Third Person Shooter Multi

Milestones

Version	Deadline	Contraintes
Gold	04/12/2020 à 17h30	Unreal Engine 4.25.3 / Perforce

Nombre d'étudiants par groupe : 2
Rendu :

- Fichiers : binaires (dans nom_du_projet\nom_du_groupe\BIN)
- Sources : Perforce

Résumé

Le projet est un shooter à la 3ème personne jouable à plusieurs en réseau sur Unreal Engine 4.25.3, plus précisément un mode team deathmatch / horde. L'équipe bleue contre l'équipe rouge. Le projet doit être réalisé à partir du template ShooterMulti fourni.

Features gameplay du template

Apparence :
- Le joueur contrôle le mannequin présent dans le projet de base, il est par défaut dans l'équipe bleue.
- Le personnage joueur a toujours l'arme "rifle" équipée.

Spawn :
Des points de spawn sont placés dans la map. Le joueur spawn à un endroit aléatoire. Quand le joueur spawn ou respawn, il est invincible pendant quelques secondes.

Character / Contrôles / Caméra :
Les 3C contiennent les mécaniques suivantes :

- Déplacements : le joueur se déplace dans toute les directions en regardant toujours dans la direction où il vise (strafe)

- Visée: Lorsque le joueur presse la gâchette gauche ou le clic droit de la souris, il passe en mode visée, il est un peu plus lent et son set d'animation est différent.
- Sprint: Lorsque le joueur presse SHIFT ou le bouton X sur le pad, il sprinte (à une vitesse constante sans viser, le joueur ne peut que tourner. S'il tire ou vise, il s'arrête automatiquement de sprinter)
- Saut: Lorsque le joueur presse ESPACE, il effectue un simple saut avec un léger air control
- Tir: Lorsque le joueur le clic gauche ou gachette droite sur la manette, il tire en rafales continues. Il s'arrête de tirer quand il relâche le bouton ou quand il n'a plus de munition dans son clip (dans ce cas il recharge automatiquement). Le joueur peut viser chaque partie des joueurs adversaires (headshot = mort instantanée)
- Recharge: Si le clip de munition du joueur n'est pas plein et que le joueur appuie sur R ou bouton A de la manette, le joueur recharge son rifle
- Coup de poing: Si le joueur appuie sur F ou sur le bouton RB de son pad, il effectue un coup de poing, l'animation n'est jouée que sur la partie haute du corps et le joueur peut se déplacer pendant ce temps. Le coup poing blesse les joueurs adverses touchés.
- Interaction bouton : Si le joueur appuie sur la touche E du clavier ou le bouton B de son pad, il actionne un bouton, l'animation de bouton est jouée même s'il n'y a pas de bouton. S'il y a un bouton devant, il est actionné.
- Dégâts: lorsque le joueur essuie des tir ennemis ou un coup de mêlée, un feedback visuel lui indiquer qu'il est touché et sa vie baisse
- Mort : quand le joueur meurt, il passe en ragdoll pendant quelques secondes avant de respawner à un endroit aléatoire

Caméra :

- Elle suit le joueur contrôlé. Elle collisionne avec l'environnement à l'aide d'un spring arm.
- Quand le joueur vise, la caméra présente un field of view plus petit
- Quand le joueur tire, il subit des shakes de caméra
- Quand le joueur sprinte, il y a un léger shake de caméra

Rifle :

- Le rifle possède une capacité maximum totale et par magasin (clip). Ex : 20 et 5. Dans ce cas le rifle ne peut pas contenir plus de 20 munitions, et le joueur peut tirer 5 munitions avant de devoir recharger.
- Le rifle tire automatiquement à une cadence choisie par le développeur (doit recharger quand le clip est vide)
- A chaque tir, un FX de muzzle (éclair) et de fumée sort du canon, ainsi qu'une lumière éclairant à l'instant du tir.
- Chaque tir doit est dirigé vers le point visé par le joueur, avec un degré de dispersion random de forme conique. La dispersion est plus petite quand le joueur vise.
- Chaque impact de tir donne lieu à un FX de fumée.

Pickups :

- Des points de spawn de munitions ou de medikits sont placés dans la map.
- Ils spawnent régulièrement un “pickup”, choisi aléatoirement entre un pack de munitions ou un medikit (si le pickup précédemment spawné sur le point n’a pas été déjà ramassé).
- Le joueur qui passe près du pickup le ramasse automatiquement (il n’est alors plus disponible pour les autres joueurs).
- Dans le cas du pack de munition, le nombre de munitions est ajouté à sa quantité actuelle. Le joueur ne peut pas ramasser les munitions s’il est déjà au max.
- Idem pour le medikit, la vie correspondante est ajoutée à la jauge de vie du joueur et ne peut pas dépasser sa valeur maximale.

Horde :

- Plusieurs boutons sont placés dans la map
- En face du bouton, un ennemi (zombie) est spawné régulièrement et automatiquement.
- Cet ennemi est un mannequin doté d’un comportement simple : il se dirige vers le joueur le plus proche
- Quand il est suffisamment proche du joueur, il le frappe avec un coup de poing à intervalles réguliers
- Il subit les dégâts des joueurs (tir et coup de poings) et tombe en ragdoll pendant quelques secondes avant de disparaître.
- Le nombre total d’IA ne dépasse pas un nombre prédéterminé
- Par défaut, les zombies sont blancs et attaquent les joueurs des 2 camps
- Si un joueur presse le bouton, le spawner ne spawn plus que des zombies de la couleur du joueur, qui attaquent uniquement le camp opposé, pendant un certain temps (i.e. au bout d’un certain temps, le spawner se remet à spawner des zombies blancs qui attaquent tout le monde)

Scoring :

- Lorsqu'un joueur meurt (que ce soit par un zombie ou un joueur adverse), cela fait un point pour l’équipe adverse
- Lorsqu'une équipe atteint un nombre de point fixé par le développeur, la partie est terminé, le texte “Red/Blue win !” est affiché pendant X secondes et la map est rechargée (nouvelle partie, score reseté)

HUD :

- Le score est affiché constamment à l’écran (“Red : 3 Blue : 1”)
- Le nombre de munitions et la quantité de munition dans le clip est affichée de façon constante à l’écran

Features obligatoires

Réplication gameplay :

Character
L'ensemble des actions des characters sont à répliquer en réseau. Le déroulement des actions du jeu doit être cohérent sur chaque client.
Sont à répliquer :

- les animations de déplacements et leur vitesse (marche, course, sprint)
- l'orientation
- la visée
- le tir ainsi que les FX/SFX associés
- les coups de mêlée
- le saut
- l'interaction avec un bouton
- la mort et le respawn

Une fois le character animé en ragdoll, son comportement physique exact n'a pas besoin d'être répliquée fidèlement (collisions désactivées)

IA
Le spawn et le comportement des IA doivent être cohérents sur chaque client

Pickups
Les spawns de pickups ainsi que leur disparition doivent être cohérents sur chaque client.

Team
La couleur d'équipe de chaque character doit être cohérente sur chaque client. Cela concerne à la fois les joueurs et les IA activées par le bouton d'interaction.

Cubes :
Vous devez placer plusieurs cubes dans le niveau. Ces cubes sont physiqués en réseau, cela signifie que n'importe quel joueur peut bouger le cube en collisionnant avec ou en tirant dessus, et le résultat doit être répliqué chez tous les clients.

Lobby :

Vous devez créer une map de lobby qui permettra aux joueurs de rejoindre le jeu, constituer une équipe et lancer la partie.
Le lobby proposera les features suivantes :

- rechercher et lister les parties en cours
- sélectionner une partie à rejoindre
- une fois une partie sélectionnée, chaque joueur pourra choisir sa couleur d'équipe (rouge ou bleue), un nom et d'indiquer s'ils sont prêt à jouer
- si tous les joueurs sont prêt, la partie sera lancée en ouvrant la map gameplay "Highrise"
- les infos de chaque joueur doivent être conservées

Vous utiliserez le système de **sessions** pour créer, lister et rejoindre les parties. L'utilisation du plugin "Advanced Session" est autorisé.
Les parties seront créées avec un **serveur dédié** auquel les clients seront connectés.
Dans un premier temps vous testerez avec l'option éditeur "Play as Client" puis en lançant un serveur dédié en ligne de commande (voir section "liens utiles").
Idéalement, à terme, vous réaliserez une build serveur séparée.

La partie "Lobby" (GUI, système de session…) peut être faite entièrement en Blueprint.

Features bonus

Une fois toutes les features précédentes implémentées avec succès, vous pouvez vous pencher sur un ou plusieurs points ci-dessous.

Gameplay

- support du mode listen server en plus du mode serveur dédié, depuis le lobby
- afficher la liste des joueurs connectés et leur équipe in-game
- afficher des messages à l'écran indiquant aux joueurs les différentes actions de la partie (joueur tué par un autre, nombre de kill restant avant la victoire d'une équipe, enchaînement de kills sans être tué…)
- ajouter l'utilisation de capacités spéciales permettant d'augmenter temporairement les statistiques de votre joueur (puissance de feu, vitesse, précision…). Ces boosts de capacité doivent afficher un feedback visuel et sonore sur les différents clients et ne pas créer d'incohérences réseau
- ajout d'une arme lente avec dégâts de zone et projection physiquée (type rocket launcher)

Epic Online Services

Explorez et utilisez la nouvelle suite de services online d'Epic Games.
Vous devrez dans un premier temps effectuer l'intégration de l'API C++ dans votre projet.
Vous pourrez ensuite gérer un ou plusieurs des aspects suivants :

- Connexion à l'aide du compte Epic Games
- Utilisation de la liste d'amis
- Matchmaking
- Statistiques
- Achievements
- ...

Résolution des tirs par "Lag Compensation"

Inspirez vous de techniques de compensation de lag ("Valve"s lag compensation" ou Sniper Riffle d'UT4) pour implémenter un système de résolution de tirs touchés / manqués lorsque le lag est important.

Réconciliation serveur par "Entity Interpolation"

Implémentez un système d'interpolation de déplacement server-side (type Overwatch).
Voir https://www.gabrielgambetta.com/client-server-game-architecture.html
!! Attention, cette partie nécessite la création d'un décalage temporel client-server et une importante refonte du fonctionnement du CharacterMovementComponent !!

- **Vous avez carte blanche pour modifier le code et les réglages du template (architecture, nomenclature, optimisations…) à condition que cela reste dans le cadre du sujet**
- **Tout doit être intégralement codé en C++ à l'exception de la GUI (menus, HUD…)**
- **Toutes les features doivent être sécurisées dans le sens où le serveur doit être décisionnaire (pour ce qui est important).**
- **Vous devez aussi bien faire attention à ce que du code inutile ne soit pas exécuté sur les répliques.**
- **A l'exception du plugin AdvancedSessions et de l'API EOS, vous utiliserez exclusivement les fonctionnalités natives d'Unreal**

Liens utiles

Programmation multi Unreal :
https://docs.unrealengine.com/
https://www.youtube.com/watch?v=TbaOyvWfJE0&list=PLQzIKg3Bi-7uB-vKCqSgLppcp3NFkZHCM
http://cedric-neukirchen.net/Downloads/Compendium/UE4_Network_Compendium_by_Cedric_eXi_Neukirchen.pdf
https://www.unrealengine.com/en-US/blog/network-tips-and-tricks
Advanced Sessions :
https://forums.unrealengine.com/community/community-content-tools-and-tutorials/41043-advanced-sessions-plugin
Dedicated Servers :
How To Test Dedicated Server Games Via Commandline - Epic Wiki
Dedicated Server Guide (Windows & Linux) - Epic Wiki
Epic Online Services
https://dev.epicgames.com/en-US/services
https://github.com/gaslightgames/UE4_EOS_Plugin
Lag compensation & interpolation
https://www.gabrielgambetta.com/client-server-game-architecture.html
https://developer.valvesoftware.com/wiki/Lag_compensation
https://www.youtube.com/watch?v=2Hi3GeCBpd0&feature=youtu.be&ab_channel=%E9%BB%84%E6%96%87%E6%B7%BC

Rappels

Perforce

Vous utiliserez perforce pour gérer le versionning de votre projet

Emplacement des ressources

\\sirius.isart.lan\JeuVideo\P_CLASSE\GP_2022\1_Supports_Cours\Unreal Multi

Questions

Pour toute question :

- Venir me voir
- Me contacter sur Discord
- Envoyer un e-mail à f.wolf@isartdigital.com