# INFORMATIONS GENERALES

|  |  |  |  |
| --- | --- | --- | --- |
| Élève : | Nom: Moser | | Prénom: Charles-Henri |
| Lieu de travail : | ETML / Avenue de Valmont 28b, 1010 Lausanne | | |
| Client UX | Nom: Curcho | | Prénom: |
| : | | |
| Client DB | Nom: Mveng | Prénom: | |
| : | | |
| Client OO | Nom: Melly | Prénom: | |
| : | | |
| Dates de réalisation : | 1er trimestre | | |
| Temps total : | 80 périodes (32 P\_OO + 24 P\_DB + 24 P\_UX) | | |

# PROCÉDURE

* Tous les apprentis réalisent le projet sur la base d'un cahier des charges.
* Le cahier des charges est présenté, commenté et discuté en classe.
* Les apprentis sont entièrement responsables de la sécurité et sauvegarde de leurs données.
* En cas de problèmes graves, les apprentis avertissent le client au plus vite.
* Les apprentis ont la possibilité d’obtenir de l’aide externe, mais ils doivent le mentionner.
* Les informations utiles à l’évaluation de ce projet sont disponibles au chapitre 8.

# TITRE

***Shoot me up !***

# SUJET

Concevoir un jeu 2D modulaire de tir à la troisième personne et en réaliser une partie.

# MATÉRIEL ET LOGICIEL À DISPOSITION

|  |  |
| --- | --- |
| * Un PC ETML * Accès à Internet | * <https://etml.icescrum.com> * <https://figma.com> |

# PRÉREQUIS

|  |
| --- |
| * Modules de programmation de base * Modules de bases de données de base * ICT-320 en cours * ICT-322 en cours * ICT-106 en cours |

# CAHIER DES CHARGES

## **Gestion de projet**

1. La planification est à faire selon les instructions spécifiques de votre chef de projet.
2. Un journal de travail devra être rendu. L’outil que vous utilisez est libre, mais les caractéristiques suivantes doivent être respectées :
   * La structure et la présentation sont claires et soignées.
   * Les sources, les fichiers, les répertoires, les commits, et autres sources d'informations concernées par le journal sont référencés.
   * L'état et les durées des tâches mentionnées sont précisés.
   * Toutes les activités planifiées, les aides extérieures, ainsi que les imprévus et les heures supplémentaires y sont mentionnés.
   * Les succès et les échecs sont mentionnés.
   * Le travail journalier et son appréciation critique, ainsi que les réflexions y figurent.

## **Qualité**

1. Réaliser un programme informatique de qualité
   * Organisé (namespace, classes, commit log,…)
   * Compacté (pas de copié/collé,…)
   * Optimisé (utilisation de structures adaptées)
   * Testé (tests unitaires)
   * Commenté
   * Complet (code, script DB, maquettes PDF, éxécutable, …)
2. Prouver que vous êtes digne de confiance lorsqu’on vous confie un projet
   * Journal de travail à jour
   * Pro-activité
     + **Poser des questions** au client
     + Faire des démonstrations
     + Utiliser un système de versioning de code (GIT)

## **Fonctionnalités requises (du point de vue client)**

### Réplica d’un « shoot’em up 2D », comme Space Invader

* 1. Maquettes
     1. Menu principal
     2. Ecran de jeu (niveau)
     3. Éditeur de niveau (voir détails ci-dessous)
     4. High scores
  2. Contraintes de réalisation
     1. Un concept de niveaux décrivant
        1. Le numéro du niveau (Level 1, Level 2, …)
        2. Le joueur
           1. Déplacements
           2. Nombre de vies
           3. Capacités de tir : direction, rafale, cooldown, décompte munitions, recharge, …
           4. Un sprite
        3. Les ennemis du niveau avec (pour chaque type)
           1. Nombre de vies
           2. Minutage d’apparition
           3. Tir (oui / non)
           4. Un sprite
        4. Les obstacles avec (pour chaque type)
           1. Une taille
           2. Une position X,Y
           3. Un sprite
           4. Le comportement en cas de dégâts (tir, collision)
     2. Structure et données des niveaux décrits et stockés dans une base de données relationnelle
  3. Fonctionnalités
     1. Au moins 2 niveaux implémentés avec
        1. Joueur
        2. Ennemis
        3. Obstacles
     2. Gestion des highscores (en base de données)

### Spécificités UX

La documentation contenue dans livraison finale du projet (Github) comprend :

* Un chapitre d’analyse de l’UX:
* Conception centrée utilisateur
* Création de deux profils de joueurs sous forme de “Personas”
* Création d’au moins un « Personas » original  spécificité liée à l’accessibilité et profil qui ne correspond pas aux « classique » du joueur
* Choix de la palette graphique
* Eco-conception
* Accessibilité
* Un chapitre de conception
* Définition de tous les écrans - maquettes base-fidélité / low-fidelity wireframes
* Ecran “Éditeur de niveau” - maquette haute-fidélité / high-fidelity wireframe
* Ajout de fonctionnalités supplémentaires tirées des Personas + un élément original
* (Optionel) Créer un prototype cliquable pour naviguer dans l’interface avec FIGMA
* Choix effectués
* ~~Un chapitre d’évaluation~~
* ~~Tests (test d’utilisabilité)~~

### Spécificités POO

La livraison finale du projet (Github) comprend :

1. Le code
   * Programmation orientée objet
   * Tests unitaires
2. Documentation (chapitres du rapport)
   * Une analyse fonctionnelle centrée utilisateur sous forme de User Stories
   * Automatique du code (manuel de référence)
   * Schémas des classes
   * Au moins un détail d’implémentation spécifique

### Spécificités DB

1. MCD
2. MLD
3. Script SQL de création
4. un jeu de données spécifique à votre jeu contenant:
   * + 1. au moins deux niveaux de jeu
       2. deux joueurs correspondants à vos personnas UX
       3. un joueur devra être présent par niveau
       4. les informations concernant les ennemis et les joueurs devront correspondre exactement à ce qui sera implémenté dans votre jeu(ex: si vous aurez 10 ennemis, nous devons retrouvez ces 10 ennemis dans votre jeu de données. Idem pour les obstacles, etc.)

### Gestion des utilisateurs

Différents utilisateurs vont devoir interagir avec la base de données. Pour ce faire, ces utilisateurs ont des missions différentes :

1. Administrateur du jeu :
   1. Peut créer, lire, mettre à jour et supprimer (CRUD) n'importe quelle table.
   2. Gérer les utilisateurs et leurs privilèges.
2. Joueur :
   1. Aucun accès direct à la base de données. Ce rôle est essentiellement géré par l’application qui contrôle l’accès aux données pour chaque joueur. L’application assure que les joueurs ne peuvent pas accéder directement aux données critiques, mais seulement aux informations qui les concernent.
3. Gestionnaire du jeu :   
   Cet utilisateur serait utilisé par l'application (le jeu) pour gérer les mises à jour en temps réel des données pendant une session de jeu.
   1. Lecture/écriture pour modifier les scores, le nombre de vies, et le niveau actuel du joueur.
   2. Insertion d'un nouveau score quand un joueur atteint un nouveau record.
   3. Lecture pour récupérer les informations sur le niveau (comme les ennemis et obstacles).
   4. Lecture pour récupérer les caractéristiques et positions des ennemis et obstacles.

             Travail demandé : Vous devez expliquer, comment mettre en place ces différents « types d’utilisateurs » afin de

              Rendre la gestion des utilisateurs la plus simple possible. Pour cela, vous devez définir un ensemble de requêtes

              SQL. Vous devez créer au moins un utilisateur par « type d’utilisateurs ».

**Création des index**

En étudiant le dump MySQL de votre DB vous constaterez que vous ne trouvez pas le mot clé INDEX.

1. Pourtant certains index existent déjà. Pourquoi ?

2. Quels sont les avantages et les inconvénients des index ?

3. Sur quel champ (de quelle table. Pensez à l'utilisation de votre DB depuis C# par exemple), cela pourrait être pertinent d’ajouter un index ? Justifier votre réponse.

Backup / Restore

Nous souhaitons réaliser une sauvegarde (Backup) de votre base de données . Ensuite, nous souhaitons nous assurer que cette sauvegarde est correcte en la rechargeant dans MySQL (opération de restauration). Donner la commande permettant de faire :

- Un backup de votre base de données

- Un restore de votre base de données En expliquant en détail chaque commande utilisée.

**Livrables**

 1 release sur Github contenant :

a. README à la racine contient votre Nom, prénom et classe \*\*\*\*

b. Une archive ZIP avec les éléments UX

c. Une archive ZIP avec les éléments DB \*\*\*\*\*

d. Une archive ZIP avec les éléments POO

e. [Code source inclus automatiquement]

\*\*\*\* c'est seulement cette partie qui sera évaluée pour P\_DB 106

## Livrables

Il n’y a qu’un livrable : une release GitHub, à laquelle est attaché le rapport, au format PDF, contenant :

* 1. Introduction
  2. Planification
  3. Analyse fonctionnelle
  4. Maquettes
  5. Schémas (DB, POO, diagrammes de classe)
  6. Manuel de référence des classes
  7. Rapport de tests
  8. Journal de travail
  9. Chapitre explicatif de l’usage fait de l’IA dans ce projet

# Évaluation

1. Auto-évaluation challengée par le client basé sur des éléments observables.
2. Le recours à des outils en ligne d’intelligence artificielle (ex. : Chat GPT) est autorisé mais ne peut servir que d’inspiration à la réalisation. Chaque développeur doit être à tout moment en mesure d’expliquer le code de manière précise et convaincante.  
   En cas d’abus, l’évaluation du projet en tiendra compte.