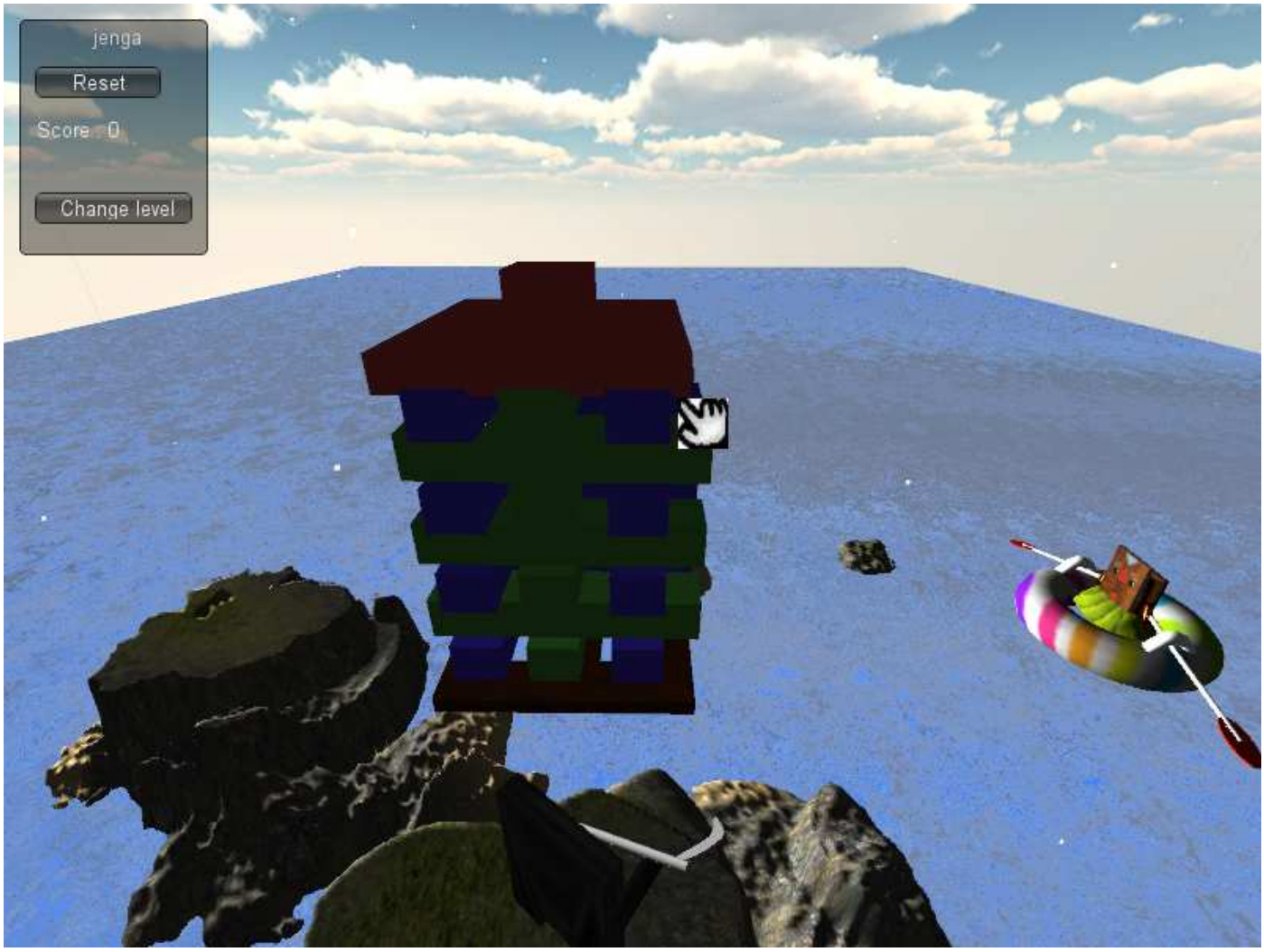
A2010

Adrien SERIN

Henri PAYNO

Coralie VESSIERES



[RAPPORT BRAS HAPTIQUE]

Création d'un jeu de type JENGA

SOMMAIRE

INTRODUCTION

Dans le cadre de l'UV VI50 nous avons fait un projet sur le thème du bras haptique. Pour utiliser au mieux toutes les possibilités de ce bras nous avons modélisé un jeu de JENGA. Le but est de créer une interaction sensori-motrice entre l'homme et la machine pour obtenir une immersion à la fois visuelle, tactile et sonore.

Le principe du jeu est de déplacer à l'aide du bras haptique les blocs d'une tour sans faire tomber celle ci. Un retour d'effort est fourni par le bras lors du déplacement de ces blocs.

Cette idée est inspirée du jeu Boom Blox sortit sur Wii.

I. CREATION D'UNE DLL

1. INTRODUCTION

Afin d'interagir avec le bras haptique dans le logiciel Unity, nous avions besoin d'une DLL. Celle ci a été créée à l'aide de visual studio et des drivers fournis avec le bras haptique. La DLL nous donne la possibilité de récupérer les déplacements avec le bras mais aussi de pouvoir lui appliquer diverses forces.

Afin de faire fonctionner le bras haptique le développement d'une dll est nécessaire. Pour cela on se base sur les modèles architecturaux proposés par Sensable.

La dll est développée en C++ et contient les fonctionnalités de base :

- position du bras, en absolu et en relatif (différence depuis la dernière position).
- Affecter des forces au bras haptique (en x, y et z).
- Obtenir la valeur des boutons.

Afin d'utiliser la dll il faut initialiser le bras haptique (via la fonction init de la dll) . Cette fonction lance une routine de mise à jour des attributs du bras (sur chaque frame) ainsi qu'un scheduler associé au bras haptique. Pour obtenir la valeur des attributs il suffit de faire appel aux getters proposé par la dll.

Lorsque l'on quitte l'application il faut libérer le bras haptique via une fonction (exitHD). Elle termine le scheduler.

2. RESULTAT DE LA DLL

La dll répond aux attentes et est assez stable. Elle peut facilement être réutilisable puisque les fonctions créées sont assez génériques. L'importation des DLL sous Unity se fait uniquement sous la version pro. Il suffit ensuite d'ajouter un import dans un script C# pour utiliser une fonction créée dans cette DLL.

II. DEVELOPPEMENT

1. CAMERA

Deux caméras ont été implémentées : une dans le plan fixe pour déplacer le curseur. Et une pour roter autour de la construction. Pour cette dernière l'angle de rotation autour de x (déplacement haut/bas) a été limité afin d'éviter les problèmes de « gimble lock » et surtout d'empêcher l'utilisateur de se situer en dessous de la carte de jeu.

2. SCRIPTS

Les scripts ajoutées ont étés développés en C# puisqu'ils incorporaient l'importation de la dll crée.

Script développé :

- OrbitHD : permet la rotation autour de la tour par le bras haptique (basé sur mouseOrbit).
- DragRigidbodyWithHD: version de DragRigidbody pour le bras haptique (permet un dragAndDrop en 3D).
- GUIScript : script pour l'environnement graphique (boutons, score et pointeur).
- GameScript : initialisation du bras haptique et fermeture + désinitialisation du bras haptique.

D'autres scripts sont utilisés pour les animations ou l'effet d'eau notamment.

III. ENVIRONNEMENT DU JEU

Le jeu est composé de plusieurs scènes jouables contenant une tour de JENGA chacun. Comme présenté dans le cahier des charges, le décor est non réaliste.

1. MODELISATION

Tous les terrains sont modélisés sous Unity, de même que les effets d'eau et que la skymap. Tous autres objets ou animations sont créés sous 3DS max et importés sous Unity. Les animations sont donc des scripts qui tournent en boucle et indépendamment de Unity.

Chaque tour est composé de deux parties obligatoires. Un socle qui permet de surélever la tour pour mieux se déplacer autour, et d'un poids qui représente le toit de la tour.

2. INTERFACE

Une interface de compteur de points a été créée pour le jeu. Ce compteur change lorsque qu'un bloc touche le sol. Si ce bloc est celui qu'on vient d'enlever on a une incrémentation du compteur, si au contraire le bloc tombe de lui même de la tour il décrémente le compteur.

Des boutons pour changer de niveau et recommencer le niveau ont étés ajoutés dans l'idée de pouvoir les cliquer via le bras haptique. Mais nous n'avons pas eu le temps de développer cette fonctionnalité.

3. PHYSIQUE

Pour implémenter la physique sous Unity on utilise les options Physx du logiciel. Pour cela chaque bloc de la tour possède un rigidbody. Ce rigidbody est sensible à la gravité. Chaque bloc a des propriétés physiques tel que sa texture (bois, métal, plastique, glace), sa masse et un coefficient de réaction au frottement. Chaque élément du décor possède une box ou un mesh collider pour indiquer que cet objet est non traversable.

IV. GAMEPLAY

1. MODE CAMERA

Avec le bras haptique on peut déplacer la caméra pour tourner autour de la construction pour avoir un meilleur point de vu. Ce mode est activé en appuyant sur le bouton gris clair du bras haptique.

Si aucun bouton n'est pressé on est en mode premier plan, c'est à dire que la caméra se déplace uniquement dans le viewport.

2. MODE SAISIE

Lorsqu'on appuie sur le bouton gris foncé on passe en mode saisie. Le bloc se trouvant alors devant le curseur est saisie. Celui ci suit alors les mouvements fait avec le bras haptique. Lorsque le bouton est relâché le bloc n'est plus sous le contrôle du bras.

Chaque bloc qui peut être pris possède le script « dragAndDropHD ». Ce script permet d'appliquer des transformations au bloc par rapport au bras haptique en trois dimensions.

Lorsque l'utilisateur se saisit d'un bloc une force est ajoutée au bras haptique afin d'assurer l'immersion.

V. PROBLEMES RENCONTRES

L'un des problèmes majeur est la synchronisation entre le bras haptique et Unity. Ce problème apparait aléatoirement et rend le bras inutilisable ce qui amène une obligation de redémarrer l'application. Toutefois ce problème a aussi été rencontré dans les exemples fournit par SenSable. Ce qui nous fait penser que le problème doit venir du lien entre l'ordinateur et le bras haptique.

La dll a été complexe à mettre en place dû au manque d'exemples adaptés à notre projet.

L'incrémentation et la décrémentation du score ne marchent pas tout le temps. Si on fait tomber la tour en entier notre score négatif n'est pas égal au nombre de blocs tombés. Lors de la saisie d'un bloc la fonction de raytracing peut parfois sélectionner plusieurs blocs à la fois avant d'en garder un seul pour le dragAndDrop, on se retrouve alors avec des blocs qui sont compté en positif alors qu'ils ne le devraient pas.

VI. AMELIORATION

Tout d'abord régler les problèmes du score.

Puis l'ajout de niveaux supplémentaires seraient un plus (dont un mode construction). On pourrait aussi concevoir des blocs de composition différente (glace, gelée…).

De plus des animations plus complexes auraient pu être ajoutées en début et en fin de partie.

D'un point de vu graphique on pourrait calculer les ombres.

Pour améliorer l'immersion sonore, des sons pourraient être associé à la saisie des personnages.

Une caméra libre pourrait être ajoutée. Toutefois sa conception d'un point de vu ergonomique est très complexe. En effet le bras haptique ne compte que deux boutons et l'activer lorsque ces deux boutons sont pressés nous a paru être une mauvaise idée.

CONCLUSION

Le jeu du jenga est utilisable et remplit les conditions établies dans le cahier des charges.

Unity3d s'est avéré riche en fonctionnalités. Ce qui nous a été profitable notamment pour l'aspect physique.

L'immersion sensori-motrice de l'application est de bonne qualité même si le retour de force pourrait être un peu amélioré. Toutefois l'application est assez cohérente dans sa réalisation.