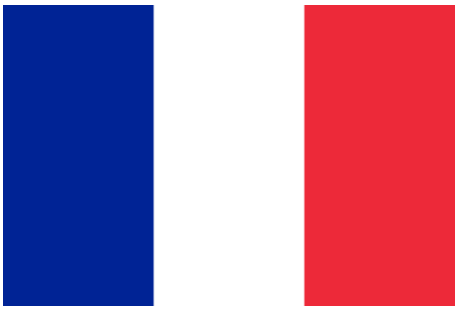
2017

Projet C++ : Elections piège à cons



Election TRUMP

POLYTECH PARIS-UPMC

MOMNOUGUI Johann

SAKHI Faïçal

UPMC SORBONNE UNIVERSITÉS

EISE4

13/01/2017

Sommaire

I. Introduction

L'objectif du projet est de répondre à la problématique imposée qui était « élections piège à cons » .Notre objectif était de construire autour de ce thème un projet C++ répondant à la problématique et respectant la grille d'évaluation imposée. Le but étant avant tout de montrer ce que nous savons faire en C++.

1. Thème

Le thème étant « élections piège à cons », nous devions trouver une manière de coder qui réponde à la problématique .Nous avons pris le sujet au premier degré en organisant une élection. Nous avons aussi pris l'initiative de jouer sur la dimension humoristique du sujet en permettant à l'utilisateur de jouer et de s'identifier à des personnages politiques connus.

2. Contraintes techniques

Nous devions durant ce projet respecter certaines contraintes que sont :

— 8 classes ;
— 3 niveaux de hiérarchie ;
— 2 fonctions virtuelles différentes et utilisées à bon escient ;
— 2 surcharges d'opérateurs ;
— 2 conteneurs différents de la STL ;
— diagrammme de classe UML complet ;
— commentaire du code ;
— pas d'erreurs avec Valgrind ;
— rendu par dépôt git, adresse à envoyer par mail avec dans le sujet le motif `[EISE 4 C++ Projet]` ;
— pas de méthodes/fonctions de plus de 30 lignes (hors commentaires, lignes vides et assert) ;
— utilisation d'un Makefile avec une règle "all" et une règle "clean".

II. L'utilisateur et le programme

1. Lancement du Programme

Le programme contient les fichiers suivants sans compter les images :

-Personnage.hh/Personnage.cc

-Electeur_non_politique.hh /Electeur_non_politique.cc

-Politique.hh/Politique.cc

-Evenement.hh /Evenement.cc

-Meeting.cc/Meeting.hh

-main.cc

-Basic_Window.hh/Basic_Window.cc

-Fenetre.hh/Fenetre.cc

-Fenetre_Choix.hh/Fenetre_choix.cc

-Fenetre_Reponse.hh/Fenetre_Reponse.cc

-Debat.cc/Debat.hh

-Election.cc/Election.hh

-Makefile

-Scenario.cc/Scenario.hh

-Fenetre_Debat.hh/Fenetre_Debat.cc

-Fenetre_Entry.hh/Fenetre_Entry.cc

- utility.cc/utility.hh

Pour pouvoir exécuter notre programme, les PC doivent disposer en plus de g++, de la librairie graphique ***Gtkmm***. C'est celle que nous avons choisie pour réaliser notre interface graphique.

Ce lien indique comment installer ***Gtkmm*** sur son PC et comment compiler un projet avec :

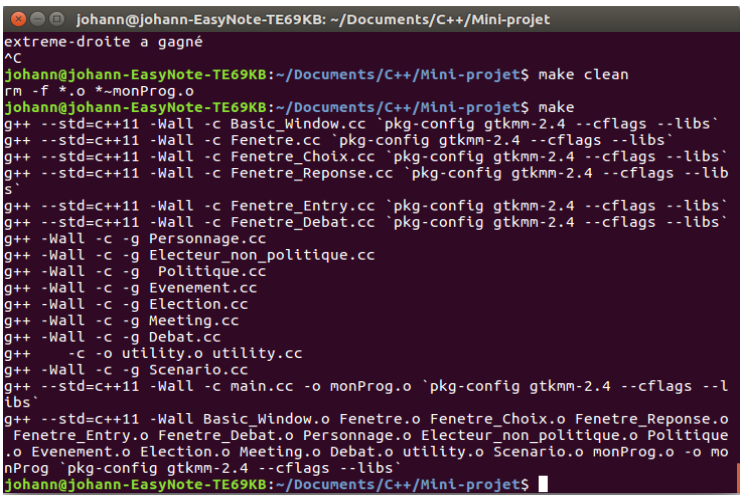
https://doc.ubuntu-fr.org/gtkmm

Pour récupérer notre programme, il suffira de télécharger le dossier présent sur **GitHub** à l'adresse suivante :

https://github.com/JohannNdecka/Mini-projet-MOMNOUGUI-SAKHI.git

Une fois le dossier téléchargé il faudra bien veiller à ce que tous les fichiers décrits précédemment soient présents dans ce même dossier.

On accède ensuite à notre dossier grâce au terminal .On compile le projet grâce au **Makefile** du projet dans le terminal. Ensuite, on lance l'éxécutable créé, ***monProg*** :



Enfin, on peut alors **jouer à notre jeu**.

2. Utilisation du programme

Afin de pouvoir jouer à notre jeu, il suffit de suivre les instructions et de se laisser guider par les fenêtres graphiques.

Une fois exécuté, le programme se lance et la première fenêtre apparait proposant de démarrer le jeu il suffit alors d'appuyer sur le bouton **Démarrer** .

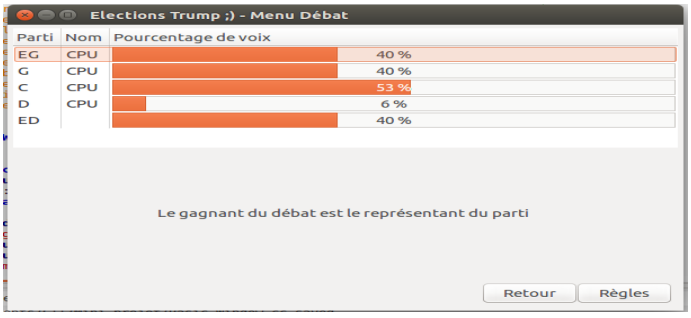
Après cela, le jeu va proposer au jouer de choisir son candidat .Il suffit alors **de cliquer sur le bouton du candidat de son choix .**

La troisième fenêtre demande au candidat d'entrer son nom de joueur .Il suffit alors d'entrer le nom puis ensuite d'appuyer sur **Continuer**.

La fenêtre suivante affiche le menu du jeu ; elle permet de choisir ce que l'on veut faire faire à son candidat. Le candidat peut alors choisir de faire un **meeting**, un **débat** ou alors de finir le jeu en lançant les **élections** qui départageront les candidats.

En fonction du choix, on accède alors à la fenêtre suivante si on fait un meeting, et si on veut revenir au menu joueur il suffit de sélectionner la touche **Continuer**.

On peut aussi accéder à la fenêtre suivante si l'on a sélectionné la touche **débat**, on sélectionne alors **Continuer** pour quitter la page et revenir **au menu principal**.



Enfin on accède à cette page si on a sélectionné démarrer élection une fenêtre transitoire apparait puis une fenêtre finale affiche le résultat des élections .Une touche **Rejouer** permet de revenir **au menu principal**.

III. Le programme

1. Principe du jeu et fonctionnement global

Le principe de base de notre application est de réaliser un jeu d'élections. Dans ce jeu, l'utilisateur incarne le représentant d'un parti qui doit se présenter aux élections. Il choisit son candidat

et c'est là que le jeu commence. Il a la possibilité d'organiser des meetings ou des débats afin de convaincre un peu plus la population puis il peut choisir de commencer les élections et le hasard décidera du sort.

Le côté amusant de ce jeu est qu'on ne sait pas qui peut gagner malgré le nombre de débats et de meetings organisés car les fonctions de vote utilisent rand(). Ce qui est intéressant dans notre jeu est que nous avons pu modéliser les votes par les préférences des citoyens. En effet, un citoyen sera partisan d'un certain camp lais cela ne garantit pas qu'il vote pour son parti. Je vais vous expliquer les rôles des meetings et des débats :

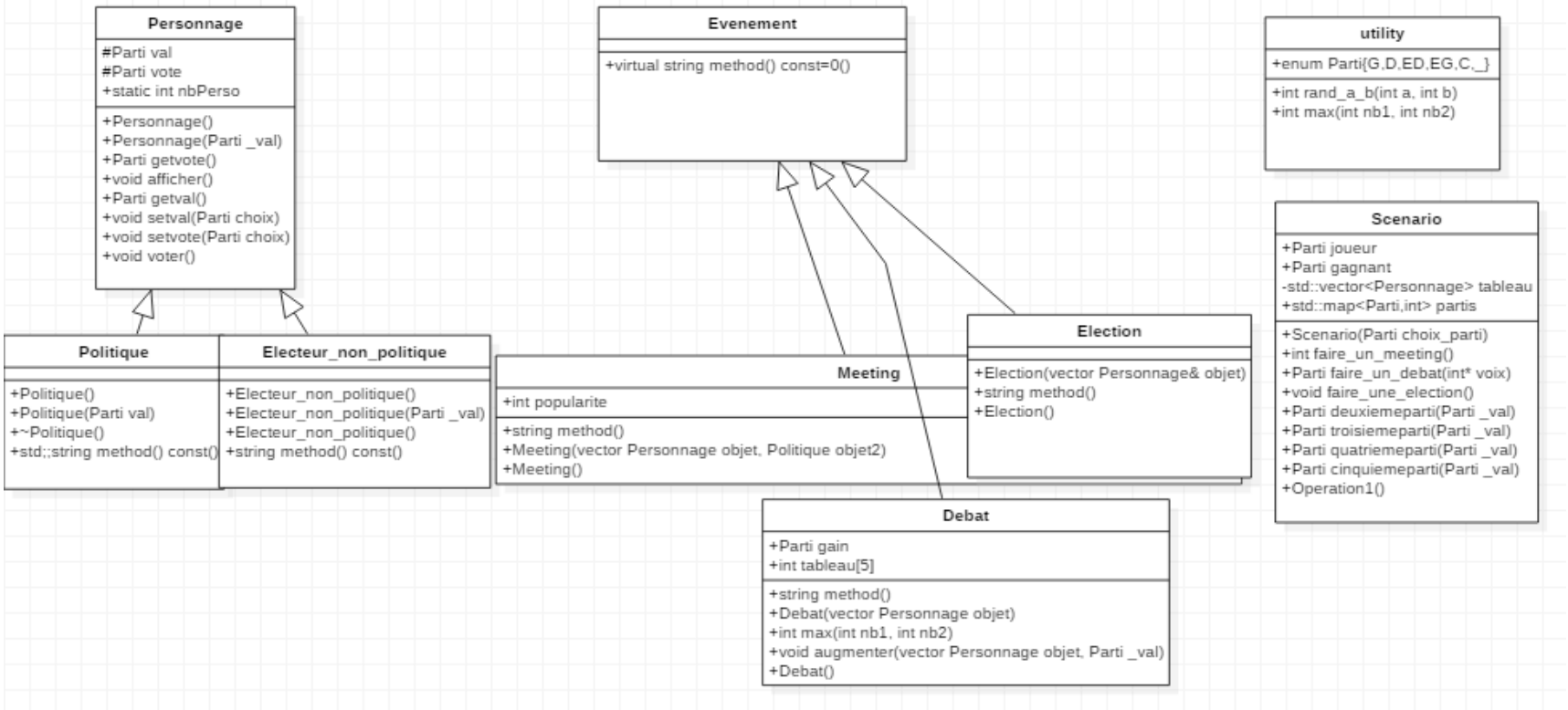
- Les meetings permettent d'avoir entre 0 et 10 partisans en plus ;
- Les débats permettent d'avoir 50 partisans en plus mais le gagnant du débat doit avoir le score maximum ; il s'agit d'un rand entre 0 et 100. C'est le gagnant qui récupère les partisans.

On remarque donc que dans chaque cas, il n'y a que la chance qui prédomine et l'on ne sait pas d'avance qui va gagner. C'est là tout le sens du jeu : c'est un piège à cons car que ce soit pour les partisans ou les candidats, nul ne peut être sûr d'assurer la victoire à son parti.

Pour présenter le jeu, nous allons présenter le code C++ contenant les classes politiques, les personnages que nous appellerons le moteur du jeu puis les fenêtres du jeu qui représentent l'interface graphique du jeu. Ces deux parties ont chacune leur diagramme UML mais dans l'application ; c'est l'interface graphique qui fait appel aux fonctions du moteur du jeu.

2. Le moteur du jeu

1.1) Le diagramme UML



1.2) Le jeu en lui-même

a) *L'héritage de la classe Personnage*

Dans ce jeu, la classe **Personnage** est une classe que l'on voulait abstraite (on a pas pu l'utiliser comme une classe abstraite à cause de son incompatibilité avec le vector) .Cette classe permet seulement de regrouper des attributs communs aux deux classes héritées que sont le parti auquel le personnage est rattaché de base et le parti pour lequel il va voter au fur et à mesure du jeu ainsi qu'une méthode commune qui est la méthode voter .On ne peut donc que créer un Personnage **Politique** qui lui aura le droit de voter, de faire des meetings, de faire des débats devra se présenter aux élections ou un personnage **Electeur_non_politique** qui lui pourra juste voter . Les classes Electeur_non_politique et Politique sont héritées de Personnage.

b) *L'héritage de la classe évènement*

La classe **Evènement** est, elle une véritable classe abstraite ; elle a comme classes filles les classes **Meeting** , **Debat** et **Election**. Ces classes interagissent elles avec des personnages ou des vector de personnages grâce aux arguments de leurs méthodes ou de leurs constructeurs.

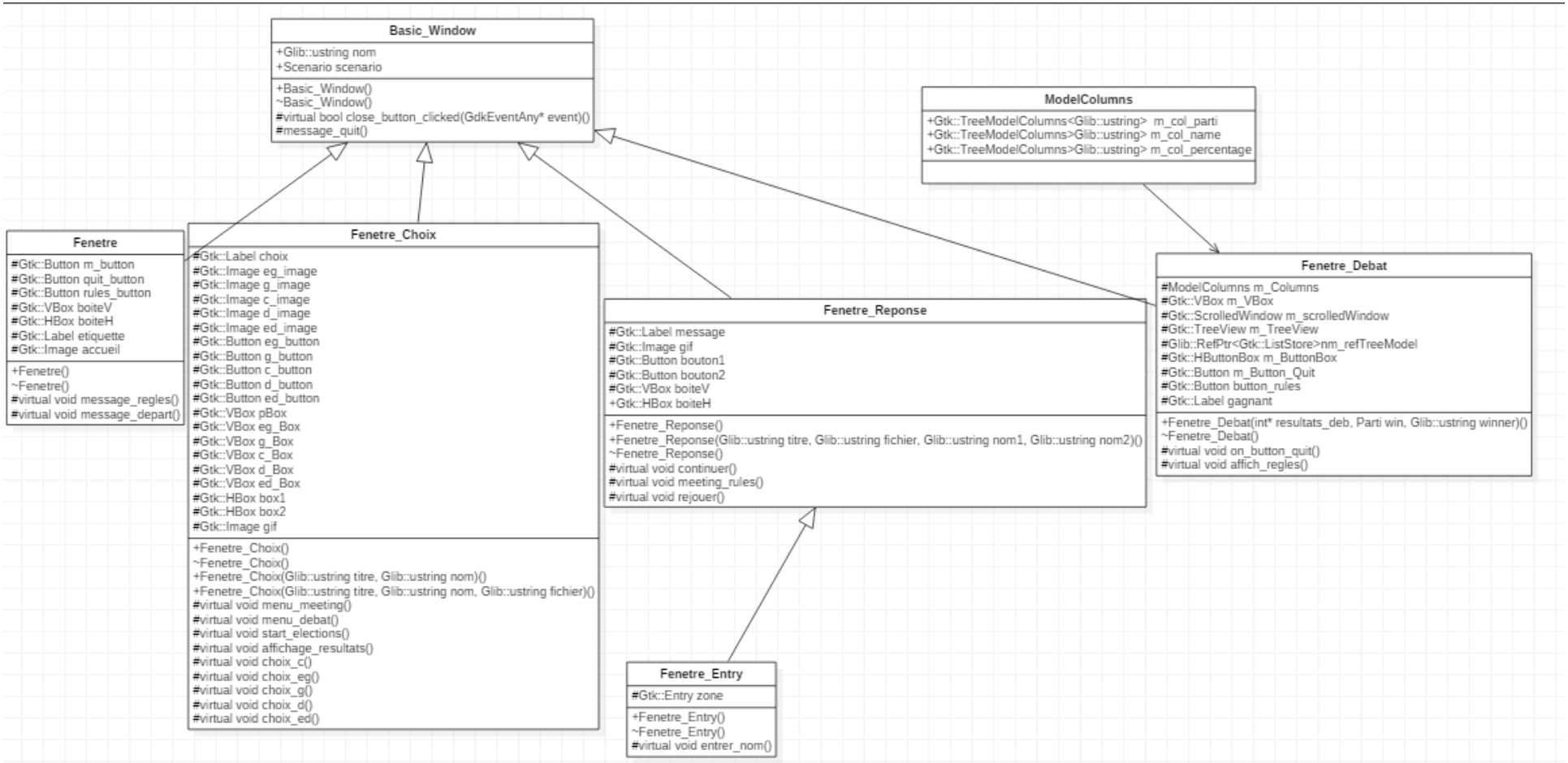
c) *La classe Scenario*

La classe **Scenario** permet de regrouper tous les éléments pour créer le moteur de jeu .On retrouve dans ses attributs un vector de personnages et ses méthodes ont pour arguments des objets issus de la hiérarchie des évènements.

La classe Scenario permet d'avoir un moteur de jeu plus fluide et plus facile à prendre en main ainsi qu'une interface de jeu plus simple.

3. L'interface graphique du jeu

1.1) Le diagramme UML



1.2) Présentation de l'interface graphique

Le moteur graphique permet de donner une dimension visuelle au programme. Grâce à lui, l'utilisateur visualise mieux ce qu'on attend de lui. Etant donné l'application que nous voulions réaliser, nous avons choisi 'utiliser ***Gtkmm***. Elle respecte la hiérarchie du C++ et donc utilise la Programmation Orientée Objet (P.O.O).

Nous allons ainsi présenter les aspects de cet interface.

a) L'héritage des objets

Nous avons choisi de faire hériter toutes les classes d'une classe principale appelée ***Basic_Window***. Ainsi, nous pouvons attribuer à toutes nos fenêtres des paramètres tels que la taille, le titre, l'icône, le message de confirmation… Cette technique a aussi été très utile car on peut lui attribuer un attribut qui sera commun à toutes les classes. C'est ainsi qu'on a inséré un attribut de type Scenario pour contrôler le jeu.

b) La modularité des objets

Un des avantages de la P.O.O. est la modularité des objets qui permet de créer des objets d'une même classe différents mais avec les mêmes caractéristiques fondamentales. On a ainsi pu créer des fenêtres utilisant les mêmes objets mais disposées différemment. L'exemple le plus frappant dans notre application est la classe **Fenetre_Choix**. Il a plusieurs attributs pour la construction de fenêtres donc il y a eu plusieurs constructeurs de cette classe qui pouvaient définir une fenêtre particulière avec des évènements différents. On peut aussi utiliser un constructeur avec paramètres et créer des objets différents comme ça a été le cas avec la classe **Fenetre_Reponse**. Tout ceci nous a permis de modéliser des fenêtres diverses mais aux caractéristiques similaires.

IV. Conclusion

Ce projet nous a permis d'approfondir et surtout d'utiliser nos connaissances en C++. Nous avons pu travailler en groupe et appliquer nos connaissances à un projet que nous avons monté nous-mêmes. De plus cela nous a vraiment permis de nous améliorer.

Ce projet nous a aussi permis de découvrir d'autres facettes du C++ comme la création d'interfaces graphiques par exemple. Nous nous sommes aussi aperçus un peu plus que le c++ demandait une grande rigueur notamment pour ce qui est de l'allocation dynamique par exemple comme avec l'utilisation des références ,des const, des pointeurs, etc… Cependant nous pensons que ce projet (et même ce module en C++) nous a vraiment été bénéfique et nous a permis de commencer à franchir un nouveau pallier en terme de programmation.