**UNIVERSITÉ DE SHERBROOKE**

**DÉPARTEMENT D'INFORMATIQUE**

**TP**

**Rapport de projet :**

# Parallélisation de la génération d'environnements dans une application en temps réel

Travail présenté à

Gabriel Girard

par

*Bertolotti Lucas, 10 235 113*

*Gallant Vincent, 11 098 714*

*Gardeur Benjamin, 09 178 640*

2013-04-26

## Problématique

Nous avons développé, dans le cadre du cours d'intelligence artificielle, un générateur d'environnements aléatoire (niveaux de jeu) pour un jeu de plateformes en deux dimensions. Cependant, la génération de nouveaux environnements peut prendre un certain ; il peut donc y avoir un blocage du jeu pendant quelque temps entre la transition des niveaux. Ceci est indésirable pour un jeu en temps réel, qui doit être mis à jour jusqu'à soixante fois par seconde. Une attente d'une seconde dans un chargement d'environnement est donc dérangeante pour le joueur.

## Solution

Pour régler ce problème, nous avons décidé de paralléliser le chargement d'environnements et la mise à jour du jeu. Ainsi, il est possible de jouer sans temps d'attente. Puisque quatre environnements successeurs sont accessibles à partir d'un environnement quelconque, jusqu'à quatre fils d'exécution de chargement s'exécutent en parallèle avec le fil d'exécution du jeu. Ceci permet d'avoir tous les environnements prêts lorsque le joueur quitte l'environnement courant, peu importe la sortie empruntée.

Si, dû à une génération d'environnement particulièrement longue, l'environnement successeur que le joueur a choisi n'est pas encore chargé, le fil d'exécution du jeu s'arrête jusqu'à ce que la tâche de génération soit terminée.

Suite au changement d'environnement, on ne génèrera pas un nouvel environnement pour l'entrée de ce dernier ; nous prendrons l'environnement précédent à sa place.

## Code source concernant la parallélisation

Dans le fichier source *« Game\ExplorationGameMode.cs » :*

- La classe *GenerateLevelTask* encapsule la création de fils d'exécution de chargement
- Dans la classe *ExplorationGameMode* :
    - La fonction *NotifiedEvent* est appelée lors du choix d'un environnement successeur par le joueur. Il appelle donc le code permettant de charger les nouveaux environnements successeurs
    - La fonction *CreateAdjLevel* s'occupe du chargement de tous les successeurs du niveau courant

# Problèmes

La génération d'environnements fut beaucoup plus performante que prévu. Il a donc fallu qu'on ajoute des attentes aux fil d'exécution afin de simuler une génération d'environnements complexe (dans la classe *ExplorationGameMode*, fonction *Start*).

# Démarrer l'application

Le redistribuable .NET version 4.5 est nécessaire pour l'exécution du logiciel.

## Depuis l'exécutable

Dans le répertoire bin, double-cliquer sur l'exécutable *« SuperVetroidMania.exe ».*

## Depuis le code source

Ouvrir la solution Visual Studio 2010 situé au chemin *« src/GameName1.sln »*. Compiler et exécuter le projet « GameName1 ».

Si le programme ne démarre pas, s'assurer que le dossier *« src/bin/WindowsGL/Debug|Release/Content »* existe. Si le dossier n'existe pas, le créer avec le contenu présent dans *« bin/Content »*.

## Commandes dans l'application

**A :** déplace le personnage vers la gauche.
**D :** déplace le personnage vers la droite.
**W :** fait sauter le personnage (plus la touche est enfoncé, plus le personnage saute haut).