ISOCRAFT STORY

Rapport de Soutenance

BERTRAND Nicolas, CHANE Romain,
CLERAULT Raphaël, DALOZ Léo

March 20, 2024

Rapport de Soutenance

Table des matières

1 Introduction

Cette section faisait originellement partie du cahier des charges. Elle apparaît ici afin de faciliter la compréhension du rapport.

1.1 Nature du projet

IsoCraft Story est le fruit d'une inspiration née de l'immensité créative de *Minecraft*, alliée à la volonté de transcender les frontières du jeu de construction. Ce projet représente une fusion audacieuse entre le sandbox classique et une vision novatrice de l'exploration. Plongeant les joueurs dans un univers alien fascinant, le jeu se distingue par sa perspective isométrique, promettant une expérience de jeu qui repousse les limites de l'ordinaire.

Le joueur incarne un intrépide explorateur de galaxies s'étant écrasé sur cette étrange planète. Il est désormais chargé de reconstruire son vaisseau, tout en tentant de percer les secrets de cet environnement extraterrestre, à travers une exploration attentive et de la résolution d'énigmes.

Un concept incontournable du jeu est la formation de la planète en "couches" parallèles et placées les unes sur les autres, formées de cartes du jeu uniques à chaque partie et abritant une faune et flore inédites. Ces couches servent de progression au joueur, qui s'aventure toujours plus profond dans cet univers hostile, à la recherche de son histoire.

Voici des concept arts réalisés par Dall-E pour représenter l'atmosphère des deux premières couches :

2ère couche - surface

2ème couche - forêt luxuriante

1.2 Notre équipe

Nicolas

Je suis Nicolas BERTRAND, 18 ans, possédant plusieurs passions dans le domaine du sport avec le Basket-Ball, dans le domaine de l'art avec le dessin. Mais aussi dans le domaine de la logique avec les mathématiques, les échecs et notamment l'informatique. Les jeux vidéo me sont aussi un centre d'intérêt fort depuis mon plus jeune âge.

Mon premier contact avec l'informatique fut au collège avec des activités en mathématiques où l'on devait utiliser scratch pour dessiner des formes géométriques et résoudre des problèmes. Même si ces activités étaient amusantes, ce n'est qu'en première que mon regard se tourna de plus en plus vers l'informatique. En effet mes spécialités de physique et de Sciences de l'Ingénieur m'ont fait découvrir des langages de programmations tels que Python, Arduino et HTML.

Cela fait aussi un peu plus d'un an que mes connaissances et compétences en dessin se sont grandement approfondies. L'étude des perspectives, des couleurs, des lumières, des volumes et de l'utilisation de l'espace m'a permis d'acquérir une meilleure compréhension de la construction cohérente de mondes et de personnages. De plus l'utilisation régulière d'une tablette graphique m'a rendu plus à l'aise dans la création sur support digital et m'a amené une maitrise intermédiaire du logiciel de dessin Krita.

Mon attrait pour la logique mathématique, l'informatique et le dessin me serviront tout le long du projet sur Unity pour pouvoir créer le plus librement possible avec des camarades qui ont tous plus d'énergie et de passion les uns que les autres.

Romain

Je suis Romain CHANE, âgé de 17 ans, passionné d'informatique depuis mon plus jeune âge. La firme Nintendo m'a introduit au monde du jeu vidéo, que je considère comme un art capable de susciter de profondes émotions chez les joueurs. Actuellement, je suis fervent admirateur de la série des Souls, qui offre une expérience de jeu captivante et mémorable.

Au collège, j'ai été initié à la programmation à l'aide de Scratch, un logiciel qui m'a permis de concevoir divers jeux. Parmi ceux-ci, se trouvent deux jeux de combat, respectivement inspirés de Super Smash Bros et de Street Fighter,

ainsi qu'un jeu de course en pseudo 3D mettant en vedette le célèbre hérisson "Sanic". Aussi, j'ai acquis une petite maîtrise des langages Python, assembleur, HTML, CSS, Java ainsi que des requêtes en SQL sur des bases de données au cours de mes études au lycée.

Dernièrement, j'ai découvert RPG in a Box, un logiciel dédié à la création de jeux de rôle. Bien que mes projets sur cette plateforme n'aient jamais abouti pour cause de motivation et que la création de jeux avec cet outil soit extrêmement limitée, j'ai apprécié le fait de pouvoir façonner mon propre univers dans un ordinateur.

C'est pour toutes ces raisons que j'attends avec impatience de débuter mon nouveau projet sur Unity, qui m'offrira une plus grande liberté de création, qui me permettra de découvrir et de me servir d'un nouveau langage de programmation, qui me contraindra cette fois-ci à le finir puisqu'une deadline est imposée et surtout, qui me permettra de travailler avec des gens qui sont tout aussi passionnés que moi.

Raphaël

Je suis Raphaël CLERAULT, 18 ans, passionné d'informatique depuis mon plus jeune âge. J'ai des connaissances dans de nombreux langages comme le C, C++, C#, python, java, javascript, html/CSS, scratch, CAML, assembleur-x86, assembleur AVR, et mon propre assembleur, et mon propre langage de programmation.

Mes premiers projets, bien que modestes, ont été fait sur Scratch, puis Python, puis Unity, puis UE4/UE5, puis sur mon propre moteur de rendu. En 3e (2019) j'ai codé un jeu entier pour mon collège, à la demande de mon école, pour rendre les cours de français plus interactifs. J'ai ensuite, entre autres, codé une IA pour résoudre des parties du jeu *Démineur*, disponible sur le Play store au nom de *Démineur solveur*. J'ai ensuite porté *DOOM II* sur ma propre console portable, que j'ai fabriquée de A à Z, puis codé un moteur de rendu 3D pour cette console, très modeste de puissance (4000x moins puissante qu'un ordinateur moderne).

Léo

Je suis Léo, 18 ans. J'ai une expérience relativement variée dans la création d'applications et de jeux divers, souvent avec un moteur de rendu bas-

niveau par rapport à Unity, comme par exemple : PyOpenGL, PyGame, tk-inter (Python), Canvas (JavaScript), SDL/SDL2 (C++). Faire un jeu avec Unity change complè-tement le champ des possibles pour moi, avec l'approche totalement différente du moteur, de son contenu tout prêt mais qui ne va jamais parfaitement, c'est pourquoi je me retrouve souvent à recoder des mécaniques essentielles, par exemple un ray tracer, ou un système de collisions avec des équations différentielles pour juste un peu plus de performances.

J'aime lier les mathématiques à l'informatique, je l'ai par exemple fait quand j'ai créé mon ray tracer et différents systèmes de collisions, utilisé la trigonométrie pour diverses animations, interpolations et mondes générés procéduralement, mais aussi pour cracker de multiples jeux grâce aux probabilités et au dénom-brement. J'ai aussi profité de cette symbiose quand j'ai représenté des fractales, créé des moteurs 3D ou encore 4D avec des calculs d'intersections de divers objets mathématiques.

J'ai développé mes connaissances en programmation en grande partie grâce à ma soif d'apprendre et à Python. L'utilisation d'un langage complètement nouveau pour moi est, je le crois, une excellente voie d'apprentissage. Par ailleurs, ayant plus ou moins de connaissances dans une quinzaine de langages de programmation allant de HTML/CSS à Arduino, en passant par l'expérience acquise pendant la réalisation de deux ordinateurs 8-bits dans *Minecraft* et deux langages d'assembleur, ainsi qu'un compilateur C-like, je pense être capable de faire évoluer le projet.

2 Avancement actuel

Cette partie se focalise sur l'état actuel du projet, et des directions déjà adressées des points de vue technique et artistique.

2.1 Histoire, Lore

L'histoire d'Isocraft Story est destinée à prendre une place très importante dans son gameplay. En effet, pour se distinguer de notre principale source d'inspiration qui est *Minecraft*, nous voulons mettre en place un système d'*archéologie vidéoludique*.

A l'instar des œuvres de Hidetaka Miyazaki ou de Fumito Ueda (qui ont pris une place majeure dans les jeux vidéo de ces deux dernières décennies),

nous voulons intégrer une véritable histoire à travers des indices laissés dans le monde que le joueur sera amené à parcourir. Le joueur sera ainsi libre de choisir un nouveau chemin de jeu parmi ceux déjà existants, en tentant de reconstruire l'histoire d'Isocraft Story à l'aide des différents types de reliques aliens dispersées dans les différents niveaux de la planète *6429-B*.

Notre équipe a pour l'instant posé les principes fondamentaux du *lore* de notre planète, il reste cependant à établir les caractéristiques de chacun des peuples extraterrestres plus en détail pour pouvoir établir des récits cohérents à l'échelle de ces derniers.

2.2 Génération du terrain

IsoCraft story se déroule dans un monde tridimensionnel composé de blocs, ou voxels. Ces blocs sont générés grâce à une technique nommée *noise*, qui permet notamment de fabriquer des mondes uniques et infinis. Nous verrons ici comment le monde est généré, puis parlerons de la manière de rendu dans le jeu, et enfin des structures multi-blocs.

2.2.1 Génération du monde

Le monde étant affiché à l'écran sous une projection isométrique (Figure 1), l'affichage de terrain complexe comme des grottes n'est pas aisé à mettre en oeuvre, car très souvent caché par la caméra en troisième personne. C'est pourquoi les joueurs devront s'adapter aux défis pratiques que ce rendu propose, mais la génération doit aussi encourager et faciliter l'exploration.

Pour cela, nous avons utilisé du noise en 2D, en tant que *heightmap* (Figure 2). Cette technique consiste à associer à chaque "colonne" verticale de blocs du monde une hauteur, définie en fonction de la valeur renvoyée par le noise. L'algorithme de noise utilisé est OpenSimplex.

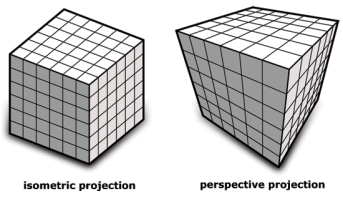


Figure 1 - projections

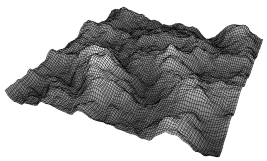


Figure 2 - heightmap

Cet algorithme, à l'opposé du white noise, qui est plus naïf et simple à implémenter (Figure 3), a comme particularité de renvoyer une valeur aléatoire entre -1 et 1, tout en préservant une certaine fluidité entre les points. Par comparaison, utiliser du white noise donnerait un terrain trop accidenté et imprévisible. Ajouter différentes heightmap ensemble permet de diminuer l'effet de fluidité excessive, et de créer un terrain réaliste sous forme de voxels.

Cette heightmap est alors utilisée lors de la génération de la grille de blocs qui constitue le monde, selon l'algorithme suivant : lors de la génération d'une portion du monde, l'algorithme procède colonne verticale par colonne. Pour chaque bloc dans cette colonne, sa hauteur est comparée à la valeur renvoyée par la heightmap.

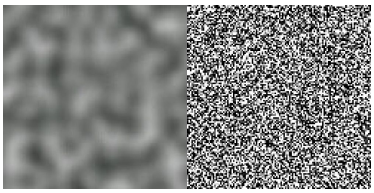


Figure 3 - perlin et white noise

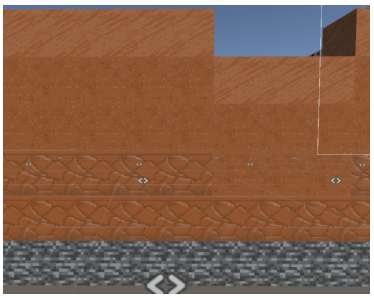


Figure 4 - colonne de blocs

Dans le cas où la hauteur est supérieure à la valeur de la heightmap, le bloc sera rempli avec de l'air. Sinon, un certain type de bloc est attribué en fonction de sa hauteur et de sa distance avec le sol (Figure 4).

Comme nous utilisons un noise en 2D, nous ne pouvons pas générer de terrain complexe comme des grottes ou des blocs en surplomb. Mais cela n'est pas un problème, la perspective et la caméra à la troisième personne rendant l'application pratique de ces concepts difficile pour les joueurs.

2.2.2 Caméra

Il est à noter que la caméra joue un rôle important dans l'exploration et l'accessibilité, et a été conçue comme intuitive à utiliser. son implémentation est passée par plusieurs itérations, jusqu'à obtenir une caméra semi-contrôlée par le joueur.

En effet, celui-ci peut tourner la caméra selon l'axe Y, à des intervalles de 45 degrés, mais la hauteur de la caméra et sa distance restent gérées automatique-

ment. Cette caméra pourra ensuite être utilisée lors d'événements particuliers, comme pour cibler des points d'intérêt ou des ennemis.

2.2.3 Rendu du monde dans le jeu

Un cube a six faces; et le monde affiché peut comprendre des milliers de colonnes de blocs, chacune jusqu'à 16 blocs de haut pour le moment. Le coût en terme de performances est donc très vite conséquent. Pour cela, un algorithme de rendu optimisé est utilisé.

Chunks

Le monde étant infini, il est donc découpé en portions, ou *chunks*. Ceux-ci sont alors affichés s'ils sont proches du joueur, et seront prochainement chargés ou déchargés quand le joueur se déplacera. Ces chunks sont aussi utilisés lors de la sauvegarde et le chargement du monde, et pour partager les informations du monde en multijoueurs.

Optimisation des faces

L'amélioration en performance la plus importante est le fait de ne pas considérer le monde comme des cubes, mais comme des faces. En effet, deux cubes côte à côte ont une face en commun, il est donc inutile de la dessiner car le joueur ne la verra pas. Un algorithme est donc mis en place, exécuté uniquement lors de la modification des données d'un chunk (pose ou destruction d'un bloc et chargement initial d'un chunk).

L'algorithme va considérer un chunk comme une seule texture, un seul mesh, qui va être généré procéduralement en fonction des blocs présents. Il permet de n'ajouter au mesh que les faces des cubes qui sont visibles par le joueur, donc les faces des blocs solides se trouvant à côté d'un bloc d'air.

Une autre optimisation est mise en place, qui consiste à utiliser une *texmap* : une seule texture contenant tous les blocs (Figure 5). Cela permet de référencer, dans le chunk, non plus une texture potentiellement différente pour chacune des faces, mais seulement une position dans la texmap, ce qui accélère le rendu des chunks car seulement un asset a besoin d'être chargé à tout moment.

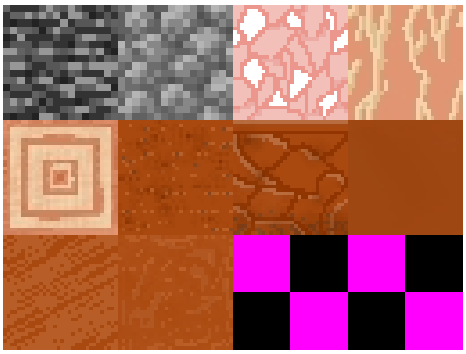


Figure 5 - texmap

Bien sûr, de multiples problèmes ont surgi lors de ces optimisations, notamment lors de la modification de blocs en bordures de chunks, car un ajustement du mesh des chunks adjacents pouvait être requise. De plus, il devenait nécessaire de savoir quels blocs allaient être présents dans les chunks, même non générés, car lors de la génération des chunks adjacents le jeu devait savoir s'il devait afficher ou non certaines faces sur les "côtés" du chunk. Tous ces problèmes ont été adressés et le monde est maintenant correctement affiché.

2.2.4 Structures

Evoquons maintenant les *structures*, qui sont des regroupements prédéfinis de blocs, par exemple un arbre. Ils se génèrent différemment d'une simple heightmap, car non pas un bloc mais plusieurs ont ici besoin d'être placés selon un arrangement précis.

Pour cela, les structures sont stockées dans des fichiers séparés, et nous avons mis en place un éditeur externe pour créer des structures plus facilement (Figure 6). Ces structures sont alors placées dans le monde, et cela en assignant à chaque colonne une certaine probabilité qu'une certaine structure apparaisse, ancrée à ces coordonnées. Pour obtenir cette probabilité, nous avons implémenté un *PRNG*, mais allons certainement changer pour utiliser du white noise.

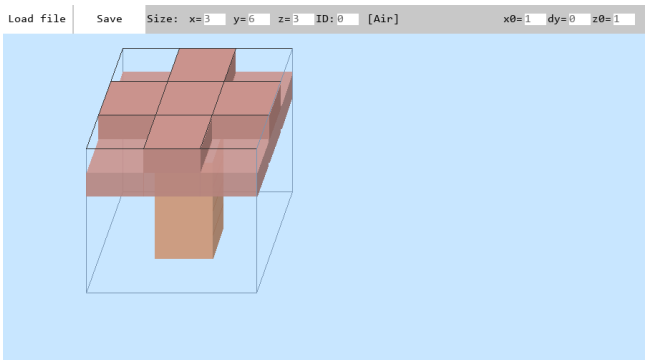


Figure 6 - éditeur de structures

Une fois qu'une structure est ancrée à un certain point, ses blocs sont calqués sur les blocs du chunk déjà généré, avec certaines conditions : certains blocs ne doivent pas être remplacés, ou alors par exemple des arbres peuvent avoir des variations dans leurs branches. Cela pose aussi quelques problèmes, lors de la génération d'un chunk : même si une structure n'est pas ancrée dans ce chunk en particulier, une autre ancrée dans un autre chunk pourrait "déborder" et cela doit être géré.

Tout ceci permet donc la génération procédurale d'un monde infini et unique, peuplé de structures et hautement optimisé.

2.3 Sauvegarde

Un élément important de tout jeu est de pouvoir sauvegarder son avancement. Pour Isocraft Story, nous avons choisi de tout sauvegarder : les paramètres, la position/rotation du joueur, son inventaire, sa vie, et le plus important, la carte, ou le monde.

Pour la sauvegarde du joueur et des paramètres, nous avons choisi la simplicité. Un dossier existe dans l'ordinateur, contentant toutes les sauvegardes du joueur, ainsi que les paramètres, qui sont universels pour toutes les sauvegardes. Une sauvegarde consiste en un dossier, du nom de la sauvegarde, contentant un fichier *.IsoSave*, et un dossier *Chunks* qui, lui, contient tous les chunks. Ces fichiers-là sont présents chez tous les joueurs.

Pour la sauvegarde du monde, une approche très différente a été choisie. En effet, le monde est divisé en chunks de 16x16x16 blocs et les blocks de la grille du monde peuvent être représentés par un id. Sauvegarder les chunks

avec le format *X.Y.Z :id* aurait fait qu'un chunk prenne 64Ko d'espace. Au vu du grand nombre de chunks qui doivent être sauvegardés, le monde aurait pris beaucoup trop de place. Nous avons donc trouvé un moyen de garder la même quantité d'information, mais en n'utilisant que 4ko par chunks, soit 16x moins.

Ce n'est pas une approche finale, seulement intermédiaire, car nous allons essayer d'optimiser encore plus, tout en gardant la possibilité de sauvegarder des *block entities*, soit des blocs avec des inventaires, et la position et l'état de différents modèles 3D, par exemple des coffres ou des gisements de minerais. Cependant, ce système de sauvegarde reste suffisant pour le moment.

Le dossier *Chunks* est présent uniquement sur le serveur pour un monde en multijoueurs, car lui seul en a besoin. Les clients ne peuvent pas récupérer de copies du monde, pour économiser de la place.

2.4 Menus, inventaire

A ce stade, les menus et l'inventaire sont fonctionnellement proches de leurs états finaux, mais loin de leurs aspects désirés. Le menu principal permet déjà de créer des nouvelles parties en solo ou multijoueurs, de charger des parties existantes, et de rejoindre des parties grâce a un code ou avec l'adresse IP du serveur. Le code est constitué de l'adresse IP et du port, mais encodés pour être plus facile a partager et retenir.

En jeu, on peut trouver 3 menus principaux :

- Le menu des paramètres, qui permet de modifier les paramètres, par exemple l'assignation des touches, du multijoueurs, de l'overlay (informations de debug, FPS et plus), et de tout ce que l'on compte rajouter par la suite. Il est actuellement fonctionnel, et il les paramètre sont sauvegardés correctement.
- L'inventaire du joueur, qui permet de visualiser les blocks que transporte le joueur, et de les manipuler, ou de les déplacer.
- Le *chat*, qui permet de communiquer entre joueurs, et de recevoir des messages d'informations du jeu. Par exemple, lorsqu'un joueur se connecte, un message dira dans le chat *Un nouveau joueur a rejoint la parie.*

2.5 Réseau

Pour la parie réseau, nous avons fait le choix dès le début d'utiliser l'asset *Mirror*, intégrant les bases du networking avec Unity. Regardons plus en détail les raisons de notre choix. Tout d'abord, les membres du groupe étant chargés du networking était déjà familiers avec cet asset. Ensuite, elle est gratuite, bien documentée, et offre une variété de solutions pour le networking avec Unity.

Nous avons fait le choix d'intégrer le networking dès le début du projet, pour toujours avancer avec le networking en tête. Notre jeu étant jouable de base en solo, mais optionnellement en multijoueurs, il aurait été facile de négliger cet aspect tout au long du développement.

Comme dit dans le cahier des charges, le multijoueurs de IsoCraft Story permet de vivre l'aventure du jeu avec des amis, en coopération. Par simplicité et économie, il n'y aura pas de serveurs dédiés au jeu, mais à la place, le joueur possédant la sauvegarde du jeu pourra host le serveur. La tâche demandée est relativement simple pour la machine du joueur et ne demande presque pas de puissance en plus.

A noter que toute partie, même solo, est en réalité une partie en multijoueurs; seulement, lors des parties solo, le nombre de joueurs maximum est fixé à 1, ainsi personne ne peut se connecter. Cela a pour avantage de gérer les modes solo et multijoueurs de la même façon, et de passer in-game une partie solo en multijoueurs, simplement en augmentant le nombre de joueurs max et en changeant la visibilité du serveur.

De ce fait, les actions des joueurs se déroulent ainsi lors de la gestion de plusieurs joueurs :

- Chaque joueur a une copie de la carte, des objets et des autres joueurs sur sa machine.
- Lorsqu'un joueur A modifie une de ses propriétés, par exemple en se déplaçant, il va de lui-même envoyer sa nouvelle position à tous les autres joueurs, qui vont mettre à jour leur copie de ce joueur A.
- Lorsqu'un joueur A modifie la carte, il va envoyer une commande au serveur, contenant par exemple les coordonnées du block à modifier. Le serveur va alors gérer la modification du bloc en question, ce qui permet de sécuriser toute modification et de prévenir de bugs de duplication, de modification de blocs dans des espaces non chargés, ou de blocs impossibles à modifier.

Le serveur renvoie une information au joueur A, correspondant au fait qu'il ait ou non modifié la carte. Par exemple, si l'opération de casser un bloc a échoué, il ne faut pas que le joueur A rajoute ce block dans son inventaire. Si ce dernier peut casser le bloc, le serveur va envoyer à tous les joueurs l'ordre de supprimer le bon bloc dans leurs copies de la carte.

- Lorsqu'un joueur A se connecte ou se déplace, il a besoin de recevoir les chunks à afficher. Il va donc envoyer une commande au serveur avec la position du chunk à recevoir. Le serveur a dans ce cas-là 2 choix. Soit le chunk existe dans la sauvegarde, dans ce cas il va simplement récupérer le chunk et l'envoyer au bon client. Soit le chunk n'a jamais été généré, et dans ce cas le serveur va le générer, l'envoyer au joueur A, et le sauvegarder au prochain auto-save.

De cette manière, le serveur est celui qui a une version mise à jour de la carte et de la position de tous les joueurs, ce qui permet de facilement sauvegarder et charger le monde, et de gérer le passage de solo à multijoueurs et inversement.

2.6 Art, design

L'avancement de la direction artistique dans notre projet progresse de manière significative. Nous avons déjà finalisé celle du premier et du second niveau, en ayant soigneusement réfléchi à chaque détail pour assurer une immersion optimale des joueurs.

Pour mettre en place ces différentes ambiances dans les niveaux de la planète *6429-B*, nous aurons besoin de modèles 3D pour représenter les créatures et objets rencontrés sur la planète. Nous aurons aussi besoin de textures qui seront appliquées aux blocs afin de représenter les différents types de matériaux sur lesquels le joueur progressera. Et enfin, nous avons aussi besoin d'effets sonores et visuels (SFX et VFX), pour immerger davantage le joueur dans les souterrains de notre planète virtuelle.

Voyons plus en détail les aspects visuels, puis nous verrons plus tard les aspects sonores.

2.6.1 Textures 2D

Le monde étant constitué de blocs, la réalisation de la texture d'un bloc nécessite les points suivants :

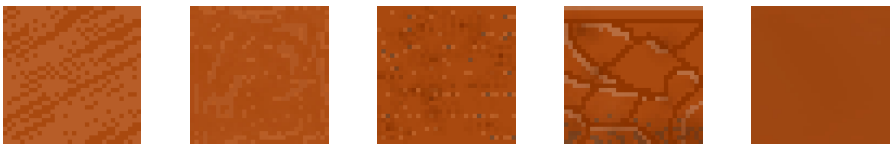
- Réaliser les différentes faces du bloc si elles ne sont pas toutes identiques.
- La texture d'un bloc doit être unique ou du moins distincte des autres, cela est d'autant plus compliqué que le design des blocs est en pixel art, de 32x32 pixels.
- La texture doit être répétable, out *tileable* sur un plan. Cela est surtout nécessaire dans le cas où plusieurs blocs seraient côte à côte, pour ne pas voir d'évidente coupure entre les côtés d'un bloc. De plus, il est préférable de ne pas voir cette coupure en dessinant une texture sans motif qui deviendrait visible lors de la répétition d'un grand nombre de blocs.

Notre jeu se basant en partie sur l'exploration des différentes couches de la planète, il était nécessaire d'avoir des textures signatures de chaque couche. La première étant la surface, un désert peu accueillant, nous avons donc commencé à travailler sur des textures dans les teintes rouges pour renforcer ce côté hostile de la surface.

De ce fait, nous avons décidé que le sol de la première couche (surface) soit du sable rouge avec ses différentes variantes. De plus, ce choix reste en lien avec l'histoire d'une planète ancienne et abandonnée : en effet, la formation du sable rouge est due à la présence d'oxyde ferrique, de ce fait plus le sable est ancien, plus il sera oxydé et plus il sera rouge.

Pour les différentes variantes du sable, nous en avons déjà trois :

- Sable fin rouge, les deux à gauche : il se trouve à la surface du sol.
- Sable rouge, au milieu : il se trouve entre le grès rouge et le sable rouge.
- Grès rouge, les deux à droite : il se trouve au plus profond de la surface.



Nous utilisons aussi d'autres textures issues de *Minecraft* temporairement, en attente de décision concernant le design de ceux-ci et de leur logique par rapport à la construction du monde, par exemple le bloc incassable se trouvant au plus profond du monde.

Nous avons aussi commencé la création de textures des ressources utiles au joueur avec celles des arbres (feuillage et tronc) :

- Tronc, à gauche et au milieu
- Feuillage, à droite

2.6.2 Modèles 3D

L'avancée dans la création des modèles 3D a été accompagnée de certains défis pour nos équipes, notamment dans l'apprentissage de Blender, l'outil que nous avons choisi pour ce projet. Au début, la courbe d'apprentissage pour maîtriser les fonctionnalités et les raccourcis de Blender a été assez raide. Les membres de l'équipe ont dû consacrer du temps à explorer les différentes fonctions et à s'habituer aux raccourcis clavier pour optimiser leur flux de travail.

Animer des personnages a également posé des défis. Nos artistes ont dû apprendre à utiliser les outils d'animation de Blender pour donner vie à nos personnages de manière convaincante. Cela impliquait de comprendre les principes de base de l'animation, tels que les *keyframes*, les poses et les cycles d'animation, et d'appliquer ces connaissances de manière créative pour créer des mouvements fluides et réalistes.

Malgré ces défis, nous avons progressé avec succès dans la création des modèles 3D. Les premiers objets du premier étage, comme les coffres et les torches, mais aussi des modèles de fleurs (Figure 7), ont été achevés avec succès, ajoutant une dimension supplémentaire à l'immersion du joueur dans la première partie du jeu. De même, les modèles 3D du joueur (Figure 8) et du vaisseau spatial ont été finalisés, offrant des visuels attrayants qui captent l'attention des joueurs.

Cependant, il reste encore à accomplir tous les modèles 3D des ennemis de la première partie du jeu, ce qui constitue un défi sur lequel nous nous concentrons dans les prochains jours. Nous sommes déterminés à garantir des sensations optimales de combat et d'exploration en concevant des ennemis aussi captivants que les autres éléments du jeu.

Figure 7 - Fleur dans Blender

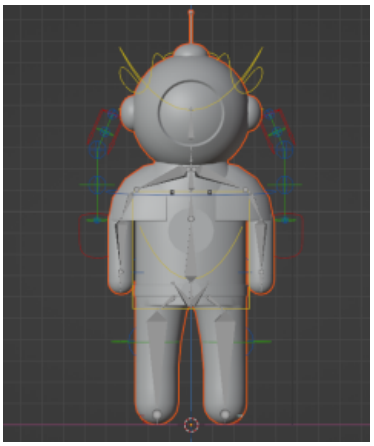


Figure 8 - Modèle du joueur

2.7 Musiques

Le jeu étant divisé en "couches", chacune abritant une ambiance unique, les musiques se doivent de refléter ce caractère. Les instruments sont pour la plupart choisis par niveau pour obtenir une certaine homogénéité au sein d'une même couche.

Par exemple, la couche externe de la planète est hostile, venteuse et désertique, c'est un terrain de souvenirs et de mélancolie, car c'est là que le vaisseau du joueur s'est écrasé. Pour cela, des musiques calmes ont été composées, à base de pianos mélancoliques et violons pour symboliser la solitude et le temps qui passe.

Quelques autres musiques ont été composées pour les niveaux suivants, soulignant notamment l'aspect mystérieux de la planète par les dissonnances, l'inconnu par le choix sombre de mélodies et d'instruments. C'est le cas en particulier pour certaines couches énigmatiques et ténébreuses, ou abritant des boss ou vestiges de civilisations disparues.

Les musiques de début et de fin ont été composées, et leur formation est opposée. Cela permet de les lier, de faire le lien entre la majuscule et le point de la phrase, pour que ces musiques et les cinématiques les accompagnant servent de prologue et d'épilogue. La formation de la musique pour le crash utilise des descentes chromatiques et une certaine amplification de la tension. Elles soulignent le désespoir, accompagné de la chute, dans la destruction du seul point d'ancrage du joueur.

Par opposition, la formation similaire mais opposée de la musique jouée pendant l'envol du vaisseau à la fin du jeu représente l'espoir, le souvenir de toutes ces aventures vécues au cours de l'exploration, l'envol et la promesse d'un sauvetage et du retour du joueur parmi les siens.

Les musiques sont donc pensées pour accompagner le joueur, le faire s'identifier et s'attacher à différents aspects du jeu. Elles visent aussi à faire prendre conscience au joueur des merveilles et de la diversité du monde sur lequel il est échoué, ainsi que de sa taille relative à ce complexe et mystérieux écosystème.

2.8 Cinématiques

Afin de raconter l'histoire au joueur, des cinématiques seront intégrées au jeu, par exemple au moment de la création d'une partie. Ces cinématiques, réalisées sur blender, n'utilisent pas d'asset extérieurs, car tout est modélisé/simulé par les membres de notre groupe.

La cinématique du début comporte pour le moment trois scènes :

- La première, où on peut voir le décollage du joueur.
- La deuxième, où le joueur voyage dans l'espace, et rencontre un nuage d'astéroïdes, qui abîme son vaisseau.
- La dernière, où son vaisseau est très abimé. Le joueur est alors éjecté, et son vaisseau s'écrase sur la planète *6429-B*.

Vous pouvez trouver une image de la cinématique du décollage en annexe.

Les modèles 3D sont créés et animés à la main, et les fumées/flammes sont simulées grâce a Blender. Cela permet un meilleur réalisme, au coût d'une animation longue à rendre pour l'ordinateur qui va créér les cinématiques initialement. Plusieurs dizaines d'heures de calculs au total seront nécessaires pour que l'ordinateur produise les cinématiques finales.

3 Evolution du planning

Nous verrons ici l'évolution du projet actuellement, comparée au cahier des charges original, de par nos avances et nos retards, tout en proposant quelques

modifications au tableau d'avancement qui nous servait de référence.

3.1 Avancement par rapport au planning, modifications

Notre planning, ou tableau d'avancement préétabli avant la première soutenance de méthodologie, a généralement été respecté, mais pas toujours. Ceci peut s'expliquer dans la mesure où certaines directions étaient facilement identifiables dans la construction du jeu, mais certaines moins.

Voici notre ancien tableau d'avancement pour référence :

Soutenance	**1**	**2**	**3**
Génération / Map	70%	90%	100%
Histoire / Mécaniques	50%	75%	100%
Modélisation	20%	80%	100%
Interface / Design	0%	60%	100%
Cinématiques	50%	75%	100%
Contrôles / Gameplay	50%	100%	100%
Textures	40%	70%	100%
Musiques / FX	20%	60%	100%
Post processing, review	0%	30%	100%
Animations	10%	30%	100%
Multijoueurs / Networking	20%	70%	100%
IA	0%	50%	100%
Site internet	30%	90%	100%

Ce tableau permettait de visualiser l'avancement des différentes tâches nécessaires au développement du projet. Les pourcentages représentent les proportions en terme de travail prévu pour les différentes tâches. Le tableau a été conçu pour parer aux éventuels écueils et obstacles qui auraient pu nous ralentir, et nous nous sommes efforcés de suivre ce tableau à titre indicatif.

Certains aspects du jeu dépendent d'autres, comme par exemple les contrôles nécessitant un certain avancement sur la génération de la map. En effet, certaines actions sont conditionnées par l'aspect du terrain. L'interface, les textures et le post-processing ne sont possibles qu'après qu'un avancement significatif ait été fait dans la partie "Histoire", car cette dernière va préciser le caractère du jeu et donc ses effets visuels.

Mais tous les aléas n'ont pas pu être prédits. On peut facilement remarquer

que nous n'avons pas judicieusement choisi les catégories et voies de travail, notamment à cause du fait que la quantité de travail que représentent toutes ces voies de travail a parfois été mal estimée.

Certaines catégories n'ont donc pas un très grand impact, ou alors sont trop vagues pour représenter quelque chose de tangible, par exemple "Contrôles / Gameplay", alors que d'autres catégories, par exemple "Histoire / Mécaniques", englobent trop de choses et méritent d'être redistribuées.

C'est pourquoi nous avons modifié ce tableau pour le remettre au goût du jour, et pouvoir mieux nous concentrer sur des tâches concrètes et équivalentes en terme de travail. Cela nous permet aussi de répartir mieux les tâches entre les membres de notre équipe. Voici ce tableau modifié :

Soutenance	**Méthodo**	**1ère**	**Mi-mai**	**Dernière**
Génération / map	50%	75%	100%	100%
Complexité des voxels	0%	20%	X%	X%
Block entities	0%	0%	100%	100%
Histoire / lore	50%	75%	90%	100%
Design des couches	30%	60%	80%	100%
Progression, ressources	20%	50%	80%	100%
Crafting, alchimie	0%	0%	30%	100%
Boss, end game	0%	20%	40%	100%
Ascenseurs, liens niveaux	0%	0%	90%	100%
PVP, combat	0%	10%	80%	100%
Spawn des mobs	0%	0%	90%	100%
Items	0%	0%	50%	100%
Modélisation	10%	30%	50%	100%
Interface/design	30%	60%	75%	100%
Cinématiques	50%	75%	85%	100%
Balancing, playtest	0%	25%	50%	100%
Musiques	10%	40%	70%	100%
FX	0%	0%	20%	100%
Post processing, review	10%	20%	50%	100%
Animations	0%	0%	X%	X%
Multijoueurs / networking	80%	90%	100%	100%
IA	0%	20%	70%	100%
Site internet	0%	50%	80%	100%

"X%" signifie que la direction correspondante n'est pas réellement nécessaire, et constitue plus une voie de recherche en cas d'avance sur le planning. C'est le cas pour les *Animations*, car nous cherchons à avoir quelque chose qui fonc-

tionne, mais ce sujet est propre à de nombreuses itérations et améliorations, car nos capacités vont évoluer au fil du temps.

On remarque aussi cette notation dans *Complexité des voxels*. Cela correspond à une amélioration de la map pour supporter non plus seulement des blocs et des modèles 3D placés dans le monde, mais aussi des mesh plus compliqués, comme des demi-blocs ou des escaliers. Cette variété est inspirée par le rendu visible dans *Minecraft*.

L'implémentation de modèles 3D dans le monde se fera quant à elle dans *Block entities*, qui servira aussi de catégorie pour la gestion de données supplémentaires dans la map, comme par exemple les objets contenus dans un coffre placé dans le monde, ou l'état de l'IA d'un monstre, pour la persistance entre les joueurs en multijoueurs et pendant la sauvegarde.

Le tableau d'avancement est modifié pour supporter un point de passage entre les deux soutenances techniques, pour mieux suivre un avancement préétabli. On peut aussi remarquer que le tableau contient beaucoup plus de sections, cela pour simplifier la répartition et redistribuer la charge de travail, comme nous le verrons plus bas.

3.2 Difficultés rencontrées, retards

Nous avons choisi de rajouter une colonne dans le tableau d'avancement pour plusieurs raisons : premièrement, comme dit précédemment, pour nous permettre de mieux nous rendre compte de notre avancement.

Ensuite, pour mieux répartir les tâches dans le temps : en effet, entre la première et la dernière soutenance techniques, de nombreuses catégories ont un fort changement en terme de pourcentage. Cela a une explication simple : nous avons pris du retard sur le planning. Il nous reste donc beaucoup de choses à faire en peu de temps, et mieux nous organiser semble être un point crucial.

L'organisation a été l'un des points faibles les plus importants dans notre organisation, car nous pouvions avoir travaillé bien plus sur le projet mais n'avions parfois peu ou pas de directions claires sur quoi avancer, ou alors pas assez de variété dans notre travail. C'est pourquoi on peut remarquer que, parmi les catégories à fort changement de pourcentage dans le tableau, approximativement une moitié de la charge de travail est répartie sur la première moitié du temps qu'il nous reste, et l'autre moitié moins essentielle est placée après.

Cependant, tous les pourcentages, mêmes pour ceux à implémenter vers

la fin du projet, augmentent continuellement. Cela s'explique par le fait que nous nous laissons une certaine liberté d'implémenter plusieurs choses en même temps, pour nous simplifier la tâche dans le futur.

Un bon exemple de cela est la gestion de la génération, qui a tout de suite été préparée en vue de l'implémentation du multijoueurs. La génération a été implémentée très tôt, car essentielle, et n'a pas posé de problème lors de l'ajout du multijoueurs. De même, implémenter le multijoueurs tôt permet la simplification de l'implémentation de fonctionallités futures.

Adressons pour finir un autre des points faibles de nos retards : la répartition entre les membres de notre équipe était mal anticipée, certains membres se retrouvant avec beaucoup moins de code que d'autres. Nous avions créé le tableau d'avancement pour équilibrer la quantité de travail, mais cela ne s'est pas avéré être une bonne solution.

Nous espérons que les modifications apportées vont nous permettre de mieux identifier les directions à prendre lors du développement et de la réalisation de notre projet.

4 Avis et conclusion

Après quelques mois de travail, voici nos impressions sur ce qui a été effectué et ce qu'il reste à venir.

4.1 Avis des membres de l'équipe

Nicolas

Le projet a été plaisant à réaliser jusqu'à présent, en passant de la réflexion à la réalisation de différentes textures, et par leur incorporation dans le jeu tout en respectant le lore. Cependant, cette étape reste longue, au vu du nombre de textures à réaliser, mais elle est aussi complexe pour les critères à respecter pour la cohérence d'une texture.

Pour ce qui est de la partie développement, la sous-estimation du temps nécessaire pour le projet a été un frein à son avancée, notamment dans la prise en main du logiciel Unity et des différentes fonctions qu'il apporte.

Romain

Dans l'ensemble, nos progrès sur Isocrat Story sont prometteurs. Toutefois, je crois qu'il est impératif d'améliorer notre organisation. Il nous arrive parfois de manquer de clarté quant aux tâches à accomplir et à la manière de les exécuter, ce qui peut entraver nos avancements. Je pense donc que mettre en place des réunions régulières et une meilleure documentation de nos responsabilités accroîtrait notre efficacité et est une nécessité.

Cela dit, travailler sur ce projet a été extrêmement agréable. Imaginer un univers, son histoire et ses peuplades avec des collègues aussi sympathiques a été une expérience particulièrement enrichissante. La collaboration avec une équipe aussi dynamique et créative a été une source d'inspiration constante, et j'ai hâte de poursuivre cette aventure ensemble.

Raphaël

J'adore travailler sur ce projet. Travailler le multijoueur m'a permis de mieux comprendre les protocoles des réseaux, et concevoir des cinématiques m'a permis d'apprendre à encore mieux utiliser des logiciels comme Blender.

Mon expérience avec Unity m'a permis de commencer rapidement les bases, et de pouvoir faire avancer le jeu rapidement. Ce jeu est un défi technique mais nous pouvons y arriver, et nous allons y arriver.

Léo

Après avoir commencé par travailler sur la génération, je me suis vite rendu compte de l'ampleur du travail. Nous nous devons de travailler régulièrement sur ce projet. J'ai presque une centaine de commits sur le repo git, mais une importante partie a été concentrée dans l'espace d'une semaine... Je pense que nous sommes capables d'adresser tous les défis que nous nous sommes donnés, à la condition que nous trouvions le temps de travailler régulièrement.

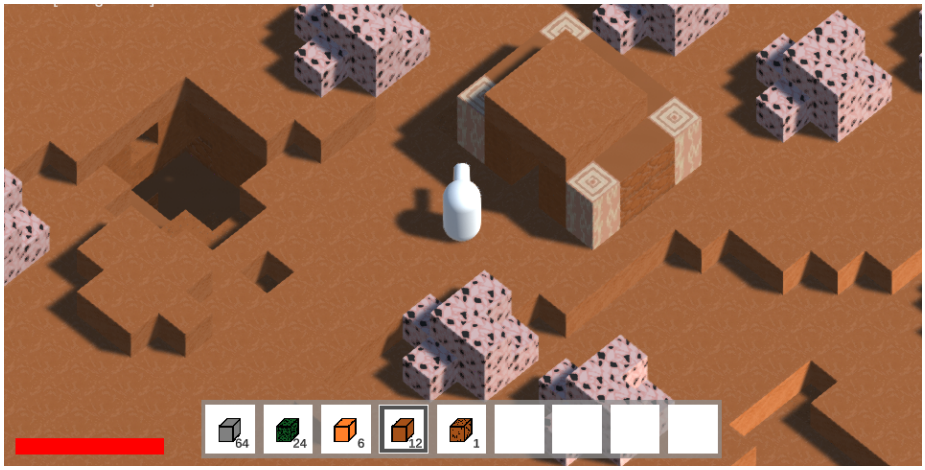
De manière générale, j'ai beaucoup apprécié le travail que ce projet demande, les recherches et algorithmes mis en oeuvre, la résolution de problèmes, et je suis prêt à continuer et mener à bien le développement !

4.2 Conclusion

IsoCraft Story, jeu d'aventure dans une planète alien, se déroule dans un monde de voxels infini et constitué de couches. La génération du monde, l'interface utilisateur et les menus, ainsi que des objets graphiques et musicaux ont été créés, mais nous sommes en retard sur quelques autres points. Nous concentrons nos efforts sur la deuxième couche en ce moment, et allons bien vite rattraper notre retard dans le planning amélioré, pour donner vie à ce jeu !

5 Annexes

Lien repo GitHub : https://github.com/RaphoufouLeFou/Isocraft_Story



Capture d'écran du jeu

Site web, en cours de construction, où l'on pourra trouver diverses informations sur le jeu

Logo du jeu, logo de WarpZone Entertainment

Extrait d'une cinématique rendue dans Blender