Cahier des charges

Thibault Gdalia
Florent Youinou
Mathilde Laplaze
Vincent Baille

17 janvier 2014

Table des matières

Introduction **2**

1 Avancement du Projet **3**
1.1 Moteur Physique . . . 3
1.2 Réseau . . . 4
1.3 Son . . . 4
1.4 État de jeux . . . 5
1.5 Éditeur de Map . . . 7
1.6 Site Web . . . 9
1.7 Graphique . . . 9

2 Et Après ? **11**
2.1 Map Editor 2.0 . . . 11
2.2 Réseaux Multijoueur . . . 11
2.3 Nouveaux Personnage . . . 11
2.4 Graphismes . . . 11

Introduction

Nous sommes la Team Girafe. Un groupe de 4 étudiants composé de Mathilde "Mattou" Laplaze, Vincent "Vincae" Baille, Florent "T4ze" Youinou, Thibault "Skeat" Gdalia.

Nous produisons actuellement un runner 2D, où le joueur parcours nos maps à l'aide d'un Houla-Houla. Cet animal est un oiseau très gourmand, qui à chaque effort, consomme de l'énergie, sous forme de sucre. Mais étant donné que cette énergie lui est nécessaire pour voler, des bonbons sont parsemé tout au long de son parcours et son taux de sucre augmente avec le temps.

Le jeu est composé de différents modes. Le mode solo où vous devez parcourir les différentes maps attention que votre oiseau reste à l'écran car si il disparait par la gauche : c'est perd. Le mode multijoueur où la map est infini et le but est donc d'aller le plus loin possible, le score étant comptabilisé selon le nombre de mètres parcourut, une fois que vous êtes mort (ça arrive à tous un jour malheureusement). Ce score est ensuite envoyé à la base de donnée de notre site web afin de pouvoir consulter le classement des joueurs.

Chapitre 1

Avancement du Projet

1.1 Moteur Physique

Dans un runner 2D, le moteur physique est en quelque sorte la base. L'oiseau doit être attiré par le sol, et doit être bloqué par les obstacles. Il était donc naturel que en fassions notre plus grand centre d'attention durant cette première partie de projet. Nous avons fait en sorte qu'il soit efficace et que l'on puisse le moduler aisément afin de pouvoir s'amuser à changer les propriétés physiques d'une map à l'autre. Par exemple, la gravité est appliquée au personnage selon un coefficient variable qui peut être modifiée en fonction du niveau de difficulté. Le vol de l'oiseau peut donc être contrôlé, tout comme ses battements d'ailes qui le font avancer en même temps qu'il s'élève dans les airs.

Étant donné que la carte de jeux à les mêmes dimensions que la fenêtre, la gestion du dépassement de map à été très simple à réaliser. Un simple test sur la position du personnage par rapport à la fenêtre nous permet donc de l'empêcher de voler plus haut que le bord supérieur, ou de tomber plus bas que le bord inférieur.

Exemple :

```
if(SpritePosition.Y <= ScreenHeight)
{
        // On peux bouger verticalement l'oiseau
}
else
{
        // On laisse l'oiseau a sa position
}
```

Du coté de la gestion des obstacles, nous les avons fais sous forme de rectangle qui contiennent les images des blocs. Nous avions dans un premier temps implémenté une collision basique, rectangulaire, qui fonctionnait avec ces rectangles. Cette collision marchait assez bien mais notre oiseaux est rond, et les images des obstacles ne sont pas forcément rectangulaire non plus, ce n'était donc pas la solution la plus optimale...

Pour remédier à ce problème, l'idée nous est donc venue de passer par une collision par pixel, bien plus adaptée à l'utilisation que l'on en fait dans notre jeux. Le problème de cette collision, c'est qu'elle est très gourmande puisque même en étant optimisée au maximum, elle parcourt beaucoup de pixel. Il a donc fallut assemblé les deux tests, collision rectangulaire puis par pixel afin de ne pas tester une collision par pixel avec le rectangle d'un bloc si il ne touche pas le rectangle du personnage. car si il n'y a pas d'obstacle nous n'avons donc pas besoin de regarder si il y a une collision á gérer, de plus nous n'avons pas besoin de gérer la collision de façon optimal pour les bonbons, donc pas besoin de perdre du temps machine pour quelque chose qui ne nous sert pas, soyons logique. Pour résumer nous avons fait un mix des deux grandes sortes de collisions (on est des oufs, ou pas...)

1.2 Réseau

Comme expliqué dans l'introduction, notre jeux permet, si l'on est connecté, d'envoyer le score à la base de données du site web.

Pour gérer cet accès, nous avons préalablement implémenté la classe DBConnect.cs dans un projet annexe afin de pouvoir facilement tester les fonctions d'ajout de données, de suppressions, de mises à jour, de tests d'existences de valeur et de listages que nous préparions. Certaines fonctions ne nous sont pas utile dans l'état actuel du jeux, mais si l'on veut par la suite rajouter des options tel que s'inscrire depuis le jeux, afficher le classement ou encore changer de pseudo, elles seront déjà prête.

La gestion de base de donnée utilise une Dll nommée "Mysql.Data" car il n'est, par défaut, pas possible de communiquer entre le jeux en C# et la base de données en MySql. Avec quelques tutos en ligne et les connaissances de T4ze dans le domaine du Web, la compréhension et l'implémentation des fonctions fut assez simple et fonctionnel rapidemment.

1.3 Son

Nous savons que tout bon jeu est accompagné d'une vraie bibliothèque de sons, c'est pourquoi Mattou s'est vraiment concentré sur cette aspect du projet, il a fallu tout

d'abord qu'elle recherche les musiques et les effets sonores adéquats pour que l'ambiance lorsque vous êtes dans le jeu soit optimale. Puis nous voulions que ce soit très structuré dans le projet pour ne pas avoir des sons en vrac dans nos Contents, notre solution : utiliser Xact.

1.4 État de jeux

Pour naviguer dans le jeux nous avons mis différents "États".

Dans les premiers jours de code, nous avons rencontré quelques difficultés dans l'agencement de ces états puisque tout était placé dans le Game1 et ce n'était donc vraiment pas simple de s'y retrouver et de modifier correctement les updates nécessaire et des bons affichage. En écoutant des étudiants de niveau supérieur, la solution la plus claire était de faire une pile d'état qui n'afficherais et ne mettrais à jour que le haut de la pile.

Après des recherches, quelques test et des ratés face à cette utilisation de pile, la combinaison d'une énumération d'états et d'un switch c'est avéré correspondre aussi bien à nos attentes. Cela nous a permis de simplifier au maximum le code dans le Game1 et de créer des Classes pour chaque états.

Morceau de code extrait de l'update() actuel du Game1 :

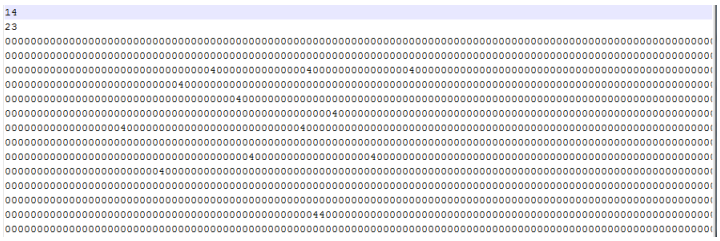
```
switch (CurrentGameState)
{
        case GameState.MainMenu:
            MenuMain.Update(gameTime);
            break;

        case GameState.Playing:
            Partie.Update(gameTime);
            break;

        case GameState.Pause:
            MenuPause.Update(gameTime);
            break;

        case GameState.Dead:
            MenuDead.Update(gameTime);
            break;
```

```
                case GameState.Win:
                    MenuWin.Update(gameTime);
                    break;
            }
```

C'est, comme vous l'aurez sans doute compris, dans le "GameState.Playing" que le gros du jeux se passe puisque c'est la partie qui correspond au jeux en lui même.

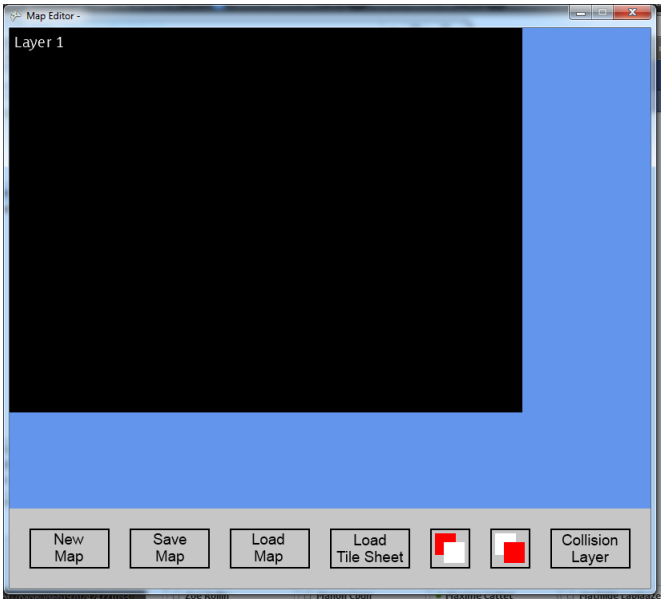
1.5 Éditeur de Map

Nos Maps sont basées sur des fichiers textes (en .txt) que nous renommons en .lvl pour level (sans blagues). Le fichier est composé de : sur la première ligne le nombre d'élements pouvant se trouvé sur une même colonne, sur la deuxième ligne le nombre d'éléments sur une ligne, puis sur le reste c'est la définition de la map



On représente la carte avec des chiffre différents, lorsqu'il y a un "0" par exemple, va représenter une case de vide à un emplacement determiner selon la ligne à dans laquelle il se trouve ainsi que la collonne, il en va de même pour les "4" qui sont eux des obstacles, tout comme les 2, tandis que les 3 sont les fameux bonbons permettant de regagner de l'énergie.

Il est bien entendu que nous n'allons pas écrire toutes nos maps à la main cela serait trop long et très vite lassant. Donc nous avons créé un éditeur de maps qui écrira dans le fichier .lvl de façon a ce que nous ayons juste à cliquer un peu partout pour avoir une map (SUPER)



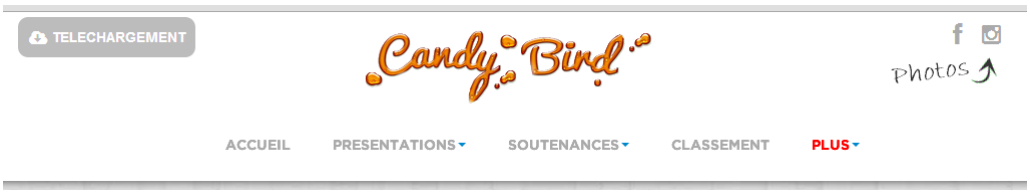
Lors de la création de l'éditeur de map nous avons différents problèmes, tout d'abord pour la création des éléments visuelle tels que les boutons, n'ayant pas de graphistes dans le groupe, cette tache nous a pris plus de temps que prévu. Au niveau de la création de la partie fonctionnel de l'éditeur de map nous ne savions pas du tout comment s'y prendre, puis après avoir étudier nos besoins et ceux des utilisateur, nous avons mis en place les différentes partie de notre éditeur. Skeat et Vincae se sont partagés le travail. Vincae s'est occupé de toute la partie de chargement de map existante et de la sauvegarde des maps, Tandis que Skeat s'est occupé de l'écriture de la map dans .lvl en fonction de l'endroit ou l'utilisateur clique avec sa souris.

1.6 Site Web

URL du site web :

`http://www.CandyBird.eu/`

Dans un esprit pédagogue, nous avons choisis de réaliser notre site à la main plutôt que d'utiliser un CMS (Wordpress ou autre). T4ze avait déjà, auparavant, monté différents site web et il a ainsi pu s'occuper de cette partie sans trop de problèmes. Le site regroupe actuellement une présentation rapide de notre projet accompagnée d'images qui illustrent les différentes parties de notre jeu. Chaque membres du groupe a sa page personnelle, avec une petite présentation ainsi qu'une photo (la classe). Pour la partie pratique, vous pouvez télécharger nos différents rapports, en LaTeX ou en PDF, ainsi que la version du code source présentée lors des différentes soutenances directement depuis le site. De plus, une page est réservée à l'inscription (avec enregistrement de carte bleu tout ça tout ça... histoire de rembourser tout nos frais de l'année). Cette partie n'est pas obligatoire mais recommandée puisque qu'il faut se connecter au jeux pour accéder au mode multijoueur et avoir l'envoie de score en ligne en fin de partie.



Pour rester à l'écoute de nos admirateurs, une page de contact est disponible. Vous avez donc la possibilité de nous laisser des commentaires ou de nous suggérer des améliorations (pas de signalement de bugs puisque nous codons assez bien pour qu'il n'y en ai pas). Et pour ceux qui aiment être toujours informé, nous tenons une page facebook dont un raccourci est visible sur le site (à gauche dans le pied de page).

1.7 Graphique

La partie a été compliqué suite au départ d'Adrien qui n'était pas prévu, il était le plus expérimenté d'entre nous avec les logiciels de graphismes (Photoshop en l'occurrence). Il fallu très vite apprendre les bases de ce logiciel afin de pouvoir réutiliser les travaux déjà produit par Adrien. Nous avons donc du faire face a ce petit problème car un jeu avec des graphismes mal finis n'ai pas très attirant et nous le savons bien.

Nous n'avons pas de personne assigné au graphismes, chacun crée les graphismes dont il a besoin au fur et à mesure que nous avançons dans nos parties. Ce n'est pas forcément très efficace mais cela évite qu'un des membres se consacre entièrement au graphismes au détriment du code ce qui n'est pas le but de ce projet. Nous souhaitons que chaque membres puisse apprendre ce qu'il souhaite, c'est pour cela que nous sommes parfois peut-être trop nombreux sur une tache, mais cette méthode a l'avantage d'impliquer l'ensemble de l'équipe a 100% dans le projet.

Nous avons tout de même défini une charte dans nos graphismes pour ne pas tomber dans un trop gros décalage entre deux partie du jeu. Nous avons donc des modèles de départ pour partir sur les mêmes bases, puis chacun les modifie, tout en montrant aux membres de l'équipe pour qu'ils puissent donner leur avis,et voir si cela reste en accord avec le reste du projet

Chapitre 2

Et Après ?

2.1 Map Editor 2.0

L'éditeur de map sera intégré au jeu afin que vous puissiez créer vos propres maps et ainsi étendre la durée de vie de notre jeu (qui est déjà infini), vous aurez aussi la possibilité de mettre au défi vos amis en partageant vos créations, ce qui rendra le jeu quelque peu plus addictif.

2.2 Réseaux Multijoueur

2.3 Nouveaux Personnage

Nous savons que nos bel oiseau est très attachant, mais nous avons décidé de rajouter de nouveaux personnages, qui auront des caractéristiques physiques différentes, pour que vous puissiez choisir un personnage qui correspondent le mieux à votre style de jeu.

2.4 Graphismes

Conclusion