GRELET--CARTIER Suzanne
VON AESCH Nicolas

4A IPS

ARBowling

Réalité augmentée et interactions hybrides - Rapport de projet

Encadrants
COUPRY Corentin
BLANCHARD EMMANUEL

2024-2025

Sommaire

Objectif de l'application

ARBowling est un jeu de bowling en réalité augmentée. Les parties se font à deux joueurs, chacun cherchant à marquer le plus de points en faisant tomber les quilles. Notre objectif était de rendre l'expérience des joueurs aussi proche de la réalité que possible, tout en gardant le matériel nécessaire aussi limité que possible. De cette manière, les joueurs peuvent s'amuser seulement avec un smartphone et une bonne vieille partie de bowling.

Par exemple, deux étudiants de l'ENSIM s'ennuient en début de semestre, n'ayant cours qu'un cours magistral à 8 h et un à 16 h 45. N'ayant pas de projet en cours ni de devoirs à préparer, ils se rendent dans le couloir menant aux salles P20 à P26 et ouvrent l'application ARBowling. Après avoir scanné le sol, ils enchaînent des parties endiablées leur permettant d'oublier leur ennui et de découvrir les possibilités infinies de la réalité augmentée.

Environnement

La partie de bowling se passe dans l'environnement de l'utilisateur. Une fois le sol détecté, les quilles sont posées dessus et le panneau des scores au-dessus.

Interface

Nous avons opté pour une interface minimaliste. Un menu d'accueil permet à l'utilisateur de lancer une partie, ce qui l'amène sur un écran lui permettant de scanner son environnement et de sélectionner le sol. Pour ces différentes étapes, l'utilisateur n'a la possibilité de cliquer que sur un bouton. Par la suite, il peut lancer les boules de bowling

d'un mouvement du doigt sur l'écran, et le score s'affiche sur le panneau des scores montré au-dessus des quilles.

Interactions

Les joueurs interagissent avec l'application avec trois types d'actions. Il peut cliquer sur des boutons, faire glisser son doigt sur l'écran et bouger son téléphone dans l'espace pour se déplacer.

L'utilisateur peut recevoir un feedback du jeu lorsqu'il fait une action particulière, par exemple une vidéo [1] se déclenche lorsqu'il obtient un score de 9 sur ce qu'on appelle un jeu (l'unité comportant deux lancers).

Répartition du travail

Responsable	Tâche

Suzanne	- Gestion du scan du sol et du filtrage des plans
	  - Sélection du bon plan
	  - Agrandir ce plan uniquement
	  - Conserver ses informations pour les utiliser dans la scène suivante
	- Menu d'accueil et interface de scan du sol
	  - Placement des boutons et des textes
	  - Gestion du partage des données nécessaires entre les différentes scènes
	- Gestion des règles d'une partie de bowling
	  - Nombre de lancers dans un jeu
	  - Nombre de jeux dans une partie
	  - Calcul des scores
	  - Cas normal
	  - Cas de spare
	  - Cas de strike
	- Placement des quilles après chaque lancer
	  - Conserver seulement les quilles debout entre deux lancers d'un jeu et les replacer au bon endroit
	  - Replacer toutes les quilles entre deux jeux
	- Gestion du lancement des vidéos lors d'une partie
	  - Vérification des conditions préalables
	  - Affichage de la vidéo et gestion de l'apparence
	- Refactoring
	- Écriture du rapport

Nicolas	- Création des modèles 3D
	  - Boule de bowling
	  - Quille
	  - Panneau des scores
	- Création de la mécanique de lancer
	  - Détection du mouvement du doigt

	- Gestion de la physique - Gestion du délai entre deux lancers - Placement des quilles dans l'espace - Calcul de la position à partir du plan utilisé comme sol - Placement du panneau des scores dans l'espace - Calcul de la position à partir de la position des quilles et du joueur - Actualisation des textes du panneau des scores - Association des textes au calcul des scores - Recherche de vidéos à implémenter - Création des éléments pour la gestion de vidéos dans le jeu - Gestion du GitHub - Écriture du rapport - Rédaction du README

Difficultés rencontrées et solutions trouvées

Détection du sol

Nous souhaitions ne détecter que le sol, ce qui posait plusieurs problèmes.

Tout d'abord, nous devions sélectionner un unique plan. Pour cela, nous choisissons le plan horizontal le plus bas parmi ceux détectés. Cependant, l'utilisateur peut vouloir l'agrandir, il nous fallait donc continuer à scanner l'environnement jusqu'à ce que l'utilisateur en décide autrement. Nous avons donc fait un algorithme de filtrage désactivant et rendant invisibles les autres plans que celui associé au sol.

Dans un souci de performance, nous avons décidé que le jeu arrêterait de scanner le sol une fois l'utilisateur satisfait, ce qui permet d'éviter des calculs inutiles. De cette manière, un seul plan est conservé lors de la phase de jeu.

Placement des éléments dynamiques dans la scène

Le placement des éléments de façon dynamique était compliqué dû à un manque de compréhension du placement relatif des objets Unity quant à une référence d'un objet lui-même dynamique.

De multiples tests nous ont alors permis de comprendre ce placement et de réussir à l'implémenter correctement dans le jeu pour les quilles ainsi que le tableau de score.

Implémentation et physique de la boule de bowling.

L'implémentation de la boule de bowling qui devait être placée dynamiquement par rapport à la caméra de l'utilisateur n'était pas triviale du tout, celle-ci étant un poil décalée vis-à-vis de la caméra réelle du téléphone.

Une fois la boule placée dans l'espace, la physique du lancer de la balle posait aussi des problèmes, par rapport à la documentation Unity qui est difficile à comprendre. Cela est renforcé par l'abstraction de la méthode GetTouch de la classe Input qui n'est pas claire.

Gestion des scores

Le bowling possède des règles particulières au niveau du calcul des scores. Comme l'affichage se fait en direct, il est nécessaire de recalculer celui-ci après chaque lancer, mais il faut aussi prendre en compte les lancers suivants (pas toujours déjà effectués) en cas de spare ou de strike.

Afin de gérer ces difficultés, nous avons séparé en plus petits éléments chaque objet (ex : lancer, jeu, partie) pour pouvoir faire des méthodes simples de calculs, permettant de calculer facilement le score une fois assemblées. Cela augmente la longueur du code, mais le rend plus simplement lisible et modulable, ce qui nous a permis d'ajouter au fur et à mesure des fonctionnalités en toute simplicité.

Placement des quilles après chaque lancer

Le placement des quilles requiert de connaître plusieurs informations :

- numéro et position de la quille
- si elle est tombée ou non
- à quel lancer le joueur est.

Selon ces informations, toutes les quilles ne sont pas replacées.

Pour connaître quelle quille est tombée, nous avons d'abord tenté de calculer sa position et son angle par rapport au sol. Cependant, cela causait de nombreux faux positifs et faux négatifs. La solution qui a été trouvée a été de garder en mémoire l'angle et la position initiale de la quille et de calculer la différence une fois le lancer fini. Si la différence est assez élevée, alors la quille est considérée comme tombée.

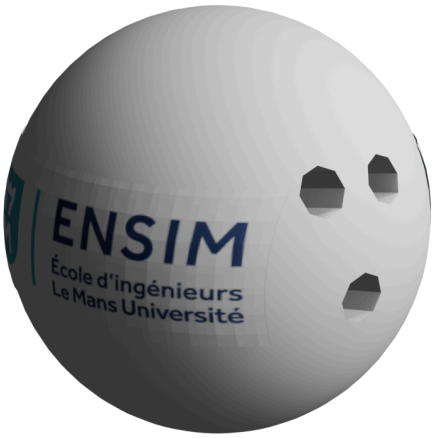
Replacer les quilles au bon endroit nous a posé des problèmes, car nous souhaitions tout d'abord supprimer l'objet entièrement, puis le recréer, comme au lancement du jeu, en s'aidant de son numéro pour savoir si la quille est à replacer ou non. Cette méthode avait pour inconvénient de ne pas toujours replacer les bonnes quilles au bon endroit, pour une raison qui nous échappe encore. Nous avons donc opté pour une solution plus simple et efficace. Si une quille est tombée, alors on désactive son mesh et son collider, si une quille est debout, on s'assure que son mesh et son collider sont activés, et toutes les quilles ont leur angle et position réinitialisés.

Ainsi, les deux solutions que nous avons trouvées à nos deux problèmes utilisent en partie le même élément : la sauvegarde de l'état initiale de la quille dans l'objet lui-même.

Modèles 3D utilisés

Panneau des scores

Boule de bowling



Quille

Sources

[1] https://www.youtube.com/watch?v=r2oXsHWrDhk , par Niko de Corridor Crew