Panic At Tortuga

Rapport de soutenance n°1

Février 2021

LIFE INVADERS

Production

Table des matières

1 Introduction **2**

2 Conception **2**
2.1 Réseau . . . 3
2.1.1 Création d'une salle . . . 3
2.1.2 Instantiation et synchronisation . . . 3
2.2 Intelligence Artificielle et NPC . . . 4
2.3 Mécaniques de jeu . . . 5
2.3.1 Environement . . . 5
2.3.2 Déplacement du personnage . . . 5
2.3.3 Attribution des cibles . . . 6
2.3.4 Système de verrouilage . . . 7
2.4 Interface . . . 7
2.4.1 HUD . . . 7
2.5 Carte du jeu . . . 8
2.6 Système de progression . . . 9
2.6.1 Système de sauvegarde . . . 9
2.6.2 Contenu débloquable . . . 10
2.6.3 Système de customisation . . . 10

3 Avances et retards **10**
3.1 Les réussites . . . 10
3.2 Les retards . . . 11

4 Prévisions **11**
4.1 Map . . . 11
4.2 Interface . . . 11
4.3 Réseau . . . 11
4.4 I.A. . . . 11
4.5 Mécaniques de jeu . . . 11
4.6 Autres . . . 11

1 Introduction

Panic At Tortuga est un jeu orienté multijoueur. Mais quel genre de est-ce ? Il s'agit du jeu s'inspirant de mécaniques de jeux divers, les principaux étant Assassin's Creed (son mode multi) et d'autres jeux comme "Guess Who ?".

Vous et d'autres joueurs êtes lachés sur la petite île de Tortuga, une île où la population vit paisiblement. Mais parmi eux, déguisés, se cachent des pirates sanguinaires, et c'est à vous de les éliminer.

Le début de production de notre équipe de dévellopement, Lifeinvaders Production, composé de Paul, Harrys, Julien et Dov ; a pu commencer à développer le jeu depuis mi-décembre. Nous avons pu apprendre de nombreuses choses, se dépaser, sortir de notre zone de confort, mais surtout prendre du plaisir à créer notre jeu !

2 Conception

Le concept de notre jeu se base sur une idée simple : Vous avez une cible et êtes celle d'un autre joueur. Vous devez éliminer chaque cible attribué (en l'occurence d'autres joueurs) et tenter de survivre dans ce petit archipel.

Nous nous sommes répartis les taches rapidement les différentes tâches pour avoir les bases. Par exemple, le déplacement du personnage et de la caméra était primordial pour Harrys et Julien qui s'occupaient du multijoueur et nécéssitait d'être livré rapidement. Pendant que Renaud-Dov réalisait l'implémentation de ces scripts, Julien et Harrys ont commencé à apprendre comment fonctionnait Photon. Paul s'occupait alors de la map.

Pour éviter de se perdre dans l'afflux de tâches à réaliser, nous décidé d'utiliser des outils de travail en équipe tel que des kanban (avec Github Projects).

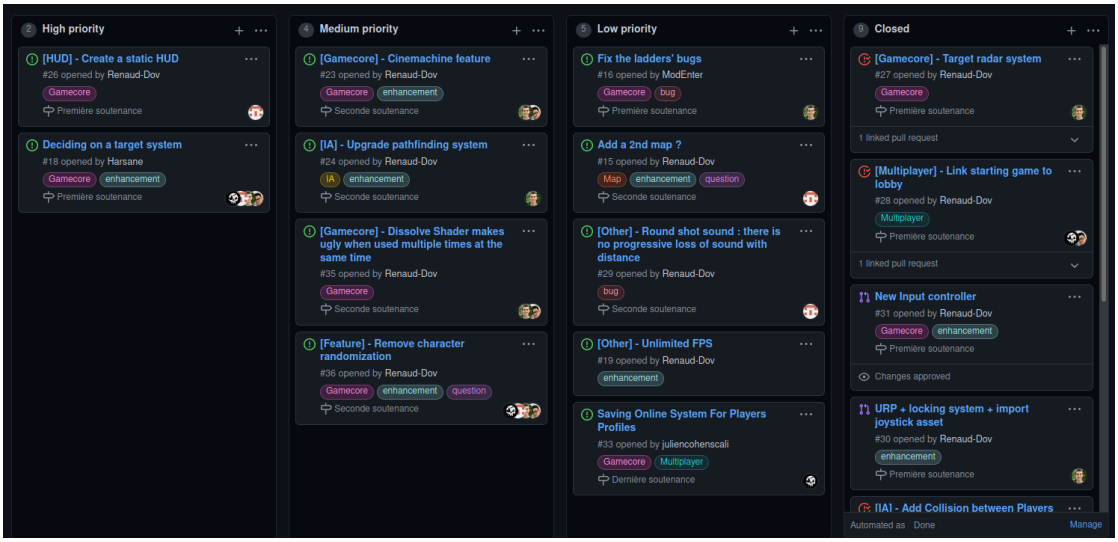

FIGURE 1 – Kanban pour la priorisation des tâches sur Github

2.1 Réseau

2.1.1 Création d'une salle

Photon se base sur un système de salle. Ces dernières permettent aux joueurs présent à l'intérieur de communiquer entre eux. Il est actuellement possible de créer ainsi que de rejoindre une salle. Ces deux processus se font automatiquement, sans effort du joueur, mais il est prévu de donner un plus grand contrôle sur ce système aux clients. En effet, il suffit d'appuyer sur un bouton pour tenter de rejoindre une salle. Si aucune salle est disponible, alors le client en crée une qui se rend disponible aux autres joueurs. Ce système a été implémenté par Julien et Harrys.

2.1.2 Instantiation et synchronisation

C'est Harrys qui s'est occupé de cette partie. Une fois que le joueur accède à une salle, celui-ci charge une scène, la même que tout les autres joueurs. Photon permet alors d'instancier son personnage, qui sera visible par tout les autres joueurs présent dans la salle. Cependant cela ne suffit pas, car il faut ensuite permettre la synchronisation des mouvements des joueurs. Pour celà, on utilise un composant appelé PhotonView. Ce dernier permet de synchroniser divers variables spécifiques à un objet comme sa position ou ses animations. De plus ce composant est nécessaire pour la communciation par "RPC" qui est essentiel à la réalisation du multijoueur. Certains éléments, tels que ceux permettant de contrôler le personnage grâce à l'utilisation du clavier ou bien d'une manette, doivent cependant être pris en compte lors de ce processus. Chaque client doit désactiver les composants problématiques des personnages qu'il ne "possède" pas. On utilise le système d'appartenance d'objet qui, tout comme la synchronisation de mouvement, se fait grâce au composant PhotonView. La scène actuel d'instantiation servant de lobby, divers ajouts "cosmétiques" ont été effectués comme l'apparition des pseudos au dessus de chaque joueur ou bien l'implémentation du choix de l'apparence. Un système de "timer", synchronisé pour tout les joueurs d'une salle est actuellement en cours de développement. Par la suite, les différentes actions pouvant être réalisés par les joueurs devront également être synchronisés. Cepdendant le travail effectué jusqu'à présent facilitera grandement cette tâche.

FIGURE 2 – Photo de groupe en multijoueur

2.2 Intelligence Artificielle et NPC

Il s'agit de Renaud-Dov qui a réalisé l'implémentation de l'IA. Pour rendre notre jeu plus vivant et permettre aux joueurs de se méler dans la foule, il nous fallait créer des IA se déplacant dans la ville. Pour cela, nous avons décider d'utiliser des NavMesh pour créer des zones où les NPC peuvent se balader d'un point A vers un point B. Pour ne pas rendre leur comportement linéaire et sans saveur, ils se déplacent de manière aléatoire vers un point quelconque défini dans une liste de coordonées. Aucun NPC n'aura donc le même trajet.

L'outil de navigation est assez poussé, et permet de définir les dimensions des personnages, ainsi que la hauteur de laquelle ils peuvent sauter et les pentes qu'ils peuvent emprunter. Ici, la pente maximale est de 33.4°, mais si ce chiffre avait été plus élevé, les pentes sur la capture d'écran seraient devenues bleues, et les personnages auraient pu monter sans utiliser les escaliers (ce qui n'est évidemment pas le but).

Pour rendre ces IA plus humaines, les joueurs peuvent les bousculer et les étourdir en courant vers eux. Ils reprenent leur trajet au bout de quelques secondes. Il faudra ainsi, dans les prochaines versions, améliorer la navigation afin que le chemin emprunté ne soit pas toujours le plus court.

Plusieurs problèmes se posent actuellement avec l'utilisation des NavMesh :

— L'IA calcule le chemin le plus court, et n'essaie pas de se déplacer sur le côté si d'autres IA marchent dans le sens contraire. Par exemple, si l'ont met deux escaliers cote à côte, toutes les IA vont prendre l'escalier le plus proche et donc se bloquer.

FIGURE 3 – Problème rencontré lorsque de nombreuses IA vont au même endroit

2.3 Mécaniques de jeu

2.3.1 Environement

Nous nous sommes beaucoup inspirés du mode multijoueur des premiers Assassin's Creed. Le système de grimpe s'étant montré trop complexe à mettre en place, nous avons réutilisé quelques idées de gameplay pour l'environement, comme les échelles et les portes.

FIGURE 4 – Joueur grimpant sur l'échelle

Les échelles permettent de créer un peu de verticalité dans nos niveaux. Elles ne peuvent être utilisées que par les joueurs, mais ont leurs avantages comme leurs inconvénients : ainsi, prendre de la hauteur permet d'emprunter des raccourcis, mais retire la discrétion (car personne de civilisé devrait être sur les toits !)

Les portes permettent de barrer le passage pour échapper à son agresseur. Elles se ferment quand le joueur passe dessus en courant et se rouvrent au bout de quelques secondes.

Ces interractions environnementales ont été réalisé par Renaud-Dov.

2.3.2 Déplacement du personnage

C'est Renaud-Dov qui a réalisé les contrôles de déplacement du personnage :

Tout jeu est très rapidement limité par la capacité de déplacement du personnage et sa vitesse. Nous avons donc réalisé un script de déplacement qui permet au joueur de marcher, courir, sauter et grimper aux échelles.

FIGURE 5 – Porte fermée après le passage d'un joueur

Pour le côté technique, nous avons dans un premier temps utilisé l'Input System de base proposé par Unity. Mais plusieurs problèmes se sont posés :

— Les configurations des keymaps sont assez limitées
— Paramétrer des périphériques autre que le clavier/souris est compliqué. Il faudrait avoir des scripts différents pour chaque périphérique, et donc des prefabs différents

C'est pour cette raison que nous avons migré vers le New Input System. Celui ci est ergonomique, multiplateformes et permet l'utilisation de différents périphériques, comme la manette ou le clavier par exemple. Certains scripts ont dû être modifiés pour devenir compatibles avec ce système.

Il est donc actuellement possible de jouer au clavier/souris ou à la manette (pas les deux à fois).

2.3.3 Attribution des cibles

Harrys a réalisé le script permettant l'attribution des cibles, qui utilise des méthodes RPC (communication inter-clients par l'intermédiaire d'un serveur). Le "MasterClient", un client qui est désigné dans chaque salle pour faire office de maître de jeu, est celui qui séléctionne la cible de chaque joueur. Il communique ensuite à chacun des clients présents dans la salle sa cible désignée. Cette attribution se fait de façon aléatoire, mais selon certaines règles :

-Un joueur ne peut (évidemment) pas être sa propre cible -Deux joueurs ne peuvent pas être la cible l'un de l'autre (Cela permet une plus grande interaction entre les joueurs et pas simplement des scénarios en "1 contre 1"). -Plusieurs joueurs ne peuvent pas avoir la même cible.

Cependant même si ce script est fonctionnel, il n'a pas encore été implémenté. Le

système d'élimination de la cible fera l'objet d'un travail important pour la prochaine soutenance.

2.3.4 Système de verrouilage

Le système qui attribue des cibles à chaque joueur n'est pas entièrement implémenté, mais le joueur peut déjà tuer une cible, qu'elle soit la bonne ou non. Pour cela, il passe en mode verrouilage, et les personnages pointés par le viseur surbrillent. Pour les sélectionner, un coup de molette suffit, et le contour devient alors jaune, pour indiquer que la cible est verouillée. Il suffit alors d'être à moins d'un mètre pour l'éliminer.

(a) Personnage au centre de l'écran en surbrillance

(b) Personnage sélectionné et verrouilé

(c) Disparition des corps après mort

FIGURE 6 – Système de verrouilage

2.4 Interface

2.4.1 HUD

Le HUD est une interface présente tout le temps à l'écran, et permettant d'afficher les informations relatives au joueur et à l'avancement de la partie directement

en jeu. Ce dernier a donc pour obligation d'être complet (au moins : informations sur la cible, points, temps de jeu restant) et assez discret, pour ne pas avoir l'impression de jouer à travers un hublot. C'est pourquoi nous avons décidé de mettre les éléments dans les coins le plus possible, afin de ne pas déranger les joueurs. Les informations fournies par le HUD n'étant pas utiles au stade actuel du jeu (encore assez expérimental), j'ai décidé de me focaliser sur la carte et sur la manipulation générale des scripts Unity avant de le commencer, et ce dernier n'est donc pas encore implémenté dans les builds de la première soutenance.

2.5 Carte du jeu

La carte étant un élément crucial du jeu, au même titre que les mécaniques, il m'a semblé important de faire des schémas et de demander l'avis des membres du groupe, ainsi que d'autres personnes, afin d'être sûr de mes plans avant de commencer. Ainsi, la carte finalement retenue devait avoir suffisamment de relief pour rendre les échelles intéressantes, et suffisamment spacieuse pour pouvoir la remplir avec au moins 150 personnages non-joueurs (afin de pimenter le jeu). Dans un premier temps, une forme ronde avait été retenue pour la carte, mais a ensuite évolué pour une forme en L, cette dernière rendant l'environnement plus naturel et augmentant les chances des personnages de se croiser.

FIGURE 7 – Vue aérienne de la carte

Ensuite, afin d'ajouter du relief à la carte, une colline a été créée. Cette dernière s'étend sur environ un quart de la carte, et possède quatre niveaux afin d'en permettre l'accès par de petits escaliers successifs. Une colline étant un terrain irrégulier, il a été décidé que les bâtiments placés sur cette dernière ne seraient pas

parallèles, mais répartis afin de créer un imbroglio de maisons rappelant le style méditerranéen dont les îles comme celle-ci sont inspirées.

FIGURE 8 – Colline de la carte, organisée par étages

Enfin, l'architecture de la ville elle-même devant elle aussi avoir une influence ibérique, les maisons ont été dessinées basses et organisées autour de places et marchés animés. Les tonnelles et les nombreuses lanternes rendent l'environnement plus chaleureux, et les arcades, dotées de portes qui se ferment lorsqu'un joueur les passe en courant, ajoutent une mécanique de fuite au jeu. L'organisation du village se fait autour de la place de l'église, qui fait office de place du marché.

2.6 Système de progression

Le système de progression, qui reste pour l'instant une tâche secondaire, a tout de même vu un avancement satisfaisant. En effet même s'il n'y a pas encore de mécanique de jeu permettant à ce système de prendre sens, l'objectif principal pour cette première soutenance a été réalisé : le système de sauvegarde.

2.6.1 Système de sauvegarde

Un système de sauvegarde complet a été implémenté. Une classe "PlayerDatabase" a été créé et permet de stocker l'ensemble des variables que nous souhaitons sauvegarder. Ce système est utilisé non seulement pour sauvegarder la progression du joueur, mais il se trouve qu'il a également une grande utilité pour sauvegarder d'autres paramètres, comme par exemple la persistence du changement des contrôles effectués par le joueur (cf. menu d'accueil). Il n'existe qu'une seule instance de cette classe pour tout le jeu (singleton). Cette instance est sérialisée(processus permettant de convertir l'information dans un autre format, ici dans le but de la stocker, permettant ainsi sa persistance) puis sauvegardée dans un fichier. Au lancement du jeu, ce fichier est désérialisé(processus inverse de la sérialisation) est l'instance récupérée

remplace donc l'instance unique présente en jeu. Il est important de faire cette démarche le plus tôt possible, afin que le chargelent de l'instance soit faite avant toute tentative d'accés à cette dernière. Il faut également faire attention à sauvegarder régulièrement, notamment lors de la modification de l'instance.

2.6.2 Contenu débloquable

Même si le système de progression n'a pas encore été créé, une réflexion sur la "récompense" de ce système de progression a eu lieu. Il serait donc intéressant d'explorer la possibilité de débloquer des effets "cosmétique" qui serait visible dans le lobby (cf. multijoueur). Un système de customisation de personnage a été créé, et nous avons pour objectif de permettre au joueur de débloquer ces éléments de customisation.

2.6.3 Système de customisation

Même si cette mécanique est encore basique, le joueur a la possibilité de choisir l'apparrence de son personnage dans le menu d'accueil(cf. menu d'accueil). Cette dernière sera vu des autres joueurs du lobby. Le fonctionnement est le suivant : chaque caractéristique est associée à une variabl. Ces variables composent un code unique associé à une combinaison spécifique qui sera alors affiché. De plus ces variables sont sauvegardés et le choix est donc persistant au redémarrage. D'ici la fin du projet, l'objectif est de bloquer l'accés à certains éléments de customisation, et de permettre au joueur de les débloquer grâce à sa progression.

3 Avances et retards

Tâche / Partie	Prévu	Réalisé
Mouvement	50%	80%
Interface/HUD	40%	40%
Cartes	40%	40%
Réseau	20%	50%
IA	30%	40%
Mécaniques de jeu	50%	40%
Progression	20%	30%

TABLE 1 – Tableau des avances et retards dans les différentes parties

3.1 Les réussites

La création du système de mouvement s'est avérée moins compliquée que nous l'avions prévu. Il manque un peu de finesse, et une physique de saut plus réaliste ne serait pas de trop non plus, mais il reste assez complet. L'utilisation du pathfinding pour les mouvements des IA nous a elle aussi étonné par sa simplicité. Un lobby

fonctionnel pour le multijoueur a aussi été réalisé ; cela nous permet non seulement de continuer sereinement la suite du projet, notamment concernant le lancement d'une partie multijoueur, mais également d'acquérir des connaisances sur Unity et sur Photon qui nous permettront d'arriver au bout de ce projet.

3.2 Les retards

Certaines mécaniques de jeu sont encore en développement et ne sont pas finalisées (réapparition, système de points...) D'autres, comme l'attribution des cibles, sont finalisées ou presque mais ne peuvent pas encore être implémentées.

4 Prévisions

4.1 Map

4.2 Interface

4.3 Réseau

4.4 I.A.

Pour la seconde période, notre objectif serait de pouvoir avoir une IA plus autonome et intelligente, qui n'emprunte plus forcément le chemin le plus court. Egalement, les joueurs peuvent bousculer les NPC qui s'arrêtent alors de marcher, mais ils ne peuvent jamais les pousser.

4.5 Mécaniques de jeu

Nous prévoyons d'intégrer pour la deuxième soutenance des pouvoirs comme un écran de fumée ou encore des couteaux, pour rendre le jeu plus intéréssant et dynamique.

Nous comptons également implémenter le script de choix de la cible. Même si celui-ci a été réalisé il n'est pas encore utilisé en jeu. Il faudra principalement donner un sens à l'attribution des cibles, sous la forme de points octroyés au joueur. Celà sera la base de laquel chaque partie sera articulée.

4.6 Autres

Pour tester le multijoueur avec des invités, nous avons réalisé un début de launcher qui télécharge les dernières mises à jour. Nous aimerions continuer à le développer pour avoir le launcher qui se mette à jour automatiquement sans à avoir à le réinstaller.

Table des figures

1 Kanban pour la priorisation des tâches sur Github 2
2 Photo de groupe en multijoueur 3
3 Problème rencontré lorsque de nombreuses IA vont au même endroit . 4
4 Joueur grimpant sur l'échelle 5
5 Porte fermée après le passage d'un joueur 6
6 Système de verrouilage 7
7 Vue aérienne de la carte 8
8 Colline de la carte, organisée par étages 9

Liste des tableaux

1 Tableau des avances et retards dans les différentes parties 10