PREMIER LOGICIEL

Type :	**PROJET**
Formations :	Ynov Informatique
Promotions :	Bachelor 1
UF :	Développement Logiciel et Base de données

---

1. CADRE DU PROJET

Ce projet permet l'évaluation des compétences acquises grâce aux modules de l'UF « Développement Logiciels » et de « Base de données ». Pour ce faire, ce projet devra être réalisé en groupe de 2.

Vous pouvez soumettre un projet personnel dont le contenu et les fonctionnalités devront respecter des conditions décrites dans la partie « Projet personnel ». Ce projet devra être validé par l'établissement.

Si vous n'avez pas d'idée de projet, vous avez le choix parmi une liste de projets proposés dans la partie « Projets au choix ».

Un bonus sera apporté aux projets personnels et aux groupes qui se challengent en proposant des fonctionnalités plus poussées.

Vous êtes totalement libre quant aux choix technologiques. Nous vous conseillons d'utiliser les langages de programmations et les outils vus avec vos intervenants mais acceptons toute technologie de Développement logiciel. Ex : C++, Java, C#, Python, NodeJS + Electron…

Seul l'aspect fonctionnel sera pris en compte et non l'aspect graphique.

Date de début : ……..

Date de rendu : ……..

2. OBJECTIFS DE FORMATION VISÉS

Vous serez évalués sur les compétences suivantes : *UF Développement Logiciel et Base de données*

PROGRAMMATION

- maîtriser les bases communes à la plupart des langages informatiques
- respecter les conventions du langage utilisé
- effectuer des choix technologiques
- créer un logiciel complet de A à Z
- utiliser des librairies tierces
- mettre en œuvre la persistance des données
- implémenter des actions CRUD

ALGORITHMES AVANCES

- répondre à un besoin fonctionnel par de l'algorithmie
- mettre en place des algorithmes complexes
- optimiser le code couteux

MATHEMATIQUE

- maîtriser la logique booléenne
- manipuler des matrices

3. PREREQUIS

Voici les modules d'enseignement prérequis à ce projet :

- programmation
- mathématique
- algorithmes avancés

4. LIVRABLES

- dépôt GIT de votre logiciel à jour
- exécutable de votre logiciel
- un document de présentation de votre projet (rôles de chacun, technologies utilisées, structure algorithmique, fonctionnalités majeures, captures d'écran…)
- slides de votre présentation

5. MODALITÉS D'ÉVALUATION DU PROJET

Vous serez évalué sur l'ensemble des productions. L'évaluation prendra aussi la forme d'une présentation orale de synthèse d'environ 15 minutes accompagnée d'un support de présentation et d'une démonstration des fonctionnalités du site mises en place.

Le jury sera composé d'une partie des intervenants des cours de l'UF « Développement Logiciels ».

Un temps de questions-réponses d'une durée de 5 minutes sera prévu à l'issue des 15 minutes de présentation.

Des évaluations intermédiaires auront également lieu au cours du déroulement du Projet.

RÉCAPITULATIF DE LA GRILLE DE NOTATION

Le projet personnel décrit ci-dessous comporte 19 points de difficulté. A vous d'y ajouter des fonctionnalités pour atteindre les 23 points minimums requis. Vous pouvez vous référer aux projets proposés pour vous faire une idée des fonctionnalités et des points qui leurs sont attribuées. Les fonctionnalités devront être validées par l'établissement.

Un point bonus sera ajouté à la note finale si vous choisissez de réaliser un projet personnel.

Un point bonus sera ajouté à la note finale si vous dépassez les 35 points de difficulté et que toutes les fonctionnalités mis en place sont fonctionnelles.

Les projets proposés devront être réalisés dans leurs intégralité pour obtenir la totalité des points.

6. DESCRIPTIF DU PROJET

Vous êtes totalement libre quant à l'apparence de vos interfaces. L'utilisation de librairies est autorisée et encouragée afin de gagner en rapidité de développement.

Vous avez la possibilité de choisir entre un projet personnel ou un projet proposé.

LISTE DES PROJETS AU CHOIX :

PROJET PERSONNEL :

Le projet personnel devra atteindre ou dépasser les 23 points en degré de difficulté. Les fonctionnalités obligatoires représentent déjà 19 points en degré de difficulté, vous devez alors trouver des fonctionnalités valant au minimum 4 points. Vous pouvez vous référer à la liste de « Projet au choix » pour vous faire une idée des points alloués par degré de difficulté.

Le projet choisit devra avoir au minimum :

- plusieurs écrans avec système de navigation dont un permettant à l'utilisateur de comprendre le logiciel *Difficulté :* 2
- utiliser une librairie graphique (pas de CLI) *Difficulté :* 3
- au moins un algorithme avancé (ex : génération, IA, analyse...) *Difficulté :* 4
- de l'interaction avec l'utilisateur (ex : bouton, champ texte, événement clavier/souris...) *Difficulté :* 2
- un CRUD sur au moins une donnée utile au logiciel *Difficulté :* 5
- un stockage de données dans un fichier (BDD autorisé mais non obligatoire, selon ordre des modules d'enseignement) *Difficulté :* 3

Degré de difficulté total : 19 points

(Bonus si le projet personnel est choisi)

1er PROJET : CARD GAME

Présentation

Projet inspiré du jeu “Castle Siege: Fantasy Card Game”. Le but est d’augmenter ses données tout en réduisant celles de l’adversaire afin de gagner la partie.

Fonctionnalités

- cinq écrans : *Difficulté :* 2
    - menu principal avec accès aux autres écrans
    - écran de jeu avec possibilité de retour
    - écran de fin de partie
    - écran d’instructions avec possibilité de retour
    - écran d’options avec possibilité de retour
- l’écran d’instructions devra présenter à l’utilisateur toutes les données nécessaires à la bonne utilisation du logiciel *Difficulté :* 1
- modèle de données :
    - deux camps opposés, qui possèdent chacun : *Difficulté :* 2
        - un index de joueur, premier ou deuxième
        - des points de vie, par défaut 100, minimum 0, maximum 100
        - des points de bouclier, par défaut 30, minimum 0, maximum 100
        - 3 ressources différentes, qui possèdent chacune :
            - un fournisseur de ressource, minimum 1
            - un stock de ressource, par défaut 10, minimum 0
        - une main de 7 cartes
    - une carte possède : *Difficulté :* 2
        - un nom
        - un type de ressource à dépenser
        - un coût
        - un type de donnée de camp à modifier (vie, bouclier, fournisseurs et stock)
        - une valeur
        - si elle affecte l’ennemi ou soi-même
        - une probabilité d’apparition
- Déroulement d’un tour :
    - le stock des ressources du joueur se voient incrémenté de 1 par fournisseur qu’il possède *Difficulté :* 1
    - si la main du joueur possède moins de 7 cartes, une carte est piochée en tenant compte des probabilités d’apparition des carte. *Difficulté :* 2
    - le Joueur a deux possibilités :
        - jouer une carte s'il peut payer le coût en ressource *Difficulté :* 2
        - jeter une des cartes de sa main *Difficulté :* 1
    - fin du tour, début du tour adverse *Difficulté :* 1

- fin d'une partie : *Difficulté :* 2
    - la partie se termine si un joueur à ses points de vie à 0 lors du début de son tour
    - le logiciel bascule alors sur l'écran de fin de partie, où diverses informations de la partie seront décrites
- écran d'option :
    - il devra permettre à l'utilisateur de créer, consulter, modifier et supprimer (CRUD) toutes les cartes utilisées dans le logiciel *Difficulté :* 5
    - ces données devront être stockées dans un fichier afin de pouvoir les réutilisées même après fermeture du logiciel *Difficulté :* 3
- deux modes de jeux devront être proposés :
    - mode deux joueurs en local *Difficulté :* 1
    - mode jouer contre une IA *Difficulté :* 5

Degré de difficulté total : 30 points

2ème PROJET : BRICK SHOOTER

Présentation

Projet inspiré d'un mini-jeu compris dans les vieilles consoles All in one.
https://youtu.be/RPTanMNGmek?t=155.

Fonctionnalités

- cinq écrans : *Difficulté :* 2
    - menu principal avec accès aux autres écrans
    - écran de jeu avec possibilité de retour
    - écran de fin de partie
    - écran d'instructions avec possibilité de retour
    - écran d'équipement avec possibilité de retour
- l'écran d'instructions devra présenter à l'utilisateur toutes les données nécessaires à la bonne utilisation du logiciel *Difficulté :* 1
- modèle de données :
    - un joueur possédant : *Difficulté :* 2
        - des crédits
        - une vitesse de tir
        - une vitesse de déplacement
        - une vitesse de défilement
    - des types de bloc possédant : *Difficulté :* 2
        - un type de donnée du Joueur, peut être nul
        - une valeur, peut être positive ou négative ou nul
        - une probabilité d'apparition (il vaut mieux que le bloc n'affectant pas le joueur soit celui de plus forte probabilité)
- pour les types de bloc, avoir au minimum :

    - un bloc neutre à forte probabilité
    - un bloc d'augmentation de vitesse de défilement à faible probabilité
    - un bloc d'augmentation de vitesse de déplacement à faible probabilité
    - un bloc de diminution de la vitesse de tir à faible probabilité
- déroulement d'une partie :
    - le Joueur apparait au milieu en bas de l'écran, il peut se déplacer de gauche à droite jusqu'aux bords de l'écran *Difficulté :* 2
    - des blocs apparaissent aléatoirement en haut de l'écran, faisant défiler vers le bas ceux déjà apparues *Difficulté :* 3
    - le Joueur peut tirer un projectile qui, une fois un bloc heurté, l'active et le casse. Un bloc activé peut modifier une donnée du joueur *Difficulté :* 3
- fin d'une partie : *Difficulté :* 2
    - la partie se termine quand le joueur le décide OU quand un bloc descendu trop bas touche le joueur
    - le logiciel bascule alors sur l'écran de fin de partie, où diverses informations de la partie seront décrites
    - si le joueur a décidé lui-même de quitter, le nombre de blocs qu'il a cassé s'ajoute à ses crédits
    - si le joueur a heurté un bloc, il ne gagne pas de crédits
- écran d'équipement :
    - il devra permettre à l'utilisateur de modifier (moins ou plus) toutes ses données (vitesse de tir, de déplacement, de défilement) monnayant les crédits qu'il a accumulé *Difficulté :* 5
    - ces données devront être stockées dans un fichier afin de pouvoir les réutilisées même après fermeture du logiciel *Difficulté :* 3

Degré de difficulté total : 23 points

7. RESSOURCES COMPLEMENTAIRES

A vous de trouver les ressources nécessaires en fonction des technologies choisies.

C++ : https://www.sfml-dev.org

C++ : https://www.libsdl.org

JAVA : https://openjfx.io

C# : https://openclassrooms.com/fr/courses/218202-apprenez-a-programmer-en-c-sur-net/217501-les-winforms-ou-windows-forms

Python :https://openclassrooms.com/fr/courses/235344-apprenez-a-programmer-en-python/234859-creez-des-interfaces-graphiques-avec-tkinter

ElectronJS : https://electronjs.org

8. BESOINS MATERIELS ET LOGICIELS

- Un IDE
- Un langage de programmation orienté objet
- GIT