UNIVERSITÉ DU QUÉBEC À MONTRÉAL

RAPPORT DU SPRINT 1

PHILIPPE LEBLANC - LEBP11129502
BRYAN CÔTÉ - COTB05079000
YASSINE H. - HASY04089702

TRAVAIL PRÉSENTÉ
À
MONSIEUR HAFEDH MILI

DANS LE CADRE DU COURS
INM5151 PROJET D'ANALYSE ET DE MODÉLISATION

9 MARS 2020

Table des matières

1. Introduction

1.1 Objectifs

L'objectif de ce document est essentiellement de décrire et communiquer les différents avancements du projet en cours et les fonctionnalités présentement implantées et celles à venir. Ainsi, cette nouvelle approche a été implémentée en réponse aux différents manquements dans l'industrie du FPS. Ce document s'adresse au commanditaire de notre projet afin de transmettre, de façon transparente, une bonne compréhension des besoins des usagers. Il s'adresse aussi aux développeurs, tout comme aux usagers, pour s'assurer que le document décrit leurs besoins de façon fidèle et prendre en compte tout changement mineur potentiel, ou bien pour une restructuration des priorités du projet.

1.2 Vue d'ensemble du produit

L'objectif du système Alpha Strike vise à combler le besoin d'entraînement au tir (à l'arme à feu) qu'un utilisateur pourrait avoir avant qu'une partie de jeu officielle se lance. Il permet à l'utilisateur d'améliorer ses réflexes et la qualité de son tir, et ce peu importe l'environnement de jeu FPS. Cet entraînement pourrait s'appliquer de près ou de loin à n'importe quel jeu de tir présent sur le marché. Ceci est justement dû au sous-système « Sandbox » que notre jeu aura. Ce qui démarque Alpha Strike de ses compétiteurs est majoritairement le côté personnalisable des différentes variables d'environnements. On entend ici la vitesse du mouvement, le recul de l'arme, la gravité ou toute autre fonctionnalité permettant de recréer l'atmosphère d'un jeu précis.

1.3 Définitions, acronymes et abréviations

FPS : Le jeu de tir à la première personne ou en vue subjective, souvent appelée Doomlike et FPS, sigle pour l'expression anglaise first-person shooter, est un genre de jeu vidéo de tir fondé sur des combats en vision subjective (« à la première personne »), c'est-à-dire que le joueur voit l'action à travers les yeux du protagoniste.

Sandbox : Le gameplay non linéaire désigne dans un jeu vidéo la possibilité qu'ont les joueurs de compléter des épreuves selon des séquences variables. Chaque joueur peut compléter qu'une partie des épreuves possibles, lesquelles pouvant être jouées dans un ordre aléatoire. Inversement, un jeu vidéo avec un gameplay linéaire confronte le joueur à une séquence fixe d'épreuves : chaque joueur rencontre chaque épreuve, et ce, dans le même ordre.

Low poly : Le fait d'utiliser un faible nombre de polygones lors de la modélisation d'éléments 3D, ce qui a l'avantage de prendre moins de ressources machine en optimisant les performances, mais d'un autre côté en créant des textures beaucoup moins détaillées. Pour certains, le style de textures qui résulte de cette technique est attrayant autant au niveau de la performance que de l'esthétique.

1.4 Documents de références

Unity. https://unity.com/

Screen Tearing, https://en.wikipedia.org/wiki/Screen_tearing

1.5 Aperçu du document

Ce document passe en revue les différentes fonctionnalités du projet qui ont été mises en place par l'équipe durant le premier sprint de développement. Un rappel des différentes fonctions du jeu Alpha Strike sera d'abord effectué ainsi que la répartition de ces tâches dans les différents sprints. Les différents éléments implémentés dans le projet seront mis en lumière comme suit :

- Cas d'utilisation du sprint actuel et leur diagramme de séquence
- Modèle de classes
- Une revue du sprint actuel suivi d'un plan pour le sprint suivant

2. Description générale du logiciel

2.1 Vue d'ensemble des fonctions du produit

La première version qui sera développée comprendra une physique fonctionnelle de base. Un personnage pouvant se déplacer et sauter. La fonctionnalité de tirer avec une arme spécifique sera incluse dans cette version. L'environnement d'entraînement sera une chambre fermée avec des cibles simples. Un score sera calculé par rapport au nombre de cibles touchées selon le temps chronométré. Le joueur pourra aussi accumuler des crédits à chaque partie jouée selon un système de devise pour ainsi magasiner des personnalisations d'armes (« skins »).

Les fonctions développées dans Alpha Strike par catégorie seront les suivantes :

- Environnement :
  - Restreindre le joueur à un bac à sable fermé : l'environnement doit être simpliste et personnalisable au goût du joueur (parmi une liste de choix).
  - Avoir un système d'éclairage simulant le soleil : un objet dans la scène doit simuler l'éclairage du soleil et ainsi créer les ombrages rendant l'environnement plus vivant.
  - Avoir un mur avec des cibles : l'environnement doit être constitué au minimum d'un mur principal sur lequel des cibles apparaissent.
  - Ajouter les textures des différents composants de la carte : selon les goûts du joueur, une carte pourrait avoir un environnement futuriste (« sci-fi »), réaliste, loufoque ou même du genre dessin animé (« low-poly »).
  - Impacts de tirs : l'environnement sera impacté par les balles tirées par le joueur, par exemple par des textures de trous causés par les munitions.

- Fonctions utilisateur (joueur) :
  - Physique de base : le joueur doit être en mesure de se déplacer dans la carte, sauter et bouger sa vision à l'aide de la souris.
  - Tirer une arme : en cliquant sur le bouton gauche de la souris, le joueur doit pouvoir tirer son arme où il vise.
  - Recharger une arme : en appuyant sur la touche « R », le joueur peut recharger son arme avec un nouveau chargeur rempli de munitions.

- Fonctions de l’arme choisie :
    - Tirer un projectile selon une trajectoire : l’arme doit pouvoir tirer des munitions selon la trajectoire qui correspond à l’endroit où le joueur vise avec son réticule.
    - Se recharger : lorsque l’arme est rechargée par le joueur, une animation appuie l’action pour améliorer l’expérience utilisateur.
    - Effets de tir : les différentes armes du jeu ont des comportements différents lorsqu’elles sont tirées par le joueur incluant les effets de recul, sonores et visuels.
- Menu
    - Lancer une partie : lorsque cette option sera choisie, le joueur sera placé dans une instance du champ de pratique et il pourra s’exercer sur les cibles.
    - Consulter son meilleur score : le joueur aura l’option de voir son plus haut score selon le mode de jeu.
    - Magasin : ce menu permettra au joueur de consulter à la fois son nombre de crédits selon le système de devise du jeu et aussi acheter des « skins », parmi ceux qui s’affichent avec le nombre de crédits possédés, pour personnaliser ses armes.

2.2 Contenu des sprints

Tout d’abord, le contenu du premier sprint a servi majoritairement à se familiariser avec l’environnement de développement pour Alpha Strike, c’est-à-dire Unity. Une recherche plus approfondie a permis dans un premier temps de trouver quelques librairies utiles à l’implémentation. Ensuite, un environnement simpliste de tir a été développé par l’équipe afin de recréer un univers sandbox voulu pour la zone de tir. Finalement, les déplacements du joueur ainsi que sa capacité de tirer ont pu être ajoutés au jeu.

Pour le deuxième sprint, il est prévu de voir apparaître des cibles pouvant être tirées par le joueur, ainsi que l'effet de recul sur les armes à feu, afin d'y ajouter un aspect plus réaliste. La partie plus optionnelle de ce sprint sera orientée vers le côté texture de l'environnement et sur le détail des armes, tant par leur couleur que par les types d'armes disponibles pour l'utilisateur.

Pour le troisième et dernier sprint, un système de devise devrait être mis en place, afin de permettre au joueur d'avoir une certaine progression dans l'application. Cette nouvelle monnaie pourrait lui permettre d'acheter à la fois de nouvelles armes dans le magasin du jeu, mais aussi lui permettre d'acheter des skins d'environnement pour le sandbox et ses armes. Finalement, un menu de jeu sera mis en place afin que l'utilisateur soit en mesure de naviguer entre une partie de tir et le magasin du jeu.

3. Contenu du sprint actuel

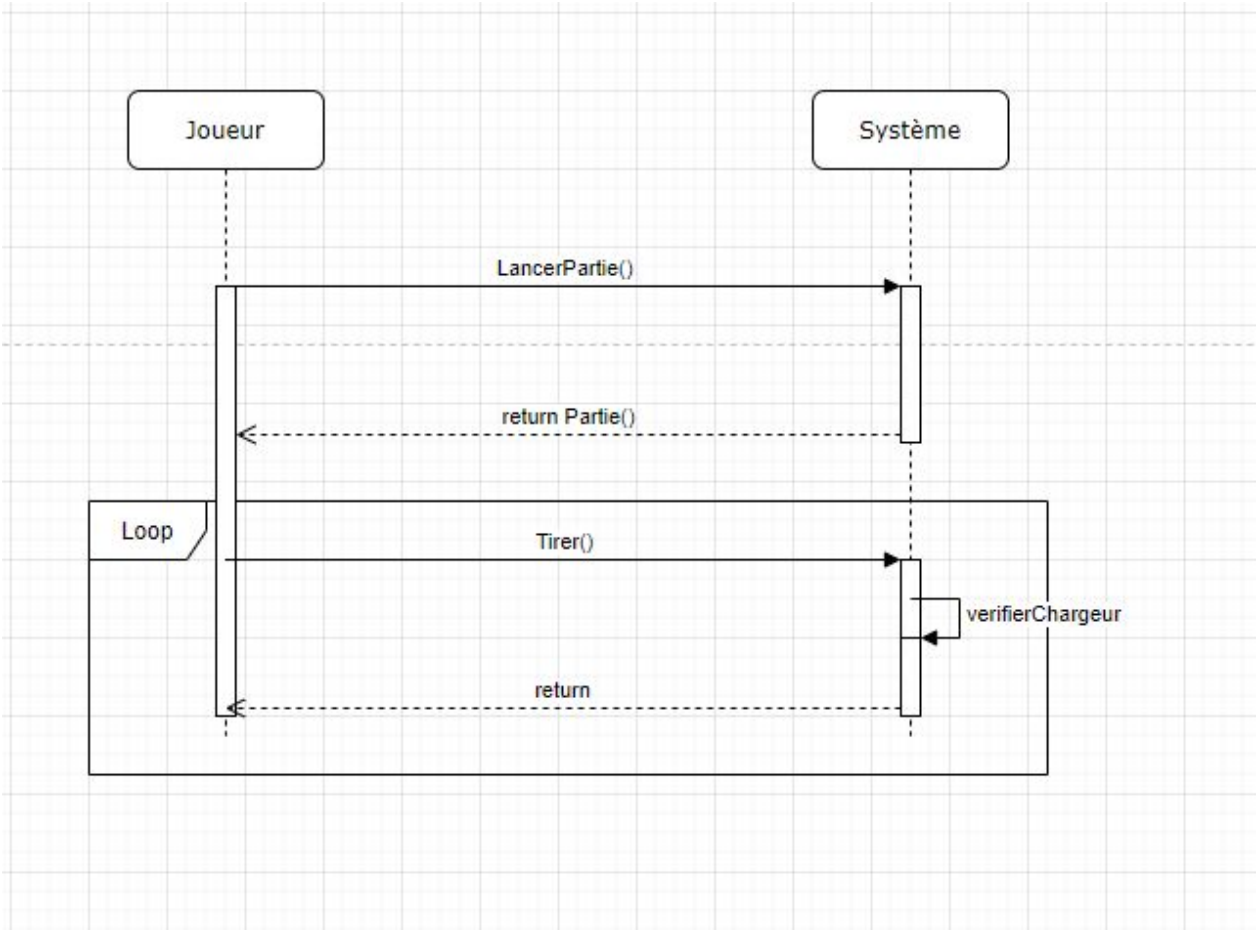
3.1 Cas d'utilisation implantés dans le sprint

Cas d'utilisation 1

Nom	Tirer
Niveau	But utilisateur
Acteurs	Usager
Pré conditions	L'usager se trouve dans le sandbox de tir
Post conditions	Un impact de balle est observable à l'endroit visé par l'usager

Scénario principal	1. Le système charge et lance la partie 2. Le système affiche aléatoirement des cibles sur la carte de jeu (prévu pour le sprint 2) 3. L'utilisateur, en appuyant sur la touche gauche de sa souris (par défaut), envoi des projectiles vers une cible 4. Le système vérifie l'impact de la balle tiré et retourne l'effet visuel au joueur
Scénarios alternatifs	4a. Le joueur ne possède plus de balles dans son chargeur 4.a.1 Le système détecte que l'arme n'a plus de munition 4.a.2 L'usager ne peut plus tirer
Fréquence d'occurrence	100 fois par partie
Autres commentaires/exigences	

Diagramme de séquence

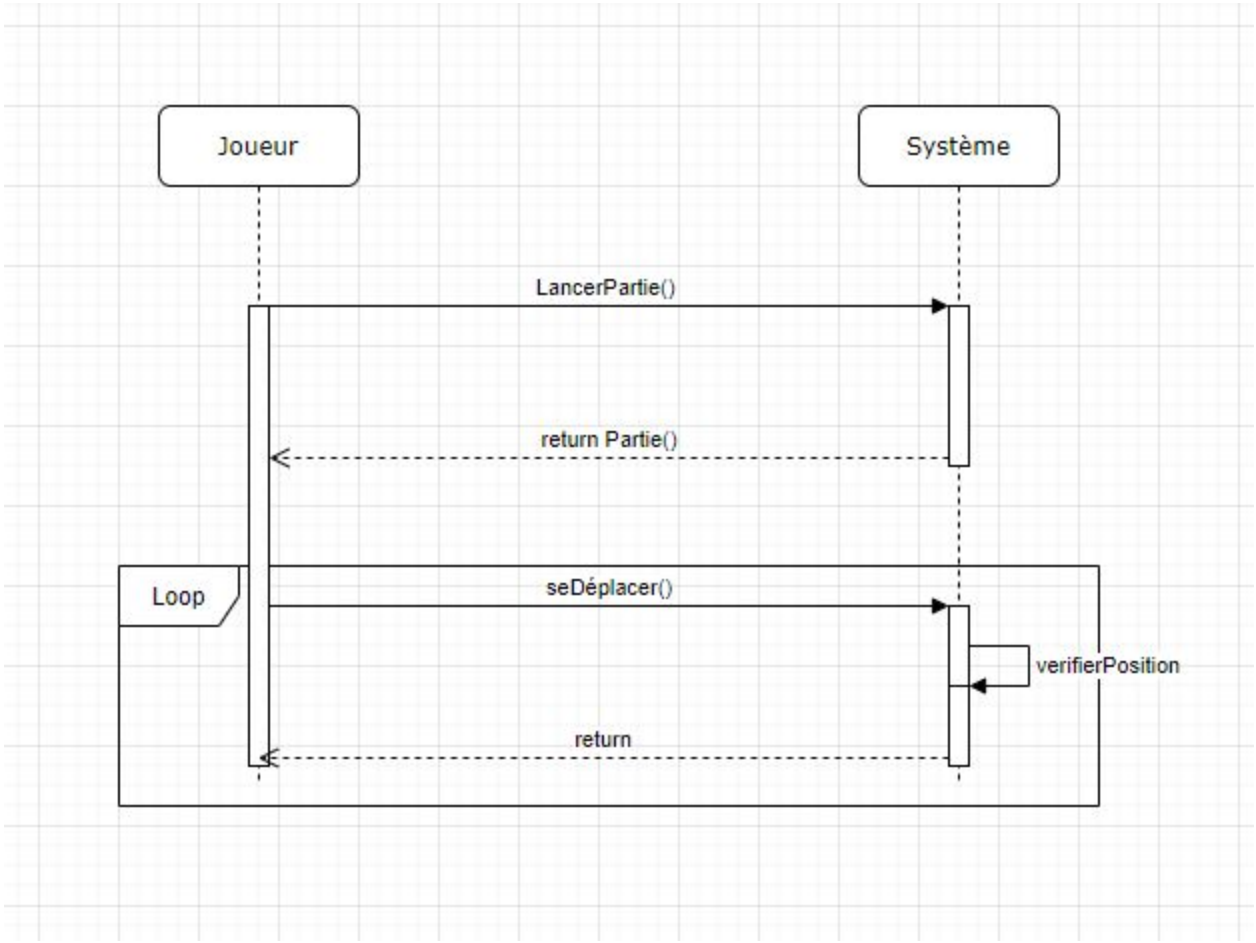


Signature	void update()
Description	Tire
Références	Cas d'utilisation Tirer (Étape 3)
Entrées	● Le nom du paramètre = ClicGaucheSouris ● Touche gauche de la souris du joueur
Sorties	Impact de balle sur la cible visée par le joueur
Pré-conditions	Le joueur est dans l'environnement de tir Le joueur possède une arme L'arme possède des munitions
Post-conditions	● Création ou destruction d'objets

Cas d'utilisation 2

Nom	Se déplacer sur la carte
Niveau	But utilisateur
Acteurs	Usager
Pré conditions	L'usager se trouve dans le sandbox de tir
Post conditions	L'usager est déplacé
Scénario principal	1. Le système charge et lance la partie 2. L'usager, à l'aide de de sa souris, ainsi que les touches a, w, s,d et espace de son clavier, tente de se déplacer 3. Le système analyse la touche appuyée par l'utilisateur 4. L'usager est déplacé
Scénarios alternatifs	4a. Le système constate que l'usager se retrouve dans l'impossibilité de se déplacer 4.a.1 L'usager n'est pas déplacé
Fréquence d'occurrence	500 fois par partie
Autres commentaires/exigences	

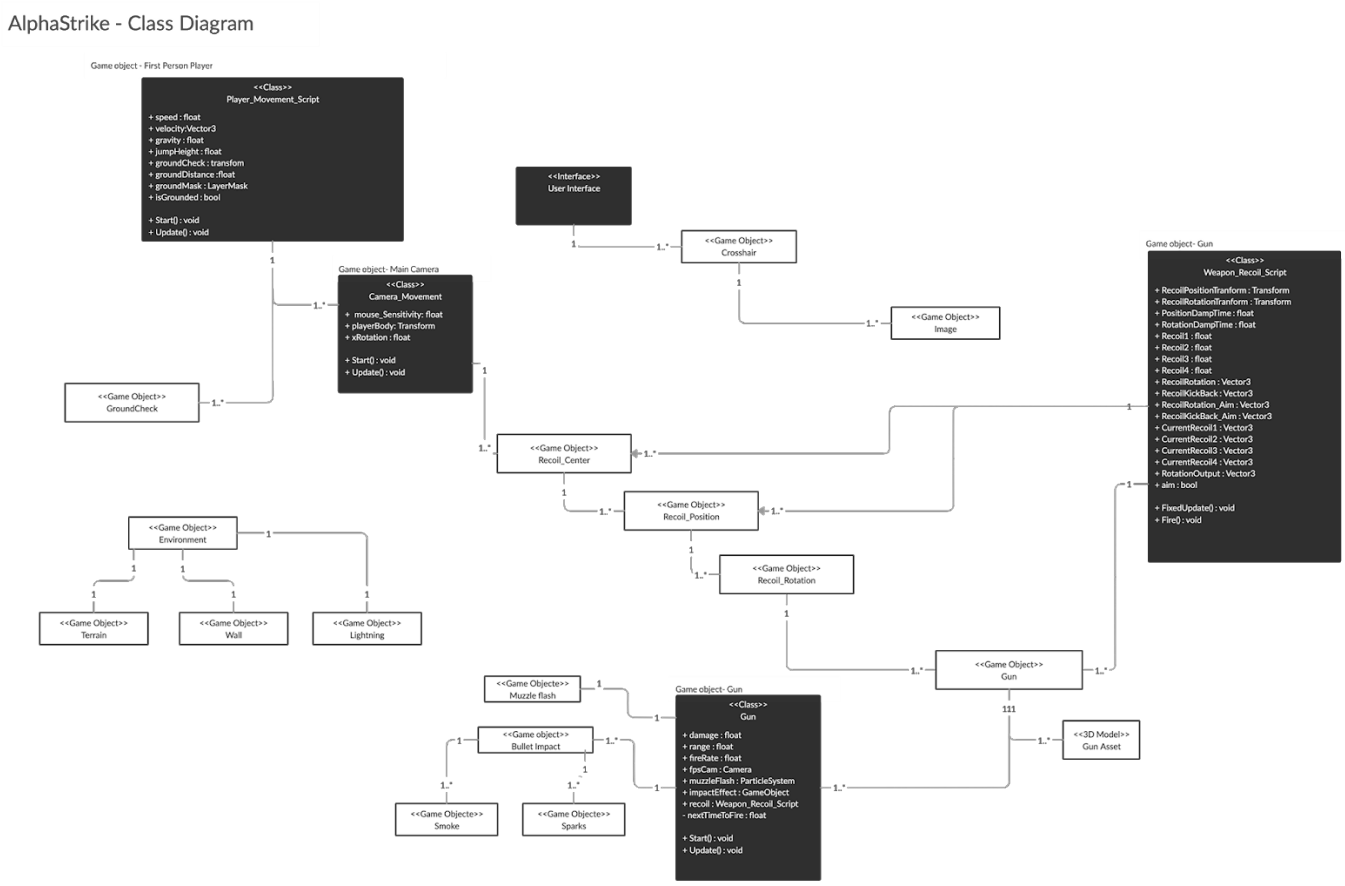
Diagramme de séquence



Signature	Void Update() - seDéplacer()
Description	Déplace le joueur dans la direction désirée
Références	Cas d'utilisation Se déplacer sur la carte(Étape 3)
Entrées	● Le nom du paramètre = ToucheDuClavier ● La touche entrée par l'utilisateur dans le but de se déplacer
Sorties	La nouvelle position de l'usager
Pré-conditions	Touches du clavier A,S,W,D par défaut
Post-conditions	● Impact de balle sur la cible du joueur dans l'environnement de tir ● Décrémente le nombre de balle dans l'arme du joueur

3.2 Modèle de classes

3.2.1 Diagramme de classes



3.2.2 Description des classes

Nom	Player_Movement_Script			
Description	La classe “Player_Movement_Script” gère les déplacements du joueur.			
Attributs				
1	**Nom**	speed	**Type**	float
	Description	Contrôle la vitesse de déplacement du joueur.		

2	**Nom**	velocity	**Type**	Vector3
	Description	Variable représentant une vélocité en 3 dimensions		
3	**Nom**	gravity	**Type**	float
	Description	représente l'accélération cause par la gravite (-9.81m/s ** 2)		
4	**Nom**	jumpHeight	**Type**	float
	Description	Hauteur d'un saut		
5	**Nom**	groundCheck	**Type**	transform
	Description	sert à détecter si le joueur touche le sol en collectant certaine informations à l'aide de la class transform pour ainsi réinitialiser la vitesse (vers le bas) dû à la gravité.		
6	**Nom**	groundDistance	**Type**	:float
	Description	distance du rayon de detection du sol par rapport au pied du joueur		
7	**Nom**	groundMask	**Type**	LayerMask
	Description	represente tout les éléments qui sont considérés comme etant un sol (regrouper dans un même nom de mask)		
8	**Nom**	isGrounded	**Type**	bool
	Description	booléen déterminant si le joueur est effectivement sur le sol ou non.		

Nom	Camera_Movement			
Description	La classe "Camera_Movement" gère les mouvement de la camera.			
Attributs				
1	**Nom**	speed	**Type**	float
	Description	Contrôle la vitesse de mouvement de la caméra dépendamment de la sensitivité de la souris.		
2	**Nom**	playerBody	**Type**	transform
	Description	modifie l'orientation du joueur en concordance avec les mouvement de caméra en modifiant des paramètre dans (playerBody).		
3	**Nom**	xRotation	**Type**	float
	Description	Sert de limite pour que la caméra ne fasse pas de rotations plus élevée que 90 degree sur l'axe x 3D.		

Nom	User Interface
Description	La classe "User Interface" pour le moment, ne fait pas grand chose excepté afficher un viseur au centre de l'écran.
Attributs	L'image du viseur.

Nom	Gun			
Description	La classe "Gun" gère l'action de tirer l'arme			
Attributs				
1	**Nom**	damage	**Type**	float
	Description	le niveau de dégât que l'arme fera		

2	**Nom**	range	**Type**	float
	Description	la distance maximale à laquelle l'arme peut tirer pour atteindre des cibles		
3	**Nom**	fireRate	**Type**	float
	Description	la cadence de tir de l'arme		
4	**Nom**	fpsCam	**Type**	Camera
	Description	l'objet dans la scène qui sert de caméra (vision première personne) du joueur		
5	**Nom**	muzzleFlash	**Type**	ParticleSystem
	Description	l'effet au bout du canon lorsque le joueur tire		
6	**Nom**	impactEffect	**Type**	GameObject
	Description	l'effet des impacts de munitions sur les objets tirés		
7	**Nom**	recoil	**Type**	Weapon_Recoil_Script
	Description	référence au script (classe) qui gère le recul de l'arme		
8	**Nom**	nextTimeToFire	**Type**	float
	Description	valeur qui permet ou non au joueur de tirer selon la cadence de tir		

Nom	Weapon_Recoil_Script			
Description	La classe "Weapon_Recoil_Script" gère l'effet de recul sur l'arme lorsque le joueur tire			
Attributs				
1	**Nom**	RecoilPositionTranform	**Type**	Transform
	Description	Sert à faire les transformations de position de l'arme		
2	**Nom**	RecoilRotationTranform	**Type**	Transform
	Description	Sert à faire les transformations de position de l'arme		
3	**Nom**	PositionDampTime	**Type**	float
	Description	Sert à faire les transformations de position de l'arme		
4	**Nom**	RotationDampTime	**Type**	float
	Description	Sert à faire les transformations de position de l'arme		
5	**Nom**	Recoil1 … Recoil4	**Type**	float
	Description	Sert à faire les transformations de position de l'arme		
6	**Nom**	RecoilRotation	**Type**	Vector3
	Description	Sert à faire les transformations de position de l'arme		
7	**Nom**	RecoilKickBack	**Type**	Vector3
	Description	Sert à faire les transformations de position de l'arme		
8	**Nom**	RecoilRotation_Aim	**Type**	Vector3
	Description	Sert à faire les transformations de position de l'arme		
9	**Nom**	RecoilKickBack_Aim	**Type**	Vector3
	Description	Sert à faire les transformations de position de l'arme		
10	**Nom**	CurrentRecoil1 … CurrentRecoil4	**Type**	Vector3

	Description	Sert à faire les transformations de position de l'arme		
11	**Nom**	RotationOutput	**Type**	Vector3
	Description	Sert à faire les transformations de position de l'arme		
12	**Nom**	aim	**Type**	bool
	Description	Ce paramètre sert à modifier le type de transformation qui sera effectué à l'arme selon le type de visée, par exemple visée éloignée (mitraillette) ou visée rapprochée (fusil sniper)		

3.2.2 Description des associations

Nom		Associations simples		
Description		Le lien entre la classe "gun" qui gère l'arme aux objets de la scène qui l'utilisent		
1	**Entité source**	Gun << classe >>	**Entité destination**	MuzzleFlash << game object >>
2			**Entité destination**	BulletImpact << game object >>
3			**Entité destination**	Gun << game object >>

Nom		Associations simples		
Description		Le lien entre la classe "Weapon_Recoil_Script" qui gère l'effet de recul sur l'arme et les objets de la scène qui l'utilisent		
1	**Entité source**	Weapon_Recoil_Script << classe >>	**Entité destination**	Recoil_Center << game object >>
2			**Entité destination**	Recoil_Position << game object >>
3			**Entité destination**	Gun << game object >>

Nom		Associations simples		
Description		Le lien entre la classe “Player_Movement_Script” et groundcheck qui détecte si le joueur est sur le sol.		
1	**Entité source**	Player_Movement_Script << classe >>	**Entité destination**	groundcheck << game object >>

Nom		Associations simples		
Description		Le lien entre la classe “Camera_Movement” et recoil_center qui qui prend en considération tout les mouvement dû au recul de l’arme.		
1	**Entité source**	Camera_Movement << classe >>	**Entité destination**	recoil_center << game object >>

4. PROCHAIN SPRINT

4.1 Revue du sprint

Revue technique (contenu du *sprint review*)

- Ce qui a été complété :
  - La base de l’environnement de tir est complétée
  - Le joueur est présent dans le champ de tir
  - Celui-ci peut se déplacer sur la carte
  - Il peut changer sa vision à l’aide de sa souris
  - Le joueur possède une arme et peut l’utiliser pour tirer
  - L’arme possède une mire aidant le joueur à savoir où il vise
  - L’arme effectue un recul lorsqu’un joueur tire
- Ce qui a bien fonctionné :
  - Les jeux de test sur l’application étaient très faciles
  - Avoir accès à de la documentation sur Unity et sur des projets semblable au

nôtre

- Ce qui n'a pas bien fonctionné :
    - Naviguer dans l'éditeur Unity n'était pas très intuitif
    - Ajouter des éléments dans un projet était souvent complexe sans créer des bogues ailleurs dans le répertoire de travail

- Obstacles rencontrés et les solutions adoptées :
    - Comme développer un projet en Unity était une première pour tout le monde, nous avons eu de la difficulté à savoir comment commencer un projet dans cet environnement. Pour remédier au problème, nous avons consulté quelques tutoriels afin de nous familiariser avec l'environnement et de comprendre les différents outils à notre disposition.
    - La collaboration sur git entre les différents membres de l'équipe a causé problème sur le versionnage du projet entre un système d'exploitation Windows et macOS. Pour régler ce problème, nous avons utilisé l'outil Unity Collab disponible avec Unity qui nous permet d'avoir une simulation de git au sein de l'application.

Revue gestion (contenu du *sprint retrospective*)

- Évaluer les personnes
    - Bonne collaboration, division du travail.
    - Projet modulaire donc aucun conflit de merge request
- Bon déroulement
    - Accès facile à de la documentation, des tutoriels
    - Tester immédiatement de nouvelles fonctions implémentées
- Améliorations potentielles
    - Avoir des objectifs hebdomadaires à respecter
    - Communication plus fréquente entre les membres de l'équipe

- Créer un plan pour la mise en œuvre/adoption de ces améliorations
    - Ajouter les rencontres comme tâche à nos sprints

4.2 Plan pour prochain sprint

Pour le deuxième sprint, voici les tâches prévues pour le groupe :

- Ajout de textures à l'environnement
- Implémentation des cibles à tirer
- Ajout de détails sur les armes
- Animation d'impact sur les objets tirés par l'arme du joueur
- Possibilité de recharger son arme

5. ANNEXES

Unity Collab