SPÉCIFICATION EXTERNES DU LOGICIEL CÔTÉ CLIENT

TRAVAIL PRÉSENTÉ À
MME SOUMAYA CHERKAOUI

DANS LE CADRE DU COURS
GEI450, PROJET DE CONCEPTION DE LOGICIELS

PAR L'ÉQUIPE SOKRATE :
SIMON BÉLANGER
YANNICK BROSSEAU
NICOLAS HATIER
CATHERINE PROULX
FRANÇOIS TREMBLAY

LE 30 MAI 2001

Université de Sherbrooke

SPÉCIFICATION EXTERNES DU LOGICIEL CÔTÉ CLIENT

TABLE DES MATIÈRES

1. Introduction

1.1. But du document

Ce document définit toutes les spécifications du didacticiel **Dedalus**. Destiné en premier lieu au clients et futurs utilisateurs de l'application, il a aussi un l'intérêt pour ses développeurs et testeurs.

1.2. Portée

Ce document couvre les spécifications de la version 1.0.0 de **Dedalus**. Il présente aussi certaines particularités sélectionnées qui seront probablement ajoutées dans des versions futures. Le but de ceci est de guider les développeurs dans la sélection d'un design qui pourra englober le didacticiel dans son ensemble.

1.3. Définitions, acronymes et abréviations

DTL – Document de test logiciel

SELD – Spécification externe du logiciel, côté développeur

PGPL – Plan de gestion du projet logiciel

1.4. Références

ERIC J.BRAUDE, Software Engineering, Wiley & Sons, 2001.

Plan de gestion du projet logiciel (PGPL)

1.5. Vue d'ensemble

Sera discutée dans la section 2.

2. Description

Dedalus se veut un didacticiel qui permettra à un utilisateur potentiel de s'initier aux concepts de la programmation structurée. Il combinera le jeu à un langage simple de programmation pour former une interface intéressante pour les utilisateurs (voir section 2.3). Il présentera des épreuves, soit des labyrinthes semés d'embûches où se déplacera un robot, desquelles le succès initiera la montée au niveau de difficulté suivant. Chaque niveau de difficulté sera caractérisé par la nécessité d'intégrer et d'utiliser une notion de programmation en particulier. Le succès de chaque niveau sera déterminé par l'efficacité des instructions produites par l'utilisateur, efficacité évaluée par rapport au parcours optimal.

Les notions de programmation à évaluer pour chaque niveau seront à déterminer plus tard, mais débuteront probablement par des commandes simples telles que **`avance`**, **`gauche`**, **`droite`**, etc. Pour bien intégrer les concepts de programmation, aucune de ces commandes ne pourra être utilisée en mode direct. L'utilisateur devra construire son script, et seulement ensuite l'exécuter pour mesurer son résultat sur le robot et sa performance.

Dans les premières versions du didacticiel, seuls quelques niveaux auront été implantés.

Une autre particularité à prévoir sera une interface permettant à un éducateur de suivre les progrès de son élève au fur et à mesure de son utilisation du didacticiel.

2.1. Interfaces

2.1.1. Interfaces utilisateur

L'interface utilisateur principale présentera trois zones importantes. La première sera une grille de jeu, plus précisément un labyrinthe, comprenant divers murs et autres obstacles qu'un petit robot devra traverser (Figure 1).

La seconde apparaîtra comme une télécommande, comprenant divers boutons permettant d'ajouter des instructions au script, à mesure que le niveau de l'utilisateur augmentera.

La dernière section contiendra le script que l'utilisateur aura déjà entrée. Il possédera, entre autres, des commandes permettant de déplacer des instructions, d'en insérer ou d'en effacer.

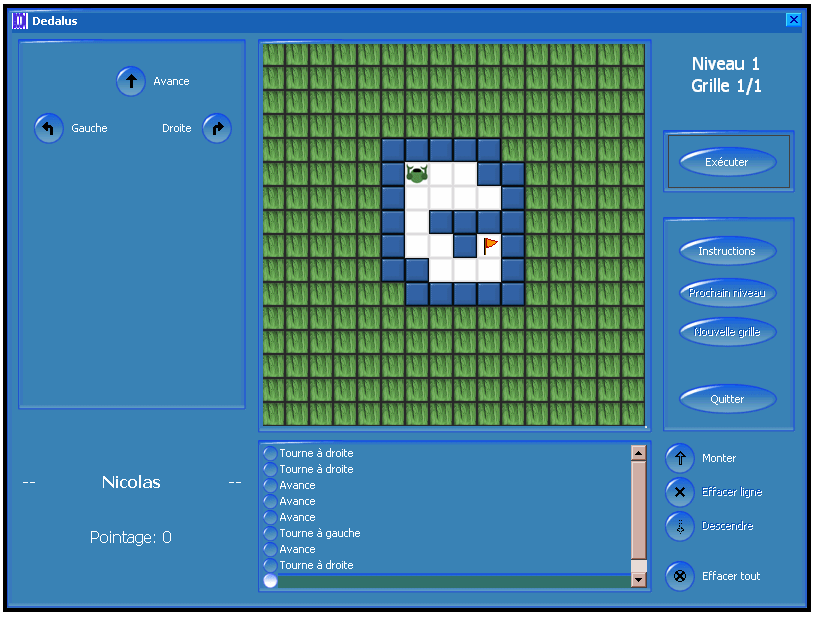

Figure 1 *– Interface principale*

Des petites zones secondaires afficheront le nom de l'utilisateur, le pointage, un résumé des instructions, ainsi que quelques commandes de contrôle du jeu.

Des fenêtres secondaires apparaîtront lors d'actions précises de l'utilisateur. Au commencement de l'exécution, une fenêtre de bienvenue apparaîtra (Figure 2). Il y aura aussi, dépendant du cas, une fenêtre d'identification ou une fenêtre d'entrée d'un nouveau nom (Figure 3). Une autre fenêtre présentera les instructions et les conseils à l'utilisateur (Figure 4).

Figure 2 *- Bienvenue*

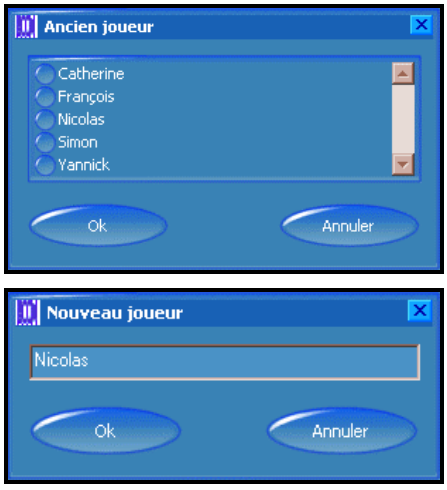

Figure 3 *– Identification / Nouveau*

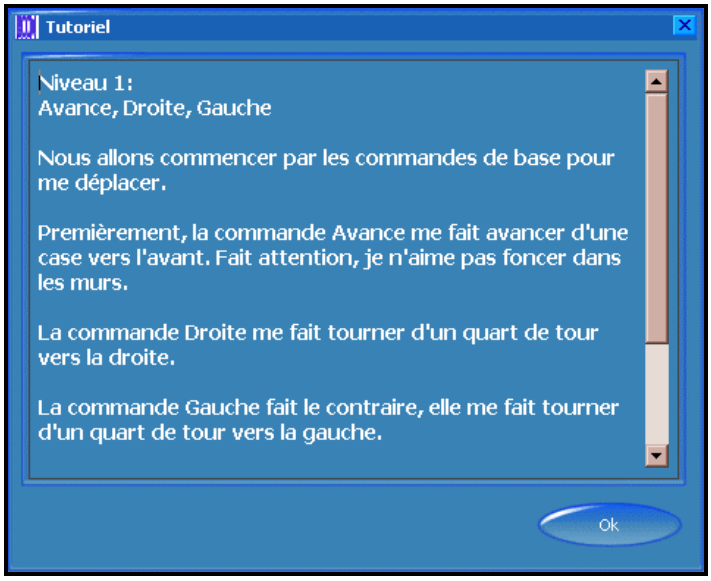

Figure 4 *- Instructions*

2.1.2. Autres interfaces

Le didacticiel ne contiendra aucune interface matérielle, logicielle ou de communications.

2.2. Contraintes mémoire

Dedalus ne devra pas prendre plus de 16 Mo de mémoire et autant d'espace disque.

Le plan de test pour s'en assurer sera détaillé dans le DTL.

2.3. Opérations

Il devra être possible pour l'utilisateur d'enregistrer son avancement afin que plusieurs personnes puissent utiliser la même installation du logiciel et avoir leur propres données.

2.4. Adaptation à l'environnement contextuel

Il est prévu que le logiciel sera disponible en plusieurs langues dans une version ultérieure.

3. Fonctions du produit

3.1. Cas d'utilisation

Les cas d'utilisation son présentés graphiquement à la figure 5

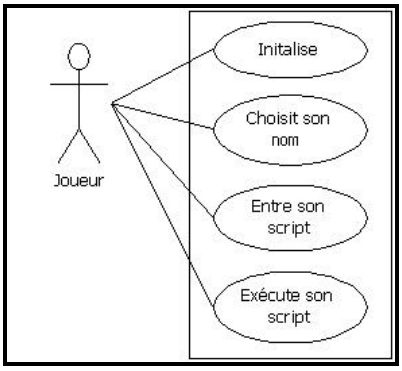

Figure 5 *– Cas d'utilisation*

3.1.1. Cas d'utilisation « Initialise »

Le système affiche la fenêtre d'accueil.

3.1.2. Cas d'utilisation « Choisit son nom »

Le système affiche une fenêtre permettant à l'utilisateur de choisir son nom dans une liste, si il l'a déjà entré auparavant, ou une fenêtre lui permettant d'écrire son nom.

3.1.3. Cas d'utilisation « Entre son script »

Le système ajoute les commandes sélectionnées par l'utilisateur dans une liste, lui permettant de bien suivre ce qu'il a entré.

3.1.4. Cas d'utilisation « Exécute son script »

Le système affiche le déplacement du robot en suivant les commandes entrées par l'utilisateur dans son script. Il affiche ensuite une fenêtre de rétroaction présentant un rapport de performance.

3.2. Phases du logiciel

Les phases du logiciel sont représentées graphiquement à la figure 6.

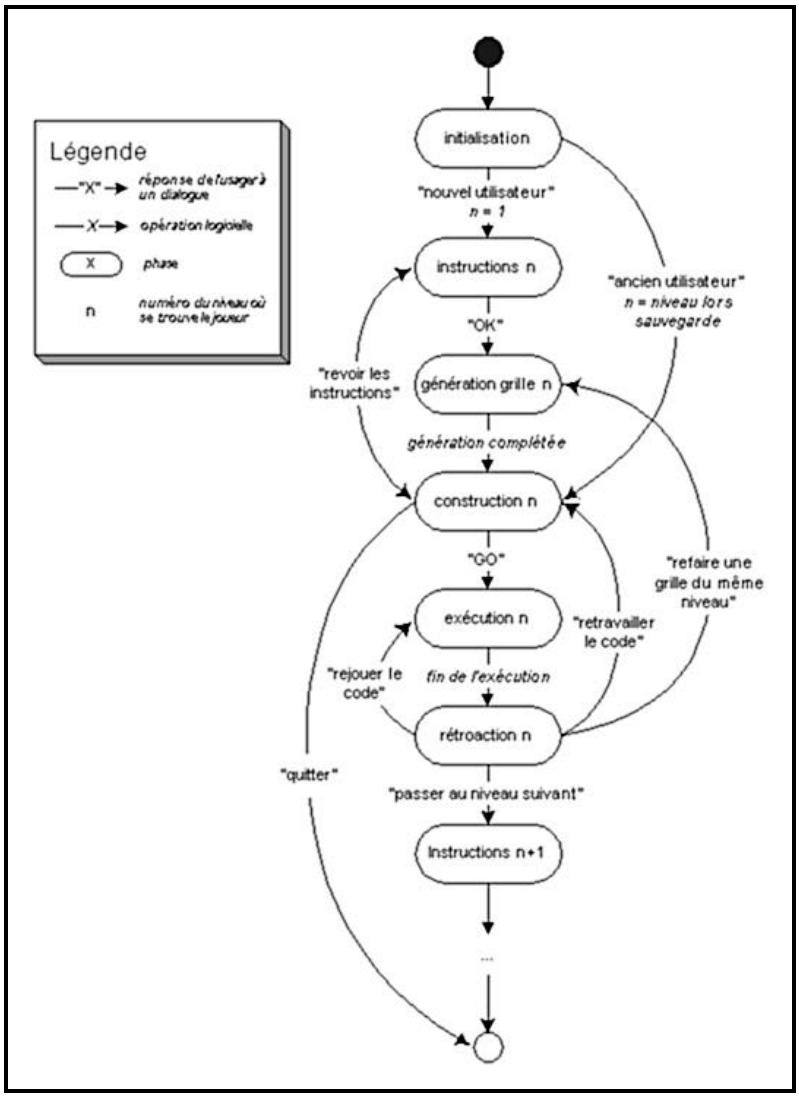

Figure 6 *– Diagramme des phases*

3.2.1. Phase « initialisation »

Cette phase survient au début de l'exécution de programme. Une fenêtre apparaît pour demander à l'utilisateur son nom, afin de charger ses données. Il a aussi la possibilité d'enregistrer son nom s'il n'a jamais utilisé le logiciel. En cliquant sur le bouton OK, le logiciel passe à la phase *instruction*.

3.2.2. Phase « instructions »

Cette phase survient au début de chaque niveau, au moment où de nouvelles commandes deviennent disponibles à l'utilisateur. Le logiciel présente les nouvelles fonctions à l'utilisateur, lui explique leur fonctionnement et lui donne des exemples de leur utilisation, ceci afin que l'utilisateur soit rapidement familier avec celles-ci. Dès que l'utilisateur a fini la lecture de ces instructions, le logiciel passe à la phase *génération.*

3.2.3. Phase « génération »

Cette phase survient à chaque fois qu'une grille de jeu doit être générée. Pendant cette phase, le logiciel génère la grille de jeu en fonction du niveau de l'utilisateur. Celui-ci ne peux pas interagir avec le logiciel pendant cette phase. Lorsque la grille est générée, le logiciel passe à la phase *construction*.

3.2.4. Phase « construction »

Le logiciel présente à l'utilisateur le labyrinthe. L'utilisateur construit son script de programmation pour tenter de le résoudre. Il a la possibilité de revoir les instructions (phase *instruction*), de générer une nouvelle grille (phase *génération*) ou de soumettre son code à la machine (phase *exécution*).

Le script que l'utilisateur peut produire est constitué d'instructions simples, qui deviennent accessible au fur et à mesure qu'il monte de niveau. Le tableau 1 présente les concepts d'instructions qui seront disponibles.

La grille est constituée de 256 cases (16 X 16). Le tableau 2 présente les différents types de cases et items que l'utilisateur pourra rencontrer. La figure 7 présente un exemple d'une grille de jeu de premier niveau.

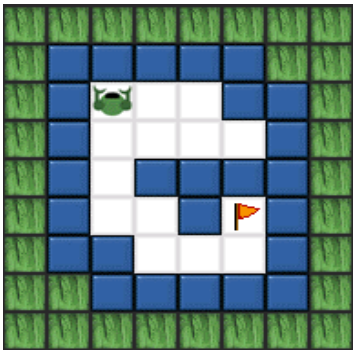

Figure 7 *– Grille de jeu*

Commandes de base	**Mots-clés de conditions et boucles**	**Conditions**
Avance Gauche Droite Ramasse Ouvre	Si Condition Sinon Tant que Condition Fais X fois	Objet / Pas Objet Porte / Pas Porte Sortie / Pas Sortie Ai Objet / Ai Pas Objet Mur / Pas Mur

Tableau 1 – *Concepts d'instructions*

Cases	**Items dans certaines cases**
Case vide Mur Porte fermée Gazon (zones inutilisées) Indéfinies (Le robot doit vérifier)	Caisse (« contenant » des morceaux de robot) Robot

Tableau 2 – *Contenu des grilles de jeu*

3.2.5. Phase « exécution »

L'utilisateur soumet son script à la machine. Le logiciel exécute œ script pas à pas en présentant ses effets sur le robot. L'utilisateur peut accélérer ou ralentir cette exécution. L'exécution terminée, le logiciel passe à la phase *rétroaction*.

3.2.6. Phase « rétroaction »

Un rapport de performance est présenté à l'utilisateur. Si le script n'était pas complet ou était erroné, ou si la performance était trop basse, il doit retourner à la phase *construction*. Sinon, il a le choix de revoir l'exécution, de retourner à la *construction* pour améliorer son script ou de passer au niveau suivant. Dans ce cas, le logiciel passe à la phase *instructions*.

4. Utilisateurs

Ce jeu sera conçu de façon à présenter un intérêt à des jeunes filles et garçons d'âge scolaire, plus précisément de la fin du primaire au milieu du secondaire (7 à 14 ans). Ces jeunes n'auront à posséder aucune connaissance particulière en informatique, si ce n'est de se servir d'une souris. Le jeu pourra aussi avoir son attrait pour des néophytes plus âgés qui voudraient apprendre la programmation sans se jeter immédiatement dans des langages existants sans préparation.

5. Contraintes

Le didacticiel devra pouvoir fonctionner sur un PC ayant le système d'exploitation Windows 95™ ou une version subséquente et une vitesse de processeur minimum de 133 MHZ et ne pas requérir de périphérique particulier [autre qu'une carte de son], afin qu'il soit utilisable sur la plupart des ordinateurs possédés par les écoles et les familles. Il devra cependant pouvoir atteindre une résolution d'écran de 800 x 600 avec 256 couleurs.

6. Hypothèses et dépendances

Non applicable

7. Priorités

Il va de soi que les spécifications décrites dans ce document devront être en tout temps consistantes avec celles décrites dans le SELD. Toute inconsistance devra être consignée comme un défaut. Dans le cas où une spécification serait définie autant dans ce document que dans le SELD, le logiciel devra être construit à partir de la définition du SELD puisqu'elle sera plus détaillée.

Les spécifications *essentielles* (telles que décrites dans le SELD) devront être implantées dans cette version du logiciel. Les spécifications *désirables* pourraient êtres implantées dans cette version si possible et si le temps le permet, mais aucun engagement n'est pris dans ce sens par les développeurs. Il est tout de même clair que ces spécifications seront incluses dans une version future. Quant au spécifications *optionnelles*, elles seront implantées à la discrétion des développeurs.