https://github.com/RallardM/Autoshooter

https://trello.com/b/SdhxLW6F/autoshooter

Autoshooter Maurice et Rémi,

Différemment à ce qui était demandé, les accès au outils de débogage se fait par les touches F1 (Menu Level Up), F2 (Menu Game Over) et F5 (Level Up au de 5 niveaux).

Les destructions d’objets se font en sortie de la ‘main loop’ du jeux et dans le destructeur de Game. Il est possible que cette partie ne soit pas encore complétée à la remise.

La pause se fait dans la ‘main loop’ à l’aide d’un booléen, ainsi que les menu de level up et Game Over (un enum pour chaque menu) .

Nous avons ajouté quatre singletons dans le but d’alléger la classe Game : **CameraManager**, **CollisionManager**, **MenuManager** et **GameObjectPool**.

**GameObjectPool** sert à alléger la classe Game, il a également servit à tenter un pool d’objet sur la branche pool\_object. J’avais un bug avec les projectiles qui changeaient tous en même temps, j’ai donc abandonné l’idée par manque de temps.

**CameraManager** s’occupe des appels aux méthodes servants à informer certaines classes sur la position et les propriétés de la caméra. Cinq classes ont un accès friend : Game, Player, Enemy (le spawn), MenuManager et Projectile (les rebonds sur le bord de l’écran).

**CollisionManager** s’occupe des détection de collisions. Par contre, nous avons remarqué que la logique n’a pas été fait en tenant compte de la symétrie des arguments. Enemy, Player et ExperienceOrb lui sont friend et y accède.

**MenuManager** s’occupe d’afficher les menus de level up et GameOver.

**Entity:GameObject** :Nous avons ajouté une classe abstraite Entity:GameObject pour avoir des méthode virtual seulement utilisées sur Enemy:Entity et Player:Entity qui sont GetHealth() et GetMaxHealth().

**ExperienceOrb**:**GameObject** est le blueprint pour les orbes d’expériences. Sous forme de d’objet on peut plus facilement contrôler leur destruction.

**Weapon:GameObject** : Classe abstraite des armes pour donner seulement aux armes les méthodes d’armes qui étaient déjà présente : Fire(), GetPosition(). Ainsi que les méthode protected : IncreaseRate(), IncreaseProjectileDamage() et IncreaseProjectileSize().

**HandGun:Weapon**, **ExplosiveGun:Weapon** et **LaserGun:Weapon** ont été mis en classe dans l’optique de pouvoir les ajouter et les retirer du joueur, mais le retrait n’a pas été implémenté durant le game play.

**Projectile:GameObject**. Nous l’avons gardé tel quel en essayant d’implémenter le struct des projectile qui été déjà là au départ. Les projectile sont attribué leurs propriétés uniques dans les armes appropriées en utilisant le m\_weaponInfos.m\_projectileInfos mit à notre disposition.

**Enumerations.h** est un fichier global ou se situe les Enums du jeux.

La classe **MathUtils** sert aux fonction mathématiques et les magie Numbers mathématique tel que les fraction sous forme de multiplication.

Il est possible que le temps nous ai manqué pour mettre à jour notre **diagramme de classe**. Celui de base est accompagné à la source de notre projet avec tous les itérations de 001 et plus.

Maurice a ajouté la distribution des tâches dans le **README.md** à la source du projet.