THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

L'apprentissage par l'exemple et à distance de la programmation d'IHM sur appareils mobiles

Van Tongelen, Sophie

Award date:
2005

Institut d'Informatique
Facultés Universitaires Notre Dame de la Paix
Namur, Belgique

L'apprentissage par l'exemple et à distance de la programmation d'IHM sur appareils mobiles

Sophie Van Tongelen

Mémoire présenté pour obtenir le diplôme de Maître en Informatique
Année Académique 2004-2005

Résumé

Pour concevoir une formation à distance sur le développement d'une interface graphique pour appareils mobiles, il est nécessaire de prendre en compte plusieurs difficultés pouvant survenir. Tout d'abord, il y a les problèmes d'autonomie et de motivation, le problème de l'accompagnement et la question du développement des dispositifs. Ces problèmes, auxquels il faut ajouter la détermination du rôle des technologies dans la formation, se pose au niveau de la méthode d'apprentissage. D'autres questions surviennent aussi aux niveaux de la programmation et de la conception d'une interface.

L'apprentissage d'un langage de programmation dans le but spécifique de concevoir une interface rencontre certains problèmes propres à l'apprentissage de la programmation. Le but est de "faire faire" une tâche à un ordinateur-exécutant, ce qui pose certains problèmes d'abstraction et de visualisation de la tâche et de l'ordinateur.

L'importance d'appliquer les principes d'utilisabilité et d'ergonomie se voit amplifiée pour des appareils dotés de petits écrans. Ces appareils ne s'utilisent pas dans le même contexte que des postes fixes et ont donc des besoins spécifiques en terme d'interface. Pour concevoir une interface graphique mobile utilisable, plusieurs principes doivent être appliqués.

Ce mémoire propose un tutoriel basé sur l'apprentissage par les exemples dans une formation à distance sans tutorat. La version proposée est un format dit "papier", c'est-à-dire, imprimable accompagné des fichiers des exemples expliqués pas à pas.

Abstract

To conceive a distance training about the development of a graphical user interface specially for mobile devices, it is necessary to consider many difficulties that could happen. First there are the problems of distance self-learning like the autonomy and the motivation, the follow-up, and the technologies part in the learning. There are others questions, about programming and about the interface design.

The programming language learning in order to design an interface have same problems than in programming learning. The objective is "having a task done by the computer" and it is a reason of some abstraction problems and task and computer visualizations problems.

It is important to put usability and ergonomy principles into practice because of the problems amplification on small-screen devices. Users don't use small-screen devices in the same context than a workstation. So handheld devices need specific interfaces.

The thesis proposes a tutorial based on learning by examples as part of distance training without any tutoring. The proposed version is printable because of its format. It contains step-by-step examples.

Je voudrais remercier particulièrement ma promotrice de mémoire, Monique Noirhomme-Fraiture, pour le suivi et les précieux commentaires donnés lors de l'écriture de ce mémoire.

Je voudrais également remercier mon maître de stage, Julio Abascal, pour son accueil et le suivi durant mon séjour à San Sebastian. Je voudrais remercier également l'équipe du laboratoire pour l'accueil et l'aide que tous m'ont apportée.

Je voudrais remercier Geof, Aurélie et Pierre pour leur aide ainsi que mes parents et Nathalie pour leur soutien.

Table des matières

Table des figures

Table des acronymes

API Application Programming Interface
AWT Abstract Window Toolkit
CDC Connected Device Configuration
CLDC Connected Limited Device Configuration
FAD Formation à Distance
FAQ Foire aux Questions
GPS Global Positioning System
GUI Graphical User Interface
IHM Interface Homme-Machine
J2EE Java 2 Enterprise Edition
J2ME Java 2 Micro Edition
J2SE Java 2 Standard Edition
MIDP Mobile Information Device Profile
OS Operating System
PDA Personal Digital Assistant
PSE Programmation sur Exemples
SMS Short Message System
TIC Technologies d'Information et de Communication
WAP Wireless Application Protocol

Introduction

De nos jours, une multitude d'appareils mobiles dotés d'écrans de taille réduite a fait son apparition : téléphones mobiles, smartphones, ou encore assistants personnels. Les utilisateurs de ces appareils sont très diverses et souvent peu spécialisés. Cependant, certains d'entre eux souhaitent développer des applications tournant sur leur téléphone mobile ou leur assistant personnel. Ils doivent alors faire face à plusieurs problèmes.

L'objectif principal du stage était de fournir un document aux utilisateurs d'appareils mobiles afin de les aider à développer des interfaces graphiques destinées à être greffées sur des applications. Ces interfaces doivent être utilisables, à cause des problèmes d'ergonomie particuliers à ces appareils, et simples à développer. Le projet était d'établir un tutoriel permettant aux utilisateurs-développeurs débutants de créer une interface graphique rapidement, sans que ça ne leur demande trop d'efforts tels que le temps et la concentration nécessaires au processus d'apprentissage. Ce tutoriel devait se faire dans le cadre d'une formation à distance sans tutorat et il devait être basé sur la méthode de l'apprentissage par les exemples. Le projet pose les problématiques liées à la conception d'interfaces graphiques sur les appareils mobiles et à son apprentissage.

Le mémoire est composé de cinq chapitres. Le premier introduit la théorie de l'apprentissage à distance par l'exemple. Plusieurs aspects sont abordés. Tout d'abord, quelques notions sur l'apprentissage en général sont introduits. La deuxième partie de ce chapitre expose les enjeux, la problématique et le rôle des technologies dans l'apprentissage à distance. Elle propose également une typologie des enseignements selon le degré d'intégration de ces technologies. Le dernier point concerne l'apprentissage de la programmation, sa problématique et son apprentissage par les exemples.

Le deuxième chapitre porte sur les problèmes spécifiques à la conception d'IHM pour appareils dotés de petits écrans, ainsi que quelques principes à appliquer lors de la création d'interfaces.

Le troisième chapitre décrit l'architecture générale de la technologie faisant l'objet du tutoriel, J2ME. Il s'attarde un peu plus sur le profil MIDP, qui est enseigné.

Le quatrième chapitre présente la problématique de tout processus d'enseignement, c'est-à-dire les questions à poser pour concevoir un programme de formation, et son application au cas du tutoriel. Il décrit comment le tutoriel répond à ces questions ainsi que sa structure et chaque module en faisant partie. Il termine par une critique sur les manques

et les erreurs.

Le cinquième chapitre expose des améliorations possibles à amener au tutoriel. Il est composé de deux parties. La première parle de la version existante au format papier et des améliorations pouvant y être apportées. La deuxième porte sur la création d'un site Internet. Un changement de support qui apporte des avantages certains par rapport à la version papier.

Chapitre 1

Etat de l'art - L'Apprentissage à distance à partir d'exemples

1.1 Principes de base

1.1.1 Définitions

Il existe une multitude de définitions de l'apprentissage dépendantes de la théorie sur laquelle on se base. Cependant, toutes théories confondues, l'apprentissage implique toujours un changement chez l'apprenant. Ce changement se manifeste dans ses connaissances mais également dans son attitude, son comportement et ses croyances.

"L'acquisition de connaissances par le texte est une modification du contenu de la mémoire à long terme : elle se traduit par une modification des structures de connaissances : réseaux sémantiques et schémas." [RIC90]

Ce changement dans les connaissances, habiletés et attitudes de l'apprenant entraîne l'acquisition de compétences et d'un certain niveau de performances.

1.1.2 Théories

Il existe plusieurs théories de l'apprentissage, trois sont exposées ci-après : la théorie comportementaliste (ou behaviorisme) de Skinner, la théorie cognitiviste et la théorie constructiviste de Piaget.

Théorie comportementaliste Le comportementalisme voit l'apprentissage comme un processus d'adaptation du comportement aux modifications de l'environnement ; c'est une "boîte noire". Apprendre se ramène à recevoir des stimulus de l'environnement et à y répondre. L'enseignant peut diriger ce processus en sélectionnant des stimuli et en récompensant ou renforçant les réponses jugées adéquates (et en décourageant les "mauvaises" réponses). L'apprenant a un rôle plutôt passif.

Cette théorie a donné naissance à l'enseignement programmé, qui vise à automatiser le processus stimulus-réponse-renforcement. Cependant, ce type d'enseignement est tombé en disgrâce.

Théorie cognitiviste Au contraire des comportementalistes, les cognitivistes mettent en lumière les processus internes de l'apprentissage. L'apprenant est mis au centre de sa formation et est considéré comme actif dans son apprentissage. En effet,il traite les informations qui lui viennent du monde extérieur : il les reconnait, les mémorise et les restitue face à certaines situations. La réalité perçue est une réalité objective et l'apprenant doit l'intégrer à ses schémas mentaux. La réaction face à cette réalité est donc différente de la réaction décrite chez les comportementalistes. L'apprentissage est défini comme un changement dans les structures mentales de l'étudiant.

La vision de l'éducation qui en découle met en avant l'importance d'un engagement mental actif des élèves durant l'apprentissage afin qu'ils puissent traiter les informations en profondeur et pas uniquement en surface.

Théorie constructiviste Pour Piaget, l'apprentissage se manifeste grâce à une divergence entre la réalité perçue et sa représentation interne. C'est la grande différence avec l'approche cognitiviste, la réalité est, ici, subjective. L'apprenant cherche à réduire ce fossé et à structurer les connaissances enseignées. L'apprentissage est vu comme une activité de construction par l'individu dans un contexte social.

Le Logo, créé par Seymour Papert, est directement issu de la théorie constructiviste. Il a comme objectif d'"apprendre à apprendre". Contrairement à l'apprentissage programmé, le Logo a connu un très grand succès dans tous les pays et parmi tous les âges. Aujourd'hui, Logo continue de se développer avec Lego-Logo qui allie informatique et robotique.

La figure 1.1 présente les différences principales entre ces trois théories.

De ces théories, trois modèles d'apprentissage sont nés : le modèle dit *classique* qui privilégie la transmission du savoir par l'enseignant, le modèle dit *moderne* qui privilégie la construction des connaissances par l'étudiant et le modèle de l'*autoformation*, un modèle un peu à part.

Modèle classique Il concerne les cours traditionnels, magistraux. La qualité de l'enseignement est fonction des capacités de l'enseignant à faire comprendre les concepts et de sa maîtrise de la matière.

Les technologies utilisées comme ressources pédagogiques sont principalement un soutien à l'enseignant en tant qu'illustration de son cours. Celui-ci peut utiliser textes, images, films ou encore videoconférences. Ce modèle s'applique surtout dans le cadre de la théorie comportementaliste où l'enseignant joue un rôle central.

Modèle moderne Ce modèle vise la variété des modes d'acquisition et de transmission des connaissances ainsi que la participation active de l'élève. Il s'applique plutôt à une approche constructiviste où l'élève a un rôle central. La technologie est vue principalement comme une aide pour l'apprenant avec des possibilités de simulation, de création, de consultation ou encore d'évaluation des connaissances.

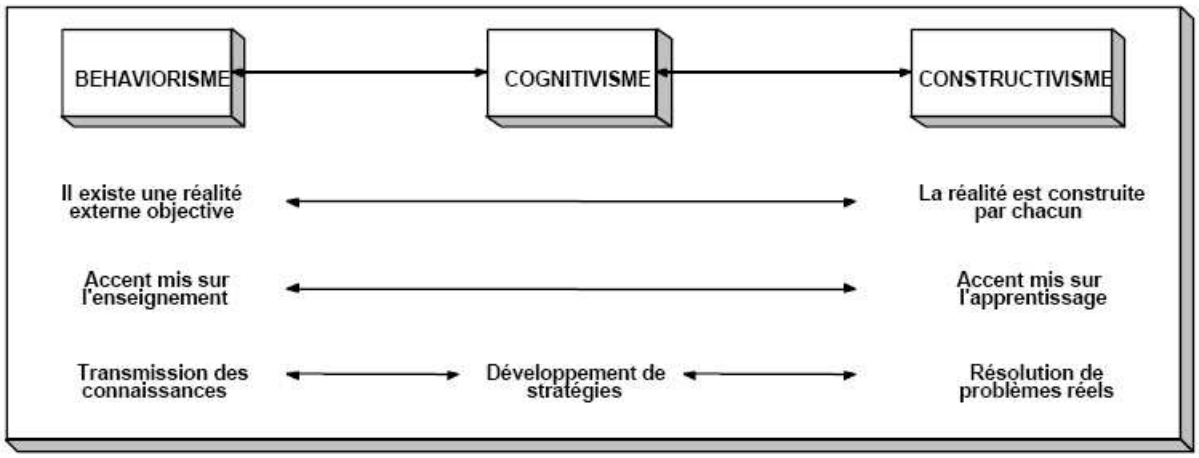


FIG. 1.1 – Evolution des approches théoriques de l'apprentissage

Modèle de l'auto-formation L'activité de l'apprenant est centrale et en ça, il se rapproche du modèle moderne, entre autres, au niveau de l'utilisation des outils technologiques. Par contre, il tend également vers un modèle classique par les ressources pédagogiques qui sont souvent une simulation du comportement de l'enseignant classique.

1.1.3 Connaissances, compétences et performances

Connaissances Représentations mentales qui permettent de se représenter des objets et des faits ou d'agir. On distingue deux types de connaissances :

- Les connaissances déclaratives (savoirs) sont descriptives et assez éloignées de l'activité concrète.
- Les connaissances procédurales (savoir-faire) forment les structures qui permettent d'agir. Elles peuvent être représentées par la formule SI-ALORS. Le SI représente le côté cognition et le ALORS le côté action.

Une personne peut avoir certaines connaissances déclaratives sur une activité sans, pour autant, avoir de compétences procédurales sur cette même activité, et inversement.

"Par analogie avec les langages informatiques, cette dichotomie résume la différence fonctionnelle entre le fait d'utiliser un langage pour décrire les relations entre les états d'un problème ou pour prescrire des transformations entre ces états."

Compétences Les compétences définissent les capacités d'une personne à accomplir une tâche donnée grâce à ses savoirs, savoirs-faire et savoirs-être.

On peut distinguer les compétences de la manières suivante :

- Les compétences de type reproduction permettent l'accomplissement de tâches

routinières et l'utilisation d'un agencement connu de savoirs, savoirs-faire et savoirs-être
– Les compétences du type production permettent l'accomplissement de tâches du type résolution de problème. L'individu doit découvrir l'ordre dans lequel utiliser ses savoirs pour accomplir la tâche.

L'enseignement sera adapté en fonction du type de compétences à acquérir. Si on veut accomplir des tâches routinières, on utilisera plutôt des démonstrations suivies d'exercices. Par contre, si on veut accomplir des tâches de type résolution de problème, on utilisera plutôt des études de cas. L'apprentissage de l'informatique étant du domaine de la résolution de problème, un enseignement adapté est un enseignement par les exemples.

Performances Lorsque la personne active ses compétences pour l'exécution d'une tâche, elle montre une certaine performance. On distingue les performances et les performances observables.

1.2 Apprentissage à distance

1.2.1 Définitions

L'apprentissage à distance fait référence à tout processus d'enseignement/apprentissage où l'enseignant et l'apprenant sont séparés par une distance physique. Le terme de formation à distance est un concept très large. Il peut se référer à un système de formation dynamique entre l'enseignant et l'apprenant mais il peut se référer également à un système où il n'existe aucun lien entre les deux protagonistes. La communication peut se faire au moyen de divers medias. On appelle e-learning l'apprentissage à distance lorsqu'il se fait via internet.

L'UNESCO propose la définition suivante : "Mode d'enseignement, dispensé par une institution, qui n'implique pas la présence physique du maître chargé de le donner à l'endroit où il est reçu, ou dans lequel le maître n'est présent qu'à certains moments ou pour des tâches spécifiques. Les communications enseignants-enseignés se font principalement par le recours à la correspondance, aux imprimés, aux divers médias audiovisuels, à l'informatique, à certains regroupements." [UNE87]

Toutes les définitions de l'enseignement à distance reconnaissent :
– une séparation spatiale et/ou temporelle entre apprenant et enseignant,
– un recours aux médias permettant l'apprentissage malgré cette séparation.

Le e-Learning est particulier dans l'apprentissage à distance car il est toujours intimement lié à Internet. Il permet d'accéder au contenu de la formation via un navigateur Web classique. L'offre de e-Formation est la plus importante dans le domaine des formations continues professionnelles.

1.2.2 Caractéristiques

Keegan, [WAL01], tirait cinq caractéristiques définissant l'enseignement à distance :

- la séparation spatiale et temporelle entre l'enseignant et l'apprenant tout au long du processus d'apprentissage.
- l'organisation et la planification du programme d'apprentissage ainsi que la préparation des matériaux pédagogiques se font par une institution enseignante.
- l'utilisation des technologies et médias.
- l'existence de mécanismes de communication bidirectionnelles entre l'enseignant et l'apprenant.
- l'apprenant est un individu isolé, la notion de groupe est inexistante ou quasi inexistante. Ce dernier point n'est cependant pas une caractéristique systématique grâce à l'émergence de l'apprentissage collaboratif, à distance ou non. Celui-ci consiste à créer un groupe dont les membres travaillent tous, et ensemble, sur le même projet. Cela dépasse donc les rares réunions qui peuvent être organisées lors d'un processus d'apprentissage à distance plus classique. Le groupe est alors un facteur de motivation et de soutien.

Une caractéristique fréquente de l'enseignement à distance est sa flexibilité : accès sans prérequis ni limite d'âge, flexibilité de programme, de lieu, de rythme, d'horaire, etc.

Contrairement à l'enseignement conventionnel dominé par l'enseignant et auquel les étudiants doivent s'adapter, l'enseignement à distance est généralement produit en équipe (et donc moins personnalisé) et s'efforce le plus souvent de répondre aux besoins et demandes très variés des apprenants, que ce soit délibérément ou suite à une concurrence avivée par la globalisation.

1.2.3 Une typologie des enseignements

Différentes formules de cours sont possibles, elles se distinguent par la présence ou l'absence d'un tutorat et par l'unicité ou la multiplicité du support :

- enseignement exclusivement en ligne
  - sans tutorat : cela permet l'autoformation et une formation "à la carte" dans le loisir ou la culture. Par exemple, les portails font partie de ce type d'enseignement.
  - avec tutorat : le tutorat peut se faire de manière synchrone ou asynchrone. Lors d'un tutorat synchrone, l'enseignant et l'apprenant se retrouve en ligne au même moment et peuvent discuter en direct. Lors d'un apprentissage avec tutorat asynchrone, ils communiquent par messagerie en différé.
- l'enseignement en ligne conjugué à un enseignement en présentiel : par exemple, à l'université, les cours en ligne sont un complément au cours présentiel.

La figure 1.2 montre les cinq types d'enseignements en allant du présentiel à l'autoformation, en passant par les formes intermédiaires. Cette typologie est établie en fonction du niveau d'intégration des TIC dans la formation.

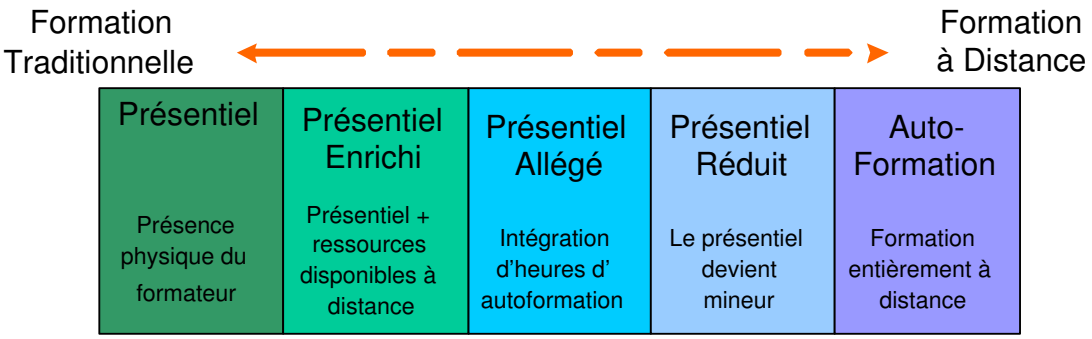


Fig. 1.2 – Typologie des enseignements selon le niveau d'intégration des TIC

Présentiel C'est le cours traditionnel qui se passe en un lieu précis avec la présence physique d'un formateur. Il peut être individuel ou collectif. Les enseignants peuvent enrichir leur cours grâce à la projection de ressources textuelles, graphiques, audio et vidéo, d'expérimentations ou de simulations.

Présentiel enrichi L'entièreté de la formation a toujours lieu en présence de l'enseignant mais celui-ci met à la disposition des étudiants différentes ressources disponibles à distance. Ces ressources peuvent être, par exemple, des exercices, un syllabus ou des dispositifs d'autoévaluation.

Présentiel allégé Le principal de la formation se réalise en présentiel avec certaines heures dédiées à l'autoformation multimédia tutorée, par exemple, en remplacement d'heures de travaux pratiques traditionnels.

Présentiel réduit Le principal de la formation se fait en dehors de la présence de l'enseignant. Cela implique donc un changement dans son rôle. Il suit les étudiants de manière synchrone et asynchrone, lors d'échanges à distance et, de manière synchrone lors d'échanges en présentiel. Les évaluations se font en présentiel.

Autoformation La formation se fait, dans sa totalité ou au moins en très grande partie, à distance. Les étudiants se déplacent uniquement pour l'évaluation finale, s'il y a lieu.

1.2.4 Les enjeux

"Pour qu'il y ait apprentissage, il faut qu'il y ait savoir apprendre, aimer apprendre et vouloir apprendre. L'absence de l'une ou de l'autre de ces dimensions entraîne très rapidement un arrêt du processus". [BER96]

L'autonomie de l'apprenant, l'accompagnement de l'apprenant et le développement des dispositifs de support pour la FAD sont les principaux enjeux de la formation à distance.

L'autonomie de l'apprenant, comme celle des autres acteurs du dispositif, peut être examinée sous deux angles :

- la motivation : sens, projet, finalités, enjeux personnels ;
- les capacités métacognitives : apprendre à apprendre, identifier et gérer des ressources, travailler avec des pairs, maîtriser les techniques et outils d'apprentissage, etc.

Autonomie et motivation L'autonomie ne peut pas être considérée comme une caractéristique fixe. Elle dépend de la formation et de son importance pour le sujet apprenant et non du dispositif pédagogique. De plus, elle évolue tout au long de l'apprentissage. Il n'y a, de ce point de vue, pas de bon ou de mauvais dispositif, mais des dispositifs plus ou moins adaptés

La motivation peut être vue en deux temps [CHIOUSSE01], d'une part au niveau du lien entre l'individu et son apprentissage, et d'autre part, au niveau du lien existant entre l'apprenant et son environnement. On parle de motivation intrinsèque et de motivation sociale liée au groupe.

Dans l'approche comportementaliste, c'est l'enseignant qui permet de motiver l'étudiant grâce à des récompenses ou des encouragements.

L'approche cognitiviste considère la motivation comme, provenant de la relation existante entre l'individu et son environnement.

L'accompagnement L'accompagnement dans les dispositifs de FAD doit à la fois prendre en compte ces problèmes de gestion et de motivation et aider l'apprenant à les gérer. En conséquence, l'accompagnement interpersonnel doit intégrer des dimensions non seulement pédagogiques mais aussi organisationnelles, professionnelles et personnelles.

Le développement des dispositifs Le développement des dispositifs de support pour la FAD est un enjeu majeur sur le plan technologique. C'est ainsi que l'on voit se développer des plates-formes de téléformation. A ce jour, on dénombre beaucoup de plates-formes dédiées à la e-formation et parfois, à la formation à distance, plus générale. On peut se poser la question d'un tel foisonnement dans ce domaine. La multiplicité des plates-formes existantes semble témoigner d'une tendance à essayer de développer une plate-forme répondant à ses besoins spécifiques sans chercher à adapter l'une ou l'autre plate-forme existante.

1.2.5 Problématique

Dans le contexte des constructivismes, il y a un nouvel idéal éducatif par le concept d'"environnement", pour la formation à distance comme pour le présentiel :

- Créer des environnements d'apprentissage qui favorisent le travail collaboratif et la communication ;
- Faire une place aux connaissances antérieures ;
- Assurer un apprentissage actif ;
- Contextualiser l'apprentissage ;

- Développer l'autonomie (apprendre à apprendre)
- Effectuer des tâches authentiques et/ou décoder des documents authentiques
- Favoriser une variété d'expériences d'apprentissage

Il s'agit d'un défi éducatif en passant du contrôle de l'apprenant à son autonomie. Il s'agit également d'une mise en relation en créant une certaine proximité malgré la distance existant au départ.

L'apprenant doit savoir se repérer dans le cours présenté via le Web, trouver les concepts clés dont il a besoin et se soumettre à des auto-évaluations continues. Selon les tâches plus ou moins complexes qu'il doit accomplir, il doit pouvoir continuer ou revenir sur des unités mal assimilées ; c'est-à-dire, qu'il doit pouvoir gérer seul une bonne partie de sa formation. Il doit bénéficier de techniques pédagogiques aptes à lui permettre d'acquérir les capacités établies par le projet de formation.

Vu la diversité des outils de la formation à distance, il faudrait prendre des décisions adéquates selon :

- les situations de formation
- les caractéristiques des différentes matières à enseigner (faire un cours pour la formation continue des médecins / enseigner une langue étrangère)
- la nécessité absolue du multimédia pour certaines matières

1.2.6 Les technologies dans l'apprentissage

L'introduction de la technologie dans le processus d'apprentissage amène à développer l'esprit critique de l'apprenant et ainsi lutter contre la désinformation et la surinformation. On appelle les nouveaux "progrès de l'esprit" [CL03] la capacité à déterminer la validité des sources documentaires, d'analyser en profondeur l'information véhiculée, de se construire une représentation personnelle, argumentée et flexible, d'acquérir la capacité de relier et d'organiser des données éparses. C'est à ce niveau que doivent intervenir les enseignants.

Les technologies peuvent assurer trois fonctions distinctes :

La fonction de consultation sites, documents, experts

La fonction de communication experts, visioconérences, conférences thématiques, travail collaboratif ou coopératif, courriel.

La fonction de production suppose d'autres types de technologie

Chacune de ces fonctions exige des démarches et des approches pédagogiques particulières au plan pédagogique.

Environnements d'apprentissage informatisés L'environnement d'apprentissage est défini comme un lieu réel ou virtuel abritant un ou plusieurs systèmes interagissant dans un but commun : l'apprentissage. L'environnement d'apprentissage est dit informatisé lorsque certaines ou l'ensemble des interactions entre les sous-systèmes sont soutenues par des ressources informatiques.

La définition énoncée ci-dessus se situe dans le cadre de la théorie des systèmes où l'environnement est un lieu abritant un ou plusieurs systèmes. A son tour, un système est un ensemble de composantes qui, sous l'effet d'un stimulus, génère une réponse et dont les actions sont orientées vers un but commun.

L'environnement offre à l'apprenant les moyens de constuire leurs connaissances grâce à des activités d'apprentissage adaptées. De ce fait, on peut dire que le concept d'environnement entre dans la théorie constructiviste.

Après l'étude de divers types d'environnement, Collins a observé la présence de trois fonctions générales [JB98b] :

- Participation au discours
- Participation aux activités
- Présentation de travaux aux fins de l'évaluation

Les ressources sont réparties dans cinq espaces virtuels, dans lesquels chaque acteur évolue d'une façon qui lui est propre. Par exemple, dans une situation typique de téléapprentissage, les espaces disponibles à l'apprenant prennent la forme suivante :

Espace de navigation et de gestion qui donne accès au navigateur de scénario pédagogique et aux outils de gestion, par exemple : carnet, agenda, plan de travail ;

Espace d'information qui regroupe les divers types de documents ou de données dont l'apprenant a besoin pour exécuter les différentes activités prévues dans son scénario pédagogique ;

Espace de production qui contient les outils nécessaires pour produire des travaux ;

Espace de communication et de collaboration qui réunit les outils permettant à l'apprenant d'échanger avec d'autres intervenants, de réaliser des travaux en équipe, de participer à des télédiscussions ou à des conférences ou séminaires à distance ;

Espace d'assistance qui permet d'obtenir de l'aide, des conseils ou une adaptation de l'environnement, de la part d'une personne-ressource en ligne ou du système informatique.

Ainsi, aux fonctions de communication, d'information et de production identifiées dans [JB98b], s'ajoutent deux autres fonctions, soit celle de navigation et de gestion, et celle d'assistance.

A ces espaces, on peut associer les cinq classes d'outils identifiées par Perkins [JB98b], soit :

Les banques d'informations (information banks) : toute ressource qui, plus que toute autre chose, est source d'information, par exemple, les bases de données ;

Les surfaces symboliques (symbol pads) : outils de support à la mémoire à court terme, qui servent à enregistrer des idées, développer des plans ou formuler et manipuler des équations, par exemple, un traitement de texte ou un logiciel de dessin ;

Les ensembles de construction (construction kits) : ressources semblables aux surfaces symboliques, à la différence qu'elles ne sont pas vierges, qui contiennent un

ensemble de composantes et de procédés prédéfinis, par exemple, un langage de programmation) ;

Les aires d'étude de phénomènes (phenomenaria) : endroits accessibles à l'investigation et à la manipulation, où sont présentés des phénomènes, par exemple, un micromonde ou un simulateur ;

les gestionnaires de tâches (task managers) : ressources qui définissent les tâches à accomplir, guident et parfois aident les apprenants dans la réalisation de ces tâches et fournissent une rétroaction sur le processus ou le produit de l'apprentissage, par exemple, un tutoriel.

Ce qui décide si un environnement est informatisé, c'est la pertinence du choix des ressources informatiques pour supporter telle ou telle fonction dans tel ou tel contexte d'apprentissage et leur intégration harmonieuse dans l'environnement d'apprentissage. Il n'est pas opportun de parler d'environnement d'apprentissage informatisé si une ressource informatique n'apporte rien à l'apprentissage.

Pour classifier les environnements d'apprentissage informatisés, peut-être faudra-t-il développer de nouvelles taxonomies. En effet, de nouveaux critères semblent nécessaires. Par exemple, il est possible de subdiviser les environnements d'apprentissage selon leur "degré d'ouverture" sur le monde. On distingue alors : les micromondes informatiques, les environnements d'apprentissage basés sur la classe et les environnements ouverts et virtuels.

Les micromondes informatiques : Les étudiants entrent dans un environnement informatisé autonome. Ces micromondes peuvent être supportés par un environnement de classe plus large, mais pas nécessairement.

Les environnements d'apprentissage basés sur la classe : La classe est l'environnement d'apprentissage premier. Différentes technologies peuvent être utilisées comme outils pour supporter les activités de la classe.

Les environnements ouverts et virtuels : Certains environnements d'apprentissage informatisés sont des systèmes relativement ouverts, permettant des interactions et des rencontres avec d'autres participants, ressources et représentations. Dans un micromonde, l'apprenant interagit avec l'ordinateur essentiellement, alors que dans un environnement ouvert et virtuel, il interagit d'abord avec d'autres participants et avec des outils répartis dans l'environnement.

1.3 Apprentissage de la programmation

Les taux d'échec ou d'abandon aux cours d'initiation à la programmation en premier cycle universitaire se situent autour de 80% de part le monde. Au vu de l'importance que l'informatique a acquise, ce problème devient un enjeu majeur.

1.3.1 Historique

Méthode empirique L'apprentissage de la programmation se faisait à travers l'apprentissage d'un langage. L'enseignant aidait l'apprenant à organiser ses idées par des organigrammes mais le reste, l'apprentissage des concepts, l'étudiant devaient les comprendre par lui-même.

Réaction à l'empirisme : émergence de l'algorithme Constatant l'impuissance de la méthode empirique, des gens comme Dijskstra, Wirth, Ledgard et Arsac ont amené l'idée que la programmation était une résolution de problèmes et qu'il fallait s'y prendre avec méthode. Avant d'écrire un programme, on écrit ou même on programme un algorithme. L'idée était que puisqu'on programmait un algorithme, les langages devaient être conçus pour programmer des algorithmes. Algol (ALGOrithmic Language) fut créer depuis cette idée, suivie de Pascal dans les années septante. L'apport principal de l'algorithmique est la généralisation de la notion de structure à tous les niveaux : structures de programme, structures de contrôle, structures de données. Si l'algorithmique fut la première méthode à prendre en compte la pédagogie, elle ne se base sur aucune théorie de l'apprentissage. Une liste de concepts à apprendre était établie et ceux-ci suivaient une certaine progression plus ou moins rapide selon le public ciblé.

Il y a quelques années cette méthode fonctionnait car le public était principalement composé d'adultes très motivés et/ou scientifiques. Le public d'aujourd'hui s'est fortement élargi vers les enfants ou d'autres segments de la population qui n'ont pas, ou peu, de connaissances scientifiques préalables.

Didactique La différence avec l'algorithmique classique vient de ce qu'on ne cherchera pas à étudier toutes les structures de contrôle ou tous les types de variable mais à ce que les élèves utilisent effectivement, c'est-à-dire comprennent, celles qui ont été étudiées. La question importante est celle du "comment" faire. En effet, le but n'est pas de faire des programmeurs, mais de développer la pensée logique, la capacité d'abstraction, par un apprentissage de la programmation.

La programmation visuelle Dans les années 80, de nombreux projets ont jeté les bases de la programmation visuelle. L'idée principale de cette approche est de remplacer les mots par des images. Cependant, son application est limitée et elle ne permet pas de visualiser l'état du programme à un moment donné de son exécution.

L'utilisation d'exemples pour concevoir des programmes constitue le second pas vers les environnements de programmation pour utilisateur final.

On différencie la programmation à partir d'exemples et la programmation sur exemples ou par démonstration. Pour la première, l'utilisateur conçoit lui-même ses fonctions sur des valeurs réelles, utilisées comme variantes par le programme. Le deuxième type de programmation trouve son origine dans le système Pygmalion réalisé par David SMITH en 1975. En 1980, dans ses travaux, Brad MYERS formalise le concept et lui donne le nom de programmation basée sur les exemples. Plusieurs travaux de recherche se sont succédés sur le sujet. En 1993, Allen CYPHER fait un récapitu-

latif des systèmes de PSE existants et introduit la notion de la programmation par démonstration. L'idée principale est d'éviter le niveau d'abstraction des variables manipulées par l'utilisateur et de ne lui permettre d'interagir qu'avec les valeurs des objets.

1.3.2 Principes de l'enseignement de l'informatique

Dans l'enseignement de l'informatique, il existe une boucle "infernale" : il est indispensable de donner un minimum de théorie (découverte et appropriation) mais ces savoirs ne sont intégrés qu'après des exercices pratiques.

L'enseignant a un rôle triple :

- Il doit veiller à ce que les séances de travaux pratiques rencontrent les concepts à apprendre. De telle manière que les apprenants mettent en pratique ces concepts et ainsi construisent les connaissances appropriées.
- Il a un rôle de personne ressource lors de ces mêmes travaux pratiques lors de problèmes rencontrés par les apprenants.
- Lors des apports "théoriques", il doit présenter le contenu à maîtriser en terme de problèmes à résoudre. Cela permet à l'apprenant de lui donner un sens et aide la mémorisation

Les connaissances sur le fonctionnement ne sont organisées qu'après une phase de manipulation de ces connaissances.

Il y a deux manières de construire les connaissances. Soit, l'apprenant part d'une expérience concrète pour y appliquer des observations pour le mener vers une conceptualisation abstraite. Soit, il part de la conceptualisation abstraite qu'il va expérimenter de façon active et aller vers une expérience concrète.

Une stratégie d'apprentissage est de débuter le processus d'appropriation par l'utilisateur avec des documents de découverte, des concepts ou des outils faisant l'objet de l'apprentissage. Le parcours peut alors être le suivant :

- documents de découverte
- séances "en salle de cours" : réponses aux questions, synthèse
- exercices de mise en oeuvre

La deuxième étape prend une importance considérable dans l'enseignement traditionnel, avec une interaction pas toujours excellente. Dans l'apprentissage à distance, le centre de gravité peut être déplacé vers un travail personnel à domicile. Le plus important est alors d'organiser ce travail, donc de fournir une structure en étapes, un "planning" à l'apprenant.

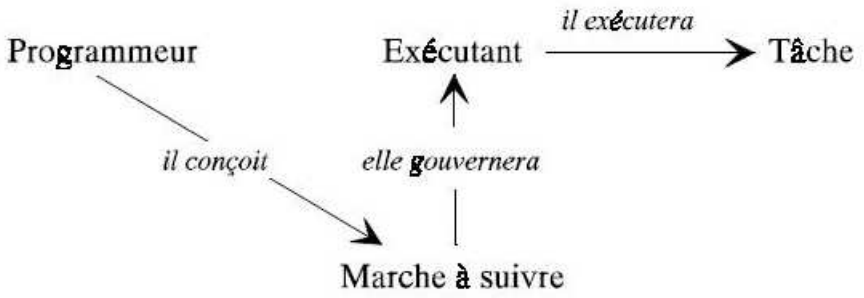


FIG. 1.3 – Problématique de la programmation

1.3.3 Problématique

Pour Charles Duchâteau [DUC02], la problématique de la programmation, représentée à la figure 1.3 se résume de la façon suivante :

"Faire faire une tâche en différé par un dispositif exécutant."

Le programmeur fait face, premièrement, à la tâche qui devra être correctement définie et précisée et, deuxièmement, à l'exécutant-ordinateur aux capacités limitées mais connues. Il s'agira alors de construire une marche à suivre qui permettra de faire faire cette tâche par cet exécutant.

Pour l'utilisateur, l'ordinateur équipé d'un programme convenable constitue un outil. L'ordinateur est une boite noire, son fonctionnement n'intéresse pas l'utilisateur.

Pour l'analyste-programmeur, l'ordinateur est un exécutant auquel il va devoir expliciter le traitement des informations à effectuer. L'ordinateur n'est plus une boite noire mais il est capable d'effectuer certains traitements élémentaires et formels d'informations au travers d'un langage de programmation.

Il faut à l'avance avoir complètement déplié la tâche à faire effectuer et la traduire dans une marche à suivre qui précise en détails, sans ambiguïté et de manière exhaustive les instructions destinées à l'exécutant et les indications permettant d'organiser la suite de ses actions.

Pour qu'un problème soit bien posé en programmation, il faut non seulement que la tâche soit correctement précisée, mais encore que l'exécutant qui sera chargé de son exécution soit complètement décrit.

Le cycle de programmation Les étapes du faire faire sont représentées sur le schéma de la figure 1.4.

Quoi faire La description de la tâche à effectuer est floue et doit être précisée grâce à diverses questions pour arriver à une description précise.

Comment faire Il s'agit ici d'expliquer les stratégies à mettre en oeuvre pour accomplir la

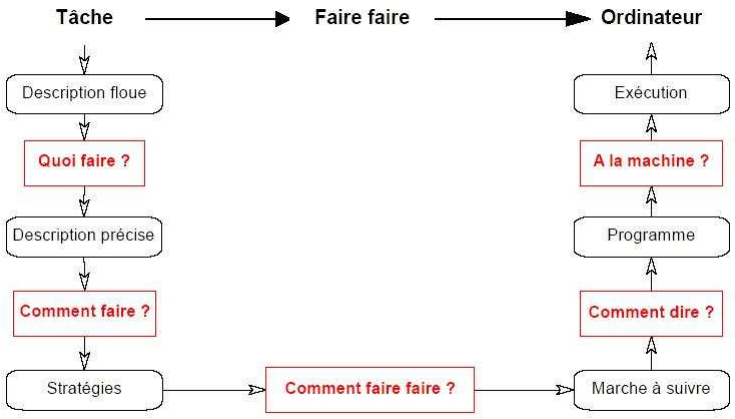


FIG. 1.4 – Cycle de programmation : les étapes du faire faire

tâche. La démarche est totalement différente entre savoir exécuter et savoir expliquer **comment** exécuter.

Comment faire faire Sur base des deux étapes précédentes, il s'agit d'établir une marche à suivre destinée à l'exécutant-ordinateur.

Comment dire On traduit la marche à suivre dans un langage de programmation, qui sera compréhensible par l'ordinateur.

A la machine Cette dernière étape est celle du travail qui se fait sur l'ordinateur, où va avoir lieu la compilation suivie de la correction des erreurs.

Problèmes possibles pour l'apprenant Lors de l'apprentissage de la programmation, on peut rencontrer trois types de problèmes :

- Problème du faire faire, erreurs dites temporelles :
  L'apprenant doit abstraire les différents comportements de la tâche et prévoir le comportement complet du programme.
- Problème de la compréhension de l'exécutant :
  Il peut en découler des erreurs dues à la distance entre la représentation analogique (domaine de la tâche) et la représentation fregeïenne (domaine numérique, ordinateur) d'un objet.
- Problème des erreurs dites "anthropomorphiques" :
  L'apprenant accorde de façon inconsciente à l'interpréteur du programme une compréhension contextuelle de son code, dépassant les instructions explicites.

Le rôle du langage de programmation

Il faut intégrer les contraintes de l'exécutant. L'apprentissage de la programmation ne peut donc pas être totalement dissociée du langage de programmation. La phase d'analyse

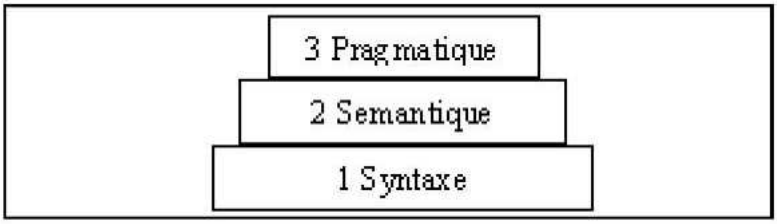


FIG. 1.5 – Décomposition des connaissances en programmation

doit, cependant, garder sa primauté et sa nécessité.

En se référant à une décomposition des connaissances de la programmation commune en didactique de la programmation, représentée à la figure 1.5 et connue sous le nom d'"échelle sémiotique", la plupart des difficultés rencontrées lors de l'apprentissage sont de nature sémantique ou pragmatique. Les erreurs sémantiques sont liées à la compréhension des principes de fonctionnement du langage tels que l'itération ou la programmation modulaire. Les erreurs pragmatiques se rapportent au transfert des connaissances en programmation pour traiter un cas concret.

La couche syntaxique génère un grand nombre d'erreurs de faible importance, à cause de défauts d'utilisabilité et elle semble être une cause importante de frustration, et donc d'échec. Plus encore, cela focalise l'attention sur les problèmes syntaxiques au détriment des couches supérieures.

L'apprentissage via un langage de programmation amène d'autres difficultés que celles énoncées plus haut. Ces difficultés sont liées à l'instrumentation :

- Le cycle édition-compilation-test nécessite de prévoir à long terme les effets du programme contre une interprétation immédiate qui viendrait d'un système interactif.
- Le mode de communication basé sur le langage est un support implicite de la vision anthropomorphique de l'exécutant-ordinateur.
- L'IHM est concentrée sur une petite partie de l'activité, le "comment dire", alors que les pas précédents sont difficiles à appréhender pour l'apprenant.

La plupart des environnements d'initiation sont les mêmes que les environnements de programmeurs avertis, ce qui engendre plusieurs problèmes.

- Il est impossible à l'apprenant de séparer les difficultés algorithmiques et de représentation des données
- L'apprenant a un modèle anthropomorphique de l'exécutant-ordinateur alors que celui-ci est erroné.
- L'attention est axée sur la réduction de la distance d'évaluation du programme final mais le programmeur débutant ne sait pas par quel bout commencer.

1.3.4 La programmation par l'exemple

Définition

La PSE est une extension du concept des enregistreurs de macros. Ces derniers sont à même d'enregistrer les actions de l'utilisateur et de les rejouer à la demande. Cependant l'exécution des macros ainsi réalisées ne prend pas en compte le contexte de création ou d'exécution. Dans le cas de la PSE, le système produit une généralisation des actions enregistrées. Durant la conception d'un programme, le système analyse les entrées de l'utilisateur, et construit le programme à même de générer l'exemple. L'utilisateur final est le programmeur qui ne programme que par l'intermédiaire d'une interface graphique.

Un système de programmation est dit basé sur exemples si l'utilisateur peut utiliser les valeurs d'un exemple d'exécution pour définir les objets sur lesquels portent le programme à construire.

La programmation sur exemples est à l'opposé de la programmation avec exemples. Un système de programmation est dit avec exemples si la visualisation des programmes, associés à des exemples, permet seulement une représentation plus concrète de ce que l'on a écrit et non, une description concrète de ce que l'on veut.

Objectifs et caractéristiques

Le but principal de la PSE est de générer interactivement des programmes. La PSE s'appuie sur le fait que l'utilisateur fait moins d'erreurs lorsqu'il raisonne sur des exemples que sur des concepts abstraits.

La PSE se place à un niveau d'abstraction beaucoup plus élevé que les enregistreurs de macros. Elle résout des problèmes d'interprétation par l'intermédiaire de procédures de dialogues sophistiquées ou par l'utilisation de règles.

Limites des systèmes de PSE

Les préférences, bien que largement utilisées, présentent des limites importantes. Elles ne peuvent répondre qu'à un nombre limité de situations prévues par le concepteur de l'application. Du fait des besoins potentiellement variés des utilisateurs, elles ne peuvent répondre à leur totalité. Pour répondre au plus de besoins possibles, il faut proposer un nombre important d'options, qui rend la configuration de l'application par l'utilisateur très difficile.

Les enregistreurs de macros sont trop littéraux. La séquence de commande enregistrée est rejouée sans discernement de leur part. Le contexte à l'exécution n'est pas pris en compte. De plus, pour comprendre ou modifier le fonctionnement d'une macro existante, il faut connaître le langage de programmation de l'application. D'un point de vue implémentation, ils possèdent donc le même problème que les langages d'application puisque, là aussi,

il faut écrire un interpréteur et, en plus, un mécanisme pour enregistrer les commandes de l'utilisateur sous forme textuelle.

Les scripts sont des langages d'application à part entière. Ils nécessitent non seulement la connaissance d'un vocabulaire et d'une syntaxe, mais surtout la maîtrise des concepts de programmation tels que les structures de données (listes, tableaux) et les structures de contrôles (boucles, conditions, procédures). Même s'ils permettent d'étendre une application dans des proportions importantes, il n'en demeure pas moins que pour automatiser des tâches simples, il ne devrait pas être nécessaire de recourir à de tels langages.

Les systèmes de PSE sont difficiles à mettre en oeuvre. Ils sont réalisés au moyen de langages d'application. Les langages d'application réclament en effet un effort de conception (choix d'une syntaxe et d'un vocabulaire appropriés) et d'implémentation (écriture d'un interpréteur ou d'un compilateur intégré à l'application) important. À ces difficultés d'implémentation liées aux langages d'applications et aux enregistreurs de macros s'ajoutent celles qui sont liées à la généralisation d'un programme. La visualisation d'un programme est problématique d'autant plus que, non seulement les commandes, mais aussi leur généralisation, doivent êtres représentés.

1.3.5 La programmation par l'exemple : quelques systèmes

L'apprentissage de la programmation comporte plusieurs problèmes comme vu ci-dessus. Pour pouvoir les résoudre, il faut développer un environnement de programmation destiné à l'initiation. Un environnement d'apprentissage, quelque soit la nature de cet apprentissage, doit, entre autres, offrir à l'apprenant la possibilité d'expérimenter, de pratiquer et d'avoir des retours sur ses performances, d'où la nécessité d'utiliser des techniques multimédias ; d'une part, pour réaliser des simulations, et y associer des techniques d'intelligence artificielle et, d'autre part, pour la notion de modèles d'apprenants.

Il existe deux grands types de systèmes d'apprentissage par l'exemple. Ceux qui utilisent une approche pragmatique et ceux qui utilisent une approche sémantique.

- Approche sémantique : représentation explicite de l'exécutant avec son état et ses capacités (technique de programmation par démonstration). Ces systèmes se situent au niveau du "comment faire faire". L'environnement est représenté par un micromonde.
- Approche pragmatique : les systèmes qui utilisent une approche pragmatique ne représente pas explicitement l'exécutant. L'état de celui-ci est plutôt mis dans une "boite noire".

Afin de tester son apport au niveau de la convivialité, la PSE a été utilisée dans un grand nombre de systèmes expérimentaux. Il convient de noter néanmoins que la PSE n'est pas utilisée à grande échelle. Les systèmes de PSE cherchent à fournir aux non-programmeurs le pouvoir d'expression d'un environnement de programmation classique (éditeur, structuration de programmes).

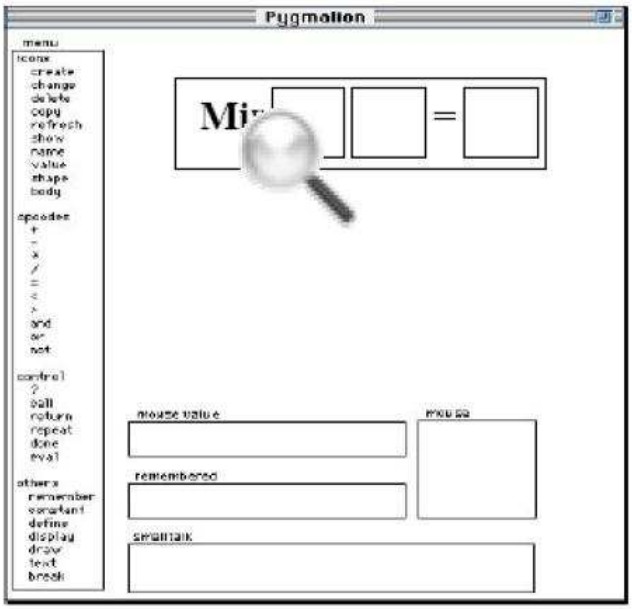


Fig. 1.6 – Environnement de programmation visuelle de Pygmalion

Pygmalion

Pygmalion est un exemple d'environnement de programmation qui adopte une approche syntaxique. On interagit graphiquement avec les instructions.

Pygmalion de David Smith a été le premier système de PSE. C'est la première tentative de programmation visuelle. Elle est basée sur la métaphore du "tableau noir" : la zone de travail est vue comme un tableau où le programmeur peut "dessiner" ses idées. L'environnement est représenté à la figure 1.6.

Pygmalion est à l'origine du concept d'icône, disposant à la fois d'une image, d'un contenu (du texte pour une icône de document), et d'un comportement (déplacement d'un fichier par glisser-déposer). Il fournit un menu permettant d'accéder aux icônes et aux opérations portant sur celles-ci. Il dispose d'un champ de texte appelé "mouse value" dans laquelle toute valeur tapée est rattachée à la souris et peut être déposée dans n'importe quelle icône. Lorsque Pygmalion est en mode enregistrement, une zone d'historique ("remembered") affiche les deux dernières actions exécutées. Le système dispose aussi d'une zone "SmallTalk" où le programmeur peut taper et évaluer des expressions écrites dans ce langage. Enfin, il existe une zone "mouse" où l'on peut définir le sens d'un click avec l'un des boutons.

Pygmalion fournit les représentations graphiques standards d'arithmétique, de relationnelle et des opérateurs booléens.

Il est destiné à des utilisateurs maîtrisant déjà la programmation. Pour créer une procédure, l'utilisateur doit montrer à Pygmalion, sur un exemple concret, comment calculer le

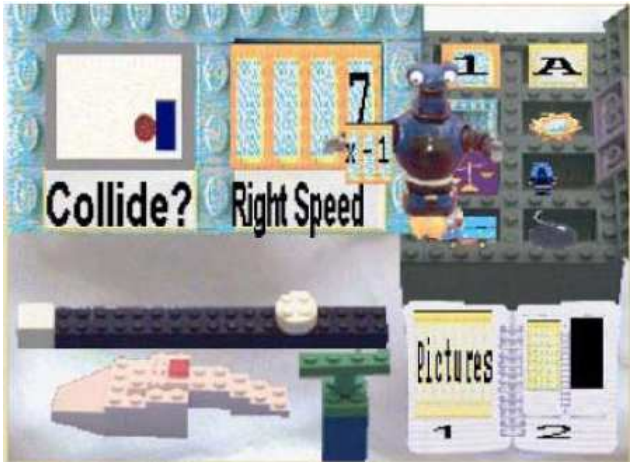


Fig. 1.7 – Environnement de programmation selon la métaphore de la ville dans Toontalk

résultat (i.e. la valeur de retour) de la procédure. A chaque étape du calcul, il doit indiquer les éventuelles structures de contrôle en pressant des boutons pour ajouter au programme en cours d'écriture branchements conditionnels, boucles, ou encore, appels récursifs. Dans le cas de procédures récursives, une seule démonstration suffit. A titre d'exemple, Pygmalion est capable de créer par démonstration la procédure récursive factorielle.

ToonTalk

ToonTalk est un environnement de programmation à l'aide d'exemples utilisant la métaphore de jeu de construction pour enfant, représenté à la figure 1.7. Il suit une approche sémantique. ToonTalk est un système de programmation iconique qui s'adresse aux enfants (à partir de six ans). Il utilise l'animation pour s'exprimer et non le texte ; il convient particulièrement bien à ceux qui ont une intelligence plus orientée vers l'action et moins verbale.

ToonTalk se base sur une approche particulière : le programmeur est un acteur dans le monde virtuel. Chaque objet abstrait de la programmation est représenté grâce à une métaphore, cfr tableau 1.1. Cette approche le place en marge des systèmes de PSE classiques. S'adressant à des enfants, ToonTalk tend à faire l'analogie entre la programmation et un jeu de construction. Dans la métaphore utilisée, chaque objet du monde de la programmation est associé à son équivalent dans le monde de ToonTalk. Un programme complet se représente sous la forme d'une ville, les sous programmes et les processus sont représentés par des maisons, que l'on peut faire communiquer à l'aide de pigeons voyageurs. Ceux-ci peuvent transporter des objets depuis leur maison jusqu'à des nids. Chaque maison peut contenir des robots ; ce sont eux qui accomplissent les tâches à l'intérieur des sous-programmes. On peut affecter à ces tâches des pré-conditions qui apparaissent sous forme de bulles.

On peut apprendre à un robot à effectuer sa tâche par l'exemple. Cette tâche peut être

Abstraction informatique	Concrétisation dans ToonTalk
programme	ville
sous-programmes ou processus ou objet	maison
méthodes	robots
messages ou vecteurs	boites
sorties d'un sous-programme	camions chargés
terminaison d'un sous-programmes	bombes
capacité des canaux de transmission	oiseaux

TAB. 1.1 – Correspondances entre les métaphores utilisées par ToonTalk et les concepts de programmation

par exemple :
- L'envoi d'un message : en donnant une boîte à un pigeon ;
- La création d'un nouveau processus (une nouvelle maison) : en chargeant une boîte et un ou plusieurs robots dans un camion ;
- Modifier une structure de données par copier-coller de boîte à boîtes ;
- Tuer un processus en envoyant une bombe sur la maison ;

EbP - Example-based Programming

EbP est un environnement de développement de programmes paramétrés, susceptibles de générer des comportements standards, représenté à la figure 1.8. Il se situe dans le domaine de la géométrie dynamique. Il permet en particulier de mettre au point par manipulation directe et visuelle les programmes générés.

EbP utilise des techniques de PSE permettant la création de familles de composants

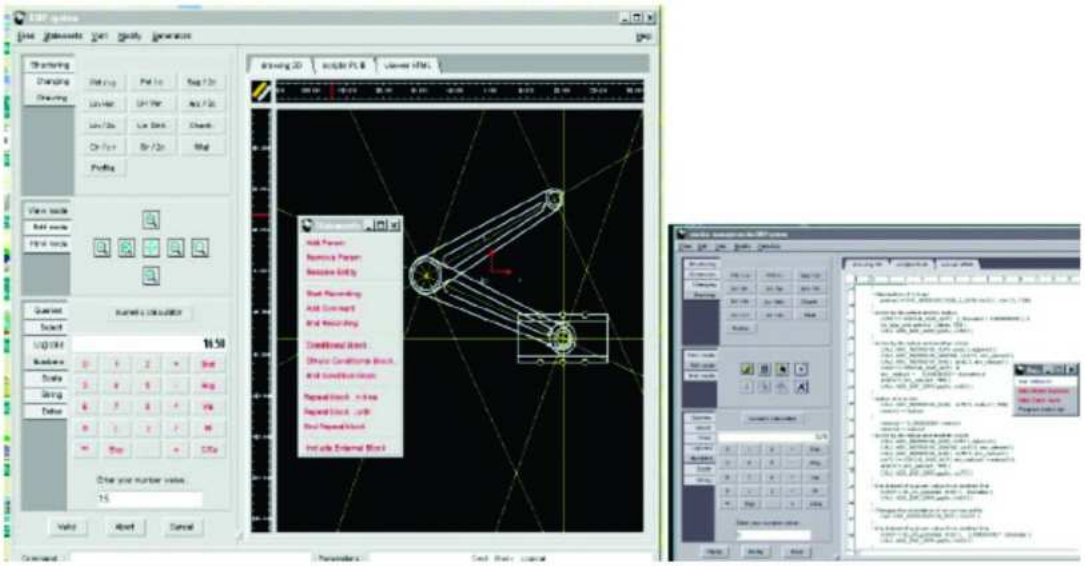


FIG. 1.8 – Environnement de programmation Ebp

standards. Basé sur une interprétation orientée vers la sémantique des interactions utilisateurs, EbP utilise comme représentation interne un arbre de syntaxe abstraite. Il utilise une double capture : au niveau lexicographique, pour filtrer les commandes et les désignations autorisées, et au niveau sémantique.

EbP comporte des structures de contrôle complètes. Les sous-programmes, les alternatives et les répétitions sont totalement disponibles. Les alternatives et les répétitions supposent la définition d'expressions booléennes. Plusieurs schémas d'itération sont fournis. L'itération d'ensembles est disponible pour des objets sélectionnés à l'aide d'un rectangle élastique, ou pour les transformations géométriques multiples.

Mondrian

Mondrian [Lieberman 1993] est un éditeur graphique orienté objet avec lequel l'utilisateur peut définir de nouvelles commandes par démonstration. Il est destiné aux personnes accomplissant des tâches graphiques et désireuses de créer de nouvelles commandes sans pour autant posséder des notions de programmation. Cet éditeur est très simple et offre une représentation graphique originale des commandes sous forme de dominos : le côté gauche d'un domino est une miniature de l'écran avant exécution de la commande, tandis que le côté droit montre l'écran après exécution. Parmi les commandes proposées on trouve la sélection, le déplacement et le groupement d'objets, la création de rectangles, le choix de couleurs, etc.

Mondrian ne détecte ni les boucles ni les conditions : une commande n'est qu'une séquence linéaire d'instructions, ce qui limite sa force d'expression.

MELBA - Metaphors-based Environment to Learn the Basics of Algorithmic

MELBA est un environnement d'apprentissage de l'algorithmique à l'usage des étudiants en début de cursus universitaire ou plus généralement de cursus d'enseignement supérieur. Le système MELBA utilise les deux approches citées ci-dessus, l'approche sémantique et l'approche pragmatique qui sont complémentaires. Il représente l'état du système plutôt que le système lui-même.

L'objectif de ce projet est de mesurer l'impact sur l'apprentissage :
- de la représentation graphique explicite des différents modèles sous-jacents à la conception de programmes.
- du support explicite par un paradigme d'interaction adapté du processus de conception et pas seulement d'évaluation.

MELBA est composé de trois espaces (dynamiques et synchrones) :
- l'espace du programme permettant l'édition du programme,
- un micromonde pragmatique représentant l'expérience,
- un micromonde sémantique représentant l'exécutant.

Chacun des micromondes a un lien dynamique avec le programme de manière à avoir un retour immédiat et animé de l'action effectuée, et ce dans les deux sens.

Programmation sur exemple pragmatique Ce micromonde représente l'expérience concrète (par exemple : expérience des verres d'eau) et la capacité de l'exécutant-ordinateur. Il n'explicite pas son fonctionnement interne puisque l'objectif de ce module est l'appréhension de la structuration temporelle des algorithmes.

Comme expliqué ci-dessus, ce micromonde est interactif et il génère l'écriture du programme dans l'éditeur. Inversement, lorsque l'apprenant édite un programme, il peut voir à tout moment quel est l'état courant de la tâche. Cela peut lui permettre aussi de comprendre la tâche effectuée par un programme grâce à l'animation. L'éditeur permet, ensuite, de restructurer le programme généré.

La figure 1.9 représente un programme qui remplit d'eau un rang de verres en utilisant une pipette. Plusieurs actions sont proposées telles que passer au verre suivant, mettre une goutte.

Grâce à ce micromonde, l'apprenant peut, à partir d'une tâche créer un programme facilement et ce, uniquement grâce à son savoir-faire sur cette tâche.

Programmation sur exemple sémantique Ce micromonde représente le modèle que l'apprenant doit acquérir. Il utilise la métaphore du bureau. Ce choix a été fait d'une part, parce que l'environnement du bureau est un environnement familier ; et d'autre part parce qu'il existe une correspondance très forte entre les objets du bureau et les concepts de programmation. Ce micromonde est composé de trois éléments :
- un gestionnaire de fichiers qui constitue la mémoire à long terme du programme
- une calculatrice symbolique
- un panel représentant le contenu de la mémoire de travail, mémoire éphémère.

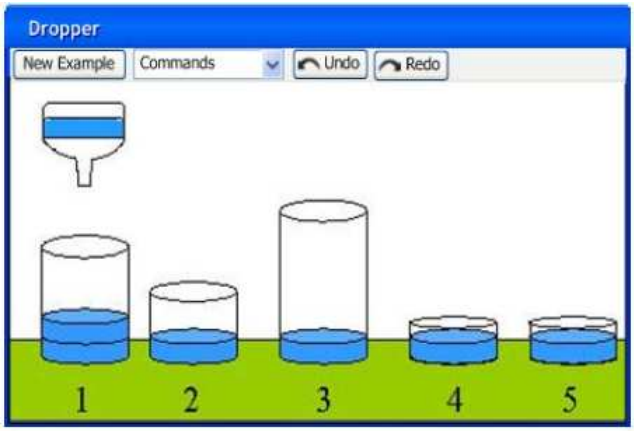


FIG. 1.9 – Représentation du micromonde pragmatique

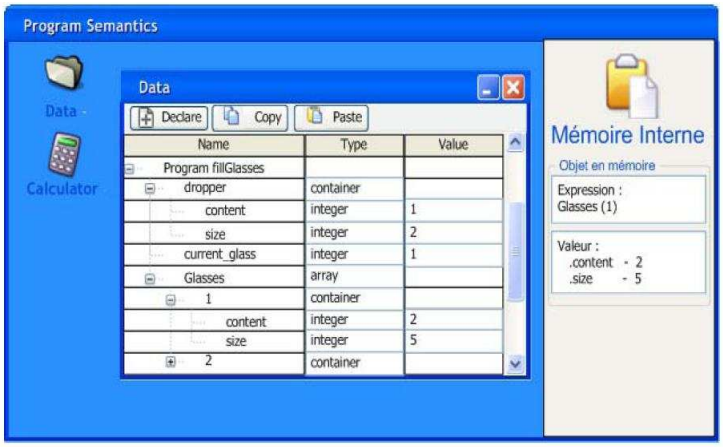


FIG. 1.10 – Représentation du micromonde sémantique

A l'instar du micromonde pragmatique, le micromonde sémantique représenté à la figure 1.10 est synchronisé avec le programme. Cela permet une meilleure appréhension des différents concepts, par la construction ou l'animation. De plus, il est relié avec une représentation pragmatique pour mettre directement en relation les représentations fregeïenne et analogique.

La figure 1.11 montre la synchronisation existant entre le micromonde pragmatique, qui représente l'exemple (panneau de droite) et la représentation du programme (panneau de gauche). Lorsque l'utilisateur sélectionne une instruction, le programme s'exécute jusqu'à ce point, avec la correspondance pour chaque instruction sur l'exemple. Lorsqu'une instruction est ajoutée, le système se re-synchronise (flèche 1), c'est la programmation avec exemples. L'utilisateur peut également montrer sur le panneau de droite ce qu'il attend que le programme fasse (flèche 2). C'est la programmation par démonstration.

Exemple du remplissage de verres avec une pipette

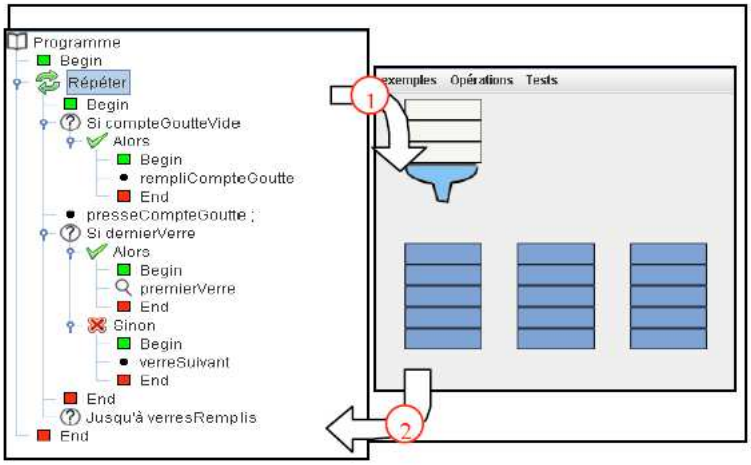


FIG. 1.11 – Synchronisation entre le micromonde pragmatique, qui représente l'exemple, et la représentation du programme

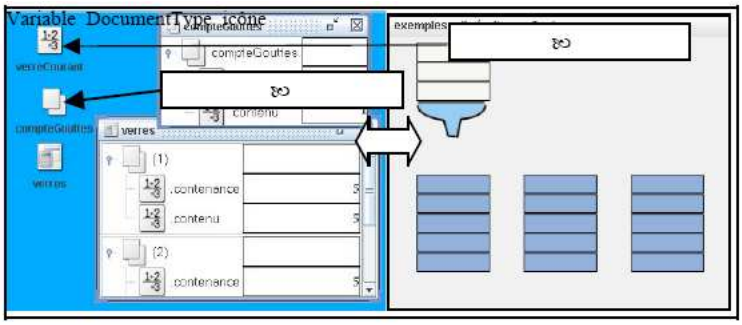


FIG. 1.12 – Synchronisation entre les micromondes pragmatique et sémantique

La figure 1.12 représente l'état des données du programme. Il utilise la métaphore du bureau, bien connue par les utilisateurs, pour leur expliquer les concepts de variable, de type, de paramètre et de référence.

L'interactivité du système permet aux utilisateurs de diminuer les erreurs grâce à la possibilité de vérifier à tout moment l'impact de leurs actions.

Chapitre 2

Interfaces homme-machine pour appareils mobiles à petit écran

2.1 Les appareils mobiles

De nos jours, une foule d'appareils mobiles dotés d'un écran de taille réduite a fait son apparition : téléphones mobiles, smartphones, ou encore assistants personnels.

Les fonctionnalités de base du téléphone mobile sont le transfert de la voix de manière numérique ; de données, par exemple, au travers du WAP ; les messages écrits (SMS) ; les services de gestion des appels, et d'autres services proposés par des sociétés extérieures telles que les banques, des informations sportives, ou des services de localisation. Les téléphones mobiles ont une interface utilisateur matérielle très limitée. Ils possèdent un pavé numérique auquel on ajoute quelques boutons pour naviguer dans les menus. Dans certains cas, ces boutons peuvent également faire office de raccourcis vers des fonctions données.

Le smartphone tend à réduire la frontière entre les assistants personnels et les téléphones mobiles. Ils possèdent des fonctionnalités avancées telles que la navigation web ou des fonctions d'agenda qui sont présentes dans les PDA's, tout en alliant la téléphonie. Il est possible de lui ajouter des applications supplémentaires créées par un développeur professionnel ou par l'utilisateur lui-même. Le système d'exploitation, de loin, le plus présent est Symbian grâce à son adoption par Nokia. Il est suivi par Microsoft et Palm. Contrairement aux PDA's, ils ne sont pas tous équipés d'écrans tactiles.

Les assistants personnels existent depuis le début des années 90. Le premier appareil a été commercialisé par Apple en 1993. Au début, ces appareils n'offraient qu'un nombre limité de fonctionnalités : agenda, répertoire/carnet d'adresses, bloc-notes, calculatrice/convertisseur, alarme/réveil, liste de tâches. Depuis, ils se sont enrichis de nombreuses fonctions multimédias,telles que la lecture de musique, ou de vidéo, et permettent même de naviguer sur internet. Pourtant, les moyens d'interaction hommes machines sur les assistants personnels n'ont guère évolué depuis leur apparition : écran tactile utilisable avec stylet ou doigt, reconnaissance vocale, clavier, boutons. Le moyen le plus utilisé est le stylet

Worldwide total smart mobile device market
Market shares Q2 2005, Q2 2004

Vendor	Q2 2005 shipments	% share	Q2 2004 shipments	% share	Growth Q2'05/Q2'04
Total	**12,185,600**	**100.0%**	**5,933,330**	**100.0%**	**105.4%**
Nokia	6,695,800	54.9%	1,967,550	33.2%	240.3%
Palm	1,057,420	8.7%	1,067,880	18.0%	-1.0%
RIM	897,280	7.4%	487,600	8.2%	84.0%
Motorola	556,050	4.6%	75,450	1.3%	637.0%
Fujitsu	527,890	4.3%	146,830	2.5%	259.5%
Others	2,451,160	20.1%	2,188,020	36.9%	12.0%

Source: Canalys estimates, © canalys.com ltd. 2004-2005
Smart mobile device market: handhelds, wireless handhelds, smart phones

FIG. 2.1 – Marché mondial des appareils mobiles intelligents selon le vendeur

que ce soit en tant que souris ou en tant que moyen d'écriture, grâce à la reconnaissance calligraphique.

Les deux principaux systèmes d'exploitation sur le marché des PDA's sont PocketPC et PalmOS. Symbian et Linux sont plus marginaux.

En ce qui concerne les IHM des PDA's, elles sont fortement dépendantes du système d'exploitation utilisé. L'interface de PalmOS a été étudiée spécialement pour les assistants personnels et est très simple d'utilisation. Les concepteurs de PocketPC ont calqués l'interface utilisateur sur celle des postes fixes utilisant Windows. Elle est donc plus familière pour l'utilisateur.

Le marché Le marché des smartphones et autres produits hybrides a progressé de 186% en un an tandis que les PDA's voient une augmentation de 18% qui, d'après l'institut Canalys, serait dû à la fonction de GPS intégrée.

Les figures 2.1 et 2.2 montrent que Nokia est le leader du marché des smartphones avec l'OS Symbian. PalmOne est majoritaire sur le marché des PDA's avec 33%.

2.2 Comparaison des interfaces selon les appareils

Une des principales différences entre PalmOs et PocketPC est que PalmOS a été conçu spécialement pour les appareils mobiles, sans base préalable tandis que l'interface du Pocket PC veut fournir aux utilisateurs de Windows sur poste fixe un environnement qui leur est familier. Il propose une suite bureautique avec, entre autres, PocketWord, PocketExcel ou encore PocketOulook. L'approche adoptée par Palm est très consistente et basée sur

Worldwide total smart mobile device market
Market shares Q2 2005, Q2 2004

OS vendor	Q2 2005 shipments	% share	Q2 2004 shipments	% share	Growth Q2'05/Q2'04
Total	**12,185,600**	**100.0%**	**5,933,330**	**100.0%**	**105.4%**
Symbian	7,648,920	62.8%	2,429,930	41.0%	214.8%
Microsoft	1,931,630	15.9%	1,360,220	22.9%	42.0%
PalmSource	1,157,720	9.5%	1,335,810	22.5%	-13.3%
Others	1,447,330	11.9%	807,370	13.6%	79.3%

Source: Canalys estimates, © canalys.com ltd. 2004-2005
Smart mobile device market: handhelds, wireless handhelds, smart phones

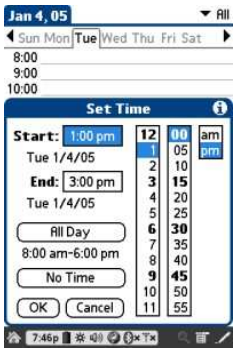
FIG. 2.2 – Marché mondial des appareils mobiles intelligents selon l'OS

un modèle conceptuel très simple. Il n'y a qu'une ou deux façons de réaliser une tâche. Il est donc très facile pour un utilisateur de prédire le résultat d'une action effectuée. L'interface qui en résulte est très utilisable. Pocket PC, au contraire, utilise une interface dite permissive. Il existe différentes façons de réaliser une tâche donnée. Cette approche est basée sur le concept que toute action raisonnable de la part de l'utilisateur donnera un résultat raisonnable. Cependant, l'utilisateur peut être désorienté s'il y a trop de modèles conceptuels différents.

Une étude de [Hübscher-Younger et al, 2001] a mesuré le temps mis par l'utilisateur pour effectuer une série de tâches sur Palm ou sur Pocket PC. Il résulte de cette étude que PalmOS est significativement plus efficient que PocketPC avec des temps beaucoup plus courts, et ce pour toutes les tâches. Il est plutôt surprenant que malgré que la plupart des participants sont des utilisateurs de Windows et l'interface de Pocket PC a été créée pour être familière à ces utilisateurs, l'utilisation du Palm semble beaucoup plus simple. Une réponse à cette différence peut se trouver dans le fait que les contextes d'utilisation d'un appareil mobile ou d'une station fixe sont complètement différents.

Les besoins des utilisateurs et les contextes d'utilisation d'appareils mobiles diffèrent totalement des utilisateurs fixes. Un utilisateur mobile ne focalise pas toute son attention sur son écran tout le temps, il est distrait par son environnement, par des événements qui se déroulent autour de lui.

L'approche de Palm est bien plus adaptée à l'utilisation et aux besoins des utilisateurs mobiles. PocketPC est parfait pour des actions complexes mais ce ne sont pas ces dernières qui sont effectuées le plus souvent sur les appareils mobiles.



2.3 Problèmes spécifiques aux appareils mobiles

Un thème commun à beaucoup de travaux passés sur les appareils mobiles est le désir d'exécuter sur ceux-ci des environnements informatiques similaires à ceux qui sont exécutés sur des postes fixes. Bien qu'exécuter beaucoup de ces mêmes applications peut être utile et souhaitable, exécuter le même environnement peut être ni souhaitable et ni même possible pour beaucoup d'appareils.

La plupart des problèmes qui se posent avec la mobilité existaient déjà avant : faible résolution, déconnexion réseau, petitesse de l'écran. Un problème nouveau qui se pose est la conception d'application pouvant tourner sur différentes plateformes. Une application doit rester cohérente avec elle-même pour une utilisation sur station fixe ou sur appareil mobile. Il faut remettre en question le paradigme WIMP (Windows, Icons, Menus, Pointing devices) qui cible plutôt une utilisation sur poste fixe :

- La taille de l'écran doit être assez grande.
- La manipulation du curseur doit pouvoir se faire de manière précise.
- Le paradigme WIMP implique que le centre d'intérêt de l'utilisateur est constitué des informations affichées à l'écran.

Toute interface-utilisateur peut être considérée comme ayant ces composants :

- Les métaphores sont des concepts fondamentaux qui sont communiqués via des mots, images, sons, et expérences tactiles. Les concepts de pages, ou de Webblogs en sont des exemples. La vitesse de création de nouvelles métaphores augmente grâce au déploiement rapide via l'internet.
- Les modèles mentaux sont les structures ou les organisations de données, fonctions, tâches, rôles, et gens présentes dans un groupe. Cela peut être les fonctions, médias, outils ou tâches hiérarchiques.
- La navigation considère le mouvement à travers les modèles mentaux, c'est à dire, à travers les contenus et outils. Cela inclut les dialogues dits techniques tels que les menus, boites de dialogue ou encore les icônes.
- L'interaction se centre sur les entrées et sorties techniques, en incluant le retour sur une action effectuée. Le clavier, la souris, le stylo ou un microphone sont différents types d'entrée. L'utilisation de la séquence glisser-déposer- sélection-action est un

modèle d'interaction.
- L'apparence concerne ce qui est visuel, auditif et tactiles comme par exemple : le choix des couleurs, les appels sonores ou le mode de vibration.

2.3.1 Apparence

Le premier problème, et sans doute le plus évident, est la taille de l'écran. Cela limite l'affichage d'un certain nombre de données.

Plus encore que la taille, la résolution de l'écran est un problème central. Le problème se pose au niveau de l'affichage des données mais également au niveau de la taille des éléments à afficher. Certains éléments tels que les icônes doivent avoir une taille et un niveau de détails minimals pour permettre à l'utilisateur de les différencier aisément.

Pour encore beaucoup d'appareils, la gamme des couleurs est limitée, or, celles-ci sont d'une grande aide pour mettre en évidence des parties de l'écran ou des informations. La résolution sonore peut également être faible.

Le nombre d'informations auxquelles le consommateur souhaite pouvoir accéder est de plus en plus importants. Par exemple, un lecteur de musique permet le stockage de 2500 items. Cela pose le problème de la visualisation de toutes ces données sur des écrans qui ne permettent pas l'affichage simultané de nombreuses informations.

Les effets d'une mauvaise interface utilisateur sont renforcés par l'environnement d'un petit écran.

2.3.2 Moyens d'interaction

La précision et la facilité d'utilisation des dispositifs d'entrée sont généralement restreintes. Il faut le prendre en compte lors de la conception de l'interface graphique qui devra combler les manques et simplifier au niveau logiciel ce qui est complexe au niveau matériel.

Un autre facteur à considérer est un défi légal qui a émergé avec les appareils mobiles utilisés en voiture ou dans des lieus où le silence est exigé. Les concepteurs doivent être capable de répondre à ces défis avec des solutions d'entrées et sorties multi-modales.

2.3.3 Complexité de la technologie et utilisateurs inexpérimentés

Les avances technologiques et les pressions du marché ont fortement complexifié les produits des technologies d'information et de communication. Ces derniers sont miniaturisés et équipés de multiples caractéristiques. Au début de la distribution des produits, les utilisateurs étaient expérimentés. L'élargissement du marché et la baisse des coûts ont fait entrer de nouveaux consommateurs. Ceux-ci sont peu spécialisés et ne désirent pas être formés à l'utilisation des produits avancés. La communication mobile n'échappe pas aux

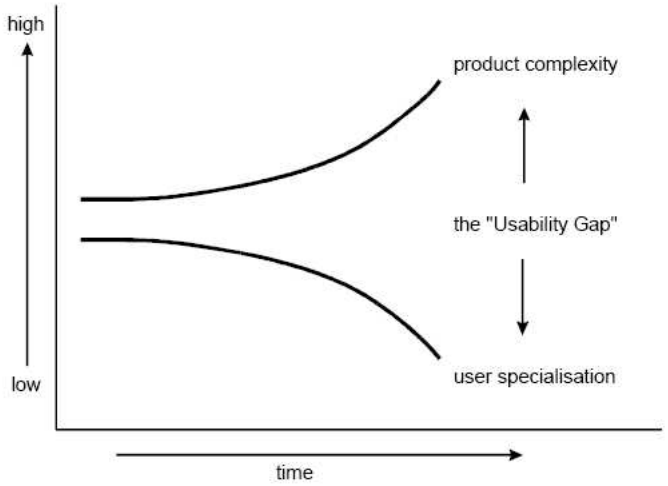


FIG. 2.3 – Evolution du fossé existant entre la spécialisation des utilisateurs et la complexité du produit

observations des deux tendances conflictuelles qui sont la complexification des terminaux mobiles et des technologies sous-jacentes et l'élargissement de la gamme des utilisateurs. De plus en plus d'utilisateurs inexpérimentés au niveau des technologies devront faire face à des terminaux de plus en plus complexes au niveau de la manipulation. Cela est dû au phénomène de miniaturisation et au nombre de caractéristiques croissant telles que le WAP ou le GPS.

Ces tendances ont souvent été observables pour d'autres technologies, et elles ont créé des problèmes qui peuvent être définis comme le "fossé d'utilisabilité", représenté à la figure 2.3.

Pour réduire le fossé, il faut diminuer la complexité perçue par l'utilisateur. Plusieurs moyens existent tels que le développement d'interfaces de qualité, ou d'interfaces intelligentes par rapport au contexte ; la simplicité de la configuration ; la personnalisation possible des capacités ; l'accomplissement aisé des tâches. Les concepteurs doivent fournir des sélections évidentes en soi et des indices quant à la signification des icônes, boutons et autres élélments de l'interface utilisateur pour la navigation et l'interaction.

Les avances technologiques et l'industrie des TIC plus mature fournissent une opportunité pour réduire le fossé d'utilisabilité grâce à des produits qui peuvent être conçu avec une grande flexibilité dans le hardware, le software et le service. Une introduction plus lente de nouvelles caractéristiques permettrait de focaliser l'attention sur les problèmes existants.

2.3.4 Attention minimale et contexte

Beaucoup d'appareils mobiles sont utilisés dans des situations d'attention minimale, où l'utilisateur a seulement une attention limitée et intermittente disponible pour l'interface. Dans de telles situations, les interactions avec le monde réel sont généralement plus importantes que les interactions avec l'ordinateur. Les mains et les yeux des utilisateurs peuvent être occupés autre part tandis que l'utilisateur est occupé à certaines tâches du monde réel, telles que la marche ou la conduite d'un véhicule. Il peut également être distrait par un événement extérieur plus fréquent en dehors d'un bureau.

La volonté de transposer sur des appareils mobiles, des applications destinées d'abord aux postes fixes, ne se matérialise pas toujours de manière aisée. Lors de l'utilisation d'une application sur poste fixe, le contexte est connu. Il peut donc être aisément pris en compte lors de la conception de l'interface utilisateur. Au contraire, le contexte d'utilisation des appareils mobiles est changeant. La conception faite pour le poste fixe ne sera pas idéale pour un appareil mobile. Développer des interfaces mobiles ne se limite donc pas à miniaturiser des interfaces existantes pour poste fixe.

Miniaturiser un site web ou une application bureau pour une utilisation mobile menace de considérer l'environnement mobile et la technologie comme un sous-ensemble de l'environnement du bureau. On peut trouver des arguments pour cette attitude : les langages mobiles sont pour la plupart des sous-ensembles de leurs équivalents bureau, les écrans sont plus petits, les dispositifs d'entrée pour les utilisateurs sont plus limités et la vitesse de connexion est plus lente.

2.3.5 Tâches exécutées et utilisation de l'appareil

De plus en plus, les utilisateurs ont besoin d'effectuer des opérations entre deux ou plusieurs appareils. Même pour des appareils ayant une interface utilisateur bien conçue, ce genre de tâche peut être difficile à effectuer.

Une analyse des tâches de l'informatique mobile peut produire un ensemble très différent de priorités des applications par rapport à une analyse de l'utilisation d'une station de travail typique de bureau. Cela produira certainement un ensemble de tâches, différent de celles réalisées par un ordinateur typique scientifique. Par exemple, une analyse peut indiquer que les tâches de base réalisées sur des appareils mobiles sont la planification, la prise de courtes notes, la consultation et la composition de courrier électronique, naviguer dans de grandes bases de données, et remplir des formulaires. On peut comparer cet ensemble de tâches avec les tâches typiquement réalisées sur un poste de travail par les chercheurs : préparation de gros documents, programmation, consultation et composition d'email, et création de présentations graphiques. Différents ensembles de tâches utilisateur peuvent affecter les décisions de design des systèmes. La plupart des tâches des postes de travail impliquent un importante quantité de données en entrée, tandis que les tâches mobiles impliquent principalement de petites quantités de données en entrée et la présentation d'informations existantes.

Le PDA ne s'utilise pas de la même manière qu'un ordinateur portable. La durée d'utilisation d'un PDA est beaucoup plus courte tandis que le nombre d'utilisation est plus élevé. Ce qui signifie que l'utilisateur du PDA allume ce dernier pour entrer une information ou y accéder, pour une action simple. Il utilisera son ordinateur portable pour des actions plus complexes, puisque ce dernier propose un clavier et un plus grand confort d'utilisation pour ce type d'action. Les informations doivent donc pouvoir être interprétées rapidement et facilement par les utilisateurs.

2.4 Conception d'interfaces pour PDA

Le problème de la taille de l'écran couplé au nombre important d'informations amène à penser à un système hiérarchique avec un contenu catégorisé et ordonné, qui permettrait de naviguer et de sélectionner les informations désirées de manière efficiente. Si le contenu peut être soumis à une recherche alphabétique, chronologique, géographique, ou numérique, cela peut être suffisant pour localiser rapidement le contenu désiré. Cependant, dans beaucoup de cas, la structure des listes hiérarchiques ou des contenus de réseaux doit être examinés. Au niveau de la navigation, l'agencement hiérarchique des écrans ne devrait pas contenir un nombre de niveaux trop élevé.

Une approche a été développée pour résoudre ce problème sur des appareils se portant au poignet (Samsung, AnyCall). Le design propose une liste ayant des caractéristiques clés connues de l'utilisateur et montrées par de petits symboles. Une ligne élargit le texte et permet à l'utilisateur de lire le contenu courant.

Lorsqu'on conçoit une interface pour PDA, un concept incontournable est la simplicité. La surcharge d'informations à l'écran entrave l'utilisateur dans la manipulation du logiciel. L'interface du PDA doit répondre aux besoins de rapidité d'accès, d'interaction et de feedback. Elle doit permettre à l'utilisateur de trouver et d'accéder aux informations de la manière la plus efficace possible. Il faut donc faire un tri dans les informations et sélectionner les plus essentielles. Il faut ensuite les agencer de façon claire, concise et pertinente.

L'écran doit être compréhensible, c'est à dire que ses éléments sont faciles à interpréter et à identifier, mais il doit également être agréable. On parle de lisibilité de l'écran. Par exemple, un moyen d'améliorer la lisibilité est de choisir une typographie adéquate qui soit facile à déchiffrer.

Un des principaux problèmes des appareils mobiles est la faible résolution de l'écran. Il y a deux méthodes possible pour contourner cette contrainte en ajustant la taille des interacteurs. La première possibilité est simplement, réduire ces interacteurs. Cependant, des expériences sur l'utilisabilité ont déterminé que la taille minimale pour une icône était de huit pixels sur six. Beaucoup d'interacteurs ne peuvent pas être réduits de manière significative. Une alternative à la réduction de la taille est de remplacer l'interacteur par un élément prenant moins de place.

Le concepteur de l'interface doit avoir une certaine empathie avec l'utilisateur de ma-

nière à ce que la conception que celui-ci a d'une interface utilisateur et des actions à effectuer corresponde avec l'interface conçue.

L'attention limitée de l'utilisateur nécessite que les interfaces aidant les utilisateurs à terminer une tâche. Cela peut être une simple aide ponctuelle sur des points critiques.

L'effort de l'utilisateur doit être limité au maximum et ce pour deux raisons. Premièrement, l'attention de l'utilisateur n'est pas concentrée totalement sur l'appareil. Deuxièmement, les interactions doivent se faire de la manière la plus efficiente possible et dans une optique d'économie de temps.

2.5 Etat actuel des produits

Le marché des appareils mobiles montre beaucoup d'innovations mais également beaucoup de chaos. De nouveaux produits font leur apparition et disparaissent.

Beaucoup de produits peuvent faire face aux défis des modèles de business en plus de la qualité de l'expérience utilisateur et du design de l'interface utilisateur. Pour beaucoup d'appareils mobiles et d'informations, la lisibilité de l'écran demeure modeste. Même si beaucoup de PDA's couleur peuvent afficher des pages Web miniatures, le contenu n'est pas affiché de manière à ce qu'il puisse facilement être lu. Dans beaucoup de téléphones ayant une connexion internet et possédant un petit écran, les caractères sont limités en nombre et lisibilité, en plus d'un contraste faible. Souvent, moins de sept lignes de texte peuvent être affichées. Il n'est pas surprenant qu'une récente étude sur les téléphones WAP connecté au Web par le groupe Nielsen Norman montre une très faible utilisabilité et une faible adoption de ces appareils par les utilisateurs.

Peu de produits mettent en pratiques les connaissances tirées des recherches sur l'utilisabilité dans le design des interfaces utilisateur depuis les dernières décennies.

Le pouvoir des appareils mobiles va de l'appareil tenant dans la main vers des versions plus petites se portant au poignet. Cette place semble l'endroit idéal pour les fonctions informatiques et de communication. Les consommateurs ont déjà l'habitude de porter une montre-bracelet. Malheureusement, le design de ces appareils semble plus approprié à des personnages de science-fiction.

FIG. 3.1 – Correspondances entre les types d'appareils et les plate-formes adaptées

Chapitre 3

Technologies

3.1 J2ME : Généralités

En 1999, Sun Microsystems présente J2ME à la conférence des développeurs JavaOne. La plate-forme J2ME cible des appareils qui ont comme contraintes : une interface utilisateur et un affichage limités, une faible mémoire, une alimentation sur batterie et un encombrement et un poids faibles. La figure 3.1 montre les différentes plates-formes et le type d'appareil qui les supporte.

De nombreux appareils supportent J2ME, comme le montre la figure 3.2

L'architecture de J2ME, représentée sur le schéma de la figure 3.3 se base sur l'utilisation de configurations et de profils. Ceux-ci sont choisis en fonction des spécifications de l'appareil sur lequel on veut développer.

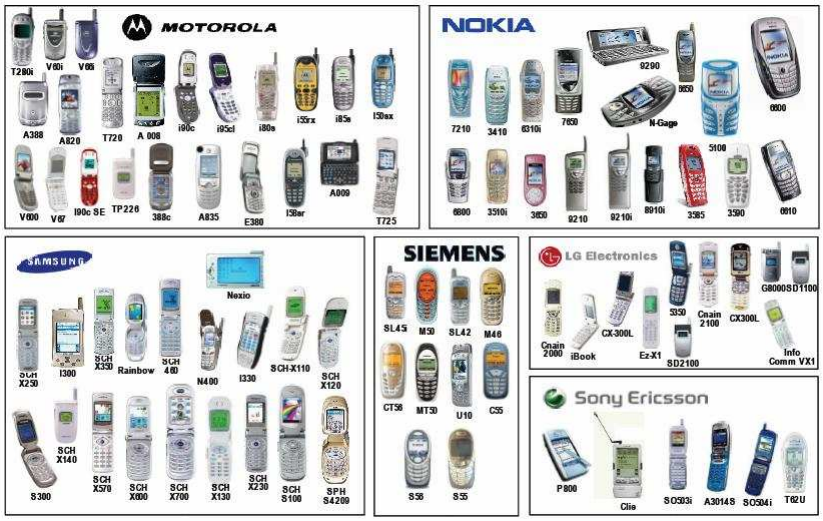


FIG. 3.2 – Echantillon d'appareils supportant la plate-forme J2ME

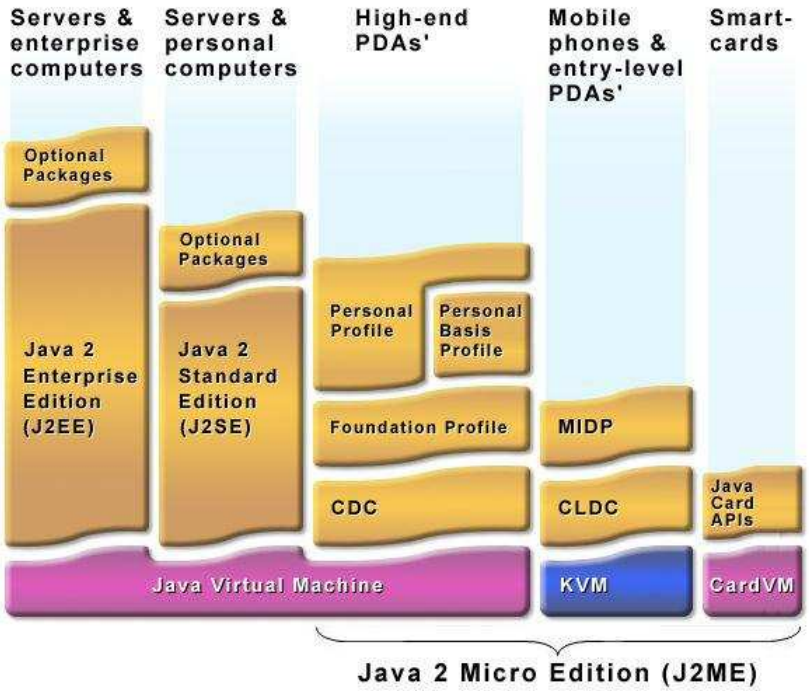


FIG. 3.3 – Architecture générale de J2ME

3.2 KVM - K Virtual Machine

La KVM est une machine virtuelle Java portable et compacte prévue pour des dispositifs avec des ressources limitées, tels que les téléphones portables, organizers personnels,

appareils ménagers, distributeurs automatiques, etc... L'objectif principal d'une telle machine virtuelle est d'être la plus petite possible (40 à 80 ko) tout en préservant l'aspect du langage de programmation Java. La KVM n'est pas implémentée que par Sun Microsystems et a déjà été portée sur plusieurs dizaines de dispositifs par des constructeurs agréés par Sun. Pour les dispositifs ne possédant pas d'interface utilisateur capable de lancer des applications, il existe un gestionnaire d'applications Java qui joue le rôle d'interface entre le système d'exploitation hôte et la machine virtuelle.

3.3 Configurations

La configuration détermine une plate-forme minimale pour une catégorie d'appareils qui ont des besoins analogues :

- type et capacité de la mémoire totale
- type de connexion réseau disponible pour l'appareil
- type et vitesse de processeur.

Chaque configuration consiste en une machine virtuelle et un ensemble de classe qui fournit un environnement de développement pour les applications logicielles.

Il existe deux configurations définies : Connected Device Configuration (CDC) et Connected Limited Device Configuration (CLDC).

3.3.1 Connected Device Configuration (CDC)

Cette configuration s'adresse aux appareils qui se situent entre les appareils à faibles ressources et les postes de travail fixes. Ce sont les appareils grand public haut de gamme. Typiquement, ces appareils ont une mémoire supérieure à 2MB, des processeurs de capacité élevée et une connexion réseau permanente avec une large bande passante Exemples : téléviseurs et visiophones Internet, systèmes de navigation et de loisirs embarqués.

3.3.2 Connected Limited Device Configuration (CLDC)

Cette configuration s'adresse aux appareils à faibles ressources dont les caractéristiques matérielles sont les suivantes : une mémoire entre 128 Ko et 512 Ko disponible pour la plate-forme Java, un processeur de 16 bits ou 32 bits, une alimentation par batterie et une connexion intermittente avec une bande passante limitée. Exemples : téléphones portables, messagers de poches, organizers personnels, scanners de code barres.

3.4 Profils

Un profil complète une configuration en ajoutant des classes supplémentaires qui fournissent une plate-forme appropriée à une famille d'appareils. L'objectif est de donner plus de flexibilité pour maintenir la portabilité des applications entre les terminaux Les profils sont définis par le "Java Community Process". MIDP et PDAP sont des profils définis pour la configuration CLDC. Foundation Profile, Personal Basis, Personal Profiles et RMI Profile sont des profils définis pour la configuration CDC.

- *Mobile Information Device Profile*
  MIDP est le premier profil qui a été développé dont l'objectif principal est le développement d'applications sur des machines aux ressources et à l'interface limitées.
  MIDP est basé sur la configuration CLDC. Ses fonctionnalités principales sont l'interface client, la persistance de stockage, la mise en réseau, et l'application modèle. Il permet la gestion de l'interface utilisateur, la gestion de l'interface réseau, et la gestion d'une base de donnée sur le mobile.
- *PDA Profile*
  PDAP est proche de MIDP mais est plutôt destiné aux PDAs plus performant. Cependant ce profil a été abandonné en tant que tel, il se présente maintenant sous forme de 2 packages optionnels. Au niveau de l'interface, il utilise un sous-ensemble d'AWT.
- *Foundation Profile*
  FP ajoute des classes à CDC pour inclure les principales librairies de la version 1.3 de Core Java 2. Comme son nom l'indique, FP est un profil utilisé comme base pour la plupart des autres profils de CDC.
- *Personal Basis et Personal Profiles*
  Le Personal Basis Profile ajoute des fonctionnalités d'interfaces utilisateur basiques au Foundation Profile. Il ne permet pas à plus d'une fenêtre d'être active au même moment. Les plates-formes qui peuvent supporter une interface utilisateur plus complexe utilise le Personal Profile. Ces profils utilisent des sous-ensembles d'AWT pour la programmation d'interfaces.
- *RMI Profile*
  RMI Profile ajoute les librairies Remote Method Invocation de J2SE au Foundation Profile.

3.4.1 MIDP

MIDP fait partie du groupe de spécification JSR 037(MIDP pour les plateformes J2ME). Il existe deux versions : MIDP 1.0 et MIDP 2.0. Il est avant tout destiné aux téléphones mobiles et aux PDA bas de gammes. Il fournit des API pour le développement d'interfaces graphiques utilisateur, la connectivité réseau, le stockage local de données et la supervision d'applications. L'API du MIDP se compose des API du CDLC (Connected Limited Device Configuration) et de 3 packages :

- javax.microedition.midlet : cycle de vie de l'application. Socle technique destiné à gérer le cycle de vie des MIDlets.
- javax.microedition.lcdui : interface homme/machine. Fournit la gestion de l'interface utilisateur telle que les contrôles.
- javax.microedition.rms : base de données persistante légère.

Les MIDlets

Une MIDlet est une application java qui s'exécute sur des appareils MIDP.

Toutes les applications pour le profil MID doivent être dérivées de la classe MIDlet qui gère le cycle de vie de l'application. Cette classe appartient au paquet javax.microedition.midlet. Elle permet le dialogue entre le système et l'application.

Pour gérer ce cycle de vie, la classe MIDlets défini les trois méthodes ci-dessous.

- startApp() ; : cette méthode est appelée à chaque démarrage ou redémarrage de l'application.
- pauseApp() ; : cette méthode est appelée lors de la mise en pause de l'application
- destroyApp(boolean unconditional) ; : cette méthode est appelée lors de la destruction de l'application. Si le boolean a comme valeur "false", la MIDlet peut ne pas se terminer et déclarer une MIDletStateException.

Ces trois méthodes doivent obligatoirement être redéfinies dans chaque MIDlet.

Le cycle de vie d'une MIDlet, représenté sur le schéma de la figure 3.4 comporte quatre états : loaded, active, paused, destroy. Quand une MIDlet est chargée sur un appareil et que le constructeur est appelé, cela se passe dans l'état "loaded". Ca peut être à n'importe quel moment avant l'appel la méthode starApp(). Après cet appel, la MIDlet passe à l'état "active". La méthode pauseApp() met la MIDlet en pause ce qui permet de libérer des ressources inutiles à la MIDlet dans cet état. Ca évite des conflits et une consommation inutile de la batterie. La méthode destroyApp() termine la MIDlet.

Le background et le foreground sont les deux états possibles d'une application. Une application en foreground consomme de l'énergie et c'est celle qui est exécutée. Les applications qui sont en attente sont en background et ne consomme pas d'énergie. La méthode pauseApp() met l'application en background.

Les API interface utilisateur

La classe `Display` est la base de toute interface utilisateur d'une MIDlet. Cependant, elle ne possède que peu de fonctionnalités et seule, elle n'est pas très utile. Il y a donc des sous-classes plus utiles. Les deux classes directement en dessous de `Displayable` sont `Screen` et `Canvas`. Elles forment les deux API's graphiques de MIDP et sont représentées à la figure 3.5.

L'API de haut niveau permet un niveau d'abstraction élevé et donc une plus grande portabilité. Elle est orientée écran et widget. Il y a peu de contrôle possible au niveau du

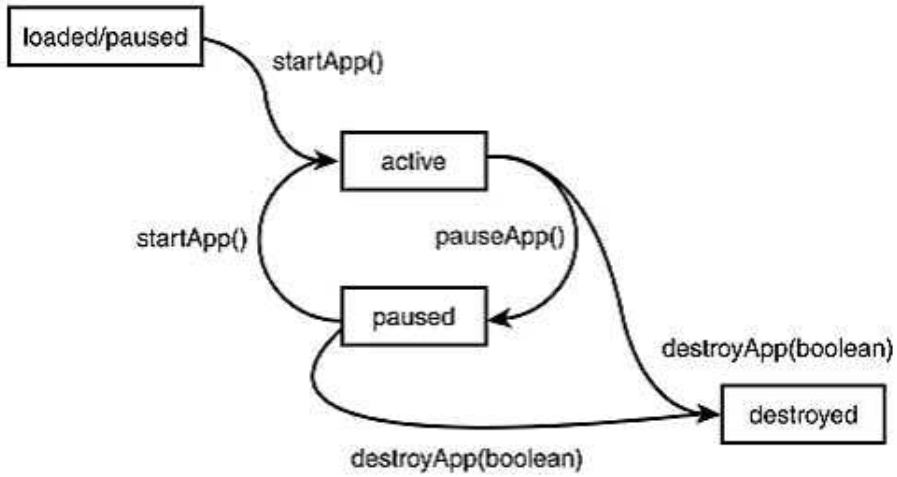


Fig. 3.4 – Cycle de vie d'une MIDlet

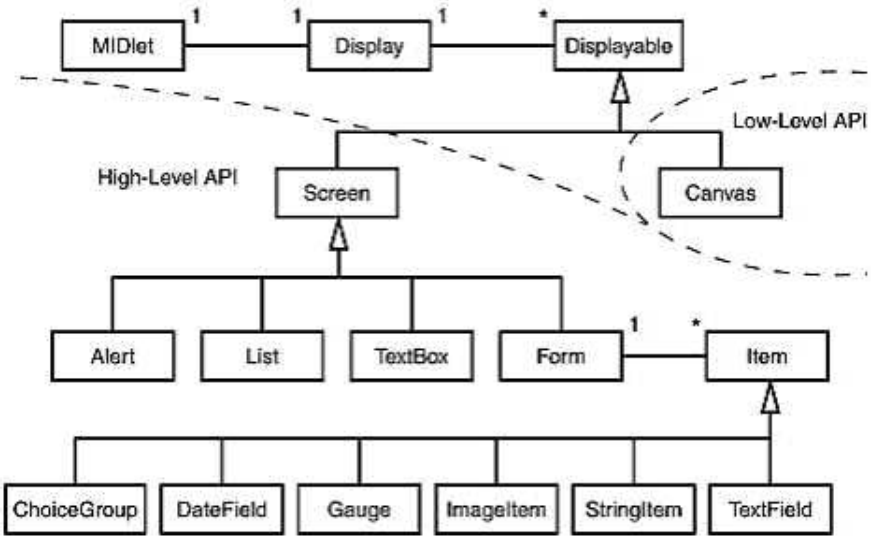


Fig. 3.5 – Représentation des différentes classes graphiques

look-and-feel. On ne peut donc pas agir sur l'apparence visuelle (couleurs, tailles, entrée) des composants de haut-niveau. Tout ça est géré par l'implémentation MIDP. Il est plus simple et plus puissant qu'AWT. Les classes qui implémentent l'API de haut niveau héritent

toutes de la classe javax.microedition.lcdui.Screen.

L'API de bas niveau offre un contrôle beaucoup plus grand de l'apparence visuelle. Il offre des primitives de dessins et permet de dessiner pixel par pixel l'interface. Cette API est définie pour les applications, telles que les jeux, qui ont besoin d'un tel contrôle des éléments graphiques et d'un accès aux événements de bas-niveau. En contrepartie, la portabilité de ces applications est très faible puisque l'accès à des détails spécifiques au niveau hardware est permis. Les classes javax.microedition.lcdui.Canvas et javax.microedition.lcdui.Graphics implémentent l'API de bas niveau.

3.4.2 PDAP

Dans une MIDlet (ou PDAlet), les composants d'AWT et du paquet javax.microedition.lcdui peuvent être utilisés dans la même application mais seul un type de composant peut être visible à l'écran à un moment donné. Si un screen lcdui se trouve dans le focus de l'entrée, il cachera toutes les frames AWT, et inversement.

Le listing 1 montre une MIDlet vide et le listing 2 converti cette MIDlet vide en une PDAlet vide en appellant la méthode getDefaultToolkit() dans la méthode startApp(). L'outil par défaut est un sous-ensemble d'AWT et est utilisé pour créer une interface graphique utilisateur pour la PDAlet. La méthode getDefaultToolkit() rend disponible l'API PDAP à la PDAlet.

Le profil PDA utilise les éléments d'interface communs d'AWT comme la solution pour les interfaces utilisateurs pour PDAlets.

AWT utilisé dans PDAP est un sous-ensemble de l'AWT de J2SE qui est "designé" pour utiliser l'espace limité de l'écran et les contraintes de mémoire des PDA. Ce sous-ensemble contient toutes les caractéristiques requises pour développer une application PDA.

Il est préférable de tester l'interface sur toutes les plates-formes sur lesquelles l'application est destinée à tourner pour s'assurer que les choix établis pour l'interface n'ont pas d'impacts négatifs sur l'application. Chaque plate-forme a des caractéristiques différentes.

Lorsque la plate-forme ne gère pas certains éléments, l'implémentation ne renvoie pas d'erreur. L'application continuera son exécution. Elle ignore silencieusement la ou les méthodes concernant le changement. Les méthodes principalement non supportées par certaines plates-formes sont surtout la gestion des fenêtres telles que les frames, fenêtres et boîtes de dialogue au niveau du redimensionnement et du positionnement.

Certaines plates-formes ne supportent que le noir et blanc tandis que d'autres supportent une palette de couleurs plus large. Cependant, ces dernières peuvent ne pas permettre le changement de couleurs des éléments GUI standards tels que les menus ou les barres de titres.

Certaines implémentations interdisent la personnalisation du pointeur que ce soit par le développeur ou par l'utilisateur.

Il est préférable d'éviter l'utilisation de fonts différents dans la GUI. Si on veut utiliser

des fonts différents, il faut s'assurer que la plate-forme les supportent bien tous et que l'application ne tournera pas sur d'autres implémentations.

Concernant la gestion des fenêtres (les frames, fenêtres, et boîtes de dialogues), il existe également des restrictions ou interdictions sur certaines plates-formes. Elles concernent en général les points suivants.

- Certaines implémentations permettent l'affichage de plusieurs fenêtres actives au même moment, mais pas toutes.
- Le redimensionnement est limité ou interdit dans beaucoup d'implémentations. Le redimensionnement concernant les boîtes de dialogue peut amener un résultat différent de celui espéré : une taille différente ou aucun changement. Il faut donc éviter au maximum les besoins de redimensionnement des boîtes de dialogue.
- Le positionnement peut également faire l'objet de restrictions. Il n'est alors pas possible de déterminer exactement l'emplacement de la boîte de dialogue. Il peut y avoir différentes restrictions ou simplement une complète interdiction.

Les méthodes `setTitle()` et `getTitle()` n'ont pas toujours d'effet sur la boite de dialogue ou sur le frame.

KAWT

kAWT est utilisable avec MIDP4Palm.

kAWT est un sous-ensemble d'AWT qui tourne sur la KVM contrairement à AWT.

Le but du projet kAWT est de fournir une version simplifiée d'AWT pour la KVM : les classes originales com.sun.kjava incluses dans J2ME CLDC Beta1 et les versions EA plus anciennes de la KVM diffèrent des composants d'interfaces utilisateur Java standards par bien des aspects. Porter des applications sur PDAs devient un peu compliqué. Le problème le plus important est que seul le support limité pour la gestion d'événements est fourni par la KVM. Cette version simplifiée d'AWT ne laisse pas tous les programmes AWT s'exécuter sur le Palm sans adaptation.

Chapitre 4

Le tutoriel

Le but premier du tutoriel est d'introduire les étudiants au développement d'IHM sur des appareils mobiles. Cela implique de prendre en compte les difficultés particulières d'utilisabilité de ceux-ci mais également les problèmes pédagogiques. La méthode proposée était l'apprentissage par les exemples, chaque exemple étant expliqué pas à pas. Ce chapitre présente la problématique de l'apprentissage et la manière dont elle s'applique au tutoriel.

4.1 Définitions

Tutoriel Initiation guidée à l'utilisation d'un ensemble de notions ou d'une technique.

Didacticiel Logiciel interactif destiné à l'enseignement ou à l'apprentissage, et pouvant inclure un contrôle de connaissance.

4.2 Problématique

Lors de la création d'une formation, il faut tenir compte de plusieurs paramètres, comme le montre le schéma pédagogique général à la figure 4.1. Ce schéma sous-tend tout processus d'enseignement. Il pose cinq questions auxquelles il faut répondre pour construire la formation.

En lisant le graphique de droite à gauche, la première question posée est celle du "à qui ?". Plusieurs questions sont à se poser : A qui est destiné la formation ; quel public va être intéressé ; qu'est-ce qui va le pousser à suivre cette formation ? Celles-ci vont définir le public dans ses choix et ses objectifs d'apprentissage. Il peut avoir plusieurs raisons de porter son choix vers une formation à distance, cela peut être pour le contenu, la flexibilité ou simplement l'attrait pour les nouvelles technologies.

La question qui vient ensuite est celle du "pourquoi ?". Celle-ci va permettre de définir les objectifs particuliers et généraux. Certains besoins se font sentir et il faut définir la

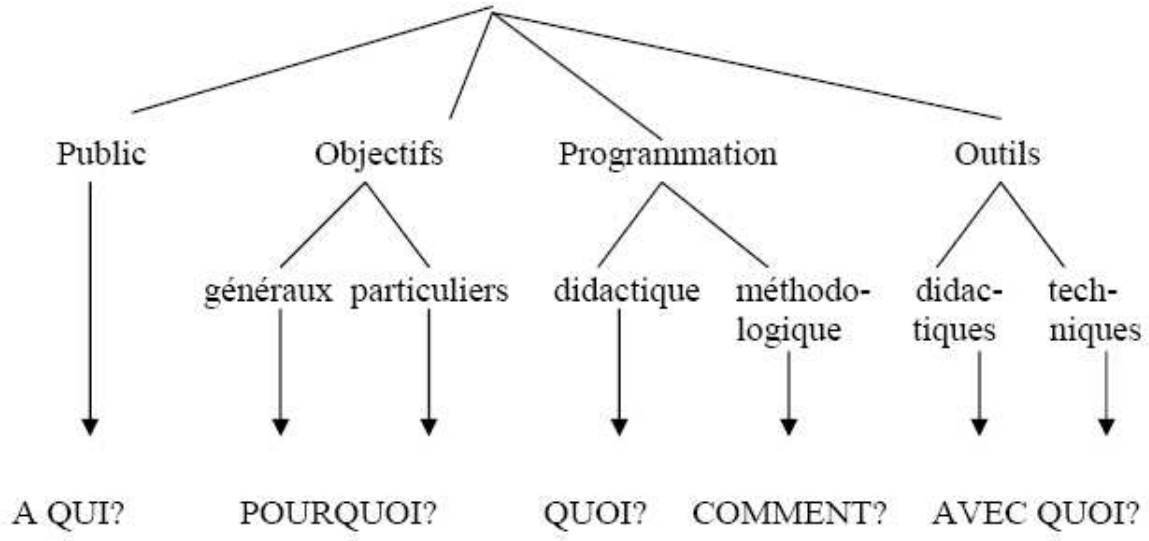


FIG. 4.1 – Schéma pédagogique de T. Cristea, 1984

réponse qui sera apportée par le projet. Le support d'une formation tel qu'un manuel scolaire ou ce tutoriel peut remplir diverses fonctions, c'est durant cette phase qu'elles sont précisées.

La question du "quoi" délimite le contenu de la formation. On détermine la matière qui va être enseignée et inscrite dans le tutoriel. C'est ce qu'on appelle la programmation didactique. Il faut s'assurer que le contenu soit cohérent. Chaque contenu a ses spécificités, comme par exemple, la quantité d'informations ou les relations entre théorie et pratique.

Les stratégies d'enseignement et d'apprentissage répondent à la question "comment". Elles définissent la manière dont les informations vont être enseignées afin de les transformer en connaissances. C'est à cette étape, appellée programmation méthodologique, que l'on détermine la complexité des informations. Le parcours de l'apprenant doit être établi en fonction de son niveau de départ, le temps dont il dispose et son rythme d'assimilation.

La dernière question, "avec quoi", pose les problèmes du support et des outils adéquats à la formation ainsi que leur mise en place. Ce paramètre détermine le type de tutorat s'il y en a un, l'existence de cours en présentiel ou non, ainsi que le contenu vu en présentiel. Le problème de l'évaluation est posé. Il faut définir la manière dont elle va s'effectuer et se poser la question de la nécessité et de l'intérêt d'une méthode d'auto-évaluation.

Une autre question peut être ajoutée, c'est celle des contraintes. Charles Duchâteau énonce, dans son article [**?**], plusieurs contraintes à prendre en compte lors de l'élaboration d'un programme de formation à distance : le public, la matière à enseigner et la disponibilité du formateur.

4.3 Le public visé

Le public visé est toute personne possédant un appareil mobile et souhaitant développer une interface graphique pour ceux-ci. Il peut y avoir des niveaux très différents mais ils seront débutants complets dans le domaine du développement avec le profil MIDP. Cependant, ils doivent avoir des connaissances du langage Java car le but du tutoriel n'est pas d'enseigner les principes de base de la programmation objet. Le langage est très simple mais pas intuitif et étant une extension de Java, il fait appel à des concepts propres à ce langage. Des concepts tels que la structure conditionnelle , l'héritage ou l'encapsulement ne sont pas expliqués et supposés connus. Il est destiné à un public qui ne désire pas apprendre le développement d'interfaces graphiques dans un but professionnel. L'apprenant possède un appareil mobile et il désire développer une application pour celui-ci, on peut donc supposer qu'il est familiarisé avec les technologies. De ce fait, il n'aura pas de problèmes majeurs à entreprendre une formation à distance.

4.4 Objectifs

L'objectif de ce tutoriel est d'introduire rapidement l'étudiant dans le développement d'interfaces utilisateur sur PDA et sur d'autres appareils mobiles équipés d'écrans de petite taille tels que les téléphones mobiles ou les smartphones. Ces derniers prennent un essor considérable. Ils possèdent des écrans plus petits que les assistants personnels. Ils ont également des dispositifs d'entrée plus restreints. Il en résulte donc plus de contraintes au niveau des interfaces utilisateur. Ce tutoriel n'entre pas dans le cadre d'une formation diplômante, mais bien d'une formation "à la carte".

Seul le fonctionnement du langage spécifique aux appareils mobiles et dans le cadre du développement d'interfaces graphiques utilisateur est exposé. Ce contenu, très ponctuel, impose à l'utilisateur de trouver un autre moyen pour apprendre à développer le reste de son application.

Le but du tutoriel est de proposer le plus d'informations différentes possibles de telle manière que l'apprenant autonome puisse chercher l'information désirée. Il n'est pas obligatoire de lire toute la partie expliquant le fonctionnement de l'outil mais si l'apprenant le désire, l'information est à sa disposition. La partie sur les guidelines est également au service de l'apprenant.

Etant donné que le tutoriel est destiné à un apprentissage de type autoformation, il est essentiel de diminuer les obstacles que le nouveau développeur pourrait rencontrer et qui risqueraient de le faire abandonner. Le choix du langage a donc un rôle primordial dans le tutoriel. Le langage devrait être assez simple et permettre de créer rapidement une interface. Pouvoir admirer son travail après seulement quelques lignes de code augmente la satisfaction personnelle. Au contraire, devoir écrire des dizaines de lignes et donc risquer de faire plus d'erreurs syntaxiques pour arriver au même résultat, peut être frustrant. Cela joue sur la motivation intrinsèque (cfr page 1.2.4) de l'apprenant, ce qui pourrait le poussser à

abandonner. Le tutoriel propose des exemples et explique les étapes pas à pas pour créer une petite application. L'apprenant peut donc directement se mettre à développer. Il va commencer à construire ses connaissances rapidement dans le processus d'apprentissage.

Fonctions remplies par le tutoriel Dans un document de [FMG94], sept fonctions identifiées dans les manuels scolaires sont appliquées en informatique pédagogique. Il faudra définir lesquelles vont remplir le projet.

1. Fonction de transmission de connaissances
2. Fonction de développement de capacités et de compétences : La découverte d'un langage tel que LOGO en est un exemple.
3. Fonction de consolidation de l'acquis, ou d'exercice
4. Fonction d'évaluation de l'acquis : Deux autre fonctions s'y rattachent :
   - Fonction de diagnostic pour l'analyse de l'erreur et l'identification des causes probables.
   - Fonction de remédiation proposant des activités permettant la correction de l'erreur.
5. Fonction d'aide à l'intégration des acquis
6. Fonction de référence
7. Fonction d'éducation sociale et culturelle

Un manuel scolaire poursuit rarement une fonction unique, un projet d'informatique pédagogique suit le même schéma. Il y a, en général, une fonction principale enrichie par une ou plusieurs fonctions secondaires.

Le tutoriel poursuit plusieurs de ces fonctions. La première et la plus évidente est le développement de capacités et de compétences. La compétence à acquérir est de pouvoir développer une interface graphique sur un appareil doté d'un petit écran en utilisant le langage enseigné. Une deuxième fonction est celle de la transmission de connaissances. Le tutoriel ne poursuit aucune autre fonction.

4.5 Programmation didactique - Choix du contenu

Une étape indispensable à l'établissement du tutoriel est l'analyse du contenu. Elle consiste à déterminer la matière à traiter tant au niveau de la difficulté que de l'étendue. Il faudra ensuite analyser ces contenus afin de les structurer de manière logique et rationnelle.

Une interface graphique est un sujet délicat car il faut choisir entre créativité et utilisabilité. Ce choix est d'autant plus critique pour les applications destinées aux petits écrans que l'utilisabilité est importante et que la créativité peut encombrer un écran qui n'a pas besoin de cela. Le choix du langage a été fait également en ce sens.

4.5.1 Contraintes

Les utilisateurs d'appareils mobiles peuvent posséder plusieurs appareils différents. S'ils développent une application, il est préférable qu'elle soit portable et donc qu'elle puisse tourner sur les différentes plates-formes.

Les développeurs sont débutants et n'ont que peu de notions de la programmation. Il leur faut donc un langage assez simple.

4.5.2 Les langages

Plusieurs langages pouvaient être utilisés mais les deux principaux sont Java et C++. Ce sont les deux langages les plus usités. Ils ont chacun leurs avantages. Le premier, Java, est portable grâce aux machines virtuelles existant sur une multitude de plate-forme. De plus Java possède des interfaces utilisateurs performantes et également portables sur des appareils divers. Le principal handicap de Java est sa vitesse, en effet, vu qu'il s'agit d'un langage interprété ses performances s'en font ressentir.

Le second est C++. C'est en quelque sorte l'opposé de Java. En effet, C++ est rapide mais par contre difficilement portable.

Le langage utilisé à l'université est Java. Ce fut le choix adopté. Java possède une plate-forme spécifique aux appareils à fortes contraintes telles que la taille de l'écran, une interface limitée ou encore une mémoire faible, appellée J2ME. Le profil de cette plate-forme adapté aux téléphones mobiles, smartphones et assistants personnels à faibles ressources se nomme MIDP. Le grand avantage de Java est qu'il est portable grâce aux machines virtuelles proposées. Ce qui permet à ce tutoriel d'être utile pour des applications destinées à PocketPC, PalmOS, Symbian ou Linux.

4.5.3 Choix du profil

Le profil adopté est MIDP, les autres utilisant un sous-ensemble d'AWT pour le développement d'interfaces. La première question portait donc sur le choix entre AWT et MIDP.

Pour la conception d'interfaces graphiques utilisateur, on peut utiliser des sous-ensembles d'AWT. Comme expliqué dans le chapitre précédent, ils sont étudiés pour être utilisés sur des appareils mobiles de haut niveau. Cependant, il existe de nombreux tutoriels destinés à l'apprentissage d'AWT. Il est sans doute plus intéressant d'inculquer aux programmeurs novices des principes d'utilisabilité et de design. Ceux-ci sont particulièrement importants dans le cas des appareils mobiles à cause de la petitesse des écrans qui fait l'effet d'une loupe sur des problèmes qui existent également sur des postes fixes mais sont moins importants.

MIDP est très simple d'utilisation et demande un apprentissage beaucoup moins long. Chaque élément a une place déterminée par la plate-forme. MIDP est très rigide au niveau de la créativité, mais permet d'avoir un écran qui va à l'essentiel et utilisable. MIDP est

destiné aux PDA's de bas niveau ainsi qu'aux téléphones et autres appareils de faible puissance. Le choix pour l'un ou pour l'autre dépend principalement des ressources de l'appareil.

En effet, AWT a été conçu pour des postes de travail et optimisé pour ceux-ci. Il génère un nombre conséquent d'objets à durée de vie courte tels que les objets d'événements. Cela cause des pénalités de performance significatives dues au garbage collector, à un appareil déjà limité en ressources.

MIDP permet de maximiser la portabilité des applications entre les différents appareils. Les éléments graphique de MIDP ne sont pas des sous-ensembles d'AWT/Swing. L'interaction avec l'utilisateur est basée sur une succession d'écran. MIDP a seulement un "single commandlistener". Il est bon de choisir MIDP pour des appareils à faible ressource tels que des téléphones mobiles ou des PDA's bas de gamme.

Il est aussi possible d'utiliser dans une même application des composants d'AWT et du paquet de développement d'interfaces utilisateur de MIDP (javax.microedition.lcdui). Cependant, les composants ne peuvent pas être affichés au même moment en temps que fenêtre active.

4.6 Approche pédagogique

Le tutoriel se base sur l'approche théorique des comportementalistes. La méthode d'enseignement se centre sur l'exposé, la pratique répétée et le renforcement [JB98a]. Le tutoriel incite l'apprenant à expérimenter par lui-même et donc à prendre une part active dans sa formation, ce qui le rapproche légèrement du cognitivisme. Cependant, la dominante reste clairement comportementaliste.

4.6.1 Caractéristiques de la matière à enseigner

Le développement d'interfaces graphiques utilisateur est une compétence de type production (cfr. page 6). En effet, le développeur est face à une tâche de type résolution de problème. Il doit établir le style visuel de l'interface, les interactions entre les éléments et faire en sorte que cette interface soit utilisable. On est dans le domaine du "faire faire" de la programmation appliqué au cas spécifique du développement d'une interface. Comme vu dans le premier chapitre, une méthode adaptée à l'acquisition de ce type de compétences est un enseignement par les exemples. Dans le tutoriel, l'apprentissage se fait plus précisément par les exemples Cela correspond à l'application d'un savoir-faire. L'étudiant apprend à reconnaître des exemples et à les associer. Cette approche permet une introduction rapide à l'outil.

4.6.2 Enseignement de type autoformation

Le choix de l'autoformation était le choix le plus logique. La formation est de courte durée et est destinée à être une introduction plus qu'une réelle formation en développement d'interfaces graphiques. La complexité peu élevée de la matière à enseigner et la faible durée de la formation ne nécessitent pas un tutorat dans la formation. De plus, la grande flexibilité que permet l'autoformation est un atout pour la matière enseignée qui porte sur un point bien précis. Cette caractérisque de flexibilité permet à un large public d'accéder au tutoriel. En autoformation, il est difficile de connaître le profil du public. L'enseignement est donc moins personnalisé.

Dans la formation à distance, l'autonomie et l'accompagnement de l'apprenant ainsi que le développement des dispositifs de support sont des enjeux majeurs. Pour que l'apprenant garde un taux de motivation élevé, de nombreux facteurs entrent en jeux. La durée de l'apprentissage est, ici, assez court l'apprenant ne verra donc pas sa motivation diminuer à cause d'une formation trop longue. Pour garder la motivation de l'apprenant assez élevée, le tutoriel propose des exemples simples. Il pourra réutiliser les exemples ou des parties d'exemples proposés dans le tutoriel pour d'autres applications. Cela permet à l'étudiant d'avoir des résultats rapides et de ne pas se trouver devant des erreurs syntaxiques ou sémantiques trop fréquentes (cfr page 1.3.3).

Un problème se pose au niveau de la pédagogie. Le tutoriel entre dans l'apprentissage à distance de type autoformation ce qui implique que les problèmes d'autonomie et de motivation inhérent à ce type d'enseignement se posent. L'apprenant doit être impliqué, motivé et attentif mais il est isolé. Cette isolation augmente le risque de décrochage, car il n'y a pas de soutien humain contrairement à l'enseignement en présentiel ou à distance avec tutorat. Cela limite la motivation dite sociale (cfr page 1.2.4).

4.6.3 Apprentissage par l'exemple

L'apprentissage par l'exemple est une méthode qui, peut-on dire, a fait ses preuves. C'est sans doute une des plus vieilles de l'humanité, avec la méthode essai-erreur. En effet, c'est celle utilisée par l'enfant pour comprendre le monde qui l'entoure. Il observe les actions et comportements de l'adulte. Les informations découvertes sont emmagasinées et triées par l'apprenant, pour ensuite les appliquer, une fois les concepts compris et intégrés. Dans cette approche, il s'agit de montrer plutôt que de démontrer. Au départ, c'est une méthode d'apprentissage très lente mais, dans le tutoriel, elle est enrichie d'explications pas à pas qui accompagnent les exemples. Les objectifs sont déterminés avant chaque exemple et permettent de guider l'étudiant dans son apprentissage, afin qu'il comprenne le but commun qui relie chacune des étapes.

4.6.4 Stratégie pédagogique

Plusieurs choix s'offraient quant à la méthode à adopter pour découper le tutoriel en modules. La première option consistait à développer une application complexe découpée en modules allant du plus simple au plus compliqué. La deuxième option consistait à aborder un thème différent dans chaque module avec toujours des applications simples. La troisième option était de développer un petit jeu. Cependant, bien que ludique, cette dernière option était trop large. Il aurait fallu développer les interactions derrière l'interface alors que le but premier du tutoriel est uniquement le développement d'interface graphique. Les deux premières options portent sur l'API de haut-niveau de MIDP, elle est portable et simple d'utilisation. Au contraire, le développement d'un jeu aurait demandé d'utiliser l'API de bas-niveau de MIDP alors que celle-ci n'est pas portable. En effet, elle permet l'accès aux composants matériels. De plus, elle demande parfois, de réinventer la roue. Cette option a donc été écartée. Entre les deux premières options, c'est la découpe en modules abordant chacun un thème qui a été abordé ainsi, l'apprenant peut prendre les modules séparément et revenir facilement sur un point précis. Cela favorise l'autonomie de l'apprenant. Celui-ci peut gérer le déroulement de sa formation mais, tout individu n'est pas égal devant l'autonomie, le tutoriel donne donc une ligne de conduite conseillée. La théorie et la pratique sont fortement imbriquées l'une dans l'autre. En effet, il est conseillé que l'apprenant applique les exemples donnés au fur et à mesure de sa lecture du tutoriel.

Il a été décidé de montrer des exemples simples pour chaque concept expliqué. Après un ensemble de concepts liés, un exemple de code, reprenant tous ou presque tous les concepts vus précédemment, est donné. A la fin du tutoriel, un exemple de code reprend tout encore une fois. L'apprenant voit donc trois fois chaque concept dans des exemples différents, ce qui lui permet de réactiver les connaissances acquises. Cela lui permet également de contextualiser les connaissances et de les retravailler pour inscrire plus profondément en mémoire les informations vues [dVM01]. Ce dernier point permet à l'apprenant de mettre en relation les diverses connaissances acquises et ainsi de pouvoir les structurer et les appliquer lorsqu'il se trouvera devant un nouveau problème.

Les exemples choisis sont toujours des exemples positifs. L'introduction de contre-exemples pourrait dans ce cas-ci semer la confusion chez l'apprenant.

4.7 Les outils

4.7.1 Supports technologiques

La partie explicative du tutoriel est un document qui peut être imprimé sur un support papier. Elle est accompagnée des fichiers des exemples vus dans cette partie. L'apprenant peut donc ouvrir un fichier et effectuer des modifications s'il le désire pour voir comment l'application se comporte. Cela permet une utilisation plus interactive du tutoriel. Comme expliqué dans la théorie constructiviste (cfr page 4), il est important que l'apprenant

construise ses connaissances. Cela peut se faire grâce à un traitement actif des savoirs de la part de l'apprenant.

4.7.2 La production de documents

Un document d'autoformation peut être conçu avec différents types de médias : texte, images fixes, son, image vidéo, simulatons ou encore, hyperliens.

Le format le plus classique des tutoriels pour apprendre un langage est le format "papier", c'est celui adopté ici. Un autre format possible aurait été un site internet proposant du texte, des hyperliens entre les différents modules et vers l'extérieur et des simulations et exercices. Le format PDF choisi est l'exacte image du format imprimé. Sous sa forme électronique, il peut fournir des moyens de navigation entre les parties du document.

Un document d'autoformation doit exposer les objectifs à atteindre, la méthode d'utilisation du document et les différents types d'activités, le lien avec d'autres documents, l'identité du document, la bibliographie et un glossaire.

La production de document soulève des points pouvant être problématiques, ceux-ci se posent particulièrement pour de gros projets de formation qui peuvent avoir des coûts conséquents.

Production "lourde" ou "légère" Une production dite "légère" se pose dans le court terme. La durée d'élaboration des documents est réduite grâce à la reprise de documents existants. Les intervenants sont réduits également et doivent avoir des compétences très diverses.

Une production dite "lourde" doit se positionner dans le long terme. Elle se pose plutôt dans un modèle industrialisé. Elle demande des moyens humains et financiers importants et donc de nouer des partenariats.

Dans le contexte de ce mémoire, la production dite "légère" s'impose de manière très claire, la matière à enseigner et le support retenu étant peu complexes.

L'actualisation Le domaine de l'informatique bouge très vite et le problème de l'actualisation des documents se posera tôt ou tard. Le document propose un point concernant la dernière évolution de MIDP assez récente. Il n'y a, pour l'instant, pas de nouvelle évolution prévue mais d'autres facteurs peuvent demander une actualisation tels que l'évolution des usages et des pratiques des apprenants.

4.8 La structure

Le tutoriel se veut complet mais non exhaustif. Il est complet en ce qui concerne le développement d'une interface graphique en général. Il offre des informations théoriques et pratiques sur un langage de programmation ainsi que des conseils sur le design et cela afin de permettre à l'apprenant de développer au mieux des interfaces simples et utilisables. Le tutoriel n'est pas exhaustif dans le sens où toutes les classes et tous les éléments de

développement graphique ne sont pas présentés. Il était peu pertinent de montrer des classes qui ont des fonctionnements très semblables. Cependant, les modules couvrent tous les éléments utiles et pertinents au niveau de leur utilisation. Les fonctionnalités secondaires viendront avec l'usage.

L'introduction expose l'objectif global de la formation et la structure du document afin d'en informer l'utilisateur et de faciliter sa navigation. Elle fait un résumé du contenu des différentes parties. Elle expose également la méthode pédagogique utilisée pour l'apprentissage.

Le corps se compose de quatre parties : une explication théorique sur l'architecture du langage, les prédispositions à prendre pour préparer l'environnement de programmation, le développement de petites applications expliquées par étapes, et des guidelines destinés au développement d'IHM sur PDA's.

La table des matières du tutoriel se trouve ci-après. La section suivante présente les chapitres 3 et 4 du tutoriel, $\mu laversioncompltesetrouveenannexe$.

Théorie, généralités sur J2ME Cette première partie tient lieu d'introduction au tutoriel, avant de passer à la partie pratique. Dans un premier temps, elle propose à l'apprenant une vue globale de J2ME et précise, ensuite, le profil utilisé, MIDP. Elle permet à l'apprenant d'avoir une vue d'ensemble de la plate-forme et de ce qu'elle offre. Cela lui permettra de choisir au mieux la configuration et le profil qu'il veut utiliser et de changer son choix si celui-ci est inadéquat. Le profil MIDP et plus particulièrement les API's de développement d'interfaces graphiques. Cette partie est utile à l'apprenant pour qu'il comprenne l'environnement et l'outil avec lesquels il va travailler. Il n'est pas absolument nécessaire de lire cette partie pour bien utiliser le tutoriel et l'utilisateur peut toujours revenir dessus plus tard. Les éléments de l'API ne sont pas exposés dans cette partie mais le seront un à un dans la partie "Développement".

Prédispositions pratiques Les pré-requis sont les besoins nécessaires avant de commencer réellement à développer. Ils commencent par énoncer les outils nécessaires et donnent les adresses où les télécharger. Le second point, l'exécution sur l'émulateur Palm, explique les téléchargements nécessaires ainsi que la configuration des chemins d'accès et la manière de créer un exécutable. Le dernier point présente un détail des étapes à accomplir pour créer une application MIDP.

Développement Le premier module est divisé en plusieurs petites parties expliquant chacune un élément simple. Cela permet de familiariser l'apprenant à l'environnement et à l'utilisation des bases de ce langage. La première sous-partie concerne le "Display" et le "Displayable" qui sont les bases de la programmation avec MIDP. Un canevas de code est donc directement fourni. Chaque fonction est expliquée, ainsi que la structure du code qui s'y rapporte. L'apprenant a donc le code indispensable à toute implémentation d'une interface utilisateur avec MIDP. Les sous-parties suivantes concernent chacune un élément simple. Le premier élément expliqué est le plus simple de tous, c'est la "TextBox". C'est un displayable, élément indispensable à l'affichage sur l'écran de n'importe quel autre élément. Les sous-parties suivantes s'intéressent à différents éléments qui permettent d'enrichir l'interface et qui ont chacune une utilité. Le module est terminé par le code d'une petite interface qui comprend tous les éléments vus avant. Le deuxième module s'intéresse aux commandes, à la manière d'interagir avec l'interface. Il se structure de la même manière que les sous-parties du module précédent avec d'abord, les définitions et les objectifs, viennent ensuite l'explication des étapes accompagnée d'un peu de théorie. Les commandes sont une étape plus complexe. En dernier, vient le code d'une interface comprenant de nouveau un exemple de tous les éléments vus avant. Le troisième et dernier module concerne la classe la plus importante, car la plus utile de MIDP, les "form" et "items". Le module suit la même structure qu'expliquée plus haut. Un exemple de code reprenant tous les concepts vus dans les différents modules clôture ce chapitre.

Commencer par la définition et l'objectif du module est important pour une vision claire de la matière et pour la structuration des connaissances chez l'apprenant.

Le tutoriel se concentre principalement sur MIDP 1.0 plus courant que son évolution MIDP 2.0. Celle-ci est cependant expliquée dans ses principaux changement et apports. Elle permet, entre autres, une plus grande créativité et ouvre de nouvelles possibilités.

Chaque module est assez court et la structure en sous-modules permet à l'utilisateur de faire une pause dans la formation. C'est essentiel en sachant que l'on admet généralement qu'un individu normal a une période de concentration moyenne d'environ une demi-heure.

Les guidelines Comme vu dans le chapitre précédent, plusieurs problèmes sont exacerbés par la petitesse des écrans. La partie concernant les guidelines semble donc inévitable pour énoncer des principes élémentaires d'utilisabilité. Elle fait référence à de nombreux sites, principalement les guidelines établis par les concepteurs des principaux systèmes d'exploitation : PalmOS, PocketPC et Symbian.

Prédispositions pratiques

Téléchargements

Avant de commencer, il faut vous procurer les outils suivants :

Un EDI : L'EDI utilisé dans ce tutorial est netbeans4 pour les facilités qu'il offre grâce à son mobile pack pour la programmation avec J2ME. Il en existe cependant beaucoup d'autres. Netbeans est disponible à l'adresse suivante :

http ://www.netbeans.org/

Sur le site de sun, il est nécessaire de télécharger :
- un JDK supérieur à 1.3 :
  http ://java.sun.com/j2se/index.jsp
- le profil MIDP :
  http ://java.sun.com/products/midp/
- la configuration CLDC
  http ://www.sun.com/software/communitysource/j2me/cldc/
- J2ME wireless toolkit :
  http ://java.sun.com/products/j2mewtoolkit/index.html

Téléchargez l'émulateur palm OS et les ROM's à l'adresse suivante :

http ://www.palmos.com/dev/tools/emulator/

Pour l'installation et la configuration des paths, il existe un très bon article en français :

http ://www.clubj2me.org/article.php ?sid=1

Exécution sur l'émulateur Palm

Il faut télécharger MIDP pour PalmOS :

http ://java.sun.com/products/midp4palm/download.html

Vous devez décompresser dossier zip, dans le répertoire PRCfiles, il y a un fichier MIPD.prc qu'il faut installer sur l'émulateur Palm. Il suffit de glisser l'icône sur l'écran de l'émulateur.

Pour pouvoir exécuter une MIDlet sur le Palm, il vous faut convertir le fichier .jad ou

.jar de votre MIDlet en un fichier .prc qui est directement exécutable sur Palm OS. Dans le répertoire Converter du dossier midp4palm1.0, vous devez exécuter le fichier converter.bat.

Si à l'exécution, le converter déclare l'erreur suivante :

```
Error: Java path is missing in your environment

       Please set JAVA\_PATH to point to your Java directory

       e.g. set JAVA\_PATH=c:\bin\jdk1.3\
```

Il faut configurer la variable d'environnement JAVA_PATH. Pour ce faire, dans Windows, il faut aller dans le panneau de configuration dans System. Dans les propriétés du système, cliquez sur l'onglet "Avancé" et aller dans Variables d'environnement. Ajoutez une nouvelle variable d'environnement JAVA_PATH avec comme valeur un chemin vers un jdk supérieur au jdk 1.3.

Créer une application MIDP

Créer un nouveau projet J2ME MIDP – Dans File, choisir New Project (Ctrl-Shift-N).
En dessous de Categories, sélectionner Mobile. Sous Projects, sélectionner Mobile Application et cliquer sur Next.

- Sous Project Name, entrer MyHello. Changer le Project Home pour une autre directory de votre système. A partir de maintenant, nous ferons référence au dossier par $PROJECTHOME.
- Vérifier la check box Create HelloMIDlet. Cliquer sur Next.
- Laisser le J2ME Wireless Toolkit avec la Target Platform sélectionnée.
- Cliquer sur Finish. L'IDE crée le $PROJECTHOME./MyHello project folder. Le dossier du projet contient toutes vos sources et meta-données du projet. Le projet MyHello s'ouvre dans la fenêtre des projets.

Éditer le code source Java – "Expand" le noeud du projet MyHello et double-cliquer sur le noeud du code source HelloMIDlet.java. Le code source s'affiche dans le Source Editor.

- Dans la méthode startApp(), remplacer "test string" avec le texte de votre choix, par exemple, "Hello World."

Compiler et exécuter un projet hoisir Run Main Project (F6) du menu Run. Double-cliquer sur la fenêtre d'Output pour la maximiser, vous pourrez voir ainsi l'entièreté de l'output. Noter que le fichier HelloMIDlet.java est construit et pré-vérifié avant d'être exécuté. Un émulateur s'ouvre pour afficher les résultats de l'exécution de la MIDlet. L'émulateur par défaut est le DefaultColorPhone.

Dans la fenêtre de l'émulateur, cliquer sur le bouton en dessous de la commande Launch. L'émulateur lance la MIDlet et affiche le texte entré précédemment.

Cliquer sur le bouton en dessous d'Exit pour fermer la MIDlet. Ensuite cliquer sur le bouton dans le coin supérieur droit de l'émulateur pour fermer la fenêtre de l'émulateur.

Pour de plus amples informations, vous pouvez aller sur le site de netbeans : http ://www.netbeans.org/k mobility-40.html

Développement d'IHM avec l'API de haut-niveau

Module 1 : Les composants simples

Introduction

Ce module a pour objectif de familiariser le développeur à l'environnement de MIDP. Il introduit les composants suivants :

- *Display et Displayable* : ce sont les composants de base dans l'API de haut-niveau. Ils permettent d'encapsuler les différents composants qui seront visibles à l'écran.
- *Ticker*
- *TextBox*
- *Alert*

Les trois derniers composants sont les composants les plus simples de l'API.

Display et Displayable

Définition

Le Display représente l'écran de l'appareil au niveau hardware. Il encapsule les éléments qui seront visibles à l'écran. Le Displayable contient les objets qui sont visibles à l'écran. Pour ajouter un élément au Display, il faut qu'il soit d'abord inclus dans un objet héritant de Displayable, un screen (TextBox, Alert, Form, List) ou un canvas (API de bas niveau).

Code

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class HelloMid extends MIDlet {
    private Display display;
```

```
        public void startApp() {
        display = Display.getDisplay(this);
        }

        public void pauseApp() {
        }

        public void destroyApp(boolean unconditional) {
        }
    }
```

Explications - Étapes

Les opérations réalisées quand la MIDlet est en background ne prennent effet que quand la MIDlet passe en foreground.

1. Déclaration de la variable :

   `private Display display;`

2. Appel de la méthode `getDisplay()` dans `startApp()` :

   `display = Display.getDisplay(this);`

   La méthode statique `getDisplay(MIDlet mid)` permet d'obtenir une référence vers la classe Display. Chaque MIDlet n'a accès qu'à une seule instance de cette classe qui lui est propre. Aussi, `getDisplay` renvoit toujours la même valeur à chaque appel.

   Une MIDlet invoque la méthode `getDisplay()` au premier appel de `startApp()`. Si la méthode `getDisplay()` doit effectivement se trouver dans startApp(), elle peut s'y trouver à n'importe quel moment.

La méthode `setCurrent()` permet d'associer un Displayable à un Display. Il ne devient visible qu'à ce moment là. La méthode `setCurrent` influence uniquement l'état interne de l'affichage Display et notifie au gestionnaire d'applications que la MIDlet voudrait voir le Displayable donné affiché à l'écran.

La méthode `getCurrent` permet de savoir ce qui est affiché à l'écran à un moment donné.

TextBox

Definition

Une TextBox est un Screen qui permet d'afficher et d'éditer du texte.

Explications - Étapes

1. Déclaration des variables

```
private TextBox text;
```

2. Initialisation des variables

```
text = new TextBox("Write all you want",
            "You cannot write more character than 120",
            120, TextField.ANY);
```

   Pour créer une TextBox, il faut spécifier les paramètres désirés dans le constructeur de la TextBox

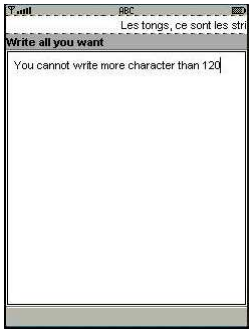
```
public TextBox(String title, String text, int maxSize,
        int constraints);
```

   - *title* est utilisé comme titre de l'écran, de la TextBox.
   - *text* détermine ce qui sera affiché à l'écran au début.
   - *maxSize* détermine le nombre de caractères maximal qu'une TextBox peut contenir.
   - *constraints* détermine le type de la TextBox. Elle peut être de type ANY, EMAILADDR, NUMERIC, PASSWORD, PHONENUMBER ou URL.

3. Ajout de la TextBox au Display

```
display.setCurrent(text);
```

Screenshots



Ticker

Definition

Le ticker implémente un texte qui défile continuellement à l'écran. Un ticker peut être attaché à un des quatres types de "screen" : "TextBox", "List", "Alert" ou "Form".

Explications - Étapes

1. Déclaration des variables

```
private String citation;
private Ticker tick;
```

2. Initialisation des variables

```
citation =
    "Les tongs, ce sont les strings des pieds.   Ph. Geluck    ";
tick = new Ticker(citation);
```

Remarquez les espaces ajoutés en fin de citation. Ils permettent une meilleure visibilité puisque une fois le texte terminé, il recommence depuis le début sans temps d'arrêt.

On crée un nouveau ticker grâce au constructeur suivant :

```
new Ticker(String str);
```

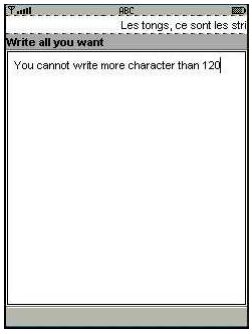
str : Le seul argument du constructeur est le texte qui défilera à l'écran. Aucun paramètres n'est fourni pour déterminer la direction ou la vitesse de défilement du texte. Ceux-ci sont déterminés par l'implémentation MIDP.

3. Ajout du Ticker à la TextBox

```
text.setTicker(tick);
```

Une fois initialisé, le ticker défile à l'écran, pour l'arrêter, il faut le supprimer au moyen de la méthode setTicker(null).

Screenshots



Alert

Definition

La classe Alert est une classe qui affiche à l'écran une boîte de dialogue. Il peut consister en un label, un texte ou une image. Il est possible de, soit définir le temps pendant lequel la boîte de dialogue est affichée avant de passer à un autre Screen, soit permettre à l'utilisateur de fermer lui-même la boîte.

Explications - Étapes

1. Déclaration des variables

```
private String title, content;
private AlertType alertTp;
```

2. Initialisation des variables

```
//Initialisation des variables nécessaires à alert
title = new String("Confirmation");
content = new String
    ("After this confirmation you have a TextBox");
alertTp = AlertType.CONFIRMATION;

Alert alert = new Alert (title,content,null,alertTp);
```

   Le constructeur utilisé ici est

```
public Alert(String title, String alertText, Image
alertImage, AlertType alertType);
```

   (a) *title* : String qui sera affiché comme titre de l'alerte.
   (b) *alertText* : Texte du corps de l'alerte.
   (c) *alertImage* : Image affichée à l'écran. Le seul format d'image accepté est le format png.
   (d) *alertType* : Les types d'alerte sont ALARM, CONFIRM, ERROR, INFO, WARNING.

3. Détermination du temps d'affichage de l'alerte

```
alert.setTimeout(Alert.FOREVER);
```

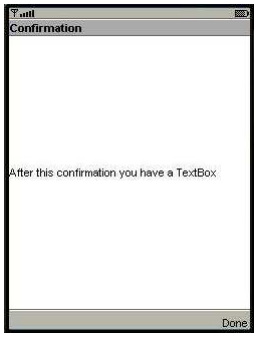
   Si on n'a pas une vision complète du message d'alerte et qu'il faut avoir recours au scroll, la limite de temps est automatiquement désactivée.

4. Ajout de l'alerte au display

```
display.setCurrent(alert, text);
```

La méthode `setCurrent` possède ici deux paramètres, elle permet d'afficher l'alerte et de définir l'écran qui sera affiché à sa fermeture, dans ce cas-ci, c'est la TextBox text définie plus haut.

Screenshots



Code

```
// Il faut d'abord importer les packages
//contenant les classes midlet et lcdui

import javax.microedition.midlet.*; import
javax.microedition.lcdui.*;

// Chaque application MID est dérivé de la classe MIDlet,
// il faut donc étendre la classe MIDlet

public class HelloMid extends MIDlet
    //implements CommandListener
{
    //déclaration des variables de base
    private Display display;
    private TextBox text;

    //déclaration des variables du ticker
    private String citation;
    private Ticker tick;

    //déclaration des variables pour l'alerte
```

```
    private String title, content;
    private AlertType alertTp;

    public void startApp() {
        display = Display.getDisplay(this);
        text = new TextBox("Write all you want",
                "You cannot write more character than 120",
                    120, TextField.ANY);

        //Initialisation et Ajout du ticker à la TextBox
        citation =
            "Les tongs, ce sont les strings des pieds.
                    Ph. Geluck    ";
        tick = new Ticker(citation);
        text.setTicker(tick);

        //Initialisation des variables nécessaires à alert
        title = new String("Confirmation");
        content = new String
                ("After this confirmation you have a TextBox");
        alertTp = AlertType.CONFIRMATION;

        //Initialisation de l'alerte
        Alert alert = new Alert (title,content,null,alertTp);
        alert.setTimeout(Alert.FOREVER);

        //Ajout de l'alerte au display, une fois fermée,
        //la TextBox sera affichée.
        display.setCurrent(alert, text);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

Module 2 : Les commandes

Définition

Pour créer des menus et ou des boutons, on utilise la classe "Commands". Elle permet d'implémenter des commandes qui, en fonction de l'environnement, seront affichées comme boutons et/ou dans un scroll menu.

Objectif

Le premier module de ce tutoriel a introduit l'environnement grâce à l'apprentissage des composants les plus simples. Vous êtes capable d'afficher des éléments à l'écran mais pas d'agir sur ces éléments, de voyager entre les screens ou de quitter l'application. Ce module va vous permettre d'avancer en ce sens.

Étapes

1. Déclaration et initialisation des variables

```
Command clear = new Command("Clear", Command.SCREEN, 0);
Command exit = new Command("Exit", Command.EXIT, 0);
```

Pour créer une commande, on utilise le constructeur et on spécifie les arguments :

```
public Command(String label, int type, int prioriy);
```

Une fois la commande créée, il est impossible de changer les arguments (label,type,priorité). Les arguments type et priorité sont utilisés lors du positionnement de la commande à l'écran.

- *label* : Le label de la commande consiste en un String qui sera affiché à l'écran pour dénommer la commande.
- *type* : Les différents types de commandes sont
  - Command.BACK permet à l'utilisateur de retourner à l'écran précédent.
  - Command.CANCEL quand on a besoin de notifier à l'écran une réponse négative.
  - Command.EXIT utilisée pour spécifier une commande de sortie de l'application
  - Command.HELP passé quand l'application requiert un écran d'aide.
  - Command.ITEM permet de dire à l'application qu'un item "explicite" a été ajouté au screen.
  - Command.OK permet de notifier au screen une réponse positive.
  - Command.SCREEN type qui spécifie une screen-specific Command de l'application. Pas de signification "generic".
  - Command.STOP interrompt une procédure en cours.

- *priority* : En fonction du niveau de priorité, les commandes seront affichées de manière plus ou moins accessible à l'utilisateur. Le niveau de priorité le plus élevé correspond à la plus petit valeur, c'est-à-dire 0.

2. Ajout des commandes au screen

```
text.addCommand(clear);
text.addCommand(exit);
```

L'ajout de la commande à n'importe quelle sous-classe de Displayable se fait facilement grâce à la méthode `addCommand(Command Cmd)`.

Une même commande peut être ajoutée à plusieurs screen. Il n'est donc pas nécessaire d'en créer plusieurs instances.

L'ordre dans lequel les commandes sont ajoutées au screen n'a aucun impact sur leur positionnement. Il n'y a aucun layout ou autre moyen de positionnement fournis par MIDP. Tout est géré par ce dernier en fonction des caractéristiques matérielles de l'appareil et est déterminé uniquement par les paramètres de type et de priorité. La commande qui aura la plus petite valeur de priorité sera la plus accessible.

Le type de la commande détermine le positionnement de la commande dans un menu. Il permet aussi de garder une certaine consistance avec les autres applications. Si le bouton EXIT est toujours en bas à gauche et que cela a été codé dans les propriétés de l'appareil, le bouton EXIT de la nouvelle application sera placé à cet endroit.

Chaque appareil a un nombre de "soft buttons" déterminé. Ce type de boutons n'a pas de fonctionnalité définie. Une fonctionnalité peut leur être assignée dynamiquement grâce aux commandes. Si le nombre de commandes dépasse le nombre de "soft buttons" disponibles, les commandes affichées à l'écran sous forme de boutons seront celles qui auront la priorité la plus importante, les autres auront leur place dans un menu.

(MIDP 2.0 Les commandes peuvent désormais être ajoutées à un item et plus seulement à un screen. Cela se fait de façon très simple. D'abord, il faut créer l'item puis y ajouter la commande et enfin enregistrer le command listener.

3. Gestion des événements

Il existe deux types d'événements : l'événement `Screen` et l'événement `ItemStateChanged`. Les listeners correspondants sont respectivement CommandListener et ItemStateListener.

Le traitement associé à une action effectuée sur une commande est effectué dans une interface CommandListener. Cette interface définit une méthode, commandAction, qui est appelée si une commande est déclenchée. Le listener correspondant est mis en place en implémentant l'interface CommandListener. Vous devez alors enregistrer cette dernière avec la méthode `setCommandListener (CommandListener myListener)`.

Toute modification interactive de l'état d'un élément de formulaire déclenche un événement itemStateChanged (exemple : modification d'un texte, sélection d'un élément

d'une liste, ...). Le listener correspondant est mis en place en implémentant l'interface ItemStateListener. Vous devez alors enregistrer l'objet ItemStateListener auprès d'un formulaire Form avec la méthode setItemStateListener (ItemStateListener myListener).
La classe `CommandsMidlet` doit implémenter l'interface `CommandListener` avec comme unique méthode `commandAction` implémentée plus bas.

```
public class CommandsMidlet extends HelloMid
       implements CommandListener
```

Pour être notifié quand un utilisateur active une Command, vous devez enregistrer un CommandListener avec le Displayable auquel la commande a été ajoutée. Vous pouvez faire cela en invoquant la méthode suivante :

```
public void
       setCommandListener(CommandListener cl);
```

Contrairement à Swing et à AWT, MIDP ne permet l'enregistrement que d'un seul CommandListener à un moment donné. Lorsque la méthode setCommandListener(CommandListener cl) est appelée, tout listener précédent est remplacé par le nouveau. Si la méthode est appelée avec l'argument `null`, le listener précédent est supprimé. Cela semble un peu limité, mais permet une meilleure gestion des ressources en évitant la prolifération de classes d'événements qui sont très chères en terme de mémoire. Cependant, même si cela semble quelque peu limité, les MIDlets peuvent faire énormément de choses avec seulement un listener par screen.
`CommandListener` est une interface dotée d'une seule méthode :

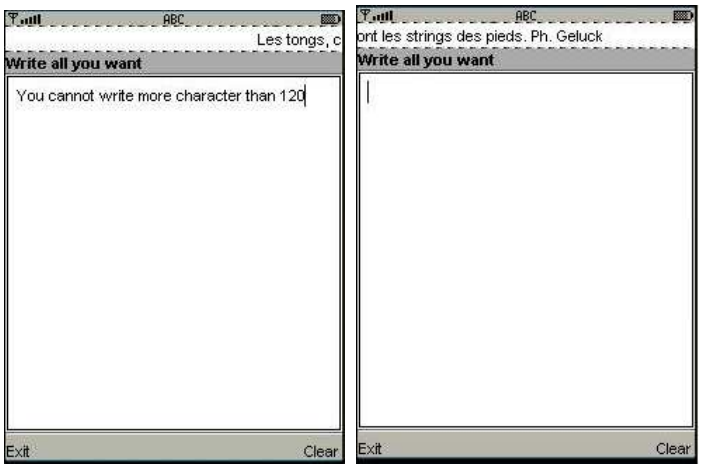
```
       public void commandAction (Command c, Displayable d) {
           if(c==exit)
           notifyDestroyed();
           else if (c==clear)
               clear();
       }
```

La méthode commandAction() est appelée quand n'importe quelle `Command` du Displayable est activée. C'est à l'intérieur de cette méthode que l'on notifie au gestionnaire quelles sont les actions à effectuer lors de l'activation d'une commande. La méthode `notifyDestroyed` notifie au gestionnaire que la MIDlet se termine volontairement.

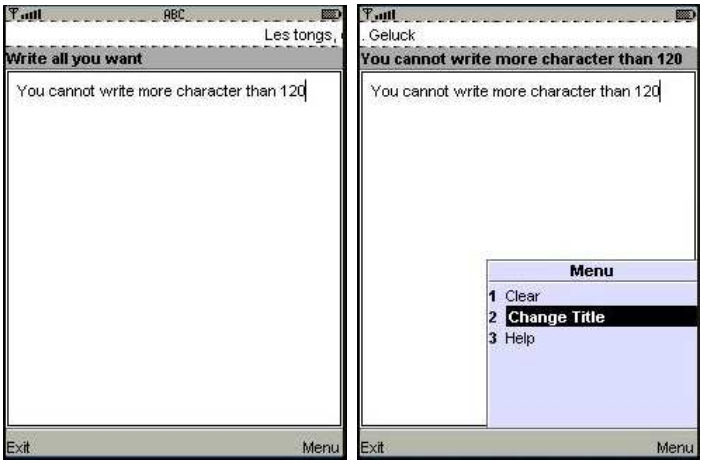
- *Command c* : Cet argument est le plus utile, il permet de savoir quelle est la commande que l'utilisateur a activée. A ce moment, on appelle la méthode correspondant à l'action que l'utilisateur veut réaliser.
- *Displayable d* : Cet argument est utile lorsqu'une même commande est assignée à plusieurs screens et que l'action à réaliser dépend du screen courant ou lorsque l'action nécessite une référence au screen pour la réalisation de sa tâche.

4. Positionnement des commandes en fonction des places disponibles

   Si l'application propose deux commandes. Les deux commandes seront affichées à l'écran.



   Une nouvelle application propose quatre commandes. Ce nombre dépasse le nombre de "soft buttons" disponible sur la plupart des appareils. Les captures d'écran montrent comment les commandes sont agencées en fonction de l'appareil et du type de la commande.



5. code avec deux commandes

   Le code ci-dessous étend la précédente classe `HelloMid`. Cela évite la reprogrammation de tout le code et permet de voir uniquement les lignes qui implémentent les commandes, objet de ce module.

```
/*
* CommandsMidlet.java
*
* Created on 8 novembre 2004, 11:56
*/
```

```
import javax.microedition.midlet.*; import
javax.microedition.lcdui.*;

/**
*
* @author  Sophie Van Tongelen
* @version
*/
public class CommandsMidlet extends HelloMid implements
CommandListener{
    //déclaration des diverses commandes

    static final Command clear = new Command("Clear", Command.SCREEN, 0);
    static final Command exit = new Command("Exit", Command.EXIT, 0);
    Display display;

    //implémentation des commandes
    public void commandAction (Command c, Displayable d) {
        if(c==exit)
        notifyDestroyed();
        else if (c==clear)
            clear();
    }

    //implémentation de l'action correspondante au bouton clear.
    public void clear(){
        text.setString("");
    }

    public void startApp() {
        boolean first = !started;
        super.startApp();

        //si c la 1ere exec de startapp, les commandes sont installées
        if(first){
        text.addCommand(clear);
        text.addCommand(exit);
        text.setCommandListener(this);
        }
    }

    public void pauseApp() {
    }
```

```
    public void destroyApp(boolean unconditional) {
    }
}
```

6. code avec quatre commandes

```
/*
* CommandsMidlet.java
*
* Created on 8 novembre 2004, 11:56
*/

import javax.microedition.midlet.*; import
javax.microedition.lcdui.*;

/**
*
* @author  Domotech
* @version
*/
public class Commands2Midlet extends HelloMid implements
CommandListener{
    static final Command clear = new Command("Clear", Command.SCREEN, 1);
    static final Command exit = new Command("Exit", Command.EXIT, 0);
    static final Command help = new Command("Help", Command.HELP, 2);
    static final Command chgTitle = new Command("Change Title", Command.SCREEN,

    Display display;

    //implem des commandes
    public void commandAction (Command c, Displayable d) {
        if(c==exit){
            destroyApp(true);
            notifyDestroyed();
        }
        else if (c==clear)
            clear();
        else if(c==help)
            help();
        else if(c==chgTitle)
            chgTitle();
```

```
        }

        public void help(){
        }

        public void chgTitle(){
            String newTitle = text.getString();
            text.setTitle(newTitle);
        }

        public void clear(){
            text.setString("");
        }

        public void startApp() {
            boolean first = !started;
            super.startApp();

            //si c la 1ere exec de startapp, les commandes sont installées
            if(first){
            text.addCommand(clear);
            text.addCommand(exit);
            text.addCommand(help);
            text.addCommand(chgTitle);
            text.setCommandListener(this);
            }
        }

        public void pauseApp() {
        }

        public void destroyApp(boolean unconditional) {
        }
    }
```

Module 3 : Form et Items

Objectif

Ce module a pour but d'introduire la classe la plus importante de l'API de haut-niveau, la classe `Form`.

Définitions des classes

La classe `Form` est la sous-classe la plus importante de MIDP. Elle peut contenir autant d'objets `Item` que désirés.

La classe `Item` est une classe abstraite qui ne peut être instanciée. De nouveau, on ne peut agir sur la taille et le positionnement des items. Tout ça est géré par MIDP. Les sous-classes de `Item` sont : `ChoiceGroup, DateField, Gauge, ImageItem, StringItem, TextField`. Chacune sera développée en son temps plus bas.

La classe `List` fournit diverses listes dont l'utilisateur peut sélectionner un ou plusieurs items. Cette classe ne sera pas présentée ici car son utilisation, excepté le fait que ce soit un `Screen` à part entière, est très proche de l'item ChoiceGroup.

Form

1. Création d'un *Form*. La classe Form s'utilise comme tout autre classe `Screen`. Pour en créer une instance, on utilise un des deux constructeurs suivants dont on spécifie les paramètres :

```
public Form(String title);
public Form(String title, Item[] items);
```

   – *title* constitue le titre du screen `Form`.
   – *items* fournit les différents items que l'on veut ajouter au screen sous forme de tableau d'items.

   Le premier constructeur permet une meilleure structuration du code aussi ce sera celui utilisé ici.

```
private Form form;

//Initialisation
form = new Form("Items");
```

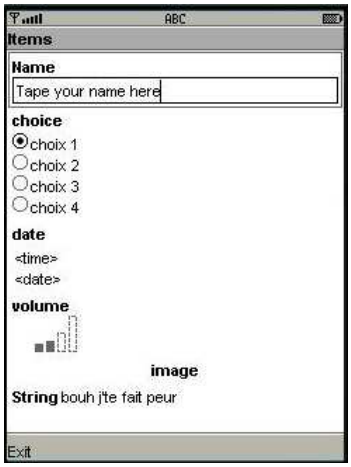
2. Ajout d'un item au screen form. Pour ajouter un item au screen form, la classe form fournit la méthode suivante :

```
public void append(Item it);
```

Une même instance de `Command` ou de `Ticker` peut être ajoutée à plusieurs screens différents. Ce n'est pas le cas pour les items. Une instance d'item ne peut être ajoutée qu'à un screen `Form` à un moment donné.

3. Ajout du screen *form* au display par la méthode `setCurrent()`.

```
d.setCurrent(form);
```



Création d'items

ChoiceGroup

1. Définition
   L'item ChoiceGroup permet de fournir une liste d'éléments que l'utilisateur peut sélectionner. Il y a deux types de ChoiceGroup : EXCLUSIVE, MULTIPLE. La classe liste en fournit un troisième qui est le type IMPLICIT.
2. Explications - Étapes
   (a) Initialisation des variables nécessaires à choiceIt

```
labChoice = "choice";
choicetype = ChoiceGroup.EXCLUSIVE;
imgChc = null;
String[] elements =
    {"choix 1","choix 2","choix 3","choix 4"};
choiceIt =
    new ChoiceGroup(labChoice,choicetype,elements,imgChc);
```

Pour créer un item ChoiceGroup, il faut passer quatre paramètres au constructeur :

```
public ChoiceGroup(String label, int choiceType,
String[] stringElements, Image[] imageElements);
```

- *label*
- *choiceType* :
  - EXCLUSIVE : ce type de ChoiceGroup fournit une liste d'éléments dont un seul peut être sélectionné à la fois.
  - MULTIPLE : dans ce type de ChoiceGroup, l'utilisateur peut sélectionner plusieurs éléments à la fois.

  Pour la liste, le type IMPLICIT permet d'envoyer une commande pour signaler le changement d'état suite à la sélection.
- *stringElements* : ce paramètre est un tableau de string où chaque string désigne un élément de la liste et en est son label. Cependant, les éléments peuvent également être ajoutés grâce à la méthode `append(String str, Image img);`. Le string fait alors office de label et on peut y associer une image. On utilise alors le constructeur suivant :

  ```
  public ChoiceGroup(String label, int choiceType);
  ```

- *imageElements* : ce tableau d'images permet d'associer une image à chaque élément.

(b) Ajout du ChoiceGroup au screen form

```
form.append(choiceIt);
```

DateField

1. Définition
   L'item DateField permet d'afficher et de modifier la date et l'heure d'un objet de type `Date`.
2. Explications - Étapes
   (a) Initialisation des variables nécessaires à *dateIt*

   ```
   labDF = "date";
   mode = DateField.DATE_TIME;
   dateIt = new DateField(labDF,mode);
   ```

   Pour créer un item `DateField`, il faut passer deux paramètres au constructeur :

   ```
   public DateField(String label,int mode,TimeZone timeZone);
   ```

   - *label* : c'est le label du DateField
   - *mode* : il y a trois mode :
     - DATE : Affiche uniquement la date.

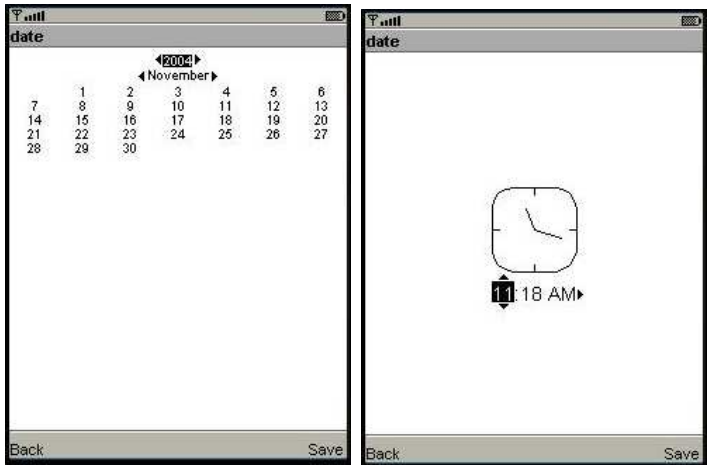
- TIME : Affiche uniquement l'heure.
- DATE_TIME : Affiche la date et l'heure.

– *timeZone* : permet de définir un fuseau horaire.

(b) Ajout du `DateField` au screen form

```
form.append(dateIt);
```

3. Screenshots



Gauge

1. Définition
   L'item `Gauge` permet la visualisation de l'avancement et/ou d'un état à un moment donné. Elle se présente sous la forme de plusieurs barres verticales de même hauteur pour une jauge non interactive, d'hauteur croissante pour une jauge interactive.
2. Explications - Étapes

   (a) Initialisation des variables nécessaires à *gaugeIt*

```
labGauge = "volume";
interactive = true;
maxValue = 8;
initialVal = 4;
gaugeIt =
    new Gauge(labGauge,interactive,maxValue,initialVal);
```

   Pour créer un item `Gauge`, il faut passer quatre paramètres au constructeur :

```
public Gauge(String label,boolean interactive,int
                maxValue,int initialValue);
```

   – *label* : c'est le label de la jauge.

- *interactive* : une jauge interactive sera par exemple une représentation du volume que l'utilisateur pourra à son gré augmenter ou diminuer. Une jauge non interactive sera par exemple la visualisation du temps restant de chargement.
- *maxValue* : c'est la valeur maximal de la jauge.
- *initialValue* : c'est la valeur de la jauge affichée à l'écran à l'état initial

(b) Ajout de l'item `Gauge` au screen form

```
form.append(gaugeIt);
```

ImageItem

1. Définition
   Cet item permet d'afficher une image non interactive. L'image ne peut être que du format png.
2. Explications - Étapes

   (a) Initialisation des variables nécessaires à *imgIt*

```
labImg = "image";
Image img = null;
layout = ImageItem.LAYOUT_CENTER;
altTxt = "altText???"; ACHTUNG
imgIt= new ImageItem(labImg,img,layout,altTxt);
```

   Pour créer un item ImageItem, il faut passer quatre paramètres au constructeur :

```
public ImageItem(String label,Image img,int
                                 layout,String altText);
```

   - *label* : c'est le label de l'image.
   - *img* : image qui sera affichée à l'écran.
   - *layout* : Il y a 6 types de layout :
     - LAYOUT_CENTER : centrée horizontalement.
     - LAYOUT_DEFAULT : dépend de l'appareil
     - LAYOUT_LEFT : alignée à gauche
     - LAYOUT_NEWLINE_AFTER : une ligne est dessinée sous l'image.
     - LAYOUT_ : une ligne est dessinée au-dessus de l'image.
     - LAYOUT_RIGHT : alignée à droite
   - *altText* : ACHTUNG

   (b) Ajout de l'ImageItem au screen form

```
form.append(imgIt);
```

StringItem

1. Définition
   L'item StringItem permet d'afficher une chaîne de caractère à l'écran qui soit non interactive.
2. Explications - Étapes
   (a) Initialisation des variables nécessaires à strIt

```
labStr = "String";
txt = "bouh j'te fait peur";
strIt= new StringItem(labStr,txt);
```

   Pour créer un item StringItem, il faut passer deux paramètres au constructeur :

```
public StringItem(String label,String text);
```

   – *label* : c'est le label de la chaîne.
   – *text* : constitue le corps du texte.

   (b) Ajout du StringItem au screen form

```
form.append(strIt);
```

TextField

1. Définition
   L'item `TextField` permet d'afficher et d'éditer du texte. C'est le pendant de `TextBox`, dans la classe `Item`
2. Explications - Étapes
   Pour créer un `TextField`, on utilise le constructeur suivant :

```
public TextField(String label,String text,int
                                    maxSize,int constraints);
```

   Les paramètres sont les mêmes que pour la `TextBox` introduite dans le premier module. Les contraintes sont également les mêmes.

Code complet

```
/*
* FormMidlet.java
*
* Created on 4 novembre 2004, 12:46
```

```
*/

import javax.microedition.midlet.*; import
javax.microedition.lcdui.*;

/**
*
* @author  Domotech
* @version
*/
public class FormMidlet extends MIDlet
    implements CommandListener{

    private Form form;
    private Display d;
    static final Command exit =
            new Command("Exit", Command.EXIT, 0);

    // Déclaration des variables pour le ChoiceGroup
    private String labChoice;
    private int choicetype;
    private Image[] imgChc;
    private ChoiceGroup choiceIt;

    // Déclaration des variables pour le DateField
    private String labDF;
    private int mode;
    private DateField dateIt;

    // Déclaration des variables pour le Gauge
    private String labGauge;
    private boolean interactive;
    private int maxValue;
    private int initialVal;
    private Gauge gaugeIt;

    // Déclaration des variables pour l'ImageItem
    private String labImg;
    private Image img;
    private int layout;
    private String altTxt;
    private ImageItem imgIt;
```

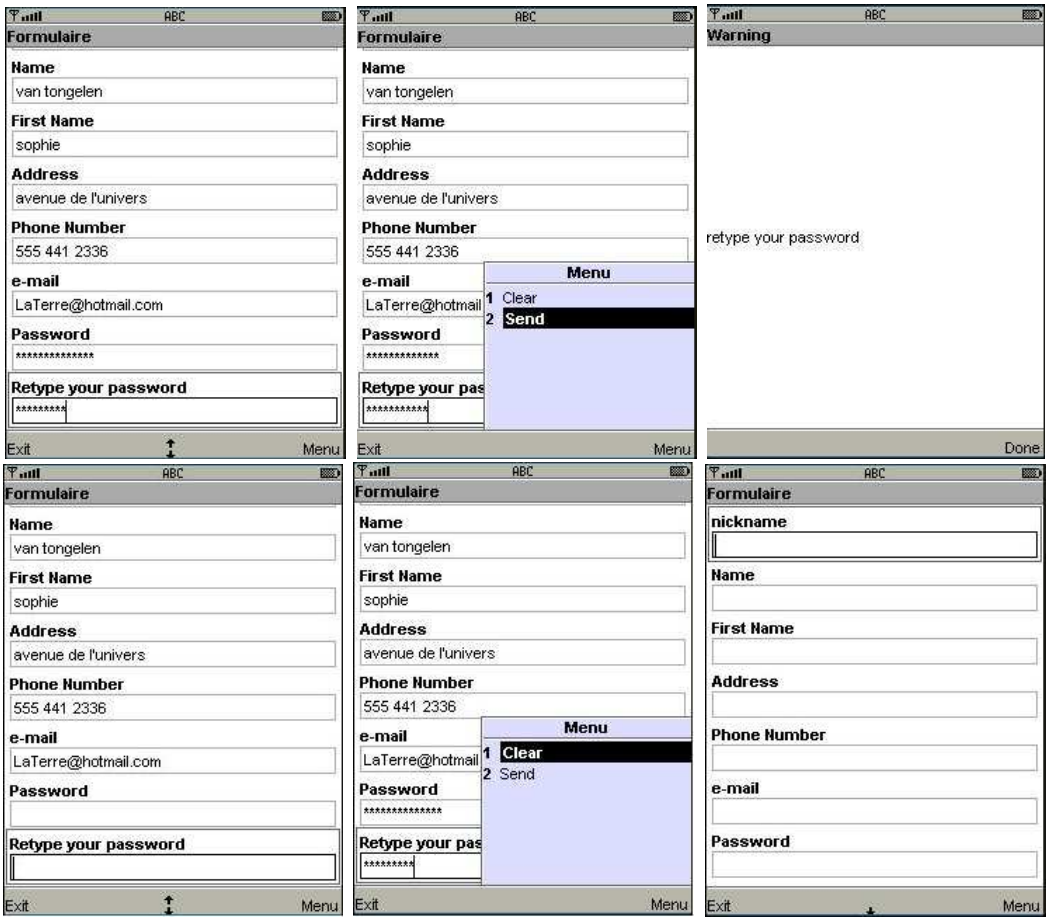
```
// Déclaration des variables pour le StringItem
private String labStr;
private String txt;
private StringItem strIt;

// Déclaration des variables pour le TextField
private String labTF;
private String txtFd;
private int maxSize;
private int constraints;
private TextField txtFdIt;

public void commandAction (Command c, Displayable d) {
    if(c==exit){
        destroyApp(true);
        notifyDestroyed();
    }
}

public void startApp() {
    d =  Display.getDisplay(this);

    //Initialisation de form
    form = new Form("Items");

    //Ajout de la commande exit au screen form
    form.addCommand(exit);
    form.setCommandListener(this);

    //Initialisation des variables nécessaires à choiceIt
    labChoice = "choice";
    choicetype = ChoiceGroup.EXCLUSIVE;
    imgChc = null;
    String[] elements =
        {"choix 1","choix 2","choix 3","choix 4"};
    choiceIt =
        new ChoiceGroup(labChoice,choicetype,elements,imgChc);

    //Initialisation des variables nécessaires à dateIt
    labDF = "date";
    %mode = DateField.DATE_TIME;
    dateIt = new DateField(labDF,mode);
```

```
        //Initialisation des variables nécessaires à gaugeIt
        labGauge = "volume";
        interactive = true;
        maxValue = 8;
        initialVal = 4;
        gaugeIt =
            new Gauge(labGauge,interactive,maxValue,initialVal);

        //Initialisation des variables nécessaires à imgIt
        labImg = "image";
        Image img = null;
        layout = ImageItem.LAYOUT\_CENTER;
        altTxt = "altText???";
        imgIt= new ImageItem(labImg, img,layout,altTxt);

        //Initialisation des variables nécessaires à strIt
        labStr = "String";
        txt = "bouh j'te fait peur";
        strIt= new StringItem(labStr,txt);

        //Initialisation des variables nécessaires à txtFdIt
        labTF = "Name";
        txtFd = "Tape your name here";
        maxSize = 20;
        constraints = TextField.ANY;
        txtFdIt= new TextField(labTF,txtFd,maxSize,constraints);

        //Ajout des différents items au screen form
        form.append(txtFdIt);
        form.append(choiceIt);
        form.append(dateIt);
        form.append(gaugeIt);
        form.append(imgIt);
        form.append(strIt);
        d.setCurrent(form);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

Formulaire

Les captures d'écran ci-dessous montre le fonctionnement du formulaire.

La première figure représente le formulaire rempli. Les champs *Password* et *retype your password* sont remplis différemment de manière à produire une erreur lors de l'envoi du formulaire. Pour envoyer le formulaire, l'utilisateur va dans le menu et sélectionne *send* (figure 2). Lors de l'envoi, l'application détecte la non correspondance des mots de passe et demande à l'utilisateur de retaper un mot de passe via une alerte (figure 3). Lorsque l'utilisateur ferme l'alerte, l'application retourne au formulaire dont les champs concernant le mot de passe sont vides (figure 4). Si l'utilisateur veut remettre les champs du formulaire à zéro, il utilise la commande *clear* dans le menu (figure 5). Les champs du formulaire sont alors vides (figure 6).



Code

```
/*
 * FormulaireMidlet.java
 *
 * Created on 10 novembre 2004, 17:10
 */

import javax.microedition.midlet.*; import
javax.microedition.lcdui.*;

/**
 *
 * @author  Domotech
 * @version
 */
public class FormulaireMidlet extends MIDlet implements
CommandListener{
    Display display;
    Form form;
    TextField nick, name, first, address, phone, mail, pw, retypePw;
    Command exit, clear, send;

    public void clear(){
        nick.setString("");
        name.setString("");
        first.setString("");
        address.setString("");
        phone.setString("");
        mail.setString("");
        pw.setString("");
        retypePw.setString("");
    }

    public void send(){
        String verifpw1 = pw.getString();
        String verifpw2 = retypePw.getString();
        verifyPW(verifpw1,verifpw2);
    }

    public void commandAction(Command c, Displayable s) {
        if(c==exit){
            notifyDestroyed();
```

```
        }
        else if(c==clear)
            clear();
        else if(c==send)
            send();
    }

    public void verifyPW(String pw1, String pw2){
        if (!(pw1.equals(pw2))){
        Alert alert = new Alert ("Warning","retype your password",null,AlertType.WARNING
        alert.setTimeout (Alert.FOREVER);
        display.setCurrent (alert,form);
        pw.setString("");
        retypePw.setString("");
        }
    }

    public void startApp() {
        display = Display.getDisplay (this);

        form = new Form("Formulaire");

        exit = new Command("Exit", Command.EXIT, 0);
        clear = new Command("Clear", Command.SCREEN, 0);
        send = new Command("Send", Command.SCREEN, 0);

        nick = new TextField("nickname","",20,TextField.ANY);
        first = new TextField("First Name","",20,TextField.ANY);
        name = new TextField("Name","",20,0);
        address = new TextField("Address","",20,0);
        phone = new TextField("Phone Number","",20,TextField.PHONENUMBER);
        mail = new TextField("e-mail","",20, TextField.EMAILADDR);
        pw = new TextField("Password","",20,TextField.PASSWORD);
        retypePw = new TextField("Retype your password","",20,TextField.PASSWORD);

        form.append(nick);
        form.append(name);
        form.append(first);
        form.append(address);
        form.append(phone);
        form.append(mail);
        form.append(pw);
        form.append(retypePw);
```

```
        form.addCommand(exit);
        form.addCommand(clear);
        form.addCommand(send);

        form.setCommandListener(this);

        display.setCurrent(form);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

MIDP 2.0

Objectifs

MIDP 2.0 amène plusieurs nouveaux éléments dans le développement des interfaces graphiques. Il permet plus de créativité. Ce module vous introduit les concepts de "CustomItem", qui permet de créer complètement un nouvel item, et offre deux nouveaux items : "Spacer" et "StringItem". MIDP 2.0 introduit également le concept de taille aux classes Item et CustomItem.

Étapes

La taille des items

Pour chaque item, standard ou personnalisé, vous pouvez éditer les tailles minimum et préférée de l'écran. La taille minimum d'un item est la plus petite taille à partir de laquelle un item peut être affiché et fonctionner correctement. La taille préférée fournit la taille optimale par rapport à l'écran. Tandis que les composants développés avec la classe Item peuvent prendre les caractéristiques standards de l'écran, les caractéristiques minimales et préférées doivent être établies manuellement pour les CustomItems.

De nouvelles méthodes ont été ajoutées pour permettre de définir la taille des items :

- setPreferredSize(int width, int height)
- int getPreferredHeight()

- int getPreferredWidth()
- int getMinimumWidth()
- int getMinimumHeight()

Spacer et StringItem

L'item "Spacer" permet d'espacer les éléments graphiques à l'écran. C'est un élément graphiques invisible dont on peut déterminer facilement la taille. L'item "StringItem" permet d'afficher un texte non-éditable par l'utilisateur.

```
import javax.microedition.lcdui.Spacer;

private Spacer spacer;

    // put some space between the items to segregate
    spacer = new Spacer(20, 20);
    form.append(spacer);
```

CustomItem

Les CustomItems peuvent prendre tous les deux des entrées au clavier numérique et au pointeur. Les modes d'interaction sont
KEY_PRESS, KEY_RELEASE, KEY_REPEAT, POINTER_DRAG,
POINTER_PRESS, et POINTER_RELEASE.

Une sous-classe de CustomItem peut rendre aisée la création interactive de composants tels que les tableurs ou les calendriers. La classe permet aux utilisateurs de mettre à jour le contenu directement dans le composant existant ou dans un composant secondaire tel qu'un TextField. Les mises à jour qui ont été entrées dans le composant secondaire sont automatiquement copiées dans le composant original lorsque la session est complète.

Pour construire un nouvel item, il faut définir une nouvelle classe qui étend la classe CustomItem. Celui-ci pourra ensuite être ajouté au "Form" comme n'importe quel autre "Item".

```
public class CustIt extends CustomItem {
    public CustIt( String exemple ){
        super( exemple );
    }
...
```

```
}

        CustIt ci = new CustIt( "label" );
        mi.setLayout( Item.LAYOUT_CENTER |
                      Item.LAYOUT_NEWLINE_BEFORE |
                      Item.LAYOUT_NEWLINE_AFTER );

        f.append( ci );
```

Vous pouvez trouver l'ensemble des modifications dans le document de Nokia que vous trouverez en annexe ou à l'adresse suivante :
http ://ncsp.forum.nokia.com/download/ ?asset$_id = 327; ref = devx$

4.9 Critiques

Les critiques portent. Certaines d'entre elles sont les conséquences directes du type de médias utilisés et de l'approche théorique suivie. En effet, la théorie comportementaliste a montré ses limites, la formation devrait donc évolué les théories cognitiviste ou constructiviste afin de mettre l'individu au centre de son apprentissage.

4.9.1 L'interactivité

Le manque d'interactivité joue sur la construction des connaissances qui se fait par la pratique, grâce à un traitement actif de l'information. Le problème de l'autonomie de l'apprenant et donc de sa motivation, problème principal de l'autoformation, n'est pas vraiment atténué. Le manque d'interactivité et de liens sociaux avec un formateur et/ou d'autres apprenants en sont les principales causes. Un engagement actif permet aussi une connaissance plus profonde de la matière (cfr page 1.1.2).

La lecture à l'écran peut rapidement devenir monotone et donc avoir un effet négatif sur la motivation de l'apprenant. Le format du tutoriel lui permet d'être imprimé facilement. Les exemples fournis qui soulignent la théorie et qui sont inextricablement liés avec elle augmentent l'interactivité possible avec l'apprenant mais celle-ci reste très limitée. Or, l'interactivité est un facteur important dans la rupture de la monotonie et permet donc de favoriser la concentration. Elle peut se faire de diverses manières telles que des "points questions" qui souligneraient des points importants de la matière, des notions plus avancées signalées comme telles ou des exemples et contre-exemples. Ce dernier point n'est pas seulement un choix dans l'interactivité mais également un choix pédagogique sur la pertinence de contre-exemples, choix qui serait peu judicieux dans le cas de cet apprentissage.

4.9.2 Contenu limité

Ce tutoriel ne s'intéresse qu'à la partie de développement d'une interface graphique. Or, lorsque le développeur crée une interface graphique, c'est généralement pour l'intégrer à une application. L'avantage de se focaliser sur l'interface graphique est que l'on peut approfondir le sujet et appuyer sur les règles ergonomiques à appliquer lors du développement. Le désavantage est que le développeur novice préférera un tutoriel abordant tous les aspects de la programmation et se désintéressera d'un tutoriel spécialisé.

Seule une technologie de développement est présentée alors que beaucoup d'autres existent. Le tutoriel pourrait présenter SuperWaba ou C++.

4.9.3 La stratégie pédagogique

L'évaluation des erreurs Le tutoriel ne propose aucune fonction d'évaluation. L'apprenant pourra détecter ses erreurs, et les corriger, en programmant lui-même, soit à la compilation soit à l'exécution. Cependant, le tutoriel pourrait prévoir certaines erreurs possibles de l'utilisateur et appuyer dessus en expliquant les raisons possibles et probables. Cela permettrait d'en éviter certaines et de donner à l'utilisateur si, malgré tout, elles ont lieu un moyen de les résoudre.

La réactivation des connaissances antérieures Le tutoriel demande aux utilisateurs de connaître le langage Java, or, il n'y fait jamais référence. La réactivation des connaissances antérieures permet aux nouvelles connaissances de se "greffer" sur les connaissances existantes et, donc, d'intégrer plus facilement et durablement les nouvelles connaissances.[dVM01]

La contextualisation Les exemples ont rarement un contexte significatif c'est à dire, une situation que l'apprenant pourrait rencontré dans la vie réelle. La contextualisation permet à l'apprenant d'enregistrer efficacement et de se remémorer plus facilement ces connaissances.[dVM01]

Chapitre 5

Perspectives futures

5.1 Améliorations du tutoriel ”papier”

5.1.1 Ajout de contenu

Un autre plus à apporter serait d'aborder d'autres technologies ou tout du moins en mentionner l'existence. Par exemple, on pourrait développer les technologies C++ ou SuperWaba.

5.1.2 Ajout d'exemples et d'exercices

La confrontation de l'utilisateur à des problèmes lui permet de faire le cheminement de la résolution. On pourrait mettre des exercices adaptés à la fin de chaque module et des exercices généraux dans un chapitre séparé. Il devrait y avoir soit plusieurs niveaux de difficultés distincts soit une évolution de la difficulté continue dans les exercices.

5.1.3 Auto-évaluation

Il est possible de créer un questionnaire général posant des questions de connaissances et ainsi vérifier l'assimilation du concept exposé. Ce questionnaire peut être découpé en sections qui reflètent la structure du tutorial afin que, s'il le désire l'utilisateur puisse faire une évaluation partie par partie. A la fin du questionnaire, une section reprenant des problèmes plus complexes et voyageant dans les différents modules peut être ajoutée.

En ce qui concerne la correction du questionnaire, deux méthodes sont possibles. La première est de donner un simple correctif avec les solutions avec un renvoi à la théorie. La deuxième est de donner des indications et/ou de renvoyer l'apprenant à la section pouvant l'aider à trouver la solution. Enfin, si c'est vraiment nécessaire, on peut donner la solution.

5.1.4 Réactivation des connaissances antérieures

Il faudra introduire un rappel sur les concepts de base de la programmation Java. Dans un premier temps, un rappel général s'intercalerait entre la théorie et les prédispositions pratiques. Il serait bienvenu, ensuite, de faire des rappels ponctuels dans les modules ou des comparaisons avec Java. Cela permettra de créer des relations entre anciennes et nouvelles connaissances et donc une meilleure mémorisation et structuration.

5.2 Création d'un site Internet

L'utilisation d'Internet dans l'apprentissage permet d'éliminer les contraintes d'espace et de temps présentes dans l'enseignement traditionnel. Grâce à un développement approrié, l'environnement pourra être accessible via une multitude de plates-formes informatiques. Cet avantage est très important au vu de la diversité de ces plates-formes. Il permet un accès immédiat aux mises-à-jour contrairement à une version sur CD-ROM.

Une plate-forme d'apprentissage permet une grande interactivité avec l'apprenant. Celui-ci a une réelle autonomie face à son apprentissage, une structure adaptée pouvant aussi le guider. Il peut suivre des liens vers les divers modules et vers des documents externes et les visualiser à l'écran, il peut télécharger des exemples, des exercices ou des documents qu'il peut exécuter ou imprimer.

La création d'un site internet réellement interactif amènera le tutoriel à devenir un environnement d'apprentissage informatisé. Ce qui n'était pas le cas du tutoriel papier ou en tout cas très faiblement informatisé. Le support informatique n'était utilisé que pour une fonction de communication des connaissances unilatérales et non d'interaction.

L'apport d'interactivité améliorera grandement le niveau de motivation de l'apprenant.

Il serait certainement intéressant de mettre le tutoriel en ligne pour un plus grand accès, et accompagné d'un forum. Le forum permettrait aux plus expérimentés d'aider les novices et ainsi de continuer la formation. Cela permettra de passer à de l'e-learning et aux avantages qu'il peut apporter, comme l'échange de connaissances et la possibilité de questionner des experts.

Le site permettra également une plus grande interactivité puisqu'il sera possible de mettre directement en relation certaines informations. Par exemple, il est possible de mettre des liens dans la partie théorique vers des exercices et inversement.

La disponibilité de deux types de documents (imprimable et de navigation) permet une plus grande flexibilité. L'utilisateur n'ayant pas de ligne internet ou un simple modem préférera sans doute une version téléchargeable du tutoriel. Le didacticiel sera plus adapté à un individu possédant une connexion haut-débit qui ne regardera pas le temps alloué à la formation.

Certaines informations pertinentes pour l'utilisateur peuvent être fournies telles que le temps normal de travail d'apprentissage par module, la distinction entre niveaux de

difficultés ou entre l'essentiel et les suppléments.

5.2.1 Conception

Structuration du contenu

Premièrement, pour faciliter la navigation et, deuxièmement, pour une meilleure assimilation des connaissances, le contenu doit être structuré et hiérarchisé. Cela permettra de représenter les relations entre les différents modules du site et les parcours possibles à travers la structure.

Cette structure doit être cohérente, compréhensible par l'utilisateur. Le contenu doit être clair et concis. Une fois le contenu défini, des unités sémantiques indépendantes sont constituées. Elles permettent de faciliter le traitement de l'information, la mise à jour et les combinaisons logiques entre ces unités.

Élaborer les stratégies pédagogiques

Cette phase pose le choix des stratégies et des médias les plus adaptés. Chaque unité de contenu peut avoir une stratégie différente, mais l'ensemble doit rester cohérent.

Une proposition de stratégie peut être, par exemple, de fournir du texte agrémenté d'exemples et d'images fixes. On peut y ajouter des hyperliens vers des exercices interactifs, accompagnés d'une aide et d'une correction possible, ou vers des sites de références.

Bâtir l'organigramme

L'établissement d'un organigramme impose de se construire une vue globale du site, ce qui permettra d'avoir plus de clarté et de cohérence. L'organigramme représente tous les liens logiques voulus. Quelques structures d'information sont représentées à la figure 5.1

Le site doit être conçu de manière à ce que l'utilisateur puisse naviguer et trouver son chemin sans devoir fournir un effort important pour comprendre la structure. L'utilisateur doit se représenter mentalement la structure du site qu'il découvre au fur et à mesure de sa visite, il est donc préférable que celle-ci soit la plus intuitive possible, simple et efficace. Cela facilitera la recherche et l'accès à l'information.

Les pièges à éviter sont les impasses et les boucles qui peuvent perdre l'utilisateur.

Quelques principes pour construire un organigramme sont de démarrer avec une page d'accueil d'où sera accessible le menu principal. Ce menu ne devrait pas comporter plus de cinq à sept unités et le nombre de sous-menus devrait se limiter à trois niveaux. Pour certains sites, il peut être nécessaire d'intégrer un outil de recherche. La page d'accueil a plusieurs objectifs dont le premier est d'introduire l'utilisateur sur le site. Elle présente les objectifs, les auteurs et le menu. La page d'accueil est une page que l'on doit pouvoir atteindre à partir de n'importe quelle autre page du site. Diverses unités peuvent être

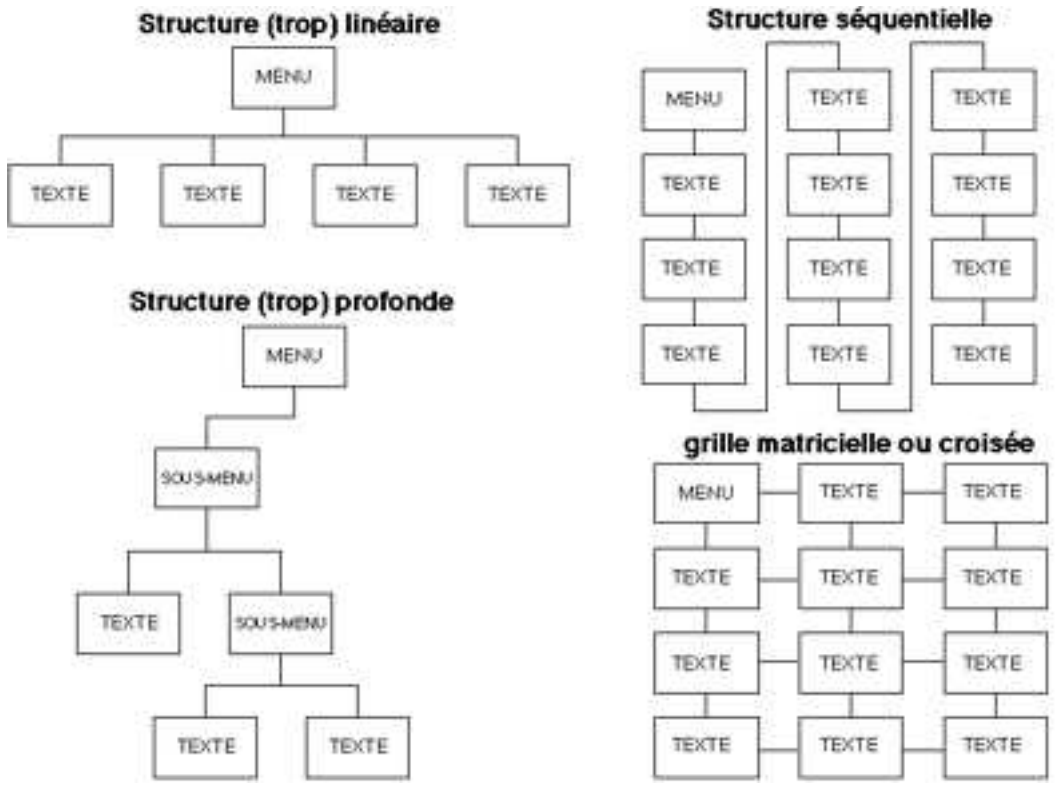


FIG. 5.1 – Exemples de structures d'information

ajoutées au site au besoin. Cela peut être une FAQ, un forum de discussion, une messagerie instantanée ou une section d'aide.

Dans le cas qui fait l'objet de ce travail, les unités peuvent être calquées sur le tutoriel "papier" et peuvent être les suivantes :

- Présentation générale de la plate-forme J2ME
- Modules
- Guidelines
- Codes
- Outils

Unités auxquelles on peut ajouter un module pour des exercices et un forum. Une FAQ et/ou une section d'aide peuvent également être intégrées.

Le site pourrait se structurer de différentes manières mais certains liens doivent être présents, ainsi que certaines hiérarchies. La section "Modules" devrait contenir au moins les trois modules établis dans le tutoriel, elle peut également contenir la section "Pré-requis". Des liens entre les sections "Exercices", "Modules" et "Codes" sont indispensables ainsi qu'un lien entre les sections "Pré-requis" et "Outils".

L'ergonomie

Le texte La lecture de données numériques ne se fait pas de la même manière qu'une version papier. Alors que pour cette dernière, la lecture est linéaire, un document électronique permet la navigation. Une version imprimable du document est toujours un plus. La version papier et la version numérique d'un texte ne sont pas un simple transposition de l'une vers l'autre. Le texte doit être adapté au support. Un texte lu à l'écran devra être morcelé en unités logiques et synthétisé. Il pourra mettre à profit les avantages interactifs du support. Le texte à imprimer sera structuré de manière plus traditionnelle.

La mise en page La disposition des éléments à l'écran détermine la lisibilité de l'écran et donc sa compréhension par l'utilisateur.

L'image L'image peut apporter une meilleure compréhension du contenu si elle est utilisée comme support visuel. Elle peut être utilisée comme métaphore et augmenter la lisibilité de l'écran. Elle peut être interactive et proposer des hyperliens. Elle peut aussi simplement rendre le design plus esthétique.

5.2.2 Propositions de structures

Voici deux manières de structurer le site :

Le schéma représenté à la figure 5.2 propose une structure constituée d'une page d'accueil proposant un menu de sept unités :
- Présentation générale de la plate-forme J2ME
- Modules
- Exercices
- Guidelines
- Codes
- Outils
- Forum

Un problème de cycle peut se produire au niveau des exercices et des modules, si on ajoute des relations de l'unité "exercices" vers l'unité "modules". Cette relation peut ne pas être nécessaire si une aide est fournie lors de la résolution.

Le schéma représenté à la figure 5.3 propose une structure constituée d'une page d'accueil proposant un menu de cinq unités :
- Présentation générale de la plate-forme J2ME
- Modules
- Guidelines
- Téléchargements
- Forum

L'inconvénient de cette structure est qu'elle ne permet pas un accès direct aux exercices.

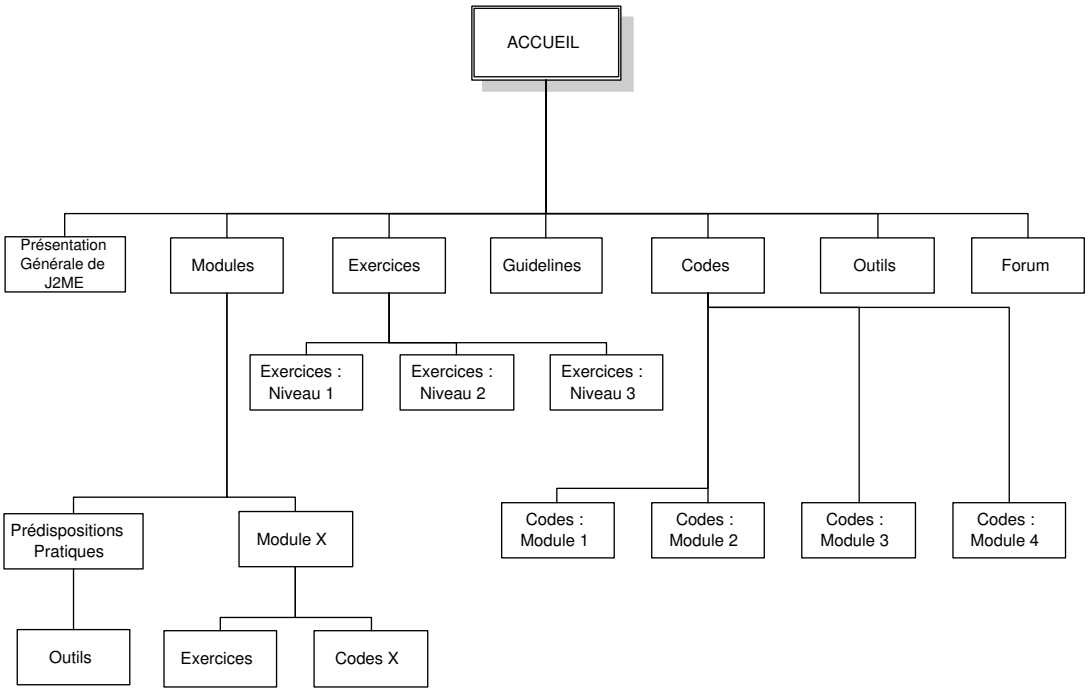


FIG. 5.2 – Structure d'un site composé d'une page d'accueil et d'un menu à sept modules

Un outil mathématique adaptable à l'informatique

Un outil pour l'apprentissage des mathématiques créé à l'université de Laval au Québec [LAVAL01] a une structure qu'il est peut-être possible d'adapter au cas du développement d'interfaces graphiques.

Trois modules constituent l'outil un tutoriel, un exerciseur et un démonstrateur. Ces modules peuvent être utilisés indépendamment mais ils sont reliés entre eux pour permettre à l'apprenant de voyager entre. L'organisation de ses liens a été étudiée de manière à ce que l'utilisateur ne se retrouve pas au point de départ à son insu. La navigation se fait via des fenêtres se superposant. Il arrivera à un moment donné dans une fenêtre sans lien, ce qui l'obligera à revenir à la fenêtre précédente.

Le tutoriel contient la théorie des concepts mathématiques vus. Dans le cas du développement graphique, le tutoriel peut être assimilé au tutoriel "papier" et, sur le site, à sa version numérique.

L'exerciseur propose à l'utilisateur divers exercices avec trois niveaux de difficultés. Cela lui permet d'évoluer dans son apprentissage.

L'adoption de plusieurs niveaux d'exercices permet une plus grande personnalisation de l'apprentissage. Les utilisateurs les plus débutants ne seront pas démotivés face à des exercices non-adaptés à leur niveau de connaissances.

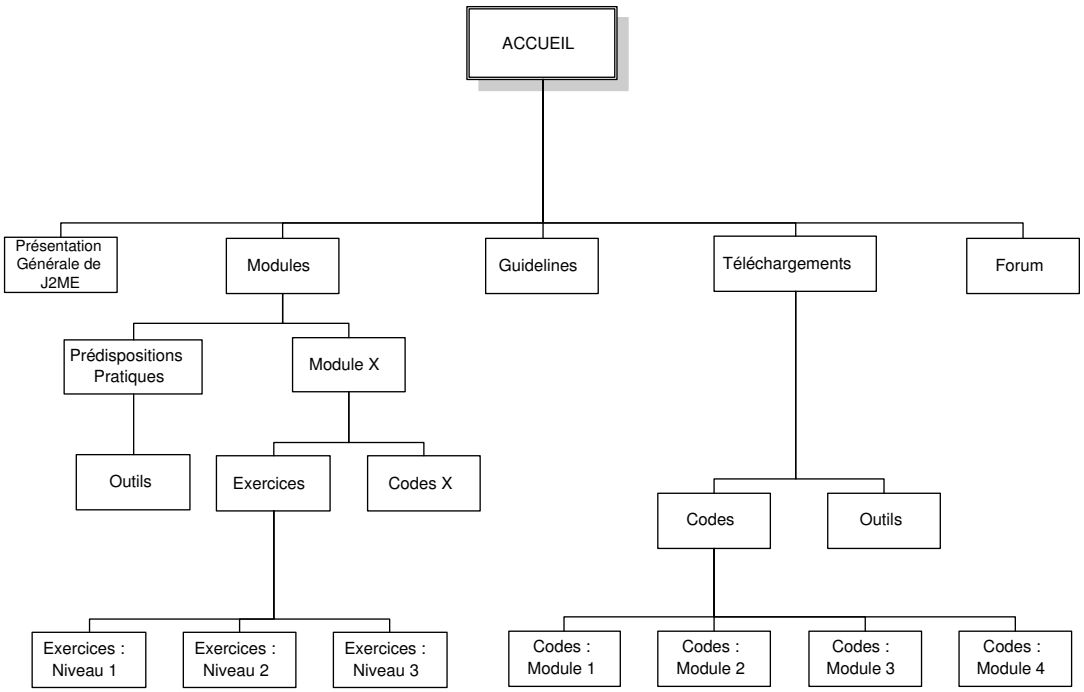


FIG. 5.3 – Structure d'un site composé d'une page d'accueil et d'un menu à cinq modules

Cet exerciseur appliqué à la programmation ne peut pas être conçu exactement de la même manière que pour les mathématiques. Dans ce cas, on peut imaginer un exercice proposant aux utilisateurs des parties de codes, en leur demandant de les compléter. Une aide pourrait leur être fournie en leur proposant plusieurs choix de réponses possibles. A un niveau de difficulté, un code complet pourrait leur être demandé en leur donnant une interface graphique qu'ils devraient essayer de coder. A ce moment, des aides pourraient leur être apportées en donnant des parties de codes. Si l'utilisateur n'est pas capable de résoudre les exercices malgré les aides qui lui sont apportées, il doit être renvoyé vers un niveau d'exercices de difficulté inférieure et/ou une solution devrait lui être proposée.

Le démonstrateur établit une correspondance entre un concept mathématique et le phénomène physique qui lui est associé. Cet outil permet à l'utilisateur d'expérimenter par lui-même les effets de variations sur les différents facteurs.

L'IDE peut faire office de démonstrateur mais le site pourrait également fournir un template avec des choix multiples proposés pour qu'il voit les différences sans devoir coder lui-même puisque ce point fait partie de l'exerciseur. L'utilisateur pourrait, par exemple, expérimenter lui-même les différences d'agencement des boutons en fonction des appareils et du nombre de boutons.

5.2.3 Le forum : impact sur la motivation

L'ajout d'un forum de discussion au sein d'un site Web éducatif permet au professeur d'échanger avec le groupe de façon asynchrone, tout en conservant un suivi de ces échanges. Pour ce faire, un logiciel de gestion du forum doit être installé sur le serveur qui héberge le site. Plusieurs solutions logicielles s'offrent à nous. Le choix s'effectuera en fonction des besoins, des budgets disponibles et de la plate-forme utilisée (Unix, Windows, Mac OS).

Le forum peut avoir un impact sur la motivation de l'apprenant, puisqu'il peut éviter un certain nombre de décrochage. D'une part, parce que l'apprenant peut obtenir des solutions à des problèmes qui pour lui sont insolubles. D'autre part, il peut trouver une certaine satisfaction à être actif dans le forum grâce à l'aide qu'il peut apporter aux autres débutants. Le traitement actif de l'information facilite la mémorisation des connaissances. L'apprenant doit donc participer activement au processus d'apprentissage. Cela se fait par différents moyens. Premièrement, il doit exécuter les exercices proposés sur son ordinateur et observer ses effets. Il doit essayer ensuite de résoudre par lui-même des problèmes. Deuxièmement, l'explication de certains points à d'autres apprenants permet une révision de la matière. L'apprenant peut se rendre compte que certains points qu'il croyait avoir compris restent en fait assez obscures. De plus, en expliquant les concepts, il acquiert une connaissance plus approfondie de la matière.

"La présence du groupe est une source d'information, un agent de motivation, un moyen d'entraide et de soutien mutuel et comme lieu privilégié d'interaction pour la construction collective des connaissances."

La présence d'un formateur peut être enrichissante pour le groupe mais ce concept n'est pas applicable ici. La présence d'experts et d'un ou plusieurs modérateurs sur le forum.

De plus, il est enrichissant de pouvoir expliquer et faire comprendre à d'autres des concepts que l'on a acquis. Comme énoncé dans la théorie constructiviste de l'apprentissage, la construction des connaissances passe par une mise en pratique. L'enseignement est une forme de mise en pratique des connaissances, car il faut les structurer pour pouvoir les réexpliquer. L'aide apportée aux autres fait donc partie du cycle d'apprentissage.

5.3 Limites du support Internet

Lors de la conception du site, il faut tenir compte du temps de chargement. Un temps trop long peut décourager certains utilisateurs surtout s'ils utilisent une connexion à faible débit. Il faut tenir compte de cette contrainte pour les images mais également pour les exercices proposés.

”Chapitre i” devant le nom du chapitre

Conclusion

L'enseignement peut prendre différentes formes allant d'un enseignement traditionnel, avec la présence d'un formateur, à l'autoformation où l'apprenant gère sa formation seul. Le tutoriel s'inscrit dans un enseignement de type autoformation pour la grande flexibilité qu'il offre. Il est important de choisir le type d'enseignement le plus adapté en fonction du public visé et de sa motivation, du contenu de la formation et de la disponibilité du formateur. Dans un contexte où tout le monde est interconnecté, la formation à distance via Internet semble naturelle. Elle offre une multitude de possibilités quant au type de formation, au contenu, et surtout offre une grande flexibilité dans le temps et l'espace.

Lors du développement d'interfaces graphiques, les utilisateurs se sont trouver face à plusieurs problèmes. Premièrement, "Mr Tout le Monde" peut être en possession d'un appareil mobile et vouloir développer sa propre application. Les utilisateurs sont donc peu spécialisés. C'est le propre des technologies déjà entrées dans une phase mature de leur existence. Les utilisateurs ont donc besoin de trouver un moyen simple pour réaliser leur objectif.

Deuxièmement, l'apprentissage de la programmation n'est pas trivial et pose des problèmes dans la résolution de tâches. L'apprentissage de la programmation pose plusieurs problèmes mais le principal se définit par "faire faire une tâche" à l'ordinateur. De là, découlent les autres problèmes. Le concepteur d'interfaces se trouvera devant ces même problèmes. Il ne s'agira pas de dessiner simplement une interface, mais de donner les instructions à l'exécutant-ordinateur pour qu'il la réalise.

Troisièmement, les interfaces homme-machine destinées aux appareils dotés de petits écrans voient une forte amplification des difficultés d'utilisabilité généralement rencontrées. Cela est principalement dû à la petite taille des écrans mais également à un contexte d'utilisation fortement différent d'un poste fixe. L'attention portée par l'utilisateur est réduite, l'interface doit donc être simple et demander peu d'effort.

Pour aider l'utilisateur à développer des interfaces utilisables, le tutoriel propose un langage permettant peu de créativité mais qui est très fonctionnel. Il offre également quelques conseils d'ergonomie.

Il a été décidé d'écrire le tutoriel en plusieurs modules abordant chacun un type d'éléments graphiques différents. Chacun propose un code se basant sur des concepts vus dans les modules précédents. Il y a donc une évolution dans l'acquisition des connaissances.

Cependant, l'apprenant peut revenir sur un module particulier, puisque le contenu d'un module forme un tout.

L'apprentissage par les exemples a été choisi pour offrir à l'utilisateur des modèles qu'il pourra reproduire pour le développement de sa propre interface. Il pourra également réutiliser certaines parties de code. C'est un gain de temps et d'énergie appréciable pour un apprenant qui ne souhaite pas entrer dans le professionnel.

Bibliographie

[ABE01] David ABELE. *Programmation de dispositifs électroniques à faibles ressources de calcul et J2ME*. 2001.

[ADA02] Lamont ADAMS. J2me : les bases de la programmation gui, 2002. http ://www.zdnet.fr/builder/programmation/java_c_cplusplus/0,39020934,2126275,00.htm.

[AND01a] Dean W. ANDREAKIS. Java 2 micro edition, Avril 2001. http ://venus.cs.depaul.edu/MS-seminar/symposium2001/andreakis1.ppt1.

[AND01b] Dean W. ANDREAKIS. *J2ME CLDC/MIDP Overview UI Proposal*. février 2001.

[BER96] J. BERBAUM. Apprendre à apprendre, février-mars, 1996. http ://www.grics.qc.ca/cles_en_main/projet/peda0398.pdf.

[CAI] Christophe CAIGNAERT. Étude de l'évolution des méthodes d'apprentissage et de programmation. http ://archive-edutice.ccsd.cnrs.fr/docs/00/03/07/38/PDF/b50p052.pdf.

[CHI01] Sylvie CHIOUSSE. Pédagogie et apprentissage des adultes : État des lieux et recommandations, février-mars, 2001. http ://www.oecd.org/dataoecd/5/44/1831501.pdf.

[CL03] Corina CILIANU-LASCU. *Techniques D'Enseignement / Apprentissage À Distance Intégration Des Nouvelles Technologies De L'Information Et De La Communication*. 2003.

[DEL02] Bruno DELB. Présentation de j2me : Première partie, janvier 2002. http ://developpeur.journaldunet.com/tutoriel/jav/020129jav_j2me1_1.shtml.

[DES05] LITTLE SPRINGS DESIGN. *User Interface Design Guidelines : J2ME MIDP 2.0*. 2005.

[DOU04] Jean Michel DOUDOUX. Développons en java, 2004. http ://perso.wanadoo.fr/jm.doudoux/java/tutorial/index.htm.

[DUC92] Charles DUCHÂTEAU. De faire à faire faire : au coeur de la programmation : quelques réflexions didactiques, 1992. Namur, Belgium.

[DUC00] Charles DUCHÂTEAU. Serveur pÉdagogique et formation a distance premiers enseignements d'une expérience de formation organisée partiellement à

distance : une initiation (au traitement de texte)en autonomie, balisée et assistée, Juillet 2000.

[DUC02] Charles DUCHATEAU. Images pour programmer, 2002. CeFIS - FUNDP.

[dVM01] CEGEP du Vieux Montréal. L'apprentissage par problèmes, Juin 2001.

[EE01] Nathalie EVEN Eric ECOUTIN. Fiche pratique n°3 : Les documents, janvier 2001. http ://ressources.algora.org/telechargement/tel/document.pdf.

[ENS04] ENSMA. *Validation d'une approche « End-user Programming » pour l'apprentissage de la programmation.* 2004.

[FIC] Assistants personnels numériques (pda). http ://www.rcip.gc.ca/Francais/Contenu_Numerique/Fiches_Techniques/Pda/fiche_tech16.html.

[FMG94] Xavier ROEGIERS François-Marie GERARD. Évaluer un projet d'informatique pédagogique : une question de questions, 1994. Louvain-La-Neuve, http ://www.bief.be/enseignement/publication/informat.html.

[GIG02] Eric GIGUERE. J2me optional packages, 2002. http ://developers.sun.com/techtopics/mobility/midp/articles/optional/.

[GIG05] Eric GIGUERE. Using custom items in midp 2.0, February 05. http ://developers.sun.com/techtopics/mobility/midp/ttips/customitem/.

[HIB05] Paul D. HIBBITTS. Small screens, big lessons, 2005. http ://www.smallscreensbiglessons.com/.

[J2Ma] Importing existing j2me midp source code into netbeans ide 4.0. http ://www.netbeans.org/kb/articles/import-mobility-40.html.

[J2Mb] J2me devices. http ://developers.sun.com/techtopics/mobility/device/device.

[J2Mc] J2me midp development for netbeans ide 4.0 quick start guide. http ://www.netbeans.org/kb/articles/quickstart-mobility-40.html.

[J2Md] J2me : Step by step. http ://www.digilife.be/quickreferences/PT/J2ME

[J2Me] Java 2 micro edition (j2me). http ://krugazor.free.fr/J2ME.pdf.

[JAL93] Todd R. KAUFMANN James A. LANDAY. *User Interface Issues in Mobile Computing.* octobre 1993.

[JBa] Christian ST-PIERRE Josée BASTIEN. Session 2 - les nouveaux outils et les nouveaux produits pour de nouveaux usages dans la pÉdagogie et l'andragogie. Québec, Canada. http ://web.gci.ulaval.ca/opea-mat/vitrine/article1.pdf.

[JBb] Christian ST-PIERRE Josée BASTIEN. *Un outil pédagogique pour l'enseignement et l'apprentissage des mathématiques de l'ingénieur.*

[JB98a] Laura WINER Josianne BASQUE, Johanne ROCHELEAU. Une approche pédagogique pour l'école informatisée, 1998. http ://www.grics.qc.ca/cles_en_main/projet/peda0398.pdf.

[JB98b] Sylvie DORE Josianne BASQUE. Le concept denvironnement dapprentissage informatisé, 1998. http ://cade.icaap.org/vol13.1/dore.html.

[JCP] Guy PIERRA Frédéric BESNARD Jean-Claude POTIER, Patrick GIRARD. Génération graphique interactive de programmes de géométrie paramétrée. Laboratoire dInformatique Scientifique et Industrielle, ENSMA, Site du Futuroscope.

[JK03] Dana NOURIE Jonathan KNUDSEN. Wireless development tutorial part i, September 2003. http ://developers.sun.com/techtopics/mobility/midp/articles/wtoolkit.

[KEE96] D. KEEGAN. Foundations of distance education, 1996.

[KEO03] Jim KEOGH. The new pda profile, jdj, janvier 2003. http ://www.sys-con.com/author/ ?id=1202.

[KON03] Mikko KONTIO. Custom gui development with midp 2.0, 2003. http ://www-128.ibm.com/developerworks/library-combined/wi-developui/tmp0000.html.

[KON04] Mikko KONTIO. Architectural manifesto : Designing mobile user interfaces, 2004. http ://www-128.ibm.com/developerworks/wireless/library/wi-arch4/index.html.

[MAH] Qusay MAHMOUD. *MIDP GUI Programming, Learning Wireless Java.*

[MAH01] Qusay MAHMOUD. chapter 5 : Midp gui programming, learning wireless java, 2001. http ://developers.sun.com/techtopics/mobility/midp/chapters/wjqusay/ch5.pdf.

[MAR01] Aaron MARCUS. Babyface design for mobile devices and the web. In *Human-Computer Interface Internat. (HCII) Conf.*, volume 2, pages 514–518, Mahwah, NJ USA, 2001. Lawrence Erlbaum Associates. http ://www.amanda.com/resources/HCII01/ HCII01_Pn_Bab_Marcus_160401.pdf.

[NGa] Patrick GIRARD Nicolas GUIBERT. Programmation sur exemple et enseignement assisté par ordinateur de l'algorithmique : le projet melba.

[NGb] Patrick GIRARD Nicolas GUIBERT, Laurent GUITTET. *Apprendre la programmation par lexemple : méthode et système.* ENSMA.

[NG05] Patrick GIRARD Nicolas GUIBERT, Laurent GUITTET. Initiation à la programmation par l'exemple : concepts, environnement, et étude d'utilité, 2005. Montpellier.

[RIC90] J. F. RICHARD. *Les Activités mentales.* Armand Colin, 1990. Paris.

[RIT] Simon RITTER. Javatm 2 micro edition, connected limited device configuration, mobile information device profile, wireless messaging api (wma) mobile media api,. htp ://www.sun.com/developers /evangcentral.

[RIV01] Michel RIVEILL. J2me, Janvier 2001. http ://rangiroa.essi.fr/cours/info-mobile/02-slides-j2me.pdf.

[SAN04] Loé SANOU. Modele generique de programmation sur exemple, Septembre 2004. Laboratoire dInformatique Scientifique et Industrielle Université de Poitiers ENSMA.

[SH02] Henrik GEDENRYD Simon HOLLAND, David R. MORSE. The application of direct combination to mobile and ubiquitous human computer interaction., 2002. Computing Department, The Open University, Walton Hall, United Kingdom.

[SOE] Jasper SOETENDAL. Mobile user interfaces.

[SP] Gillian HAYES Shwetak PATEL, Heather MAHANEY. Ui environments. http ://www.cc.gatech.edu/classes/AY2005/cs4470_fall/lectures/lecture12-uienvironments.ppt.

[SPL] Marielle RICHE-MAGNIER Serge -LAJUS. Les technologies éducatives, une occasion de repenser la relation pédagogique. http ://www.txtnet.com/ote/text0007.htm.

[SUN] What's new in midp 2.0. http ://java.sun.com/products/midp/whatsnew.html.

[SUN02] Midp style guide mobile information device profile (midp) 1.0a, Août, 2002. http ://java.sun.com/j2me/docs/alt-html/midp-style-guide7/.

[TOP02] K. TOPLEY. *J2ME in a Nutshell.* 2002.

[UNE87] UNESCO. Glossaire des termes de technologie éducative, 1987. http ://www.educnet.education.fr/dossier/eformation/distance3.htm.

[VAN] Jean VANDERDONCKT. Les constructions d'interfaces par génération et par démonstration sont-elles conciliables ? Facultés Universitaires Notre Dame de la Paix, Institut d'Informatique.

[WAL01] Marc WALCKIERS. L'enseignement à distance, 2001. http ://www.ipm.ucl.ac.be/multimedia/MARC/ED-WebIPM-Defin.html.

[WIK] Wikipedia. http ://fr.wikipedia.org.

Annexe A

Vous trouverez dans les pages qui suivent, un exemplaire complet du tutoriel.