Qui est-ce ?
Projet de programmation

Groupe *AB*
Martinet Hugo, Xihao Wang, Ziyu Gao
https://github.com/m-hugo/GuessWho
L2 informatique
Faculté des Sciences
Université de Montpellier.

16 avril 2022

1 Technologies utilisées et organisation

1.1 Choix du langage

— Le language rust est utilisé pour l'integralité du projet
— La bibliotheque graphique fltk-rs est utilisée pour l'affichage de la fenetre et des widgets, la fonctionalité de gestion des evenements par message incluse avec les widgets est très pratique

La blibliothèque graphique egui à été fortement considérée et un prototype à été crée mais le concept d'"immediate mode" n'était pas facile à utiliser quand on fait une application qui doit initialiser beaucoup de donnés pendat l'execution, garder beaucoup d'états en memoire et avoir des widgets qui en influencent beaucoup d'autres
— Le framework rocket est utilisé pour avoir un serveur similaire à node.js (mais avec la vitesse de rust !) pour le mode multijoueur
— La blibliotheque reqwest est utilisée coté client pour acceder au serveur
— La bibliotheque rand est utilisée pour tirer un personage aléatoire, le tirage de nombres aléatoires n'etant pas present dans la bibliotheque standard de rust
— La bibliotheque serde est utilisée pour importer et exporter le json (planches et sauvegarde)

1.2 Organisation du travail

1. Répartition du travail au sein du groupe
J'ai commencé à faire le prototype tout seul avant de trouver un groupe puis j'ai travaillé avec Xihao Wang, Ziyu Gao à partir de la première séance, nous avons fait beaucoup de code ensemble mais ils ont aussi fait du code tout les 2, publié sous xihao-wang, et j'ai aussi fait pas mal de code moi même, publié comme le travail de groupe sous m-hugo
J'ai fait l'architecture des programmes ainsi que le choix des bibliotheques, les deux parties les plus compliquées quand on fait un projet en rust car une bonne architecture nous permet de rendre le programme beaucoup plus interactif en evitant les problemes de "borrowing", d'"ownership" et de durée de vie des variables. Les bibliotheques rust sont très nombreuse et la plupart ne marchent pas ou manquent de fonctionalités

2. Rythme de travail, mode de fonctionnement ... A la première seance je leur ai appris les bases du rust, nous avons ensuite travaillé ensemble 2-3 heures par semaine le lundi et nous finissions certaines taches de notre coté pendant la semaine

2 Étape 1 : permettre à l'utilisateur de jouer

— Fonctionnalités de l'application : interactions possibles de l'utilisateur.
— Format du fichier JSON, contraintes éventuelles.

```
{
"attrs": {
 "Nom":["nom1", "nom2"],
 "attribut1":["possibilité1", "possibilité2"],
 "attribut2":["possibilité1"]
},
"liste":[
  {
    "Nom":"nom1",
    "attribut1": "possibilité1",
    "attribut2": "possibilité1",
    "image": "./personnages/nom1.png"
  },
  {
    "Nom":"nom2",
    "attribut1": "possibilité2",
    "attribut2": "possibilité1",
    "image": "./personnages/nom2.png"
  },
  (22 autres "{}")
]
}
```

"attrs" contient tout les attributs et toutes les possibilités pour chaque attribut "liste" contient 24 blocs, chacun contenant tous les attributs definis dans "attrs" suivi de "image" avec le chemin de l'image depuis ce dossier ou un chemin absolu

3 Étape 2 : aider à la saisie des personnages

— Scénario des interactions avec l'utilisateur.
Nous avions prevu de permettre aux utilisateurs de valider les attributs/valeurs avec la touche entrée pour pouvoir en saisir plusieurs à la suite sans utiliser la souris (il est possible de savoir quelle boite à le focus) mais nous n'avons pas reussi, la saisie des attributs reste quand meme assez intuitive

4 Étape 3 : Mode Multijoueur

l'extension réalisée est le mode multijoueur : lors du choix du jeu de cartes, le joueur peut cocher un case qui active le mode multijoueur, il n'a plus qu'à rentrer l'ip du serveur, choisir j1 ou j2 et selectioner le même jeu de cartes que son adversaire, une fenetre avec le plateau de l'adversaire s'affiche alors et lors du tour de l'adversaire, je joueur peut voir quelle question il pose et quels personnages il renverse.
Nous avons seulement eu le temps de mettre en place les contraintes necessaires au bon fonctionnement du programme comme l'affichage d'un message d'erreur si l'ip ne pointe pas vers un serveur valide, le programme n'empeche pas à un joueur de poser une seconde question avant que l'autre n'ai pu poser la sienne mais c'est nous supposons que les joueurs respectent les règles du jeu, cela permet egalement de laisser son adversaire poser plusieurs questions pour chaque question qu'il pose lorsqu'il joue contre un enfant par exemple.

Nous utilisons des routes rattachées à des fonctions comme node.js, le seveur execute differentes fonctions selon le format de l'url utilisée pour y acceder et renvoie une chaine de characteres

5 Bilan et Conclusions

L'utilisation de callbacks (add) pour la gestion des evenements boutons n'a pas très bien fonctionné car en rust deux boutons ne peuvent pas écrire dans la meme zone de texte car, sans enrober les variables dans des types complexes, elle ne peuvent être empruntées en lecture en écriture que si elles ne sont pas empruntées en lecture ailleurs
mais après les avoir remplacés par les emmisions de messages (add_emit) j'ai pu faire un systeme assez elegant : les boutons ecrivent dans des tuyaux et la boucle principale du programme est la seule à effectuer des operations sur les elements en fonction du type du message et de son contenu.