Créez votre premier jeu vidéo avec Swift !

100 heures

Énoncé

Mise en situation

Vous voulez devenir un développeur d'application iOS expert ? Pour cela vous devez devenir des magiciens de la Programmation Orientée Objet en démontrant votre compréhension du concept et votre connaissance de Swift !

Dans ce projet nous allons créer un jeu où deux équipes vont s'affronter dans un combat à mort !

> Nous n'allons pas faire d'interface graphique pour ce jeu. Donc tout se passera dans la console.

Voyons à quoi devra ressembler votre jeu et comment il fonctionnera.

Etape 1 : Les équipes

Au début de la partie, chaque joueur va constituer son équipe en choisissant parmi plusieurs types de personnages.

- **Combattant** : L'attaquant classique. Un bon guerrier !
- **Mage** : Son talent ? Soigner les membres de son équipe.
- **Colosse** : Imposant et très résistant, mais il ne vous fera pas bien mal
- **Nain :** Sa hache vous infligera beaucoup de dégâts, mais il n'a pas beaucoup de points de vie.

Consigne

- Chaque équipe doit avoir 3 personnages (peu importe le type).

- Les personnages doivent être nommés par les joueurs. Leur nom doit être unique parmi les personnages de la partie.
- Le personnage "Combattant" démarre avec 100 points de vie et une épée qui ôte 10 points à son adversaire.
- A vous de définir les paramètres des autres personnages !

Etape 2 : Au combat !

Une fois les équipes constituées, la partie démarre. Chacun à son tour, les joueurs effectuent la boucle d'action suivante :

1. Choisir un personnage de son équipe
2. Choisir un personnage de l'équipe adverse à attaquer ou un personnage de sa propre équipe à soigner dans le cas du Mage.

Le programme va ensuite effectuer l'attaque (ou le soin) et indiquer aux joueurs ce qu'il vient de se passer.

Lorsqu'un personnage n'a plus de point de vie, il est mort… Et ne peut pas être réanimé par le mage ! Lorsque tous les personnage d'une équipe sont morts, le joueur a perdu et la partie s'arrête ! 

> Avant l'étape 2, il faut donner la liste des personnages de l'équipe adverse avec les propriétés utiles, pour que le joueur puisse faire un choix informé !

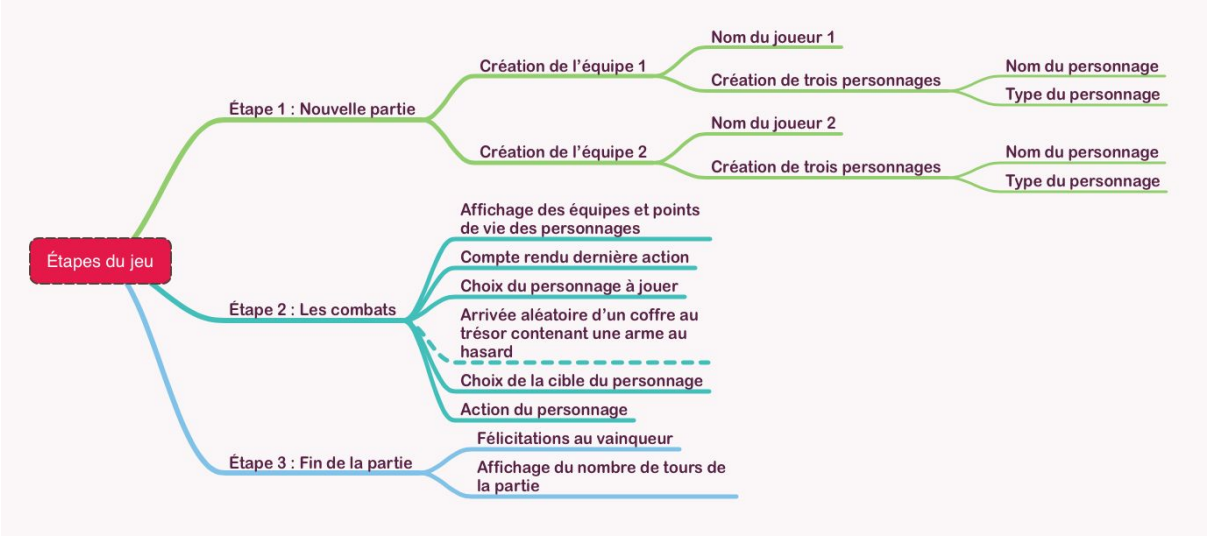
Etape 3 : Changeons d'armes !

Rajoutons un peu de piment à la partie. Dans la boucle d'action du joueur décrite précédemment, nous allons ajouter une étape après l'étape 1.

1. *Choisir un personnage de son équipe*
2. ***Un coffre apparaît devant le personnage, il l'ouvre et… il s'équipe d'une nouvelle arme !***
3. *Choisir un personnage de l'équipe adverse à attaquer ou un personnage de sa propre équipe à soigner dans le cas du Mage.*
   - Cette nouvelle étape aura lieu aléatoirement pour plus de fun !
   - Chaque personnage ne peut porter qu'une arme à la fois
   - Le mage ne peut pas recevoir une arme de type Attaque mais seulement de type Soin (et inversement pour les autres personnages)

En résumé

Merci à **Matthieu Despres** (étudiant du parcours iOS) qui a concocté cette très belle carte mentale qui résume le déroulement complet de la partie.



Bonus

Faire une application, c'est avant tout de la créativité ! Donc montrez votre créativité en ajoutant une fonctionnalité de votre choix !

Quelques exemples pour vous inspirer :

- Un nouveau personnage avec un pouvoir spécial : effrayer, empoisonner, séduire, etc.
- Les statistiques du jeu à la fin de la partie (nombre de tours, etc.)
- Les personnages ont des types (feu, eau, etc) et sont plus ou moins sensibles à certaines attaques
- A vous de jouer !

Je vous suggère de développer le programme en suivant les étapes dans l'ordre (Equipes, puis combat, puis armes, puis bonus). Ce sera plus facile !

Le bonus est obligatoire. Mais vous êtes complètement libre d'implémenter la fonctionnalité de votre choix.

Contraintes

- Le code est sur **Github** avec un historique de commits cohérent.

- Le code est **documenté** avec des commentaires pour chaque fonction, propriété, classe au minimum.
- Le code est écrit en **anglais** : commentaires, variables...
- Le projet ne contient **aucun warning ni erreur**.

Livrables

Vous fournirez les fichiers suivants :

- Le projet Xcode
- Une page (PDF) décrivant brièvement le bonus que vous avez choisi et la façon dont vous l'avez intégré

Pour ce projet, la validation s’effectuera par un mentor validateur lors d’une soutenance filmée.

Votre mentor accompagnateur fera la demande de soutenance quand vous aurez de la visibilité sur la complétion du projet.

Soutenance

La présentation de votre projet : 15-20 minutes

Un moment questions/réponses avec le mentor : 10 minutes

Compétences à valider

- Implémenter un programme Swift
- Comprendre et utiliser la programmation orientée objet