

Feuille de TD n°2

Space Shooter

M2 GAMAGORA

16 octobre 2025

1 Scène principale

Objectif : fournir une scène d'entrée minimale pour lancer et tester le jeu.

Contenu minimal :

- nœud racine (**Node2D**) ;
- sol de test (**StaticBody2D**) ;

2 Faire le joueur

Objectif : définir un personnage jouable simple avec déplacements et animations.

- scène joueur avec racine **CharacterBody2D** (ou **KinematicBody2D** pour Godot 3.x) ;
- **CollisionShape2D** pour les collisions ;
- sprite/animations (**AnimatedSprite2D** ou **AnimationPlayer**) ;
- script gérant le déplacement horizontal, le saut et éventuellement un dash/roulade ;
- camera attachée au joueur ou configurée pour le suivre.

3 Créer un niveau avec TileMap

Objectif : concevoir un niveau jouable en utilisant un système de tuiles.

- créer une nouvelle scène avec un nœud racine **TileMap** ;
- créer un **TileSet** contenant les tuiles du niveau ;
- configurer les collisions pour les tuiles pertinentes
- utiliser l'auto-tile pour faciliter la création et la gestion du niveau(documentation)
- placer le joueur et les éléments interactifs dans la scène.

4 Créer des ennemis

Objectif : ajouter des adversaires contrôlés par l'ordinateur et gérer leurs interactions avec le joueur.

- créer une nouvelle scène avec un nœud racine **CharacterBody2D** ;
- placer les ennemis dans le niveau, deux possibilités :
 - manuellement comme des instances de scène ;
 - via **TileMap** pour un placement en grille (documentation).
- gérer le déplacement des ennemis :
 - mouvement de patrouille lorsque le joueur est éloigné ;
 - poursuite du joueur lorsqu'il entre dans une zone définie ;
 - retour au mouvement initial si le joueur s'éloigne ;
- gérer les dégâts du joueur :
 - perte de vie du joueur au contact de l'ennemi ;
 - invincibilité temporaire après avoir été touché ;
 - indicateur visuel de la perte de vie et de l'invincibilité (clignotement) ;
- gérer l'attaque du joueur :
 - dégâts au contact de l'arme du joueur ;
 - mort de l'ennemi ;

5 Gérer le changement de scène

Objectif : permettre la transition entre différents niveaux et gérer les limites de la caméra.

- créer un deuxième niveau ;
- créer un niveau de base (vide ou minimal) et utiliser l'héritage de scène pour faciliter la maintenance ;
- mettre en place un système de changement de scène avec un **Area2D** ;
- éviter de charger tous les niveaux en même temps :
 - ne pas exporter directement une **PackedScene** pour indiquer le niveau à charger ;
 - utiliser un node singleton pour stocker les niveaux et gérer le changement ;
 - indiquer simplement un identifiant de niveau à charger dans l'**Area2D** ;
 - obtenir l'identifiant à partir du nom du fichier de la scène ou d'une variable sur la racine du niveau ;
 - réfléchir à la gestion de la position de spawn du joueur dans le nouveau niveau ;
- gérer la position du joueur dans le nouveau niveau :
 - placer correctement les niveaux pour éviter les problèmes de transition ;
 - déplacer le joueur à la bonne position dans le nouveau niveau ;
- gérer les limites de la caméra :
 - utiliser un rectangle dans le niveau pour indiquer les limites ;
 - empêcher la caméra de sortir de ce rectangle en utilisant ses propriétés de limites ;

6 Faire des ameliorations

7 Faire des améliorations

7.1 Créer des plateformes mouvantes

Objectif : ajouter des plateformes qui se déplacent pour enrichir le gameplay.

- utiliser **AnimatableBody2D** et **Path2D** pour définir le mouvement ;
- configurer la vitesse et le trajet des plateformes ;
- tester l'interaction avec le joueur (embarquement, collision).

7.2 Ajouter d'autres types d'ennemis

Objectif : diversifier les adversaires pour rendre le jeu plus intéressant.

- créer des ennemis volants ;
- créer des ennemis qui tirent des projectiles ;
- créer des ennemis qui foncent sur le joueur pour exploser ;
- adapter les comportements et les animations selon le type d'ennemi.

7.3 Créer un boss

Objectif : concevoir un adversaire principal avec des mécaniques avancées.

- définir une scène spécifique pour le boss avec une zone de boss ;
- ajouter des phases d'attaque et des comportements variés ;
- gérer la barre de vie et les effets visuels spécifiques ;
- prévoir une récompense ou une transition après la victoire.
- Faire en sorte que le joueur ne puisse pas sortir de la zone tant que le boss n'est pas battu.

7.4 Améliorer les limites de la caméra par niveau

Objectif : rendre la gestion des limites plus flexible et précise.

- délimiter le contour du niveau avec un polygone ;
- utiliser ce polygone pour restreindre le mouvement de la caméra ;
- tester le comportement dans différents niveaux.

7.5 Créer un menu de pause

Objectif : permettre au joueur d'interrompre la partie et d'accéder à des options.

- ajouter une interface de pause accessible via une touche ;
- proposer des options de reprise, retour au menu principal, et quitter ;
- gérer l'arrêt des animations et du gameplay pendant la pause.

7.6 Créer des options de configuration des touches

Objectif : offrir la possibilité de personnaliser les contrôles du jeu.

- ajouter un menu dédié à la configuration des touches ;
- permettre la modification et la sauvegarde des préférences ;
- afficher les touches configurées dans l'interface du jeu.