SPÉCIFICATION EXTERNES DU LOGICIEL CÔTÉ DÉVELOPPEUR

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 5 JUIN 2001

Université de Sherbrooke

SPÉCIFICATION EXTERNES DU LOGICIEL CÔTÉ DÉVELOPPEUR

TABLE DES MATIÈRES

TABLE DES FIGURES ET DES TABLEAUX

1. Spécifications des interfaces externes

1.1. Interface utilisateur

Dedalus se déroule principalement sur la même interface, qui contient tous les contrôles du jeu. Cette interface occupe une surface de 800 par 600 pixels, donc tout l'écran sur les ordinateurs affichant la résolution minimale.

Les contrôles du jeux sont des boutons fonctionnels. Cliquer sur ces boutons ajoute la commande correspondante dans la zone de script.

L'entière collection des interfaces s'énumère ainsi :

a. Une interface utilisateur principale qui contient les contrôles du jeu, le labyrinthe et la zone de script, telle que décrite dans la section 2.MU.

b. Une fenêtre de bienvenue, telle que décrite dans la section 2.ID.

c. Deux fenêtres d'identification, selon que le joueur soit nouveau ou déjà enregistré, telles que décrites dans les sections 2.ND et 2.ED.

d. Une fenêtre d'instructions, telle que décrite dans la section 2.TD.

e. Une fenêtre de paramètres pour les commandes de script, telle que décrite dans la section 2.PD.

f. Une fenêtre de rétroaction, telle que décrite dans la section 2.FD

1.2. Autres interfaces

Le logiciel ne contient pas d'interface matérielle, logicielle ou communication.

2. Précision des spécifications

2.1. Diagrammes de séquence

2.1.1. Phase *initialisation*

Les diagrammes de séquence pour la phase *initialisation* est présenté aux figures 1 et 2. Cette phase requiert les classes *DedalusGame*, *IntroDlg*, *NewUserDlg*, *ExistingUserDlg* et *MainUI*. Une seule instance de chacune de ces classes sera créée.

2.1.2. Phase *instructions*

Le diagramme de séquence pour la phase *instructions* est présenté à la figure 3. Cette phase requiert les classes *DedalusGame et TutorialDlg*. Une seule instance de chacune de ces classes sera créée.

2.1.3. Phase *génération*

Le diagramme de séquence pour la phase *génération* est présenté à la figure 4. Cette phase requiert les classes *DedalusGame, Grid, GridAnimator et MainUI*.

2.1.4. Phase *construction*

Le diagramme de séquence pour la phase *construction* est présenté à la figure 5. Cette phase requiert les classes *DedalusGame, ScriptContainer et MainUI*.

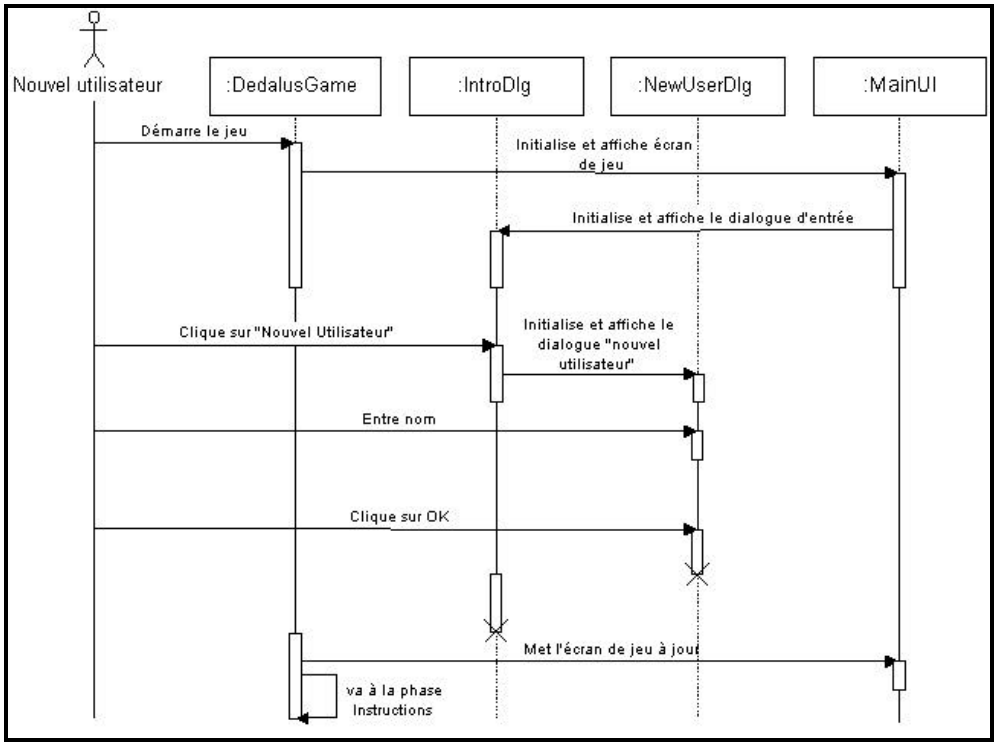

Figure 1 *– Diagramme de séquence pour la phase* initialisation *(Nouvel utilisateur)*

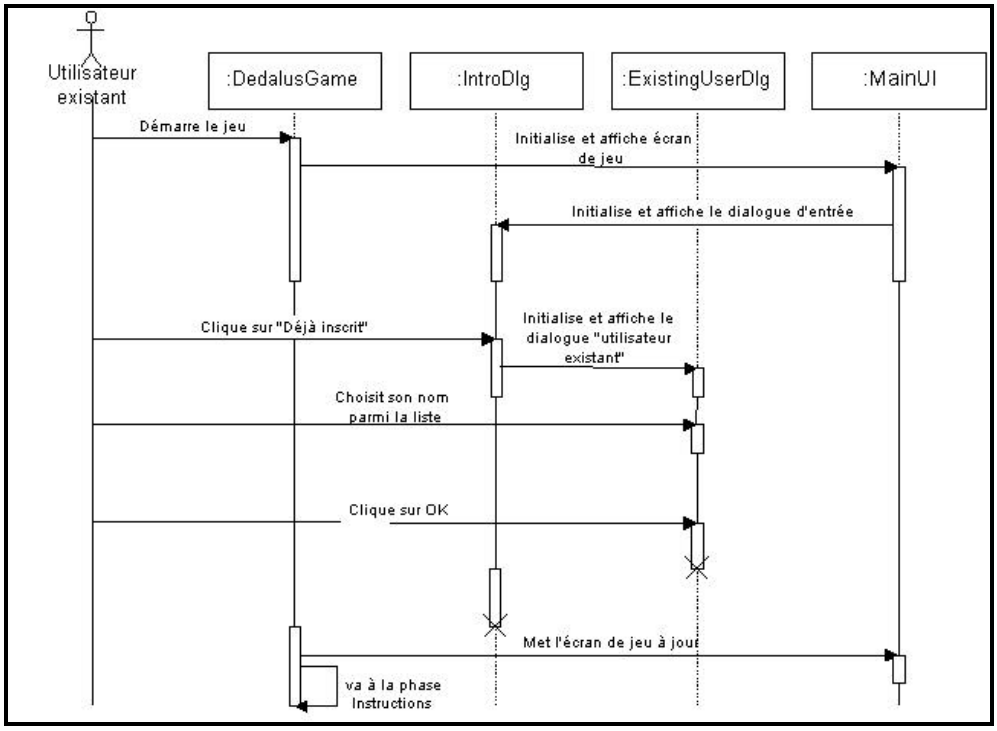

Figure 2 *– Diagramme de séquence pour la phase* initialisation *(Utilisateur existant)*

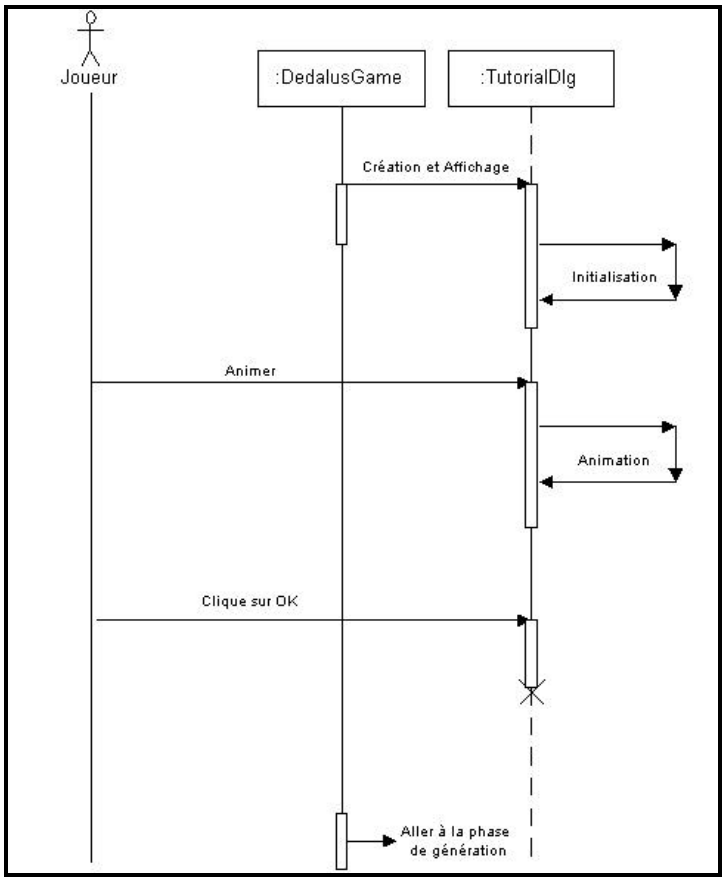

Figure 3 *– Diagramme de séquence pour la phase* instructions

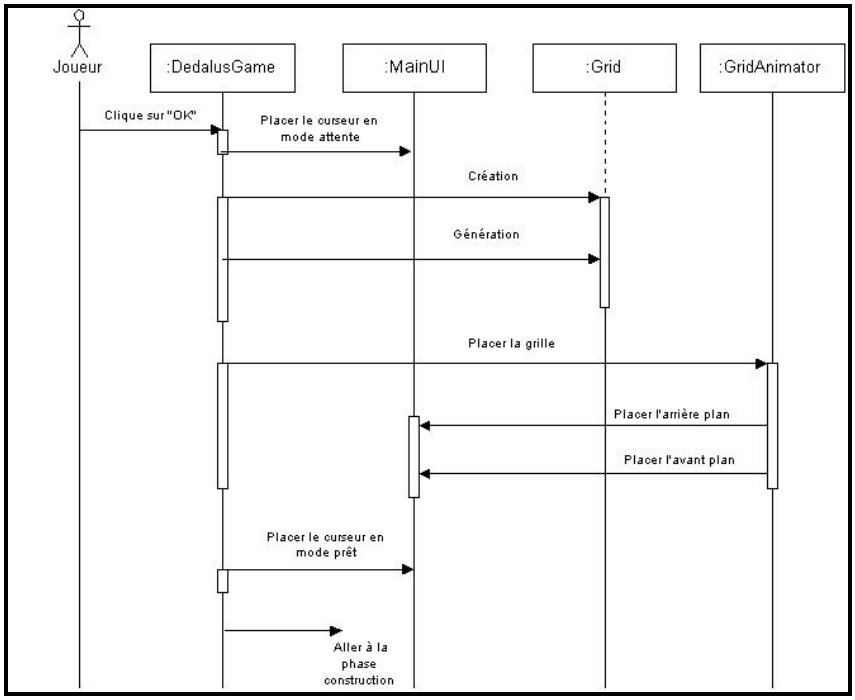

Figure 4 *– Diagramme de séquence pour la phase* génération

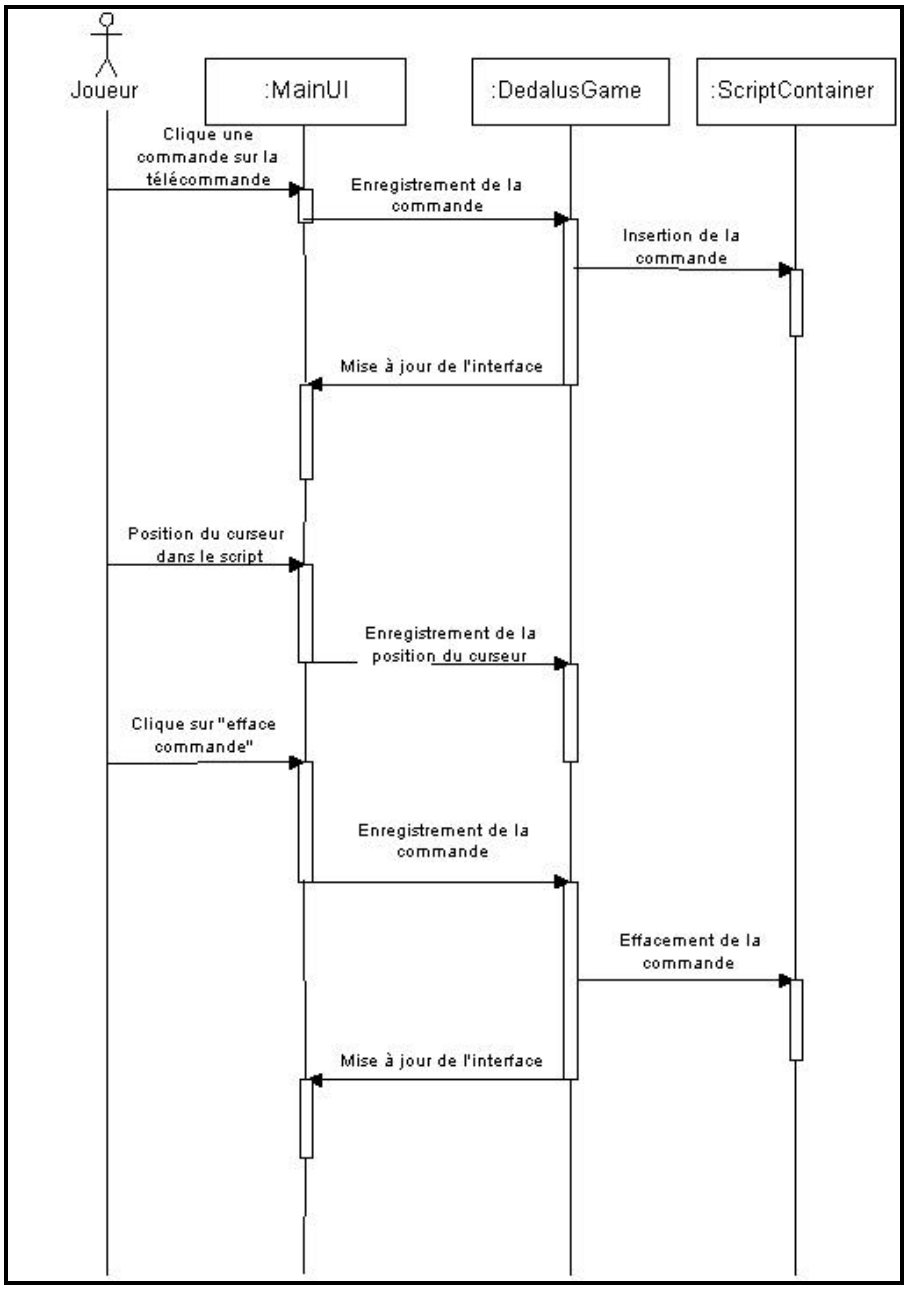

Figure 5 *– Diagramme de séquence pour la phase* construction

2.1.5. Phase *exécution*

Le diagramme de séquence pour la phase *exécution* est présenté à la figure 6. Cette phase requiert les classes *DedalusGame, ScriptContainer, ScriptAnalyser, GridAnimator et MainUI*.

2.1.6. Phase *rétroaction*

Le diagramme de séquence pour la phase *rétroaction* est présenté à la figure 7. Cette phase requiert les classes *DedalusGame, ScriptContainer, ScriptAnalyser, GridAnimator et MainUI*.

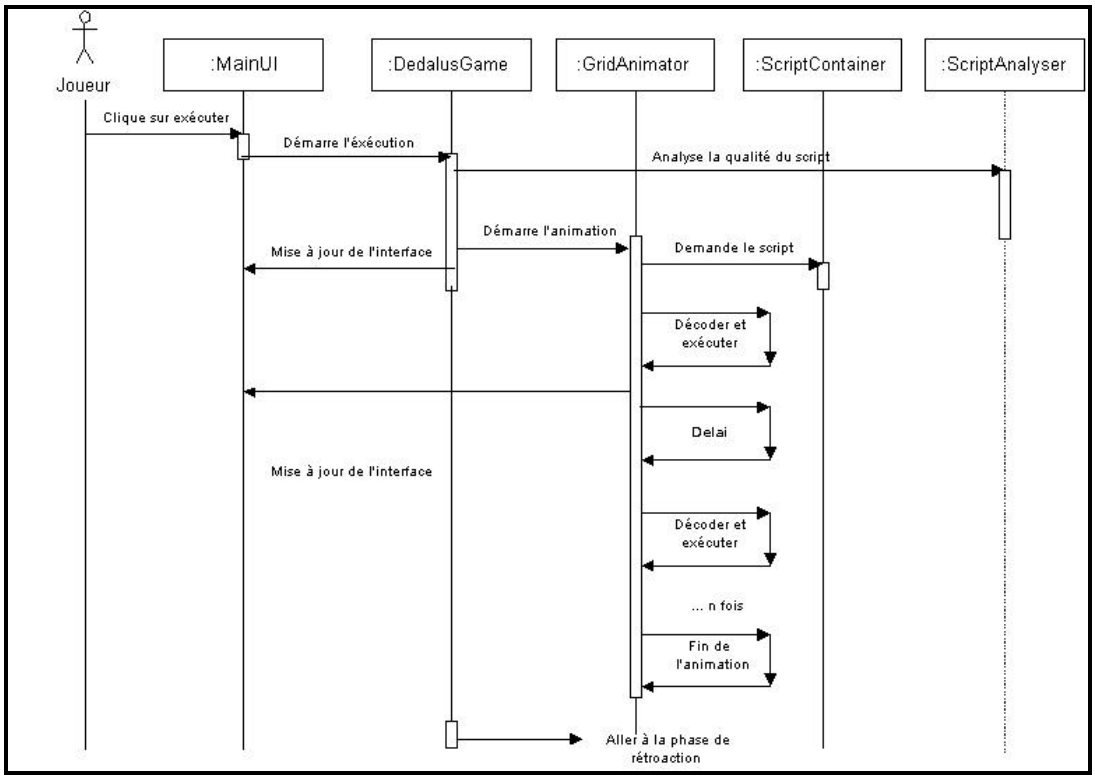

Figure 6 *– Diagramme de séquence pour la phase exécution*

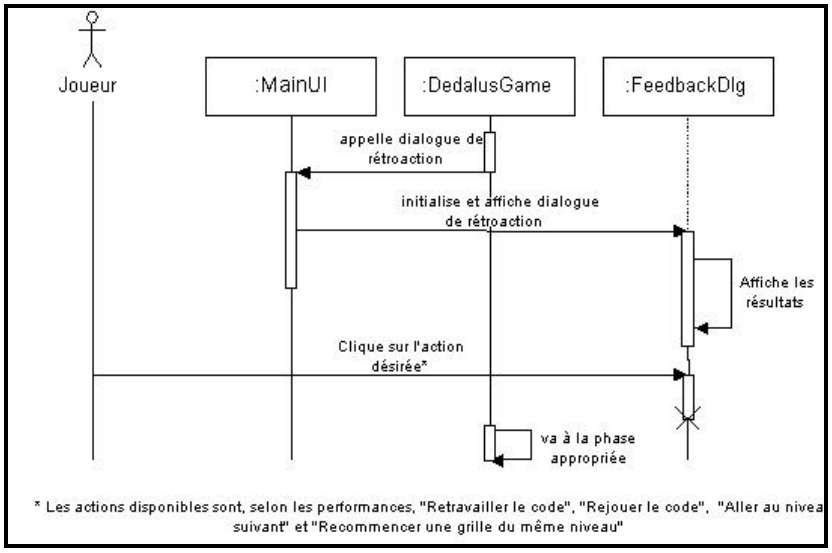

Figure 7 *– Diagramme de séquence pour la phase* rétroaction

2.2. Classes de classification des précisions sur les spécifications

Les classes du jeu Dedalus qui, ensemble, sont suffisantes pour représenter les spécifications sont *DedalusGame*, *GridAnimator*, *Grid, ScriptAnalyser*, *ScriptContainer*, *MainUI*, *IntroDlg*, *NewUserDlg*, *ExistingUserDlg*, *FeedBackDlg*, *TutorialDlg*, *ParametersDlg*.

Les associations entre ces classes sont présentées à la figure 8. Il est à noter qu'aucune hiérarchie n'est présente entre ces classes. Seuls les regroupements de modules sont présentés, ainsi que les associations entre modules.

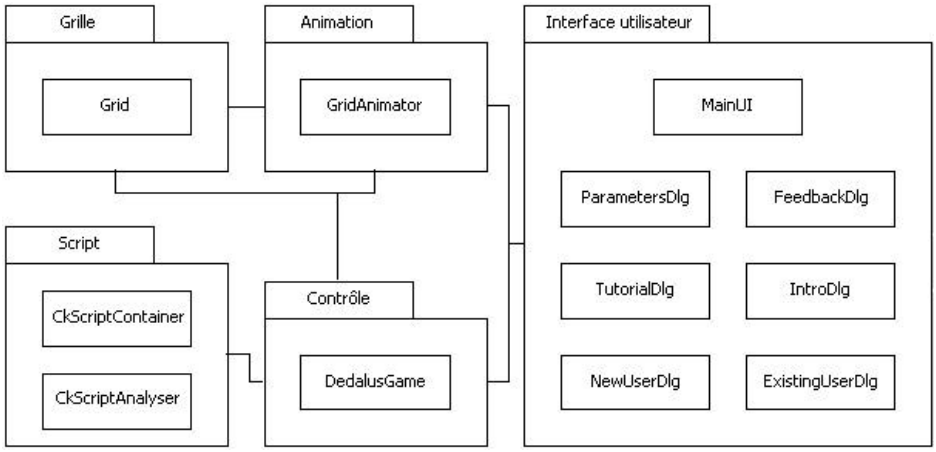

Figure 8 *– Diagramme des associations des classes* (« Object model »)

2.DG. DedalusGame : Module de contrôle

2.DG.1. Attributs

En tout temps, DG contrôle l'évolution du jeu, en gardant la trace des informations sur l'usager, le niveau atteint et la phase en cours.

2.DG.2. Entités

Une seule entité de cette classe est présente en tout temps.

2.DG.3. Fonctionnalités

Au cours du jeu, l'utilisateur atteint des objectifs, donc des niveaux de complexité. DedalusGame contient ce niveau et gère les modules qui lui sont rattachés en prenant ce niveau en considération. Les différents niveaux sont expliqués dans le tableau 1

Niveau	NR Min[1]	Concepts de programmation ajouté	Types de case ajoutés
1	1	Avance, Droite, Gauche	Vide, Mur, gazon, sortie
2	2	Ramasse	Trésor
3	3	Fais X Fois	-
4	2	Ouvre Porte	Porte
5	3	Si [mur]	Quelques cases mystère
6	2	Sinon	
7	2	[objet]	Plus de cases mystère
8	2	[porte]	
9	4	Tant que	
10	2	[sortie], [ai objet]	
Bonus	-		Rempli de cases mystère

Tableau 1 *– Niveaux de complexité du jeu*

[1] NR Min : Nombre minimal de réussites pour passer au niveau suivant

2.GA. GridAnimator : Animateur de la grille

2.GA.1. Attributs

GridAnimator contiendra deux copies de la grille de jeu : une étant l'originale, et l'autre celle sur laquelle il agit. Il connaîtra aussi le script de l'utilisateur.

2.GA.2. Entités

Une seule entité de cette classe est présente en tout temps.

2.GA.3. Fonctionnalités

2.GA.3.1. Reproduction des mouvements du robot *[Essentielle]*

GridAnimator reproduira les mouvements du robots à partir des commandes de script, sur la grille. Il devra réagir aux erreurs (robot essaie de passer au travers d'un mur, etc.)

2.GA.3.2. Interruption *[Désirable]*

GridAnimator devra avoir la possibilité d'être interrompu dans son exécution.

2.GR. Grid : Grille de jeu et générateur

2.GR.1. Attributs

Grid connaît les contraintes du niveau en cours et possède une grille de jeu interne.

2.GR.2. Entités

Une seule entité de cette classe est présente en tout temps.

2.GR.3. Fonctionnalités

2.GR.3.1. Génération de la grille *[Essentielle]*

Grid doit générer une grille de jeu d'après les contraintes de niveau.

2.GR.3.2. Vérification *[Essentielle]*

Grid doit vérifier la faisabilité de la grille de jeu qu'il a construite.

2.GR.3.3. Chemin optimal *[Optionnelle]*

Grid devra calculer le chemin optimal pour le comparer à celui de l'utilisateur.

2.SA. ScriptAnalyser : Analyseur de script

2.SA.1. Attributs

ScriptAnalyser connaît le script composé par l'utilisateur.

2.SA.2. Entités

Une seule entité de cette classe est présente en tout temps.

2.SA.3. Fonctionnalités

2.SA.3.1. Optimisations *[Optionnelle]*

ScriptAnalyser cherche dans le script les optimisations possibles pour en présenter un rapport à l'utilisateur.

2.SA.3.2. Erreurs de codage *[Optionnelle]*

ScriptAnalyser détecte les erreurs de codage (boucles infinies, incohérences)

2.SC. ScriptContainer : Conteneur de script

2.SC.1. Attributs

ScriptContainer connaît en tout temps le script entré par l'utilisateur.

2.SC.2. Entités

Une seule entité de cette classe est présente en tout temps.

2.SC.3. Fonctionnalités

2.SC.3.1. Organisation *[Essentielle]*

ScriptContainer est responsable de l'organisation et de la mise en page du script, ainsi que de l'insertion et de l'effacement de lignes.

2.SC.3.2. Mise à jour

Lorsqu'une commande est insérée ou enlevée, ScriptContainer doit mettre à jour MainUI pour que l'utilisateur voie son script.

2.SC.3.3. Sauvegarde *[Désirable]*

ScriptContainer est responsable de l'enregistrement du script sur disque ainsi que de son chargement.

2.ED. ExistingUserDlg : Dialogue de sélection utilisateur

2.ED.1. Attributs

ExistingUserDlg possède une liste déroulante contenant les noms des joueurs enregistrés sur la machine. Le nom du dernier utilisateur est sélectionné par défaut. Un bouton OK et un bouton Annuler sont aussi présents. Une version future pourra demander un mot de passe pour la sélection d'un utilisateur, afin que chacun joue seulement sur ses propres données.

2.ED.2. Entités

Une seule entité de cette classe existe.

2.ED.3. Événements

2.ED.3.1. OK *[Essentielle]*

Presser sur *OK* ferme ExistingUserDlg et IntroDlg. DedalusGame passe alors à la phase suivante.

2.ED.3.2. Annuler *[Essentielle]*

Presser sur *Annuler* ferme ExistingUserDlg et retourne à IntroDlg.

2.FD. FeedbackDlg : Dialogue de rétroaction

2.FD.1. Attributs

FeedbackDlg possède une étiquette texte contenant le texte de rétroaction, une image où sera affichée une médaille de réussite, le robot fatigué ou brisé selon la réussite du niveau. Il contient aussi les boutons *Retravailler le code*, *Rejouer le code, Aller au niveau suivant* et *Recommencer une grille du même niveau.*

2.FD.2. Entités

Une seule entité de cette classe existe.

2.FD.3. Événements

2.FD.3.1. Retravailler le code *[Essentielle]*

Cliquer sur *Retravailler* ferme FeedbackDlg et retourne à la phase *Construction*.

2.FD.3.2. Rejouer le code *[Essentielle]*

Cliquer sur *Rejouer* ferme FeedbackDlg et va à la phase *Exécution*.

2.FD.3.3. Recommencer une grille du même niveau *[Essentielle]*

Cliquer sur *Recommencer* ferme FeedbackDlg et va à la phase *Génération*.

2.FD.3.3. Aller au niveau suivant *[Essentielle]*

Cliquer sur *Niveau Suivant* ferme FeedbackDlg, augmente le niveau et va à la phase *Génération*. Ce bouton n'est disponible que si l'utilisateur a complété le niveau.

2.ID. IntroDlg : Dialogue de bienvenue

2.ID.1. Attributs

IntroDlg affiche une image du logo Dedalus, ainsi que deux boutons : *Nouveau joueur* et *Ancien joueur*.

2.ID.2. Entités

Une seule entité de cette classe existe.

2.ID.3. Événements

2.ID.3.1. Nouveau joueur *[Essentielle]*

En cliquant sur *Nouveau joueur*, NewUserDlg apparaît.

2.ID.3.2. Ancien joueur *[Essentielle]*

En cliquant sur *Ancien joueur*, ExistingUserDlg apparaît.

2.ND. NewUserDlg : Dialogue nouvel utilisateur

2.ND.1. Attributs

NewUserDlg possède une boite de texte initialement vide, un bouton *OK* ainsi qu'un bouton *Annuler*.

2.ND.2. Entités

Une seule entité de cette classe existe.

2.ND.3. Événements

2.ND.3.1. OK *[Essentielle]*

Presser sur OK vérifie si la boîte de texte contient un nom et s'il n'existe pas déjà. Si tout est ainsi, ferme NewUserDlg et IntroDlg; DedalusGame passe à la phase suivante. Sinon, affiche un message d'erreur et retourne à NewUserDlg

2.ND.3.2. Annuler *[Essentielle]*

Presser sur Annuler ferme NewUserDlg et revient à IntroDlg.

2.PD. ParametersDlg : Dialogue spécification des paramètres

ParametersDlg s'affiche lors de l'insertion dans le script d'une commande qui nécessite un ou plusieurs paramètres : le *Si [x]*, le *Tant que [x]* et le *Fais [x] fois*. ParametersDlg n'a pas été inclus dans les diagrammes de séquence.

2.PD.1. Attributs

ParametersDlg contient une liste des paramètres disponibles pour le *Si* et le *Tant que*, une case à cocher pour ajouter un *Sinon* au *Si*, une boîte de texte et un bouton incrément (*SpinButton*) pour le *Fais x fois.*

Il contient aussi les bouton de commande *OK* et *Annuler*.

2.PD.2. Entités

Trois entités de cette classe existent.

2.PD.2.1. Si *[Essentielle]*

Cette entité contient la liste, la case à cocher et les boutons de commande.

2.PD.2.2. Tant que *[Essentielle]*

Cette entité contient la liste et les boutons de commande.

2.PD.2.3. Fais x fois *[Essentielle]*

Cette entité contient la boîte de texte, le bouton incrément et les boutons de commande.

2.PD.3. Événements

2.PD.3.1. Sélection d'une condition

Sélectionner une condition dans la liste met celle-ci en surbrillance.

2.PD.3.2. Bouton incrément

Cliquer sur les flèches du bouton incrément change la valeur dans la boite de texte. Cette valeur peut aller de 1 à 100.

2.PD.3.3 OK *[Essentielle]*

Cliquer sur *OK* ajoute la commande et ses paramètres dans le ScriptContainer, ferme ParametersDlg et retourne à MainUI.

2.PD.3.4 Annuler *[Essentielle]*

Cliquer sur *Annuler* ferme ParametersDlg et retourne à MainUI.

2.TD. TutorialDlg : Dialogue des instructions

2.TD.1. Attributs

TutorialDlg affiche le titre du concept de programmation et le niveau auquel il est associé. Une zone de texte présente l'explication de cette nouvelle fonctionnalité. Une autre présente un exemple de script l'utilisant. Une image affiche l'effet de ce script dans une grille de jeu miniature. TutorialDlg possède aussi les bouton *OK*, *Précédent*, *Suivant*, *Animer*.

2.TD.2. Entités

Une seule entité de cette classe existe.

2.TD.3. Événements

2.TD.3.1. OK *[Essentielle]*

Presser sur *OK* ferme TutorialDlg et va ou revient à la phase de construction.

2.TD.3.2. Précédent / Suivant *[Désirable]*

Les boutons *Précédent* et *Suivant* font passer au concept de programmation précédent ou suivant dans la liste des niveaux. *Suivant* sera désactivé si le concept présenté correspond au niveau en cours. *Précédent* sera désactivé si le concept présenté correspond au premier niveau.

2.TD.3.3. Animer *[Désirable]*

Le bouton Animer fait exécuter le script exemple dans la grille miniature.

2.MU. MainUI : Interface utilisateur principale

2.MU.1. Attributs

MainUI affiche la grille de jeu, le script sous forme de liste, le boutons de commandes de script, les boutons de contrôle du script (effacer ligne, monter ligne, descendre ligne, effacer tout, éditer paramètres, enregistrer, ouvrir), les boutons de contrôle du jeu (quitter, niveau suivant, générer une autre grille, changer de joueur, afficher les instructions), ainsi que des étiquettes affichant l'état du joueur (nom, niveau atteint, pointage).

2.MU.2. Entités

Une seule entité de cette classe existe.

2.MU.3. Événements

2.MU.3.1. Commandes de script *[Essentielle]*

Appuyer sur une commande de script affiche ParametersDlg si besoin est, sinon ajoute la commande dans ScriptContainer.

2.MU.3.2. Contrôle du script

Appuyer sur une commande de contrôle du script envoie l'instruction correspondante à ScriptContainer, pour *Effacer* *[Essentielle]*, *Monter* *[Essentielle]*, *Descendre* *[Essentielle]*, *Enregistrer* *[Désirable]*, *Ouvrir* *[Désirable]*. La commande *Paramètres* *[Essentielle]* affiche ParametersDlg pour l'instruction sélectionnée. Cette commande n'est disponible que lorsqu'une instruction telle que *Si*, *Tant que* ou *Fais x fois* est en surbrillance.

2.MU.3.3. Boutons de contrôle du jeu

La commande *Quitter* *[Essentielle]* affiche un message de confirmation puis ferme le jeu.

La commande *Niveau suivant* *[désirable]* affiche un message de confirmation puis passe au niveau suivant et à la phase *Génération*. Cette commande n'est disponible que si l'utilisateur a réussi un niveau mais a décidé de revenir travailler son code ou travailler une nouvelle grille.

La commande *Générer une autre grille* *[désirable]* affiche un message de confirmation puis passe à la phase *Génération*. Cette commande n'est disponible que si l'utilisateur a déjà choisi de revenir travailler son code après une exécution.

La commande *Changer de joueur* *[désirable]* affiche un message de confirmation puis passe à la phase *Initialisation*.

La commande *Afficher les instructions* *[désirable]* affiche TutorialDlg pour le niveau en cours.

2.MU.4 Fonctionnalités

2.MU.4.1 Gestion de l'interface *[Essentielle]*

MainUI est responsable de gérer la disponibilité des boutons en fonction du niveau et des interactions avec l'utilisateur. Il doit aussi rafraîchir son interface et la grille de jeu au besoin et sur demande.

3. Spécifications de performance

Le logiciel devra se charger en mémoire et afficher l'écran de bienvenue en moins de 20 secondes sur un ordinateur respectant les exigences minimales. La génération de la grille devra prendre moins de 5 secondes.

4. Contraintes de design

Dedalus devra être implémenté en C++. Toutes les parties non IU du logiciel devront être codées de façon portable entre différentes plate-formes. Les interfaces utilisateur de la première version et des versions Windows subséquentes seront codées en C++ en utilisant les classes MFC (Microsoft Foundation Classes).

L'implantation et le design des classes devra être fait de façon modulaire de façon à ce que chaque partie puisse fonctionner séparément et que tout composant soit interchangeable. Cette façon de procéder favorisera la portabilité du code non IU.

5. Attributs systèmes du logiciel

5.1. Fiabilité

Dedalus ne devrait pas perdre sa stabilité plus d'une fois par 500 utilisations.

5.2. Compatibilité

Dedalus devra pouvoir être joué sur un PC opérant sous Windows 95 ou version subséquente seulement (aucune autre applications dans la mémoire), ayant au minimum 16 Mo de mémoire, une vitesse d'horloge de 133 MHz et une résolution de 800x600x256.

5.3. Sécurité

Une version subséquente demandera un mot de passe pour charger les données d'un utilisateur.

5.4. Maintenabilité

5.4.1. Génération des grilles de jeu

Il devra être très facile de changer le générateur de grille de jeu pour un autre ou d'en modifier ses paramètres, afin de pouvoir générer des grilles d'un niveau non prévu à ce jour

5.4.2. Ajout de commandes / concepts de programmation

Il devra être possible d'ajouter ou de retirer des commandes de script afin de changer les concepts de programmation à apprendre à l'usager. Ainsi, il pourra être un jour possible d'inclure les concepts d'objets et de classes.

5.4.3. Aspect visuel

L'aspect visuel de l'application devra pouvoir être modifié rapidement.

5.4.4. Analyse des scripts

Il devra être possible de modifier l'analyseur de script afin d'obtenir d'autres niveaux d'analyse, tels que des suggestions ou modifications à apporter pour rendre le script plus performant, plus lisible, etc..

6. Autres spécifications

Aucune autre spécification.