RÉALITÉ AUGMENTÉE : LES TOURS DE HANOI

Simon Le Floch, Iwan Derouet

Contents

1. Introduction

Le jeu des tours de Hanoi est un jeu qui se joue avec 3 cylindres verticaux (les tours) et 4 disques toriques de tailles toutes différentes. Au début du jeu, les 4 pièces sont empilées sur l'une des tours, de la plus grande à la plus petite. Le but du jeu est de décaler l'empilement sur un autre cylindre, en respectant les règles suivantes : Une pièce ne peut être posée que sur une tour vide ou sur une pièce plus grande qu'elle. Ce jeu n'est ni particulièrement difficile à résoudre ni difficile à implémenter en soi. Tout l'intérêt de ce projet vient de la contrainte de le faire en réalité augmentée (AR). Le principe repose donc sur le fait d'avoir des éléments réels filmés et détectés par l'appareil faisant tourner le jeu, qui sont retransmis à l'écran et par dessus lesquels viendront s'afficher les éléments du jeu. Le programme doit donc reconnaître des marqueurs et gérer l'affichage et la logique du jeu selon leur position. Ce rapport analyse donc notre processus de développement, nos idées, nos choix et nos résultats.

2. Étapes de développement

La première étape du développement consiste en le choix de l'environnement de travail. Nous avons choisi d'utiliser AFrame, permettant un set-up rapide et un déploiement cross plateforme facile, avec la bibliothèque AFrame-AR. L'étape suivante est le choix du type de gameplay, duquel dépendra l'essentiel du code. Le point qui restera constant tout au long du projet est le fait que la base des tours de Hanoi est fixée sur un marqueur

Hiro. Le reste, en revanche, a connu plusieurs itérations. L'idée sur laquelle nous sommes initialement partis était d'avoir un plan 2D vertical dans lequel les pièces pourraient se déplacer librement. Celles-ci seraient soumises à la physique, et leur déplacement serait géré par un clic sur l'écran, qui permettrait de sélectionner la pièce visée et de la déplacer en projetant la position du doigt ou du curseur, selon l'appareil, sur le plan 2D. Les tours étaient affichées à partir d'un modèle 3D intégré à la scène, et les prototypes des pièces n'étaient que de simples cubes. Cependant, nous avons eu de nombreux déboires avec cette tentative, notamment causés par des versions de bibliothèques incompatibles entre elles, ainsi que des difficultés pour gérer correctement la projection du curseur sur le plan 2D souhaité. La physique marchait toutefois partiellement.

Contraints d'abandonner cette idée, nous avons brièvement exploré une seconde version, dans laquelle nous aurions eu un 2e marqueur pour la main du joueur. Lorsque celui-ci entre en contact avec une tour celle-ci est sélectionnée, lorsqu'une 2e tour est choisie une pièce est déplacée, si possible, de la première vers la deuxième. Cependant, nous avons simultanément regardé une troisième idée que nous avons retenue, qui est une version simplifiée de la première : un curseur est positionné en permanence au centre de l'écran. Lorsque celui-ci, par raycast, croise une tour celle-ci est sélectionnée. Lorsqu'une deuxième est choisie, le transfert de pièce a lieu de la première vers la deuxième instantanément.

3. Solution retenue

La solution retenue, dont le principe est expliqué au paragraphe précédent, fut codée en 4 fichiers. `index.html` contient la définition des éléments de la scène : 4 pièces toriques, 3 cylindres verticaux, un viseur, un plan servant de base, et une caméra. Pour des raisons peu claires, nous observons que le curseur proposé par a-frame ne nous permet pas d'avoir un gameplay fonctionnel. En effet, son raycast ne se fait pas toujours correctement, ce qui conduit à des tours parfois non détectées lorsqu'elles le devraient et inversement. Par conséquent, le viseur final n'est qu'un cercle dessiné au centre de l'écran en css, et le raycast est fait de façon classique depuis le centre de la caméra, dans un des fichiers suivants.

Les trois autres fichiers, en JavaScript, contiennent le code des trois principaux composants. On trouve tout d'abord les pièces, possédant 2 fonctions : une permettant de leur donner une position par défaut, et une permettant de les positionner selon la tour sur lesquelles elles sont et leur place sur la tour en question. Le deuxième composant est celui des tours. Celui-ci gère les pièces possédées par chaque tour ainsi qu'une partie de la logique d'échange de pièces. Chaque tour a donc à sa disposition une liste de longueur 4 indiquant les pièces en sa possession. Chaque entrée correspond à un emplacement possible. La valeur de l'entrée vaut -1 si celle-ci est vide, ou un nombre entre 0 et 3 pour indiquer la présence d'une pièce. Le nombre indique la taille de la pièce, une pièce plus petite aura une valeur plus grande. Les tours ont également accès aux fonctions `Select` et `Deselect`, qui n'ont pour effet que de changer leur couleur en vert en cas de sélection. Une fonction `updateDisplay` parcourt la liste de pièces et appelle la fonction d'affichage des pièces correspondantes, pour les placer sur la tour à la bonne hauteur. La fonction

`TrySendPiece` regarde s'il est possible d'enlever une pièce (ce qui est toujours le cas si la tour en possède au moins une). Si oui, la pièce est enlevée et la fonction renvoie le numéro de la pièce. Sinon, -1 est retourné. Enfin, la fonction `TryPlacePiece` essaie de rajouter une pièce à la tour, le fait si cela est possible selon les règles du jeu et renvoie le numéro de la pièce en cas de réussite, -1 sinon. Le troisième composant, check-in-viseur, était initialement simplement destiné à lancer un raycast depuis le curseur et à appeler les fonctions `Select` et `Deselect` des tours au besoin. Il s'occupe finalement également d'une partie de la logique du jeu, en appelant les fonctions d'échange de pièces des tours lorsque 2 d'entre elles ont été sélectionnées. La fonction `tryPieceTransfer` appelle les fonctions d'envoi puis de réception de pièces des tours, et annule le transfert si ce dernier n'est pas faisable.

4. Résultats et améliorations possibles

Notre implémentation permet bel et bien de jouer au jeu du début jusqu'à la fin. Le marqueur est détecté, les tours sont correctement sélectionnées au survol par le curseur, et les pièces sont bien déplacées tel que nous l'avions prévu. Si ces tests sont concluants sur ordinateur ainsi que sur tablette, nous n'avons en revanche pas réussi à obtenir une détection correcte du marqueur sur android, sans avoir de piste pour la cause. Il n'est pas impossible que la version de la bibliothèque utilisée pour AFrame ou pour l'AR soit elle-même en cause.

FIGURE 1. État du jeu au début de la partie

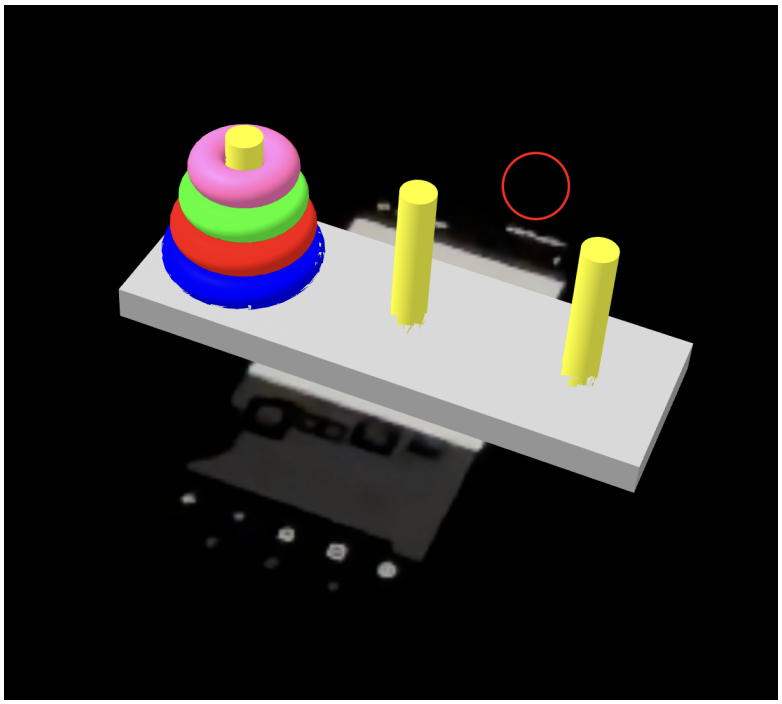


FIGURE 2. État du jeu en cours de partie. La tour de droite est sélectionnée

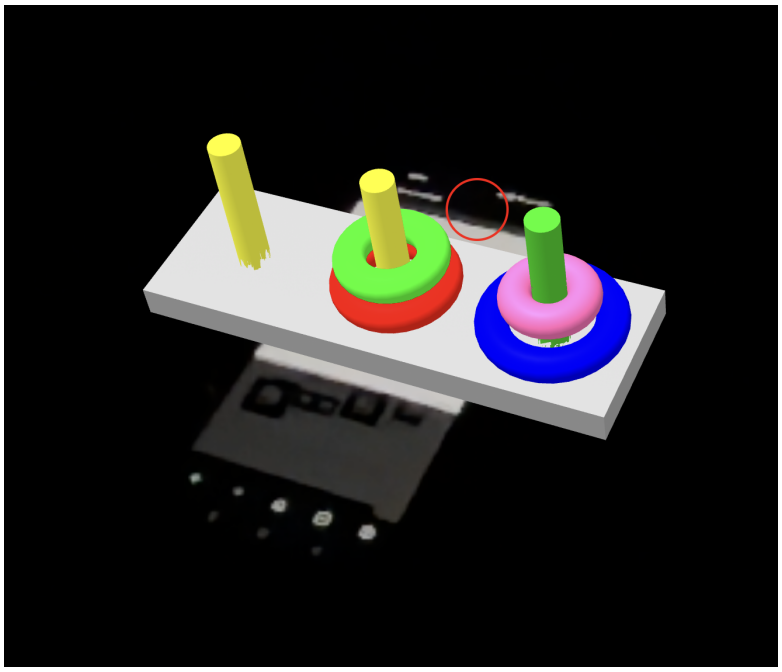
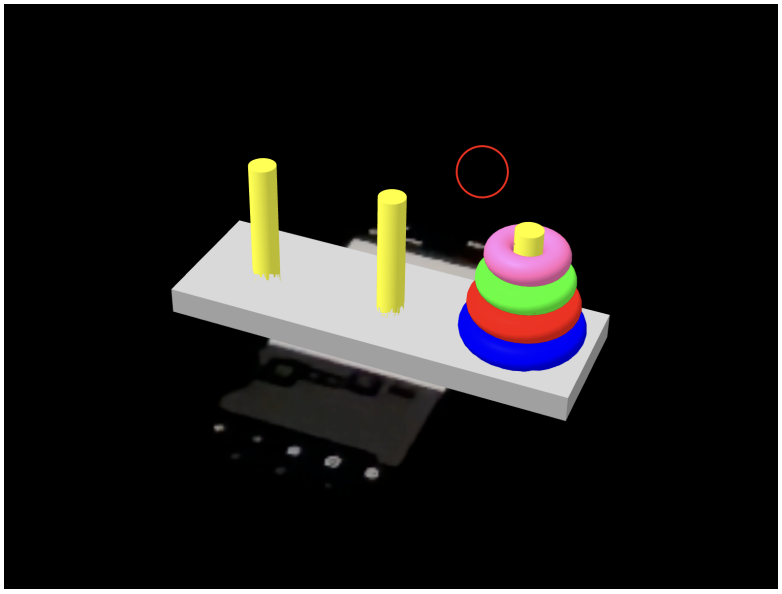


FIGURE 3. État du jeu en fin de partie



Par ailleurs, il est certain que des améliorations sont possibles. Tout d'abord, l'implémentation des différentes fonctions a été faite assez rapidement, avec pour seul but de fonctionner correctement. Les performances n'ont pas été prises en considération et peuvent certainement être améliorées, telles que dans la fonction `updateDisplay`, qui fait 2 boucles `for` lors de son exécution. Il est également possible d'imaginer une version plus proche de l'idée d'origine, dans laquelle une tour sélectionnée voit sa pièce la plus haute suivre le curseur, permettant de mieux voir le déplacement et donc une meilleure

compréhension de la situation par l'utilisateur. Nous pourrions également imaginer une version faisant intervenir plus de marqueurs, avec possiblement un marqueur par pièce ou bien un marqueur par tour. Cependant, cela risque de rendre difficile la gestion des différentes actions possibles, et le risque de ne pas pouvoir progresser car au moins un marqueur n'est pas correctement détecté est important.

Un autre point d'amélioration possible serait l'ajout d'une véritable interface. Cela permettrait d'offrir aux joueurs la possibilité de configurer, par exemple, le nombre de tours et de pièces totales, ainsi que de donner un moyen de réinitialiser la partie autre que recharger la page web. Cela permettrait de plus d'augmenter les retours faits à l'utilisateur, en indiquant clairement par exemple si un mouvement n'a pas eu lieu parce qu'il est contraire aux règles du jeu, ou afficher clairement la victoire et le nombre de coups effectués, pour permettre ensuite d'enregistrer un score à battre.

5. Conclusion

En conclusion, nous avons pu explorer un certain nombre de façons d'interagir avec un programme en réalité virtuelle, et nous sommes parvenus à créer une expérience simple en AR permettant de jouer et de résoudre le problème des tours de Hanoi. Ceci étant, l'implémentation que nous avons choisie est assez basique, et il existe de nombreuses façons de l'améliorer. Il serait également intéressant d'explorer les possibilités de porter ce type de programme sur des plateformes très différentes, telles que des casques de réalité augmentée, et d'étudier les interactions possibles avec ce genre d'appareils.

---

5 janvier 2025

Simon Le Floch, Iwan Derouet