KIRADY Amélie
SHEHADEH Fadi

Projet C++
Le Monde d'Après

- **Install**

Nous avons créé ce projet en développant sur un éditeur de texte. La partie nouvelle dans notre méthode de travail a été d'utiliser un logiciel pour ce qui concerne la partie graphique. Le logiciel s'appelle Qt Creator. Ce logiciel est un IDE cross-platform puissant et gratuit.

Nous vous expliquons en quelques étapes l'installation du logiciel pour pouvoir l'utiliser:

- La première commande a effectuer est la suivante :

sudo apt-get install qt5-default
Sudo apt-get install qtcreator

- Ensuite, pour être sur d'avoir les différents fichiers nécessaires à la bonne compilation du programme il faut faire :

sudo apt-get install build-essential
sudo apt-get install qtmultimedia5-dev

- Pour lancer le logiciel il faudra ouvrir Qt creator, ouvrir un nouveau projet, et sélectionner le fichier *Projet.pro* dans le dossier fourni. Une erreur peut apparaître, indiquant un problème de compatibilité. Cliquer sur *ok* et *configurer* avec les paramètres déjà proposés.

- Pour effectuer la compilation, il faut d'abord aller dans l'onglet compiler en haut à gauche, et cliquer sur *Exécuter qmake.*

- L'étape d'après consiste à compiler tout le projet, en cliquant sur la flèche verte en bas à gauche de l'écran.

- Si tout se déroule bien, le jeu devrait se lancer.

● Description de l'application

- *L'histoire*

Le principe de notre application est de représenter "le monde d'après", dans lequel nous avons imaginé la présence du vaccin contre le Divoc-67, un virus apparu en 2067. Ce virus a affecté de nombreuses personnes sur la planète et le vaccin a été introduit dans la ville de Sorssieu pour la première fois. Au tout début nous voulions simplement faire une simulation sur la population de Sorssieu et voir le nombre de personnes contaminées en comparant avec la présence ou non du vaccin et de voir ses influences en général. Puis, nous avons souhaité faire participer l'utilisateur à cette simulation en lui faisant choisir certaines actions. En fonction de celles qu'il choisit, il aurait plus ou moins de chance de contracter le virus et le résultat serait annoncé à la fin ou au cours de sa partie s'il est finalement contaminé.

- *Les différents personnages*

Après son introduction dans la ville, la population est composée de plusieurs types de personnages : ceux qui sont vaccinés et ceux qui ne le sont pas. Les personnages qui ne sont pas vaccinées, n'ont pas le droit de se rendre au cinéma par rapport à celles qui le sont. Parmi les personnes vaccinées, certaines ont bien réagi au vaccin : ce sont les personnes "normales", qui sont immunisées. Cependant, d'autres personnes ont mal réagi à ce vaccin : certaines ont eu un troisième bras qui a poussé, les rendant plus contaminantes, puisqu'elles touchent plus de surface. D'autres ont perdu tous leurs poils et leurs cheveux, les rendant dénuées de barrières protectrices naturelles et ainsi, plus contaminables. En plus d'avoir eu un effet secondaire considérable sur ces personnes, le vaccin ne les a pas immunisées. Seulement, personne ne le sait : elles sont donc autorisées à se rendre dans tous les lieux et n'ont aucune restriction.

- *Les différents lieux*

Pour rendre plus réaliste cette simulation, nous avons souhaité créer différents lieux puisque les facteurs de contamination sont différents en fonction du lieu où l'on se trouve comme dans sa propre maison ou dans le métro, dans un lieu fréquenté ou peu fréquenté. Dans ces lieux, différemment accessibles pour les personnages en fonction de s'ils sont vaccinés ou pas, le personnage possède un attribut *donneeLieu* qu'il remplit lors de son entrée dans le lieu.

Les différents lieux sont destinés :

- ➔ Aux courses : le supermarché et la pharmacie. Dans ces lieux, la donnée est le nombre d'articles que le joueur souhaite acheter.
- ➔ Aux loisirs : la salle de sport, le cinéma et le bar. Dans ces lieux, la donnée est respectivement le nombre de machines utilisées, le nombre de films visualisés et le nombre de bières bues.
- ➔ Au travail : l'entreprise et l'université. Dans ces lieux, la donnée est le nombre d'heures passées à travailler sur le lieu.
- ➔ Aux transports : le métro et le bus. Dans ces lieux, la donnée est le nombre de kilomètres parcourus avec ce moyen de transport.

Chaque lieu redéfinit une fonction virtuelle "*impact*", qui actualise les données de l'état physique, de l'état mental et de l'argent du personnage une fois qu'il s'est rendu dans ce lieu. Pour cela, on récupère la map du lieu en question pour obtenir le nombre de personnes non immunisées et on calcule l'impact sur le physique : plus l'affluence du lieu est élevée, plus la donnée du lieu est grande, plus le facteur de contamination est élevé, plus cela a un impact négatif sur le personnage. Pour certains lieux, lorsque la donnée du lieu du personnage est égale à la valeur maximale du lieu, cela peut avoir un effet doublement négatif.

- *La ville*

Nous avons créé une dernière classe : la classe Ville. Dans cette classe, tous les lieux composant la ville sont créés, ainsi que les personnages composant les lieux. Elle possède également une map : la clé indique le nom du lieu, à laquelle on associe une liste de Personnages. Cette liste est initialisée après le parcours de la liste des personnages : on récupère le nom du lieu dans lequel se trouve le personnage pour l'ajouter à la liste propre au lieu et actualiser l'affluence du lieu de la ville. On attribue chacun de ces vectors de Personnages à chacune des clés de la map. Par la suite, on initialise les map de chaque lieu pour répertorier le nombre de personnes immunisées et non immunisées présentes dans chaque lieu.
Cette classe possède plusieurs fonctions : la première "impactDuLieu", qui recrée les différents lieux pour permettre d'appeler la bonne fonction "impact" par la suite sur le personnage. On parcours dans cette fonction la liste des lieux, puis en fonction du nom du lieu, du lieu dans

lequel se trouve le personnage, on met à jour l'affluence du lieu et son nombre de personnes immunisées pour appeler la fonction impact avec les bons paramètres mis à jour.
Elle possède également une fonction "ajoutPersoVille", qui permet d'ajouter un personnage dans le bon lieu de la ville en fonction du lieu qu'on lui a attribué lors de sa construction.

- *L'évolution du joueur dans le jeu*

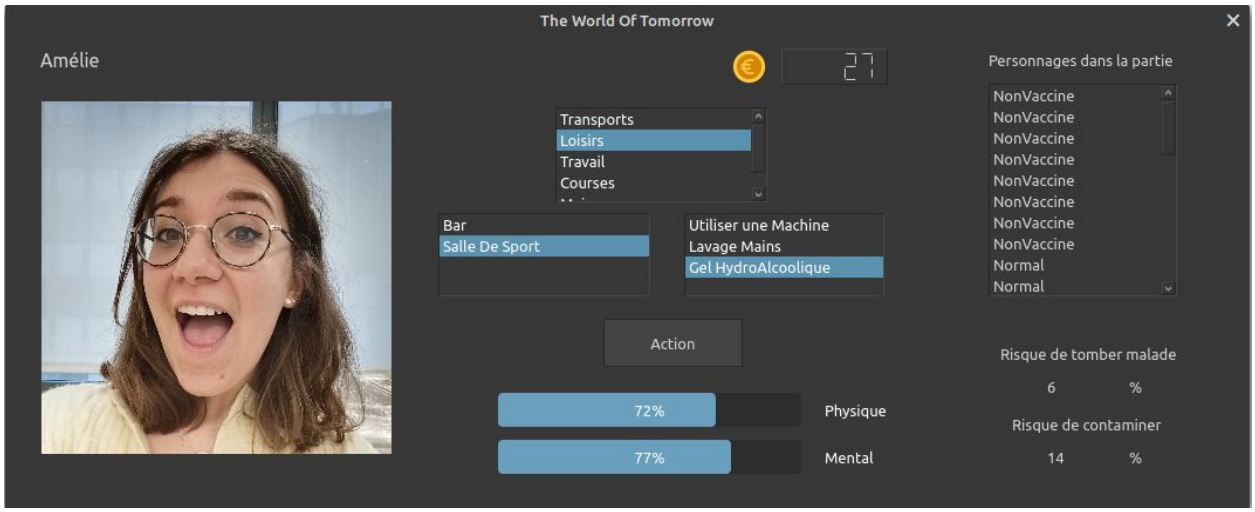
Le joueur choisit tout d'abord s'il souhaite être le personnage "Fadi" ou le personnage "Amélie" au début de la partie. Avec "Fadi", il possède un physique de 65 et un mental de 36 et avec "Amélie", il possède un physique de 36 et un mental de 65. Il choisit également un niveau de difficulté : "hard", qui correspond à un nombre de 10 pièces pour la partie et "easy", qui correspond à un nombre de 50 pièces. Ces données sont évidemment modifiables dans la classe Personnage.

Voici à quoi ressemble la fenêtre de démarrage du choix du joueur et de la difficulté. On notera que le joueur est obligé de sélectionner un joueur et un niveau de difficulté pour commencer la partie sinon, une fenêtre l'avertira du fait qu'il manque une des deux données.

Interface graphique lors du démarrage

Dans cette première version du jeu, nous avons décidé que le joueur ne pouvait évoluer qu'en tant que personnage "*NonVaccine*" : il ne peut donc pas se rendre au cinéma.
Une fois les différents paramètres de jeu choisis, le joueur accède à la ville. Les différents types de lieux apparaissent (travail, loisir, course, maison), puis les différents lieux de ce type lorsqu'il est choisi apparaissent et l'utilisateur peut choisir dans lequel il veut se rendre.

Une fois arrivé dans le lieu, il clique sur l'action qu'il veut effectuer et clique sur le bouton "Action" . Une fenêtre apparaît afin de récupérer la donnée du lieu, qui dépend du lieu dans lequel le personnage se trouve. C'est en fonction de la donnée saisie au clavier ou à la souris que la donnée du lieu se met à jour. Ensuite, il lui suffit de cliquer sur "Ok" pour exécuter la commande, ou bien sur "Cancel" pour l'annuler. A partir de ce moment, la fonction *impactDuLieu* est appelée et met à jour les barres du physique et du mental, le nombre de pièces du joueur et le pourcentage de chance d'être contaminé, ou de contaminer quelqu'un.
Le joueur peut voir également les taux de contamination le concernant (contaminé) ou concernant les autres (contaminant) à sa droite, qui évoluent en fonction des actions qu'il effectue.
Le joueur a également un visuel de son personnage représentant l'état courant de son personnage : la photo change en fonction du chiffre de son état mental et de son état physique.
La partie se termine si une des barres du mental ou du physique atteint la valeur 0 ou si le joueur n'a plus de pièce.



Interface graphique lors du déroulement du jeu

- **Fiertés**

Nous avons plusieurs fiertés concernant ce projet. Tout d'abord, nous sommes fiers de l'évolution de notre idée de base, à savoir l'apport de la dimension interactive avec l'utilisateur. Celui-ci peut choisir de se rendre dans différents lieux, dans lesquels il peut choisir la distance qu'il veut parcourir, le nombre d'articles qu'il veut acheter, le nombre d'heures qu'il veut travailler ou encore le nombre d'éléments qu'il souhaite avoir pour se divertir. Il possède également ses propres indicateurs concernant son physique, son mental et son argent. Tout cela est visible sur notre interface graphique.

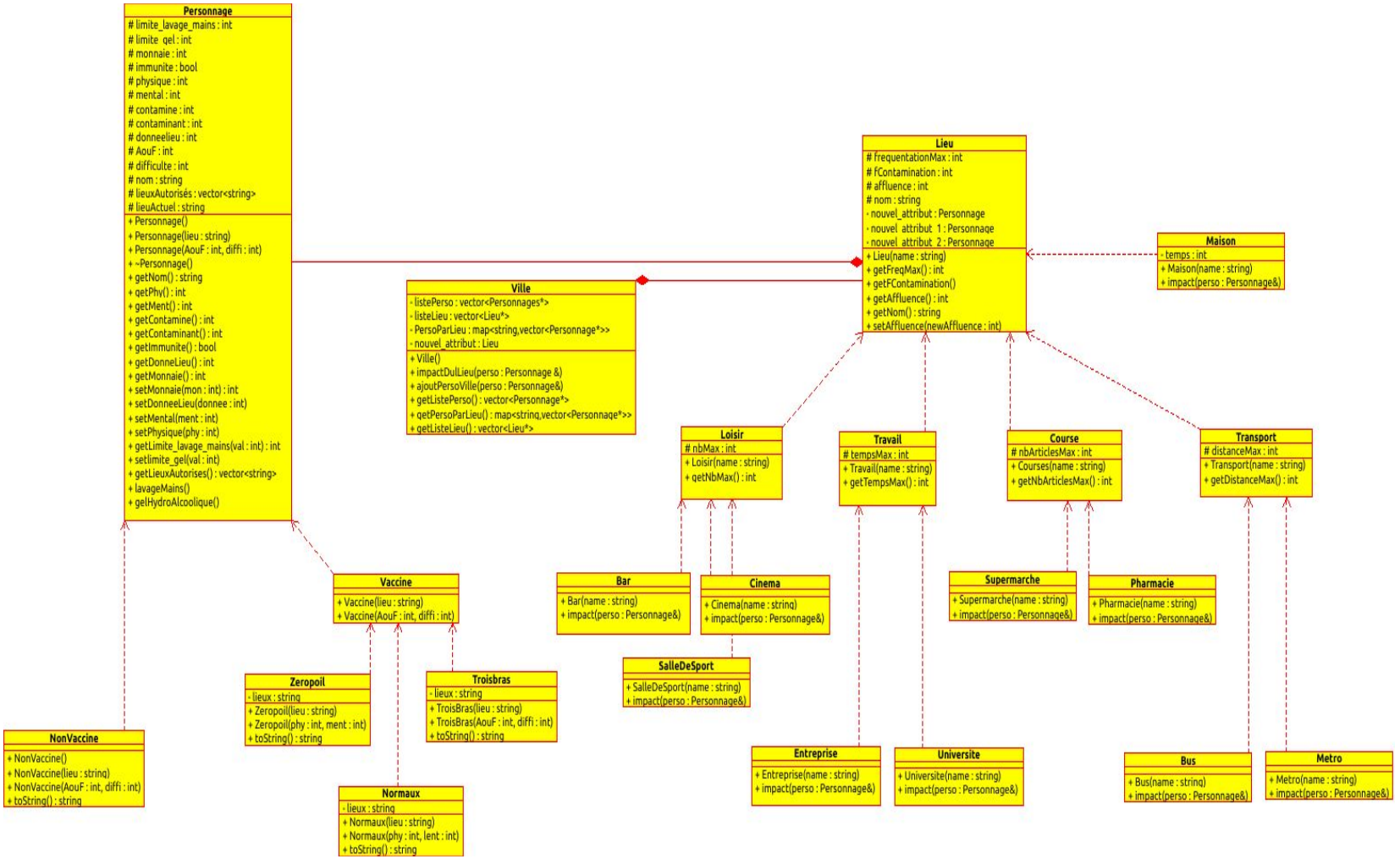
Cette interface graphique fait également partie de nos fiertés. En effet, nous avons pu découvrir un nouvel outil et sommes fiers d'avoir réussi à manipuler ces nouveaux outils. Cela a, de plus, représenté une difficulté tout au long du projet : de la prise en main du logiciel jusqu'à l'intégration du code développé sur SublimeText.

De plus, nous sommes fiers du code concernant les maps dans les classes des lieux et dans la classe Ville. Nous pensons avoir acquis les connaissances concernant ces outils du C++.
Enfin, nous pensons que notre jeu n'est pas exploité à son maximum : nous avons voulu faire au plus simple pour cette première version et maximiser les objectifs de base fixés. Pour autant, nous sommes sûrs qu'il possède plusieurs axes d'améliorations et de possibilités. Par exemple, la surcharge de l'opérateur "<" n'est pas très utile pour l'application actuelle, mais pour une autre où il faudrait effectuer un choix pour vacciner une personne en priorité par rapport à une autre, au cas où il y aurait des pénuries de vaccin par exemple, cela serait intéressant de savoir lequel des deux est le plus faible. De plus, il pourrait être intéressant d'exploiter plus les indicateurs "contaminé" et "contaminant" des personnages pour avoir une simulation plus réaliste. De la même manière, les chiffres pourraient être plus réalistes concernant les fonctions "impact" de chaque lieu et pourrait ainsi servir à réaliser de vraies simulations avec d'autres types de personnages plus "réels", comme des personnages âgées, des étudiants ou des nourrissons.

Nous avons cependant des insatisfactions vis-à-vis de ce projet, notamment de ne pas avoir réussi à appeler la bonne fonction impact sur le lieu et d'avoir été obligé de recréer tous les lieux avec les bonnes données dans la fonction "*impactDuLieu*" dans la classe Ville. Enfin, nous aurions aimé pouvoir récupérer le nom de la classe pour que certaines actions soient facilitées comme éviter de passer par le nom du lieu pour identifier sa classe et pouvoir ainsi lui donner un nom un peu plus intéressant comme "Sorbonne" pour un objet Université ou "Ligne7" pour un objet Métro.

- **Diagramme UML**

Comme nous pouvons le voir sur le diagramme UML ci-dessous, nous avons bien respecté le minimum de 8 classes, avec un total de 22 classes, et bien les 3 niveaux de hiérarchie. Cela a été difficile à suivre à cause de la quantité de classes lors du développement, mais pratique cependant, puisque toutes les fonctions étaient bien à leur place, permettant une certaine fluidité dans le code. Le diagramme est fourni en plus pour une meilleure qualité de visualisation.


Personnage
limite_lavage_mains : int
limite_gel : int
monnaie : int
immunite : bool
physique : int
mental : int
contamine : int
contaminant : int
donneelieu : int
AouF : int
difficulte : int
nom : string
lieuxAutorisés : vector<string>
lieuActuel : string
+ Personnage()
+ Personnage(lieu : string)
+ Personnage(AouF : int, diffi : int)
+ ~Personnage()
+ getNom() : string
+ getPhy() : int
+ getMent() : int
+ getContamine() : int
+ getContaminant() : int
+ getImmunite() : bool
+ getDonneLieu() : int
+ getMonnaie() : int
+ setMonnaie(mon : int) : int
+ setDonneeLieu(donnee : int)
+ setMental(ment : int)
+ setPhysique(phy : int)
+ getLimite_lavage_mains(val : int) : int
+ setlimite_gel(val : int)
+ getLieuxAutorises() : vector<string>
+ lavageMains()
+ gelHydroAlcoolique()
Lieu
frequentationMax : int
fContamination : int
affluence : int
nom : string
- nouvel_attribut : Personnage
- nouvel_attribut_1 : Personnage
- nouvel_attribut_2 : Personnage
+ Lieu(name : string)
+ getFreqMax() : int
+ getFContamination()
+ getAffluence() : int
+ getNom() : string
+ setAffluence(newAffluence : int)
Maison
- temps : int
+ Maison(name : string)
+ impact(perso : Personnage&)
Ville
- listePerso : vector<Personnages*>
- listeLieu : vector<Lieu*>
- PersoParLieu : map<string,vector<Personnage*>>
- nouvel_attribut : Lieu
+ Ville()
+ impactDuLieu(perso : Personnage &)
+ ajoutPersoVille(perso : Personnage&)
+ getListePerso() : vector<Personnage*>
+ getPersoParLieu() : map<string,vector<Personnage*>>
+ getListeLieu() : vector<Lieu*>
Loisir
nbMax : int
+ Loisir(name : string)
+ getNbMax() : int
Travail
tempsMax : int
+ Travail(name : string)
+ getTempsMax() : int
Course
nbArticlesMax : int
+ Courses(name : string)
+ getNbArticlesMax() : int
Transport
distanceMax : int
+ Transport(name : string)
+ getDistanceMax() : int
Vaccine
+ Vaccine(lieu : string)
+ Vaccine(AouF : int, diffi : int)
Bar
+ Bar(name : string)
+ impact(perso : Personnage&)
Cinema
+ Cinema(name : string)
+ impact(perso : Personnage&)
Supermarche
+ Supermarche(name : string)
+ impact(perso : Personnage&)
Pharmacie
+ Pharmacie(name : string)
+ impact(perso : Personnage&)
NonVaccine
+ NonVaccine()
+ NonVaccine(lieu : string)
+ NonVaccine(AouF : int, diffi : int)
+ toString() : string
Zeropoil
- lieux : string
+ Zeropoil(lieu : string)
+ Zeropoil(phy : int, ment : int)
+ toString() : string
Troisbras
- lieux : string
+ TroisBras(lieu : string)
+ TroisBras(AouF : int, diffi : int)
+ toString() : string
SalleDeSport
+ SalleDeSport(name : string)
+ impact(perso : Personnage&)
Normaux
- lieux : string
+ Normaux(lieu : string)
+ Normaux(phy : int, lent : int)
+ toString() : string
Entreprise
+ Entreprise(name : string)
+ impact(perso : Personnage&)
Universite
+ Universite(name : string)
+ impact(perso : Personnage&)
Bus
+ Bus(name : string)
+ impact(perso : Personnage&)
Metro
+ Metro(name : string)
+ impact(perso : Personnage&)