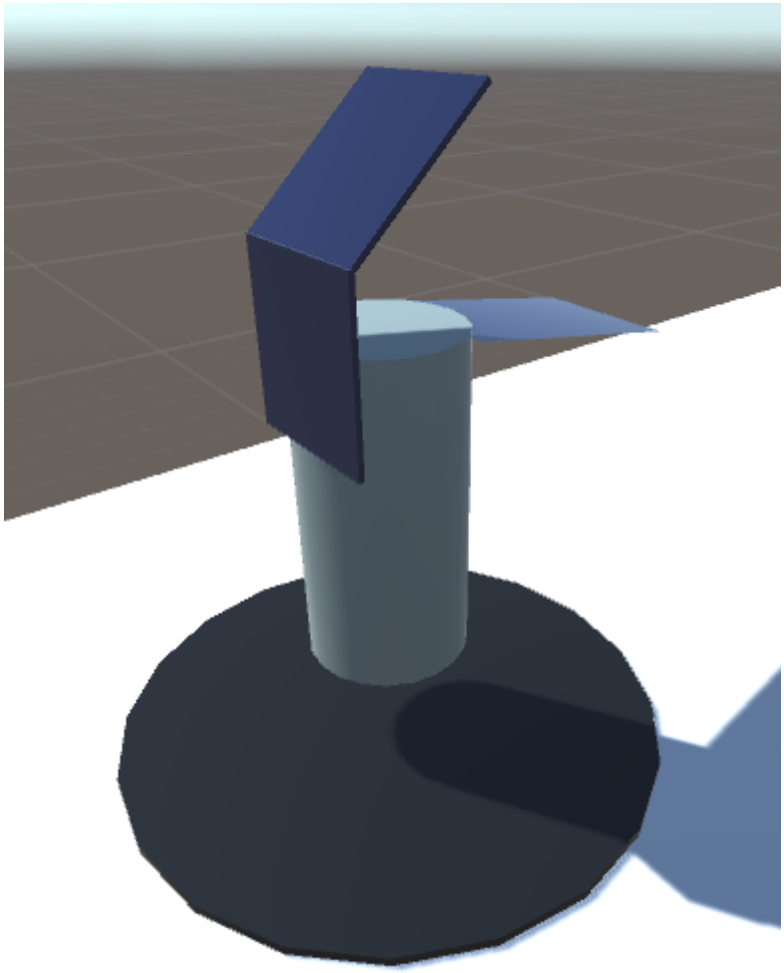
RAPPORT TIA
2017/2018
Obstacle-Race



Timothée Merlet-Thomazeau
Christophe Planchais

I- Game-play et composants

Game-play

Victoire

Notre jeu est de type basket ball, l'utilisateur doit placer des ascenceurs gravitationnels (les "aspirateurs") pour tenter de marquer un panier. Un label texte apparaît lorsque la partie est gagnante. Le code de “arrivee.cs” n’est pas compliqué (utilise seulement un *TextMesh*), mais nous a permis de valider l’entrée en collision avec les objets.

Echec

Le temps est illimité mais un échec peut survenir si la balle tombe du terrain. Pour ce faire nous avons utilisé un deuxième composant *plane* qui détecte la collision et déclenche l'événement échec (une notification à l’utilisateur).

Composants

Aspirateur

Le composant incontournable pour gagner.

Escalator

Permet de gagner en altitude. Il a été ajouté au deuxième niveau (c.f. Section IV).

Pente

Tapis magique

Permet d’augmenter artificiellement la vitesse de la balle. (à faire pour le troisième niveau)

II- Placements des composants

Utilisation

Un "aspirateur" est util pour gagner. Un clique continue sur l’aspirateur le fait tourner, tandis qu’un drag and drop le fait bouger.

De même avec la pente, on peut la faire tourner. Une fois satisfait de l’orientation des composants, il faut appuyer sur le bouton virtuel rouge pour lancer la balle.

Attention, à bien travailler en mode AR (avec un terrain-photo), sinon, la balle se lance toute seule tant qu'elle n'a pas trouvé la photo-terrain.

Extension pour android : la souris

À faire…...

III- Calculs physiques

On peut ajouter un script *.cs* à chaque objet à partir de l'*Inspector* en cliquant sur *Addcomponent*.

Deltatime

Les calculs des forces et vitesses sont extrêmement dépendants du temps. D'autre part, chaque frame peut prendre un temps quelconque pour se dessiner, d'où l'utilisation de *deltaTime* : le temps en secondes qu'il a fallu pour terminer la dernière image : le jeu devient indépendant du *frameRate*.

Transform

Chaque objet d'une scène a un *transform* qui est utilisé pour stocker et manipuler la position, la rotation et l'échelle de l'objet. "GetComponent<Transform>();". On utilise entre autre ce composant dans "aspirateur.cs" pour créer une force *anti-centrifuge* dans le *disque d'aspiration* qui attire la balle vers son centre.

```
void OnTriggerStay(Collider other)  {
    Rigidbody rb = other.GetComponent<Rigidbody>();
    Vector3 dir = transform.position - rb.position;
```

RigidBody

L'ajout de composant à un objet, permet le contrôle de la position de celui-ci grâce au moteur de simulation physique de Unity. L'obtention se fait à partir de son *collider*, et non de l'objet lui-même.

```
void OnTriggerStay(Collider other){
    Rigidbody rb = other.GetComponent<Rigidbody>();
```

Collider

La forme englobante (approximation géométrique) de l'objet, destinée au calcul des collisions. Exemple : *Box Collider*, *Sphère Collider* et *Capsule Collider*. On assigne un *collider* graphiquement à l'aide de l'*Inspector.*

Repère local

La première difficulté fut l'écriture de la pente. Au niveau de l'axe, nous la faisions tourner sans que l'angle de pente soit modifié. Unity fournit cependant des opérateurs de vecteurs :

Forward : raccourci de Vector3(0, 0, 1).

Left : raccourci de Vector3(-1, 0, 0).

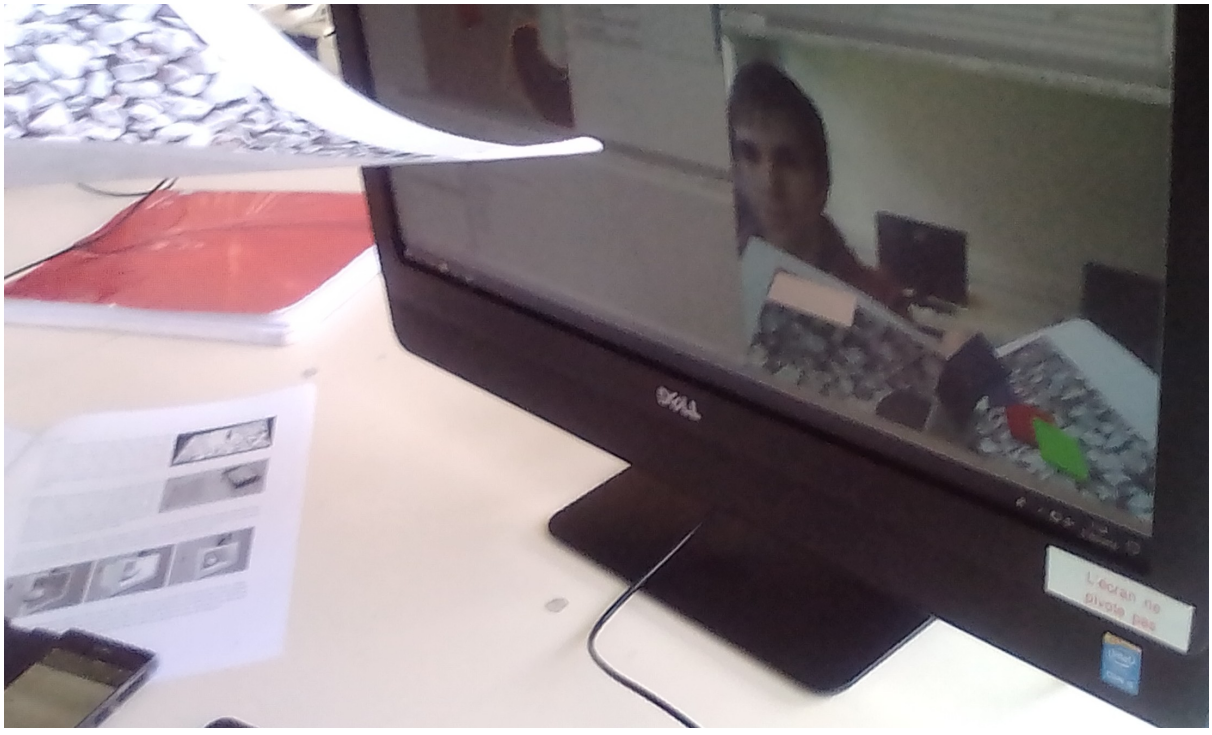
Forces

Si l'énergie de la balle est suffisamment faible pour être attirée par notre *aspirateur*, nous la faisons monter dans la fonction *OnTriggerStay* de "elevateur.cs",:

```
 b rb.AddForce(transform.up * force * 15 * Time.deltaTime);
```

IV- Augmented Reality (RA)

Premier niveau (développement agile)



Après quelques soucis avec la webcam, nous avons une version jouable en RA. On appui sur le bouton rouge pour commencer la partie (cependant, la victoire est assurée pour chaque jeu((.

Deuxième niveau

Deux "UI-Button" ont été placés pour afficher ou cacher notre popup contextuel "tenir enfoncé l'objet afin de le déplacer". "Arrivee.cs" et sa fonction *update* mettent à jour le temps écoulé. Ce-dernier sera affiché avec la notification de victoire.

V- Conclusion

Nous avons rencontré des difficultés avec l'installation d'Unity après une réinstallation globale, par l'Istic, des machines Windows et il ne nous a pas été possible de tester l'export en format de fichier apk pour le système Android.

Ce TP nous a permis cependant de découvrir le langage C# et Unity et nous sommes assez satisfaits de notre application, en terme de rendu AR.

PS: 26.02.2078 : nous avons finalement réussi à exporter l'apk sous Android avec l'aide de notre professeur.