

Rapport de deuxième soutenance

Nom de Projet : *Pricefield Garden*

Nom de groupe : Arcadia

Belkhder Ilias

Martinez Alexandre

Poelger Jonathan

Brice Benoît

Avril 2020



Encadrants : Remi Vernay, Philippe Roussille

Table des matières

1	Introduction	2
1.1	Cahier des charges	2
1.2	Résumé de l'avancée	2
2	Découpage du projet	3
2.1	Avancement des tâches	3
3	Avancement	4
3.1	Menu (Alexandre M.)	4
3.2	Menu in-game	6
3.2.1	Graphique (Ilias B.)	6
3.2.2	Fonctionnement (Jonathan P.)	6
3.3	Objets Examinables (Jonathan P.)	8
3.4	Enigmes (Ilias B.)	9
3.5	Personnage (Jonathan P.)	10
3.5.1	Graphiques	10
3.5.2	Camera	11
3.5.3	Animations	11
3.5.4	Esprit	12
3.6	Map (Benoit B.)	13
3.7	Site Web (Jonathan P.)	16
4	Récit de la réalisation	17
5	Soutenance finale	19
6	Conclusion	19

1 Introduction

Le projet Pricefield Garden a connu un bon nombre d'évolution pendant cette seconde période, entre la première et la seconde soutenance. Elle a permis de développer davantage de fonctionnalités et d'améliorer celles d'ores et déjà existantes. Notre jeu ressemble désormais plus à l'idée que nous en avions lors de l'écriture du cahier des charges. Les avancées faites jusqu'à maintenant ont été possibles à l'aide des différentes connaissances que nous avons acquis en travaux pratiques à l'école, ainsi qu'à l'aide de nos recherches personnelles sur Godot et les outils que nous utilisons dans le cadre de ce projet. L'avancement à ce jour sera davantage détaillé dans la suite de ce rapport.

Pour rappel, 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.

1.1 Cahier des charges

Actuellement, nous ne prévoyons aucune modification quant au cahier des charges, sauf pour la cape du personnage qui est annulé ou que nous ferons seulement si nous finissons le projet à l'avance.

1.2 Résumé de l'avancée

L'avancement de notre travail pour cette deuxième période est détaillé ci-dessous dans leurs sections respectives. Ces sections correspondent aux catégories de fonctionnalités que nous avions mentionnés dans le cahier des charges. Pour chaque section qui s'y prête, le développement de chaque fonctionnalité sera détaillé individuellement, en précisant leur fonctionnement ou leurs enjeux techniques, ainsi que les contributeurs à ces dernières.

2 Découpage du projet

2.1 Avancement des tâches

Ce tableau montre la répartition actuelle des tâches.

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

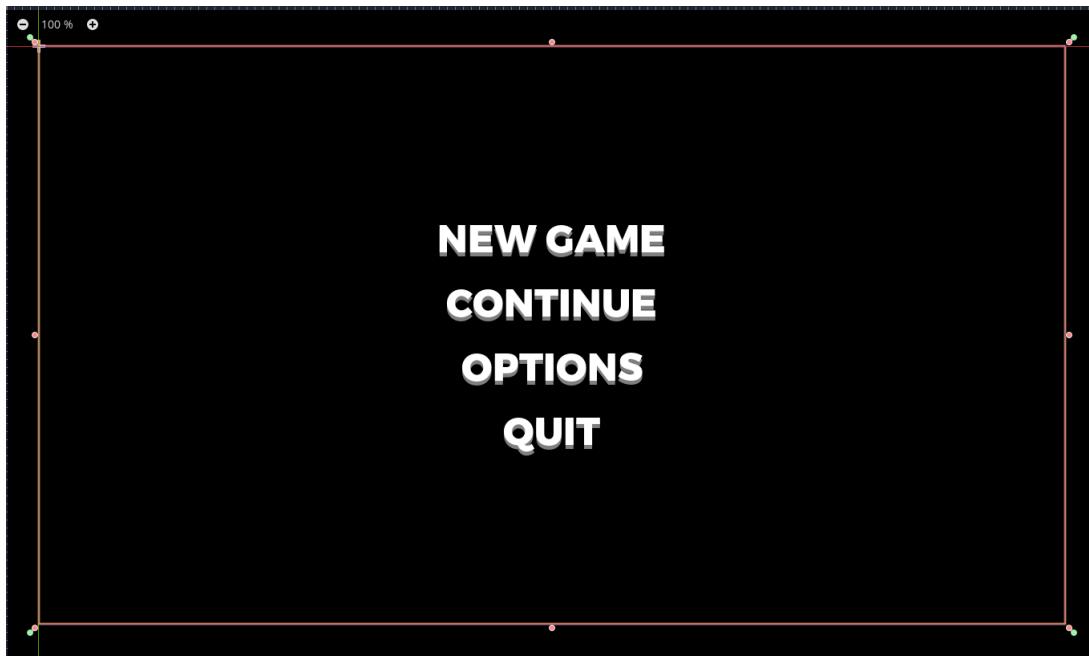
Ce tableau montre l'avancée actuelle des différentes tâches du projet.

Tâches	Avancement réel	Avancement prévu
Menu	70	50
Interface joueur	70	50
Graphismes	75	66
Soundtrack	25	50
Enigmes	70	75
Personnage	90	100
Sauvegarde	0	50
Level design	100	100
Story telling	100	100
Site Web	90	100

3 Avancement

3.1 Menu (Alexandre M.)

Voici l'avancement du menu principal.



Je ne me suis pas attardé sur la partie graphique du menu mais plutôt sur le code en C# qui permet le fonctionnement des boutons.

Premièrement, le bouton NEW GAME envoie le joueur sur la map du jeu :

```
1  using Godot;
2  using System;
3
4  public class NewGameButton : Node
5  {
6      private void _on_NewGameButton_button_up()
7      {
8          GetTree().ChangeScene("res://NewGame.tscn");
9      }
10 }
```

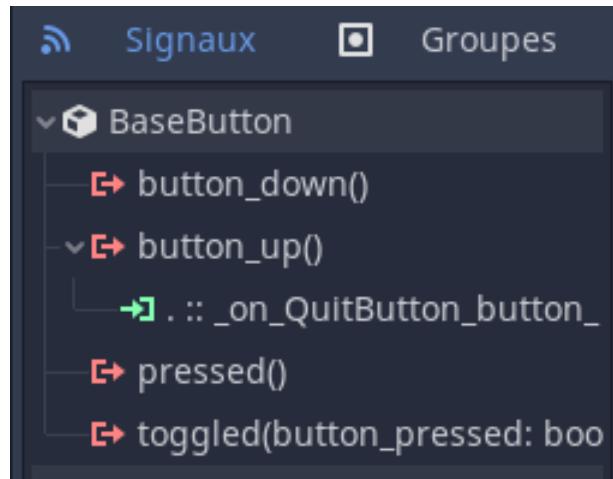
Ensuite, le bouton CONTINUE marche de la même façon sauf qu'il utilise la sauvegarde, je dois m'occuper de cela pour la prochaine soutenance.

Les options sont également à faire pour plus tard. Ce n'est pas un point important du menu principal.

Enfin, le bouton QUIT est également fonctionnel. Il suffit de quitter le jeu comme ceci :

```
1  using Godot;
2  using System;
3
4  public class QuitButton : Node
5  {
6      private void _on_QuitButton_button_up()
7      {
8          GetTree().Quit();
9      }
10 }
```

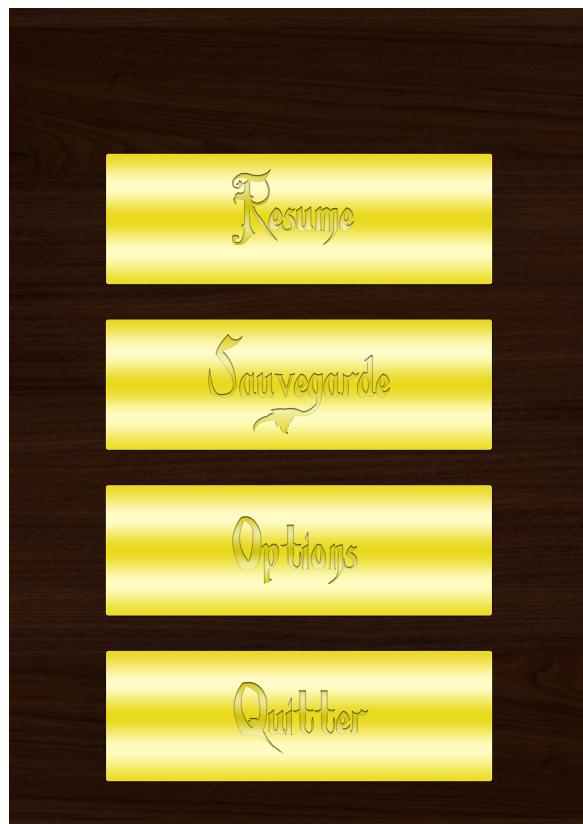
Mon plus gros problème a été de comprendre les différents signaux de Godot. En effet j'en utilise pour relier les scripts aux boutons mais cela avait l'air assez étrange à première vue. Voici quelques exemples de signaux utilisés :



3.2 Menu in-game

3.2.1 Graphique (Ilias B.)

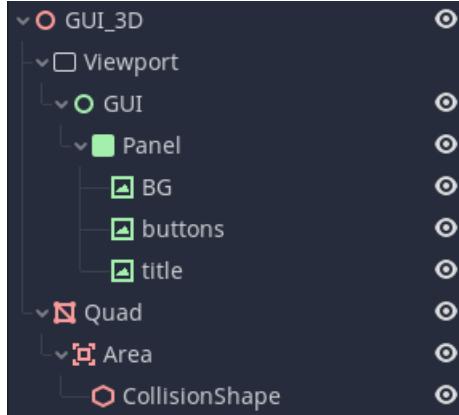
Pour cette soutenance, je me suis occupé de la création graphique du Menu in-game, en veillant à faire des couleurs raccords à celles-déjà présente pour notre “boîte à énigmes” afin de donner une cohérence globale à la partie graphique de notre jeu. Il est composé de “Resume” qui permettra à l’utilisateur de revenir au jeu, de “Sauvegarde” qui permettra au joueur de garder sa progression ou encore d’”Options” qui permettra au joueur de naviguer parmis les différents paramètres disponibles tel que le son ou encore de modifier ses touches.



3.2.2 Fonctionnement (Jonathan P.)

Afin que je joueur puisse sauvegarder, retourner au menu ou aller dans les options, un autre menu que le principal est nécessaire, un menu “in-game”. Celui-ci s'affiche lorsque le joueur appuie sur `echap` (un booléen change de valeur : à `true` on rend le menu visible et on appelle la fonction qui gère le menu, à `false` on le rend invisible et on appelle la fonction qui gère le déplacement. Il est placé dans la scène du personnage, de sorte à toujours faire face à la caméra), et on navigue dedans avec les contrôles pour avancer et reculer le personnage (`w` et `s`), et entrée sert à valider le choix.

Afin de pouvoir afficher ce menu, il a fallu simuler de la 2D dans un monde en 3D. Pour que cela soit plus simple, tous les procédés sont encapsulés dans une scène de type simple : spatial. Le type dont dérive tous les objets 3D.



Cette scène est donc composée de deux grandes parties : la partie qui “existe” dans le monde en 3D, et la partie qui affiche du 2D. L’objet Viewport permet d’afficher un écran dans un monde en 3d, ici cet écran montre le menu. les objets fils du Viewport sont les éléments à afficher. Le “Quad” est une mesh (un objet avec une forme) 3D et cela permet d’avoir un support pour le Viewport. Ces deux objets ensemble permettent donc d’afficher de la 2D dans un monde en 3D.

La dernière chose à prévoir est ce qui va permettre à l’utilisateur de naviguer dans le menu. Pour cela, une variable peut aller de 0 à 3 (4 renvoie à 0 et -1 renvoie à 3), et pour chaque valeur, deux flèches se placent de chaque côté du menu actuellement sélectionné, et la touche entrée déclenche une action différente :

- Resume : ferme le menu
- Save : Sauvegarde la partie (non implémenté)
- Options : ouvre le menu des options (non implémenté)
- Quit : retourne au menu principal

Il reste donc à connecter ce menu aux éléments auxquels il doit accéder, et lorsque ces derniers seront en place, ajouter des éléments de “confort” de jeu, comme des popup si l’on s’apprête à quitter une partie non sauvegardée par exemple.

3.3 Objets Examinables (Jonathan P.

Les objets examinables sont les objets qui lanceront les énigmes. Ils peuvent prendre n'importe quelle forme étant donné qu'ils sont de type mesh 3D, et possèdent plusieurs capteurs :

- une “area” qui capte quand le joueur est proche de l'objet
- un “visibility notifier” qui capte si l'objet est proche du centre du champ de vision du joueur

Si ces deux conditions sont vérifiées, un point d'exclamation apparaît au dessus afin de signaler au joueur qu'une action peut être réalisée.

```
public override void _Process(float delta)
{
    var expoint = GetNode("exclamPoint") as MeshInstance;
    var visibility = GetNode("VisibilityNotifier") as VisibilityNotifier;
    var area = GetNode("Area") as Area;

    //on vérifie que l'objet est visible
    insight = visibility.IsOnScreen();
    //le point d'exclamation apparaît si l'objet est visible et proche
    expoint.Visible = inSight && near;
    //si l'objet est visible et provoque que le joueur appuie sur la touche action
    //on émet un signal, qui sera récupéré dans le code de la map
    action = inSight && near && Input.IsActionJustPressed("action");
    if (action)
    {
        EmitSignal("Action");
    }
}

//L'objet Area marche avec des signaux naturellement: il peut en envoyer lorsque l'on
//y entre ou lorsque l'en sort. ici cela permet de déterminer si le joueur est
//"proche" de l'objet
0 references
public void _on_Area_body_entered(Node body)
{
    near = true;
}

0 references
public void _on_Area_body_exited(Node body)
{
    near = false;
}
```

Le système de signaux de Godot permet deux choses :

- Déetecter certaines actions relatives aux noeuds existants. Ces signaux sont déjà existant lorsque l'on crée un nouvel objet, c'est le cas de onAreaBodyEntered() par exemple.
- Faire le lien entre des codes attachés à différents objets. Dans ce cas on parle de signaux personnalisés, qui appellent des fonction dans n'importe quel code actif, et peuvent même faire passer des arguments. Ici, EmitSignal("Action") appellera la fonction onExaminableAction() qui se trouve dans un autre script.

Les fonctions appelés par des signaux sont appelées des listener, comme il en existe dans beaucoup d'autre domaines, mais sont la majorité du temps utilisées pour réagir aux actions de l'utilisateur. Ces signaux sont ici très pratique de par leur facilité d'utilisation et la large panel d'utilisation possible grâce à la personalisation.

3.4 Enigmes (Ilias B.)

Nous allons vous montrer dans cette partie comment a été fait le système d'énigmes.

```
10     private uint EnigmeNumber;
11     private TypeE type;
12     private string awnser;
13     private string title;
14     private string text;
15     private bool complete;
16
17     private Tuple<TypeE, string, string, string, bool, string[]>[] enigme;
18     //          Type   Title   Text    Awnser   complete   Buttons
19
20     private Label Title;
21     private Label Text;
22     private CheckBox Awnser1;
23     private CheckBox Awnser2;
24     private CheckBox Awnser3;
25     private CheckBox Awnser4;
26     private LineEdit AwnserEdit;
27     private Control Quiz;
28     private Control MCQ;
29
30
31     public override void _Ready()
32     {
33         Title = GetNode("Viewport/GUI/Title") as Label;
34         Text = GetNode("Viewport/GUI/Text") as Label;
35         Quiz = GetNode("Viewport/GUI/Quiz") as Control;
36         MCQ = GetNode("Viewport/GUI/MCQ") as Control;
37
38         if(type == TypeE.MCQ)
39         {
40             Quiz.Visible = false;
41             Awnser1 = GetNode("Viewport/GUI/MCQ/Awnser_1") as CheckBox;
42             Awnser2 = GetNode("Viewport/GUI/MCQ/Awnser_2") as CheckBox;
```

Chaque énigme aura un type (elle sera soit à choix multiples, soit le joueur devra entrer une valeur), un titre, un texte la décrivant et un booléen indiquant si oui ou non le joueur a complété l'énigme. Ensuite, chaque énigme est entrée dans une liste de tuples contenant le type(enumération), le titre(string), l'énoncé de l'énigme(string), la bonne réponse(string), l'état de complétion de l'énigme(bool) et les choix de réponses(string[]) s'il s'agit d'un QCM, null sinon). Pour faire apparaître l'énigme il suffira ainsi de faire appel à la liste à une n-ème place.

Ensuite Le titre et l'énoncé iront se mettre dans les cases prévues à cet effet, le système de réponse se mettra en place, et lorsque le joueur tentera une réponse, ce qu'il a écrit sur le bouton ou dans l'éditeur de texte sera comparé au string réponse, et il pourra recommencer en cas d'échec ou bien continuer son aventure, revenant au jeu, s'il a réussi. Cela sera aussi utile à la sauvegarde plus tard, car il suffira de stocker dans un fichier texte (en plus de la position du joueur) une liste des énigmes réussies ou pas, pouvant être une suite de 0 par exemple.

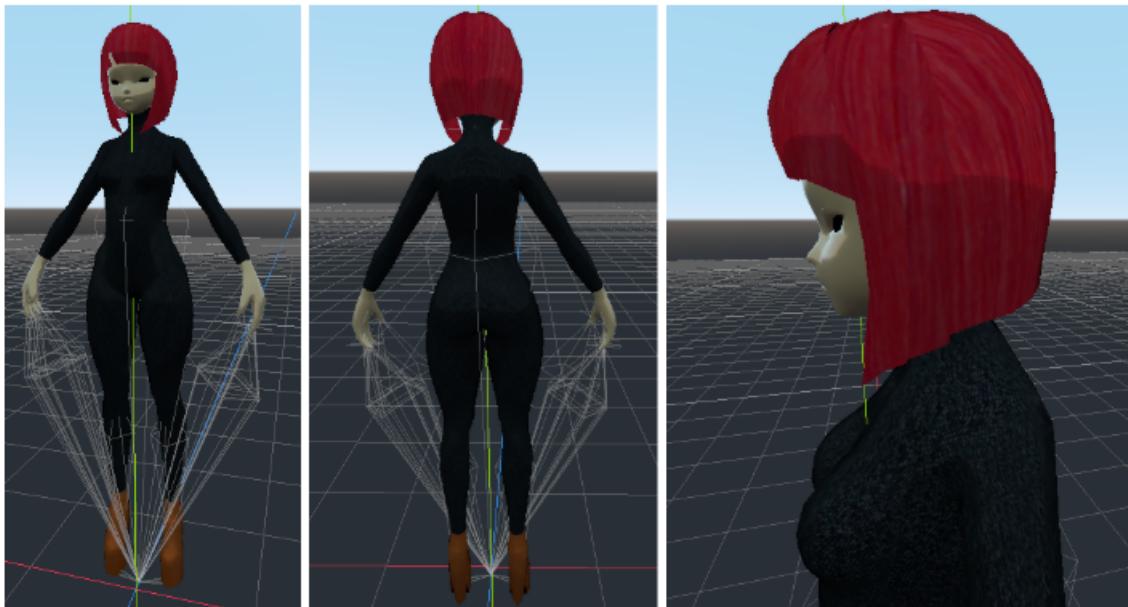
Avances et Retards : Au vu de l'avancée du projet, on peut considérer que l'implémentation des énigmes est presque finie en ce qui concerne les fonctionnalités basiques mises en place dans le jeu. C'est-à-dire l'affichage d'un texte, d'un titre et de la possibilité de résoudre l'énigme. Nous avons donc théoriquement dépassé les prévisions.

Prévisions : Néanmoins, si les fonctionnalités nécessaire au jeu actuel sont implémentées, cela ne veut pas dire que la partie des énigmes est bouclée. En effet, il nous faudra donner à notre code les informations de chaque énigme et là et seulement là, cette partie sera finie. De plus, nous avons commencé à réfléchir à de nouveaux éléments de gameplay qui nécessiteront potentiellement un retour sur cette partie et son fonctionnement. Nous réfléchissons à ajouter des indices notamment.

3.5 Personnage (Jonathan P.)

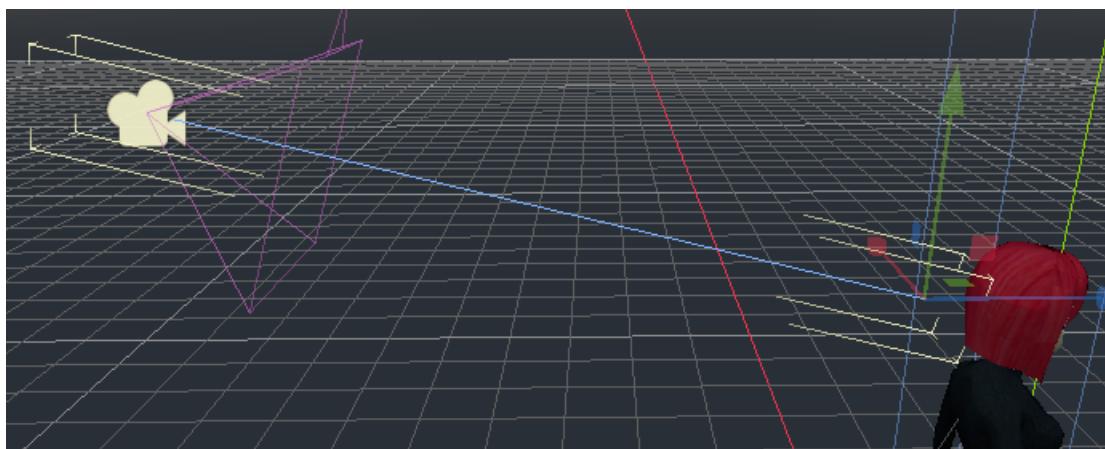
3.5.1 Graphiques

Pour la partie purement graphique du personnage, sur un commun accord, nous avons décidé de ne pas faire une longue cape au personnage, car cela n'était pas un élément crucial du jeu et apporte beaucoup de problèmes. Les simulations de tissus sont peu adaptées aux jeux vidéos car très gourmandes en performances, et les différents soft body (objets mous) sur Blender ou Godot ne sont pas adaptés aux vêtements. En effet, si l'on veut créer une petite cape accroché au dos d'un personnage et qui virevoltera, cela ne pose pas de problèmes. Mais si nous le voulons, comme c'est le cas ici, un vêtement qui couvre tout le corps et qui reste plus ou moins en place, cette solution est peu adapté, car mettre peu de contraintes à un soft body le fait voler au moindre mouvement et en mettre beaucoup cause des problèmes de collision et des bug graphiques. S'il reste du temps en fin de projet, il pourrait être consacré à cela, mais en attendant une autre solution a été retenue : des textures à même le personnage, qui font effet d'un habit moulant en tissus pour le corps, des cheveux roux, les chaussures à talon en cuir et de la peau beige.



3.5.2 Camera

La caméra était presque opérationnelle mais il manquait la réaction au décors, afin qu'elle ne passe pas à travers les murs mais qu'elle se rapproche lorsque le personnage est dos à un obstacle. Pour cela, il existe un objet godot qui remplit très bien cette tâche et qui a d'ailleurs principalement été créé pour les caméras à la 3e personne comme la nôtre, dont la gestion est soumise à beaucoup plus de contraintes que les caméras à la première personne ou même les caméras 2D. Cet objet s'appelle une "SpringArm", et son principe est le suivant : sa base est fixe et un rayon rectiligne en sort dans une direction choisie, avec une distance maximale. Tous les objets enfants sont placés au bout du rayon, et si quelque chose vient couper ce dernier, les enfants sont avancés au point où le rayon est coupé.



3.5.3 Animations

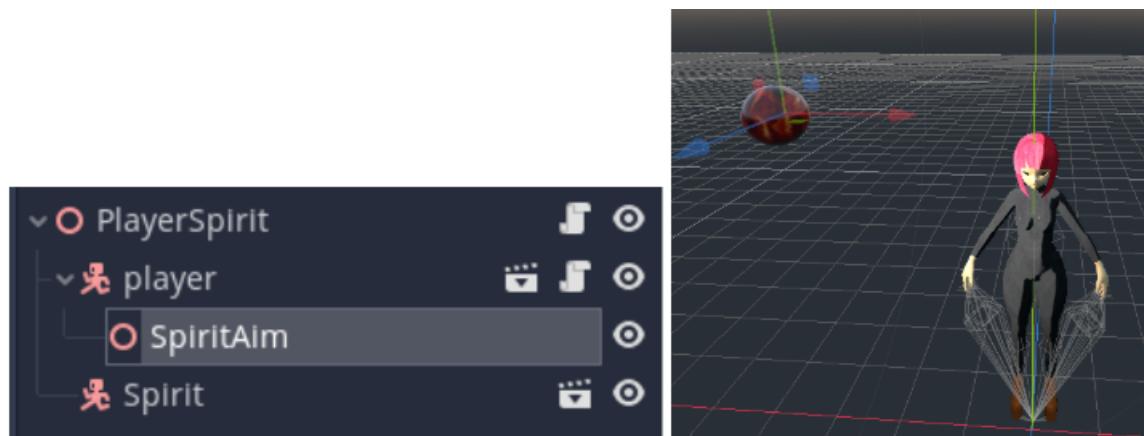
Pour les animations, celles qui étaient inutiles d'un point de vue de gameplay supprimées (le saut, la chute, regarder autour de soi), et les interactions entre les autres améliorées (marcher en arrière joue l'animation de marche inversée, il est impossible de courir en arrière, à la fin d'une animation qui n'est pas un déplacement, le personnage se remet à marcher au lieu de glisser sur le sol, etc.).

Les animations restantes sont les suivantes :

- marcher et courir, la base du déplacement
- rester sur place, pour ne pas que le modèle soit parfaitement immobile
- hocher la tête, lorsque l'on interagit avec un élément examinable
- se relever, seulement au tout début du jeu
- rire malsain, seulement pour la scène de fin du jeu
- remercier, à la fin d'une énigme

3.5.4 Esprit

Le personnage est en permanence suivi par un esprit qui l'aide tout au long de l'histoire. Il prend la forme d'une sphère lumineuse et flotte près du personnage. Il s'agit d'une sphère avec une texture de lave/feu et une source de lumière au milieu, ce qui permettra notamment d'illuminer les endroits sombres. Pour pouvoir suivre le personnage, ils sont tous les deux enfants d'une même scène, qui sera celle instanciée dans le jeu, afin qu'ils aient la même position dans l'espace mais qu'ils bougent indépendamment de l'autre.



L'esprit se dirige vers "SpiritAim" qui est fixe par rapport au personnage et bouge en même temps que ce dernier, et qui est situé en haut à droite de ce dernier. Sur les axes x et z (les horizontales) il se dirige donc vers la SpiritAim, et sur l'axe y (le verticale) il suit une sinusoïde.

Voici le code associé et son explication :

```
//On crée le vecteur allant de la position réelle à la position souhaitée.
var aim = Starget.GlobalTransform.origin - Spirit.GlobalTransform.origin;
//On réduit sa taille (et donc la vitesse de l'esprit).
//Le choix de ne pas utiliser la méthode .Normalized() viens du fait que l'esprit doit
//rester à proximité du joueur, donc plus il est loin, plus il est rapide.
|   aim = aim/40;
//une fonction cosinus sur l'axe vertical créera l'effet de flottement
float Ylevit = (float)Math.Cos(Math.PI*time/120)/20;
//On crée le vecteur de déplacement final, avec à l'horizontale déplacement
//vers l'aim et à la verticale le cosinus
var levit = new Vector3(aim.x, Ylevit, aim.z);
//Enfin, on applique ce vecteur en translation à l'esprit
Spirit.Translate(levit);
```

Ces actions sont appelées à chaque frame, la variable time s'incrémentant donc à chaque frame¹. C'est pour cette raison que toutes les valeurs sont très réduites, afin de privilégier les micro déplacement et garder un effet le plus fluide possible (le déplacement divisé par 40 et le cosinus par 20)

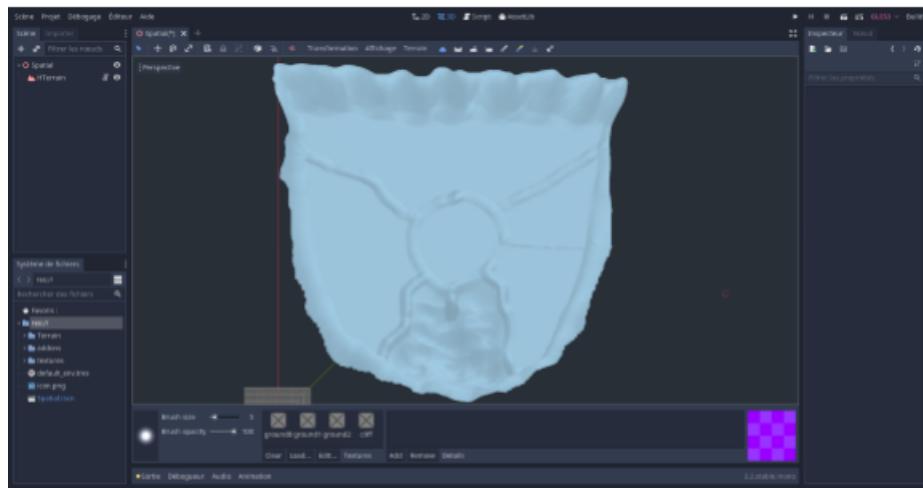
1. on ne se préoccupe pas du fait qu'elle puisse atteindre maxInt, cela prendrait environ 9942h de jeu d'affilée, soit un an, un mois et 18 jours

3.6 Map (Benoit B.)

Création de la map :

A la fin de la première soutenance, on avait élaboré un début de map qui devait nous servir de modèle sauf que celle-ci n'était pas exploitable car nous ne pouvions projeter notre personnage sur une map comme tel.

Ci-dessous l'ancienne map :



Nous avons donc, au début, décidé de le faire à partir de blender et de l'addon “Blender ESCN exporter” qui nous permettrait d’exporter tous nos objets 3D sur Godot.

Cependant, nous avons rencontré beaucoup de difficultés quand à l’élaboration de celle-ci sur Blender, donc nous avons cherché d’autres solutions.

Après de nombreuses recherches, nous avons remarqué qu'il existait un logiciel de création de map 3D très bien optimisé et qui nous permettait de l’exporter sur tous les moteurs de jeu ainsi que sur Blender. Ce logiciel se nomme World Machine.

Contrairement aux éditeurs de terrain traditionnels, World Machine utilise une approche procédurale. En effet, celui-ci peut former le terrain en utilisant des blocs de construction de base tels que des fractales.

On pouvait aussi appliquer des textures à nos différentes îles assez aisément après avoir formé celle-ci.

Après un peu de pratique, nous avons obtenu la map ci-dessous avec des textures :



Cependant, la map étant exactement comme on le souhaitait, nous devions ensuite exporter cette map sur Blender ou sur Godot afin de l'exploiter pour notre jeu. Pour cela, le logiciel nous a permis de mettre notre map sous le format .obj, qui était un format exploitable par Godot ou Blender.

Ensuite, nous devions créer un nouveau projet sous Godot, puis insérer le fichier .obj dans le même dossier que celui du projet qui a été créé par le logiciel.

Création de la forêt :

Nous avons ensuite cherché des assets d'arbre et d'herbe afin de commencer notre forêt, qui se situe dans l'île au nord de la map.

Nous avons décidé ensuite de prendre un modèle assez réaliste qui convenait à l'idée qu'on voulait de la map.

Sachant que la forêt au nord de la map devrait être dense, presque impénétrable, nous avons dû jouer avec la taille et l'épaisseur des arbres afin de la faire paraître la plus réaliste possible.

Après l'ajout des arbres, il nous fallait une texture de couleur verte qui nous donnerait l'impression d'être en forêt. Pour cela, nous avons repris la texture dont disposait notre map sur World Machine, puis nous sommes allés sur Blender, et avons appliqué un matériau à l'objet.

Une fois le matériau obtenu, il faut lui appliquer la texture que l'on a choisi, et enfin, choisir comment l'appliquer.

Nous avons obtenu la forêt ci-dessous après avoir réalisé la manipulation précédente :

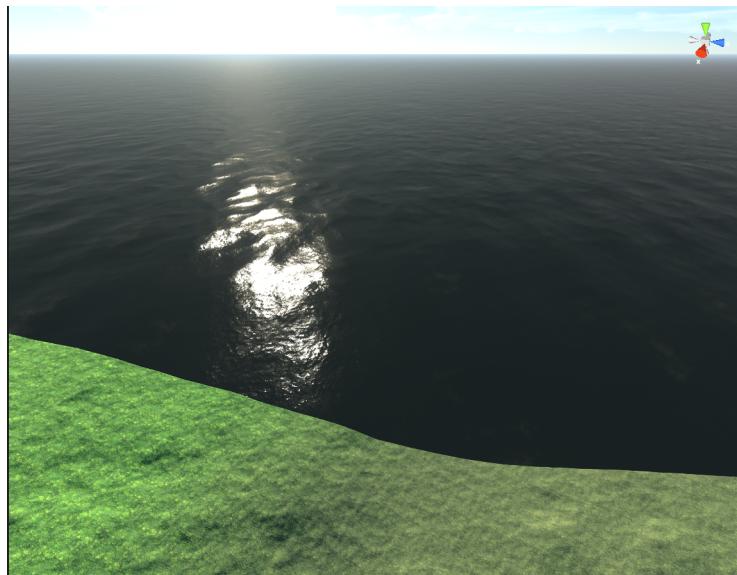


Une fois les textures et les arbres ajoutés, nous avons ensuite ajouté un modèle d'herbe réaliste afin de finir l'élaboration de cette forêt. Pour cela, nous avons procédé exactement de la même manière que pour les arbres.

Création de la mer :

Après avoir modélisé notre terrain, nous avons importé des assets que nous avons trouvé sur internet qui contenait plusieurs modèle d'eau afin de modéliser notre environnement et dans notre cas la mer.

Nous avions à notre disposition avec ces assets des outils de redimensionnement qui nous permettait d'étirer l'eau, c'est-à-dire modifier la largeur de l'eau ainsi que du niveau de l'eau par rapport au terrain. Après avoir importé ce modèle d'eau et appliquer les réglages qui nous convenait, on obtient une plage comme on peut l'apercevoir ci-après.



3.7 Site Web (Jonathan P.)

Le site internet est opérationnel, et possède les éléments important à son bon fonctionnement, même si la “beauté” graphique reste discutable. Un menu latéral permet de naviguer entre les pages, qui contiennent un titre et un contenu en plus de ce menu.

- L'accueil : Il contient l'artwork de base du jeu, celui qui a été créé au premier jour de ce projet. Evidemment, les idées ont changé depuis et le jeu sera sans doutes différent de cette idée.
- Le développement : Il contient les liens des rendus en PDF et du cahier des charges.
- La galerie : Elle présente un pèle-mèle d'images prises tout au long du projet.
- Téléchargement : Il contiendra le lien de téléchargement du jeu une fois celui-ci achevé.
- La soluce : Il y aura dans cette partie toutes les solutions des énigmes avec une explication lorsque celles ci-seront terminées.
- Les contacts : Les visages, noms, et adresses mail de toute l'équipe.

4 Récit de la réalisation

Général

La période entre la première et la seconde soutenance a surtout permis de continuer tous les aspects du jeu, qui avancent tous à un rythme différent. Cette période a été marquée par le confinement qui a causé des problèmes pour travailler à certains membres du groupe, mais dans l'ensemble tout avance plutôt bien, malgré qu'il y ait un peu de retard sur certaines parties clés du projet.

Jonathan Poelger

Cette soutenance a été pour moi l'occasion de finaliser le personnage et commencer à faire d'autres tâches, qui étaient assez différentes comme le menu in-game et l'avancement du site web. Le confinement ne m'a pas posé de problèmes particulier pour travailler, et j'ai pu trouver le temps d'avancer régulièrement en dehors des cours, TP, DM, révisions, etc. Je suis assez satisfait du personnage, les cinématiques se feront plus tard, mais s'intéresser à d'autres aspects en tout genres pose des problèmes bien différents et pousse donc à progresser dans d'autres domaines. La tâche principale maintenant sera de faire tout l'intérieur de la map, et nous y travaillerons sans doute tous. Cependant, je pense que le temps peut vite finir par manquer et il ne faudra surtout pas relâcher le travail pour pouvoir finir le projet à temps.

Alexandre Martinez

Pour ma part je me suis occupé du menu principal (surtout le code en C#). J'ai rencontré quelques difficultés à comprendre l'utilisation des signaux de Godot mais sinon j'ai bien aimé créer un menu et cela n'a pas été très compliqué.

Pour la prochaine soutenance je compte faire la sauvegarde ainsi que les options du menu même si les options ne sont pas très importantes pour le moment.

Je trouve cela sympathique et enrichissant de faire un projet comme celui-ci qui est assez complet. Malheureusement le confinement n'aide pas trop mais on essaye d'avancer comme on peut. Nous espérons pouvoir finir notre projet à temps malgré cette contrainte.

Benoît Brice

Durant cette période de la première soutenance à la seconde, je me suis surtout occupé de l'élaboration de la map.

Au début, malgré les recherches, on a pas trouvé de moyens de créer la map, mais par la suite, avec un peu d'aide extérieure au groupe, j'ai pu trouver des solutions à nos problèmes.

Je suis très content d'avoir pu travailler durant cette période à l'élaboration de cette map car je trouve cela très intéressant voir divertissant. Ainsi, je prends actuellement beaucoup de plaisir sur ce que je produis.

Cependant, j'ai rencontré beaucoup de difficultés à exporter notre map sur Godot car mon ordinateur est peu performant.

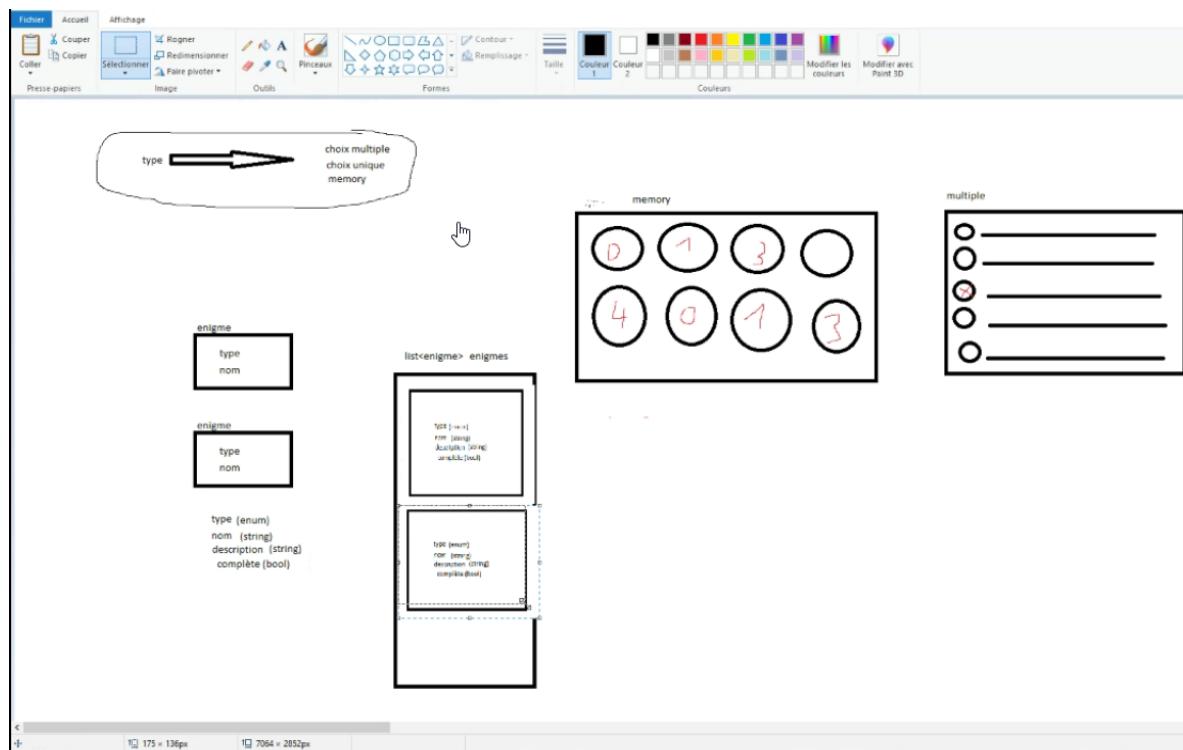
J'ai aussi eu des difficultés due à mes problèmes de connexion.

Ilias Belkhder

Pour cette soutenance j'ai été confronté à quelques problèmes pour faire mes tâches. En effet, d'une part, comment réaliser le menu in-game du jeu en bonne et due formes sans tablette graphique ? Suite à cela, il fallait tenter de faire un brouillon de ce menu : À quoi allait-t-il ressembler ? Qu'allait-il montrer/faire ressentir au joueur ? Les réponses à ces questions trouvées, je me suis atteler à la tâche au plus vite.

Par la suite, ce fut le tour des énigmes, comment est-ce qu'on allait réussir à les implémenter ? Comment les réunir autour d'un même code mais pouvoir rendre leur utilisation et appel facile ? Je me suis ensuite entretenu avec Jonathan puis lui ai montré un brouillon de mon code et une image paint de ce que je m'imaginais faire. Après avoir imaginés tout ce dont nous avions besoin, tel que le booléen qui permettra de savoir si oui ou non le joueur a réussi l'éénigme, lui donnant ainsi accès à l'éénigme suivante (à contrario, l'éénigme sera bloquée). Nous avons par la suite fait face à quelques problèmes : Comment tout mettre dans une seule liste puis pouvoir l'appeler ou encore, la boîte à énigmes ne voulait pas apparaître dans Godot.

(Cf : le paint en question)



5 Soutenance finale

Maintenant que la carte est presque finie, nous prévoyons que la prochaine soutenance nous permettra de terminer tous les détails et outils restants. 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 énigmes et leur placement sur la map
- Les connexions entre les différentes parties du projet
- Les éléments de décors

6 Conclusion

Ainsi, le projet a bien avancé pendant ces quelques semaines de développement, de nombreuses fonctionnalités ont été ajoutées et celles déjà présentes ont été améliorées. Au vu de nos progrès, nous pouvons dire que nous nous rapprochons de notre objectif qui était de présenter un jeu fonctionnel et dont nous sommes fier. Cela ne veut néanmoins pas dire qu'il est fini et comme expliqué dans ce rapport, il reste de nombreux éléments que nous souhaitons ajouter au jeu. Néanmoins, nous avons fait réalisé le plus difficile et nous allons nous efforcer de continuer sur notre lancé pour améliorer ce qui a été fait et ajouter bien sûr d'autres éléments dans le but de finir notre jeu, voire même d'aller plus loin et peut-être même le continuer après la soutenance finale.