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 en 3D.
Contenu minimal :

- nœud racine (`Node3D` / `Spatial`) ;
- sol / environnement de test (`StaticBody3D` avec `CollisionShape3D` et un `CSGBox3D` pour le rendu) ;
- caméra 3D (`Camera3D`) ;
- lumière directionnelle (`DirectionalLight3D`) ;
- environement 3D de base (`WorldEnvironment`) ;

2 Faire le joueur

Objectif : définir un personnage jouable simple en 3D avec déplacements et animations.

- scène joueur avec racine `CharacterBody3D`
- `CollisionShape3D` (capsule/box) pour les collisions
- un `MeshInstance3D` pour le visuel ;
- deplacement et caméra 3D, deux positions possibles :
  - third-person (caméra derrière le joueur) ;
    - deplacement sur X et Z depedendant de la rotation de la caméra ;
    - la caméra suit le joueur avec un léger smoothing (script ou `SpringArm3D`) ;
    - rotation de la caméra avec la souris ;
  - side-on 3D (plateformer 2.5D) : contraindre le mouvement du joueur à un plan X/Y ou X/Z mais avec rendu 3D ;
    - deplacement sur X avec saut sur Y valeur en Z fixe ;
    - la caméra suit le joueur avec un léger smoothing (script ou `SpringArm3D`) ;
    - rotation de la caméra fixe (pas de rotation avec la souris) ;
    - possibilite d'ecarter la camera du joueur avec la souris (optionnelles) ;

3 Créer un niveau 3D

Objectif : concevoir un niveau jouable en 3D.

- méthodes possibles pour construire un niveau 3D :
  - utiliser des `CSG` nodes et ensuite les assembler en un meshInstance ;
  - utiliser `GridMap` avec une `MeshLibrary` (ou `TileMap 3D` si disponible) pour placer rapidement des blocs/tiles 3D (GridMap docs).
  - utiliser des `StaticBody3D` + `MeshInstance3D` pour chaque élément des assets importés (modèles .glb/.obj).
- configurer correctement les `CollisionShape3D` pour les sols, murs et plateformes ;
- placer le joueur et les éléments interactifs (pickup, traps, ennemis) ;

4 Créer des ennemis

Objectif : ajouter des adversaires contrôlés par l'ordinateur et gérer leurs interactions avec le joueur en 3D.

- créer une nouvelle scène avec un nœud racine `CharacterBody3D` ;
- placement des ennemis : manuellement (instances) pas possible d'utiliser les gridmap pour ça malheureusement ;
- comportements possibles :
  - patrouille le long d'un trajet (utiliser `Path3D` + `PathFollow3D`) ;
  - détection du joueur via un `Area3D` (déclencher la poursuite) ;
  - poursuite avec navigation (`NavigationAgent3D` ou `NavigationServer`) ;
- gestion des dégâts et de la vie :
  - collision corps-à-corps (détectée via `body_entered` ou checks) ;
  - invincibilité temporaire et feedback visuel (émission de particules, shader, clignotement du matériau) ;
- gerer l'attaque du joueur :
  - attaques au corps à corps.
  - detection des degats sur les ennemis (Area3D ou checks a la main via distance)
- animations 3D : utiliser `AnimationTree` et transitions d'animation adaptées (idle, walk, attack, die).

5 Gérer le changement de scène

Objectif : permettre la transition entre différents niveaux 3D et gérer la caméra.

- créer plusieurs niveaux 3D (scènes séparées) héritant de la même scène pour les éléments communs (HUD, gestionnaire de niveaux) ;
- zones de transition : utiliser `Area3D` pour déclencher le chargement du niveau suivant et transmettre un identifiant de spawn ;
- pattern recommandé : ne pas exporter de `PackedScene` directement depuis l'éditeur des zones, mais stocker des identifiants et charger dynamiquement via un singleton (`Autoload`) ;
- gérer la position du joueur : prévoir des points de spawn (`Marker3D` ou noeuds dédiés) dans chaque niveau et restaurer la rotation/position souhaitée ;
- limites de la caméra en 3D :
  - pour une caméra third-person, restreindre la translation via volumes (`CollisionShape3D` ou checks manuels) ou calculer les limites projetées ;
  - pour un side-on 3D, contraindre la caméra et le joueur sur les axes voulus et utiliser un volume pour empêcher la caméra de sortir ;

6 Améliorations et extensions (optionnelles)

6.1 Créer des plateformes mouvantes

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

- utiliser `Path3D` + `PathFollow3D` ou animer la position via `Tween`/`AnimationPlayer` ;
- configurer la vitesse, le trajet et l'interpolation des plateformes ;
- tester l'interaction avec le joueur (embarquement, collision, transfert d'impulsion si nécessaire).

7 Ajout des animations 3D

Objectif : rendre le joueur et les ennemis plus vivants avec des animations 3D.

- utiliser un `AnimationTree` pour gérer les états d'animation (idle, walk, run, jump, attack, die) ;
- configurer les transitions entre les animations selon les actions du joueur et des ennemis ;
- importer des modèles 3D animés (formats .glb/.gltf) et configurer les squelettes et animations ;
- ajuster les animations pour qu'elles correspondent aux mécaniques de jeu (vitesse, hauteur de saut, etc.).

7.1 Ajouter d'autres types d'ennemis

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

- ennemis volants (contrôler Y/Z) ;
- ennemis à projectiles (création et vélocité de projectiles 3D) ;
- ennemis qui foncent pour exploser (timers, zones d'effet avec `Area3D`) ;
- adapter les comportements et les animations 3D selon le type d'ennemi.

7.2 Créer un boss

Objectif : concevoir un adversaire principal avec des mécaniques avancées.

- scène spécifique pour le boss avec une zone de boss (`Area3D`) ;
- ajouter des phases d'attaque et comportements variés ;
- gérer la barre de vie (HUD 2D superposée) et effets visuels (particles, shaders) ;
- bloquer les sorties tant que le boss n'est pas vaincu (collision/Area3D) ;
- prévoir récompense ou transition après la victoire.

7.3 Améliorer les limites de la caméra par niveau

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

- définir des volumes (`Area3D`) ou boîtes de limites pour restreindre la caméra ;
- calculer des collisions ou utiliser des contraintes pour que la caméra ne traverse pas le décor ;
- tester le comportement pour différentes résolutions et rapports d'aspect.

7.4 Créer un menu de pause

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

- ajouter une interface de pause en `CanvasLayer` (UI 2D superposée) ;
- options : reprendre, retourner au menu, quitter, réglages audio/graphique ;
- arrêter la physique ou mettre en pause les agents si nécessaire.

7.5 Créer des options de configuration des touches

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

- menu de configuration des touches (UI 2D) ;
- sauvegarder les préférences dans un `ConfigFile` ou via `ProjectSettings` ;
- afficher les touches configurées dans l'UI.