UNIVERSITE PARIS-SUD

Master Informatique 1ere Année

Année 2017-2018

Rapport de Programmation Web Avancée

Par

Gao Jiaxin

Xue Di

Jeu de catapulte

Enseignant : Kim Nguyễn, Université Paris-Sud

I - Introduction

Ce projet vise à réaliser une plateforme de jeu de catapulte en utilisant des connaissances de HTML et Javascript. Dans ce jeu, nous devons créer un personnage (bird), des actions de personnage doivent être contrôlées par le souris. Le but du jeu est de passer des niveaux du jeu en frappant les cibles par le personnage et laissez-la disparaître.

La reste de rapport est organisée comme suit : dans la deuxième partie, nous vous présentons la répartition de travail, ensuite, nous vous expliquons la structure des documents et à quoi cela sert des documents. Dans la troisième partie, nous vous présentons des fonctions que nous avons réalisées dans ce projet. Enfin, nous vous donnons une conclusion, des difficultés que nous avons rencontrées et ce que nous avons appris grâce à ce projet.

II - Répartition de travail

Ce projet est réalisé en binôme.

Gao Jiaxin a réalisé:

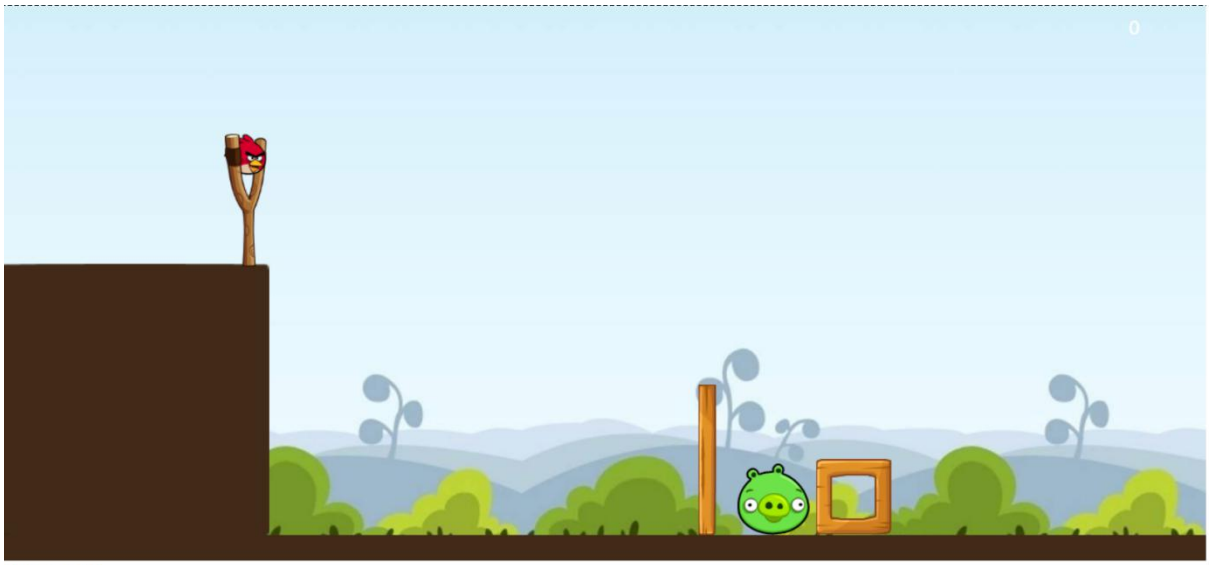
1. Familiariser avec l'utilisation des differentes modèles de la bibliothèque “box2d”, y compris“b2world”, “b2body”, “b2FixtureDef”, “b2shape”, “b2collison”, “b2contacelistener”, etc.
2. Familier avec l'architecture du projet cocos2d, familier avec les éléments basic y compris “director”, “scene”, “layer”, “sprite”.
3. Concevoir les diffierentes scenes.
4. Fichier level4.json, level5.json, level6.json.

Xue Di a réalisé:

1. Developper le page d’accueil en utilisant HTML/CSS et créer le “canvas”.
2. Construire un serveur Apache.
3. Contrôler des actions du personnage et la collision entre le personnage et d’autres objets, contrôler des touches et concevoir le module “score”.
4. Fichier level1.json, level2.json, level3.json.

III - Règles et méthodes d'exécution du jeu

Pour exécuter notre programme, d'abord on doit créer un serveur et placer le dossier dans le répertoire racine du serveur. Après on entrée dans le répertoire “main”, et il va charger le page index.html, et le page est comme ça:



FPS: 60
Mettre à jour: Recharger
Niveaux: 1 OK

La partie supérieure est la partie de l'interface de jeu, la première ligne de la partie inférieure montre que le FPS de notre jeu est de 60, la button du deuxième ligne est mettre à jour le page, la troisième ligne est utilisée pour sélectionner les niveaux.

Notre jeu a 6 niveaux. Chaque fois que nous tuons un ennemi, nous obtenons 100 points. Pour les trois premiers niveaux, 100 est le score parfait. Pour les trois derniers niveaux, 200 est le score parfait.

III - Structure des documents

Dans le répertoire “main” :

index.html:

Ce fichier est la page d’accueil, il a créé le canvas, et charger le document cocos2d.js

```
<script type="text/javascript" src="cocos2d.js" ></script>
```

cocos2d.js:

```
engine: '../cocos2d/cocos2dx.min.js',
```

Par preload, nous pouvons obtenir le engine, nous passons à cocos2dx.min.js.

Cette bibliothèque définit que nous devons lancer main.js, c‘est l‘entrée de notre code. Cocos2d est un framework logiciel ouvert. Il peut être utilisé pour créer des jeux, des applications et d'autres programmes interactifs.

main.js:

```
var cocos2dApp = cc.Application.extend(
```

“extend” est similaire au prototype, nous ajoutons deux parties pour “Application” et il devient la classe “cocos2d”.

```
ctor: function (scene) {
```

```
applicationDidFinishLaunching:function(){
```

La première partie est “ctor”, un constructeur pour précharger les scènes et les ressources.

La deuxième partie est “applicationDidFinishLaunching” qui définit le “directeur” du jeu, définir quelques attributs comme “frameRate”, etc.

```
var App = new cocos2dApp(GameScene);
```

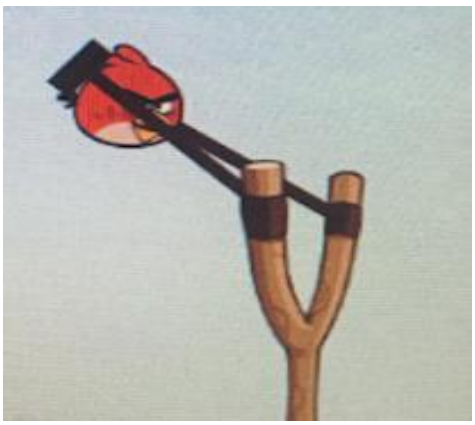
Enfin, nous pouvons charger la scène pour accéder à game.js

game.js:

1.GameScene: Dans Gamescene, nous déclarons une “layer” et l'initialisons, et finalement l'ajoutons à la GameScene.

2.GameLayer:

a. **getTexture():** Renvoie le nom de chemin de l'image correspondant au nom
b. **addObject():** Ajouter un “sprite” avec la description “desc” (paramètres de la fonction) au Gamelayer. Si le “sprite” a une propriété “shape”, nous devons l'appliquer à “b2.enablePhysicsFor” pour simuler.
c. **init():** Initialiser tous les “sprite”,y compris “birdsprite”, “scoreLabel”
d. **update():** 60 images par seconde, chaque seconde change chaque image
e. **onTouchesBegan()**:Lorsque nous touchons bird, il y aura une troisième sling3.



f. onTouchesMoved(): Selon le mouvement de la souris, la position de ‘’bird‘ et la position de “sling1”, “sling2”, “sling3” sont mises à jour.

g. onTouchesEnded(): Contrôler les modifications de “sling1”, “sling2”, “sling3”.

physical.js: Définir les actions physiques, et rendre un système physique. b2 est une classe que nous avons définie pour simuler le monde physique réel. D’abord, on définit un écouteur d'événement “contactListener”. C’est pour detecter les contactes et les collisions. Cette classe peut rendre quelques fonctions importantes.

a. **initWorld():** Initialiser la scène globale ”world” et réglez la direction de gravité vers le bas, son valeur est 10. Après on ajoute “contactListener” à “world”. Et on définit un array vide “bodies” pour ajouter l‘objet “body” dans le futur.
b. **enablePhysicsFor(desc):** Creer l‘objet “body” en base de paramètre “desc”. Et le ajouter dans l'ensemble des “bodies”.
c. **simulate():** Simuler de vraies scènes de mouvement, y compris la position de déplacement, manipulation d'objets morts, etc.

imgs.js: Définit toutes les ressources(images) nécessaires.

level1.json, level2.json, level3.json, level4.json, level5.json, level6.json: Tous les fichiers “json”, conçus pour différents niveaux.

./img: Toutes les images nécessaires.

./cocos2d: La bibliothèque “cocos2d”.

../box2d: La bibliothèque “box2d”.

IV - Fonctionnalités réalisées

Fonctionnalités de base

1. Nous avons terminé toutes les fonctionnalités de base. Le “bird” peut déplacer horizontalement et verticalement, nous pouvons utiliser la souris pour contrôler l'angle et l'intensité du vol de la “bird”.
2. Nous avons conçu 5 niveaux, chacun avec au moins un cible. Le “wood1” et “wood2” est utilisé pour protéger la cible. Donc, en fonction de la position du “wood1” et “wood2”, différentes difficultés sont conçues.
3. Nous avons écrit 5 fichiers de “json” pour l’appeler.
4. Le mouvement de personnage réalistes, nous avons fait le parabolique pour que les actions de personnage peuvent être plus naturelles.
5. Le rendu est fait dans l’élément “canvas”.
6. Le rendu est 60 FPS.

Fonctionnalités avancées

1. **Cibles ou obstacles mobiles:** nous avons donné au “wood1” “wood2” des caractéristiques de mouvement physiques qu'ils pouvaient déplacer. Lorsque nous définissons l'attribut de l'objet comme dynamique, l'objet sera déplacé par la collision comme s'il s'agissait d'un monde réel, ce qui est l'avantage de la bibliothèque “box2d”.
2. **Blessures:** Nous pouvons ajuster différentes force d’impact de collision pour changer la difficulté de la disparition des cibles. Mais nous n'avons qu'un “bird”, donc cette fonction pas vraiment fini.

VI - Conclusion

Ce projet nous permet de familiariser et d'appliquer des connaissances de Javascript que nous avons appris en cours. Nous avons rencontré des difficultés, par exemple,
comment vont disparaître les cibles. Grâce à ce projet, nous pouvons mieux comprendre ce que nous avons appris en cours et c'est aussi une chance nous permettant de réviser et de préparer l'examen final.