**Rapport Projet Video Poker - Developpement B1 2009-2010**

**SuPoker**

**Introduction**

Ce rapport a pour but de présenter nos travaux concernant le projet de développement B1 2009-2010 qui consiste à coder un jeu de vidéo poker en C ainsi qu'un site de promotion en HTML, CSS et Javascript.

Dans un premier temps nous présenterons le projet, les membres qui y ont participés et la répartition du travail.

Puis nous parlerons de la partie développement en C du projet, c'est à dire les algorithmes et le codage du jeu. Donc nous présenterons les choix théoriques et techniques de notre projet. En outre nous expliquerons les différents algorithmes utilisés (qui sont également disponibles dans le dossier "Algorithme" ou dans la section "Algorithme" du site web.), et nous décrirons les différents modules (jalons) composant le jeu. Suite à cela, nous aborderons les difficultés rencontrées, ainsi que les solutions apportées. Pour terminer sur cette partie du projet, nous présenterons les optimisations réalisées, ainsi que les ajouts par rapport au cahier des charges. Et enfin nous énumérerons les ajouts possibles.

Nous passerons ensuite à la partie web du projet. Comme précédemment, après avoir présenté le rôle du site web, nous présenterons les différentes technologies utilisées ainsi que les différents aspects techniques de cette partie.

Enfin, nous terminerons par une conclusion, qui en outre, expliquera ce que nous a apporté ce projet.

**Partie I - Présentation du projet**

**A. Avant propos**

Ce projet s’inscrit dans la dynamique des projets de développement de la promotion B1 de Supinfo. Le but est de créer un jeu de vidéo poker en langage C, ainsi qu'un site web de promotion, en HTML/CSS et JavaScript.

L'équipe de ce projet est constituée de Clément Stroppa ( 92692 ), Gautier Labarre ( 93762 ), Cyril Maitre ( 92807 ), Raphael Walter ( 92582 ) et d'Alexandre Dubois ( 95222 ).

En outre, nous avons donné à ce projet le nom de SuPoker pour la concaténation de Supinfo et Poker.

**B. Répartition du travail**

Nous avons scindé le projet en 5 grandes parties.

**1. Elaboration et Ecriture des Algorithmes**

Cette partie consiste à élaborer les algorithmes que nous utiliserons pour coder le jeu.

Tous les membres du groupe y ont participé en exposant leurs idées.

*NB: Ces algorithmes sont expliqués plus loin dans le rapport. En outre ils sont également disponible dans le dossier "Algorithme", ou bien dans la section "Algorithme" du site web.*

**2. Ecriture du Code C**

C'est la partie « codage », c'est-à-dire l'écriture du programme (ici en langage C). Nous avons bien sur utilisé les algorithmes écrit dans la première partie, cependant ils n'ont pas tous été respectés à la lettre. Certains ont été modifiés et repensés, pour améliorer ou simplifier le code, au fur et à mesure de son écriture et des problèmes rencontrés.

Cette partie a été réalisée par Cyril Maitre et Raphael Walter.

*NB: Vous trouverez les codes source dans la section "Sources" du site web, ou bien directement dans le dossier du même nom.*

**3. Optimisation du Code C**

Une fois l'écriture du code terminé et le programme fonctionnel, nous avons réalisé plusieurs optimisations pour rendre le jeu plus fluide et plus opérationnel.

Nous avons tous participé à cette partie.

**4. Ecriture du Site**

C'est dans cette partie qu'a été réalisé tout le site, du plan au design. Gautier Labarre et Clément Stoppa ont travaillés dessus.

**5. Finalisation du projet**

Une fois toutes les étapes précédentes terminées, nous avons finalisé le projet, c'est à dire corriger les derniers petits bugs du programme et du site web. Mais nous avons passé aussi du temps à faire un PowerPoint pour la soutenance de projet, ainsi que pour rédiger ce rapport. Nous avons en outre rassemblé tous les documents du projet pour constituer le rendu final.

Nous avons tous participés à cette finalisation.

En découpant le projet en ces 5 étapes, nous nous sommes partagés les tâches de façon optimal. En effet nous pensons qu'il vaut mieux qu'une personne reste sur une tâche assez précise (codage en C ou le site Web), plutôt que de passer de l'un à l'autre (ce qui met plus de temps). Aussi le fait d'avoir tous participé à l'élaboration des algorithmes a permis d'avoir plus d'idées et d'apprendre à travailler à plusieurs. En outre nous avons passé aussi du temps à mettre nos travaux en commun pour que chacun prenne connaissance des techniques utilisé pour réaliser telle ou telle tâche.

**Partie II - Algorithme et Codage en C**

**A. Choix théorique et technique**

**1. IDE (Environnement de développement intégré)**

Nous avons utilisé Visual Studio 2008 puisqu'il est l'un des meilleurs (voire le meilleur) IDE pour la programmation C / C++. Ceci a garanti un gain de temps et ainsi de productivité. De plus Supinfo nous fournit une licence gratuite.

**2. Librairie graphique**

Nous avons utilisé la SDL. En effet c'est une bibliothèque simple, efficace et libre qui est également très utilisée. De plus un tutoriel complet est disponible sur le SiteduZero (http://www.siteduzero.com/tutoriel-3-14080-installation-de-la-sdl.html)

**3. Langage Web**

* xHTML / CSS
* Javascript

**4. Autre logiciel**

* Gcc - Pour la compilation sous Linux.
* Geany et notepad++ - Editeur de texte pour le codage du site web.
* Photoshop, Paint, et Gimp - Editeur de dessin.
* Microsoft Office - Pour l'édition de texte, le PowerPoint et le tableau Excel
* Subversion - Pour le partage de fichier

**5. Système d'exploitation**

* Windows Seven - 64 bit
* Ubuntu 10.04 - 64 bit

**B. Code - Les différents modules**

Nous avons découpé le code en plusieurs modules. Ceci permet d'une part de scinder, de manière logique, le code et ainsi fournir une meilleure lisibilité. Et d'autre part, la possibilité de travailler sur 2 modules différents, simultanément.

**1. Analyse des cartes**

Ce module permet d'analyser les cartes qui sont sur le tapis et ainsi connaitre la combinaison du joueur.

Pour cela, nous avons préalablement attribués, de façon logique, des id aux différentes cartes (ces ID sont disponible dans le header defineCarte.h). Ensuite les cartes du tapis sont rangées dans un tableau en fonction de leurs ID (du plus petit au plus grand). Ensuite à l'aide d'une cascade de "if... else", nous pouvons déterminer la combinaison obtenue par le joueur.

*/!\ Pour plus de détails concernant ce module, je vous invite à lire le fichier "Algorithme - Analyse des cartes".*

**2. La gestion des cartes**

Ce module permet de gérer les cartes, c'est à dire la distribution aléatoire du premier et du second tirage.

Le mélange des cartes se fait à l'aide d'indice généré aléatoirement et attribué à chaque carte. Ces cartes sont ensuite rangées dans une liste en fonction de leurs indices. Ceci permet donc d'avoir une liste contenant des cartes dans un ordre aléatoire. Cela représente le sabot du jeu.

Pour le premier tirage, il suffit de transférer les 5 premières cartes de la liste sabot. Ces cartes sont ensuite transférées dans une liste "tapis", et sont rangées en fonction de leur ordre de tirage. Puis les id contenus dans ces listes sont placés dans un tableau dans l'ordre croissant des indices.

Pour le second tirage, on analyse si la carte est gardée ou non (à l'aide de l'état des boutons "Garder"). Si la carte n'est pas gardée, elle est retiré de la liste tapis et est remplacé par la première carte de la liste sabot. Donc concrètement, on retire les cartes non gardées et on les remplace par les premières cartes de la liste sabot

*/!\ Pour plus de détails concernant ce module, je vous invite à lire les fichiers "Algorithme - Ajout d'une carte à une liste de façon aléatoire", "Algorithme - Ajout d'une carte à une liste en fonction de son indice", et "Algorithme - Attribution d'un indice unique et généré aléatoirement à une carte".*

**3. La gestion des events**

Ce module permet de gérer tous les événements, c'est à dire toutes les interruptions du jeu. En fonction des actions du joueur, certains "flags" (indicateurs), seront modifiées et cela affectera le jeu (notamment au niveau graphique). En effet la structure STRUCT\_etatSurface et STRUCT\_statistiques, contiennent des variables de type int qui sont des flags, en fonction de la valeur qu'ils contiennent, permet de changer l'aspect du jeu. Par exemple, lorsque le joueur clique sur un bouton ("aide" par exemple), l'état de ce bouton est passé à "down" (appuyé). Ensuite lorsqu'il sera affiché, ce sera l'image du bouton appuyé qui sera affiché, rendant ainsi un effet de pression.

**4. Gestion des menus**

Ce module permet d'afficher les éléments nécessaires pour chaque menu, en appelant les fonctions adéquates du module "interfaceGraphique". Ainsi il suffit d'appeler les bonnes fonctions pour avoir le menu souhaité. Par exemple pour afficher le menu de mise, on appellera la fonction affichageBackground (qui permet d'afficher le background du jeu), affichageJeton (qui permet d'afficher les jetons pour la mise) ... etc.

En outre l'affichage du menu "Aide" et "A Propos" est quelque peu différent, puisqu'il permet d'afficher le menu sans perturber le tour de jeu. En effet si le joueur clic sur le bouton "Aide" lorsqu'il a tiré ses cartes, lorsqu'il quittera le menu Aide, il retrouvera les cartes précédemment tirées.

*/!\ Pour plus de détails concernant ce module, je vous invite à lire le fichier "Algorithme - Affichage du menu Aide et A propos".*

**5. La gestion des Mises et Gains**

Ce module permet de gérer la mise et le gain. Lorsque le joueur clique sur un jeton pour augmenter sa mise, ce module permet de savoir si le joueur a assez de jetons par exemple.

En outre si le joueur n'a effectivement pas assez de jetons (par exemple s'il clique sur le jeton 200 alors qu'il n'en a que 50), la totalité de ses jetons sera ajoutée à la mise, et ses jetons passeront alors à 0.

Enfin cette fonction permet de gérer les gains du joueur. Pour cela un switch est fait sur la variable combinaison (qui contient la combinaison du joueur). En fonction du résultat, son nombre de jetons variera.

**6. Gestion du texte**

Dans ce module, sont regroupées toutes les fonctions d'affichage du texte. Il permet en partie la gestion des variables Jeton, Mise, Gain, Nombre de main gagnée, Nombre de main perdue, Gain maximum et Gain total.

**7. L'interface graphique**

Ce module est composé de fonctions qui permettent chacune d'afficher un élément précis de l'interface graphique. Dans chaque fonction, on analyse l'état de l'élément (qui est contenue dans la structure etatSurface) à afficher. En fonction de cet état, telle ou telle image sera chargée à l'écran. Par exemple si l'état du background est "backgroundStart", on affichera le background correspondant au menu de démarrage. Pour déterminer l'état de chaque surface, on utilise un switch. Ces fonctions sont appelées pour former l'aspect graphique du jeu.

**Voici les différentes fonctions:**

affichageBackground: Permet d'afficher le background du jeu. La variable background enregistre l'état du background, ce qui permet d'afficher tel ou tel fond en fonction de la phase de jeu. Par exemple si on se trouve dans la phase Mise, le background sera différent que si l'on se trouve dans le menu principal.

affichageBackTexte: Permet d'afficher le texte "Jeton", "Mise", "Gain", "Nombre de main gagnée", "Nombre de main perdue", "Gain Maximum" et "Gain Total". Ce texte n'est en fait qu'une image qui est affichée via cette fonction. Il n'y a pas d'état de surface pour cet élément puisqu'il est toujours affiché de la même façon.

affichageTapis: Permet d'afficher les 5 cartes du tapis. Un tableau état surface de 5 cases permet d'enregistrer les 5 ID des cartes du tapis. Ensuite les cartes sont affichées à l'écran en analysant ces ID (en parcourant le tableau) avec un switch.

affichageBoutonGarder: Permet d'afficher les 5 boutons "Garder" en dessous des 5 cartes. Quand le joueur clique sur un bouton, son état est inversé (Si le bouton est UP, il passe à DOWN, et vice versa). Ensuite en analysant ces états (qui sont contenus dans un tableau), on peut afficher les 5 boutons.

affichageBoutonAide: Permet d'afficher le bouton "Aide" en bas à gauche de l'écran. L'état de cette surface est UP ou DOWN (Appuyé ou relâché).

affichageBoutonApropos: Permet d'afficher le bouton "A Propos" en bas à gauche de l'écran. L'état de cette surface est UP ou DOWN (Appuyé ou relâché).

affichageBoutonCarte: Permet d'afficher le bouton "Carte" en bas à droite. L'état de cette surface est UP, DOWN ou DISABLED (Appuyé, relâché ou désactivé).

affichageTexteBas: Permet d'afficher le texte en bas de l'écran. Ce texte n'est en fait qu'une image. Il change en fonction de la phase de jeu et de la combinaison du joueur. En effet lors de la phase "Second Hand", cet élément indique au joueur la combinaison qu'il obtenue.

affichageTexteHaut: Permet d'afficher le texte en haut de l'écran. Ce texte est aussi une image et son état change en fonction de la phase de jeu.

affichageBoutonStart: Permet d'afficher le bouton "Nouvelle Partie", "Reprendre la Partie" et "Options" du menu principal. Le bouton "Reprendre la Partie" peut avoir l'état UP, DOWN ou DISABLED (Appuyé, relâché ou désactivé), car il est désactivé pour le moment. Les deux autres boutons peuvent avoir les états UP ou DOWN (Appuyé ou relâché).

affichageBoutonContinuer: Permet d'afficher le bouton "Continuer" en bas à gauche dans le menu Game over. L'état de ce bouton est UP ou DOWN (Appuyé ou relâché).

affichageBoutonRetour: Permet d'afficher le bouton "Retour" du menu "Aide", "A Propos" et "Options". L'état de ce bouton est UP ou DOWN (Appuyé ou relâché).

affichageJetons: Permet d'afficher les différents jetons pour la mise (1, 2, 5, 10, 20, 50, 100, 200, All-In et Reset). Ces jetons ont pour état, UP ou DOWN (Appuyé ou relâché).

*/!\ Pour plus de détails concernant ce module, je vous invite à lire le fichier "Algorithme - Affichage d'un élément à l'écran".*

**8. Le main**

Ce module permet de mettre en relation tous les modules précédents et ainsi permettre le bon fonctionnement du jeu.

*/!\ Pour plus de détails concernant ce module, je vous invite à lire le fichier "Algorithme - Un tour de jeu".*

**C. Les difficultés**

**L'interface graphique**

La gestion des menus et la transition d'un menu à l'autre était assez difficile.

**Tirage de carte de manière aléatoire**

Il était aussi difficile de mettre en place un système qui permet de tirer des cartes de manière aléatoire. En effet la fonction rand et srand ne sont pas infaillible.

**Memory Leak (fuite mémoire)**

Les fuites mémoire ont posé quelques soucis. En effet, le jeu prenait de plus en plus de place mémoire au fur et à mesure qu'il était exécuté.

**Analyse des cartes**

C'est à dire déterminer la combinaison obtenue par le joueur.

**Compilation sur Linux**

La création d'un makefile est assez complexe.

**D. Solution:**

**1. L'interface graphique:**

Pour résoudre le problème de l'interface graphique nous avons d'abord choisi une librairie qui permet de la générer simplement et avec efficacité. De plus, nous avons découpé l'interface graphique en différentes fonctions. Cela simplifie la "construction" des différents menus, puisqu'il faut simplement appeler les fonctions adéquates pour afficher les éléments voulus à l'écran.

De plus l'utilisation de variables (regroupées dans une structure) qui contient les différents états de toutes les surfaces, a grandement simplifié l'implémentation de l'interface graphique.

**2. Tirage de carte de manière aléatoire**

Pour le tirage de cartes aléatoire, nous avons fait plusieurs tests sur la fonction RAND et SRAND. Ainsi nous avons trouvé une solution pour le mélange des cartes et ainsi avoir des tirages complètements aléatoires.

En effet, comme expliqué précédemment, la fonction rand, nous permet de générer 52 nombres aléatoire et unique. A la suite de quoi, nous attribuons ces nombres aux différentes cartes qui sont ensuite rangées en fonction de leurs nombres attribués.

**3. Memory Leak (fuites mémoire)**

Pour régler le problème de la memory leak, nous avons utilisé le système de fonction mis en place pour l'interface graphique. En effet la mémoire est gérée dans chaque fonction, c'est à dire que nous allouons la mémoire en début de fonction, et nous la restituons en fin de fonction. Ainsi la mémoire est mieux gérée, ce qui a permit de ne plus avoir aucunes fuites mémoire.

**4. Analyse des cartes**

Pour résoudre le problème de l'analyse des cartes, nous avons simplement passé beaucoup de temps à analyser le problème pour élaborer un algorithme à la fois simple et efficace. Cet algorithme est expliqué dans le fichier "Algorithme - Analyse des cartes".

**5. Compilation sur Linux**

Concernant la compilation sur linux, nous nous sommes aidés d'informations trouvées sur internet.

**E. Optimisation**

Nous avons effectué quelques optimisations au cours du projet. Les deux principales sont l'utilisation de listes au lieu de tableaux. Ainsi l'espace mémoire est mieux utilisé.

Mais aussi le rafraichissement des éléments nécessaires, c'est à dire uniquement des éléments cliquables. Par exemple, lors de la phase mise, lorsque le joueur cliquera sur un jeton, seul les jetons, les boutons et les statistiques seront rafraichit à l'écran. Le background par exemple ne sera pas rechargé. Ceci permet d'augmenter la fluidité et d'alléger le jeu.

**F. Les améliorations apportées**

Nous avons également ajouté quelques éléments au jeu qui n'était pas demandé dans le cahier des charges. Ces ajouts sont mineurs et peuvent être retiré, selon le souhait du client, sans problème de la solution.

Ces ajouts comportent:

Un menu principal

L'ajout de statistiques (Gain max, gain total, main perdue, main gagnée)

Les jetons ALL-IN et RESET

Le menu "A propos"

**G. Les améliorations possibles**

En outre, d'autres ajouts sont possibles pour enrichir la solution:

Ajout d'options dans le menu option (Jetons de départ, sélection d'un thème graphique, choisir son pseudo.. etc.)

Sauvegarde du jeu lorsque le joueur quitte la partie. Et reprise de la partie lorsque l'on clique sur le bouton « reprendre ». Ceci peut-être fait avec la lecture / écriture de fichier. Quand on clique sur ce bouton, on regarde dans un fichier s'il y a des données sauvegardées. Si oui on les charges. Quand on quitte on met les données dans ce fichier.

**Partie III - Le site web**

**A. Présentation**

Le site, codé en xHTML, CSS et JavaScript, offre une interface graphique, simple d'utilisation, qui permet d'accéder à tous les éléments du projet. Ainsi grâce à cette plateforme, vous pourrez télécharger tous les éléments du projet. Ceci comprend entre autres, l'exécutable du jeu, le cahier des charges, ce rapport, les différents algorithmes utilisés, le document de mise en ligne ... etc.

Le site permet donc la promotion du projet.

**B. Technologies utilisées, et aspect technique**

Le site utilise du xHTML, du CSS (pour l'aspect graphique) et du JavaScript (pour l'aspect dynamique). Nous avons utilisé l'éditeur de texte Geany et Notepad++.

Le JavaScript est utilisé dans la page "Cahier des Charges" pour afficher des images lorsque l'utilisateur passe sa souris sur le nom d'un logiciel.

Nous l'avons aussi utilisé dans la page "Source" pour sélectionner le type de fichier à afficher.

**Partie IV - Conclusion:**

**A. Objectif atteint ?**

Oui les objectifs ont été atteints. Le jeu respecte le cahier des charge fournit dans le PDF, et propose même des options supplémentaires (le menu principal, les statistiques ...etc. CF: Amélioration). Aussi après plusieurs tests par plusieurs personnes sur différents pc, nous pouvons dire que le jeu est stable et fonctionnel.

Concernant le répertoire à rendre, celui-ci contient tous les objets demandés.

**B. Coût de la solution**

La durée de réalisation du projet est d'environ 250 heures. De plus en tenant compte que cela est notre premier projet, nous pouvons déterminer un prix à l'heure d’environs 10 €. Soit un prix égal à 2500 € pour la solution complète.

**C. Ce que nous a apporté ce projet ?**

Ce projet nous a appris plusieurs choses. Il nous a appris à travailler ensemble, à se partager les différentes tâches et ensuite à mettre en commun le travail de chacun.

Ensuite il nous a permit d'approfondir ou même d'apprendre certains langages. Entre autre l'approfondissement du langage C et l'apprentissage du JavaScript (qui n'a été qu'introduit dans le cours).

En outre nous avons aussi apprit à utiliser de nouveaux logiciels. Par exemple Visual Studio 2008 pour la partie codage du jeu en C, mais aussi Photoshop comme éditeur graphique.

Enfin nous avons appris à prendre de bonnes décisions qui ont parfois été capital dans l'accomplissement à bien de ce projet. Mais aussi à être soumit à des contraintes de temps et techniques.

En conclusion ce projet à été riche en apprentissage et nous a procuré une expérience non négligeable pour notre futur métier, Ingénieur en Informatique.