SMP – Rapport de TP : Ricochet Robot

Etudiants :

- Adji Touré
- Amaury Lejolivet
- Baptiste Dubot
- Guillaume de Maricourt

Ce TP n'a pas pu être terminé dans les temps impartis par manque de temps. Par ailleurs nous certifions ne pas avoir eu recours à chatGPT (ou autre…) afin de créer nos différents modules.

Repository github : https://github.com/ECN-SEC-SMP/tp-note-2023-bapt-amo-adji-gui.git

Lien replit : https://replit.com/join/pvzvinqaph-baptdbt

Note 1 : Il est déconseillé de faire tourner le script sur replit car l'affichage ne peut pas être fait sur une si petite fenêtre.

Note 2 : Il est nécessaire de changer de branche et d'aller sur la branche « Board » qui est encore en cours de développement en suivant le lien Github…

Details techniques sur ce projet

Certains tests ont été réalisés pour les classes simples et sont disponibles dans le dossier « tests/ ». Ces tests ne sont pas des tests unitaires mais basés sur des assertions. Nous n'avons pas appris à « stubber » des fonctions et ne sommes donc pas capables de réaliser ces tests unitaires (qui sont aussi très chronophages). Des ébauches d'algorithmes papier ont été réalisés mais non documentés.

Nous avons utilisé Github pour ce TP d'une manière plutôt optimale avec un mécanisme de branches et de pull-request/reviews. Ce type de travail est équivalent au travail réalisé dans la plupart des entreprises qui ont des équipes de développement logiciels composées de bon développeurs juniors et seniors.

Nous avons réalisé une documentation détaillée avec l'outil Doxygen. Cette documentation se trouve sur le repository Github dans le dossier « html ». Les fichiers qui ont été détaillés sont les suivants :

- Helper.cpp/h
- Player.cpp/h
- Display.cpp/h
- Board.cpp/h

Problèmes rencontrés

Lors de ce TP nous avons rencontré plusieurs soucis techniques (comme les inclusions circulaires infinies, qui empêchent toute compilation) mais aussi humain. Le travail le plus difficile a été de relier les modules ce que nous n'avons pas réussi à faire malgré le diagramme de classes mais aussi la gestion du temps. Les modules, au fur et à mesure qu'on les développe, sont amenés à évoluer (pour mieux correspondre aux besoins du programmeur.se) et il est difficile de se tenir à jour des besoins de chacun et de faire évoluer l'architecture. Cela démontre la difficulté du travail de développement avec la dépendance forte des modules entre eux. Cela a aussi été constaté sur les périodes en entreprise.

Inclusions circulaires infinies : Ce problème consiste à inclure un fichier dans un second qui inclus lui-même ce premier fichier. Nous n'avions pas utilisé les directives préprocesseur avec notamment le

« #pragma once » sur l'un des fichiers. Cela nous a valu l'échec de la compilation avec un « timeout » remonté par le thread de compilation (sur mingw32 notamment).

Implémentation du sablier : L'implémentation du sablier n'a pas été faite. Nous n'avons pas réussi à activer un thread qui déclenche un timeout de 60 secondes ou même réussi à initialiser un timer périphérique comme ce que nous avions vu en MAC.

Génération du plateau : La génération du plateau a été fastidieuse. Il nous a semblé que bien que **complet** le cahier des charges n'était pas assez explicite/détaillé. Cela nous a valu de fausses interprétations sur les besoins dans la génération de ce plateau et nous a fait réécrire plusieurs fois les algorithmes de génération.

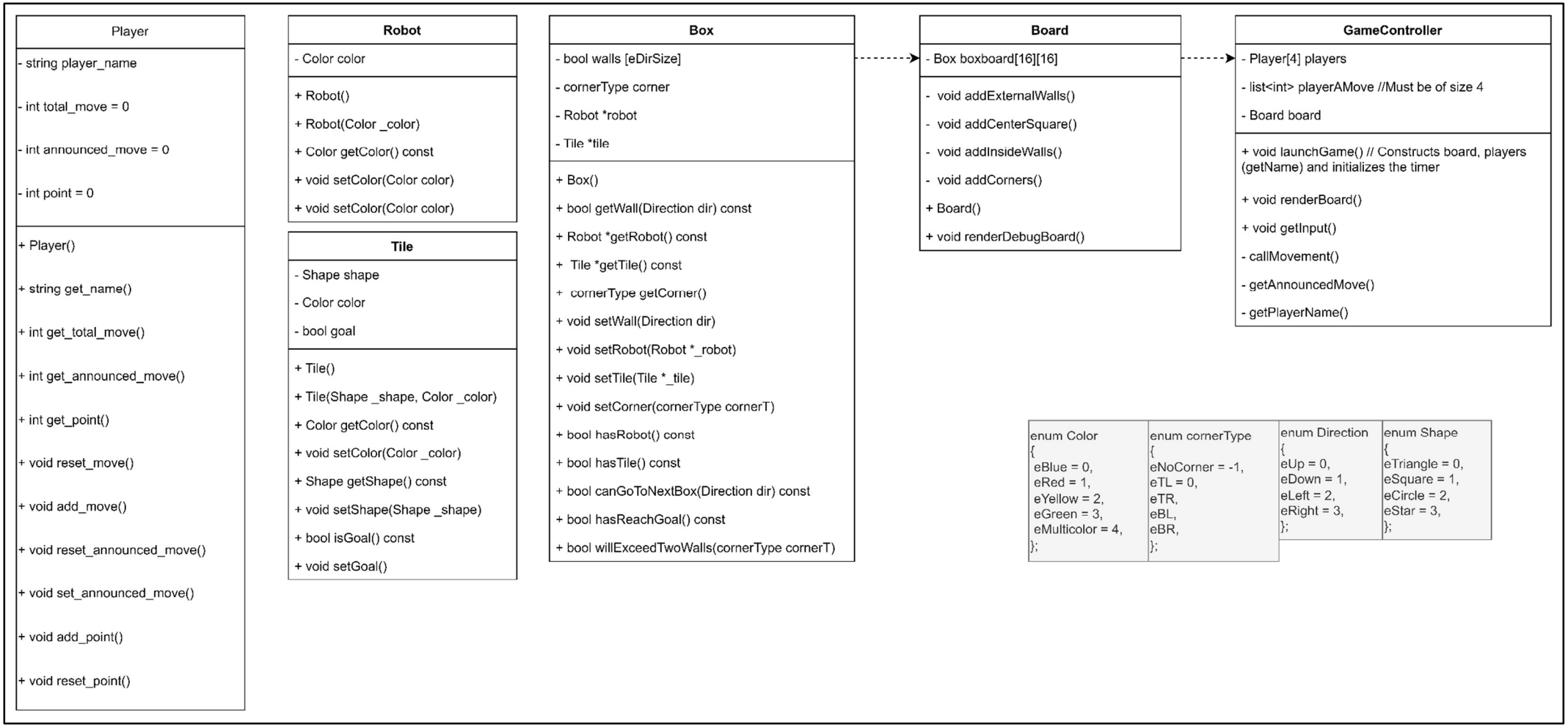
Utilisation de Replit : Replit est un très bon outil, très performant. Cependant, bien qu'utile au début de la formation, il nous a longtemps caché les affres de la compilation et l'utilisation de git avec des interfaces qui simplifient le tout. Il a été difficile de faire la transition entre replit et mingw/git. Cependant nous avons maintenant la compréhension nécessaire pour nous retirer de Replit.

Diagramme de classe et architecture du projet

Le diagramme de classe qui est actuellement proposé n'est pas forcément optimal. La génération du plateau cache par exemple beaucoup de fonctions qui simplifient l'écriture du code dans le fichier « class/board/helper.cpp ».

Cependant ce diagramme de classe à la qualité de rendre sa compréhension simple et clair (du moins de notre point de vue, c'est qui est discutable bien entendu). Nous avons décidé de simplifier le tout en utilisant des types énumérés pour gérer la direction, les formes, les couleurs et les types d'angle. Cependant l'implémentation actuelle n'est pas optimale d'un point de vue de la complexité en temps et de la complexité en espace. Cela reste entièrement améliorable et optimisable.

Le diagramme de classes est disponible à la page suivant


Player
- string player_name
- int total_move = 0
- int announced_move = 0
- int point = 0
+ Player()
+ string get_name()
+ int get_total_move()
+ int get_announced_move()
+ int get_point()
+ void reset_move()
+ void add_move()
+ void reset_announced_move()
+ void set_announced_move()
+ void add_point()
+ void reset_point()
Robot
- Color color
+ Robot()
+ Robot(Color _color)
+ Color getColor() const
+ void setColor(Color color)
+ void setColor(Color color)
Tile
- Shape shape
- Color color
- bool goal
+ Tile()
+ Tile(Shape _shape, Color _color)
+ Color getColor() const
+ void setColor(Color _color)
+ Shape getShape() const
+ void setShape(Shape _shape)
+ bool isGoal() const
+ void setGoal()
Box
- bool walls [eDirSize]
- cornerType corner
- Robot *robot
- Tile *tile
+ Box()
+ bool getWall(Direction dir) const
+ Robot *getRobot() const
+ Tile *getTile() const
+ cornerType getCorner()
+ void setWall(Direction dir)
+ void setRobot(Robot *_robot)
+ void setTile(Tile *_tile)
+ void setCorner(cornerType cornerT)
+ bool hasRobot() const
+ bool hasTile() const
+ bool canGoToNextBox(Direction dir) const
+ bool hasReachGoal() const
+ bool willExceedTwoWalls(cornerType cornerT)
Board
- Box boxboard[16][16]
- void addExternalWalls()
- void addCenterSquare()
- void addInsideWalls()
- void addCorners()
+ Board()
+ void renderDebugBoard()
GameController
- Player[4] players
- list<int> playerAMove //Must be of size 4
- Board board
+ void launchGame() // Constructs board, players (getName) and initializes the timer
+ void renderBoard()
+ void getInput()
- callMovement()
- getAnnouncedMove()
- getPlayerName()
enum Color
{
eBlue = 0,
eRed = 1,
eYellow = 2,
eGreen = 3,
eMulticolor = 4,
};
enum cornerType
{
eNoCorner = -1,
eTL = 0,
eTR,
eBL,
eBR,
};
enum Direction
{
eUp = 0,
eDown = 1,
eLeft = 2,
eRight = 3,
};
enum Shape
{
eTriangle = 0,
eSquare = 1,
eCircle = 2,
eStar = 3,
};