XNA – Frogger

Sommaire

1. Description et mode d'emploi

1.1. But du jeu

Le but du jeu du Frogger est simple, il suffit de traverser le niveau en évitant de se faire écraser par une voiture ou de se noyer dans la rivière.

Pour cela le joueur doit déplacer une grenouille qui peut se déplacer et peut monter sur des éléments de l'environnement de jeu pour traverser la rivière.

1.2. Eléments du jeu

Voici les éléments qui composent cette version du jeu :

- **Les moyens de transport :**

Les troncs, se déplacent sur les rivières, la grenouille peut sauter sur ces derniers.

Les nénuphars, se déplacent également et peut servir de transport. Attention, ils disparaissent souvent sous l'eau ce qui a pour effet de tuer la grenouille qui resterait trop longtemps sur ces derniers.

- **Les dangers :**

Les voitures, se déplacent sur les routes et tue la grenouille qui entre en collision avec ces dernières.

1.3. Les touches utiles

Pour jouer au Frogger, rien de plus simple :
Joueur 1 :
Les touches directionnelles sont à utiliser.

Joueur 2 :
Touche QSZD (Q = gauche, S = bas, D = droite, Z = haut)

Pour démarrer après la mort appuyer sur **Entrée.**

La touche **Echap**. Permet d'afficher le menu de fermeture du jeu, disponible à tout moment du jeu.

1.4. Les menus

Pour choisir les options et valider celles-ci dans les menus, il faut utiliser les touches haut et bas (touches directionnelles). Appuyez sur Entrée pour valider l'option mis en avant.

1.5. Les astuces

Changement de niveau
Les touches F1 à F5 permettent de se placer sur le niveau correspondant (de 1 à 5).

Modification des niveaux
Vous pouvez modifier les niveaux en éditant le fichier levels.xml dans le dossier de l'exe et le sous dossier levels. Il est alors possible de changer le sens, la vitesse ainsi que customiser complètement les niveaux.

2. Analyse du travail effectué

2.1. Réflexion sur l'architecture

L'architecture semble être cohérente et évolutive, les managers répondent à un besoin et donnent accès à leurs méthodes une fois instancié. Une solution de classe statiques aurait pu éviter de les instancier, mais dans certains cas, le besoin d'effectuer le rendu de ces managers (GameLogicManager par exemple) implique qu'il soit des classes dérivant de DrawableGameComponent instanciables.

La plupart de ces managers communiquent parfaitement entre eux en passant par la référence vers l'objet MainGame. Cela évite ainsi d'avoir des méthodes multiples dans la classe principale afin d'effectuer des opérations de base dans les managers.

2.2. Justification de certains choix techniques

Support unique du clavier
Les méthodes de gestion des entrées utilisateurs sont conçues de façon à pouvoir facilement supporter les autres périphériques. Il suffit alors de créer les différentes méthodes de gestion de la souris ou du contrôleur Xbox360 et d'ajouter l'appel de ces dernières dans la méthode UpdateAllInputs() de la classe principale.

Import des niveaux
J'ai volontairement choisi de coder une méthode d'import un peu rude et longue en n'utilisant pas de sérialisation directement sur mes classes entités.
La solution de sérialisation certes élégantes nous fait nous éloigner du Framework XNA, déjà en utilisant un fichier XML directement via un objet XMLReader, le code réalisé ne devient plus portable sur plateforme Xbox360. Je voulais ainsi conserver le reste de mon code « standard » à la philosophie portable de XNA.
Pour être parfaitement compatible XNA, il aurait été préférable de coder notre propre importer dans le content pipeline, l'utilisation de System.IO est une faute en soit.

Gestion des collisions
Je n'utilise pas la méthode Intersect() sur mes entités (dérivant de DrawableGameComponent), j'ai choisi de réaliser une méthode vérifiant cette intersection afin de pouvoir par la suite, modifier cette détection de collisions (implémenter par exemple une tolérance suivant le nombre de pixel impliquant la collision, etc.).

2.3. Problèmes connus

Score
Le score semble rester à 0.
En fait ce score correspond au nombre de grenouille ayant réussi à traverser le niveau, par défaut j'avais fixé la limite à 3, il fallait donc 3 grenouilles pour atteindre le niveau suivant, je trouvais ça un peu long, du coup j'ai réduis ce nombre minimum à 1.
Ainsi la grenouille arrivant au bout du niveau, le niveau supérieur est chargé et le compteur de grenouille affiché comme score reste nul.

Il n'y a pas de HighScore et score final au jeu, et ce dernier n'est donc pas sauvegardé sur disque (d'autant plus qu'il aurait fallu utiliser le namespace Storage de XNA afin de répondre au besoin le plus proprement possible, utiliser System.IO est une faute).

Rivière capricieuse au premier chargement
La rivière semble afficher un décalage visuel ennuyeux lors du premier chargement, celui-ci peut apparaître de temps à autres. Le décalage imposé dans le LevelManager semble corriger ce souci la plupart du temps.

Gestion des modes de jeu et fonctionnalités
La gestion des modes aurait dû être plus poussée.
En proposant par exemple des modes différents pour le multi-joueurs : mode versus (premier arrivé et score), mode coopératif (les 2 doivent arriver), écran splitté.

Mais aussi des fonctionnalités sur le jeu en solo : niveau de difficulté, mode endurance (parcours défilant infini par ex.), bonus et items, ennemis supplémentaires traqueurs.

Animation, son, et réseau
N'étant pas une priorité imposée, j'ai décidé de ne pas m'en occuper dans l'immédiat, une prochaine version pourra mettre en œuvre les fonctionnalités d'animation plus sympathique, des sons sur événements et ambiance.
Le mode 2 joueurs pourraient exploiter les possibilités réseau proposées par XNA 2.0.

3. Aperçu du jeu et résumé

Menu principal :

Niveau 1 en mode solo :

Niveau 2 en mode multi :

Niveau 3

Niveau 4

Niveau 5