**IDENTIFICATION**

|  |  |  |
| --- | --- | --- |
| **Nom du projet** | **Sprite** | |
| **Nom de l’équipe** | **Sprite** | |
| **Membres de l’équipe** | Francis Trépanier, 15 102 504 | Jacque Goupil, 16 103 958 |
| **Professeur encadrant** | **Mikaël Fortin** | |

**MANDAT**

Notre mission dans le cadre de ce projet était de concevoir un jeu vidéo d’exploration en vue « top-down » en s’inspirant des jeux *Legend of Zelda*. Ce jeu vidéo servira éventuellement à garnir un portfolio de projets personnels. Nous tenions donc à produire un résultat d’apparence professionnelle même si nous ne visions que la production d’un prototype avec une durée de jeu très limitée. Ce jeu vidéo a été développé avec le moteur Unity avec le langage C#, avec lesquels Francis Trépanier était déjà familier.

**Objectifs du projet**

* Créer un prototype de jeu vidéo fonctionnel.
* Découvrir ou approfondir nos connaissances quant au moteur de jeu Unity et au langage C#.
* Produire un résultat que l’on peut inclure dans un portfolio.

**LIVRABLES ATTENDUS ET RÉSULTATS ATTEINTS**

**État des livrables**

|  |  |  |
| --- | --- | --- |
| **LIVRABLES** | **ÉTAT** | **DATE** |
| Jalon 1 : Le joueur peut déplacer un personnage à l’écran et dispose d’une attaque à courte portée (coup d’épée) et une attaque à distance (projectile). | Terminé | 2019-07-03 |
| Jalon 1 : L’environnement de jeu est rendu grâce au principe de tuiles. | Terminé | 2019-05-30 |
| Jalon 2 : Au moins trois types d’ennemis sont présents et chacun a son propre comportement. Exemples : un ennemi qui fait des allers-retours, un qui fait une ronde, un qui s’approche du joueur lorsqu’il le voit, etc. | Terminé | 2019-07-24 |
| Jalon 2 : Le joueur et les ennemis disposent de points de vie et meurent lorsque ce nombre atteint zéro. | Terminé | 2019-07-23 |
| Jalon 3 : Le joueur peut récupérer des objets qui s’ajoutent à son inventaire et ont divers effets. Exemples : modifier ses statistiques de vitesse ou force d’attaque, changer son apparence, ajouter un nouveau type d’attaque, ajouter des munitions, etc. | Terminé | 2019-08-05 |
| Jalon 3 : L’environnement de jeu consiste en une suite de salles où il faut éliminer tous les ennemis pour pouvoir passer à la suivante. | Terminé | 2019-07-30 |
| Jalon 3 : Au moins une salle nécessite la résolution d’un puzzle pour progresser, par exemple un défi de Sokoban. | Terminé | 2019-08-08 |

**Notes sur les écarts (positif ou négatif)**

Quelques fonctionnalités ont débordé légèrement des dates prévues. Par exemple, la troisième arme du joueur et les effets sonores ont été implémentés au jalon 3 alors que nous avions prévu les ajouter plus tôt. Nous avons fait le choix de repousser ces tâches car elles étaient simples à effectuer, et car il était plus prioritaire d’avoir un environnement à explorer.

Nous avons réussi à implémenter toutes les fonctionnalités prévues dans nos trois jalons. Un léger écart peu cependant être noté sur l’objectif de créer une salle de résolution de puzzle. Le donjon principal du jeu présente quelques puzzles simples à résoudre comme l’interrupteur pour faire descendre les pieux pour accéder à la clef de la porte de la salle de combat, ou encore la salle secrète qui permet d’obtenir le chakram. Nous avons également entamé un prototype de salle semblable au *Color Dungeon* de *Legend of Zelda : Link’s Awakening DX*. Dans cette salle, des interrupteurs permettent d’ouvrir et fermer des portes de différentes couleurs. Cependant, cette salle n’est pas reliée au reste du jeu, et pourrait faire partie d’un autre donjon que nous implanterions dans le futur.

Une fonctionnalité bonus a été ajoutée à la toute fin, comme nous avions à la fois un peu de temps de disponible et voulions impressionner Mikaël. Nous avons pris la décision d’implémenter une salle avec un combat de boss contre une licorne volante. Le boss lance des projectiles et se téléporte lorsqu’il subit des dégâts. Le combat de boss est débloqué lorsque le joueur a vaincu les ennemis de la salle de combat du donjon. Vaincre le boss fait apparaître le coffre contenant les bombes dans la salle.

**Problèmes rencontrés**

Le plus grand facteur qui a ralenti le développement est le fait que nous ne sommes pas des artistes et que nous avions de la difficulté à produire les animations des personnages. Nous avons passé beaucoup de temps à apprendre à utiliser l’outil d’animation de Unity, et animer le personnage principal et son attaque à l’épée a demandé la majorité du temps du premier jalon. Les ressources graphiques que nous avons trouvées étaient également dans des formats que Unity avait de la difficulté à importer, si bien qu’il nous arrivait parfois de devoir manuellement découper un fichier de tuiles.

Un autre problème qui a affecté notre premier mois était davantage au niveau humain. Les deux membres de l’équipe n’avaient pas la même image en tête quant à l’objectif à atteindre. Jacque imaginait davantage un jeu similaire à *The Binding of Isaac* ou au premier *Legend of Zelda* sur NES tandis que Francis orientait le développement vers un *Legend of Zelda : Minish Cap*. Éventuellement, nous avons pris un moment pour regarder des images de nos sources d’inspirations mutuelles afin de s’enligner dans la même direction.

Unity a été pensé principalement pour faire de la 3D, bien qu’il soit également capable de faire du 2D. Certaines des fonctionnalités du moteur comme les algorithmes de prévision de trajectoires ne fonctionnent qu’en 3D, ce qui nous a forcé à implémenter notre propre solution au moyen d’une librairie externe. Aussi, les outils de création de niveaux sont plus pratiques à utiliser pour faire des jeux avec une caméra de face et non de vue aérienne, si bien que l’éditeur de niveaux était plus pénible à utiliser qu’un éditeur de niveaux 2D typique.

Finalement, Unity et Git ne font pas exactement très bon ménage avec l’énorme quantité de fichiers de métadonnées qui sont générés pour les assets. De plus, fusionner des fichiers de préfab ou encore des scènes dans Unity est une tâche quelque peu désagréable lorsqu’il y a des conflits à résoudre. Le format de type YAML utilisé ne s’y prête pas bien et nécessite souvent une fusion manuelle pour éviter des soucis de corruption.

**BILAN TECHNIQUE**

* **Méthodologie**
  + Notre méthodologie reposait principalement sur un horaire de présentation hebdomadaire à faire à Mikaël. Nous nous donnions des objectifs à accomplir pour la semaine suivante à la fin de chacune des rencontres et nous nous séparions les tâches à accomplir lors de la semaine entre nous.
* **Architecture** 
  + Notre architecture utilise principalement le principe d’orienté-objet et de « component » de Unity. Nous avons également utilisé le pattern « Delegate » pour la création du système d’automate de comportement qui utilise des classes réutilisables.
* **Logiciels et outils utilisés**
  + Unity : Moteur de jeu
  + Visual Studio : Éditeur de code
  + Rider : Éditeur de code
  + Gimp : Éditeur d’images
  + Pyxel Edit : Éditeur d’images
  + Game Character Hub : Créateur de sprite de personnage
  + GitLab : Dépôt de code et outil de gestion de projet
  + Git : Outil de versionnage
  + Discord : Logiciel de communication
  + Google Drive : Logiciel de stockage de fichiers

**LITTÉRATURE**

Unity Manual: <https://docs.unity3d.com/Manual/UnityManual.html>

Tutoriel sur les State Machine : <https://unity3d.com/fr/learn/tutorials/topics/navigation/finite-state-ai-delegate-pattern>

Tutoriel sur des mécaniques de base de Zelda : <https://www.youtube.com/watch?v=F5sMq8PrWuM>

Sources d’Assets : <https://opengameart.org/>

Plugin Unity 2D: <https://github.com/Unity-Technologies/2d-extras>

**CONTINUITÉ ET ÉVOLUTION**

* **Qu’est-ce qui reste à faire**

Il resterait principalement à ajouter davantage de contenu, car certaines des mécaniques de jeu que nous avons programmées sont présentement sous-exploitées par les niveaux de jeu en place. Il serait possible de créer de nouveaux ennemis rapidement au moyen du système d’AI, d’ajouter des niveaux et éventuellement d’inventer une histoire.

Il faudrait également faire un peu plus de travail de UI sur le jeu, le UI actuel étant très minimaliste. Il n’y a également aucuns menus qui existe pour consulter des informations sur les objets possédés ou encore le statut du joueur dans sa quête.

Finalement, si possible trouver un artiste 2D pour refaire nos animations et nos sprites afin d’améliorer la qualité du produit fini.

* **Si le projet était à refaire, qu’est-ce que vous feriez de différent?**

Nous aurions dû faire un peu moins de travail sur l’aspect graphique et peut être utiliser un style minimaliste afin de réduire le temps qu’on a passé sur les animations. Le manque de sprites pour l’animation a fait en sorte que l’on a passé beaucoup trop de temps à animer les attaques à l’épée qui ont nécessité trois versions avant d’être fonctionnelles.

* **Qu’est-ce qui pourrait être amélioré?**

Notre gestion du Git pourrait améliorée, nous n’avons pas du tout utilisé le système de branches et avons directement fait nos commits dans la branche Master ce qui est considéré comme une pratique non saine en entreprise.

Le code a besoin d’être réorganisé. Tous les fichiers C# sont à la racine du projet et il y a peu de réutilisation de code. Il faudrait définitivement prendre un moment pour planifier la séparation des objets et comportements de façon plus intelligente.