CentraleSupélec

2e année d'école d'ingénieur - Projet long

Conception d'un logiciel de simulation de circuit électrique

Pierre BIRET et Victor FAUTH, promotion 2019

Année 2017-2018

Professeur encadrant : Amir ARZANDÉ

Table des matières

Introduction

Dans le cadre de notre projet long de la deuxième année du cursus ingénieur à Supélec, nous devons développer un logiciel de simulation de circuit électrique. Il s'agit d'un logiciel qui permet à l'utilisateur de créer graphiquement un circuit électrique (à l'aide de composants génériques ou personnalisés) puis de simuler son fonctionnement dans différentes conditions. Un logiciel commercial de ce type parmi les plus connus est LTspice.

Ce projet ayant déjà été proposé l'an dernier, il a été décidé que nous commencerions à partir de zéro afin de bien maîtriser chaque étape de la conception du logiciel. Cependant, nous conservons le langage de programmation précédemment utilisé. Le logiciel sera donc codé en C#. Il s'agit d'un langage orienté objet, très inspiré du C++, mais développé par Microsoft dans le cadre de la plateforme .NET et présentant de nombreuses différences. Pour gérer le front-end, nous utiliserons WPF (Windows Presentation Foundation).

D'un point de vue organisationnel, le projet se déroulera en plusieurs phases : tout d'abord une phase de montée en compétences en codant un petit jeu, puis l'implémentation d'une interface graphique basique pour le programme. Cela nous permettra ensuite d'ajouter la simulation du circuit. S'il nous reste du temps, nous pourrons ajouter des fonctionnalités au programme pour le rendre plus agréable, intuitif et rapide à utiliser.

I. Montée en compétences : programmation d'un programme simple

Afin d'apprendre à coder en C# avec WPF, nous avons décidé de commencer par un projet bien plus modeste : le développement d'un jeu de Memory. Il s'agit d'apprendre à coder en C# avec WPF (et donc à utiliser XAML, comme expliqué au prochain paragraphe), mais aussi de se familiariser avec l'IDE utilisé. Nous avons choisi pour cela Microsoft Visual Studio : le langage C# a été créé pour être utilisé avec cet IDE, il permet de générer le code XAML à partir d'une interface graphique, possède des outils de déboguage très puissants et une licence est fournie aux élèves de Supélec. Comme gestionnaire de versions, nous utilisons Git.

FIGURE 1.1 – Les logos de Visual Studio 2017 (gauche) et de Git (droite).

1.1 La dualité C#/XAML

Lorsqu'un programme graphique est créé en utilisant WPF, deux langages sont utilisés. Les éléments graphiques (fenêtre, boutons, images, etc...) sont codés en XAML. Il s'agit d'un langage descriptif dérivé du XML. Il nous permet de décrire les attributs de chaque élément : par exemple, la fenêtre a un nom (utilisé pour l'appeler dans le code), un titre, une taille, une position, etc... Elle possède aussi des « children » : par exemple, un bouton. Ce bouton possède des attributs similaires, mais aussi certains attributs spécifiques, tels l'animation à effectuer lors de l'activation. Si une fonction à lancer au clic (ou lors de n'importe quel événement ayant lieu sur l'élément) peut être définie pour la fenêtre, cet attribut est obligatoire pour un bouton.

Le XAML est un langage qui permet ainsi de définir rapidement une interface utilisateur, sans qu'il soit nécessaire de taper trop de code. Tous les éléments sont aussi personnalisables si nécessaire, bien que cela soit assez complexe.

Le code lui-même est écrit en C#. Il gère toute la logique et l'interactivité du programme. Si une interactivité est nécessaire, par exemple pour changer le titre de la fenêtre au clic, il est possible de modifier tous les éléments depuis le code, d'en créer ou d'en supprimer.

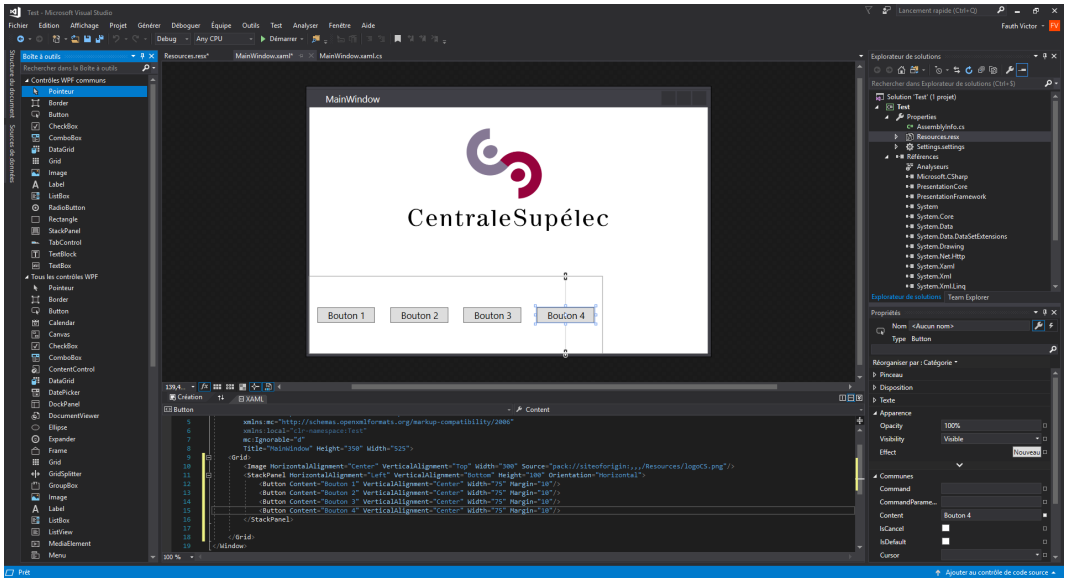

FIGURE 1.2 – L'éditeur graphique de XAML. Ici, une fenêtre avec une image et une ligne de boutons a été créée grâce à l'outil graphique, le code XAML correspondant est affiché dans la fenêtre inférieure. La fenêtre de droite permet de modifier les propriétés de l'élément sélectionné.

1.2 Règles du jeu

Il s'agit d'un jeu très simple dans lequel un certain nombre de cartes sont posées, face cachée. Ces cartes vont par paire : chaque motif est représenté sur deux cartes. Les cartes sont posées au hasard, puis le joueur retourne deux cartes. Si elles sont identiques, la paire est retirée du jeu. Sinon, elles sont retournées face cachée, au même endroit (après un temps, ici deux secondes, permettant au joueur de mémoriser les cartes). Le jeu s'arrête lorsque toutes les paires ont été trouvées, le but étant de minimiser le nombre d'essais.

1.3 Implémentation

1.3.1 Le modèle MVC

Pour coder le jeu, nous avons décidé d'utiliser l'architecture MVC (Modèle-Vue-Contrôleur). Le principe est de séparer le code en trois composantes distinctes : le modèle contient les données du programme, le contrôleur gère la logique et la vue interagit avec l'utilisateur.

1.3.2 Le modèle

Le modèle contient l'état du jeu a un moment donné. Pour cela, nous définissons deux classes : la classe `Card` et la classe `Board`.

1.3.3 Le contrôleur

1.3.4 La vue

L'interface graphique statique en XAML

L'interactivité en C#

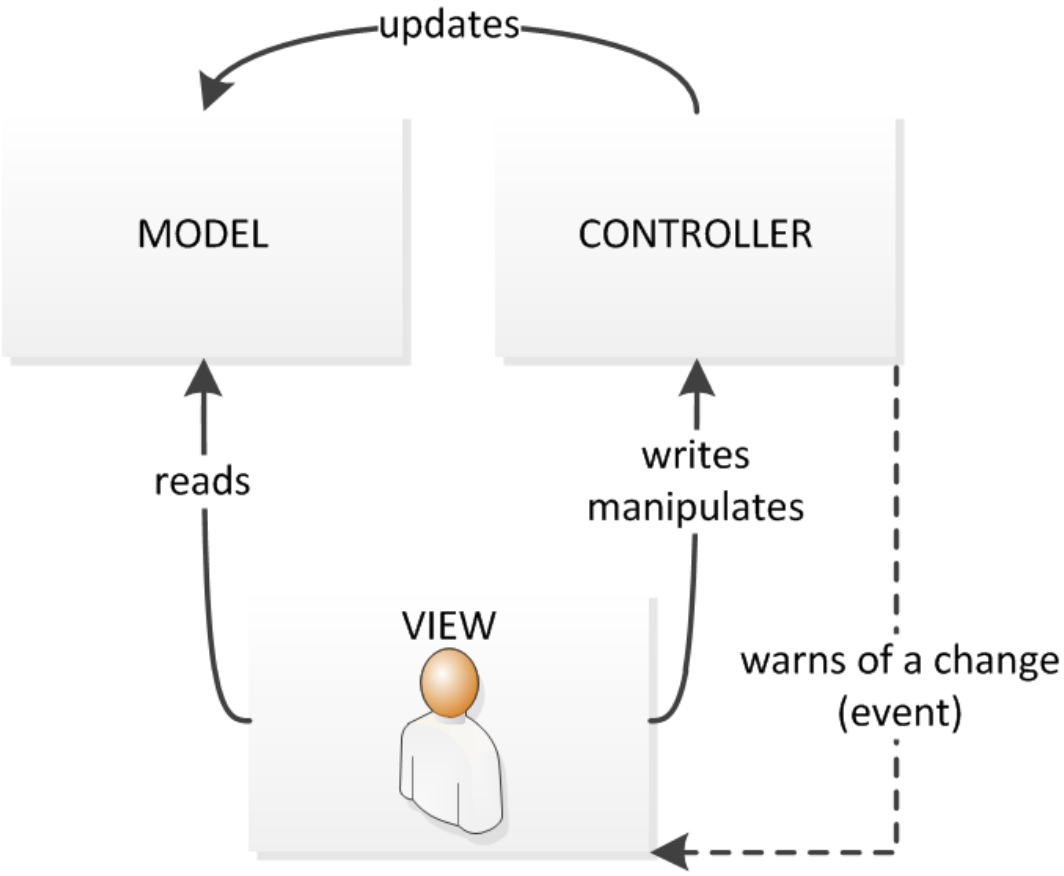

FIGURE 1.3 – Les interactions de l'architecture MVC. Image tirée de Wikipédia.

II. Cahier des charges

2.1 Test

2.1.1 Test2

III. Stage

Conclusion

Conclusion

Glossaire

EDIe L'environnement de développement intégré (IDE en anglais) est un logiciel permettant à la fois d'éditer du code et de le compiler. Il comprend souvent des outils simplifiant le développement, tel un débogueur.