Perret Louis
Orillon Mathilde
B3-A3

IUT CLERMONT - FD
UNIVERSITÉ
Clermont Auvergne

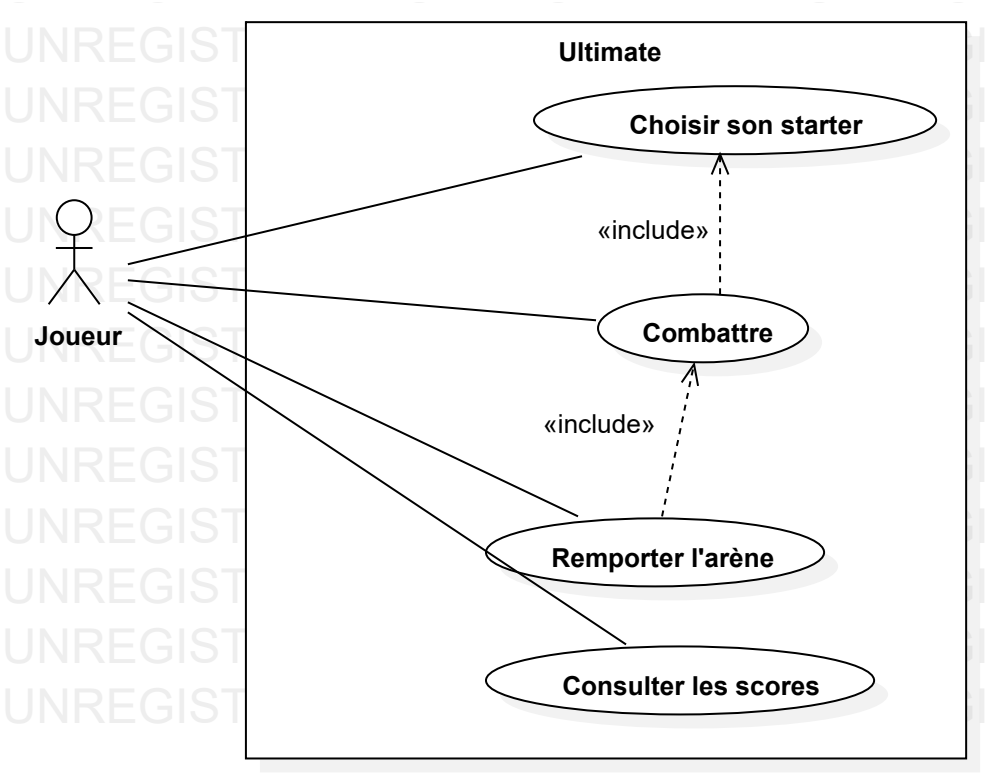
Projet Android : Ultimate

Sommaire :

Contexte

Le but du jeu est de survivre à plusieurs vagues de matériels informatiques (PC, tablette, portables…) sauvages au sein d'une arène. Le joueur pourra choisir son starter parmi les 3 disponibles. Chacun de ces trois starter à ses propres forces et faiblesses. Le joueur devra ensuite se diriger vers un portail, puis une fois entré, il devra vaincre tous les ennemis de la vague. Les combats permettent à 2 adversaires de s'affronter à l'aide de leurs attaques.
Les adversaires s'attaquent chacun leur tour, et la vitesse permet de déterminer qui attaquera en premier. Ces attaques font plus ou moins de dégâts en fonction de leur type et du type de l'ennemi. Une fois sorti vainqueur de la vague, le joueur sera soigné et gagnera de l'expérience lui permettant d'évoluer. Il y a deux paliers d'évolution pour chaque starter jouable. Le joueur devra ensuite vaincre les vagues suivantes dans l'arène, vagues composées d'ennemis de plus en plus puissants, souvent des évolutions d'adversaires déjà vaincus dans des vagues précédentes. Une fois toutes les vagues remportées, le joueur sera sacré maître informatique !

Diagramme de cas d’utilisation



Description des cas d'utilisations

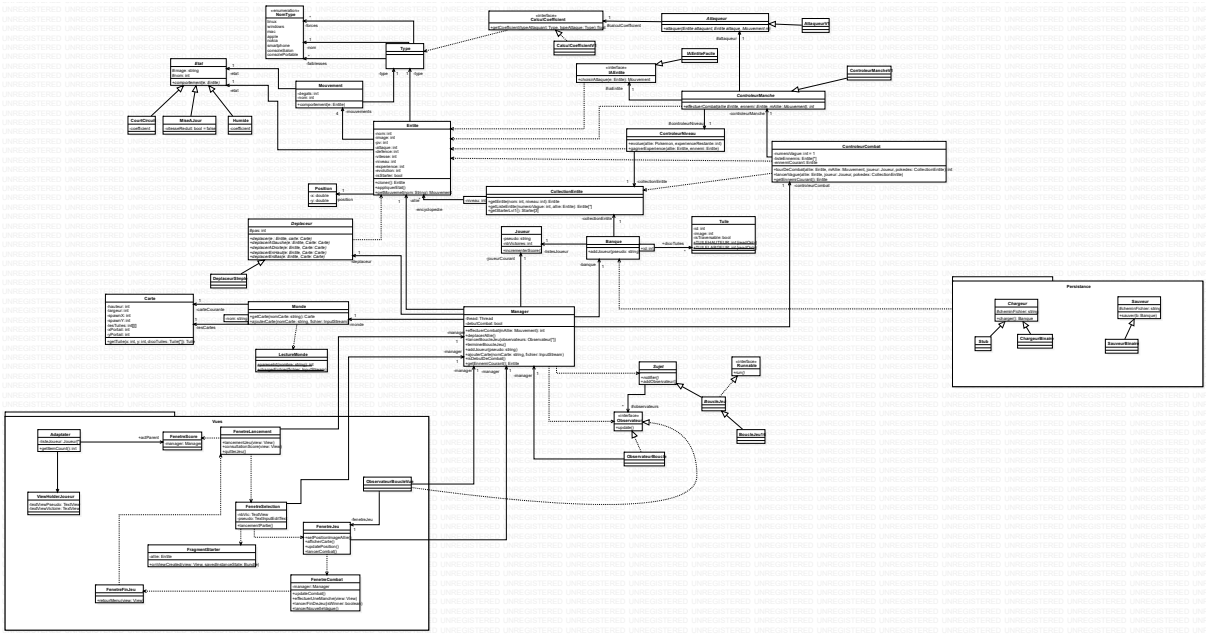
Nom du cas	Choisir son starter
Objectif	Sélectionner le personnage avec lequel on combattra
Acteurs principaux	Joueur
Acteurs secondaires	Aucun
Condition initiale	Le joueur doit avoir cliqué sur le bouton "play"
Déroulement	Les trois starters possible s'affichent à l'écran, puis le joueur clique sur le bouton correspondant à son choix.
Condition de fin	Le joueur a cliqué sur le starter qu'il souhaitait

Nom du cas	Combattre
Objectif	Survivre aux différentes vagues d'ennemis
Acteurs principaux	Joueur
Acteurs secondaires	Aucun
Condition initiale	Le joueur doit avoir choisi son starter
Déroulement	Une fois le starter choisi, le jeu dirige automatiquement le joueur vers le portail qui permettra de lancer la première vague. Puis, le joueur pourra alors attaquer chaque adversaire se présentant à lui afin de remporter la vague. Une fois une vague finie, une nouvelle se lance.
Condition de fin	-Le starter du joueur est mis KO -Le joueur a fini le jeu

Nom du cas	Remporter l'arène
Objectif	Finir le jeu
Acteurs principaux	Joueur
Acteurs secondaires	Aucun
Condition initiale	Remporter toutes les vagues
Déroulement	Affiche une fenêtre montrant la victoire du joueur tout en incrémentant son nombre de victoire de 1
Condition de fin	Cliquer sur le bouton "retourner au menu"

Nom du cas	Consulter les scores
Objectif	Vérifier ces statistiques
Acteurs principaux	Joueur
Acteurs secondaires	Aucun
Condition initiale	Le joueur doit avoir cliqué sur le bouton "Score"
Déroulement	Une fois le bouton cliqué, les scores de chaque joueur s'affichent.
Condition de fin	Appuyer sur le bouton retour

Diagramme de classes



Description diagramme de classes

A. Partie Entité

Cette partie permet d'identifier ce qu'est une Entité:

- Classe Entite: représente une entité d'après beaucoup de données comme son nom, ses points de vies (pv), son attaque, etc
- Classe Mouvement : représente les attaques d'une entité
- Classe Type : Représente le type d'une entité et de ses attaques
- Enum NomType : Énumération composée des noms de chaque type existant
- Classe Position : Représente la position en (x,y) de l'entité
- Classe Etat (et ses filles) : Représente les différents statuts que l'entité peut subir durant un combat. Nous avons utilisé un patron Etat puisque ce dernier nous permet de changer dynamiquement cet état lors d'un combat et modifier le comportement que cela peut avoir sur le entite . En effet, une entité peut n'avoir aucun état, puis une fois empoisonné, il commencera à perdre des points de vie au fil du combat.
- Classe CollectionEntite : Permet de stocker toutes les entités du jeu par niveau dans un dictionnaire.

B. Fonctionnalité combat:

Cette partie permet de gérer les combats entre les entités :

- Interface Attaqueur : Permet de gérer les dégâts que va causer une attaque à une entité suivant plusieurs paramètres : les dégâts de l'attaque ainsi que l'attaque de l'attaquant, la défense de l'attaqué et les affinités des types.
- Interface CalculCoefficient : Permet de calculer le coefficient de dégâts que l'on va ajouter/enlever aux dégâts de l'attaque suivant les affinités des types des deux entités .
- Interface IAEntite : Permet de simuler le choix d'une attaque pour les entités adverses au joueur.
- Classe ControleurNiveau : Permet de contrôler le gain d'expérience, la montée d'un niveau et l'évolution d'une entité .
- Classe ControleurManche : Permet de contrôler une manche d'un combat entre deux entités
- Classe ControleurCombat : Permet de coordonner toutes ces classes entre elles afin d'effectuer correctement un combat entre deux entités .

C. Partie déplacement:

Cette partie gère le déplacement de l'entité du joueur dans notre jeu :

- Classe DeplaceurEntite: Cette classe contient des méthodes changeant les coordonnées du personnage dans 4 directions (haut, bas, gauche, droite) et une méthode appelant ces méthodes en fonction des touches pressées.

D. Partie carte/monde:

Cette partie gère la création des cartes dans lesquelles le joueur joue :

- Classe Tuile: Classe métier de l'objet Tuile, un objet avec une image et un id. Une tuile peut être traversable et des évènements peuvent être attachés à ces tuiles. Les tuiles composent l'environnement graphique dans lequel évolue le joueur.
- Classe LectureMonde: Classe permettant de charger un fichier à partir de son nom et de le parser.
- Classe Carte: Permet de récupérer une hauteur, une largeur, des points d'apparition et un tableau de tuiles à partir du parsage d'un fichier texte.
- Classe Monde: Représente une Map contenant toutes les cartes et leurs noms.

E. Partie Boucle de Jeu

Cette partie permet de gérer notre boucle de jeu que nous avons implémentée afin d'actualiser la position et l'affichage de l'entité sur notre fenêtre. Pour ce faire nous avons implémenté un patron Observateur afin de pouvoir observer cette boucle et d'être notifié à chaque boucle de cette dernière. Cette notification aura pour but de venir actualiser la position du personnage sur la fenêtre :

- Classe Sujet : Représente le sujet observé et possède une collection d'observateurs à venir notifier.
- Interface Observateur : Représente les observateurs qui effectuent un traitement particulier lorsqu'ils sont notifiés.
- Classe BoucleJeu : Représente notre boucle de jeu.

F. Partie persistance:

Cette partie permet de gérer la persistance de notre jeu. Nous avons implémenté un patron Stratégie qui nous permet de changer facilement la façon dont on persiste à l'exécution. A l'heure actuelle, nous sauvegardons nos données sous format binaire, mais si nous souhaitons le faire à l'aide d'une base de données par exemple, cela ne demandera pas de changement conséquent dans notre code :

- Classe Chargeur : Gère le chargement de nos données
- Classe Sauveur : Gère la sauvegarde de nos données

G. Partie vue:

La partie vue correspond à l'interface graphique de notre jeu. Elle est donc composée des classes permettant l'affichage de notre jeu sur un assemblage de conteneurs mais également les contrôleurs.

- Activité:
  - FenetreLancement: Activité appelée en première lors du démarrage du jeu. Elle permet de quitter le jeu où de se diriger vers l'activité gérant la sélection des entités.
  - FenetreScore : Permet l'affichage du score de chaque joueur via l"utilisation d'une recycler view.
  - FenetreSelection: Activité qui permet à l'utilisateur de pouvoir entrer son nom et choisir son starter. Elle possède un fragment FragmentStarter qui permet l'affichage du choix des starter.
  - FenetreJeu: Correspond à la fenêtre de jeu, l'activité sur laquelle notre personnage se déplace. Cette classe réalise l'affichage de la carte en ajoutant des ImageView au conteneur de l'activité.
  - FenetreCombat: Correspond à la fenêtre de combat, là où sont affichés les entités participant à un combat ainsi que l'interface de combat(attaques, points de vie des entités, état). C'est également depuis cette classe que l'on appelle les méthodes permettant de simuler les attaques et ici, que l'action à réaliser en fonction du résultat du combat est réalisée.
  - FenetreFinDeJeu : Activité appelée en cas de défaite/victoire du joueur lors d'un combat. Elle permet de revenir à la fenêtre de lancement.