RAPPORT DE PROJET

Louis HUORT  
Grégoire DOLIDON  
Mathis GUESSARD  
Hakim IZM

Table des matières

[I. Notre jeu 2](#_Toc74813173)

[A. Le sujet du projet 2](#_Toc74813174)

[B. CrossyFrog 2](#_Toc74813175)

[C. Règles du jeu 2](#_Toc74813176)

[II. Périmètre du projet 3](#_Toc74813177)

[A. Cahier des charges de la V1 3](#_Toc74813178)

[1. Détails techniques du jeu : 3](#_Toc74813179)

[2. Mode de jeu : 3](#_Toc74813180)

[3. Choisir les personnages : 3](#_Toc74813181)

[4. Écran de chargement : 3](#_Toc74813182)

[5. Menu du jeu : 4](#_Toc74813183)

[6. Pattern : 4](#_Toc74813184)

[7. Blocs : 4](#_Toc74813185)

[8. Grenouille : 4](#_Toc74813186)

[9. Scores : 5](#_Toc74813187)

[10. Graphismes : 5](#_Toc74813188)

[B. Cahier des charges de la V2 5](#_Toc74813189)

[1. Mode de jeu : 5](#_Toc74813190)

[2. Effets sonores 6](#_Toc74813191)

[3. Graphismes 6](#_Toc74813192)

[4. Expérience utilisateur 6](#_Toc74813193)

[III. Conception du projet 7](#_Toc74813194)

[A. Diagrammes de classes 7](#_Toc74813195)

[B. Diagrammes de cas d’utilisation 8](#_Toc74813196)

[C. Diagrammes de séquence 10](#_Toc74813197)

# Notre jeu

## Le sujet du projet

Le sujet du PTS2 est de créer un jeu d’arcade inspiré d’un jeu existant ou en reproduisant un.

## CrossyFrog

Notre choix s’est porté sur Frogger, un jeu d’arcade paru en 1981 dans lequel le joueur contrôle une grenouille partant d’un point A et devant traverser une route et une rivière tout en évitant de se faire écraser par la circulation et de se noyer auquel cas la grenouille meurt et le joueur perd la partie.

Nous tâcherons de réaliser une version basée sur un mélange entre Frogger 1 et Frogger 2 puisque ce deuxième jeu prend en charge la génération d’une carte de jeu infinie et que nous aimerions implémenter cette fonctionnalité dans notre jeu.

De plus, le choix entre deux modes de jeu est prévu : le premier sera le mode « solo endless » où le joueur jouera seul et devra parcourir la plus grande distance possible pour obtenir le meilleur score tandis que le second mode de jeu sera le mode « 1 vs 1 » où deux joueurs s’affronteront sur le même plan de jeu, la caméra suit le joueur placé devant et celui-ci gagnera dès lors que l’autre joueur ne sera plus visible à l’écran ou si l’un des deux joueurs meurt, l’autre remporte la partie.

Puis, concernant l’interface du jeu, nous pensons à afficher un menu principal à l’ouverture dans lequel le joueur aura la possibilité de démarrer une partie, puis devra faire le choix entre les deux modes de jeu précédemment expliqués avant de choisir son personnage. Il aura également la possibilité de consulter le temple de la renommée où seront répertoriés les meilleurs scores, de régler les paramètres de jeu ou de quitter le jeu.

## Règles du jeu

Puisque le jeu est un « endless runner[[1]](#footnote-2) », le joueur ne peut pas gagner. En effet, la partie se termine dès lors que le joueur perd, c’est-à-dire dès qu’il touche un obstacle.

Les obstacles entraînant une fin de partie sont les suivants :

* Les voitures ;
* L’eau ;
* Le train ;
* Les buissons.

À la fin de la partie, le score du joueur peut être enregistré, s’il le souhaite. Les quatre meilleurs scores enregistrés seront affichés dans le tableau du « hall of fame ».

# Périmètre du projet

## Cahier des charges de la V1

### Détails techniques du jeu :

* Frames par seconde : 60 (60Hz = 16ms)
* Taille de l’écran : 17\*15 cases
* Taille d’une case : 52 pixels

### Mode de jeu :

Crossy Frog V1 n’aura qu’un seul mode de jeu fonctionnel, c’est un mode de jeu avec une carte sans fin (endless) où le joueur devra réussir à aller le plus loin possible, la vitesse de défilement de la carte augmente au fil du temps. Le score correspond au nombre de mètres parcourus (1 case = 1 mètre). La grenouille ne doit pas toucher les blocs d’eau ou les obstacles (comme les voitures).

Détails techniques :

* Vitesse du jeu : 10% - 99%
* Augmentation de la vitesse du jeu : +1% toutes les 12secondes
* Score en mètres, +1 à chaque bloc de haut en plus
* Si la grenouille est sur un bloc infranchissable (eau, voiture, etc.), la partie s’arrête. Détection des positions à chaque tour de boucle.
* L’affichage des patterns se fait à l’aide de 3 patterns comme vus sur les maquettes, dès qu’un pattern sort de l’écran, il retourne en haut et la disposition des blocs est changée.

### Choisir les personnages :

Dans n’importe quel mode de jeu, les personnages pourront être choisis parmi une liste de grenouilles réalisées par notre équipe. Le personnage au centre sera le personnage actuellement sélectionné et les autres seront disposés sur le côté, on naviguera entre eux de gauche à droite.

Détails techniques :

* Les grenouilles seront faites au format 32\*32 pixels
* Elles seront redimensionnées à la taille des cases du jeu soit 52\*52 pixels
* La navigation se fera à l’aide des touches ZQSD, des touches directionnelles et Entrée

### Écran de chargement :

Le jeu disposera d’un écran de chargement à son lancement car les images sont nombreuses et peuvent prendre du temps à charger lors du lancement du jeu.

Détails techniques :

* L’écran de chargement s’affichera avec la progression du chargement des items (progress bar)
* Lancement du menu après le chargement complet des items.

### Menu du jeu :

Le jeu disposera d’un menu contenant les éléments suivants : Play/Hall Of Fame/Settings/Exit. Chacun de ces menus donnera accès au widget correspondant.

Détails techniques :

* Tout se fera à l’aide d’un QPainter : l’option du menu en cours sera surlignée d’un carré blanc
* La navigation se fait à l’aide des touches ZQSD ou des touches directionnelles et de la touche entrée pour valider la sélection.

### Pattern :

Les patterns sont composés de cases de taille fixe, ces cases sont en fait des blocs avec une image et des propriétés correspondantes.

Détails techniques :

* Le pattern fait une taille de 17\*7 cases (donc la moitié de l’écran)
* Il y a donc 3 patterns au total pour avoir un jeu fluide

### Blocs :

Les blocs ont comme attribut une image et des propriétés comme (s’il est traversable, s’il donne de la vitesse, etc.)

Détails techniques :

* L’image du bloc est celle qui a été chargée au lancement du jeu pour éviter les lags lors de la génération.

### Grenouille :

La grenouille correspond à l’image choisie par le joueur avant la partie, elle a des attributs de position et une image.

Détails techniques :

* La grenouille correspond à l’image chargée au lancement du jeu
* Son changement de position se fait à l’aide des touches ZQSD ou des touches directionnelles et des QPressEvent associés, elle bouge à chaque mouvement de la taille d’un bloc soit : 52pixels
* À chaque mouvement, le jeu vérifie si celui-ci est possible, par exemple que ça ne ferait pas sortir la grenouille de la carte
* Si la grenouille est en dehors de la carte, alors la partie d’arrête

### Scores :

Les scores pourront, si l’utilisateur le souhaite, être enregistré à chaque fin de partie et, s’ils font partie des meilleurs, se retrouver dans le hall of fame.

Détails techniques :

* Le score sera enregistré dans un fichier local.
* Ce fichier sera une base de données avec l’extension “.db”.

### Graphismes :

Tous les graphismes du jeu seront réalisés par notre équipe.

Details techniques :

* Le format des images différera mais respectera toujours les tailles suivantes : 16x16 ou 32x32 ou 64x64 ou 128x128 (unité : pixel).

## Cahier des charges de la V2

### Mode de jeu :

Deux modes de jeu sont prévus pour la V2 : le mode de jeu initialement programmé en V1 (solo endless) et un mode multijoueur local 1vs1. Dans ce second mode de jeu, les deux joueurs s’affrontent sur le même ordinateur et jouent sur le même clavier : un joueur contrôle sa grenouille avec les touches (ZQSD) et l’autre contrôle l’autre grenouille avec les touches directionnelles. Aussi, l’objectif dans ce mode de jeu est de rester en vie le plus longtemps : le premier joueur mourant perd la partie et son adversaire la gagne.

Détails techniques :

* Les deux joueurs ne peuvent pas sélectionner le même modèle pour leur grenouille
* La partie se termine quand un des deux joueurs meurt

### Effets sonores

Des effets sonores seront omniprésents dans la V2 : des bruits pour la sélection dans les menus, pour le déplacement des grenouilles, pour le passage des trains et des voitures ainsi que pour la mort d’une grenouille. De plus, une musique de fond accompagnera les joueurs durant leur partie.

Détails techniques :

* Les musiques et effets sonores seront des issus de banques de données libres de droits

### Graphismes

Cette deuxième version apporte des améliorations graphiques diverses permettant un meilleur confort visuel et une meilleure immersion pour les joueurs. Parmi ces nouveautés comptent : une animation pour la mort d’une grenouille, une animation pour le déplacement d’une grenouille, une animation en fin de partie où la grenouille gagnante apparaît en faisant une célébration, l’ajout de différents biomes [[2]](#footnote-3)et l’ajout de rondins de bois sur les cours d’eau.

Détails techniques :

* Les changements de biomes s’opèrent à partir d’un certain nombre de points obtenus par le joueur.
* Création de nouvelles textures sur Krita.
* Quand une grenouille est sur un rondin de bois, elle se déplace en même temps que le rondin.

### Expérience utilisateur

De nouvelles fonctionnalités permettront au joueur de bénéficier d’une expérience de jeu personnalisée. En l’occurrence, les utilisateurs pourront créer et modifier des patterns qui constituent la carte de l’environnement de jeu. De plus, le joueur pourra tricher au jeu à l’aide de codes de triches. Ensuite, la configuration des touches pourra également être modifiée par le joueur à sa guise. Et la vitesse des voitures sur les routes sera aléatoire en fonctions des différentes lignes de route dans le jeu

Détails techniques :

* Un nouveau menu sera ajouté à l’écran de démarrage afin de pouvoir modifier les touches
* Les codes de triches seront cachés, le seul endroit où ils seront visibles sera la documentation technique du jeu

# Conception du projet

## Diagrammes de classes

Nous avons fait un diagramme de classes pour poser une base qui nous servira dans le codage de notre jeu.

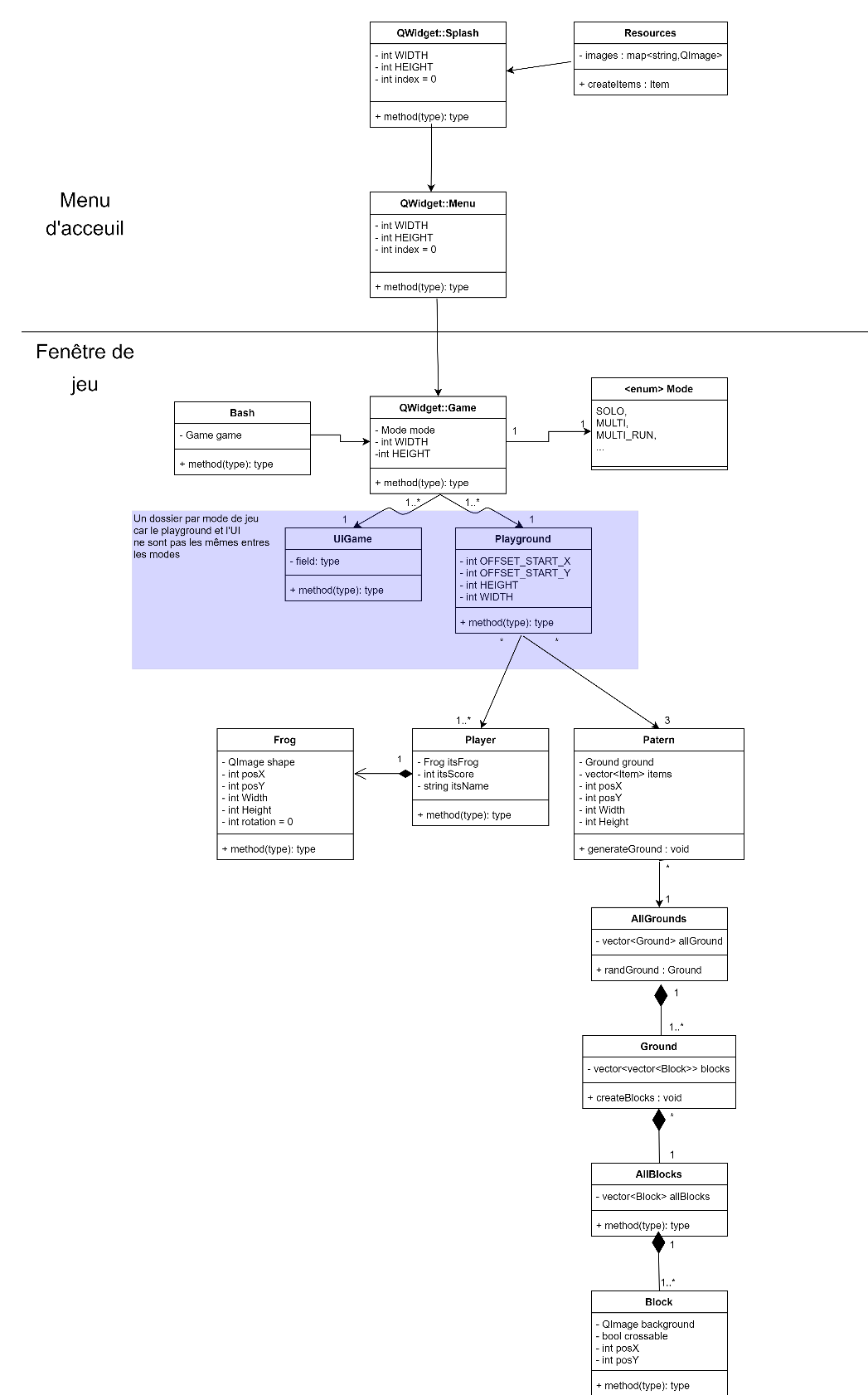


Figure Diagramme de classe conçu avant la phase de programmation

Ensuite, après la phase de programmation, nous avons recréé un diagramme de classes à l’aide de l’outil de rétro-ingénierie de BOUML afin d’observer les différentes modifications qui ont été opérées.

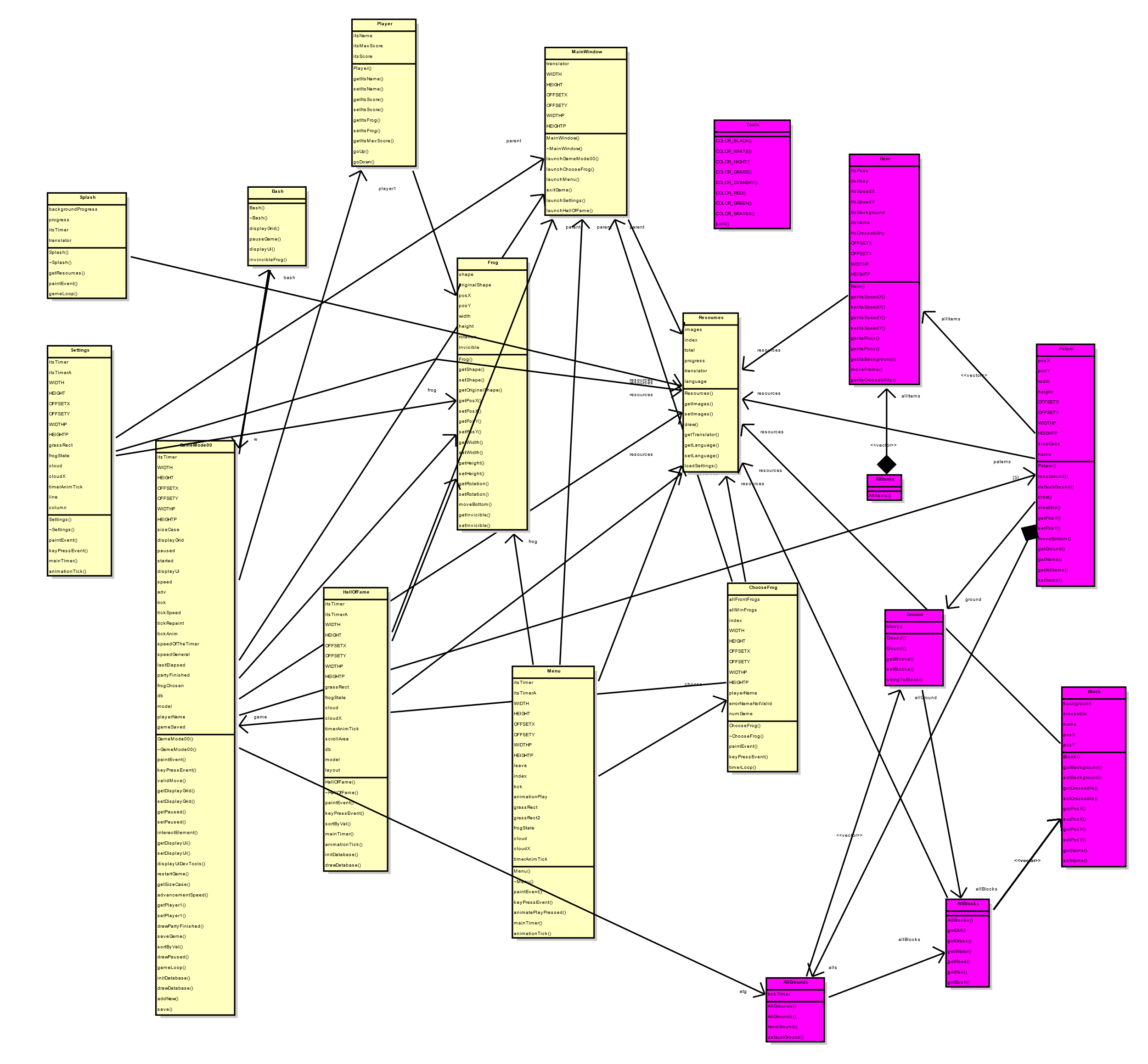


Figure Diagramme de classes issu de la rétro-ingénierie du code du jeu

## Diagrammes de cas d’utilisation

Nous avons réalisé deux diagrammes de cas d’utilisation, un pour chaque mode de jeu prévu : le premier correspond au cas du mode de jeu « solo endless » tandis que le deuxième représente le cas du mode « 1 vs 1 ».

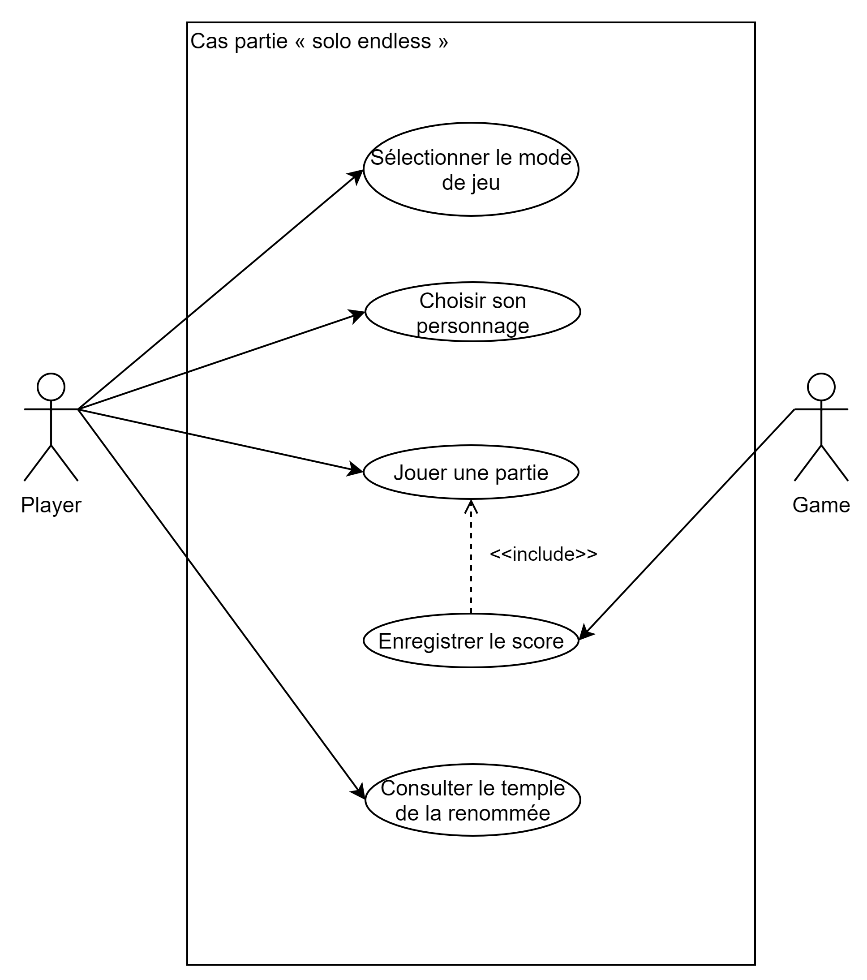


Figure Diagramme de cas d'utilisation pour le cas partie "solo endless"

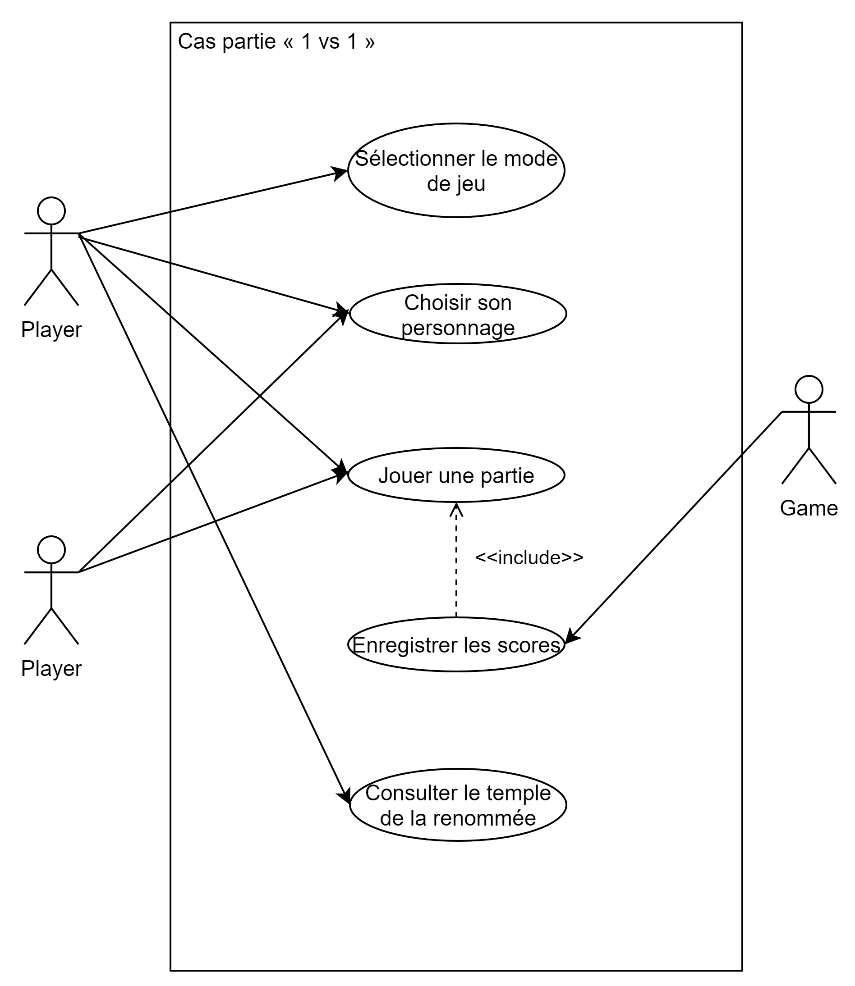


Figure Diagramme de cas d'utilisation pour le cas partie "1 vs 1"

## Diagrammes de séquence

Parmi les diagrammes de séquences que nous avons réalisés, comptent celui du déplacement de la grenouille et celui de la sélection dans le menu principal du jeu.

Une image contenant texte, capture d’écran, moniteur

Description générée automatiquement

Figure Diagramme de séquence pour le déplacement de la grenouille

Une image contenant texte

Description générée automatiquement

Figure Diagramme de séquence pour la sélection dans le menu principal

# Gestion de projet

1. Mode de jeu où le jeu joueur avance sans fin sur un monde généré de manière aléatoire. [↑](#footnote-ref-2)
2. Différents types de terrain : (foret, désert, etc.) [↑](#footnote-ref-3)