Technical design document

Arcade sport game project

12 décembre 2012

Table des matières

Revisions

Version	Date	Description
v1.0	10.07.2012	Creation du document
v1.1	10.12.2012	Ajout de la norme
v1.2	10.28.2012	Quelques mots sur les challenges
v2.0	11.23.2012	Mise à jour IA/Caméra
v2.1	11.23.2012	Ajout Outils/Ressources utilisées
v2.2	12.12.2012	Ajout liens, outils communication

Première partie

Introduction

Ce document sera rempli au fur et à mesure que le projet avancera pour expliquer comment nous avons résolu les problèmes et les contraintes du Game Design.

Pour le moment, nous avons prévu de coder le jeu à l'aide de Unity3d et de C#. Pour la partie multijoueur, nous utiliserons le réseau (avec Internet Protocol).

1 Norme

Pour s'y retrouver plus facilement, nous avons établi une norme basée sur les contraintes d'Unity et de ce document : 50 tips for working with unity

1.1 Scripts

Ils seront en priorité écrits avec C#. On utilisera les autres langages que si nous n'avons pas le choix.

1.2 Hiérarchie

1.2.1 Fichiers

Tout type de fichier doivent se retrouver ensemble. Les textures avec les textures, les scripts avec les scripts, etc..

Il ne faudra pas non plus hésiter à créer des dossiers, peu importe si le chemin pour accéder aux fichiers est long.

1.2.2 Scène

Une fois de plus : ne pas hésiter à ranger les GameObjects dans des GameObjects vides. Les scripts demanderont quel GameObject devra accueillir quels GameObjects dynamiques

Tout, sauf les GameObjects vides, devront être instanciés par des prefab.

1.3 Nommage

Tout ce qui peut, dans le projet, suivre cette convention, devra suivre cette convention de nommage :

1. Tout devra être en anglais.
2. Tout devra commencer par une majuscule, ne contenir aucun caractere special (comme l'underscore "_" ou l'espace).
3. Si le nom contient plusieurs mots, les coller ensemble et le faire commencer par une majuscule.
4. On part de l'élément le plus spécifique
5. Aucune abréviation.

6. Le fichier lié à l'objet devra avoir le même nom de base

Exemple :
- RedButton : OK
- ButtonRed : Red est l'élément le plus spécifique
- RedBtn : Pas d'abréviation
- redButton : On commence par une majuscule
- Red Button : Pas d'espace
- Red_Button : Pas d'underscore ou autre caractère spécial
- Redbutton : Faire commencer le second mot button par une majuscule
- RougeBouton : En anglais

1.3.1 Version

Dans le cas où un fichier contient plusieurs versions :

1. Les fichiers archivés devront se situer dans le dossier Version, puis dans un dossier ayant le nom de base
2. Les fichiers utilisés seront dans le dossier /Archive
3. Les fichiers auront pour suffixe "_X" où X est l'indice de version
4. L'indice de version a pour seule contrainte de devoir suivre l'ordre alphabétique (table ascii) mais peut ne pas suivre les règles de convention de nommage citées ci-dessus

Exemple :

```
/
├── Archive/
│   └── Prefabs/
│       └── Buttons/
│           └── RedButton/
│               ├── RedButton_v1.0_201210121234
│               └── RedButton_v1.0_201210040930
└── Current/
    └── Prefabs/
        └── Buttons/
            └── RedButton
```

1.3.2 Extension de fichier

Pour les fichiers, le nom de base devra respecter la convention de nommage ci-dessus et devra contenir une extension selon le type du fichier.

Type	**Ext.**	**Description**
Script en C#	.cs	Morceau de code utilisé par unity
Image	.jpg .png .psd	Image utilisée pour une texture ou une map
GameObjects	.prefab	Les GameObjects préenregistrés
Scene	.unity	Une scène (Ex : menu, terrain, ..)
Shader	.shader	Méthode de rendu d'un material
Material	.mat	Une matière (shader paramétré)
Animation	.fbx	Mouvement du RIG[1] d'un objet

1. Squelette d'un objet, lié au meshes pour permettre une animation

2 Outils

Pour notre projet, nous utiliserons plusieurs outils :

Nom	Version	Utilité
Moteur de jeu		
Unity	4.0	L'outil principal de développement pour le jeu
Programmation		
Visual Studio	2012	Editeur de texte pour les scripts
MonoDevelop		
Graphismes		
Maya		Modelisation, animation, set-up
Z-Brush		Sculpt, texturing
Photoshop		painting, texturing
Paint.NET		Dessin
Illustrator		Dessin vectoriel, design d'interface
Documentation		
Office		Ensemble de logiciels de création de document
LaTeX		Langage de création de documents
Espace de stockage		
git		gestionnaire de version en ligne
dropbox		système de cloud pour les versions clés.
Organisation		
Trello	site	Liste de tâches (scrum)
Facebook	site	Communication (persistant)
Doodle	site	Sonde les disponibilités
Skype		Communication en temps réel

3 Assets utilisées

3.1 Scripts

3.1.1 Spirite Manager

Bibliothèque de 3 scripts qui gèrent de manière optimisée l'affichage des spirites. L'utiliser nous permettra de rendre plusieurs animations simultanément sans augmenter de façon exponentielle le nombre de Draw Calls.

Lien : Wiki d'Unity

Deuxième partie

Challenges

Dans le développement du jeu, nous avons prévu deux grandes difficultés en particulier : La gestion de l'IA [1] et la gestion de la caméra. Ce seront ces deux grands sujets qui nous prendront le plus gros de notre temps, et que nous reverrons donc, après avoir sortit une première version du jeu pour ne pas freiner le reste de l'équipe.

4 Intelligence Artificielle

Dans un premier temps nous étions partis sur un automate fini dont les états pouvaient soit appeler un sous-automate, soit générer des ordres qui seraient gérés par les state-machines des unités ; les états étant reliés par des 'conditions' (delegates). Cette solution avait pour avantage de pouvoir être gérée par un 'éditeur'. Mais cela impliquerait que le résultat soit stocké dans un fichier, être chargé (parsé), de devoir créer l'éditeur (surement un programme en C avec la SDL [2]) qui crée le fichier. Bien que cela ne devrait, en soi, pas être compliqué à créer, et encore moins à utiliser (même pour les GD [3]), cela risquerait de prendre vraiment trop de temps à fabriquer..

Nous allons donc créer la state-machine pour chaque unité, nous allons faire également un système agent mais sans éditeur, ils auront un code fixe (on reprendra notre ancienne idée que si nous sommes en avance).

5 Gestion de la caméra

Une fois de plus, nous allons partir sur une caméra basique en vue de dessus (inclinée) qui suit la balle à une certaine distance, et qui est dans le sens de l'équipe ayant la balle ou l'ayant eu en dernier si elle est par terre.
Nous nous pencherons sur la fluidité du suivi, les différentes positions de caméra (et la transition vers ces positions) plus tard.

Nous allons nous inspirer de la gestion de la caméra du jeu Jonah Lomu Rugby Challenge que l'on peut voir sur YouTube

1. Intelligence Artificielle
2. Simple DirectMedia Layer
3. Game Designer