

Pyramid



Ferreira Dantas Filipe
Alipio Penedo João Paulo

Table des matières

| | | |
|-----|--|----|
| 1 | Introduction..... | 3 |
| 1.1 | Cadre, description et motivation | 3 |
| 1.2 | Organisation | 3 |
| 1.3 | Objectifs..... | 3 |
| 1.4 | Planification | 4 |
| 2 | Analyse..... | 5 |
| 2.1 | Description de l'application | 5 |
| 2.2 | Analyse concurrentielle..... | 6 |
| 2.3 | Résumé de l'analyse concurrentiel..... | 9 |
| 2.4 | Maquette graphique..... | 10 |
| 2.5 | Use cases et scénarios..... | 11 |
| 3 | Implémentation | 12 |
| 3.1 | Dossier de conception | 12 |
| 3.2 | Fonctionnalités | 12 |
| 3.3 | Description des tests effectués..... | 15 |
| 3.4 | Erreurs restantes | 15 |
| 4 | Mise en service..... | 16 |
| 4.1 | Installation | 16 |
| 4.2 | Liste des documents fournis | 16 |
| 5 | Conclusions | 16 |
| 6 | Annexes..... | 17 |
| 6.1 | Sources | 17 |
| 6.2 | Archives du projet..... | 17 |

1 Introduction

1.1 Cadre, description et motivation

Ce projet est réalisé dans le cadre du module Projet C# au cours de la 3^{ème} année d'apprentissage d'informaticien au CPNV. Nous avons choisi de réaliser pendant ce projet un jeu de cartes appelé Pyramid.

1.2 Organisation

Élèves :

- Ferreira Dantas Filipe
- Filipe.FERREIRA-DANTAS@cpnv.ch
- Alipio Penedo Joao Paulo
- Joao-Paulo.ALIPIO-PENEDO@cpnv.ch

Responsables de projet :

- Ithurbide Julien
- Julien.ITHURBIDE@cpnv.ch
- Andolfatto Frederique
- Frederique.ANDOLFATTO@cpnv.ch

1.3 Objectifs

Définition de l'application :

- Définir les tenants et aboutissants de l'application qu'on développe
- Planification (brainstorming)

Analyse et conception de l'application :

- Phase d'analyse et conception (objet d'un rapport après 3 semaines)

Implémentation de l'application :

- Rédaction du rapport de mise en service

1.4 Planification

| | | |
|---|---|-----|
| Sprint 4 🕒 Updated 5 minutes ago | Correction de bugs + commentaires dans le code Finalisation de l'application + tests Finalisation de la documentation Préparation de la présentation finale Échéance : 25/01/2019 | *** |
| Sprint 3 🕒 Updated 7 days ago | Continuation de la documentation Création des fonctionnalités : actions joueur + tests Ajout des scores dans le fichier des scores + tests Échéance : 11/01/2019 | *** |
| Sprint 2 🕒 Updated on 14 Dec 2018 | Conception de l'interface graphique Implémentation de l'affichage des cartes + test Continuation de la documentation Échéance : 14/12/2018 | *** |
| Sprint 1 🕒 Updated on 27 Nov 2018 | Analyse - conception globale de l'application Création des uses cases Modélisation de la base de données Début de la documentation Échéance : 30/11/2018 | *** |
| Planning général 🕒 Updated on 23 Nov 2018 | Sprint 1 : Analyse - conception globale de l'application Sprint 2 : Conception et implémentation Sprint 3 : Conception et implémentation Sprint 4 : Finalisation de la conception et de l'implémentation et résolution de bug Échéance : 25/01/2019 | *** |

2 Analyse

2.1 Description de l'application



Ce dernier est composé d'un plateau de cartes disposées en forme de pyramide sur 7 lignes. La 1^{ère} ligne de 7 cartes est la seule accessible en début de partie, jusqu'à la 4^{ème} ligne composée d'une seule carte. Les cartes de la ligne qui se situe derrière celle avec laquelle on interagit deviennent accessibles une fois que la/les cartes situées devant sont éliminées.

Le joueur a accès à une pile de cartes, il peut voir 2 cartes de sa pile au même temps.

Pour éliminer des cartes, le joueur doit sélectionner une carte sa pile et une carte de la pyramide pour former un résultat égal à 13 en additionnant la valeur des deux cartes.

Le joueur peut également éliminer deux cartes de la pyramide.

Le joueur ne peut pas reculer dans sa pile de cartes, il a accès à sa pile 3 fois par pyramide. Si le joueur n'a pas terminé la pyramide au bout du 3^{ème} tour de sa pile, il peut jouer une nouvelle pyramide.

Le joueur a droit à 2 nouvelles pyramides en cas de de pyramide non terminée. Les pyramides réussies ne sont pas décomptées dans le compte des pyramides. Les pyramides réussies comptent comme des panneaux réussis.

Chaque élimination de deux cartes additionne 5 points au score du joueur. À la fin de la partie le score de points et de panneaux réussis est enregistré. Chaque ligne de cartes de la pyramide que le joueur élimine valent aussi des points :

- 1^{er} ligne : 25
- 2^{ème} ligne : 50
- 3^{ème} ligne : 75

- 4^{ème} ligne : 100
- 5^{ème} ligne : 150
- 6^{ème} ligne : 250
- 7^{ème} ligne : 500

Le joueur ne peut pas éliminer des cartes en utilisant plus que deux cartes.

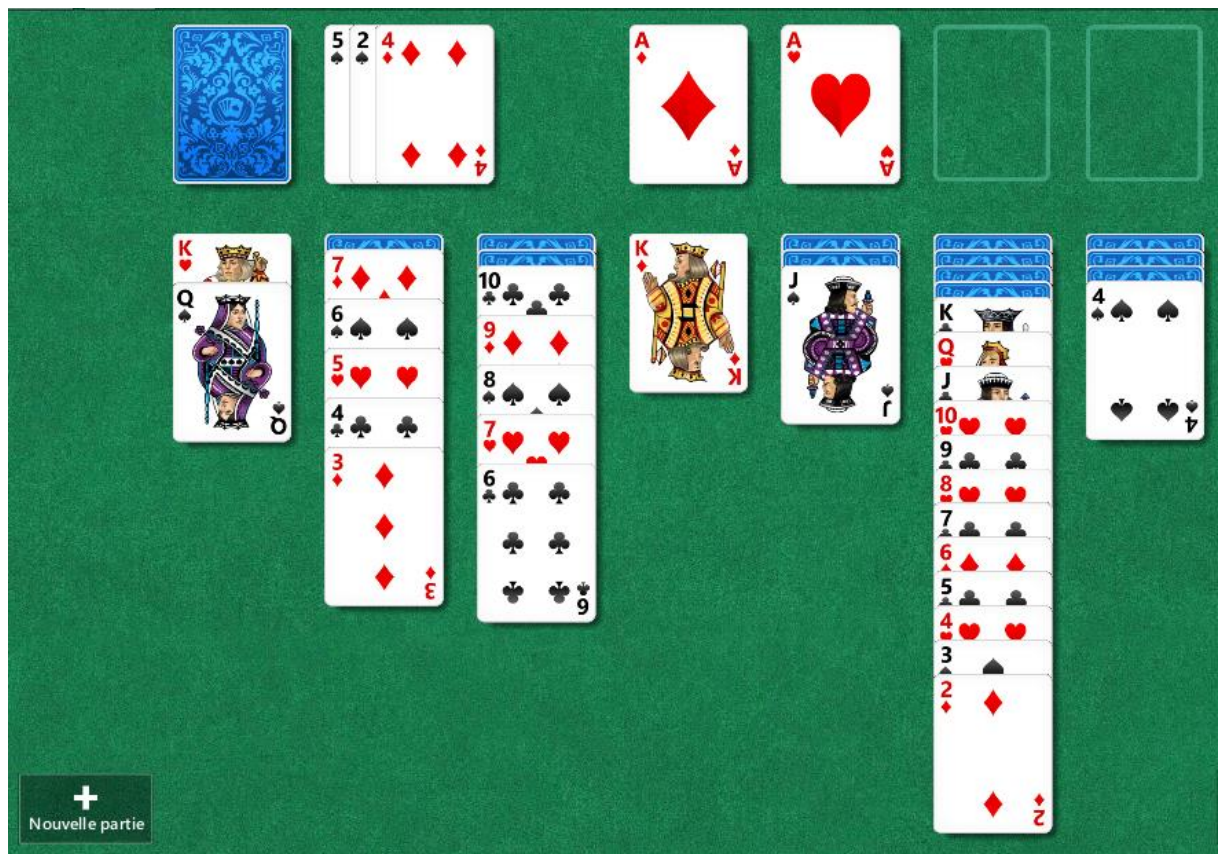
Exemple :

- 7 + 6 = 13 : la carte est éliminée
- 7 + 3 + 3 = 13 : la carte n'est pas éliminée
- Roi = 13 : la carte est éliminée

Chaque carte à la valeur de son chiffre et les cartes d'images (Roi, Reine, Valet et As) valent 13, 12, 11 et 1.

2.2 Analyse concurrentielle

- Klondike : le but de Klondike est de créer 4 piles croissantes de cartes sur chaque pile de base au coin supérieur droit. Chaque pile ne peut contenir une couleur.



Avantages :

- Tout comme Pyramide ce jeu est proposé de base sur les machines Windows 8 et ultérieur.

Inconvénients :

- Ce jeu est notamment plus compliqué que Pyramid, il est destiné à un public plus âgé.

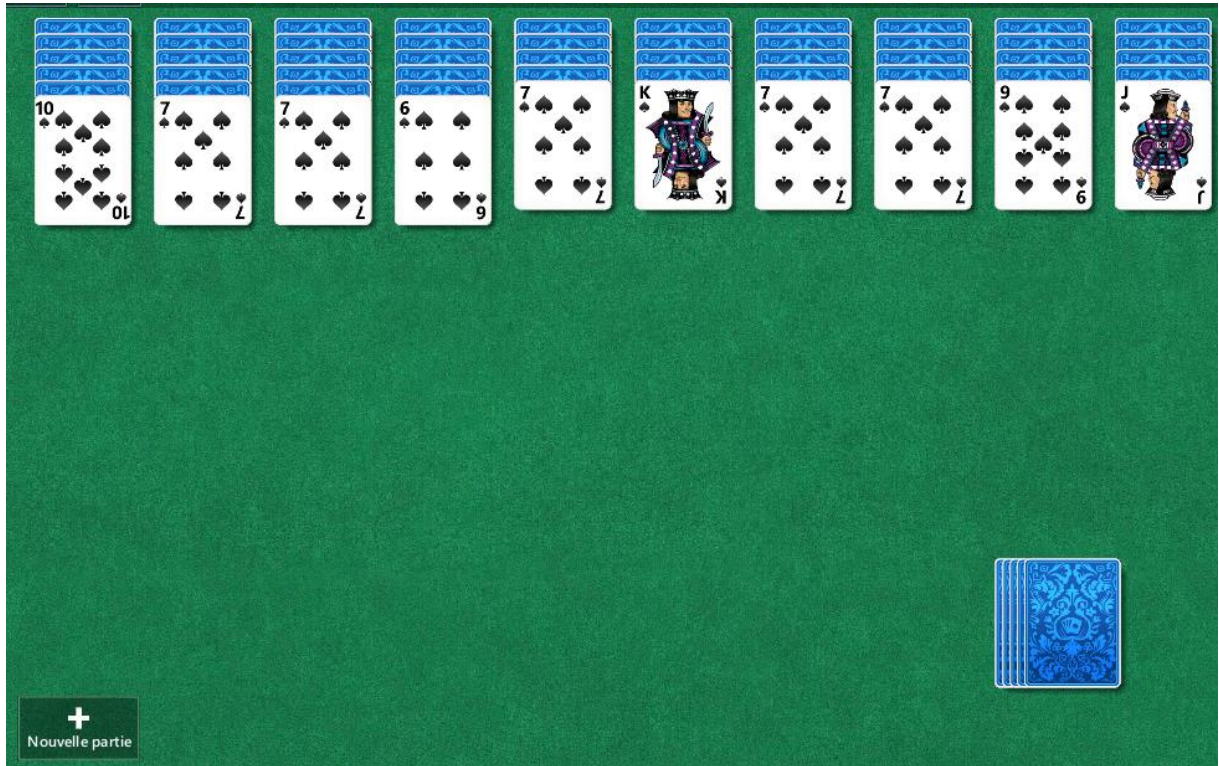
- FreeCell : le but de ce jeu est de créer quatre piles de cartes de base, une par couleur, au coin supérieur droit. Chaque pile doit être construite en ordre croissant.



Avantages :

- Tout comme Pyramide ce jeu est proposé de base sur les machines Windows 8 et ultérieur.

- Spider : le but du jeu est d'enlever les cartes de la table en créant des suites. Chaque suite doit être en ordre décroissant, du Roi à l'As.



Avantages :

- Tout comme Pyramide ce jeu est proposé de base sur les machines Windows 8 et ultérieur.
- Facile à apprendre et à jouer

- TriPeaks : le but est de dégager le plus de tableaux possibles. Supprimer les cartes dont on voit le recto en appuyant sur celles directement supérieures ou inférieures à la carte du dessus de la défausse, laquelle se trouve au bas de l'écran.



Avantages :

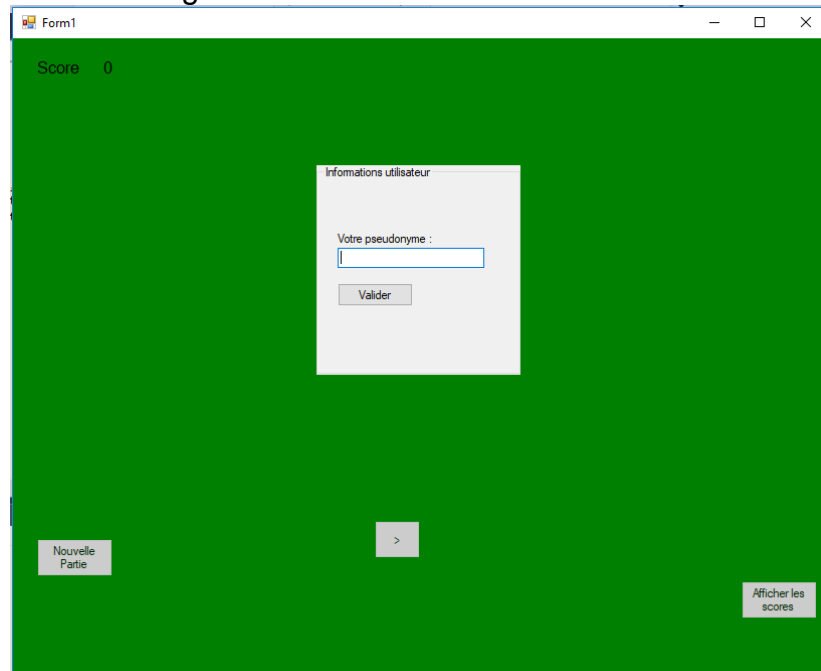
- Tout comme Pyramide ce jeu est proposé de base sur les machines Windows 8 et ultérieur.

2.3 Résumé de l'analyse concurrentiel

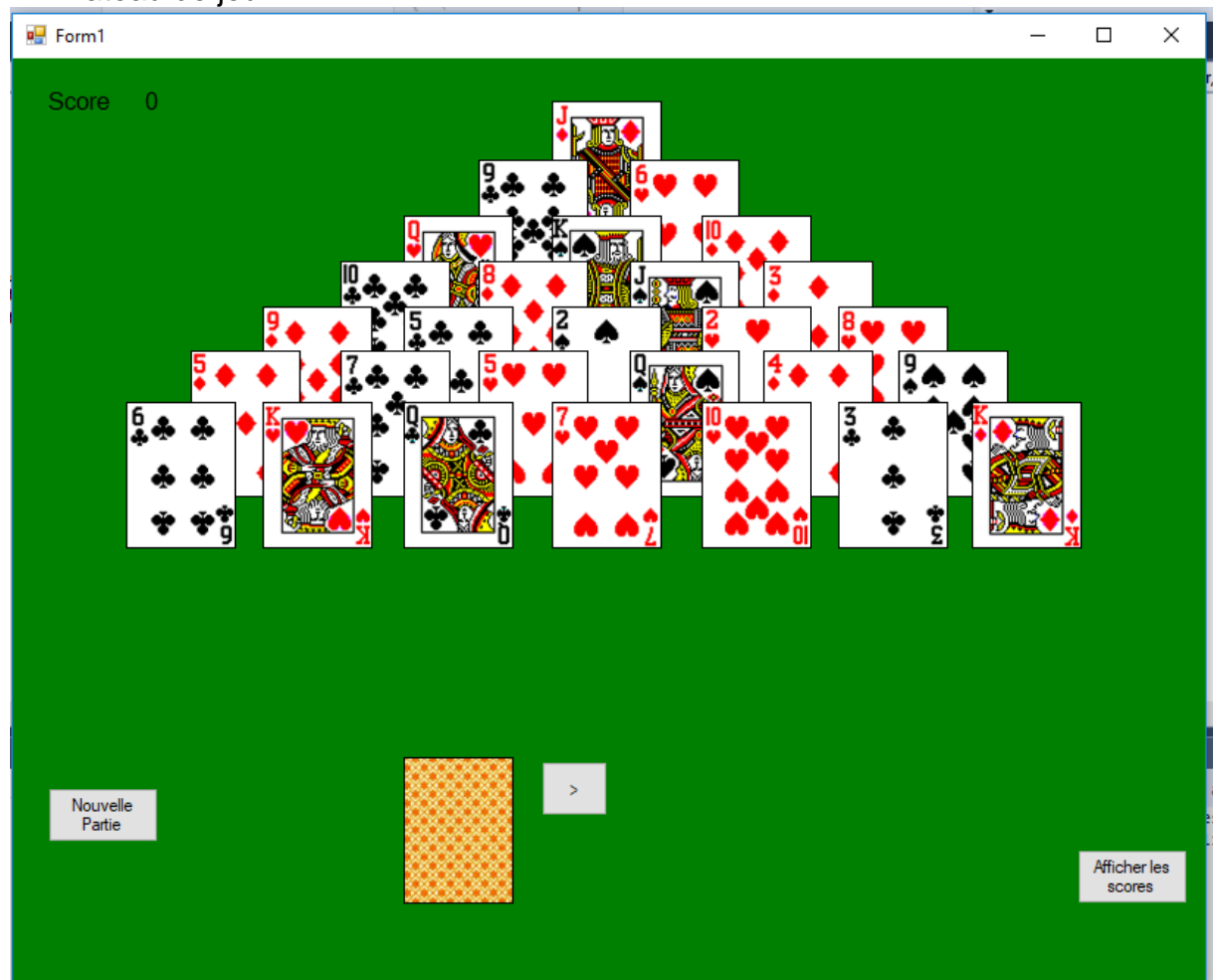
Tous les jeux abordés dans ce chapitre sont proposés dans les machines Windows 8 et ultérieur. Ils sont aussi disponibles sur internet. Tous les jeux sont assez faciles à prendre en main apart Klondike et FreeCells. Ils sont adaptés pour tous les publiques. Les jeux préinstallés sur Windows ont une interface graphique plus ergonomique et agréable grâce à des animations qui ne seront pas implémentés dans l'application C# que nous allons développer.

2.4 Maquette graphique

- Démarrage



- Plateau de jeu



2.5 Use cases et scénarios

| Use case 1 : Lancement de l'application | |
|---|--|
| Action | Résultat |
| L'utilisateur lance l'application | La fenêtre de l'application s'affiche, un pop-up demande le pseudo du joueur, pour pouvoir enregistrer son score à la fin de son jeu. le premier plateau de cartes s'affiche de manière aléatoire ainsi que la pile de cartes de l'utilisateur de manière |
| L'utilisateur clique sur « Valider » sur le pop-up de demande du pseudo | La première pyramide de cartes s'affiche, les cartes sont affichées de manière aléatoire, la pile de cartes du joueur est aussi mélangée. Les cartes de la pyramide ne sont pas doublées dans la pile de cartes du joueur. |
| L'utilisateur clique sur le bouton « Nouvelle partie » | Une nouvelle pyramide de jeu s'affiche ainsi que la pile de cartes du joueur. |
| L'utilisateur clique sur le bouton « > » | La prochaine carte de la pile de l'utilisateur s'affiche |

| Use case 2 : Jeu | |
|--|---|
| Action | Résultat |
| L'utilisateur sélectionne deux cartes dont la somme est égal à 13 | Les deux cartes que l'utilisateur a sélectionnées disparaissent du plateau. Le score de l'utilisateur augmente de 5 points. |
| L'utilisateur sélectionne deux cartes dont la somme n'est pas égal à 13 | Rien ne se passe. |
| L'utilisateur clique sur le bouton « Nouveau plateau » | Un nouveau plateau apparait de manière aléatoire. |
| L'utilisateur élimine toutes les cartes de la première ligne de son plateau | 25 points sont ajoutés au score de l'utilisateur |
| L'utilisateur élimine toutes les cartes de la deuxième ligne de son plateau | 50 points sont ajoutés au score de l'utilisateur |
| L'utilisateur élimine toutes les cartes de la troisième ligne de son plateau | 75 points sont ajoutés au score de l'utilisateur |
| L'utilisateur élimine toutes les cartes de la quatrième ligne de son plateau | 100 points sont ajoutés au score de l'utilisateur |
| L'utilisateur élimine toutes les cartes de la cinquième ligne de son plateau | 150 points sont ajoutés au score de l'utilisateur |
| L'utilisateur élimine toutes les cartes de la sixième ligne de son plateau | 250 points sont ajoutés au score de l'utilisateur |
| L'utilisateur élimine la carte de la septième ligne de son plateau | 500 points sont ajoutés au score de l'utilisateur |
| L'utilisateur fini un plateau | Une nouvelle pyramide s'affiche ainsi qu'une nouvelle pile de cartes, sans décompter le nombre de piles du joueur |

| | |
|--|----------------------------------|
| | en cas de pyramide non terminée. |
|--|----------------------------------|

| Use case 3 : Fin du jeu | |
|--|---|
| Action | Résultat |
| L'utilisateur n'a plus de solution possible pour continuer d'éliminer ses cartes | Le score de l'utilisateur est enregistré dans le fichier de scores. |

3 Implémentation

3.1 Dossier de conception

Durant ce projet, nous travaillons sur les machines du CPNV qui ont la configuration suivante :

- Dell OptiPlex 7050 :
 - o Processeur : Intel Core i7-6700 3.4 GHz
 - o RAM : 16384 MB
 - o Graphique : Intel HD Graphics 530
 - o OS : Windows 10 Education 64bits

Pour notre projet, nous utilisons les outils suivants :

- Programmation : Visual Studio 2017
- Documentation : Word 2016
- Logiciel de gestion de versions décentralisés : Git

3.2 Fonctionnalités

Fonctions dans le fichier « Form1.cs » ;

MyPaint :

- Cette fonction permet de « dessiner » les cartes dans le fenêtré de l'application, cette fonction fait aussi appel à la fonction « Shuffle » qui permet de mélanger les cartes.

Form1Load :

- Cette fonctionnalité affiche la message box qui demande un pseudo à l'utilisateur.

cmdAfficherScores_Click :

- Dans cette fonction on trouve le code du bouton qui permet d'afficher les scores des utilisateurs qui sont inscrits dans le fichiers des scores (Scores.txt).

cmdCacherScores_Click :

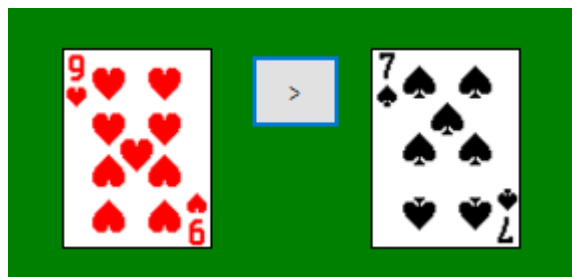
- Dans cette fonction on trouve le code du bouton qui permet de « cacher », n'est plus afficher, les scores des utilisateurs.

Shuffle :

- Cette fonction permet de mélanger une liste d'objets, nous l'utilisons pour mélanger les cartes de la pyramide avant de les afficher.

cmdNextCard_Click :

- Dans cette fonction on trouve le code du bouton « Prochaine carte », lors du clic sur ce bouton, l'utilisateur peut voir la prochaine carte de sa pile de cartes.



imgLastCarte_Click :

- Cette fonction contient le code du bouton « Dernière Carte », c'est la carte de droite dans l'image ci-dessus (7 de piques). La fonction permet d'éliminer cette carte si c'est un Roi (valeur du roi = 13) ou en cliquant sur un 6 ($6 + 7 = 13$).

imgNouvelleCarte_Click :

- Cette fonction a la même fonctionnalité que la fonction `imgLastCarte_Click`, mais cette fois-ci, c'est lors du clic de la carte de gauche (9 de cœur).

cmdNouvellePartie_Click :

- Dans cette fonction on trouve le code qui va permettre à l'utilisateur de commencer une nouvelle partie au clic du bouton « Nouvelle Partie ».

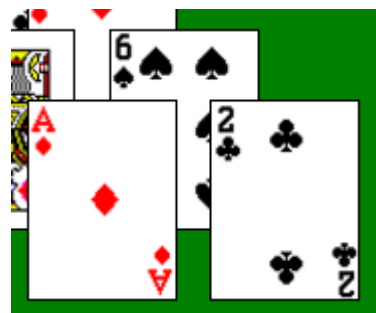


ClickOnCard :

- Cette fonction contient le code qui permet de gérer le clic avec toutes les cartes de la pyramide, cette fonction est appelée à chaque clique sur une des cartes de la pyramide. Cette fonction fait aussi appel à la fonction qui permet de calculer le score de l'utilisateur.

CanClick :

- Cette fonction contient le code qui vérifie si l'utilisateur peut ou ne peut pas cliquer sur n'importe quelle carte de la pyramide.
- Exemple (image à côté) : dans ce cas l'utilisateur peut cliquer sur le As de carreau et le Deux de trèfles, au contraire si l'utilisateur essaye d'appuyer sur le Six de piques et ensuite sur un sept il ne pourra pas éliminer aucune des deux cartes qu'il vient de sélectionner en cliquant.



ScorePoints :

- Cette fonction permet d'ajouter des points au score de l'utilisateur lorsqu'il va éliminer toutes une ligne de cartes de la pyramide.

Fonctions dans le fichier « Carte.cs »

Ce fichier contient le code de la classe des cartes, il contient les fonctions suivantes :

Carte :

- Carte est le constructor de la classe Carte

GetValeurCarte :

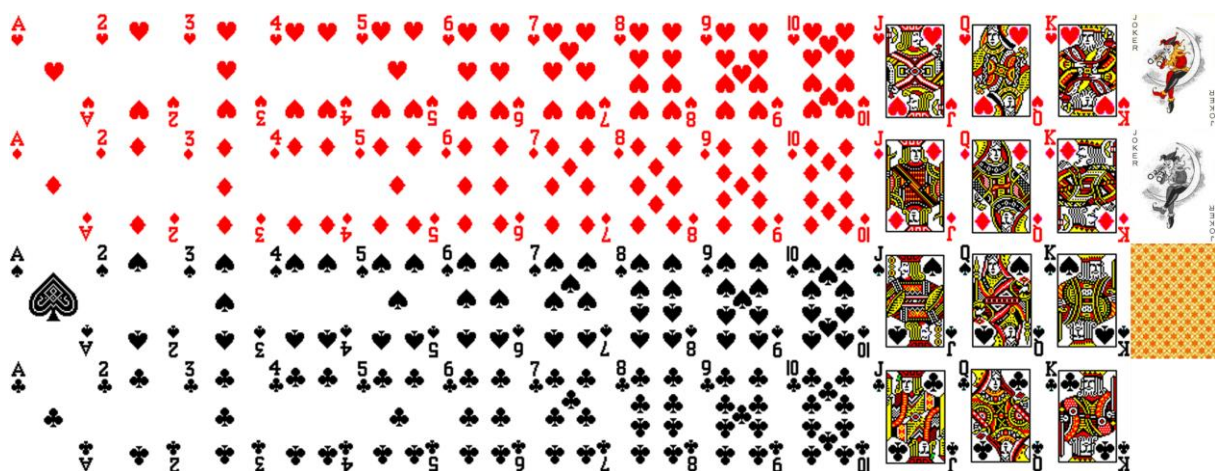
- Cette fonction permet de retourner la valeur de la carte sélectionnée.

GetImage :

- Cette fonction permet de retourner l'image de la carte sélectionnée.

Fonctions dans le fichiers « CarteGenerator.cs »

Dans ce fichier on trouve les fonctions qui nous permettent de dessiner les images des cartes, en les découpant sur une seule image qui contient toutes les cartes, « Debugdeck.png ». Ce fichier se trouve dans le dossier de la conception du code « Pyramid/Test/bin/ ».



ImagePath :

- Cette fonction permet de récupérer le chemin de l'image « Debugdeck.png ».

enum SorteCarte :

- Cette énumération contient les quatre sortes d'un tas de cartes, l'ordre de cette énumération est important.

enum ValeurCarte :

- Cette énumération contient toutes les valeurs d'un tas de cartes, l'ordre est aussi important.

getCartelImage :

- Cette fonction permet de récupérer de découper les cartes dans le fichier de l'image des cartes, avec la taille indiquée dans les paramètres de celle-ci.

getCarte :

- Cette fonction permet de récupérer la carte que l'on veut en lui donnant les paramètres de la sorte de la carte et sa valeur. En utilisant la fonction getCarteImage.

getJoker :

- Cette fonction permet de découper et retourner l'image la carte Joker dans le fichier de l'image des cartes.

getDos :

- Cette fonction permet de découper et retourner l'image du dos des cartes qui se trouve aussi dans le fichier de l'image des cartes

getToutesCartes :

- Cette fonction retourne toutes les cartes dans une liste de cartes, en utilisant les fonctions décrites antérieurement. La fonction crée aussi les objets Carte.

3.3 Description des tests effectués

Tous les tests en rapport avec notre application se font durant le développement. Notre temps limité ne nous permet pas de prévoir une séance de test à part.


3.4 Erreurs restantes

L'utilisateur ne peut pas utiliser les deux dernières cartes de sa pile, quand il essaye de le faire l'application crash et l'erreur suivante s'affiche :

```
private void imgLastCarte_Click(object sender, EventArgs e)
{
    score = Int32.Parse(lblScore.Text);
    if (i < Cartes.Count())
    {
        value1 = Cartes.ElementAt(i - 2).GetCarteValeur();
    }

    //MessageBox.Show(value1.ToString());

    if (value1 == 13)
    {
        imgLastCarte.Image = UsedCarte.Last();
        Cartes.RemoveAt(i - 2);
        score += 5;
        lblScore.Text = score.ToString();
        value1 = 0;
    }
}
```



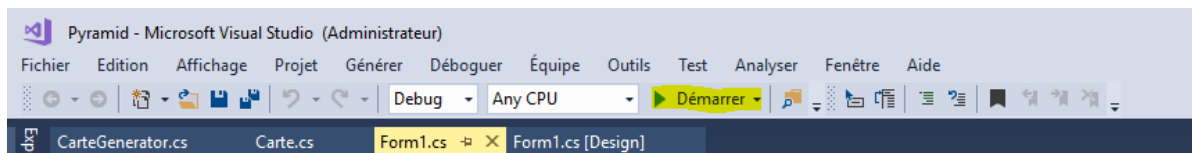
Par manque de temps nous n'avons pas réussi à corriger cette erreur.

Lorsque l'utilisateur joue, il se peut que quelques fois il ne puisse pas éliminer deux cartes dont la somme est égale à 13. Nous n'avons pas réussi à trouver la provenance de cette erreur.

4 Mise en service

4.1 Installation

Un dossier « Pyramid » vous sera fourni. Pour exécuter « Pyramid.sln » qui se trouve dans le dossier mentionné précédemment et qui vous permettra de lancer notre application, il vous faudra installer Visual Studio 2017. La dernière étape sera de lancer l'exécution de l'application en appuyant sur « Démarrer » dans Visual Studio suite au chargement du projet



4.2 Liste des documents fournis

Un dépôt Git est accessible. Ce dernier contient :

- Un dossier « Documentation »
 - o Contenant la documentation du projet
- Un dossier « Pyramid »
 - o Contenant les fichiers qui composent l'application « Pyramid »

5 Conclusions

Objectifs atteints :

- La majorité des objectifs décrits dans le planning ont été atteints. Les objectifs non-atteints sont décrits ci-dessous.

Objectifs non-atteints :

Comparaison (prévu et réellement passé) :

- Il n'y a pas eu de complication au niveau du planning,
- Nous avons essayé de respecter les sprints au maximum, nous avons toujours respectés les délais.
- Nous avons bien estimé nous sprints et nous sommes arrivés à nous y tenir.
- Nous n'avons pas manqué de temps pour finir le projet.

Points-positifs :

- Nous avons perfectionné nous compétences en C#.

Difficultés particulières :

- Nous avons été bloqués pendant le sprint 1 parce que nous ne connaissions pas encore l'hérité avec le C#, nous avons réussi à trouver une solution au problème qui est bien plus compliqué que simplement faire de l'hérité. Par manque de temps nous avons gardé notre solution de départ, ce qui nous l'affichage des cartes nettement plus compliquée.

Suites possibles :

- Les suites possibles à ce projet seraient de corriger les petits bugs restants, améliorer l'interface graphique et changer la méthode de création de l'objet carte en utilisant l'hérité avec les « PictureBox »

6 Annexes

6.1 Sources

Nous avons utilisé comme sources le site officiel de Microsoft, nous avons aussi parfois utilisé des forums sur internet et nous avons utilisé un document mit à disposition par le responsable du projet (ce document se trouve aussi dans repository).

Source : <https://docs.microsoft.com/fr-fr/>

6.2 Archives du projet

Un repository a été créé au début de ce projet sur GitHub. Tous les fichiers et documents ainsi que le programme du projet sont stockés dans ce repository.

Lien repository GitHub : <https://github.com/FerreiraDantas-AlipioPenedo/ProjetC--FerreiraDantas-AlipioPenedo>