IMAC
INGÉNIEUR
ESIPE

Programmation et Algorithmique
Projet
PokImac

1 Présentation

Maintenant que vous êtes des développeurs plus affirmés, il est temps de mettre en pratique vos connaissances dans un projet informatique qui va consister en l'implémentation d'un petit jeu d'aventure basé sur la franchise *Pokemon* de *Nintendo* et *Game Freak*. Voilà une description tirée du site Pokebip
"La série principale Pokémon est celle par laquelle tout a commencé. Le plus souvent développés par Game Freak, ces jeux ont toujours le même principe : vous incarnez un jeune Dresseur de Pokémon à qui l'on offre un Pokémon de départ, et vous partez pour une aventure semée d'embûches. Vous parcourez une région peuplée de Pokémon sauvages que vous pouvez capturer, faire combattre pour les entraîner, et peut-être les faire évoluer."

1.1 Déroulement du jeu

Pokimac va différer du jeu original, nous n'attendons pas un jeu aussi complet que celui présenté dans les captures d'écrans.
Notre jeu affiche une carte dans la console sous forme d'une grille indexée ainsi que les éléments présents sur la carte. Ces éléments sont des créatures qu'on appelle des "PokImacs" et le personnage que l'utilisateur incarne qu'on appellera "Dresseur PokImac".
Il n'y a pas réellement d'objectif précis, le joueur explore la carte pour défier des PokImacs ou juste les découvrir, c'est lui qui s'impose son objectif, soit obtenir les PokImacs les plus forts, soit découvrir tous les PokImacs, soit les collectionner. Tout dépend de son imagination ou du votre !

1.2 Modalité

Le but du projet est donc de faire un programme en $C/C++$ qui va permettre à un utilisateur de jouer à *PokImac*. Le rendu attendu devra comporter les sources du programme et de quoi le compiler (un Makefile ou autre) ainsi qu'un bref rapport décrivant le mode d'utilisation (surtout s'il diffère du sujet), les méthodes utilisées dans l'implémentation, les difficultés rencontrées et votre méthode de travail (qui a fait quoi comment ?).
Le rendu sera suivi d'une soutenance comportant une courte démonstration du fonctionnement et une présentation générale d'implémentation (approx. 5 minutes) qui sera suivie de questions et/ou de retours de notre part.

- Date de rendu : <>
- Date de soutenance : <>

2 Fonctionnalités

2.1 Jeu

2.2 Carte

Une carte va répertorier les PokImacs et le dresseur positionnés sur deux dimensions x et y. Cette carte est délimitée par une certaine longueur et une certaine largeur. Le dresseur peut se déplacer sur chaque case en utilsant quatre directions : haut, bas, gauche, droite.
Le programme doit afficher la carte sous forme d'une grille mais il peut la stocker de n'importe quelle façon, tableau 1D ou 2D, avec ou sans pointeurs, avec une liste chainée ou un tableau associatif (pour les plus téméraires d'entre vous). Le choix de votre représentation virtuelle devra être justifié.
1D ou 2D ?
Un tableau deux dimensions déclaré de la façon suivante

```
int tableau[20][50];
```

en fait un ensemble de tableaux positionnés à la suite dans la mémoire (ou au moins répertoriés par le système). Quand vous faites

```
int a = tableau[1][2];
```

Cela revient à peu près à faire

```
int a = *(tableau + 50 * 1 + 2); // acces a la deuxieme ligne donc apres les 50
    premieres case
```

En supposant que le compilateur ne vous envoie pas balader.
Vous accédez à un certain emplacement mémoire situé après l'adresse pointée par $tableau$. Mais pour que cela fonctionne, le compilateur doit connaitre les dimensions du tableau pour savoir comment indexer les cases. **Tout ça pour dire que un tableau 2D ne peut pas être un double pointeur là où un tableau 1D peut se transférer facilement à un pointeur** et donc se passer de fonctions en fonctions. Vous devez donc y réfléchir avant de choisir comment représenter votre carte.
Si vous faite un tableau 1D, vous pouvez le traiter comme un tableau 2D comme ceci

```
int tableau[largeur*longueur]; // ou malloc(sizeof(int)*largeur*longueur)
int a = tableau[longueur * y + x];
```

2.2.1 PokImac

Le déplacement du dresseur pourra déclencher un événement s'il rentre dans la case d'un PokImac (qu'on peut voir sur la carte ou non, à vous de voir). A ce moment là, le dresseur pour interagir avec. On vous laisse définir les détails de cette interaction, vous pouvez soit le combattre, soit le capturer, soit lui demander de venir avec vous si vous êtes plutôt pacifiste. Vous pouvez aussi proposer ces différents choix à l'utilisateur.
Dans tous les cas, il faudra au moins un affichage qui permet de voir les caractéristiques du PokImac rencontré.
Les PokImac sont des créatures qui se différencient par certaines caractéristiques :

- Son espèce (un nom d'espèce)
- Sa reprensentation dans la console (soit juste un nom, soit un ascii art)
- Son endurance (qui vont définir des points de vie)
- Sa force
- Sa défense

(C'est une liste non-exhaustive, vous pouvez vos rapprocher du jeu original ou rajouter d'autres caractéristiques.)
Selon l'interaction choisie, il faudra utiliser ces caractèrestiques pour définir la challenge de l'interaction. Par exemple si vous choississez la capture, la défense peut définir le taux de chance de pouvoir le capturer. **Pour les oufs** :
Vous pouvez vous rapprocher du jeu original en reprenant le côté stratégique, chaque Pokimac possède 1 à 4 attaques, avec certaines caractéristiques comme son type, sa puissance et ses effets. Demandez à l'utilisateur d'utiliser l'une de ces attaques, ces attaques auront plus ou moins d'efficacité selon l'attaquant et l'opposant.

2.2.2 Joueur

Dans notre jeu, le joueur n'aura trois caractéristiques, son nom, son équipe de pokémon et un inventaire. Dépendant du style de jeu que vous avez choisi, vous définirez les objets qu'il peut avoir dans son inventaire ainsi que leurs finalités. L'objet le plus courant est la Pokiball, elle permet d'apprivoiser (pour ne pas dire capturer) les Pokimacs. On peut aussi imaginer de la nourriture pour nourrir des Pokimacs sauvages.
Au début du jeu, il faudra au moins un pokémon dans son équipe et l'inventaire minimal pour pouvoir jouer.

2.3 UX

Ce projet ne nécessite pas d'interface graphique (Enfin cela dépend de vous, c'est quand même plus chouette d'avoir un jeu graphique). En revanche, il faut un minimum de User-Experience, un minimum d'interaction Homme-Machine.

2.3.1 Menu

Votre programme peut et devrait afficher un menu qui permet à l'utilisateur d'avoir une brève présentation du jeu et de sa configuration. Notamment il peut rentrer le nom de chaque joueur et définir la taille de la carte.

2.3.2 Tour de jeu

Votre programme doit permettre à l'utilisateur de jouer son tour en choisissant une direction pour son déplacement, si le déplacement mène à une interaction il faudra changer le menu en conséquence pour rentrer en mode interaction.
Ce mode affiche le Pokimac rencontré (ou un autre événement) et propose des actions à l'utilisateur. Vous pouvez afficher aussi l'inventaire et l'équipe actuel du joueur. **Note** : Vous pouvez utilisez ce site pour afficher des ASCII Art de pokemon.

2.3.3 Fin de jeu

Lorsque le programme conclut que le jeu est terminé, affichez un écran de fin.

2.4 Pour aller plus loin

Nous avons simplifier le jeu de base pour qu'il soit valable en tant que projet parcourant les bases de la programmation C/C++. Vous pouvez et nous vous encourageons à vous rapprocher un peu plus du jeu d'origine en terme de complexité ou de personnaliser le jeu pour y mettre votre empreinte parce que c'est plus fun :p et c'est un plus pour la notation.
Notamment vous pouvez

- Ajouter d'autres dresseurs Pokimacs
- Ajouter le côté stratégique des combats en ajoutant d'autres caractéristiques et des attaques
- Etoffer l'inventaire
- Ajouter une histoire
- ...

2.5 SDL

La librairie SDL est une librairie, c'est à dire un ensemble de code (compilé ou non) qui répondent à une problématique, qui permet d'abstraire la gestion des fenêtres et des entrées utilisateurs (souris, clavier, manette, ...). Cette abstraction permet notamment d'avoir le même code peu importe sur quel système on code (Linux, Windows, Mac OS, ...), il s'agit d'une librairie *cross-plateform*.

2.5.1 installation

Unix

```
$ sudo apt-get install libsdl2-dev
```

Windows Téléchargez le package SDL https ://www.libsdl.org/release/SDL2-devel-2.0.10-VC.zip ou https ://www.libsdl.org/release/SDL2-devel-2.0.10-mingw.tar.gz. Puis copier les dossiers ***include, bin*** et ***lib*** dans le dossier de votre compilateur où ces dossier sont déjà présents.

2.5.2 Compilation

Unix Lorsque vous compilez votre executable

```
$ g++ `sdl-config --cflags` ...
```

ou

```
$ g++ -lSDL2 ...
```

Windows Lorsque vous compilez votre executable

```
$ g++ -lmingw32 -lSDLmain -lSDL2 ...
```

Pour plus d'info : https ://wiki.libsdl.org/Installation

2.5.3 Code

Une application avec une GUI se construit avec l'architecture suivante :
initialisation
Tant que L'application n'est pas fermé **faire**
faire un pas dans le temps (mettre à jour les différents acteurs de l'application selon le temps et les entrées utilisateurs)
redessiner la fenêtre

Traiter les entrées utilisateurs
Pause de quelques millisecondes pour laisser l'ordinateur souffler
fin Tant que

Avec SDL cela se traduit grossièrement par

```
// Consultez la documentation de SDL pour connaitre les parametres
/*** Init ***/
SDL_Init(...); // Initialisation de la librairie
window = SDL_CreateWindow(...); // Ouvre une nouvelle fenetre
renderer = SDL_CreateRenderer(window, ...); // utilitaire pour l'affichage de
    la fenetre cree
texture = SDL_CreateTextureFromSurface(renderer, IMG_Load(...)); // charger une
     image

running = true;

while(running) // L'application n'est pas ferme
{
  ... // faire un pas dans le temps
  /*** redessiner la fenetre ***/
  SDL_SetRenderDrawColor(renderer, 0, 0, 0, 255); // definit la couleur
      courante (r,g,b,a)
  SDL_RenderClear(renderer); // rempli la fenetre avec la couleur courante
  SDL_RenderCopy(renderer, texture, ...); // affiche une texture dans la
      fenetre
  SDL_RenderPresent(renderer); // rafraichit la fenetre

  /*** Traiter les entrees utilisateurs ***/
  SDL_Event event;
  while(SDL_PollEvent(event)){ // recupere les evenements disponibles
    if(event.type == SDL_QUIT) { // si l'utilisateur ferme la fenetre
      running = false;
    }
  }
  SDL_Delay(30); // Pause de quelques millisecondes
}
SDL_Quit(); // nettoyage des utilitaires SDL
```

Pour plus d'information : https ://wiki.libsdl.org/MigrationGuide

2.5.4 SDLApp

Vous pouvez trouver sur le e-learning une mini sur-couche de la SDL pour simplifier son utilisation. Il suffit d'inclure sdl_app.h et de compiler sdl_app.cpp avec vos propre sources. Le dossier contient une documentation qui décrit le fonctionnement des fonctions présentes dans sdl_app.h.
Le code précédent peut ainsi être remplacé par

```
if(!initialize_app(...))
  return 1;

texture = loadTexture("./resources/tile.jpg");
SDL_Event events[256];

while(app_is_running()) {
```

```
    events_count = fetch_events(events, true); // traite l'evenement SDL_QUIT
    for(int i=0; i<events_count; i++)
    {
      ...
    }
    draw_texture(texture, 0, 0);
    refresh_app_window();
}
return 0;
```

3 Notation

Ce sujet vous donne les grandes lignes du jeu que nous attendons. En donnant un jeu fonctionnel respectant les différents points du sujet (On peut faire une partie, trouver des Pokimacs et terminer le jeu) et une présentation claire et pertinente, vous aurez une note autour de 15-16.
Vous pouvez simplifier certains points comme il est indiqué à certains moment, mais dans ce cas là la note tournera plutôt autour de 12 (vous devez quand même avoir un jeu fonctionnel).
À contrario, si vous ajoutez des fonctionnalités en plus comme il vous ai proposé dans la section 2.4, vous pouvez viser le 20.

Attention : vous avez une certaine liberté sur les traits du jeu **mais** vous devez respecter à minima les fonctionnalités listées dans le sujet. Si on ne retrouve pas les différents points abordés dans le sujet, il nous manquera des points à attribuer. Si vous tenez à vous eloigner du sujet, il faudra en parler d'abord aux responsables du cours.

Note : Votre chargé de TD sera à votre disposition pour répondre à vos questions, n'hésitez pas à lui envoyer un message

4 Exemples

A venir