SPÉCIFICATION INTERNE DU LOGICIEL 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 20 JUIN 2001
Université de Sherbrooke

SPÉCIFICATION INTERNE DU LOGICIEL, CONCEPTION DÉTAILLÉE
Table des matières

TABLE DES FIGURES

6. Design détaillé

6.1. Design détaillé par module

Le diagramme détaillé de l'architecture du logiciel Dedalus est présenté ici, ainsi que les diagrammes de séquences pour chacune des étapes du jeu. Nous nous référerons à ces figures tout au long du document.

La **Figure 1** présente l'architecture de Dedalus avec chacun de ses modules. Ceux-ci sont regroupés dans le paquetage dont ils font partie.

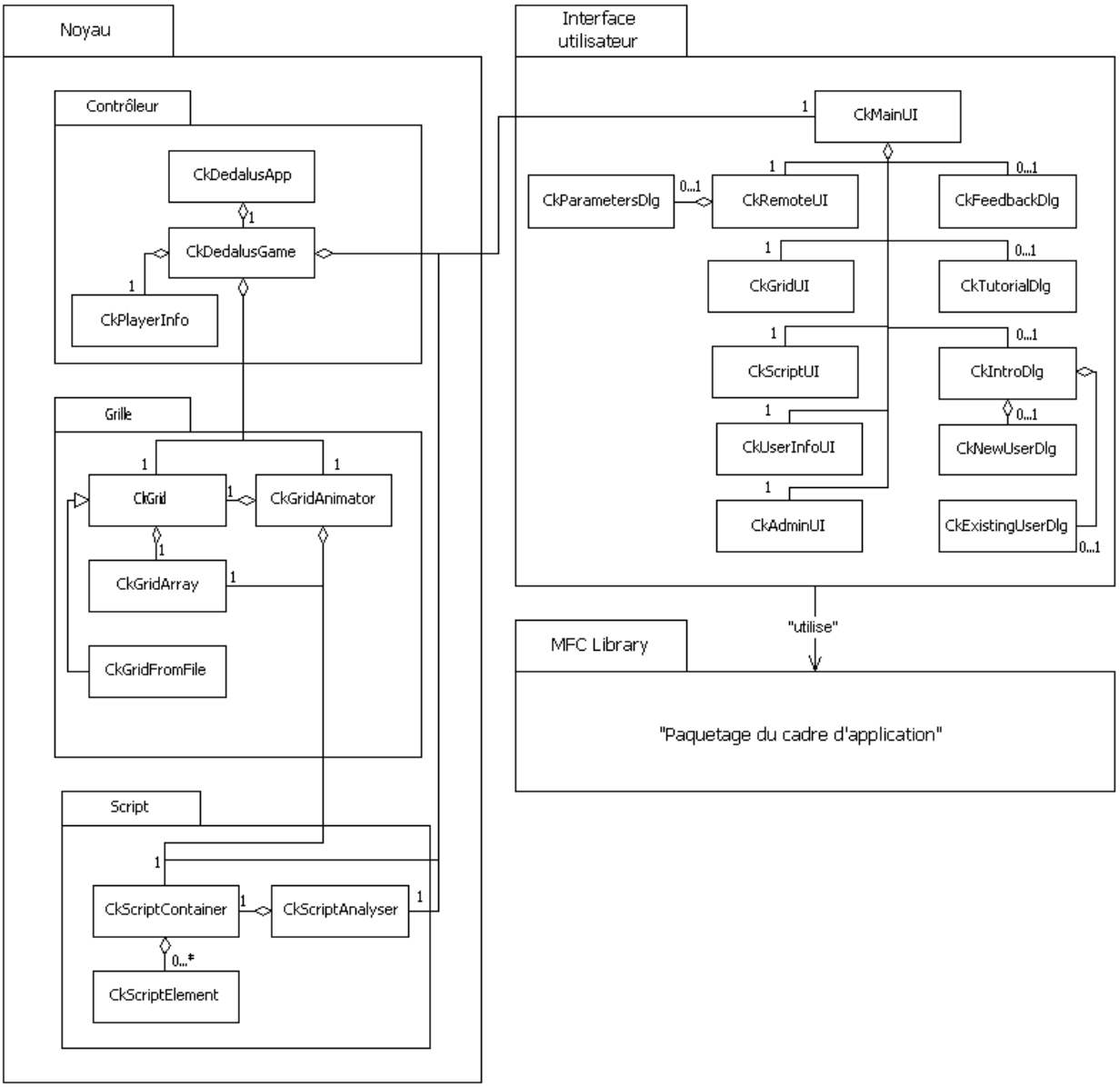

Figure 1 Diagramme de l'architecture par module de Dedalus

La **Figure 2** décrit la séquence selon laquelle les objets des différentes classes de l'ensemble des interfaces graphiques utilisateurs sont reliées lors de l'entrée d'un nouvel utilisateur.

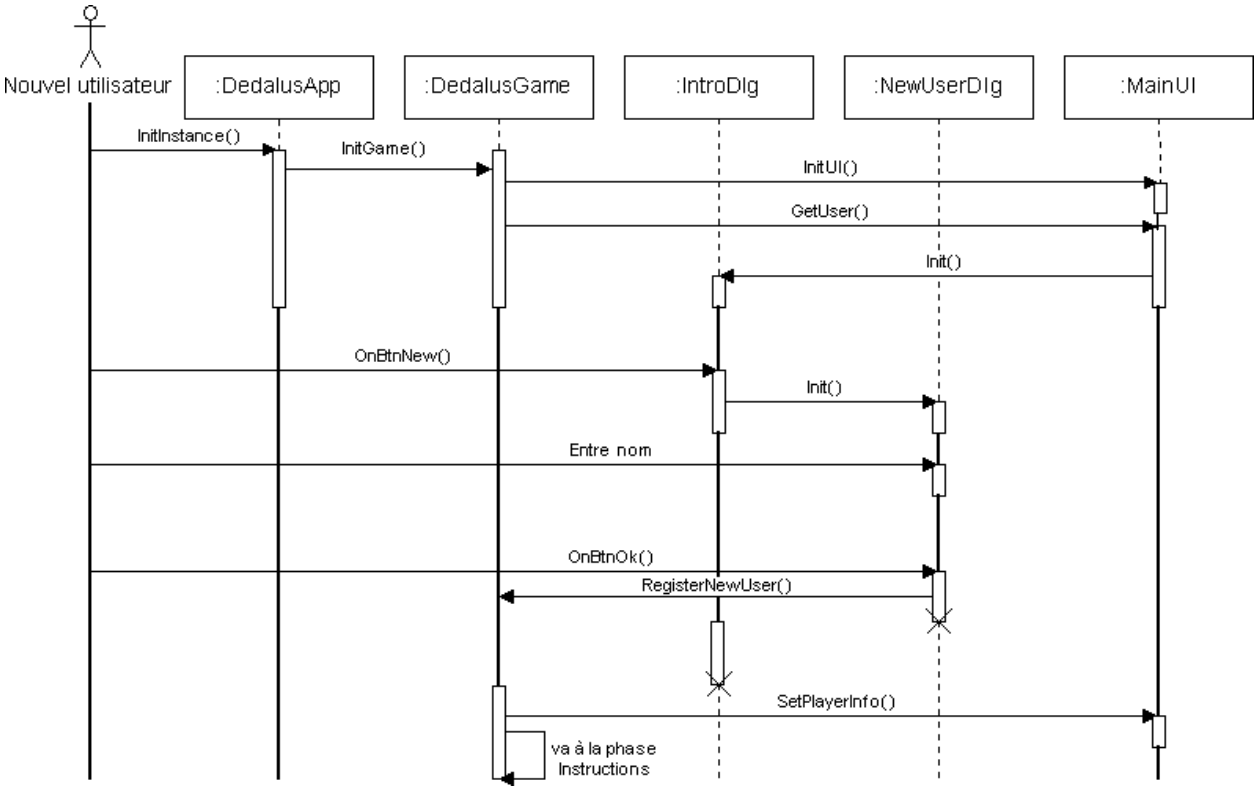

Figure 2 Diagramme de séquence, étape initialisation avec un nouveau joueur

La **Figure 3** décrit la séquence selon laquelle les objets des différentes classes de l'ensemble des interfaces graphiques utilisateurs sont reliées lors de l'entrée d'un ancien utilisateur.

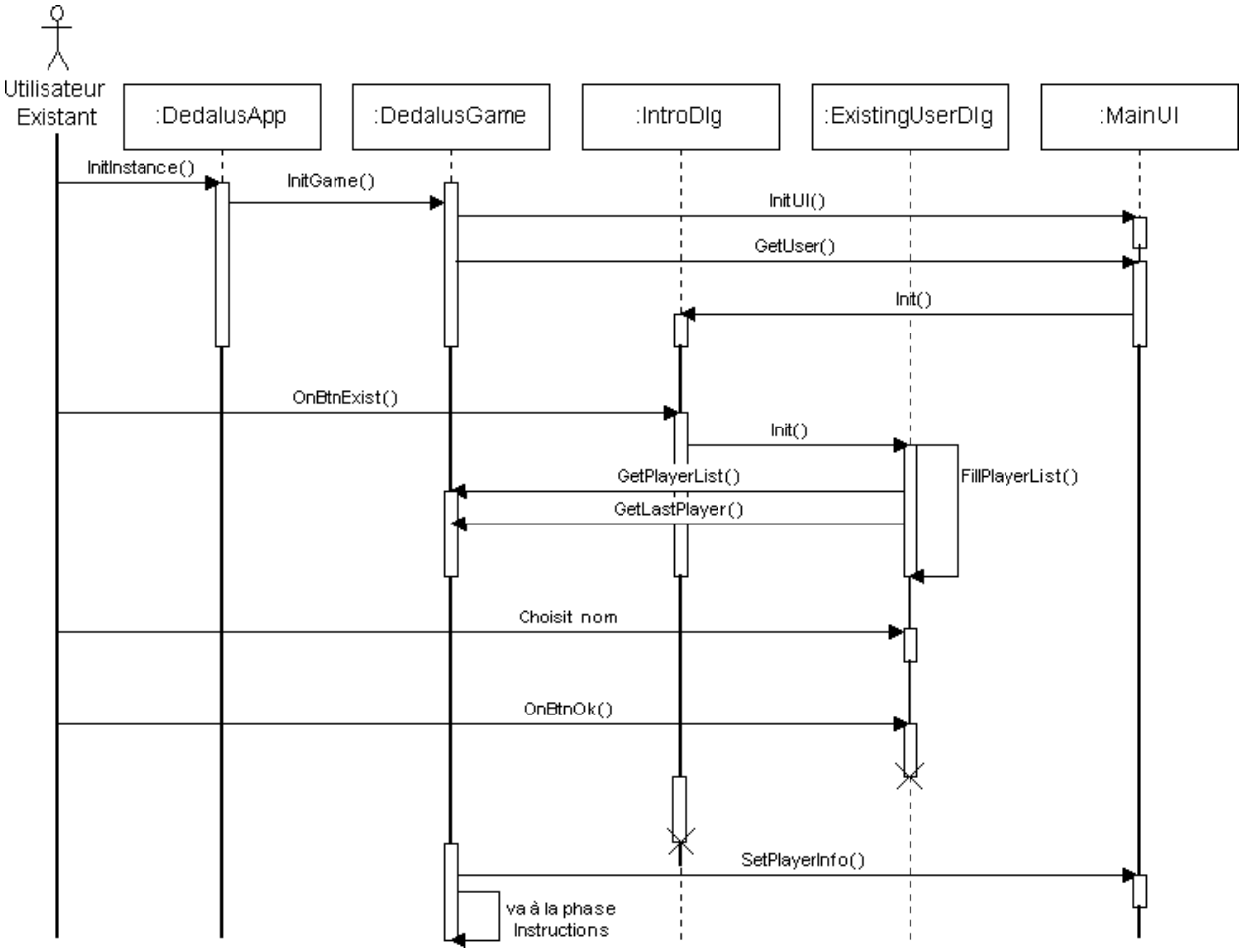

Figure 3 Diagramme de séquence, étape initialisation avec un joueur existant

La Figure 4 décrit la séquence des opérations lors de l'étape des instructions.

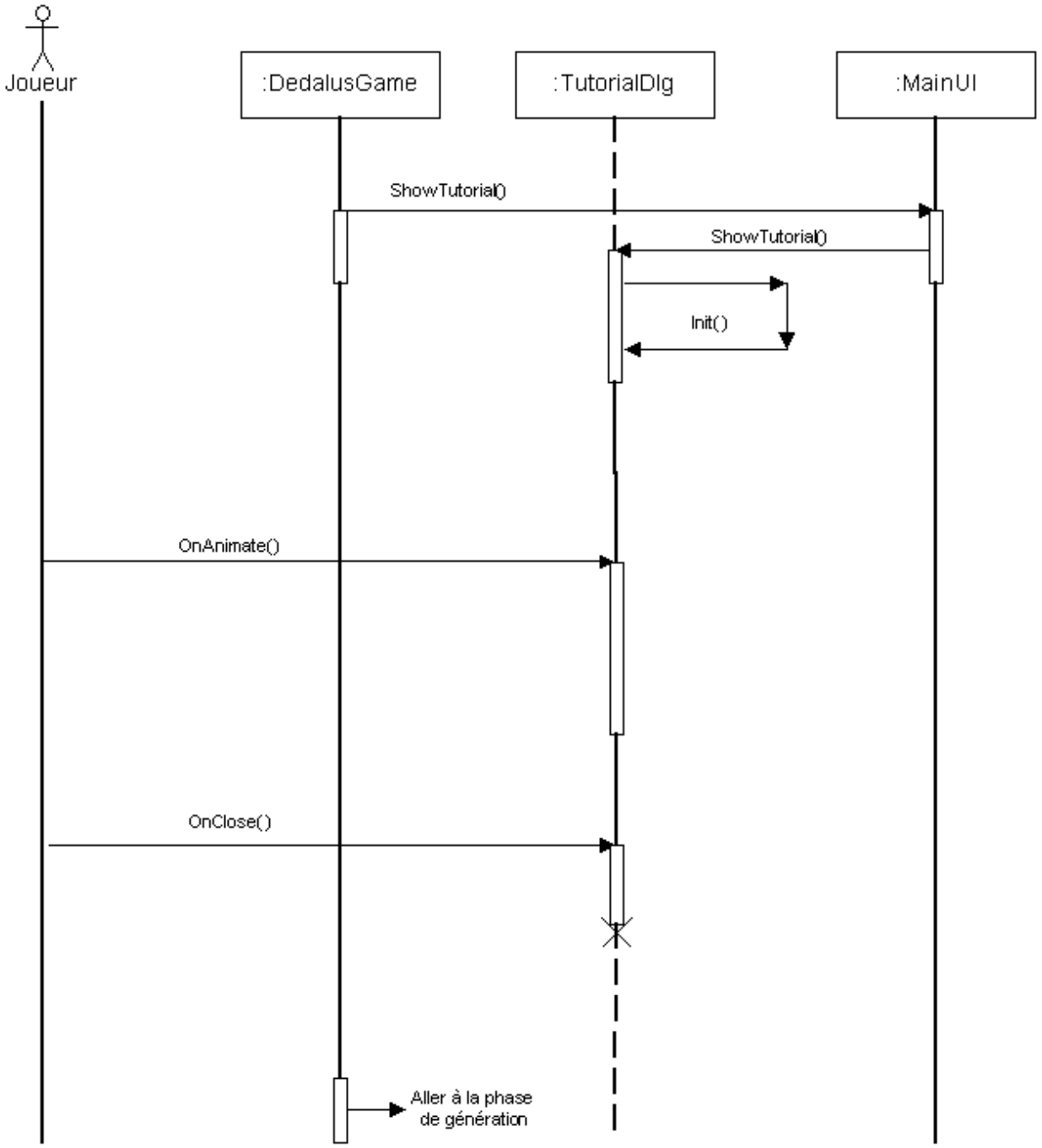

Figure 4 Diagramme de séquence, étape instructions

La **Figure 5** décrit la séquence des opérations lors de l'étape de la génération.

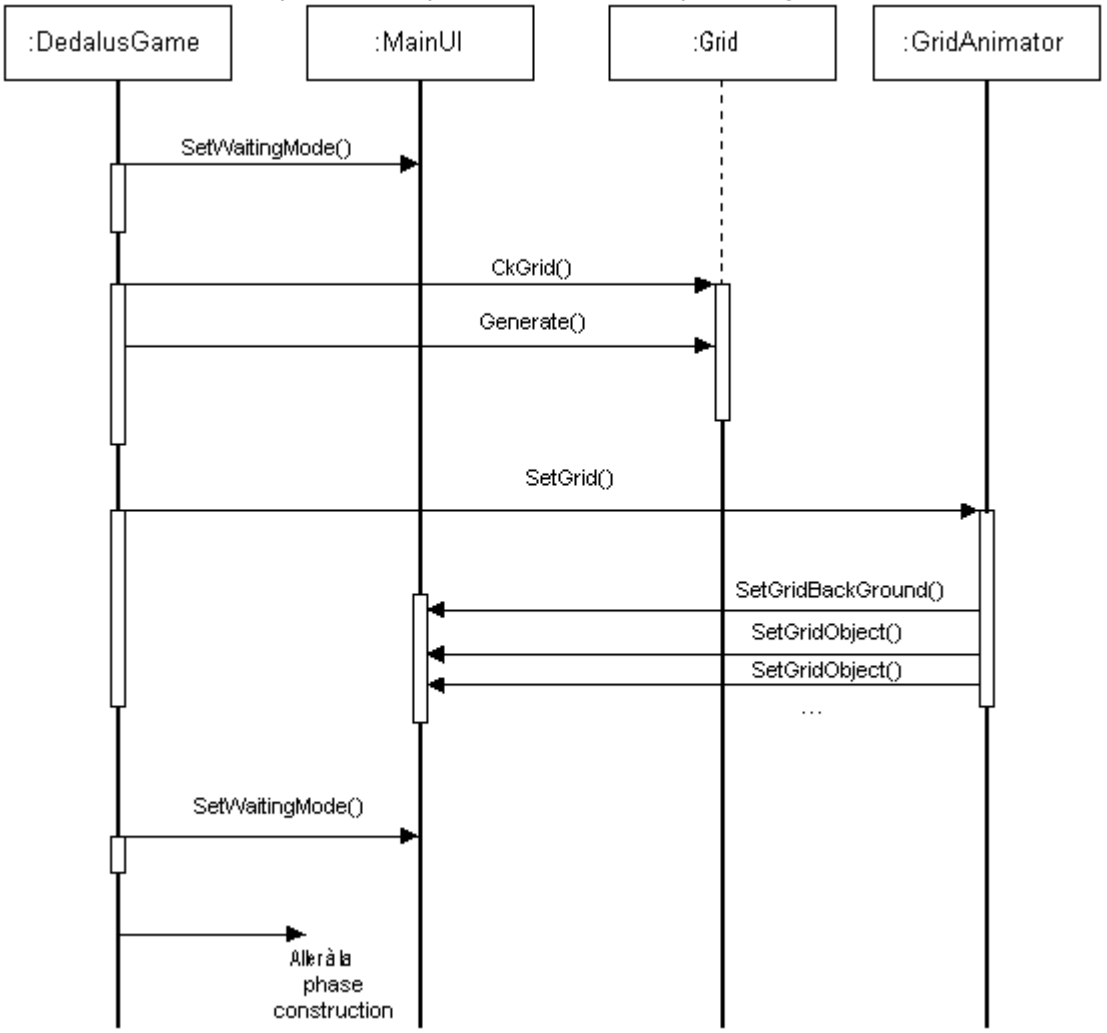

Figure 5 Diagramme de séquence, étape génération

La **Figure 6** décrit la séquence des opérations lors de l'étape de la construction.

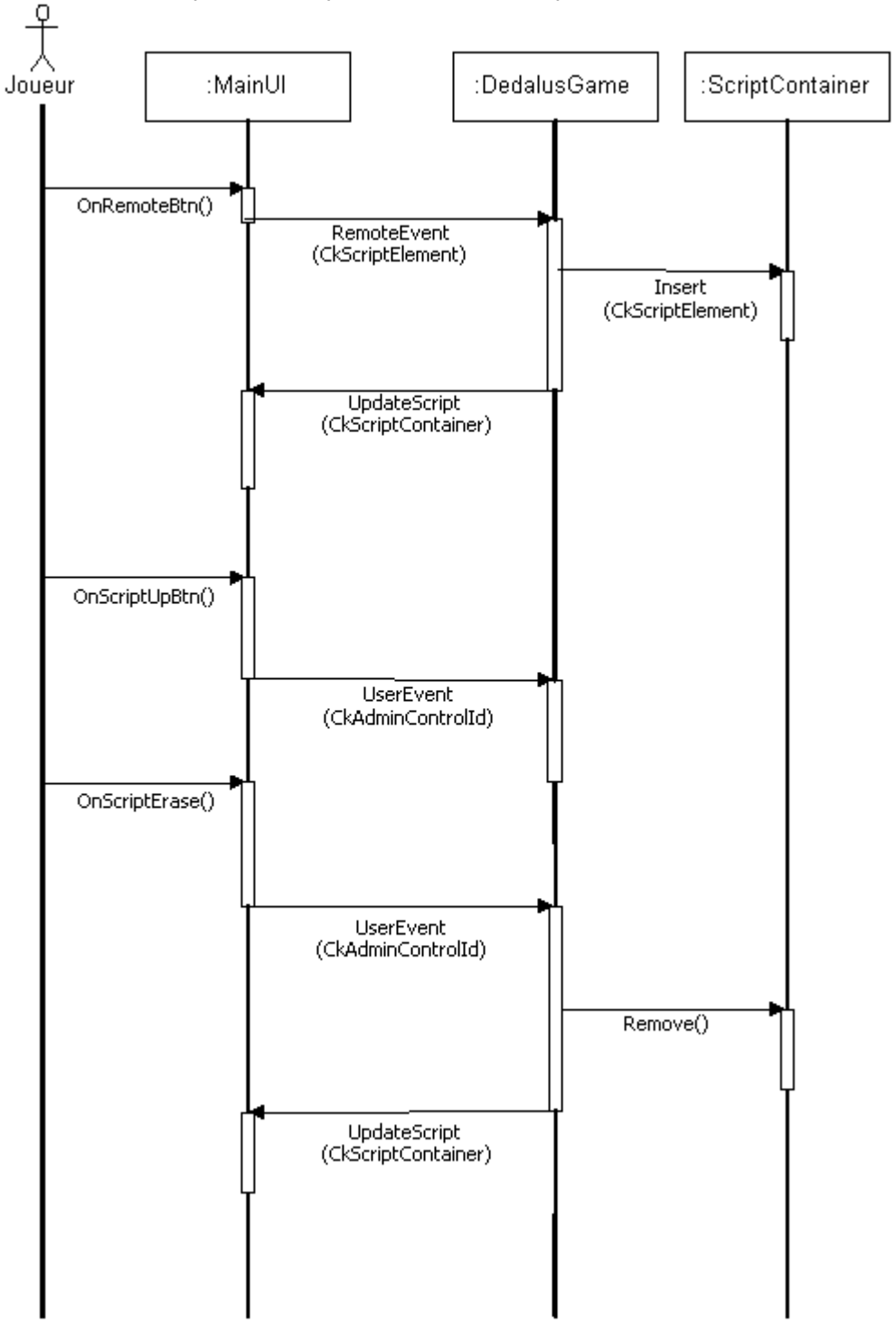

Figure 6 Diagramme de séquence, étape construction

La **Figure 7** décrit la séquence des opérations lors de l'étape de l'exécution.

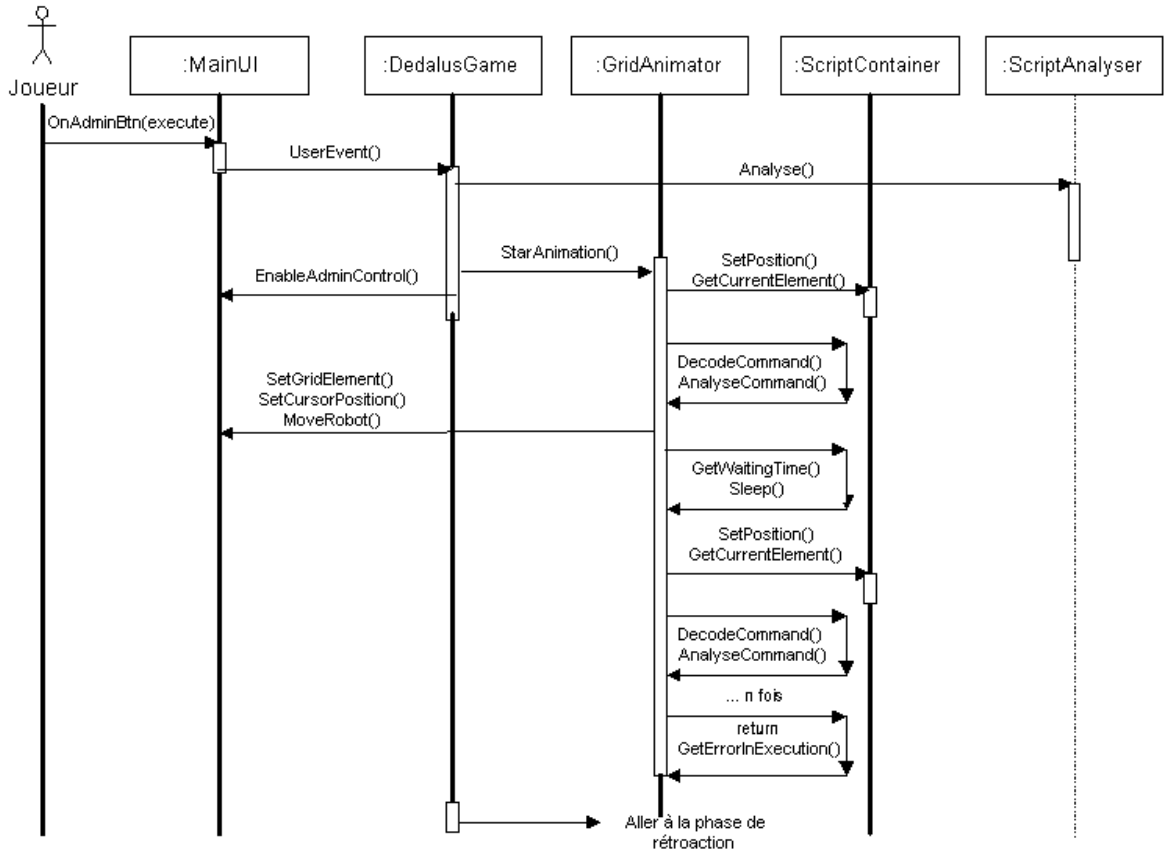

Figure 7 Diagramme de séquence, étape exécution

La **Figure 8** décrit la séquence des opérations lors de l'étape de la rétroaction.

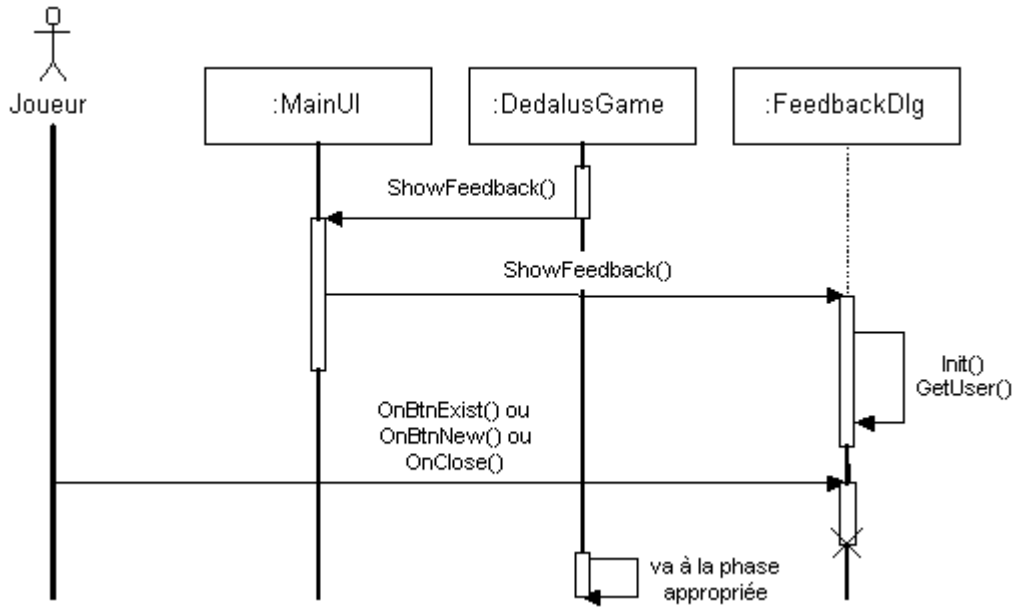

Figure 8 Diagramme de séquence, étape rétroaction

6.1.1. L'ensemble des interfaces graphiques utilisateurs

6.1.1.RU RemoteUI

Héritage : Cette classe dérive de CDialog, une classe des Microsoft Foundation Classes.
Données : Aucunes
Fonctions :

```
public :
void EnableRemoteControl(CkRemoteId, bool)
```
Action : Affiche et active un bouton de la télécommande

```
private :
void OnRemoteBtn(CkRemoteId)
```
Action : Crée un CkScriptElement basé sur le CkRemoteId en vérifiant avec l'objet ParametersDlg s'il y a une condition. Puis, appel Dedalus::RemoteEvent (CkScriptElement) pour ajouter le CkScriptElement tel que spécifié à la section 2.MU.3.1 du SELD.

6.1.1.GU GridUI

Héritage : Cette classe dérive de CDialog, une classe des Microsoft Foundation Classes.
Données : Aucunes
Fonctions :

```
public :
void SetGridBackground(CkGridArray)
```
Action : Place les bitmaps d'arrière-plan dans GridUI selon les données dans le CkGridArray.

```
void SetGridObject(CkGridElement, CkVector)
```
Action : Place un bitmap représentant un CkGridElement à la position donnée par un CkVector.

```
void MoveRobot(CkVector, CkVector)
```
Action : Enlève le bitmap du robot de la position du premier CkVector et place un bitmap du robot à la position donnée par le deuxième CkVector.

6.1.1.SU ScriptUI

Héritage : Cette classe dérive de CDialog, une classe des Microsoft Foundation Classes.
Données : Aucunes
Fonctions :

```
public :
void UpdateScript(CkScriptContainer)
```
Action : Met à jour la listbox du script avec le contenu du CkScriptContainer.

```
void SetCursorPosition(long)
```
Action : Place le curseur du listbox à la position donnée.

6.1.1.UU UserInfoUI

Héritage : Cette classe dérive de CDialog, une classe des Microsoft Foundation Classes.
Données : Aucunes

Fonctions :

```
public :
void SetPlayerInfo(CkPlayerInfo)
```

Action : Affiche les données de l'usager dans les champs de UserInfoUI.

6.1.1.AU AdminUI

Héritage : Cette classe dérive de CDialog, une classe des Microsoft Foundation Classes.
Données : Aucunes
Fonctions :

```
public :
void EnableAdminControl(CkAdminControlId, bool)
```

Action : Active un bouton d'administration sur l'interface tel que spécifié à la section 2.MU.4.1 du SELD.

```
private :
void OnAdminBtn(CkAdminControlId)
```

Action : Appelle DedalusGame::UserEvent(CkAdminControlId) pour signaler au contrôleur qu'un bouton a été cliqué tel que spécifié à la section 2.MU.3.3 du SELD.

6.1.1.MU MainUI

Cette classe met en œuvre la spécification 2.MU.1 du SELD.

Héritage : Cette classe dérive de CFrameWnd, une classe des Microsoft Foundation Classes.
Données : Aucunes
Fonctions :

```
public :
void EnableAdminControl(CkAdminControlId, bool)
```

Action : Appelle AdminUI ::EnableAdminControl(CkAdminControlId, bool).

```
void EnableRemoteControl(CkRemoteId, bool)
```

Action : Appelle RemoteUI::EnableRemoteControl(CkRemoteId, bool).

```
void InitUI(void)
```

Action : Initialise l'interface utilisateur et affiche toutes les sous-fenêtres de MainUI et MainUI lui-même.

```
void SetGridBackground(CkGridArray)
```

Action : Appelle GridUI::SetGridBackground(CkGridArray).

```
void SetGridObject(CkGridArray)
```

Action : Appelle GridUI::SetGridObject(CkGridArray).

```
void MoveRobot(CkVector, CkVector)
```

Action : Appelle GridUI::MoveRobot(CkVector, CkVector).

```
void SetPlayerInfo(CkPlayerInfo)
```

Action : Appelle UserInfoUI::SetPlayerInfo(CkPlayerInfo).

```
void SetWaitingMode(bool state, float Percentage)
```

Précondition : Percentage doit être entre 0 et 1.

Action : Transforme le curseur de souris en un sablier standard si la valeur bool est vraie, sinon, remet le curseur en flèche standard. Une version future pourra afficher une barre de progression avec la valeur de Percentage.

`CkFeedbackAction ShowFeedback(CkFeedback)`
Action : Appelle FeedbackDlg::Showfeedback(CkFeedback).

`CkUserName GetUser(void)`
Action : Appelle IntroDlg::GetUser(void).

`void ShowTutorial(CkLevel)`
Action : Appelle TutorialDlg::ShowTutorial(CkLevel).

`void UpdateScript(CkScriptContainer)`
Action : Appelle ScriptUI::UpdateScript(CkScriptContainer).

`void SetCursorPosition(long)`
Action : Appelle ScriptUI::SetCursorPosition(long).

`CkUserName GetUser(void)`
Action : Appelle IntroDlg::GetUser(void).

`void ShowFeedback(CkFeedback)`
Action : Appelle FeedbackDlg::ShowFeedback(CkFeedback).

6.1.1.ID IntroDlg

Cette classe met en œuvre la spécification 2.ID.1 du SELD.

Héritage : Cette classe dérive de CDialog, une classe des Microsoft Foundation Classes.
Données : Aucunes
Fonctions :

`public :`
`void Init(void)`
Action : Initialise la fenêtre sans l'afficher.

`char * GetUser(void)`
Action : Affiche la fenêtre et attend qu'elle soit fermée par un événement utilisateur. Retourne la valeur donnée par cet événement.

`private :`
`void OnBtnExist(void)`
Action : Survient quand l'utilisateur clique sur "Utilisateur existant" tel que spécifié à la section 2.ID.3.2 du SELD.
Pseudocode : Appelle ExistingUserDlg::GetUser().
S'il y a une valeur de retour, la donne comme retour de GetUser() et ferme la fenêtre.
Sinon, ne fait rien.

`void OnBtnNew(void)`
Action : Survient quand l'utilisateur clique sur "Nouvel utilisateur" tel que spécifié à la section 2.ID.3.1 du SELD.
Pseudocode : Appelle NewUserDlg::GetUser().

S'il y a une valeur de retour, la donne comme retour de GetUser() et ferme la fenêtre
Sinon ne fait rien.

`void OnClose(void)`
Action : Survient quand l'utilisateur clique sur le X de la fenêtre. Fait fermer l'application par DedalusGame.

6.1.1.ED ExistingUserDlg

Héritage : Cette classe dérive de CDialog, une classe des Microsoft Foundation Classes.
Données : Aucunes
Fonctions :

`public :`
`void Init(void)`
Action : Initialise et crée la fenêtre sans l'afficher.

`CkUserName GetUser(void)`
Action : Affiche la fenêtre, appelle FillPlayersList() et attend un événement utilisateur. Retourne la valeur donnée par cet événement

`private :`
`void FillUserList(void)`
Action : Appelle DedalusGame::GetPlayersList(char[][]) et place cette liste dans le ListBox. Appelle DedalusGame::GetLastPlayer() et sélectionne le nom qui correspond à cette valeur.

`void OnBtnCancel(void)`
Action : Survient quand on clique sur "Annuler". Ferme la fenêtre et fait que GetUser() ne renvoie rien tel que spécifié à la section 2.ED.3.2 du SELD.

`void OnBtnOk(void)`
Action : Survient quand on clique sur "OK". Ferme la fenêtre et fait que GetUser() renvoie le nom sélectionné tel que spécifié à la section 2.ED.3.1 du SELD.

`void OnClose(void)`
Action : Survient quand on clique sur le « X » de la fenêtre. Ferme la fenêtre et fait que GetUser() ne renvoie rien.

6.1.1.ND NewUserDlg

Cette classe met en œuvre la spécification 2.ND.1 du SELD.

Héritage : Cette classe dérive de CDialog, une classe des Microsoft Foundation Classes.
Données : Aucunes
Fonctions :

`public :`
`void Init(void)`
Action : Initialise et crée la fenêtre sans l'afficher.

`CkUserName GetUser(void)`
Action : Affiche la fenêtre et attend un événement utilisateur. Retourne la valeur donnée par cet événement.

```
private :
void OnBtnCancel(void)
```
Action : Survient quand on clique sur "Annuler". Ferme la fenêtre et fait que GetUser() ne renvoie rien tel que spécifié à la section 2.ND.3.2 du SELD.

```
void OnBtnOk(void)
```
Action : Survient quand on clique sur "Ok" tel que spécifié à la section 2.ND.3.1 du SELD.
Pseudocode : Appelle DedalusGame::RegisterNewUser(char*).
Si la réponse est positive, ferme la fenêtre et fais que GetUser() renvoie le nom tapé.
Sinon affiche une boîte de message avec un avertissement "Nom déjà utilisé", puis ne fait rien.

```
void OnClose(void)
```
Action : Survient quand on clique sur le X de la fenêtre. Ferme la fenêtre et fait que GetUser() ne renvoie rien.

6.1.1.PD ParametersDlg

Cette classe met en œuvre la spécification 2.PD.1 du SELD.

Héritage : Cette classe dérive de CDialog, une classe des Microsoft Foundation Classes.
Données : Aucunes
Fonctions :

```
public :
void Init(void)
```
Action : Initialise et crée la fenêtre sans l'afficher.

```
bool GetParameter(CkScriptElement *)
```
Action : Affiche la fenêtre avec les paramètres disponibles pour le ScriptElement et attend un événement utilisateur. Retourne la valeur donnée par cet événement.

```
private :
void OnBtnCancel(void)
```
Action : Survient lorsqu'on clique sur "Annuler". Ferme la fenêtre et fait que GetParameter retourne faux tel que spécifié à la section 2.PD.3.4 du SELD.

```
void OnBtnOk(void)
```
Action : Survient quand on clique sur « Ok ». Affiche un message d'erreur si aucun paramètre n'est sélectionné, sinon ferme la fenêtre, place le paramètre sélectionné dans le CkScriptElement, et fait retourner vrai par GetParameter()tel que spécifié à la section 2.PD.3.3 du SELD.

```
void OnClose(void)
```
Action : Survient quand on clique sur le « X » de la fenêtre. Ferme la fenêtre et fait que GetParameter() retourne faux.

6.1.1.FD FeedbackDlg

Cette classe met en œuvre les spécifications de la section 2.FD.1 du SELD.

Héritage : Cette classe dérive de CDialog, une classe des Microsoft Foundation Classes.
Données : Aucunes

Fonctions :

```
public :
void Init(void)
```

Action : Initialise et crée la fenêtre sans l'afficher.

```
CkAdminControlId ShowFeedback(CkFeedBack, CkPlayerInfo)
```

Action : Affiche la fenêtre avec les données de CkFeedback et de CkPlayerInfo dans ses champs. Active ou désactive les boutons conformément aux données de CkFeedback. Puis attend un événement utilisateur.

```
private :
void OnNextLevel(void)
```

Action : Survient quand on clique sur "Aller au prochain niveau". Ferme la fenêtre et fait que ShowFeedback() retourne NextLevelID tel que spécifié à la section 2.FD.3.4 du SELD.

```
void OnReplay(void)
```

Action : Survient quand on clique sur "Rejouer l'exécution". Ferme la fenêtre et fait que ShowFeedback() retourne ReplayID tel que spécifié à la section 2.FD.3.2 du SELD.

```
void OnNewGrid(void)
```

Action : Survient quand on clique sur "Faire une autre grille du même niveau". Ferme la fenêtre et fait que ShowFeedback() retourne NewGridID tel que spécifié à la section 2.FD.3.3 du SELD.

```
void OnContinueWork(void)
```

Action : Survient quand on clique sur "Retravailler le script". Ferme la fenêtre et fait que ShowFeedback() retourne ContinueID tel que spécifié à la section 2.FD.3.1 du SELD.

```
void OnClose(void)
```

Action : Survient quand on clique sur le « X » de la fenêtre. Annule cet événement (l'utilisateur doit choisir une réponse valide).

6.1.1.TD TutorialDlg

Cette classe met en œuvre la spécification 2.TD.1 du SELD.

Héritage : Cette classe dérive de CDialog, une classe des Microsoft Foundation Classes.
Données : Aucunes
Fonctions :

```
public :
void Init(void)
```

Action : Initialise et crée la fenêtre sans l'afficher.

```
void ShowTutorial(CkLevel)
```

Action : Crée une instance de CkGridUI et de CkGridAnimator et les connecte ensemble. Place les données (texte, images, script) du niveau courant dans les champs pour la page en cours, et active les boutons "PageSuivante" et "PagePrécédente" selon le niveau et la page de tutoriel affichée.

```
private :
void OnAnimate(void)
```

Action : Appelle CkGridAnimator::StartAnimation() en donnant le script d'exemple en paramètre tel que spécifié à la section 2.TD.3.3 du SELD.

`void OnClose(void)`

Action : Survient quand on clique sur le X de la fenêtre ou presse sur OK. Ferme la fenêtre tel que spécifié à la section 2.TD.3.1 du SELD.

`void OnNextPage(void)`

Action : Survient quand on clique sur "Page Suivante". Appelle SetTutorialText(), SetTutorialScript() et SetTutorialGrid() pour la page suivante, et active les boutons "PageSuivante" et "PagePrécédente" selon le niveau et la page de tutoriel affichée tel que spécifié à la section 2.TD.3.2 du SELD.

`void OnPrevPage(void)`

Action : Survient quand on clique sur "Page Précédente". Place les données (texte, images, script) du niveau courant dans les champs pour la page en cours, et active les boutons "PageSuivante" et "PagePrécédente" selon le niveau et la page de tutoriel affichée tel que spécifié à la section 2.TD.3.2 du SELD.

6.1.2. Le module Contrôleur

On retrouve à la **Figure 6** le diagramme de séquence qui démontre l'interaction entre le MainUI et DedalusGame lors de l'étape de la construction du script.

6.1.2.DA CkDedalusApp

Cette classe démarre et termine l'exécution du jeu. Elle est nécessaire pour permettre au jeu de fonctionner dans un environnement Windows.

Héritage : Cette classe dérive de CWinApp, une classe des Microsoft Foundation Classes.
Données :

```
private :
CkDedalusGame * m_pDedalusGame
    // Un pointeur au contrôleur du jeu unique, tel stipulé à la section 2.DG.2 du SELD.
```

Fonctions :

```
public :
bool InitInstance(void)
```

Action : Démarre l'application.

```
int ExitInstance(void)
```

Action : Interrompt l'application.

6.1.2.DG CkDedalusGame

Cette classe coordonne l'activité du jeu et les interactions entre les modules. Se référer aux figures 2 à 8 pour l'interaction avec les classes de l'ensemble des interfaces graphiques utilisateurs .

Héritage : Aucun
Données :

```
private :
CkPlayerInfo m_CurrentPlayerInfo
```

// Les informations relatives au joueur tel stipulé à la section 2.DG.1 du SELD.

`CkMainUI * m_pMainUI`
// Un pointeur à l'interface principale tel que spécifié à la section 2.MU.1 du SELD.

`CkGrid * m_pGrid`
// Un pointeur à la grille de jeu courante unique tel que spécifié à la section 2.GR.2 du SELD.

`CkGridAnimator * m_pGridAnimator`
// Un pointeur à l'animateur de grille unique tel que stipulé à la section 2.GA.2 du SELD

`CkScriptContainer * m_pScript`
// Un pointeur au script courant unique tel que spécifié à la section 2.SC.1 du SELD.

`CkScriptAnalyser * m_pScriptAnalyser`
// Un pointeur à l'analyseur de script unique tel que spécifié à la section 2.SA.1 du SELD.

Fonctions :

`public :`
`CkMainUI * InitGame(void)`
Action : Initialise le jeu.

`void UserEvent(CkAdminControlId ControlId)`
Action : Répond à une action du joueur sur un des contrôles dits "administratifs", soit tout ce qui est extérieur à la télécommande (sortir, exécuter, interrompre l'exécution, regénérer une grille, passer au niveau suivant, effacer un élément du script, passer à l'élément supérieur du script, passer à l'élément inférieur du script, effacer tout le script et lire les instructions). Transfère la commande au module responsable de son exécution.

`void RemoteEvent(CkScriptElement Element)`
Action : Répond à l'entrée d'une commande par le biais de la télécommande en l'insérant dans le ScriptContainer.

`void GetPlayerList(unsigned char[][])`
Action : Ouvre le fichier contenant la liste des joueurs inscrits, la formate sous la forme tableau de chaînes de caractères et la retourne.

`long GetLastPlayer(void)`
Action : Ouvre le fichier contenant la liste des joueurs inscrits. Lit l'index placé en début de fichier indiquant le dernier joueur ayant joué et retourne cette valeur.

`bool RegisterNewUser(char *)`
Action : Vérifie si le nouveau joueur a un nom unique, et si oui, l'inscrit dans la liste des joueurs et retourne VRAI.

6.1.2.PI CkPlayerInfo

Cette classe contient toute l'information relative à un joueur.

Héritage : Aucun
Données :

`private :`
`char m_PlayerName[CK_PLAYER_NAME_LENGTH]`

```
        // Le nom du joueur

    CkLevel m_Level
        // Le niveau courant du joueur

    CkScore m_Score
        // Le nombre de points accumulés par le joueur
```

Fonctions :

```
    public :
    CkLevel GetLevel(void)
```
Action : Retourne le niveau courant de ce joueur.

```
    char * GetPlayerName()
```
Action : Retourne le nom de ce joueur.

```
    Long GetScore()
```
Action : Retourne le nombre de points de ce joueur.

```
    void SetPlayerName(char * PlayerName)
```
Action : Spécifie le nom du joueur (m_PlayerName).

```
    void SetLevel(CkLevel)
```
Action : Spécifie le niveau du joueur (m_Level).

```
    void SetScore(CkScore)
```
Action : Spécifie le nombre de points du joueur (m_Score).

```
    void IncrementScore(unsigned long NbrToIncrement)
```
Action : Incrémente le pointage du joueur d'un nombre donné.

```
    void IncrementLevel()
```
Action : Incrémente le niveau du joueur.

6.1.3. Le module Grille

La **Figure 7** décrit l'étape exécution. On y retrouve les interactions du GridAnimator avec les autres objets lors de cette étape.

La **Figure 5**Erreur! Source du renvoi introuvable. décrit la séquence des événements lors de l'étape de génération de la grille.

6.1.3.GA CkGridAnimator

Cette classe sert à exécuter le script de l'utilisateur sur la grille. Ses données sont définies à la section 2.GA.1 du SELD.

Héritage : Aucun
Données :

```
    private :
    CkGrid m_AnimatorGrid
        // La grille de jeu pour le module animator

    CkGridArray m_AnimatorGridArray
```

```
        // L'état courant de la grille

    CkGridAnimatorErrorType m_ErrorInExecution
        // L'erreur rencontrée lors de l'exécution

    unsigned long m_WaitingDelay
        // Le délai d'attente en millisecondes

    CkScriptContainer * m_ScriptContainer
        // Pointeur vers le conteneur de script du joueur
```

Fonctions :

```
    public :
    CkGridAnimatorErrorType StartAnimation(bool*, CkScriptContainer *)
```
Postcondition : La valeur de retour doit être l'une des CkGridAnimatorErrorType.
Action : Exécute le script de l'utilisateur tel que spécifié la section 2.GA.3.1 du SELD.

```
    void SetGrid(CkGrid AnimatorGrid)
```
Action : Met à jour l'objet CkGrid du module GridAnimator.

```
    CkGridAnimatorErrorType GetErrorInExecution(void)
```
Postcondition : La valeur de retour doit être l'une des CkGridAnimatorErrorType.
Action : Donne l'erreur détectée s'il y en a une.

```
    void SetWaitingTime(unsigned long DelayInMilliseconds)
```
Précondition : La valeur entrée doit être positive et entière.
Action : Enregistre le temps d'attente pour le rafraîchissement de l'IU lors de l'analyse.

```
    private :
    void AnalyseCommand(long)
```
Précondition : La valeur entrée doit correspondre à l'une des entrées possibles du script.
Action : Exécute le script de l'utilisateur, commande par commande, jusqu'à la fin du script, ou la détection d'une erreur, ou une interruption de l'utilisateur.

```
    long DecodeCommand(CkScriptElement)
```
Action : Traduit la commande pour permettre l'analyse contextuelle.

```
    void SetErrorEncountered(CkGridAnimatorErrorType ErrorEncountered)
```
Précondition : La valeur entrée doit être l'une des CkGridAnimatorErrorType.
Action : Enregistre l'erreur si détectée.

6.1.3.GD CkGrid

Cette classe gère la création de grilles. Il s'agit d'une classe virtuelle puisqu'elle ne peut être instanciée directement. On utilisera plutôt une des versions dérivées de CkGrid selon la méthode de génération désirée.

Héritage : Aucune
Données :

```
    private :
    CkGridArray * m_pGrid
        //  Pointeur à la grille définissant la position des murs, des planchers, des portes et du gazon
        tel que spécifié à la section 2.GR.1 du SELD.
```

```
CkVector m_ExitPosition
```
// Position et orientation de la sortie de la grille

```
CkVector m_ObjectPosition
```
// Position et orientation de l'objet à ramasser

```
CkVector m_RobotPosition
```
// Position et orientation du robot

```
unsigned long m_Width
```
// Largeur de la grille

```
unsigned long m_Height
```
// Hauteur de la grille

Fonctions :

```
public :
virtual void Generate(CkLevel Level)
```
Action : Génère la grille appropriée selon le niveau tel que spécifié à la section 2.GR.3.1 du SELD et vérifie la faisabilité de cette grille tel que spécifié à la section 2.GR.3.2 du SELD.

```
CkVector GetExitPosition(void)
```
Action : Retourne la position et l'orientation de la sortie dans la grille.

```
CkVector GetObjectPosition(void)
```
Action : Retourne la position et l'orientation des objets à ramasser dans la grille.

```
CkVector GetRobotPosition(void)
```
Action : Retourne la position et l'orientation du robot dans la grille.

```
const CkGridArray * GetGrid(void)
```
Action : Retourne un pointeur à la grille décrivant le fond de la grille (murs, planchers, portes, zones inaccessibles)

6.1.3.GF CkGridFromFile

Cette grille génère la grille en récupérant un fichier prédéfini sur le disque. Dans une version subséquente, une classe CkGridRandom, pouvant générer et valider dynamiquement une grille, sera mise en oeuvre.

Héritage : Cette classe dérive de CkGrid.
Données : Aucunes
Fonctions :

```
public :
void Generate(CkLevel Level)
```
Action : Récupère un fichier sur le disque et remplit la grille avec les valeurs lues selon le niveau.

6.1.3.GE CkGridElement

Héritage : Aucun

Données :

```
private :
CkGridElementType m_Type
```

// Spécifie le type de l'élément, qui peut être un mur, un plancher, une porte (horizontale ou verticale) ou du gazon (zône inacessible).

```
bool m_Hidden
```

// Spécifie la visibilité de l'élément. Un élément caché est considéré comme une case mystère alors qu'un élément visible n'a pas besoin d'être vérifié avant que le robot y accède.

```
bool m_Open
```

// Spécifie si l'élément peut être traversé par le robot(ouvert) ou pas (fermé).

Fonctions :

```
public :
CkGridElementType GetType(void)
```

Action : Retourne le type de l'élément.

```
bool IsHidden(void)
```

Action : Indique si l'élément est caché.

```
bool IsOpen(void)
```

Action : Indique si l'élément est ouvert.

```
void SetType(CkGridElementType Type)
```

Action : Spécifie le type de l'élément et met m_Hidden et m_Open à leurs valeurs par défaut pour ce type d'élément.

```
void Hide(void)
```

Action : Marque l'élément comme étant caché.

```
void Show(void)
```

Action : Marque l'élément comme étant dévoilé.

```
void Open(void)
```

Action : Marque l'élément comme étant ouvert.

```
void Close(void)
```

Action : Marque l'élément comme étant fermé.

6.1.3.GA CkGridArray

Héritage : Aucun

Données :

```
private :
CkGridElement * m_pGrid
```

// Pointeur à un tableau de m_Height * m_Width éléments

```
unsigned long m_Height
```

// Hauteur de la grille

```
unsigned long m_Width
```

// largeur de la grille

Fonctions :

```
public :
CkGridElement GetElement(unsigned long x, unsigned long y)
```
Action : Retourne une copie d'un élément donné de la grille.

```
void SetElement(unsigned long x, unsigned long y, CkGridElement Element)
```
Action : Spécifie l'élément à placer à un endroit donné de la grille.

```
void Open(unsigned long x, unsigned long y)
```
Action : Appelle la fonction Open() de l'élément.

```
void Close(unsigned long x, unsigned long y)
```
Action : Appelle la fonction Close() de l'élément.

```
void Show(unsigned long x, unsigned long y)
```
Action : Appelle la fonction Show() de l'élément.

```
void Hide(unsigned long x, unsigned long y)
```
Action : Appelle la fonction Hide() de l'élément.

6.1.3.VT CkVector

Cette classe permet d'identifier une position et une orientation dans un système cartésien.

Héritage : Aucun

Données :

```
private :
unsigned long m_X
    // Position X

unsigned long m_Y
    // Position Y

CkOrientation m_Orientation
    // Orientation selon un système Nord-Est-Sud-Ouest
```

Fonctions :

```
private :
unsigned long GetX(void)
```
Action : Retourne la position X.

```
unsigned long GetY(void)
```
Action : Retourne la position Y.

```
CkOrientation GetOrientation(void)
```
Action : Retourne l'orientation.

```
SetX(unsigned long X)
```
Action : Spécifie la position X.

```
SetY(unsigned long Y)
```
Action : Spécifie la position Y.

```
SetOrientation(CkOrientation Orientation)
```

Action : Spécifie l'orientation.

`Set(unsigned long X, unsigned long Y, CkOrientation Orientation)`
Action : Spécifie les trois paramètres de l'objet.

6.1.4. Le module Script

Les classes de ce module représentent le script que le joueur compose.

6.1.4.SA CkScriptAnalyser

Cette classe sert à analyser le script composé par l'utilisateur.

Héritage : Aucun
Données : Aucunes
Fonctions :

```
public :
CkScriptResult Analyse(CkScriptContainer Script)
```
Action : Analyse le script composé par l'usager et retourne un résultat tel que spécifié aux sections 2.SA.1, 2.SA.3.1 et 2.SA.3.2 du SELD.

6.1.4.SC CkScriptContainer

Cette classe représente le script composé par le joueur comme stipulé à la section 2.SC.1 du SELD.

Héritage : Aucun
Données :

```
private :
vector<CkScriptElement> m_ScriptElementList
    // la liste des CkScriptElement tel que spécifié à la section 2.SC.1 du SELD.
long m_Index
    // index indiquant la position courante dans la liste
```

Fonctions :

```
public :
```

Les fonctions suivantes correspondent à la section 2.SC.3.1 du SELD.

`void Insert(CkScriptElement Element)`
Action : Ajoute un élément à la liste à l'endroit indiqué par le curseur courant.

`void SetPosition(long Index)`
Précondition : La valeur de Index ne doit pas dépasser la grosseur de ScriptElementList.
PostCondition : L'attribut `index` prend la valeur de Index.
Action : Positionne la curseur courant à l'endroit spécifié dans la liste.

`CkScriptElement GetCurrent(void)`
Action : Retourne l'élément positionné au curseur courant.

`CkScriptElement GetNext(void)`
Précondition : Le curseur courant doit être positionné à l'avant dernier objet ou moins dans la liste.

Action : Incrémente de un le curseur de l'élément courant et retourne l'élément positionné au curseur courant.

`void Next(void)`
Précondition : Le curseur courant doit être positionné à l'avant dernier objet ou moins dans la liste .
Action : Incrémente de un le curseur de l'élément courant.

`void Remove(void)`
Précondition : La liste doit contenir au moins un objet.
Action : Enlever de la liste l'élément positionné au curseur courant.

`void RemoveAll(void)`
Précondition : La liste doit contenir au moins un objet.
Action : Enlever de la liste tous les éléments.

6.1.4.SE CkScriptElement

Cette classe représente les éléments pouvant faire partie du CkScriptContainer. Cet objet enregistre le type de commande tel que décrit à la section 3.4 du SELC. De plus, l'utilisation d'`enum` permet de respecter la spécification 5.4.2 du SELD.

Héritage : Aucune
Données :

```
private :
CkScriptElementCommand m_Command
    // le type de commande sous forme d'un enum

CkScriptElementCondition m_Condition
    // le type de condition sous forme d'un enum reliée à la commande

unsigned long m_IterationNbr
    // le nombre d'itération reliée à la commande
```

Fonctions :

`public :`
`CkCommand GetCommand(void)`
Action : Retourne le type de commande de l'objet CkScriptElement.

`CkCondition GetCondition(void)`
Action : Retourne la condition reliée à la commande.

`void GetCommand(CkScriptElementCommand Command)`
Postcondition : L'attribut `m_Command` prend la valeur de `Command`.
Action : Modifie le type de commande de l'objet.

`void SetCondition(CkScriptElementCondition Condition)`
Précondition : L'attribut `m_Command` doit faire partie de la liste des commandes ayant une condition.
Postcondition : L'attribut `m_Condition` prend la valeur de `Condition`.
Action : Modifie le type de condition reliée à la commande.

6.2. Design détaillé par données

6.2.1 CkLevel

Cette classe représente le niveau courant du joueur tel que stipulé à la section 2.DG.3 du SELD.

C'est un objet de type `unsigned long`. Il représente le niveau courant du jeu.

6.2.2 CkScore

C'est un objet de type `long`. Il représente le pointage du joueur.

6.2.3 CkScriptElementCommand

C'est un `enum` pouvant prendre les valeurs suivantes : `FORWARD`, `TURN_LEFT`, `TURN_RIGHT`, `IF`, `FOR`, `WHILE`, `TAKE_OBJECT`, `OPEN_DOOR`. Il représente les types de commande disponible dans le script.

6.2.4 CkScriptElementCondition

C'est un `enum` pouvant prendre les valeurs suivantes : `WALL`, `NOT_WALL`, `HAVE_OBJECT`, `HAVE_NOT_OBJECT`, `DOOR`, `NOT_DOOR`, `EXIT`, `NOT_EXIT`, `OBJECT`, `NOT_OBJECT`. Il représente les paramètres des commandes du script.

6.2.5 CkScriptResult

C'est un `enum` pouvant prendre les valeurs suivantes : `ERROR`, `OK`. Il s'agit du résultat retourné par le CkScriptAnalyser.

6.2.6 CkFeedback

CkFeedback est une structure de données contenant les éléments suivants :

```
CkPerformance Performance,
CkError Error,
char * Commentaires de CkScriptAnalyser,
bool Can_NextLevel,
bool Can_Replay,
bool Can_NewGrid,
bool Can_ContinueWorking.
```

Elle sert à transmettre l'information relative à la performance du joueur à l'interface utilisateur.

6.2.7 CkPerformance

C'est un `enum` pouvant prendre les valeurs suivantes : `GOOD`, `BAD`, `AVERAGE`,`ECHEC`. Il est utilisé dans le CkFeedback

6.2.8 CkError

C'est un `enum` pouvant prendre les valeurs suivantes : `NO_ERROR`, `WALK_IN_WALL`, `INFINIT_LOOP`, `OBJECT_MISS`, `EXIT_NOT_FOUND`. Il est utilisé dans le CkFeedback.