RAPPORT DE PROJET

Louis HUORT  
Grégoire DOLIDON  
Mathis GUESSARD  
Hakim IZM

Table des matières

[I. Notre jeu 3](#_Toc74834750)

[A. Le sujet du projet 3](#_Toc74834751)

[B. CrossyFrog 3](#_Toc74834752)

[C. Règles du jeu 3](#_Toc74834753)

[D. Vision produit 4](#_Toc74834754)

[II. Périmètre du projet 4](#_Toc74834755)

[A. Cahier des charges de la V1 4](#_Toc74834756)

[1. Détails techniques du jeu : 4](#_Toc74834757)

[2. Mode de jeu : 4](#_Toc74834758)

[3. Choisir les personnages : 4](#_Toc74834759)

[4. Écran de chargement : 5](#_Toc74834760)

[5. Menu du jeu : 5](#_Toc74834761)

[6. Pattern : 5](#_Toc74834762)

[7. Blocs : 5](#_Toc74834763)

[8. Grenouille : 6](#_Toc74834764)

[9. Scores : 6](#_Toc74834765)

[10. Graphismes : 6](#_Toc74834766)

[B. Cahier des charges de la V2 6](#_Toc74834767)

[1. Mode de jeu : 6](#_Toc74834768)

[2. Effets sonores 7](#_Toc74834769)

[3. Graphismes 7](#_Toc74834770)

[4. Expérience utilisateur 7](#_Toc74834771)

[III. Conception du projet 9](#_Toc74834772)

[A. Diagrammes de classes 9](#_Toc74834773)

[B. Diagrammes de cas d’utilisation 10](#_Toc74834774)

[C. Diagrammes de séquence 12](#_Toc74834775)

[IV. Gestion de projet 14](#_Toc74834776)

[A. Notre groupe 14](#_Toc74834777)

[B. Outils utilisés 14](#_Toc74834778)

[C. Calendrier prévisionnel 15](#_Toc74834779)

[D. Rétrospectives des différentes séances 15](#_Toc74834780)

[1. Rétrospective des semaines 20 et 21 15](#_Toc74834781)

[2. Rétrospective de la semaine 22 16](#_Toc74834782)

[3. Rétrospective de la semaine 23 16](#_Toc74834783)

[V. Réalisation 17](#_Toc74834784)

[A. Programmation 17](#_Toc74834785)

[1. Fonctionnalités de la V1 17](#_Toc74834786)

[2. Fonctionnalités de la V2 17](#_Toc74834787)

[3. Problèmes rencontrés 17](#_Toc74834788)

[B. Graphismes 17](#_Toc74834789)

[1. Dessin des textures 17](#_Toc74834790)

[2. Problèmes rencontrés 18](#_Toc74834791)

[C. Maintenance du jeu 18](#_Toc74834792)

[1. Modification des textures 18](#_Toc74834793)

[2. Création et modifications des patterns 18](#_Toc74834794)

# 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 ».

## Vision produit

Le jeu sera simple à utiliser pour que n’importe qui puisse y jouer. Le but du projet étant de faire un jeu d’arcade, nous avons donc choisi de ne pas utiliser la souris et de tout contrôler avec le clavier. Les parties seront simples au début puis plus le joueur gagne des points, plus le jeu devient rapide et donc difficile. Le jeu sera néanmoins compétitif ; le joueur aura le choix d’enregistrer son score et de pouvoir le comparer aux autres. Nous voulons que le jeu soit personnalisable et que le joueur puisse facilement modifier les textures et personnaliser le terrain de jeu en créant de nouveau patterns.

# 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.

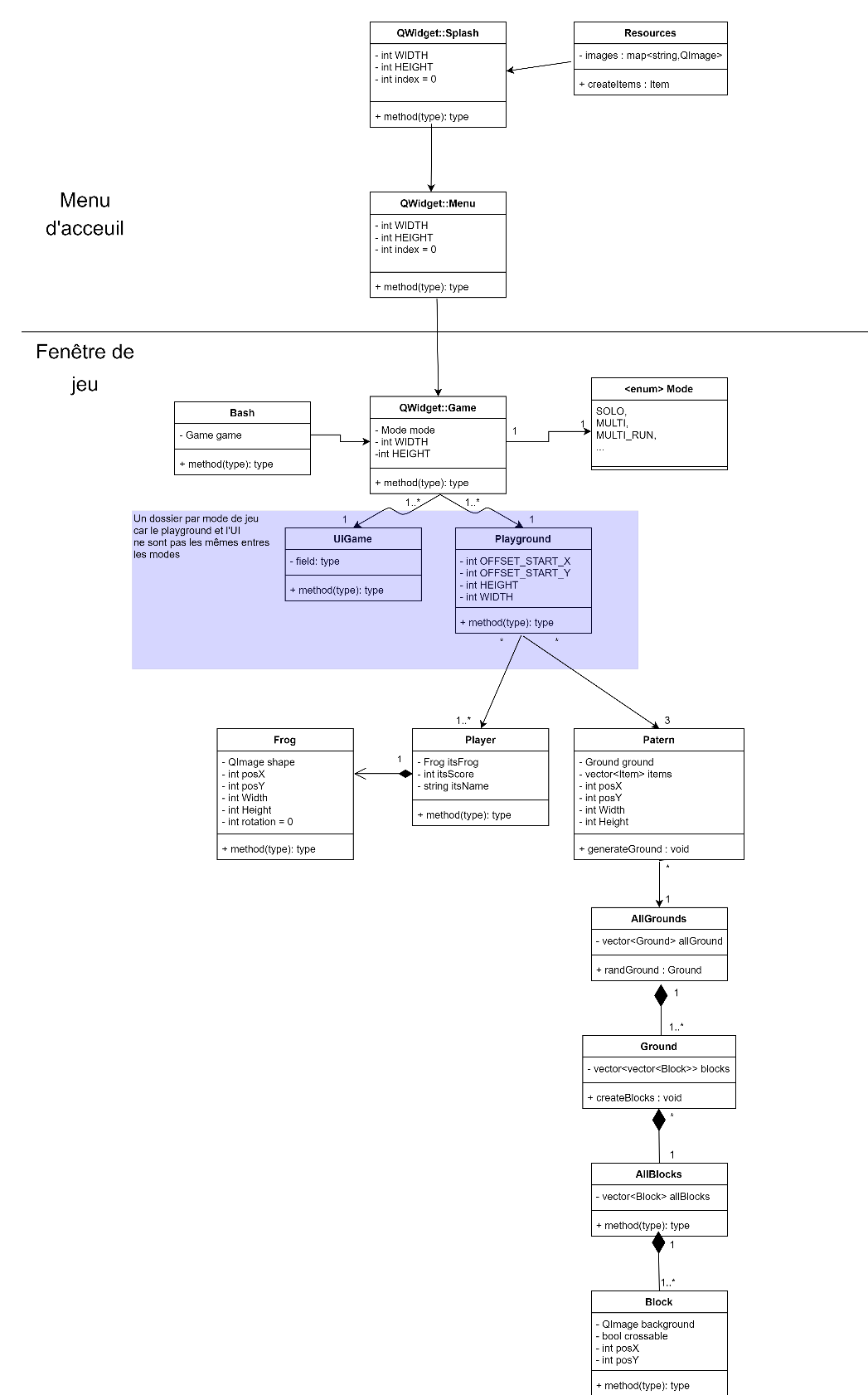


Figure Diagramme de classe conçu avant la phase de programmation

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.

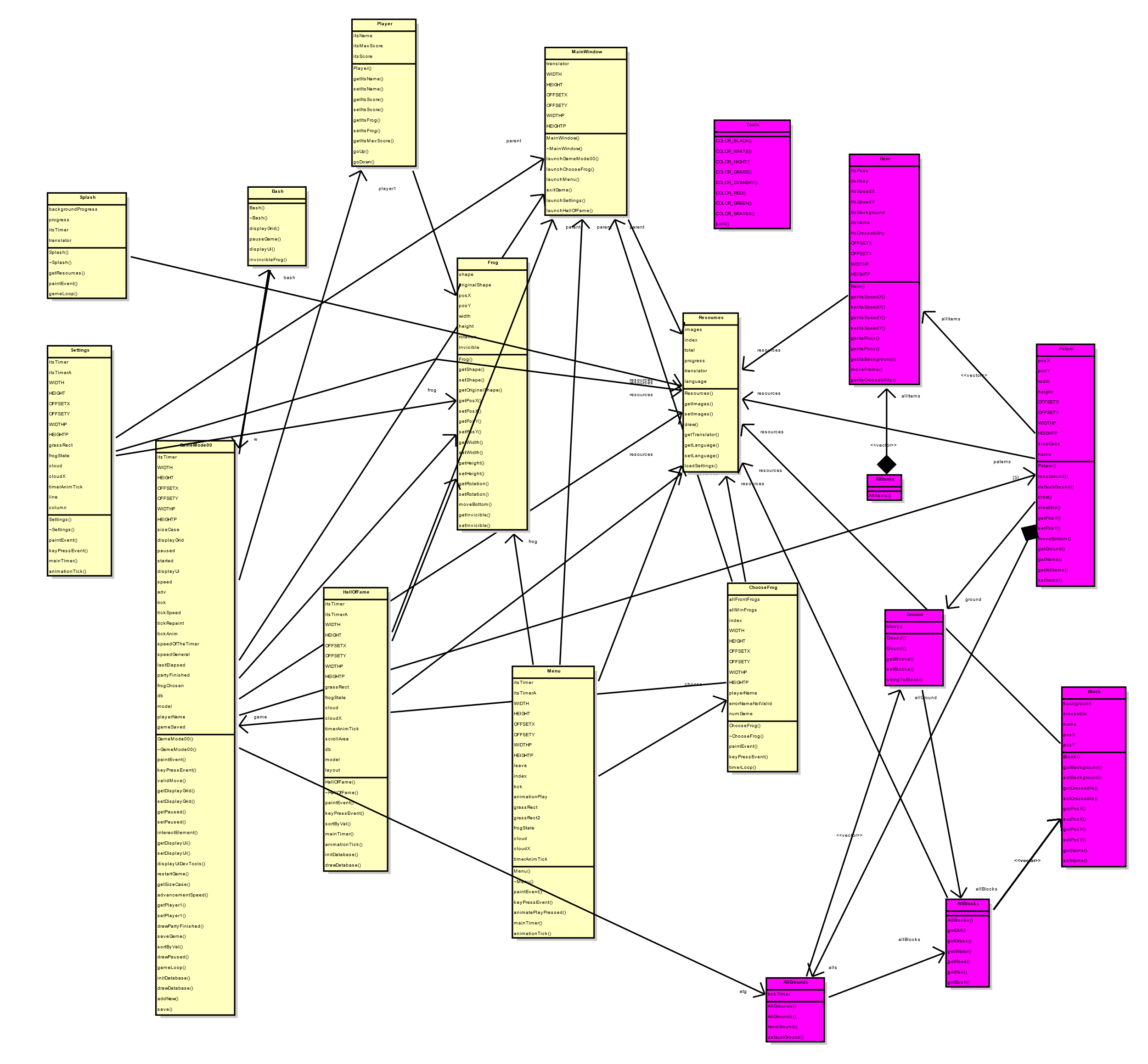


Figure Diagramme de classes issu de la rétro-ingénierie du code du jeu

## 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 ».

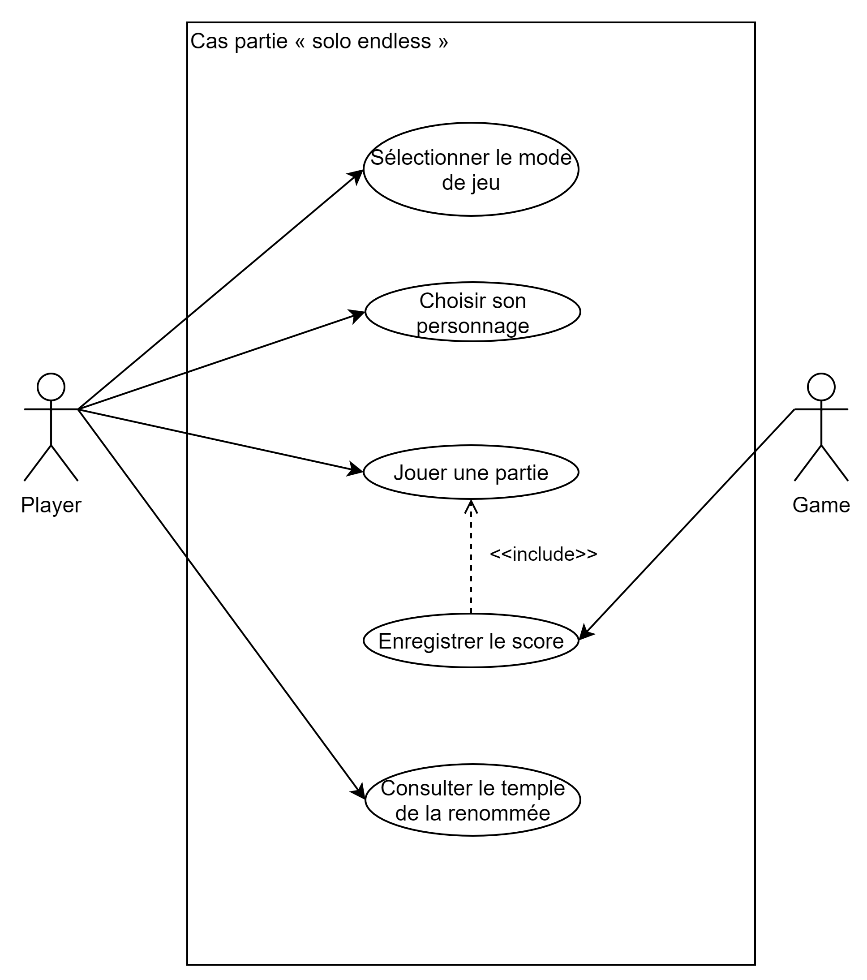


Figure Diagramme de cas d'utilisation pour le cas partie "solo endless"

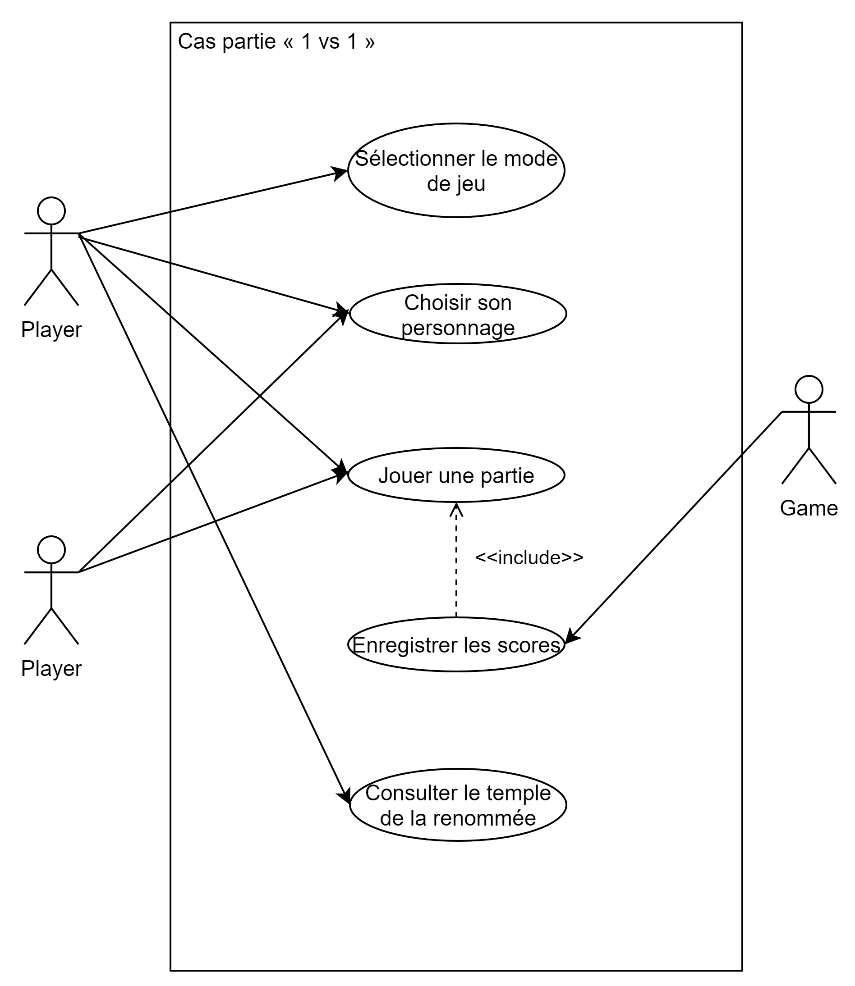


Figure Diagramme de cas d'utilisation pour le cas partie "1 vs 1"

## 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

## Notre groupe

Nous avons nommé notre groupe « Satrari » en référence au studio de développement de jeux-vidéo Atari qui était très actif à l’époque des jeux d’arcade en produisant des jeux régulièrement et en concevant leurs propres consoles de salon entre les années 70 et 90.

Les rôles dans notre groupe ont été répartis de la manière suivante :

* Louis HUORT (chef de projet, développeur) ;
* Grégoire DOLIDON (graphiste) ;
* Mathis GUESSARD (chargé de production) ;
* Hakim IZM (développeur, chargé de production) ;

## Outils utilisés

Pour mener à bien la conception et la production de CrossyFrog, nous avons utilisé différents outils tels que :

* Figma : une plateforme permettant de créer des maquettes visuelles, celle-ci nous a permis de trouver des idées pour avoir l’IHM la plus user-friendly ;
* Draw.io : un site permettant dessiner des diagrammes en tous genres, nous l’avons utilisé pour faire les diagrammes de classes, de cas d’utilisation et de séquence avant le développement du jeu ;
* BOUML : un logiciel de création de diagrammes UML, qui nous a été utile à la fin du développement du jeu pour la génération du diagramme de classes final à titre de comparaison grâce aux fonctionnalités de rétro-ingénierie du logiciel ;
* Krita : un outil d’édition graphique, qui nous a servi pour la création de nos textures ;
* Qt Creator : un IDE, nous ayant servi pour la programmation du jeu en C++.

Puisque nous travaillons collaborativement sur le même projet, nous devons partager des fichiers rapidement et simplement. Pour cela, nous avons utilisé :

* GitLab : notamment la Forge de l’IUT, qui nous sert à partager les fichiers relatifs à la conception et à la programmation ;
* Discord : un logiciel de messagerie instantanée, qui nous est utile lorsqu’il s’agit de communiquer entre nous en dehors des heures consacrées au projet et quand on doit partager des fichiers dans l’immédiat ;
* Onedrive : est un service de stockage sur le cloud qui nous permet de travailler sur des fichiers au format Word ou Powerpoint relatifs à la gestion de projet (cahier des charges, diaporama, etc.) en même temps.

Les outils nous permettant de nous organiser sont les suivants :

* Miro : un outil de travail collaboratif permettant la recherche d’idées, le brainstorming à plusieurs et l’organisation des tâches en catégories ;
* Gantt Project : un outil de création et de gestion d’un calendrier prévisionnel pour les tâches afférentes au produit.

## Calendrier prévisionnel

Nous avons utilisé Gantt Project pour la création de notre calendrier prévisionnel au début de notre projet.

La première étape concernait les débuts du projet où nous prévoyions la mise en place des outils de travail et le choix du jeu (entre le 17 et le 19 mai).

Ensuite, l’étape de réflexion se divisait en trois tâches majeures : la rédaction du cahier des charges (entre le 19 mai et le 2 juin), la réflexion sur les graphismes (entre le 21 mai et le 2 juin) et l’élaboration des diagrammes de conception (entre le 2 et le 9 juin).

Puis vient l’étape de production qui consiste en quatre grands axes : la création des textures (entre le 2 et le 10 juin), la programmation d’une V1 jouable (entre le 10 et le 14 juin), la programmation d’une v2 jouable (entre le 14 et le 18 juin) et la rédaction de la documentation destinée à l’utilisateur (entre le 16 et le 18 juin).

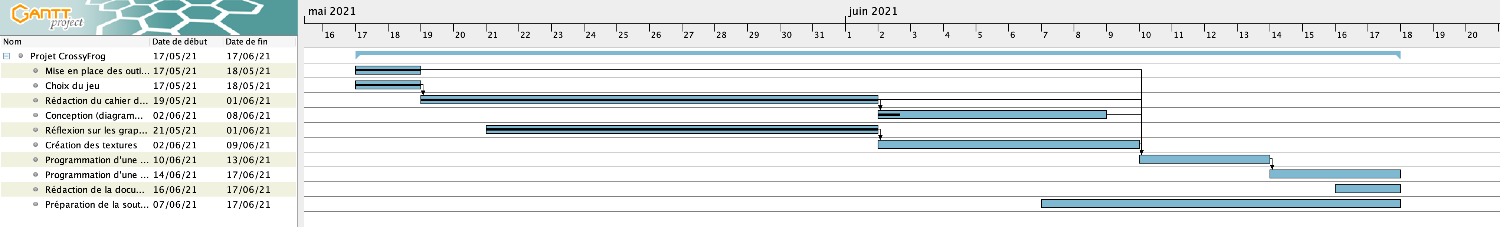


Figure Capture d'écran du calendrier prévisionnel (GanttProject)

## Rétrospectives des différentes séances

### Rétrospective des semaines 20 et 21

Lors de la semaine 20, nous étions au tout début de notre projet : nous avons tout d’abord mis en place les outils qui nous permettront de suivre notre avancée et de réunir les meilleures conditions pour permettre ce travail collaboratif (cf. cahier des charges).

La semaine d’après, nous avons débuté l’élaboration du cahier des charges pour la première version du jeu, nous avons également entamé la réflexion sur la vision produit et commencé la création de maquettes des interfaces IHM du jeu sur Figma. Nous avons créé un calendrier de projet sur Gantt et le débuté le codage dans le but d’évaluer les potentielles difficultés. Ainsi que fait les premiers dessins du jeu : les personnages utilisé lors de la sélection.

Pour la semaine 22, notre objectif est d’évaluer la durée nécessaire pour programmer le jeu, pour cela nous chercherons à terminer les maquettes et d’écrire les premières lignes de code pour nous permettre d’évaluer cette durée. Pour cela notre objectif est le codage du terrain de jeu.

### Rétrospective de la semaine 22

En cette semaine de cours, notre travail s’est déroulé pendant les horaires dédiés, nous avons avancé sur le code et les premières fonctionnalités majeures du jeu : c’est à dire la page du menu, et le déroulement du jeu.

Seuls les boutons “Play” et “Exit” sont fonctionnels. Le jeu se joue de manière infinie grâce à l’implémentation de la fonctionnalité du défilement des patterns. Ceux-ci sont sélectionnés de manière aléatoire parmi quatre patterns existants. Ces patterns sont dessinés à l’avance et situés dans un fichier texte à part.

Les textures des grenouilles affichées pendant la partie et celles présentes sur l’écran d’accueil ont été dessinées et les maquettes sur Figma ont été également finies.

En parallèle, nous avons continué l’amélioration du cahier des charges puis la finalisation de la vision produit. Nous avons au même moment décidé de rajouter One Drive comme outil de ma

Ainsi les objectifs que nous nous étions donnés la semaine précédente ont été atteints et nous avons pu même avancer au-delà de nos espérances.

Nos objectifs pour la semaine 23 sont les suivants : la création d’autres patterns (au moins une dizaine) pour diversifier les décors, implémenter la fonctionnalité d’accélération croissante de la vitesse de jeu ainsi que la création des obstacles.

### Rétrospective de la semaine 23

Cette semaine, nous nous sommes consacrés à l’avancement technique du jeu afin de répondre aux exigences du cahier des charges dans le temps imparti. Les fonctionnalités implémentées ou améliorées sont les suivantes :

* L'accélération de la vitesse de jeu jusqu’à une vitesse maximale ;
* L’apparition et le défilement des plateformes et des obstacles (tels que les rondins de bois sur l’eau et les voitures sur les routes) ;
* Le déplacement de la grenouille ;
* Remise à zéro lors de la mort du joueur à la suite d’un contact avec un obstacle ou l’eau.

En dehors de la création de nouveau pattern. Tous les objectifs que nous nous sommes donnés pour cette séance ont été atteints avec même des suppléments.

Pour la semaine 24, nous souhaiterions avancer sur la V2 du jeu. Les fonctionnalités voulues pour cette V2 sont :

* Les musiques et les sons ;
* Le hall of fame (tableau des meilleurs scores) ;
* La traduction (anglais/français) ;
* Les paramètres (sélection : sons, langue, etc.) ;
* L’écran de game over (score + possibilité d’enregistrer le score et de relancer une partie) ;
* Un obstacle supplémentaire (train) ;
* Grenouille entrainée par les rondins de bois ;
* Plus de variété dans les patterns ;
* Règles.

Pour l’ensemble du projet il nous reste à rédiger rapport et soutenance ainsi que rassembler les différents livrables (doc).

# Réalisation

## Programmation

### Fonctionnalités de la V1

La V1 du jeu est une version fonctionnelle ; en effet le jeu dispose d’un menu quasiment fonctionnel, il lui manque les paramètres et le classement qui viendront en V2. Une fois le jeu lancé (bouton play), nous ne pouvons pas choisir notre grenouille ou encore écrire un pseudo personnalisé. Le jeu, quant à lui, fonctionne correctement bien qu’il manque encore les trains n’étant pas encore implémentés au mode de jeu. Malgré cela la difficulté est déjà assez élevée.

La texture de la grenouille a été trouvée sur Internet, tout comme les blocs qui sont tirés du fameux jeu « Minecraft ». Toutes ces textures sont dites des « placeholders » puisqu’en effet ces textures sont remplacées au fur et à mesure de l’avancement des graphismes.

À la fin de la partie, le joueur peut sauvegarder son score dans le tableau des scores qui est une base de données. Celle-ci va trier tous les scores dans l’ordre décroissant pour n’afficher que les meilleurs. Le score est enregistré dans la base de données mais le pseudo du joueur reste le même puisque comme dit précédemment, la fonctionnalité de mettre un nom personnalisé n’est pas encore possible. En jeu la grenouille peut se déplacer dans les 4 directions disponibles et l’image de la grenouille se tourne dans le sens où la grenouille se dirige.

### Fonctionnalités de la V2

La V2 du jeu comporte tous les éléments du jeu attendus. Les graphisme sont complètement crée par notre équipe. Les différents éléments du menu comme les paramètres ou encore le classement sont terminés. Le classement affiche les 5 meilleurs score de Crossy Frog. Dans les paramètres nous pouvons choisir la langue du jeu. En effet tous les textes présent dans le jeu sont traduisible, les langues disponibles sont anglais et français. Depuis les paramètres nous pouvons aussi accéder aux règles ou encore aux crédits. Les différents mode de jeux sont grisés car ils ne sont pas encore implémenté dans le jeu, il faut voir à comme un truc dans le turfu frero, la sang de la veine. Nous pouvons désormais

### Problèmes rencontrés

## Graphismes

### Dessin des textures

L’intégralité des textures ont été dessinées avec Krita. Toutes les images utilisées pour le jeu sont des images conçues pour l’occasion, de ce fait nous avons les droits pour toutes les textures. La police d’écriture est « 8-bit arcade », crée par Damien GOSSET,c’est une police libre de droit pour une utilisation personnel et commerciale . La police d’écriture est « 8-bit arcade », crée par Damien GOSSET, c’est une police libre de droit pour une utilisation personnelle et commerciale.

Les textures des blocks et des personnages en jeu sont en 32x32px. Nous avons décidé de rester dans le visuel des jeux d’arcade retro en choisissant une résolution basse. De plus ce choix permet aux jeux d’être plus fluide car les texture de basse résolutions demande moins de ressources graphiques.

Le plus grand problème rencontré lors de la création des graphismes a été la cohérence entre chaque texture pour que l’ensemble soit cohérent pour cela plusieurs textures que chaque éléments ont été dessiner puis afficher aléatoirement avec une rotation aléatoire

### Problèmes rencontrés

## Maintenance du jeu

### Modification des textures

Il vous est possible de personnaliser les textures du jeu. Toutes les textures du jeu sont disponibles suivant ce chemin : "\crossy-frog\QT\CrossyFrog\res".

Les textures du jeu sont en 32x32 px, mais cette taille n'est pas fixe ; vous pouvez modifier les fichiers. Il faut cependant faire attention à la resolution : plus elle est élevée, plus le jeu nécessitera de resources.

Les textures sont toutes en format ".PNG", les fichiers sources ".KRA" (fichiers sources de Krita) sont disponibles pour pouvoir être modifiés.

### Création et modifications des patterns

L'ajout de pattern (ou motifs) permet de personnaliser le jeu pour créer des cartes uniques.

La création et la suppression de patterns s'effectue dans les fichiers au format ".txt" situés dans le répertoire suivant : "crossy-frog\QT\CrossyFrog\res\ground".

Ceux-ci comportent une matrice de 17x7 caractères séparés par des espaces. Chaque caractère correspond à une case en jeu.

Les correspondances sont les suivantes :

* G : Grass
* D : Dirt
* W : Water
* T : Train
* B : Bush
* R : Road

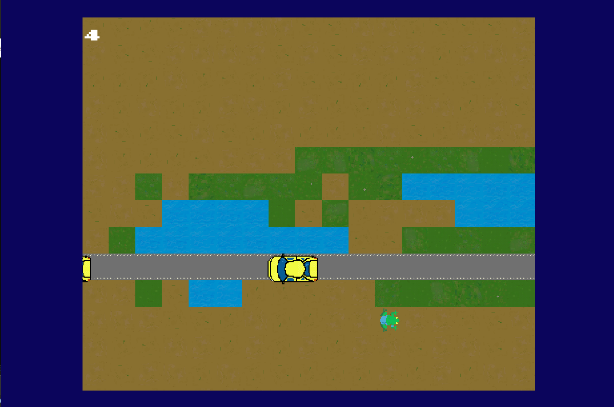
Les voitures apparaissent automatiquement si une ligne entière de route est posée, en revanche si la ligne n'est pas complète, la route sera vide.

Le même fonctionnement est appliqué à l'eau pour la création des rondins de bois, et les rails pour les trains.

Des patterns vides sont mis à disposition dans le fichier pour simplifier la création.

Une fois que le fichier contenant le nouveau pattern est créé, il faut l'ajouter au jeu. Pour cela, il suffit d'ajouter une ligne de code dans le fichier "crossy-frog\QT\CrossyFrog\tools\allgrounds.cpp" à l'emplacement indiqué par les commentaires dans le fichier. Le fichier sera alors lu au lancement du jeu et le pattern sera créé.

Il est important de faire attention à la difficulté des patterns créés ; il est possible de créer des patterns qu'on ne peut pas traverser, ce qui rend le jeu injouable.

Une image contenant texte

Description générée automatiquement

Figure 8 Exemple d'un fichier décrivant un pattern

Figure 9 Affichage en jeu du pattern décrit en figure 8

# Bilan

## Projet

Vue trop grand

Le projet

## Conditions de travail

Manque de temps / difficile de programmer un jeu en une semmaine + s’entrener pour un oral + réviser les controles

Un manque de connaisance / d’aide

## Nos compétences

Nous azvont bien travaille run equipe

Il est plus simple de travailler en presentile par raport au distantiel du pts1

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)