SPÉCIFICATION INTERNE DU LOGICIEL ARCHITECTURE

TRAVAIL PRÉSENTÉ À
MME SOUMAYA CHERKAOUI

DANS LE CADRE DU COURS
PROJET DE CONCEPTION DE LOGICIELS
GEI450

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

LE 20 JUIN 2001
Université de Sherbrooke

SPÉCIFICATION INTERNE DU LOGICIEL, ARCHITECTURE

Table des matières

TABLE DES FIGURES

1. Introduction

1.1. But

Ce document décrit l'architecture du logiciel Dedalus.

1.2. Portée

Ce design concerne la version qui sera remise à la fin du cours *Projet de conception de logiciel*. Cette architecture est conçue de façon à faciliter la réalisation de versions améliorées dans le futur.

1.3. Définitions

Aucune

2. Références

[BRAUDE] A.J. Braude, Software Engineering, An Object-Oriented Perspective. Wiley, 2001.

3. Description par décomposition

L'architecture du programme Dedalus est décrite à l'aide du modèle d'état, du modèle des classes et d'un diagramme de flux de donnée. L'ensemble de l'interface graphique utilisateur utilise le cadre d'application MFC de Microsoft.

3.1. Décomposition par modules

Le programme Dedalus contient deux ensembles de classes : les interfaces graphiques utilisateurs et le noyau du jeu. Ces deux ensembles communiquent entre eux via une interface spécifique décrit à la section 5 de ce document. Les classes internes à ces ensembles sont uniquement accessibles via les interfaces définies.

3.1.1. L'ensemble des interfaces graphiques utilisateurs

Cet ensemble contient les classes nécessaires à la communication entre l'usager et le programme Dedalus. Ces classes gèrent les interactions avec l'usager.

3.1.2. L'ensemble du noyau du jeu

Cet ensemble contient les classes qui sont nécessaires pour contrôler la dynamique du jeu et les données relatives au logiciel. Ces classes gèrent chacune des étapes du jeu Dedalus. Cet ensemble est divisé en trois sous-modules :

3.1.2.1. Contrôleur

Ce sous-module gère la progression du jeu ainsi que l'interaction entre les autres modules.

3.1.2.2. Grille

Ce sous-module contient toute l'information en rapport avec le contenu de la grille de jeu, ainsi que son traitement et sa génération.

3.1.2.3 Script

Ce sous-module fait l'entreposage du script du joueur et son analyse au point de vue efficacité.

3.2. Décomposition par processus concurrent

Cette section est non pertinente car le programme Dedalus comporte qu'un seul fil d'exécution.

3.3. Décomposition par données

Il y a cinq types de données principales qui circulent à l'intérieur du programme Dedalus : CkScriptContainer, CkGrid, CkGridArray , CkPlayerInfo et les étiquettes de commande. On retrouve à la **Figure 1** le diagramme de flux de données qui caractérise le mouvement de ces données**.**

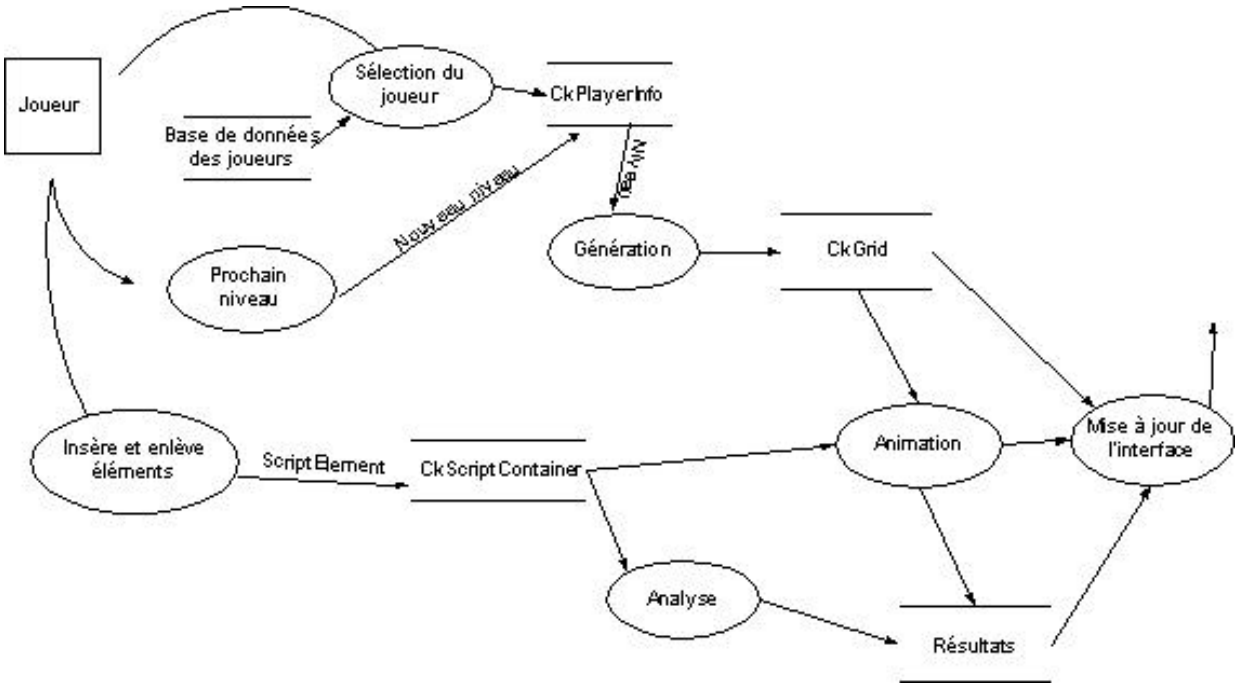

Figure 1 Diagramme de flux de données

3.3.1. CkScriptContainer

CkScriptContainer contient les éléments du script que l'usager a créé pour permettre au robot de sortir du labyrinthe. Il est interne à l'ensemble du noyau du jeu. Autrement dit, uniquement les classes composant l'ensemble du noyau du jeu s'échangent des objets CkScriptContainer.

3.3.2. CkGrid

CkGrid représente la grille du jeu. Il contient toute l'information relative à la grille du jeu, c'est à dire, le labyrinthe dans lequel le robot évolue. Il est interne à l'ensemble du noyau du jeu. Autrement dit, uniquement les classes composant l'ensemble du noyau du jeu s'échangent des objets CkGrid.

3.3.3. CkGridArray

CkGridArray est un tableau représentant la grille de jeu. Cet objet évoluera à l'intérieur du noyau du jeu et du noyau du jeu vers l'interface utilisateur.

3.3.4. CkPlayerInfo

CkPlayerInfo contient toute l'information relative à l'utilisateur : nom, niveau, etc. Les objets de type CkGrid évolueront du noyau du jeu vers l'interface utilisateur et vice versa.

3.3.5. Étiquette de commande

Cette liste contient la version par clef numérique du script. C'est à dire, chacune des commandes sera représentée par une valeur numérique associée. Cette liste évoluera à l'intérieur du noyau du jeu et du noyau du jeu vers l'interface utilisateur.

3.4. Décomposition

Le programme Dedalus peut être découpé selon les états de la **Figure 2.**

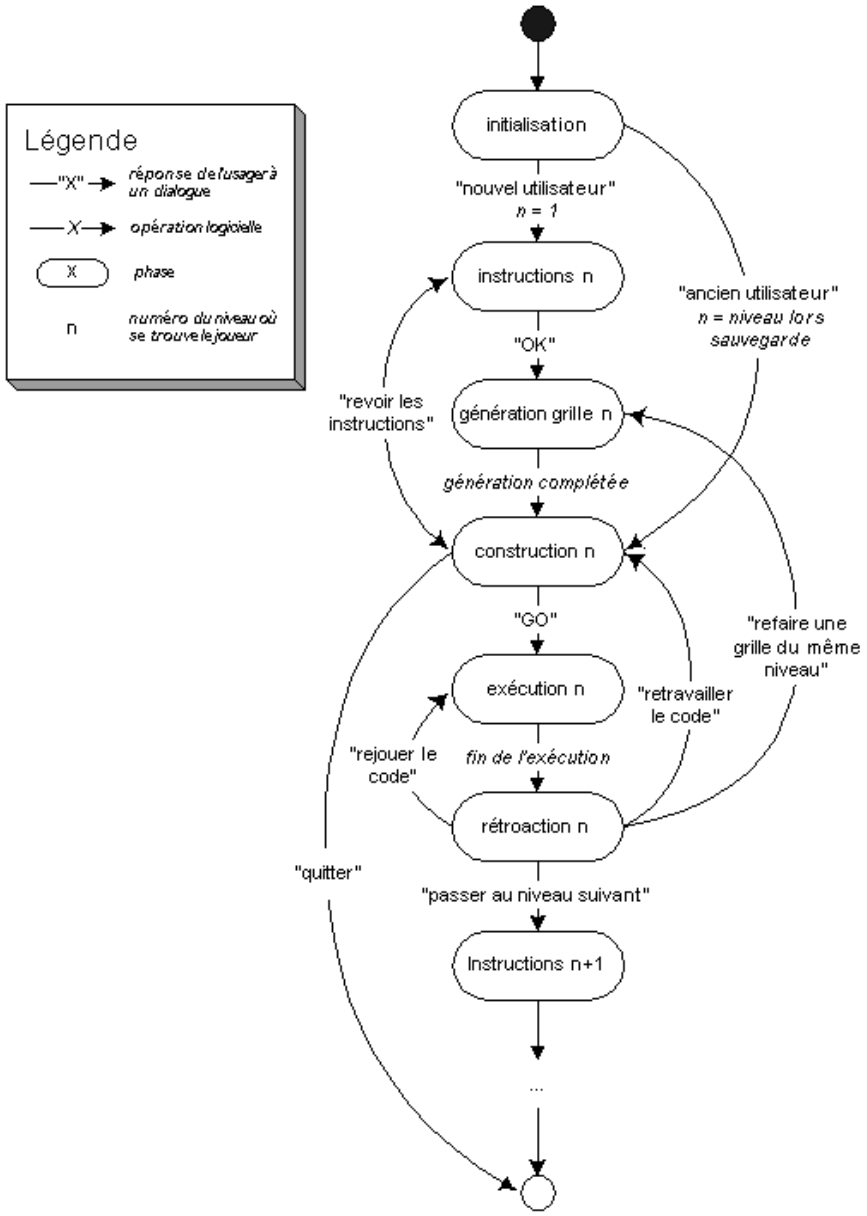

Figure 2 Diagramme d'états

4. Description par dépendance

Cette section décrit les dépendances pour chacune des décompositions de la section 3.

4.1. Dépendances entre modules

Les dépendances entre les modules sont montrées à la **Figure 3**. L'ensemble du noyau du jeu dépend de l'ensemble de l'interface graphique utilisateur pour les actions faites par l'usager et l'ensemble de l'interface graphique utilisateur dépend du noyau du jeu car c'est ce dernier qui contrôle l'état du jeu.

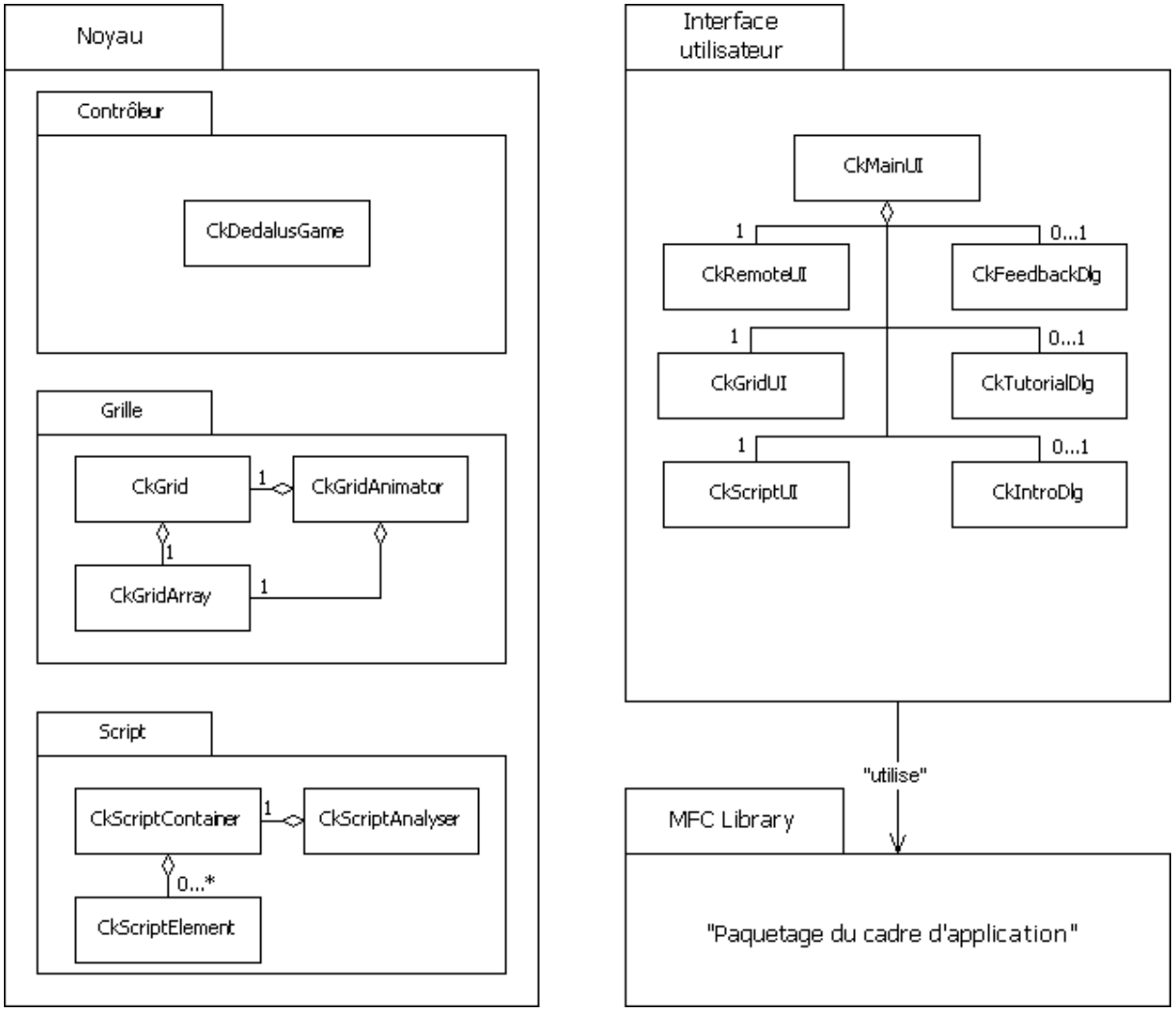

Figure 3 Architecture de Dedalus

4.2. Dépendances interprocessus

Le programme Dedalus ne comporte qu'un seul fil d'exécution. Par conséquent, il n'y a pas de dépendances interprocessus.

4.3. Dépendances des données

Les données qui circulent à l'intérieur et entre les deux ensembles principaux, sont définies à l'aide des classes qui se retrouvent à la section 6 de ce document.

4.4. Dépendances des états

L' état dans lequel se trouve le programme Dedalus dépend de l'état précédent comme démontré à la figure 1 de ce document.

5. Description des interfaces

Cette section décrit les interfaces du programme Dedalus.

5.1. Interface par module

Cette section contient les différentes interfaces divisées par ensemble et par les modules contenus dans ses ensembles.

5.1.1. Interface de l'ensemble de l'interface graphique utilisateur

L'interface de l'ensemble de l'interface graphique utilisateur est fournie par la classe MainUI. Cette interface est visible par tous les objets de l'ensemble du noyau du jeu. Elle est contituée des méthodes suivantes :

1. `EnableAdminControl(CkAdminControlId, bool)`
   // mettre en opération ou non les contrôles du jeu
2. `EnableRemoteControl(CkRemoteId, bool)`
   // mettre en opération ou non la télécommande du robot
3. `InitUI()`
   // initiliser l'interface graphique utilisateur
4. `SetGridBackground(CkGridArray)`
   // mettre à jour l'arrière plan de la grille de jeu à l'écran
5. `SetGridObject(CkObjectId, CkVector)`
   // positionner un objet sur la grille
6. `SetPlayerInfo(CkUserInfo)`
   // mettre à jour les informations du joueur
7. `SetWaitingMode(bool, CkPourcentage)`
   // mettre on enlever le mode attente
8. `ShowFeedback()`
   // afficher la fenêtre de rétroaction
9. `ShowInstruction()`
   // afficher la fenêtre d'instruction
10. `ShowIntroduction()`
    // afficher la fenêtre d'introduction
11. `ShowTutorial()`
    // afficher la fenêtre du tutoriel
12. `UpdateScript(CkScriptContainer)`
    // mettre à jour le script à l'écran
13. `MoveRobot(CkVector, CkVector)`
    // déplacer le robot sur le grille

5.1.2. Interface du module contrôleur

L'interface du module du jeu est fournie par la classe DedalusGame. Cette interface est visible par tous les objets de l'ensemble de l'interface graphique utilisateur. Elle est contituée des méthodes suivantes :

1. `UserEvent(CkAdminControlId)`
   // signaler un événement général de l'usager

2. `RemoteEvent(CkScriptElement)`
   // signaler un événement provenant de la télécommande
3. `InitGame()`
   // initialiser la partie
4. `CkPlayerList GetPlayerList()`
   // acquérir les liste de tous les joueurs

5.1.3. Interface du module Script

L'interface du module Script est accessible uniquement aux objets à l'intérieur de l'ensemble du noyau du jeu. Cette inteface est fournie par la classe CkScriptAnalyser et CkScriptContainer. Elle est constituée de la méthode suivante :

1. `CkResult Analyse(CkScript, Cklevel)`
   // analyser le script à exécuter
2. `SetPosition(long)`
   // placer le curseur courant à l'intérieur du contenant
3. `CkElement GetCurrentElement()`
   // acquérir l'élément courant du CkScriptContainer
4. `CkElement GetNextElement()`
   // acquérir l'élément suivant du CkScriptContainer
5. `Next()`
   // placer le curseur courant sur l'élément suivant
6. `Remove()`
   // enlever l'élément courant du CkScriptContainer
7. `RemoveAll()`
   // enlever tous les éléments du CkScriptContainer
8. `Insert(CkScriptElement)`
   // insérer un nouvel élément

5.1.4. Interface du module Grille

L'interface du module Grille est accessible uniquement aux objets à l'intérieur de l'ensemble du noyau du jeu. Cette inteface est fournie par la classe CkGridAnimator et CkGrid. Elle est constituée des méthodes suivantes :

1. `SetGrid(CkGrid)`
   // mettre à jour l'objet CkGrid du module CkGridAnimator
2. `StartAnimation(bool*)`
   // démarrer ou arrêter l'animation de la grille
3. `Generate(CkLevel)`
   // générer une nouvelle grille d'un niveau spécifique
4. `CkGridArray GetGrid()`
   // acquérir l'objet CkGridArray de CkGrid

5.2. Interface par processus

Dedalus étant composé d'un seul fil d'exécution, il n'y a pas d'interface relatif aux processus.