

Rapport de première soutenance

Nom de Projet : *Pricefield Garden*

Nom de groupe : Arcadia

Belkhder Ilias

Martinez Alexandre

Poelger Jonathan

Brice Benoit

Mars 2020



Encadrants : Remi Vernay, Philippe Roussille

Table des matières

1	Introduction	2
1.1	Le projet	2
1.2	Résumé de l'avancée	2
2	Découpage du projet	3
2.1	Avancement des tâches	3
2.2	Story telling	4
2.3	Menu (Alexandre M.)	4
2.4	Interface joueur (Ilias B.)	5
2.5	Graphisme	6
2.6	Soundtrack (Benoit B. et Ilias B.)	8
2.7	Enigmes (Ilias B. et Benoit B.)	10
2.8	Personnage (Jonathan P.)	11
2.9	Level Design	15
2.10	Site web (Jonathan P.)	16
3	Récit de la réalisation	17
4	Deuxième soutenance	19
5	Conclusion	19

1 Introduction

Ce compte-rendu a pour but de présenter l'avancement de notre groupe, Arcaidia, sur notre projet Pricefield Garden. Il détaillera notamment le travail effectué depuis la remise du cahier des charge en janvier, et nos objectifs pour la prochaine soutenance.

1.1 Le projet

Pour ce projet, nous avons décidé de réaliser un jeu s'inspirant des jeux à mondes ouvert et d'éénigmes. Pricefield Garden est ainsi un jeu solo en vue à la troisième personne, où le joueur incarne un personnage qui se réveille dans une plaine aux côtés d'une lupiotte. Rapidement, il se rend compte qu'il est bloqué sur une île, entouré d'une part d'une mer qui n'en finit pas, et de l'autre d'un immense mur. Il comprend que pour s'échapper, il lui faudra résoudre les différentes énigmes qui arrive sur son chemin et trouver la sortie. Ici, le joueur devra explorer la carte et les environnements mis à sa disposition, tout en résolvant les énigmes qui lui seront proposés afin d'avancer dans cette quête. Il va devoir s'entraider avec la lupiotte afin de franchir les différents niveaux qui se succèdent. Chaque région de la carte consiste en une succession d'énigmes.

Il devra, par ailleurs, trouver une explication à ces épreuves.

1.2 Résumé de l'avancée

Afin d'avancer rapidement, nous nous sommes réparties les tâches pour converger vers la création du jeu. En effet, il fallait que d'un côté un début de carte soit fait, et de l'autre qu'un personnage et que des menus soit créés afin d'obtenir les outils de travaux. La première tâche que nous avons réalisée (passée la prise en main des différents logiciels) était la création du personnage, puis les interactions que peut faire le joueur en possession de ce dernier. Ensuite, nous avons fait un début de carte sans texture et nous avons créé la mer que nous allons mettre autour.

2 Découpage du projet

2.1 Avancement des tâches

Ce tableau montre la répartition réelle des tâches qui est un peu différente de celle prévue au départ.

Répartition	Jonathan	Alexandre	Illias	Benoît
Menu		X		x
Interface Joueur			X	
Graphismes	X	X	X	X
Soundtrack			x	X
Enigmes			X	X
Personnage	X	x		
Sauvegarde	x	X		
Level design	X	X	X	X
Story telling	X	X	X	X
Site	X			

Ce tableau montre l'avancée actuelle des différentes tâches du projet.

Tâches	Avancement réel	Avancement prévu
Menu	25	25
Interface joueur	33	25
Graphisme	25	33
Soundtrack	25	25
Enigmes	40	20
Personnage	50	50
Sauvegarde	0	0
Level Design	75	75
Story telling	90	100
Site Web	66	50

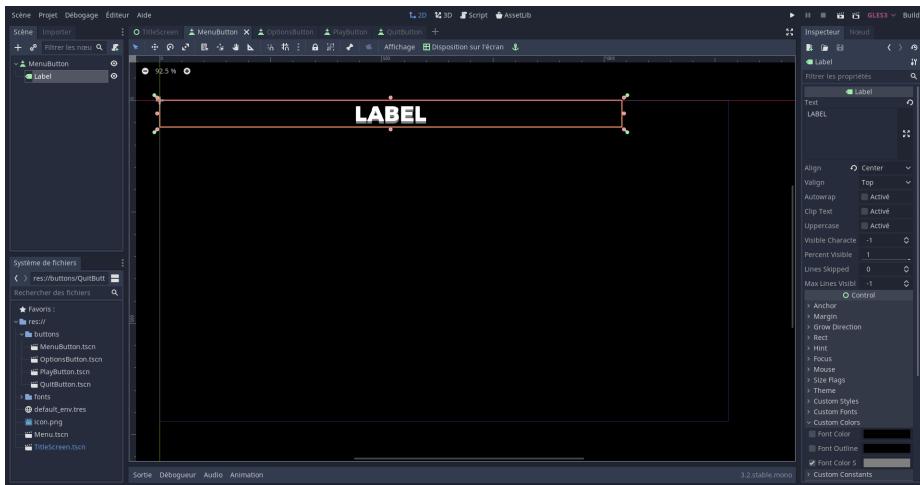
2.2 Story telling

Au début du jeu, une jeune fille avec un long capuchon de toile se réveille au milieu d'une prairie. Près d'elle, une sorte d'esprit prenant la forme d'une sphère lumineuse lui apprend qu'elle est emprisonnée sur une presqu'île séparée du reste du continent par un immense mur. Au centre de ce dernier, une grande porte permet l'accès à cette "prison". Pour pouvoir accéder à cette porte il faudra donc explorer l'île et en découvrir les secrets en résolvant des énigmes notamment. Dans un premier temps, il faudra trouver le moyen d'accéder à la partie centrale de l'île où se trouve une grande tour qui la surplombe. À partir de là, trois chemins s'offrent à elle : le premier mène à une forêt qui semble se répéter à l'infini, le second à un labyrinthe et le troisième à une grotte. Au fond de la grotte se trouvent des cellules et cela permet de finalement récupérer un morceau de clef. À la sortie du labyrinthe se trouve un village qui semble abandonné et où se trouve la seconde partie de la clef. Avec ces deux parties, la jeune fille peut ouvrir la tour et à son sommet, un mécanisme permet de rendre la forêt "pénétrable". Elle peut donc accéder à la dernière partie de l'île, et la grotte constituera la dernière énigme. À la fin du jeu, plot twist, l'esprit et le joueur se rendent compte que cette fillette était enfermée pour une bonne raison...

2.3 Menu (Alexandre M.)



Le menu principal a été créé dans un canvas qui est le cadre vu dans ce menu. Plusieurs boutons ont été fait à partir du template ci-dessous. Les différents onglets montrent que les boutons sont séparés, permettant ainsi de les personnaliser si on le souhaite. Par exemple, ces instances servent ici à changer le label. Visuellement ces boutons ont une ombre qui crée une impression de relief.



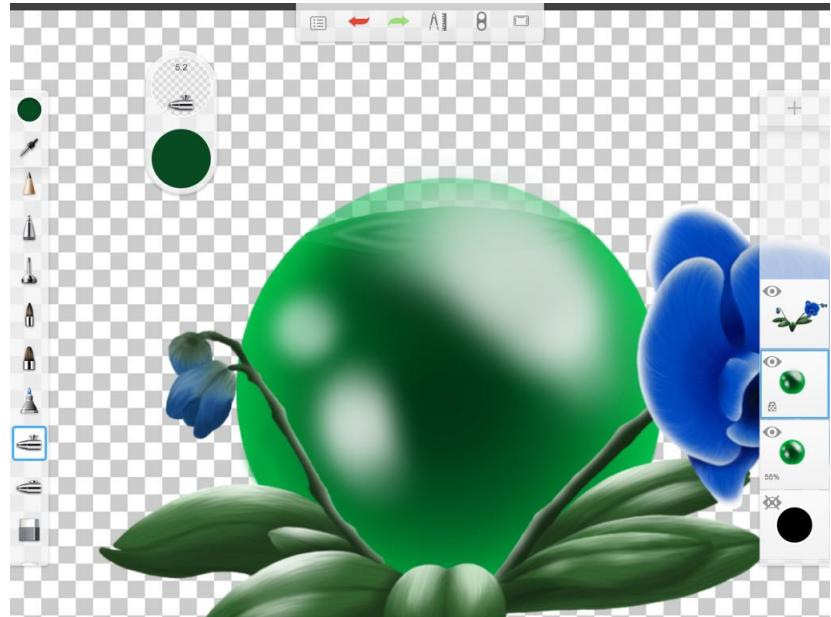
Pour la deuxième soutenance il faut avoir fini le menu hormis les options. Pour cela il faut créer la redirection à ce menu via le bouton OPTIONS sans pour autant remplir l'interface. De plus, nous devons charger une partie ou en créer une nouvelle grâce au bouton PLAY et quitter le jeu en cliquant sur QUIT. Pour cela il faut créer un système de sauvegarde pour ne pas avoir à recommencer le jeu à chaque fois qu'on le redémarre.

2.4 Interface joueur (Ilias B.)

L'affichage tête haute (en anglais : Head-up display - HUD) est un ensemble d'informations affiché en périphérie du centre de l'écran et renseignant le joueur sur son personnage ou son environnement : score, niveau, santé de son personnage, arme utilisée, nombre de munitions restantes, carte, position du joueur, position des ennemis ou des opposants, informations de mission, communication alliée, chat avec joueurs en ligne. Les différents éléments prédisposés seront ainsi important pour l'expérience du joueur. Nous avons voulu mettre en place un objet important pour l'histoire, qui sera là au lieu d'une quelconque barre de vie ou autre. Au départ, nous hésitions entre une boule de différentes couleurs. Puis nous nous sommes dit que nous voulions que la couleur de cette dernière ait une double signification. Nous avons ainsi décidé d'utiliser la couleur émeraude : Signifiant la générosité, la chance, ou encore l'obscurantisme et le malheur.

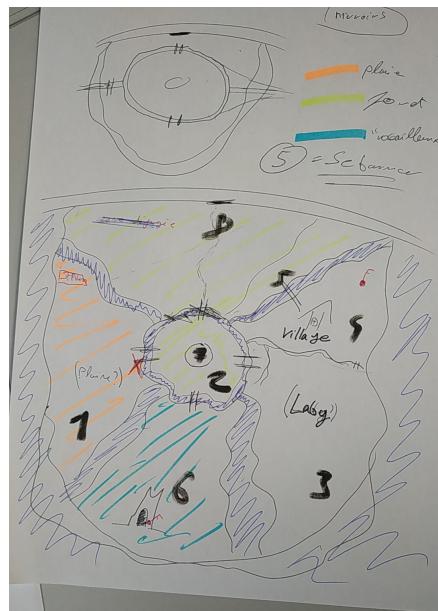


Cet objet sera placé en haut à gauche de l'écran. Il servira pour l'histoire et l'avancée du jeu, et aura divers stade. Elle pourra se remplir/se vider, et/ou avoir les fleurs l'ornant qui se fanent.



2.5 Graphisme

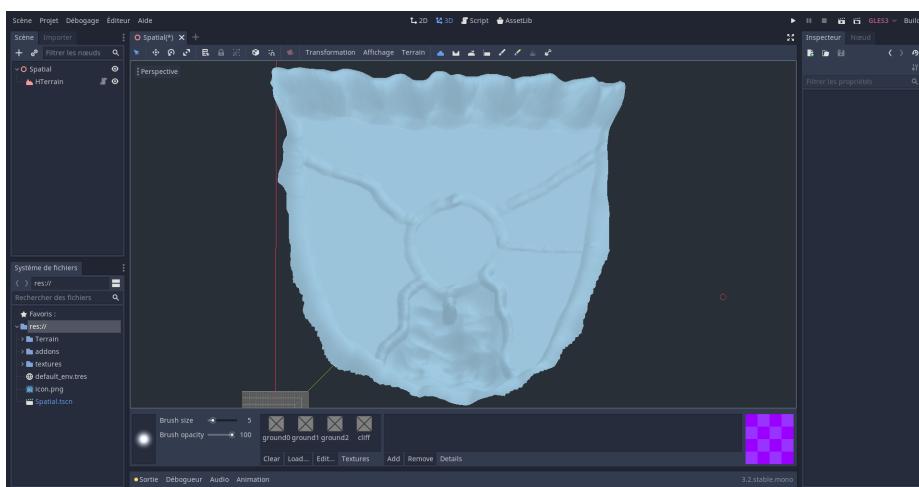
Pour la conception de la map, nous avons décidé de mettre plusieurs zones afin que les phases d'exploration du joueur soit plus intéressantes. Celle-ci serait composée de 5 zones : Les plaines, la grotte, celle qui contient le labyrinthe et le village, la tour centrale et enfin la zone finale contenant la porte de sortie. Sachant que la map serait une île prison où l'objectif du joueur serait de s'enfuir à la fin du jeu.



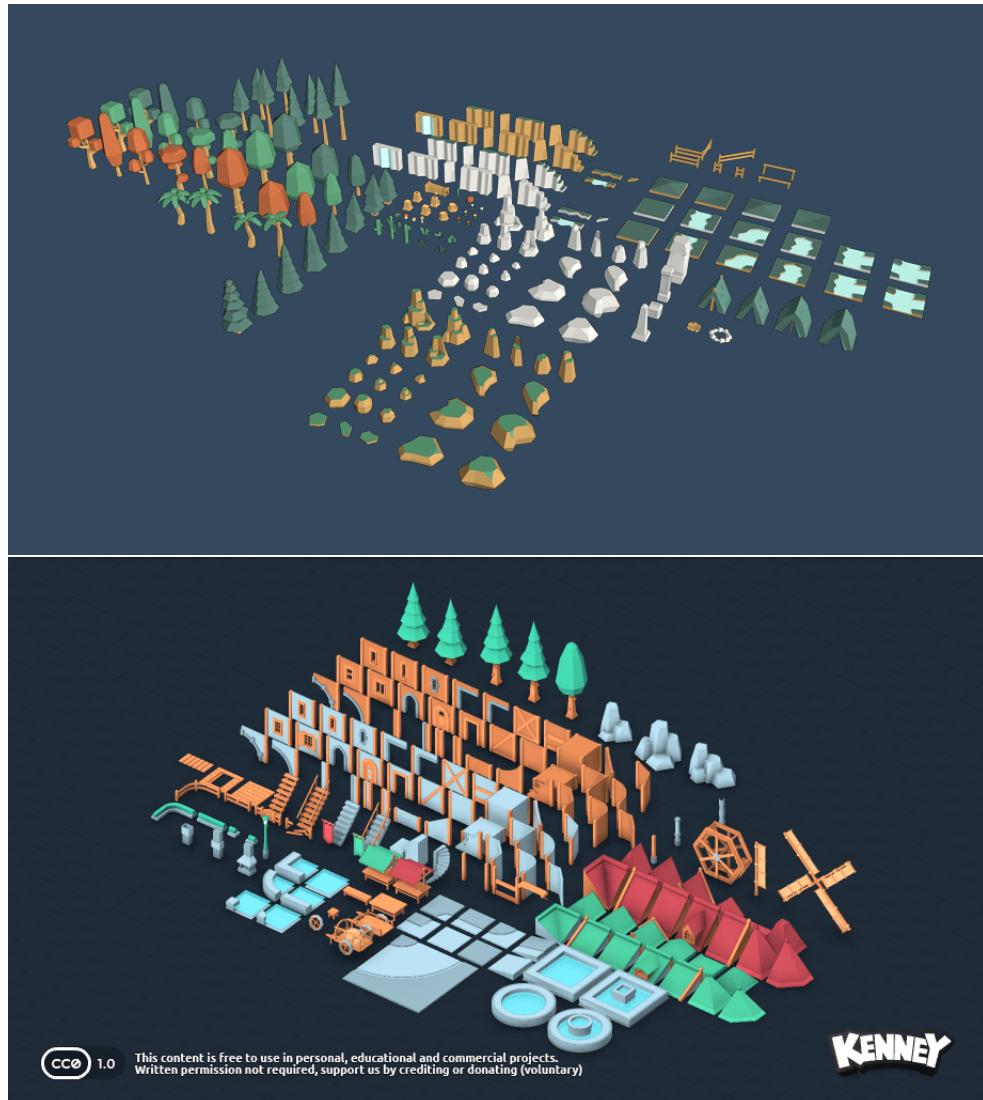
Pour les limites du jeu, nous hésitions entre des “murs” invisibles, ou une gigantesque étendue d’eau. C’est donc l’étendue d’eau qui a été choisie car nous voulions rendre plus vivant notre monde ouvert. Cette dernière a été créée en utilisant la fonction “New SpatialMaterial”, en mettant en place une couleur bleu assez basique puis une transparence afin de la faire paraître comme réelle. De plus, elle est dans la capacité de réfléchir et de se distordre.



La map a été commencée en téléchargeant des addons permettant de la modéliser. Ensuite un terrain plat a été créé et la construction de la map a démarré à partir de ce plan en utilisant les outils disponibles “Raise” et “Lower” pour les reliefs. On peut donc y voir un mur en haut servant de sortie, cela étant notre objectif, ainsi que des creux servant à y accueillir des rivières. Dans la zone juste en dessous du mur et dans le cercle central nous voulons y mettre une forêt, ainsi qu’une tour au centre du cercle. La zone de gauche servira à créer une plaine, la zone du bas ayant quelques reliefs servira quant à elle à créer une grotte. Un village et un labyrinthe prendront place respectivement en haut à droite et en bas à droite.



Voici des assets dont nous allons nous servir pour notre map. Nous allons surtout nous servir des arbres pour faire une grande forêt et des maisons pour un village.



2.6 Soundtrack (Benoit B. et Ilias B.)

Pour la partie soundtrack, nous n'avons pour le moment rien fait de concret si ce n'est des recherches que nous allons évoquer tout de suite et qui vont être appliqués avant la seconde soutenance. Ce qui nous permettrait ainsi d'améliorer la qualité auditive de notre jeu.

Dans toute application ou jeu, la lecture du son et de la musique aura un léger retard. Pour les jeux, ce délai est souvent si faible qu'il est négligeable. Les effets sonores sortiront quelques millisecondes après l'appel d'une fonction play (). Pour la musique, cela n'a pas d'importance car dans la plupart des jeux, il n'interagit pas avec le gameplay.

Pourtant, pour certains jeux (principalement les jeux de rythme), il peut être nécessaire de synchroniser les actions des joueurs avec quelque chose qui se passe dans une chanson (généralement en synchronisation avec le BPM). Pour cela, il est utile d'avoir des informations de synchronisation plus précises pour une position de lecture exacte.

Il est difficile d'obtenir une précision de synchronisation de lecture très faible car de

nombreux facteurs sont en jeu lors de la lecture audio :
 L'audio est mélangé en morceaux (pas en continu), selon la taille des tampons audio utilisés (vérifiez la latence dans les paramètres du projet).
 Les morceaux audio mixtes ne sont pas lus immédiatement.
 Les API graphiques affichent deux ou trois images en retard.
 La solution à ce problème serait de réduire la latence consiste à réduire les tampons audio (là encore, en modifiant le paramètre de latence dans les paramètres du projet). Le problème est que lorsque la latence est trop faible, le mixage sonore nécessitera considérablement plus de CPU. Cela augmente le risque de sauter (une fissure dans le son car un rappel de mixage a été perdu).
 Godot est livré avec des valeurs par défaut sensibles qui ne devraient pas avoir besoin d'être modifiées.
 Le problème pour notre jeu n'est pas ce léger retard mais la synchronisation graphique et audio.
 Ensuite, nous avons appris qu'en appelant `AudioStreamPlayer.play()`, le son ne commencera pas immédiatement, mais seulement lorsque le thread audio traitera le morceau suivant.
 Ce délai ne peut pas être évité mais il peut être estimé en appelant `AudioServer.GetTimeToNextMix()`.
 La latence de sortie (ce qui se passe après le mixage) peut également être estimée en appelant `AudioServer.GetOutputLatency()`.

Ajoutez ces deux et il est possible de deviner presque exactement quand le son ou la musique commencera à jouer dans les haut-parleurs :

```
var actual_play_time = AudioServer.get_time_to_next_mix() + AudioServer.get_output_latency
$Song.play()
```

Ceci sont des codes réalisés sur GDScript pour le moment mais dont on devrait par la suite le faire en C pour la seconde soutenance.

```
var time_begin
var time_delay

func _ready()
    time_begin = OS.get_ticks_usec()
    time_delay = AudioServer.get_time_to_next_mix() + AudioServer.get_output_latency()
    $Player.play()

func _process(delta):
    # Obtain from ticks.
    var time = (OS.get_ticks_usec() - time_begin) / 1000000.0
    # Compensate for latency.
    time -= time_delay
    # May be below 0 (did not being yet).
    time = max(0, time)
    print("Time is: ", time)
```

Dans le cas des phases durant laquelle le joueur devra résoudre une énigme, une chanson commence et se termine après quelques minutes, cette approche est très bien.

Dans le cas où la lecture peut durer beaucoup plus longtemps, par exemple quand le joueur explorera la carte, le jeu finira par se désynchroniser et une approche différente est nécessaire.

On devrait utiliser l'horloge matérielle audio pour la synchronisation. Pour cela, on devra utiliser `AudioStreamPlayer.getPlaybackPosition()` pour obtenir la position actuelle du morceau semble idéal, mais ce n'est pas très utile en l'état. Cette valeur augmentera en morceaux (chaque fois que le rappel audio sera mélangé à un bloc de son), de nombreux appels peuvent donc retourner la même valeur. Ajouté à cela, la valeur sera également désynchronisée avec les haut-parleurs pour les raisons mentionnées précédemment.

Pour compenser la sortie «fragmentée», il existe une fonction qui peut nous aider : `AudioServer.getTimeSinceLastMix()`.

L'ajout de la valeur de retour de cette fonction à `getPlaybackPosition()` augmente la précision :

Pour augmenter la précision, nous alloçns devoir enlever les informations de latence (c'est-à-dire combien il faut de temps pour que l'audio soit entendu après son mixage) :

```
var time = $Player.get_playback_position() + AudioServer.get_time_since_last_mix()
```

```
var time = Player.getPlaybackPosition () + AudioServer.getTimeSinceLastMix () -  
          AudioServer.getOutputLatency ()
```

Le résultat peut être un peu nerveux en raison du fonctionnement de plusieurs threads. Vérifiez simplement que la valeur n'est pas inférieure à celle de l'image précédente (jetez-la si c'est le cas).

C'est également une approche moins précise que la précédente, mais elle fonctionnera pour des chansons de n'importe quelle longueur ou pour synchroniser n'importe quoi (effets sonores, par exemple) avec la musique.

2.7 Enigmes (Ilias B. et Benoit B.)

Nous avons, par la suite, recherché comment nous allions mettre en place nos énigmes. Nous nous sommes tout d'abord demandé ce que nous pouvions mettre en place. Ce qui nous a amené au schéma suivant : Un texte mettant en place “l'intrigue” de l'énigme, un endroit pour y répondre et de mini-menus disposant d'indices (qu'il faudra payer à l'aide de la monnaie du jeu), et un autre pour quitter. Le menu des énigmes suivant a alors été fait afin de palier à ces besoins.



Pour la prochaine soutenance, le but sera d'implémenter ce menu et ses sous-menus afin de rendre notre interface d'énigmes utilisable pour le reste du développement du jeu.

Nous avons pour le moment élaboré l'éénigme de la tour qui sera l'une des énigmes principales afin d'accéder à une nouvelle zone du jeu. Nous devons donc modéliser la tour puis créer une éénigme d'observation afin de trouver des éléments nous permettant d'avancer dans l'intrigue.

De plus, nous avons réfléchi à quelques autre éénigmes que nous mettrons au centre du labyrinthe, donnant ainsi des indications afin de trouver la sortie du labyrinthe.

2.8 Personnage (Jonathan P.)

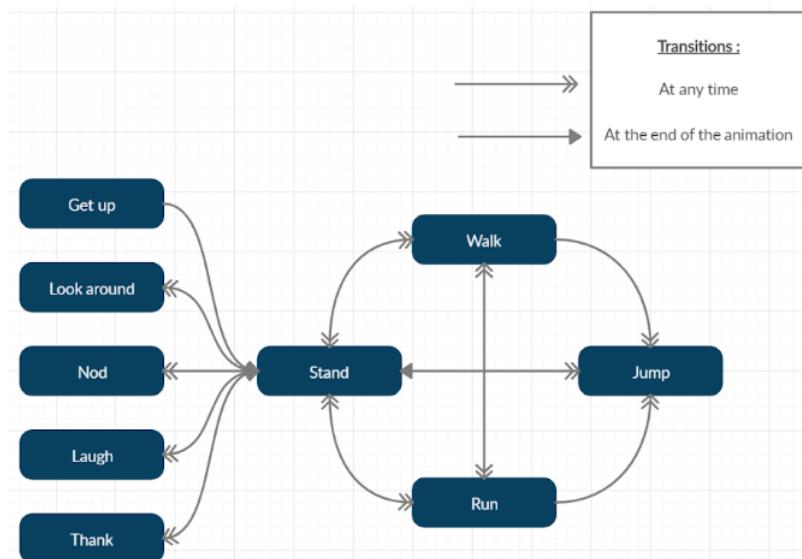
Modelisation et animations



Le modèle du personnage vient du site Free3D, un site internet qui répertorie des centaines de modèles 3D en tout genres. Les animations ont été implémentées avec Mixamo, un site permettant de rigger¹ automatiquement les modèles 3D et pouvoir leur faire exécuter des dizaines d'animations. Les animations choisies sont les suivantes :

- "Rester debout" avec un léger mouvement de respiration.
- "Marcher" et "courir" pour les déplacements basiques.
- "Sauter" et "tomber", ces actions ne sont pas définitives et ne seront pas forcément utilisées, la seconde disparaîtra certainement et n'est pas présente dans la gestion des animations.
- "Hocher la tête", "regarder les alentours", "remercier" et "rigoler" en réponse à certaines actions comme examiner un élément, parler à un PNJ, etc.
- "Se relever" ne sera utile qu'au début du jeu quand le personnage se réveille.

Pour ce qui est de la gestion de animations, après des tests avec une machine à état et par la programmation uniquement, le résultat étant quasiment identique, la solution par programmation a été retenue. Comme aucune de ces solutions ne semble avoir de réel avantage par rapport à l'autre un choix presque arbitraire a dû se faire et cette solution semblait plus intuitive et flexible. Bien que les animations soient gérées programmatiquement, on peut les représenter sous forme de machine à état comme ci dessous :



La position de base est donc "Stand", depuis laquelle toutes les autres sont accessible (sauf se relever comme précisé plus tôt). Pour les 4 animations où le personnage ne se déplace pas ainsi que le saut, il faut attendre la fin de l'animation et repasser en "Stand" avant de pouvoir faire autre chose. Pour la marche et la course, les deux peuvent se démarrer en état debout ou en déplacement, et l'on peut marcher ou courir puis sauter sans s'arrêter. Par la suite il faudra consilier toutes ces animaions avec les evenements du jeu, surtout des Input de l'utilisateur.

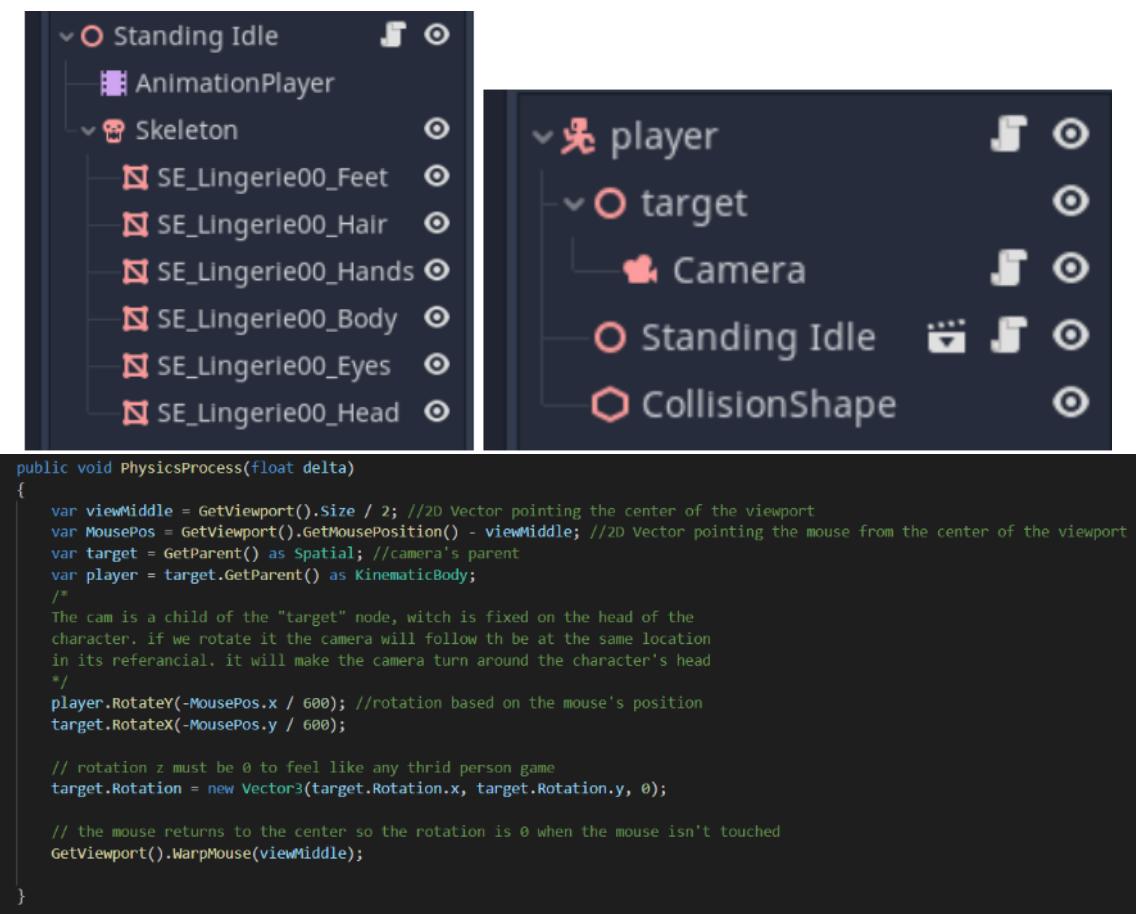
1. Le squelettage ou rigging est un procédé en animation 3D qui dote un objet à animer d'un squelette interne mobile qui déformera sa maille (mesh) de surface

Habits et tissus

Après de nombreux tests et essais, l'idée de base pour les habits du personnage a été modifiée. En effet une animation avec une simulation de tissus a un très bon rendu et semble vraiment réaliste mais il ralentit énormément le programme et est en général trop lourd pour un jeu vidéo. Nous allons donc opter pour un tissus simulé avec un simple soft body, avec une mesh créée sous Blender. Le capuchon sera en deux parties, la capuche et la cape, toute deux fixés au cou du personnage. Cette partie n'est pas encore terminée mais elle sera la prochaine priorité.

Caméra

Le principe de gestion de la caméra repose sur un noeud spatial (nommée target) placé environ au centre de la tête du personnage. Celui ci sert de référentiel à la caméra qui est son fils dans l'arborescence, donc lorsque l'on applique une rotation au noeud target, la caméra tourne pour rester au même endroit dans son référentiel local, résultant en une rotation autour de la tête du personnage. Afin de calculer la rotation en question, nous récupérons les coordonnées (x,y) de la caméra par rapport au centre du viewport, et on applique la rotation sur x au noeud target et celle en y au personnage, pour qu'il soit toujours droit à la caméra mais que celle-ci puisse tourner, puis on force la souris à se replacer au centre du viewport pour qu'il n'y ai pas de déplacement pendant que la souris ne bouge pas.



Déplacements

Pour les déplacements, on prend le vecteur allant de la caméra à la target, on lui retire sa composante en y et on le normalise, ce qui donne un vecteur unitaire dans la direction où est tournée la caméra. Ensuite on applique ce vecteur en translation au personnage pour le faire bouger. Pour cela, on utilise la fonction MoveAndCollide(Vector3*delta) du noeud KinematicBody (noeud permettant de créer des objets répondant aux collisions et aux lois de la physique). Cette fonction permet une translation qui réagit à certains objets comme les autres noeuds du type KinematicBody, ou bien les RigidBody par exemple.

```
public void Move(float delta)
{
    var cam = GetNode<Camera>("target/Camera").GlobalTransform.origin;
    var target = GetNode<Spatial>("target");
    Vector3 Strait = (target.GlobalTransform.origin - cam); //3D vector from the cam to the target
    Strait.y = 0; //makes it only on x and z axis
    Strait = Strait.Normalized(); //makes it one unit long

    if (Input.IsActionPressed("walk_front") ||
        Input.IsActionPressed("walk_right") ||
        Input.IsActionPressed("walk_left") ||
        Input.IsActionPressed("walk_back")) //include running behaviour
    {

        //if you are running you go twice as fast as when you're walking
        if (running)
        {
            | this.MoveAndCollide(Strait *20* delta);
        }
        else
        {
            | this.MoveAndCollide(Strait * 10 * delta);
        }
    }
}
```

Avancements futurs

Comme dit plus tôt, la priorité actuelle est le capuchon qui servira de vêtement au personnage. Cependant, d'autres tâches restent encore à faire. Voici les prochains ajouts lorsque la cape sera opérationnelle :

- La distance de caméra : dans les endroits clos (grotte, tour, etc.) ou lorqu'un mur se trouve derrière le personnage, il ne faut pas que la caméra passe à travers ou derrière l'obstacle, et donc qu'elle s'approche ou s'éloigne du personnage.
- Actions du personnage avec son environnement : il faut que le personnage puisse parler aux PNJ et examiner des objets, donc faire des actions relatives a des objets dans son champ de vision
- Cinématiques : Au début et à la fin, ainsi que peut être à d'autres endroits clef du jeu, le personnage devra enchaîner des animations avec un mouvement spécifique de caméra comme pour une cinématique.

2.9 Level Design

La première partie du jeu sera une plaine un peu valonnée avec seulement une ou deux énigmes pour bien apprendre à prendre le jeu en main et se lancer dans l'histoire, l'ambiance sera assez calme.

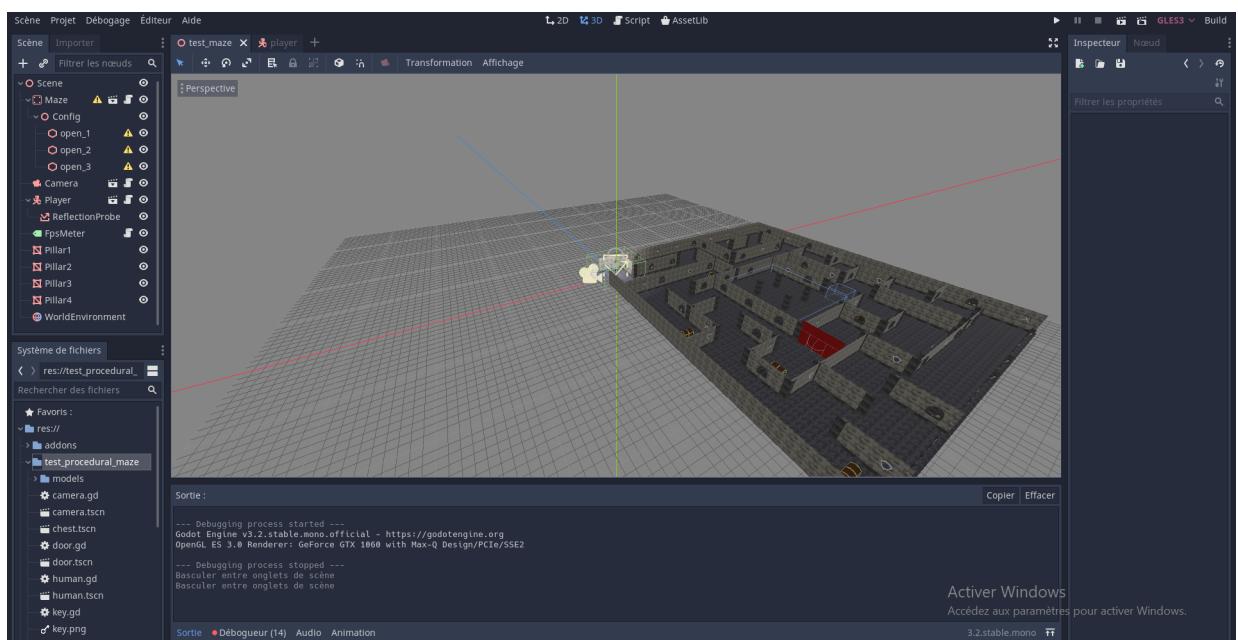
La partie de la grotte se passera en intérieur et aura une ambiance différente du reste de la map qui sera beaucoup plus ouverte. Dans cette partie l'ambiance sera lourde, presque glauque, et la luminosité sera très faible.

La partie du village aura aussi une ambiance un peu lourde mais surtout assez triste, et il faudra beaucoup fouiller pour continuer dans l'aventure.

La partie du Labyrinthe :

Pour commencer à former le labyrinthe de notre map, qui est un élément important dans l'intrigue, nous avons commencé à télécharger des addons permettant de le modéliser.

Puis, on utilise les variables “size-x” et “size-y” qui sont le nombres de lignes et de colonnes, “corridor-width” étant la largeur des lignes et des colonnes, “wall-width” la largeur des murs générés, “hauteur” la hauteur des murs afin de créer les proportions du labyrinthe. Nous avons ensuite créé un nouveau script qui contient “maze.gd” à partir du répertoire add-ons / procedural-maze et nous créons un noeud avec un nouveau StaticBody.



On utilise ce code pour initialiser le labyrinthe.

```
26
27 ~ func initialize(_size_x, _size_y, _corridor_width, _wall_width, _height):
28     size_x = _size_x
29     size_y = _size_y
30     corridor_width = _corridor_width
31     wall_width = _wall_width
32     height = _height
33     cells = []
34 ~| for x in range(size_x):
35     | var line = PoolIntArray()
36 ~| | for y in range(size_y):
37     | | line.append(0)
38     | cells.append(line)
39 ~| objects = []
40
```

Pour la seconde soutenance, on souhaite améliorer notre labyrinthe en lui assignant de meilleures textures ressemblant plus au “low-poly” que l’on souhaite, et enfin le complexifier afin que cela apporte plus de difficultés au joueur afin de sortir du labyrinthe.

2.10 Site web (Jonathan P.)

Le site Web est assez simple, il possède un menu fixe sur toute les pages et chacunes d’elles un titre et du contenu. Les différentes pages sont les suivantes :

- Accueil : avec l’artwork principal du jeu.
- Avancement du projet : des images de code, de recherches et de sprites mises tout au long du projet.
- Téléchargement : pour pouvoir télécharger le jeu une fois fini.
- Galerie : des images in-game seront mises quand le projet sera terminé, il y aura probablement un trailer.
- Soluce : tous les emplacements des objets et énigmes, avec la solution de celles-ci.
- Contacts : des informations sur l’équipe de développement et de quoi nous contacter.

Le site sera alimenté tout au long du projet et n’a pour l’instant que peu de contenu. Toutes les pages ont le même squelette (Le header avec le logo et le titre de la page, le menu avec toutes les différentes pages du site, c’est la seule partie qui ne scroll pas avec le reste, le contenu qui englobe tout ce qui est relatif à cette page, et enfin le Footer avec les noms des membres de l’équipe). Le Css est le même pour toutes les pages.

3 Récit de la réalisation

Général

Dans un premier temps, il a surtout fallu appréhender et comprendre Godot, qui est très différent de tous les environnements de développement que nous avions pu utiliser. Cela s'est révélé assez frustrant au tout début de ne rien comprendre, mais après une revue en cours de programmation, nous avions suffisamment de bases pour nous y lancer pleinement. Le système de noeuds et Godot en général est au final assez agréable à utiliser.

Jonathan Poelger

La partie site web a été assez courte et n'a pas posé de problèmes particulier car le site n'est pas très complexe. Pour ce qui est du personnage, j'ai eu des difficultés à beaucoup d'étapes. Au tout début il a fallu choisir entre beaucoup de modèles différents et nous avons fini par opter pour un modèle avec des graphismes peu détaillés mais pour autant assez réalistes. Quand est venu le temps de le faire s'animer, le choix des animations à utiliser a été assez long mais l'implémentation sur Godot s'est fait assez facilement car elle est plus ou moins automatique. La gestion des animations n'a pas été très complexe, quelques tests sur une machine à états et par le code ont suffit. Le problème principal s'est posé lors de l'implémentation des déplacements et de la caméra : j'ai trouvé pour la caméra une astuce avec un objet target sur la tête du modèle du personnage car je ne comprenais pas vraiment les exemples sur internet, étant tous en gdscript. Le problème a été l'ordre des rotations sur les différents axes, puis le mouvement du personnage en fonction de la position de la caméra. Cependant, le problème qui m'a tenu en haleine le plus longtemps (presque 4 semaines) a été la rotation du personnage pour qu'il se tourne vers là où il se déplace, un problème d'autant plus étrange qu'il semble très simple à résoudre. LookAt(), Rotation(), Rotate(), RotationY(), LookAtFromPosition(), RotationDegrees(), etc. , rien n'y faisait, il y avait à chaque fois des bugs en tout genre, allant du crash complet à la rotation aléatoire extrêmement rapide en passant par un blocage de la direction de la caméra. Au final la solution a tenu en une ligne, il fallait gérer la rotation du modèle en même temps que la rotation de la caméra. Au final malgré les problèmes, je trouve ça vraiment très satisfaisant de voir le travail prendre vie et de voir le projet prendre doucement forme grâce au travail de tout le groupe.

Alexandre Martinez

Nous avons d'abord fait un croquis de la map tous ensemble et j'ai trouvé cela sympa de commencer tous ensemble. Je me suis dit que ce serait génial si je m'occupais de modéliser ce que nous avons dessiné.

D'ailleurs j'ai dû faire la map deux fois à cause de problèmes de fichiers mais je trouve cela intéressant car cela m'apprend à modéliser et le fait de donner forme à notre dessin est satisfaisant.

De plus, j'ai fait des recherches d'assets pour créer notre environnement et j'aime beaucoup les assets que j'ai trouvé durant mes recherches pour la partie graphique. Mon gros problème se trouvait sur le menu car j'ai mis du temps avant de réussir à régler mes problèmes d'héritage entre les différentes scènes. En revanche tout fonctionne parfaitement maintenant donc je n'ai plus de soucis là dessus. Je sens que le projet commence plutôt bien pour une première utilisation de Godot malgré nos problèmes car pour l'instant je vois que ce que j'ai fait marche.

Ilias Belkhder

J'ai dû proposer plusieurs boules de couleurs différentes aux membres du groupe et les faire refaire à plusieurs reprises afin de remplir nos exigences artistiques.

J'ai par ailleurs dû faire plusieurs recherches afin de savoir comment faire l'étendue d'eau. Au départ, j'hésitais à faire une mer avec des vagues qui ondulent. Cependant, au fur et à mesure que je découvrais la méthode pour la réaliser, contenant des fonctions cosinus et sinus hyperbolique, je me rendis compte de la difficulté de cette tâche puis je suivis le conseil de mon professeur de programmation de faire un shader.

Ensuite vint le problème de notre "boîte à énigmes". Il fallait d'une part faire quelque chose qui est assez beau graphiquement, mais aussi quelque chose qui pourrait contenir un texte, avec des indices, et des outils pour y répondre. Après plus d'une dizaine d'essais, nous avions enfin réussi à trouver quoi appeler la "boîte à énigmes" de notre jeu.

Benoît Brice

Nous avons d'abord commencé à se mettre d'accord vis à vis du scénario ainsi que de la map, et j'ai trouvé cela vraiment intéressant car nous respections les idées de chacun tout en se mettant d'accord.

J'ai ensuite commencé à faire des recherches sur la création de map 3D en "Open World" et j'ai eu beaucoup d'informations grâce aux réseaux sociaux dont les membres de certains groupes m'ont été d'une grande aide.

J'ai tout de même eu des problèmes avec la création de la map en utilisant plusieurs moyens tels que des générateurs aléatoires de map, ou bien en utilisant des addons. Mais au final, on s'est répartie le travail et s'était plus faisable.

Par la suite, j'ai commencé à chercher un moyen de créer le labyrinthe, un élément important dans notre jeu et j'ai encore une fois eu l'occasion de bénéficier de l'aide des membres de mon groupe ainsi que d'autres personnes expérimentées afin d'aboutir à un résultat satisfaisant.

4 Deuxième soutenance

La prochaine soutenance nous permettra de terminer, en théorie, la map et de peaufiner certains éléments. En effet, nous prévoyons pour cette soutenance de rajouter les éléments de gameplay que nous avions prévu au début du développement du jeu et de corriger les bugs au maximum ou du moins de les minimiser. Les éléments à rajouter en priorité sont :

- les décors et textures de la map
- les premières énigmes, donc le menu de celles-ci, ainsi que leur accessibilité depuis la map (présence de panneaux d'affichage, décors ou PNJ lançant l'énigme)
- les derniers éléments du personnage, il devra approcher de la fin en termes de finitions, donc avoir les vêtements, les actions avec l'environnement et les cinématiques
- les premiers effets sonores et musiques de fond
- le menu terminé, sauf la partie options
- le système de sauvegardes

5 Conclusion

Pour cette première soutenance, nous avons réussi à globalement atteindre les objectifs que nous nous étions fixés dans le cahier des charges. Même s'il nous reste encore beaucoup de travail, nous avons déjà des bases solides ainsi que les principales mécaniques de jeu, nous permettant d'avoir d'ores et déjà un aperçu global du projet. La suite sera majoritairement un travail de création, et d'interactions entre les éléments du jeu, en effet toutes les parties individuelles doivent fonctionner ensemble. nous ne sommes donc limités que par le temps.