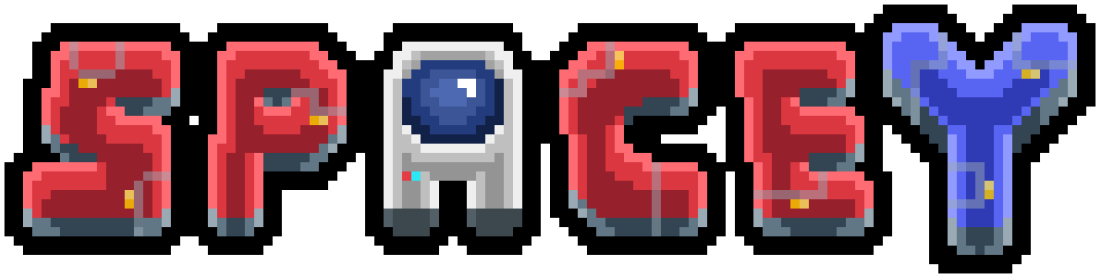
SpaceY

Rapport de Soutenance n*1



By Epit'chad

Maurin AUDENARD (Chef de Projet)
Maxandre BERSON-LEFUEL
Madin AMRANE
Théodore BECHETOILLE
Olaf PINIARSKI

spacey-epitchad.github.io

Cadre : Projet de 1ère année EPITA

Table des matières

Aspect visuel et graphique du jeu

Dans la majorité des jeux vidéos, l'aspect visuel possède une place très importante, donnant aux joueurs des indications sur quoi faire, comment le faire, ou encore lui permettant de facilement comprendre ce qui a lieu pendant la partie. C'est donc naturellement que nous lui avons également accordé une importance capitale dans notre jeu. Cette partie se concentrera donc sur les différents aspects graphiques de notre jeu et des procédés utilisés.

1.1 Les équipes

L'aspect visuel des différentes équipes a été un des premiers points sur lesquels nous avons travaillé. Nous avons alors opté pour le choix des couleurs bleus et rouges, très fréquemment utilisées pour symboliser une rivalité entre 2 équipes car dans l'imaginaire commun, ces 2 couleurs symbolisent souvent une opposition (par exemple en politique, où le rouge est associé à la gauche et le bleu à la droite). Nous la retrouvons alors dans plusieurs média tels que le cinéma (par exemple Captain America : Civil War opposant Captain America en bleu à Iron man en rouge), mais surtout dans les jeux vidéo servant à représenter 2 équipes qui s'affrontent (Utilisé dans Super Smash Bros et Clash Royale par exemple)

Nous remarquons directement les différences de couleurs entre les personnages d'une équipe, étant donné la nature multijoueur de notre jeu, il est important pour les joueurs de pouvoir facilement identifier les personnages de sa propre équipe et de l'équipe adverse. Il en est de même pour les couleurs des fusées, on ne voudrait pas que les joueurs confondent leurs fusées avec celle de l'adversaire, mettre des fusées de même couleur que les joueurs est donc une solution qui nous est venue naturellement.

1.2 Déplacement

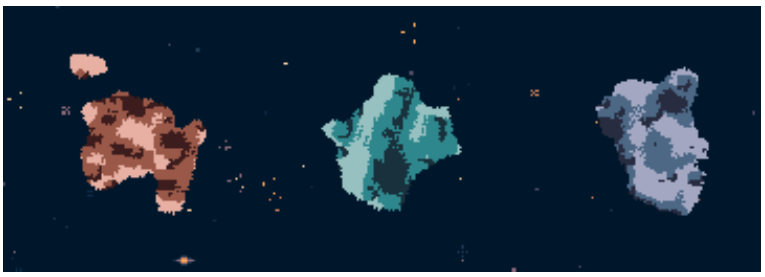
Concernant le déplacement nous devions rendre le déplacement des personnages cohérent et dynamique, pour ne pas donner l'impression que le jeu n'est pas fini. Cependant, étant dans l'espace, nous ne pouvions pas utiliser des simples pas. Nous avons alors opté pour un propulseur à l'arrière du personnage dont l'opacité augmente puis

diminue pour rendre l'animation plus dynamique. Nous avons également tourné le personnage vers la direction vers laquelle il avance pour rester cohérent. La gestion de l'opacité a été grâce à la propriété "Modulate" des objets de Godot, quant à la direction nous avons utilisé la "Rotation" des noeuds de Godot pour la contrôler

1.3 Astéroïdes

Les astéroïdes sont des éléments cruciaux du jeu, et il était important de permettre au joueur de différencier entre les différents types d'astéroïdes. Nous avons été capable de les générer grâce à un logiciel spécialisé "PixelPlanetWindows", ce qui nous a permis de créer des astéroïdes de différentes tailles et différentes formes. Les différentes forment et la rotation des astéroïdes servent à rendre l'environnement du jeu plus vivant et ne pas simplement donner l'impression qu'ils ont été clonés. Concernant leurs contenus nous avons donc décidé de les différencier par leurs couleurs :

1. Les astéroïdes gris sont des astéroïdes qui ne possèdent pas de ressource, c'est la couleur des astéroïdes dans l'imaginaire collectif.
2. Les astéroïdes oranges rougeâtres représentent les astéroïdes possédant du fer, il s'agit de la couleur du minerai de fer avant d'être traité.
3. Les astéroïdes bleus représentent les astéroïdes possédant des cristaux, il s'agit de la couleur du cristal dans l'imaginaire collectif et avec laquelle il est le plus souvent représenté dans les jeux vidéo.



1.4 Aliens

Le jeu représente des humains dans l'espace, nous nous sommes donc dit que les ennemis les plus intuitifs qu'ils pourraient avoir serait une forme de vie extraterrestre hostile. Nous avons donc simplement pris des aliens en soucoupe volante, un cliché assez répandu dans la culture populaire, et qui donc parlerait à tout le monde, permettant de facilement se rendre compte de la menace qu'ils représentent pour l'avancée du joueur

1.5 Conclusion

En ce qui concerne l'avancement par rapport au cahier des charges, nous avions prévu d'avoir trouver 100% des ressources graphiques. Nous avons effectivement trouvé toutes les ressources nécessaires même si elles n'ont pas toutes été implémentées car nous les implémentons au fur et à mesure de l'avancement du développement du jeu.

Génération de la map

Dans SpaceY la map est la pierre angulaire du jeu. C'est d'ailleurs pour cela qu'on a décidé de la faire en tout premier.
En effet, la faire plus tard ralentirai le développement car comme tout prend vie sur la carte nous, sans nous n'aurions pas pu faire toutes les intéracions entre les différentes parties du jeu.

Pour l'instant sur la map est constituée des trois éléments suivant :

1. Le départ et l'arrivée.
2. Les limites de la map, constituées par des "colliders".
3. les astéroïdes qui permettent la récolte des ressources ainsi que les mouvements de la fusée.

2.1 Composition basique

Le départ se positionne tout à gauche de la map, et l'arrivée est quant à elle positionnée tout à droite, au même endroit que l'objectif, la Lune.
Nous avons choisi ce placement, car c'est pour nous le plus naturel car il suit le sens du temps qui est très souvent représenté de gauche à droite. Ainsi, les joueurs ont l'impression d'avancer à la fois dans leur objectif et dans le temps qui est une ressource précieuse dans une course.

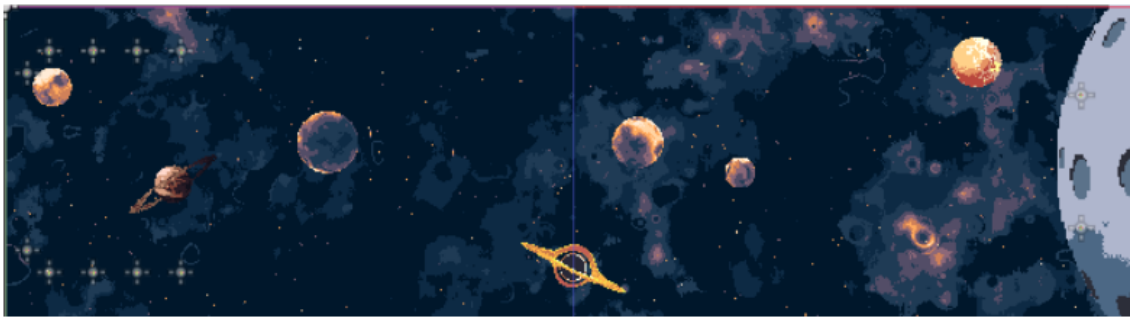


FIGURE 2.1 – Départ au début de la partie

Afin que rien ne sorte de la map, on a installé des colliders, des limites, qui bloquent les joueurs et les aliens trop aventureux.
Ces limites physiques sont constituées d'objets fait pour, dans godot, ce sont les StaticBody2D. Les StaticBody2D sont des corps, des objets physiques qui sont immuables pour les autres composantes du jeu. Ainsi, les joueurs ne pourront pas se perdre et les aliens eux ne pourront pas s'enfuir.

Pour que les lasers ne consomment pas trop de stockage nous avons décidé de mettre en place un système afin de les faire disparaître dès lors qu'ils ne sont plus utiles.
Pour ce faire, nous avons mis en place une Area2D qui s'étend sur toute la zone jouable, ainsi lorsque les lasers sortent de cette zone, ils s'autodétruisent. Bien que ces lasers ne réservent pas beaucoup de stockage dans la mémoire, nous tenions à les gérer proprement pour que le jeu soit le moins gourmand en ressources possible.

Voici un shéma représentant la disposition de ces limites

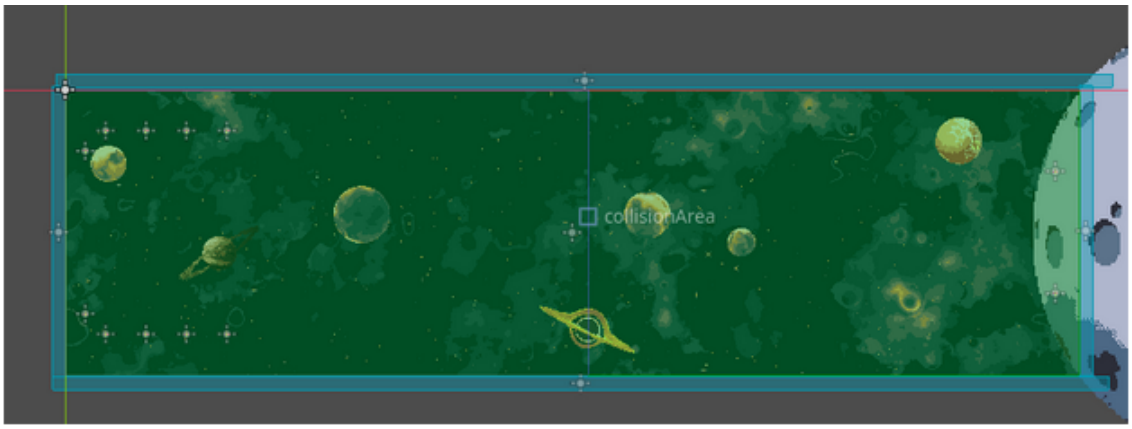


FIGURE 2.2 – Départ au début de la partie
En bleu : colliders, En vert : Area2D.

Le réceptacle des astéroïdes est l'un des éléments de la map qui a été la plus complexe à choisir.
En effet, nous avons dû nous y reprendre à plusieurs fois, tout d'abord, nous voulions constituer un Array remplie de type astéroïdes, ce qui à été fait.
Cependant après s'être renseigné sur les algorithmes de pathfinding destinés aux IA du jeu, nous nous sommes rendu compte qu'il était plus simple d'utiliser un objet godot : une "TileMap". Les TileMap présentent beaucoup d'avantages par rapport aux Array pour le cas de notre utilisation.

En effet, elle est facile à mettre en place, facile d'accès, a une taille infinie et des réglages faciles à modifier.
Cependant elle présente aussi des désavantages : les éléments de la TileMap sont forcément sur la grille alors que dans le jeu, nous voudrions que les astéroïdes soient, certes disposés en grille, mais pas parfaitement afin d'avoir un sentiment plus naturel face aux placements des astéroïdes.
Aussi les actions possibles sur les éléments des TileMap ne sont pas aussi simple que celle des Array, c'est aussi le cas du parcours des TileMap, qui est très restreint cela a pour effet de complexifier l'accès des éléments de la TileMap ce qui peut être handicapant lors du développement.

2.2 Génération des astéroïdes

La génération des astéroïdes est complétement aléatoire, cela lui permet d'être simple et complètement impartiale ce qui nous paraissait important pour un jeu de course comme SpaceY qui se concentre sur l'accessibilité.
À fin de faire des ajustements et pour s'assurer que toutes les maps sont jouables nous avons mis plusieurs curseurs que nous pouvons ajuster à notre convenance, ils sont au nombre de trois et chacun contrôle respectivement une facette de la génération de la map :

1. Le pourcentage de remplissement de la map.
2. Le pourcentage d'apparition des astéroïdes ferreux.
3. Le pourcentage d'apparition des astéroïdes contenant du cristal.

2.3 À venir ?

Cependant, nous voudrions aussi faire d'autres modes de génération : une génération dites personnalisable. Cette génération consisterait en une panoplie de choix qui influencerait la disposition des astéroïdes. Par exemple, les astéroïdes suivraient ce paterne :

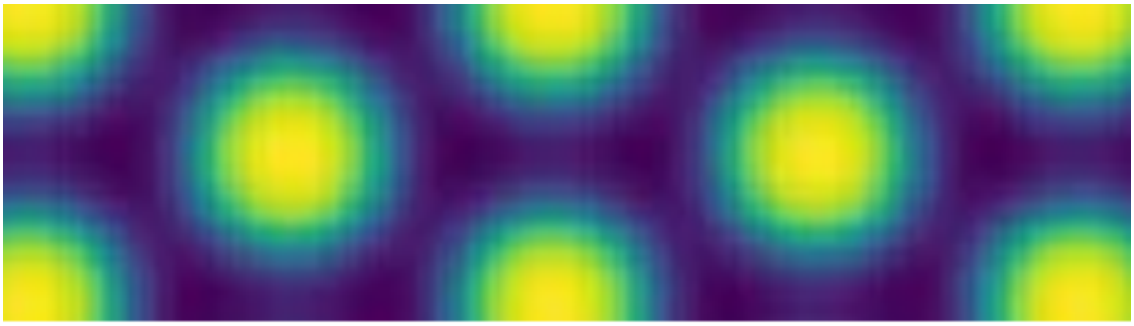


FIGURE 2.3 – Graphe voulu de la répartition d'astéroïdes.
zones sombres : zones d'astéroïdes | zone claires :zones vides.

Dans ce paterne les zones sombres seraient les zones peuplées d'astéroïdes et celles en clair le seraient moins voir pas du tout.
C'est avec cet objectif que nous avons cherché des fonctions mathématiques qui pourraient nous aider. Après des recherches et beaucoup d'expérimentation, nous avons trouvé cette fonction :

$$g(x,y) = 0.2(\sin(\tfrac{\pi}{\sqrt{3}}(x+y+1.8)) + \cos(\tfrac{\pi}{\sqrt{3}}(x-y)) + \sin(\tfrac{2\pi}{\sqrt{3}}y)) + 0.1$$

FIGURE 2.4 – Fonction mathématique trouvée.

Ce qui donne, sur la calculatrice de GeoGebra, ce graphe :

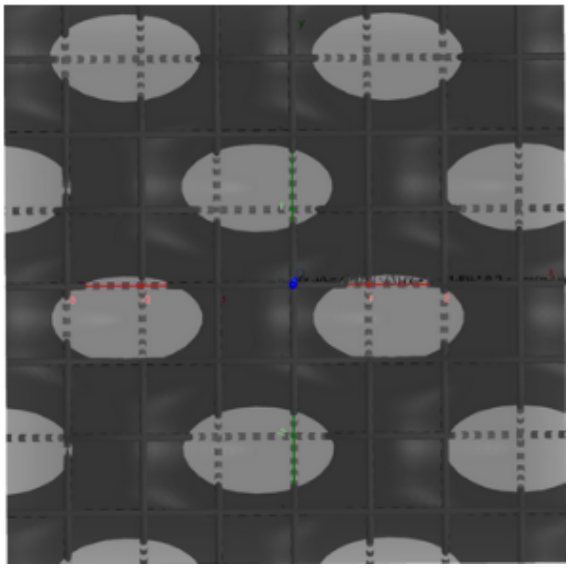


FIGURE 2.5 – Graphe obtenu vu de haut.

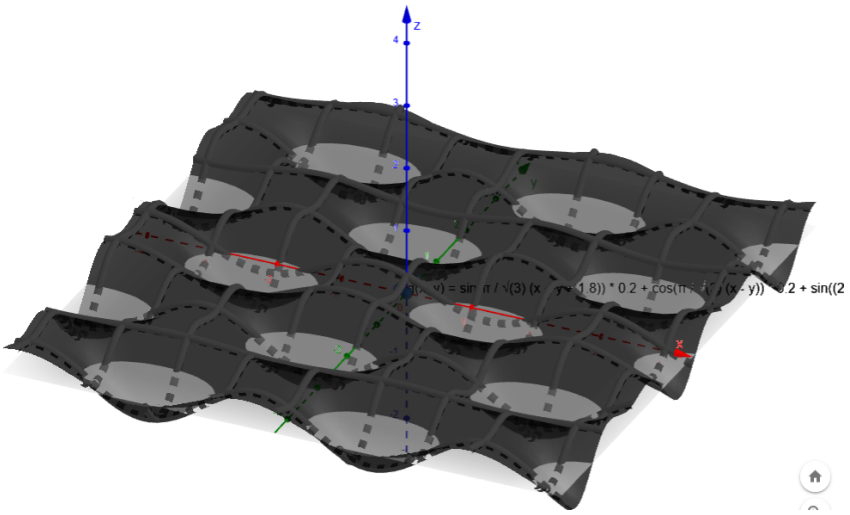


FIGURE 2.6 – Graphe obtenu vu de diagonale.

Comme vous pouvez le voir, la fonction mathématiques est trouvée, cependant il faudrait en trouver plus, avec des paternes différents afin d'apporter un vrai choix de maps aux joueurs. Mais pour l'instant ce n'est que le début de cette fonctionnalité, elle n'est pas implémenter et il faut encore faire des recherches avant de commencer son potentiel développement.

En ce qui concerne l'avancement de la map, nous avions prévu d'avoir fini 90% de la map, et nous avons bien estimé l'avancement, car pour l'instant elle est complétement finie, il ne reste qu'à voir si la génération personnalisée est possible et faire quelques ajustement sur l'apparition des astéroïdes.

Astéroide

Dans cette partie nous nous intéresserons aux astéroïdes et leurs importants rôles dans une partie de notre jeu tel que celui dans l'avancement de la fusée, ou bien de sa défense .

3.1 La génération des ressources

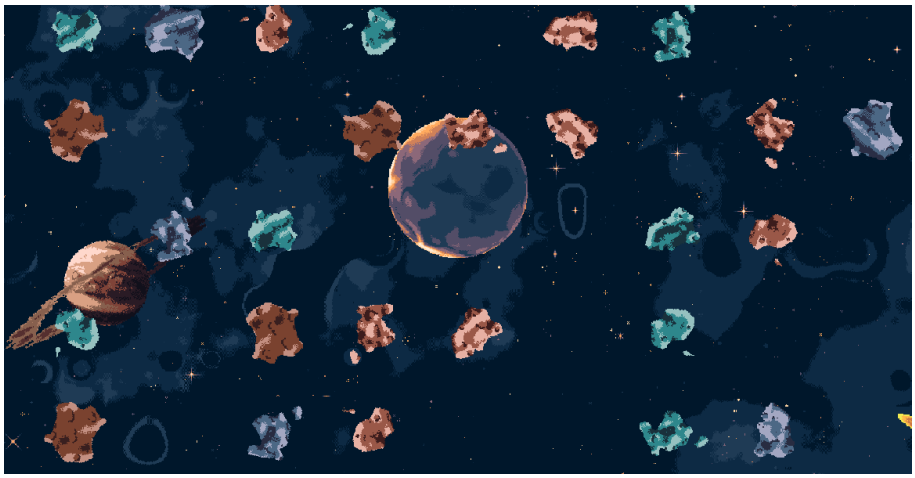
La génération des astéroïdes se fait en grande partie lors de la génération de la carte, s'est à ce moment là qu'il est instancié. A sa création, deux valeurs aléatoires entre 0 et 100 lui sont associées qui déterminent sa forme et sa nature (les ressources qu'il contient). La variété de forme et de texture a pour but de rendre le jeu plus dynamique et ne pas donner d'impression de clonage, quant aux ressources il s'agit d'un aspect dont la variété est primordiale dans le bon fonctionnement d'une partie afin de ne pas rendre les parties répétitives.

3.2 Les ressources

Comme dit précédemment, une valeur aléatoire entre 0 et 100 décide des ressources que possèdera un astéroïde :

1. Entre 0 et 29 : L'astéroïde ne possède aucune ressource (gris).
2. Entre 30 et 59 : L'astéroïde possède du fer (Orange rougeätre).
3. Entre 60 et 100 : L'astéroïde possède du fer et du cristal (bleu)

Mais à quoi servent donc ces ressources ? Et bien il s'agit de la clef de la victoire d'une équipe, le fer sert à créer des stations de passages qui permettent de faire avancer les fusée, il s'agit du seul moyen pour faire avancer cette dernière, donc sans fer il est impossible de gagner et c'est pour cela que c'est une ressource abondante. Le cristal lui permet de lancer un laser destructeur lorsque le joueur est positionné autour de sa fusée, mais nous rentrerons dans les détails de ce que peut faire le laser plus tard dans une partie dédiée.



3.3 Station de passage

La relation entre les astéroïdes et les stations de passage ne s'arrête pas aux matériaux, en effet nous sommes dans l'espace, et les seules plateformes à dispositions sont... les astéroïdes ! Une fois que le joueur a récupéré suffisamment de fer pour construire une station, se positionner près d'un astéroïde et être dans une certaine zone (une " Area-Body2D"), maintenir une touche qui lancera la construction d'une station. Pour se faire nous avons utilisé un chronomètre (un "Timer" sur Godot), et une barre de chargement qui se remplit au fur et à mesure que le chronomètre avance pour représenter l'avancement de la construction. Une fois que la barre de chargement est remplie, la station est construite.

FIGURE 3.1 – La barre de chargement

FIGURE 3.2 – Une station créée sur un asteroïde

3.4 Conclusion

En ce qui concerne les astéroïdes, nous avions prévu d'avoir fini 80% des astéroïdes, et nous sommes assez proches de nos estimations. Nous avons fini la majorité de ce que nous avons à faire pour le jeu final, la seule interaction qui n'a pas encore implémentée pour finaliser le développement des astéroïdes est la récolte de ressource, qui représente la future grande étape dans le développement du jeu

Fusée

Dans une course de fusées, il faut des fusées !
Et des fusées contrôlables de préférence…

Dans SpaceY, les fusées apparaissent sur le départ qui se situe toute à droite. Chaque équipe à sa propre fusée ainsi l'appartenance de la fusée à une équipe est signifiée par sa couleur.
Les fusées sont très espacées pour que les équipes ne se rencontrent pas tout de suite, cela permet de prendre en main les touches, et de s'organiser lors du début de la partie. L'organisation étant très importante dans SpaceY, nous devions permettre de la mettre en place en laissant du temps aux joueurs. En jeu, le début de la partie ressemble à ça :

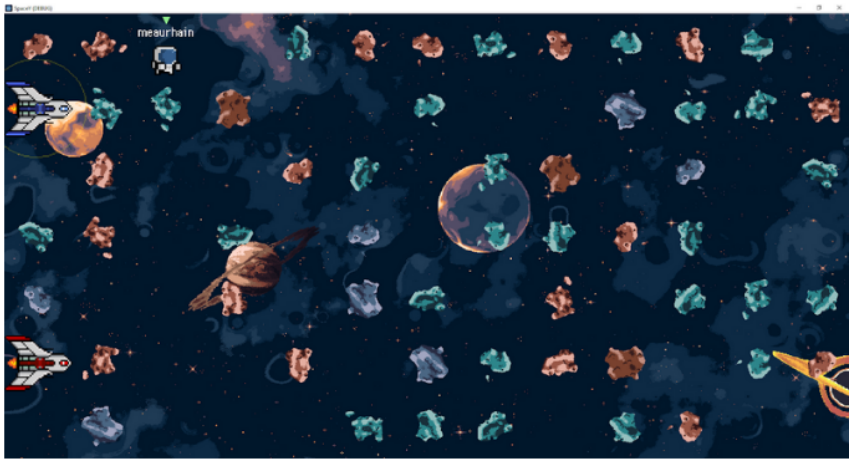


FIGURE 4.1 – Départ au début de la partie

Une fusée c'est bien, mais c'est encore mieux quand elle avance, surtout dans une course. Afin de faire avancer la fusée, les joueurs peuvent poser des "Waypoints" (aussi appelés des stations de passage) sur n'importe quels astéroïdes libres, qui construisent le futur itinéraire de la fusée.
Cependant, afin que la fusée se dirige vers un Waypoint, ce dernier ne doit pas être posé très loin. En effet, la fusée détecte les Waypoints qui se trouvent dans un certain périmètre autour d'elle. Cela oblige les joueurs à s'organiser pour aller chercher des ressources qui permettent de poser les Waypoints aux bons endroits, afin de faire le meilleur itinéraire.

Afin que les joueurs puissent se représenter la porté de la fusée, nous avons mis en place des pointeurs sur les astéroïdes atteignables. Les pointeurs d'une fusée ne sont visible que pour les joueurs possédant la fusée.

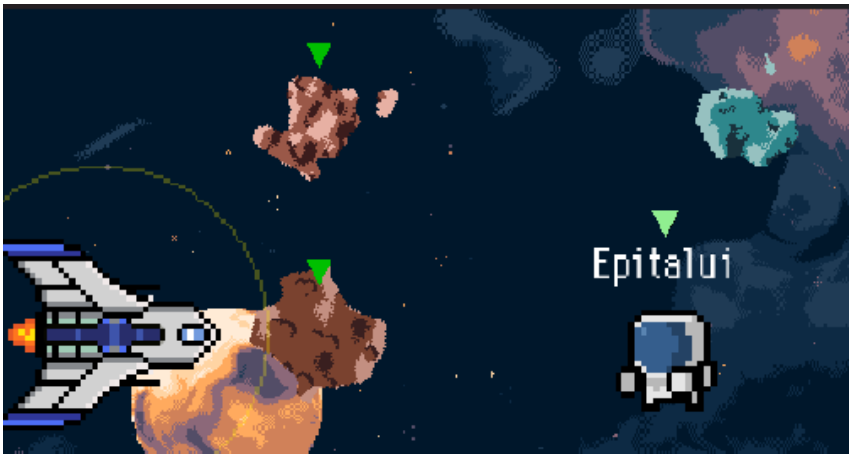


FIGURE 4.2 – Visuel des pointeurs

4.1 Détection des Astéroïdes

Pour ce faire nous avons mis en place une Area2D centrée sur la fusée en permanence et une autre qui est centrée sur le Waypoint. La première est la zone de détection de la fusée et la seconde est la zone d'accessibilité au Waypoint Ainsi un Waypoint est considéré comme atteignable par la fusée lorsque ces deux zones se chevauchent.

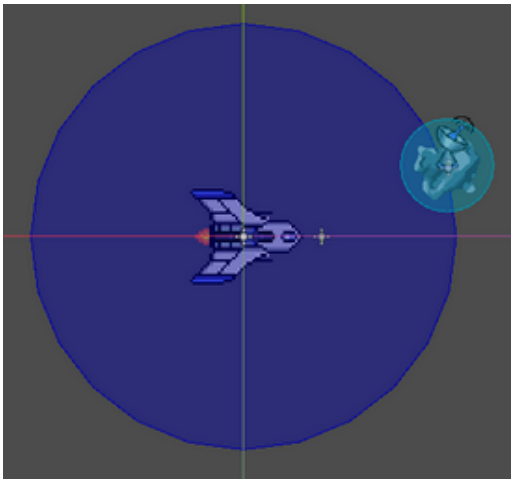


FIGURE 4.3 – Astéroïde invisible pour la fusée

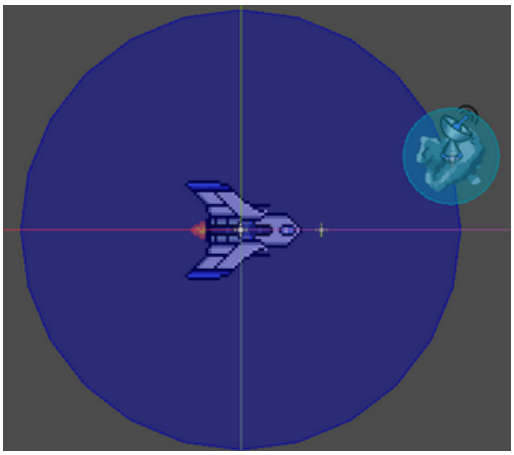


FIGURE 4.4 – Astéroïde dans le champ de détection

Le fonctionnement derrière est très simple :

1. Lorsque deux Area2D entre en contacte la zone de la fusée émet un signal :

   ```
   _on_checkpoint_area_entered()
   ```

2. Ce signal déclenche une fonction de la fusée qui prend en paramètre l'Area2D qui vient d'entrer (area) :

```
public void _on_checkpoint_area_entered(Area2D area) {
    if (area.IsInGroup($"Rocket{equipNumber}")
        && !area.IsInGroup($"AlreadyUsedStation"))
    {
        area.AddToGroup($"AlreadyUsedStation");
        if (!gotTarget)
        {
            _navigationAgent.TargetPosition = area.GlobalPosition;
            LookAt(area.GlobalPosition);
            gotTarget = true;
        }
        else
            checkpoints.Add(area);
    }
}
```

4.2 Avancement

Pour faire avancer nous utilisons un fonctionnalité itinérante à godot : les “NavigationAgent”. Ces nœuds permettent de récupérer simplement le vecteur à appliquer afin de se rapprocher de la cible. Ainsi, nous avons juste à appliquer ce vecteur sur la fusée pour faire son déplacement. Afin qu’on n’ait pas l’impression que la fusée glisse vers la cible nous avons fait en sorte qu’elle pointe toujours vers sa cible, comme le ferait une vraie fusée ! Pour le faire, nous utilisons la fonction "LookAt()".

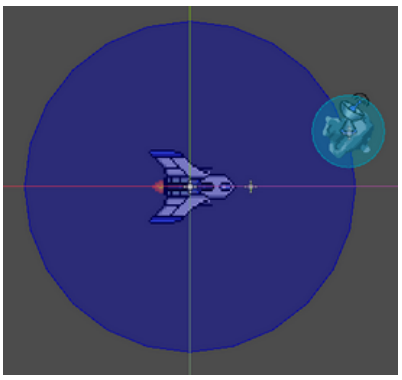


FIGURE 4.5 – Sans LookAt()

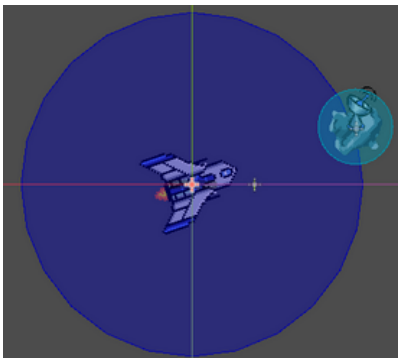


FIGURE 4.6 – Avec LookAt()

4.3 À venir.

Pour l’instant les fusées sont fonctionnelles, nous avions prévus que les fusée soient fini à 90%, ce qui est respecté car il ne manque que les intéractions avec les ressources. En effet, comme les ressources ne sont pas encore implémentées dans le jeu les fusées seront modifiées quoi qu’il arrivent. Que ce soit pour "manufacturer" les ressources comme les consommer, il faudra adapter les fusées.

Personnage jouable

Dans cette partie nous nous attarderons sur les actions que sont capables d'effectuer les joueurs au travers de leurs personnages afin de faire avancer leur fusée et d'entraver l'avancement de leur adversaire.

5.1 Caméra

La caméra est un élément important dans le jeu. En effet, chaque joueur à sa propre caméra, que l'on pourrait appeler son champ de vision. L'implémentation de la caméra en elle même est très simple, il s'agit simplement d'ajouter le noeud "caméra" présent sur Godot. Mais 3 éléments sont notables. Dans un premier temps, il a fallu synchroniser la position de la caméra afin de faire en sorte que le joueur reste au centre, ensuite, le deuxième point est de faire en sorte que cette caméra reste dans le champ de la map. Pour ce faire, nous avons précisé des positions à ne pas dépasser dans les propriétés de l'élément caméra. Et le troisième point concerne le multijoueur, il a fallu faire correspondre une caméra par joueur. Afin de pallier ce problème, nous avons développé un algorithme permettant de récupérer l'identifiant de l'utilisateur actuel afin de lui associer sa propre caméra.

```
if (multiSync.GetMultiplayerAuthority() == Multiplayer.GetUniqueId())
{
    camera.MakeCurrent();
    indicator.Show();
}
```

5.2 Récolte de ressources

Comme évoqué plus haut, et comme nous le détaillerons plus tard, pour faire avancer la fusée, nous devons poser des stations de passages pour faire avancer notre fusée, ils auront également la possibilité de tirer des laser de leurs fusée. Mais ces deux actions ne se feront pas sans coût et nécessiteront de récupérer des ressources sur les différents astéroïdes.

Comme expliqué plus tôt, les astéroïdes possèdent chacun une certaine quantité d'une certaine ressource. Celles-ci, lorsque leurs valeurs n'est pas nulle, pourront être récoltées par les joueurs puis amenées à la fusée afin d'être " raffinées " dans la fusée (de manière passive), permettant la création de points d'accès et le tir de laser.

Cependant cette mécanique n'a toujours pas été implémentée, elle représente alors notre prochain objectif majeur dans le développement de notre jeu.

5.3 Tirer des lasers

Une des mécaniques principales du jeu est de pouvoir tirer un laser. Il pourra d'une part paralysé les joueurs des 2 équipes, et faire disparaître les aliens qui menacent de faire perdre ses ressources à la fusée.

L'idée derrière ces laser a été de faire en sorte qu'il avance en direction du curseur (représenter par un viseur) cela en utilisant la méthode " GetMouseGlobalPosition" et en

augmentant sa vélocité. Pour éviter qu'il ère indéfiniment dans le jeu en dehors de l'écran nous avant également créé une méthode qui fait en sorte de faire le faire disparaître une fois qu'il sort des bordures de la carte pour éviter d'avoir plusieurs laser en train d'être gérer par la machine ce qui aurait été coûteux en ressource particulièrement pour l'hôte.

En ce qui concerne son effet sur les aliens, le procédé a été assez simple, nous avons utilisé le noeud " AreaBody2D " qui représente la zone dans laquelle l'alien peut se faire toucher par le laser, lorsqu'il se fait toucher, le noeud le représentant est détruit le faisant donc disparaître.

Concernant son effet sur les joueurs, il les paralysera pour une durée de 1,5 secondes, pour se faire nous avons forcé la vélocité du personnage à 0, et annulant son animation pendant la durée de l'effet. Cet effet ne prend pas en compte l'équipe du personnage et s'applique dès qu'il rentre dans la zone associée d'un personnage.

Dans le cahier des charges, nous avions prévu d'avoir fini 20% du laser. Cependant, nous avons finalement avancé beaucoup plus que prévu, nous avons réussi à finir son interaction avec les aliens et une partie de son interaction avec les personnages. Néanmoins, il nous reste encore certaines choses à implémenter. Tout d'abord il faut rendre l'utilisation du laser impossible lorsque le joueur ne possède pas la ressource nécessaire. Ensuite, il nous faudra coder l'effet qu'il aura concernant les ressources lorsqu'il rentrera en contact avec une fusée ou un joueur, mais les ressources n'ayant toujours pas implémentées, nous ne sommes pas en capacité d'implémenter ces interactions à l'heure actuelle, cela sera alors notre prochaine étape dans la conception du jeu et du laser plus particulièrement.

Interface

L'interface graphique est un élément primordial lors du développement d'un jeu, puisqu'il contribue grandement au plaisir visuel renvoyé par le jeu et son image général. En plus de cet aspect, l'interface sert tout d'abord à améliorer le confort du joueur lors du lancement d'une partie et c'est pourquoi l'objectif de notre interface était de la rendre la plus simple possible. Nous avons créé 4 scènes principales pour les menus : le menu principal sur lequel le joueur arrive en premier lorsqu'il lance SpaceY, le menu des paramètres, le menu de connexion à une partie, et enfin, la salle d'attente qui est un menu particulier sur lequel nous reviendrons plus tard dans la partie "Réseau et multijoueur".

6.1 Page d'accueil

La page d'accueil correspondant au menu principal, c'est-à-dire la première image du jeu que les joueurs verront. Elle permet à travers deux boutons différents, un accès aux deux scènes suivantes : la scène de connexion à une partie et la scène des paramètres.

6.2 Paramètres

Le menu des paramètres n'a pas une importance particulière pour SpaceY. Actuellement, il est seulement disponible pour changer son pseudonyme affiché pendant une partie. Nous avons aussi implémenté des réglages pour le son que nous n'avons pas encore codés. En effet, toute la dimension sonore n'a pas encore été réalisée.

6.3 Menu de connexion

Le menu de connexion est un menu simple mais efficace qui donne accès à deux boutons : un bouton “Host” qui permet de créer une partie à laquelle les autres joueurs peuvent se connecter, et le bouton “Join” qui permet de rejoindre une partie. Ce dernier est différent du premier car il requiert un paramètre supplémentaire. Afin de rejoindre une partie, il est demandé de fournir un code correspondant à l’identificateur de la partie. C’est grâce à ce code que le joueur peut se connecter à l’ordinateur de l’hôte de la partie.

6.4 Conclusion

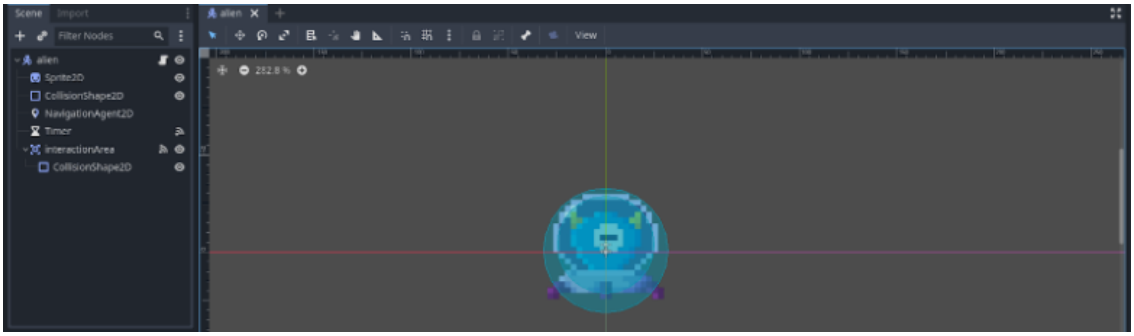
Enfin, le GUI (Graphical User Interface) ou interface utilisateur graphique en français est une partie dont nous sommes fiers. Nous avons pu la terminer comme nous le souhaitons tout en respectant le cahier des charges. Elle n'est pas entièrement terminée car il nous reste le menu des paramètres à finaliser comme expliqué ci-dessus.

IA

Afin d'obtenir une dynamique de jeu agréable, le multijoueur permet aux joueurs de rejoindre leurs amis dans des parties personnalisées. Nous avons rencontré beaucoup de difficultés lors de l'implémentation de cette fonctionnalité, nous y reviendrons prochainement.
L'objectif principal était de garantir une expérience fluide et immersive pour tous les joueurs. Cependant, la synchronisation entre les joueurs s'est révélée être un défi de taille. La coordination des actions simultanées, la transmission des données en temps réel et la gestion des latences ont été des aspects cruciaux à prendre en compte.
De plus, nous avons dû optimiser la gestion des ressources réseau pour minimiser la charge sur l'ordinateur hôte, tout en maintenant une qualité de jeu optimale pour l'ensemble des participants. Pour ce faire, nous avons décidé de réduire le nombre d'échanges entre l'hôte et les clients. Les paquets envoyés sont alors toujours les mêmes, mais le délai d'envoi est plus long. Afin de garder une certaine fluidité dans les mouvements, nous avons utilisé un lerp. Le lerp est une opération mathématique qui permet de calculer une position moyenne sur laquelle afficher la position du joueur. De ce fait, les déplacements sont toujours fluides.

7.1 Implémentation

Le modèle de réseau que nous avons utilisé est appelé peer-to-peer. Très simplement, il s'agit d'un système qui repose sur la connexion à un ordinateur hôte.
Lors de l'initialisation, un utilisateur crée une partie, il sera alors hôte de la partie. Son ordinateur héberge alors la partie et les autres joueurs peuvent la rejoindre via un identifiant de partie généré à partir de l'adresse IP de l'hôte.
Lors de la connexion, hôte ou client, plusieurs signaux sont envoyés. Tout d'abord, une confirmation de la création du "serveur" (ordinateur de l'hôte) et de la connexion à celui-ci. Une fois que les autres joueurs se connectent à l'hôte, ils reçoivent eux aussi une confirmation qui va se caractériser par l'affichage de la salle d'attente ainsi que la transmission des informations du joueur à l'hôte (pseudonyme, équipe du joueur etc...).
Et enfin lors de la déconnexion, des signaux sont envoyés à chacun des joueurs afin de prendre en compte le fait qu'un joueur se soit déconnecté de l'ordinateur de l'hôte.



7.2 Conclusion

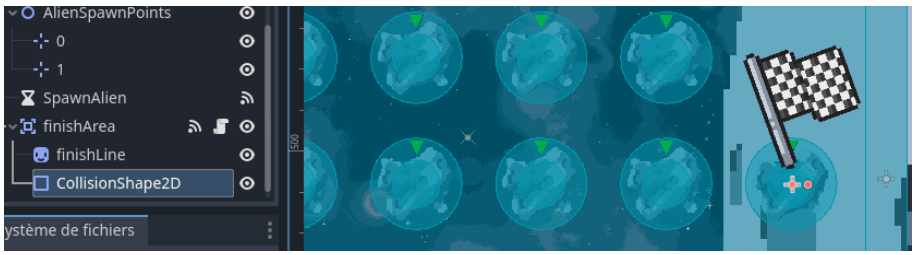
Le peer-to-peer utilisé pour ce jeu fait donc appel à un ordinateur hôte. L'hôte va recevoir toutes les informations nécessaires afin de faire fonctionner la partie, puis les transmet à tous les joueurs connectés. Etant donné que les joueurs ont besoin de l'adresse IP de l'hôte afin de rejoindre la partie, nous avons décidé d'utiliser une méthode de chiffrement pour sécuriser les connexions. L'adresse IP est donc transformée en une suite de chiffres calculés par l'ordinateur.
À chaque connexion, l'hôte reçoit des informations sur les joueurs (et plus précisément l'ordinateur du joueur), tel que le pseudonyme choisi par le joueur, l'équipe dans laquelle il se trouvera ainsi que son adresse IP. Cette dernière information n'est pas visible par l'hôte, mais reste une donnée enregistrée dans son ordinateur lors de la partie. L'hôte est le seul joueur qui puisse partager le code de la partie, ainsi que lancer cette dernière.

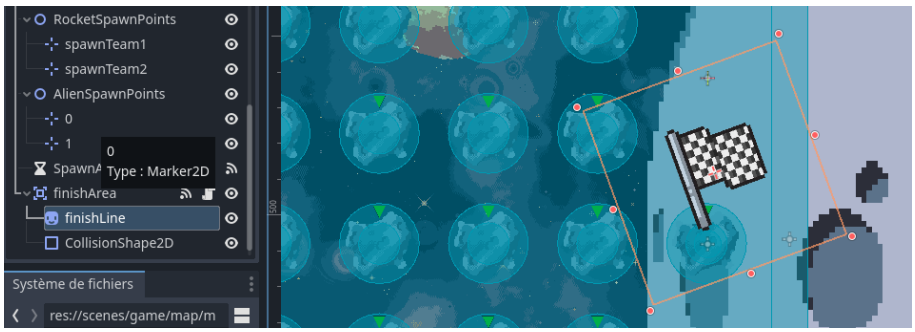
Conditions de Victoire

Les conditions de victoire sont un point clé dans la finition de notre jeu. En effet, dans un jeu tel que le nôtre, où l'on affronte une équipe adverse dans une course, définir un vainqueur à la fin de la partie semble indispensable. Cela permet de clore la partie en indiquant quelle équipe à gagné.

8.1 Conditions

Tout d'abord nous avons défini un sprite, ici un drapeau à damier, pour indiquer aux joueurs où se situe l'arrivée, avec un astéroïde pour pouvoir y accéder. Nous avons ensuite établi les conditions de victoire. Pour cela une zone d'arrivée à été établie. Si un objet rentre en collision avec cette zone, un signal sera émis, qui appellera alors une fonction. Cette fonction vérifiera en premier lieu si l'objet présent est une fusée. Si l'objet n'est une fusée il ne se passera rien, pour éviter que la victoire ne soit obtenue d'une autre manière (par exemple si un joueur rentre dans cette zone, il ne fera pas gagner son équipe, c'est bien sa fusée qui doit y aller). Pour différencier la fusée d'un autre objet, nous avons créé un groupe dans lequel les deux fusées sont présentes. Ainsi, si l'objet est bien une fusée, alors la fonction étudiera à quelle équipe cette fusée appartient. Il ne reste alors plus qu'à afficher un message de fin avec le nom de l'équipe victorieuse.





8.2 Message de Fin

Lorsque les conditions sont remplies et qu'une équipe a gagné, un message de fin est affiché pour informer les joueurs que la partie est finie et quelle équipe a gagné. Nous avons tout d'abord créé un message, affiché directement sur la map, que nous avons ensuite caché pour ne pas qu'il soit visible toute la partie. C'est seulement lorsqu'une équipe passe la ligne d'arrivée que le message est affiché, avec un nouveau texte de victoire. Pour cela nous avons créé une fonction prenant en paramètre la couleur de

l'équipe gagnante puis renvoyant le nouveau message sur l'écran en utilisant le nom de l'équipe.

8.3 Retour Menu

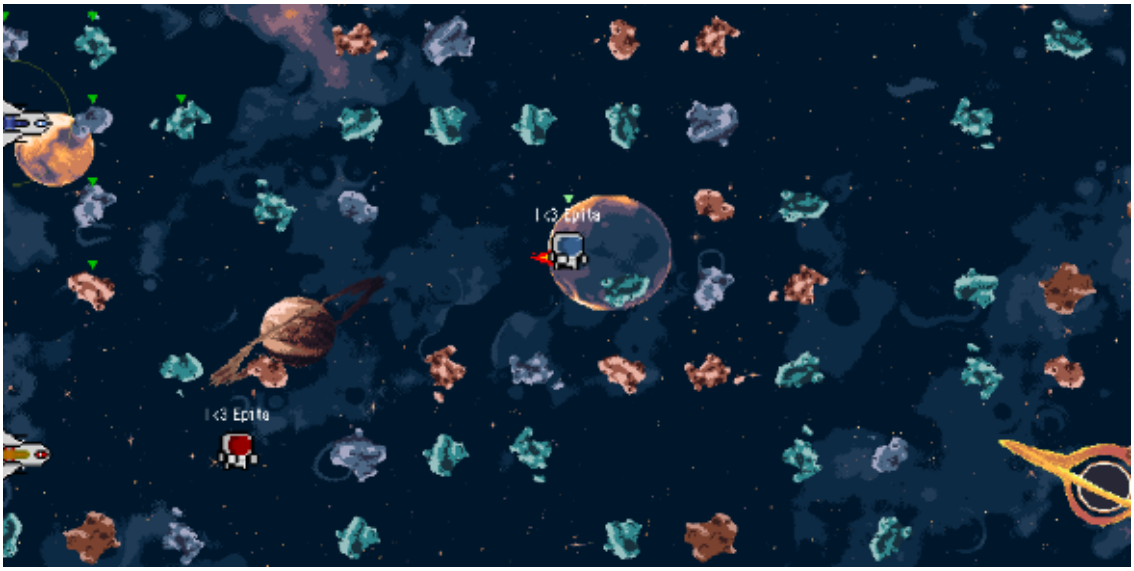
En plus du message de fin qui est affiché, un bouton "menu" a été ajouté pour permettre aux joueurs de retourner sur le menu du jeu pour relancer une nouvelle partie. Faisant partie du message de fin, lui aussi ne sera affiché que lorsqu'une équipe aura franchi la ligne d'arrivée.

8.4 Conclusion

Pour finir nous avons donc mis en place des conditions de victoire ainsi qu'un message de fin pour mettre en avant l'équipe victorieuse. Nous avons rencontré quelques petits problèmes tel que l'affichage du message de fin. Nous ne savions où le placer, et avons cherché quel placement permettait de le garder sur l'écran du joueur qu'importe ces mouvements. Un autre problème rencontré qui lui est toujours d'actualité, est avec le bouton "menu" qui ne déconnecte pas le joueur de la partie. Autrement dit, les joueurs sont bien sur leur menu, comme le suggère le bouton mais sont toujours connectés à la partie à laquelle ils jouaient les empêchant d'en rejoindre ou d'en créer une nouvelle.

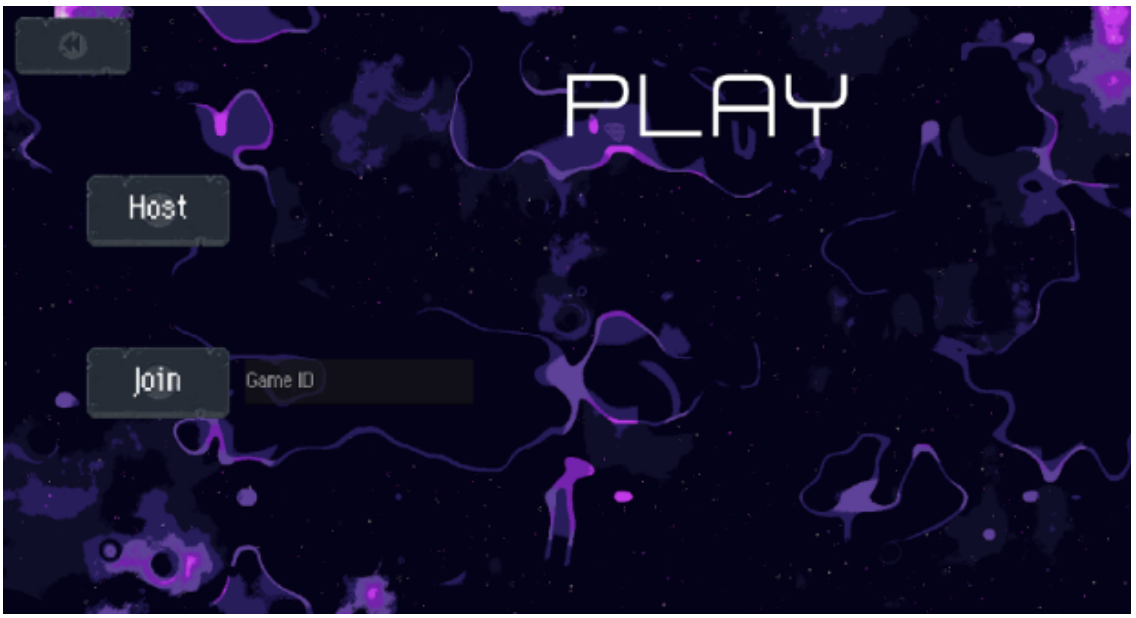
Réseau et Multijoueur

Afin d'obtenir une dynamique de jeu agréable, le multijoueur permet aux joueurs de rejoindre leurs amis dans des parties personnalisées. Nous avons rencontré beaucoup de difficultés lors de l'implémentation de cette fonctionnalité, nous y reviendrons prochainement.
L'objectif principal était de garantir une expérience fluide et immersive pour tous les joueurs. Cependant, la synchronisation entre les joueurs s'est révélée être un défi de taille. La coordination des actions simultanées, la transmission des données en temps réel et la gestion des latences ont été des aspects cruciaux à prendre en compte.
De plus, nous avons dû optimiser la gestion des ressources réseau pour minimiser la charge sur l'ordinateur hôte, tout en maintenant une qualité de jeu optimale pour l'ensemble des participants. Pour ce faire, nous avons décidé de réduire le nombre d'échanges entre l'hôte et les clients. Les paquets envoyés sont alors toujours les mêmes, mais le délai d'envoi est plus long. Afin de garder une certaine fluidité dans les mouvements, nous avons utilisé un lerp. Le lerp est une opération mathématique qui permet de calculer une position moyenne sur laquelle afficher la position du joueur. De ce fait, les déplacements sont toujours fluides.



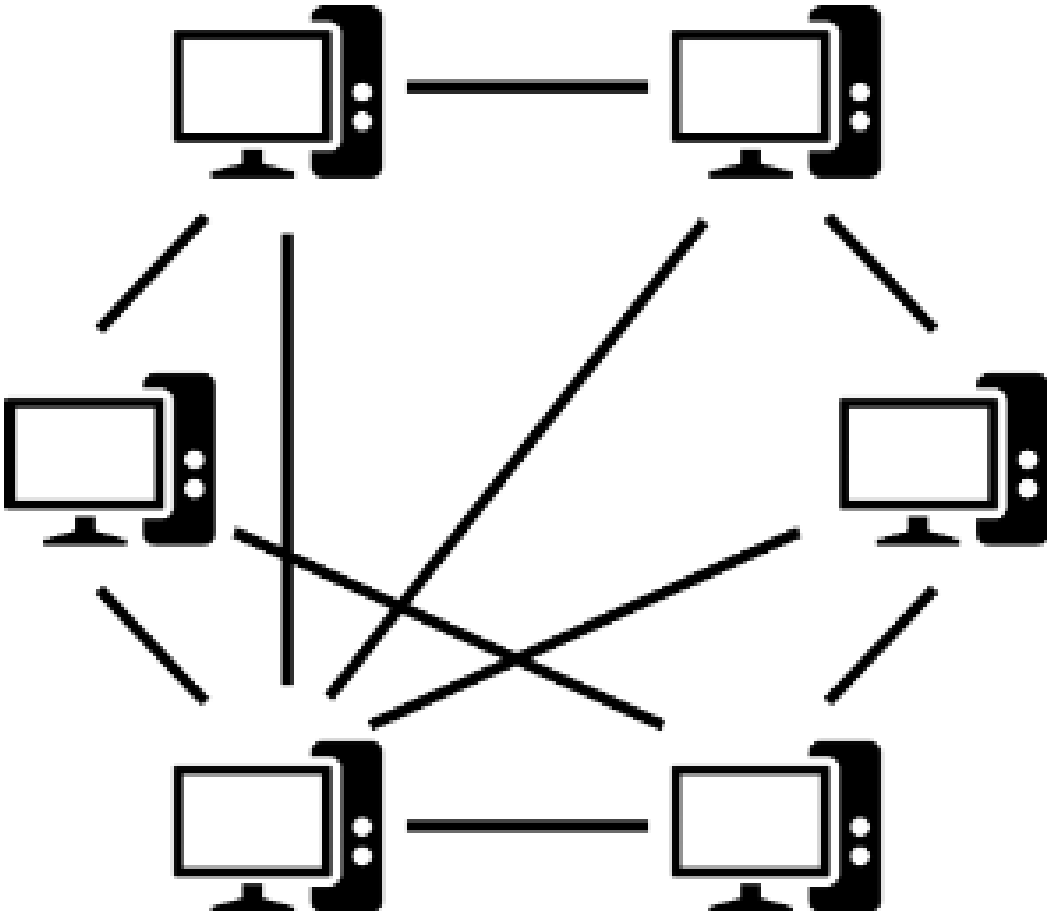
9.1 Initialisation du réseau

Le modèle de réseau que nous avons utilisé est appelé peer-to-peer. Très simplement, il s'agit d'un système qui repose sur la connexion à un ordinateur hôte.
Lors de l'initialisation, un utilisateur crée une partie, il sera alors hôte de la partie. Son ordinateur héberge alors la partie et les autres joueurs peuvent la rejoindre via un identifiant de partie généré à partir de l'adresse IP de l'hôte.
Lors de la connexion, hôte ou client, plusieurs signaux sont envoyés. Tout d'abord, une confirmation de la création du "serveur" (ordinateur de l'hôte) et de la connexion à celui-ci. Une fois que les autres joueurs se connectent à l'hôte, ils reçoivent eux aussi une confirmation qui va se caractériser par l'affichage de la salle d'attente ainsi que la transmission des informations du joueur à l'hôte (pseudonyme, équipe du joueur etc…).
Et enfin lors de la déconnexion, des signaux sont envoyés à chacun des joueurs afin de prendre en compte le fait qu'un joueur se soit déconnecté de l'ordinateur de l'hôte.



9.2 Hôte de la partie

Le peer-to-peer utilisé pour ce jeu fait donc appel à un ordinateur hôte. L'hôte va recevoir toutes les informations nécessaires afin de faire fonctionner la partie, puis les transmet à tous les joueurs connectés.
Etant donné que les joueurs ont besoin de l'adresse IP de l'hôte afin de rejoindre la partie, nous avons décidé d'utiliser une méthode de chiffrement pour sécuriser les connexions. L'adresse IP est donc transformée en une suite de chiffres calculés par l'ordinateur.
À chaque connexion, l'hôte reçoit des informations sur les joueurs (et plus précisément l'ordinateur du joueur), tel que le pseudonyme choisi par le joueur, l'équipe dans laquelle il se trouvera ainsi que son adresse IP. Cette dernière information n'est pas visible par l'hôte, mais reste une donnée enregistrée dans son ordinateur lors de la partie. L'hôte est le seul joueur qui puisse partager le code de la partie, ainsi que lancer cette dernière.



9.3 Salle d'attente

Nous avons décidé d'ajouter une salle d'attente afin de visualiser tous les joueurs qui rejoignent une partie. L'hôte de la partie est le seul à pouvoir partager l'identifiant de jeu mais aussi et surtout, il est le seul à pouvoir lancer le début de la partie.
Lors de cette période d'attente, nous avons décidé d'ajouter un mini-jeu afin de faire patienter les joueurs. Il s'agit d'un simple shooter, il faut tirer sur les aliens qui apparaissent de part et d'autre de l'écran. Ce mini-jeu n'a aucun objectif en dehors de faire patienter les joueurs, et ne doit en aucun cas être pris pour une fonction importante de SpaceY. Nous l'avons implémenté à la suite à des tests effectués sur la fusée lors du développement de la mécanique de tir.

9.4 Conclusion

Comme indiqué dans le cahier des charges, nous avions prévu de fournir environ 60% du travail sur la partie réseau, ce qui a été respecté. En effet, nous n'avons pas rencontré de retard particulier malgré le fait que le début de l'implémentation du multijoueur fut compliqué. Nous avons beaucoup cherché sur internet et sur la documentation Godot comment envoyer des informations sur des machines connectées à un même réseau. Nous nous tenons à jour afin de faire progresser la partie réseau en même temps que les autres. Il nous est impossible de dire que cette partie est terminée étant donné que les données transmises à travers le réseau se trouvent dans n'importe quel aspect du jeu. Ainsi, à chaque nouvelle avancée, nous implémentons simultanément les échanges de données.

Site Internet

- `https://spacey-epitchad.github.io/`

Le site internet est un élément important de notre projet. Il nous permet de rediriger nos joueurs vers notre jeu ainsi que vers les différents liens sources. Il est important qu'il soit simple et efficace car il est la première image que nos joueurs ont d'Epit'Chad.

10.1 Développement

Ce site internet a été réalisé sur la base d'un template. Afin de le personnaliser, nous avons ajouté diverses pages que nous développerons par la suite. Le template utilise le langage SCSS que nous n'avons pas utilisé par la suite. En revanche, nous avons utilisé le code de balisage HTML5, ainsi que les langages de programmation JavaScript et CSS pour la mise en forme et les animations. Le site internet utilise BootStrap, une librairie d'animation. Nous avons souhaité l'utiliser afin de rendre la lecture plus agréable. Tout d'abord, le site comprend une page d'accueil sur laquelle il est possible de retrouver toute la structure du site internet. Puis une page "Jeu" sur laquelle il est possible de télécharger la dernière version publiée de SpaceY ainsi que les rapports de soutenance. Ensuite la page "Équipe" présente les différents membres de l'équipe ainsi que des liens menant à leurs réseaux sociaux et leurs propres sites internets. Pour finir, une page "à propos" contient toutes les informations concernant l'historique du projet, ainsi que des explications et des spécifications sur le projet SpaceY en plus des différentes sources utilisées pour la réalisation du projet.

10.2 Conclusion

Nous sommes fiers de pouvoir fournir un site internet complet. En effet, dans le cahier des charges nous avions prévu de fournir un site à 70% terminé. En revanche, il est entièrement terminé. Il reste tout de même des textes à éditer au fur et à mesure de l'avancée du projet, ainsi que les liens du projet à modifier, mais le développement du site est entièrement terminé.

Conclusion

Comme démontré dans ce rapport, SpaceY avance à grands pas vers une version finale et optimale. En effet, pour chaque partie de façon individuelle, nous avons plus ou moins tenu les dates inscrites dans le cahier des charges. Et globalement, nous sommes en avance sur le résultat final qui devrait être terminé en mai. Nous sommes fiers de pouvoir fournir une version efficace pour cette soutenance afin de prouver que SpaceY ira au bout de ses ambitions.