

Aegina : rapport de projet

Florian AMSALLEM (amsall_f) , Théo ISSARNI (issarn_t),
Julien MOUNIER (mounie_a), Romain MOUNIER (mounie_r)

AIM²

14 juin 2016



Remerciements

Nous tenons à faire des remerciements à Pierre-Alexis Mandégué pour son aide dans la création des musiques d'Aegina et à pour son aide dans la création de modèles 3D des créatures d'Aegina.

Introduction

Ce document est le rapport de projet d'Aegina, projet du second semestre de la première année de la prépa intégrée d'EPITA. Ce projet est réalisé par le groupe d'étudiants AIM² constitué de Florian Amsallem, Théo Issarni, Julien Mounier et Romain Mounier.

Le but de ce projet était de travailler en groupe et de créer un jeu de survie dans un environnement hostile dans lequel le joueur incarnerait une personne se retrouvant sur une île volante et ne sachant rien de ce qui a bien pu lui arriver. Cette personne aurait alors pour but de survivre et de tenter de rentrer chez lui.

Ce jeu a été développé sur unity3D en C# en utilisant visual studio. Nous avons également utilisé Blender pour faire les modèles 3D ainsi que GIMP pour faire les textures 2D.

Dans ce rapport nous allons commencer par présenté l'univers d'Aegina car nous avons beaucoup développé l'histoire de ce monde et des différents éléments le composant. Puis nous allons de notre environnement visuel et sonore car presque tous les éléments composant cet environnement ont été fait spécialement pour le jeu Aegina par les membres de notre groupe. Ensuite nous allons parler de tous les systèmes développés dans notre jeu ce qui englobe la plus grande partie de notre travail. Enfin nous allons parler des différents moyens de communication que nous avons mis en place afin de partager notre création et de trouver des joueurs intéressés pour tester notre jeu.

1 Univers du jeu

1.1 Le monde d'Aegina

Le monde d'Aegina est constitué d'une quasi infinité d'îles volantes se trouvant au-dessus d'un vide béant appelé les *abysses*. Ce monde, contrairement au nôtre, est instable. Il ne partage pas nos lois de l'équilibre et de nombreux phénomènes étranges s'y produisent sans que l'on puisse les expliquer. La présence des *abysses* n'étant qu'un de ces phénomènes.

Bien que ne vivant pas dans ce monde et n'ayant aucune information sur celui-ci, les êtres humains on trouvé un moyen de « téléporter » leurs déchets dans un néant qui n'est autre qu'Aegina. Lorsque les déchets arrivent directement dans les *abysses*, ils n'ont aucune influence sur la stabilité d'Aegina. S'ils tombent sur une île, ils peuvent représenter un apport en ressource pour Aegina. L'aspect néfaste pour Aegina provient d'une sorte très particulière, presque barbare, de déchets.

Certaines autorités humaines utilisent la « téléportation » pour se débarrasser d'autres humains. Malheureusement une fois dans le monde d'Aegina ceux-ci tentent désespérément de survivre. Ils finissent en général par arriver à activer les cristaux dont les pouvoirs quasi divins leur permettent d'échapper à la faim et à la mort. Certains ont alors voyagé d'île en île, créant quelques infrastructures telles que des ponts, laissant leur trace dans ce monde, mais l'amenant petit à petit à sa fin sans le savoir.

Une des grande marque de l'instabilité d'Aegina est son climat. Le climat dans l'univers d'Aegina est très étrange. Il peut changer brusquement d'une zone à une autre mais est stable dans chacune des zones. Cette stabilité a permis à de nombreuses espèces de s'installer et de survivre dans ce monde sans craindre un brusque changement de climat. Les différents climats d'Aegina, au nombre de quatre, sont souvent appelés les quatre saisons d'Aegina pour leur ressemblance aux saisons du climat tempéré de notre monde.

1.1.1 Primtemps : forêt

Le printemps d'Aegina est un climat dans lequel les températures sont stables entre 6 ° C et 20 ° C. De plus les pluies ne se font pas rares ce qui permet aux îles du printemps d'Aegina d'accueillir une faune et une flore abondantes et variées.

Ces îles aux apparences tranquilles semblent les zones les plus agréables pour les voyageurs d'Aegina qui cherchent un endroit pour se reposer. Cependant détrompez-vous, les créatures d'Aegina sont très agressives et le printemps d'Aegina est la saison de la « mort » pour les aventuriers qui s'y aventurent.

Les cristaux se trouvant dans le printemps d'Aegina ont très souvent la capacité de rendre

la terre plus fertile et d'augmenter le nombre d'animaux et de végétaux dans l'île transformant ainsi les forêts de ces îles en jungles dangereuses pour quiconque s'y aventurerait. Le printemps d'Aegina serait une véritable mine d'or pour des entrepreneurs cherchant à récolter du bois ou cultiver des plantes mais malheureusement ce type de personne survit rarement longtemps dans Aegina.

1.1.2 Automne : forêt automnale

L'automne d'Aegina est un climat où les températures varient entre 2 °C et 25 °C. Un fort ensoleillement et d'autres raisons obscures ont permis à une végétation assez particulière de se développer dans ce climat. Cette végétation possède une couleur rouge-orangé donnant l'impression d'un automne constant sur l'île. De plus certaines espèces végétales comme les champignons et les citrouilles sont endémiques de l'automne d'Aegina.

Malgré la présence de créatures plus faibles dans l'automne d'Aegina que dans le printemps d'Aegina, ces zones n'en restent pas moins dangereuses car l'automne d'Aegina échange la densité de créatures contre une plus grande diversité de celles-ci, rendant ainsi la chasse plus difficile même pour des chasseurs aguerris. Ces zones n'en restent pas moins les zones favorites des aventuriers du monde d'Aegina car elles présentent le meilleur rapport « confort de vie »/« danger des créatures ».

Les cristaux se trouvant dans l'automne d'Aegina peuvent avoir n'importe quelle capacité rendant ces zones encore plus variées et uniques qu'elles ne le sont déjà. De plus tous les ans, les aventuriers d'Aegina vont sur ces îles pour y récolter et sculpter des citrouilles en pleurant et en se souvenant de leur monde. Cette pratique vaut à l'automne d'Aegina le nom de « la saison de la mélancolie ».

1.1.3 Hiver : neige

L'hiver d'Aegina est un climat extrêmement froid qui atteint des températures avoisinant les -20 °C et qui ne dépassent jamais les 0 °C. Ce Climat rend la faune et la flore sur ces îles peu fréquentes. Etrangement, les îles influencées par l'hiver possèdent une abondance de minéraux. Cela serait dû au fait que, en raison de la faible présence d'êtres vivants pouvant absorber les énergies étranges d'Aegina, ces énergies sont absorbées par les roches et transformées en minéraux.

Malheureusement la faible présence de créatures dans l'hiver d'Aegina n'en fait pas un lieu de rêve pour les aventuriers car le climat y est tout de même redoutable et la nourriture rare. Seules les rares aventuriers capables de forger y viennent pour récupérer des minéraux.

Les cristaux se trouvant dans l'hiver d'Aegina ont de plus fortes chances de posséder

des pouvoirs divins capables de briser l'espace et le temps. Ceux-ci peuvent permettent de se déplacer vers d'autres cristaux se trouvant aussi dans l'hiver d'Aegina. Ce qui est très utile pour les aventurier possédant une base près de ces zones et veulant explorer le monde d'Aegina.

1.1.4 Eté : désert

L'été d'Aegina est un climat extrêmement chaud et aride. La température y est toujours au dessus de 30 ° C et atteint régulièrement les 50 ° C. La faune et la flore de l'île sont alors très particulières, elles sont toutes les deux globalement constituées de cactus. Etrangement, c'est aussi l'un des seuls endroits où l'on peut trouver de l'eau car les cactus d'Aegina retiennent une énorme quantité d'eau en leur intérieur.

Le climat de l'été d'Aegina n'est pas vraiment le meilleur pour les aventurier mais la présence de cactus, et surtout de l'eau dont ils regorgent, peut transformer cette zone en un lieu très agréable pour se reposer. Malheureusement quelques créatures rodent dans l'été d'Aegina et celle-ci peuvent profiter d'un moment d'égarement pour se repaître d'aventuriers encore chauds.

Les cristaux se trouvant dans l'été d'Aegina ont la particularité d'être le plus souvent des cristaux de guerre. Ceux-ci renforcent les aventuriers d'Aegina en les rendant capables d'abattre un sanglier d'un seul coup d'épée bien affutée. Ces aventuriers se retrouvent alors également avec un corps résistant comme l'acier, ce qui leur permet de combattre une armée de monstres sans périr. Malheureusement, l'effet ressenti est bien souvent supérieur à l'effet réel amenant alors les aventurier, surestimant leur force et leur résistance, à une mort prématurée face à une horde de monstres.

1.2 Les minerais

Dans le monde d'Aegina on peut trouver de nombreux minerais qui permettront d'aider le joueur dans son aventure. Certains de ces minerais existent dans notre monde, mais Aegina possède aussi des minerais uniques. Dans cette partie nous allons vous présenter les minerais exclusifs au monde d'Aegina notamment du cristal.

1.2.1 Le cristal

L'instabilité de ce monde est principalement due à la présence de cristaux sur certaines îles. Ceux-ci sont capables de déchirer les trames de l'espace et du temps permettant de réaliser des miracles comme le retour dans le temps ou la téléportation. La puissance des cristaux est amplifiée lorsqu'ils sont activés.

Les cristaux dans Aegina sont un élément très important car ils possèdent d'énormes pouvoirs et sont responsables de l'instabilité du monde. Ils sont en apparence indestructibles, mais il est toutefois possible de les rendre inactifs. Pour s'activer ils ont besoins d'absorber divers minerais ce qui arrive très rarement naturellement, mais peut être facilement réalisé par des êtres conscients.

Le cristal semble réagir avec tous les minerais spécifiques du monde d'Aegina ainsi qu'avec de nombreuses créatures qui, en présence d'un cristal actif, perdent toute raison et tentent de le désactiver.

Peu d'informations supplémentaires sont disponibles sur les cristaux. Leur composition est totalement inconnue et personne ne sait d'où ils viennent. Ils ne semblent pas exister depuis la création du monde car leur forme géométrique et leurs pouvoirs donne l'impression qu'ils ont été créé par des êtres conscients possédant une technologie très avancée. Mais il ne s'agit que de suppositions qui n'ont que peu d'importance pour notre histoire.

1.2.2 Mithril

Le mithril, dans le monde d'Aegina, est un mineraï vert clair un peu plus solide que le fer. Assez rare à la surface des îles, on en trouve toujours en petite quantité en sous-sol puisqu'il est l'un des deux minerais nécessaires à la « survie » d'une île. Le mithril a la particularité de modifier les phénomènes étranges d'Aegina parfois en les amplifiant, d'autres fois en les stabilisant. Il est impossible de savoir comment celui-ci va réagir avant de l'utiliser.

A l'état naturel, il catalyse les instabilités et les relâche sous forme de vagues d'énergie rendant le mineraï chaud au toucher. Une fois épuré, le mithril continue de catalyser les instabilités, mais peut alors relâcher son énergie sous de nombreuses autres formes que la chaleur, comme le ferait un cristal, mais à bien plus faible échelle.

D'ailleurs l'énergie catalysée et stockée par le mithril semble être la même que celle se trouvant dans les cristaux car les deux matériaux peuvent échanger cette énergie. Ainsi quelques chercheurs hérétiques qui se sont trouvés sur Aegina ont supposé que les cristaux étaient formés d'un alliage composé en majorité de mithril. Ces chercheurs ont trouvé la mort quelques jours plus tard, tués par des sangliers après avoir transféré toute l'énergie de leur cristal dans du mithril.

1.2.3 Floatium

Le floatium est un mineraï quasi transparent que l'on peut trouver dans Aegina. Les humains qui arrivent dans Aegina le confondent souvent avec du quartz lorsqu'ils en voient dans la roche. Mais celui-ci est bien plus transparent et s'apparenterait plus à du verre ou de la glace. La vrai particularité du floatium se remarque une fois le mineraï nettoyé de toute impureté. La première constatation est que celui-ci est bien plus solide que ce à quoi on peut penser, sa résistance égale celle de l'acier, mais surtout il est impossible de le faire tomber par terre. En effet, une fois celui-ci lâché, il flotte.

Ce mineraï est encore un mystère d'Aegina car personne ne sait pourquoi il flotte. Il n'est pas plus léger que l'air puisqu'il reste à une distance fixe du sol, mais il ne semble pas s'agir non plus de force magnétique. De toute façon peu d'expériences ont eu lieu sur le floatium car, en Aegina, en plus d'être un mineraï très rare, les personnes capables de faire des expérimentations scientifiques le sont encore plus.

La seule information sûre est que le floatium est, avec le mithril, l'un des deux composants permettant aux îles de voler. Cependant, les cristaux et le floatium peuvent réagir entre eux, le cristal semble pouvoir absorber les capacités du floatium tandis que le floatium est responsable de la rotation et de la lévitation du cristal, mais tout cela reste de la théorie.

Néanmoins, il est possible de conjecturer que si un cristal a une trop grande influence sur une île, il pourrait provoquer sa chute. Mais les seules personnes connaissant la vérité sont sûrement mortes avec leurs îles.

1.2.4 Sunkium

Le sunkium est un mineraï d'une couleur mauve très sombre. C'est un mineraï légendaire qui possède de nombreuses qualités car il est le mineraï le plus rare, le plus lourd, le plus sombre, le plus solide, le plus cool, le plus classe, le plus *dark* et le plus unique d'Aegina. Pourtant, malgré sa solidité, il n'est pas si difficile à extraire des roches car il semble repousser les autres minerais lors de sa formation, ce qui permet de le trouver dans des alvéoles formées par la roche. Cependant une fois extrait son poids reste tout de même un problème pour le

transporter.

Mais ce qui en fait le minerai le plus cool, c'est qu'il peut être utilisé très simplement pour la création d'épées ou d'armures. En effet, il a la même température de fusion que le fer et une fois mélangé à du mithril il chauffe et refroidit à une vitesse exceptionnelle. De plus la couleur sombre des armures formées avec le sunkium apporte un côté cool et *dark* à toute personne les portant. Mais malheureusement cette impression ne dure que quelques secondes car rares sont ceux capables de se mouvoir dans une telle armure. La plupart se retrouvent coincés dans celle-ci, incapables de bouger ou de l'enlever, amenant à la mort la plus stupide d'Aegina.

Le sunkium, comme le mithril et le floatium, réagit avec les cristaux mais sa réaction est peut être la plus impressionnante des trois, après tout c'est le minerai le plus cool. Le sunkium change de volume et de masse sous l'influence du cristal mais pas comme tout le monde s'y attendrait car plus son volume diminue, plus il devient lourd. Il se met alors à absorber de la lumière, assombrissant les alentours. Le cristal quant à lui peut absorber le sunkium mais personne ne sait réellement ce que ça lui apporte.

Certaines théories stipulent que les pouvoirs du cristal viendraient de minuscules particules de sunkium qui déformerait l'espace temps. Mais cette théorie s'appuyant sur d'autres théories qui n'ont pas été prouvées, sa véracité est plus que discutable. Sans la présence des cristaux, le sunkium serait le minerai le plus mystérieux d'Aegina.

1.3 La Faune

Malgré toutes ces instabilités un écosystème s'est tout de même développé. Des animaux tels que des lapins, pingouin et sanglier peuplent les îles d'Aegina. Mais d'autres espèces plus agressives et plus exotiques sont apparues suite à l'instabilité du monde. Parmi elles nous trouvons les *slimes*. Les créatures de l'univers d'Aegina sont toutes des êtres vivants ayant muté à cause de l'influence des cristaux. Cela donne à Aegina un faune unique que ses aventureurs n'apprécient pas forcément. Nous allons présenté la faune d'Aegina en commençant par notre personnage principal.

1.3.1 Ille

Ille est le personnage que le joueur incarne dans Aegina. De père et de mère inconnus, Ille survit depuis son plus jeune âge seul dans une forêt. Bien que n'ayant pas d'amis, sauf si l'on peut considérer un écureuil comme un ami, Ille est très chaleureux et amical. Cependant, Ille n'est pas un enfant sauvage comme son histoire le laisse penser, il est légèrement civilisé et a lui-même choisi de vivre en ermite à cause de sa particularité. En effet, depuis sa petite enfance, Ille possède une force surhumaine (et un certain talent pour la construction). Il a lui-même construit sa maison à l'âge de neuf ans et est capable d'abattre des arbres comme un bûcheron chevronné depuis qu'il sait manier une hache. Cependant, cette force effraie facilement les enfants de son âge et même les adultes, mais malgré tout Ille vit de façon paisible. Les repas de Ille ne sont pas très extravagants, un jour cela peut être un sanglier et le lendemain un simple champignon. Une seule chose concernant sa nourriture lui importe, cette dernière doit être préparée car Ille tient toujours au fond de lui à se rapprocher des siens et l'action de préparer la nourriture lui donne l'impression d'être vraiment humain.

Son seul défaut, si l'on peut considérer ceci comme un défaut, est d'être simplet. C'est d'une certaine façon une chance car c'est ce qui à permis à l'aventure d'Aegina de commencer.

Lors d'un hiver très rude, Ille n'eut pas d'autre choix que de voler de la nourriture dans un village avoisinant. Malheureusement, la discréction n'étant pas son fort, il fut surpris en plein méfait et un garde tenta de l'arrêter. Ille essaya de passer le garde, mais il ne contrôla pas sa force et, en le bousculant, il le tua. Ille n'avait pas réalisé que le garde était extrêmement affaibli par le froid et la faim. Il fut alors poursuivi pour meurtre et rapidement arrêté car, malgré sa force, Ille restait humain. Il fut condamné à mort et envoyé dans le néant qui n'était autre qu'Aegina. C'est à son réveil que le jeu commence.

Cependant, Ille n'est pas forcément seul en Aegina. Il pourra rencontrer d'autres personnes car la mort par envoi dans le néant est une pratique assez courante. Retrouver des amis dans ce monde dangereux peut être une bonne chose car dans Aegina le nombre de vos

amis fait en partie votre force.

1.3.2 Les créatures agressives

Certaines créatures d’Aegina sont hostile au humain et peuvent décider de les attaqués. C'est créatures sont appelé les créatures agressives. Ces créatures peuvent s'organiser en meute et avoir alors un chef de meute possédant des capacités supérieurs aux autres membres de son espèce.

Sanglier Le sanglier d’Aegina est une créature présente dans la plupart des biomes. Le seul climat qu'il évite est le froid de l'hiver d’Aegina. Il se porte très bien dans des climats chauds comme l'été d’Aegina où il n'hésite pas à détruire des cactus pour boire.

Contrairement à un sanglier normal, il a plus l'apparence d'un phacochère et se montre très agressif envers les aventuriers d’Aegina. Son régime alimentaire est constitué de tout ce qu'il peut trouver allant du cactus à la chair humaine et en passant même par les minerais de mithril qu'il affectionne pour des raisons que l'on ignore.

Une particularité que les aventuriers d’Aegina ont trouvée à ces sangliers est qu'ils ont une peur des abysses telle qu'ils n'osent pas s'approcher du bord des îles et n'empruntent les ponts que lorsque leurs ennemis ne se trouvent pas en leur présence de peur que ceux-ci les poussent dans le vide.

Ces sangliers, bien que très dangereux, sont très appréciés des aventuriers d’Aegina car leur chair est un met succulent et peut être facilement cuisinée en de nombreux plats. Ces sangliers sont les meilleurs mets de tout l'univers d’Aegina loin devant les autres créatures qui, parfois, ne peuvent même pas être mangées.

Pampi Le pampi est un mystère pour tous les biologistes ayant mis les pieds dans le monde d’Aegina. Cette créature vivant dans le désert ressemble à un cactus humanoïde. Personne ne sait si les pampis sont des cactus ayant acquis la capacité de se mouvoir, des créatures prenant l'apparence de cactus pour se camoufler ou des humains ayant été transformés en cactus.

Ces trois hypothèses restent plausibles car le régime alimentaire du pampi reste inconnu et pourrait bien se limiter à un peu d'eau fraiche et de soleil comme les autres plantes. Cependant, les pampis possèdent une bouche et, même si celle-ci ne semble pas servir à manger, elle marque bien l'appartenance du pampi au règne animal.

De plus le comportement du pampi semble confirmer l'hypothèse qu'il a pris l'apparence d'un cactus pour se camoufler car il est d'une nature très peureuse et ne montre ses cotés agressifs qu'en deux occasions : quand il est acculé et n'a plus de possibilité pour s'enfuir et

quand on l'appelle pampa. En effet, le pampi est une espèce orgueilleuse qui n'aime pas que l'on déforme son nom.

Enfin la dernière hypothèse vient en partie de ce caractère orgueilleux typiquement humain mais surtout des cris que pousse le pampi, ceux-ci rappelant très fortement des cris humains.

Les aventuriers d'Aegina qui chassent le pampi ont deux règles d'or : ne pas acculer un pampi et ne pas prononcer le mot pampa. Un pampi agressif est l'une des créatures les plus dangereuses d'Aegina, il est capable de propulser ses épines avec la puissance d'une mitraillette. Les squelettes que l'on peut trouver dans le désert sont les vestiges d'une espèce détruite par les pampi. Malheureusement pour elle, cette espèce poussait un cri ressemblant à s'y méprendre au mot pampa.

Slime

1.3.3 Les créatures pacifiques

D'autres créatures d'Aegina préfèrent adopter un comportement moins agressif vis à vis des humains et vont donc fuirent leur présence . C'est créatures sont appelé les créatures pacifiques.

Lapin

Pingouin

1.4 l'histoire dans le jeu

1.4.1 Le tutoriel (Théo et Romain)

Un défaut majeur de la première version de notre jeu était que celui-ci permettait de nombreuses actions (comme ouvrir un inventaire, se déplacer, jeter des objets) mais aucune information ne disait comment les réaliser. Nous avons donc décidé de créer un tutoriel expliquant au joueur comment effectuer les différentes actions dans le jeu. Nous profitons également de ce tutoriel pour introduire l'histoire du jeu puisque celle-ci s'enchaînera directement à la suite du tutoriel. Seulement contrairement au tutoriel, qui est propre au joueur puisque chaque joueur doit pouvoir apprendre à jouer, l'histoire sera propre au monde et ne se répétera pas pour chaque joueur.

Au lancement d'une partie tout nouveau joueur voit une fenêtre apparaître lui demandant si oui ou non il veut faire le tutoriel. Ne pas faire le tutoriel n'aura aucune influence sur le reste du jeu car celui-ci sera alors considéré comme réussi.



FIGURE 1 – présence du tutoriel en jeu

Dans le jeu, le tutoriel est visuellement présent grâce à deux zones de texte. La première zone de texte dans le rectangle rouge de la figure 1 raconte le développement de l'histoire

du tutoriel. C'est dans cette partie que les raisons des actions de Ille sont expliquées, ce qui donne une logique à l'existence du tutoriel. Ce texte est temporaire et ne reste que peu de temps à l'écran car celui-ci prend beaucoup de place et risquerait d'encombrer la vision du joueur. Plus tard dans le développement du jeu cette partie sera également lue oralement par un narrateur.

La seconde zone de texte dans le rectangle vert de la figure 1 est directement adressée au joueur. Ce texte explique concrètement quels sont les objectifs du joueur et comment les accomplir. Il donne par exemple des indications sur quelle touche appuyer pour ouvrir l'inventaire ou sauter. Tant que les objectifs ne sont pas tous accomplis cette zone de texte ne disparaît pas. En appuyant sur la croix se trouvant en bas à droite de cette zone une fenêtre apparaît nous demandant si on souhaite terminer automatiquement le tutoriel. Cette option a été rajoutée pour les joueurs qui auraient malencontreusement lancé le tutoriel ou pour les joueurs voulant sauter la dernière étape du tutoriel car celle-ci demande d'utiliser tout ce qui a été appris dans le tutoriel et est donc une partie assez longue.

Si un joueur quitte le jeu au milieu du tutoriel sa progression sera sauvegardée et il pourra continuer le tutoriel en relançant sa partie sans tout recommencer depuis le début.

1.4.2 Le système de Succès (Julien)

Les succès représentent les exploits réalisés par les joueurs, ils sont propres à une partie et sont donc partagés entre les joueurs qui peuvent coopérer pour réaliser un succès. Notre jeu laissant beaucoup de liberté au joueur, il est possible que celui-ci ne sache plus quoi faire. C'est là que les succès interviennent, car ils proposent différents défis auxquels le joueur peut se confronter. Ces exploits peuvent porter sur différents aspects de la survie comme le nombre de créatures tuées ou encore la quantité de potions consommées.

D'un point de vue technique, tous les différents succès sont stockés comme attribut static d'une classe particulière. Mais ces succès étant accomplis par les joueurs à n'importe quel moment, il fallait trouver un moyen de mettre à jour ces attributs à chaque instant tout en y passant le moins de temps possible. Les mises à jour doivent donc répondre à quelques contraintes :

- ne pas tester les succès déjà accomplis ;
- ne pas tester les succès dépendant de succès non accomplis (l'objectif *Tuer 10 monstres* ne peut pas être réussi avant l'objectif *Tuer 1 monstre*) ;
- indiquer au joueur le moment où il réussit un succès.

Pour régler ces premiers problèmes, nous avons opté pour l'utilisation de deux structures. Tout d'abord, un nouvel objet appelé *Succes* qui contient : un booléen indiquant si le *Succes*

est réussi, la liste de toutes les conditions nécessaires à l'accomplissement de celui-ci, son nombre de parents et une liste de *fils*. Ces deux derniers éléments permettent de créer une hiérarchie entre les différents succès et de faciliter l'affichage.

La deuxième structure est une liste de succès à mettre à jour, présente uniquement sur le serveur. A intervalles de temps réguliers, le serveur teste si les succès présents dans la liste sont accomplis. Si c'est le cas, alors il est retiré de la liste à tester et tous ses fils y sont ajoutés s'ils n'étaient pas déjà présents. De plus, l'affichage d'un pop-up en haut à gauche de l'écran est demandé sur chaque client attestant de l'accomplissement du succès (voir figure 2). La demande d'affichage est mise dans une file pour que les pop-up ne se superposent pas.

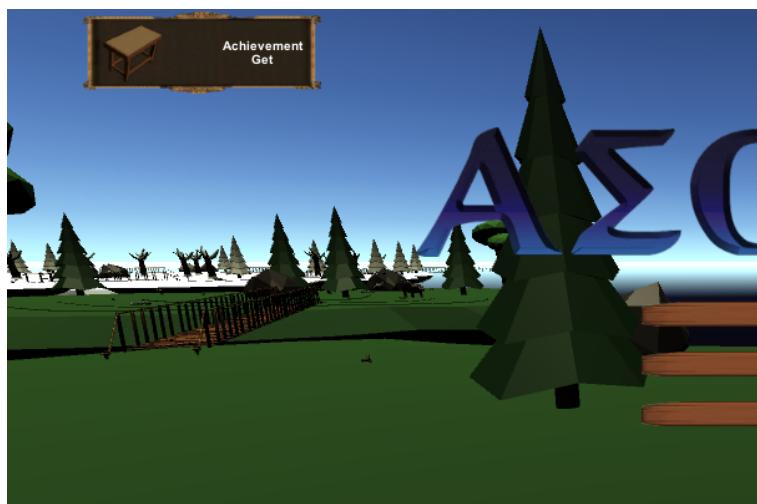


FIGURE 2 – Affichage du Pop-up

Une interface est également présente pour permettre aux joueurs de contempler les succès qu'ils ont accomplis (voir figure 3). Dans cette interface, chaque succès est représenté par une image et leur hiérarchie est visible grâce à des lignes. A terme, une si petite fenêtre ne sera pas suffisante pour afficher tous les succès, c'est pourquoi des barres de défilement seront présentes à droite et en dessous de la zone d'affichage (celle de droite est déjà visible).

La distinction entre les succès accomplis, les succès présents dans la liste (pouvant être accomplis) et ceux ne pouvant pas l'être se fait par l'utilisation de trois *Shader*. Comme expliqué dans le premier rapport de soutenance, un *Shader* indique à la carte graphique comment calculer la couleur de chaque pixel de l'écran.

Le premier *Shader* affiche l'image du succès sans la modifier (*succès accompli*), le deuxième ne tient compte que de la transparence de l'image et affiche un gris là où l'image n'est pas transparente (*succès pouvant être accompli*) et le dernier n'affiche pas du tout l'image (*succès impossible à accomplir pour le moment*).



FIGURE 3 – Interface des succès

Pour nous aider à tester les conditions des succès nous avons décidé de créer un objet regroupant des informations sur l'évolution d'une partie. Parmi ces informations, nous trouvons le nombre de créatures et de joueurs tués par un joueur ainsi le nombre de joueurs morts, la quantité d'objets fabriqués, etc.

Comme vous pouvez vous en douter, ce sont sur ces statistiques que les succès s'appuient. Elles sont donc, comme les succès, propres à la partie et non au joueur.

1.4.3 Le scénario

1.4.4 Le boss Final

2 L'environnement graphique et sonore

2.1 Interfaces 2D

2.2 Modèles 3D

Nous avons décidé de doter notre jeu d'un style graphique original. Nous avons cherché l'inspiration en parcourant l'asset store. Nous y avons trouvé le modèle de notre personnage principal et nous nous sommes accordés pour un univers cartoon.

Nous avons donc commencé la réalisation de notre première île mais ce fut un échec (figure 4). Nous n'avions pas assez bien défini la forme que nous voulions lui donner et il a donc fallu recommencer le modèle car celui-ci n'était pas vraiment cartoon et n'était pas adapté à la génération aléatoire du monde.

Nous avons alors décidé de réaliser tous nos futurs modèles grâce au logiciel Blender et d'importer le moins possible depuis l'asset store. Nous nous sommes donc recentrés sur l'aspect cartoon et en avons profité pour réaliser nos premiers éléments à savoir les arbres et un rocher (figure 5).

La réalisation des modèles suivants s'est faite et continuera à se faire petit à petit en fonction des demandes engendrées par l'avancement du projet.

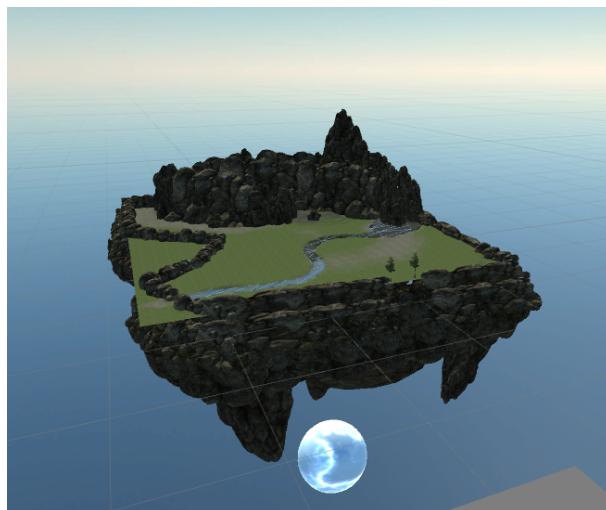


FIGURE 4 – Première île

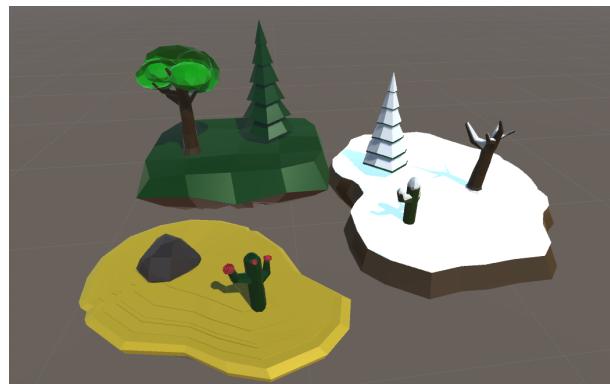


FIGURE 5 – Premiers éléments

2.3 Animations

Ille est un personnage que nous avons téléchargé depuis l'asset store de unity qui possédait initialement deux animations, « marcher » et « rester immobile ». Cependant ces deux seules animations ne nous suffisaient pas. Nous avons alors décidé de faire les animations de nos personnages nous-même. Le premier problème que nous avons rencontré est que notre personnage n'était pas accessible depuis notre version de *blender*, trop récente pour le modèle 3D que nous avons. Il nous a donc fallu trouver une ancienne version de *blender* compatible avec notre model 3D. Depuis nous essayons d'améliorer nos animations dans le but d'être le plus réaliste possible mais sans expérience dans ce domaine il est difficile de faire des animations convenables et celles-ci sont systématiquement remises en questions par les membres de l'équipe et donc modifiées.

2.4 Musiques et sons

Musique Afin de compléter l'univers que nous essayons de créer à travers notre jeu, nous avons ajouté des musiques de fond. Nous avons d'abord ajouté une musique qui est associée au menu principal uniquement. Trois autres musiques ont par la suite fait leur apparition, chacune d'elles est associée à un thème différent, la forêt, le désert, la neige ou l'automne, qui correspond chacun à un des biomes possibles pour les îles. Nous avons alors fait en sorte que les musiques se lancent en fonction du biome sur lequel le joueur se trouve. En multi-joueurs chaque personne peut se retrouver avec des musiques différentes au même moment.

Son Afin de ne pas avoir un jeu fade, nous avons introduit quelques sons. Ces sons nous permettent d'avoir une base que l'on pourra enrichir au fur et à mesure de l'avancée du développement de notre jeu. Nous avons tout d'abord ajouté un son pour l'ouverture et la fermeture de l'inventaire et du menu. Par la suite nous avons ajouté des bruits de pas afin d'entendre notre personnage marcher ou courir. Nous avons également ajouté un léger bruit lorsque le joueur récupère un objet.

L'ajout de divers nouveaux bruitages se sont faits en même temps que l'ajout de gameplay au jeu. Il a fallu rendre cohérent les actions du personnage car ce dernier ne pouvait pas obtenir du bois sans couper un arbre, il nous a alors fallu créer des bruitages pour les diverses actions indispensables à notre jeu comme couper du bois, ramasser des objets...

Ces multiples ajouts de bruitages et musiques ont nettement amélioré l'environnement sonore du jeu. Au final cet enrichissement sonore et ces nouvelles animations permettent une assez bonne cohérence entre ce que l'on souhaite réaliser et ce que l'on peut observer dans le jeu.

3 Systèmes de jeu

3.1 Systèmes fondamentaux

3.1.1 Déplacement du personnage et de sa caméra (Théo et Julien)

Le joueur détermine les déplacements de son personnage. Celui-ci peut avancer (avec la touche « w » ou « up »), reculer (avec la touche « s » ou « down »), s'orienter vers la droite (avec la touche « q » ou « right ») et s'orienter vers la gauche (avec la touche « d » ou « left »). Notre personnage peut courir si ce dernier se déplace et que l'utilisateur appuie sur la touche « shift », il peut aussi sauter si l'utilisateur appuie sur la barre d'espace.

Le saut à d'ailleurs été un élément problématique à implémenter car notre personnage ne devait pouvoir sauter qu'à partir du sol pour éviter qu'il ne s'envole en enchaînant des sauts dans les airs. Nous avons donc créé des restrictions permettant au personnage de ne sauter que depuis certains éléments du décors (qui regroupent presque tous les éléments).

A partir de ce moment notre personnage était devenu opérationnel mais à quoi cela servait-il si nous ne pouvions pas le voir ?

Nous avons alors décidé de créer une caméra qui suivrait notre personnage où qu'il aille. Cette caméra n'est pas une simple caméra fixe, elle possède des caractéristiques variées grâce auxquelles une grande liberté est accordée au joueur. Elle s'oriente grâce à la souris et même si on dit qu'elle suit le joueur, l'orientation des mouvements du joueur ne dépendent que de l'orientation de la caméra. Par exemple, faire reculer le joueur le fait en réalité se diriger vers la caméra.

La vue à la troisième personne permet au joueur de voir son personnage de coté, de devant ou encore de derrière. Tout cela en étant proche ou éloigné de son personnage, ce qui permet aussi d'avoir une vision plus générale de l'environnement. L'éloignement de la caméra et du personnage se contrôle avec la molette de la souris, il est donc malheureusement indispensable de posséder une souris à molette pour profiter à 100% de la caméra (le trackpad d'un ordinateur portable peut lui aussi permettre de profiter de la liberté de la caméra).

La vue à la première personne permet au joueur de voir exactement ce que son personnage voit. La passage d'une vue à l'autre se fait automatiquement en approchant ou éloignant la caméra du personnage.

3.1.2 Génération aléatoire du monde (Florian et Julien)

Comme nous l'avions spécifié dans notre cahier des charges, le monde du jeu est généré de manière aléatoire. Pour cette première soutenance, nous avons implémenté la génération

aléatoire d'un *chunk*, et la liaison des *chunks* entre eux, pour pouvoir générer un monde différent à chaque début de partie.

Pour le moment, le monde contient quatre *chunks* reliés par des ponts, chacun de ces *chunks* est choisi aléatoirement parmi des *chunks* préfabriqués composés d'îles, de ponts et d'ancres. Par la suite un biome sélectionné aléatoirement est appliqué sur le *chunk*, ce qui détermine le type des éléments qui pourront être mis à la place des ancre. Pour finir chaque ancre est donc remplacée par un élément cohérent avec le biome. Si le biome est *désert* par exemple alors une ancre peut être remplacée par un cactus, un rocher, du sable... ou rien (c'est à dire un élément vide). Cette méthode de génération implique des milliers de possibilités pour un monde composé seulement de quatre *chunks*.

Nous avons donc tous les éléments pour créer des *chunks*, il ne restera plus qu'à faire apparaître le monde de manière procédurale avec un agencement des *chunks* correct. Puis il sera aussi nécessaire de sauvegarder le monde avec les modifications que les joueurs y ont apportées.

3.1.3 La sauvegarde (Florian)

La sauvegarde est une des grandes implantations pour cette deuxième soutenance, celle-ci permet aux joueurs d'arrêter une partie et de la reprendre quand ils veulent. Le joueur qui crée une partie en est en quelque sorte le « propriétaire ». La partie est hébergée sur son ordinateur qui devient le serveur pour cette partie. C'est sur ce même ordinateur que les sauvegardes seront effectuées.

La première chose que nous avons dû modifier est la sauvegarde du monde. Pour cela nous avons utilisé un système de *seed*. Le principe est le suivant, lorsque le joueur crée un monde alors un nombre va être choisi aléatoirement. Ce nombre est appelé la *seed* du monde. À partir de celui-ci nous pouvons en déduire toutes les informations sur le monde, c'est-à-dire, les îles et leur position mais aussi leur biome et leur environnement. Ce système rend la sauvegarde légère en mémoire et donc rapide. Nous avons dû compléter le système de génération du monde et plus précisément d'un chunk.

Chaque chunk va calculer son propre *seed* à partir du *seed* du monde et de sa position, à partir de ce nouveau *seed* nous allons pouvoir connaître ses îles et son biome. Puis pour chacune des ancre nous savons l'élément qui doit être à cet emplacement.

Nous avons uniquement besoin de sauvegarder la *seed* pour sauvegarder notre monde. Cependant les modifications du joueur ne peuvent pas être déterminées grâce à la *seed*. Nous avons donc besoin de sauvegarder des informations supplémentaires telles que l'heure de la journée, les éléments détruits et posés, mais aussi les caractéristiques des cristaux. Enfin

l'arrivée des succès et des statistiques de jeu doit être elles aussi sauvegardées. Pour finir chaque joueur a des informations à sauvegarder en voici la liste :

- la position ;
- le contenu de son inventaire ;
- sa vie, sa faim et sa soif ;
- les effets (régénération, poison ...) et leur temps restant ;
- son équipe en mode *PvP* .

Enfin la sauvegarde s'effectue automatiquement lors de l'arrêt du serveur, mais cela ne suffit pas. En cas de crash ou si le « propriétaire » ne ferme pas correctement l'application, la sauvegarde ne peut pas s'effectuer. Il y a donc une sauvegarde automatique toutes les minutes. Celle-ci ne provoque aucun ralentissement du jeu car elle est très légère.

3.1.4 Menus (Romain et Florian)

Un élément essentiel à tout jeu est indéniablement de pouvoir le quitter et le commencer. Cela paraît évident mais il a tout de même fallu créer des interfaces permettant ces actions au début du jeu (pour commencer une partie) et au milieu du jeu (pour arrêter de jouer).

Pour cela nous avons créé des menus permettant non seulement de démarrer une partie ou de quitter le jeu mais aussi de modifier les options disponibles.

Actuellement le menu permet de modifier la langue, le volume sonore du jeu , la sensibilité de la caméra et la distance de rendu. En cours de jeu, ces menus s'ouvrent avec la touche *escape* quand aucune autre interface n'est ouverte et se ferment avec la même touche ou avec le bouton *continuer*. Lorsque le menu est ouvert, le joueur et la caméra du joueur ne peuvent pas bouger. Lors du lancement du jeu on arrive directement à un menu mais celui-ci ne se ferme pas avec la touche *escape*. Ce menu principale est plus complexe que le menu en jeu car il possède sa propre scène et de nombreux éléments de personnalisation.

Menu Principale Dans ce menu principal nous avons rajouté une petite interface donnant accès à la liste des mondes disponibles. De même, on peut enregistrer des adresses de serveurs qui nous permettront de ne pas devoir ré-entrer les adresses IP des serveur auxquels on veut avoir accès en prenant le risque de se tromper.

Sur la figure 6, la partie encadrée en rouge n'apparaît que lorsque l'on appuie sur le bouton « Join » et montre la liste des serveurs que l'on a enregistrés. De même, une interface similaire apparaît lorsque appuie sur le bouton « Play » mais cette fois avec une liste de mondes. Cette liste de mondes est générée automatiquement en parcourant les fichiers de la sauvegarde.



FIGURE 6 – Menu Principal

Dans cette interface, il est possible de sélectionner un serveur ou un monde en cliquant dessus. On ne le voit pas vraiment mais une fine ligne blanche forme alors le contour du monde sélectionné. Il est alors possible de supprimer le monde ou le serveur de la liste en appuyant sur le bouton « Delete » ou de lancer le jeu en appuyant sur le bouton « Start ». Lorsque la liste possède plus de 3 éléments l'écran n'en affiche que trois mais il est possible de parcourir toute la liste en appuyant sur les flèches se trouvant dans la partie droite du rectangle rouge.

Enfin il est possible d'accéder à une interface de création de monde ou de serveur en appuyant sur le bouton « Create ».

Dans l'interface de création de monde (voir figure 7) Il faut donner un nom à son monde, choisir le type de jeu que l'on veut faire en appuyant sur le bouton « Coop/PVP » pour changer de type et si l'on veut on peut préciser la *seed* de la map en base 36 (*seed* composée d'au maximum six chiffres et lettres en minuscules). Cette *seed* est propre à un monde particulier et permet donc de régénérer le même monde (sans prendre en compte les modifications faites par le joueur). Il y a donc 2 176 782 336 mondes différents qui peuvent être générés. Il suffit alors d'appuyer sur le bouton « Create » pour créer un nouveau monde et commencer une partie. Pour des raisons de comptabilité des sauvegarde il est impossible de créer plusieurs sauvegardes du même nom.

Dans l'interface d'enregistrement de serveur (voir figure 8) il nous est seulement demandé de choisir une adresse IP et un nom sous laquelle l'enregistrer puis d'appuyer sur le bouton « Create » pour enregistrer un nouveau serveur. Contrairement à la création de monde on peut créer plusieurs serveurs de même nom ou de même adresse IP sans créer de conflit.

Notre génération de monde commençant à être longue avec la génération des îles et de leurs graphes Nous avons rajouté un écran de chargement sous la forme d'une vidéo format



FIGURE 7 – Création de monde

MP4 tournant en boucle.

Enfin en jeu, nous avons rajouté des options permettant d'influencer la caméra. La première permet de modifier la sensibilité de la souris sur les mouvements de la caméra et la deuxième permet de modifier la distance maximum de vision de la caméra. Cette dernière option a été rajoutée car avec l'agrandissement de notre monde les plus vieux ordinateurs n'étaient pas capable de faire tourner le jeu sans laguer.



FIGURE 8 – Enregistrement d'un serveur

première scène La scène du menu, aussi appelée première scène, a une valeur très particulière à nos yeux. Même si cette scène ne sert que de paysage à la création de partie, elle nous permet de mettre en avant tout l'univers graphique de notre jeu et de montrer au joueur les différents éléments présents dans le jeu.

Lors du dernier rapport nous vous avions présenté une île unique avec une caméra orbitant autour. Aujourd’hui, en plus d’évoluer dans un vrai archipel, la caméra suit un mouvement ne pouvant plus être réduit à une simple fonction mathématique. Nous avons décidé que la caméra suivrait un parcours bien plus complexe (voir pointillés verts sur la figure 9) et bien plus intéressant. Plutôt que de se limiter à une orbite circulaire, la caméra va serpenter entre les îles, s’en rapprocher, s’en éloigner, voire passer en dessous.

Chaque point sur la figure 9 est en réalité une balise. La première balise connaît la deuxième balise qui connaît la troisième balise et ainsi de suite jusqu’à la dernière balise qui elle connaît la première balise. Nous obtenons ainsi un circuit fermé et il ne reste plus qu’à faire avancer la caméra le long de ce circuit. Cette technique permet de résumer des mouvements très complexes en un enchaînement de petits déplacements linéaires ce qui est bien plus simple à réaliser. Plus il y aura de balises le long du circuit, plus le déplacement paraîtra fluide et naturel.

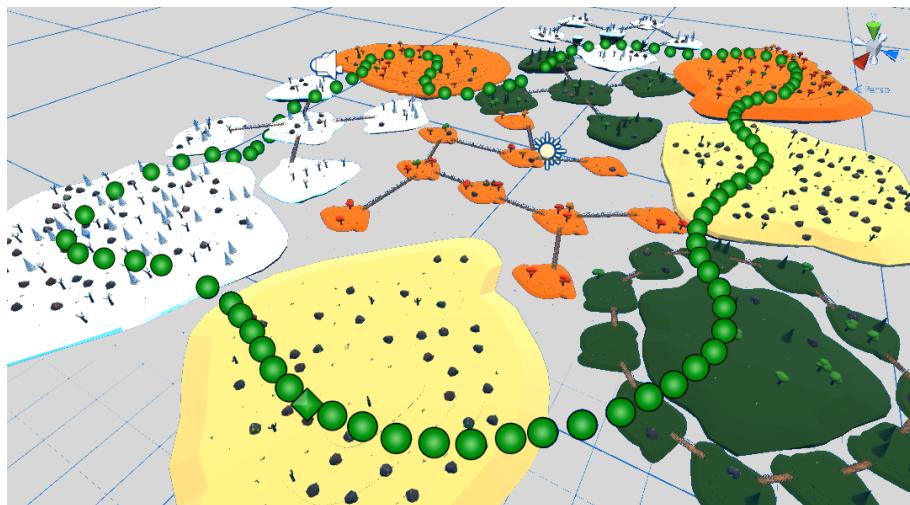


FIGURE 9 – La Première Scène

personnalisation du personnage

3.2 Le système d'inventaire

3.2.1 Les différents types d'objets

Pour pouvoir gérer les ressources que possède un joueur, nous avons dû créer une classe que nous avons tout simplement choisi d'appeler *Item*.

La classe *Item* permet de créer un patron d'objet ne contenant que des constantes immuables qui sont le nom (dans chacune des langues disponibles, pour le moment il s'agit du français et de l'anglais), la description (elle aussi dans chacune des langues proposées), un identifiant unique, une quantité maximale par tas d'objets, une texture 2D et un élément correspondant à l'objet 3D. Cependant, le joueur peut posséder plusieurs exemplaires d'un même objet. Les objets qu'il possède sont donc une nouvelle classe appelée *ItemStack* contenant une quantité (inférieure ou égale à la quantité max de l'objet) et un pointeur vers l'objet qu'il représente. Ainsi, les informations communes à plusieurs exemplaires d'un même objet ne sont pas dupliquées, seules la quantité de l'objet peut changer. Il y a alors un gain de mémoire.

Nous avons ensuite fait la distinction entre deux catégories d'objets, les premiers qui ne sont que des ressources et les seconds qui peuvent être utilisés et ont un effet particulier soit sur l'environnement soit sur le joueur lui-même. Ces derniers objets peuvent être équipés, c'est-à-dire mis dans la main du joueur. Pour se faire, il suffit que l'objet soit placé dans la barre d'outils et que le sélecteur soit sur la case du dit objet. Ainsi, l'objet apparaîtra dans la main du personnage et tous les joueurs pourront le voir.

Une fois un objet équipé, il suffit de faire un clic droit pour l'utiliser. Il existe plusieurs types d'interaction :

- **Les consommables** : qui lors d'un clic droit sont consommés et produisent un effet sur le joueur ;
- **Les outils** : qui permettent d'interagir avec certains éléments de l'environnement ;
- **Les objets pouvant être posés** : qui une fois équipés, apparaissent dans le monde et peuvent être posés avec le clic droit.

Commençons par les consommables. Leurs effets sont très variés, ils peuvent influer sur les caractéristiques d'un personnage, allant de sa vitesse jusqu'à sa santé en passant par sa hauteur de saut. Ce sont les consommables qui permettent entre autre de se nourrir et de boire.

Passons ensuite aux outils. Jusqu'à présent, il n'était possible d'interagir qu'avec les cristaux et à condition d'être suffisamment proche d'eux. Maintenant, il est possible d'interagir

avec n'importe quel élément de l'environnement à condition d'être équipé d'un outil approprié. Il est en effet impossible de couper un arbre à la main, il faut une hache pour cela. Si l'élément est trop éloigné, le joueur peut garder son clic droit enfoncé, son personnage va alors s'approcher des ressources l'entourant et les récolter jusqu'à ce qu'il n'y ait plus de ressources dans les environs ou que son outil équipé ne soit pas approprié à la récolte.

Enfin, pour ce qui est de la pose des objets, lorsqu'un objet pouvant êtreposé est équipé, celui-ci apparaît en transparence dans le monde. Cet objet transparent est situé à une unité de distance devant le personnage et est tourné vers lui. Mais ce n'est pas l'objet réel, c'est une prévisualisation de l'objet qui permet au joueur de choisir où il veut réellement le poser. Un indicateur de couleur permet d'informer le joueur sur la validité de la position actuelle de l'objet (voir figure 10). Il n'est pas possible de poser une table à travers un arbre ou sur un rocher. Le joueur peut confirmer la pose de l'objet par un simple clic droit.

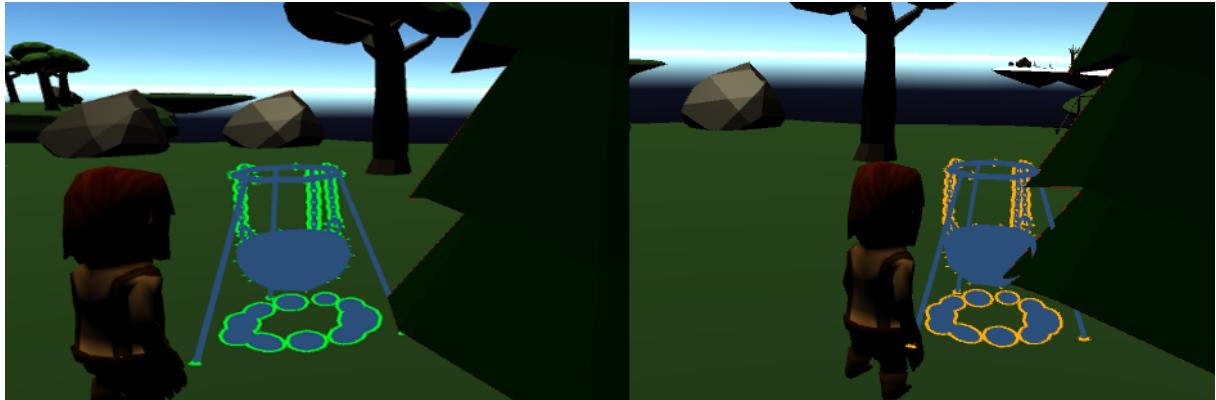


FIGURE 10 – La prévisualisation d'un objet

3.2.2 Inventaire et stockage d'objets

Pour que notre personnage puisse posséder un grand nombre d'objets de toutes sortes sans avoir à les porter sur lui, et donc se déplacer assez facilement, nous avons créé un inventaire (figure 11). Celui-ci est constitué de 24 cases sous forme de 4 lignes et 6 colonnes. Chaque case peut contenir un élément, qui peut être déplacé grâce à un *drag and drop*, ou plusieurs objets identiques, ils forment alors un *tas*. De plus le joueur peut avoir une description de chacun des éléments en passant sa souris sur celui-ci. Cet inventaire s'ouvre lorsque l'on appuie sur la touche *inventaire* (touche *i* par défaut) et peut ensuite être fermé avec la même touche *inventaire* ou la touche *escape*.

Lorsque l'inventaire est ouvert le joueur et la caméra ne peuvent plus bouger via les touches de déplacement ou la souris (mais le joueur continuera de tomber si celui-ci était

en train de chuter). L'inventaire se sauvegarde automatiquement en local lorsque le jeu est quitté normalement ou lorsque qu'une modification de son contenu s'effectue (quand un objet est jeté ou récupéré). Ce même inventaire est alors récupéré d'une partie sur l'autre.



FIGURE 11 – Format de l'inventaire

Lorsque nous sommes en train de bouger il est impossible de voir ce que contient notre inventaire et il faut nécessairement l'ouvrir, et donc s'arrêter, pour interagir avec lui. Actuellement ce n'est pas un problème mais quand il sera nécessaire d'utiliser des ressources pour survivre et interagir avec l'environnement, nous voulons que ces dernières soient facilement accessibles sans devoir s'arrêter (pour pouvoir boire une potion de santé alors qu'un slime nous poursuit par exemple).

Nous avons donc ajouté une barre de sélection (figure 12) synchronisée avec la 4^e ligne de notre inventaire. Le joueur peut sélectionner une case de cette barre en utilisant les touches alphanumériques correspondantes et utiliser l'objet s'y trouvant. Cette barre apparaît même quand le joueur n'a pas affiché son inventaire puisque son but est justement d'utiliser des objets hors de l'inventaire.

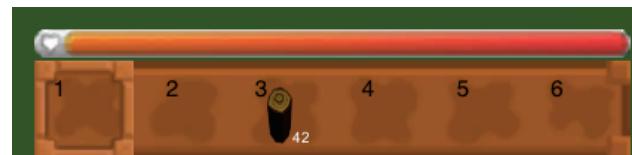


FIGURE 12 – barre d'outil

3.2.3 La création d'objets (Romain)

Une nécessité en survie, et donc dans tous les jeux de survie, est l'obtention d'outils primitifs pour fabriquer un abri ou chasser. Pour cela, il faut pouvoir créer ces outils. Nous avons donc rajouté un système de création d'objets. Chaque objet peut être créé à partir d'une recette définissant le type et le nombre d'ingrédients nécessaires à sa réalisation. Les objets qu'il est possible de créer se classent en cinq catégories : les établis qui permettant de fabriquer plus d'outils, les ressources de construction qui sont principalement utilisées pour fabriquer des objets plus évolués, les consommables, les outils et les armures

La réalisation de chacune de ces recettes nécessite plusieurs conditions. D'abord, il faut posséder les composants nécessaires à la fabrication de l'objet souhaité. Cela paraît naturel, il est impossible de faire une pioche à partir de rien tout comme on ne peut pas faire de ragout sans viande. Il faut aussi posséder une place vide dans l'inventaire. Même si le joueur possède déjà les composants de base de l'objet qu'il veut fabriquer, la transformation d'un bâton en une pioche ne se fait pas de manière directe. Il faut d'abord créer un manche et forger un fer de pioche puis assembler les deux. Cet espace vide dans l'inventaire est nécessaire pour toutes ces actions. Enfin, il faut se trouver à proximité des établis nécessaires à la fabrication des objets souhaités. Aucune indication n'est donnée pour savoir quel établi est nécessaire pour faire un objet mais cela reste évident : il faut par exemple être à proximité d'une forge pour fabriquer des lingots ou un autre objet en fer.

Mais malgré tout cela, il manque une chose pour pouvoir fabriquer facilement des objets : l'expérience. Réaliser une recette avec succès ne se fait pas dès la première tentative, fabriquer du verre ou forger une épée sans expérience demande plus de ressources que nécessaire. Tant que la recette n'est pas maîtrisée, le coût en ressources sera doublé. Pour maîtriser une recette il n'existe actuellement qu'une seule possibilité : La chance.

Enfin certaines recettes sont cachées et il est impossible de savoir qu'elles existent sans remplir certaines conditions ou rechercher sur le wiki.

En jeu, il est possible d'accéder au système de craft en même temps que l'inventaire. Sur la partie gauche de l'écran cinq boutons permettent de choisir quelle catégorie d'objets on veut créer (entourée en bleue sur la figure 13). Après avoir cliqué sur ce bouton, une liste d'éléments pouvant être créés apparaît (entourée en rouge sur la figure 13). Les flèches en haut et en bas permettent de faire défiler les éléments. En sélectionnant sur l'un de ces éléments sa recette s'affiche sous la forme d'une liste de composants (entourée en vert sur la figure 13). Dans cette liste, la dernière case est cochée en vert si l'objet peut être créé, ou occupée par une croix rouge dans le cas contraire. La création d'un objet se fait en cliquant sur la case cochée en vert.

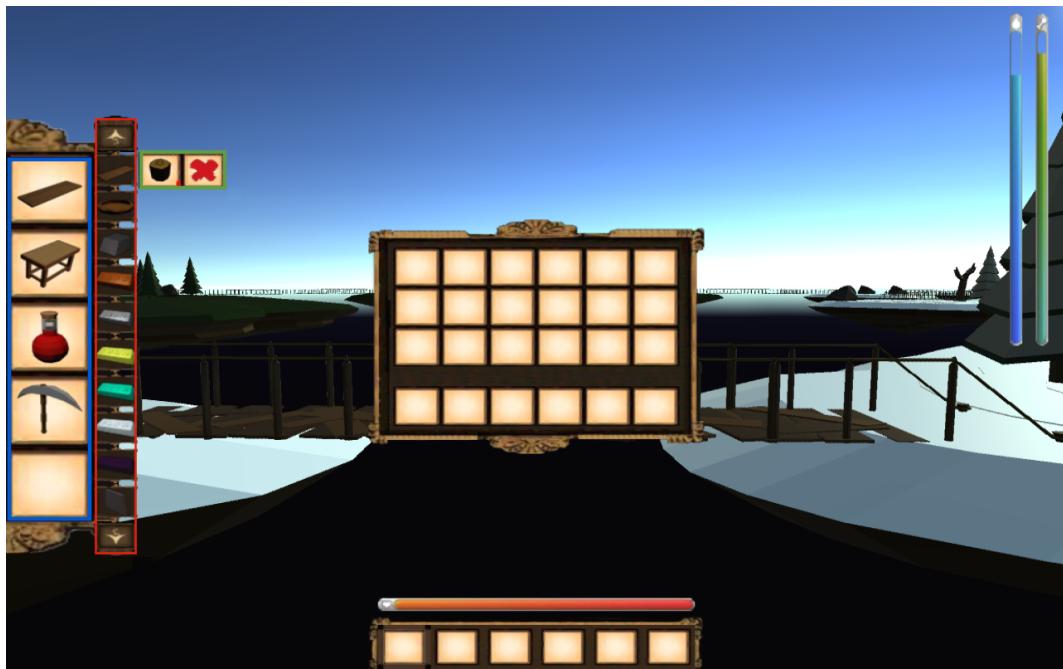


FIGURE 13 – Interface de la création d'objet

Pour permettre au joueur d'améliorer les capacités de son personnage, nous lui donnons la possibilité de réaliser des potions, qui sont bien sûr des consommables.

3.2.4 Drop d'Item (Florian et Julien)

Le drop d'un item correspond à son passage de l'inventaire au monde 3D. Pour cela, le joueur à la possibilité, depuis son inventaire, de glisser un *tas* d'objets hors de l'inventaire. Il peut aussi grâce à une touche (la touche « Q » par défaut), lancer l'objet sélectionné dans sa barre d'outil. Ces actions vont faire apparaître l'objet dans le monde en le lançant à quelques mètres du personnage. Dans cet état, n'importe quel joueur peut alors ramasser l'objet dès qu'il est assez proche de lui. La récupération de l'objet se fait par des échanges d'information entre serveur et clients pour garantir que seul un joueur ramasse l'objet, et qu'il a suffisamment de place dans son inventaire pour le stocker. De plus si deux objets identiques sont assez proches alors ils vont se rassembler. Cela permet d'afficher moins d'objets et donc d'améliorer les performances du jeu. D'autre part les objets « hors inventaire » depuis plus d'une minute disparaissent, cela permet encore une fois d'améliorer les performances et de ne pas avoir une île envahie de ressources non utilisées.

3.3 Système de survie

3.3.1 Cristal (Romain et Julien)

Une particularité du monde d’Aegina est la présence, sur certaines de ses îles, d’un cristal. Ce cristal n’est pas juste comme les autres éléments du décors tels que les arbres ou les pierres qui sont là pour être récoltés et transformés par les joueurs. Les cristaux eux sont indestructibles. Dans la nature, les cristaux ont une couleur grise indiquant qu’il n’ont pas été activés par un joueur, ils n’ont alors par d’influence visible sur l’environnement. Lorsqu’un joueur tente d’interagir avec un cristal inactif, une fenêtre (figure 16) apparaît proposant au joueur d’activer le cristal en échange de ressources. Les informations contenues dans cette fenêtre ne sont pas encore toutes définies, mais elle a été dimensionnée pour pouvoir contenir les prochains ajouts prévus.

Une fois activé, le cristal change de couleur pour prendre celle de l’équipe du joueur (bleu lorsque tout les joueurs sont dans la même équipe). A l’activation, un cristal se spécialise dans l’une des trois catégories suivantes de manière aléatoire :

- **Attaque (60%)** : catégorie des cristaux d’attaque (ou de défense ce qui revient au même).
- **Récolte (30%)** : catégorie des cristaux de production
- **Portail (10%)** : catégorie des cristaux de portail.

Dans le cadre bleu de la figure 14 se trouvent les informations sur le type du cristal et ses pouvoirs. Les pouvoirs du cristal se distinguent en trois partie :

- le pouvoir de guerre qui renforce la puissance du joueur. Si un joueur se trouve sur l’île, il bénéficie d’un bonus en attaque et en armure significatif. En effet si le pouvoir de guerre d’un cristal est maximisé, les joueurs sur l’île verront leur attaque multipliée par deux par rapport à un joueur sans cristal. De plus, ils réduiront également de moitié tous les dégâts qu’il recevront ;
- le pouvoir de la récolte qui renforce la production du joueur. Si un joueur sur l’île tente de fabriquer un objet, il aura plus de chance de maîtriser une recette. De plus les ressources autour de ce genre de cristal auront plus de chance de réapparaître ;
- le pouvoir divin qui a une influence plus tactique sur les joueurs. Actuellement ce pouvoir permet juste de désigner un cristal comme un point de réapparition possible pour le joueur. Plus tard, il permettra également de se téléporter vers les autres cristaux possédant un pouvoir divin.

Il est possible d’augmenter n’importe quel pouvoir du cristal en appuyant sur le bouton « Upgrade » de la colonne du pouvoir voulu et en consommant les ressources présentent dans le cadre rouge sur la figure 14. Cependant un cristal après son activation ne peut être



FIGURE 14 – Interface de l’interaction avec le cristal

amélioré que trois fois et chaque amélioration est plus chère que la précédente.

En mode « joueur contre joueur », un cristal est associé à l’équipe du joueur qui l’a activé. Ses pouvoirs ne s’appliquent qu’aux joueurs de cette équipe.

3.3.2 La survie du joueur dans l’environnement

vie et mort La survie étant le thème de notre jeu, il nous fallait construire de bonnes bases pour le développement de cette dernière. Aujourd’hui les éléments clés de notre survie sont présents. Parmi ces éléments, trois sont visibles directement en jeu grâce à des barres (figure 15). Deux de ces barres, l’une symbolisant la soif et l’autre la faim du personnage, se situent en haut à droite de l’écran tandis que la troisième barre, symbolisant la santé (les points de vie), se trouve au dessus de la barre de sélection.

Lorsque la barre de faim ou de soif est vide, les point de vie du personnage commencent à diminuer. Cependant ces deux attributs (soif et faim) ne sont pas les seuls à pouvoir s’attaquer aux points de vie. Si le personnage décide de sauter dans le vide, sa santé en souffrira également car malgré sa force et sa volonté, Ille ne sait pas voler.

Quand le personnage se retrouve sans point de vie (barre de santé vide), il sentira ses forces disparaître, c’est ce que nous appelons communément la mort. Cependant, dans notre grande bonté, nous avons décidé que la mort ne serait pas tout de suite définitive et donc

chaque mort sera accompagnée d'un nouveau début. Pour le moment, le joueur *respawn*, il est téléporté sur la première île qu'il a traversée et ses attributs (santé, soif et faim) sont remis au maximum. Son inventaire est identique à celui qu'il avait au moment de sa mort.



FIGURE 15 – Disposition des différentes barres

En tant que développeur d'un jeu de survie nous prenons tout aussi soin de la survie de nos joueurs que de leur mort. Il faut que la mort soit assez terrifiante pour que le joueur évite de mourir intentionnellement dans le seul but de revenir plus rapidement à son point de réapparition. Lors de la première soutenance, lorsque le joueur mourait il revenait à son point de réapparition sans aucune contrepartie et récupérait sa fin, sa soif et sa vie.

Maintenant, lorsque le joueur meurt son personnage disparaît et une fenêtre apparaît devant l'écran du joueur lui demandant si il veut réapparaître ou si il veut quitter. Si il décide de continuer la partie il sera transporté sur un lieu de réapparition mais tous les objets de son inventaire seront jetés au sol sur le lieu de sa mort. Si il décide de quitter, la prochaine fois qu'il lancera le jeu, il se retrouvera sur un point de réapparition mais tous les objets de son inventaire auront été jetés sur son lieu de mort avant de quitter la partie. Comme il n'existe pas d'autre moyen de transporter des objets que dans son inventaire, la mort revient presque à recommencer une partie à zéro. Heureusement, si vous avez un allié avec vous, celui-ci pourra récupérer tous vos objets mais si un ennemi se trouve dans les parages celui-ci pourra décider de vous tuer pour récupérer vos équipements.

Les points de réapparition sont les îles possédant un cristal de la même équipe que le joueur et la première île.

interaction avec l'environnement Il a été décidé que le joueur ne pourrait interagir avec son environnement que d'une manière très particulière. En effet, la plupart du temps,

le joueur utilise un curseur pour sélectionner l'objet avec lequel il veut interagir. Mais ce mécanisme a un inconvénient majeur : la souris doit être réservée au déplacement du curseur et ce n'est pas envisageable dans notre cas car la souris est utilisée pour le déplacement de la caméra.

Il a donc fallu trouver une autre méthode pour sélectionner les objets et bien faire comprendre au joueur quel est l'objet actuellement sélectionné. Après concertation, nous avons remarqué que l'objet avec lequel le joueur veut interagir est souvent l'objet le plus proche de lui (*nearElement*). Nous avons donc décidé qu'un joueur ne pourrait interagir qu'avec le *nearElement* et ce en utilisant le clic droit. L'avantage de n'interagir qu'avec le *nearElement* est que celui-ci est unique. Il ne peut y avoir qu'un seul objet le plus proche et si plusieurs objets sont à égale distance, alors le premier rencontré sera considéré comme *nearElement*.

Maintenant que la méthode théorique est en place, il faut l'appliquer. Pour ce faire, il suffit de comparer les distances séparant le personnage des objets environnants. La sélection du *nearElement* se résume donc à une recherche de minimum. Mais il faut définir l'ensemble de recherche et, pour cela, être capable de différencier les objets de l'île par exemple. La solution que nous utilisons est la suivante : nous ajoutons un *tag* particulier à tous les objets avec lesquels le joueur peut interagir. Pour ne pas avoir à considérer l'ensemble des objets de monde, nous définissons une sphère d'interaction. Le joueur ne peut interagir qu'avec les objets se trouvant à l'intérieur de cette sphère, les autres sont trop éloignés. L'objet le plus proche du joueur se trouve donc obligatoirement à l'intérieur de cette sphère.

Le *nearElement* est maintenant identifié mais il est impossible pour le joueur de faire la différence entre le *nearElement* et les autres éléments. Pour mettre en évidence le *nearElement*, nous utilisons un *shader* particulier. Les *shaders* indiquent au système de rendu comment un objet doit être affiché à l'écran. Le *shader* que nous utilisons ajoute un contour rouge à la silhouette de l'objet et assombrit les couleurs de l'objet. Il devient donc très facile de distinguer le *nearElement* des autres objets.

évolution de l'environnement Pour réaliser un cycle jour nuit, nous avons dû déterminer la durée d'une journée que l'on a divisée en 30 phases. Chacune d'elle correspondant à une image de *skybox*. Seulement cette méthode à présenté quelques inconvénients. Entre autre, le changement de *skybox* était trop brusque et les différentes *skybox* n'étaient pas homogènes. C'est pourquoi nous avons dû trouver une autre méthode : la *directional light*.

La *directional light* est un *gameobject* proposé par unity. Son intérêt majeur est que l'orientation de cette lumière influence la *skybox* du monde. De plus nous pouvons choisir la couleur et l'intensité de cette lumière. Pour ce faire nous utilisons une fonction qui convertit

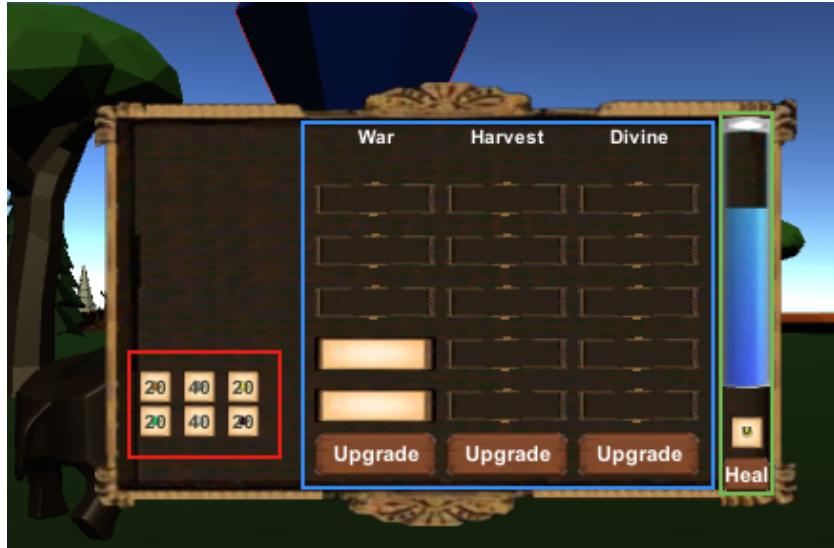


FIGURE 16 – Interface du cristal

la longueur d'onde des rayons lumineux en couleur RGB¹ et nous représentons le spectre solaire à l'aide de la fonction mathématique suivante où λ est la longueur et t le temps :

$$\text{spectre}(\lambda, t) = 255 * ((3.25t - 4)\lambda^2 + (-4t + 4)\lambda + (0,08(t - 1)))$$

La figure 17 montre les évolutions du spectre lumineux. L'arc des abscisses représente la longueur d'onde et celui des ordonnées l'intensité. Pour chaque valeur de t (moment de la journée), nous pouvons donc calculer un spectre lumineux. La courbe bleue ($t = 0$) représente le spectre quand le soleil est au zénith et la courbe rouge ($t = 1$) correspond au spectre au moment du coucher du soleil. Les courbes vertes ($t = 0,3$ et $t = 0,6$) montrent quant à elles l'évolution de la courbe. A chaque instant, nous échantillonnons la courbe actuelle en prenant 40 valeurs équiréparties, ce qui nous donne un tableau des couleurs (en RGB) que notre astre est censé émettre. Ce tableau contient environ 40 valeurs or une *directional light* ne possède qu'une seule couleur. Il a donc fallu trouver un moyen pour réunir ces 40 couleurs en une seule. Pour ce faire, nous parcourons toutes les valeurs du tableau et nous formons une nouvelle couleur en choisissant les plus grandes valeurs de rouge, de vert et de bleu rencontrées.

Nous utilisons cette même fonction en fixant $t = 0$ (courbe bleue, figure 17) pour définir l'intensité lumineuse. Dans ce cas, λ est une représentation du temps et ainsi, $\lambda = 0$ représente l'intensité au levé du soleil et $\lambda = 1$ l'intensité au couché.

1. <http://stackoverflow.com/questions/1472514/convert-light-frequency-to-rgb>

Nous avons ensuite implémenté la rotation et le déplacement de la *directional light*. Ce fut assez simple car nous étudions les mouvements circulaires en cours de physique. Il nous a donc suffi de copier les équations paramétriques horaires en base cartésienne. C'est-à-dire :

$$x(t) = R\cos(wt)$$

$$z(t) = R\sin(wt)$$

Où R est le rayon de l'orbite et w la vitesse angulaire.

Ainsi, la *directional light* suit une orbite circulaire autour du point de coordonnées (0,0,0). L'orientation de la *directional light* est définie grâce à la fonction *lookat* qui oriente un *gameobject* vers un autre *gameobject*, le point (0,0,0) dans notre cas.

Un procédé similaire est utilisé pour la première scène du jeu. En effet, dans cette scène, la caméra orbite autour d'une île et nous nous servons de la fonction *lookat* pour orienter la caméra vers l'île. Mais, contrairement au cas de la *directional light*, le déplacement utilise les propriétés de la base polaire. En effet, comme l'objet est toujours orienté vers le centre de son orbite, il suffit de lui appliquer une vitesse constante pour qu'il décrive une trajectoire circulaire.

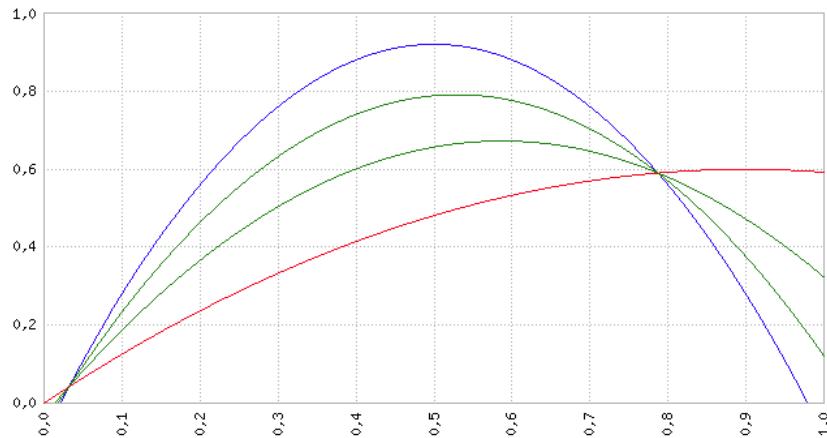


FIGURE 17 – évolution du spectre lumineux en fonction du temps

3.3.3 Les Créatures (Florian)

Chaque type de créature est défini par un ensemble de caractéristiques permettant de les générer dans les biomes adaptés, de récupérer des ressources lors de leur mort, ... Voici l'ensemble de ces caractéristiques :

- Une probabilité d'apparition ;
- Des dégâts ;
- Une vision d'agro ;
- Une vision de fuite ;
- Une vitesse de déplacement en marche ;
- Une vitesse de déplacement en course ;
- Une vitesse d'attaque ;
- Une liste de biomes sur lesquels la créature peut apparaître ;
- Un ensemble d'objets que la créature peut produire à sa mort ;

L'arrivée des créatures dans le monde d'Aegina a imposé la création d'intelligence artificielle, leur comportement devant être automatiquement géré par le jeu.

Pour gérer les déplacements des créatures, nous avions besoin de connaître les chemins possibles et donc la topographie des îles. Nous avons commencé par la génération de graphes lors du chargement des chunks. Chaque chunk possède son graphe qui contient un certain nombre de noeuds comportant tous plusieurs informations telles que la présence d'éléments, par exemple un arbre ou un rocher, et les liens entre ces éléments (voir figure 18).

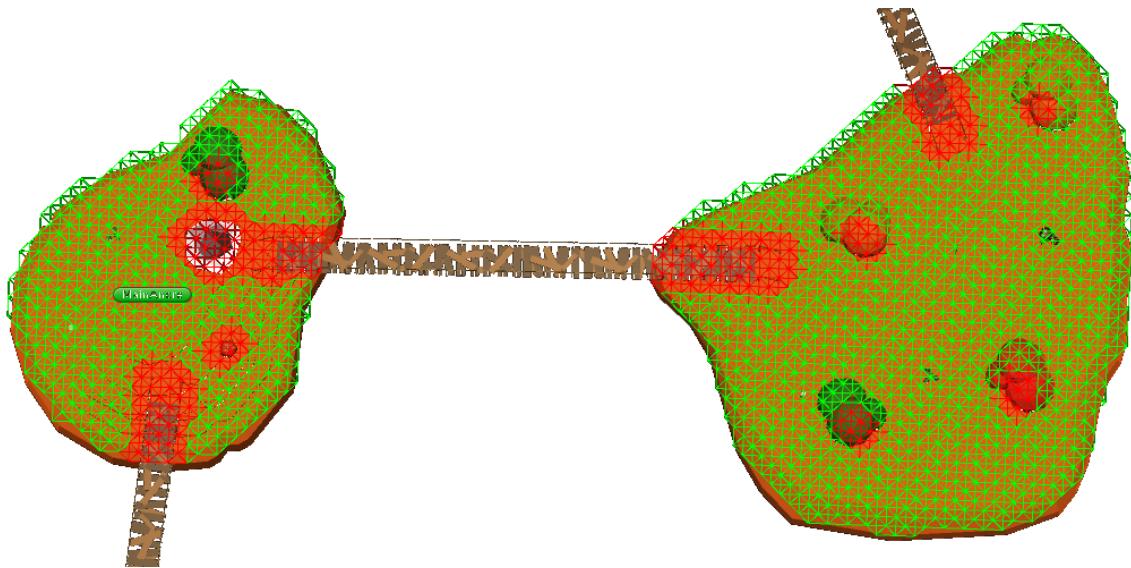


FIGURE 18 – Graphe

Une fois le graphe généré pour chaque chunk, nous avons pu coder un algorithme de *path finding*, le *A**. Ceci nous permet de trouver le meilleur chemin pour aller d'un point A à un point B du graphe. De plus les îles séparées par du vide ou un pont ont des graphes disjoints, cela permet entre autre que les créatures restent sur leur île.

Une fois le déplacement implémenté, nous nous sommes occupés de l'appliquer aux créatures.

Nous avons la possibilité de déplacer nos créatures intelligemment en tenant compte de leurs caractéristiques. Pour déterminer le comportement d'un créature, poursuite du joueur, fuite ou attaque, nous tenons uniquement compte de la distance entre la créature et le joueur le plus proche pour le moment. Tant qu'une créature est loin de tout personnage, elle va se promener en choisissant aléatoirement un point de destination sur le graphe (si le point de destination choisi n'est pas accessible, un nouveau choix aléatoire est réalisé). Dès qu'un joueur est trop proche alors la créature va fuir (le point de destination sera éloigné du joueur). Cependant si le joueur est très proche ou s'il l'attaque, la créature va elle aussi chercher à l'attaquer (ou se défendre). Elle poursuit alors le joueur en évitant tous les obstacles. Une fois à portée de celui-ci, elle va l'attaquer. On peut noter que cette implémentation va permettre de varier les comportements en fonction des créatures en utilisant un seul script d'IA. Par exemple, les sangliers vont pouvoir attaquer sans fuir dès qu'un joueur approche alors que le pampi fuira et n'attaquera le joueur que s'il est agressé. Nous pouvons aussi imaginer des créatures futures qui ne feront que fuir sans jamais attaquer le joueur.

3.3.4 La chasse

Avec l'ajout des créatures, il nous faut aussi le moyen de nous défendre. C'est pour cela que nous avons donné à Ille le moyen d'attaquer. En étant immobile ou en se déplaçant le joueur peut, en utilisant un clic gauche, déclencher l'attaque de Ille. Celui-ci va alors porter un rapide coup en avant avec l'objet dans ses mains ou à main nue si il ne possède pas d'objet. Cependant il faut faire attention, si Ille saute, déclenche un menu ou utilise un objet pendant les deux premiers dixièmes de seconde après le début de son attaque, celle-ci sera annulée. De plus, Ille n'est pas un barbare et cela ne sert à rien de cliquer le plus vite possible avec la souris car Ille ne peut déclencher une attaque que tous les huit dixièmes de seconde même si la précédente attaque a été annulée.

Avec l'ajout de la capacité d'attaque deux nouvelles valeurs sont mises en jeu : l'attaque et l'armure. Ille possède une attaque prédéfinie qui est influencée par l'arme qu'il possède grâce à la formule :

$$DommageDeL'attaque = AttaqueDuJoueur \times CoefficientD'attaqueDeL'arme$$

L'arme du joueur a donc un grand impact sur les dégâts qu'il inflige.

Cependant les dommages d'une attaque ne sont pas directement de la vie perdue par la créature ou le joueur attaqué car celui-ci peut utiliser son armure pour se défendre. La vie

perdue suit alors la formule :

$$ViePerdue = DommageDeL'attaque \times \frac{1}{1 + Armure * CoefficientD'armure}$$

Lorsqu'il s'agit d'une créature, son *CoefficientD'attaqueDel'arme* est égal à 1 et son *Armure* est égale à 0. Pour rendre des créatures plus résistantes que d'autres, nous augmentons leurs points de vie.

3.4 système multijoueur

3.4.1 plateforme multijoueur

Le mode multi-joueurs est une partie importante du projet car celle-ci permettra la coopération des joueurs pour progresser dans le jeu. Nous avons dans un premier temps utilisé les fonctionnalités de *Unet* pour pouvoir créer un serveur (ou un host dans notre cas), mais aussi pouvoir rejoindre un serveur grâce à son adresse IP.

Nous avons dans un second temps synchronisé la position, la rotation et les actions des joueurs. Pour cela on a d'abord utilisé le composant de unity *NetworkTransform*, mais celui-ci ne permettait pas la synchronisation fluide des personnages. Nous avons donc codé nous-mêmes un script permettant cette synchronisation. Son principe est d'envoyer au serveur la position et la rotation du personnage dès que le joueur les modifie, puis celui-ci envoie ces mêmes informations à tous les autres clients. De plus pour rendre les déplacements le plus fluide possible nous utilisons le principe d'interpolation. Cela consiste à déduire les positions intermédiaires d'un joueur à partir de son ancienne et de sa nouvelle position.

Pour finir nous avons synchronisé les éléments, les sons et les caractéristiques du monde, que ce soient les îles, les biomes, les bruits de pas des joueurs, l'état de la journée (jour/nuit) ou bien les arbres.

3.4.2 interface de communication

Comme déjà précisé, la coopération est un élément important du jeu. Pour cela la communication est primordiale. Nous avons donc ajouté un chat qui permet aux joueurs de s'envoyer des messages sans passer par un logiciel tiers.

Nous pouvons noter qu'il est utile de connaître la liste des joueurs connectés. Nous avons donc donné la possibilité aux joueurs d'afficher la liste des joueurs connectés en appuyant sur la touche *tabulation* de leur clavier, elle apparaît en haut de l'écran (voir la figure 19).



FIGURE 19 – Liste des joueurs

3.4.3 Commandes et système d'opérateur

le « propriétaire » de la partie est un opérateur, cela lui permet de faire des commandes de triche telles que /GIVE. Cependant l'ensemble des joueurs possèdent des commandes utilitaires et de communication.

Disponible pour tous les joueurs :

/help {page (int)} : énumère les commandes avec leurs arguments si la page n'est pas spécifiée alors affiche la première page ;
 /msg <player (string)> <message (string)> : envoie un message privé au joueur dont le nom du personnage est passé en paramètre ;
 /seed : affiche dans le chat le seed du monde actuel ;
 /music <clip (int)> : lance la musique correspondant au clip ;
 /team <message (string)> : envoie un message uniquement lisible par les coéquipiers du joueur.

Disponible uniquement pour les opérateurs :

/time <value (int)> : détermine l'heure de la journée ;
 /give {player (string)} <id (int) ou nom (string)> {quantity (int)} : ajoute à l'inventaire d'un joueur (ou nous-même s'il n'est pas spécifié) la quantité indiquée (ou un seul s'il n'est pas spécifié) de l'objet dont l'identifiant correspond au paramètre ;
 /tp <player (string)> : nous téléporte vers le personnage souhaité ;
 /kick <player (string)> : fait sortir du serveur le joueur dont le personnage est passé en paramètre ;
 /save : force la sauvegarde du monde ;
 /kill <player (string)> : tue le personnage passé en paramètre ;
 /effect <player (string)> <id (int)> {power (int)} : affecte le joueur passé en paramètre de l'effet correspondant à l'identifiant avec une puissance choisie (la puissance est d'un si elle n'est pas spécifiée) ;
 /op <player (string)> : donne les droits d'opérateur au joueur passé en paramètre ;
 /deop <player (string)> : supprime les droits d'opérateur au joueur passé en paramètre ;
 /choseteam <id (int) ou nom (string)> {player (string)} : attribue à une équipe choisie le joueur passé en paramètre (nous s'il n'est pas spécifié) .

3.4.4 Les modes de jeu multijoueur

4 La communication

4.1 Phase de test alpha et bêta

4.1.1 Alpha (Florian)

La dernière étape avant la soutenance était de partager notre première version, *v1.0-alpha*. Ce partage nous a permis d'avoir des retours de bugs et quelques avis. Les bugs importants rapportés étaient :

- Le respawn du joueur qui parfois ne fonctionnait pas ;
- Un bug graphique au niveau de l'élément proche lorsque la caméra était à l'intérieur ;
- La musique qui en multi-joueurs pouvait se superposer avec la musique des autres joueurs alentour.

Nous avons corrigé la totalité des bugs rapportés pour produire la version *v1.1-alpha* puis grâce aux avis, nous avons ajouté et modifié quelques détails artistiques. Ces modifications ont abouti à la version *v.1.2-alpha* qui est la version présentée lors de la première soutenance.

4.1.2 Bêta (Florian)

La dernière étape avant la soutenance était de partager notre version actuelle, *v1.0-beta*. Ce partage nous a permis d'avoir des retours de bugs et quelques avis. Les bugs importants rapportés étaient :

- L'animation d'attaque par le joueur ;
- L'interaction avec des petits éléments ;
- Un fort déséquilibrage du jeu, celui-ci était beaucoup trop dur.

Nous avons corrigé la totalité des bugs rapportés, puis grâce aux avis, nous avons ajouté et modifié quelques détails artistiques et de gameplay. Ces modifications ont abouti à la version *v.1.1-beta* qui est la version présentée lors de la seconde soutenance.

4.2 Le site web (Théo)

Le site web que nous avons créé (accessible à l'adresse <http://jmounier.github.io/Aegina-Website/>) avait pour notre équipe plusieurs objectifs à remplir :

- nous devions respecter l'obligation qui était de construire un site web ;
- créer un lieu où nos testeurs et nos, nous espérons, futurs joueurs puissent trouver toutes les informations nécessaires afin de rendre leur expérience en jeu agréable ;
- permettre à ces personnes de télécharger notre jeu mais aussi de nous transmettre leurs expériences.

Tout ceci est réalisable depuis les différentes pages présentes sur le site, que nous allons maintenant vous présenter.

4.2.1 la page d'accueil

Sur cette page (voir figure 20) vous pourrez apercevoir, au premier plan plusieurs éléments :

- un bref descriptif du jeu ;
- des accès pour télécharger la version béta de Aegina en fonction de votre système d'exploitation, Windows, Mac et GNU/Linux ; -
- la dernière actualité ajoutée sur le site.

Vous pouvez aussi voir sur le coté un menu, il permet de naviguer entre les différentes pages du site web, il est positionné au même endroit sur toutes les pages.

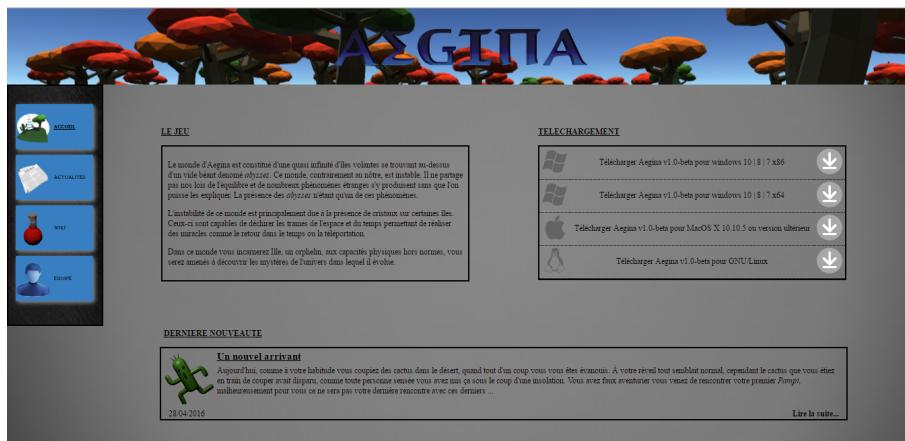


FIGURE 20 – Page d'accueil du site web

4.2.2 Les actualités

Sur la pages des actualités (voir figure 21), que nous tiendrons à jour vous pourrez trouver les différents ajouts qu'il y aura dans le jeu.

Les actualités pourront concerter l'ajout d'un nouveau biome, de monstres ou encore le lancement d'une nouvelle version disponible au grand public.

Nous espérons que les différentes annonces qui seront faites sur cette page rempliront les attentes que vous avez envers Aegina.



FIGURE 21 – Actualités du site web

4.2.3 Le wiki

La page wiki (voir figure 22) est pour notre équipe une page très importante, nous connaissons Aegina et donc nous connaissons la difficulté de rester en vie dans ce monde hostile même pour des personnes aguerries, c'est pourquoi nous avons décidé de mettre en place cette page.

Dans ce wiki vous pourrez trouver toutes sortes d'informations, toutes plus utiles les unes que les autres pour survivre.

Vous y trouverez les différents objets que vous pouvez posséder dans le jeu ainsi que leurs possibles effets, les éléments et monstres qui existent dans Aegina.

Le wiki est actuellement en quatre parties : la liste des objets, la liste des éléments que l'on trouve sur les îles, la liste des créatures et la liste des effets. On peut passer d'une partie à l'autre en cliquant sur le nom de la partie en haut de la page.

Il faut cependant savoir que ce wiki est rempli au fur et à mesure de notre développement, il peut donc manquer certaine informations.

Ressources			
Nom	Icône	ID	Comment l'obtenir ?
Bûche		0	Récolte
Pierre		1	Récolte
Sable		2	Récolte
Cuivre		3	Récolte
Fer		4	Récolte

FIGURE 22 – Wiki du site web

4.2.4 L'équipe

Sur la dernière page de notre site web vous pourrez trouver diverses informations sur les membres de l'équipe AIM², ces informations présenterons les membres de l'équipe et leur rôle dans la création du jeux Aegina.

Conclusion

Table des matières

Introduction	1
1 Univers du jeu	2
1.1 Le monde d'Aegina	2
1.1.1 Primtemps : forêt	2
1.1.2 Automne : forêt automnale	3
1.1.3 Hiver : neige	3
1.1.4 Eté : désert	4
1.2 Les minéraux	5
1.2.1 Le cristal	5
1.2.2 Mithril	5
1.2.3 Floatium	6
1.2.4 Sunkium	6
1.3 La Faune	8
1.3.1 Ille	8
1.3.2 Les créatures agressives	9
1.3.3 Les créatures pacifiques	10
1.4 l'histoire dans le jeu	11
1.4.1 Le tutoriel (Théo et Romain)	11
1.4.2 Le système de Succès (Julien)	12
1.4.3 Le scénario	14
1.4.4 Le boss Final	14
2 L'environnement graphique et sonore	15
2.1 Interfaces 2D	15
2.2 Modèles 3D	16
2.3 Animations	17
2.4 Musiques et sons	18
3 Systèmes de jeu	19
3.1 Systèmes fondamentaux	19
3.1.1 Déplacement du personnage et de sa caméra (Théo et Julien)	19
3.1.2 Génération aléatoire du monde (Florian et Julien)	19
3.1.3 La sauvegarde (Florian)	20
3.1.4 Menus (Romain et Florian)	21

3.2	Le système d'inventaire	25
3.2.1	Les différents types d'objets	25
3.2.2	Inventaire et stockage d'objets	26
3.2.3	La création d'objets (Romain)	28
3.2.4	Drop d'Item (Florian et Julien)	29
3.3	Système de survie	30
3.3.1	Cristal (Romain et Julien)	30
3.3.2	La survie du joueur dans l'environnement	31
3.3.3	Les Créatures (Florian)	35
3.3.4	La chasse	37
3.4	système multijoueur	39
3.4.1	plateforme multijoueur	39
3.4.2	interface de communication	39
3.4.3	Commandes et système d'opérateur	40
3.4.4	Les modes de jeu multijoueur	40
4	La communication	41
4.1	Phase de test alpha et bêta	41
4.1.1	Alpha (Florian)	42
4.1.2	Bêta (Florian)	42
4.2	Le site web (Théo)	43
4.2.1	la page d'accueil	43
4.2.2	Les actualités	44
4.2.3	Le wiki	44
4.2.4	L'équipe	45
Conclusion		45