TECH ART

CAHIER DES CHARGES

Mini-studio

INTRODUCTION

Nom du projet : Mini-studio inter école
N° du Projet : 12
Nom d'UE : Développement personnel et projets communs pour G-ART // Projets appliqués pour G-TECH
N° d'UE : 5 pour G-ART // 3 pour G-TECH
Date : 10/03/2025 – 21/03/2025
Classe : G-ART 1 + G-TECH 1

MODALITÉS

Projet en groupe imposé de 4 avec 2 G-Tech et 2 G-Art.

CONTEXTE ET DESCRIPTION DU PROJET

L'objectif de ce projet est de réaliser **un shoot'em ou shooter like** en vue de dessus (top-down) avec un style cohérent et une direction artistique efficace.
Les **G-ART** auront la charge de toute la partie graphique du jeu. Ils devront produire tous les éléments visuels nécessaires à l'intégration par les **G-TECH** dans le jeu vidéo en usant du moteur développé.

Pour résumer le principe d'un shooter : le joueur contrôle un objet graphique (vaisseau, personnage, autre...) et il est attaqué par différents types d'ennemis qui peuvent lui faire des dégâts au corps à corps ou à distance ou même des dégâts de zone.

Les possibilités d'environnements d'un shooter sont grandes, vous pourrez décider que le joueur puisse se mouvoir sur une carte finie ou infinie ou même faire en sorte que le joueur se déplace en bottom-top à la space invaders.

Contraintes techniques

- **Pour les G-Tech**

L'objectif de ce projet est donc de réaliser un jeu vidéo avec un **rendu graphique**. Le jeu sera bien entendu, développé entièrement en C++ et vous aurez droit seulement à l'utilisation de la **librairie SFML** en plus de **la librairie standard.** Vous pourrez utiliser le dernier standard C++23 si vous le souhaitez. On autorise ImGUI si nécessaire mais ne perdez pas de temps avec cette dernière librairie.

- **Pour les G-Art**
  - Format des assets
    - Il devra y avoir des spritesheet par type d'entités.

- **Besoins communs**
  - Éléments requis :
    - **Vaisseau/personnage principal** (avec animations si nécessaire)
    - **3 types d'ennemis minimum** (formes variées et animations simples)
    - **Effets de tirs et explosions** (projectiles, impacts, explosions visuelles)
    - **Décors** (fonds animés pour le scrolling, éléments de décor statiques ou en parallax)
    - **Bonus et upgrades** (tir amélioré, boucliers, vitesse accrue, etc.)
    - **Interface minimaliste** (barre de vie, score, power-ups visibles)

Choix du Thème

Les équipes doivent choisir un thème parmi ceux proposés :

- **Médiéval** (dragons volants, sorciers en tapis volant, etc...)
- **Futuriste** (vaisseaux spatiaux, drones de combat, lasers...)
- **Zombie** (hélicoptères ou survivants, vagues de zombies, ruines...)
- **Aérien** (avions de chasse, biplans rétro, dogfights...)
- **Autres suggestions possibles** :
  - Steampunk
  - Cyberpunk
  - Mythologique
  - Insectoïdes
  - Biomécanique...

Chaque thème devra avoir une cohérence graphique et une palette de couleurs adaptée.

Specs

Vous devez implémenter dans votre application :

- **Gameplay**
    - Menu d'écran d'accueil
        - Quand le joueur lance le jeu, un écran d'accueil doit s'afficher et il est possible de sélectionner un élément parmi les suivants :
            - Play
                - Permet de lancer le jeu
            - Exit
                - Permet de quitter **proprement** le jeu
    - Menu pause
    - Navigation joueur
        - Le joueur utilisera des touches pour se déplacer.
        - La souris pourra être utilisée pour orienter le projectile et pour effectuer des rotations du player en fonction du shooter proposé.
        - Une touche sera utilisée pour attaquer
        - Des touches ou autres inputs pourront être utilisées pour changer d'armes ou capacités.
    - ATH joueur
        - Le joueur doit disposer d'un élément visuel capable de montrer l'état de sa santé avant de mourir.
        - Le joueur doit être en mesure de voir son score affiché tout au long de la partie.
    - Mort du joueur
        - Lorsque le joueur perd, un game over doit apparaitre.
    - Bonus/Malus system
        - Mettre en place un mécanisme où le player peut avoir des power-up ou power-down qui affecteront le joueur (déplacement, vie, dégâts, etc...)
    - Multi-weapon
        - Mettre en place un mécanisme qui permettra au joueur d'avoir différentes armes avec différentes caractéristiques. Celles-ci, pourront être sélectionnables via un mécanisme suffisamment ergonomique.

- Ennemis
    - Gestion de la vie
        - Vous pouvez ajouter des ennemis qui ne meurt pas en un seul coup mais qui disposent d'une résistance plus forte.
    - Patterns
        - Les ennemis peuvent avoir des comportements divers et variés.
    - Boss
        - Lorsque le joueur atteint des paliers, il peut faire face à des boss qui auront leurs propres comportements.

- **Performances**
    - Le jeu doit être synchronisé sur 60fps et on doit pouvoir afficher le framerate via une touche clavier.
        - Celui-ci ne doit pas chuter lorsque des nouveaux ennemis poppent dans la zone de combat.

- **Collisions**
    - Au minimum, un système de collision AABB devra être mis en place pour gérer la collision des objets.
    - Celui-ci devra être débuggable via l'appui d'une touche

- **Outils**
    - Les étudiants auront la possibilité d'utiliser la librairie standard C++17/C++ 20 ou C++23
    - Ils devront versionner leur code en utilisant **gitlab** afin d'avoir une phase de développement saine.
        - Ils devront créer des branches avec des merge request.
        - Ils devront avoir un git léger qui contient le strict minimum requis pour cloner et recompiler.
    - Pour les assets, ils seront fournis par les G-ART et devront être versionnés dans le git.
    - Afin que les membres de l'équipe communiquent facilement le long de la réalisation de leur projet, il est indispensable d'utiliser des outils propres au développement logiciel :
        - Traces UML
        - Traces des décisions
        - **Et un board type scrum / kanban disponible sur gitlab**
        - Les G-Tech auront en charge la gestion du projet

- **Approche créative et logiciels**
    - Vous pourrez utiliser différentes techniques pour réaliser vos assets :
        - **Pixel Art :** Aseprite, Piskel, Photoshop (grille et crayon pixel)
        - **Voxel Art :** MagicaVoxel, VoxEdit
        - **Dessin digital :** Photoshop, Krita, Procreate
        - **Dessin papier/scanné :** puis retravaillé numériquement

Ressources G-Tech

1. **A installer/prérequis techniques** :
    - Visual Studio 2022
    - Git
    - Helix (p4merge)
    - Vcpkg
    - SFML

CMake

Tutoriels recommandés G-Art

Pour vous aider à vous débrouiller en autonomie, voici quelques tutoriels utiles :

- Pixel Art
    - https://www.youtube.com/results?search_query=tutorial+pixel+art+photoshop
- Création d'un Spritesheet
    - https://www.youtube.com/watch?v=6Emb5KhBbuY
- Voxel art
    - https://www.youtube.com/watch?v=5tqBNAr52SA

Temps de travail en autonomie

Vous devez travailler en dehors des séances pour affiner votre travail et respecter les deadlines.

- Chaque séance servira de point de contrôle pour valider l'avancement.
- Une bonne gestion du temps est essentielle pour éviter le rush final.

Cohésion d'équipe

- **Communication constante G-TECH / G-ART** pour garantir une intégration fluide.
- **Échanges et feedbacks** pour homogénéiser le style visuel.
- **Travail collaboratif :** pas de travail isolé, chaque membre doit contribuer activement.

PLANNING DU MODULE MINI-STUDIO POUR LES G-ART

Semaine 1

- **Séance 1 (2h) :** Présentation du projet, choix du thème, moodboard et croquis rapides.
- **Séance 2 (2h) :** Esquisse des assets principaux (vaisseau/personnage, ennemis, premiers éléments de décor).
- **Séance 3 (2h) :** Raffinement des visuels et début de digitalisation.

Semaine 2

- **Séance 4 (3h) :** Finalisation des assets, intégration dans le sprite sheet, tests de lisibilité et ajustements.
- **Séance 5 (2h30) :** Soutenance (présentation des assets, processus créatif, choix graphiques...).

- **Collaboration avec les G-TECH :** Adaptation aux contraintes techniques.
- **Conclusion :** Réflexion sur le travail accompli et points d'amélioration.

LES 10 COMMANDEMENTS POUR RÉUSSIR LE PROJET

1. **En retard, jamais tu ne seras !**
    a. Aucun retard ne sera toléré et il faudra se forcer à respecter les deadlines qui sont une priorité pour le bon déroulé d'un projet.
2. **Travailler en équipe tu devras !**
    a. TEAM = Together Everything Advance More
    b. Personne ne travaille dans son coin, la communication est essentielle et même si seul on pense travailler plus vite, n'oubliez pas qu'à plusieurs, on va plus loin !
3. **Penser aux yeux des lecteurs tu devras !**
    a. Il faudra créer des visuels lisibles en priorisant la clarté et non la complexité.
4. **Straight to the point tu iras !**
    a. Prioriser l'efficacité et aller à l'essentiel, il faut éviter les détails inutiles.
    b. Même si une feature n'est pas réalisée pour du long-terme, en cas de manque de temps il faut capitaliser sur le requis.
5. **Cohérent tu seras !**
    a. Il faut vérifier la cohérence du style et contrôler que l'ensemble est homogène.
6. **Tes doutes tu lèveras !**
    a. Il faudra tester régulièrement l'affichage, les assets en jeu, le gameplay pour s'assurer du rendu.
7. **Les contraintes techniques tu respecteras !**
    a. Il faudra prendre en compte les contraintes techniques de chaque corps de métiers comme respecter la résolution, les spritesheet, etc...
8. **Les retours tu anticiperas !**
    a. G-TECH et G-ART devront anticiper les feedbacks entre-eux et échanger régulièrement pour éviter les erreurs et les non-dits.
9. **Ton temps tu organiseras !**
    a. Il est essentiel d'organiser son temps et de travailler régulièrement en autonomie pour éviter le rush final.
    b. Un board commun tu devras utiliser ainsi que des méthodes de gestion de projet.
10. **Une bonne présentation avec <u>amour</u> tu réaliseras !**
    a. Afin d'exposer le travail accompli, tu expliqueras clairement ton projet et tes choix (artistiques, techniques, etc...)

OBJECTIFS PÉDAGOGIQUES ET PROFESSIONNELS DU PROJET

A l'issue de ce projet, les étudiants seront capables de :

Savoirs

- **Modéliser** des problèmes architecturaux de plus haut niveau.
- **Identifier** les problématiques techniques usuelles de l'industrie.
- **Identifier et développer** les principales briques logicielles nécessaires au développement de jeux vidéo.
- **Comprendre** les notions de performances et de framerate.
- **Discriminer** les parties logicielles qui affectent les performances.
- **Expliquer** ce qu'est la notion de gameplay.
- **Faire le lien** entre le code et le plaisir en jeu.
- **Exploiter** du contenu fait par des artistes et les intégrer dans un jeu.
- **Communiquer** avec d'autres corps de métier.
- **Gérer** des projets incluant différents domaines.
- **Écouter et comprendre** les besoins et les contraintes non habituels.
- **Vulgariser** la technicité pour être audible.

Savoir-être

- **Construire** une solution propre et pérenne pour développer en équipe.
- **Découper et paralléliser** des tâches en adéquation avec l'architecture.
- **S'organiser** pour avoir des MVP (Minimum Viable Product) le plus souvent possible.

Savoir-faire / Compétences

- **Programmer** en utilisant des concepts nécessaires dans l'industrie.
- **Utiliser** des ressources artistiques.
- **Consommer** des librairies tierces

ROADMAP ET RENDUS

GART

À la fin du module, les G-ART devront livrer un spritesheet finalisé, prêt à être intégré par les G-TECH.

PowerPoint pour la soutenance :

- **Introduction :** Présentation du projet, choix du thème et justification.
- **Direction artistique :** Moodboard, inspirations et palette de couleurs.
- **Processus de création :** Croquis, esquisses, techniques utilisées (pixel art, voxel, etc.).
- **Assets réalisés :** Vaisseau/personnage, ennemis, tirs, décor, interface, etc.
- **Organisation et gestion du temps :** Méthodologie et répartition du travail.

GTECH

DESCRIPTION

Les dates limites fixées doivent être rigoureusement respectées afin de garantir une évaluation équitable.

Jalon	Livrables attendus	Date limite	Moyens / formats
1	**Remise du projet**	21/03/2025 à 13h00	• Repository git avec la liste des membres du groupe envoyés sur le fil de rendu discord.
2	**Soutenance**	21/03/2025 de 16h30 à 19h00	• Soutenance orale

EVALUATION

L’évaluation est conçue pour être holistique, prenant en compte non seulement le produit final, mais aussi le processus, les compétences acquises et les attitudes démontrées tout au long du projet. Les compétences sont classées en trois catégories principales : “savoirs” (connaissances théoriques), “savoir-faire” (compétences pratiques), et “savoir-être” (compétences interpersonnelles et attitudes).

SYSTÈME DE NOTATION

Savoirs (Connaissances)

- **Compréhension théorique** : Évaluée soit en amont du projet pendant la semaine théorie soit lors de la soutenance et restitution du projet. Cela permet de mesurer la compréhension des concepts fondamentaux et des connaissances liées au projet des étudiants.

Savoir-faire (Compétences)

- **Compétences techniques et application** : Évaluées à travers la soumission finale du projet. Cela inclut la qualité, la fonctionnalité, et la précision technique du travail produit.
- **Gestion de projet** : Évaluée en fonction de l’organisation, du respect des échéances, et de l’utilisation efficace des ressources. Cela peut être évalué à travers la documentation du projet et les journaux de processus.

Savoir-être (Attitudes/Compétences interpersonnelles)

- **Travail d'équipe et collaboration** : Évalués à travers des évaluations par les pairs et les membres du groupe. Les critères incluent la communication, la coopération, et la contribution aux tâches du groupe.
- **Autonomie et initiative** : Évaluées en fonction des contributions individuelles, de la capacité à travailler de manière autonome, et de la résolution proactive des problèmes.

- **Soutenance** : Évaluées lors de la présentation finale du projet. Les critères incluent la clarté, la cohérence, et la capacité à articuler et défendre les résultats du projet.

GRILLE D'ÉVALUATION

Évaluation du produit fini	**Note**
Respect du cahier des charges et fonctionnalités implémentées	10
Interface utilisateur	5
Gestion des ressources	5
Total	**20**

Évaluation de la soutenance	**Note**
Aisance à l'oral	2
Justification des choix	4
Présentation de l'architecture	4
Respect du temps de parole	2
Organisation du travail	3
Améliorations proposées et problèmes rencontrés	3
Qualité des slides	2
Total	**20**

Questions/réponses	**Note**
Total	**20**

Notation sur le code	**Note**
Architecture	10
Algorithmique	10
Build system/git	10
Qualité du C++	10
Total	**40**

La note finale est la note sur 100 ramenée sur 20.

Modalités de la soutenance :

- 30 minutes réparties de la manière suivante
    - 10 minutes de présentation PowerPoint ou similaire
    - 5 minutes de test du jeu
    - 15 minutes de questions