IUT Lyon 1
Département Informatique

Rapport de Projet Tutoré

Jeu Minetest
Course Cubique

Groupe :

- PELOUTIER Maxime
- SAINT-SORNY Evan
- STCHETININE Océane
- TON Hoang Chi

Tuteur :

- CHAMPIN Pierre-Antoine

Intitule du Projet :

Jeu Minetest

Descriptif du sujet :

Notre sujet consiste en l'utilisation du moteur Minetest, un moteur 3D open source inspiré du jeu-vidéo Minecraft, dans le but de créer une application 3D originale en étendant le moteur.

Version de Minetest :

Windows 5.1.0 (sortie le 12/10/2019)

Gestion de version :

GIT hébergé sur la forge de l'université https://forge.univ-lyon1.fr/

Licence :

Code : LGPL 2.1

Éléments artistiques : CC-BY-SA

Résumé du jeu :

Course Cubique est un ensemble de 2 mini-jeux basés sur le mouvement, Square **Mile** et **Splash Block**. Square Mile est un parcours dont le but est d'arriver du début à la fin le plus vite possible et Splash Block est un mini-jeu d'agilité où le joueur doit sauter de plateforme en plateforme sans tomber dans l'eau le plus longtemps possible. Ces deux mini-jeux seront situés dans le même monde et accessibles directement depuis un hub central.

Table des matières

I. Introduction

Tout d'abord, bonjour à vous qui lisez ce rapport,

Le projet que nous allons aborder à travers la suite de ces pages reste un projet qui a eu des rebonds dans tous les sens du terme, malgré tout, nous sommes assez fiers de pouvoir le présenter.
Alors de la part de toute l'équipe, Bonne lecture !

-L'équipe de Course Cubique.

II. Présentation du Projet

A. Contexte du Projet

Notre sujet consiste à créer un jeu sur le moteur Minetest. Il s'agît d'un moteur 3D en « Voxel » inspiré du jeu Minecraft. Ce moteur étant open-source et scriptable dans le langage Lua, nous devons l'exploiter afin de créer un jeu original en y ajoutant nos propres fonctionnalités.
Nous avons choisi ce sujet pour plusieurs raisons, la première étant que certains membres du groupe considèrent le développement de jeux-vidéo dans leur carrière professionnelle. De plus, le langage Lua se différencie considérablement des langages de programmation que l'on voit en cours (C, Java, etc.). Il est donc intéressant d'apprendre à l'utiliser, et cela dans un moteur 3D.

B. Description brève du jeu

Nous avons décidé de créer le jeu **Course Cubique**, un ensemble de deux mini-jeux, situés dans le même monde prédéfini, qui pourront être lancés depuis un même hub central où arrivera initialement le joueur. Ainsi ne sera pas nécessaire de retourner sur le menu principal de Minetest pour changer de mini-jeu.

Le premier mini-jeu, nommé **Square Mile** consiste à aller d'un point A à un point B à travers différents obstacles, bonus et malus, dans le but d'avoir le meilleur temps.

Le deuxième mini-jeu se nomme Splash Block. Dans ce jeu, le joueur devra éviter de tomber en sautant sur des plateformes qui disparaissent peu de temps après être passé dessus, dans le but de tenir le plus longtemps et faire le plus de sauts possibles.
Au cours du déroulement de la partie, le jour rencontrera des blocs boots (des blocs glissants ainsi que des blocs de tremplins (propulsions), ce qui lui compliquera la tâche.

III. Environnement Technique

A. Moteur du jeu

Le sujet impose d'utiliser Minetest. Ce moteur 3D est gratuit, open-source et inclut une API Lua pour développer des mods, c'est-à-dire un ensemble de scripts qui permettent
D'ajouter ou de modifier des fonctionnalités du moteur, telles que des types de blocs, des mécaniques de jeu, des interfaces, etc.

Dans le cadre de Minetest, un « jeu » est un ensemble de mods. Nous allons nous appuyer sur **Minetest Game**, le jeu inclus dans l'installation de base du moteur, ce qui va nous permettre d'avoir des objets, mécaniques et fonctionnalités basiques. Nous pourrons également inclure et/ou s'appuyer sur des mods créés par la communauté de Minetest à notre jeu, si les licenses de ces mods le permettent, pour ajouter des fonctionnalités complexes à développer mais utiles dans le cadre de notre jeu.

B. Langages de programmation

Le moteur en lui-même a été écrit en C++, mais pour des raisons de complexité et de compatibilité avec les autres mods, nous n'allons pas le modifier.

L'API de modding de Minetest utilise le langage de scripting Lua, qui est donc le langage que nous allons utiliser pour créer notre jeu (www.lua.org).

C. Gestion de Version

Nous avons choisi d'utiliser le système de gestion de version Décentralisé Git pour notre travail collaboratif. Nous estimons que ses fonctionnalités telles que l'historique de versions, les Branches, et un dépôt distant de référence commun sont indispensables pour mener à bien notre projet et faciliter la distribution des itérations du jeu.

Au cours du premier et deuxième semestre, nous avons appris à utiliser les commandes Git ainsi que l'utilisation de GitLab sur la Forge mise à disposition par l'Université Lyon 1. Nous allons donc notre projet sur la Forge GitLab de l'Université.
Une des raisons pour lesquelles nous avons choisi GitLab est la gratuité de la création de dépôts privés, contrairement à GitHub qui est limité à 4 collaborateurs sur un dépôt privé sans version payante.

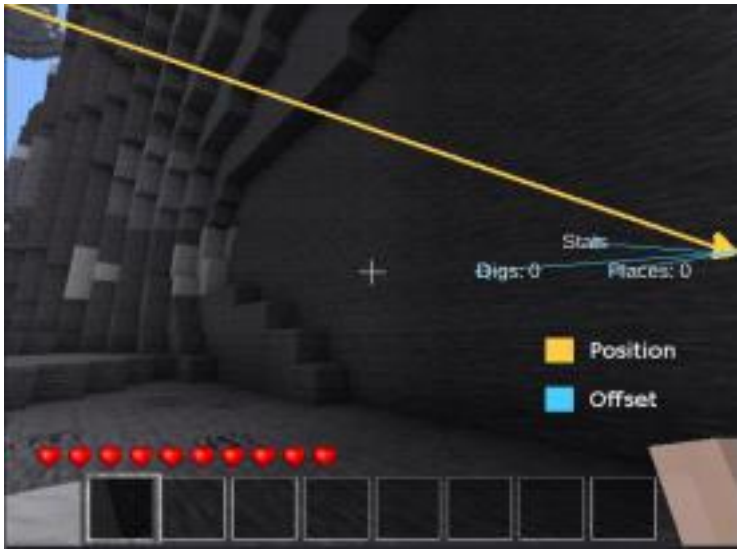
D. Possibilités et limitations de l'API de modding

Minetest offre des possibilités très intéressantes dans le cadre de la création de nos mini-jeux grâce à son API. Voici quelques exemples extraits de **Minetest modding book**, écrit par **rubenwardy** :

- Modifier la physique du joueur en surchargeant ses valeurs de base :

```
minetest.register_chatcommand("antigravity", {
    func = function(name, param)
        local player = minetest.get_player_by_name(name)
        player:set_physics_override({
            gravity = 0.1, -- set gravity to 10% of its original value
                           -- (0.1 * 9.81)
        })
    end,
})
```

Intégrer des éléments à l'interface du joueur, par exemple pour afficher des compteurs de temps ou de score :



- Créer des nouveaux blocs avec leurs propres propriétés et apparences :

Cependant, cette API comporte des limitations qui rend difficile l'implémentation de certaines fonctionnalités, notamment :

- La création d'animaux vivants et de PNJ avec lesquels interagir ;
- La création et la conduite de véhicules.

IV. Point Technique

A. Splash Block

1. Mécaniques d'apparition du joueur

Lorsqu'il lance Splash Block depuis le menu principal ou qu'il réapparaît à la fin du mini-jeu, le joueur sera toujours placé à la même position. Cependant, sa caméra sera positionnée à un endroit différent s'il apparaît pour la première fois ou s'il réapparaît après avoir terminé une partie.

En pratique, cela signifie que dans le premier cas, il regardera dans la direction d'un panneau en métal qui affiche du texte indiquant le but du jeu et sera incité à regarder les panneaux en bois en-dessous, qui eux affichent du texte sur les effets des plateformes situées dans l'arène, pour ensuite regarder derrière-lui, vers la direction de la ligne de départ. Dans le deuxième cas, il regardera directement dans la direction d'un panneau qui lui indique d'aller sur la ligne de départ pour commencer.

Figure A-1.1 : Vue subjective du joueur lors de sa première apparition.

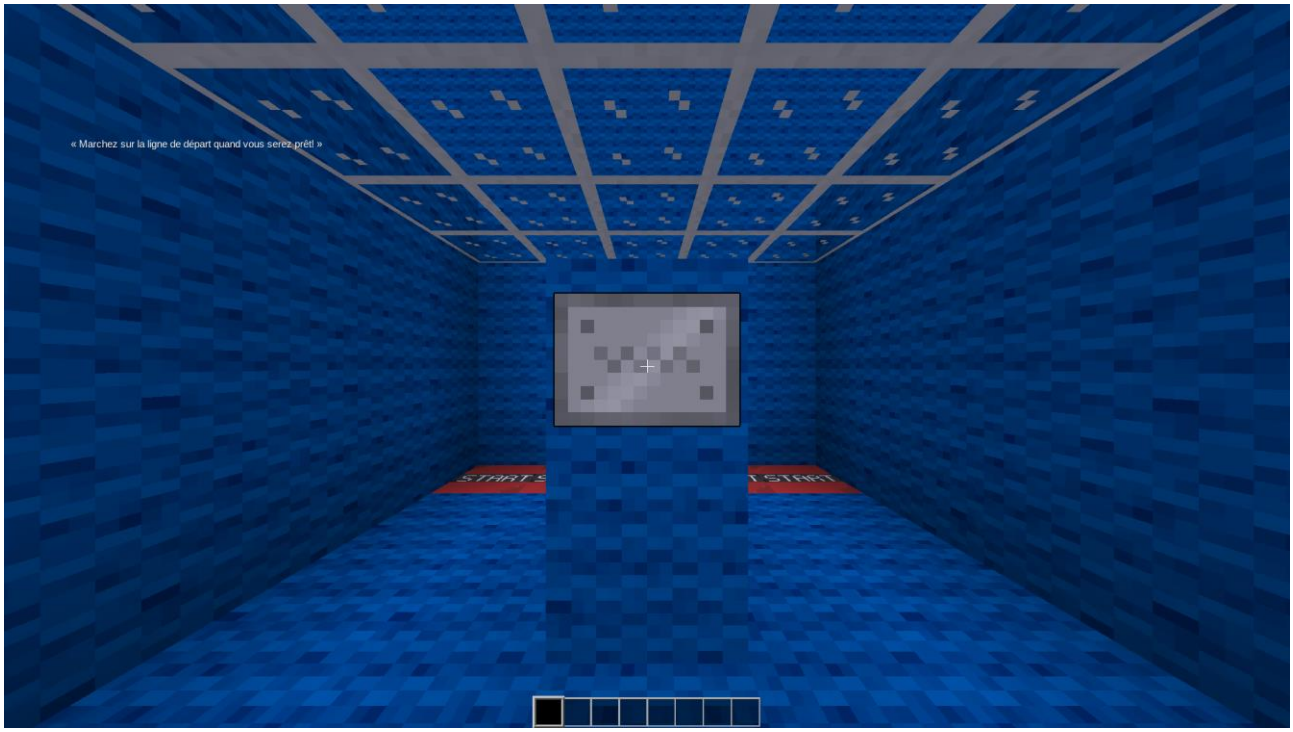


Figure A-1.2 : Vue subjective du joueur lors de sa réapparition après la fin du mini-jeu.

2. Bloc de départ

Le type de bloc constituant la ligne de départ effectue plusieurs actions lorsque le joueur marche dessus :

- Le joueur est téléporté au-dessus de l'arène et commence à tomber.
- Un chronomètre est lancé pour mesurer le temps de « survie » du joueur, ainsi qu'un compteur de plateformes sur lesquelles le joueur est passé.
- La caméra du joueur est positionnée vers le bas pour qu'il puisse observer les plateformes dès le départ.
- L'effet de la gravité subi par le joueur est grandement réduit, ce qui lui permet de prendre plus de temps pour atterrir une première fois sur une plateforme et ne pas échouer directement en tombant brusquement dans l'eau.
- Un compteur de score lié au temps de « survie » apparaît en haut à droite de l'écran du joueur.

Figure A-2.1 : Ligne de départ, qui lance le mini-jeu lorsque le joueur marche dessus.

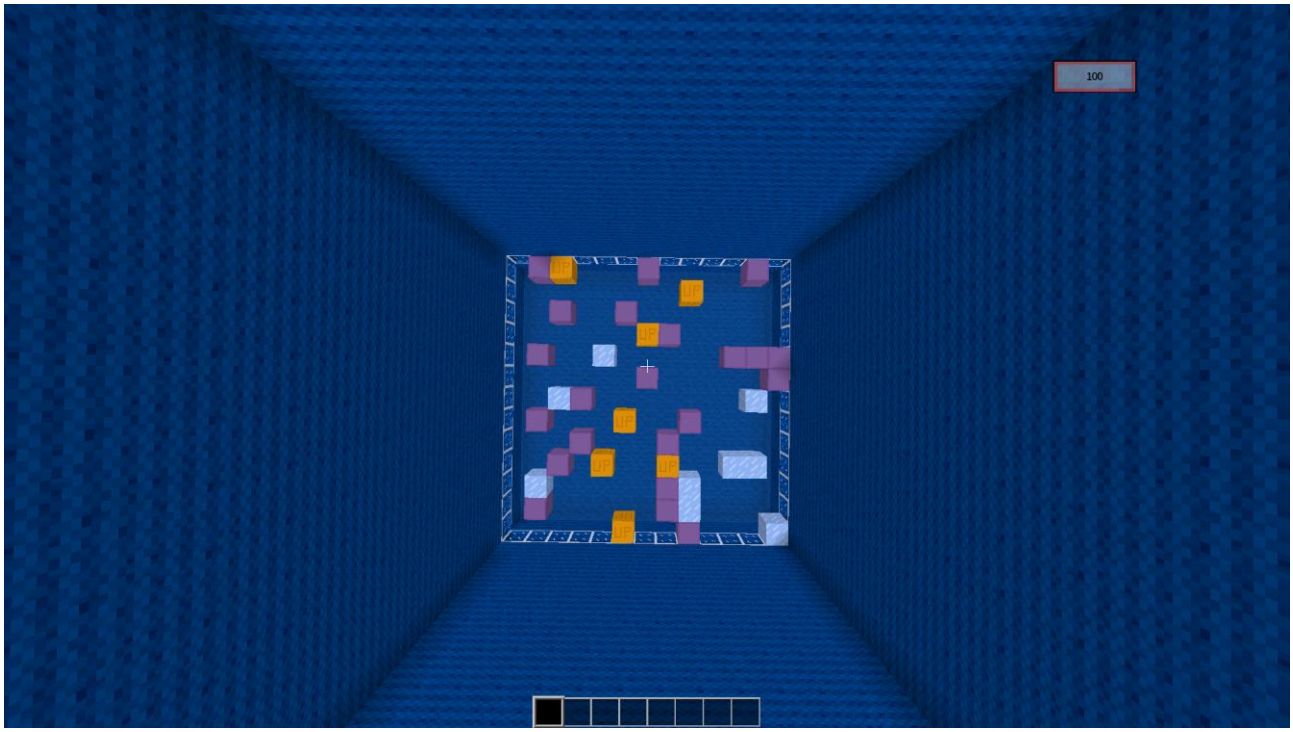


Figure A-2.2 : Vue subjective du joueur peu après avoir marché sur la ligne de départ.

3. Plateformes

L'arène est constituée de trois type de plateformes :

- Des plateformes normales violettes, sans propriétés physiques particulières.
- Des plateformes de glace glissantes, qui perturbent les mouvements du joueur.
- Des plateformes de propulsion, qui envoient le joueur vers le haut en lui ajoutant une grande quantité de vélocité verticale.

Quand le joueur marche sur n'importe laquelle d'entre elles, elles sont remplacées par une version transparente du bloc correspondant, pour indiquer qu'elles sont sur le point de disparaître (les versions transparentes restent solides). Le compteur de blocs sur lesquels le joueur passe est également incrémenté. Pour limiter l'avantage des propulsions au joueur, la version transparente des plateformes de propulsion n'envoie pas le joueur vers le haut.

Après quelques secondes, le jeu sélectionne tous les blocs non occupés par une plateforme autour du bloc sur le point de disparaître, et choisit un de ces blocs aléatoirement pour y placer un bloc du même type que le bloc disparu. Cela permet à l'arène de changer constamment pour que le joueur reste actif, tout en évitant de remplacer une plateforme existante par une autre ou de placer une plateforme en dehors de l'arène. Ainsi, il y a toujours la même quantité de blocs de chaque type au sein de l'arène, et le joueur peut réutiliser ces blocs pour créer sa propre arène.

Le reste des blocs sont des blocs fournis de base par Minetest et servent à délimiter l'arène.

Figure A-3.1 : Les trois différents types de plateformes et leurs versions transparentes. De gauche à droite : bloc de propulsion, bloc normal et bloc glissant.

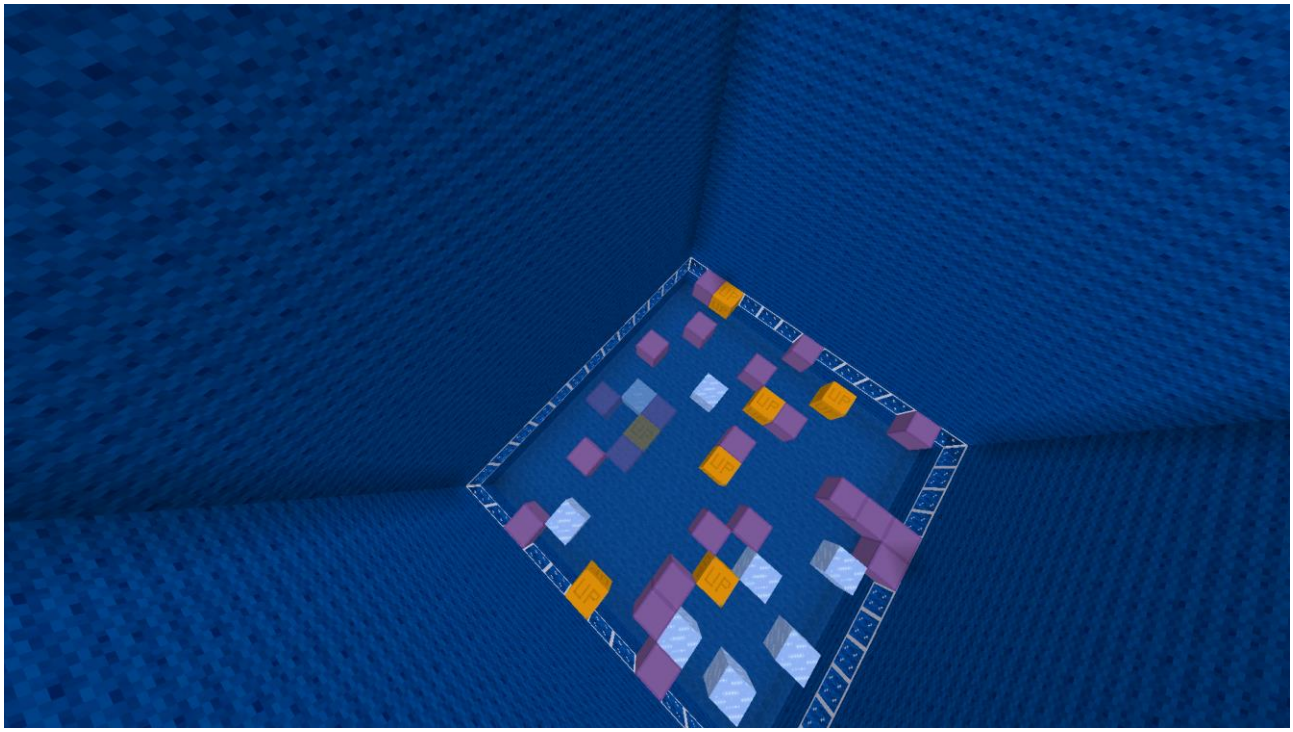


Figure A-2.2 : Vue subjective du joueur (interface désactivée pour plus de visibilité) en hauteur après avoir été propulsé.
On peut y voir des plateformes transparentes sur le point de disparaître.

4. Fin du mini-jeu

Lorsque le joueur tombe dans l'eau, plusieurs actions sont effectuées :

- Le chronomètre est arrêté et sa valeur est récupérée.
- Le compteur en haut à droite de l'écran disparaît.
- Une fenêtre apparaît au-dessus du centre de l'écran pour indiquer au joueur le temps qu'il a tenu sans tomber, le nombre de blocs sur lesquels il est passé, et le score total, calculé en fonction de ces deux facteurs.
- Après quelques secondes, cette fenêtre disparaît et le joueur réapparaît au point de départ (voir fig. A-1.2).
- Les compteurs de temps et de blocs sont réinitialisés.

Figure A-2.2 : Affichage de la fenêtre sur l'écran du joueur à la fin du mini-jeu

5. Limitations

Nous nous sommes confrontés à quelques limitations de l'API de Minetest lors de la réalisation de ce mini-jeu, difficiles à contourner à notre niveau :

- Il est très difficile d'observer les mouvements du joueur avec précision, notamment pour savoir quand il passe sur un bloc en particulier.
- Nous n'avons pas pu afficher un chronomètre directement sur l'écran du joueur.
- La taille du texte apparaissant à l'écran n'est pas ajustable, ce qui cause des problèmes de visibilité.

6. Mods de la communauté utilisée

Pour contourner une partie de ces limitations pour la réalisation de ce mini-jeu, nous avons utilisé quelques mods de la communauté :

- **Poschangelib** (https://framagit.org/karamel/poschangelib), utilisé pour ajouter des « écouteurs » aux blocs permettant de détecter et récupérer les informations du joueur qui passe dessus.
- **Timer** (https://github.com/minetest-mods/timer), utilisé pour compter le temps de survie du joueur.
- **Superflat** (https://github.com/srifqi/superflat), utilisé pour générer un monde plat sur lequel nous avons construit notre zone de jeu.

B. Square Miles

1. Mécanisme d'apparition du joueur

Le joueur comme dans Splash block est toujours téléporté au même endroit de la map. La map est située sur une plaine avec à côté le parcours, mais ne sera pas limité à ce seul parcours : Il aura la possibilité de créer lui-même son propre parcours. Toutefois, des moyens sont donnés afin d'encadrer les parcours.

2. Changements par rapport à Splas hBlock

- Le mod bloc_timer a été modifiée afin de permettre une décrémentation du score.
- L'ajout des blocs de boost
- La structure d'empilage des blocs qui sont différents car on crée plusieurs types de parcours.

3. Blocs de boost

- Bloc de vitesse permettant de réécrire temporairement la stat de vitesse afin de donner au jour une allure plus rapide

- Bloc de saut permettant de réécrire temporairement la statistique de saut et de sauter plus haut pendant un temps donné.

- Bloc de gravité permettant de au joueur de modifier sa gravité permettant des saut plus longs.

4. Structure du parcours

Nous avons proposé qu'un seul type de parcours. Mais le joueur a la possibilité de créer d'autres parcours en fonction de sa créativité.

5. Fin du mini-jeu

Le mini-jeu se finit lorsque le joueur atteint la ligne d'arrivée. Et reçoit les mêmes informations que dans le splash block.

6. Limitations du jeu

- La caméra peut se coincer dans les zones restreintes, perturbant la perception dans l'espace du joueur.

- Il est difficile de s'orienter rapidement et avec précision à la souris.

- Il y a un manque de sons, ce qui diminue l'expérience du joueur.

- Un contrôle rigide du joueur

V. Retour sur le cahier des charges du s2

A. Organisation du Projet

Notre projet est divisé en lots de fonctionnalités, définies pour chaque mini-jeu et Course Cubique en général, à créer et à rendre selon des échéances durant le semestre 3. A la base le projet devait être fonctionnel pour le début de s4, malheureusement dû à un contre temps et à une perte d'une partie de l'équipe on a dû refaire le cahier des charges au niveau des différents lots, des équipes et des deadlines.
Au final, étant 4, nous avons divisés par binôme pour chaque jeu. Du côté de Splash block le binôme est constitué de Maxime P et Océane S. Evan S ainsi que Hoang Chi T. sont sur le mini-jeu Square Miles.

B. Liste des lots

N° du lot / Intitule :
1. Environnement de base
2. Compteur de temps et de score
3. Disparitions / Apparitions des blocs
4. Types de blocs
5. Map de Jeu
6. Test

Lot n°1 : Environnement de base :

CC Bloc qui téléporte le joueur à une coordonnée quand il passe dessus.

Déclencheur de début et un déclencheur de fin du mini-jeu, intégrés dans un parcours avec des obstacles simples.
Déclencheur de début et un déclencheur de fin du mini-jeu, intégrés dans une zone de jeu avec des plateformes inactives.

Instructions d'installation de Course Cubique dans Minetest ainsi qu'un tutoriel du jeu dans un fichier texte README.

Lot n°2 : Compteur de Temps et de score

CC Empêchement du déclenchement d'un mini-jeu lorsqu'un autre est actif.

Compteur de temps et de score dans l'interface du joueur. Le score dépendra du temps réalisé entre le début et la fin du parcours.
Compteur de temps et de score dans l'interface du joueur. Le score dépendra du temps passé sans tomber et du nombre de sauts réalisés.

Lot n°3 : Mécaniques centrales

CC Disparition des plateformes après un certain délai quand le joueur passe dessus, et faire réapparaître un bloc aléatoirement dans la zone.

Lot n°4 : Mécaniques avancées

CC Types de blocs aux propriétés physiques différentes (glissants, rebondissant, etc.) et réapparition des blocs avec un type aléatoire et à une certaine distance d'une autre plateforme.
Tutoriels dans un menu accessible directement dans le jeu et informations sur les chemins alternatifs et les différents types de blocs.

Lot n°5 : Map du Jeu

CC Création de la Map avec le hub pour accéder à chaque jeu, et ainsi pouvoir commencer une partie de Course-Cubique.

Lot n°6 : Test

CC Tableau des scores pour chaque jeu et polissage de l'environnement visuel et sonore.
Finalisation de la documentation utilisateur ainsi que des différentes options.

C. Répartition de l'équipe & Tâches

Chef de Projet :

Océane S.

Tâches : répartir les tâches au sein du groupe et valider les modifications de ce document.

Splash Block:

Maxime P. (Lead Développer) et Océane.S

Tâches : encadrer le développement du jeu, être le référent en matière de code et apporter de l'aide aux autres développeurs.

Square Miles:

Hoang Chi T. & Evan.S

VI. Bilan Personnel

A. Maxime Peloutier

Malgré les nombreux imprévus et difficultés, j'ai trouvé ce projet très intéressant et instructif. J'ai appris le Lua, un langage de programmation dont la syntaxe et le fonctionnement sont considérablement différents de ceux vus en cours à l'IUT. J'ai également appris à travailler avec l'API du moteur Minetest, et en particulier, ses limitations. Le confinement m'a poussé à mieux utiliser Git et Gitlab pour la gestion de version, notamment travailler avec plusieurs branches et la résolution de conflits. Tous les imprévus lors du déroulement du projet m'ont forcé à savoir m'adapter à la situation et prendre des décisions par rapport à ce qui était prévu initialement.

Il était très intéressant d'apporter sa propre contribution à un projet déjà existant. Pouvoir s'appuyer sur l'API du moteur ainsi que celles de quelques mods créés par la communauté permet de ne pas avoir à partir de rien, ce qui a rendu le procédé plus agréable.

Cela dit, j'ai trouvé Minetest assez décevant quant à ses possibilités à mon niveau. Malgré le fait que Minetest est conçu pour le modding et que le moteur a certainement beaucoup de potentiel avec de l'expérience, j'estime que les fonctionnalités de base limitent un peu trop les ambitions. Si je devais refaire un projet personnel de modding, j'utiliserais sans doute le jeu dont il a été inspiré, Minecraft. Même si le langage de programmation est différent et en général moins utilisé dans l'industrie du jeu-vidéo de nos jours (Java), je suis déjà familier avec ce langage et la communauté de modding de Minecraft est colossalement plus grande que celle de Minetest en raison de sa popularité. Il devient alors plus facile de trouver des solutions à mes problèmes sur Internet. De plus, Minecraft est un jeu beaucoup plus complet de base et mis à jour régulièrement, ce qui donne une base plus solide pour s'appuyer. Je joue moi-même à beaucoup de jeux appartenant au genre « Sandbox », et Starbound et l'un d'entre eux. Contrairement à Minecraft et Minetest, s'agît d'un jeu en 2D, mais il utilise Lua comme langage de script et a un bon support de modding par les développeurs. Je songe également à me lancer dans un projet de modding de Starbound en utilisant ce que j'ai appris avec ce projet tutoré pour approfondir et varier mes connaissances et compétences.

J'ai trouvé que s'organiser en groupe a été très difficile. La nature de notre projet a fait que nous devions travailler sur des fichiers très proches les uns des autres, et il devient alors difficile de travailler à plusieurs sur une même fonctionnalité. En plus de difficultés de communication déjà présentes, deux membres ont quitté le groupe entre le S3 et le S4, donc nous avons dû revoir notre organisation et nous objectifs en grande partie.

Globalement, je reste plutôt satisfait du résultat du projet, et les connaissances et compétences que j'ai acquises auront beaucoup de valeur à mettre en avant pour un éventuel stage ou une poursuite d'études dans le domaine du jeu-vidéo.

B. Océane Stchetinine

J'aimerais tout d'abord remercier Maxime qui a été présent sans faille quand il le pouvait du début à la fin, et je pense que je n'aurai pas pu espère meilleur synchro que celle qui s'est passée pour ce projet.
C'est un projet que nous avons choisi pour son côté ludique (vis-à-vis de l'environnement de jeu, qu'on connaissait dans les grandes lignes par Minecraft) mais également par les possibilités que nous offrez le langage. Et pour tout dire nous n'en avons pas été déçu. Le Lua est un langage qui est fort intéressant à travers sa simplicité d'apprentissage et qui offre nombreux horizons et pouvoir s'approprie l'environnement de jeu comme nous le voulions était un petit objectif à atteindre dans nos quêtes de joueur. Du côté de l'API, je rejoins Maxime sur le fait qu'elle a des limites un peu trop restreintes par rapport à ce qu'on s'attend à pouvoir faire avec, et la communauté par sa taille ne peut pas être au rendez-vous, ce qui a rendu pas mal de moments un peu compliqué, surtout sur les détails.
Malgré tout, ce projet nous a beaucoup appris à devoir nous organiser avec des deadlines bien définie et beaucoup de personnes simultanément, ce que j'avais déjà pu entrevoir en association, mais vraiment mettre en place avec le PTUT. Devoir canaliser 5 personnes en même temps était compliqué, surtout au début quand on a dû mettre toutes nos visions du projet à plat pour en construire une commune, mais l'a été encore plus au confinement. Devoir se marcher sur les pieds (on avait revu les ambitions du projet à la baisse.) en étant sur des parties d'un même script est, je pense une des organisations les moins productives.

Après tout ce chemin, je garde quand même un bon souvenir de ce projet, et surtout du résultat qu'on arrive à vous présenter à travers ce cahier des charges. On aurait pu certes faire mieux si tour à tour on avait mis plus de temps pendant le confinement pour l'avancer, mais la santé qu'elle soit mentale ou physique reste le plus important.

Voulant plus m'orienter développement web et animation, je ne pense pas recroiser la route de ce charmant langage un jour qu'est le lua dans mon ide, mais je reste comblée de cette découverte et je pense m'intéresser à l'évolution des projets de la communauté.

Pour finir, je souhaite le meilleur à Alyssia.B (une des personnes qui nous a quittés durant ce ptut) pour son nouveau ptut en espérant que tout se passera au mieux.

C. Hoang Chi Ton

Ce projet m'a permis d'acquérir en premier lieu des compétences en LUA, en plus de devoir gérer les versions avec git.
De plus, cela m'a obligé de m'organiser dans mon emploi du temps et à coopérer avec les autres même si dans le contexte dans lequel nous avons travaillés n'a pas amélioré la situation. Ce qui a échoué, c'est l'implémentation de sons qui est inaudible, ce qui rend le jeu moins vivant. Difficulté à coopérer due au covid, car nous ne pouvions pas nous retrouver en présentiel pour parler des points importants.
L'équipe n'était pas au complet, nous avions dû revoir à la baisse le cahier des charges.
La communauté autour de minetest est petite donc pour trouver des solutions, c'est difficile.
Donc du coup ça prend du temps de devoir gérer les cours et les projets/TP.

D. Evan Saint-Sorny

Le projet tutoré a rassemblé mes connaissances et les compétences introduites en cours de DUT informatique et m'a permis aussi aussi de simuler le travail dans le monde professionnel.
De la planification en passant par la conception puis par le développement ainsi que la communication, le travail d'équipe, les méthodes de suivi et de gestion de projets ce fût un projet riche et intéressant qui m'a apporté de l'expérience.
Cela dit, Minetest ne possède qu'une communauté restreinte et une documentation incomplète peu mise à jour. Ainsi, parfois, pour réaliser une tâche simple, il faut faire beaucoup de lignes de code Lua et/ou celle-ci s'avère compliquée à coder.
Le projet a été restructuré et adapté au cours de temps avec l'apparition d'évènements imprévisibles et naturels. Avec le COVID 19 et le confinement, nous avons tous dû faire face de notre côté (gérer les problèmes personnels, moins de communication et d'échange).
Ce fût un bon entrainement et une bonne simulation, j'aurais préféré néanmoins que le projet se fasse avec des technologies plus récentes et plus utilisées par des entreprises.