SYNDICAT QUÉBÉCOIS DE LA CONSTRUCTION SIMULATOR 2023:
Analyse Préliminaire

Par
Louis-Charles Gaumond
Jonathan Trottier
Claudel D. Roy
Frédéric Léger
Kyle Lussier
Mathieu Duval
Marc-Éric Martel

Travail remis à
Arthur Ouellet
Enseignant

Dans le cadre du cours 420-5DE-HY
Évolution des applications
Groupe 1

Cégep de Saint-Hyacinthe
11 septembre 2023

TABLE DES MATIÈRES

BESOINS, DEMANDES ET SUGGESTIONS

Les améliorations demandées se divisent en quatre catégories principales; l'implémentation d'une boucle de jeu, l'implémentation d'un serveur multijoueur, l'amélioration de l'interface utilisateur et l'ajout de contenu:

BOUCLE DE JEU

Il faudra implémenter au minimum un mode *match à mort* pour ainsi avoir un vrai jeu; il faudrait aussi un mode en équipe et d'autres modes si le temps le permet.

Chaque partie devra avoir un compte à rebours ainsi qu'un système de pointage, avec la possibilité de mettre un *last stand* (étouffement du *respawn*) dans un monde qui se rétrécit, un *last stand* sans rétrécissement de monde ou simplement une fin de partie à la fin du compte à rebours.

Il faudra aussi implémenter des *boosters* - des objets consommable qui apportent une modification temporaire à l'expérience du joueur (dans le style d'une étoile dans *Super Mario Bros.*et les *pickups* dans la plupart des *FPS*), et possiblement des armes avec des comportements différents (au lieu de l'arme du style mitraillette de base).

SERVEUR MULTIJOUEUR

La boucle de jeu reposera sur une architecture multijoueur avec une topologie client/serveur. La communication se fera en *UDP* et le serveur devra pouvoir gérer la transmission des paquets et sa latence.

Le but à atteindre sera de pouvoir configurer une partie directement sur un serveur (en *CLI*) et jouer une partie avec plusieurs joueurs sur ce dernier.

Si possible, il pourrait y avoir une interface de configuration de partie accessible par le client qui permettrait de laisser le serveur rouler sans l'intervention d'un utilisateur pour la configuration d'une partie.

Il faudra faire la conceptualisation de l'architecture des paquets et des procédés (communications avant/pendant/après la partie et déclencheurs). Il faudra aussi décider du modèle d'entitée distante (autre joueur ou objet) dans le client ainsi que du modèle de connecteur du client.

L'INTERFACE UTILISATEUR

Il y aura - pour permettre l'accès aux autres améliorations, l'ajout d'un menu de démarrage pour la configuration d'une partie multijoueur ou en solo ainsi qu'un menu pour les paramètres du jeu.

Le menu de configuration de partie multijoueur devra permettre au minimum de se connecter à un serveur de jeu distant avec une adresse IP ou un nom de domaine, de donner son nom de joueur à celui-ci et pouvoir participer à une partie *match à mort* ou *match à mort en équipe* pré-configurée par le serveur.

Si le temps le permet et que le serveur a le code requis, il serait possible d'avoir un menu pour créer ou joindre une partie multijoueur à partir d'un client avec un système de *lobby* - la topologie restera client/serveur et le serveur fera toujours la gestion de la partie. Le joueur créateur pourra choisir les options de mode de jeu, le *seed*, la grandeur de la carte et le nombre de joueurs requis/maximal.

Le but à atteindre pour le menu de configuration d'une partie seul est de laisser le joueur jouer seul dans une carte avec un *seed* pouvant être choisi par le joueur ou au hasard et des options de grandeur de carte sans la boucle complète de partie comme en multijoueur - une sorte de mode *free roam* comme le jeu est en ce moment.

Si jamais nous trouvons le temps de faire une intelligence artificielle pour les ennemis, il serait possible et relativement simple d'intégrer cette IA et permettre les modes de jeu *match à mort* et autres en solo.

Le menu de paramètres de jeu aura deux versions, une version en partie qui n'aura que les options qui ne requiert pas de redémarrage de l'engin et une qui sera accessible que lors du démarrage du jeu (hors-partie) avec toutes les modifications possibles;

Le menu de paramètres de jeu en partie devra permettre la modification de:

- la sensibilité de la souris,
- le volume de la musique ainsi que des sons diégétiques,
- la présence de flash lors du tir,
- la distance de vision (si jamais les cartes sont assez grosses pour que le rendu puisse être affecté);

le menu complet aura aussi:

- résolution de l'écran,
- limite d'images/seconde,
- résolution des textures.

Les changements dans ce menu devront être sauvegardés dans un fichier et devront être chargés lors du démarrage du jeu.

Il y aura aussi la création d'un *heads-up display* pour afficher l'état du joueur et de la partie (vie, pointage, temps restant, arme, etc.) ainsi qu'un système de messagerie provenant du serveur (*i.e.:* «_____ a tué _____.», «30 secondes restantes à la partie.») qui, *si le temps le permet*, pourrait être utilisé pour implémenter un *chat* entre les joueurs.

À cheval avec la partie *Contenu*, il y aurait la création d'un écran *splash* affichant un logo pour le jeu ainsi qu'un écran de chargement et un écran de fin de partie avec les statistiques finales de partie.

CONTENU

Cette catégorie englobe le contenu qui n'est pas principalement du code, les améliorations de code déjà présent sans changer l'étendue de ce code et les corrections de *bugs* présents dans le produit initial.

Il faudra modifier le code relié à la génération des *chunks* pour l'améliorer et rendre le monde plus propice à une bonne expérience de jeu avec la nouvelle boucle de jeu.

Il serait possible de faire des structures qui seront générées dans le monde s'il reste du temps.

L'implémentation des sifflements de balles avec le son 3D devra être fait; les entitées de balles sont déjà positionnées dans le monde, donc cette implémentation ne devrait pas être trop compliquée.

Il serait possible aussi de faire de la physique ballistique si le temps le permet.

Il y aura aussi la création de contenu visuel relié au joueur (un bras avec une arme), aux autres joueurs (*sprites* animées à la *Doom (1993)*), aux *boosters* et - si possible, refaire les textures existantes et la *skybox*.

Dans les corrections, il y aura de gérer le point de commencement des joueurs dans une partie multijoueur, de permettre l'escalade d'un bloc (au lieu de rester pris devant) et de corriger un problème dans la génération des *meshes;* ces *meshes* ne sont pas tout le temps rafraîchies quand un autre *chunk* est crée à côté lors du roulement de monde (mais ça ne sera pas nécéssaire si la fonctionnalité de roulement de monde infini est enlevé).

PLANIFICATION PRÉLIMINAIRE

Le développement sera fait en mode *sprint* et aura quatres itération de 3 semaines (calculs faits avec 5 heures/semaine/personne - 5 heure/semaine pour les tests : 5hrs*7pers*3sem-5hrs*3sem = 90 heures/itération);

Itération 1 *Cette itération est pricipalement de la conceptualisation et de la création de contenu visuel.*

#	Catégorie	Dépendances	Tâche	Temps
1	Contenu		Faire l'écran *splash*/l'écran titre.	3h
2	Contenu		Faire le bras du joueur/armes.	3h
3	Contenu		Faire les *boosters*.	3h
4	Contenu		Faire les animations des autres joueurs	10h
5	IU		Faire l'architecture générale des menus.	9h * 2 pers.
6	Serveur		Faire l'architecture générale (paquets, classe joueur/objet, procédures)	9h * 2 pers.
7	IU		Faire la conception du *heads-up display*.	5h * 2 pers.
8	IU		Faire afficher un message à l'écran pendant 2 secondes lors de la modification d'une *string «message»*.	6h
9	IU		Faire une menu de départ avec un bouton «Jouer» qui démarre une partie.	5h
10	Boucle		Faire l'implémentation d'un compte à rebours de partie.	5h
11	Contenu		Faire la correction de l'escalade de bloc.	4h
12	Serveur	S1-6 (9h)	Créer le projet serveur.	1h
13	Serveur	S1-12 (10h)	Faire l'implémentation des sockets de connexion client/serveur.	4h
---	Tests		Faire des tests pertinents aux tâches du sprint 1.	15h
			Total	**105h**

Itération 2

#	Catégorie	Dépendances	Tâche	Temps
14	Serveur	S1-6	Implémenter la classe autre joueur	5h
15	Serveur	S1-13	Faire la gestion du transfert de paquets	10h * 2 pers.
16	Serveur	S2-14 (5h)	Faire la gestion des modes de jeu.	10h * 2 pers.
17	IU	S1-7	Implémenter le *heads-up display*.	10h
18	Boucle	S2-14 (5h)	Faire les mécaniques de combat.	10h
19	Boucle	S1-3	Implémenter les *boosters*.	7h
20	Contenu		Implémenter les sons 3D	8h
21	IU	S1-9	Menu démarrage - faire l'IU pour jouer seul et la connexion multijoueur.	5h
22	IU	S1-9	Menu pause - interface menu retour à la partie/quitter partie.	5h
---	Tests		Faire des tests pertinents aux tâches du sprint 2.	15h
			Total	**105h**

Itération 3

#	Catégorie	Dépendances	Tâche	Temps
23	Serveur	S2-15, S2-16, S2-18	Copie de la gestion de mécanique de combat du client	5h * 2 pers.
24	Serveur	S3-23	Implémenter la gestion de la latence/des erreurs.	10h * 2 pers.
25	Serveur	S2-16	Implémenter la configuration de serveur à partir du client (*lobby*)	10h
26	IU	S2-21, S2-22	Faire le menu de paramètres	5h
27	IU	S3-26 (5h)	Faire la sauvegarde des paramètres	5h
28	Boucle	S2-18	Implémenter le rétrécissage du monde.	7h
29	Contenu		Améliorer la génération de monde.	5h
30	Contenu	S3-30 (5h)	Ajouter des structures à la génération de monde.	10h
31	IU	S2-16	Faire l'écran de fin de partie.	5h
32	Contenu		Faire les corrections des *meshes*.	3h
33	IU	S3-25 (10h)	Faire l'écran de *lobby* pour créer une partie.	5h
34	IU	S3-25 (10h)	Faire l'écran de *lobby* pour joindre une partie.	5h
---	Tests		Faire des tests pertinents aux tâches du sprint 3.	15h
			Total	**105h**

Itération 4

#	Catégorie	Dépendances	Tâche	Temps
35	Serveur		Faire une base de données pour les *leaderboards* et les *logins*.	5h * 2 pers.
36	Serveur	S4-35 (5h)	Faire une procédure de *login* dans le serveur.	10h * 2 pers.
37	Serveur	S2-16	Implémenter la configuration de serveur à partir du client (*lobby*)	10h
38	Boucle	S2-18	Faire l'implémentation d'un mode un joueur (PvE)	10h * 2 pers.
39	IU	(S4-35 et S4-36 pas annulés)	Faire un écran de *login* et de *leaderboard*.	7h
40	Boucle		Faire des armes différentes.	10h
41	Serveur	S4-40 (10h)	Copier les mécaniques d'arme dans le serveur	5h
42	Contenu		Peaufiner le contenu.	8h
---	Tests		Faire des tests pertinents aux tâches du psrint 4.	15h
			Total	**105h**

TECHNOLOGIES

Voici les technologies présentes et les technologies qui seront ajoutées au projet;

LANGAGES DE PROGRAMMATION

Le projet est principalement écrit en C++ (standard C++17) avec des *shaders* en GLSL.

Le serveur sera aussi en C++ et utilisera du code recyclé du projet (principalement pour la physique et la génération de monde).

LIBRAIRIES EXTERNES

Ce projet utilise quatre librairies externes;

- OpenGL/GLEW/GLU; l'API OpenGL est utilisé pour le rendu de graphisme, GLEW et GLU sont des librairies pour accéder aux fonctionnalités de l'API,
- SFML est une librairie servant à la gestion de l'interface utilisateur (création de fenêtre, gestion d'évenements clavier/souris, etc.),
- DevIL est une librairie de manipulation d'image; dans ce projet, ça sert à gérer les textures pour faciliter le rendu de celles-ci,
- irrKlang est une librairie audio pour la gestion des sons 2D et 3D.

ENVIRONNEMENT DE DÉVELOPPEMENT INTÉGRÉ

L'*EDI* utilisé pour le projet est *Microsoft Visual Studio 2022*.

COMPILATEUR

Le compilateur est *Microsoft Visual C++* avec support pour le standard C++17.

Il serait possible de changer le compilateur pour *Clang* si le changement s'avère pertinent.