Projet Blast

Rapport de première soutenance

— Groupe Quadro —

Nicolas FROGER
Mathieu GUÉRIN
Mattéo DEMICHELE

11 mars 2019

Table des matières

1 Introduction

Cette première période avant la première soutenance a permis de développer les bases de notre jeu vidéo et de prendre en main les outils que nous utilisons dans le cadre du projet : principalement Unity et Git, avec Github et en collaboration.

Ce rapport de soutenance présentera les fonctionnalités décrites dans le cahier des charges et notre avancement sur ces dernières, avec le rôle de chaque membre du groupe dans leurs développements. Le travail qui a été fait jusqu'alors a pour principal objectif de nous donner une base concrète du jeu vidéo que nous imaginons, sur laquelle nous seront en mesure d'ajouter plus facilement de nouvelles fonctionnalités et améliorer celles existantes.

2 Reprise du cahier des charges

Le groupe est aujourd'hui constitué de 3 personnes :
— Nicolas FROGER (chef de projet)
— Mathieu GUÉRIN
— Mattéo DEMICHELE

La répartition des tâches a été un peu plus difficile que prévue et est donc un peu floue, mais nous n'y avons pas apporté de modification depuis le dernier rendu. Nous essayons de tenir au mieux l'avancement des fonctionnalités, bien que cela soit parfois difficile.

Cependant, nous n'avons pour le moment que travaillé sur les bases du jeu et nous pouvons difficilement imaginer les potentiels futurs obstacles que nous rencontrerons qui pourraient nécessiter d'une adaptation du cahier des charges.

3 Avancement des fonctionnalités

L'avancement de chaque fonctionnalité du projet sera détaillé ci-dessous dans leurs sections respectives. Pour chaque section qui s'y prête, le développement de tous les aspects implémentés de la fonctionnalité sera détaillé chronologiquement, en précisant les auteurs des modifications, ainsi qu'une explication technique dans certaines d'entre elles.

Certaines fonctionnalités ne contiennent encore que peu d'éléments, mais seront améliorées par la suite. Plus de détails seront donnés dans la partie sur les prévisions pour le projet.

3.1 Gestion des sauvegardes

La gestion des sauvegardes n'a pour le moment pas encore besoin d'être implémentée. En effet, celle-ci ne sera utile que lorsqu'une progression sera possible, avec l'ajout d'objets de personnalisation et de missions, entre autres. Cette fonctionnalité permettra au joueur de garder son avancement dans le jeu et de reprendre plus tard pour avancer encore plus dans sa partie. Seront gardés en mémoire sa position dans le monde, son inventaire, l'état de son vaisseau et ses niveaux lorsque ceux-ci seront implémentés.

3.2 Contrôles du vaisseau

Participants au développement :
— Nicolas FROGER
— Mathieu GUÉRIN

Le vaisseau est un objet dans l'espace que le joueur peut contrôler. Une caméra y est virtuellement attaché, à la troisième personne, et permet au joueur de se déplacer dans le jeu. **Les mouvements** de ce dernier ont été configurés dès le début par Nicolas et améliorés ensuite. Pour le moment, le joueur est ainsi capable de contrôler :
— L'accélération et les mouvements du vaisseau (avec Z Q S D)
— Le roulis du vaisseau (avec A E)
— La direction du vaisseau (avec les mouvements de souris)
Un *boost* est également disponible pour le joueur et lui permet actuellement de multiplier par 10 sa vitesse maximale. Pour activer le boost, le joueur doit maintenir Shift ⇑ pendant son accélération.

L'accélération est d'ailleurs progressive : c'est à dire que le joueur mettra un certain temps à atteindre une vitesse rapide, et encore plus pour atteindre sa

vitesse maximale. Cette vitesse maximale est de 19.6 m/s en mode normal, elle est donc de 196 m/s en mode *boost*.

Le joueur est aussi capable de regarder ce qu'il se passe autour de lui sans modifier la direction de son vaisseau, il s'agit du mode « caméra libre ». Pour se faire, il suffit d'appuyer sur le clic molette avant de déplacer la souris.

Les mouvements ont été compliqués à bien gérer. En effet, le jeu se déroule dans l'espace, c'est-à-dire un environnement sans gravité, et donc sans notion de haut et de bas. Cela représente une vraie problématique car il ne faut pas que le joueur se perde. Il faut aussi que les contrôles se fassent relativement au joueur et non au monde, ce qui a posé beaucoup de problèmes au début car la caméra ne réagissait pas de la manière souhaité lorsque le vaisseau était incliné par exemple et a nécessité beaucoup de recherches pour les régler. Les solutions ont cependant été assez simples. Pour garder l'exemple de la caméra, nous utilisons la fonction `Transform.LookAt()` pour que la caméra regarde le vaisseau du joueur. Lorsque le joueur était incliné ou était à l'envers, la caméra restait orientée de la même façon et les contrôles devenaient très confus. Nous avons appris par la suite que cette fonction dispose d'un paramètre permettant de choisir le vecteur qui représente le haut, par défaut fixé à `Vector3.up`, il a donc suffit d'utiliser le vecteur haut relatif au joueur (`player.transform.up`) pour régler ce problème.

Des mécaniques de base pour **le combat** ont ensuite été ajoutées par Mathieu. Ce qui était à la base un simple canon est devenu deux canons, capables de tirer sur demande **des projectiles**, à l'aide du clic gauche et droit pour, respectivement, le canon gauche et droit. Ces projectiles tirés iront heurter les objets qu'ils rencontreront, avant de disparaître à l'impact. Les collisions sont détectées à l'aide de la méthode prédéfinie `OnCollisionEnter()` et supprime le projectile lorsqu'elle est appelée. Si aucune collision n'est détectée, le projectile est supprimé au bout d'un certain temps (environ 5 secondes).

Pour résoudre ensuite un problème pratique de gameplay soulevé par Nicolas, une aide à la visée avec un **viseur dynamique** a été ajouté à l'interface.

De plus, des **effets de traînées** ont été ajoutés au vaisseau, sur ses réacteur, pour donner au joueur une réelle impression de mouvement (que le vide de l'espace seul ne permet pas). Des effets similaires ont été ajoutés aux projectiles pour leur donner aussi un effet de vitesse, mais aussi de puissance. Ces deux traînées utilisent les composants *Trails emitter* de Unity.

3.3 Objets du jeu

Participants au développement :
— Mathieu GUÉRIN

Parmi les objets actuellement implémentés dans le jeu, on trouve **le vaisseau** lui-même et **les projectiles** tirés par ce dernier.

Par la suite, d'autres objets modifiants par exemple les propriétés du vaisseau pourront être ajoutés dans le jeu.

3.4 Gestion du vaisseau

En prévision de futurs additions au jeu, un indicateur de **santé du vaisseau** a été ajouté au jeu. Le joueur devra y faire attention lors des missions et autres évènements du jeu.

D'autres facteurs viendront influencer d'autres propriétés et statistiques du vaisseau, que le joueur devra apprendre à gérer.

3.5 Gestion des missions

Les missions ne sont pour le moment qu'au stade d'idée. Des objectifs devront être ajoutés pour diriger le *gameplay*. Ils offriront au joueur une progression à accomplir, et lui permettront de récupérer des récompenses pour modifier et améliorer les caractéristiques de son vaisseau.

3.6 Monde et Carte

Participants au développement :
— Mathieu GUÉRIN

Le monde sera rempli par la suite en même temps que l'implémentation d'objectifs et de missions.

Actuellement, une *skybox* est présente dans le jeu. Cette *skybox* a été créée à l'aide du logiciel *Spacescape*. Il s'agit d'un outil de création de *skybox* spatiales avec des étoiles et des nébuleuses. Ce logiciel est complet mais est difficile à prendre en main. Pour obtenir un bel univers, il faut créer différents calques, avec par exemple des points pour les étoiles. L'addition de plusieurs calques donne un résultat plus esthétique. Lorsque la création de calques est terminée, le logiciel dispose d'une fonction d'exportation pour des moteurs de jeux vidéos dont Unity. L'importation dans Unity est ensuite très simple : 6 images sont créées et il suffit de suivre les instructions d'Unity.

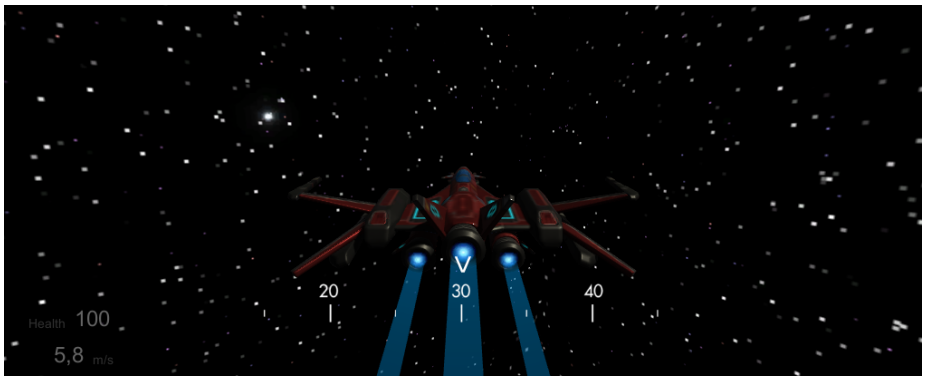
Pour le moment, seule une base pour la carte a commencé à être développée par Mathieu avec un mode de vision de la carte. La touche 

permet de basculer entre deux modes de vue :

— La vue normale : une vue à la troisième personne, derrière le vaisseau.
— La vue "carte" : une vue de loin, au dessus du joueur.
Dans cette vue, les contrôles du vaisseau sont désactivés et le joueur peut avoir un aperçu de ce qui se trouve autour de lui.

3.7 Interface

Participants au développement :
— Nicolas FROGER
— Mathieu GUÉRIN



L'interface est gardée volontairement simple pour permettre au joueur de garder une bonne visibilité dans le jeu.

Pour permettre au joueur de se repérer dans l'espace, **une boussole** a été ajouté à l'interface par Nicolas. Elle se met à jour selon les mouvement du joueur. La boussole est constitué d'un indicateur qui indique la valeur actuelle et la boussole en elle-même en dessous. Il s'agit d'une très grande image de près de 8000 pixels de largeur qui contient toutes les valeurs de 0 à 350. Cette image est masquée dans Unity à l'aide du composant *Mask* et est déplacée en fonction de l'angle du joueur entre un vecteur arbitraire qui représente le nord (arbitraire car il n'y a pas de nord dans l'espace) et l'orientation du vaisseau. L'image fait une boucle car en arrivant à la toute droite de celle-ci on retourne au début, c'est-à-dire à gauche. Cette boussole sera utile par la suite pour se rendre quelque part sur la carte et servira d'indicateur de direction.

En addition, Mathieu a ajouté l'affichage d'informations sur le vaisseau dans le bord inférieur gauche de l'écran. Actuellement, la santé et la vitesse du vaisseau du joueur sont affichés. La vitesse est donnée en mètres par seconde, 1 mètre équivalent à une unité dans le monde en 3D.

De plus, pour résoudre un problème de difficulté à la visée, Mathieu a ajouté un **viseur dynamique** à l'interface. Ce dernier est affiché quand le vaisseau pointe en direction d'un objet dans l'espace. Cela est rendu possible grâce au

tracé de rayons virtuels avec *Physics.Raycast()*. Cette fonction retourne le point d'impact avec un objet, s'il y en a un. Le point d'impact est ensuite projeté sur l'interface en deux dimensions, ce qui nous donne la position à jour du viseur, qui se placera aux coordonnées correspondantes. Dans le cas où le rayon virtuel ne rencontre aucun objet, la fonction retourne faux et le viseur est masqué jusqu'à ce que le joueur dirige son vaisseau vers un nouvel objet.

3.8 Multijoueur

Participants au développement :
— Nicolas FROGER

Une base fonctionnelle a été implémentée pour le multijoueur par Nicolas. Cette fonctionnalité utilise l'outil **Photon** pour permettre la communication entre les joueurs et faciliter le développement. L'avantage de Photon est, en plus de sa facilité d'utilisation, le *cloud*. En effet, l'éditeur de ce module s'occupe lui-même d'héberger les utilisateurs. Au lieu de fonctionner par partage d'IP comme dans de nombreux jeux, il suffit ici d'entrer le nom d'une salle pour rejoindre une partie. Il est possible de créer une salle mais celle-ci sera hébergée directement chez Photon au lieu d'être hébergée directement chez le joueur, ce qui permet une grande simplicité pour que plusieurs joueurs se rejoignent : il n'est par exemple pas nécessaire d'ouvrir des ports ou d'échanger des adresses IP.

Le multijoueur est requis dans le cadre de ce projet et est une partie de travail non négligeable. Il nous a été recommandé à plusieurs reprises de faire de cette fonctionnalité une priorité et de tout implémenter autour d'elle pour ne pas avoir à tout recommencer par la suite. Ces différentes raisons expliquent

pourquoi cette partie a été travaillée dès le début du projet. Le premier objectif était que deux joueurs puissent se rejoindre dans une même partie et puissent évoluer dans le même monde. Il a donc fallu en premier lieu créer un menu afin de créer ou de rejoindre une salle de jeu. Il a ensuite été nécessaire de faire des différents éléments qui composent un joueur un *prefab*, avec à l'intérieur le modèle 3D du joueur, sa caméra, ses scripts et ses effets. Il s'agit, comme son nom l'indique, d'un élément « préfabriqué ». Cet élément est ajouté à la scène à chaque fois qu'un joueur rejoint une partie, il est « instancié ». Photon adapte des fonctions déjà existantes de Unity pour qu'elles fonctionnent dans l'environnement multijoueur. C'est par exemple le cas pour créer une instance de joueur : la fonction de base est `Instantiate()` et celle de Photon est `PhotonNetwork.Instantiate()`. Ces fonctions prennent en général les mêmes paramètres que celles de base, avec parfois quelques paramètres en plus pour avoir plus de contrôles sur la session.

Une partie importante dans le développement de cette fonctionnalité est la résolution de conflits. En effet, il faut s'assurer qu'un joueur ne voit et ne contrôle que son vaisseau et pas celui d'un autre joueur. Il faut donc dans de nombreux scripts, notamment celui qui gère les déplacements du vaisseau, vérifier que celui-ci appartient au joueur avant d'effectuer le script. Dans le cas contraire, la vérification se faisant en chaque début de script, la suite du code n'est en général pas effectuée. La caméra a posé le plus de problèmes malgré que la solution soit simple, il fallait seulement la laisser désactivée dans tous les cas et l'activer uniquement lorsqu'elle appartenait au joueur. La propriété `PhotonView.IsMine` s'est avérée très utile dans ce cas.

Pour que des objets soient transmis à travers le réseau, il faut ajouter des composants du module Photon appelés *Photon View*. Il faut ensuite relier à ce composants les autres composants à synchroniser qui composent l'objet, tels que les *Rigidbody* qui permettent de leur appliquer de la physique. Des *Photon View* sont par exemple présents sur le joueur et les projectiles. Des éléments tels que les éléments immobiles de décor de la carte n'ont pas besoin d'être transmis à travers le réseau car ils sont communs à toutes les instances du jeu. Il n'y a aucun risque que deux joueurs voient ces éléments différemment dans leurs parties respectives, ils est donc inutile de gâcher des ressources dans la transmission de ces éléments.

3.9 Site web

Participants au développement :
— Mathieu GUÉRIN

Le **site web** est pour le moment très simple. Il n'est composé que d'une page principale présentant très brièvement le projet et d'une autre présentant les membres du groupe. Quelques captures d'écran du jeu permettent d'illustrer

à quoi ressemble le jeu à l'heure actuel.

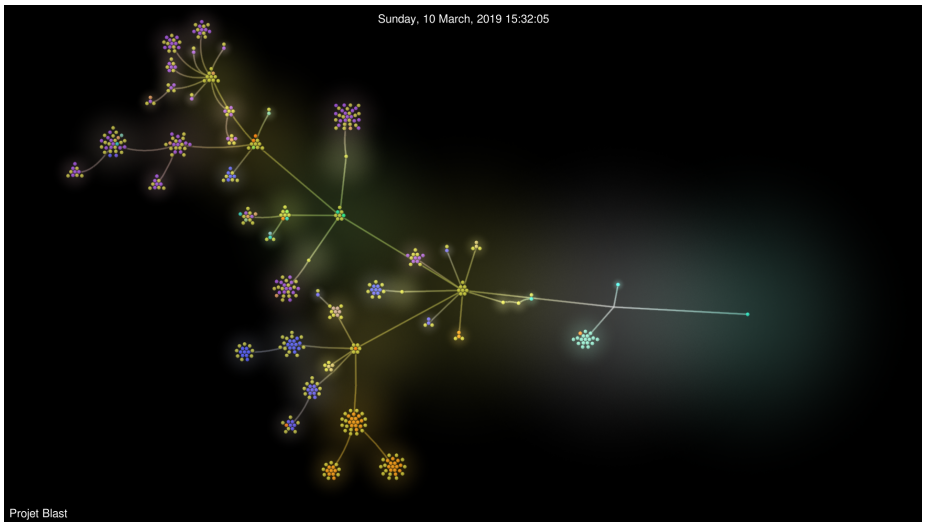
Le site internet est hébergé grâce à *Github Pages* et est accessible publiquement à cette adresse : `https://g00pix.github.io/ProjetBlast/`. Cette méthode est très accessible car la simple création d'une branche appelée « *gh-pages* » dans le repository du projet construit le site. Celui-ci se met automatiquement à jour lorsqu'un commit est effectué dans la branche.

Pour le moment, le site utilise la bibliothèque *Bootstrap* pour simplifier le développement et ajouter des éléments esthétiques sans avoir à perdre beaucoup de temps sur du code CSS qui n'est pas l'objectif de ce projet. Son utilisation est simple et bien documentée, il suffit d'ajouter des éléments HTML contenant les noms de classes correspondant pour que le site prenne forme.

Le développement du site se fait à l'aide de templates HTML qui seront remplies par *Github Pages* lorsque du contenu est ajouté. Le contenu s'écrit en *Markdown*, ce qui rend l'écriture plus simple. Il est possible d'ajouter des petites parties de code HTML dans les fichiers de contenu pour différencier certaines pages et y ajouter des éléments en plus.

4 Structure du repository

La structure du repository Git, c'est à dire du projet, est représentée par cet arbre ci-dessous. Cette visualisation a été générée parà l'aide de l'outil Gource à partir de l'historique de modification Git. Chaque branche représente un dossier, chaque élément représente un fichier et chaque couleur représente un type de fichier.



À l'extrême droite, on trouve la racine droite. 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 développés.
- À gauche, les deux grandes branches correspondent aux deux librairies principales que nous avons utilisé :
  - StarSparrow, un librairie avec les vaisseaux.
  - TextMeshPro, pour améliorer les textes de l'interface.
  - Photon, qui nous aide pour le multijoueur.

5 Expériences personnelles

Nicolas : Pour moi, la création d'un jeu vidéo est une grande première. En premier lieu, j'appréhendais l'idée de ce projet. J'ai lancé le développement en créant le repository Git et en faisant les premiers pas pour amorcer le travail du groupe. J'ai commencé par créer un environnement simple avec un vaisseau et un cube. J'avais déjà une vague idée de que je voulais faire. J'ai ajouté une skybox pour donner une impression d'être dans l'espace et j'ai commencé à expérimenter. Mon premier but était d'avoir un vaisseau contrôlable au clavier et la souris, avec une caméra à la troisième personne. Cette première fonctionnalité est très importante et a nécessité beaucoup de recherches car quelques aspects n'étaient pas toujours évidents. Ce fût de même pour la caméra. J'ai ensuite voulu me pencher sur le mutltijoueur car je savais que cela allait représenter une masse importante de travail et de recherche. Je me suis appuyé sur les recommandations d'autres élèves de niveau supérieur ainsi que celles des professeurs en m'y penchant dès le début du projet. Heureusement, beaucoup de documentation est disponible et apprendre à utiliser le module multijoueur *Photon* ne m'a pas posé de problème. Mathieu m'a rejoint dans la conception du jeu et a ajouté de nouvelles fonctionnalités. Ayant déjà quelques bases sur Git, ce début de projet m'a déjà permis d'améliorer ces connaissances et la collaboration sur cet outil m'est déjà plus familière.

Mathieu : La conception d'un jeu vidéo m'a toujours attiré, et ce depuis très jeune, mais je ne m'étais jamais engagé dans un réel projet de cette nature. Le projet de S2 était donc à mon sens l'occasion de découvrir les outils lié au développement d'un jeu vidéo et des méthodes de travail qui en découlent.
J'ai commencé à participer dans le développement du projet après Nicolas. En effet, n'ayant jamais utilisé Unity auparavant, j'ai commencé avoir avoir assisté à la conférence Unity de Gconfs, qui m'a permise de comprendre les bases de l'outil et de démarrer doucement à travailler sur notre projet. Git est un outil de collaboration que je maîtrisais déjà correctement et que j'utilisais depuis longtemps, alors j'ai pu commencer à travailler sur les nouvelles fonctionnalités que je souhaitais ajouter à la base existante que Nicolas avait conçue. Ma méthode de développement se base sur mon expérience de jeu en tant que joueur. Parmi les premières fonctionnalités que j'ai ajouté se trouve des éléments d'interactions et d'esthétique comme les projectiles tirés sur commande ou les traînées laissées par le vaisseau.

6 Avances et Retards

Par rapport au prévisions présentées par la cahier des charges, nous pouvons noter un retard dans :

— La gestion des sauvegardes
— La gestion des missions
— Le monde et la carte

Cependant, les bases et les idées sont déjà là. Toutes ces fonctionnalités seront améliorés et implémentés progressivement par la suite.

La gestion des missions implique le développement d'élément intelligents comme des ennemis, la création d'objectifs dans le jeu. Tous ces éléments demandent une interaction avec le joueur et une gestion de la progression que nous n'avons pas encore développé. Une fois ces éléments développés, la gestion des sauvegarde pourra être mise en place.

La création de missions et d'objectifs permettra aussi de remplir le monde d'éléments de gameplay, et aussi d'améliorer la gestion de la carte, qui n'est pas encore au point de manière pratique.

D'autres fonctionnalités sont même plutôt en avance par rapport à nos prévisions, comme l'interface ou le multijoueur, qui peuvent déjà permettre des modifications rapide au besoin.

7 Prévisions pour la suite

Pour finir, cette partie présentera rapidement un aperçu de ce que nous prévoyons de faire dans notre projet pour la suite et la prochaine soutenance.

Le retard sera rapidement rattrapé car nous avons déjà les idées et nous savons quelles méthodes utiliser en général pour les prochaines fonctionnalités que nous prévoyons de développer.

Gestion des missions : Le développement d'ennemis a déjà plus ou moins commencé, mais il faudra les doter d'une intelligence artificielle concrète avant de pouvoir les implémenter au jeu.

Gestion du vaisseau : Le vaisseau aura besoin de réserves pour fonctionner. Il aura par exemple un temps limité d'accélération rapide qui devra se recharger lorsque le vaisseau ira moins vite.

Objets du jeu : Des objets de jeu permettant de modifier des caractéristiques des éléments de vaisseau pourront être ajoutés. Certains objets pourront apporter des bonus et modifier certaines propriétés du vaisseau, comme par

exemple le temps de rechargement pour l'accélération rapide mentionnée précédemment.

Gestion des sauvegardes : La joueur aura la possibilité de sauvegarder la position et la configuration de son vaisseau, ainsi que les objets qu'il possède et ses statistiques. Sa progression dans les missions et objectifs devra bien sûr elle aussi être sauvegardée, un fois implémentée.

Interface : L'interface, visuellement simple pour le moment, pourra être améliorée. Les éléments qui la compose déjà seront mis à jour et de nouveaux pourront être ajoutés. Parmi les éléments existants :

— La boussole pourra afficher les futurs objets environnants, comme des éléments du décor ou des ennemis, ainsi que les objectifs assignés au joueur.
— L'interface pourra aussi afficher la liste des autres joueurs connectés ainsi que leur état.
— Les joueurs pourront choisir un pseudo lors de leur connexion, celui-ci apparaîtra au dessus de leur vaisseau dans leur partie en multijoueur.

L'interface générale pourra éventuellement aussi intégrer les fonctionnalités du multijoueur comme le chat. Un menu principal sera développé, celui actuellement présent n'ayant qu'un rôle de test et donc n'a pas pour but d'être très avancé. Ce nouveau menu, plus complet, contiendra les parties déjà existantes, c'est à dire un bouton pour le mode solo et un autre pour le mode multijoueur, mais aussi de nouveaux boutons tels qu'un menu d'options pour ajuster certains paramètres. Le menu sera aussi optimisé, comme par exemple afficher un écran de connexion pour éviter que l'utilisateur pense que son jeu a planté.

Monde et Carte : Le monde sera amélioré avec les missions, des éléments y seront donc ajoutés en même temps. Les joueurs pourra interagir avec certains de ces éléments, et d'autres resteront des élément dits « de décor ». Nous pensons par exemple à des planètes et des astéroïdes, mais aussi des grandes structures telles que des stations spatiales où le joueur devra se rendre pour effectuer des missions. La vue carte sera également bien sûr à améliorer, la base existante étant utile mais pas du tout représentative du résultat final puisque encore très limitée. Le joueur devra pourra se déplacer virtuellement et librement dans le mode de vue carte, de manière à ce qu'il puisse voir ce qu'il y a autour de lui, ainsi qu'en dessous, au dessus, et plus loin.

Site web : Le site sera à améliorer et à compléter. Son objectif sera de présenter le jeu, ainsi que de détailler et d'illustrer ses fonctionnalités. Des captures d'écrans seront par exemple ajoutées. Les membres du projet pourront aussi se présenter sur la page d'information sur le groupe. Le jeu sera disponible au téléchargement par la suite.

8 Conclusion

Malgré des débuts difficiles à cause de modifications dans la constitution du groupe, un retard pris à cause d'un début tardif et l'absence de travail d'un des membres, le jeu *Blast* a déjà bien commencé. Des éléments essentiels tels que le multijoueur sont déjà bien implémentés et de nombreux autres posent les bases de ce que nous prévoyons de faire par la suite. Ce projet est pour nous déjà quelque chose d'intéressant et nous prenons du plaisir à y travailler. Nous apprenons beaucoup à l'aide des nombreuses recherches que nous effectuons pour mieux connaître le moteur Unity mais nous améliorons également nos compétences en programmation, ce qui est le but principal de ce projet. Nous apprenons également à travailler en collaboration avec Git, un outil puissant qui nous sera utile dans notre futur. Par la suite, nous prévoyons de rattraper tout retard et d'approfondir les bases existantes. Nous cherchons également à ajouter un but dans le jeu que les missions ajouteront lorsqu'elles seront développées. Il y a encore du travail à fournir mais l'état actuel des choses est positif.