Cégep de Sainte-Foy – Automne 2024
Programmation de jeux vidéo II – 420-W51-SF

# Travail Pratique 1

25 septembre 2024

Préparé par
François Paradis

# 1 Résumé

Créer un jeu de tir à la troisième personne. Dans ce jeu, des aliens viennent envahir l'univers du joueur. Le joueur doit éliminer les aliens et détruire tous les portails où ils apparaissent pour gagner.

# 2 Conditions de réalisation

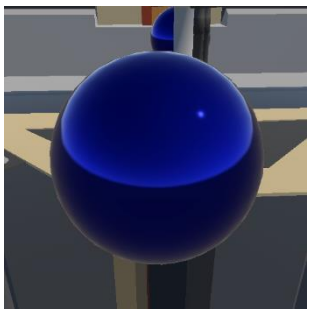
| Valeur de la note finale | Contexte | Durée |
|---|---|---|
| 20% | En équipe de 2 | 2 ½ semaines |

# 3 Fonctionnalités

Cette section présente les fonctionnalités obligatoires de votre jeu. Vous avez la liberté de les bonifier si vous le souhaiter. Les métriques (vitesse, vie, nombre d'aliens, etc.) sont laissées à votre discrétion.

## 3.1. Portails

Des portails bleus doivent être disposés dans le monde. Les aliens sortent de ces portails pour attaquer le joueur. Le joueur doit détruire ces portails en tirant dessus pour qu'ils arrêtent de générer des aliens.



## 3.2. Aliens (Les ennemis)

Les aliens surgissent des portails, tombent au sol, et se mettent à poursuivre le joueur. Un alien meurt s'il entre en contact avec le joueur (le *Space Marine*). En poursuivant le joueur, les aliens doivent éviter les obstacles (et les autres aliens).

## 3.3. Space Marine (Le joueur)

Le joueur peut sur les aliens et les portails pour les éliminer. Sa cadence de tir est assez élevée. Il peut aussi sauter (mais pas de double saut). S'il saute sur un alien, il l'écrase et l'élimine en un coup façon « Mario », sans prendre de dégâts. Quand le joueur prend des dégâts, il devient invincible pendant 0.5 sec.

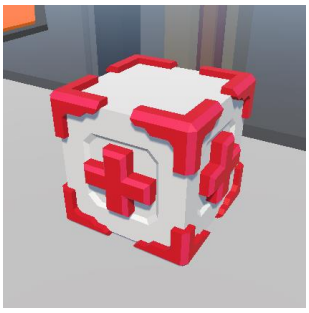
> **Astuce**
> Lors d'une collision entre le joueur et un alien, comparez la position en $Y$ des deux entités. Cela devrait vous permettre de savoir si le joueur doit prendre des dégâts.

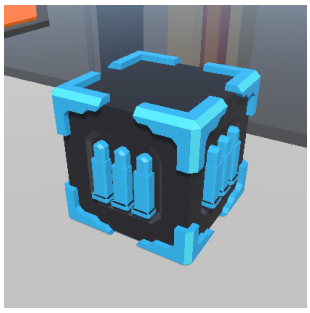
## 3.4. Collectibles (*Pickups*)

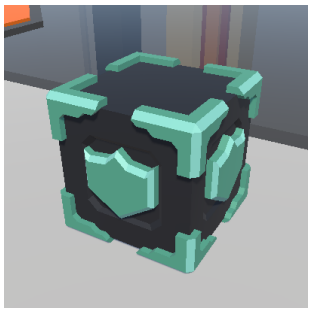
Les aliens peuvent faire tomber des collectibles :

- La boite de soin redonne des points de vie au joueur.
- La boite de munitions augmente la cadence de tir pendant 10 secondes.
- La boite d'armure donne des missiles qui font du dommage dans un certain rayon (personne n'a jamais dit que le jeu était « logique »).

Les collectibles disparaissent au bout de 15 secondes s'ils ne sont pas récoltés.





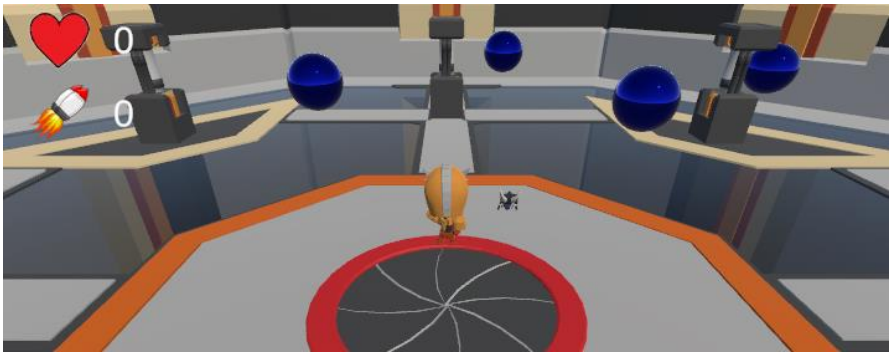


## 3.5. Fin du jeu

Si tous les portails et tous les aliens sont éliminés, alors le joueur gagne la partie. Une musique de victoire doit jouer et un message doit être affiché. Si les points de vie du joueur atteignent 0, alors le joueur perds la partie. Dans tous les cas, le joueur doit quitter le jeu pour recommencer une partie.

### 3.6. Affichage tête-haute (HUD)

Le nombre de points de vie ainsi que le nombre de missiles que le joueur possède doit être visible en tout temps à l'écran. Ces images sont fournies avec le projet.




# 4 Technicalités

Cette section vise à présenter les détails d'implémentation du jeu. Ce ne sont pas des fonctionnalités à proprement parler, mais il faut tout de même y penser pendant le développement.

## 4.1. Projectiles

Les projectiles doivent démarrer leur course à partir du canon de l'arme du joueur. Le plus simple est de prévoir un **`GameObject`** vide attaché au canon en guise de « point d'ancrage ». Faites ensuite usage de la propriété **`forward`** de son **`transform`** afin de savoir dans quelle direction envoyer le projectile.

> ⚠ Important
> Les projectiles doivent se déplacer en utilisant le moteur de physique. Si vous utilisez le **`transform`**, la détection des collisions risque de ne pas fonctionner.

## 4.2. Aliens

Les aliens doivent naviguer dans le monde en utilisant un **`NavMesh`**. Pour qu'ils puissent se déplacer sans trop de problèmes, changez la valeur de **`Step Height`** pour **`1`** (voir les « *Agent Types* »).

Notez que dès que le **`NavMesh Agent`** est actif, il se téléporte immédiatement sur le graphe (donc, au sol). Pour que l'alien descende lentement du portail, il serait judicieux de désactiver le **`NavMesh Agent`** au départ pour l'activer plus tard quand l'alien touche le sol.

### 4.3. Portails

Les aliens apparaissent dans un portail au hasard. Le rythme de création n'est donc pas relié à chaque portail individuellement : il est global. Prévoyez un script spécialement pour ça. Une fois le nombre maximal d'aliens atteint (environ 20), les portails ne doivent plus créer d'aliens.

Les portails détruits ne créent plus d'aliens.

### 4.4. Caméra

La caméra doit utiliser *Cinemachine*. C'est un élément complexe à mettre en place et peut demander beaucoup de raffinage. Ne mettez pas trop d'énergie sur ce point : concentrez-vous sur le reste.

### 4.5. Collectibles

Un collectible a une chance sur $X$ d'apparaitre quand un alien est éliminé ($X$ doit être configurable dans l'inspecteur). N'importe quel des trois types de collectable doit pouvoir apparaitre au hasard.

Les collectibles pivotent sur place en continu. Ils ont une durée de vie de 15 secondes.

Le joueur peut récolter plus de points de vie qu'il n'en avait au départ. Il peut donc avoir autant de points de vie qu'il le souhaite. La boite de missiles donne 5 missiles à la fois (il n'y a pas de limite non plus).

### 4.6. Missiles

Les missiles explosent et font des dégâts tout autour d'eux. Faites usage de la fonction statique `Physics.SphereCast` pour obtenir la liste de toutes les entités impactées par l'explosion. Il existe des effets de particule que vous pouvez utiliser en guise de visuel pour cette explosion.

### 4.7. Sons

Jouez les sons appropriés selon le contexte. Par exemple, il y a un son quand un alien meurt, quand le joueur meurt, quand il ramasse un collectible, quand il tire, etc.

## 5 Conseils

- Faites des *prefabs* pour vos objets et entités. Ici, mieux vaut trop que pas assez.
- Pensez au sens de la communication entre les entités. Ce n'est pas le joueur qui devrait détecter un collectible, mais plutôt le collectible qui devrait détecter le joueur. Même chose pour les dégâts.
- Faites usage de routines asynchrones (ou coroutines). N'oubliez pas les différences entre les deux : ce n'est pas juste une question de performances.

# 6 Contraintes et obligations

| # | Description | OK |
|---|---|---|
| 1 | Classez vos ressources de manière appropriée dans le dossier « Assets ». Gardez aussi votre scène bien organisée, avec des regroupements si nécessaire pour s'y retrouver. | |
| 2 | Le nouveau « *Input System* » de Unity doit être utilisé. Votre jeu doit se jouer minimalement avec la souris. Le mouvement de la caméra doit être géré avec « Cinemachine ». | |
| 3 | Les scripts (`MonoBehaviour`) doivent être courts et bien découpés. Chaque script ne doit s'occuper que d'une petite partie du jeu. Bien entendu, cela reste relatif : le script du joueur sera nécessairement plus gros que les autres, mais il ne devrait pas non plus en faire trop. | |
| 4 | Vous avez vu en classe une technique pour trouver facilement des composants globaux. Vous avez aussi vu une technique pour publier des événements globaux. Il faudra en faire usage. | |
| 5 | La stabilisation de la mémoire et le recyclage d'objet doivent être implémentés. Tous les objets de la scène de jeu doivent être présents en mémoire en début de partie. | |
| 6 | Les scripts doivent exposer un maximum de paramètres (`SerializeField`). Il ne devrait y avoir aucune constante (en dehors ce qui est vraiment universel). | |
| 7 | Utilisez un `CharacterController` pour gérer les déplacements et les collisions du *Space Marine*. La mécanique de saut (et la gravité) devra être implémentée manuellement. | |

# 7 Fonctionnalités facultatives

Vous pouvez ajouter des fonctionnalités au jeu. Vous avez carte blanche. Ces fonctionnalités ne donnent pas de points bonis sur la note finale. Par exemple : musique, animations, effets de particules, etc.

L'un des fonctionnalités que vous pouvez aussi ajouter est un boss final. Le boss final est un robot apparaissant quand tous les portails et les aliens sont éliminés. Il s'agit d'un élément de gameplay facultatif qui peut prendre pas mal de temps à implémenter : ne le commencez que si vous êtes prêts à y mettre beaucoup d'heures. Implémenter le *boss* final donne des points bonis (15 %) sur la note.

Voici ce qu'il y a à savoir :

- Le Boss a 100 points de vie.
- Il descend de son point surélevé quand tous les aliens et les portails ont été éliminés.
- Il circule au hasard entre différents points d'intérêts, tout en faisant face au joueur.
- Il tire vers le joueur des missiles. Les missiles font 5 points de dégâts.
- Une fois le robot vaincu, le joueur gagne la partie.

# 8 Collaboration et plagiat

Une épreuve finale à caractère synthèse en équipe ne peut **en aucun temps** être le produit d'une collaboration ou de partage de code entre des équipes différentes. Toute collaboration, partage de code[1], ou autre interaction sera sanctionnée selon la clause 6.1.12 du plan de cours :

> Tout acte de plagiat, de tricherie et de fraude sera sanctionné.
>
> Constitue notamment un plagiat, une tricherie ou une fraude l'acte de :
>
> - faire passer pour sien un travail, ou toute autre production constituant une évaluation, réalisé en tout ou en partie par une autre personne ou par une **intelligence artificielle**;
> - copier, fournir ou recevoir volontairement de l'information lors d'une évaluation;
> - obtenir, posséder, utiliser ou diffuser des questions ou réponses d'évaluation;
> - remplacer une personne ou de se faire remplacer lors d'une évaluation;
> - falsifier les résultats d'évaluations;
> - falsifier des données de laboratoire et de journal de bord;
> - utiliser des outils non autorisés en contexte d'évaluation, par exemple une **intelligence artificielle**, un cellulaire, un ordinateur, une calculatrice, etc.;
> - reproduire en tout ou en partie le travail d'une autre personne ou un travail généré par une **intelligence artificielle**, qu'il s'agisse d'un document imprimé, audiovisuel ou numérique, sans y faire expressément référence.
>
> En cas de plagiat, de fraude ou de tricherie, ou en cas de coopération à un plagiat, à une fraude ou à une tricherie lors d'une évaluation, la personne étudiante obtient la note « 0 » pour cette évaluation, sans exclure la possibilité d'autres sanctions compte tenu de la gravité de la faute ou du fait qu'il s'agisse d'une récidive.
>
> Chaque cas de plagiat, de fraude ou de tricherie doit être signalé à la Direction des études ou à la Direction de la formation continue et des services aux entreprises selon les modalités en vigueur.
>
> https://sites2.csfoy.ca/presentationtravaux/limportance-de-citer-ses-sources/

Si de tels comportements sont observés en classe, le professeur se réserve le droit de mettre fin à l'évaluation des étudiants fautifs en cours de réalisation et d'attribuer la note 0.

[1] Du code identique avec des variables renommées et/ou des espaces introduits est considéré comme du plagiat.

# 9 Modalités de remise

Remettez votre projet sur LÉA, dans la section travaux, à l'intérieur d'une archive *Zip*. Une pénalité de 15 % est appliqué en cas de retard de moins d'une journée. Au-delà de ce délai, le travail est refusé et la note de 0 % est automatiquement attribuée.

Supprimez les fichiers et dossiers inutiles, c'est-à-dire :

- Les dossiers `.vs`, `Library`, `Logs`, `obj`, `Temp`, `UserSettings`, et `Build`.
- Les fichiers `.sln`, `.csproj`, et `.user`.
- Cet énoncé.

# 10 Grille dévaluation

| Critères | Pondération |
|---|---|
| Utilisation correcte du C# et des scripts<br>Code clair et algorithmes compréhensibles (documentées si nécessaire). Respect de l'encapsulation. Respects des standards du langage. Code de qualité. Exploitation judicieuse des possibilités du langage (ne pas recoder ce qui existe déjà). Faire des petits scripts reliés à un comportement donné. Bonnes solutions pour la communication entre scripts. (Peut être ajusté si pas suffisamment de contenu). | 20 % |
| Utilisation correcte de Unity et de ses concepts.<br>Utilisation correct et judicieuse des *Game Objects* et des composants. Respect de la boucle de jeu Unity et de ses parties. Utilisation judicieuse des *colliders*, *triggers* et de la physique. Communication entre *Game Objects*. Recyclage d'objets et stabilisation de la mémoire. Gestion des entrants (inputs). Utilisation judicieuse du moteur Unity (*Project*, *Hierarchy* et *Inspector*). (Peut être ajusté si pas suffisamment de contenu). | 30 % |
| Partie fonctionnelle.<br>Partie complète, jouable et sans bug. Le *fun* du gameplay n'est pas évalué, mais une expérience particulièrement désagréable pourrait être pénalisée. | 50 % |
| Points bonus<br>Implementation complete et sans bogues du boss final. | + 15 % |

| Pénalités et rigueur |
|---|
| Qualité générale du code (incluant, mais sans s'y limiter) :<br>• Propreté générale du code.<br>• Mauvaises pratiques de programmation.<br>• Formatage ou indentation déficiente.<br>• Standard de nommage non respecté.<br>• Cohérence de l'ensemble de l'œuvre.<br>• Orthographe et grammaire (0.5% jusqu'à concurrence de 20 %). |
| Remise du travail (incluant, mais sans s'y limiter) :<br>• Erreur ou avertissement à l'interprétation du code.<br>• Non-respect des consignes de remise.<br>• Fichier inutile ou temporaire remis avec le projet.<br>• Travail remis en retard. |

**Important**

Le professeur se donne le droit de questionner tout étudiant sur le code qu'il a rédigé. La note finale pourra alors être pondérée si les réponses sont jugées insatisfaisantes.