Analyse

Baptiste Bertout – author · Pierre Planchon – author · Arthur Keller – author · Gaspard Souliez – author · Mathis Decoster – author

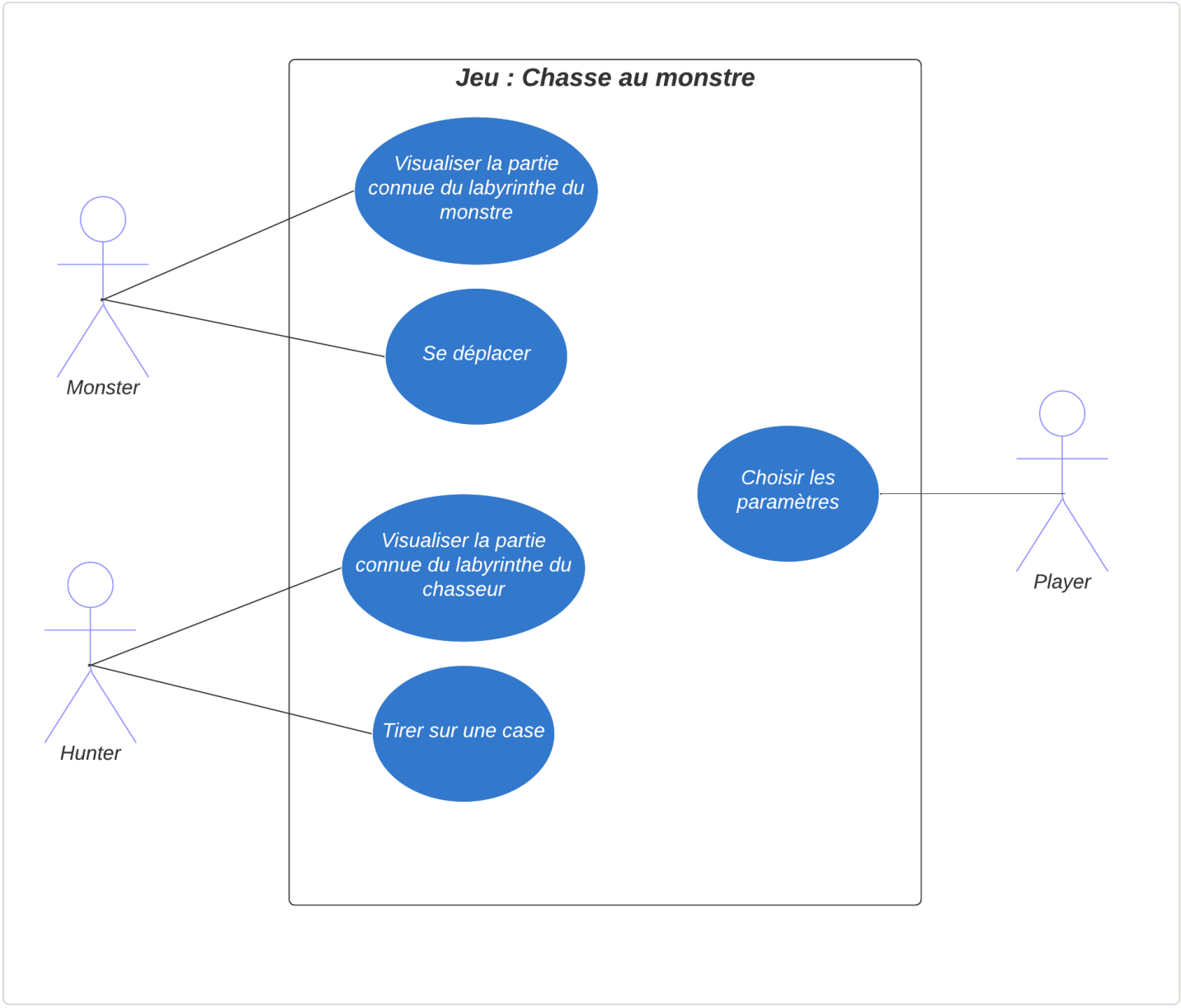
Table des matières

1. Présentation de l'équipe

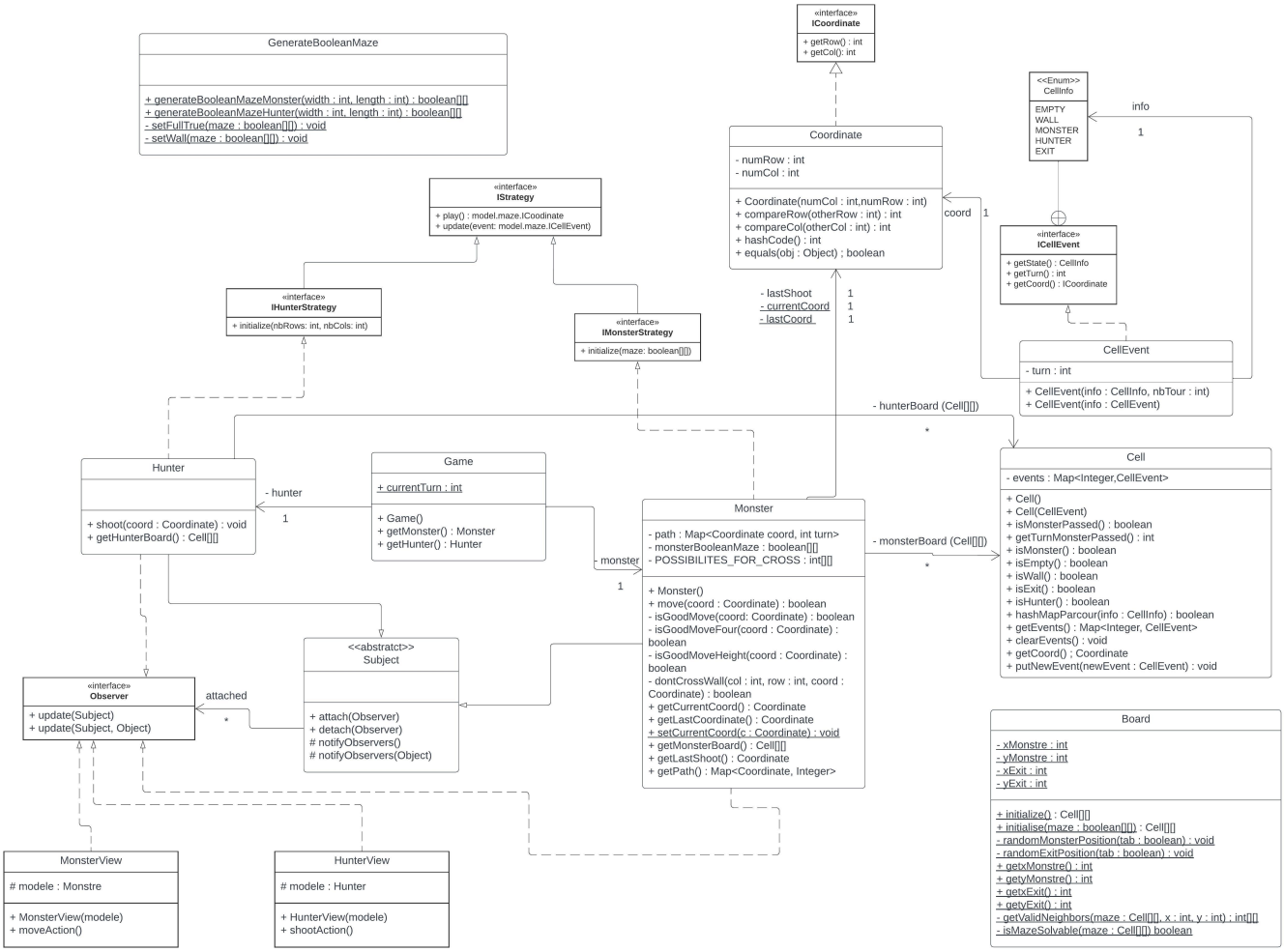
Groupe G5

- Baptiste Bertout
- Pierre Planchon
- Arthur Keller
- Gaspard Souliez
- Mathis Decoster

2. Diagramme de cas d'utilisation



3. Diagramme de classe



4. Description de l'implémentation

I. Tour du chasseur

Début du tour du montre

Selon le mode de vue que le joueur aura choisit dans les paramètres, il aura accès a certaines informations du labyrinthe ou à toutes les informations. Il aura toujours accès à la taille du labyrinthe, l'emplacement de la sortie et son propre emplacement. La vue du labyrinthe se fait par une IHM simple.

```
DEBUT
    Le joueur décide de se deplacer sur une case en utilisant l'interface IHM, avec un simple
clique.
    Coordinate coord <- coordonnées de la case choisit
    APPEL de la méthode move(coord)
FIN
```

Méthode move (Classe Monster)

```
FONCTION move(Coordinate coord)

    SI isGoodMove()
        lastCoord <- currentCoord
        currentCoord <- coord
        Ajout dans la Map path de la case choisit
        Si isExit()
            Le joueur gagne la partie
        Sinon
            notifyObservers() : On met à jour la vue du Monstre
    FIN SI

FIN FONCTION
```

Méthode isGoodMove (Classe Monster)

```
FONCTION isGoodMove(Coordinate coord)

    Si le paramètre de mouvement est définit sur 4
        isGoodMoveFour() On vérifie que le déplacement est possible à gauche, en haut, à droite et
en bas.
    Sinon
        isGoodMoveEight() On vérifie la même chose que pour le mouvement en 4 avec les digonales en
plus.

FIN FONCTION
```

Méthode isGoodMoveFour (Classe Monster)

```
FONCTION isGoodMoveFour(Coordinate coord)

    boolean goodCol <- on vérifie que la colonne soit à une colonne de plus ou de moins que celle du
monstre.
    boolean goodRow <- on vérifie que la ligne soit à une ligne de plus ou de moins que celle du
monstre.
    boolean isntWall <- on vérifie que la case sur laquelle on veut se déplacer n'est pas un mur.
    Si
        la case selectionnée est en diagonale : RETOURNER false
    RETOURNER goodCol ET goodRow ET isntWall;

FIN FONCTION
```

Méthode isGoodMoveEight (Classe Monster)

```
FONCTION isGoodMoveEight(Coordinate coord)

    boolean goodCol <- on vérifie que la colonne soit à une colonne de plus ou de moins que celle du
monstre.
    boolean goodRow <- on vérifie que la ligne soit à une ligne de plus ou de moins que celle du
monstre.
    boolean isntWall <- on vérifie que la case sur laquelle on veut se déplacer n'est pas un mur.
    RETOURNER goodCol ET goodRow ET isntWall ET dontCrossWall();

FIN FONCTION

    * La méthode dontCrossWall() vérifie que le joueur ne traverse pas deux murs en diagonale de la
façon suivante :

    M | W
    -----
    W | *

    Avec M étant le monstre, W un mur et * l'endroit ou il veut aller.
```

II. Tour du chasseur

Début du tour du chasseur

Le chasseur n'a accès à aucune informations, si ce n'est que la taille du labyrinthe. La vue du labyrinthe se fait par une IHM simple.

```
DEBUT
    Le joueur décide de tirer sur une case avec l'interface IHM, avec un simple clique.
    Coordinate coord <- coordonnées de la case choisit
    APPEL shoot(coord)
FIN
```

Méthode shoot (CLasse Hunter.java)

```
FONCTION shoot(Coordinate coord)

    notifyObservers(coord)

FIN FONCTION
```

Méthode Update de Monster

notifyObservers(coord) appelle la méthode update(Subejct, Object) de l'objet Monster Pour faciliter la communication entre les classes Hunter et Monster et leurs vues respectives un traitement est fait au début de cette méthode. Ce traitement consiste à vérifier que le sujet qui envoie le signale est bien une instance de la classe Hunter et que la donnée transmise est bien une instance de Coordinate

```
FONCTION Update(Subject sujet, Object data)

    Si (traitement du sujet et de la donnée)
        lastShoot <- (Coordinate) data
        notifyObservers() : On informe la vue du Monstre que le chasseur a tiré ce qui permet de
l'afficher.

        Si currentCoord == data
            notifyObservers() : On informe le chasseur qu'il a touché le Monstre
        Sinon si le chemin traversé par le Monstre contient les coordonnées demandées
            notifyObservers() : On informe le chasseur de l'information de la case sur laquelle il
vient de tirer
        Sinon si la case est un mur
            notifyObservers() : On informe le chasseur que c'est un mur
        Sinon
            notifyObservers() : On informe le chasseur que c'est une case vide

FIN FONCTION
```

Méthode Update de Hunter

notifyObservers(coord) appelle la méthode update(Subejct, Object) de l'objet Hunter Pour faciliter la communication entre les classes Hunter et Monster et leurs vues respectives un traitement est fait au début de cette méthode. Ce traitement consiste à vérifier que le sujet qui envoie le signale est bien une instance de la classe Monster et que la donnée transmise est bien une instance de CellEvent

```
FONCTION Update(Subject sujet, Object data)

    Si (traitement du sujet et de la donnée)
        Si isMonster()
            Le chasseur gagne la partie.
        Sinon
            notifyObservers() : On informe la vue du chasseur pour qu'il mette à jour la case avec
l'inforations donnée.

FIN FONCTION
```

Last updated 2023-11-10 00:01:09 +0100