Projet d'Algorithmique
L'Ascenseur

Victor ALAIN - Arthur BLAISE - Amélie GUEDES
- Loann POTTIER

Mai-juin 2019

Table des matières

Introduction

1.1 Avant-propos

Le jeu de l'ascenceur est un jeu de cartes complexe qui nécessite, pour y jouer, de bien maîtriser les règles. Pour reproduire ce jeu il nous fallait donc évidemment savoir y jouer, c'est dans ce but que nous avons essayé de synthétiser les règles essentielles à la création de l'algorithme. L'une des difficultés majeures a sans doute été la distribution des points à la fin de chaque manche, ou le listage des cartes qu'il est possible de jouer.

Mais au final, cette partie réelle nous a permis de relever les points importants, et d'avoir une schématisation plus concrète du déroulement du jeu. Nous avons essayé de résumer les étapes principales du développêment ici.

1.2 Problèmes à résoudre

Le but de ce projet est de réaliser un algorithme permettant d'exécuter le jeu de l'Ascenseur. Voici la liste des problématiques qui se sont présentées :

- La première partie concerne la configuration du jeu : en effet il existe plusieurs variantes des règles, comme le nombre de joueurs, ou le nombre de points gagnés par manche... Il nous faut donc trouver un moyen pour laisser aux joueurs la possibilité de les modifier.

  Ce moyen a pris la forme d'un fichier config.txt qui résume sept options différentes : le nombre maximal de joueurs (humains et bots), le nombre de points gagnés par défaut à une manche remportée, le nombre de points gagnés ou perdu par pli, la taille en pixels de la fenêtre de jeu, et les limites d'âge des joueurs (qui sont demandées pour départager les joueurs).

- Ensuite vient le développement d'une partie : gérer le nombre de manches par phase (qui varie en fonction du nombre de joueurs), le nombre de tours par manche, la distribution ou la remise des cartes... à ce stade même la question de qui doit recevoir ses cartes en premier semble complexe.

Lors d'une manche il faut aussi compter le nombre de plis remportés par chaque joueur, sans oublier de lui demander son pari au début. De surcroît chaque joueur ayant remporté une manche doit débuter la manche suivante. Il est important également de contrôler quelle carte chaque personne joue, pour vérifier que la couleur soit bien valide. Chaque joueur joue une carte de la couleur demandée s'il en possède une, sinon il doit jouer un atout. Cependant, un joueur n'est pas obligé de poser un atout s'il n'a pas de carte de la même couleur, il peut déposer une carte quelconque mais il perdra le tour. Il est aussi à noter que les jokers n'ont pas été inclus dans le jeu.

- La fin de la partie est heureusement simple à créer : le joueur ayant le plus de points gagne, point final.

- En annexe, une autre partie de code est ajoutée : la gestion des bots. Maintenant que les humains peuvent jouer, il faut leur permettre de jouer contre un ordinateur. C'est une partie assez ardue à modéliser puisqu'il faut prévoire plusieurs 'personnalités' d'Intelligence Artificielle, et créer l'algorithme correpondant.

FIGURE 1 – Jeu de carte utilisé lors d'une partie d'ascenseur

Explication des algorithmes

Pour commencer, voici comment est construit le code. Tout part d'un programme principal *start.pas* : c'est ce programme qui lance toutes les fonctions secondaires et les appellent dans le bon ordre, afin de permettre le bon déroulement du jeu. De surcroît ce programme utilise des unités, chacune s'occupant d'une partie spécifique du jeu.

2.1 Structure du programme

Programme principal

Le programme principal *start.pas* demande à chaque joueur de renseigner son pseudo et son âge, qui seront utilisés dans le reste de la partie. Dans ce programme il y a également l'utilisation des *unités*, ce qui évite d'avoir un fichier trop long (en segmentant le code, il devient plus lisible, et plus facile à tester).

Ensuite, durant le déroulement du programme les choses se complexifient car le déroulement du jeu est définit dans deux unités. L'une demande si le joueur souhaite afficher les règles (unit *intro.pas*), et l'autre contient chaque étapes du jeu (unit *deroulement.pas*).

Afin d'assurer le déroulement visuel du jeu l'aspect graphique est contenu dans une autre unité : *graph.pas*. Celle-ci fonctionne un peu comme une lib graphique spécialement conçue pour le jeu de l'Ascenceur, en appelant d'elle même les unités gLib2D, SDL et autres. Le fait de l'avoir concentrée en un seul fichier permet de ne pas avoir trop d'appels ou d'imports dans le reste du code. Dans ce même but, l'unité *classes.pas* rassemble toutes les classes utilisées dans le jeu.

2.2 Unités utilisées

Introduction

Cette unité est particulièrement simple car elle se contente de proposer à l'utilisateur la lecture des régles du jeu. Rapide et efficace.

Déroulement

Cette unité est initalement le coeur du programme ; elle fait donc appel à 11 fonctions et 9 procédures différentes. Ces dernières sont :

Initjoueur : Demande le nombre de joueurs présent dans la partie

Initplimanche : Permet de ramener le nombre de plis des joueurs a zéro avant la nouvelle manche

Inarray : Vérifie la présence d'une carte dans la liste

Init : Crée le paquet de carte

Distribuer : Distribue les cartes aux joueurs et empêche le doublon de carte

InitAtout : COOKIE

NombreManche : Calcul le nombre de manches nécessaires vis-à-vis du nombre de joueur

VerifcouleurExiste : Vérifie la couleur de cartes séléctionner par le joueur

VerifvaleurExiste : Vérifie la valeur de la carte choisie

VerifieCareAjoueur : Vérifie que le joueur possède la carte qu'il a selectionné

ChoixCarte : Applique les verifications sur le choix des cartes

VerifDroitDePoser : Vérifie si la cartes peut-être poser sans enfreindre les régles

Pli : A la fin d'un tour, renvoie un entier pour désigner le gagnant

RetirePaquet : Retire la carte du paquet une fois sélectionner

OrdreJoueur : Permet de determiné et d'appliquer l'ordre de jeu des joueurs pour le prochain pli

AffricheScore : COOKIE

Manche : Applique les etats d'une manche a savoir le nombre de cartes au joueurs et l'enregistrement des paris

Ascendant : Donne une carte de plus a chaque joueur

Descandant : Retire une carte a chaque joueur

ComptageDePoint : Compte le nombre de point associé aux joueurs

Creerjoueurs : COOKIE

Partie : COOKIE

Unité graphique

C'est le morceau de code qui permet de gérer toute l'interface graphique : de l'affichage du fond d'écran jusqu'au clic sur une carte, en passant par la saisie des paris ou les animations discrètes, tout est géré ici puis appelé dans le programme.

init : Initialise la fenêtre avec une taille, et crée la référence de l'image de fond

set_deck : Initialise la liste des cartes présentes dans le deck

set_cartes_main : Définit les cartes dans les mains du joueur

set_joueur : Définit le joueur qui doit être affiché en focus (indicateur en haut à gauche de la fenêtre)

set_fps : Initialise le nombre d'images par seconde

convert_carte : Complète les informations d'une carte pour la rendre utilisable par la lib graphique

load_players : Complète les informations d'un joueur pour le rendre utilisable par la lib graphique

convert_text : Convertit un texte en élément graphique

convert_couleur Convertit une couleur en bits (utilisé par le terminal) en une couleur graphique

afficher_cartes : Affiche la liste de cartes définie par *set_deck*

afficher_joueurs : Affiche la liste des joueurs autours de la "table"

afficher_background : Affiche l'image de fond

afficher_texte : Affiche un message

refresh : Rafraîchit l'affichage de la fenêtre

focus_joueur : Affiche le pseudo du joueur focus en haut à gauche de l'écran

afficher_atout : Affiche la couleur de l'atout actuel

afficher_manche : Affiche la liste des cartes définies par *set_main*

afficher_cadre : Affiche un cadre autour de la carte survolée par la souris

on_click : Retourne la carte où le joueur a cliqué

saisir_txt : Demande à l'utilisateur de saisir du texte

afficher_score : Affiche le score des joueurs dans une zone blanche, au milieu de la fenêtre

Difficultés

3.1 Coeur du programme

COOKIE

3.2 Unité grahique

La principale difficulté rencontrée lorsque nous avons commencé la partie graphique est l'installation : malgré de nombreuses heures de tests, il nous a été impossible d'utiliser la lib SDL sur nos ordinateurs personnels. Cela nous a donc forcé à la travailler uniquement depuis les postes de l'école, ralentissant la progression de cette section du programme.

D'autres problèmes sont apparus au fur et à mesure du code : par exemple il fallait pouvoir modifier la taille de la fenêtre afin de l'adapter à la taille de l'écran, ce que ne proposait pas nativement gLib2D. Nous avons donc dû modifier cette unité pour ajouter une fonction qui modifie l'échelle de la fenêtre, elle-même appelée par la procédure *init*.

Certains défis se sont révélés intéressants à relever. Comment afficher les 52 cartes différentes sans avoir à les sauvegarder une par une sous forme d'image ? Nous avons opté pour la formation d'un rectangle blanc avec du texte superposé pour afficher la valeur et la couleur. Comment afficher les cartes, joueurs et autres items plus de 50 fois par seconde sans mettre à plat la mémoire libre de l'ordinateur ? Ils sont générés une seule fois puis stockés dans une variable de l'unité. Comment demander à l'utilisateur de saisir du texte ? Nous affichons une sous-fenêtre invitant l'utilisateur à écrire, puis on interrompt le programme afin de détecter les clics sur le clavier et afficher le résultat en temps réel. Et encore bien d'autres...

Avantages et limites

Contribution des étudiants

Arthur Blaise

Arthur s'est concentré sur l'utilisation de l'unité gLib2D, et la création de la lib graphique, pour pouvoir l'implémenter plus tard dans le programme. En annexe, il y a aussi eu la gestion du fichier de configuration.

Amélie Guedes et Victor Alain

Amélie et Victor se sont occupés du plus important, le coeur du programme. De la création des joueurs à la fin de la partie, en passant par le déroulé de chaque manche, chaque pli.

Loann Pottier

Loann a rejoint le projet en cours de route et, faute de pouvoir coder, s'est appliqué à comprendre le programme pour le résumer dans ce rapport.

Bilan personnel

6.1 Amélie Guedes

6.2 Arthur Blaise

Pour ma part, j'ai beau avoir une certaine expérience de la programmation, ce projet a été mon premier vrai travail avec une interface graphique. J'ai pu découvrir le monde du SQL, la gestion des événements, l'utilisation du cache pour limiter les fuites de mémoire, et plein d'autres choses passionnantes. Même si j'ai passé de longues heures à m'arracher les cheveux sur des problèmes d'apparence simples et à pester contre Pascal, je n'en reste pas moins fier du travail accompli. Ce jeu est tellement complexe à modéliser, sans même parler de l'affichage, que notre programme vallait la peine d'être créé. Et tant pis pour les longues heures noctures derrière mon ordinateur...

6.3 Victor Alain

6.4 Pottier Loann

Conclusion

QUI AIME LES COOKIES ???