ISTIC M2 IL

3D BALL OBSTACLE

PROJET TIA

Chakib BENKEBIR -- Youssouf ROUDANI
04/03/2018

Introduction

Ce projet consiste en la réalisation d'un jeu en réalité augmentée sous Android. Le développement a été effectué avec Unity, un moteur de jeux multiplateformes qui se base sur l'utilisation de script en C#. Le logiciel dispose d'un éditeur de scène qui permet de créer des objets et les manipuler.

Pour la réalisation d'une application de réalité augmentée, nous utiliserons Vuforia, un SDK orienté AR. Cet outil rend possible d'utiliser des objets dans la réalité et leurs attacher un certain comportement dans le programme développé.

Le projet propose deux sujets, une course d'obstacle et la Rube Goldberg Machine. Nous avons opté pour le premier, la course d'obstacle d'une balle pour atteindre un but et gagné le jeu.

Le principe du jeu

Le plateau de notre jeu comporte une balle, un plan, des barrières autour avec une sortie, le but est de faire en sorte que la balle atteigne la sortie en utilisant des objets de soutien comme les rampes et les murs.

A moment de l'apparition du plateau un compteur se déclenche, une fois que le joueur ai fini de placer les objets il peut appuyer sur le buttons Go , c'est qui lancera la balle et arrêtera le compteur.

Le score du joueurs est défini par deux critère qui sont le nombre d'objets de soutient utilisés et le temps passé avant le lancement de la balle.

Des fonctionnalités tel le déplacement et la rotation des objets ont été ajouté afin de leur donner plus de liberté. Pour le déplacement il suffit de glisser l'objet à l'endroit voulu quant à la rotation il faut sélectionner l'objets cible avec un click en suite appuyer une fois sur le bouton rotation pour le faire tourner et une seconde fois pour arrêter la rotation.

Choix techniques

I. Les différents objets

1. ARCamera

   Un objet du SDK Vuforia représentant notre camera de réalité augmenté

2. ImageTarget

   Il contient l'image que ARCamera peut détecter et suivre.

3. Sphere

   Cette objet représente la balle, et nous l'avons déclarer en tant que *Rigidbody* afin qu'il soit sous le contrôle du moteur physique d'Unity

4. Terrain

   Il représente notre plan du jeu, et il est lié directement à l'Image Target cela permet de l'afficher dès le moment ou l'image est détectée. Nous lui avons attaché un Collider pour gérer les collisions.

5. Canvas

   C'est ici que nous mettons tous nos éléments UI tel que les boutons et les textes

II. Les scènes :

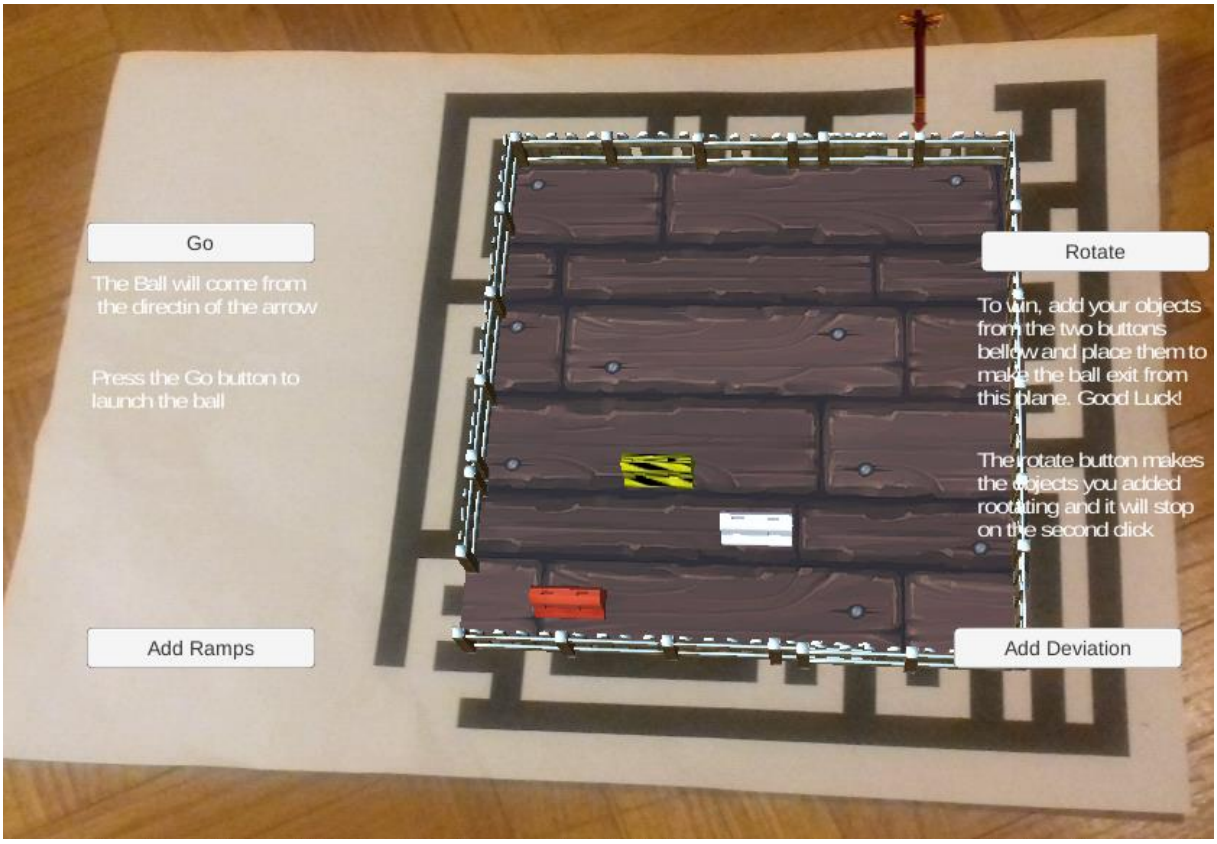
Nous avons décomposé notre jeu en 4 scènes :

1. *Scène Init :*

   Ici le joueur se retrouve avec un menu en deux dimensions lui permettant de lancer le jeu soit en mode Easy ou en mode Hard

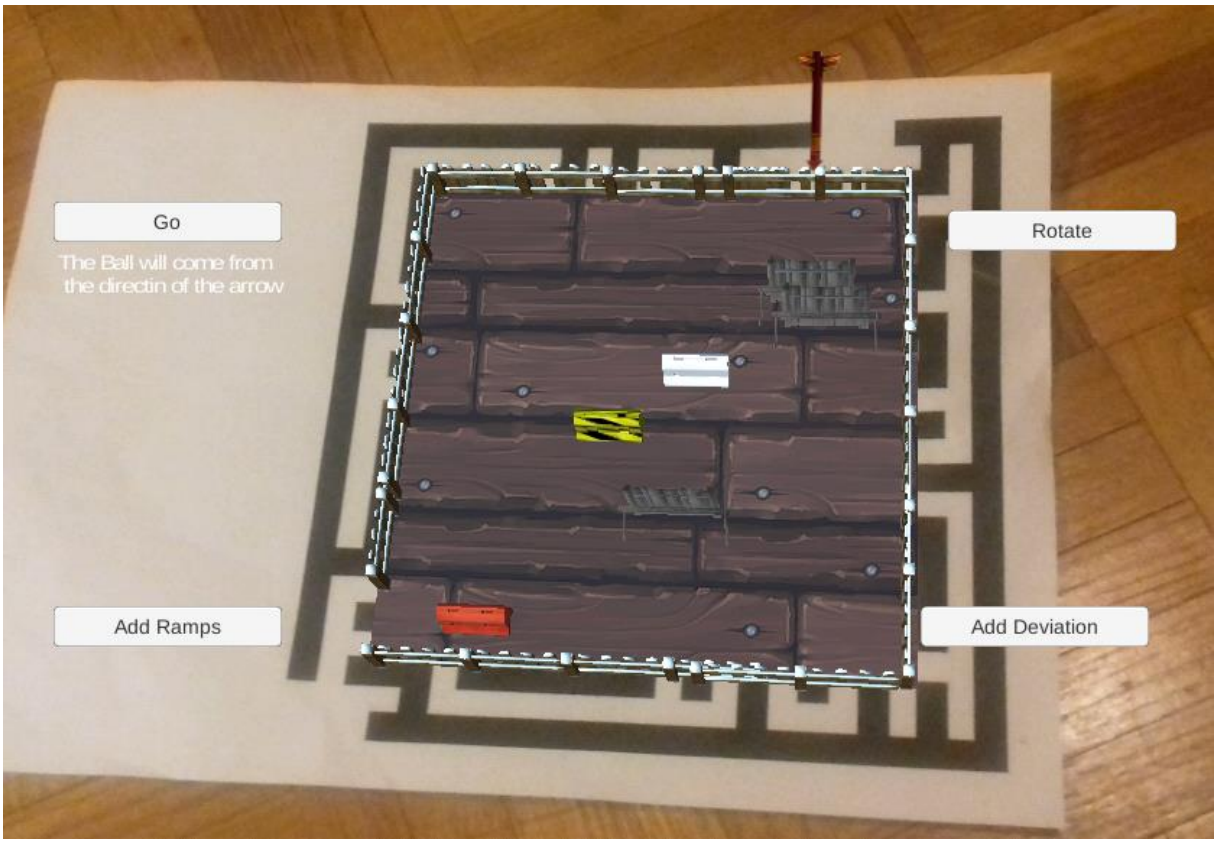
2. Scène Easy :

Cette scène représente le niveau facile du jeu, nous avons défini la difficulté par le nombre d'obstacles générés aléatoirement sur le plan du jeu, donc dans ce mode nous générons trois obstacles.



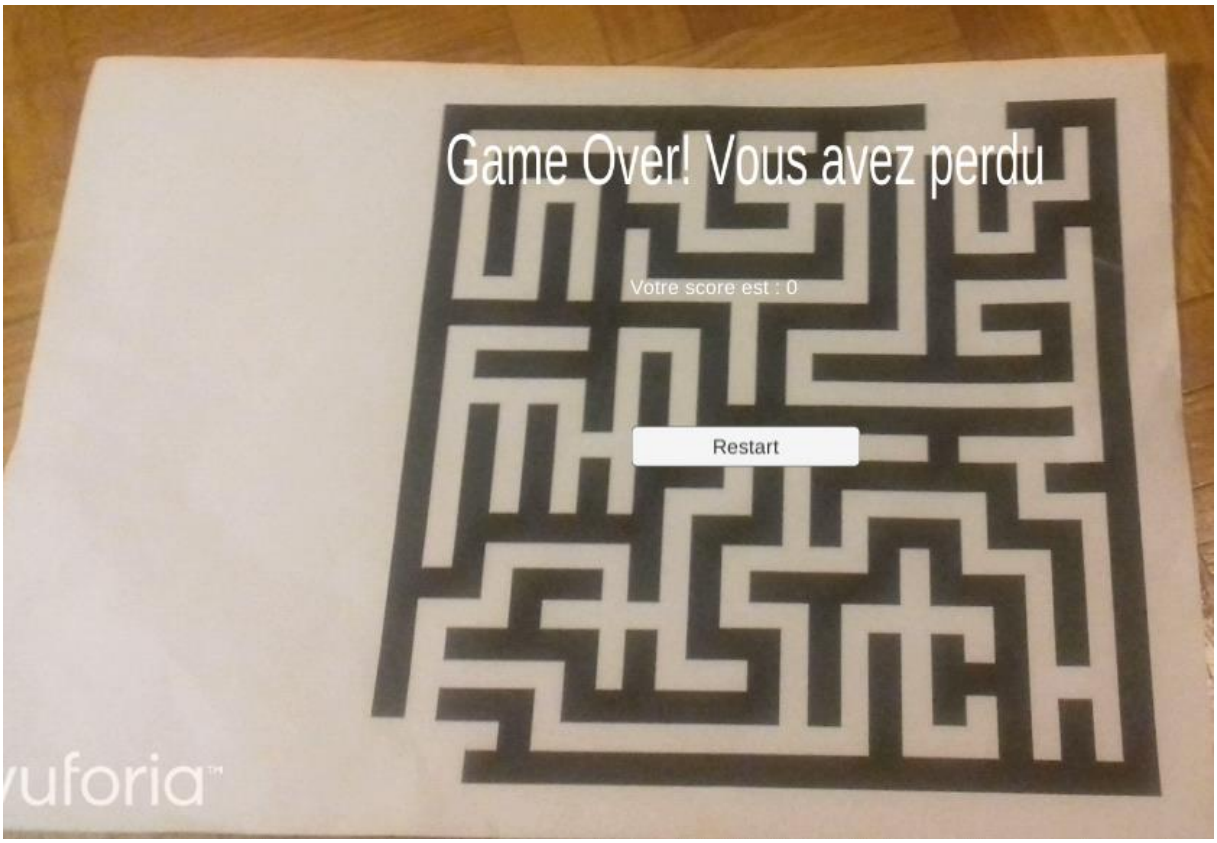
3. Scène Hard :

Ici nous avons les mêmes éléments que la scène Easy avec pour seule différence le nombre d'obstacles générés qui passe à 6 au lieu de 2.



4. Scène End :

Une fois la partie terminée nous affichons une fenêtre avec un message décrivons le résultat, dans le cas de succès par exemple nous affichons un message de félicitation accompagné du score.



III. Les fonctionnalités

1. Drag & Drop

   Pour pouvoir déplacer les objets librement nous avons implémenté les méthodes OnMouseDown() et OnMouseUp() dans un scripte que nous lions aux objets de soutient en suite on met ajoure la position dans Update().

2. Rotation

   Ici nous sauvegardons une références à l'objets sélectionner dans une variable globale, puis au moment d'appuyer sur le bouton « rotate » on procède à une transformation sur l'objet sélectionner. Ex :

```
objectRotate.transform.Rotate(Vector3.up, rotateSpeed * Time.deltaTime);
```

IV. Les scripts

1. Application.sc

   Cette classe permet le partage d'informations entre les différents objets et scènes, elle contient par exemple les variables des compteurs (nombre d'objets, temps).

2. MoveBall.cs

   Attaché directement à la balle ce scripte à pour fonctionnalité première le lancement de la balle en utilisant la méthode AddForce. C'est ici également que le résultat de la partie se décide.

3. AddObject.cs

   Le rôle du scripte est de générer les obstacle dans un premier temps puis de créer les objets de soutiens demandé par l'utilisateur.

4. Score.sc

   Ici on met à jour les messages qui seront jouées dans la scène End, cela va dépendre des variables se trouvant dans Application.sc

V. Fin de la partie

Pour gagner on vérifie si la position de la balle est extérieure au plan, seulement sur l'axe Z car nous avons une seule sortie et elle est dépendante de cet axe.

C'est la vitesse de la balle atteigne une valeur assez petite on considère que la balle s'est arrêté et donc que la partie est perdue.

```
speed = ball.GetComponent<Rigidbody>().velocity.magnitude; speed < minSpeed
```

Conclusion

Bien que le début de ce projet a été extrêmement compliqué, à commencer par l'utilisation de nouveaux outils et surtout un problème avec les salles de TPs censées avoir la plateforme Unity, mais finalement au fur et à mesure de l'avancement du projet on devenait plus curieux et intéressait par les possibilités qu'offrait la réalité augmenté, nous commencions même à nous dire que si nous avions plus de temps on aurait recommencé et fait le projet différemment avec toutes les connaissances que nous avons assimilé.

Mais cela ne nous empêche pas d'être satisfait de notre première expérience dans ce domaine. Sans oublier qu'à présent nous pouvons ajouter C# à la liste des langages de programmation que nous connaissons .