Projet Clash Royal

Introduction

Nous allons créer une version très simplifiée d'un jeu pour téléphone mobile appelé « Clash Royale » par SuperCell. Le jeu consiste à voir vos unités avancer vers le camp ennemi et lorsqu'elles sont à portée d'une unité ennemie, elles s'engagent dans le combat. Vous pouvez trouver de nombreuses vidéos sur YouTube pour vous donner une idée du jeu de base « Clash Royale ». Le jeu se termine lorsque la tour du roi de l'un des joueurs est détruite. Notre jeu sera automatique (ordinateur vs ordinateur).

Le type Tunite

Une unité peut être à la fois une tour, la tour du roi, un archer, un dragon, un chevalier ou une gargouille. Ce sont les valeurs des champs qui différencient les unités et le champ « nom », de type « TuniteDujeu »

typedef enum{tour, kingTower, archer, chevalier, dragon, gargouille} TuniteDuJeu

Les unités peuvent attaquer une cible terrestre ou aérienne, ou les deux, selon le type Tcible : typedef enum{ground, groundAndAir , air} »

« Le type Tunite est une structure :

typedef struct {

Nom de TuniteDuJeu;

Tcible attaquableCible; indique la position des unités qui peuvent être attaquées

Cartographie Tcible; indique soit « air » ou « sol », utile à savoir

Qui peut nous attaquer

int hitPoints;

float attackSpeed; En quelques secondes, plus c'est petit, plus vite

dommages int;

gamme int; en mètres, distance à laquelle nous pouvons atteindre un

cible

mouvement de flottementVitesse; en m/s

int posX, posY; Position sur le plateau de jeu

int canAttack; gère le fait que chaque unité attaque une fois par tour;

0 = a déjà attaqué, 1 = peut attaquer ce tour

à réinitialiser à 1 au début de chaque tour

int elixirCost;

} Tunite ;

Le lecteur Type

Les joueurs seront vus comme des listes d'unités liées (Tunite) de type TListePlayer.

typedef struct T_cell{

struct T_cell *suivant;

Tunite *pdata; //pointeur vers une unité

} TListePlayer;

Le plateau de jeu

Le plateau de jeu est un tableau bidimensionnel contenant des pointeurs vers des unités.

typedef Tunite **TplateauJeu;

Le plateau de jeu comporte11 colonnes et 19 lignes.

Une tour du roi et une tour défensive feront partie de la liste des unités pour chaque joueur. Les unités (autres que les tours) apparaîtront juste devant la tour

de leur roi (nous ne choisirons pas le lieu d'apparition de manière interactive) et elles emprunteront un seul chemin central pour aller l'un envers l'autre.

Voici un aperçu de ce à quoi pourrait ressembler l'interface de notre jeu, avec

Les coordonnées de quelques points utiles:

Le plateau de jeu (un tableau 2D contenant des pointeurs vers des unités) sera reconstruit à

Chaque tour de jeu grâce aux deux listes d'unités des joueurs. Chaque carré du plateau sera initialisé

avec la valeur NULL (absence d'unité par défaut). Le tableau de jeu n'est utilisé que comme base pour gérer

l'affichage du jeu (en mode console ou en mode graphique). Les combats seront gérés à travers les listes de

unités de chaque joueur.

Chaque unité ayant ses coordonnées (posX et posY), il suffit d'aller à ces coordonnées dans le tableau

de la commission d'écrire l'adresse vers l'unité en question (au lieu de NULL).

Voici un exemple de ce qui pourrait arriver en termes de code pour la corrélation de la carte avec

Les listes d'unités des deux joueurs :

Jeu TplateauJeu;

jeu = AlloueTab2D(11,20); //malloc dans cette fonction...

initPlateauAvecNULL(game); //écrit une valeur NULL dans chaque case du plateau

(absence d'unité)

PositionnePlayerOnPlateau(player1, jeu); //réalise le lien entre le tableau et chaque

unité: game[x,y]=unitecourante->pdata; player1 étant une liste d'unités

PositionnePlayerOnPlateau(player2, jeu); //idem pour le joueur 2

affichePlateauConsole(jeu,11,20); //un affichage console du jeu (avec printf) »

« Nous allons faire un remake très simplifié d'un jeu pour téléphone mobile: 'Clash Royale' de SuperCell.

Le jeu consiste à voir vos unités avancer vers le camp ennemi et lorsqu'elles sont à portée d'une unité ennemie : elles s'engagent dans le combat.

Vous trouverez de nombreuses vidéos sur YouTube pour vous faire une idée du jeu de base 'Clash Royale'. Le jeu se termine lorsque la tour du roi de l'un des joueurs est détruite. Notre jeu sera automatique (ordinateur vs ordinateur).

Une fonction utile, qui devra probablement être codée, est:

Bool tourRoiDetruite (joueur TListePlayer);

Progression du jeu

À chaque tour :

• Phase de combat, qui se décompose en:

o Détection de qui peut frapper qui (qui est à portée). La fonction suivante doit être codée et appliquée à chaque unité de chaque camp :

TListePlayer quiEstAPortee(TListePlayer player, Tunite *uneUniteDeLautreJoueur);

La fonction quiEstAPortee crée et renvoie la liste des unités qui peuvent attaquer uneUniteDeLautreJoueur

o Facultatif : tri de la liste (via un tableau pour gérer le tri) pour déterminer qui frappe en premier (paramètre vitesseAttaque des unités).

o Attaques (soustraction des dégâts des points de vie) des unités de la liste générée par quiEstAPortee sur uneUniteDeLautreJoueur, en vérifiant si chacune peut encore attaquer ce tour (field peutAttaquer). Voici l'en-tête :

TListePlayer combat(TListePlayer player, Tunite *uneUniteDeLautreJoueur);

o Vous devrez supprimer une unité (lorsqu'elle a 0 point de vie) :

Void supprimerUnite(TListePlayer *player, Tunite *UniteDetruite);

Si la dernière unité d'un joueur est supprimée, sa liste est NULL (il est donc nécessaire de passer un pointeur vers cette liste qui peut devenir égal à NULL).

• Phase de mouvement: Une seule unité par case de plateau de jeu sera acceptée. Si la case de destination est déjà occupée, l'unité avance d'une case de moins.

En résumé, vous cherchez à implémenter une version simplifiée du jeu mobile « Clash Royale », où deux joueurs informatiques s'affronteront. Le jeu se compose de tours, où chaque tour se compose d'une phase de combat, d'une phase de mouvement et d'une phase de création.

Dans la phase de combat, le jeu détectera quelles unités peuvent attaquer quelles autres unités, et les unités s'attaqueront les unes les autres si elles sont à portée. Les unités qui peuvent attaquer seront déterminées par la fonction « quiEstAPortee », et l'attaque réelle sera effectuée par la fonction de combat. Toutes les unités vaincues seront supprimées du jeu par la fonction supprimerUnite.

Dans la phase de mouvement, les unités se déplaceront vers le camp ennemi. Les unités ne pourront occuper qu'un seul espace sur le plateau de jeu à la fois, donc si un espace est déjà occupé, l'unité se déplacera d'un espace de moins.

Dans la phase de création, chaque joueur dispose d'une certaine quantité d'élixir, qui est utilisée pour créer de nouvelles unités. Il y a 50% de chances qu'une nouvelle unité soit créée dans cette phase, et l'unité sera sélectionnée au hasard parmi celles que le joueur peut se permettre avec son élixir actuel. La fonction AcheteUnite sera utilisée pour acheter la nouvelle unité, et la fonction AjouterUnite sera utilisée pour l'ajouter à la liste des unités du joueur.

Le jeu se termine lorsque la tour du roi d'un joueur est détruite. La fonction tourRoiDetruite sera utilisée pour déterminer si cela s'est produit.

Phase élixir : une quantité aléatoire d'élixir sera attribuée à chaque camp, entre 1 et 3 (vous pouvez modifier ces valeurs pour améliorer les jeux selon vos avis).

Phase d'affichage du jeu : vous appellerez votre fonction void affichePlateau console (TplateauJeu jeu, int width, int height) pour l'affichage dans la console.

Les routines d'affichage « graphique » suivantes doivent être appelées à chaque tour :

efface_fenetre (pWinSurf);

prepareAllSpriteDuJeu (jeu, WIDTH GAME, GAME HEIGHT, pWinSurf);

maj_fenetre (pFenêtre); SDL_Delay (300); »