

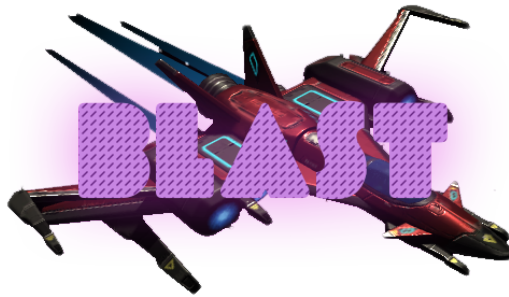
# Projet Blast

## Rapport de deuxième soutenance

— Groupe Quadro —

Nicolas FROGER  
Mathieu GUÉRIN  
Pierre DE LA RUFFIE  
Clément PERICAT

26 avril 2019



## Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Mise à jour du cahier des charges</b>	<b>4</b>
<b>3</b>	<b>Avancement des fonctionnalités</b>	<b>5</b>
3.1	Gestion du vaisseau . . . . .	5
3.2	Gestion des missions . . . . .	8
3.3	Interface . . . . .	9
3.4	Améliorations du vaisseau . . . . .	10
3.5	Sauvegardes . . . . .	11
3.6	Multijoueur . . . . .	12
3.7	Monde et Carte . . . . .	13
3.8	Site web . . . . .	15
<b>4</b>	<b>Structure du repository</b>	<b>16</b>
<b>5</b>	<b>Expériences personnelles</b>	<b>17</b>
<b>6</b>	<b>Avances et Retards</b>	<b>18</b>
<b>7</b>	<b>Prévisions pour la suite</b>	<b>19</b>
<b>8</b>	<b>Conclusion</b>	<b>21</b>

## 1 Introduction



Le projet Blast a connu un bon nombre d'évolution pendant cette seconde période, entre la première et la seconde soutenance. Elle a permis de développer davantage des fonctionnalités et d'améliorer celles existantes. Notre jeu ressemble désormais plus à l'idée que nous en avions lors de l'écriture du cahier des charges.

Les avancées faites jusqu'à maintenant ont été possibles à l'aide des différentes connaissances que nous avons acquies en travaux pratiques à l'école, ainsi qu'à l'aide de nos recherches personnelles sur Unity et les outils que nous utilisons dans le cadre de ce projet. L'avancement à ce jour sera davantage détaillé dans la suite de ce rapport.

Cette période a été aussi marquée par la fusion de deux groupes avec deux projets différents :

- *Blast*, développé par :
  - Nicolas FROGER
  - Mathieu GUÉRIN
- *Blitzkrieg*, développé par :
  - Pierre DE LA RUFFIE
  - Clément PERICAT

La question s'est donc posée de quel projet nous allions conserver. Au final, le projet retenu, comme le titre du rapport le laisse savoir, a été **Blast**.

Les deux groupes fonctionnaient de manière différentes : le premier à l'aide de *Git* et de *Github*, et le second à l'aide de *Unity Collab*. Un temps d'apprentissage et de mise en accord sur l'utilisation des outils de collaboration a donc été nécessaire.

## 2 Mise à jour du cahier des charges

Depuis la précédente soutenance, l'évolution du groupe a nécessité une réflexion de groupe à propos du cahier des charges que nous avons défini jusqu'alors.

En effet, un membre du groupe est parti, puis un autre groupe de deux personnes a été intégré au notre. Nous sommes désormais quatre membres et le nouveau groupe est alors constitué de :

- Nicolas FROGER (chef de projet)
- Mathieu GUÉRIN
- Pierre DE LA RUFFIE
- Clément PERICAT

Suite à cette fusion de deux groupes, il a fallut choisir lequel des deux projets allait être continué et lequel allait être laissé de côté. Le projet **Blast** a été retenu et c'est donc le développement de celui-ci qui est poursuivi. Il a donc été nécessaire aux deux nouveaux membres de s'approprier le jeu et se mettre au courant de ce qui avait déjà été fait pour être en mesure de commencer à ajouter des fonctionnalités.

La section "Objets du jeu" a été renommée en "Améliorations du vaisseau" pour mieux refléter l'évolution du projet.

### 3 Avancement des fonctionnalités

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

#### 3.1 Gestion du vaisseau

Participants au développement :

- Mathieu GUÉRIN
- Pierre DE LA RUFFIE
- Nicolas FROGER

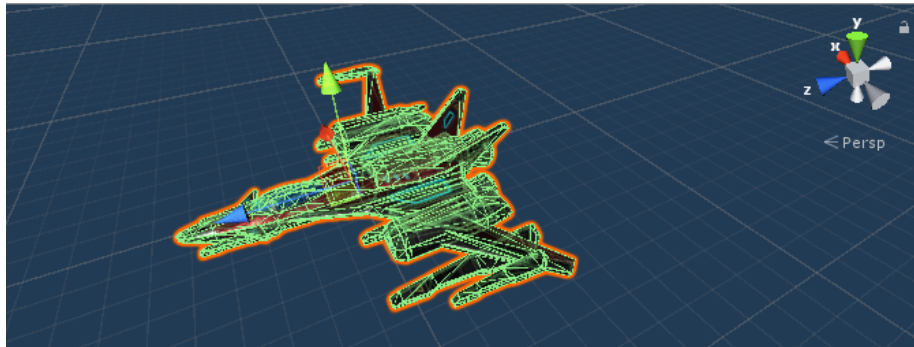
**Interactions entre entités** (Mathieu) :

Les vaisseaux sont désormais destructibles. Précédemment, nous avons introduits les points de vie pour les vaisseaux, mais ces points ne pouvaient pas évoluer. Désormais, une collision entre un objet quelconque et le vaisseau fera baisser son niveau de vie. Cela fonctionne notamment avec les armes équipées sur les vaisseaux. Il est donc possible de tuer un autre joueur dans une partie multijoueur à l'aide de ses armes.

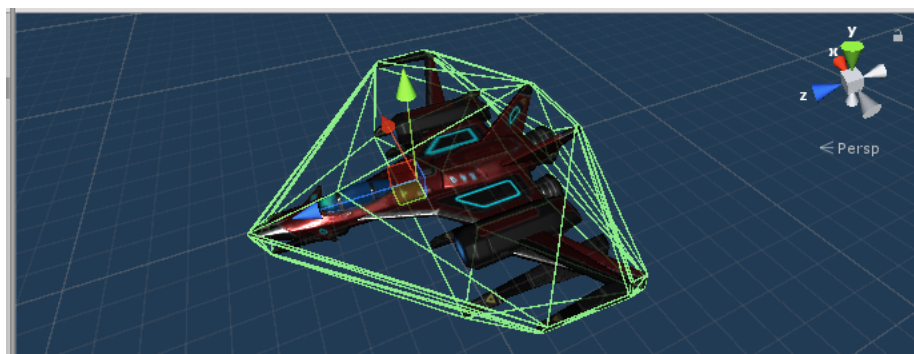
Cette fonctionnalité a été étendue à l'aide d'une *interface C#*. Cette notion, abordée en travaux pratiques à l'EPITA, permet de définir des méthodes qui seront communes à différents objets. Dans ce cas précis, l'*interface C#* (nommée `ILivingEntities`) définit des méthodes communes à toutes les entités "vivantes", c'est-à-dire des objets possédant un niveau de vie qui peut baisser au cours de la partie jusqu'à zéro, ce qui engendre la mort de l'entité. Les joueurs sont considérés comme des entités vivantes et implémentent donc cette *interface*.

**Collisions** (Pierre) :

Les vaisseaux ont subi une refonte totale de leurs systèmes de collision. Lors de la première partie de développement, la boîte de collision du vaisseau était composée de plusieurs cube, et n'était donc pas très précise (il arrivait d'avoir un bout d'aile ou de canon qui traversait un astéroïde). Nous avons ensuite décidé d'utiliser une fonctionnalité de Unity, le *Mesh Collider*, qui permet à partir d'un mesh de pouvoir calculer une boîte de collision plus précise que de simples cubes. Cela donne le collider suivant :



Cependant un problème subsiste. Si on s'en réfère à la documentation Unity, les Mesh Collider marqués comme étant *Convex* peuvent entrer en collision avec d'autres Mesh Collider, ce qui n'est pas le cas du vaisseau ci-dessus. Les autres vaisseaux (et les astéroïdes) possédant aussi des Mesh Collider, il est évident que celui-ci doit être Convex ou sinon les vaisseaux pourront se traverser. Les Mesh Collider Convex possèdent une limite de 255 triangles, ce qui explique la diminution de la précision de la version finale de la boîte de collision du vaisseau (il en va de même pour les astéroïdes). Celle-ci reste tout de même suffisante pour ce projet :



#### Mécaniques de tir (Mathieu) :

La mécanique de tir a aussi été mise à jour : il est maintenant possible de charger les canons avant le tir, et ce de manière individuelle. Ainsi, maintenir le clic gauche ou droite démarrera, respectivement, la charge de l'arme gauche ou droite. Ce chargement est perçu par le joueur à l'aide d'une orbe translucide qui augmente progressivement de taille au bout du canon. Au bout d'un court instant, des particules apparaîtront en plus pour rendre l'effet plus visible au joueur. On peut voir l'effet au bout des deux canons du vaisseau sur l'image ci-dessous.



Le chargement des armes du vaisseau du joueur peut être utile à ce dernier puisqu'il a des effets sur les dégâts et sur la vitesse des projectiles. Le chargement devient maximal au bout de 2 secondes, c'est à dire qu'au delà de 2 secondes, les effets seront les mêmes. Le chargement maximal permet, au tir, la propulsion de projectiles avec les effets suivants qui leur sont appliqués :

- Les projectiles font **3 fois** plus de dégâts aux ennemis ; et
- Les projectiles se déplacent **2 fois** plus rapidement.

#### Physique des projectiles (Nicolas) :

La vitesse de propulsion des projectiles était auparavant gérée indépendamment de celle du vaisseau, ce qui causait parfois le phénomène absurde que le vaisseau allait plus vite que le projectile qu'il tire devant lui.

Ce problème a été corrigé en ajoutant la vitesse du vaisseau à la vitesse de propulsion du projectile.

## 3.2 Gestion des missions

Participants au développement :  
— Nicolas FROGER

Un début de système de missions a été implémenté par Nicolas. Le joueur peut sélectionner une mission à effectuer dans une fenêtre contenant une liste de missions disponibles, avec le nombre de points d'expérience à la clef si elle est réussie.

Le premier type de mission implémentée est la mission de livraison. Le joueur doit se rendre dans une zone pour récupérer un colis et doit l'amener dans une autre zone. Les positions des zones sont déterminées aléatoirement tout en restant dans des distances atteignables par le joueur.

Les missions sont implémentées à l'aide de classes abstraites. Cette notion a également été vue en travaux pratiques à l'EPITA. Ce type de classe permet de définir une classe partielle, dans ce cas *Mission*, qui doit être étendue. Dans notre cas, cela nous permet de développer différents types de missions, tout en gardant des propriétés et méthodes communes, ce qui évite de gérer tous les types manuellement pour faire la même chose. Tous les types de missions partagent certaines propriétés, comme le nombre de points d'expérience qu'elles donnent au joueur quand celui-ci la finit. Les missions disposent toutes de mêmes méthodes comme la méthode *Begin()* qui correspond au démarrage de la mission.



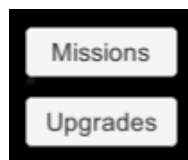
### 3.3 Interface

Participants au développement :

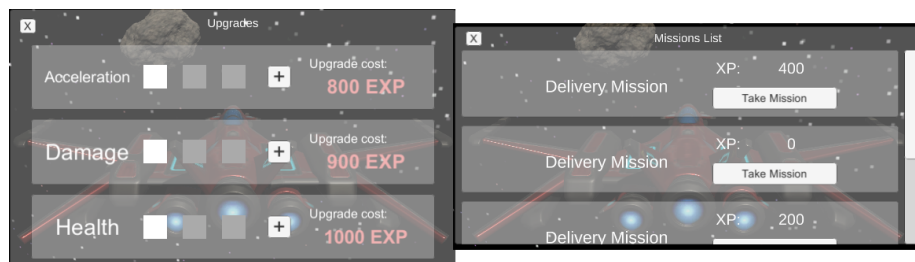
- Nicolas FROGER
- Pierre DE LA RUFFIE

**Interfaces utilisateur** (Nicolas) :

De nouvelles interfaces ont été implémentées dans le jeu. Tout d'abord, il a bien évidemment fallu créer de toutes nouvelles interfaces en jeu pour les missions et les améliorations évoquées précédemment.



C'est pourquoi l'interface de jeu possède maintenant deux boutons permettant d'accéder aux deux fenêtres *Upgrades* et *Missions*, montrés ci-dessous.

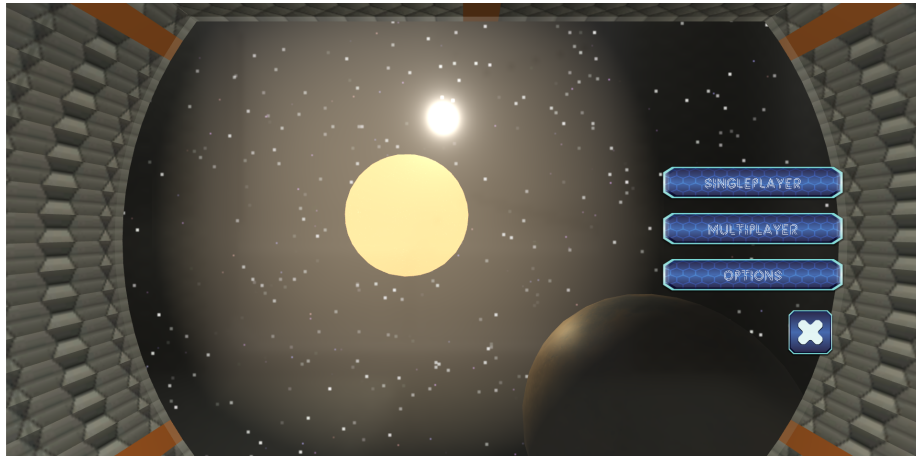


Sur l'image, on peut voir à gauche la fenêtre des *upgrades* et à droite la fenêtre des *missions*. Les caractéristiques des *missions* et des *upgrades* seront expliquées dans les sections suivantes.

Nous avons aussi ajouté un menu de pause, accessible en appuyant sur la touche **[Esc]**. Celui-ci permet de mettre le jeu en pause pour le reprendre ensuite, ou le quitter.

**Menu principal** (Pierre) :

Enfin, l'évolution du projet a aussi nécessité une remise en forme du menu. Les boutons suffisaient pour la première partie du développement mais cette nouvelle période nécessite un menu un peu plus poussé. La structure du menu en général est resté la même (*Singleplayer* - *Multiplayer* - *Options*) même si les options ne sont pas encore implémentées.



Le menu prend place dans une station spatiale. Chaque sous-menu correspond à un endroit différent de cette station où la caméra va se déplacer pour afficher les différents boutons.

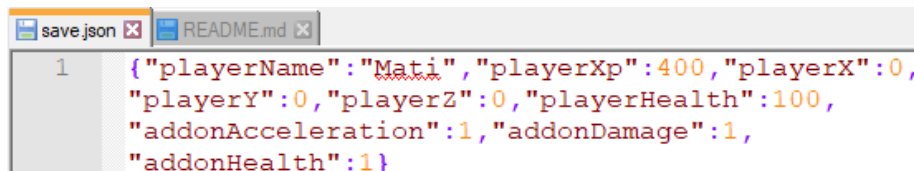
### 3.4 Améliorations du vaisseau

Participants au développement :  
— Nicolas FROGER

Cette partie était anciennement dénommée "les objets" car nous avions en idée que les objets seraient utilisés pour améliorer le vaisseau. Cependant, nous avons préféré faire un système d'amélioration de vaisseaux plus simple. À l'aide des points d'expériences pouvant être acquis dans les missions, il est désormais possible "d'acheter" des améliorations. L'amélioration sera appliquée au vaisseau après l'échange des points. Actuellement, les améliorations implémentées concernent l'accélération du vaisseau, les points de vie du joueur et les dégâts qu'infligent les armes du vaisseau. Chacune d'entre elle dispose de trois niveau d'amélioration.

### 3.5 Sauvegardes

Participants au développement :  
— Mathieu GUÉRIN





```
1 {"playerName": "Mati", "playerXp": 400, "playerX": 0,
  "playerY": 0, "playerZ": 0, "playerHealth": 100,
  "addonAcceleration": 1, "addonDamage": 1,
  "addonHealth": 1}
```

Avec l'ajout des missions et des améliorations, il était nécessaire de permettre au joueur de sauvegarder son expérience ainsi que les attributs de son vaisseau. Ces attributs sont vastes et comprennent les améliorations appliquées par le joueur (attributs préfixés par "addon"), ainsi que :

- le nom du joueur (playerName) ;
- sa position (playerX, playerY, playerZ) ;
- son niveau de vie (playerHealth) ;
- son niveau d'expérience (playerXp)

Cette liste est non-exhaustive et sera amenée à être étendue par la suite pour permettre la sauvegarde de davantage d'attributs.

Pour le moment, le chargement de la sauvegarde, si elle existe, est fait automatiquement au lancement d'une partie multijoueur ou solo. La sauvegarde est manuelle et se fait en appuyant sur la touche  . À terme, la sauvegarde sera ajoutée au menu de pause.

Il est également possible d'effacer la sauvegarde existante en appuyant sur la touche  .

### 3.6 Multijoueur

Participants au développement :

- Nicolas FROGER
- Mathieu GUÉRIN

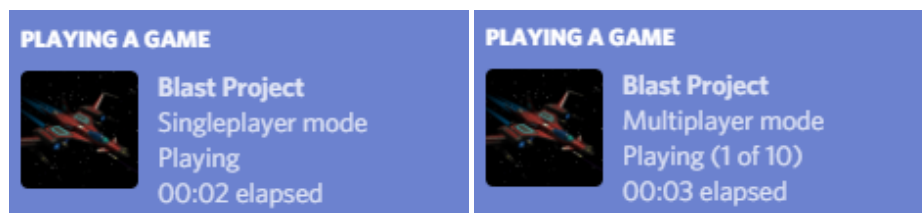
**Mode multijoueur du jeu** (Nicolas) :

Le multijoueur a subi quelques modifications. En effet, certaines optimisations ont été effectuées permettant de ne pas faire des calculs inutiles en arrière plan. Une nouvelle fonctionnalité a été ajoutée, les joueurs peuvent désormais choisir un nom qui sera affiché en jeu au dessus de leur vaisseau. Il a également été nécessaire d'assurer que toutes les nouvelles fonctionnalités étaient compatibles avec le multijoueur, et si ce n'était pas le cas de les retravailler. Le multijoueur est donc toujours fonctionnel et permet à des joueurs de se rejoindre dans une même partie.

**Intégration avec Discord** (Mathieu) :

Comme nous utilisons beaucoup Discord et que le service se connecte assez bien aux jeux vidéos, nous avons pensé intégrer le service *Rich Presence* de Discord à notre jeu vidéo. Cela a pour effet d'afficher sur le profil de l'utilisateur la notification qu'il est en train de jouer à Blast lorsqu'il lance une partie dans le jeu.

L'information contient un certain nombre d'informations, appelées contexte. Sur les deux images ci-dessous, on peut voir que l'affichage diffère selon si la partie dans le jeu est lancé en mode solo ou multijoueur.




### 3.7 Monde et Carte

Participants au développement :

- Pierre DE LA RUFFIE
- Mathieu GUÉRIN

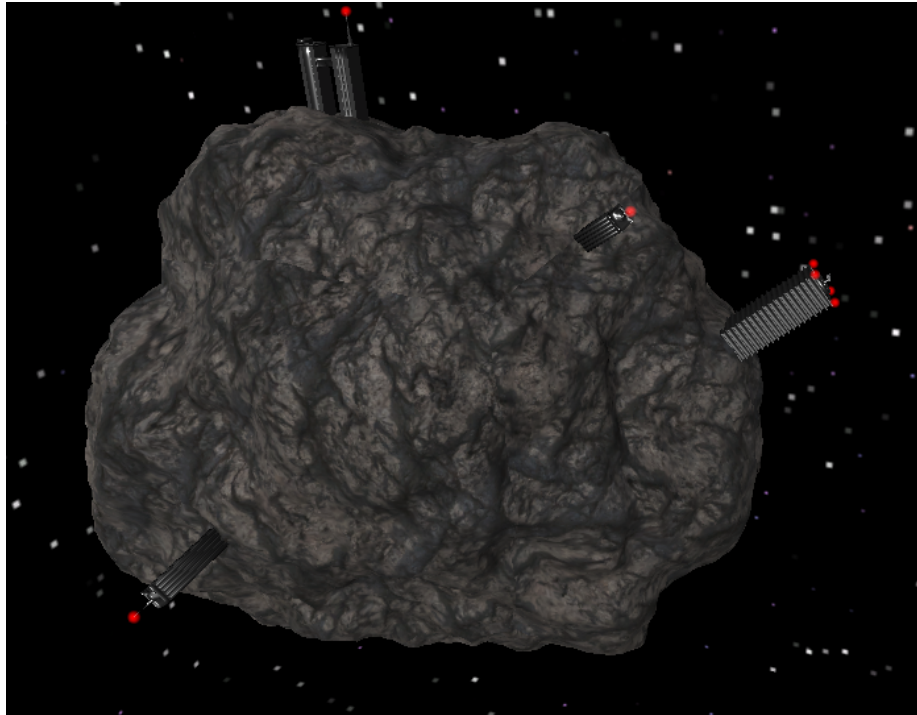
Vue d'ensemble autour du joueur (Mathieu) :



La possibilité a maintenant été donné au joueur de passer de la vue "normale" (derrière le vaisseau) à une vue au dessus du vaisseau, dite "vue d'ensemble". Cette nouvelle vue permet au joueur de voir les éléments autour de lui, mais aussi au dessus en en dessous. Le joueur est en mesure de passer d'une "vue" à l'autre en appuyant sur la touche .

Pour permettre une plus grande flexibilité, le joueur peut, lorsque qu'il en mode de vue d'ensemble, zoomer ou dézoomer à l'aide de la molette de sa souris.

**Base ennemie (Pierre) :**



Il s'agit d'un astéroïde sur lequel est placé plusieurs bâtiments, ainsi qu'un bâtiment plus gros, le QG (*que l'on peut voir sur la droite sur l'image*) sur lequel est posé une tourelle automatique. Tout les bâtiments possèdent un halo qui émet une lumière rouge. Le but sera d'implémenter une mission de destruction de bâtiments, et une fois qu'un bâtiment sera détruit le halo deviendra vert. Pour l'instant, tout cela fait partie d'une branche qui n'a pas encore été merge à la branche master : il reste encore beaucoup à faire mais nous en reparlerons plus loin.

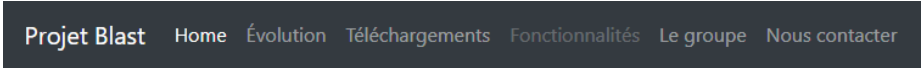
### 3.8 Site web

Participants au développement :

- Mathieu GUÉRIN
- Nicolas FROGER

Le site web a également été travaillé et amélioré. En effet, il contient de nouvelles pages :

- Une page qui retrace les évolutions majeures du groupe et du projet.
- Une page des téléchargements contenant des liens pour télécharger le jeu et les fichiers en rapport avec le projet avec notamment le cahier des charges et les rapports de soutenance.
- Une page qui présente les fonctionnalités majeures du jeu (encore en développement).
- Une page de présentation du groupe, qui parle de ce dernier et de ses membres.
- Une page de contact, qui permet à un visiteur de nous envoyer un mail (encore en développement).

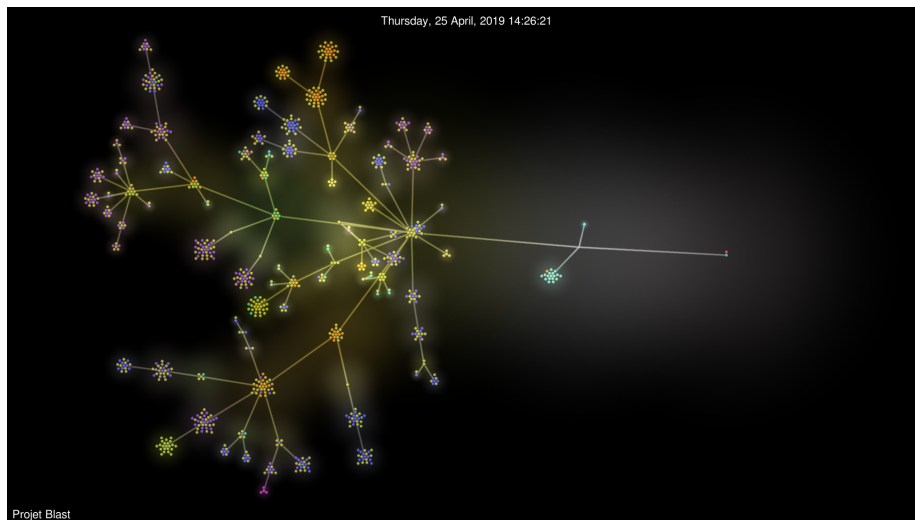
A dark horizontal bar containing a navigation menu with the following items: **Projet Blast**, Home, Évolution, Téléchargements, Fonctionnalités, Le groupe, and Nous contacter.

**Projet Blast** Home Évolution Téléchargements Fonctionnalités Le groupe Nous contacter

L'image ci-dessus correspond au menu tel qu'il est affiché sur le site web que nous avons développé.

## 4 Structure du repository

Comme dans notre rapport de première soutenance, voilà la représentation schématisée du *repository Git* de notre projet. Cette visualisation a été générée à l'aide de l'outil Gource (<https://gource.io/>) à partir de l'historique des modifications de notre projet. Chaque branche représente un dossier, chaque élément représente un fichier et chaque couleur représente un type de fichier.



Au bout droit, on trouve la racine du projet. De droite à gauche, on trouve sur l'arbre :

- Les fichiers de configuration Unity, dont la majorité sont en bleu ciel.
- Au centre, les fichiers du jeu que nous avons développé, y compris scripts et éléments du jeu.
- À gauche, les grandes branches de la partie haute correspondant aux trois librairies principales que nous avons utilisé :
  - StarSparrow, une librairie avec les vaisseaux.
  - Photon, qui nous aide pour le multijoueur.
  - TextMeshPro, pour améliorer les textes de l'interface.
- Enfin, toujours à gauche, dans la partie basse du schéma, la dernière grande branche correspond à tous les *assets* qui ont été nécessaires à la conception du nouveau menu de notre jeu.

À but informatif, voici quelques statistiques de notre projet sur Github :

- À ce jour, le *repository* du projet possède plus de **200 commits**,
- Pour une taille totale de près de **150 Mo**.

Pour permettre une bonne collaboration au sein du groupe, nous avons également mis à profit les outils mis à disposition par Github : à savoir la création d'*issues* et de *pull requests*. Les *issues* nous permettent de signaler et de discuter des bugs rencontrés, ainsi que des futures fonctionnalités que nous prévoyons.



d'ajouter à notre jeu. Les *pull requests* sont créées à chaque fois que le développement d'une fonctionnalité majeure est terminé. Cela permet à l'équipe de se mettre au courant du développement faits par d'autres, ainsi que de donner leur avis sur leur travail.

En termes de chiffres, pour le moment, on compte :

- **20 pull requests**
- **7 issues (dont 4 résolues).**

## 5 Expériences personnelles

**NICOLAS :** Cette deuxième période a été l'occasion pour moi d'améliorer mes connaissances en programmation avec Unity, car la première période n'avait pas permis d'approfondir cet aspect. En avançant dans le développement du jeu, je me suis rendu compte que le langage de script de Unity ne fonctionne pas toujours de la même manière que le C#. Il faut donc parfois réfléchir sur des problèmes différemment que si nous étions en travaux pratiques par exemple. Un exemple concret pour cela pourrait être au niveau des classes qui sont parfois un peu différentes dans Unity que dans le C# classique. C'est un bon exercice car cela nécessite de pouvoir s'adapter rapidement. Ce projet me permet surtout de m'entraîner et d'apprendre sur la réalisation d'idées. En effet, lorsqu'il faut transformer une idée en code, il est difficile de trouver le moyen le plus efficace et le plus optimisé de le faire. Ce projet étant un premier projet de grande envergure pour moi, il m'arrive souvent de devoir recommencer une fonctionnalité qui me semble irréalisable car je la développe de la mauvaise façon.

**MATHIEU :** Cette nouvelle période a été intéressante pour moi puisque j'ai pu continuer à travailler sur les fonctionnalités que j'avais déjà commencé et qui me tiennent à cœur. Le travail de programmation pour la "standardisation" des entités vivantes a été très stimulant puisqu'il qu'il met en jeu *interfaces C#*, *classes abstraites* et *classes d'objet Unity*. Ces fonctionnalités ne demandent qu'à être améliorées pour la prochaine soutenance. Certaines mécaniques de jeu ont nécessité des recherches pour comprendre comment les implémenter dans Unity. Mes ajouts viennent souvent ajouter des fonctionnalités proposées au joueur dans le but de renforcer le travail déjà fait par mes coéquipiers. Aussi, j'aurais eu besoin de revoir la structure du projet pour l'implémentation du support des sauvegardes dans le jeu. Beaucoup de travail reste à faire, mais une base solide est bien là. Enfin, expérimentant avec un jeu vidéo pour la première fois, j'en ai profité pour ajouter une intégration avec Discord, ce qui a été plutôt amusant à faire.

**PIERRE :** J'ai eu personnellement beaucoup de mal à effectuer la transition d'un groupe à l'autre. Déjà, j'ai dû apprendre à utiliser *git* correctement car j'ai toujours des problèmes avec pour les travaux pratiques de programmation. Je me suis aussi rendu compte de la difficulté qu'a été les conflits de merge qui se sont succédés. Ensuite, je me suis vite rendu compte de la différence entre mes scripts et ceux écrits par mes camarades, qui étaient plus épurés (notamment avec les opérateurs conditionnels ternaires). J'ai donc essayé de me mettre à leur niveau pour ne pas avoir trop de différences entre les différents scripts même si ce n'est pas encore tout à fait le cas. Je ressors aussi très refroidi de cette seconde partie, en ayant l'impression d'avoir passé plus de temps à chercher des modèles 3D qu'à travailler réellement en C#. Mais je sais que la dernière partie sera différente, puisque la majorité de ce qu'il reste à faire sera sur Unity maintenant que tous les modèles 3D sont dans le projet. Au fond, la recherche et la manipulation des modèles 3D notamment concernant le menu m'a intéressé et j'ai pu découvrir beaucoup de fonctionnalités un peu discrètes de Unity comme les Mesh Renderer ou les Mesh Collider, et j'espère pouvoir en apprendre d'avantages dans cette dernière partie.

## 6 Avances et Retards

Pour l'instant, nous sommes placés sur les prévisions suivantes concernant l'avancement de certaines parties.

### Avance :

Nous avons maintenu notre avance sur la partie **Multijoueur**, qui continue d'être développée avec le reste du jeu. Le multijoueur fonctionne correctement, malgré les quelques problèmes mineurs que nous avons identifié au sein du gameplay en multijoueur et qu'il faudra que nous corrigions.

### Retard :

Nous avons un maigre retard sur la partie **Monde et Carte**, mais la plupart des éléments qui nous permettront de constituer cette partie sont en développement et donc le retard pourra être rattrapé.

## 7 Prévisions pour la suite

Cette partie exposera un aperçu de ce que nous prévoyons de faire dans notre projet pour la suite et la soutenance finale.

### Gestion du vaisseau :

Une simple adaptation par rapport au travail qui sera fait en parallèle sera nécessaire. Le reste du travail effectué pour cette section nous satisfait déjà.

### Gestion des missions :

En ce qui concerne les missions, nous pouvons envisager de faire plus de missions. En effet, le seul type de missions disponible pour le moment est la livraison. L'utilisation des classes abstraites mentionnées précédemment permettront d'étendre facilement ces nouveaux types.

### Interface :

Le bouton *Option* du menu principal ne contient aucune fonctionnalité, il faudrait que ce menu contienne un réglage du son, au minimum. De plus, les boutons du menu en jeu ainsi que ceux pour accéder aux missions et aux améliorations du vaisseaux devront être les mêmes que ceux du menu principal.

### Multijoueur :

De la même manière que lors de cette deuxième période, le multijoueur sera adapté à chaque nouvelle fonctionnalité et sera optimisé sur certains points.

### Monde et Carte :

Concernant l'astéroïde, la partie la plus visible est faite mais il reste quelques détails à régler. Tout d'abord, il faut implémenter la mission de destruction de bâtiment, rendre les bâtiments destructibles et afficher leurs états détruits. Il faut aussi améliorer la rotation de la tourelle car elle finit très vite par être à l'envers à force de rotations successives. Il faudrait aussi rajouter des bâtiments qui, eux, ne seront pas destructibles afin de rendre vraiment le halo rouge comme un indicateur d'un bâtiment.

Nous aimerions aussi écrire un script capable de générer des champs d'astéroïdes avant la génération de la map.

Un problème majeur à régler concernant la boussole est que celle-ci ne s'affiche pas correctement lorsque le jeu est dans une résolution plus haute que celle de l'éditeur *Unity*.

**Site web :**

Pour le site web, le contenu de la plupart des pages devra être complété, pour rendre le site web plus vivant. La page des téléchargements devra être complétée avec des liens à jour et une meilleure clarté dans la navigation. Enfin, d'autres pages auront besoin d'améliorations : les fonctionnalités et la page de contact devront être améliorées ou terminées.

- Le développement de la page sur les fonctionnalités majeures du jeu devra être finalisé.
- La page de présentation du groupe devra être complétée.
- Une page de contact devra être rendue fonctionnelle.

**Son et musiques :** Pour l'instant, le jeu ne contient aucune musique ni aucun son. Il est donc à prévoir l'ajout de bruitages quand les vaisseaux tirent, se prennent des dommages etc. Une/Des musiques sont aussi prévues.

## 8 Conclusion

Cette deuxième période a donc permis d'avancer considérablement dans notre projet. Après des difficultés sur la structure du groupe avec un départ et des arrivées de membres, nous avons réussi à ajouter des fonctionnalités que nous avions prévu ainsi que d'améliorer certaines que nous avions développé auparavant. Le jeu a désormais un but, des missions sont réalisables, et une progression du joueur est possible avec les améliorations de vaisseau.