MASTER EN SCIENCES INFORMATIQUES

Utilisation du multilmédia dans le réapprentissage des aphasiques

Deghouys, Johan; Thise, Frédéric

Award date:
1995

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique
rue Grandgagnage, 21, 5000 Namur

Utilisation du multilmédia dans le réapprentissage des aphasiques.

Deghouys Johan **Thise Frédéric**

Promoteur : **Madame Monique Noirhomme-Fraiture**

Mémoire présenté en vue de l'obtention du grade de
Licencié et Maître en Informatique

Année académique 1994-1995

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique
rue Grandgagnage, 21, 5000 Namur

Utilisation du multilmédia dans le réapprentissage des aphasiques.

Deghouys Johan **Thise Frédéric**

Promoteur : **Madame Monique Noirhomme-Fraiture**

Mémoire présenté en vue de l'obtention du grade de
Licencié et Maître en Informatique

Année académique 1994-1995

Abstract.

Ce travail présente le réapprentissage de l'aphasie grâce à l'outil informatique et au multimédia. Notre approche est d'abord théorique, par la description de la maladie et de la rééducation associée. Elle est ensuite pratique par la présentation d'un logiciel développé sous Director 4.0 pour une personne aphasique particulière, Gun Andersson. Enfin, l'évaluation du logiciel auteur employé a permis de définir son potentiel comme aide à la conception d'exercices pour personnes aphasiques. Un cahier des charges type pour un logiciel auteur dédié en a été dégagé.

This paper presents the rehabilitation of aphasia through the use of the computer and multimedia. Our approach is, first of all, theroretical with the description of the disease and its therapy. Then, our approach is practical with the presentation of a software designed under Director 4.0 for an aphasic person, Gun Andersson. Finally, this authoring tool has been evaluated and we defined its potential as a guide to design exercices for aphasic persons. According to that, a requirements document for a specialized authoring tool has been determined.

Remerciements.

Nous voudrions témoigner toute notre gratitude à Madame Monique Noirhomme-Fraiture qui a assuré la supervision de ce mémoire et qui nous a guidé de ses conseils éclairés.

Nous désirons aussi remercier Monsieur Jörn Nilsson qui nous a été d'une aide inestimable tout au long de notre séjour en Suède. De même nous remercions Mesdames Gunilla Knall et Gun Andersson qui nous ont éclairé sur le terrible fléau qu'est l'aphasie.

Nous remercions également Monsieur Bertil Harriman pour nous avoir orienté dès notre arrivée en Suède.

Nous remercions aussi Monsieur Luc Goffinet pour l'attention qu'il nous a accordée.

Nous remercions le docteur David Otogali et son équipe pour nous avoir reçu au CTR et pour les précieuses informations qu'ils nous ont communiquées.

Nous remercions de même les personnes qui, quoique dans l'ombre, ont participé à l'élaboration de ce mémoire.

Table des matières.

Introduction.

Le terme multimédia est généralement employé pour qualifier les applications gonflées d'images, de sons, d'animations, de séquences vidéos, etc. que l'on voit de plus en plus fleurir sur nos ordinateurs. L'évolution technologique, réalisée dans ce domaine par le CD-ROM et l'apparition de micro-ordinateurs surpuissants (à base de Pentium ou de Power PC), permet à tout un chacun de goûter aux joies du multimédia.

Cependant, le multimédia est bien plus qu'une simple collection hétéroclite de différents médias. Malgré cela, bon nombre d'applications actuelles semblent s'en contenter. Il suffit qu'un logiciel renferme quelques bruitages inutiles et l'une ou l'autre image pour se voir affublé du logo multimédia par les « commerciaux ».
Selon nous, le multimédia doit avant tout être considéré comme une nouvelle manière d'envisager la conception d'applications. Grâce à lui, l'importance que l'on accorde à l'interface atteint des proportions jusqu'alors impensables. L'interface devient le centre du logiciel, elle ne peut plus être négligée comme c'est parfois encore le cas.

D'un autre coté, les possibilités graphiques et sonores quasi-illimitées du multimédia intéressent de plus en plus le milieu médical. De nombreux centres de traitement du handicap semblent y voir le moyen de diversifier leurs programmes de réapprentissage et de seconder de manière efficace les thérapeutes.
Considérer l'informatique comme un outil d'apprentissage potentiel est encore relativement récent, utiliser le multimédia comme moyen d'effectuer cet apprentissage est de surcroît assez peu envisagé. Nous allons tâcher dans ce mémoire de déterminer l'apport du multimédia pour la rééducation de personnes souffrant d'une maladie peu connue mais extrêmement présente, l'aphasie.

Dans la première partie de ce mémoire, nous présenterons le contexte général de l'aphasie. Nous en déterminerons les causes et les conséquences. Ensuite, nous envisagerons le canevas type d'un réapprentissage et la place que peut y occuper l'informatique et le multimédia. Nous nous concentrerons sur certains troubles particuliers pour le réapprentissage desquels nous avons développé un logiciel.

Dans la seconde partie, nous introduirons le logiciel auteur avec lequel nous avons travaillé pendant notre stage en Suède. Nous en ferons une critique et nous le comparerons avec les langages de programmation classiques. Nous essaierons de déterminer son utilité et son potentiel pour les équipes de thérapeutes. Ensuite, nous présenterons un cahier des charges

type pour un logiciel auteur spécialisé dans la conception par des thérapeutes d'applications destinées à des personnes souffrant d'un handicap en général et d'aphasie en particulier.

Dans la troisième partie, nous décrirons en détail le cas de Gun Andersson et le logiciel que nous avons développé pour elle, « Gun's Garden ». Le logiciel sera d'abord décrit du point de vue de son interface et, ensuite, de son implémentation.

Partie 1 :
L'aphasie : Théorie et réapprentissage.

Dans le premier chapitre de cette partie, nous allons tâcher de présenter l'aphasie en évitant les aspects trop « techniques » et en restant à un niveau relativement global.
Nous essaierons de mettre en évidence les aspects les plus utiles des théories relatives à la maladie pour la mise en place d'un réapprentissage général dans un premier temps et informatique par la suite.

Le réapprentissage sera l'objet du deuxième chapitre dans lequel nous présenterons les procédures communément employées lors de la rééducation des personnes aphasiques hors de tout contexte informatique. Ensuite, nous présenterons les apports de l'informatique et du multimédia dans la mise en place d'un programme de rééducation. Nous nous concentrerons tout particulièrement sur la rééducation de l'orientation spatiale et temporelle et sur la rééducation de la mémoire des mots.

Chapitre 1. L'aphasie.

Pour la rédaction de ce chapitre, nous nous sommes appuyés principalement sur les ouvrages référencés en [3] et en [12].

1.1 Définition.

On entend par aphasie l'ensemble des cas de perturbation du langage par exaspération ou par limitation de la fonction. Ces troubles affectent aussi bien la compréhension que l'expression des signes verbaux oraux et écrits. Ils sont déterminés par des lésions cérébrales mais sont indépendants de toute atteinte des organes périphériques d'exécution et de réception.

Les cas de perturbation du langage par exaspération sont caractérisés par une inflation de la parole combinée à une distorsion des mots, à une désorganisation de la syntaxe et à des troubles de compréhension et d'intégration.
Les cas de perturbation du langage par limitation sont caractérisés par une affection des ressorts de la combinatoire verbale, soit au niveau de l'articulation élémentaire (phonèmes), soit au niveau de la composition des mots et de la phrase. Ils s'accompagnent ordinairement de troubles parallèles dans l'exécution des signes graphiques.

L'étude de l'aphasie est aujourd'hui au carrefour de plusieurs disciplines scientifiques telles que la neurologie, la psychologie, l'orthophonie, la linguistique et même l'informatique. Il résulte de

cette convergence un ensemble de données, de modes d'approche, de techniques et de méthodes scientifiques qui conduisent à utiliser le terme d'*aphasiologie*.

Pour bien comprendre la nature des troubles affectant le langage aussi bien parlé qu'écrit, il est nécessaire de posséder une certaine connaissance des mécanismes du langage.

1.2 Le langage.

1.2.1 La zone du langage.

La neurologie du siècle dernier a vainement tenté d'associer avec précision diverses fonctions psychologiques à divers centres anatomiques du cerveau humain. Même si cette notion de *centre-siège d'une fonction* n'a plus réellement cours de nos jours, il est indéniable que, notamment dans le cas du langage, certaines structures jouent un rôle prépondérant.
En outre, il est souvent possible de déduire plus ou moins précisément le siège d'une lésion cérébrale d'une étude des désordres qui en résultent. Cependant, grâce à l'évolution de la technologie médicale, cette fonction est de moins en moins assurée par les neuropsychologues.

On appelle *zone du langage*, une zone délimitée par certaines portions du cortex de l'hémisphère dominant du cerveau. L'hémisphère dominant est dans plus de 99% des cas l'hémisphère gauche pour les droitiers (représentant environ 90% de la population) et est dans environ 50% des cas l'hémisphère droit pour les gauchers (représentant donc 10% de la population).

Toute lésion dans cette zone peut entraîner des troubles du langage que l'on qualifie d'aphasiques et, en raison de la fragmentation physique dans le cerveau des différentes fonctions psychologiques, d'autres troubles souvent associés. En effet, les troubles aphasiques sont rarement les seuls troubles présentés par le malade.

1.2.2 Les mécanismes du langage.

Nous allons, pour mieux comprendre la formation du langage aussi bien parlé qu'écrit, en présenter un modèle linguistique[1] simplifié. Ce modèle permet une description adéquate des séquences du langage parlé et écrit en termes d'unités simples s'articulant en unités plus

[1] Roch Lecours, A et Lhermitte, F., *L'aphasie*, Flammarion médecines-sciences, 1979, p. 59.

complexes. A partir de ce modèle, nous allons identifier et localiser un certain nombre de troubles aphasiques qui se trouveront combinés de diverses manières chez chaque patient.

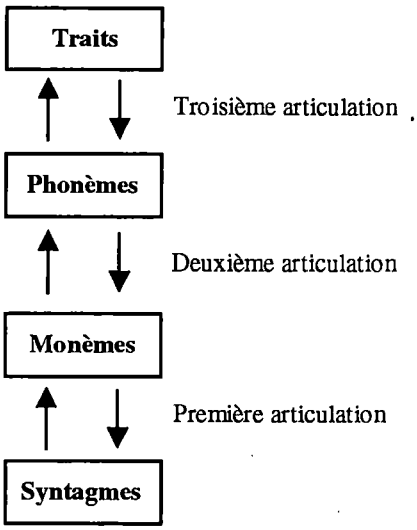


Figure 1.1. Un modèle linguistique à quatre unités et trois articulations.

Les quatre unités du modèle :

Le trait.

Le trait est l'unité atomique du langage. Dans le langage parlé, il résulte de l'action ou de la non-action de différents muscles de l'appareil buccophonatoire (le mouvement d'expiration, l'occlusion, la vibration audible des cordes vocales, la contraction du voile du palais sont autant d'exemples de traits).
Dans le langage écrit, le trait graphique est constitutif de la lettre.

Le phonème ou graphème.

Dans le langage parlé, le phonème est le résultat audible d'un groupe bien défini de traits. Chaque élément de l'alphabet phonétique est un phonème (par exemple, /n/ comme dans /ne/ (nez) ou /j/ comme dans /maj/ (maille)).
Dans le langage écrit, le graphème est représenté par une ou plusieurs lettres réalisant la transcription d'un phonème (par exemple, dans le mot PHARE, le graphème PH est la transcription du phonème /f/).

Le monème.

C'est le signe minimum, c'est-à-dire le plus petit segment linguistique signifiant ayant une face signifié (par exemple, /fam/ (femme) et /bij/ (bille) sont deux monèmes). La face signifiant du monème est, dans la majorité des cas, constituée par une série linéaire de plusieurs phonèmes ou graphèmes.

Le syntagme.

Ce terme désigne tout ensemble de monèmes sériés suivant les lois de la syntaxe. Il recouvre des unités linguistiques de divers degrés de complexité tel que le mot, le groupe de mots et la phrase.

Les trois articulations du modèle :

La première articulation ou niveau morphosyntaxique.

Cette articulation permet la sélection d'un certain nombre de monèmes et leur intégration sérielle, suivant un système morphologique (syntaxe du mot) et syntaxique conventionnel, en séquences progressivement plus complexes, les syntagmes (le syntagme /kɛlbɛlfam/ (Quelle belle femme !) est constitué des monèmes /kɛl/, /bɛl/ et /fam/).

La deuxième articulation ou niveau phonémique.

Ce niveau induit la sélection d'un certain nombre de phonèmes et leur intégration sérielle en séquences plus complexes (par exemple, le monème /kij/ (quille) est constitué des phonèmes /k/, /i/ et /j/)..

La troisième articulation ou niveau phonétique.

Ce niveau implique la sélection d'un certain nombre de traits et leur intégration, suivant une convention phonétique, en groupes constitutifs d'unités plus complexes, les phonèmes (par exemple, le mouvement d'expiration, l'occlusion, l'articulation apico-alvéolatoire, la

vibration audible des cordes vocales, la contraction du voile du palais sont les traits descriptifs du phonème /d/).

1.2.3 Les troubles du langage.

Nous pouvons dorénavant envisager une description d'un certain nombre de troubles du langage dans son aspect expressif suivant la ou les articulations où ceux-ci ont lieu.

Au niveau de la troisième articulation, on retrouve :

- les transformations phonétiques qui se caractérisent par des anomalies dans la réalisation de certains traits constitutifs des phonèmes émis (ce sont, par exemple, les phénomènes plus ou moins complets de nasalisation, d'occlusion, d'assourdissement, etc.).

- les troubles du graphisme qui se caractérisent par diverses déformations des symboles linguistiques écrits. Ces troubles ne touchent pas le choix des lettres mais leur exécution.

Au niveau de la deuxième articulation, on retrouve :

- les paraphasies phonémiques qui se caractérisent par l'omission et/ou l'ajout, le remplacement voir le déplacement d'unités phonémiques dans un mot ou segment de phrase (par exemple, /inaktif/ (inactif) est remplacé par /inaktik/ ou bien /episri/ (épicerie) est remplacé par /esipri/).

- les paraphasies verbales formelles qui se caractérisent par l'emploi d'un mot appartenant à un inventaire de la langue à la place d'un autre mot appartenant aussi à un inventaire de la langue et qui lui ressemble sur le plan de la forme (par exemple, /kavjar/ (caviar) au lieu de /kadavr/ (cadavre)).

- les paragraphies littérales et graphémiques qui se caractérisent par un comportement similaire à celui des paraphasies phonémiques mais transposé au niveau de la lettre pour la paragraphie littérale et du graphème pour la paragraphie graphémique.

Au niveau de la première articulation, on retrouve :

- les paraphasies monémiques qui se caractérisent par l'emploi d'une association, n'appartenant pas au dictionnaire de la langue, de monèmes appartenant au dictionnaire de la langue (par exemple, l'association incorrecte /italar/ est formée des monèmes /ital/ et /ar/ valides).

- les néologismes qui se caractérisent par l'emploi de mots n'existant dans aucun inventaire de la langue.

- les paraphasies verbales sémantiques qui se caractérisent par l'emploi d'un mot appartenant à un inventaire de la langue à la place d'un autre mot appartenant aussi à un inventaire de la langue et qui lui ressemble au niveau de la sémantique (par exemple, « pomme » est substitué à « poire », « bon » à « mauvais », etc.).

- les mots de prédilection qui se caractérisent par l'emploi d'un même mot à la place de n'importe quel mot appartenant au même inventaire.

- les paraphasies syntagmiques qui se caractérisent par des substitutions entre unités linguistiques plus complexes que le mot (par exemple « un éclairage intime » est remplacé par « la lumière tamisée »). Ce trouble survient presque exclusivement lors de la lecture à voix haute.

- les paragraphies verbales qui se caractérisent par un comportement similaire à celui des paraphasies verbales sémantiques mais transposé au niveau du langage écrit.

Les troubles aphasiques qui ont été présentés dans cette section sont des troubles du pôle expressif du langage que le modèle linguistique a permis de classifier. De plus, ce modèle a permis de le faire de manière hiérarchique. Dans la section 1.4, nous présenterons les troubles de la compréhension du langage ainsi que d'autres troubles expressifs qui ne peuvent être clairement attachés à une articulation particulière du modèle.

1.3 Les causes de l'aphasie.

Nous allons dans cette section donner un aperçu des atteintes cérébrales déterminant les troubles aphasiques.

En effet, il est important lors de l'étude de l'aphasie de tenir compte de son origine physique car il est extrêmement rare que l'aphasie soit le seul trouble provoqué par cette origine. Ce point est fondamental lors de l'élaboration d'un réapprentissage.

1.3.1 La thrombose cérébrale.

La thrombose cérébrale peut se définir comme l'occlusion d'une ou de plusieurs artères nourricières du cerveau par formation d'un caillot. Le tissu vasculaire, alentour, cesse d'être normalement irrigué, et, selon l'importance de l'atteinte initiale combinée à la finesse de la topographie vasculaire, le ramollissement qui s'en suit est plus ou moins étendu et plus ou moins important.

1.3.2 L'hémorragie intracrânienne.

Ce type d'hémorragie a pour cause soit l'hypertension artérielle, soit la rupture d'un anévrisme artériel. La masse sanguine libérée par l'hémorragie peut endommager les régions avoisinantes en les comprimant.

1.3.3 Les traumatismes crâniens.

Les traumatismes crâniens, en raisons de leur nombre croissant (accidents de la route, du travail, etc), représentent une cause importante d'aphasie. Avec ce genre d'atteinte, il est cependant moins aisé de déterminer l'importance des lésions cérébrales.

1.3.4 Les tumeurs cérébrales.

Les tumeurs cérébrales peuvent entraîner selon leur localisation diverses manifestations aphasiques. Ces manifestations se caractérisent surtout, dès le début, par une perturbation du pouvoir d'évocation verbale.

1.4 Les formes de l'aphasie.

Il existe de nombreuses classifications dues à d'éminents chercheurs (Wernicke, Freud, Pierre Marie, etc.) répertoriant les différents types d'aphasie mais il serait assez peu intéressant d'en faire la présentation dans le cadre de ce mémoire et ce pour différentes raisons. Premièrement, il est très difficile de faire « rentrer » chaque cas particulier d'aphasie dans une seule catégorie et, deuxièmement, les équipes de thérapeutes chargées de la rééducation des personnes aphasiques se basent avant tout sur les troubles observés chez le patient pour effectuer leur travail.

Nous allons articuler notre analyse des troubles aphasiques autour de quatre en-têtes génériques : troubles de l'expression orale, troubles de la compréhension du langage parlé, troubles de l'expression écrite et troubles de la compréhension du langage écrit.

Sous les deux en-têtes relatifs à l'expression du langage, vont se retrouver tous les troubles du langage identifiés à la section 1.2.3.

1.4.1 Troubles de l'expression orale.

a) Suppression et autres anomalies du débit.

Il s'agit, dans le cas de la suppression, d'une absence totale de production linguistique. Elle n'est guère observée que dans les formes d'aphasie à début brutal (dont la cause est vasculaire ou traumatique). Elle est en générale transitoire et évolue habituellement vers une réduction quantitative du débit (c'est-à-dire le nombre de mots prononcés par unité de temps).

b) Réduction qualitative et stéréotypies.

La réduction qualitative désigne une diminution dans le nombre de mots différents disponibles. Le phénomène de réduction peut dans certains cas être si marqué que l'éventail des mots accessibles se ramène à quelques phonèmes ou quelques mots que le malade va répéter inlassablement dans la conversation.

c) Manque du mot.

Ce terme désigne une difficulté, voire une impossibilité pour le malade de produire le mot adéquat. Le manque du mot est patent lorsqu'il y a réduction qualitative mais l'inverse n'est pas vrai. En effet, le manque du mot diffère de la réduction par le fait qu'il peut n'apparaître que dans une activité linguistique particulière (par exemple, le manque du mot peut n'apparaître que lors d'exercices de dénomination). Le manque du mot est un trouble commun à toutes les variétés d'aphasie observées et doit donc être traité chez toutes les personnes aphasiques.

d) Dysprosodies.

Le terme « prosodie » recouvre les phénomènes suivants : intonation affective, particularismes régionaux, accents toniques dans certaines langues, etc.
L'aphasie peut modifier dans certains cas la mélodie de la langue, on parlera alors de dysprosodie ou encore d'aprosodie. Cela se marque habituellement par une atténuation sensible de l'intonation donnée à la phrase qui devient en quelque sorte lente et monotone.

e) Transformations aphasiques du langage oral.

Ce titre recouvre en fait les troubles du langage oral identifiés dans la section 1.2.3 tel que les transformations phonémiques, les paraphasies verbales, les paraphasies monémiques, les néologismes, les transformations verbales sémantiques, les mots de prédilection et les paraphasies syntagmiques.

f) Jargonaphasie.

Les jargonaphasies consistent en un discours souvent abondant et précipité, logorrhéique, mais dont le contenu est habituellement incompréhensible. Plusieurs degrés de gravité peuvent être distingués selon que le discours est composé d'une variété de phonèmes associés sans règles et privés de sens, ou de termes inconnus dans la langue avec une apparence d'organisation syntaxique, ou de termes appartenant à la langue mais utilisés sans aucun sens des règles de la grammaire.

g) Dyssyntaxie.

La dyssyntaxie se caractérise par une utilisation incorrecte des règles de la syntaxe lors de l'élaboration des phrases. En fait, la dyssyntaxie ne désigne pas une maladie spécifique de la syntaxe mais qualifie plus avant certaines paraphasies.

h) Agrammatisme.

L'agrammatisme désigne un type particulier de comportement linguistique que l'on peut caractériser comme suit : ralentissement du débit, réduction générale du vocabulaire disponible, réduction et simplification des structures syntaxiques disponibles, brièveté des phrases, élisions et substitutions de certains monèmes.

1.4.2 Troubles de la compréhension du langage parlé.

Contrairement aux comportements aphasiques observés dans l'expression orale, ceux observés dans la compréhension du langage parlé sont moins clairement et précisément identifiés. Cela est dû en grande partie au fait que ces comportements sont plus difficilement observables. L'ingéniosité de l'observateur entre alors largement en ligne de compte.

On sait cependant que suivant les types d'aphasies, il peut exister ou non un trouble de la discrimination et de la reconnaissance des stimuli sonores linguistiques ou non-linguistiques. Ce trouble est appelé *surdité verbale*. La surdité verbale peut ne se limiter qu'à la compréhension de la langue parlée ou à la reconnaissance des sons non-articulés (bruits d'animaux, musiques, etc.). Cependant, elle peut aussi porter sur les deux types de stimuli. Ce point aura notamment une importance cruciale lors de la conception de logiciel de réapprentissage comme nous le verrons plus loin.

1.4.3 Troubles de l'expression écrite.

En général, la langue écrite est plus touchée que la langue parlée et la récupération en est plus lente et moins complète.
Une des raisons pouvant expliquer ce phénomène est le fait que la langue écrite est acquise plus tardivement que la langue parlée, elle pourrait de ce fait être plus fragile.

De plus, chez la plupart des gens, la langue écrite est quantitativement moins utilisée que la langue parlée. La langue écrite est aussi plus riche et variée que la langue parlée; les mots y sont souvent plus divers, les tournures syntaxiques complexes y sont plus fréquentes et les syntagmes figés (locutions toutes faites) moins nombreux.

a) Anomalie du débit.

La suppression de l'expression linguistique écrite est plus fréquente et plus souvent définitive que celle de l'expression orale. Quoi qu'il en soit, le débit d'écriture de l'aphasique est en général bien moindre que celui d'une personne normale.

b) Réduction qualitative.

La réduction qualitative du langage écrit et celle du langage parlé vont de pair chez presque tous les aphasiques qui en sont affectés. Il arrive cependant que la première soit plus marquée que la seconde.

c) Manque du mot.

La majorité des faits signalés pour le manque du mot relatif à l'expression oral est aussi valable pour le manque du mot relatif à l'expression écrite.

d) Transformations aphasiques du langage écrit.

Les transformations aphasiques du langage écrit, tout comme celles du langage oral, ont déjà été discutées à la section 1.2.3. Ces transformations sont les suivantes : paragraphies littérales et graphémiques, paragraphies verbales, néologismes écrits, troubles du graphisme.

e) Jargonagraphie, dyssyntaxie et agrammatisme.

Le terme Jargonagraphie est le pendant pour le langage écrit du terme Jargonaphasie et recouvre le même type de trouble bien que transposé à la forme écrite. Les termes dyssyntaxie et agrammatisme sont similaires à ceux définis dans la section 1.4.1.

Il n'existe cependant pas une symétrie parfaite entre les troubles des deux types d'expression, certains malades peuvent avoir une conversation à peu près exempte de paraphasies et de néologismes mais une expression écrite franchement jargonnée. Le contraire étant nettement plus rare.

1.4.4 Troubles de la compréhension écrite.

Tout comme pour les troubles de la compréhension du langage parlé, il existe des troubles portant sur la discrimination et la reconnaissance des stimuli visuels cette fois. Ces troubles portent le nom de *cécité verbale*. La cécité verbale se caractérise par la difficulté pour le sujet de comprendre les messages linguistiques écrits ou de discriminer les stimuli visuels non-linguistiques tels que les couleurs. Tout comme pour la surdité verbale, la cécité verbale peut porter sur les deux types de stimuli à la fois.
Cependant, la cécité verbale est un syndrome relativement rare.

1.5 Conclusion.

Comme nous l'avons vu, l'aphasie est un trouble acquis de la fonction du langage mais ne serait-elle pas aussi une forme de détérioration intellectuelle?

Pour certains, les troubles aphasiques ne sauraient être considérés comme des signes de l'affaiblissement ou de la dissolution d'autres fonctions supérieures. Car selon eux, ou bien ces fonctions se maintiennent effectivement intactes ou bien, si elles sont altérées, ce ne peut être que dans la mesure où leur exercice normal implique l'utilisation du langage. Lotmar[2] a écrit que les troubles aphasiques sont, non pas « *le résultat d'un déficit intellectuel, mais un déficit autonome qui agit par contrecoup sur l'intelligence à laquelle il enlève un de ses meilleurs instruments de travail* ».

Pour d'autres, par contre, comme Pierre Marie[3], « *...il y a chez les aphasiques quelque chose de bien plus important et de bien plus grave que la perte des mots ; il y a une diminution très marquée de la capacité intellectuelle en général. ...Si, pour ma part, j'avais à donner une définition de l'aphasie, le fait que je m'efforcerais surtout de mettre en lumière serait la diminution de l'intelligence* ».

[2] Cazayus, P., *L'aphasie du point de vue du psychologue*, Dessart & Mardaga, Bruxelles, 1977, p. 97.
[3] Roch Lecours, A et Lhermitte, F., *L'aphasie*, Flammarion médecines-sciences, Paris, 1979, p. 495.

En fait, il semble que la vérité se trouve entre les deux extrêmes. Elle dépend notamment de la notion que l'on a de l'intelligence. Si l'on « mesure l'intelligence » de l'aphasique à l'aide exclusive d'épreuves faisant appel à la compréhension et à la production du langage, on arrive indubitablement à la conclusion que l'aphasie est une détérioration intellectuelle. En revanche, si l'on emploie exclusivement des épreuves où la compréhension et la production du langage n'entrent que très peu en ligne de compte, la conclusion à laquelle on arrivera pourra varier suivant les cas. Les observations tendent en fait à prouver que l'aphasie n'entraîne pas nécessairement, de son fait, une déchéance de l'intelligence générale.

Cette conclusion est cruciale pour la conception d'un logiciel de réapprentissage. En effet, doit-on considérer l'aphasique comme une personne intellectuellement diminuée et ainsi concevoir un logiciel similaire dans sa présentation à ce qui est actuellement fait pour les personnes handicapées mentales[4] ou doit-on considérer l'aphasique comme une personne normale souffrant d'un déficit communicatif?
Comme nous le verrons dans la description du logiciel que nous avons développé, nous avons pris le parti de la deuxième considération.

[4] Lepoutre, T. et Roquet J., *La personne handicapée mentale et la connaissance du corps humain : un logiciel d'apprentissage*, Institut d'informatique, 1990.

Chapitre 2. Le réapprentissage.

Toute aphasie ayant pour origine une lésion de l'hémisphère dominant du cerveau, nous allons dissocier dans un premier temps la récupération physique de la lésion du réapprentissage de l'aphasie à proprement parlé. De plus, il est important de préciser d'emblée qu'un réapprentissage ne portera jamais uniquement sur l'aphasie et ceci pour plusieurs raisons. Puisqu'il n'y a pas correspondance entre une fonction psychologique ou physique et une zone bien délimitée du cerveau, toute atteinte cérébrale portera très probablement sur plusieurs de ces fonctions.
En outre, certains types de troubles ne découlant pas de l'aphasie tel que les problèmes d'orientation et de structuration spatiale et temporelle ont une incidence sur le réapprentissage de l'aphasie. Il en est de même pour certains troubles physiques allant souvent de pair avec l'aphasie tel que l'hémiplégie.

Il est donc difficile d'extraire l'aphasie de l'ensemble des troubles observés ; un programme de réapprentissage pour les personnes aphasiques devra être élaboré dans la perspective d'un programme plus global et mettra donc en présence des équipes pluridisciplinaires de thérapeutes.

2.1 Problème de la récupération physique ou fonctionnelle.

Chez l'adulte, le principal facteur de récupération est la réversibilité des lésions qui ont touché les neurones de la zone du langage (cfr. section 1.2.1). La rééducation accroit sans conteste la vitesse de récupération mais cette récupération se butera dans la majorité des cas à un plafond. En effet, la lésion à l'origine de l'aphasie a, la plupart du temps, provoqué la destruction irréversible de tissus cérébraux. La récupération ne peut se faire que sur les tissus abîmés mais non détruits.
Cependant, le cerveau possède d'autres moyens de pallier l'insuffisance engendrée par la destruction d'une partie de sa structure. Il possède en effet une étonnante plasticité : la destruction irréversible de certaines parties de cet organe, essentielles à l'accomplissement de fonctions d'un haut niveau d'élaboration, se trouve compensée par la mise en jeu de l'autre hémisphère. Malheureusement, cette capacité s'amenuise avec l'âge.

Le caractère réversible et l'importance de la compensation hémisphérique déterminent la récupération physique et par conséquent psychologique du patient.

2.2 Forme générale du réapprentissage.

Il n'existe pas a priori de méthode générale de réapprentissage des personnes cérébro-lésées en général et aphasiques en particulier. On peut seulement observer que les équipes de thérapeutes se conforment à un canevas très général. Chaque cas est traité indépendamment et individuellement et un réapprentissage personnalisé est élaboré.
Nous allons dans la suite de cette section présenter le canevas général de l'élaboration d'un programme de réapprentissage tel que proposé en [7] et [24].

2.2.1 Première étape : examen aphasiologique.

Le but premier de cet examen est de reconnaître l'existence du syndrome aphasique et, le cas échéant d'en permettre une description exhaustive.
C'est à partir d'une observation de tous les aspects élémentaires de la fonction linguistique que l'examinateur identifie la forme clinique de l'aphasie.
Un premier examen permet, en général, de dresser un diagnostic et d'établir un pronostic quant à l'état de la maladie ou de la lésion responsable de l'aphasie mais il ne permet pas d'établir un pronostic quant au degré de récupération du trouble aphasique. Pour ce faire, un second examen sera nécessaire.
L'examen aphasiologique va détecter de manière exhaustive les troubles aphasiques en passant en revue les quatre grands pôles du langage à savoir l'expression orale, l'expression écrite, la compréhension du langage parlé et la compréhension du langage écrit.

L'examen se base principalement sur une conversation que viennent renforcer une série de tests écrits.
L'interprétation des résultats, notamment de ceux portant sur l'expression orale et écrite, doit se faire en référence à une norme relativement souple. Il ne s'agit pas de faire son évaluation d'après la grammaire du bon usage mais d'après une grammaire communément employée dans la vie de tous les jours. Cette grammaire dépend de différents facteurs comme la région de provenance du patient, son niveau d'éducation, son milieu social, etc. De plus, l'interprétation doit tenir compte de l'état général du malade, de son degré de fatigue, etc.

Le thérapeute établit enfin un diagnostic et peut tenter de poser un pronostic.
Ce pronostic va prendre en compte différents paramètres conditionnant la rééducation :

- la meilleure capacité de récupération des gauchers et sujets peu latéralisés.
- l'état de santé générale du patient.
- la taille, l'emplacement et le caractère unique ou multiple des lésions.

- le type clinique de l'aphasie.
- la cause de la lésion.
- l'âge du patient.
- la personnalité et les facteurs socio-culturels.

Afin d'élaborer une image relativement fidèle de l'état du patient, le diagnostic aphasiologique est confronté au diagnostic des autres troubles psychologiques et physiques. Ce dernier diagnostic sera basé sur une série de tests psychométriques comprenant entre autres :

- des tests d'audiométrie : ces tests permettent de définir les limites de la capacité d'audition du sujet.
- des tests de périmétrie : ces tests visent à définir les limites du champ visuel.
- des tests de vision chromatique : ces tests permettent de vérifier la capacité de reconnaissance des couleurs.
- des tests d'intelligence : ces tests permettent de mesurer le quotient intellectuel verbal et non verbal du patient.
- des tests de mémoire : ces tests visent à définir quantitativement les capacités mnésiques du sujet.
- des tests de localisation spatiale et temporelle.

A partir de cette image relativement précise, un programme de réapprentissage aphasiologique sera mis au point.

2.2.2 Deuxième étape : rééducation des troubles observés en association avec l'aphasie.

La prérééducation consiste en une série d'exercices employés dans le but de minimiser les manifestations cliniques des divers troubles neuropsychologiques souvent observés en association avec l'aphasie et de préparer le malade à tirer le plus grand bénéfice possible de la rééducation de son trouble proprement linguistique.

La prérééducation se concentre tout particulièrement sur les troubles de l'attention, sur ceux de l'orientation dans le temps et dans l'espace, de la mémoire, du calcul et sur les déficits parétiques et/ou dyspraxiques pouvant faire obstacle à la rééducation de la parole et de l'écriture. Ces troubles sont notamment détectés par la batterie de tests psychométriques mentionnés dans la section précédente.

Par exemple, chez un grand nombre de malades, des exercices préparatoires à la rééducation de l'écriture sont indiqués : ils visent à contrer les effets d'une séquelle d'hémiplégie affectant la main dominante ou à entraîner la main non-dominante (changement de latéralité).

a) Le problème de l'orientation et de la structuration dans l'espace et le temps.

Ce problème est fondamental. Sans une orientation spatio-temporelle correcte, toute tentative d'expression orale ou écrite est soit rendue extrêmement compliquée, soit vouée purement et simplement à l'échec.
En effet, ce n'est que par le biais d'une planification adéquate dans l'espace et dans le temps de ses actions qu'une personne peut effectuer une activité donnée. On comprend aisément que, faute de coordination, toute production écrite provenant d'une personne souffrant d'un grave déficit dans son orientation spatio-temporelle ne pourrait qu'être illisible (par exemple, un 'b' transformé en 'd', un 'p' en 'q', etc.).

Sans en arriver à cet extrême, toute personne se retrouverait démunie si elle était incapable de dire où elle se trouve et l'heure qu'il est. C'est principalement ce point qui, dès la prise en charge, est testé par les thérapeutes.

Lorsque l'orientation temporelle est déficitaire, l'utilisation de calendriers, par exemple, dans les chambres, indiquant les heures des différentes séances permet au patient, grâce à une concentration sur une tâche quotidienne concrète, de pratiquer un réapprentissage continu et indolore. De plus, les thérapeutes essaient de prévoir pour chaque patient des séances régulières afin de faciliter la reconstruction d'un positionnement temporel. De même, des exercices d'orientation peuvent être pratiqués sur des patients munis de la carte du bâtiment indiquant l'emplacement de la prochaine séance. Lorsque la structuration spatiale est gravement déficiente, des exercices basés sur l'utilisation de puzzles et des exercices de dessin ou basés sur des dessins sont généralement mis en place.

Il s'agit donc à ce niveau, en accordant une marge de manoeuvre confortable au patient, de lui redonner une certaine indépendance dans ses déplacements et dans la gestion de son horaire.

Pour de plus amples informations sur les troubles de l'orientation et de la structuration spatiale et temporelle, le lecteur pourra consulter avantageusement [7] et [22].

b) Les troubles de la mémoire.

Si la fonction du langage est un sous-système de l'intellect, la mémoire en est un autre. La neuropsychologie a permis d'opérer une distinction entre un système de mémoire à court terme et un système de mémoire à long terme en montrant que certains patients cérébro-lésés peuvent présenter des déficits dans un des deux systèmes et pas dans l'autre. Cette double dissociation suggère que des processus ou des systèmes distincts sont impliqués dans le maintien temporaire et à plus long terme de l'information.

De plus, différentes études ont montré que la mémoire à court terme ne constitue pas un système unitaire. En effet, certains patients peuvent manifester un déficit spécifique dans la rétention à court terme d'un matériel verbal tout en ayant une mémoire à court terme normale pour un matériel visuo-spatial et inversement (cfr. figure 2.1).

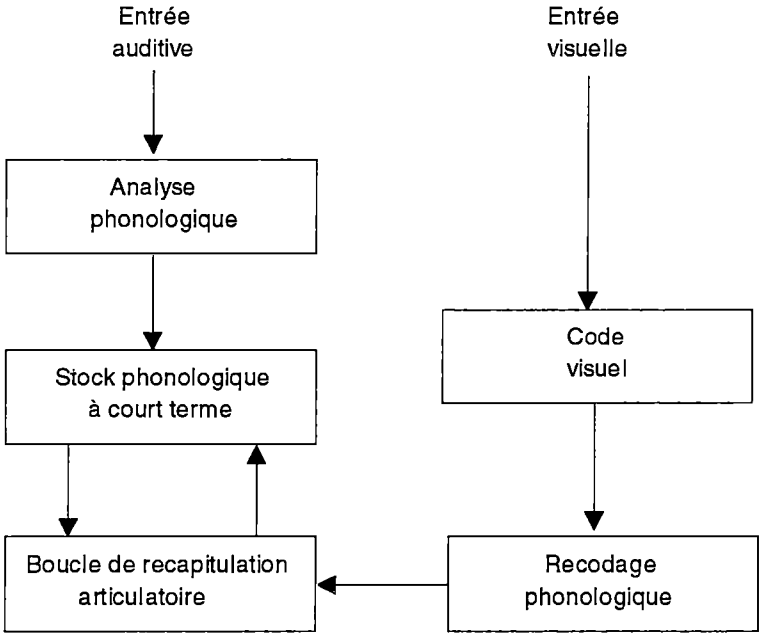


Figure 2.1. Architecture fonctionnelle de la mémoire de travail : le système de la boucle phonologique.

Le système de la boucle phonologique[5] est spécialisé dans le stockage temporaire de l'information verbale. Il se compose d'un stock phonologique et d'un processus de récapitulation articulatoire. Le stock phonologique reçoit directement et obligatoirement l'information verbale présentée auditivement qu'il stocke sous la forme de codes phonologiques. L'information n'y est maintenue que durant une période très brève (une à deux secondes) mais peut être réintroduite continuellement via un processus de récapitulation

[5] Seron, X., *La neuropsychologie cognitive*, Presses Universitaires de France, 1993.

articulatoire. Ce stock phonologique reçoit également de l'information verbale présentée visuellement mais celle-ci doit préalablement être convertie en un code phonologique par le biais du processus de récapitulation articulatoire.

Un registre visuo-spatial effectue un travail similaire pour le stockage de l'information visuo-spatiale mais son fonctionnement est actuellement moins bien compris.

La mémoire de travail, d'une capacité limitée, est destinée au maintien temporaire et à la manipulation de l'information pendant la réalisation de tâches cognitives diverses de compréhension, de raisonnement ou de résolution de problème.
Un déficit de ce système de mémoire risque d'hypothéquer tout réapprentissage ultérieur des fonctions psychologiques touchées par l'aphasie ou par des troubles associés. En effet, disposer d'une mémoire de travail acceptable est un prérequis à la compréhension des exercices de rééducation proposés de manière classique ou à la compréhension et la manipulation d'une interface de logiciel.
Si la rééducation de la mémoire de travail n'est pas satisfaisante, la conception de logiciels de réapprentissage des différents troubles aphasiques présentés par le sujet devra à tout prix tenir compte de la limitation des capacités mnésiques. De plus, comme le montre l'aspect complémentaire de l'entrée auditive et visuelle, l'utilisation de plusieurs canaux de communication pour transmettre la même information permet un renforcement de la mémorisation par le biais de la boucle de récapitulation articulatoire.

2.2.3 Troisième étape : mise en place de la rééducation.

Alors que les progrès de la thérapeutique médicale permettent le traitement de certaines des maladies à l'origine de l'aphasie, ceux des disciplines d'où pourrait provenir une théorie de la rééducation du langage sont encore très fragmentaires.
Bien qu'il existe de nombreux partisans d'approches contradictoires de la rééducation des aphasiques, tous s'accordent à dire que celle-ci doit être dirigée dans un souci constant de la personnalité globale du malade.
En fait, d'après ce que nous avons pu récolter dans [7], [12] et [24] et d'après nos entretiens avec des thérapeutes en Belgique et en Suède, il ressort très nettement qu'en matière de rééducation, l'arbitraire et le pragmatisme sont de rigueur.

a) Quand commencer le réapprentissage?

Il n'est à peu près jamais opportun de commencer la rééducation durant la phase aiguë de la maladie : l'état général du patient et le fait que l'on doive alors s'occuper surtout du traitement médical ne le permettent d'ailleurs généralement pas.
De plus, certains types d'aphasie laissent le malade, durant la période aiguë, dans un état d'excitation incompatible avec la mise en train d'une rééducation systématique.

b) Caractéristiques du réapprentissage chez l'aphasique.

L'identification de ces caractéristiques permet notamment aux développeurs de logiciels de réapprentissage d'orienter leur conception de manière à éviter les écueils du réapprentissage classique.

- Il a été démontré, et ceci vaut aussi bien pour les activités verbales que pour les activités non-verbales, que la majorité des aphasiques manifeste une grande lenteur dans l'apprentissage et ne satisfait aux exigences posées par le thérapeute qu'après une période anormalement longue de pratique. La courbe d'apprentissage est souvent très irrégulière (variabilité de la performance) et les erreurs sont plus nombreuses que chez le sujet normal. Il est toutefois fréquent que l'aphasique soit sensible à l'effet de pratique et qu'il lui soit possible, pour certaines tâches, de faire le transfert et la généralisation d'un acquis. De plus, les performances d'apprentissage des aphasiques peuvent varier avec la localisation des lésions cérébrales dont ils sont porteurs et, par conséquent, avec la forme clinique de l'aphasie qu'ils présentent.

- Une autre caractéristique du réapprentissage est que les aphasiques sont, dans l'ensemble, très sensibles aux effets de la fatigue. Ces derniers sont révélés notamment par l'apparition de comportements de persévération. Le patient n'est alors plus capable que de répéter le même phonème encore et toujours. Lorsqu'une certaine fatigue est décelée, il est inutile de vouloir poursuivre sous peine de provoquer des réactions catastrophiques. Ces réactions peuvent avoir sur la rééducation des effets très marqués allant parfois jusqu'à inhiber tout apprentissage. L'utilisation de nombreuses pauses judicieusement réparties permet un déroulement optimal des séances. En outre, pour réduire les risques de fatigue, il importe de limiter autant que faire se peut les renforcements négatifs, générateurs de frustrations.

- Les thérapeutes essaient le plus possible de baser les séances sur un matériel linguistique concret. En effet, ils ont classiquement recours à une panoplie d'objets, de dessins, d'images

et de photographies qui servent de support aux exercices de langage. Ce matériel doit pouvoir être manipulé par le malade afin de fournir un maximum d'information (c'est surtout valable pour les objets). De plus, il importe d'en contrôler la qualité visuelle. Ce contrôle des qualités physiques des stimuli est également nécessaire quant aux textes écrits, listes de mots, de locutions, de phrases qui sont régulièrement employées à des fins de rééducation. En plus d'être parfaitement lisibles, ces stimuli doivent être adaptés aux besoins et caractéristiques de l'aphasique. Dans certains cas, il faut absolument proscrire un matériel à l'allure trop infantile. Comme nous le verrons plus loin, c'est parfois le défaut de certains logiciels de réapprentissage.

- Toujours au niveau du matériel, outre l'aspect concret de ce dernier, les thérapeutes essaient que celui-ci fasse référence au vécu du patient, à sa vie antérieure. Ils essaient de travailler avec des objets lui appartenant, des photographies de sa maison ou d'objets se trouvant dans celle-ci, des photographies de sa famille, etc. Les thérapeutes se sont très vite rendu compte qu'il était nettement plus difficile de travailler avec des objets ou photographies impersonnelles ne véhiculant aucun bagage affectif que de travailler directement en référence à l'environnement du malade.

- Pour les personnes polyglottes, on veillera à ce que le réapprentisage se fasse toujours dans la langue maternelle, ceci pour la simple raison que les connaissances plus anciennes sont mieux ancrées que les connaissances plus récentes.

- Il existe divers procédés relatifs à la procédure de présentation des stimuli. Par exemple, pour favoriser la production des mots faisant défaut dans un exercice de dénomination, le rééducateur peut recourir à différentes formes de facilitation :

  - *l'ébauche orale* : peut être phonique ou muette, se basant dans le dernier cas sur l'information visuelle. Cela se fait par exemple au travers de l'énoncé du début du mot et/ou de sa présentation sous forme écrite.

  - *le contexte verbal* : cette forme peu être rendue plus efficace en la doublant d'une exagération prosodique (intonation et accentuation).

  - *les gestes et les jeux de mimiques* : il peut s'agir d'une gesticulation symbolique, incitant l'évocation de tel ou tel élément de la phrase. Il peut s'agir d'une gesticulation codifiée (à la manière d'un chef d'orchestre) dont le

déroulement spatio-temporel suggère celui de la phrase.

- *le code visuel* : il s'agit par exemple de l'usage de jetons de couleurs et de formes variées dont chacun a été associé à un des constituants syntaxiques de la phrase (par exemple, un triangle pour un verbe, un carré pour un sujet, etc.).

- *le code phonémique* : certaines associations auditives formelles peuvent permettre de contourner certaines difficultés de production. Par exemple, un malade incapable d'évoquer la structure interrogative « Qu'est-ce que... » peut y arriver après avoir travaillé sur le stimulus « caisse ».

Graduellement, le thérapeute visera à restreindre l'emploi de ces procédés de facilitation et tendra à développer chez le malade une utilisation de plus en plus volontaire du langage.

- Contrairement à l'enseignement classique, il est conseillé d'utiliser un éventail assez large de stimuli différents au cours d'une même séance. Il serait en effet inapproprié de tenter vainement d'obtenir une parfaite stabilité des réponses pour un très petit nombre de stimuli. Cela risquerait d'engendrer chez l'aphasique une réaction de découragement dû aux innombrables renforcements négatifs générés. De plus, il est important d'adapter le système de présentation des stimuli à chaque malade.

- Différents accessoires audiovisuels peuvent constituer des auxiliaires précieux pour la rééducation : le magnétophone, le projecteur de diapositive, la caméra, le magnétoscope, etc. L'informatique tend de plus en plus à remplacer tout ou partie de ces accessoires.

2.2.4 Un problème particulier du réapprentissage : la rééducation lexicale.

Etant donné le type de logiciel que nous avons eu à développer lors de notre stage, nous allons nous concentrer sur la rééducation du manque du mot (cfr. section 1.4.1.c et 1.4.3.c) en ce qui concerne les troubles aphasiques.

En effet, on observe dans toutes les aphasies un comportement plus ou moins marqué de manque du mot.

Dans certains cas, les sujets peuvent appréhender les liens qui unissent les signifiés aux signifiants correspondants mais ils éprouvent des difficultés à les évoquer et à les émettre. Dans ce cas, la facilitation par l'ébauche orale permet de contourner ces difficultés de manière relativement efficace.
Dans d'autres cas, les sujets émettent les mots sans la moindre difficulté mais éprouvent des difficultés à établir des liens entre signifiés et signifiants. Il en résulte souvent un usage paraphasique des mots (cfr. section 1.4.1.e). Dans ce cas, la facilitation par l'ébauche orale présente en général de moins bons résultats.

La première phase de la rééducation lexicale consiste presque toujours en l'application d'exercices de dénomination : cette phase précoce vise, lorsqu'il y a un manque du mot manifeste, à la réacquisition d'un lexique élémentaire. Pour ce faire, le support de stimuli imagés (dessins, photographies, ...) est en général essentiel.
Dans ce cas, il est commun, lors des premières phases du traitement du manque du mot, de joindre à la présentation de stimuli imagés leur contrepartie textuelle et/ou sonore.
Dans les phases suivantes de la rééducation, le thérapeute essaie de diminuer la fréquence de ces facilitations pour finir par les supprimer totalement.

Afin d'effectuer cet apprentissage, le thérapeute va proposer au malade une série d'exercices qui permettront notamment de s'assurer de l'ancrage mnésique du lexique appris.

Ces exercices se basent sur :
- *les antonymes* : un mot est présenté au patient et celui-ci doit en évoquer le contraire ou le choisir parmi une liste (par exemple, beau ↔ laid).
- *les synonymes* : ces exercices sont analogues à ceux sur les antonymes (par exemple, sale → souillé, taché, crotté, etc.).
- *les dérivés* : à partir d'un mot présenté oralement ou par écrit, le patient doit essayer d'évoquer quelques dérivés de ce mot (par exemple, front → frontière, affront, effronté, etc.).
- *les composés* : un mot est présenté au patient ; celui-ci doit tâcher de trouver des mots composés dont le premier élément est le mot présenté (par

exemple, garde → garde-fou, garde-chasse, garde-malade, etc.).

- *les classes* : le stimulus est un mot désignant un ensemble d'éléments de même nature ; la tâche du patient est d'évoquer le nom de quelques-uns d'entre eux (par exemple, outils → marteau, scie, tournevis, etc.).

2.3 Le réapprentissage informatique.

L'informatique est de plus en plus utilisée pour la rééducation de l'aphasie. De nombreux centres possèdent des machines capables de faire tourner des logiciels de base c'est-à-dire ayant peu d'exigences au niveau du matériel. Cependant, parmi ces centres, rares sont ceux munis d'ordinateurs aptes à recevoir des applications multimédia.
Pourtant, d'après les thérapeutes, l'informatique et, plus particulièrement, le multimédia sont extrêmement intéressants pour la rééducation de certains troubles aphasiques, principalement ceux liés à des déficits de la mémoire des mots. Les budgets alloués étant ce qu'ils sont, ces thérapeutes doivent se contenter la plupart du temps de logiciels ne traitant que des aspects les plus abstraits et répétitifs des exercices de rééducation et n'étant pas aussi accessibles au malade (en raison d'une interface bien souvent trop absconse).

2.3.1 Place de l'informatique dans le réapprentissage.

L'utilisation de l'informatique suppose de la part de l'utilisateur un minimum de capacité intellectuelle et notamment une bonne structuration spatiale et temporelle. C'est pourquoi elle n'est jamais utilisée dans les premières périodes de la rééducation.
Elle ne peut pas non plus être appliquée à tous les patients. Certains de ceux-ci ne pourraient pas l'utiliser en raison du type d'aphasie dont ils soufrent (caractérisé, par exemple, par une alexie, une apraxie, etc.) ou en raison de certains troubles associés tel qu'une hémiplégie, une déficience de la mémoire à court terme, etc.
De plus, l'informatique est surtout utilisée pour la rééducation des sujets jeunes, ne nourrissant aucune crainte vis à vis de l'ordinateur.

2.3.2 Intérêt de l'informatique comme outil de réapprentissage.

D'après les thérapeutes, l'informatique est d'un apport indéniable pour différentes raisons :

- L'ordinateur est souvent perçu comme un troisième intervenant dans la relation thérapeute-patient. Il est considéré par ce dernier comme neutre par rapport au thérapeute. Il joue en quelque sorte un rôle de médiateur en raison de son aura d'infaillibilité, ce qui a pour effet de réduire les tensions parfois observées entre le thérapeute et son patient. En outre, son caractère impersonnel ne semble pas être une gêne.

- L'ordinateur ne manifeste aucune impatience et est entièrement sous les ordres du malade. Il répétera inlassablement les mêmes exercices routiniers et rébarbatifs à longueur de séance sans jamais modifier sa relation avec le patient. Il est un fait que les thérapeutes sont avant tout des êtres humains et sont par là même susceptibles de changements d'humeur et d'une certaine lassitude qui seront immanquablement perçus par le malade. En évitant cela, le patient ne se sentira jamais dans un environnement « hostile ». De plus, cette automatisation des exercices les plus simples décharge les thérapeutes d'une partie de leur travail ; ce qui leur permet de se concentrer sur des aspects beaucoup plus riches de la rééducation.

- Les logiciels développés sont en général très simples et permettent de se concentrer sur un point particulier ; ce qui permet leur réutilisation pour d'autres patients. Chaque logiciel étant alors un élément interchangeable d'un programme de réapprentissage informatique.

- L'informatique procure, de par son aspect technologique et ludique, une certaine motivation intrinsèque. Les malades effectuent les exercices de moins mauvaise grâce que normalement et cette motivation entraîne des résultats souvent supérieurs à ceux des exercices effectués habituellement sur papier.

- Les logiciels permettent le calcul immédiat et la conservation des résultats des différentes séances. De plus, ils permettent au thérapeute et au patient de discuter de la séance à partir de tableaux représentant les résultats sous une forme agrégée. Ils permettent aussi de discuter de l'amélioration ou de la dégradation des résultats par rapports aux séances précédentes. Tout cela était bien évidemment possible avant mais était beaucoup plus lent et difficile et ne pouvait donc en général pas se faire à la fin de la séance considérée.

2.3.3 Apports du multimédia.

L'introduction du multimédia dans un logiciel de réapprentissage procure aussi un certain nombre d'avantages :

- Par rapport à la section 2.2.2.b, un des avantages immédiat du multimédia est de pouvoir rendre beaucoup plus concrets les exercices pilotés par logiciel grâce à la possibilité d'intégrer des photographies, des images, des sons, des voix digitalisées, des séquences vidéos, etc. Pour les logiciels personnalisés, le thérapeute peut y inclure du matériel audiovisuel provenant directement du malade ou de sa famille. Il n'est bien sur toujours pas possible de rendre ce matériel manipulable, ne s'agissant que de représentation, mais l'intérêt en est indéniable.

- L'utilisation du multimédia permet aussi une plus grande variété et richesse des exercices proposés. Le thérapeute n'est plus limité par un affichage en mode texte mais peut aussi inclure dans le logiciel les exercices qui exigeaient jusqu'à présent la manipulation de différentes sources sonores et visuelles (photographies, enregistrements audios, etc.).

- Le multimédia permet de rendre les interfaces beaucoup plus ergonomiques et donc par la même beaucoup plus accessibles aux malades, notamment aux personnes âgées. Les thérapeutes aussi pourront profiter de cet avantage et utiliser avec beaucoup plus d'aisance l'outil informatique.

- Le multimédia permet globalement de concentrer tous les outils informatiques classiques et audiovisuels au sein du même support. Le thérapeute pourra jongler très aisément avec ces outils sans aucun délai de mise en place et pourra ainsi passer d'un type d'exercice à un autre très rapidement. A terme, des appareils comme les magnétophones, les magnétoscopes, les projecteurs de diapositives, ... en fait tous les appareils audiovisuels de restitution ne seront plus nécessaires.

- Grâce aux effets visuels et sonores qu'il supporte, le multimédia permet de rendre les exercices plus ludiques et attrayants. La motivation et la satisfaction pour le patient d'utiliser l'outil informatique en sont d'autant plus accrues.

- Les progrès du multimédia devrait permettre de généraliser l'utilisation de ses fonctions d'entrées (auditives notamment) jusque là habituellement délaissées. Cela devrait permettre au malade lors d'exercices d'expression à voix haute, comme pour certains logiciels multimédia d'apprentissage des langues étrangères, d'enregistrer sa répétition du mot ou de

l'expression prononcée par la machine. Ensuite, le patient aurait le loisir de comparer les deux versions et ainsi de vérifier directement sa performance.

2.3.4 Troubles traités par l'informatique et le multimédia.

L'informatique est principalement utilisée pour rééduquer la mémoire des malades. Elle traite donc tous les troubles aphasiques ou associés qui ont leur origine dans une déficience de la mémoire.

Certains logiciels développés se concentrent particulièrement sur le réapprentissage de la mémoire lexicale par la réacquisition d'un vocabulaire courant permettant au patient une certaine autonomie de communication et ainsi de se débrouiller dans la vie de tous les jours. En Suède, nous avons eu l'occasion de voir fonctionner différents logiciels permettant le réapprentissage de mots courants. Un de ces logiciels permettait à partir d'une représentation d'une maison au milieu d'un jardin, de sélectionner un élément (par exemple, la cheminée) et d'en obtenir le nom écrit et parlé accompagné pour les éléments « vivants » d'un bruitage (miaulement pour un chat). Ce type de logiciel permet d'entraîner la capacité de dénomination du malade souffrant d'un manque du mot (cfr. sections 1.4.1.c et 1.4.3.c).

D'autres logiciels permettent le réentraînement de la mémoire à court terme du malade. Plusieurs logiciels observés au CTR (Centre de Traumatologie et de Réadaptation) de Bruxelles permettent ceci grâce à des exercices très simples de mémorisation. Un des logiciels présente une figure géométrique au patient pendant un certain laps de temps et, ensuite, ce dernier doit retrouver la bonne figure parmi un ensemble de figures d'abord totalement dissemblables et ensuite, au fur et à mesure du réapprentissage, similaires.
Comme nous l'avons déjà vu cet aspect adaptatif et évolutif des exercices proposés par les logiciels est fondamental.

Certains logiciels permettent aussi de réentraîner la structuration spatiale par l'utilisation de puzzle et d'autres exercices de constructions. Un des logiciels du CTR permet ce genre d'exercice et propose de surcroît une fonction d'aide au malade. Lorsque celui-ci est bloqué, cette aide indique, au moyen d'une petite animation, le positionnement correcte des différentes pièces du puzzle.

2.3.5 Caractéristiques des logiciels de réapprentissage.

Nous allons, dans cette section, présenter un ensemble de caractéristiques qui devraient se retrouver dans tout logiciel de réapprentissage :

- Les logiciels de réapprentissage doivent permettre une évolution de la difficulté des exercices afin de pouvoir s'adapter au niveau de chaque patient et de leur permettre de progresser continuellement.

- Les logiciels personnalisés, bien que représentant une minorité des logiciels développés pour le réapprentissage, doivent tenir compte des caractéristiques de l'aphasie du malade. Par exemple, un sujet souffrant d'alexie (cfr. section 1.4.4) ne pourra déchiffrer les affichages de texte. Un sujet souffrant d'un certain type de cécité verbale (cfr. section 1.4.4) pourra éprouver certaines difficultés à reconnaître les couleurs affichées et ainsi distinguer des zones aux couleurs différentes. Un autre souffrant de surdité verbale (cfr. section 1.4.2) pourra ne pas être capable de comprendre soit les messages sonores linguistiques (voix digitalisées), soit les messages sonores non-linguistiques (sons et musiques digitalisées).

- Les logiciels doivent aussi tenir compte des différents troubles non-aphasiques observés tel que les troubles de la mémoire à court terme. En effet, l'utilisation de toute interface présuppose de la part de l'utilisateur un minimum de capacité mnésique à court terme pour la manipulation des menus, des boutons de commande, etc.

- Lors de la conception de l'interface, il ne faut pas perdre de vue que les personnes aphasiques sont particulièrement sensibles aux effets de la fatigue et de la frustration. Le concepteur de l'application doit donc éviter de recourir systématiquement aux messages d'erreur. Lorsque c'est possible, une erreur devrait être signalée d'une manière moins « brutale ».

Partie 2 :
Les logiciels auteurs.

Dans un premier temps, nous allons présenter le logiciel auteur MacroMedia Director 4.0. Pour ce faire nous allons décrire l'environnement de travail proposé par le logiciel ainsi que ses modes d'utilisation. Nous parlerons, entre autres, du paradigme de programmation induit par son langage de conception dédié, LINGO.
Ensuite, nous traiterons des avantages et des inconvénients du logiciel pour la conception d'applications multimédias et nous essaierons de voir quelles améliorations il serait possible de lui apporter.

Chapitre 3. Le logiciel MacroMedia Director 4.0.

3.1 MacroMedia Director, un logiciel auteur.

Dans la suite du texte, nous emploierons le terme Director, nettement plus commode, pour désigner le logiciel auteur MacroMedia Director 4.0.

Le logiciel auteur Director a été développé par la société MacroMedia. Il s'agit d'une firme de San Francisco qui développe une gamme très complète de produits, couvrant à la fois la création d'animation 3D (avec le logiciel Macro Model) et les langages auteurs (comprenant aussi le logiciel auteur Authorware Professional). Director reste cependant la clé de voûte de l'offre de MacroMedia.

Director est un système auteur qui permet la réalisation d'applications multimédias interactives ou de films (non-interactifs). Grâce à sa très grande facilité d'utilisation et à la métaphore véhiculée par son interface, les développeurs d'applications sur ce logiciel ne doivent pas nécessairement être des informaticiens. Director a en effet été conçu pour être accessible à tous (artistes, étudiants, éducateurs, publicistes, etc.).

Les applications Director peuvent contenir à peu près tous les médias existants tel que les éléments audios et vidéos, les images, les musiques et sons digitalisés, les textes, les graphiques et les animations. Director possède un langage de programmation assez complet et puissant basé sur les scripts, LINGO.

Director, considéré comme un environnement de programmation multimédia, est un des logiciels les plus utilisés pour le développement d'applications multimédias, notamment sur CD-ROM. Il tourne sur les plates-formes Macintosh et PC, mais il est plutôt utilisé lors du développement sur la plate-forme Macintosh du fait de l'aptitude intrinsèque de cette dernière à

manipuler les éléments multimédias. De fait, Director permet la conversion automatique d'une application Macintosh en une application Windows.

Director appartient à la classe des systèmes auteurs se basant sur la métaphore de la table de montage : « *Les différents médias appartenant à l'application que l'on réalise sont affichés ou joués en fonction d'un scénario temporel qui est spécifié par son concepteur. Les objets apparaissent et disparaissent de l'écran selon un schéma temporel bien défini. On retrouve ici le concept de scène, d'acteurs et de scénario* »[6].

La table de montage proposée par Director est appelée *Score Window* (cfr. figure 3.1).

« *La table de montage est le premier niveau auquel le concepteur peut travailler, c'est-à-dire intégrer ses médias et définir leurs interactions mutuelles avec l'utilisateur final, sans devoir passer par un langage de programmation* »[7].

Director n'est pas le seul logiciel permettant la création d'applications multimédias. Authorware Professional est un autre exemple de logiciel auteur très employé (principalement dans le monde Macintosh). Authorware (disponible sur les plates-formes Macintosh et PC) à l'opposé de Director utilise des icones pour représenter le cheminement de l'application (cfr. figure 3.2).

« *Dans la logique du montage par diagramme d'icones, les écrans et les objets qui y sont liés s'enchaînent en fonction d'un organigramme défini par le concepteur de l'application. On part donc de la dynamique de l'interaction avec l'utilisateur pour construire, de façon visuelle, son application multimédia. La colonne vertébrale de l'application est un diagramme d'icones qui décrit les événements qui s'enchaînent dynamiquement* »[8].

[6] Goffinet, L., *Le multimédia et ses systèmes auteurs*, Institut d'Informatique, 1995, p. 7.
[7] Goffinet, L., *Le multimédia et ses systèmes auteurs*, Institut d'Informatique, 1995, p. 5.
[8] Goffinet, L., *Le multimédia et ses systèmes auteurs*, Institut d'Informatique, 1995, p. 8.

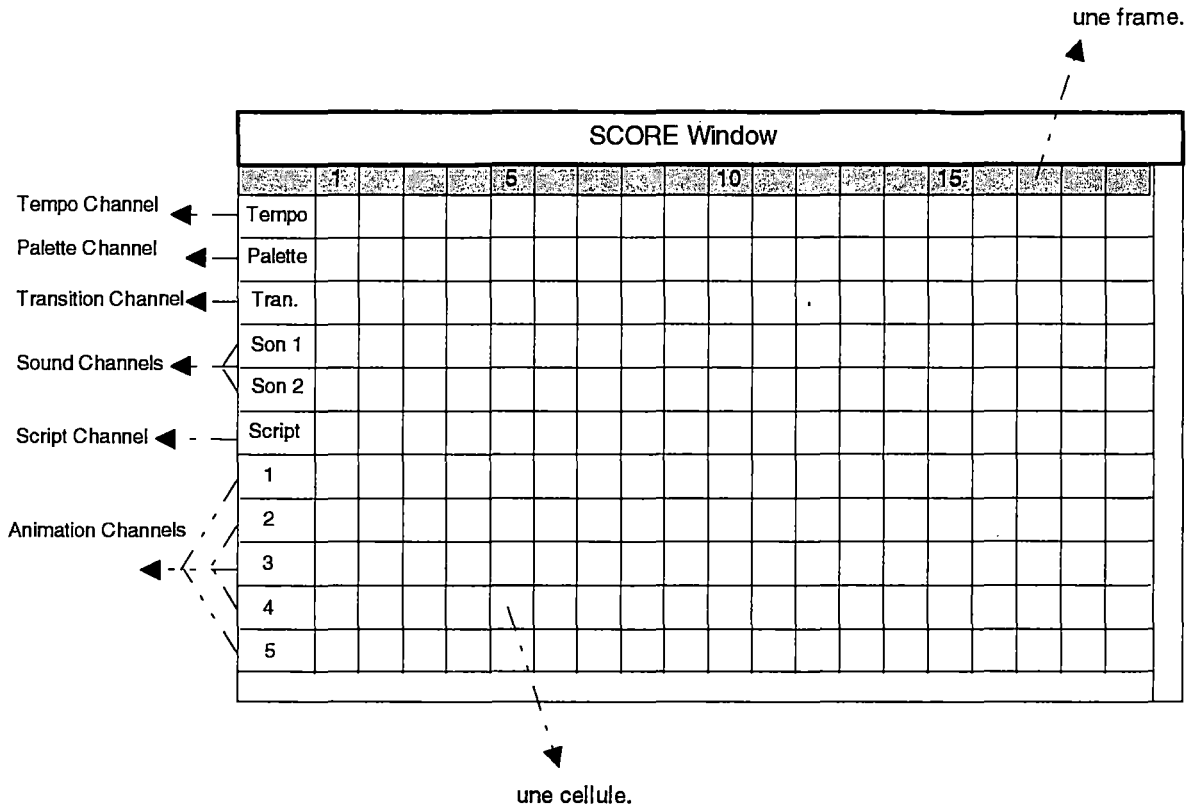


Figure 3.1. La *Score Window.*

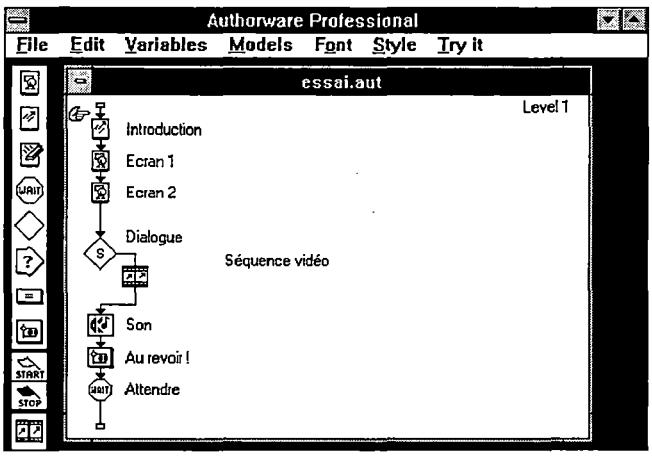


Figure 3.2. L'écran principal de Authorware Professional.

3.2 Description de l'environnement de travail.

L'interface utilisateur de Director repose sur un certain nombre de « fenêtres ».

a) Stage Window.

Cette fenêtre contient l'écran courant de l'application sur laquelle est en train de travailler le concepteur. On peut y voir les différentes images et objets graphiques que le concepteur a placés dans une colonne ou *frame* de la *Score Window* (figure 3.1). Toute modification apportée à un des éléments de la *Stage Window* est immédiatement reportée sur la *Score Window* et inversement.

b) Cast Window.

Cette fenêtre regroupe tous les éléments multimédias qui sont utilisés dans l'application (images, sons, objets graphiques, séquences vidéo, etc.) ainsi que les scripts (cfr. figure 3.3). Chaque case de cette fenêtre affiche une miniature ou un icone symbolisant l'élément contenu ainsi que son nom. La taille des cases est réglable de même que le nombre d'éléments affichables simultanément.
Chaque *Cast Member* possède un numéro identifiant unique. Un double clic sur le *Cast Member* permet de le visualiser et de le modifier si nécessaire (s'il s'agit d'une image, la *Paint Window* est appelée et se charge d'afficher l'image et de permettre sa modification, s'il s'agit d'un script, c'est l'éditeur de script qui est chargé, etc.).
La *Cast Window* peut contenir 64.000 éléments.

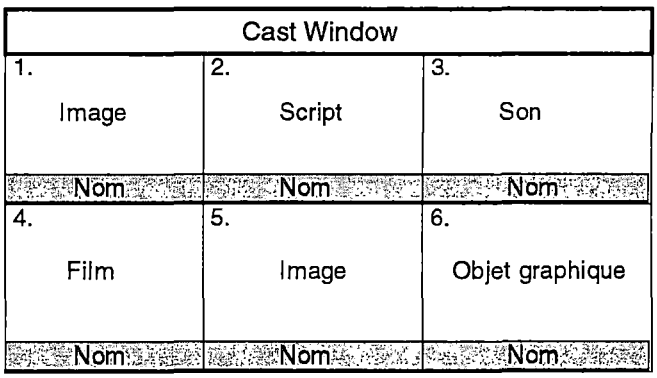


Figure 3.3. Un schéma typique de la *Cast Window*.

c) Score Window.

C'est sans aucun doute la fenêtre la plus importante de l'environnement de travail (figure 3.1).

Le modèle qui se trouve derrière une application Director est appelé son *Score*. Il s'agit d'une grille à deux dimensions. Les colonnes représentent les *frames* et sont ordonnées de telle manière que la progression du temps se fait de la gauche vers la droite, *frame* par *frame*. Les lignes sont appelées des *channels*. Il y en a de différents types et il n'est pas nécessaire de tous les utiliser dans chaque *frame*.
Une application Director est donc une suite structurée de *frames*, chaque *frame* pouvant contenir des éléments graphiques et sonores, des scripts, des transitions, etc. Les éléments insérés dans une *frame* représentent le contenu d'un écran de l'application en cours de développement.

Voici un aperçu des *Channels* disponibles :

- *Tempo Channel* : Les cellules se trouvant dans le *Tempo Channel* contiennent les instructions relatives à l'écoulement du temps dans l'application comme la vitesse de défilement en *frames* par seconde, une éventuelle pause, une synchronisation, etc.

  Ex. `Delay 3 secs`
  Cela signifie que lorsque l'exécution de l'application passe par la *frame* dont la cellule du *Tempo Channel* contient cette consigne, elle attendra trois secondes avant de poursuivre. (Ceci peut être utile lorsque l'on veut synchroniser une animation et une musique associée).

- *Palette Channel* : Le *Palette Channel* manipule les palettes de couleurs utilisées dans les différents écrans de l'application. Cela permet notamment de pouvoir afficher les images que l'on désire insérer dans l'application sans devoir se référer à une seule palette définie une fois pour toutes. Il existe au maximum une palette par *frame* et il n'est pas possible d'en changer en son sein.

- *Transition Channel* : Ce *Channel* spécifie le type de transition visuelle à effectuer lorsque l'exécution entre dans la *frame* où une transition a été placée.

  Ex. Pour passer d'un écran à un autre, Director propose une cinquantaine de transitions comme faire apparaître le deuxième écran en dissolvant le premier graduellement ou bien en exécutant un zoom avant.

- *Sound Channels* : Il y en a deux disponibles par *frame*. Ils sont utilisés pour ajouter un fond sonore à l'application. Ce peut être des voix digitalisées, des musiques ou des bruitages de différents formats de codage.

- *Script Channel* : Le *Script Channel* supporte l'interactivité et les autres comportements complexes globalement au niveau de la *frame* c'est à dire que le comportement spécifié dans le *Script Channel* n'est applicable que lorsque l'exécution se trouve dans la *frame* considérée (cfr. section 3.4.1).

- *Animation Channels* : Il y en a 48 par *frame*. Les *Animation Channels* spécifient les caractéristiques des *sprites* issus des *Cast Members* (ce n'est jamais le *Cast Member* qui est utilisé mais une copie). Dans la suite du texte, nous omettrons cette distinction entre *sprite* du *Cast Member* et *Cast Member* pour plus de lisibilité et de concision.
  Parmi les informations que l'on peut trouver dans ces cellules, on remarque le numéro du *Cast Member* (un *Cast Member* possède un numéro identifiant unique), son type (si c'est une image, un texte, un script, ...), sa position dans la *Stage Window* ainsi que le ou les effets qu'on lui a appliqués (par exemple, transparent à 50 % ou opaque).

Il existe une certaine hiérarchie d'affichage dans les *Animation Channels*. Un *Cast Member* se trouvant dans l'*Animation Channel* n° 2 apparaîtra à l'écran devant un *Cast Member* qui se trouve dans l'*Animation Channel* n° 1 et ainsi de suite. L'image de fond de l'écran sera donc la plupart du temps placée dans l'*Animation Channel* n° 1.

d) Paint Window.

Cette fenêtre est en fait un petit outil d'édition graphique permettant de créer les différents objets graphiques nécessaires sans devoir constamment passer de Director à un logiciel de création graphique. Cette fenêtre offre les possibilités que l'on retrouve généralement dans les programmes de dessin tel que PaintBrush (gomme, pinceau, vaporisateur, ...) ainsi que la possibilité d'utiliser différents effets (transparence, opacité, distorsions, changement d'échelle, etc.).

e) Control Panel.

Cette fenêtre est le panneau de contrôle de l'application et se présente sous la forme d'un panneau de commande de type « magnétoscope » avec ses touches 'Play', 'Stop', 'Pause', 'Rewind' et 'Forward'. Elle permet, à n'importe quel moment, de visualiser une partie ou l'entièreté de l'application. Ce type de panneau renforce l'utilisation de la métaphore de la table de montage.

f) Sound Window.

Elle sert pour l'écoute et l'enregistrement, via le micro incorporé dans le Macintosh, des signaux sonores. Cet outil, comparativement moins complet que la *Paint Window*, sera rarement utilisé pour l'édition.

g) Movie Window.

Cette fenêtre permet de visualiser les séquences vidéo et d'en choisir la partie qui va être placée dans l'application.

h) Palette Window.

Director offre une petite série de palettes prédéfinies. On y retrouve notamment la palette standard Windows, ce qui montre bien la volonté des concepteurs de permettre un passage aisé du monde Macintosh au monde PC. Si celles-ci ne conviennent pas, l'utilisateur de Director a la possibilité de les éditer ou d'en créer de nouvelles.

i) Message Window.

C'est une des fenêtres les plus utilisées lorsque l'on programme avec LINGO.
Elle sert au *debuggage* de l'application.
Ceci est très utile pour connaître le contenu des différentes variables ou bien pour savoir si l'application fait vraiment ce que l'on veut qu'elle fasse (la *Message Window* affiche le numéro et le nom de chaque script appelé lors de l'exécution). Elle peut rester présente à l'écran et être interrogée à n'importe quel moment.

j) Button Window.

Director offre la possibilité d'utiliser trois types de bouton prédéfinis : - le bouton normal,
- le bouton radio,
- la boîte à cocher.

Bien entendu, Director ne se limite pas à la simple gestion graphique de ces trois boutons standards ; leur comportement est également géré automatiquement. On peut toujours en créer de nouveaux suivant son imagination à l'aide de la *Paint Window* ou de tout autre programme graphique mais, dans ce cas, l'utilisateur va devoir en programmer la gestion par le biais de LINGO.

k) Text Window.

Director propose un petit éditeur de texte vectoriel permettant notamment de spécifier la police de caractère, la taille et l'emplacement du texte.

3.3 Souplesse d'utilisation.

Nous allons dans cette section illustrer la facilité d'utilisation de Director en proposant une description (dans les grandes lignes) de la manière de réaliser un petit « film » (sans l'utilisation de LINGO).
Ceci démontrera la très grande simplicité apportée par la métaphore de la table de montage.

❶ Avant de commencer la génération d'un « film », l'utilisateur doit disposer de ses différents éléments constitutifs (images, séquences vidéo, objets graphiques, sons). Pour notre petit exemple, il aura besoin d'une image de fond et de deux objets graphiques.

Prenons comme image de fond, une vue d'un désert et comme objets graphiques, un cactus et un soleil.
Ces différents éléments sont soit créés grâce à la *Paint Window* ou un autre logiciel graphique (si on est bon dessinateur), soit scannés à partir de photographies.

❷ L'utilisateur de Director prend le *Cast Member* du désert et le place dans l'*Animation Channel* n° 1 de la première *frame* (figure 3.4)
Une fois cela fait, cette image se trouve, par défaut, centrée dans la *Stage Window*.
Si cela s'avère nécessaire, l'utilisateur peut l'agrandir, la réduire et/ou la repositionner.
Toutes ces modifications sont automatiquement mises à jour dans la cellule de la *Score Window* contenant le *Cast Member*.

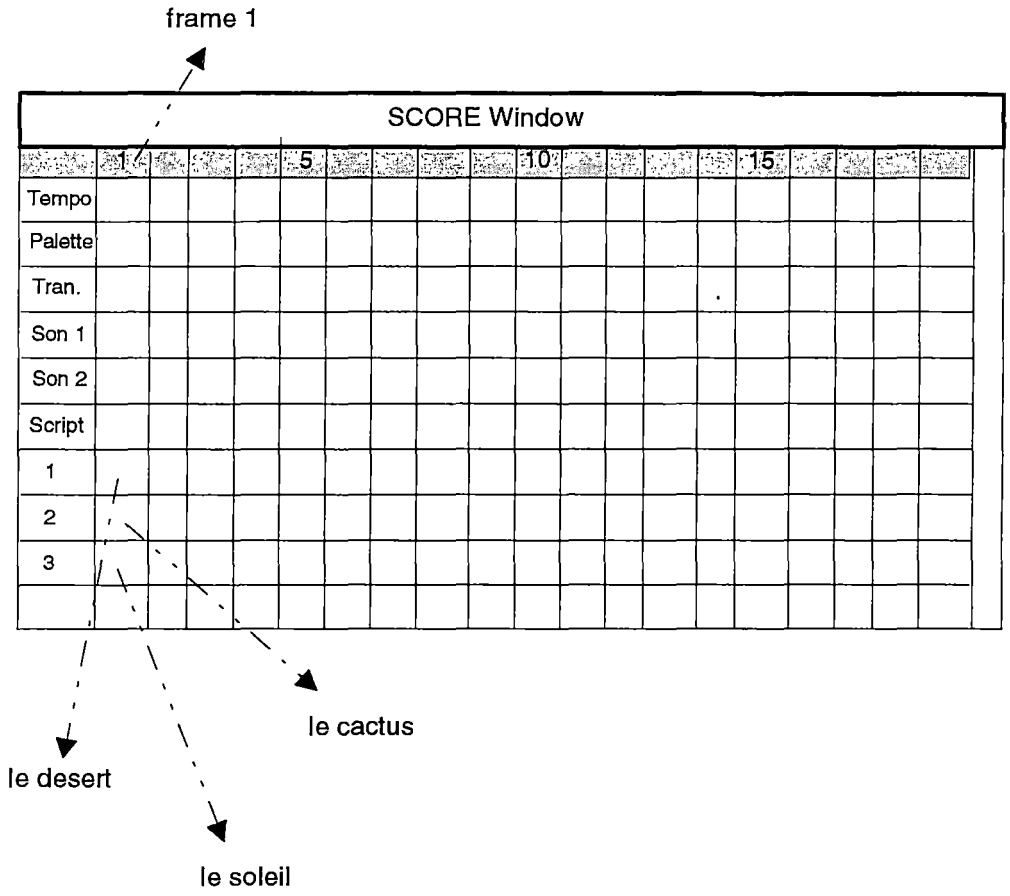


Figure 3.4. Un exemple d'utilisation de la *Score Window*.

Ensuite, il suffit de dupliquer (en utilisant le copier-coller) la cellule (contenant toutes les caractéristiques de l'image) sur le nombre de *frames* voulu (par exemple 40).

❸ L'utilisateur prend le *Cast Member* du cactus et le place dans la cellule de l'*Animation Channel* n° 2 de la *frame* 1 (figure 3.4). Ensuite, comme pour le désert, l'utilisateur modifiera la taille et dupliquera la cellule. L'image du désert et celle du cactus sont donc placé sur 40 *frames* respectivement à des positions constantes. Ces deux éléments représentent donc le décor statique de l'animation qui va être ajoutée.

❹ Comme pour les deux éléments précédents, l'utilisateur place le soleil dans la cellule de l'*Animation Channel* n° 3 de la *frame* 1 (figure 3.4).
Il positionne le soleil au commencement de sa montée au zénith (au milieu à gauche).
Il reprend le *Cast Member* du soleil pour le placer dans la cellule de l'*Animation Channel* n° 3 mais cette fois de la *frame* 40. Il le positionne au zénith.

Il a donc ainsi les deux positions extrêmes de l'animation.
Il va ensuite sélectionner les 40 *frames* de l'*Animation Channel* n° 3 et utiliser une fonction qui réalisera le calcul automatique des positions intermédiaires de l'animation.

❻ L'utilisateur n'a plus qu'à utiliser le *Control Panel* et cliquer sur 'Play', le soleil effectuera sa montée au zénith.

Director, comme on l'a vu plus haut, est un logiciel auteur accessible à tous (aux informaticiens comme aux non-informaticiens). Cependant, la non-utilisation du langage de programmation LINGO limitera la complexité des applications qui pourront être développées. En effet, toutes les applications mettant en place une interactivité requièrent l'utilisation de LINGO. Le developpeur n'utilisant pas ce langage ne pourra produire que des films (sans interaction) qui consisteront en de simples animations ou en une suite d'images séparées par des transitions (*slide show*).
Pour réaliser des applications plus complexes ou plus interactives, l'utilisation de LINGO est donc incontournable.

3.4 LINGO.

LINGO est un langage de programmation relativement simple (pour un informaticien) et de haut niveau permettant au concepteur de retirer la presque totalité du contrôle de l'application à la *Score Window*. LINGO se fonde sur le paradigme des systèmes « pilotés par événements » et est, de ce fait, un langage de scripts. A chaque événement se produisant dans le système, le concepteur peut associer un script. A chaque type d'événement correspond un type de script.

3.4.1 Le script.

Un script est un morceau de code exécuté lors de l'occurrence d'un événement particulier.

Un script de Director a la forme suivante :

```
on événement
  code à exécuter
end
```

« C'est le niveau auquel on peut écrire, dans un langage de haut niveau fourni avec la plupart des systèmes auteurs, des procédures permettent de spécifier un comportement plus

complexe en réaction à une action bien précise de l'utilisateur. Ces scripts apportent puissance et souplesse à l'application »[9].

Dans Director, il existe différents niveaux de scripts :

- le script global : ce niveau de script porte sur l'ensemble du film. Il existe deux types de scripts de ce niveau, les scripts `on startmovie` et `on endmovie`.

- le *handler* : ce type de script recouvre les scripts exécutés à la suite d'événements définis par l'utilisateur. Ce type de script réalise la même fonction qu'une procédure en Pascal. Le nom donné à un événement est utilisé pour appeler le *handler* à partir d'un autre script et il est possible de passer des paramètres. Ces scripts ont une portée globale et peuvent donc être appelés de n'importe où.

- le *frame script* : ce type de script apparaît dans le *Script Channel* et porte sur l'entièreté de la *frame* où il est placé. Ce sont les *scripts* `on enterframe` (lorsque l'exécution entre dans une *frame*, les différentes instructions contenues dans ce script sont exécutées avant toute chose) et `on exitframe` (même comportement mais cette fois lorsque l'exécution quitte la *frame*).

- le *score script* : c'est le type de script qui peut être associé à un *Cast Member* placé dans un des 48 *Animation Channels*. Ce type de script est déclenché lorsque l'utilisateur effectue un clic de souris sur le *Cast Member* en question. Ce type recouvre les scripts `on mousedown` (exécuté lorsque le bouton de la souris est enfoncé) et `on mouseup` (exécuté lorsque le bouton de la souris, après avoir été enfoncé, est relâché).

Ex.
```
on mouseup
  play frame 10
end
```

L'exécution de ce script effectuera un branchement vers la *frame* n° 10.

[9] Goffinet, L., *Le multimédia et ses systèmes auteurs*, Institut d'Informatique, 1995, p. 5.

- le *Cast Member script* : il est possible d'associer un script directement à un *Cast Member*. Cela permet de lui adjoindre un comportement global « par défaut ». En effet, si un *sprite* issu du *Cast Member* est l'objet d'une interaction et si aucun *score script* ne lui est associé dans la *frame*, c'est le script du *Cast Member* qui sera exécuté.

3.4.2 L'événement.

Lorsqu'un événement survient dans une *frame*, Director génère un message décrivant l'événement. Le message est ensuite envoyé aux différents types de scripts. Pour chaque type de script, Director essaie d'identifier un script répondant à l'événement particulier. S'il n'en trouve pas, il passe au type de script suivant, sinon il exécute le script et le message est détruit à moins que le contraire ne soit explicitement spécifié dans le script exécuté.
Les types de scripts sont organisés hiérarchiquement. Un message généré par un événement suit cet ordre hiérarchique (cfr. figure 3.5). Dans cette figure, un événement « bouton de souris enfoncé » est envoyé d'abord aux scripts attachés à un *sprite* (*score scripts*), puis à ceux attachés à un *Cast Member* (*Cast Member scripts*), puis à ceux attachés à la *frame* (*frame script*) et enfin aux *movie scripts* (scripts globaux et *handlers*).

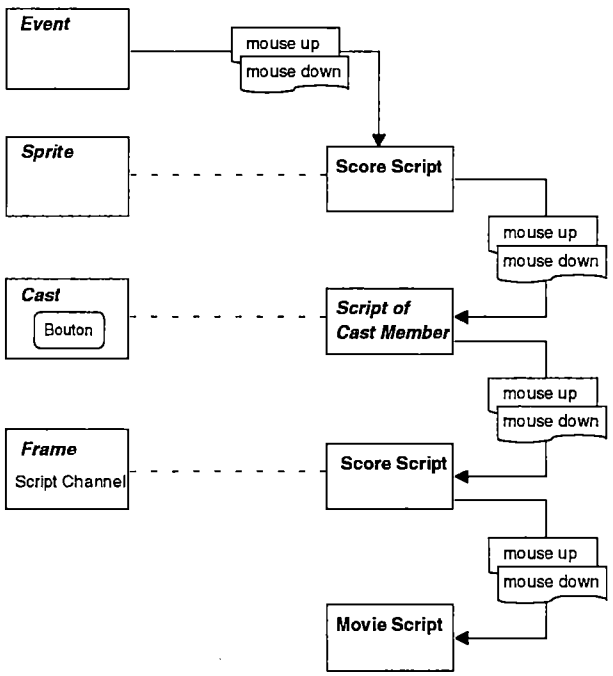


Figure 3.5. Le diagrame de passage des messages.

3.5 Avantages et inconvénients de MacroMedia Director 4.0.

Dans cette section, nous allons essayer de présenter les avantages et les inconvénients d'un logiciel auteur comme Director par rapport aux langages de programmation conventionnels en ce qui concerne la création d'applications multimédias.

3.5.1. Avantages.

a) La richesse.

Director offre une large gamme d'outils de traitement des différents médias (outil de dessin, outil pour contrôler une vidéo, outil permettant la création de nouvelles palettes de couleurs, outil pour contrôler et générer le son). Ces outils, quoique relativement efficaces, ne sont pas les plus performants dans leur catégorie. Néanmoins, ils suffisent largement lorsque l'on n'adopte pas une approche professionnelle de la création d'applications multimédias.

De plus, Director permet l'intégration de manière relativement transparente (il reconnaît les formats les plus utilisés) des sons, images, objets graphiques, séquences vidéos, ... créés à l'aide d'autres logiciels.

Par exemple, pour l'application « Gun's Garden » (cfr. chapitre 6), l'image d'introduction a été conçue sur le logiciel Alias Sketch (logiciel de dessin permettant la création d'images en trois dimensions). De même, les sons et la musique de l'application ont été digitalisés et édités à partir de Sound Edit Pro.
Par contre, les différents dessins de boutons, cadres, ... ont été réalisés à l'aide de l'outil de dessin fourni par Director (la *Paint Window*).

Tous ces différents médias peuvent être gérés directement et de manière transparente à partir d'un tableau de contrôle, la *Score Window,* ou à partir de LINGO. La puissance et la facilité d'utilisation de ce dernier permettent la réalisation d'applications interactives complexes et puissantes.
Director offre donc un environnement de travail intégré et complet libérant le concepteur des inutiles allers et retours entre logiciels de développement et d'édition.

b) La convivialité.

Un grand nombre de personnes peuvent utiliser Director même si elles n'ont aucune notion de programmation. En effet, il n'est pas utile de comprendre le langage de programmation LINGO pour pouvoir réaliser des applications multimédias. Il est ainsi possible de réaliser très aisément des slide shows, des films de présentation, etc.
Il est aussi possible pour peu que le concepteur connaisse une ou deux instructions de LINGO de créer des applications interactives simples.
Director est donc destiné à une grande variété d'utilisateur, du plus néophyte au plus aguerri.

c) La simplicité.

Director propose à l'utilisateur un certain nombre de fenêtres ayant chacune une fonction propre.

De plus, Director a été conçu en prenant une approche de studio de montage. Grâce à ces deux techniques, la création et le positionnement d'un élément se font très simplement (voir la section 3.3).
Tous les changements (taille, position) effectués par l'utilisateur sur la *Stage Window* sont automatiquement mis à jour dans la *Score Window*. Ce qui est intéressant, c'est que ces changements (surtout les changements liés à la taille) n'affectent que la ou les cellules sélectionnée(s) de la *Score Window*. Ils n'affectent nullement le *Cast Member* contenant l'élément (seule une copie du *Cast Member* est placée dans la *Stage Window*).

Ex. Grâce à cela, un objet graphique représentant un cercle peut servir à afficher un petit, un moyen et un grand cercle.

LINGO est assez simple à apprendre et avec un minimum de volonté et le manuel d'utilisation, ce langage ne devrait pas poser trop de problèmes aux différents développeurs et en particulier aux non-informaticiens (quoiqu'un minimum de connaissance de la programmation soit fortement conseillé). De plus, un compilateur automatique est intégré. Chaque fois que l'on termine l'édition d'un script, celui-ci est automatiquement compilé et les erreurs de syntaxe sont instantanément signalées.

La *Score Window*, comme toutes les autres fenêtres, est très simple à manipuler. Elle présente toutes les informations nécessaires à l'utilisateur de manière synthétique.

d) Une visualisation rapide.

Director permet une visualisation très rapide de ce que l'on a déjà réalisé. Il suffit pour cela d'utiliser le *Control Panel.* Cela permet, en outre de ne visualiser qu'une séquence particulière sans devoir exécuter entièrement l'application. En effet, le concepteur peut « forcer » le commencement de l'exécution à n'importe quelle *frame.*

e) Interactivité.

Director propose toute une série d'objets graphiques standards permettant de gérer l'interactivité. Outre les boutons de commande déjà présentés, il permet la construction de menus déroulants standards et la gestion des entrées au clavier. Il ne reste plus à l'utilisateur qu'à spécifier l'action à accomplir en réponse au moyen de LINGO.

f) Une gestion directe de la mémoire.

Director permet une gestion directe de la mémoire allouée aux éléments.
Dans les versions antérieures, Director récupérait l'espace mémoire inutilisé de manière globale. Il attendait la fin d'une séquence pour reprendre de la mémoire libre.
A présent, il est possible de lui indiquer qu'il est autorisé à évacuer un élément de la mémoire lorsque celui-ci n'est plus utilisé (par exemple, l'utilisateur peut attribuer un niveau de priorité à un *Cast Member* ; Ceux de bas niveau seront évacués avant ceux de plus haut niveau).
Au lieu d'une récupération globale lente, Director libère ainsi plus rapidement l'espace mémoire inutilisé sans inférer sur le bon déroulement de l'application.

g) Une aide à tous les niveaux.

Après avoir sélectionné l'option adéquate dans le menu d'aide, il suffit de cliquer sur une zone de l'écran de travail pour obtenir de l'information s'y rapportant. C'est une aide très utile pour les débutants qui ne devront pas toujours chercher les explications dans le manuel. De plus, l'aide en ligne reprend toutes les instructions de LINGO accompagnées de plusieurs exemples.

h) Une intégration de modules hétérogènes.

LINGO permet l'intégration de parties de programme écrits et compilés dans un autre langage (C++, Turbo Pascal). Cela permet de pouvoir réutiliser des parties de programmes sans devoir les réécrire dans le langage LINGO. Ainsi, il est possible de passer outre certaines limitations du langage (cfr. section 3.5.2).

i) Possibilité d'échapper aux interfaces WIMP classiques.

En effet, les environnements classiques (Windows et System 7) supportent la plupart du temps des interfaces de type WIMP. Les interfaces des applications développées dans ces environnements sont conçues à partir de librairies de fonctions implémentées dans les langages de programmation utilisés . Le résultat en est que l'on peut constater une certaine homogénéité peu stimulante des logiciels développés. Lorsque, par exemple, on examine les premiers logiciels multimédias sous Windows, on constate qu'ils se contentent pour la plupart d'utiliser l'interface Windows sans plus (notamment les encyclopédies multimédias).
Des logiciels comme Director permettent de concevoir assez facilement des interfaces totalement originales tout en permettant leur fonctionnement dans un environnement standard.

3.5.2 Inconvénients.

a) La lenteur.

Director demande, pour tourner d'une manière correcte, un Macintosh (ou un PC) puissant (un Quadra ou un Power PC). De fait, Director est un outil général de création d'applications multimédias et comme tous les outils de conception de haut niveau, il compile des applications largement moins rapides que si elles avaient été écrites dans un langage de programmation plus proche de la machine comme le C++. Cette limitation réserve donc l'utilisation de Director aux développeurs équipés de machines hauts de gamme.

rem : cette lenteur est aussi un inconvénient d'Authorware Professional (un autre système largement utilisé) et de probablement tous les logiciels auteurs à usage général.

b) Une relative pauvreté du langage LINGO.

Il existe une carence sensible du langage LINGO principalement au niveau des structures de données. Les seules structures de données offertes par LINGO sont les listes simples et les listes avec propriété.
La liste simple correspond à un tableau à une dimension de n éléments.
La liste avec propriétés correspond à une table c'est à dire un tableau à n lignes et deux colonnes.

Cette pauvreté implique une multiplication des listes lorsque des structures de données complexes sont nécessaires, ce qui peut parfois mener à une certaine confusion et un manque de lisibilité des programmes. En effet, pour mettre en place une simple matrice à trois dimensions, il est nécessaire d'utiliser une série de listes avec propriétés imbriquées.

c) Le module de compilation.

Le module de compilation qui accompagne LINGO est efficace pour déceler les erreurs mais lorsqu'elles se produisent, il se contente de les signaler sans indiquer l'endroit de leur occurrence dans le code. Il faut alors la chercher dans le script ce qui peut parfois se révéler assez rebutant.

d) Difficulté de lecture d'une Score Window.

Bien qu'il soit facile d'utiliser cette fenêtre, il est paradoxal qu'une personne n'ayant pas participé à l'élaboration de l'application ait beaucoup de mal à en lire le « scénario ».
Une application Director n'est, en effet, pas séquentielle puisqu'elle est composée de bouts de programme disparates et c'est l'occurrence des événements dans le système qui conditionne l'ordonnancement de ces morceaux de codes. La représentation utilisée par les concepteurs de Director est parfaitement adaptée à la conception d'animations et de films interactifs mais ne permet pas de structurer convenablement une application interactive complexe.

Afin de rendre les choses un peu plus lisibles, Director permet d'associer un label à une *frame*. Dans le cas d'un branchement, on utilisera l'instruction suivante :

Ex.
```
on mouseup
  play "toto"
end
```

`toto` étant le label associé à la *frame* 150, par exemple.

e) Interactivité de zone.

Ce problème crucial n'est pas propre à Director mais est commun à tous les logiciels auteurs.

Ex. On souhaite qu'une zone d'une image puisse permettre une interaction (par exemple, sur une image représentant une forêt, le fait de cliquer sur un arbre déclenche l'affichage de ses caractéristiques).

Les différents moyens offerts par Director pour permettre cette « interactivité de zone » sont :

- l'utilisation d'un bouton transparent c'est-à-dire que l'on essaie plus ou moins d'encadrer la zone avec une forme géométrique (polygone régulier, cercle, ...) transparente qui jouera le rôle de bouton.
  Ainsi, lorsque l'utilisateur cliquera sur un arbre, ce sera le script associé à ce bouton transparent qui sera déclenché.

- l'utilisation de plusieurs de ces formes mais de tailles très réduites afin d'épouser le mieux possible le contour de la zone. Cependant, cela multiplie le nombre de *channels* nécessaires pour permettre l'interaction.

- le dessin d'une forme qui épouse parfaitement les contours de la zone. Celui-ci peut être réalisé au moyen d'un logiciel graphique ou de la *Paint Window* (sa réalisation peut se révéler assez fastidieuse si la zone est complexe). Pour dessiner cette forme, il suffit d'éditer l'image la contenant et de la détourer au pixel près. Director permet, ensuite, de la rendre transparente et de la faire coïncider avec la zone désirée sur la *Stage Window*.

- le calcul de toutes les coordonnées de la zone et leur stockage dans une liste. Dès que l'utilisateur effectuera un clic de souris, LINGO vérifiera si le pointeur de la souris se trouve dans la zone visée en comparant les

coordonnées de la souris avec celles se trouvant dans la liste (il est extrêmement pénible de calculer toutes les coordonnées de toutes les zones). Cependant, cette méthode pose les mêmes problèmes que les deux premières mais avec l'avantage de ne pas utiliser le moindre *Animation Channel.*

En résumé, il est assez difficile de délimiter parfaitement le contour d'une zone que l'on veut rendre interactive.

Cela peut se révéler être un inconvénient lors de l'utilisation de ces applications par des personnes souffrant d'un handicap.
En effet, un sentiment de frustration risque de survenir chez une personne qui, en cliquant sur le bord d'une zone interactive délimitant imprécisément une portion d'image, ne constate aucune réaction de la part du système. Inversement, un problème similaire se produit si cette personne clique à coté d'une portion d'image et que, malgré cela, une action liée à cette portion est déclenchée.
Ce problème est difficilement soluble car soit les zones sont très précisément délimitées et la surcharge de travail pour les concepteurs peut devenir rédhibitoire, soit les zones sont grossièrement délimitées et le risque de frustration chez l'utilisateur peut devenir très important.

3.6 Suggestion d'améliorations.

Nous allons tenter de suggérer dans cette section un certain nombre d'améliorations qui, à notre avis, sont tout à fait envisageables pour un logiciel comme Director. Pour ce faire, nous nous basons sur une collection d'articles récents traitant des progrès de l'outil multimédia et sur notre expérience personnelle. Certains de ces articles peuvent être trouvés en [17], [21] et [23].

a) Une structuration de l'animation.

Cette structuration permettrait une lisibilité accrue du cheminement de l'application (comme dans Authorware Professional).

Il serait assez intéressant que Director fournissent en même temps que l'approche « table de montage » (qui facilite grandement la conception d'une application de type « animation ») une

structuration plus hiérarchique (qui offre l'avantage de pouvoir visualiser d'un seul coup d'oeil la représentation des branchements ainsi que des tests interactifs).

b) L'interactivité de zone.

Director pourrait fournir un moyen de rapidement détourer une zone d'une image. Ceci serait utile pour les développeurs d'application pour les utilisateurs souffrant d'un handicap. Certains logiciels graphiques permettent de délimiter d'une manière relativement aisée et rapide les traits constitutifs d'une image complexe. Director pourrait inclure une version adaptée de cette fonction. Ensuite, le concepteur n'aurait qu'à sélectionner une partie de l'image ainsi épurée et la définir comme bouton graphique.

c) Structures de donnée.

Le langage de programmation LINGO est un langage puissant et simple à apprendre.
Toutefois, il lui manque, comme nous l'avons vu plus haut, la possibilité d'utiliser des structures de données complexes. Fournir ces structures est probablement l'amélioration la plus simple à implémenter.

d) Edition de graphismes vectoriels.

Director, dans sa forme actuelle, ne gère que des graphismes *bitmap*. Bien qu'il permette au concepteur de redimensionner les images de ce type directement sur la *Stage Window*, le résultat n'est cependant pas toujours probant.
En effet, l'image résultante est souvent déformée ; lorsqu'elle est grossie, les pixels deviennent apparents. Les images *bitmap* ne se prêtent pas aisément à ce genre de manipulation.
Pour pallier ce problème, Director devrait proposer au concepteur la possibilité de créer des images vectorielles. Ce pourrait être d'une grande utilité pour la conception des différents objets graphiques pilotant l'interactivité comme les boutons, les fenêtres, les menus, etc.
La possibilité d'ajouter par la suite des textures *bitmap* à ces objets vectoriels simples constituerait également un plus indéniable et permettrait d'arriver aux mêmes qualités visuelles qu'une image *bitmap* pure.

e) Une conception Orientée-Objet des objets graphiques.

Director ne permet pas de concevoir des objets graphiques complexes. En effet, tout concepteur désirant par exemple concevoir une fenêtre doit la simuler par un assemblage de ses différents composants (boutons, parties du cadre, éléments textuels, etc.). S'il est un tant soit peu prévoyant, il peut, via LINGO, définir la localisation de ces différents composants de manière relative (c'est-à-dire par rapport à un élément de référence).
Cependant, s'il veut modifier la taille de la fenêtre ou d'un de ses éléments ou s'il veut ajouter certains éléments ou effets, il doit bien souvent repositionner et redessiner manuellement la plupart des composants de la fenêtre. Comme les changements de cet ordre sont très fréquents, il risque de perdre un temps précieux en « édition ».

Une approche orientée-objet permettrait de rendre les éléments d'une fenêtre solidaires et permettrait de dupliquer celle-ci dynamiquement autant de fois que désiré. Si la possibilité de créer certains éléments vectoriels, comme le cadre, était fournie, il serait possible de la rédimensionner à loisir et les autres éléments se repositionneraient automatiquement.

Director propose bien la possibilité de définir des objets mais ceux-ci doivent être simples c'est-à-dire doivent contenir un seul élément graphique, ce qui est très insuffisant.

f) Intégration d'objets 3D.

C'est probablement une des améliorations les plus difficiles à mettre en oeuvre mais elle pourrait faciliter la vie du concepteur en accroissant sensiblement la puissance de conception graphique de Director.
Si durant la phase de conception, Director était capable de manipuler des objets en trois dimensions, les possibilités de réalisation de l'interface s'en trouveraient sensiblement accrues. On peut imaginer la possibilité de travailler, lors de la conception, avec des objets en fil de fer (on gagne une dimension sur les objets vectoriels présentés plus haut) et de leur assigner des textures.
Le concepteur pourrait modifier la taille, l'angle de vue, l'éclairage d'un objet très facilement et, ensuite, une fois ces paramètres définitivement fixés, effectuer un *rendering* de l'objet (bien entendu, le modèle 3D serait conservé pour modifications ultérieures).

g) Guide des couleurs.

Il serait intéressant d'adjoindre aux outils de créations graphiques proposés par Director une sorte de « guide des couleurs ». Ce guide pourrait entre autres évaluer l'harmonie des couleurs qui additionnées doivent donner un gris neutre si elles ont bien été choisies.
Ce guide pourrait aussi conseiller le graphiste non-professionnel dans ses choix et utilisations de couleurs (comment établir des contrastes, comment juxtaposer des couleurs vives et ternes, comment éclaircir ou foncer une certaine couleur, etc.).

3.7 Utilisation de Director par des thérapeutes.

Pour les non-informaticiens, Director est un outil parfait de conception d'animations. Pour ceux qui ont le temps de se pencher sur LINGO, il peut devenir un outil puissant de création de logiciels multimédias.
Malheureusement, c'est rarement le cas des thérapeutes. De plus, ils doivent concevoir une interface qui va permettre de piloter les exercices, avant même de pouvoir aborder leur conception. C'est cette interface qui pose problème. Elle exige pour sa réalisation une bonne connaissance de LINGO. Or, les thérapeutes ont des besoins ponctuels en termes d'exercices. Ils traitent des malades qui vont avoir besoin d'un type particulier d'exercice pour la semaine suivante. Ils se rabattent alors sur des logiciels du commerce ou développés pour d'autres personnes qui, de toute façon, seront en partie inadaptés au cas particulier qui les intéresse.

Parfois, certains d'entre eux possèdent des connaissances en informatique et sont à même de concevoir les interfaces et les exercices désirés. Cependant, ils sont peu nombreux et, de toute façon, les délais dont ils disposent pour mettre au point ces exercices sont en général trop courts car il subsiste le problème épineux de l'édition. C'est en fait, pour quelqu'un qui connaît l'outil de programmation, ce qui prend le plus de temps, du moins pour un non-graphiste. Par contre, si le thérapeute, en plus d'être un informaticien amateur se doublait d'un graphiste, ... Mais on se rend compte assez vite de la difficulté d'une telle chose. Ce qu'il faudrait, c'est permettre aux thérapeutes de se concentrer sur le contenu des exercices en les déchargeant au maximum de la réalisation de l'interface.

La solution idéale serait, pour les thérapeutes, de disposer d'un outil spécialisé de conception d'exercices multimédias de réapprentissage, un logiciel auteur dédié.

Chapitre 4. Architecture pour un logiciel auteur dédié.

Dans ce chapitre, nous allons tenter de dégager de manière informelle les spécifications d'un logiciel auteur dédié à la conception d'applications pour personnes aphasiques. Pour ce faire, nous déterminerons un cahier des charges type pour ce genre de logiciel. Ce cahier des charges ne traite que des parties présentation et contrôle de la création d'applications et laisse de coté la partie abstraction (nous faisons ici référence au modèle PAC).

4.1 Cahier des charges.

Dans cette section, nous allons décrire ce que devrait permettre, selon nous, un logiciel auteur destiné à l'usage des thérapeutes. Nous nous basons pour ce faire sur notre propre expérience d'utilisation d'un logiciel auteur (MacroMedia Director 4.0) et de programmation d'applications pour personnes aphasiques. Le but ultime de ce genre de logiciel est de limiter au maximum l'utilisation d'un quelconque langage de programmation qui sera de toute façon toujours trop compliqué pour des personnes n'ayant pas le temps ou la volonté de l'apprendre.

Nous allons d'abord présenter les outils d'aide à la conception nécessaires et ensuite les différents objets que le logiciel auteur devrait pouvoir traiter et les actions qui devraient être permises par et sur ces objets.

4.1.1 Les outils intégrés.

Le genre de logiciel auteur que nous considérons ici est destiné aux centres de traitements du handicap dont les budgets sont le plus souvent très serrés.
C'est pourquoi, outre le parc de machines installées très réduit, les logiciels disponibles dans ces centres sont souvent très peu nombreux et ne concernent que très rarement l'édition multimédia.
Il est donc primordial qu'un logiciel auteur de ce genre propose l'ensemble des fonctions nécessaires à l'importation et à l'édition des différents médias que l'on veut inclure dans une application. Ces fonctions peuvent être réalisée par des outils de PAO (Publication Assistée par Ordinateur), de DAO (Dessin Assisté par Ordinateur), de MAO (Musique Assistée par Ordinateur) et de génération d'animations intégrés dans le logiciel auteur. Leur présence doit permettre une puissance d'édition suffisante pour garantir une parfaite autonomie du logiciel.

a) Outil de PAO.

Cet outil ne doit pas renfermer l'ensemble des fonctionnalités d'un traitement de texte à part entière. Il lui suffit de permettre l'édition de textes au format vectoriel, de permettre le choix de la police, de la taille des caractères et d'autres paramètres du même ordre. Les possibilités offertes sont donc très réduites mais suffisantes pour la génération de messages et autres libellés destinés au futur utilisateur de l'application en cours de développement.

b) Outil de DAO.

Dans un logiciel auteur propre à la conception d'applications multimédias, cet outil est probablement le plus important car c'est sur lui que repose la presque totalité de la conception graphique de l'application.
C'est pourquoi il doit être relativement complet, ne laissant de coté que les fonctions destinées plus spécialement aux graphistes professionnels.
Cet outil doit supporter notamment les trois grandes étapes[10] de toute création graphique, à savoir :

- l'acquisition : il s'agit de proposer à l'utilisateur la capture de photographies, images et autres dessins sur support papier au moyen d'un scanner à plat ou à main. Cette fonction permet ainsi aux thérapeutes de travailler à partir de matériels visuels fournit par le malade ou sa famille et relatif à son environnement prémorbide.

- la conversion : cette fonction doit permettre de récupérer au sein du logiciel des images, photographies, etc. déjà numérisées dans un certain format de codage et de les convertir dans un autre format (dans le cas qui nous occupe, cette fonction revêt assez peu d'importance mais cela permet de choisir un format assurant une meilleure compression du fichier graphique et ainsi de limiter la place occupée par ce fichier).

- l'édition : cette fonction est la clé de voûte de tout outil de DAO. Il permet soit de créer son propre matériel visuel soit de retoucher le matériel importé ou scanné.

[10]Gibbs, S. et Tsichritzis, D., *Multimedia Programming*, Addison-Wesley, 1994, p. 10.

c) Outil de MAO.

Cet outil doit permettre de traiter les sons digitalisés de tout type. La source audio pouvant aussi bien en être la voix qu'une musique ou un bruit quelconque. Pour supporter cela, les trois étapes de l'outil de DAO sont parfaitement transposables à la MAO. L'acquisition peut se faire à partir d'un magnétophone, d'une platine laser ou de tout autre équipement audio. L'édition doit avant tout permettre de retoucher le matériel auditif importé ou digitalisé (Il ne s'agit donc pas ici de « création » à proprement parlé).

d) Outil de génération d'animations.

Il s'agit ici de créer des animations toutes simples et pas d'obtenir des résultats auxquels pourraient prétendre des stations Silicon Graphics. Cet outil doit notamment permettre d'effectuer un *morphing* entre deux photographies, de calculer les images intermédiaires entre deux images de référence, etc.

4.1.2 Les objets traités par le logiciel.

Les objets de contrôle.

a) La fenêtre.

Le concepteur de l'application doit être capable de lier des objets au sein d'une même fenêtre. Ceci afin de lui permettre de repositionner ultérieurement cette dernière sans autres modifications. La fenêtre peut ainsi être considérée comme un méta-objet.

- Le concepteur doit être capable de modifier à loisir la taille et la position de la fenêtre. Il doit aussi pouvoir choisir la couleur ou la texture de cette dernière ainsi que d'autres paramètres d'affichage.

- Il doit pouvoir définir dans la fenêtre des zones d'édition pour le texte, des zones d'affichage pour les images et les animations. Il peut aussi y placer les différents boutons de commande.

- Le logiciel doit lui permettre de rendre la fenêtre modale s'il le désire c'est-à-dire qu'un clic en dehors de la fenêtre demeure sans effet.

b) Le bouton de commande.

Les boutons de commande peuvent être élaborés par l'outil de dessin offert par le logiciel ou par des outils externes puis importés dans le logiciel. Chaque bouton de commande comprend deux positions donc deux images, une image en position haute et une image en position basse. Un comportement par défaut leur est associé gérant le passage d'une position à l'autre. Ce comportement est compatible avec les boutons standards du System 7 et de Windows mais il peut éventuellement être redéfini. Ces boutons sont donc cliquables par le moyen d'interaction utilisé.

- Le concepteur doit pouvoir soit placer ses boutons dans une fenêtre et les lier à celle-ci (c'est-à-dire que les coordonnées du bouton seront relatives à la position de la fenêtre) soit les placer dans un écran (dans ce cas, les coordonnées du bouton seront absolues).

- Le concepteur de l'application doit pouvoir leur associer une ou plusieurs actions parmi celles qui sont proposées.
  Les actions permises sont les suivantes :
  - jouer un son lors du clic.
  - jouer un son en réponse au clic.
  - jouer/stopper une animation.
  - faire apparaître/disparaître une fenêtre donnée.
  - effectuer un branchement vers un autre écran.
  - quitter l'application.
  - afficher/effacer un texte.
  - afficher/effacer une image.

c) Le menu.

Le logiciel doit proposer au concepteur un ensemble de menus standards de différentes formes (menus déroulants, menus camembert, etc.) et doit lui permettre de concevoir ses propres menus.

- Le concepteur doit spécifier le nombre d'items du menu. Pour le guider dans sa tâche, le nombre maximum d'items peut être limité (à cinq, par exemple).

- Il doit aussi indiquer le nom de chaque item et, s'il utilise un type non-standard de menu comme le menu camembert, les icones de chaque item.

- Le logiciel doit demander la ou les actions à attacher à chaque item.
  Les actions valides sont les suivantes :
  - faire apparaître/disparaître une fenêtre donnée.
  - jouer/arrêter un son.
  - afficher/effacer un texte.
  - afficher/effacer une image.
  - activer/désactiver une option.

Les objets multimédias.

d) L'image.

Le logiciel doit pouvoir traiter tout type d'image (*bitmap* ou vectorielle) codée dans les formats les plus courants. Il pourra donc s'agir de dessins, de photographies, de graphiques, d'images de synthèse, etc. Une image pourra être créée à l'aide de l'outil offert par le logiciel ou bien pourra être importée.

- Le concepteur doit pouvoir déplacer l'image comme bon lui semble sur l'écran en cours de construction. Il doit aussi pouvoir en modifier l'échelle, l'image peut ainsi être agrandie ou réduite à souhait sans aucune retouche.

- Le concepteur doit pouvoir délimiter différentes zones de l'image. Cette délimitation pourra se faire de différentes manières (cfr. section 3.5.2.e). Les différentes zones doivent absolument être liées à l'image c'est-à-dire que tout déplacement ou changement d'échelle de l'image doit entraîner un changement correspondant des zones. Ceci afin d'éviter les fastidieuses corrections qu'un déplacement de l'image de quelques *pixels* entraîne inéluctablement.

- Une fois les zones délimitées, le concepteur doit avoir la possibilité de leur assigner un comportement particulier :

  - il peut rendre les zones sélectionnables de manière exclusive ou non. Dans ce cas, lorsque l'utilisateur pressera le bouton du moyen d'interaction utilisé lors du survol d'une zone par son pointeur, aucune action ne sera directement déclenchée. Il s'agit donc d'offrir la possibilité à l'utilisateur de choisir une ou plusieurs zones et ensuite de déclencher une action à partir d'un des objets de contrôle de l'interface.

Ex. Dans un ensemble d'images, s'il faut retrouver des images représentant des objets de la même famille (par exemple des outils), l'utilisateur de l'application va devoir sélectionner plusieurs images en séquence.

- il peut rendre les zones cliquables c'est-à-dire qu'il leur associe le comportement d'un bouton de commande transparent (voir actions associées au bouton de commande).

Ex. Supposons une image représentant une forêt composée de différents arbres. Le concepteur de l'application délimitera d'abord les arbres puis leur associera une action.
Ainsi, lorsque l'utilisateur de l'application cliquera sur un arbre, l'action associée se déclenchera (par exemple, l'apparition d'un écran affichant les caractéristiques de l'arbre)

- L'image doit pouvoir être masquée et affichée à volonté et il doit être possible de temporiser cet affichage.

Ex. Lors de notre visite au CTR, nous avons eu l'opportunité de voir fonctionner différentes applications utilisées par leurs patients. Une de celles-ci affichait une image pendant un certain laps de temps. Le patient devait alors la retrouver parmi un ensemble d'images.

e) Les objets manipulables.

Nous avons choisi de ne pas reprendre ces objets sous l'appellation « image » car nous considérons qu'ils revêtent un comportement différent des objets que nous avons décrit dans la section précédente.
Ces objets sont en fait les différents drapeaux et éléments graphiques mobiles que l'on peut trouver dans une application multimédia.

Ex. Dans l'exercice de reconstitution d'un corps humain[11], les différentes parties du corps sont les objets manipulables.

A chaque objet manipulable, l'utilisateur du logiciel doit pouvoir, s'il le désire, associer une zone valide de positionnement.

[11]Lepoutre, T. et Roquet, J., *La personne handicapée mentale et la connaissance du corps humain : un logiciel d'apprentissage*, Institut d'informatique, 1990

Ex. - Pour la reconstitution d'un puzzle, chaque pièce à sa zone valide de placement. Si l'utilisateur de l'application place un élément du puzzle à un mauvais endroit, celui-ci peut par exemple revenir à sa position initiale.

- Si l'application est un petit programme de dessin avec un certain nombre d'objets graphiques (fleur, oiseau, chien, chat, etc.), l'utilisateur de cette application pourra les placer au grès de son imagination, ces objets n'ayant pas de zone valide de placement.

- Dans le cas où le concepteur a associé une zone valide de positionnement, il doit être possible, lorsqu'une erreur est commise par l'utilisateur, de rendre visible le retour de l'objet à sa position initiale ou son placement dans sa zone valide.

f) L'animation.

Le terme animation reprend ici aussi bien les animations de *sprite* et les séquences vidéos que les simples déplacements de *sprite*.
Une animation peut avoir lieu aussi bien dans un emplacement bien déterminé de l'écran que suivant une trajectoire plus ou moins complexe dans ce même écran.

- Le logiciel doit permettre de spécifier la localisation temporelle de l'animation. La localisation temporelle permet soit de faire jouer l'animation en réponse à une action exécutée par l'utilisateur (clic sur un bouton ou sur une portion d'image, etc.), soit de faire jouer l'animation à un moment donné du temps et/ou suivant une période.

- Si l'animation est fixe, le concepteur doit pouvoir en spécifier la localisation spatiale c'est-à-dire l'endroit de l'écran où elle sera jouée.

- Si l'animation n'est pas fixe, le concepteur doit pouvoir aisément spécifier une trajectoire en indiquant par exemple les différents points de passage.

- Indépendamment du type d'animation, le concepteur doit se voir offrir la possibilité d'en spécifier la durée d'exécution et la vitesse de défilement (en images/seconde pour une séquence vidéo et en centimètres/seconde pour un déplacement de *sprite*).

g) Le son.

Sous cette appellation, nous rassemblons tout matériel auditif tel que les bruitages, la musique et la voix. Le son sert principalement à deux choses ; il permet d'abord de consolider l'ambiance créée par le matériel visuel et ensuite, il est un instrument fondamental de feed-back.

- Tout comme pour l'animation, le son peut résulter soit d'une action de l'utilisateur, soit de l'écoulement du temps. Le jeu d'un son peut aussi dépendre de l'écran dans lequel se trouve l'utilisateur. Dans le premier cas , un son pourra être attaché à un bouton de commande ou à une zone cliquable d'une image (il sera déclenché soit lors du clic, soit en résultat du clic). Il pourra aussi être généré lorsque l'utilisateur commet une erreur. Dans le deuxième cas, le concepteur peut spécifier à quel moment jouer le son et, éventuellement selon quelle période. Il peut aussi introduire un élément de hasard et rendre l'occurrence du son apériodique.

  Ex. Si le concepteur désire faire entendre des chants d'oiseau dans son application, il peut ainsi rendre le choix et l'occurrence du chant aléatoire.

  Dans le troisième cas, le concepteur peut associer un son (souvent une musique) à chaque écran de l'application.

- Le concepteur doit aussi pouvoir spécifier la durée du son et éventuellement le faire boucler (pour une musique).

- Le concepteur doit aussi avoir la possibilité de faire jouer plusieurs sons en même temps. Bien entendu, cette possibilité dépend grandement des capacités de la machine faisant tourner l'application.

h) Le texte.

Le concepteur doit être à même de placer du texte dans son application soit de manière statique (titres, messages d'information permanents, etc.) ou dynamique (messages d'aide contextuels, etc.). Le texte pourra ainsi apparaître systématiquement dans un écran ou une fenêtre donnée ou en réponse à une action de l'utilisateur.

- Le concepteur peut spécifier la localisation spatiale du texte en délimitant des zones d'édition.

- Le concepteur peut aussi spécifier la localisation temporelle du texte de la même manière que pour le son.

  Ex. Au CTR, une de leurs applications sert à entraîner la capacité de mémorisation à court terme de leurs patients. Avant de commencer l'exercice, l'application demande combien de temps la liste de mots à mémoriser doit rester visible.

- Le concepteur peut attacher un texte à un écran ou une fenêtre. Lorsque l'écran ou la fenêtre en question apparaîtra, le texte fera de même.

i) La transition.

Lors du passage d'un écran à un autre ou lors de l'apparition d'un nouvel objet graphique à l'écran, le logiciel doit permettre au concepteur de choisir entre un certain nombre de transitions prédéterminées (zoom avant ou arrière, fondu enchaîné, etc.).

4.1.3 Les moyens d'interactions.

Le moyen d'interaction principal d'une application multimédia est sans conteste la souris. Le clavier est parfois utilisé durant un bref moment pour les entrées de texte (valeurs numériques, nom et prénom, etc.). L'utilisation de la souris ou du clavier ne requiert aucune fonction particulière du logiciel auteur.
Cependant, le concepteur de l'application doit pouvoir substituer au pilotage par la souris un autre type de pilotage. Nous considérons ici le balayage des différentes zones pouvant être cliquées ou sélectionnées (zones d'une image, boutons de commande, etc.). Cette technique de balayage se base sur l'utilisation d'une sorte de *bumper* et permet à des personnes ne pouvant pas utiliser la souris pour des raisons bien souvent motrices de contourner leur handicap. Dans le logiciel auteur doit être prévue une fonction permettant la gestion du balayage et ce de manière transparente pour le concepteur.

- Lorsque le moyen d'interaction principal est la souris, le logiciel doit offrir la possibilité de changer la forme de son pointeur en fonction des zones survolées ou cliquées.

  Ex. Dans l'application « Gun's Garden », lorsque le pointeur de la souris passe au-dessus d'une zone représentant un parterre de fleurs, le curseur se

change en loupe. Dans un autre écran, le curseur prend la forme d'une main ouverte lors du survol d'un objet manipulable et d'une main fermée lorsque le bouton de la souris est enfoncé.

- Il doit aussi être possible d'associer une ou plusieurs actions, lorsqu'une zone est survolée par le pointeur du moyen d'interaction considéré.
  Ces actions sont les suivantes :
  - afficher/effacer une image.
  - afficher/effacer un texte.
  - faire apparaître/disparaître une fenêtre donnée.

Partie 3 :
Etude de cas : l'application développée en Suède.

Dans cette troisième partie, nous allons présenter les résultats de notre stage en Suède. Il s'agit en quelque sorte d'une confrontation au réel des principes mis en avant dans la première partie de ce mémoire.

Dans un premier temps, nous allons introduire de manière synthétique le cas de Gun Andersson, pour qui le logiciel a été développé, et ensuite nous présenterons l'application proprement dite, « Gun's Garden ».
L'application sera décrite d'abord du point de vue de son interface et ensuite du point de vue de son implémentation, bien que plus succinctement.

Chapitre 5. Présentation du cas de Gun Andersson.

Gun Andersson, âgée d'une cinquantaine d'année, était secrétaire académique auprès de l'école supérieure technique de Lund (Lunds Tekniska Högskola) en Suède. Elle parlait couramment plusieurs langues étrangères dont l'anglais et l'allemand.

5.1 L'atteinte cérébrale et les troubles aphasiques observés.

En 1991, lors d'un voyage d'agrément en Allemagne, elle fut terrassée par une hémorragie intracérébrale qui la laissa hémiplégique[12] et aphasique. Cela se concrétisa entre autres par une perte temporaire de l'usage de la parole et par une amnésie partielle.
A la suite de cette attaque, elle éprouva de grandes difficultés à s'exprimer de manière cohérente en suédois et perdit toute connaissance des autres langues.

Elle fut traitée à l'hôpital universitaire de Lund pour ses lésions physiques. Ensuite, elle fut prise en charge par un centre de recherche sur le handicap, le CERTEC (Centrum för Rehabiliteringsteknik) qui se chargea de lui faire recouvrer ses facultés intellectuelles.

Lors de notre première rencontre en septembre 1994, Gun semblait ne plus souffrir d'aucun handicap physique, avait recouvert l'usage de la parole mais souffrait toujours d'une perte partielle du langage. Les troubles observés du langage comprennent entre autres le manque du mot, diverses transformations aphasiques du langage oral tel que la paraphasie verbale normale et sémantique (cfr. sections 1.2.3 et 1.4). Elle était toujours incapable de parler la moindre langue étrangère et éprouvait de grandes difficultés à se rappeler le nom des choses (anomie).

[12]L'hémiplégie est une paralysie frappant une moitié du corps provoquée par des lésions des centres nerveux moteurs.

Si l'on devait faire coincider son cas avec une des catégories de l'aphasie répertoriées par les chercheurs, on pourrait qualifier l'aphasie de Gun d'*aphasie amnésique*.

5.2 Rééducation mise en place.

Plusieurs fois par semaine, elle participe à des séances de rééducation au CERTEC afin de lui redonner un vocabulaire courant qui lui permettra de se débrouiller dans le quotidien. Plusieurs logiciels sur Macintosh ont été réalisés par un centre de réhabilitation du handicap de Malmö afin de faciliter ce réapprentissage.
Nous avons eu l'occasion de voir travailler Gun avec l'un d'eux. Ce logiciel lui proposait entre autres une série de dessins représentant des adjectifs courants et elle devait les reconnaître. Elle pouvait s'aider en demandant à l'ordinateur de prononcer la première syllabe du mot. Ensuite, elle avait la possibilité de vérifier sa réponse en demandant une prononciation complète et/ou un affichage du mot à l'écran.

5.3 Les raisons du logiciel « Gun's Garden ».

Cet apprentissage semblait très bien fonctionner et Gunilla Knall, la psychologue responsable du cas de Gun au CERTEC, semblait très satisfaite des progrès réalisés grâce à cette méthode. Elle désirait donc le développement d'un autre logiciel qui, lui, serait pleinement multimédia.

Premièrement, il ne devait pas s'agir d'un programme d'apprentissage au sens strict du terme mais plutôt d'un logiciel de communication qui prendrait comme base de cette communication le jardin de Gun.
En effet, le CERTEC avait remarqué depuis un certain temps que Gun se fatiguait assez vite en utilisant les logiciels à sa disposition et ce à cause principalement du renforcement négatif causé par les erreurs qu'elle commettait (il s'agissait d'un message d'erreur affiché à l'écran).

C'est pourquoi Gunilla désirait un logiciel qui aurait pour cadre les loisirs de Gun sans fonctions d'apprentissage apparentes et qui lui permettrait de dialoguer avec son entourage sur un sujet qui lui tiendrait à coeur et qui la motiverait. En effet, grâce à cette motivation, un apprentissage implicite pouvait être envisagé. Il s'agissait donc de réaliser un logiciel comportant des fonctions d'apprentissage mais sans que Gun les reconnaissent comme tels.

5.4 Pourquoi le multimédia dans le cas de Gun?

Les logiciels antérieurs réalisés pour Gun n'utilisaient pas une représentation graphique réaliste des concepts à illustrer mais plutôt symbolique. Les images représentant les mots étaient des dessins caricaturaux en noir et blanc.
Le CERTEC voulait un logiciel procurant la représentation la plus fidèle possible de la réalité. Le logiciel devait être multimédia.

L'usage du multimédia ajoutait un certain nombre de possibilités :

- une représentation graphique fidèle permettrait à Gun de se sentir plus proche du logiciel, de ne pas ressentir l'interface comme un obstacle à l'utilisation du logiciel.

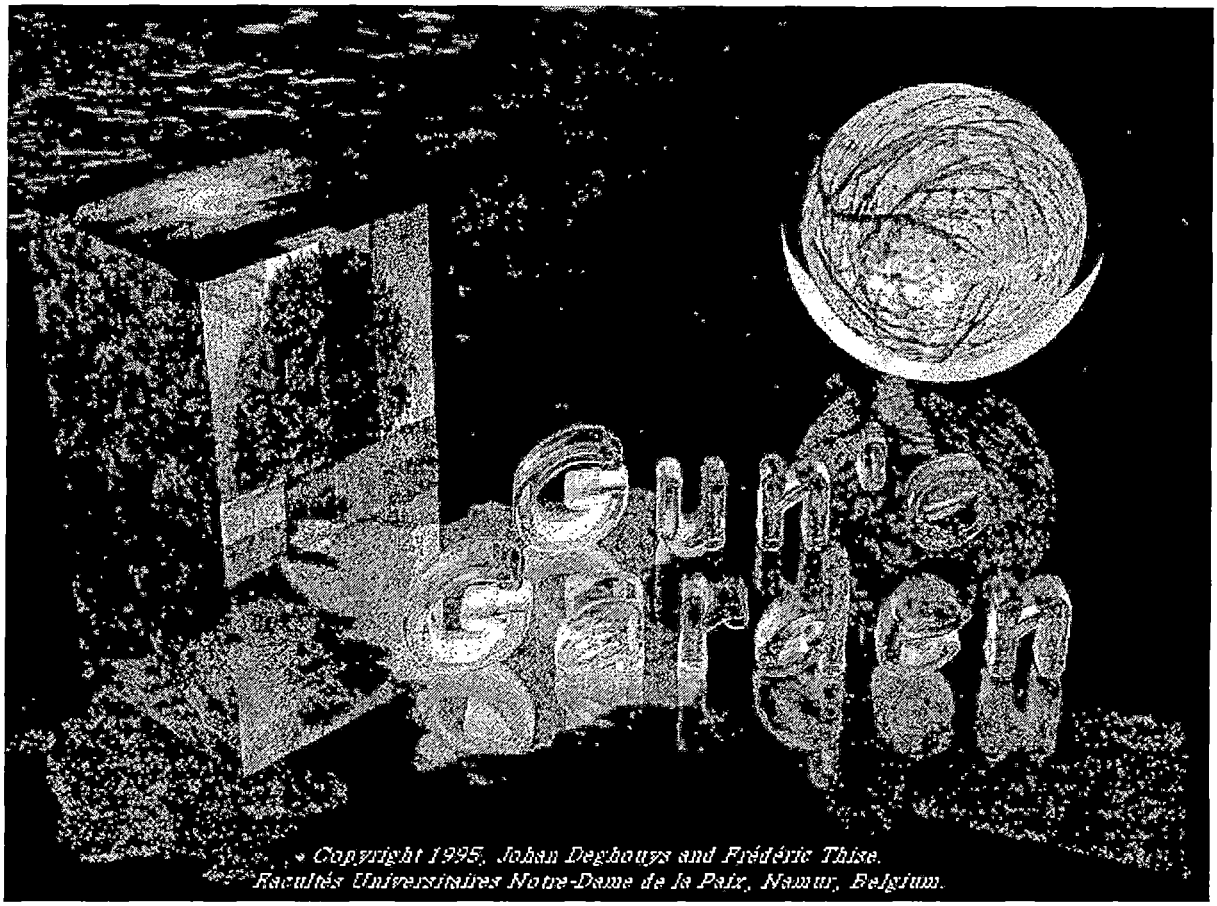
- le fait d'offrir un maximum de canaux de communication (sons, voix, images,...) permettrait de faire travailler équitablement les différents sens sans surcharger un seul.

- la possibilité de pouvoir travailler selon ses envies avec un logiciel qui se substituerait partiellement au rééducateur humain en ce qui concerne les tâches répétitives (par exemple, demander à l'ordinateur de prononcer le nom d'une chose encore et encore). C'est bien sur une caractéristique commune à tous les logiciels mais le multimédia leur donne une nouvelle grandeur.

- pendant l'été, Gun avait photographié la totalité de son jardin sous presque tous les angles. L'utilisation de ces photos, plutôt que de photos tirées de livres d'horticulture, apporte un plus évident au niveau de la motivation.

- le multimédia, en raison de sa diversité, est évidemment plus attractif, entraînant une motivation plus importante de l'utilisateur potentiel.

Chapitre 6. Présentation du logiciel « Gun's Garden ».



Le logiciel permet à l'utilisateur de communiquer avec son entourage au sujet de son jardin. Il représente en fait la base de cette communication mais ne la supporte pas fonctionnellement. L'apprentissage sera, nous l'espérons, inhérent à cette communication. Cependant, le logiciel a été prévu pour pouvoir être utilisé en solitaire, contrairement à la plupart des logiciels de réapprentissage, et est donc destiné à un cadre domestique (aucun exercice, à proprement dit, n'est proposé et aucune évaluation des résultats n'est calculée).

Il offre la possibilité de se balader dans le jardin et d'en admirer les différentes fleurs ainsi que d'obtenir quelques informations sur celles-ci. Cette ballade peut se faire classiquement de manière spatiale en cliquant sur une vue aérienne du jardin et des différents parterres ou de manière temporelle à partir de la période de floraison des différentes fleurs.
Ces « recherches » débouchent toujours sur les images des fleurs et la possibilité d'en faire afficher et prononcer le nom latin et/ou suédois ainsi qu'une brève description.
Notons que ces deux types de recherche offrent un prolongement des exercices d'orientation spatiale et temporelle généralement effectués au tout début du réapprentissage des personnes

cérébro-lésées en générale et des aphasiques en particulier. Nous reviendrons sur ce point dans une section ultérieure.

Cette visite se fait au moyen d'une interface qui laisse un maximum de place aux différents médias en se restreignant à la périphérie de l'écran.

Le logiciel « Gun's Garden » a été développé sur des Mac Quadra 840AV au moyen du logiciel auteur MacroMedia Director dans sa quatrième version. Le développement a eu lieu au département d'informatique de l'université de Lund en Suède sous la coordination de Jörn Nilsson, professeur de HCI (Human Computer Interaction) dans ce département.

6.1. Approche de conception.

Avant toute chose, nous nous sommes renseignés sur les limitations qu'engendrait la maladie de Gun au niveau de sa capacité à utiliser un logiciel multimédia. La rééducation de son aphasie étant dans une phase déja bien avancée, les troubles persistants étaient principalement d'ordre mnésique. De ce fait, nous avons pu concevoir l'interface comme si elle était destinée à une personne « normale ».

Malgré cela, nous avons été confrontés à un cas où la méthodologie classique de conception symbolisée par le modèle de la cascade[13] (figure 6.1) était inapplicable telle quelle.
Il était en effet impossible de déterminer lors des premiers contacts l'ensemble des fonctionnalités que devait offrir le logiciel. Nous n'étions pas familiarisés avec l'outil de conception et le paradigme de programmation qu'il proposait. En outre, la maladie de Gun l'empêchait de définir ses propres besoins et desiderata (rappelons qu'elle ne parlait plus que le suédois, langue que nous ne maîtrisons qu'imparfaitement...).
Nous avons donc plutôt eu affaire à la psychologue responsable de Gun au CERTEC pour savoir vers quoi orienter notre conception. En outre, le futur logiciel devait se placer dans le cadre général du plan de réapprentissage mis en place.

[13]Dix et al, *Human-Computer Interaction*, Prentice Hall, 1993, p. 148-.

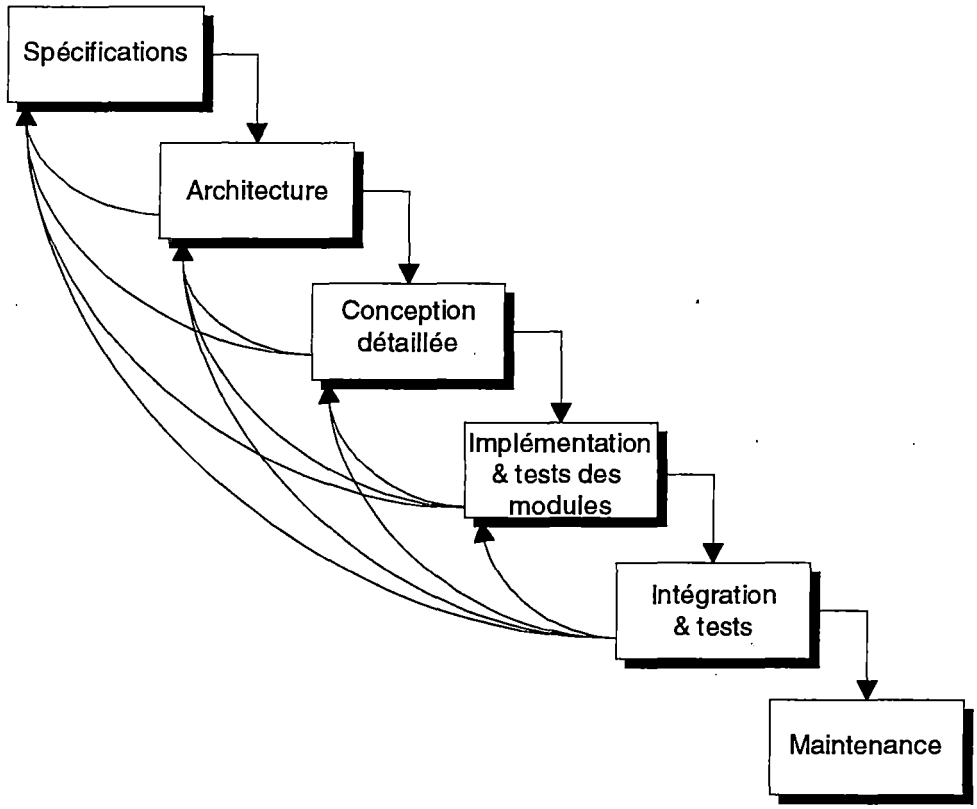


Figure 6.1 Le modèle de la cascade.

La démarche la plus appropriée à ce tâtonnement forcé nous a semblé être l'approche par prototypage[14].
Dix et al. ont identifié trois principales approches au prototypage :

- **prototype « jetable »** : un prototype est concu et testé. L'expertise gagnée de cet exercice est utilisée pour construire le produit final mais le prototype est abandonné.

- **prototype incrémental** : le produit final est construit à partir de composantes séparées et indépendantes. Un prototype est concu pour chaque composante.

- **prototype évolutif** : Cette méthode est similaire à la première mais ici le prototype (figure 6.2) n'est pas jeté et sert de base à la prochaine itération. Dans ce cas, le système est vu comme évoluant d'un nombre réduit de versions initiales vers sa réalisation finale.

[14]Dix et al, *Human-Computer Interaction*, Prentice Hall, 1993, p. 173-.

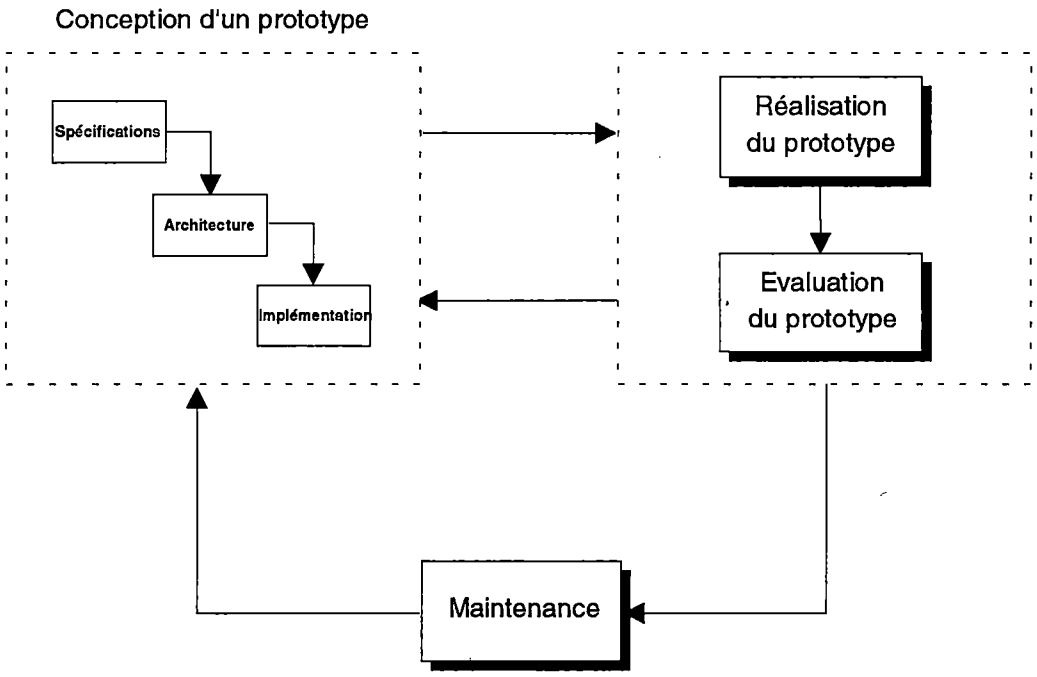


Figure 6.2 Un modèle de prototypage incrémental actualisé.

Pour la conception du logiciel, nous avons opté pour la troisième approche et ceci pour plusieurs raisons.
Premièrement, nous ne pouvions pas nous permettre de gaspiller du temps en adoptant la première approche car nous étions dans l'impossibilité d'évaluer la durée nécessaire à la réalisation du logiciel. D'une part il s'agissait de notre premier logiciel et d'autre part, comme nous l'avons déjà dit, nous n'étions pas familiarisés avec le logiciel auteur Director 4.0.
Deuxièmement, aucun cahier des charges ni aucunes contraintes particulières n'ayant été imposée, il nous était impossible de voir le logiciel comme une collection de composantes autonomes et donc d'appliquer la deuxième approche.

C'est pourquoi la conception par prototypage évolutif nous a semblé la plus appropriée. Un prototype, ne contenant qu'un nombre limité de fonctionnalités, a donc été conçu au plus vite afin de fournir une base à laquelle se référer lors des discussions ultérieures. Cela a aussi permis à Gunilla d'appréhender de meilleure manière le potentiel de l'outil multimédia.

Le logiciel auteur employé étant particulièrement agréable à utiliser lorsqu'il s'agit de concevoir des programmes où l'interaction est relativement simple, le prototype a été réalisé en l'espace de quelques jours. Il permettait à partir d'une carte sommaire du jardin, de choisir entre deux parterres de fleurs; ensuite, d'accéder à un écran contenant une carte fictive (nous n'avions pas encore eu le loisir d'effectuer un relevé de la localisation des différentes fleurs du jardin) du parterre à partir de laquelle il était possible de faire afficher les photos des fleurs du parterre.

Cette technique de prototypage est notamment identifiée par Dix et al. comme étant un excellent moyen de s'assurer, dans les premières phases de la conception d'un système, de l'adéquation des aspects interactifs.
Le prototype a ensuite été critiqué par Gunilla Knall et Gun a eu la possibilité de le tester. Enfin, de nouveaux besoins ont été identifiés, ce qui nous a permis d'apporter différents changements au logiciel. La totalité du jardin a été modélisée de manière fidèle et une recherche sur base de la période de floraison a été spécifée. Tout le reste du travail a consisté en la réalisation d'une interface gérant l'interactivité de manière optimale.

En ce qui concerne la conception proprement dite, nous avons vu le logiciel avant tout comme une interface qui se devait d'être la plus souple possible et qui devait être capable de s'adapter aux besoins de Gun tout en lui fournissant le moyen de se repérer à tout moment au sein du logiciel. Cette interface se devait d'être la plus ergonomique possible et c'est cette considération qui nous a principalement guidé lors du développement du logiciel. Les fonctionnalités demandées étant peu nombreuses, nous avons eu tout le loisir de peaufiner l'interface et d'utiliser tous les moyens nécessaires pour en faciliter l'accès.

Notre architecture de logiciel est avant toute chose une architecture de l'interface, une architecture des écrans.

Il y a donc un démarquage assez net avec le modèle de conception de la cascade en raison de cette identification progressive des besoins. Chaque nouveau besoin se traduisant par une mise à jour du logiciel d'où une conception itérative par raffinements successifs. C'est en effet le seul moyen d'être sûr que l'interface sera bien adaptée à l'utilisateur, qui plus est unique, dans notre cas.
Il est un fait évident que s'il devait un jour être utilisé par d'autres personnes que Gun Andersson, son interface devrait être « généralisée ». Dans ce cas particulier, il ne faut pas perdre de vue que les choix de conception du dialogue et de l'interface ont été guidés non pas exclusivement par un quelconque code ergonomique mais directement par l'utilisateur et son représentant. Il est donc possible que certains aspects de l'interface entrent en conflit avec l'un ou l'autre principe d'ergonomie de logiciel communément acceptés bien qu'à notre sens elle soit relativement conforme.

6.2. Architecture du logiciel.

6.2.1. Architecture des écrans.

Voici l'organisation des différents écrans au sein du logiciel.
Les écrans *Recherche spatiale* et *Recherche temporelle* sont uniques tandis que les autres écrans sont dépendants du parterre du jardin ou de la période choisie.
Nous avons opté pour une présentation de l'architecture des écrans plutôt qu'une autre parce que cela découle directement de la manière dont MacroMedia Director conçoit une interface.
Pour de plus amples détails sur chaque écran, il suffit de se reporter sur les figures 6.4 à 6.10 dans les pages suivantes.

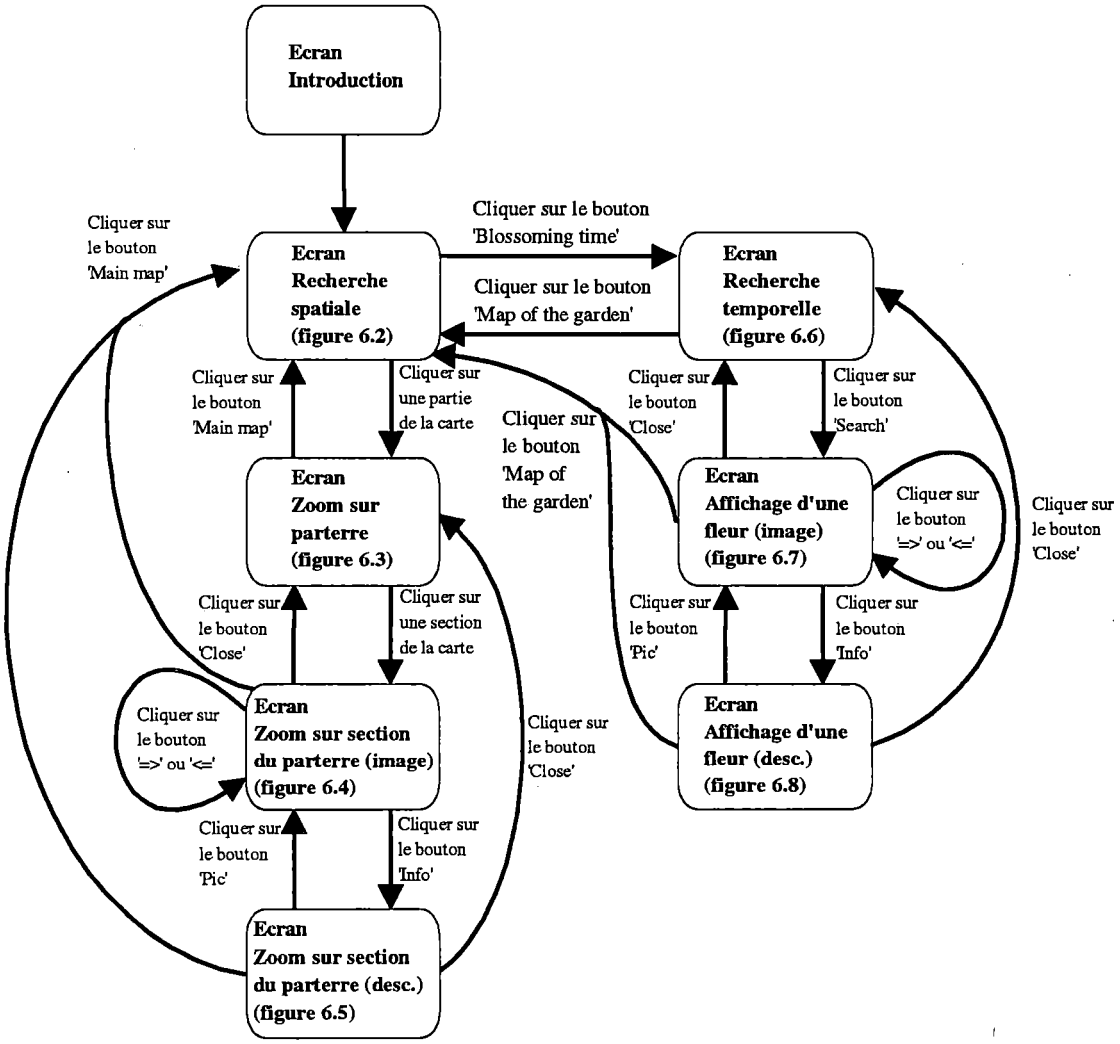


Figure 6.3. L'architecture des écrans.

6.2.2. Ecran de la recherche spatiale (écran de départ).

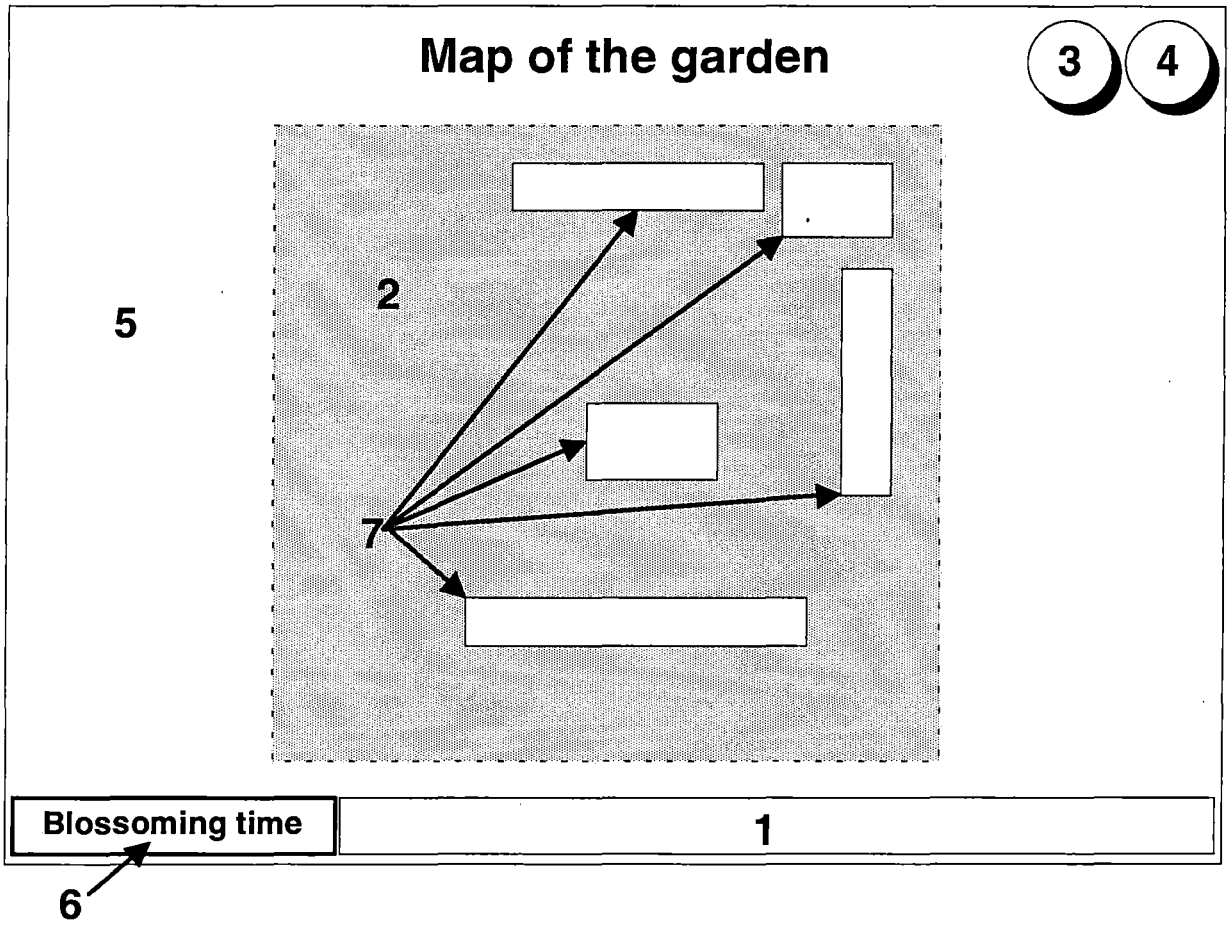


Figure 6.4. L'écran de recherche spatiale.

Légende.

1. Barre d'aide contextuelle.
2. Carte du jardin.
3. Icone cliquable d'indication de fonctionnement de la musique.
4. Icone cliquable d'indication de fonctionnement des bruitages.
5. Image de fond de l'écran représentant une vue satellite de la Scandinavie.
6. Bouton de bascule vers l'écran de recherche temporelle (figure 6.8).
7. Zones cliquables de la carte.

6.2.3. Ecran de zoom niveau 1 sur un parterre.

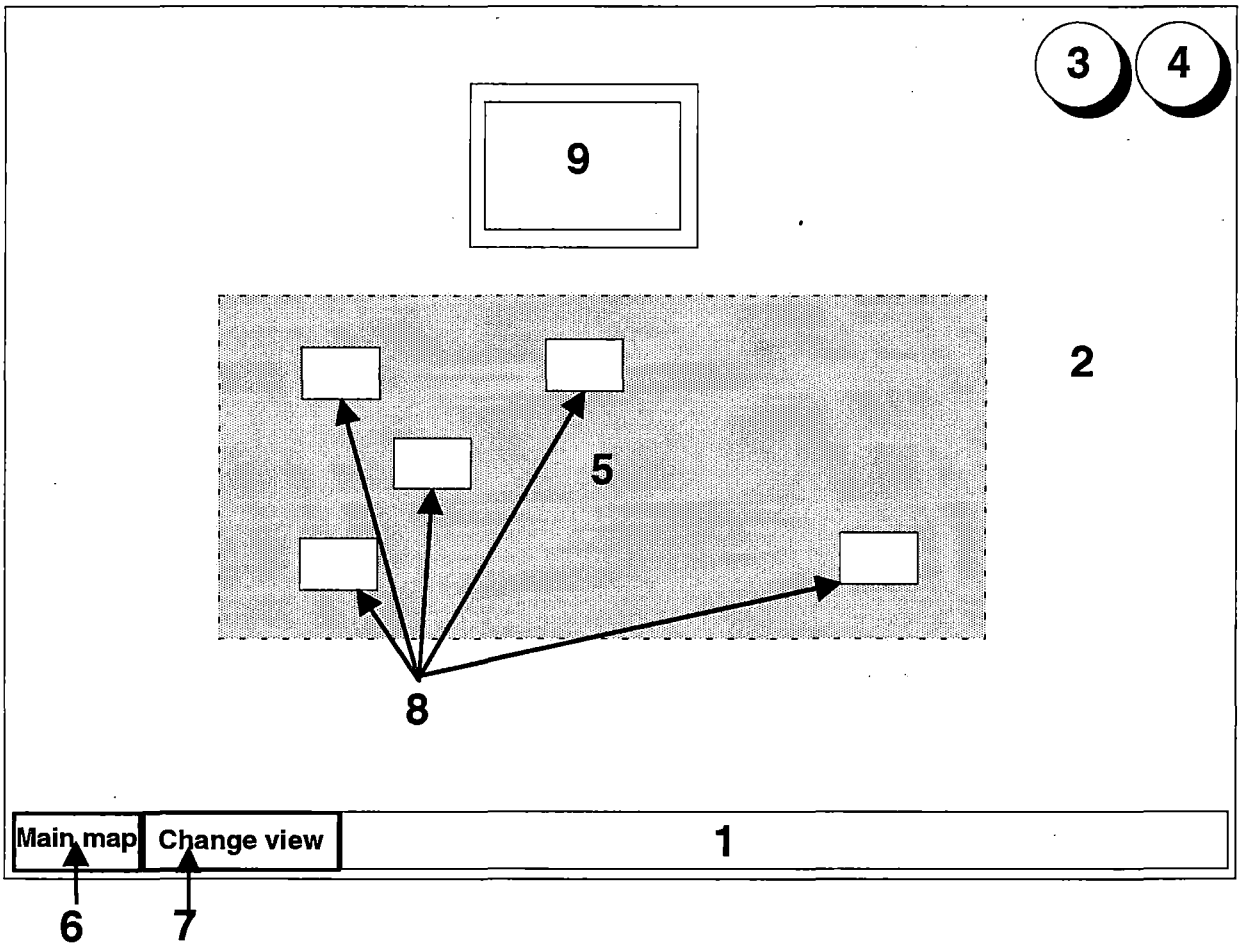


Figure 6.5. L'écran de zoom sur un parterre.

Légende.

1. Barre d'aide contextuelle.
2. Image de fond de l'écran représentant une vue grand angle du parterre.
3. Icone cliquable d'indication de fonctionnement de la musique.
4. Icone cliquable d'indication de fonctionnement des bruitages.
5. Carte du parterre.
6. Bouton de retour vers l'écran de recherche géographique (figure 6.4).
7. Bouton de changement de l'image de fond de l'écran.
8. Zones cliquables de la carte.
9. Fenêtre de prévisualisation (visible si le curseur survole les zones 8, invisible sinon).

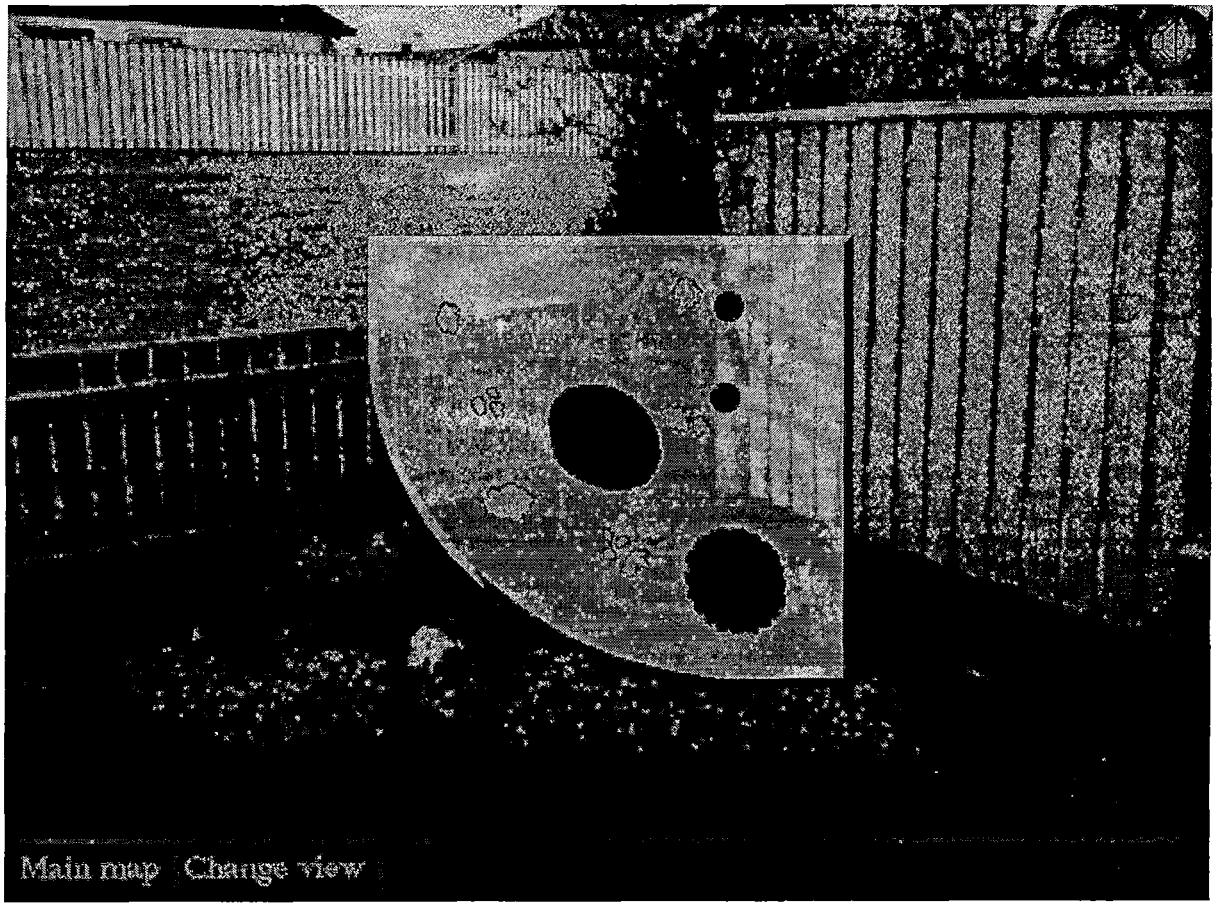


Figure 6.5 bis. Capture d'un écran de type zoom sur parterre.

6.2.4. Ecran de zoom niveau 2 sur l'image d'une section du parterre.

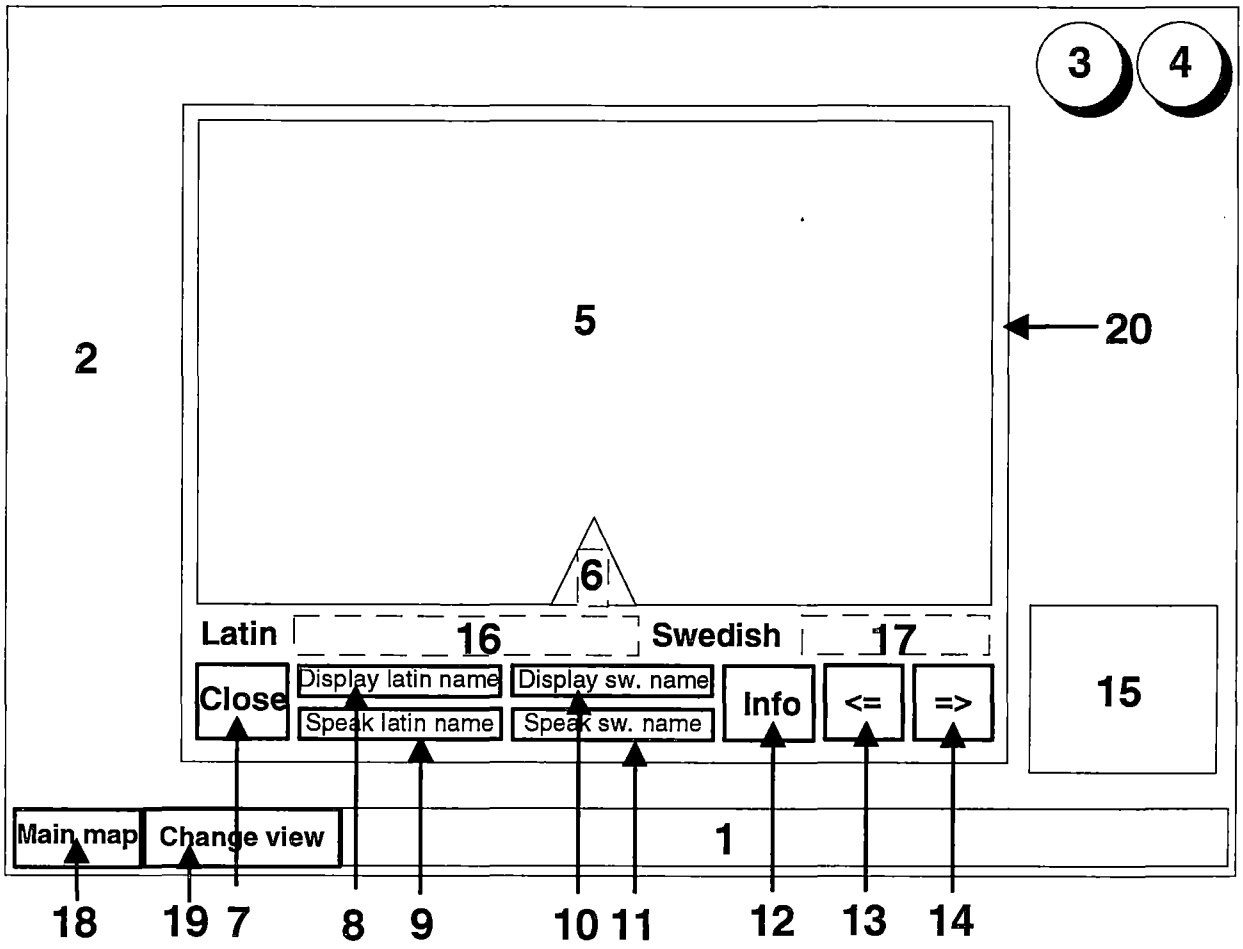


Figure 6.6. L'écran de zoom sur une section du parterre.

Légende.

1. Barre d'aide contextuelle.
2. Image de fond de l'écran représentant une vue grand angle du parterre.
3. Icone cliquable d'indication de fonctionnement de la musique.
4. Icone cliquable d'indication de fonctionnement des bruitages.
5. Image représentant un zoom sur une section du parterre.
6. Indicateur du nombre de fleurs présentes et cliquables de la section du parterre affichée.
7. Bouton de fermeture de la fenêtre d'affichage.
8. Bouton permettant d'afficher dans la cartouche 16 le nom latin de la fleur sélectionnée. Lorsque ce bouton est pressé, il se transforme en un bouton permettant de cacher le nom latin.
9. Bouton permettant l'audition d'une voix digitalisée prononçant le nom latin de la fleur sélectionnée.

10.Bouton permettant d'afficher dans la cartouche 17 le nom suédois de la fleur sélectionnée. Lorsque ce bouton est pressé, il se transforme en un bouton permettant de cacher le nom suédois.
11.Bouton permettant l'audition d'une voix digitalisée prononçant le nom suédois de la fleur sélectionnée.
12.Bouton permettant de remplacer l'image par une description de la fleur sélectionnée (figure 6.7).
13.Bouton permettant d'obtenir l'image de la section précédente du parterre.
14.Bouton permettant d'obtenir l'image de la section suivante du parterre.
15.Miniature de la carte du parterre sur laquelle un ou plusieurs pointeurs indiquent la ou les positions occupées par la fleur sélectionnée.
16.Cartouche dans laquelle sera affiché le nom latin de la fleur sélectionnée.
17.Cartouche dans laquelle sera affiché le nom suédois de la fleur sélectionnée.
18.Bouton de retour vers l'écran de recherche spatiale (figure 6.4).
19.Bouton de changement de l'image de fond de l'écran.
20.Fenêtre d'affichage de l'image.

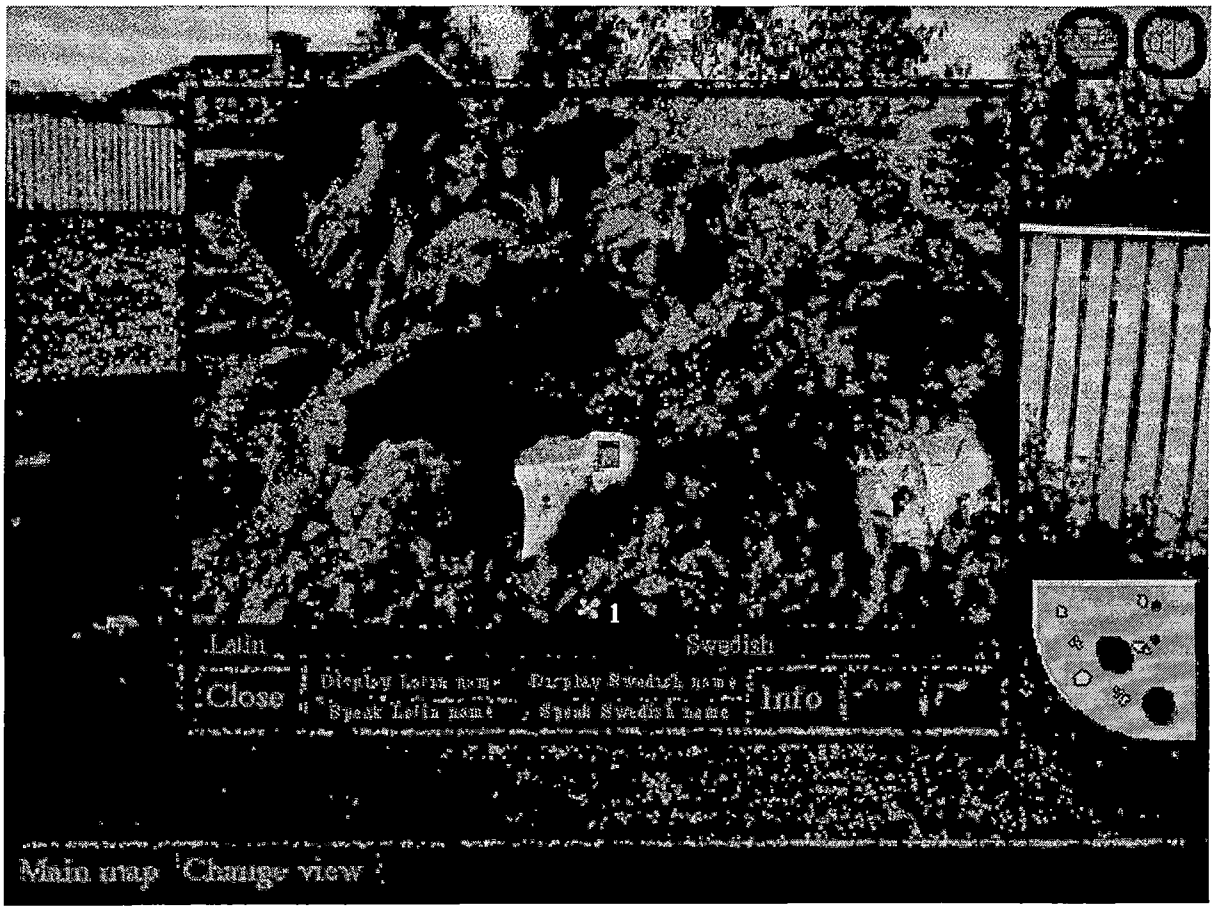


Figure 6.6 bis . Capture d'un écran de type zoom sur section d'un parterre.

6.2.5. Ecran de zoom niveau 2 sur la description d'une fleur du parterre.

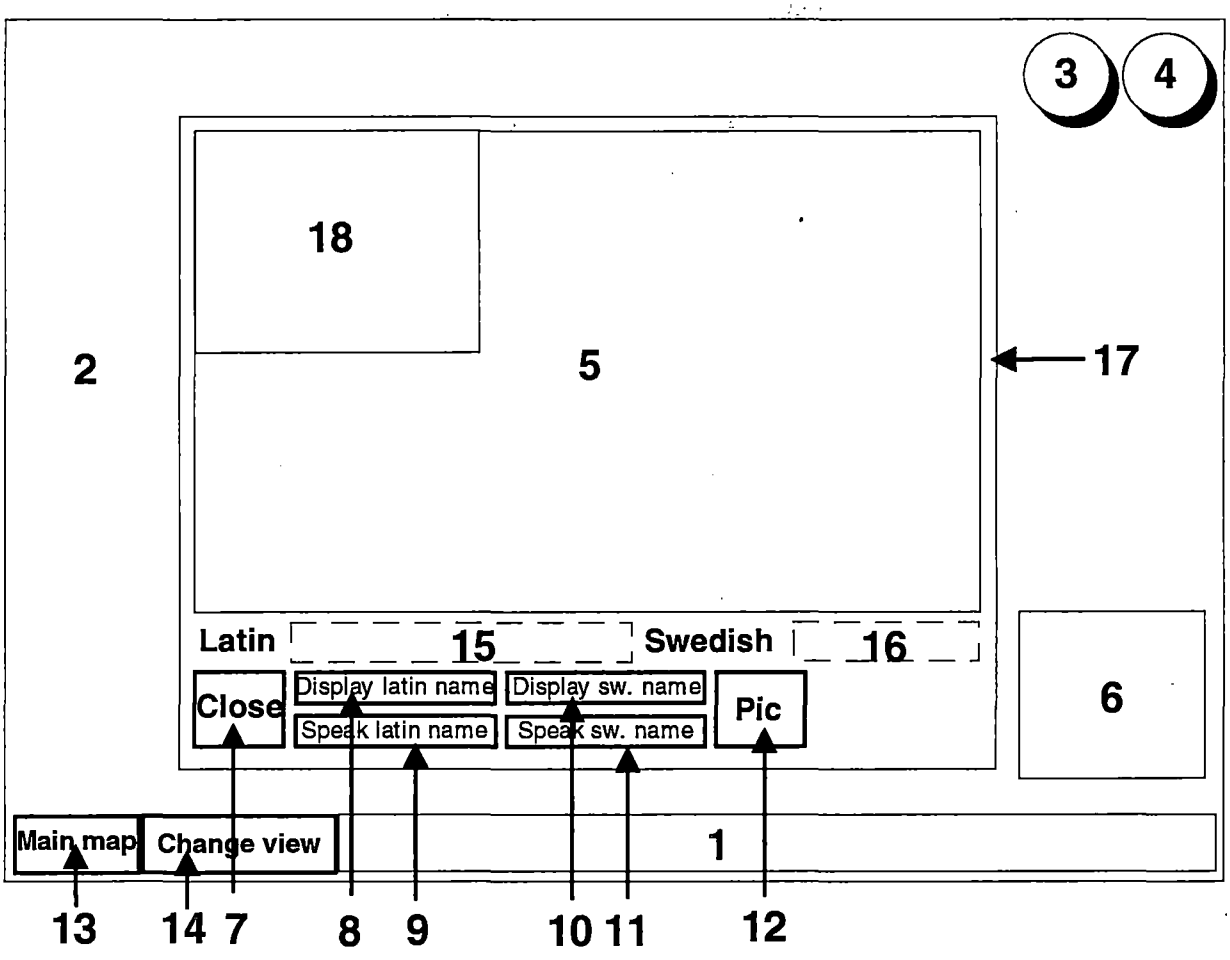


Figure 6.7. L'écran de description d'une fleur du parterre.

Légende.

1. Barre d'aide contextuelle.
2. Image de fond de l'écran représentant une vue grand angle du parterre.
3. Icone cliquable d'indication de fonctionnement de la musique.
4. Icone cliquable d'indication de fonctionnement des bruitages.
5. Description de la fleur sélectionnée.
6. Miniature de la carte du parterre sur laquelle un ou plusieurs pointeurs indiquent la ou les positions occupées par la fleur sélectionnée.
7. Bouton de fermeture de la fenêtre d'affichage.
8. Bouton permettant d'afficher dans la cartouche 15 le nom latin de la fleur sélectionnée. Lorsque ce bouton est pressé, il se transforme en un bouton permettant de cacher le nom latin.

9. Bouton permettant l'audition d'une voix digitalisée prononçant le nom latin de la fleur sélectionnée.
10. Bouton permettant d'afficher dans la cartouche 16 le nom suédois de la fleur sélectionnée. Lorsque ce bouton est pressé, il se transforme en un bouton permettant de cacher le nom suédois.
11. Bouton permettant l'audition d'une voix digitalisée prononçant le nom suédois de la fleur sélectionnée.
12. Bouton permettant de remplacer la description par une image de la fleur sélectionnée (figure 6.6).
13. Bouton de retour vers l'écran de recherche spatiale (figure 6.4).
14. Bouton de changement de l'image de fond de l'écran.
15. Cartouche dans laquelle sera affiché le nom latin de la fleur sélectionnée.
16. Cartouche dans laquelle sera affiché le nom suédois de la fleur sélectionnée.
17. Fenêtre d'affichage de la description.
18. Miniature de l'image affichée dans la fenêtre de la figure 6.6.

6.2.6. Ecran de la recherche temporelle par périodes de floraison.

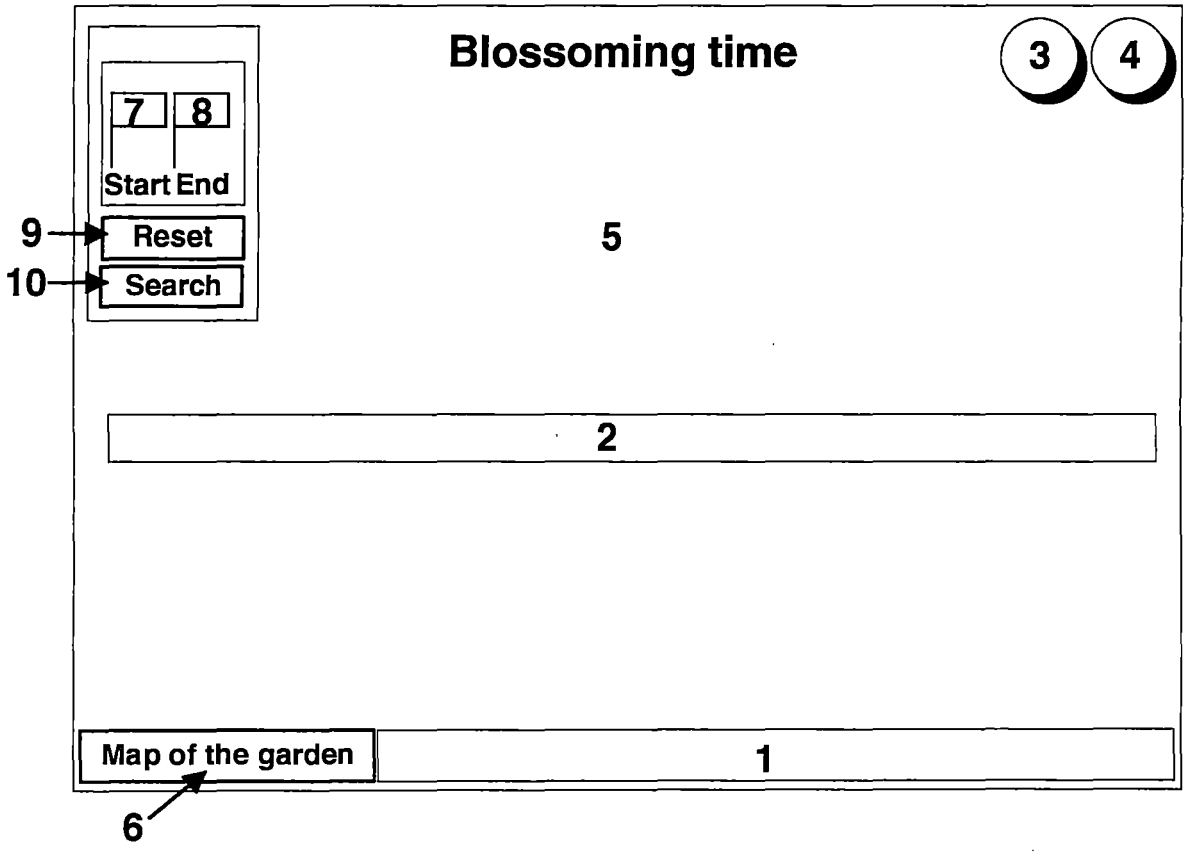


Figure 6.8. L'écran de recherche temporelle.

Légende.

1. Barre d'aide contextuelle.
2. Echelle de temps graduée de janvier à décembre sur laquelle l'utilisateur peut définir des périodes au moyen de drapeaux « Start » et « End ».
3. Icone cliquable d'indication de fonctionnement de la musique.
4. Icone cliquable d'indication de fonctionnement des bruitages.
5. Image de fond de l'écran représentant une recomposition d'un arbre à partir de quatre « quarts » photographiés lors de chaque saison.
6. Bouton de bascule vers l'écran de recherche spatiale (figure 6.4).
7. Drapeaux déplacables d'indication de début de période.
8. Drapeaux déplacables d'indication de fin de période.
9. Bouton de reinitialisation de l'échelle temporelle.
10. Bouton de lancement de la recherche sur base des périodes délimitées par les drapeaux.

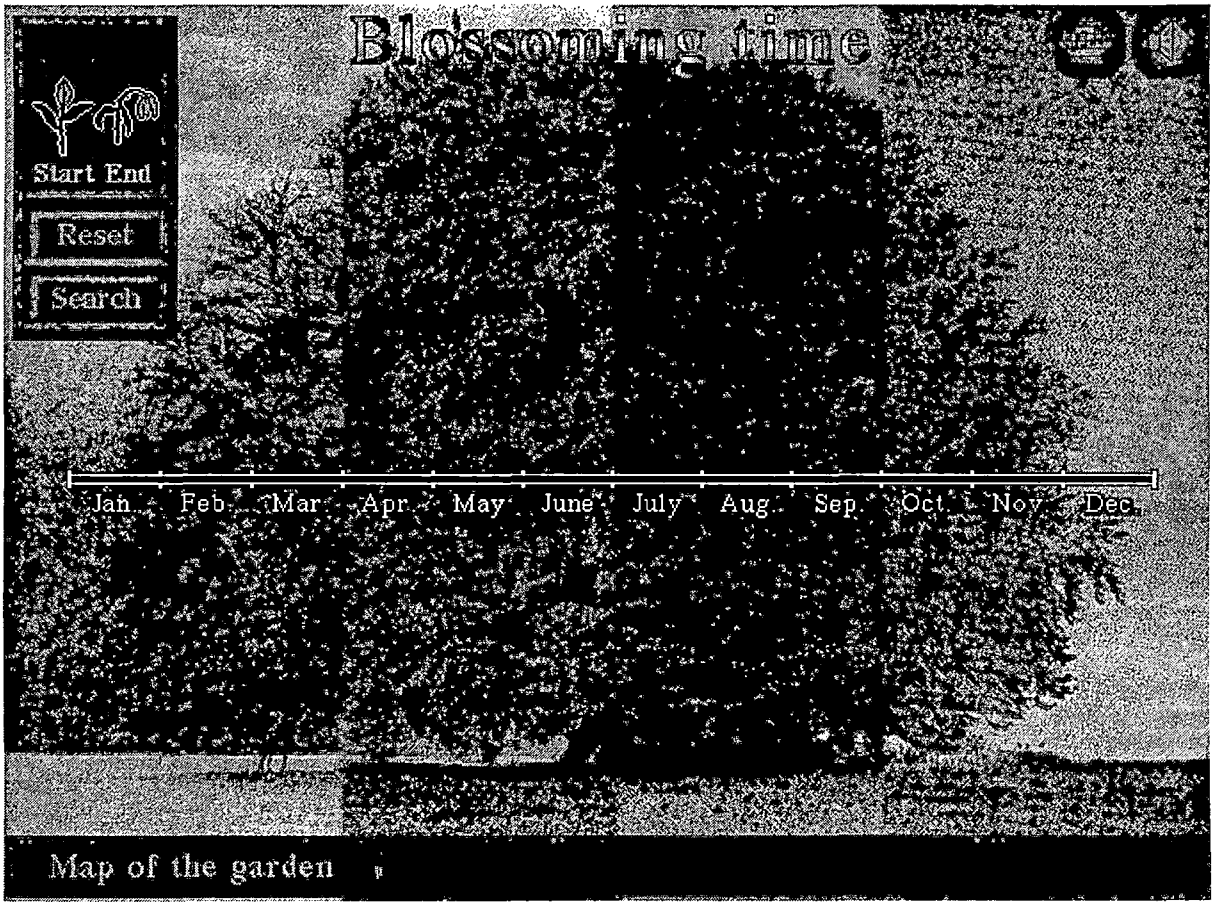


Figure 6.8 bis. Capture de l'écran de recherche temporelle.

6.2.7. Ecran de zoom sur l'image d'une fleur du jardin.

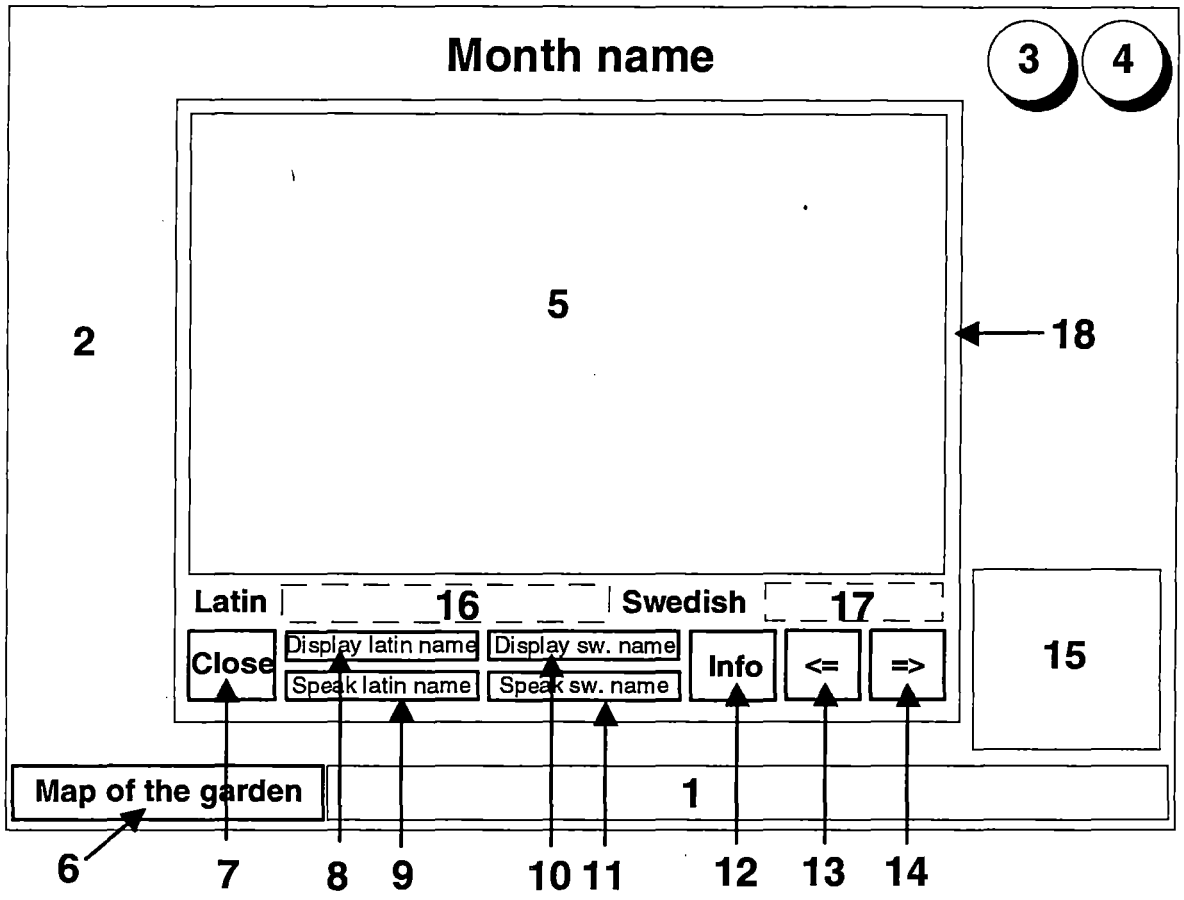


Figure 6.9. L'écran de zoom sur l'image d'une fleur du jardin.

Légende.

1. Barre d'aide contextuelle.
2. Image de fond de l'écran représentant un arbre photographié lors de la saison considérée.
3. Icone cliquable d'indication de fonctionnement de la musique.
4. Icone cliquable d'indication de fonctionnement des bruitages.
5. Image représentant un zoom sur une section du parterre.
6. Bouton de bascule vers l'écran de recherche spatiale (figure 6.4).
7. Bouton de fermeture de la fenêtre d'affichage.
8. Bouton permettant d'afficher dans la cartouche 16 le nom latin de la fleur affichée. Lorsque ce bouton est pressé, il se transforme en un bouton permettant de cacher le nom latin.
9. Bouton permettant l'audition d'une voix digitalisée prononçant le nom latin de la fleur affichée.

10. Bouton permettant d'afficher dans la cartouche 17 le nom suédois de la fleur affichée. Lorsque ce bouton est pressé, il se transforme en un bouton permettant de cacher le nom suédois.
11. Bouton permettant l'audition d'une voix digitalisée prononçant le nom suédois de la fleur affichée.
12. Bouton permettant de remplacer l'image par une description de la fleur affichée (figure 6.10).
13. Bouton permettant d'obtenir l'image de la fleur précédente florissant pendant le mois courant.
14. Bouton permettant d'obtenir l'image de la fleur suivante florissant pendant le mois courant.
15. Miniature de la carte du jardin sur laquelle un ou plusieurs pointeurs indiquent la ou les positions occupées par la fleur affichée.
16. Cartouche dans laquelle sera affiché le nom latin de la fleur affichée.
17. Cartouche dans laquelle sera affiché le nom suédois de la fleur affichée.
18. Fenêtre d'affichage de l'image.

6.2.8. Ecran de zoom sur la description d'une fleur du jardin.

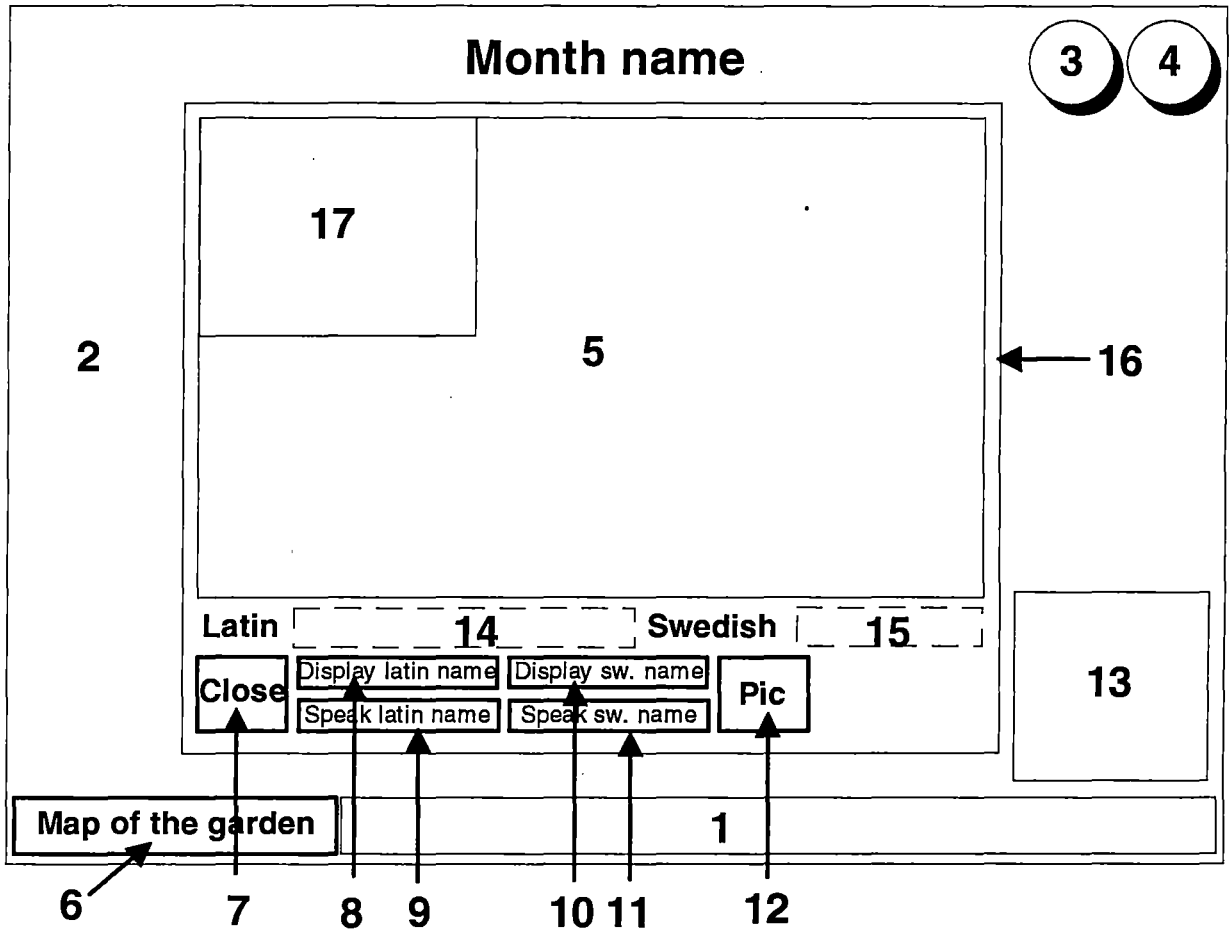


Figure 6.10. L'écran de description d'une fleur du jardin.

Légende.

1. Barre d'aide contextuelle.
2. Image de fond de l'écran représentant un arbre photographié lors de la saison considérée.
3. Icone cliquable d'indication de fonctionnement de la musique.
4. Icone cliquable d'indication de fonctionnement des bruitages.
5. Description de la fleur.
6. Bouton de bascule vers l'écran de recherche spatiale (figure 6.4).
7. Bouton de fermeture de la fenêtre d'affichage.
8. Bouton permettant d'afficher dans la cartouche 16 le nom latin de la fleur affichée. Lorsque ce bouton est pressé, il se transforme en un bouton permettant de cacher le nom latin.
9. Bouton permettant l'audition d'une voix digitalisée prononçant le nom latin de la fleur affichée.

10. Bouton permettant d'afficher dans la cartouche 17 le nom suédois de la fleur affichée. Lorsque ce bouton est pressé, il se transforme en un bouton permettant de cacher le nom suédois.
11. Bouton permettant l'audition d'une voix digitalisée prononçant le nom suédois de la fleur affichée.
12. Bouton permettant de remplacer la description par une image de la fleur affichée (figure 6.9).
13. Miniature de la carte du jardin sur laquelle un ou plusieurs pointeurs indiquent la ou les positions occupées par la fleur affichée.
14. Cartouche dans laquelle sera affiché le nom latin de la fleur affichée.
15. Cartouche dans laquelle sera affiché le nom suédois de la fleur affichée.
16. Fenêtre d'affichage de la description.
17. Miniature de l'image affichée dans la fenêtre de la figure 6.9.

6.3. Justification ergonomique de l'interface.

Le dialogue mis en place par le logiciel est basé sur la métaphore du mini-monde[15] (Model-world metaphor) implémentée grâce à la manipulation directe et l'interaction iconique.

En effet, l'interface est elle-même le monde dans lequel l'utilisateur peut interagir. En l'occurrence, le monde est ici réduit au jardin de Gunr. Ce jardin est explicitement représenté à l'aide de nombreuses photographies. L'impression de se trouver réellement dans le jardin est encore renforcée par un contexte sonore composé de différents chants d'oiseaux et d'une musique qui, quoique non présente dans la réalité, apporte une plus grande quiétude d'esprit par le thème choisi (*Les quatre saisons*, thème du printemps, de Vivaldi). De plus, l'utilisateur peut se balader dans le jardin en zoomant sur différentes parties de celui-ci.

La métaphore employée ici est donc celle du jardin en lui-même et les objets manipulables ont été construits en conséquence. Cependant, cette dernière n'est pas applicable telle quelle dans tout le programme. En fait, elle ne s'applique que pour la partie recherche spatiale. En ce qui concerne la partie recherche temporelle, la métaphore employée est moins évidente. On pourrait parler de la métaphore du panneau de contrôle à partir duquel on choisit une période de temps et on peut lancer la recherche. Dans cet écran, l'utilisateur agit toujours directement sur le logiciel mais le lien avec la réalité est nettement plus relâché. Cependant le contact avec la nature est toujours représenté explicitement notamment par le biais de l'image en arrière-plan

[15]Hutchins, E., Hollan, J. et Norman, D., *Direct Manipulation Interfaces*, dans Norman, D. et Draper, S., *User Centered System Design*, Lawrence Erlbaum Associates, 1986, p. 87-124.

représentant la recomposition d'un arbre à partir de « quarts » photographiés au cours des quatre saisons de l'année.

L'interface s'articule donc autour de métaphores composites[16]. L'une venant à l'aide de l'autre lorsque la dernière devient inapplicable. La connaissance ou reconnaissance de ces métaphores par l'utilisateur représente en quelque sorte sa compréhension de base des mécanismes de fonctionnement du logiciel.

rem : lorsque dans la suite de cette section nous parlerons d'apprentissage, nous ferons référence à l'apprentissage de l'interface et non au réapprentissage de l'aphasie mis en place par le logiciel.

6.3.1. Description des moyens d'interaction[17].

Toute l'interaction permise par le programme, en ce qui concerne la communication dans le sens utilisateur-ordinateur, s'articule autour de la seule souris du Macintosh (référencée comme moyen d'interaction d'affichage indirect). Cela veut dire que toute action de la part de l'utilisateur passe par le bouton de la souris.

Le clavier n'est ici en aucun cas utilisé et cela pour plusieurs raisons.

Premièrement, le logiciel ne requiert à aucun moment une saisie de texte de la part de l'utilisateur.
Deuxièmement, l'utilisation du clavier peut s'avérer relativement compliquée, obligeant souvent l'utilisateur novice à détourner les yeux de l'écran afin de trouver la touche appropriée.
Ces petites distractions sont autant d'écueils à la continuité du cheminement mental de l'utilisateur dans la compréhension de l'interface.
Cependant, le clavier permet l'utilisation de « raccourcis » ce qui nous amène à la troisième objection.
Les actions permises dépendent souvent du contexte c'est-à-dire de l'écran particulier dans lequel l'utilisateur se trouve. En effet, il se peut que dans l'un ou l'autre écran certains boutons soient désactivés.
De plus, nous ne sommes pas ici dans une optique de maximisation du rendement chère aux logiciels de gestion mais dans une optique de dialogue/apprentissage.

---

[16]Caroll, J., Mack, R. et Kellogg, W., *Interface Metaphor and User Interface Design*, dans Helander, M., *Handbook of Human-Computer Interaction*, Elsevier Science Publishers, 1988, p. 67-84.
[17]Sacré, B., Provot-Sacré, I. et Vanderdonckt, J., *Une description orientée objet des objets interactifs abstraits utilisés en Interface Homme-Machine,* Institut d'informatique, 1992.

Finalement, le nombre d'actions à entreprendre et la complexité réduite du logiciel ne justifie pas l'utilisation de ces « raccourcis ».

6.3.2. Description des objets interactifs[18].

a) Objets d'action.

Le menu camembert.

Un seul menu est accessible dans l'ensemble du logiciel mais il peut être accédé à tout moment. Il s'agit d'un menu en forme de camembert permettant quatre actions générales sur le système. Trois des options du menu sont des options de configuration, l'utilisateur peut ainsi activer ou désactiver le son, la musique et/ou l'aide.
La dernière option permet à l'utilisateur de quitter le logiciel.

Ces quatre options sont représentées de manière graphique et textuelle. Chacune des options représente un quart du camembert avec au centre une cinquième pseudo-option permettant de fermer ce menu.
Les représentations graphiques ont été conçues afin d'être le plus explicite possible tout en faisant si possible référence à la métaphore du jardin.

Ces représentations sont les suivantes :

- pour l'option *Music* : une partition pleine ou vide, signifiant respectivement que le choix de cette option va activer ou désactiver la musique.

- pour l'option *Sound* : un haut-parleur émettant ou non des ondes sonores signifiant respectivement que le choix de cette option va activer ou désactiver les bruitages.

- pour l'option *Help* : une vue d'un arbrisseau avec ou sans tuteur signifiant que le choix de cette option va activer ou désactiver l'aide contextuelle.

[18]Sacré, B., Provot-Sacré, I. et Vanderdonckt, J., *Une description orientée objet des objets interactifs abstraits utilisés en Interface Homme-Machine*, Institut d'informatique, 1992.

- pour l'option *Quit* : une vue d'une maison avec une flèche pointant sur la porte d'entrée signifiant que le choix de cette option va permettre de quitter le jardin, donc le logiciel.

Pour y accéder, il suffit, lorsque le curseur de la souris prend une forme bien définie (une miniature du menu camembert) de presser le bouton de celle-ci et de le garder appuyé.
A partir de ce moment, le menu fonctionne comme un menu déroulant classique, les options survolées passant au mode « vidéo inverse ».
Si par mégarde l'utilisateur relâche le bouton sans avoir déplacé le curseur, rien n'est déclenché, le menu est simplement fermé car il apparaît toujours centré autour du curseur. Ceci afin de conserver d'une part la cohérence spatiale du curseur et de placer l'option *Cancel* sous celui-ci (cfr. figure 6.11).

Figure 6.11. Le menu camembert.

L'activation ou la désactivation d'une des trois options de configuration altère la représentation graphique du quart correspondant et ce de manière à représenter l'état des options courantes (cfr. figure 6.13).

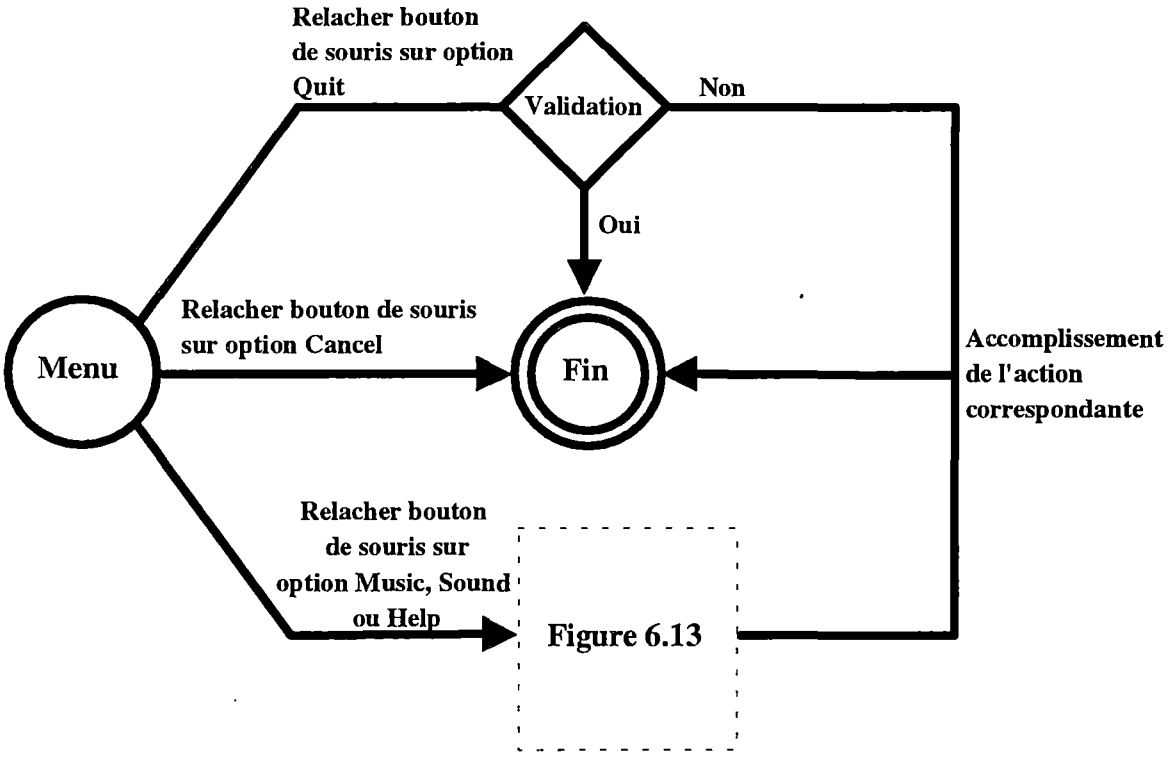


Figure 6.12. Schéma du dialogue proposé par le menu camembert.

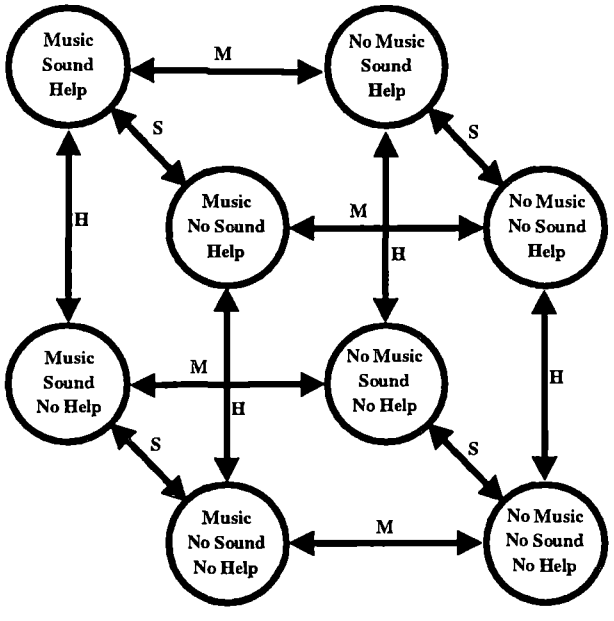


Figure 6.13. Schéma des transitions valides entre les états possibles des options du menu.

b) Objets de défilement.

La flèche de défilement.

Ces flèches sont utilisées afin d'implémenter un *browser* accessible dans les fenêtres d'affichage d'une image/fleur. Elles permettent d'accéder à l'image/fleur précédente ou suivante.

c) Objets statiques.

Le libellé.

Les libellés se retrouvent tous dans les fenêtres d'affichage et sont destinés à contenir les noms latins et suédois des fleurs ainsi que le nombre de fleurs disponible dans l'image pour les écrans de recherche spatiale.

L'icone.

L'icone est utilisé notamment pour représenter les indicateurs sonores et musicaux (cfr. références 3 et 4 de la figure 6.10).
En effet, le son et la musique ne peuvent être désactivés ou activés qu'à partir du menu camembert et ces icones sont des copies conformes des représentations utilisées dans les quarts supérieurs du menu. Cette continuité dans la représentation permet à l'utilisateur de percevoir plus aisément le lien de causalité entre ces deux objets.

d) Objets de contrôle.

L'échelle.

Une échelle est mise à la disposition de l'utilisateur lorsqu'il clique sur l'icone sonore ou sur l'icone musical afin de lui permettre de régler le volume sonore de la source correspondante. Cette échelle contient 8 positions en comptant la position nulle (volume à 0).
La figure 6.14 présente la fenêtre de réglage du volume sonore des bruitages (c'est-à-dire des chants d'oiseau). La molette de réglage se trouve à la quatrième position sur cette figure.

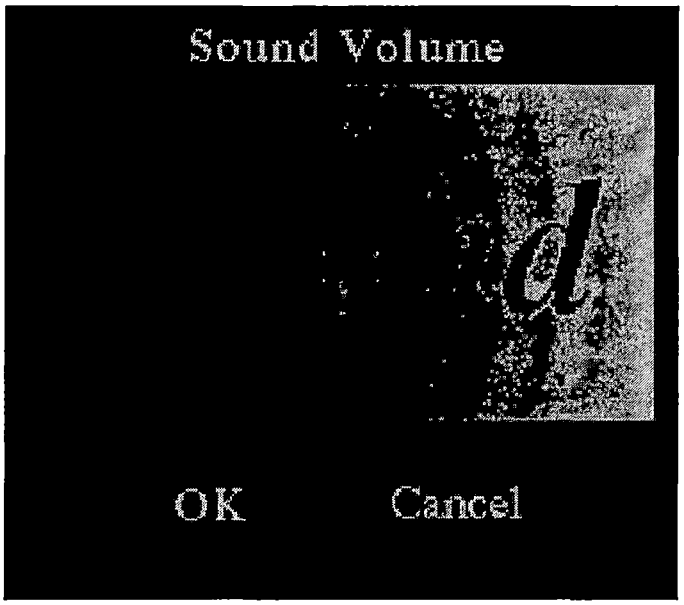


Figure 6.14. La fenêtre de réglage du volume sonore.

Le bouton de commande.

Un certain nombre de boutons de commande apparaissent dans le programme. La figure 6.14 présente deux exemples de boutons de commande ; ce sont les boutons libellés 'OK' et 'Cancel'.
Ces boutons obéissent tous aux mêmes règles de fonctionnement.
Ces règles sont celles appliquées aux boutons de commande utilisés dans le System 7 du Macintosh.
Un clic sur un bouton de commande n'est pris en compte que si et seulement si le bouton de la souris est relâché alors que le curseur se trouve toujours sur le bouton de commande. Cela permet à l'utilisateur, qui, alors que le bouton de la souris est toujours enfoncé, se rend compte qu'il ne devait pas cliquer ce bouton de commande, de se raviser. Il a ainsi la possibilité de déplacer le curseur et de relâcher le bouton de la souris à coté du bouton de commande ce qui annule le clic.

Le bouton graphique.

Il existe plusieurs boutons graphiques dans le logiciel, ce sont par exemple les deux indicateurs de fonctionnement du son et de la musique supportant les deux icones du même type.
Lorsqu'un de ces boutons est cliqué, une fenêtre de dialogue (cfr. figure 6.14) apparaît permettant de régler le volume sonore de la source correspondante. L'indicateur pour le son ou la musique n'est disponible à l'écran que si le son ou la musique est activé. En effet, il serait

ridicule de permettre de régler le volume d'une source sonore inactivée même si cela aurait été plus cohérent avec les icones des quarts supérieurs du menu camembert.

Les zones cliquables des cartes constituent aussi des boutons graphiques (étant donné qu'une erreur de clic sur ceux-ci peut s'avérer frustrante, ils obéissent aux mêmes règles que les boutons de commande).

e) Objets de dialogue.

La fenêtre.

Il s'agit, dans le logiciel, d'une fenêtre d'affichage statique (c'est à dire que son apparence et sa position ne peuvent être modifiées) apparaissant dans les écrans décrits par les figures 6.6, 6.7, 6.9 et 6.10.
Cette fenêtre permet d'afficher une image de fleur ou sa description. Les fenêtres affichant une description sont légèrement différentes des fenêtres affichant une image. De plus, une fenêtre affichant une image dans un écran de recherche temporelle est différente de son homologue dans un écran de recherche spatiale.
A la figure 6.15 et 6.16, on peut voir représenté le dialogue permis par ces fenêtres.
Le dialogue est décrit selon une forme actualisée du formalisme des réseaux état-transition[19].

Une fenêtre est en fait une agrégation d'objets interactifs tel que le bouton de commande, le libellé, la flèche de défilement, etc. Cette fenêtre constitue en quelque sorte une boîte de regroupement signifiant bien que les différents boutons inclus n'ont une portée que sur la fenêtre et les autres objets s'y trouvant intégrés.
Les fenêtres affichant une image dans les écrans de recherche spatiale requièrent, dans le cas ou plus d'une fleur est présente (cela est signalé par un indicateur du nombre de fleur), la sélection d'une fleur avant toute action de consultation (affichage du nom latin ou suédois, prononciation du nom latin ou suédois ou affichage de la description). Les fleurs de l'image sélectionnable sont signalées par la mise en évidence de la fleur lorsque le curseur de la souris la survole. Cette mise en évidence est effectuée au moyen d'un cercle bleu clair assez fin entourant la fleur. Lorsque la fleur est sélectionnée, le cercle s'épaissit et devient vert clair. Ce changement de forme et de couleur est utilisé afin de suppléer à toute carence perceptive éventuelle que pourrait induire l'aphasie (cfr. section 1.4.4). Toute ambiguïté doit donc être évitée.

[19]Dix et al, *Human-Computer Interaction*, Prentice Hall, 1993, p. 256-.

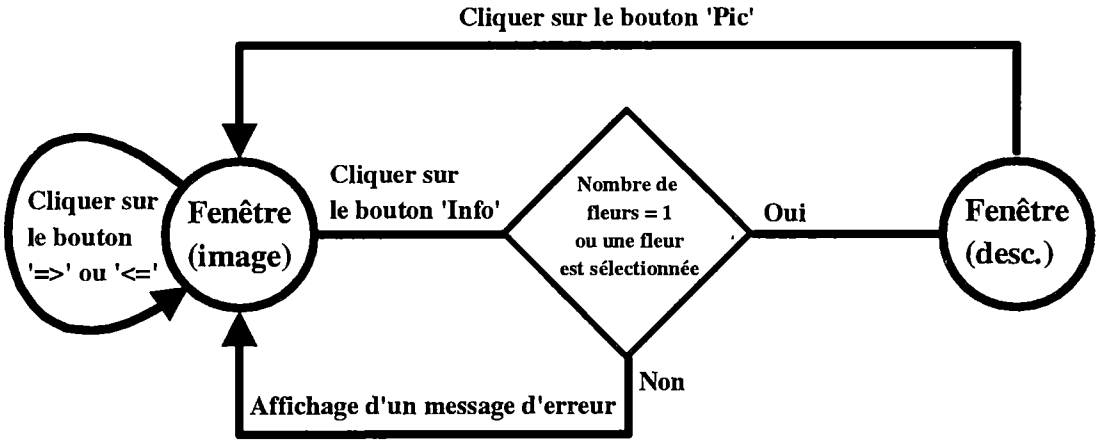


Figure 6.15. Le dialogue proposé par la fenêtre d'affichage.

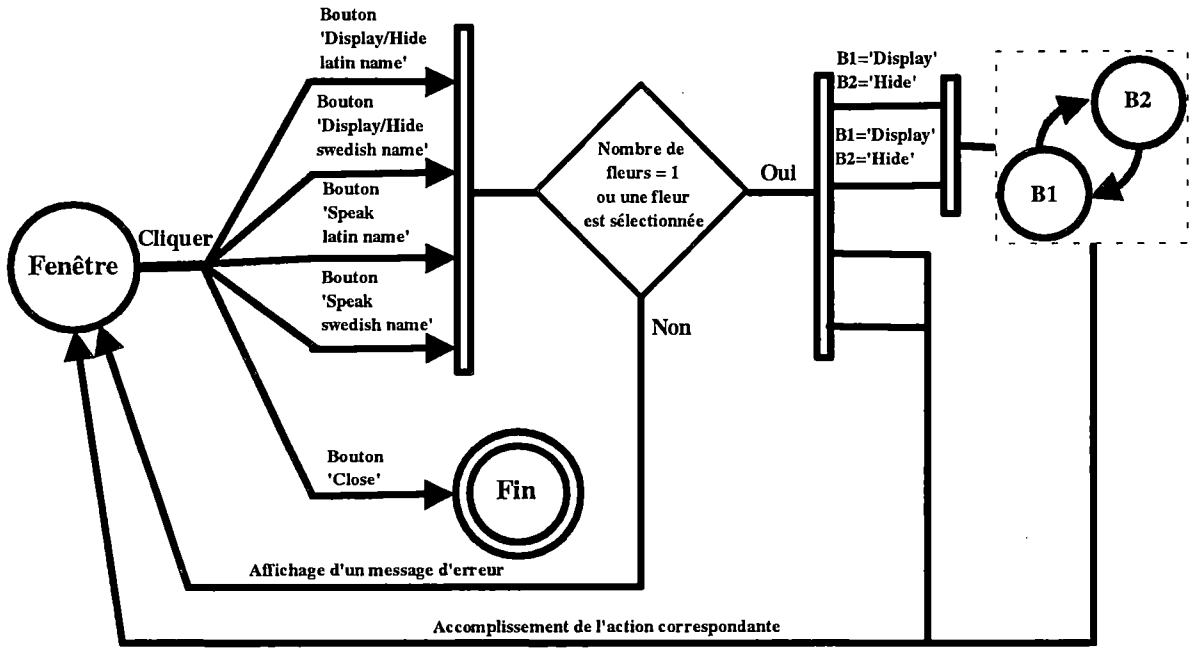


Figure 6.16. Le dialogue proposé par la fenêtre d'affichage (suite).

Le panneau de contrôle.

Ce sont les fenêtres permettant de régler le volume sonore des bruitages ou de la musique (cfr. figure 6.14). Elle contient une échelle graduée et deux boutons de commande, l'un de validation et l'autre d'annulation des changements apportés.

f) Objets de feed-back.

Les messages.

Plusieurs types de message guident l'utilisateur dans sa tâche :

- le message de confirmation : ce message apparaît lorsque l'utilisateur choisit l'option 'Quit' du menu camembert.

- le message d'erreur : ce message apparaît lorsque l'utilisateur commet une erreur de manipulation, notamment dans une fenêtre d'affichage ou dans l'écran de positionnement des drapeaux (écran de recherche temporelle).

Curseur de souris contextuel.

Un des grands principes de conception de l'interface fut de rendre toute action permise aussi visible et explicite que possible c'est-à-dire que l'utilisateur puisse à tout moment connaître ce que lui permet d'accomplir l'interface et comment l'accomplir. Un des moyens mis en oeuvre dans l'ensemble du logiciel est le curseur contextuel.
En effet, la forme du curseur de la souris change en fonction de ce qu'il survole. La nouvelle forme adoptée a été pensée de manière à rendre l'action permise à cet endroit de l'écran aussi claire que possible.

Contraintes techniques: Bien entendu, il fallait rendre ce changement de forme instantané. Tout délai entraînerait inéluctablement des erreurs de manipulation de la part de l'utilisateur. De plus, Director dans sa version 4.0 ne permet de dessiner que des curseurs monochromes et de 16x16 points au maximum.

Formes prises par le curseur:

Ceci est la forme par défaut du curseur. Il adopte cette forme lorsque le curseur survole une zone non-interactive de l'écran et lorsqu'il survole un bouton.

Le curseur adopte cette forme lorsqu'il survole une zone non-interactive à partir de laquelle le menu camembert peut être appelé.

Cette forme est prise lorsque le curseur survole une zone d'une carte à partir de laquelle l'utilisateur peut effectuer un zoom.

Le curseur prend la forme d'un niveau sur un « S » majuscule lorsqu'il survole l'indicateur de fonctionnement du son. Si l'utilisateur presse le bouton de la souris, une fenêtre de réglage du volume apparaît.

Le curseur prend la forme d'un niveau sur un « M » majuscule lorsqu'il survole l'indicateur de fonctionnement de la musique. Si l'utilisateur presse le bouton de la souris, une fenêtre de réglage du volume apparaît.

Le curseur revêt cette forme lorsqu'il survole un objet interactif déplaçable (ici un drapeau).

Le curseur prend cette apparence lorsqu'il survole un objet interactif déplaçable et que le bouton de la souris est enfoncé (cfr. infra).

Le curseur se change en doigt lorsqu'il survole une zone/objet interactif sélectionnable.

Le curseur devient une montre lorsque le système reprend la main pour entrer dans une phase non-interactive.

Barre d'aide contextuelle.

Conséquence logique des contraintes techniques sur le curseur et de la petite taille inhérente à celui-ci, la signification qui lui est attachée n'est pas toujours conforme à la réalité.
La barre d'aide contextuelle est venue suppléer cette carence. En effet, lorsque le curseur survole une région de l'écran, non seulement il change de forme si une action peut être entreprise mais en plus un bref texte explicatif est affiché sur la barre d'aide, soit pour rendre explicite le changement de forme du curseur, soit lorsque celui ci garde sa forme par défaut (pointeur) pour décrire la nature de l'objet graphique survolé.

La présence de cette barre permet de se passer totalement de toute autre forme d'aide plus classique fonctionnant par écrans interposés (ce qui fait souvent perdre le cours de sa navigation à l'utilisateur).

En outre, la nature même du logiciel n'exige pas un trop gros effort cognitif de la part de l'utilisateur au niveau de l'apprentissage, rendant amplement suffisante l'utilisation de cette aide textuelle unique.

Feed-back sonore et/ou visuel de chaque action.

Une fois le choix de l'opération à effectuer accompli, l'utilisateur doit l'entreprendre le plus souvent par une simple pression sur le bouton de la souris.
Le feed-back obtenu dépend de l'action entreprise :

- Si l'utilisateur clique sur une zone non-interactive de l'écran, rien ne se passe. Le choix a été fait de ne pas indiquer explicitement l'erreur par un message qui, en outre, retarderait l'utilisateur ou par un signal sonore qui bien que ne provoquant aucun ralentissement n'en conserve pas moins son caractère de renforcement négatif. De plus, dans le cas de Gun, il n'aurait pas été très judicieux de l'inclure à cause de la démotivation, de la frustration et de l'accroissement de fatigue qu'il provoquerait immanquablement.

- Si l'utilisateur clique sur un bouton graphique, plusieurs feed-back sont générés. Premièrement, lorsque le bouton de la souris est enfoncé, un clic sonore « descendant » retenti, accompagné du changement d'apparence du bouton (il est visuellement enfoncé, la métaphore du mini-monde est affirmée). Ensuite, lorsque l'utilisateur relâche le bouton de la souris, un clic sonore « ascendant » est généré ainsi que le changement de forme du bouton (il est maintenant revenu à sa position de départ). L'utilisateur sait donc par ces deux canaux de communication que le bouton a bien été enfoncé et que l'opération est en cours d'exécution. Connaissance renforcée par l'adoption par le curseur de la forme d'une montre afin de rassurer l'utilisateur sur l'état du système.

- Si l'utilisateur clique sur un des deux icones sonores, une mélodie particulière se fait entendre qui débouche sur l'apparition d'une fenêtre de réglage du volume. La dite mélodie est exclusivement dédiée à ces deux icones.

- Si l'utilisateur presse et garde enfoncé le bouton de la souris lorsque le curseur a la forme d'une main ouverte (ce qui se passe lorsqu'il survole un drapeau), ce curseur change à nouveau pour devenir une main fermée signifiant que le drapeau a été virtuellement saisi et qu'il peut être déplacé.

- Si l'utilisateur « dépose » le drapeau sur une zone non prévue à cet effet de l'écran, celui-ci retourne aussitôt dans la boîte d'où il provient. Ce retour est visuel (le drapeau suit une trajectoire linéaire), afin de faire comprendre à l'utilisateur qu'il n'a pas commis l'irréparable et que son drapeau ne s'est pas égaré.

- Lorsqu'une erreur de manipulation dont la nature plus complexe ne peut être expliquée par un simple signal sonore ou visuel seul survient, une fenêtre d'erreur expliquant la nature de l'erreur et le moyen de l'éviter dans le futur apparaît à l'écran. Cette fenêtre est toujours précédée d'un bref signal sonore d'avertissement. Ce signal a été choisi afin d'être le moins crispant possible.

g) Objets divers.

Cette section regroupe tous les objets interactifs non repris dans le rapport de recherche. Cette absence s'explique probablement par la spécificité de ces objets n'existant tel quel ni dans le monde System 7 du Macintosh ni dans le monde Windows.

Le drapeau.

Les drapeaux sont présents dans l'écran de départ de la recherche temporelle (cfr. figure 6.8). Leur but est de permettre à l'utilisateur de sélectionner de manière très intuitive et visuelle une ou plusieurs périodes sur l'échelle du temps.
Cette sélection se fait au moyen de deux types de drapeau, des drapeaux de début de période dont la représentation est celle d'une fleur bourgeonnant et des drapeaux de fin de période dont la représentation est celle d'une fleur fanée. Ces représentations renforcent la métaphore du jardin. L'utilisateur n'a donc pas besoin de changer de contexte, de métaphore pour comprendre le fonctionnement de ces drapeaux.

Ces drapeaux sont donc manipulables, l'utilisateur peut les « saisir » au moyen du curseur de la souris dont la forme s'adapte en conséquence et le relâcher n'importe où sur l'écran. Si l'endroit ne convient pas au positionnement d'un drapeau, il retourne à sa position de départ dans la boîte à outils.
De plus, le placement des drapeaux obéit à un certain nombre de règles que nous avons jugé être à même d'aider l'utilisateur au cours de cette opération. Par exemple, Un drapeau de début doit toujours être placé avant un drapeau de fin, le nombre de drapeau de début ne peut donc être à tout moment que de un supérieur au nombre de drapeaux de fin.
Toute erreur de manipulation qui ne peut être expliquée visuellement (comme par le retour à la position de départ) l'est au moyen d'un message d'erreur explicatif de la situation et de la manière de l'éviter par la suite.

Ces drapeaux ainsi que la boîte à outils les contenant sont placés à un endroit de l'écran qui pourrait paraître incongru étant donné la loi de Fitts[20] . En effet, étant placés à l'extrême gauche de l'écran, ils ne sont pas équidistants des positions possibles de l'échelle du temps. Cependant, les placer au milieu supérieur de l'écran ne nous semblait pas un choix très judicieux. Nous avons joué la carte de l'impact visuel maximal afin de concrétiser la métaphore du jardin et de réduire le sentiment d'utiliser une interface potentiellement contraignante. En effet, l'image de fond de cet écran apporte beaucoup à la compréhension de sa nature et de ce qu'il permet d'obtenir (cfr. description de cette image dans la légende de la figure 6.8).

L'échelle de temps.

Cette échelle apparaît donc elle aussi dans l'écran décrit par la figure 6.8.

[20]Fitts, P.M. et Posner, M.I., *Human Performance*, Wadsworth Publishing, 1967.

Elle représente une année et est divisée en douze intervalles de un mois libellés de janvier à décembre et en treize séparations. Ces séparations vont servir de points d'attraction pour les drapeaux c'est à dire que les drapeaux se retrouveront toujours sur un de ces supports, peu importe l'endroit de l'échelle où ils ont été placés. En effet, les périodes de temps que l'utilisateur peut déterminer sont des multiples de un mois.

Cette attraction citée quelque ligne plus haut est concrétisée par une zone d'attraction de 1 mois centrée sur la séparation. Tout drapeau relâché dans la zone d'attraction d'une séparation se repositionne automatiquement sur la séparation. La treizième séparation ne peut être occupée que par un drapeau de fin et une même séparation ne peut être bien évidemment occupée que par un drapeau à la fois.

Cette échelle de temps est positionnée de manière à coïncider avec les saisons représentées sur la photo de fond.

La fenêtre de prévisualisation.

Dans les écrans représentés à la figure 6.5, une carte contenant des zones cliquables permet à l'utilisateur de choisir grâce à des repères visuels la section du parterre à examiner. Ces repères étant des éléments de décor non cliquables (représentés en noir) et des formes des fleurs contenues dans la section cliquable. Mais ces formes ne sont pas toujours suffisantes (loin de là même) pour reconnaître la fleur.
C'est pourquoi nous avons décidé d'y ajouter une fenêtre de prévisualisation. Cette fenêtre de taille assez réduite n'apparaît pas en permanence à l'écran mais uniquement lorsque le curseur de la souris survole une des zones cliquables. L'image en miniature qui apparaîtra dans la fenêtre d'affichage (figure 6.6) est ainsi dévoilée dans la fenêtre de prévisualisation. Si le curseur se trouve sur une autre partie de l'écran, la fenêtre reste invisible pour la même raison expliquant le positionnement de la boîte à outils de la section précédente.

6.3.3. Grands principes ergonomiques de conception.

Pour exprimer les principes ergonomiques de conception que nous avons suivi tout au long du développement du logiciel, nous allons nous référer à certaines des 8 règles d'or de la conception du dialogue[21].

[21] Schneiderman, B., *Designing the User Interface - Strategies for Effective Human-Computer Interaction*, Addison Wesley, 1992.

a) Cohérence intra-application.

Un des grands principes de conception de l'interface fut de rendre toute interactivité aussi prévisible que possible. A cette fin, les écrans ont été conçus afin d'être cohérents entre eux. Pour ce faire, nous avons d'abord défini un ensemble de comportements et d'objets interactifs qui seront présents dans tous les écrans.

Ces objets sont les suivants :

- les deux icones sonores présents au même endroit tout au long du logiciel sous réserve, bien entendu, de l'activation ou de la désactivation du son et de la musique (cfr. description des objets interactifs).

- la barre d'aide contextuelle qui est toujours de même taille et visible au bas de l'écran si, bien sûr, l'aide est activée.

- le ou les boutons commandant le passage d'un écran à un autre et le changement de l'image de fond forment un bloc toujours de même taille et placé au même endroit de l'écran (cfr. prévisibilité et loi de Fitts[22]).

- le menu camembert est accessible de la même façon à partir de n'importe quel écran et présente toujours les mêmes options à la même place.

- les fenêtres d'affichage apparaissent toujours au même endroit et présentent les mêmes boutons à la même place (à quelques détails près suivant qu'il s'agit d'une fenêtre d'affichage simple ou d'affichage/sélection d'image ou d'affichage simple de la description d'une fleur).

Les comportements sont les suivants :

- lorsqu'une erreur de nature complexe survient, le système la signale toujours au moyen d'un message d'erreur affiché dans une fenêtre centrée sur l'écran et contenant un seul bouton afin de la fermer. L'utilisateur est averti au préalable par un signal

[22]Fitts, P.M. et Posner, M.I., *Human Performance*, Wadsworth Publishing, 1967.

sonore qu'il vient de commettre une erreur et qu'il va avoir à lire et valider (pression du bouton 'OK') un message d'erreur.

- tout le dialogue repose sur une conception orientée objet de l'interactivité, c'est-à-dire que lorsque l'utilisateur veut effectuer une opération, il doit d'abord sélectionner l'objet de cette interaction et ensuite l'action à accomplir (par exemple, dans une fenêtre d'affichage-sélection (cfr. figure 6.6), l'utilisateur doit d'abord sélectionner une fleur sur l'image si plus d'une est présente et ensuite cliquer sur un des boutons au bas de la fenêtre afin d'afficher les différents noms, de les prononcer ou d'obtenir une description. Un message d'erreur le lui signale s'il ne se conforme pas à cette procédure).

- l'exploration des différents parterres du jardin se fait de la même manière. Dans chacun de ces écrans, les boutons sont toujours les mêmes et placés au même endroit au pixel près. Cependant certains boutons peuvent être différents (par exemple, si une seule image de fond est disponible, le bouton 'Change view' devient grisé et l'information affichée dans la barre d'aide reflète cet état particulier) mais même s'ils deviennent inutiles, restent présent à l'écran afin de ne pas changer le positionnement des autres objets graphiques et d'avertir l'utilisateur du cas d'exception tout en le justifiant.

b) Offrir un feed-back informatif.

Dans la conception de notre logiciel, nous avons prévu un feed-back pour tout type d'action de l'utilisateur, ce feed-back peut être uniquement visuel ou sonore lorsque l'action est simple ou coutumière mais il peut être un message d'erreur lorsque l'action entreprise est plus complexe. Le but était de ne jamais laisser l'utilisateur dans l'expectative.

Par exemple, toute pression du bouton de la souris sur un objet cliquable renvoie un « beep » sonore significatif :

- un clic sur un bouton de commande renvoie un bruit de pression et de relâchement de bouton.
- un clic sur un des deux icones sonores renvoie une brève mélodie.
- un clic sur une fleur sélectionnable d'une image renvoie un bruit de pression de bouton.

Ce feed-back est aussi fourni en permanence par la barre d'aide contextuelle et par le curseur contextuel (cfr. sections correspondantes).
Lorsque le curseur de la souris survole les zones cliquables d'une image (cartes ou photographies de fleurs), un cercle ou un cadre délimite la partie qui sera sélectionnée si l'utilisateur clique sur le bouton.

De plus, une autre forme de feed-back est offerte à l'utilisateur, nous pourrions l'appeler permanence visuelle du contexte. Lorsque l'utilisateur clique sur une partie de la carte du jardin (référence 2 de la figure 6.4), il aboutit dans un écran dont l'image de fond représente une vue de l'endroit visité. A partir de là, il peut sélectionner une section du parterre à explorer à partir d'une nouvelle carte. Lorsqu'il clique sur une des sections, une fenêtre apparaît sur l'image de fond, contenant une image de la section choisie. A coté de cette fenêtre, une miniature de la dernière carte rencontrée sur laquelle sont placés un ou plusieurs pointeurs indiquant la ou les positions occupées par les fleurs de la section choisie est visible en permanence. Si l'utilisateur *browse* au travers des différentes sections/fleurs du parterre à l'aide des boutons de commande '=>' et '<=' de la fenêtre, le ou les pointeurs sont mis à jour.

Cette procédure est appliquée de manière similaire dans les écrans décrits par les figures 6.9 et 6.10.

c) Offrir une gestion simple des erreurs.

Comme nous l'avons vu, le système a été prévu afin que l'utilisateur fasse le moins d'erreurs de manipulation ou d'interprétation possibles. Chaque action possible est rendue claire grâce notamment au curseur contextuel et à la barre d'aide contextuelle. Dans les cartes de parterre (figure 6.5), une fenêtre de prévisualisation permet de choisir la zone à sélectionner avec un grand degré de certitude.
L'encadrement de la zone cliquable actuellement survolée par le curseur de la souris avertit l'utilisateur de ce qu'il va sélectionner et des limites géographiques de cette sélection.

Tous ces éléments déjà analysés plus haut sont autant d'indicateurs du résultat que l'accomplissement d'une action donnera.

Cependant une erreur peut toujours se produire lors de manipulations particulièrement complexes que de simples indicateurs ne peuvent expliciter. Ces manipulations se rencontrent dans les écrans décrits par les figures 6.6 et 6.8. Lorsqu'une erreur est commise, un message d'erreur en décrivant la cause et le remède est affiché à l'écran.

6.4 Description de l'implémentation du logiciel.

Bien que nous ayons vu le logiciel avant tout comme une interface lors de la conception détaillée et de l'implémentation, nous avons essayé de garder à l'esprit un certain nombre de principes d'abstraction et de généralisation.
Ces principes portent notamment sur le caractère modulaire et réutilisable de certaines procédures ou *handlers* pour conserver la terminologie propre à Director et sur la possibilité de modifications ultérieures par d'autres personnes.

6.4.1 Conception orientée LINGO.

Dès le début de l'implémentation, nous avons essayé de tirer parti au maximum de la puissance de LINGO. En effet, comme nous l'avons vu au chapitre 3, le concepteur peut, lors du développement d'une application sous Director, choisir entre simplicité et puissance.

Il lui est ainsi loisible d'utiliser au maximum les possibilités de la *Score Window* en ne faisant appel à LINGO que pour gérer les branchements et les fonctionnalités propres à l'application. Dans ce cas, le concepteur va utiliser au maximum les 48 *Animation Channels* et va avoir besoin d'un très grand nombre de *frames*. De fait, chaque écran différent c'est-à-dire chaque écran où ne fut ce qu'un détail diffère, va occuper une *frame*.

Ex. Dans le passage de l'écran de la figure 6.5 à un écran de la figure 6.6, il aurait été extrêmement simple de créer un nouvel écran (occupant une nouvelle *frame*) pour chaque nouvelle photographie à afficher dans la fenêtre de visualisation (référence 5 de la figure 6.6). Pour ce faire, il aurait suffi d'exécuter un copier-coller de tous les éléments constants de cet écran (c'est-à-dire le contenu de la totalité des cellules de la *frame* à l'exception de celle contenant l'image) et de ne changer que la photographie à afficher dans chaque écran.

Un des désavantages évidents de cette approche est une surmultiplication des *frames* nécessaires, rendant de ce fait la lecture et la maintenance d'une application nettement plus ardues. En effet, il peut arriver que des dizaines de *frames* soient identiques au niveau du comportement mais légèrement différentes au niveau du contenu graphique ou sonore de l'écran. Les scripts de ces *frames* seraient donc semblables à peu de chose près. Si le concepteur de l'application décide de modifier légèrement le comportement de ce type d'écran et ainsi de modifier le ou les scripts en assurant la gestion, il est invariablement forcé de reporter manuellement la modification sur tous les scripts similaires (pour garantir la cohérence entre écrans de même type). Il en va de même pour toute modification de la présentation d'un de ces écrans. De ce fait, une surcharge de travail totalement inutile pourrait résulter de cette approche.

Un autre désavantage est d'arriver très vite à une saturation des *Animation Channels*. Le nombre *d'Animation Channels* disponibles (48) peut paraître de prime abord assez confortable mais, au fur et à mesure, on se rend compte que c'est finalement assez peu et que, si l'on n'y prend pas garde, cela peut se révèler nettement insuffisant.

Ex. Pour résoudre le problème posé dans l'exemple précédent, une autre possibilité est de n'utiliser qu'une seule *frame* pour représenter un écran du type de celui de la figure 6.6 et d'utiliser une *Animation Channel* par photographie possible pour cet écran. Si le parterre modélisé par l'écran en question n'est composé que de quelques photographies différentes alors cela ne pose pas de problème et c'est assurément la meilleure solution. Cependant, dans le logiciel développé, un des parterres comprend une vingtaine d'images différentes requérant ainsi une vingtaine *d'Animation Channels* rien que pour pouvoir afficher à l'écran et à un moment donné un seul objet graphique (il suffit dans ce cas de laisser la seule photographie à afficher visible et de rendre les autres transparentes).

A contrario, le concepteur peut décider d'adopter une approche centrée beaucoup plus sur LINGO.

Ex. La solution que nous avons apportée à l'exemple précédent obéit à cette approche. Une seule *frame* est utilisée pour l'affichage de toutes les photographies d'un même parterre. Il existe autant de *frames* contenant un écran du type représenté à la figure 6.6 qu'il y a de parterres dans le jardin et donc d'écrans du type de celui modélisé à la figure 6.5. Cette solution nécessite cependant une bonne maîtrise de LINGO et l'utilisation de nombreuses listes pour gérer les noms des photographies à afficher.

En fait, nous avons opté pour cette approche tout au long du développement de l'application « Gun's Garden ». Cela permet notamment d'accroitre la lisibilité de la *Score Window* par une réduction du nombre de *frames* et de faciliter la modification des écrans par une réduction de la redondance. Cela permet aussi de limiter le nombre *d'Animation Channels* occupés et ainsi d'accorder plus de liberté au concepteur. Cependant, cette approche implique aussi des désavantages. Cette approche induit notamment une difficulté accrue de la programmation. Dès lors, il faut posséder une assez grande maîtrise de la programmation en générale et de LINGO en particulier pour s'en sortir tout en conservant une architecture « propre ». Et ce d'autant plus que, comme nous l'avons vu dans le chapitre 3, LINGO n'offre qu'une gestion élémentaire des structures de données.

Si l'on analyse la *Score Window* de notre application, on constate qu'il aurait encore été possible de réduire le nombre de *frames* utilisés mais cela aurait, à notre avis, grandement nuit à la lisibilité de la structure de l'application. Notre choix d'implémentation respecte un certain équilibre entre les deux approches que nous venons d'exposer. Cela permet donc de conférer à l'application une certaine lisibilité tout en conservant un code relativement concis et par là même simple à maintenir.

6.4.2 Modularisation.

Toujours dans un souci de faciliter toute modification de notre part au cours du développement et toute maintenance ultérieure, nous avons essayé de pratiquer une certaine modularisation des scripts.
Il ne faut pas entendre modularisation dans l'acception habituelle et complète de ce terme mais plutôt comme la mise en évidence (et en procédure) de morceaux de code utilisés par plusieurs scripts. Cela nous a permis, notamment, de réduire la taille des scripts et ainsi de faciliter la conception de l'application. En effet, plutôt que de reporter une même modification dans plusieurs scripts, une seule modification au sein d'un même script suffit.

Les « modules » ainsi créés ne sont pas forcément des « unités autonomes et indépendantes ». Un script de ce type peut ne contenir que quelques lignes de texte représentant une gestion partielle d'une tâche mais commune à de nombreux scripts.

Ex. Le script 346 placé en annexe est un bon exemple de ces « modules d'aisance ». Ce script effectue une partie (la partie commune) de la gestion du test du survol des différents objets graphiques par le pointeur de la souris et est appelé par le script 403 dans les écrans du type de celui présenté à la figure 6.6.

Suivant l'objet survolé, un numéro de message d'aide contextuelle (affiché sur la barre d'aide contextuelle) est ainsi affecté à une variable retournée comme résultat.

Un « module » ainsi défini peut aussi être un module au sens classique, à savoir une unité d'exécution autonome réalisant une tâche déterminée. Nous allons, pour illustrer ce point, analyser un script représentatif à maints égards de ce concept. Bien que Director propose des boutons de commande standards dont le comportement est géré de manière transparente par le logiciel auteur, nous avons reprogrammé ce comportement afin qu'il puisse s'adapter à n'importe quelle paire de *sprites* (représentant un bouton en position « haute » et un bouton en position « basse »).

Script 380 : Click_button.

```
on click_button button_name, channel

  click_1    Appel du script « click_1 » jouant un son représentant une pression du bouton
             de la souris.

  set name to button_name&"_dw"      Name contient le nom du sprite représentant un
                                     bouton (button_name) en position « basse »
                                     (extension "_dw").
  set c1 to the left of sprite channel
  set c2 to the top of sprite channel
  set c3 to the right of sprite channel
  set c4 to the bottom of sprite channel
  set the castnum of sprite channel to the number of cast name
  spritebox channel, c1, c2, c3, c4
  updatestage    Le sprite du bouton en position « basse » de nom name est affiché aux
                 anciennes coordonnées du sprite du bouton en position « haute » qui
                 occupait l'Animation Channel numéro channel.

  set ok to true
  repeat while the mousedown  Tant que le bouton de la souris est maintenu enfoncé,
    if rollover(channel) then
      set ok to true
      set name to button_name&"_dw"
      set the castnum of sprite channel to the number of cast name
```

```
      spritebox channel, c1, c2, c3, c4
      updatestage
```
Si le pointeur de la souris survole le sprite du bouton alors afficher le sprite du bouton en position « basse ».

```
    else
```
Si le pointeur de la souris ne survole plus le sprite du bouton alors afficher le sprite du bouton en position « haute ».

```
      set ok to false
      set name to button_name&"_up"
      set the castnum of sprite channel to the number of cast name
      spritebox channel, c1, c2, c3, c4
      updatestage
    end if
  end repeat

  if ok then
```
Si le bouton de la souris est relaché alors que le pointeur survolait le sprite du bouton, appeler le script « click_2 » jouant un son représentant un relachement du bouton de la souris. Ensuite, afficher le sprite du bouton en position « haute » et signaler l'accomplissement de l'action « cliquer sur un bouton ».

```
    click_2

    set name to button_name&"_up"
    set the castnum of sprite channel to the number of cast name
    spritebox channel, c1, c2, c3, c4
    cursor 4
    updatestage
    set complete to true
  else
```
Si le bouton de la souris est relaché alors que le pointeur ne survolait plus le bouton, signaler le non-accomplissement de l'action « cliquer sur un bouton ».

```
    set complete to false
  end if

  return complete
```
Retourner comme résultat le signal de l'accomplissement ou du non-accomplissement de l'action « cliquer sur un bouton ».

```
end
```

Comme on peut le voir, ce script est d'un usage tout à fait général puisqu'il gère n'importe quel bouton composé d'une paire de *cast members* (paramètre `button_name` avec extension `"_dw"` et `"_up"`) et situé sur n'importe quel *Animation Channel* (paramètre `channel`) d'une *frame*. De fait, tous les boutons de commande de l'application « Gun's Garden » sont gérés par ce script.

Des scripts similaires gèrent le comportement des différents objets d'action, de contrôle et de dialogue que l'on retrouve tout au long de l'application tel que le menu camembert, la fenêtre de visualisation, les boutons graphiques, etc.

6.4.3 Centralisation des déclarations.

Afin de faciliter d'une part la maintenance et les modifications de conception et d'autre part la traduction du logiciel en suédois (rappelons que l'application a été développée en anglais), nous avons rassemblé au sein d'un même script la totalité des déclarations des variables globales, des constantes, des listes et des messages d'aide contextuelle.
Ce script est du type `on startmovie`, ce qui signifie qu'il est exécuté une seule fois au lancement de l'application, et porte le numéro 452 dans les annexes.
Cette façon de faire, bien qu'entraînant une difficulté accrue dans le passage des paramètres aux scripts et dans la gestion des variables globales, nous a permis de centraliser une partie des modifications du code inhérentes aux modifications du prototype.

Nous avons aussi placé les descriptions des fleurs du jardin dans un unique fichier ASCII aisément modifiable. Il n'est donc pas nécessaire d'éditer le code source pour modifier les différentes valeurs décrivant une fleur.

Grâce à cela, la traduction de « Gun's Garden » nous a pris tout au plus une heure (en tenant compte de la modification des objets graphiques contenant du texte au format bitmap).

6.4.4 Positionnement relatif des objets graphiques.

Dans notre critique de MacroMedia Director 4.0, nous avons signalé l'inexistence de la possibilité de concevoir des objets graphiques complexes composés d'objets graphiques plus simples (cfr. section 3.6).

Pour pallier ce manque, nous avons décidé de désigner un élément de référence (qui est un objet graphique simple) pour chacun des objets complexes employés et de positionner les autres relativement par rapport à ses coordonnées.

Ex. Une fenêtre de visualisation du type de celle présentée aux figures 6.6, 6.7, 6.9 et 6.10 a été conçue de cette manière. Nous avons pris pour référence les coordonnées du coin supérieur gauche du *sprite* représentant le cadre et nous avons positionné les autres éléments de la fenêtre (boutons de commande, libellés, images affichées, etc.) relativement à ces coordonnées.

Ex. Dans l'écran de recherche temporelle (cfr. figure 6.8), le coin supérieur gauche de l'échelle de temps a été choisi comme référence. Les positions valides des drapeaux sur l'échelle ont été calculées à partir de cette référence.

Cette méthode de programmation peut paraître évidente mais, dans la pratique, la relative facilité offerte par Director pour la composition des écrans peut induire le concepteur à préférer le positionnement manuel des différents objets graphiques. Il s'agit en fait d'un piège. Bien que dans un premier temps la création des écrans soit nettement plus rapide puisque toute programmation est superflue, dans un second temps, une modification de la position d'un élément d'un objet complexe force le concepteur à faire de même avec les autres éléments. Cela peut se révéler fastidieux lorsque ce genre de modification survient régulièrement ou lorsque le nombre d'éléments composant l'objet complexe est élevé.

6.5 Réapprentissage proposé par le logiciel.

Le logiciel « Gun's Garden » a été conçu de manière à ne pas être ressenti par Gun comme un logiciel de réapprentissage au sens propre. Gunilla Knall, la thérapeute de Gun au CERTEC, considérait le futur produit comme une interface de communication entre Gun et son entourage prenant comme sujet le jardin de Gun. Le logiciel devait en quelque sorte faciliter le partage de la passion de Gun avec son entourage en suppléant aux carences lexicales engendrées par son manque du mot. Grâce à cette aide, Gun effectuerait de manière totalement transparente un réapprentissage se situant à deux niveaux.

6.5.1 Réapprentissage lexical.

Ce réapprentissage à pour objectif une atténuation du manque du mot observé chez Gun. Le lexique proposé par le logiciel est composé de noms de fleurs en latin et en suédois.
Gun peut au cours d'une discussion désirer parler d'une fleur particulière dont elle ne parvient pas à se rappeler le nom. Elle en connaît approximativement la localisation dans son jardin et va pouvoir, via plusieurs niveaux de zoom, accéder à une fenêtre en affichant une photographie. Ensuite, en manipulant les boutons de commande, si elle ne parvient toujours pas à la nommer, elle va pouvoir activer plusieurs types de facilitation. Elle peut faire afficher le nom latin de la fleur et, éventuellement, le faire prononcer. Elle peut de la même manière en obtenir le nom suédois.
Comme on peut le voir, c'est Gun qui pilote ce « réapprentissage ». Elle choisit quelle fleur afficher et quelle facilitation utiliser. Elle n'est tenue par aucun questionnaire ni par aucun résultat à améliorer. Aucun message d'erreur n'intervient si elle ne parvient pas à nommer une fleur car ces exercices de dénomination sont totalement implicites.

6.5.2 Rééducation de l'orientation spatiale et temporelle.

En utilisant le logiciel, Gun effectue un réapprentissage de l'orientation spatiale et temporelle. En effet, pour accéder à la fenêtre de visualisation des fleurs, Gun peut effectuer une recherche sur base de deux critères. Soit elle connaît l'emplacement d'une fleur dans le jardin et elle va effectuer une recherche spatiale, soit elle connaît sa période de floraison et elle va effectuer une recherche temporelle.

Dans le premier cas, elle va pouvoir se balader dans son jardin par l'intermédiaire de différents écrans proposant une vue plus ou moins du jardin ou d'un parterre. Elle commence sa recherche à partir d'une pseudo-vue aérienne de son jardin, puis elle effectue un zoom sur un parterre et enfin sur une section du parterre pour arriver à une fenêtre de visualisation affichant une photographie de la section. Cette photographie, suivant les cas, renferme une ou plusieurs fleurs (pour un maximum de six) qu'elle devra sélectionner avant d'obtenir la moindre information. Cet « exercice » est tout à fait similaire à ce qui est pratiqué dans les centres de rééducation où le malade doit retrouver son chemin à l'aide d'une carte. Un moyen de facilitation unique est proposé dans l'écran où est affichée la fenêtre de visualisation. Il s'agit d'une miniature du parterre visité où se trouve indiqué par un marqueur l'emplacement de la section dont la représentation est affichée dans la fenêtre.

Dans le second cas, elle va avoir la possibilité de sélectionner une ou plusieurs périodes de temps sur une échelle graduée en mois et couvrant l'ensemble de l'année. Un moyen de facilitation est tout de suite proposé puisque, en arrière plan et en correspondance avec l'échelle, une image représentant explicitement les quatre saisons est affichée. Elle pourra ensuite valider cette sélection et accéder directement à la fenêtre de visualisation.
Un autre moyen de facilitation, spatiale celui-là, est proposé dans l'écran où est affichée la fenêtre de visualisation. Il permet de faire le lien avec l'orientation spatiale. Il s'agit d'une miniature de la carte du jardin où est indiqué la localisation du ou des parterres contenant la ou les fleurs affichées.

Conclusion.

Le développement de logiciels pour personnes soufrant d'un handicap est un problème épineux pour un informaticien. En effet, il doit faire table rase de tous ses préjugés en matière d'ergonomie des logiciels. Il doit envisager le problème d'un point de vue nouveau : la personne handicapée est limitée dans ses capacités. Ses limites peuvent avoir des conséquences sur son utilisation des moyens d'interactions. Elles peuvent aussi se répercuter dans sa compréhension d'une interface.
Pour bien déterminer ces limitations, en raison de son ignorance de tous les aspects du handicap envisagé, l'informaticien doit souvent se référer au représentant de la personne handicapée. De ce fait, la méthodologie classique de conception est souvent inapplicable et la seule approche valable est celle par prototypage. Nous avons été confrontés à cette situation et nous avons pu nous rendre compte de toutes les difficultés que cela implique.

Ce mémoire a essayé de présenter en quelque sorte l'approche idéale de conception d'applications pour personnes aphasiques. En effet, la maladie doit d'abord être clairement définie et les principaux aspects utiles pour la conception doivent être déterminés. De même, le canevas général d'un réapprentissage non-informatique doit être identifié afin de bénéficier de toute cette expérience accumulée. Nous avons tenté de réaliser cela dans les deux premiers chapitres de ce document.
L'informaticien doit ensuite déterminer, d'après ce qui lui est demandé, le meilleur outil pour le développement de l'application particulière. Pour ce faire, il a besoin de connaître les potentialités des différents standards du marché et il doit en permanence garder à l'esprit ce qu'il aimerait que l'outil puisse faire. Bien que nous n'ayons traité dans le chapitre 3 que d'un seul logiciel auteur, nous avons essayé de le comparer avec les langages de programmation classiques. Dans le chapitre 4, nous avons établi le cahier des charges type d'un logiciel auteur dédié (décrivant le logiciel auteur idéal pour ce type d'application).
Enfin, le fruit de nos efforts est présenté au chapitre 6.

Bien entendu, cette approche a été dans notre cas établie a posteriori. Il est, de plus, évident que cette approche est très peu praticable telle quelle. Il est, en effet, impensable d'espérer confier la réalisation de ce genre de projet à un informaticien disposant de solides connaissances dans le traitement d'un handicap particulier. C'est pourquoi, à notre avis, ce genre de projet devrait être destiné à des équipes pluridisciplinaires composées non seulement de médecins et d'informaticiens mais aussi de graphistes et de musiciens professionnels. Ce type d'équipe assurerait les plus grandes chances de succès à un projet de cette envergure dans les délais les plus brefs.

Malheureusement, la réalité est encore assez éloignée de l'idéal. Des équipes similaires à ce que nous venons de dire existent bel et bien mais développent la plupart du temps des logiciels destinés au plus grand nombre d'utilisateurs possible (rentabilité oblige) alors que c'est d'applications personnalisées et développées ponctuellement qu'on besoin les thérapeutes. Il est de ce fait assez difficilement envisageable que cela puisse s'améliorer, le jeu de l'offre et de la demande étant rarement en faveur des personnes handicapées.

Une solution de rechange que nous avons envisagée est le développement d'un logiciel auteur dédié à la conception d'exercices pour personnes handicapées. Ce logiciel devrait pouvoir être maîtrisé très facilement par un thérapeute qui pourrait ainsi, suivant ses besoins, réaliser ses propres exercices. Cette solution n'est pas nouvelle[23] mais doit encore être améliorée afin d'être assez puissante pour ne laisser comme limites au thérapeute que celles de son imagination...

[23] Platteeuw, P., *Adaptation du système « Auteur! » pour la réalisation d'exercices destinés aux personnes handicapées,* Institut d'informatique, 1994.

Bibliographie.

[1] Bègue, D-G., *Director 4, le plein d'énergie multimédia*, dans SVM MAC, juin 1994.

[2] Caroll, J., Mack, R. et Kellogg, W., *Interface Metaphor and User Interface Design*, dans Helander, M., *Handbook of Human-Computer Interaction*, Elsevier Science Publishers, 1988.

[3] Cassagne, J. et Roux, I., *Director 4.0, Apple Media Tool 1.1, Hypercard 2.2, Trois voies pour la création multimédia*, dans SVM, mars 1995.

[4] Cazayus, P., *L'aphasie du point de vue du psychologue*, Dessart & Mardaga, 1977.

[5] Denning, P. et Dargan, P., *A Discipline of Software Architecture*, dans *Interactions*, janvier 1994.

[6] Dix, A., Finlay, J., Abowd, G. et Beale, R., *Human-Computer Interaction*, Prentice Hall, 1993.

[7] Dumortier, A-M., *Rééducation des troubles associés chez un patient cérébro-lésé*, CESEPT-IPSKT section ergothérapie, 1994.

[8] Fitts, P. et Posner, M., *Human Performance*, Wadsworth Publishing, 1967.

[9] Fontesse, A. et L'Homme, B., *Connaissance du corps interne : Réalisation d'un logiciel d'apprentissage pour personnes ayant une déficience mentale*, Institut d'informatique, 1991.

[10] Gibbs, S. et Tsichritzis, D., *Multimedia Programming*, Addison-Wesley, 1994.

[11] Goffinet, L., *Le multimédia et ses systèmes auteurs*, Institut d'informatique, 1995.

[12] Howard, D. et Hatfield, F., *Aphasia therapy*, Lawrence Erlbaum Associates, 1987.

[13] Hutchins, E., Hollan, J. et Norman, D., *Direct Manipulation Interfaces*, dans Norman, D. et Draper, S., *User Centered System Design*, Lawrence Erlbaum Associates, 1986.

[14] Knall, G. et Andersson, G., *Gun Andersson och hennes väg framåt*, Internrapport från CERTEC, 1994.

[15] Lepoutre, T. et Roquet, J., *La personne handicapée mentale et la connaissance du corps humain : un logiciel d'apprentissage*, Institut d'informatique, 1990.

[16] Myers, B., *Challenge of HCI Design and Implementation*, dans *Interactions*, janvier 1994.

[17] Murray, J., *Some Perspectives on Visual Depth Perception*, dans *Computer Graphics*, mai 1994.

[18] Nielsen, J., *Noncommand User Interfaces*, dans *Communications of the ACM*, avril 1993.

[19] Nielsen J., *The Usability Engineering Life Cycle*, dans *Computer*, mars 1992.

[20] Oléron, P., *Langage et développement mental*, Dessart, 1972.

[21] Phillips, R., *Producing Interactive Computer-Based Learning Projects*, dans *Computer Graphics*, février 1994.

[22] Platteeuw, P., *Adaptation du système « Auteur! » pour la réalisation d'exercices destinés aux personnes handicapées,* Institut d'informatique, 1994.

[23] Robertson, G., Card, S. et Mackinlay, J., *Information Visualization Using 3D Interactive Animation*, dans *Communication of the ACM*, avril 1993.

[24] Roch Lecours, A et Lhermitte, F., *L'aphasie*, Flammarion médecines-sciences, 1979.

[25] Sacré, B., Provot-Sacré, I. et Vanderdonckt, J., *Une description orientée objet des objets interactifs abstraits utilisés en Interface Homme-Machine*, Institut d'informatique, 1992.

[26] Schneiderman, B., *Designing the User Interface - Strategies for Effective Human-Computer Interaction*, Addison-Wesley, 1992.

[27] Seron, X., *Aphasie et neuropsychologie*, Mardaga, 1979.

[28] Seron, X., *La neuropsychologie cognitive*, Presses Universitaires de France, 1993.

Annexes

Script n° 339 :Back to mi_band

```
on mousedown
  Back_to("mi_band")
end
```

340 : Back to ne_corner

```
on mousedown
  Back_to("ne_corner")
end
```

341 : Back to Nelh_band

```
on mousedown
  Back_to("ne_lhband")
end
```

342 : Back to Neuh_band

```
on mousedown
  Back_to("ne_uhband")
end
```

343 : Back to Pl_band

```
on mousedown
  Back_to("pl_band")
end
```

344 : Back to S_band

```
on mousedown
  Back_to("s_band")
end
```

345 : Back to se_band

```
on mousedown
  Back_to("se_band")
end
```

346 : Buttons&icons test

```
on Buttons_icons_test
  global gsound, gmusic

  set mess_num to 1

  if rollover(30) then set mess_num to 4
  if rollover(12) then set mess_num to 18
  if rollover(13) then
    set cnum to the castnum of sprite 13
    if the name of cast cnum = "Button_hide_lat_name_up" then set mess_num to 59
    else set mess_num to 19
  end if
  if rollover(14) then set mess_num to 20
  if rollover(15) then
    set cnum to the castnum of sprite 15
    if the name of cast cnum = "Button_info_up" then set mess_num to 23
    else if the name of cast cnum = "Button_pic_up" then set mess_num to 24
  end if
  if rollover(16) then
    set cnum to the castnum of sprite 16
    if the name of cast cnum = "Button_prev_up" then set mess_num to 21
```

```
    else if the name of cast cnum = "Button_noprev" then set mess_num to 51
  end if
  if rollover(17) then
    set cnum to the castnum of sprite 17
    if the name of cast cnum = "Button_next_up" then set mess_num to 22
    else if the name of cast cnum = "Button_nonext" then set mess_num to 50
  end if
  if rollover(21) then set mess_num to 46
  repeat with n = 1 to 3
    if rollover(n+21) and the visible of sprite n then set mess_num to 47
  end repeat
  if rollover(18) then
    set cnum to the castnum of sprite 18
    if the name of cast cnum = "Button_hide_sw_name_up" then set mess_num to 60
    else set mess_num to 52
  end if
  if rollover(19) then set mess_num to 53
  if rollover(35) then
    set cnum to the castnum of sprite 35
    if the name of cast cnum = "Button_CV_up" then set mess_num to 17
    else set mess_num to 26
  end if
  if rollover(36) then set mess_num to 16

  return mess_num

end
```

347 : Change view mi_band

```
on mousedown
  global gview_mi

  Change_view(gview_mi,3)
  set gview_mi to the result
end
```

348 : Change view ne_corner

```
on mousedown
  global gview_ne

  Change_view(gview_ne,6)

  set gview_ne to the result
end
```

349 : Change view pl_band

```
on mousedown
  global gview_pl

  Change_view(gview_pl,3)
  set gview_pl to the result
end
```

350 : Change view s_band

```
on mousedown
  global gview_bs

  Change_view(gview_bs,2)
  set gview_bs to the result
end
```

351 : Click button back to main map

```
on mousedown

  click_button("Button_mmap",36)

  if the result then
    put " " into cast "Flower name latin"
    put " " into cast "Flower name swedish"
    set the visibility of sprite 33 to 1
    puppetsprite 10, false
    puppetsprite 13, false
    puppetsprite 18, false
    puppetsprite 24, false
    puppetsprite 22, false
    puppetsprite 23, false
    deselect_flower("Mask_frame")
    reset_frames
    close_frames
    set the visibility of sprite 11 to 1
    set the visibility of sprite 12 to 1
    set the visibility of sprite 13 to 1
    go to "start"
  end if

end
```

352 : Click button big ok music

```
on mousedown
  global gmusic_volume, gpos_curs_mus, gmusic_volume_prev,
      gpos_curs_mus_prev

  click_button("Button_ok_big",47)

  if the result then
    set gpos_curs_mus_prev to gpos_curs_mus
    set gmusic_volume_prev to gmusic_volume
    puppetsprite 45, false

    play done
  end if

end
```

353 : Click button big ok sound

```
on mousedown
  global gsound_volume, gpos_curs_sound, gsound_volume_prev,
      gpos_curs_sound_prev

  click_button("Button_ok_big",47)

  if the result then
    set gpos_curs_sound_prev to gpos_curs_sound
    set gsound_volume_prev to gsound_volume
    puppetsprite 45, false

    play done

  end if

end
```

354 : Click button blossoming

```
on mousedown
  click_button("Button_bloss",35)
  if the result then
```

```
    repeat with n = 21 to 26
      set the Visible of sprite n to false
    end repeat
    repeat with n = 7 to 20
      set the visibility of sprite n to 1
    end repeat

    go to frame "Blossoming"
  end if

end
```

355 : Click button cancel music

```
on mousedown
  global gmusic_volume, gpos_curs_mus, gmusic_volume_prev,
      gpos_curs_mus_prev

  click_button("Button_cancel",48)

  if the result then

    set the visibility of sprite 37 to 1

    set gpos_curs_mus to gpos_curs_mus_prev
    set gmusic_volume to gmusic_volume_prev

    set the volume of sound 2 to gmusic_volume

    puppetsprite 45, false

    play done
  end if

end
```

356 : Click button cancel sound

```
on mousedown
  global gsound_volume, gpos_curs_sound, gsound_volume_prev,
      gpos_curs_sound_prev

  click_button("Button_cancel",48)

  if the result then

    set the visibility of sprite 37 to 1

    set gpos_curs_sound to gpos_curs_sound_prev
    set gsound_volume to gsound_volume_prev

    set the volume of sound 3 to gsound_volume
    puppetsprite 45, false

    play done

  end if

end
```

357 : Click button change view

```
on Change_view view, number

  click_button("Button_CV",35)

  if the result then
    put view + 1 into view
```

```
    if view > number then set view to 1
    repeat with n = 1 to number
      if (n = view) then set the visibility of sprite n+3 to 1
      else set the visibility of sprite n+3 to 0
    end repeat
  end if
  return view
end
```

358 : Click button close

```
on Back_to part
  global gflowercast, gfselect

  click_button("Button_close",12)

  if the result then
    put " " into cast "Flower name latin"
    put " " into cast "Flower name swedish"
    set the visibility of sprite 33 to 1
    puppetsprite 10, false
    puppetsprite 13, false
    puppetsprite 18, false
    puppetsprite 24, false
    puppetsprite 22, false
    puppetsprite 23, false
    deselect_flower("Mask_frame")
    reset_frames
    close_frames
    go to part
  end if
end
```

359 : Click button display latin name

```
on mousedown
  global gflowername_lat, gflowername_sw

  if gflowername_lat <> "" then set noname to 0
  else set noname to 1

  set cnum to the castnum of sprite 13
  if the name of cast cnum = "Button_disp_lat_name_up" then
    switch_button("Button_disp_lat_name","Button_hide_lat_name",noname,13)
    set hide to 0
  else
    switch_button("Button_hide_lat_name","Button_disp_lat_name",noname,13)
    set hide to 1
  end if

  if the result then
    if not noname then
      if not hide then
        put gflowername_lat into cast "Flower name latin"
      else
        put " " into cast "Flower name latin"
      end if
    else
      -- warning sound --
      warning
      -- management of the error window --
      error_win_man("Noselect_error")
    end if
  end if
end
```

360 : Click button display swedish name

```
on mousedown
```

```
  global gflowername_lat, gflowername_sw

  if gflowername_lat <> "" then set noname to 0
  else set noname to 1

  set cnum to the castnum of sprite 18
  if the name of cast cnum = "Button_disp_sw_name_up" then
    switch_button("Button_disp_sw_name","Button_hide_sw_name",noname,18)
    set hide to 0
  else
    switch_button("Button_hide_sw_name","Button_disp_sw_name",noname,18)
    set hide to 1
  end if

  if the result then
    if not noname then
      if not hide then
        put gflowername_sw into cast "Flower name swedish"
      else
        put " " into cast "Flower name swedish"
      end if
    else
      warning
      error_win_man("Noselect_error")
    end if
  end if
end
```

361 : Click button info

```
on Info part
  global gflowers_desc, gflowername_lat, gflowername_sw

  click_button("Button_info",15)
  if the result then
    if gflowername_lat <> "" then
      put "" into cast "Description"
      set desc to
       getat(gflowers_desc,findpos(gflowers_desc,gflowername_lat))
      set lines to line 2 of desc
      put "Luminosity : "&lines into line 1 of cast "Description"
      set lines to line 3 of desc
      put "Size : "&lines&" cm" into line 3 of cast "Description"
      set lines to line 4 of desc
      decode(lines)
      put "Blossoming : "&the result into line 5 of cast "Description"
      put "" into cast "Desc Headlines"
      put gflowername_lat into line 1 of cast "Desc Headlines"
      put "("&gflowername_sw&")" into line 2 of cast "Desc Headlines"
      reset_frames

      spritebox 10, the left of sprite 11 + 10, the top of sprite 11 + 10,¬
                  the left of sprite 11 + 176, the top of sprite 11 + 137

      go to frame part
    else
      warning
      error_win_man("Noselect_error")
    end if
  end if
end
```

362 : Click button map of the garden

```
on mousedown
  global gBusy_spots_list, gHighlights_list, gFree_spots_list

  click_button("Button_map",35)
  if the result then
    repeat with n = 1 to 26
```

```
      puppetsprite n, false
    end repeat

    deselect_flower("Mask_frame")
    reset_frames
    close_frames

    set gBusy_spots_list = list (0,0,0,0,0,0,0,0,0,0,0,0,0)
    set gHighlights_list = list (0,0,0,0,0,0,0,0,0,0,0,0)
    set gFree_spots_list = list (0,0,0,0,0,0,0,0,0,0,0,0,0)

    go to frame "Start"
  end if
end
```

363 : Click Button mmap

```
on mousedown

  click_button("Button_mmap",36)

  if the result then

    set the visibility of sprite 11 to 1
    set the visibility of sprite 12 to 1
    go to "start"
  end if

end
```

364 : Click button Next

```
on Next_area gcoord_part, gcpos_part
  global gflowers_num, gflowercast, gnum_fr, gflowername_lat,    gflowername_sw

  click_button("Button_next",17)

  if the result then
    set gflowers_num to gflowers_num +1
    if gflowers_num > count(gcpos_part) then set gflowers_num = 1

    set gflowercast to getpropat(gcpos_part, gflowers_num)

    set the castnum of sprite 10 to the number of cast gflowercast
    spritebox 10, the left of sprite 11 + 10, the top of sprite 11 + 9,¬
                  the right of sprite 11 - 10, the bottom of sprite 11 - 22

    if gflowercast="Fl3/1" then
      set nf to 3
    else
      if gflowercast="Fl3/9" then
        set nf to 2
      else
        if gflowercast="Fl4/6" then
          set nf to 2
        else
          set nf to 1
        end if
      end if
    end if
    pointer_map(gcoord_part,gcpos_part,gflowercast,nf)

    put "" into cast "Flower name latin"
    put "" into gflowername_lat
    put "" into cast "Flower name swedish"
    put "" into gflowername_sw

    deselect_flower("Mask_frame")
    reset_frames
    close_frames
```

```
    set_frames(gflowercast)

    reset_buttons_disp

    if gnum_fr=0 then
      set gflowername_lat to
            getat(gflowers_list,findpos(gflowers_list,gflowercast))
      set desc to
      getat(gflowers_desc,findpos(gflowers_desc,gflowername_lat))
      set gflowername_sw to line 1 of desc
      put 1 into cast "Number of flowers"
    else
      put gnum_fr into cast "Number of flowers"
      set gflowername_lat to ""
      set gflowername_sw to ""
    end if

    updatestage
  end if
end
```

365 : Click button next in frames "Seasons_..."

```
on mousedown
  global gflowercast, gflowername_lat, gflowername_sw, gcurrent_flower,
      gseason, gframe, gflower_num, gmonths_list

  if gflower_num < total(gmonths_list) then

    click_button("Button_next",17)

    if the result then

      reset_frames
      close_frames

      next_flower

      set gframe to
            getpropat(gflowers_list,getpos(gflowers_list,gcurrent_flower))
      rem_tail(gframe)
      set gflowercast to the result

      set gflower_num to gflower_num + 1
      if gflower_num = total(gmonths_list) then
        set the castnum of sprite 17 to the number of cast "Button_nonext"
      else
        set the castnum of sprite 17 to the number of cast "Button_next_up"
      end if
      set the castnum of sprite 16 to the number of cast "Button_prev_up"

      set_frame(gframe)

      set the castnum of sprite 10 to the number of cast gflowercast
      spritebox 10, the left of sprite 11 + 10, the top of sprite 11 + 9,¬
                  the right of sprite 11 - 10, the bottom of sprite 11 - 22


      put "" into cast "Flower name latin"
      put "" into gflowername_lat
      put "" into cast "Flower name swedish"
      put "" into gflowername_sw

      reset_buttons_disp

      set gflowername_lat to gcurrent_flower
      set desc to
      getat(gflowers_desc,findpos(gflowers_desc,gflowername_lat))
      set gflowername_sw to line 1 of desc
```

```
      pointer_mmap(gflowername_lat)

      test_seasons
      if the result then go to frame label(gseason)+1

    end if
  end if

end
```

366 : Click button no in the confirmation window

```
on mousedown

  click_button("Button_no",48)

  if the result then

    set the visibility of sprite 37 to 1
    play done
  end if

end
```

367 : Click button Ok

```
on mousedown

  click_button("Button_ok",47)

  if the result then

    set the visibility of sprite 37 to 1
    play done
  end if

end
```

368 : Click button Pic

```
on mousedown
  global gflowercast

  click_button("Button_pic",15)
  if the result then
    spritebox 10, the left of sprite 11 + 10, the top of sprite 11 + 9,¬
                the right of sprite 11 - 10, the bottom of sprite 11 - 22
    set_frames(gflowercast)
    go to the frame - 3
  end if

end
```

369 : Click button Pic in the frames "Seasons"

```
on mousedown
  global gframe

  click_button("Button_pic",15)
  if the result then
    spritebox 10, the left of sprite 11 + 10, the top of sprite 11 + 9,¬
                the right of sprite 11 - 10, the bottom of sprite 11 - 22
    set_frame(gframe)

    go to the frame - 3
  end if
```

```
end
```

370 : Click button Previous

```
on Prev_area coord_part, cpos_part
  global gflowers_num, gflowercast, gnum_fr, gflowername_lat,    gflowername_sw

  click_button("Button_prev",16)

  if the result then
    set gflowers_num to gflowers_num -1
    if gflowers_num < 1 then set gflowers_num = count(cpos_part)

    set gflowercast to getpropat(cpos_part, gflowers_num)
    set the castnum of sprite 10 to the number of cast gflowercast
    spritebox 10, the left of sprite 11 + 10, the top of sprite 11 + 9,¬
                  the right of sprite 11 - 10, the bottom of sprite 11 - 22

    if gflowercast="Fl3/1" then
      set nf to 3
    else
      if gflowercast="Fl3/9" then
        set nf to 2
      else
        if gflowercast="Fl4/6" then
          set nf to 2
        else
          set nf to 1
        end if
      end if
    end if

    pointer_map(coord_part,cpos_part,gflowercast,nf)

    put "" into cast "Flower name latin"
    put "" into gflowername_lat
    put "" into cast "Flower name swedish"
    put "" into gflowername_sw

    deselect_flower("Mask_frame")
    reset_frames
    close_frames

    set_frames(gflowercast)

    reset_buttons_disp

    if gnum_fr=0 then
      set gflowername_lat to
            getat(gflowers_list,findpos(gflowers_list,gflowercast))
      set desc to
       getat(gflowers_desc,findpos(gflowers_desc,gflowername_lat))
      set gflowername_sw to line 1 of desc
      put 1 into cast "Number of flowers"
    else
      put gnum_fr into cast "Number of flowers"
      set gflowername_lat to ""
      set gflowername_sw to ""
    end if

    updatestage
  end if
end
```

371 : Click button previous in frames "Seasons_..."

```
on mousedown
  global gflowercast, gflowername_lat, gflowername_sw, gcurrent_flower,
       gseason, gframe, gflower_num, gmonths_list
```

```
  if gflower_num > 1 then
    click_button("Button_prev",16)

    if the result then

      reset_frames
      close_frames

      previous_flower

      set gframe to
            getpropat(gflowers_list,getpos(gflowers_list,gcurrent_flower))
      rem_tail(gframe)
      set gflowercast to the result

      set gflower_num to gflower_num - 1
      if gflower_num = 1 then
        set the castnum of sprite 16 to the number of cast "Button_noprev"
      else
        set the castnum of sprite 16 to the number of cast "Button_prev_up"
      end if
      set the castnum of sprite 17 to the number of cast "Button_next_up"

      set_frame(gframe)

      set the castnum of sprite 10 to the number of cast gflowercast
      spritebox 10, the left of sprite 11 + 10, the top of sprite 11 + 9,¬
                  the right of sprite 11 - 10, the bottom of sprite 11 - 22


      put "" into cast "Flower name latin"
      put "" into gflowername_lat
      put "" into cast "Flower name swedish"
      put "" into gflowername_sw

      reset_buttons_disp

      set gflowername_lat to gcurrent_flower
      set desc to
       getat(gflowers_desc,findpos(gflowers_desc,gflowername_lat))
      set gflowername_sw to line 1 of desc

      pointer_mmap(gflowername_lat)

      test_seasons
      if the result then go to frame label(gseason)+1

    end if
  end if

end
```

372 : Click button reset

```
on mousedown
  global gBusy_spots_list, gHighlights_list, gFree_spots_list

  click_button("Button_reset",3)
  if the result then
    set gBusy_spots_list = list (0,0,0,0,0,0,0,0,0,0,0,0,0,0)
    set gHighlights_list = list (0,0,0,0,0,0,0,0,0,0,0,0,0,0)
    set gFree_spots_list = list (0,0,0,0,0,0,0,0,0,0,0,0,0,0)

    repeat with n=7 to 20
      if n <14 then
        set the loch of sprite n to the loch of sprite 13
        set the locv of sprite n to the locv of sprite 13
      else
        set the loch of sprite n to the loch of sprite 20
        set the locv of sprite n to the locv of sprite 20
      end if
```

```
    end repeat

    repeat with n = 21 to 26
      set the Visible of sprite n to false
    end repeat

  end if
end
```

373 : Click button Search

```
on mousedown
  global gmonths_list, compteur, Empty_scale, gpos_months_list,
       gpos_current_list, gJanuary_list, gFebruary_list,  gMarch_list,gApril_list,
gMay_list , gseason,¬
       gJune_list, gJuly_list, gAugustus_list, gSeptember_list,  gOctober_list,
gNovember_list, gDecember_list, gcurrent_list,           gBusy_spots_list,
gHighlights_list, gFree_spots_list

  click_button("Button_search",4)

  if the result then
    set Empty_scale to true
    calccompteur

    if (compteur <> 0) then
      warning
      error_win_man("Notequal_error")
    else
      if (Empty_scale = true) then
        warning
        error_win_man("Empty_scale_error")
      else
        user_choice

        set gpos_months_list to 0
        set gcurrent_list to []

        next_month

        if count(gcurrent_list) <> 0 then

          set gpos_current_list to 0
          next_flower

          repeat with n = 1 to 26
            puppetsprite n, false
          end repeat

          set gBusy_spots_list = list (0,0,0,0,0,0,0,0,0,0,0,0,0,0)
          set gHighlights_list = list (0,0,0,0,0,0,0,0,0,0,0,0,0,0)
          set gFree_spots_list = list (0,0,0,0,0,0,0,0,0,0,0,0,0,0)

          test_seasons
          go to frame gseason
        else
          warning
          error_win_man("Empty_list_error")
        end if
      end if
    end if
  end if
end
```

374 : Click button speak latin name

```
on mousedown
  global gflowername_lat, gpathname, gmusic_volume, gsound_volume

  click_button("Button_speak_lat_name",14)
```

```
  if the result then
    if gflowername_lat <> "" then
      set filename to ""
      set n to 1
      set end to false
      repeat while not end
        set ch to char n of gflowername_lat
        if ch<>" " then
          set filename to filename&ch
          if length(filename)=10 then set end to true
        end if
        set n to n+1
      end repeat

      set the volume of sound 2 to 0
      set the volume of sound 3 to 400

      sound playfile 3, gpathname&"Voices:"&filename&"_la"
      repeat while not soundbusy(3)
        NOTHING
      end repeat
      repeat while soundbusy(3)
        NOTHING
      end repeat

      set the volume of sound 2 to gmusic_volume
      set the volume of sound 3 to gsound_volume

    else
      -- warning sound --
      warning
      -- management of the error window --
      error_win_man("Noselect_error")
    end if
  end if
end
```

375 : Click button speak swedish name

```
on mousedown
  global gflowername_lat, gpathname, gmusic_volume, gsound_volume

  click_button("Button_speak_sw_name",19)

  if the result then
    if gflowername_lat <> "" then
      set filename to ""
      set n to 1
      set end to false
      repeat while not end
        set ch to char n of gflowername_lat
        if ch<>" " then
          set filename to filename&ch
          if length(filename)=10 then set end to true
        end if
        set n to n+1
      end repeat

      set the volume of sound 2 to 0
      set the volume of sound 3 to 400

      sound playfile 3, gpathname&"Voices:"&filename&"_se"
      repeat while not soundbusy(3)
        NOTHING
      end repeat
      repeat while soundbusy(3)
        NOTHING
      end repeat

      set the volume of sound 2 to gmusic_volume
```

```
      set the volume of sound 3 to gsound_volume

    else
      warning
      error_win_man("Noselect_error")
    end if
  end if
end
```

376 : Click button yes in the confiramtion window

```
on mousedown

  click_button("Button_yes",47)

  if the result then quit
end
```

377 : Click down sound

```
on click_1
  puppetsound "c1"
  updatestage
end
```

378 : Click part management

```
on click_part_management start_ch, end_ch, number, part_name_list,     cpos_part,
partname

  set total to 0
  if end_ch <> 0 then
    repeat with n = start_ch to end_ch
      set total to total + rollover(n)
    end repeat
  end if

  if rollover(29) and (total=0) then
    puppetsprite 48, true
    menu_sprite ("")
    set the loch of sprite 48 to the mouseh
    set the locv of sprite 48 to the mousev
    cursor 0
    set the visibility of sprite 48 to 1
    menu_cursor (the loch of sprite 48, the locv of sprite 48)
    menu_choice (the result)

  else
    set n to 1
    set choice to 0
    repeat while n<number+1 and choice = 0
      if rollover(n+10) then
        set choice to n
        set part_sprite to getat(part_name_list,n)
        click_part_sprite("Mask_" & part_sprite,10+n)
        if the result then
          puppetsprite 39 ,false
          set part_sprite to getat(part_name_list,n)
          set the visibility of sprite 38 to 0
          set the visibility of sprite 39 to 0
          map_button (partname,part_sprite,cpos_part)
        end if
      end if
      set n to n+1
    end repeat

  end if
end
```

379 : Click up sound

```
on click_2
  puppetsound "c2"
  updatestage
end
```

380 : click_button

```
on click_button button_name, channel

  -- mouse down sound --
  click_1

  -- place the button ( necessary because of a bug from director --
  set name to button_name&"_dw"
  set c1 to the left of sprite channel
  set c2 to the top of sprite channel
  set c3 to the right of sprite channel
  set c4 to the bottom of sprite channel
  set the castnum of sprite channel to the number of cast name
  spritebox channel, c1, c2, c3, c4
  updatestage

  -- test if the button is really pressed by the user, it means that the
      user has the --
  -- possinility to change his mind by releasing the mouse button in
      another area than the --
  -- one occupied by the button --

  set ok to true
  repeat while the mousedown
    if rollover(channel) then
      set ok to true
      set name to button_name&"_dw"
      set the castnum of sprite channel to the number of cast name
      spritebox channel, c1, c2, c3, c4
      updatestage

    else
      set ok to false
      set name to button_name&"_up"
      set the castnum of sprite channel to the number of cast name
      spritebox channel, c1, c2, c3, c4
      updatestage
    end if
  end repeat

  if ok then
    -- mouse up sound --
    click_2

    set name to button_name&"_up"
    set the castnum of sprite channel to the number of cast name
    spritebox channel, c1, c2, c3, c4
    cursor 4
    updatestage
    set complete to true
  else
    set complete to false
  end if

  -- complete equals 1 if the button is fully pressed, equals 0 otherwise --
  return complete

end
```

381 : click_part

```
on click_part_sprite part_name, channel

  -- mouse down sound --
  click_1

  set name to part_name&"_dw"
  set the castnum of sprite channel to the number of cast name
  set the blend of sprite channel to 80
  updatestage

  -- test if the part is really clicked by the user, it means that the user
      has the --
  -- possinility to change his mind by releasing the mouse button in
      another area than the --
  -- one occupied by the part --

  repeat while the mousedown
    if rollover(channel) then
      set name to part_name&"_dw"
      set the castnum of sprite channel to the number of cast name
      set the blend of sprite channel to 80
      updatestage

    else
      set name to part_name&"_up"
      set the castnum of sprite channel to the number of cast name
      set the blend of sprite channel to 38
      updatestage
    end if
  end repeat

  if rollover(channel) then
    -- mouse up sound --
    click_2

    set name to part_name&"_up"
    set the castnum of sprite channel to the number of cast name
    set the blend of sprite channel to 38
    cursor 4
    updatestage
    set complete to true
  else
    set complete to false
  end if

  -- complete equals 1 if the part is fully clicked, equals 0 otherwise --
  return complete
end
```

382 : Convert month_name to month_list

```
on convert_month month
  global gJanuary_list, gFebruary_list, gMarch_list,gApril_list, gMay_list    ,
gJune_list, gJuly_list, gAugustus_list, gSeptember_list,  gOctober_list,
gNovember_list, gDecember_list

  if month="January" then
    set result to gjanuary_list
  else
    if month="February" then
      set result to gFebruary_list
    else
      if month="March" then
        set result to gMarch_list
      else
        if month="April" then
          set result to gApril_list
        else
          if month="May" then
```

```
          set result to gMay_list
        else
          if month="June" then
            set result to gJune_list
          else
            if month="July" then
              set result to gJuly_list
            else
              if month="Augustus" then
                set result to gAugustus_list
              else
                if month="September" then
                  set result to gSeptember_list
                else
                  if month="October" then
                    set result to gOctober_list
                  else
                    if month="November" then
                      set result to gNovember_list
                    else
                      set result to gDecember_list
                    end if
                  end if
                end if
              end if
            end if
          end if
        end if
      end if
    end if
  end if

  return result

end
```

383 : Cursor management

```
on cursor_management type
  global gclosed_hand, gmusic_volume, gpos_curs_mus, gsound_volume,
       gpos_curs_sound

  cursor gclosed_hand

  repeat while the mousedown
    if the mousev < the top of sprite 44 + 49 then
      set the locv of sprite 45 to the top of sprite 44 + 49
    else
      if the mousev > the top of sprite 44 + 133 then
        set the locv of sprite 45 to the top of sprite 44 + 133
      else
        set the locv of sprite 45 to the mousev
      end if
    end if
    updatestage
  end repeat

  set max to the top of sprite 44 + 49
  set lapse to 12

  if the locv of sprite 45 <= max + lapse/2 then
    set the locv of sprite 45 to max
    set volume to 7
  else
    if the locv of sprite 45 <= max + lapse then
```

```
      set the locv of sprite 45 to max + lapse
      set volume to 6
    else
      if the locv of sprite 45 <= max + 3*lapse/2 then
        set the locv of sprite 45 to max + lapse
        set volume to 6
      else
        if the locv of sprite 45 <= max + 2*lapse then
          set the locv of sprite 45 to max + 2*lapse
          set volume to 5
        else
          if the locv of sprite 45 <= max + 5*lapse/2 then
            set the locv of sprite 45 to max + 2*lapse
            set volume to 5
          else
            if the locv of sprite 45 <= max + 3*lapse then
              set the locv of sprite 45 to max + 3*lapse
              set volume to 4
            else
              if the locv of sprite 45 <= max + 7*lapse/2 then
                set the locv of sprite 45 to max + 3*lapse
                set volume to 4
              else
                if the locv of sprite 45 <= max + 4*lapse then
                  set the locv of sprite 45 to max + 4*lapse
                  set volume to 3
                else
                  if the locv of sprite 45 <= max + 9*lapse/2 then
                    set the locv of sprite 45 to max + 4*lapse
                    set volume to 3
                  else
                    if the locv of sprite 45 <= max + 5*lapse then
                      set the locv of sprite 45 to max + 5*lapse
                      set volume to 2
                    else
                      if the locv of sprite 45 <= max + 11*lapse/2 then
                        set the locv of sprite 45 to max + 5*lapse
                        set volume to 2
                      else
                        if the locv of sprite 45 <= max + 6*lapse then
                          set the locv of sprite 45 to max + 6*lapse
                          set volume to 1
                        else
                          if the locv of sprite 45 <= max + 13*lapse/2 then
                            set the locv of sprite 45 to max + 6*lapse
                            set volume to 1
                          else
                            if the locv of sprite 45 <= max + 7*lapse then
                              set the locv of sprite 45 to max + 7*lapse
                              set volume to 0
                            end if
                          end if
                        end if
                      end if
                    end if
                  end if
                end if
              end if
            end if
          end if
        end if
      end if
    end if
  end if
end if
if type = "Music" then
  set gpos_curs_mus to volume - 7
  if gpos_curs_mus < 0 then set gpos_curs_mus to - gpos_curs_mus
  set gmusic_volume to 20*volume
  set the volume of sound 2 to gmusic_volume
else
  set gpos_curs_sound to volume - 7
  if gpos_curs_sound < 0 then set gpos_curs_sound to - gpos_curs_sound
```

```
    set gsound_volume to 20*volume
    set the volume of sound 3 to gsound_volume
  end if
end
```

384 : Error windows

```
on enterframe
  global ghelp

  test_sound
  cursor -1
  if rollover(47) then set mess_num to 28
  else set mess_num to 1

  if ghelp then put getat(gmessage_list,mess_num) into cast "Help-message"
end

on exitFrame
  go to the frame
end
```

385 : Error windows management

```
on error_win_man name
  global ghelp

  set v to [0,0,0]
  repeat with n=1 to 48
    if n<46 then
      if n<>37 then
        setat gpuppet_list, n, the puppet of sprite n
        puppetsprite n, true
      end if
    else
      setat v, n-45 , the visibility of sprite n
      set the visibility of sprite n to 1
    end if
  end repeat

  play frame name

  repeat with n=1 to 48
    if n<46 then
      puppetsprite n, getat (gpuppet_list, n)
    else
      set the visibility of sprite n to getat (v,n-45)
    end if
  end repeat
end
```

386 : File_io

```
-- low level object oriented procedures for the management of a text file --
on openfile filename
  global greadobject

  put fileio(mnew,"read",the pathname & filename) into greadobject

end
on readfile
  global greadobject

  put greadobject(mreadline) into text

  return text
end
on closefile
```

```
  global greadobject

  greadobject(mdispose)
end
```

387 : Flags management

```
on Start_Flag gblue

  global gBusy_spots_list, Compteur, Notgood

  set i to 1
  set Trouve to false
  set Notgood to false

  repeat while (i < 14) and ( Trouve = false )
    if ( GetAt ( gBusy_spots_list , i ) = gblue ) then
      SetAt ( gBusy_spots_list , i , 0 )
      set Trouve to true
      set j to i
    else
      set i to i+1
    end if
  end repeat

  CALCCOMPTEUR

  if  (Compteur < 1 ) and ( Compteur > -7 ) then

    DESURLIGNEMENT j
    SURLIGNEMENT

    repeat while the mouseDown
      set the locV of sprite gblue to the mouseV
      set the locH of sprite gblue to the mouseH
      updatestage
    end repeat

  else
    set Notgood to true
    Positionnement(gblue)
    warning
    error_win_man("BF_error")
  end if

end

on End_flag ggreen

  global gBusy_spots_list, Compteur, Notgood, ghelp

  set i to 1
  set Trouve to false
  set Notgood to false

  repeat while (i < 14) and ( Trouve = false )
    if ( GetAt ( gBusy_spots_list , i ) = ggreen ) then
      SetAt ( gBusy_spots_list , i , 0 )
      set Trouve to true
      set j to i
    else
      set i to i+1
    end if
  end repeat

  CALCCOMPTEUR

  if  ( Compteur < 7 ) and ( Compteur > 0-trouve ) then

    DESURLIGNEMENT j
    SURLIGNEMENT
```

```
    repeat while the mouseDown
      set the locV of sprite ggreen to the mouseV
      set the loch of sprite ggreen to the mouseH
      updatestage
    end repeat

  else
    set Notgood to true
    Positionnement(ggreen)
    warning
    error_win_man("GF_error")
  end if

end
```

388 : Flags positionning

```
on POSITIONNEMENT Vari

  Global Ok , gBusy_spots_list , Find , LL , LR

  set Ok to false
  set Find to false
  if ( the bottom of sprite Vari < the top of sprite 6 - 29 ) or ¬
     ( the bottom of sprite Vari > the bottom of sprite 6 + 29 ) or ¬
     ( sprite_coord(vari) < the left of sprite 6 - 6 ) or ¬
     ( sprite_coord(vari) > the right of sprite 6 + 7 ) then
    REPOSITIONNEMENT(Vari)
  else

    set Num to 1


    repeat while ( Find = false ) and ( Num < 14 )
      TERE Vari, Num
      set Num to Num+1
    end repeat

    TROUVEDRAP Vari
    SURLIGNEMENT
  end if

end

on REPOSITIONNEMENT Vari  -- Place the flag to its initial position.

  if  ( Vari < 14 ) then
    set coord1 to the locv of sprite 13
    set coord2 to the loch of sprite 13
  else
    set coord1 to the locv of sprite 20
    set coord2 to the loch of sprite 20
  end if

  if ( the locV of sprite Vari > coord1 ) then

    set j to ( the locV of sprite Vari )
    set k to ( the locH of sprite Vari )
    set the locV of sprite Vari to the locV of sprite Vari -1

    repeat while ( the locV of sprite Vari > coord1 )
      set x to (the locV of sprite Vari-coord1) * (k - coord2) + (coord2) *
      ( j - coord1)
      set x to x / ( j - coord1 )

      set the locH of sprite Vari to x
      updatestage
      set the locV of sprite Vari to the locV of sprite Vari -10
    end repeat
```

```
    set the locH of sprite Vari to coord2
    set the locV of sprite Vari to coord1

  else
    if ( the locV of sprite Vari = coord1 ) then

      set j to ( the locV of sprite Vari )
      set k to ( the locH of sprite Vari )
      set the locH of sprite Vari to the locH of sprite Vari -1

      repeat while ( the locH of sprite Vari  > coord2 )
        set y to (the locH of sprite Vari-coord2) * (j - coord1) + (coord1)
       * ( k - coord2)
        set y to y / ( k - coord2 )

        set the locV of sprite Vari to y
        updatestage
        set the locH of sprite Vari to the locH of sprite Vari -10
      end repeat

      set the locH of sprite Vari to coord2
      set the locV of sprite Vari to coord1

    else
      set j to ( the locV of sprite Vari )
      set k to ( the locH of sprite Vari )
      set the locV of sprite Vari to the locV of sprite Vari +1

      repeat while ( the locV of sprite Vari < coord1 )
        set x to (the locV of sprite Vari-coord1) * (k - coord2) + (coord2)
       * ( j - coord1)
        set x to x / ( j - coord1 )

        set the locH of sprite Vari to x
        updatestage
        set the locV of sprite Vari to the locV of sprite Vari +10
      end repeat

      set the locH of sprite Vari to coord2
      set the locV of sprite Vari to coord1

    end if
  end if

end

on TEST Vari , Number
  global Ok , gFree_spots_list

  set Ok to false

  if (GetAt(gBusy_spots_list, Number) <> Vari) then

    POSLIBRE Vari
    if vari<14 then
      if number = 13 then
        warning
        error_win_man("SF_lastspot_error")
        set ok to true
        setat gfree_spots_list , Number, 0
      end if
    end if

    if ( GetAt ( gFree_spots_list , Number ) <> 0 ) then
      set Ok to true

      if ( Vari > 13 ) then
        if (GetAt(gBusy_spots_list, Number) < 14) and
       (GetAt(gBusy_spots_list, Number) <> 0 ) then
          set j to ( GetAt ( gBusy_spots_list , Number ) )
          REPOSITIONNEMENT j
          SetAt ( gBusy_spots_list , Number , 0 )
```

```
          DESURLIGNEMENT Number
        end if

      else
        if ( GetAt ( gBusy_spots_list , Number ) > 13 ) then
          set j to ( GetAt ( gBusy_spots_list , Number ) )
          REPOSITIONNEMENT j
          SetAt ( gBusy_spots_list , Number , 0 )
          DESURLIGNEMENT Number

        end if
      end if
    end if
  end if
end

on TERE Vari, Num
  global OK, Find, Notgood , gScale_Coord


  if (((the left of sprite 6) + (GetAt(gScale_Coord,Num)) -
       (sprite_coord(vari))) < 25 ) ¬
       and (((the left of sprite 6) + (GetAt(gScale_Coord,Num)) -
       (sprite_coord(vari))) > -24 ) then

    if Notgood = false then
      TEST Vari , Num
    end if

    if ( Ok = true ) then
      REPOSITIONNEMENT Vari
    else

      set base to sprite_coord(vari) - the left of sprite vari
      set height to (the bottom of sprite vari - the top of sprite vari)
      set length to (the right of sprite vari - the left of sprite vari)

      spriteBox Vari,GetAt(gScale_Coord,Num) + the left of sprite 6 -
       base, the top of sprite 6 - (Height-3),¬
       GetAt(gScale_Coord,Num) + the left of sprite 6 + (Length -
       base), the top of sprite 6 + 3

      SetAt ( gBusy_spots_list , Num , Vari )
    end if
    set Find to true
  end if

end
```

389 : Frame "Blossoming"

```
on enterframe
  global ghelp, gmenu, gsound, gmusic, gmessage_list, gopened_hand,
       gclosed_hand, gvolume_cursor_music, gvolume_cursor_sound

  test_music
  test_sound
  set mess_num to 1

  -- cheese menu zone rollover test --
  repeat with n = 7 to 20
    if rollover(n) then
      set rf to true
      set num to n
    end if
  end repeat

  if rollover(29) then
    if rollover(2) then
      if rf then
```

```
        cursor gopened_hand
        if num <= 13 then set mess_num to 27
        else set mess_num to 36
      else
        if rollover(3) then
          set mess_num to 31
          cursor 0
        else
          if rollover(4) then
            set mess_num to 32
            cursor 0
          else
            cursor 0
          end if
        end if
      end if
    else
      if rf then
        cursor gopened_hand
        if num <= 13 then set mess_num to 27
        else set mess_num to 36
      else
        if rollover(6) then
          cursor 0
          set rb to 0
          repeat with n=21 to 26
            if rollover(n) then set rb to n
          end repeat
          if rb<>0 and the visible of sprite rb then set mess_num to 49
          else set mess_num to 33
        else
          cursor gmenu
          set mess_num to 2
        end if
      end if
    end if
  else
    if rf then
      cursor gopened_hand
      if num <= 13 then set mess_num to 27
      else set mess_num to 36
    else
      if rollover(27) and gsound then
        cursor gvolume_cursor_sound
        set mess_num to 6
      else
        if rollover(28) and gmusic then
          cursor gvolume_cursor_music
          set mess_num to 7
        else
          cursor 0
        end if
      end if
    end if
  end if

  -- icons&Buttons rollover tests --
  if rollover(30) then set mess_num to 4
  if rollover(35) then set mess_num to 16

  if ghelp=1 then put getat(gmessage_list,mess_num) into cast "Help-   message"

end

on exitFrame
  go to the frame
end

on mousedown
  global gblue, ggreen, gclosed_hand

  set total to 0
```

```
  repeat with n = 2 to 6
    if n<>5 then set total to total + rollover(n)
  end repeat

  set end to false
  set n to 7
  repeat while n <= 20 and not end
    set total to total + rollover(n)
    if rollover(n) then
      cursor gclosed_hand
      set end to true
      if n<=13 then
        if n<>13 then
          set gblue to n
          SFlag_management
        else
          warning
          error_win_man("Full_error")
        end if
      else
        if n <> 20 then
          set ggreen to n
          EFlag_management
        else
          warning
          error_win_man("Full_error")
        end if
      end if
    end if
    set n to n+1
  end repeat

  if rollover(29) and (total=0) then
    puppetsprite 48, true
    menu_sprite ("")
    set the loch of sprite 48 to the mouseh
    set the locv of sprite 48 to the mousev
    cursor 0
    set the visibility of sprite 48 to 1
    menu_cursor (the loch of sprite 48, the locv of sprite 48)
    menu_choice (the result)
  end if
end
```

390 : Frame "NE_corner"

```
on enterframe
  global gnecorner_name_list

  part_management(gnecorner_name_list,11,16)
end

on exitFrame
  go to the frame
end

on mousedown
  global gnecorner_name_list, gcpos_necorner

  click_part_management(11,16,6,gnecorner_name_list,gcpos_necorner,"SF_ne")

end
```

391 : Frame "NELH_band"

```
on enterframe
  global gnelhband_name_list

  part_management(gnelhband_name_list,11,12)
```

```
end

on exitFrame
  go to the frame
end

on mousedown
  global gnelhband_name_list, gcpos_nelhband

  click_part_management(11,12,2,gnelhband_name_list,gcpos_nelhband,"SF_nelh")

end
```

392 : Frame "Neuh_band"

```
on enterframe
  global gneuhband_name_list

  part_management(gneuhband_name_list,11,11)

end

on exitFrame
  go to the frame
end

on mousedown
  global gneuhband_name_list, gcpos_neuhband

       click_part_management(11,11,1,gneuhband_name_list,gcpos_neuhband,"SF_
       neuh")

end
```

393 : Frame "NWL_band"

```
on enterframe

  part_management([],0,0)

end

on exitframe
  go to the frame
end

on mousedown

  click_part_management(0,0,0,[],[],"")

end
```

394 : Frame "pl_band"

```
on enterframe
  global gplband_name_list

  part_management(gplband_name_list,11,14)

end

on exitFrame
  go to the frame
end

on mousedown
  global gplband_name_list, gcpos_plband

  click_part_management(11,14,4,gplband_name_list,gcpos_plband,"SF_pl")
```

```
end
```

395 : Frame "Quit"

```
on enterframe
  global ghelp

  test_sound
  cursor -1

  if rollover(47) then set mess_num to 29
  else if rollover(48) then set mess_num to 30
  else set mess_num to 1

  if ghelp then put getat(gmessage_list,mess_num) into cast "Help-message"
end

on exitFrame
  go to the frame
end
```

396 : Frame "S_band"

```
on enterframe
  global gsband_name_list

  part_management(gsband_name_list,11,24)

end
on exitFrame
  go to the frame
end

on mousedown
  global gsband_name_list, gcpos_sband

  click_part_management(11,24,14,gsband_name_list,gcpos_sband,"SF_bs")

end
```

397 : Frame "SE_band"

```
on enterframe
  global gseband_name_list

  part_management(gseband_name_list,11,14)

end

on exitFrame
  go to the frame
end

on mousedown
  global gseband_name_list, gcpos_seband

  click_part_management(11,14,4,gseband_name_list,gcpos_seband,"SF_se")

end
```

398 : Frame "SF_mi"

```
on enterframe
  global gmiband_name_list

  part_management(gmiband_name_list,11,14)
```

```
end


on exitFrame
  go to the frame
end

on mousedown
  global gmiband_name_list, gcpos_miband

  click_part_management(11,14,4,gmiband_name_list,gcpos_miband,"SF_mi")

end
```

399 : Frame "Start"

```
on enterframe
  global glens, gmenu, gmessage_list ,ghelp, gsound_number, gpathname

  test_music
  test_sound

  rollover_GP(14,21)
  set mess_num to the result

  -- icons&Buttons rollover tests --
  if rollover(30) then set mess_num to 4
  if rollover(35) then set mess_num to 5

  if ghelp=1 then put getat(gmessage_list,mess_num) into cast "Help-   message"

end

on exitFrame
  go to the frame
end


on mousedown
  global gview_ne, gview_bs, gview_mi, gview_pl, gview_se, gview_nwl,  gview_nelh,
gview_neuh, gpart_name_list

  set total to 0
  repeat with n = 14 to 21
    set total to total + rollover(n)
  end repeat

  if rollover(29) and (total=0) then
    puppetsprite 48, true
    menu_sprite ("")
    set the loch of sprite 48 to the mouseh
    set the locv of sprite 48 to the mousev
    cursor 0
    set the visibility of sprite 48 to 1
    menu_cursor (the loch of sprite 48, the locv of sprite 48)
    menu_choice (the result)

  else

    set n to 1
    set choice to 0
    repeat while n<9 and choice = 0
      if rollover(n+13) then
        set choice to n
        click_part_sprite(getat(gpart_name_list,n),13+n)
      end if
      set n to n+1
    end repeat

    if choice = 1 and the result then
```

```
    repeat with n = 4 to 9
      if n = gview_ne+3 then set the visibility of sprite n to 1
      else set the visibility of sprite n to 0
    end repeat

    repeat with n = 10 to 21
      set the visibility of sprite n to 1
    end repeat

    repeat with n = 45 to 47
      set the visibility of sprite n to 1
    end repeat

    go to frame "ne_corner"

  else
    if choice = 2 and the result then

      repeat with n = 4 to 5
        if n = gview_bs+3 then set the visibility of sprite n to 1
        else set the visibility of sprite n to 0
      end repeat

      repeat with n = 6 to 21
        set the visibility of sprite n to 1
      end repeat
      repeat with n = 45 to 47
        set the visibility of sprite n to 1
      end repeat

      go to frame "s_band"
    else
      if choice = 3 and the result then

        repeat with n = 4 to 6
          if n = gview_mi+3 then set the visibility of sprite n to 1
          else set the visibility of sprite n to 0
        end repeat

        repeat with n = 7 to 21
          set the visibility of sprite n to 1
        end repeat
        repeat with n = 45 to 47
          set the visibility of sprite n to 1
        end repeat

        go to frame "Mi_band"
      else
        if choice = 4 and the result then

          repeat with n = 4 to 4
            if n = gview_nelh+3 then set the visibility of sprite n to 1
            else set the visibility of sprite n to 0
          end repeat

          repeat with n = 5 to 21
            set the visibility of sprite n to 1
          end repeat
          repeat with n = 45 to 47
            set the visibility of sprite n to 1
          end repeat

          go to frame "ne_lhband"
        else
          if choice = 5 and the result then

            repeat with n = 4 to 6
              if n = gview_pl+3 then set the visibility of sprite n to 1
              else set the visibility of sprite n to 0
            end repeat

            repeat with n = 7 to 21
```

```
            set the visibility of sprite n to 1
          end repeat
          repeat with n = 45 to 47
            set the visibility of sprite n to 1
          end repeat

          go to frame "pl_band"
        else
          if choice = 6 and the result then

            repeat with n = 4 to 4
             if n = gview_se+3 then set the visibility of sprite n to 1
              else set the visibility of sprite n to 0
            end repeat

            repeat with n = 5 to 21
              set the visibility of sprite n to 1
            end repeat
            repeat with n = 45 to 47
              set the visibility of sprite n to 1
            end repeat

            go to frame "se_band"
          else
            if choice = 7 and the result then

              repeat with n = 4 to 4
               if n = gview_nwl+3 then set the visibility of sprite n
                                                                to 1
                else set the visibility of sprite n to 0
              end repeat

              repeat with n = 5 to 21
                set the visibility of sprite n to 1
              end repeat
              repeat with n = 45 to 47
                set the visibility of sprite n to 1
              end repeat

              go to frame "nwl_band"
            else
              if choice = 8 and the result then

                repeat with n = 4 to 4
                  if n = gview_neuh+3 then set the visibility of sprite
                                                              n to 1
                  else set the visibility of sprite n to 0
                end repeat

                repeat with n = 5 to 21
                  set the visibility of sprite n to 1
                end repeat
                repeat with n = 45 to 47
                  set the visibility of sprite n to 1
                end repeat

                go to frame "ne_uhband"
              end if
            end if
          end if
        end if
      end if
    end if
  end if
end
```

400 : Frame Volume settings

```
on enterframe
```

```
  global ghelp, gopened_hand

  test_sound

  if rollover(47) then
    set mess_num to 57
    cursor -1
  else
    if rollover(48) then
      set mess_num to 58
      cursor -1
    else
      if rollover(45) then
        cursor gopened_hand
        set mess_num to 56
      else
        set mess_num to 1
        cursor -1
      end if
    end if
  end if

  if ghelp then put getat(gmessage_list,mess_num) into cast "Help-message"
end

on exitFrame
  go to the frame
end
```

401 : Frames "DF_..."

```
on enterframe
  global ghelp, gsound, gmusic, gmessage_list, gvolume_cursor_music,
       gvolume_cursor_sound

  test_music
  test_sound

  set mess_num to 1

  -- cheese menu zone rollover test --
  if rollover(29) then
    cursor gmenu
    set mess_num to 2
  else
    if rollover(27) and gsound then
      cursor gvolume_cursor_sound
      set mess_num to 6
    else
      if rollover(28) and gmusic then
        cursor gvolume_cursor_music
        set mess_num to 7
      else
        cursor 0
        set mess_num to 1
      end if
    end if
  end if

  -- icons&Buttons rollover tests --
  Buttons_icons_test
  if the result <> 1 then set mess_num to the result

  if ghelp=1 then put getat(gmessage_list,mess_num) into cast "Help-   message"

end

on exitFrame
  go to the frame
end
```

```
on mousedown
  global gframes_pic, gflowercast, gflowername_lat, gfselect, gnum_fr

  if rollover(29) then
    puppetsprite 48, true
    menu_sprite ("")
    set the loch of sprite 48 to the mouseh
    set the locv of sprite 48 to the mousev
    cursor 0
    set the visibility of sprite 48 to 1
    menu_cursor (the loch of sprite 48, the locv of sprite 48)
    menu_choice (the result)
  end if

  updatestage
end
```

402 : Frames "DF_Seasons"

```
on enterframe
  global ghelp, gsound, gmusic, gmessage_list, gvolume_cursor_music,
      gvolume_cursor_sound

  test_music
  test_sound

  set mess_num to 1

  -- cheese menu zone rollover test --
  if rollover(29) then
    cursor gmenu
    set mess_num to 2
  else
    if rollover(27) and gsound then
      cursor gvolume_cursor_sound
      set mess_num to 6
    else
      if rollover(28) and gmusic then
        cursor gvolume_cursor_music
        set mess_num to 7
      else
        cursor 0
      end if
    end if
  end if

  -- icons&Buttons rollover tests --
  Seasons_buttons_icons_test
  if the result <> 1 and the result <> 17 then set mess_num to the result

  if ghelp=1 then put getat(gmessage_list,mess_num) into cast "Help-message"

end

on exitFrame
  go to the frame
end

on mousedown
  global gframes_pic, gflowercast, gflowername_lat, gfselect, gnum_fr

  if rollover(29) then
    puppetsprite 48, true
    menu_sprite ("")
    set the loch of sprite 48 to the mouseh
    set the locv of sprite 48 to the mousev
    cursor 0
    set the visibility of sprite 48 to 1
    menu_cursor (the loch of sprite 48, the locv of sprite 48)
    menu_choice (the result)
    updatestage
```

```
  end if

end
```

403 : Frames "SF_..."

```
on enterframe
  global ghelp, gsound, gmusic, gmessage_list, gnum_fr, gfselect

  test_music
  test_sound

  if gnum_fr <> 0 then
    if gfselect<>0 then set the visibility of sprite 37+gfselect to true
  end if

  rollover_frame(gnum_fr)
  set mess_num to the result

  -- icons&Buttons rollover tests --
  Buttons_icons_test
  if the result <> 1 then set mess_num to the result

  if ghelp=1 then put getat(gmessage_list,mess_num) into cast "Help-   message"

end

on exitFrame
  go to the frame
end

on mousedown
  global gframes_pic, gflowercast, gflowername_lat, gfselect, gnum_fr

  set total to 0
  if gnum_fr <> 0 then
    repeat with n = 38 to 37+gnum_fr
      set total to total + rollover(n)
    end repeat
  end if

  if rollover(29) and total=0 and not rollover(26) then
    puppetsprite 48, true
    menu_sprite ("")
    set the loch of sprite 48 to the mouseh
    set the locv of sprite 48 to the mousev
    cursor 0
    set the visibility of sprite 48 to 1
    menu_cursor (the loch of sprite 48, the locv of sprite 48)
    menu_choice (the result)
  else
    if gnum_fr <> 0 then
      set num to 0
      repeat with n = 1 to gnum_fr
        if rollover(n+37) then
          set num to n
        end if
      end repeat
      if num<>gfselect and num<>0 then
        click_1
        reset_buttons_disp
        deselect_flower("Mask_frame")
        select_flower("Mask_frame",num)
      end if
    end if
  end if
end
```

404 : Frames "SF_seasons"

```
on enterframe
  global ghelp, gsound, gmusic, gmessage_list, gmonths_list, gflower_num, ¬
         gvolume_cursor_music, gvolume_cursor_sound

  test_music
  test_sound

  puppetsprite 16, true
  puppetsprite 17, true

  if gflower_num = 1 then
    set the castnum of sprite 16 to the number of cast "Button_noprev"
  end if
  if gflower_num = total(gmonths_list) then
    set the castnum of sprite 17 to the number of cast "Button_nonext"
  end if

  updatestage

  set mess_num to 1

  -- cheese menu zone rollover test --
  if rollover(29) then
    cursor gmenu
    set mess_num to 2
  else
    if rollover(27) and gsound then
      cursor gvolume_cursor_sound
      set mess_num to 6
    else
      if rollover(28) and gmusic then
        cursor gvolume_cursor_music
        set mess_num to 7
      else
        cursor 0
      end if
    end if
  end if

  -- icons&Buttons rollover tests --
  Seasons_buttons_icons_test
  if the result <> 1 then set mess_num to the result

  if ghelp=1 then put getat(gmessage_list,mess_num) into cast "Help-   message"

end

on exitFrame
  go to the frame
end

on mousedown
  global gframes_pic, gflowercast, gflowername_lat, gfselect, gnum_fr

  if rollover(29) then
    puppetsprite 48, true
    menu_sprite ("")
    set the loch of sprite 48 to the mouseh
    set the locv of sprite 48 to the mousev
    cursor 0
    set the visibility of sprite 48 to 1
    menu_cursor (the loch of sprite 48, the locv of sprite 48)
    menu_choice (the result)
    updatestage
  end if

end
```

405 : Frames and selections management

```
on select_flower framemask, num
  global gfselect, gflowers_list, gflowercast, gflowername_lat, gflowername_sw

  transf(num)
  set gflowername_lat to
      getat(gflowers_list,findpos(gflowers_list,gflowercast&the result))
  set desc to getat(gflowers_desc,findpos(gflowers_desc,gflowername_lat))
  set gflowername_sw to line 1 of desc

  set castname to framemask&"_dw"
  set the castnum of sprite num+37 to the number of cast castname
  set gfselect to num

  updatestage

end

on deselect_flower framemask
  global gfselect, gflowername_lat
  if gfselect<>0 then
    set castname to framemask&"_up"
    set the visibility of sprite gfselect+37 to 0
    set the castnum of sprite gfselect+37 to the number of cast castname
    set gfselect to 0
    put " " into cast "Flower name latin"
    put " " into cast "Flower name swedish"
    put "" into gflowername_lat
    put "" into gflowername_sw
  end if
end

on set_frames flowercast
  global gframes_pic, gflowername_lat, gflowers_list, gpos_coord_frames,
      gcoord_frames, gnum_fr, gfselect

  set gnum_fr to get_frame_number(flowercast)

  repeat with x = 1 to gnum_fr

    transf(x)
    set frame to flowercast&the result
    set pos to getat(gpos_coord_frames,findpos(gpos_coord_frames,frame))
    set c1 to getat(gcoord_frames,pos*4-3)
    set c2 to getat(gcoord_frames,pos*4-2)
    set c3 to getat(gcoord_frames,pos*4-1)
    set c4 to getat(gcoord_frames,pos*4)

    puppetsprite 37+x, true
    spritebox 37+x, the left of sprite 11 + c1, the top of sprite 11 + c2,¬
                    the left of sprite 11 + c3, the top of sprite 11 + c4
    set the blend of sprite 37+x to 100
  end repeat

  repeat with n = 1 to 6
    if gfselect<>n then set the visibility of sprite 37+n to false
    else set the visibility of sprite 37+n to true
  end repeat

end

on reset_frames
  global gnum_fr, gfselect

  if gfselect <> 0 then set the visibility of sprite gfselect+37 to 0

  set gnum_fr to 0
end

on close_frames
```

```
  repeat with n = 38 to 43
    puppetsprite n, false
  end repeat
end

on set_frame frame
  global gpos_coord_frames, gcoord_frames, gnum_fr, gfselect

  set gnum_fr to back_transf(frame)
  set gfselect to 1

  if gnum_fr<>0 then

    set pos to getat(gpos_coord_frames,findpos(gpos_coord_frames,frame))
    set c1 to getat(gcoord_frames,pos*4-3)
    set c2 to getat(gcoord_frames,pos*4-2)
    set c3 to getat(gcoord_frames,pos*4-1)
    set c4 to getat(gcoord_frames,pos*4)

    puppetsprite 38, true
    spritebox 38, the left of sprite 11 + c1, the top of sprite 11 + c2,¬
                the left of sprite 11 + c3, the top of sprite 11 + c4
    set the blend of sprite 38 to 100

    set the visibility of sprite 38 to true

  else

    set the visibility of sprite 38 to false

  end if

end

on get_frame_number flowercast
  global gframes_pic

  set nfr to getat(gframes_pic,findpos(gframes_pic, flowercast))

  return nfr
end
```

406 : Highlights management

```
on SURLIGNEMENT

  global gBusy_spots_list , Inter , Inter2 , PosB , PosV

  set i to 1
  set Inter to 0
  set Inter2 to 0
  set PosB to 0
  set PosV to 0

  repeat while ( i < 14 )
    if ( GetAt ( gBusy_spots_list , i ) < 14 ) and ( GetAt (
       gBusy_spots_list , i ) <> 0 ) then
      set j to ( GetAt ( gBusy_spots_list , i ) )
      set PosB to i
      set i to i+1
      set Trouve to false

      repeat while ( i < 14 ) and ( Trouve = false )
        if ( GetAt ( gBusy_spots_list , i ) > 13 ) then
          set jj to ( GetAt ( gBusy_spots_list , i ) )
          set PosV to i

          LIGNE PosB , PosV

          set Trouve to true
          set i to i+1
```

```
        else
          if ( GetAt ( gBusy_spots_list , i ) = 0 ) then
            set i to i+1
          else
            set Trouve to true
          end if

        end if

      end repeat

      if ( Trouve = false ) then
        set k to 1
        -- Then we begin at the begin of the
        repeat while ( k < PosB ) and ( Trouve = false )
          if ( GetAt ( gBusy_spots_list , k ) > 13 ) then
            set Posv to k
            set Trouve to true
            set Inter2 to 610

            LIGNE PosB , 13

            set jj to ( GetAt ( gBusy_spots_list , k ) )
            set Inter2 to sprite_coord(jj)
            set Inter to 36

            LIGNE PosV , 1

          else
            if ( GetAt ( gBusy_spots_list , k ) = 0 ) then
              set k to k+1
            else
              set Trouve to true
            end if

          end if

        end repeat

      end if

    else
      set i to i+1
    end if

  end repeat

end

on LIGNE PB , PV
  global gHighlights_list , Posb, Posv

  set Bon to false
  set i to 1

  repeat while ( i < 13 ) and ( Bon = false )
    if ( GetAt ( gHighlights_list , i ) = PB )  then
      set Bon to true
    else
      set i to i+2
    end if
  end repeat

  if ( Bon = false ) then
    set i to 21
    set x to 1

    repeat while ( i < 27 )
      if ( the visible of sprite i = false ) then
        if posb < posv then
          set coordh_1 to GetAt(gScale_Coord,pb)
          set coordh_2 to GetAt(gScale_Coord,pv)
```

```
          setAt ( gHighlights_list , x , PB )
          setAt ( gHighlights_list , x+1 , PV )
        else
          if pv=1 then
            set coordh_1 to GetAt(gScale_Coord,posb)
            set coordh_2 to GetAt(gScale_Coord,13)
            setAt ( gHighlights_list , x , Posv )
            setAt ( gHighlights_list , x+1 , Posb )
          else
            set coordh_1 to GetAt(gScale_Coord,1)
            set coordh_2 to GetAt(gScale_Coord,posv)
            setAt ( gHighlights_list , x , PosB )
            setAt ( gHighlights_list , x+1 , PosV )
          end if
        end if

        set the stretch of sprite i to true
        Spritebox i , coordh_1 + the left of sprite 6, the locv of sprite
      6-3 , coordh_2 + the left of sprite 6-2, the locv of sprite 6+1

        set the visible of sprite i to true
        updatestage

        set i to 27
      else
        set i to i+1
        set x to x+2

      end if

    end repeat

  end if

end

on DESURLIGNEMENT me

  global gHighlights_list

  set i to 1

  repeat while ( i < 13 )
    if ( GetAt ( gHighlights_list , i ) = me ) or ( GetAt (
       gHighlights_list , i+1 ) = me ) then
      if ( i = 1 ) then
        set the Visible of sprite 21 to false
        setAt ( gHighlights_list , i , 0 )
        setAt ( gHighlights_list , i+1 , 0 )
      else
        if ( i=3 ) then
          set the Visible of sprite 22 to false
          setAt ( gHighlights_list , i , 0 )
          setAt ( gHighlights_list , i+1 , 0 )
        else
          if ( i=5 ) then
            set the Visible of sprite 23 to false
            setAt ( gHighlights_list , i , 0 )
            setAt ( gHighlights_list , i+1 , 0 )
          else
            if ( i=7 ) then
              set the Visible of sprite 24 to false
              setAt ( gHighlights_list , i , 0 )
              setAt ( gHighlights_list , i+1 , 0 )
            else
              if ( i=9 ) then
                set the Visible of sprite 25 to false
                setAt ( gHighlights_list , i , 0 )
                setAt ( gHighlights_list , i+1 , 0 )
              else
                set the Visible of sprite 26 to false
                setAt ( gHighlights_list , i , 0 )
```

```
                setAt ( gHighlights_list , i+1 , 0 )
              end if
            end if
          end if
        end if
      end if
    end if
    set i to i+2
  end repeat

end
```

407 : Info DF ne_corner

```
on mousedown
  Info("DF_ne")
end
```

408 : Info mi_band

```
on mousedown
  Info("DF_mi")
end
```

409 : Info nelh_band

```
on mousedown
  Info("DF_nelh")
end
```

410 : Info neuh_band

```
on mousedown
  Info("DF_neuh")
end
```

411 : Info pl_band

```
on mousedown
  Info("DF_pl")
end
```

412 : Info s_band

```
on mousedown
  Info("DF_bs")
end
```

413 : Info se_band

```
on mousedown
  Info("DF_se")
end
```

414 : Initialize frame "Season_..."

```
on enterframe
  global gflowercast, gflowername_lat, gflowername_sw, gcurrent_flower,¬
         gpos_current_list, gflowers_list, gmonths_list, gpos_months_list,¬
         gframe, gflower_num

  repeat with n = 10 to 16
    set the visibility of sprite n to 0
  end repeat
```

```
  set gframe to     getpropat(gflowers_list,getpos(gflowers_list,gcurrent_flower))
  rem_tail(gframe)
  set gflowercast to the result

  puppetsprite 10, true
  set the castnum of sprite 10 to the number of cast gflowercast
  spritebox 10, the left of sprite 11 + 10, the top of sprite 11 + 9,¬
                the right of sprite 11 - 10, the bottom of sprite 11 - 22

  repeat with n = 10 to 26
    set the visibility of sprite n to 1
  end repeat

  set gflower_num to 1

  set gflowername_lat to gcurrent_flower
  set desc to getat(gflowers_desc,findpos(gflowers_desc,gflowername_lat))
  set gflowername_sw to line 1 of desc

  put getat(gmonths_list,gpos_months_list) into cast "Month name"

  set_frame(gframe)
  pointer_mmap(gflowername_lat)

end enter frame
```

415 : Initialize the frame "SF_mi"

```
on exitFrame
  global gcoord_miband, gcpos_miband

  SF_launch(gcoord_miband,gcpos_miband)

end
```

416 : Initialize the frame "SF_necorner"

```
on exitFrame
  global gcoord_necorner, gcpos_necorner

  SF_launch(gcoord_necorner,gcpos_necorner)

end
```

417 : Initialize the frame "SF_nelh_band"

```
on exitFrame
  global gcoord_nelhband, gcpos_nelhband

  SF_launch(gcoord_nelhband,gcpos_nelhband)

end
```

418 : Initialize the frame "SF_neuh"

```
on exitFrame
  global gcoord_neuhband, gcpos_neuhband

  SF_launch(gcoord_neuhband,gcpos_neuhband)

end
```

419 : Initialize the frame "SF_pl"

```
on exitFrame
  global gcoord_plband,gcpos_plband
```

```
  SF_launch(gcoord_plband,gcpos_plband)

end
```

420 : Initialize the frame "SF_sband"

```
on exitFrame
  global gcpos_sband, gcoord_sband

  SF_launch(gcoord_sband,gcpos_sband)

end
```

421 : Initialize the frame "SF_se"

```
on exitFrame
  global gcoord_seband, gcpos_seband

  SF_launch(gcoord_seband,gcpos_seband)

end
```

422 : Kind beep sound

```
on warning
  puppetsound "Beep_kind"
  updatestage
end
```

423 : map_button

```
on map_button map_name, flower_name, flower_list
  global gflowercast, gflowers_num, ghelp

  -- this procedure launch the displaying of the area clicked on the map of  a
part --

  set flower to ""
  set n to 1
  set ch to char n of flower_name
  -- the point was used at the end of the name of the picture because some
      pictures represent flowers which have several occurences in the
      current part. So the      point was used to point
            the different occurences (.1, .2, ...) --

  repeat while ch<>"." and n<=length(flower_name)
    set flower to flower&ch
    set n to n+1
    set ch to char n of flower_name
  end repeat

  set gflowercast to flower
  set gflowers_num to findpos(flower_list,flower)

  cursor 4

  go to frame map_name

end
```

424 : Menu_management

```
on menu_cursor coord_h, coord_v
  global ghelp, gsound, gmusic, gmessage_list

  set mess_num to 1
  -- the default choice is camcel --
  set choice to 3
  -- test what part of the pie menu is currently under the mouse point and
       shadows the part in consequence --
  repeat while the mousedown

    if ( the mouseh < coord_h + 15 ) and ( the mouseh > coord_h - 15 ) ¬
            and ( the mousev < coord_v + 15 ) and ( the mousev > coord_v -
             15 ) then
      -- shadows the part "cancel"--
      menu_sprite("")
      set mess_num to 8
      set choice to 3

    else if (the mouseh < coord_h ) and ( the mousev > coord_v ) then
      -- shadows the part "Quit"--
      menu_sprite("Quit")
      set mess_num to 9
      set choice to 2

    else if (the mouseh < coord_h ) and ( the mousev < coord_v ) then
      -- shadows the part "Music"--
      menu_sprite("Music")
      if gmusic = 0 then set mess_num to 10
      else set mess_num to 11
      set choice to 1

    else if (the mouseh > coord_h ) and ( the mousev < coord_v ) then
      -- shadows the part "Sound"--
      menu_sprite("Sound")
      if gsound = 0 then set mess_num to 12
      else set mess_num to 13
      set choice to 4

    else if (the mouseh > coord_h ) and ( the mousev > coord_v ) then
      -- shadows the part "Help"--
      menu_sprite("Help")
      if ghelp = 0 then set mess_num to 14
      else set mess_num to 15
      set choice to 5
    end if
    if ghelp=1 then put getat(gmessage_list,mess_num) into cast "Help-
      message"
  end repeat

  set the visibility of sprite 48 to 0
  puppetsprite 48, false
  updatestage

  cursor 200

  return choice
end

on menu_choice choice
  global gsound, gmusic, ghelp, gmusic_name, gpuppet_list, gpos_curs_mus,
       gpos_curs_sound, gmusic_volume, gsound_volume, gmusic_volume_prev,
       gsound_volume_prev, gpos_curs_mus_prev, gpos_curs_sound_prev

  -- manage the choice of the user --

  if choice = 1 then
    -- display an info window to the user and apply a non-door to the  boolean
gmusic --
    set gmusic to ( gmusic = 0 )
    set the visibility of sprite 32 to gmusic
```

```
    set the visibility of sprite 28 to gmusic
    if gmusic = 0 then
      set the castnum of sprite 47 to the number of cast "Music_off"
    else
      set the castnum of sprite 47 to the number of cast "Music_on"
      set gpos_curs_mus to 2
      set gmusic_volume to 100
      set gmusic_volume_prev to gmusic_volume
      set gpos_curs_mus_prev to gpos_curs_mus
      set the volume of sound 2 to gmusic_volume
    end if

    set the visibility of sprite 47 to 1
    updatestage

    if gmusic=0 then sound stop 2
    else sound playfile 2, gmusic_name

    delay(2)
    set the visibility of sprite 47 to 0

  else
    if choice = 2 then
      -- display an confirmation window to the user located in the frame
       "Quit"--

      set the visibility of sprite 37 to ghelp

      -- store the current puppetsprite states of the different sprites --
      set v to [0,0,0]
      repeat with n=1 to 48
        if n<46 then
          if n<>37 then
            setat gpuppet_list, n, the puppet of sprite n
            puppetsprite n, true
          end if
        else
          setat v, n-45 , the visibility of sprite n
          set the visibility of sprite n to 1
        end if
      end repeat

      play frame "Quit"

      repeat with n=1 to 48
        if n<46 then
          if n<>37 then
            puppetsprite n, getat (gpuppet_list, n)
          end if
        else
          set the visibility of sprite n to getat (v,n-45)
        end if
      end repeat
    else
      if choice = 4 then
        -- display an info window to the user and apply a non-door to the
       boolean gsound --
        set gsound to ( gsound = 0 )
        set the visibility of sprite 31 to gsound
        set the visibility of sprite 27 to gsound

        if gsound = 0 then
          set the castnum of sprite 45 to the number of cast "Sound_off"
        else
          set the castnum of sprite 45 to the number of cast "Sound_on"
          set gpos_curs_sound to 2
          set gsound_volume to 100
          set gsound_volume_prev to gsound_volume
          set gpos_curs_sound_prev to gpos_curs_sound
          set the volume of sound 3 to gsound_volume
        end if
```

```
        set the visibility of sprite 45 to 1
        updatestage
        delay(2)
        set the visibility of sprite 45 to 0

      else
        if choice = 5 then
          -- display an info window to the user and apply a non-door to the
       boolean ghelp --
          set ghelp to ( ghelp = 0 )
          put "" into cast "Help-message"
          set the visibility of sprite 30 to ghelp
          set the visibility of sprite 37 to ghelp

          if ghelp = 0 then set the castnum of sprite 46 to the number of
       cast "Help_off"
          else set the castnum of sprite 46 to the number of cast "Help_on"

          set the visibility of sprite 46 to 1
          updatestage
          delay(2)
          set the visibility of sprite 46 to 0

        end if
      end if
    end if
  end if

end

on menu_sprite selected
  global gmusic, gsound, ghelp, menu_name

  -- diplay the pie menu which represents the currently shadowed part --

  set menu_name to "Menu"
  if gmusic+gsound+ghelp <> 0 then set menu_name to menu_name & "-"
  if gmusic = 1 then set menu_name to menu_name & "m"
  if gsound = 1 then set menu_name to menu_name & "s"
  if ghelp = 1 then set menu_name to menu_name & "h"
  if selected <> "" then set menu_name to menu_name & "-"
  set menu_name to menu_name & selected

  set the castnum of sprite 48 to the number of cast menu_name
  updatestage
end
```

425 : Month lists creation management

```
on List_of_Months
  global gflowers_desc, gJanuary_list, gFebruary_list, gMarch_list,gApril_list,
gMay_list ,¬
         gJune_list, gJuly_list, gAugustus_list, gSeptember_list, gOctober_list,
gNovember_list, ¬
         gDecember_list, gFlower

  set gJanuary_list    = [0]
  set gFebruary_list   = [0]
  set gMarch_list      = [0]
  set gApril_list      = [0]
  set gMay_list        = [0]
  set gJune_list       = [0]
  set gJuly_list       = [0]
  set gAugustus_list   = [0]
  set gSeptember_list  = [0]
  set gOctober_list    = [0]
  set gNovember_list   = [0]
  set gDecember_list   = [0]

  set Length to count(gflowers_desc)
```

```
  set i to 1

  repeat while ( i <= Length )
    set gFlower to ( Getpropat ( gflowers_desc , i ))
    set desc to ( Getat ( gflowers_desc , i ))
    set months_Flower to line 4 of desc
    ANALYZE_STRING months_Flower
    set i to i+1
  end repeat

  DeleteAt gjanuary_list , 1
  DeleteAt gfebruary_list , 1
  DeleteAt gmarch_list , 1
  DeleteAt gApril_list , 1
  DeleteAt gMay_list , 1
  DeleteAt gjune_list , 1
  DeleteAt gjuly_list , 1
  DeleteAt gaugustus_list , 1
  DeleteAt gseptember_list , 1
  DeleteAt goctober_list , 1
  DeleteAt gnovember_list , 1
  DeleteAt gdecember_list , 1

end

on ANALYZE_STRING months_flower

  set Size to Length(months_flower)

  set j to 1
  set Trouve to false
  set Inf_months to ""
  set Sup_months to ""

  repeat while ( j < Size +1 )
    set Ch to ( char j of months_flower )
    if ( Ch <> "-" ) then
      set Inf_months to Inf_months & Ch
      set j to j+1
    else
      set j to j+1
      set Trouve to true
    end if

    if ( Trouve = true ) then
      repeat while ( j < Size+1 )
        set Ch to ( char j of months_flower )
        set Sup_months to Sup_months & Ch
        set j to j+1
      end repeat
    end if
  end repeat
  set Inf_months to ( Value ( Inf_months ))
  set Sup_months to ( Value ( Sup_months ))
  FILL_MONTHS_LIST Inf_months , Sup_months

end

on FILL_MONTHS_LIST Inf_months , Sup_months

  if ( Inf_months = 0 ) then
  else
    if ( Inf_months < Sup_months ) then
      repeat while ( Inf_months < Sup_months+1 )
        FILL2_LIST Inf_months
        set Inf_months to Inf_months+1
      end repeat
    else
      if ( Sup_months = 0 ) then
        FILL2_LIST Inf_months
      else
        repeat while ( Inf_months < 13 )
```

```
          FILL2_LIST Inf_months
          set Inf_months to Inf_months+1
        end repeat
        set j to 1
        repeat while ( j < Sup_months+1 )
          FILL2_LIST j
          set j to j+1
        end repeat
      end if
    end if
  end if

end

on FILL2_LIST Inf_months
  Global gJanuary_list, gFebruary_list, gMarch_list, gApril_List,      gMay_list,
gJune_list, gJuly_list, gAugustus_list, gSeptember_list,  gOctober_list,
gNovember_list, gDecember_list, gflower

  if ( Inf_months = 1 ) then
    Append gJanuary_list , gFlower
  else
    if ( Inf_months = 2 ) then
      Append gFebruary_list , gFlower
    else
      if ( Inf_months = 3 ) then
        Append gMarch_list , gFlower
      else
        if ( Inf_months = 4 ) then
          Append gApril_list , gFlower
        else
          if ( Inf_months = 5 ) then
            Append gMay_list , gFlower
          else
            if ( Inf_months = 6 ) then
              Append gJune_list , gFlower
            else
              if ( Inf_months = 7 ) then
                Append gJuly_list , gFlower
              else
                if ( Inf_months = 8 ) then
                  Append gAugustus_list , gFlower
                else
                  if ( Inf_months = 9 ) then
                    Append gSeptember_list , gFlower
                  else
                    if ( Inf_months = 10 ) then
                      Append gOctober_list , gFlower
                    else
                      if ( Inf_months = 11 ) then
                        Append gNovember_list , gFlower
                      else
                        Append gDecember_list , gFlower
                      end if
                    end if
                  end if
                end if
              end if
            end if
          end if
        end if
      end if
    end if
  end if

end
```

426 : Next and previous flower

```
on next_flower
  global gcurrent_list, gcurrent_flower, gpos_current_list, gmonths_list,¬
```

```
        gpos_months_list

  set gpos_current_list to gpos_current_list + 1

  if gpos_current_list > count(gcurrent_list) then
    next_month
    put getat(gmonths_list,gpos_months_list) into cast "Month name"
    if the result = 0 then set gpos_current_list to 1
    else set gpos_current_list to count(gcurrent_list)
  end if
  set gcurrent_flower to getat(gcurrent_list,gpos_current_list)
end

on previous_flower
  global gcurrent_list, gcurrent_flower, gpos_current_list, gmonths_list,¬
        gpos_months_list

  set gpos_current_list to gpos_current_list - 1

  if gpos_current_list < 1 then
    previous_month
    put getat(gmonths_list,gpos_months_list) into cast "Month name"
    if the result = 0 then set gpos_current_list to count(gcurrent_list)
    else set gpos_current_list to 1
  end if
  set gcurrent_flower to getat(gcurrent_list,gpos_current_list)
end
```

427 : Next and previous month management

```
on next_month
  global gcurrent_list, gmonths_list, gpos_months_list


  set end to false
  set nonext to 0
  set temp1 to gcurrent_list
  set temp2 to gpos_months_list

  set gpos_months_list to gpos_months_list + 1

  repeat while (not end)
    if gpos_months_list<=count(gmonths_list) then
      set month to getat(gmonths_list,gpos_months_list)

      convert_month(month)
      set result to the result

      if count(result)<>0 then set end to true
      else set gpos_months_list to gpos_months_list + 1
    else
      set end to true
      set nonext to 1
      set result to temp1
      set gpos_months_list to temp2
    end if
  end repeat
  set gcurrent_list to result
  return nonext
end

on previous_month
  global gcurrent_list, gmonths_list, gpos_months_list

  set end to false
  set noprevious to 0
  set temp1 to gcurrent_list
  set temp2 to gpos_months_list

  set gpos_months_list to gpos_months_list - 1
```

```
  repeat while (not end)
    if gpos_months_list>=1 then
      set month to getat(gmonths_list,gpos_months_list)

      convert_month(month)
      set result to the result

      if count(result)<>0 then set end to true
      else set gpos_months_list to gpos_months_list - 1
    else
      set end to true
      set noprevious to 1
      set result to temp1
      set gpos_months_list to temp2
    end if
  end repeat
  set gcurrent_list to result
  return noprevious
end
```

428 : Next area mi_band

```
on mousedown
  global gcoord_miband, gcpos_miband

  Next_area(gcoord_miband, gcpos_miband)
end
```

429 : Next area ne_corner

```
on mousedown
  global gcoord_necorner, gcpos_necorner

  Next_area (gcoord_necorner, gcpos_necorner)
end
```

430 : Next area nelh_band

```
on mousedown
  global gcoord_nelhband,gcpos_nelhband

  Next_area(gcoord_nelhband, gcpos_nelhband)
end
```

431 : Next area pl_band

```
on mousedown
  global gcoord_plband, gcpos_plband

  Next_area(gcoord_plband, gcpos_plband)
end
```

432 : Next area s_band

```
on mousedown
  global gcoord_sband, gcpos_sband

  Next_area (gcoord_sband, gcpos_sband)
end
```

433 : Next area se_band

```
on mousedown
  global gcoord_seband, gcpos_seband

  Next_area(gcoord_seband, gcpos_seband)
```

```
end
```

434 : Number->char, Char->number

```
on transf number

  if number = 1 then
    set result to "_a"
  else
    if number = 2 then
      set result to "_b"
    else
      if number = 3 then
        set result to "_c"
      else
        if number = 4 then
          set result to "_d"
        else
          if number = 5 then
            set result to "_e"
          else
            set result to "_f"
          end if
        end if
      end if
    end if
  end if

  return result
end

on back_transf string

  set n to 1
  set ch to char 1 of string
  set end to false
  set result to 0
  set res to ""

  repeat while not end and n<=length(string)

    if ch = "_" then
      set res to char n+1 of string
      if res = "a" then
        set result to 1
      else
        if res = "b" then
          set result to 2
        else
          if res = "c" then
            set result to 3
          else
            if res = "d" then
              set result to 4
            else
              if res = "e" then
                set result to 5
              else
                set result to 6
              end if
            end if
          end if
        end if
      end if
      set end to true
    else
      set n to n + 1
      set ch to char n of string
    end if

  end repeat
```

```
  return result
end
```

435 : Part management

```
on part_management part_list, start_ch, end_ch
  global ghelp, gmessage_list

  test_music
  test_sound

  rollover_FP(part_list,start_ch,end_ch)
  set mess_num to the result

  -- icons&Buttons rollover tests --
  if rollover(30) then set mess_num to 4
  if rollover(35) then
    set cnum to the castnum of sprite 35
    if the name of cast cnum = "Button_CV_up" then set mess_num to 17
    else set mess_num to 26
  end if
  if rollover(36) then set mess_num to 16

  if ghelp=1 then put getat(gmessage_list,mess_num) into cast "Help-   message"
  if mess_num <> 3 then
    puppetsprite 39, false
    set the visibility of sprite 38 to 0
    set the visibility of sprite 39 to 0
  end if
end
```

436 : pointer_map

```
on pointer_map coord_list, pos_list, flowername, nf

  -- in the frames "SF_...", points the location(s) of a flower or area on   the
small map --

  set x to getat(pos_list,findpos(pos_list,flowername))

  repeat with m = 1 to nf
    set c1 to getat(coord_list,x*4-3)
    set c2 to getat(coord_list,x*4-2)
    set c3 to getat(coord_list,x*4-1)
    set c4 to getat(coord_list,x*4)

    puppetsprite 21+m, true
    spritebox 21+m, c1 + the left of sprite 21, c2 + the top of sprite 21,   c3 +
the left of sprite 21, c4 + the top of sprite 21
    set the blend of sprite 21+m to 60
    set x to x + 1
  end repeat
  set the visibility of sprite 23 to (nf>=2)
  set the visibility of sprite 24 to (nf=3)

end

on pointer_mmap flowername
  global gcoord_mmap_parts, gflowers_list

  -- in the frames "Season_..." points the location(s) of a flower on the
      small garden map --

  set n to 1
  set end to false
  set previous_number to 0
  set temp_list to [:]
```

```
  -- for each occurence of the flower in the general list, points its  location on
the map --

  repeat while not end
    set test to getone(gflowers_list,flowername)
    if test = 0 then
      set end to true
    else
      set pos to findpos(gflowers_list,test)
      setaprop temp_list, test, getat(gflowers_list,pos)
      deleteat gflowers_list, pos
      set flowercast to test
      set number to char 3 of flowercast

      -- correct only because a flower has a maximum of 2 different
       locations --

      if number <> previous_number then

        set c1 to getat(gcoord_mmap_parts,number*4-3)
        set c2 to getat(gcoord_mmap_parts,number*4-2)
        set c3 to getat(gcoord_mmap_parts,number*4-1)
        set c4 to getat(gcoord_mmap_parts,number*4)
        puppetsprite 3+n, true
        spritebox 3+n, c1 + the left of sprite 3, c2 + the top of sprite 3,
       c3 + the left of sprite 3, c4 + the top of sprite 3
        set the blend of sprite 3+n to 60
        set n to n+1
        set previous_number to number
      end if

    end if
  end repeat

  set the visibility of sprite 5 to (n-1>=2)

  repeat with n = 1 to count(temp_list)
    setaprop gflowers_list, getpropat(temp_list,n), getat(temp_list,n)
  end repeat

end
```

437 : Prev area mi_band

```
on mousedown
  global gcoord_miband, gcpos_miband

  Prev_area(gcoord_miband,gcpos_miband)
end
```

438 : Prev area ne_corner

```
on mousedown
  global gcoord_necorner,gcpos_necorner

  Prev_area(gcoord_necorner,gcpos_necorner)
end
```

439 : Prev area nelh_band

```
on mousedown
  global gcoord_nelhband, gcpos_nelhband

  Prev_area(gcoord_nelhband,gcpos_nelhband)

end
```

440 : Prev area pl_band

```
on mousedown
  global gcoord_plband, gcpos_plband

  Prev_area(gcoord_plband,gcpos_plband)
end
```

441 : Prev area s_band

```
on mousedown
  global gflowers_num, gflowercast, gcoord_sband, gcpos_sband, gfselect,
       gnum_fr, gflowers_list, gflowername_lat

  Prev_area(gcoord_sband,gcpos_sband)

end
```

442 : Prev area se_band

```
on mousedown
  global gcoord_seband, gcpos_seband

  Prev_area(gcoord_seband,gcpos_seband)
end
```

443 : Read Flowers File

```
on read_flowersfile filename, list_flowerscode
  global gflowers_desc

  -- read the file containing the description of the flowers and update the
       general list of --
  -- the flowers by adding their names in regard of the name of the frame    in a
picture. --
  -- create the list containing the descriptions --

  openfile(filename)

  set n to 1
  readfile
  set flow_desc to ""
  set name to ""
  set line_desc to 1
  set EOF to false

  repeat while not EOF

    if line_desc = 1 then
      set code to word 1 of the result
      set line_desc to line_desc + 1
    else
      if line_desc = 2 then
        set name to take_words(the result)
        set end to false

        -- replace the code of the general list by the name of the flower
      contained in the file --
        -- for each occurence of the code in the list --

        repeat while end <> true
          if getone(list_flowerscode,code) = 0 then
            set end to true
          else
            setaprop list_flowerscode,
                   getpropat(list_flowerscode,getpos(list_flowerscode,code)),
                   name
          end if
        end repeat
```

```
        set line_desc to line_desc + 1
      else
        if the result contains "eod" then
          addprop gflowers_desc, name, flow_desc
          set n to n+1
          set flow_desc to ""
          set line_desc to 1
        else
          set result to take_words(the result)
          put result into line line_desc-2 of flow_desc
          set line_desc to line_desc + 1
        end if
      end if
    end if
    readfile
    if the result contains "eof" then set EOF to true

  end repeat

  closefile

end
```

444 : Remove tail of a string

```
on rem_tail string

  set n to 1
  set ch to char 1 of string
  set end to false
  set result to ""

  repeat while not end and n<=length(string)

    if ch <> "_" then
      set result to result & ch
    else
      set end to true
    end if
    set n to n + 1
    set ch to char n of string

  end repeat

  return result

end
```

445 : reset buttons disp

```
on reset_buttons_disp
  set c1 to the left of sprite 13
  set c2 to the top of sprite 13
  set c3 to the right of sprite 13
  set c4 to the bottom of sprite 13
  set the castnum of sprite 13 to the number of cast "Button_disp_lat_name_up"
  spritebox 13, c1, c2, c3, c4

  set c1 to the left of sprite 18
  set c2 to the top of sprite 18
  set c3 to the right of sprite 18
  set c4 to the bottom of sprite 18
  set the castnum of sprite 18 to the number of cast "Button_disp_sw_name_up"
  spritebox 18, c1, c2, c3, c4
end
```

446 : Rollover_fp

```
on rollover_fp flowers_list, start_ch, end_ch
  global glens, gvolume_cursor_music, gvolume_cursor_sound, gsound, gmusic

  -- manages the mouse cursor shape, the help message and the miniature
       picture displayed --
  -- in the frames reached from the main map of the garden --
  -- start_ch represents the first channel containing the sprite of a  flower
shape on the map --
  -- and end_ch do the same for the last channel... --
  -- flowers_list represents the global list containing the name of the
       picture/cast members of --
  -- this part. --

  set rf to 0
  if end_ch <> 0 then
    repeat with n = start_ch to end_ch
      if rollover(n) then set rf to n
      else set the visibility of sprite n to 0
    end repeat
  end if

  if rollover(29) then
    if rf = 0 then
      cursor gmenu
      set message to 2

    else
      cursor glens
      set the visibility of sprite rf to 1
      updatestage

      set fl to getat(flowers_list,rf-10)
      set flower to ""
      set n to 1
      set ch to char n of fl
      repeat while ch<>"." and n<=length(fl)
        set flower to flower&ch
        set n to n+1
        set ch to char n of fl
      end repeat

      set the castnum of sprite 39 to the number of cast flower
      spritebox 39, the left of sprite 38 + 10, the top of sprite 38 + 9,¬
                  the right of sprite 38 - 10, the bottom of sprite 38 - 9

      puppetsprite 39, true
      set the visibility of sprite 38 to 1
      set the visibility of sprite 39 to 1
      set message to 3
    end if
  else
    if rollover(27) and gsound then
      cursor gvolume_cursor_sound
      set message to 6
    else
      if rollover(28) and gmusic then
        cursor gvolume_cursor_music
        set message to 7
      else
        cursor 0
        set message to 1
      end if
    end if
  end if

  return message

end
```

447 : Rollover_frame

```
on rollover_frame num
  global glens, gfselect, gfinger, gvolume_cursor_music,
       gvolume_cursor_sound, gsound, gmusic

  -- determine the shape of the mouse cursor and the message displayed in    the
frames "SF_..."

  set rf to 0
  if num <> 0 then
    repeat with n = 38 to 37+num
      if rollover(n) then set rf to n
      else if n<>gfselect+37 then set the visibility of sprite n to 0
    end repeat
  end if

  if rollover(29) then
    if rf = 0 then
      if rollover(26) then
        cursor -1
        set message to 48
      else
        cursor gmenu
        set message to 2
      end if
    else
      cursor gfinger
      set the visibility of sprite rf to 1
      set message to 25
    end if
  else
    if rollover(27) and gsound then
      cursor gvolume_cursor_sound
      set message to 6
    else
      if rollover(28) and gmusic then
        cursor gvolume_cursor_music
        set message to 7
      else
        cursor 0
        set message to 1
      end if
    end if
  end if

  return message
end
```

448 : Rollover_GP

```
on rollover_gp start_ch, end_ch
  global glens, gvolume_cursor_sound,gvolume_cursor_music, gmusic, gsound

  -- manages the mouse cursor shape, the help message displayed in the
       frames "SF_ ...".--

  set rf to 0
  repeat with n = start_ch to end_ch
    if rollover(n) then set rf to n
    else set the visibility of sprite n to 0
  end repeat

  if rollover(29) then
    if rf = 0 then
      cursor gmenu
      set message to 2
    else
      cursor glens
      set the visibility of sprite rf to 1
      set message to 3
```

```
    end if
  else
    if rollover(27) and gsound then
      cursor gvolume_cursor_sound
      set message to 6
    else
      if rollover(28) and gmusic then
        cursor gvolume_cursor_music
        set message to 7
      else
        cursor 0
        set message to 1
      end if
    end if
  end if

  return message

end
```

449 : Seasons buttons&icons test

```
on Seasons_buttons_icons_test
  global gsound, gmusic

  set mess_num to 1

  if rollover(2) then set mess_num to 37
  if rollover(3) then set mess_num to 38
  if rollover(4) then set mess_num to 39
  if rollover(5) and the visible of sprite 5 then set mess_num to 39
  if rollover(30) then set mess_num to 4
  if rollover(12) then set mess_num to 18
  if rollover(13) then
    set cnum to the castnum of sprite 13
    if the name of cast cnum = "Button_hide_lat_name_up" then set mess_num
      to 61
    else set mess_num to 40
  end if
  if rollover(14) then set mess_num to 41
  if rollover(15) then
    set cnum to the castnum of sprite 15
    if the name of cast cnum = "Button_info_up" then set mess_num to 44
    else if the name of cast cnum = "Button_pic_up" then set mess_num to 45
  end if
  if rollover(16) then
    set cnum to the castnum of sprite 16
    if the name of cast cnum = "Button_prev_up" then set mess_num to 42
    else if the name of cast cnum = "Button_noprev" then set mess_num to 35
  end if
  if rollover(17) then
    set cnum to the castnum of sprite 17
    if the name of cast cnum = "Button_next_up" then set mess_num to 43
    else if the name of cast cnum = "Button_nonext" then set mess_num to 34
  end if
  if rollover(18) then
    set cnum to the castnum of sprite 18
    if the name of cast cnum = "Button_hide_sw_name_up" then set mess_num
      to 62
    else set mess_num to 54
  end if
  if rollover(19) then set mess_num to 55
  if rollover(35) then set mess_num to 16

  return mess_num
end
```

450 : SF launch

```
on SF_launch coord_part, cpos_part
```

```
  global gflowercast, gnum_fr, gflowername_lat, gflowername_sw

  repeat with n = 10 to 16
    set the visibility of sprite n to 0
  end repeat

  puppetsprite 10, true
  set the castnum of sprite 10 to the number of cast gflowercast
  spritebox 10, the left of sprite 11 + 10, the top of sprite 11 + 9,¬
                the right of sprite 11 - 10, the bottom of sprite 11 - 22

  repeat with n = 10 to 26
    set the visibility of sprite n to 1
  end repeat

  if gflowercast="F13/1" then
    set nf to 3
  else
    if gflowercast="F13/9" then
      set nf to 2
    else
      if gflowercast="F14/6" then
        set nf to 2
      else
        set nf to 1
      end if
    end if
  end if

  pointer_map(coord_part,cpos_part,gflowercast,nf)

  set_frames(gflowercast)

  if gnum_fr=0 then
    set gflowername_lat to
      getat(gflowers_list,findpos(gflowers_list,gflowercast))
    set desc to getat(gflowers_desc,findpos(gflowers_desc,gflowername_lat))
    set gflowername_sw to line 1 of desc
    put 1 into cast "Number of flowers"
  else
    put gnum_fr into cast "Number of flowers"
    set gflowername_lat to ""
    set gflowername_sw to ""
  end if
end
```

451 : Sprite coord

```
on sprite_coord flag_num
  if flag_num<=13 then
    set result to 18 + the left of sprite flag_num
  else
    set result to 14 + the left of sprite flag_num
  end if
  return result
end
```

452 : Start_Movie

```
on startmovie
  global gview_ne, gview_bs, gflowername_lat, gfinger, gflowername_sw, ¬
              glens, gmenu, gsound, gmusic, ghelp, gmessage_list, gmusic_name,
              gflowers_desc, gpart_name_list, gnecorner_name_list,
       gsband_name_list, gview_mi, gmiband_name_list, gview_pl,
       gplband_name_list, gview_se, gview_neuh, gneuhband_name_list,
       gcpos_neuhband, gseband_name_list, gview_nwl, gview_nelh,
       gnelhband_name_list, gcoord_neuhband, gcoord_necorner,
       gcpos_necorner, gcoord_sband, gcpos_sband, gcpos_miband,
       gcoord_miband, gcoord_plband, gcpos_plband, gcoord_seband,
       gcpos_seband, gcoord_nelhband, gcpos_nelhband, gcoord_frames,
```

```
     gflowers_list, gpos_coord_frames, gframes_pic, gfselect,
     gnum_fr, gsound_list, gsound_number, gpathname,
     gBusy_spots_list, gHighlights_list, gFree_spots_list,
     gopened_hand, gclosed_hand, gpuppet_list, gScale_Coord,
     ggo_start, gcoord_mmap_parts, gpos_curs_mus, gmusic_volume,
     gmusic_volume_prev, gpos_curs_mus_prev, gpos_curs_sound,
     gsound_volume, gsound_volume_prev, gpos_curs_sound_prev,
     gvolume_cursor_music, gvolume_cursor_sound

cursor 4

set gpos_curs_mus to 2
set gmusic_volume to 100
set gmusic_volume_prev to gmusic_volume
set gpos_curs_mus_prev to gpos_curs_mus
set gpos_curs_sound to 2
set gsound_volume to 100
set gsound_volume_prev to gsound_volume
set gpos_curs_sound_prev to gpos_curs_sound

set the soundlevel to 7
set gpathname to the pathname&"Sounds&Music:"
set gmusic_name to "Music_bg"
set the volume of sound 2 to 200
set the volume of sound 3 to gsound_volume
set gsound_list =
     list("Bird1","Bird2","Bird3","Bird4","Bird5","Bird6","Bird7","Bird8",
     "Bird9","Bird10","Bird11","Bird12","Bird13","Bird14")
set gsound_number to 14
sound playfile 2, gpathname&"Music_intro"
repeat while not soundbusy(2)
  NOTHING
end repeat

go to frame 2

set gnum_fr to 0
set gfselect to 0
set gview_ne to 1
set gview_bs to 1
set gview_mi to 1
set gview_pl to 1
set gview_se to 1
set gview_nwl to 1
set gview_nelh to 1
set gview_neuh to 1

set ggo_start to 0

set gflowername_lat to ""
set gflowername_sw to ""

put " " into cast "Flower name latin"
put " " into cast "Flower name swedish"
put " " into cast "Help-message"
put " " into cast "Description"
put " " into cast "Number of flowers"
put " " into cast "Month name"
put " " into cast "Desc Headlines"

set the textfont of cast "Help-message" to "Times"
set the forecolor of cast "Help-message" to 0
set the textsize of cast "Help-message" to 13
set the textstyle of cast "Help-message" to "Plain"
set the textfont of cast "Flower name latin" to "Geneva"
set the textsize of cast "Flower name latin" to 10
set the forecolor of cast "Flower name latin" to 0
set the textfont of cast "Flower name swedish" to "Geneva"
set the textsize of cast "Flower name swedish" to 10
set the forecolor of cast "Flower name swedish" to 0
set the textfont of cast "Description" to "Times"
set the forecolor of cast "Description" to 0
```

```
set the textsize of cast "Description" to 20
set the textfont of cast "Number of flowers" to "Times"
set the textstyle of cast "Number of flowers" to "Bold"
set the forecolor of cast "Number of flowers" to 0
set the textsize of cast "Number of flowers" to 14
set the textfont of cast "Desc Headlines" to "Times"
set the forecolor of cast "Desc Headlines" to 0
set the textsize of cast "Desc Headlines" to 28
set the textfont of cast "Month name" to "Times"
set the textstyle of cast "Month name" to "Plain"
set the forecolor of cast "Month name" to 0
set the textsize of cast "Month name" to 30

set gpuppet_list=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
set gflowers_desc = [:]
set gcoord_necorner =
    list(10,10,22,22,53,6,65,18,17,26,29,38,52,30,64,42,42,56,54,68,21,46,3
        3,58)
set gcpos_necorner =
    ["Fl1/12":1,"Fl1/7":2,"Fl1/8":3,"Fl1/6":4,"Fl1/2":5,"Fl1/4":6]
set gcoord_sband =
    list(16,1,38,16,27,15,38,28,26,24,37,36,161,23,171,33,86,24,98,38,31,
    33,104,44,127,31,140,44,99,24,111,38,103,31,117,44,110,1,122,10,115,
    31,128,45,131,2,165,17,148,18,164,32,138,30,173,44)
set gcpos_sband =
    ["Fl3/16":1,"Fl3/15":2,"Fl3/1":3,"Fl3/9":6,"Fl3/14":8,"Fl3/13":9,
        "Fl3/22":10,"Fl3/11":11,"Fl3/5":12,"Fl3/3":13,"Fl3/7":14]
set gcoord_miband = list(4,5,34,34,80,2,109,32,52,1,64,17,63,1,75,16)
set gcpos_miband = ["Fl4/6":1,"Fl4/3":3,"Fl4/5":4]
set gcoord_plband = list(10,2,22,16,11,20,23,33,63,31,75,44,87,23,101,36)
set gcpos_plband = ["Fl6/5":1,"Fl6/6":2,"Fl6/1":3,"Fl6/7":4]
set gcoord_seband = list(1,1,8,9,26,10,38,24,35,2,52,16,30,22,42,35)
set gcpos_seband = ["Fl5/6":1,"Fl5/2":2,"Fl5/1":3,"Fl5/4":4]
set gcoord_nelhband = list(66,6,82,23,78,1,98,20)
set gcpos_nelhband = ["Fl2/2":1,"Fl2/1":2]
set gcoord_neuhband = list(83,1,97,14)
set gcpos_neuhband = ["Fl8/2":1]

set gpart_name_list =
    list("Mask_ne_corner","Mask_s_band","Mask_m_part","Mask_ne_lhband",
    "Mask_plat_part","Mask_se_part","Mask_nw_lband", "Mask_ne_uhband")
set gnecorner_name_list = list ("Fl1/12","Fl1/7","Fl1/8","Fl1/6","Fl1/2",
                                "Fl1/4")
set gsband_name_list = list ("Fl3/16","Fl3/15","Fl3/1.1","Fl3/9.1",
                        "Fl3/1.2","Fl3/14","Fl3/13","Fl3/22",
                "Fl3/11", "Fl3/9.2","Fl3/5","Fl3/7",
        "Fl3/3", "Fl3/1.3")
set gmiband_name_list = list ("Fl4/6.1","Fl4/3","Fl4/5","Fl4/6.2")
set gplband_name_list = list ("Fl6/5","Fl6/6","Fl6/1","Fl6/7")
set gseband_name_list = list ("Fl5/6","Fl5/2","Fl5/4","Fl5/1")
set gnelhband_name_list = list ("Fl2/2","Fl2/1")
set gneuhband_name_list = list ("Fl8/2")

set gframes_pic =
    ["Fl3/5":4,"Fl3/15":6,"Fl3/9":0,"Fl3/3":0,"Fl3/14":0,"Fl3/16":0,"Fl3/
    13":3,"Fl3/22":0,"Fl3/1":0,"Fl3/11":0,"Fl3/7":0,"Fl1/4":0,"Fl1/8":0,
    "Fl1/12":0,"Fl1/6":0,"Fl1/7":5,"Fl1/2":4,"Fl5/6":0,"Fl5/2":0,
    "Fl5/4":0,"Fl5/1":4,"Fl4/6":0,"Fl4/5":2,"Fl4/3":4,"Fl2/1":0,"Fl2/2":3
    ,"Fl8/2":0,"Fl6/1":0,"Fl6/5":0,"Fl6/6":0,"Fl6/7":0]
set gflowers_list =
    ["Fl3/15_a":"F1","Fl3/15_b":"F2","Fl3/15_c":"F3","Fl3/15_d":"F4",
    "Fl3/15_e":"F5","Fl3/5_a":"F6","Fl3/5_b":"F7","Fl3/5_c":"F8",
    "Fl3/16":"F9","Fl3/1":"F10","Fl3/9":"F11","Fl3/14":"F12","Fl3/13_a":"
    F13","Fl3/22":"F14","Fl3/11":"F15","Fl3/7":"F16","Fl3/3":"F17",
    "Fl1/4":"F18","Fl1/8":"F19","Fl1/12":"F20","Fl1/6":"F21",
    "Fl1/7_a":"F22","Fl5/6":"F26","Fl5/2":"F25",
    "Fl5/4":"F24","Fl5/1_a":"F27","Fl5/1_b":"F28","Fl5/1_c":"F29",
    "Fl5/1_d":"F30","Fl4/6":"F31","Fl4/5_a":"F32","Fl4/5_b":"F33",
    "Fl4/3_a":"F34","Fl4/3_b":"F35","Fl4/3_c":"F36","Fl4/3_d":"F14",
    "Fl2/1":"F38","Fl2/2_a":"F39","Fl2/2_b":"F40","Fl2/2_c":"F38",
    "Fl8/2":"F41","Fl3/5_d":"F42","Fl3/15_f":"F43","Fl1/7_b":"F33",
```

```
     "Fl1/7_c":"F44","Fl1/7_d":"F45","Fl1/7_e":"F46","Fl1/2_a":"F5",
     "Fl1/2_b":"F18","Fl1/2_c":"F35","Fl1/2_d":"F19","Fl6/1":"F47",
     "Fl6/6":"F48","Fl6/7":"F49","Fl6/5":"F50","Fl3/13_b":"F12",
     "Fl3/13_c":"F11"]
set gpos_coord_frames =
     ["Fl3/15_a":1,"Fl3/15_b":2,"Fl3/15_c":5,"Fl3/15_d":4,"Fl3/15_e":3,
     "Fl3/5_a":6,"Fl3/5_b":7,"Fl3/5_c":8,"Fl5/1_a":9,"Fl5/1_b":10,
     "Fl5/1_c":11,"Fl5/1_d":12,"Fl4/3_a":13,"Fl4/3_b":14,"Fl4/3_c":15,
     "Fl4/3_d":16,"Fl4/5_a":17,"Fl4/5_b":18,"Fl2/2_a":19,"Fl2/2_b":20,
     "Fl2/2_c":21,"Fl3/5_d":22,"Fl3/15_f":23,"Fl1/7_a":24,"Fl1/7_b":25,
     "Fl1/7_c":26,"Fl1/7_d":27,"Fl1/7_e":28,"Fl1/2_a":29,"Fl1/2_b":30,
     "Fl1/2_c":31,"Fl1/2_d":32,"Fl3/13_a":33,"Fl3/13_b":34,"Fl3/13_c":35]
set gcoord_frames =
     [121,134,258,243,253,81,393,191,103,194,129,228,292,14,387,79,
     199,78,234,110,108,171,293,232,366,107,422,156,69,36,230,130,
     196,42,339,210,88,115,199,252,17,179,101,265,343,7,426,133,
     197,159,342,255,102,73,254,169,41,12,181,69,283,29,377,130,
     136,114,299,270,131,13,308,114,163,95,277,260,98,15,156,132,
     252,15,354,131,259,28,357,159,209,237,415,284,161,33,295,131,
     84,55,161,121,345,23,429,103,327,87,402,148,283,101,365,233,
     18,28,190,183,74,191,159,256,207,30,309,136,246,164,338,237,
     149,115,344,247,116,53,245,117,15,86,141,233]

set gcoord_mmap_parts =
     [77,7,87,21,33,9,79,15,22,85,78,96,48,31,69,42,68,70,90,79,
     24,32,48,42,83,19,88,60,44,4,85,9]

set gmessage_list =
     list("","Main Menu: Keep the mouse button down to access it",¬
                          "Click on this part to make a zoom","Help Bar",¬
                          "Blossoming time",¬
                          "Sound indicator: sound on, click on it to
                    access the volume settings",¬
                          "Music indicator: music on, click on it to
                    access the volume settings",¬
                          "Cancel","Quit the application","Turn the music
                    on",¬
                          "Turn the music off","Turn the sound on","Turn
                    the sound off",¬
                          "Turn the help on","Turn the help off","Back to
                    the main map",¬
                          "Change the background view","Close the display
                    window",¬
                          "Display the latin name of the selected
                    flower",¬
                          "Speak the latin name of the selected flower",¬
                          "Previous area","Next area","Information about
                    the selected flower",¬
                          "Picture of the area","Click on this flower to
                    select it",¬
                          "No more background view","Dragable Start
                    Flag",¬
                          "Close the error window","Definitely quit the
                    application",¬
                          "Don't quit the application","Reset the time
                    scale",¬
                          "Search for flowers with blossoming time within
                    the considered period",¬
                          "The time scale where you can place the Flags",¬
                          "No next flower",¬
                          "No previous flower","Dragable End Flag",¬
                          "The month during which the displayed flower is
                    blossoming",¬
                          "Map of the garden",¬
                          "The part of the garden where the displayed
                    flower is located",¬
                          "Display the latin name of the displayed
                    flower",¬
                          "Speak the latin name of the displayed flower",¬
                          "Previous flower","Next flower",¬
                          "Information about the displayed flower",¬
                          "Picture of the displayed flower",¬
```

```
                    "Map of this part of the garden",¬
                    "Spot of this part of the garden where the
displayed area is located",¬
                    "Number of flowers available on this picture",¬
                    "Selected period of time",¬
                    "No next area","No previous area",¬
                    "Display the swedish name of the selected
flower",¬
                    "Speak the swedish name of the selected
flower",¬
                    "Display the swedish name of the displayed
flower",¬
                    "Speak the swedish name of the displayed
flower",¬
                    "Drag the cursor up or down to set the volume",¬
                    "Accept the changes made to the volume",¬
                    "Cancel the changes made to the volume",¬
                    "Hide the latin name of the selected flower",¬
                    "Hide the swedish name of the selected flower",¬
                    "Hide the latin name of the displayed flower",¬
                    "Hide the swedish name of the displayed flower")

set gBusy_spots_list = list (0,0,0,0,0,0,0,0,0,0,0,0,0)
set gHighlights_list = list (0,0,0,0,0,0,0,0,0,0,0,0,0)
set gFree_spots_list = list (0,0,0,0,0,0,0,0,0,0,0,0,0)
set gScale_Coord  = list
          (3,51,99,147,195,243,291,339,387,435,483,531,579)

set lens to the number of cast "Lens"
set lens_mask to the number of cast "Mask Lens"
set menu to the number of cast "Menu-curs"
set menu_mask to the number of cast "Mask Menu"
set opened_hand to the number of cast "Opened hand"
set opened_hand_mask to the number of cast "Opened hand mask"
set closed_hand to the number of cast "Closed hand"
set closed_hand_mask to the number of cast "Closed hand mask"
set finger to the number of cast "Finger"
set finger_mask to the number of cast "Finger mask"
set volume_cursor_music to the number of cast "Volume cursor music"
set volume_cursor_music_mask to the number of cast "Volume cursor music
    mask"
set volume_cursor_sound to the number of cast "Volume cursor sound"
set volume_cursor_sound_mask to the number of cast "Volume cursor sound
    mask"

set gvolume_cursor_music to
    [volume_cursor_music,volume_cursor_music_mask]
set gvolume_cursor_sound to
    [volume_cursor_sound,volume_cursor_sound_mask]
set gfinger to [finger,finger_mask]
set gclosed_hand to [closed_hand,closed_hand_mask]
set gopened_hand to [opened_hand,opened_hand_mask]
set glens to [lens,lens_mask]
set gmenu to [menu,menu_mask]

set gsound to true
set gmusic to true
set ghelp to true

repeat with n = 1 to 48
  set the visibility of sprite n to true
end repeat

set the visibility of sprite 28 to gmusic
set the visibility of sprite 27 to gsound
set the visibility of sprite 30 to ghelp
set the visibility of sprite 31 to gsound
set the visibility of sprite 32 to gmusic
set the visibility of sprite 45 to false
set the visibility of sprite 46 to false
set the visibility of sprite 47 to false
set the visibility of sprite 48 to false
```

```
  read_flowersfile("flowers_english.desc",gflowers_list)
  List_of_Months

  repeat while soundbusy(2)
    NOTHING
  end repeat
  set the soundlevel to 5
  set the volume of sound 2 to gmusic_volume

  set ggo_start to 1
end
```

453 : switch button

```
on switch_button button_name_st, button_name_end, noname, channel

  -- mouse down sound --
  click_1

  -- place the button ( necessary because of a bug from director ) --
  set name to button_name_st&"_dw"
  set c1 to the left of sprite channel
  set c2 to the top of sprite channel
  set c3 to the right of sprite channel
  set c4 to the bottom of sprite channel
  set the castnum of sprite channel to the number of cast name
  spritebox channel, c1, c2, c3, c4
  updatestage

  -- test if the button is really pressed by the user, it means that the     user
has the --
  -- possinility to change his mind by releasing the mouse button in   another
area than the --
  -- one occupied by the button --

  set ok to true
  repeat while the mousedown
    if rollover(channel) then
      set ok to true
      set name to button_name_st&"_dw"
      set the castnum of sprite channel to the number of cast name
      spritebox channel, c1, c2, c3, c4
      updatestage
    else
      set ok to false
      set name to button_name_st&"_up"
      set the castnum of sprite channel to the number of cast name
      spritebox channel, c1, c2, c3, c4
      updatestage
    end if
  end repeat

  if ok then
    -- mouse up sound --
    click_2

    if noname then
      set name to button_name_st&"_up"
    else
      set name to button_name_end&"_up"
    end if

    set the castnum of sprite channel to the number of cast name
    spritebox channel, c1, c2, c3, c4
    cursor 4
    updatestage
    set complete to true
  else
    set complete to false
  end if
```

```
  -- complete equals 1 if the button is fully pressed, equals 0 otherwise--
  return complete

end
```

454 : take words

```
on take_words li

  set end to false
  set wo to ""
  set result to ""
  set n to 1
  set space to 0
  set wo_num to 0

  repeat while not end and n<=length(li)
    set ch to char n of li
    if ch=" " then set space to space + 1
    else set space to 0
    if space = 0 then
      set n to n + 1
    else
      if space = 1 then
        set wo_num to wo_num + 1
        set n to n + 1
      else
        set end to true
      end if
    end if
  end repeat
  if space = 0 then set wo_num to wo_num + 1
  set result to word 1 to wo_num of li

  return result

end
```

455 : Test music

```
on test_music
  global gmusic, gmusic_name, gpathname

  if gmusic then
    if not soundbusy(2) then sound playfile 2, gpathname & gmusic_name
  end if

end
```

456 : Test seasons

```
on test_seasons
  global gcurrent_list,gJanuary_list, gFebruary_list,
       gMarch_list,gApril_list, gMay_list ,gJune_list, gJuly_list,
       gAugustus_list, gSeptember_list, gOctober_list, gNovember_list,
       gDecember_list, gseason

  if gcurrent_list=gjanuary_list or gcurrent_list=gfebruary_list or¬
              gcurrent_list=gmarch_list then
    set result to "SF_Winter"
  else
    if gcurrent_list=gapril_list or gcurrent_list=gmay_list or¬
                gcurrent_list=gjune_list then
      set result to "SF_Spring"
    else
      if gcurrent_list=gjuly_list or gcurrent_list=gaugustus_list or¬
                  gcurrent_list=gseptember_list then
        set result to "SF_Summer"
```

```
      else
        set result to "SF_Fall"
      end if
    end if
  end if
  if gseason = result then
    set change to 0
  else
    set gseason to result
    set change to 1
  end if
  return change
end
```

457 : Test_sound

```
on test_sound
  global gsound, gsound_list, gsound_number, gpathname

  if gsound then
    if random(60)=1 then
      if not soundbusy(3) then
        set sound_name to getat(gsound_list,random(gsound_number))
        sound playfile 3, gpathname & sound_name
      end if
    end if
  end if

end
```

458 : Time scale management

```
on CALCCOMPTEUR

  global Compteur , gBusy_spots_list ,Empty_scale

  set i to 1
  set Compteur to 0
  repeat while ( i < 14 )
    if ( GetAt ( gBusy_spots_list , i ) < 14 ) and ( GetAt (
       gBusy_spots_list , i ) <> 0 ) then
      set Compteur to Compteur + 1
      set Empty_scale to false
    else
      if ( GetAt ( gBusy_spots_list , i ) <> 0 ) then
        set Compteur to Compteur - 1
      end if
    end if
    set i to i+1
  end repeat

end

on POSLIBRE Vari

  Global gFree_spots_list , gBusy_spots_list , gHighlights_list

  set gFree_spots_list = list (0,0,0,0,0,0,0,0,0,0,0,0,0,0)

  set i to 1

  repeat while ( i < 14 )
    if ( GetAt ( gBusy_spots_list , i ) <> 0 ) then
      SetAt ( gFree_spots_list , i ,1 )
    end if
    set i to i+1
  end repeat
```

```
  if ( Vari < 14 ) then
    set i to 1
    SetAt ( gFree_spots_list , 13 , 1 )

    repeat while ( i < 14 )
      if ( GetAt ( gBusy_spots_list , i ) <> 0 ) and ( GetAt (
       gBusy_spots_list , i ) < 14 ) then
        set j to 1
        SetAt ( gFree_spots_list , i , 1 )

        if ( i = 1 ) then
          SetAt ( gFree_spots_list , 12 , 1 )
        else
          SetAt ( gFree_spots_list , i-1 , 1 )
        end if

      end if
      set i to i+1
    end repeat

  else
    set i to 1

    repeat while ( i < 14 )
      if ( GetAt ( gBusy_spots_list , i ) > 13 ) then
        SetAt ( gFree_spots_list , i , 1 )
        if i=1 then setAt ( gFree_spots_list , 13 , 1 )
        set Find2 to false
        set j to i+1

        repeat while ( j < 14 ) and ( Find2 = false )
          if ( GetAt ( gBusy_spots_list , j ) = 0 ) then
            SetAt ( gFree_spots_list , j , 1 )
            set j to j+1
          else
            set Find2 to true
          end if
        end repeat


        if ( Find2 = false ) then
          set j to 1

          repeat while ( j < 14 ) and ( Find2 = false )
            if ( GetAt ( gBusy_spots_list , j ) = 0 ) then
              SetAt ( gFree_spots_list , j , 1 )
              set j to j+1
            else
              set Find2 to true
            end if
          end repeat

        end if
      end if
      set i to i+1
    end repeat
  end if

end

on TROUVEDRAP Vari

  Global gBusy_spots_list

  set i to 1
  set Trouve to false

  repeat while ( i < 14 )
    if ( Getat ( gBusy_spots_list , i ) = Vari ) then
      set j to i
      set i to 15
    end if
```

```
    set i to i+1
  end repeat

  set j to j-1
  repeat while ( j > 0 ) and ( Trouve = false )
    if ( Getat ( gBusy_spots_list , j ) < 14 ) and ( Getat (
       gBusy_spots_list , j ) <> 0 ) then

      DESURLIGNEMENT j
      set Trouve to true
    else
      if ( Getat ( gBusy_spots_list , j ) > 13 ) then
        set Trouve = true
      else
        set j to j-1
      end if
    end if

  end repeat

  if ( Trouve = false ) then
    set k to 12

    repeat while ( Trouve = false ) and ( k > 0 )
      if ( Getat ( gBusy_spots_list , k ) < 14 ) and ( Getat (
       gBusy_spots_list , k ) <> 0 ) then
        DESURLIGNEMENT k
        set Trouve to true
      else
        if ( Getat ( gBusy_spots_list , k ) > 13 ) then
          set Trouve to true
        else
          set k to k-1
        end if
      end if
    end repeat

  end if

end
```

459 : Total

```
on total months_list

  set result to 0
  repeat with n = 1 to count(months_list)
    set month to getat(months_list,n)
    convert_month(month)
    set result to result + count(the result)
  end repeat

  return result
end
```

460 : User choice management

```
on MONTHS me
  global Mon

  if ( me = 1 ) then
    set Mon to "January"
  else
    if ( me = 2 ) then
      set Mon to "February"
    else
      if ( me = 3 ) then
        set Mon to "March"
      else
        if ( me = 4 ) then
```

```
          set Mon to "April"
        else
          if ( me = 5 ) then
            set Mon to "May"
          else
            if ( me = 6 ) then
              set Mon to "June"
            else
              if ( me = 7 ) then
                set Mon to "July"
              else
                if ( me = 8 ) then
                  set Mon to "Augustus"
                else
                  if ( me = 9 ) then
                    set mon to "September"
                  else
                    if ( me = 10 ) then
                      set Mon to "October"
                    else
                      if ( me = 11 ) then
                        set Mon to "November"
                      else
                        set Mon to "December"
                      end if
                    end if
                  end if
                end if
              end if
            end if
          end if
        end if
      end if
    end if
  end if
end

On user_choice
  global gbusy_spots_list , Mon , gmonths_list


  set gmonths_list = [0]
  set Mon to ""
  set i to 1
  repeat while ( i < 14 )
    if ( GetAt ( gbusy_spots_list , i ) < 14 ) and ( GetAt (
       gbusy_spots_list , i ) <> 0 ) then
      MONTHS i
      Append gmonths_list , Mon
      set Trouve to false
      set i to i+1

      repeat while ( Trouve = false ) and ( i < 14 )
        if ( GetAt ( gbusy_spots_list , i ) = 0 ) then
          MONTHS i
          Append gmonths_list , Mon
        else
          set Trouve to true
        end if
        set i to i+1
      end repeat

    else
      set i to i+1
    end if

  end repeat

  if ( Trouve = false ) then
    set k to 1

    repeat while ( Trouve = false )
```

```
      if ( GetAt ( gbusy_spots_list , k ) = 0 ) then
        MONTHS k
        Append gmonths_list , Mon
      else
        set trouve to true
      end if
      set k to k+1
    end repeat
  end if

  DeleteAt gmonths_list , 1
end
```

461 : Volume settings call

```
on volume_settings type
  global gpuppet_list

  set v to [0,0,0,0,0]

  repeat with n=1 to 48
    if n<44 then
      if n<>37 then
        setat gpuppet_list, n, the puppet of sprite n
        puppetsprite n, true
      end if
    else
      setat v, n-43 , the visibility of sprite n
      set the visibility of sprite n to 1
    end if
  end repeat

  play frame type

  repeat with n=1 to 48
    if n<44 then
      if n<>37 then
        puppetsprite n, getat (gpuppet_list, n)
      end if
    else
      set the visibility of sprite n to getat (v,n-43)
    end if
  end repeat
end
```

462 : Volume sound

```
on volume
  puppetsound "Volume_sound"
  updatestage
end
```

463

```
on enterFrame
  repeat with n = 45 to 47
    set the visibility of sprite n to 0
  end repeat

  set the visibility of sprite 48 to 0
  set the visibility of sprite 38 to 0
  set the visibility of sprite 39 to 0

  repeat with n = 11 to 16
    set the visibility of sprite n to 0
  end repeat

end
```

464

```
on enterFrame
  repeat with n = 45 to 48
    set the visibility of sprite n to 0
  end repeat

  set the visibility of sprite 38 to 0
  set the visibility of sprite 39 to 0

  repeat with n = 11 to 24
    set the visibility of sprite n to 0
  end repeat
end
```

465

```
on enterFrame
  repeat with n = 45 to 47
    set the visibility of sprite n to 0
  end repeat

  set the visibility of sprite 48 to 0
  set the visibility of sprite 38 to 0
  set the visibility of sprite 39 to 0

  repeat with n = 11 to 14
    set the visibility of sprite n to 0
  end repeat

end
```

466

```
on enterFrame
  repeat with n = 45 to 47
    set the visibility of sprite n to 0
  end repeat

  set the visibility of sprite 48 to 0
  set the visibility of sprite 38 to 0
  set the visibility of sprite 39 to 0

  repeat with n = 11 to 14
    set the visibility of sprite n to 0
  end repeat

end
```

467

```
on enterFrame
  repeat with n = 45 to 47
    set the visibility of sprite n to 0
  end repeat

  set the visibility of sprite 48 to 0
  set the visibility of sprite 38 to 0
  set the visibility of sprite 39 to 0

  repeat with n = 11 to 14
    set the visibility of sprite n to 0
  end repeat

end
```

468

```
on enterFrame
  repeat with n = 45 to 47
    set the visibility of sprite n to 0
  end repeat

  set the visibility of sprite 48 to 0
  set the visibility of sprite 38 to 0
  set the visibility of sprite 39 to 0

end
```

469

```
on enterFrame
  repeat with n = 45 to 47
    set the visibility of sprite n to 0
  end repeat

  set the visibility of sprite 48 to 0
  set the visibility of sprite 38 to 0
  set the visibility of sprite 39 to 0

  repeat with n = 11 to 12
    set the visibility of sprite n to 0
  end repeat

end
```

470

```
on decode string

  set n to 1
  set part1 to ""
  set part2 to ""
  repeat while n<= length(string)
    set ch to char n of string
    if ch ="-" then
      if n<>1 then
        transf_num_month(part2)
        set part1 to the result&ch
        set part2 to ""
      else
        set part1 to ch
      end if

    else
      set part2 to part2&ch
    end if
    set n to n+1
  end repeat

  if part2 <> "" then
    transf_num_month(part2)
    set result to part1&the result
  else
    set result to part1
  end if

  return result
end
on transf_num_month me

  if ( me = 1 ) then
    set Mon to "January"
  else
    if ( me = 2 ) then
      set Mon to "February"
```

```
    else
      if ( me = 3 ) then
        set Mon to "March"
      else
        if ( me = 4 ) then
          set Mon to "April"
        else
          if ( me = 5 ) then
            set Mon to "May"
          else
            if ( me = 6 ) then
              set Mon to "June"
            else
              if ( me = 7 ) then
                set Mon to "July"
              else
                if ( me = 8 ) then
                  set Mon to "Augustus"
                else
                  if ( me = 9 ) then
                    set mon to "September"
                  else
                    if ( me = 10 ) then
                      set Mon to "October"
                    else
                      if ( me = 11 ) then
                        set Mon to "November"
                      else
                        set Mon to "December"
                      end if
                    end if
                  end if
                end if
              end if
            end if
          end if
        end if
      end if
    end if

  return mon

end
```

471

```
on SFlag_management
  global gblue

  Start_Flag(gblue)

  POSITIONNEMENT(gblue)
end

on EFlag_management
  Global ggreen

  End_Flag(ggreen)

  POSITIONNEMENT(ggreen)
end
```

472

```
on EnterFrame

  repeat with n = 1 to 26
    PuppetSprite  n,true
  end repeat
```

```
end
```

473

```
on enterframe
  repeat with n = 45 to 47
    set the visibility of sprite n to 0
  end repeat

  set the visibility of sprite 48 to 0
  set the visibility of sprite 38 to 0
  set the visibility of sprite 39 to 0

  repeat with n = 11 to 11
    set the visibility of sprite n to 0
  end repeat
end
```

474

```
on exitFrame
  global ggo_start

  if ggo_start then go to frame "Start"
  else go to the frame
end
```

475

```
on exitFrame

  repeat with n = 1 to 13
    set the visibility of sprite n to 1
  end repeat

  repeat with n = 14 to 21
    set the visibility of sprite n to 0
  end repeat

  repeat with n = 45 to 47
    set the visibility of sprite n to 0
  end repeat

  repeat with n = 35 to 37
    set the visibility of sprite n to 1
  end repeat

  set the visibility of sprite 48 to 0
  set the visibility of sprite 38 to 0

end
```

476

```
on exitFrame
  puppetsprite 13, true
  puppetsprite 18, true
end
```

477

```
on mousedown
  repeat with n = 21 to 26
    set the Visible of sprite n to false
  end repeat
  repeat with n = 7 to 20
```

```
    set the visibility of sprite n to 1
  end repeat
  puppetsprite 4, false
  puppetsprite 5, false
  puppetsprite 16, false
  puppetsprite 17, false

  Back_to("Blossoming")
end
```

478

```
on mousedown
  puppetsprite 16, false
  puppetsprite 17, false
  Info("DF_Winter")
end
```

479

```
on mousedown
  puppetsprite 16, false
  puppetsprite 17, false
  Info("DF_Spring")
end
```

480

```
on mousedown
  puppetsprite 16, false
  puppetsprite 17, false
  Info("DF_Summer")
end
```

481

```
on mousedown
  puppetsprite 16, false
  puppetsprite 17, false
  Info("DF_Fall")
end
```

482

```
on mousedown
  cursor_management("Music")
end
```

483

```
on exitFrame
  global gpos_curs_mus, gmusic_volume,, gmusic_volume_prev,
      gpos_curs_mus_prev

  puppetsprite 45, true

  set max to the top of sprite 44 + 49
  set lapse to 12
  set gpos_curs_mus to gpos_curs_mus_prev
  set gmusic_volume to gmusic_volume_prev
  set the loch of sprite 45 to the left of sprite 44 + 128
  set the locv of sprite 45 to gpos_curs_mus*lapse + max
```

```
  set the blend of sprite 45 to 90

end
```

484

```
on mousedown
  cursor 4

  -- volume settings sound --
  volume

  volume_settings("Music")
end
```

485

```
on exitFrame
  global gpos_curs_sound, gsound_volume, gsound_volume_prev,
      gpos_curs_sound_prev

  puppetsprite 45, true

  set max to the top of sprite 44 + 49
  set lapse to 12
  set gpos_curs_sound to gpos_curs_sound_prev
  set gsound_volume to gsound_volume_prev
  set the loch of sprite 45 to the left of sprite 44 + 128
  set the locv of sprite 45 to gpos_curs_sound*lapse + max
  set the blend of sprite 45 to 90

end
```

486

```
on mousedown
  cursor 4
  -- volume settings sound --
  volume

  volume_settings("Sound")
end
```

487

```
on mousedown
  cursor_management("Sound")
end
```

488

```
on delay number
  starttimer
  repeat while the timer < number*60
    nothing
  end repeat
end
```