Table des matières

1. Fonctionnement de l’équipe et organisation du travail

- **Présentation du groupe**

1. Tertilov Vadim APP1 TD-f TP-7

2 - Safaa Benmoussa APP1 TD-f TP-7

3 – Kopylov Mykyta APP1 TD-f TP-7

- **Répartition du travail**

Par tâche :

Mykyta:

Menu principal (partie visuelle)
Algorithme de recherche en largeur (BFS)
Rapport

Tertilov Vadim:

Architecture du projet
Logique de base
Lier les parties front-end et back-end.

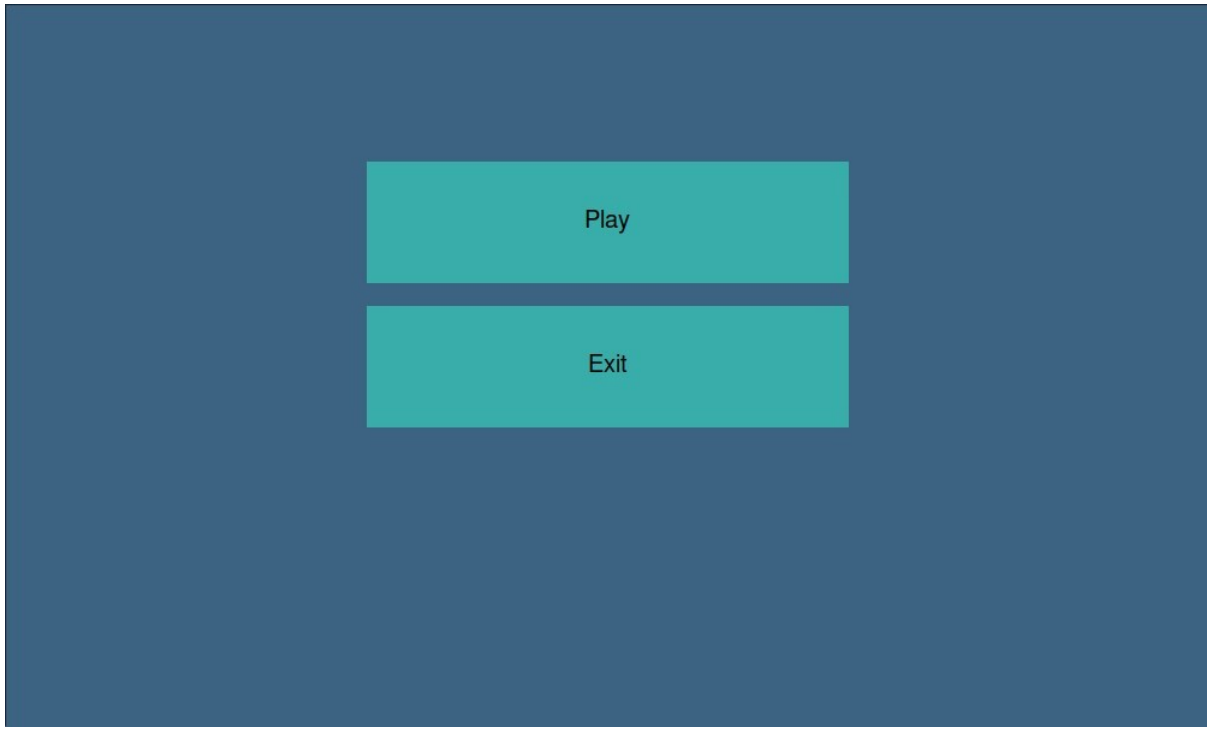
Safaa Benmoussa:

Interface graphique du terrain de jeu
Rapport
Textures supplémentaires

- **Difficultés rencontrés**

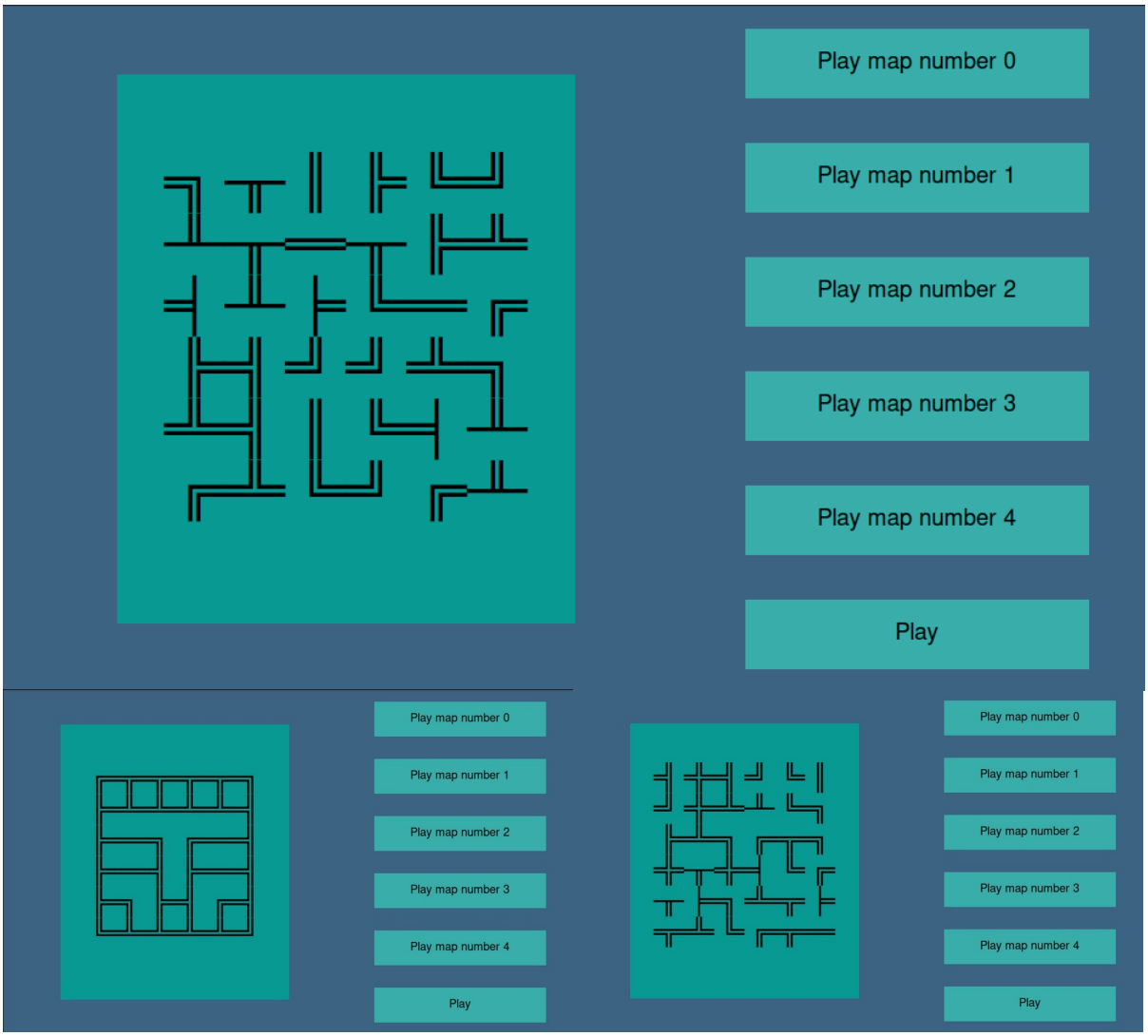
1. *Communication*
   La barière linguistique fut un frein à la communication durant le projet. Malgré le fait que nous parlons tout trois anglais de manière correcte.

Utilisation



L'interface principale est banale : un arrière-plan uniforme et deux boutons. En appuyant sur le bouton Play, vous accédez au menu de sélection de la carte (que vous pouvez trouver/ajouter/remplacer dans le répertoire maps à la racine du projet).
Cliquer sur exit fermera simplement la fenêtre du jeu et arrêtera l'exécution du programme.

Lors de l'écriture de l'interface principale, nous avons essayé de nous en tenir aux principes de base des combinaisons de couleurs, de sorte que seules trois couleurs ont été choisies comme base.
Voici les codes HEX de chacune d'entre elles :
Fond - #3c6382
Couleur des boutons - #38ada9
Couleur du texte des boutons - #000000



Vous pouvez maintenant voir le menu de sélection de la carte. Sur le côté droit de l'écran se trouve une série de boutons permettant de sélectionner la carte que vous souhaitez jouer.
Sur le côté gauche de l'écran, vous pouvez voir une fenêtre de prévisualisation qui change en fonction de la carte que vous avez sélectionnée.
Vous pouvez choisir parmi un ensemble de 5 cartes standard, que vous pouvez ajouter ou modifier en entrant le dossier approprié à la racine du projet.

État d'avancement du projet

La première chose que nous voulions faire était de trouver la structure de données la plus efficace, pour une mise en œuvre confortable du jeu et un processus de débogage simplifié par la suite. Après avoir discuté de plusieurs options, nous avons opté pour une implémentation utilisant des opérations binaires.
L'utilisation d'opérations binaires nous a permis d'éviter un grand nombre d'opérations dans le projet pour faire pivoter les pièces et leur affichage ultérieur.
Au lieu de cela, nous mettons tous les caractères ASCII (représentant notre carte) dans un dictionnaire et nous écrivons ensuite une méthode pour les manipuler (par exemple, nous utilisons l'extrait de code suivant pour faire pivoter une pièce spécifique vers la droite :

```
def rotate_right(self) -> None:
    """
    Rotates the room right
    """
    self.value = (self.value << 1) & 0xF | (self.value >> 3)
```

De cette façon, avec un décalage bit à bit, nous pouvons effectuer un "retournement" d'un caractère, ce qui évite un grand nombre de données pré-remplies et rend le code beaucoup plus court et plus facile à lire.

L'ensemble du projet a été écrit selon une structure orientée objet.
Classes principales :
Room - une classe avec des méthodes de base pour travailler avec des pièces (comme tourner une pièce, vérifier les pièces adjacentes et ainsi de suite),
Entity - classe abstraite décrivant le fonctionnement des unités principales : le Dragon et le Chevalier. Dans notre cas : le processus de mouvement et de mort.
Donjon - la méthode principale pour travailler avec les cartes et combiner tous les objets du jeu (y compris le joueur, le chevalier, le dragon et l'ensemble des récompenses).

Un système de statistiques a été écrit pour mettre en œuvre les changements d'état (quelque chose de similaire peut être vu dans les bibliothèques aiogram et pyTelegramBotAPI pour développer des robots télégraphiques). Dans notre cas, il s'agit du système de base pour implémenter la commutation entre les éléments de l'interface et la transition de la phase d'installation jusqu'à la phase de lancement du jeu.

La principale donnée transférée est le nom du fichier .txt, qui est la carte de l'utilisateur.

La partie visuelle est écrite dans la bibliothèque FLTK, également selon les principes OOP.

Opinion sur le projet : le projet peut être assez divertissant, mais ce n'est pas pour autant que nous l'avons apprécié. Tout d'abord, cela est dû à la tâche technique qui, à notre avis, aurait pu être beaucoup plus intéressante et beaucoup plus proche de l'idée originale.
Il nous semble également que l'utilisation de FLTK n'est pas la décision la plus correcte, puisque la bibliothèque elle-même limite le développeur avec ses fonctionnalités et qu'il est beaucoup plus logique d'écrire ce projet sur tkinter (sur lequel fltk a été basé) ou sur pygame, piglet, arcada, QT et des bibliothèques similaires, qui ont une mise en cache d'image correcte, un ensemble d'outils nécessaires pour créer l'interface et également une structure de type OOP, permettant de créer deux ou plusieurs fenêtres et de contrôler chaque fenêtre séparément, ce qui rend le processus de développement beaucoup plus flexible.