# Abstract

James Gouin et la banane sacrée est un jeu de réflexion en 2D, le joueur doit diriger un pingouin à travers des énigmes dans un univers au thème glacial. La résolution d’une énigme permet de passer à l’énigme suivante. Le but du joueur est de terminer tout les niveaux en résolvant les énigmes les unes après les autres pour finalement récupérer l’objet ultime et résoudre le mystère derrière celui-ci.

Le déroulement du projet est décrit en détail dans ce document, les choix qui ont été fait et les différentes possibilités que nous avions sont commentées.

# Introduction

James Gouin et la Banane Sacrée est jeu d’infiltration en 2D sans combats sous forme de puzzle game. Le joueur manipule un pingouin agent-secret envoyé sur un iceberg pour récupérer la « banane sacrée » volée par un singe. Son personnage peut marcher et glisser dans les quatre directions. Il fera face à un niveau de tutoriel, 6 niveaux normaux et un niveau final.

Il a pour objectif de récupérer des blocs de glace auprès du boss normal de chaque niveau afin de se créer un passage jusqu’à l’iceberg central abritant l’igloo du singe. Pour cela, le joueur devra résoudre des problèmes logiques en se frayant un chemin tout en évitant d’entrer dans le champ de vision des ennemis.

## La longue histoire de James Gouin

James Gouin est un Jeune Pingouin avec beaucoup de potentiel. Il fut recruté à la fin de ses études de poissonnerie par une organisation secrète capitaliste. Ces compétences, hors du commun, en esquive de queues de poissons enragés le propulsèrent au titre d’espion pour les projets spéciaux. Il est cependant un grand craintif de l’eau, il est impossible pour lui de mouiller son nœud papillon qui lui fût donné par son maitre Poissonnier.

Un jour, alors que notre héro pêchait, il reçu par une fusée glaçon un nouvel ordre de mission…

Contenu du message glacé :

**Ordre de mission pour James Gouin**

Docteur Douceur, le méchant Singe, a volé la banane Sacrée ! Il est de ton devoir de la récupérer. Pour cela, nous t’envoyons en mission sur le très dangereux … Iceberg glacé, la base du monstrueux singe Docteur Douceur! Tu as deux jours avant qu’il n’utilise la banane sacrée pour faire un délicieux banana split. Empêche-le ! Dans 1min, ce message glacé s’auto-fondera. Nous nierons tout lien avec cette mission. Vous êtes maintenant seul. Bonne chance.

Le pingouin décide d’accepter la mission et part aussitôt pour rejoindre le QG de l’organisation. Il lui est mis à disposition une Banana Rocket pour rejoindre l’antre du Docteur Douceur. Une fois le voyage terminé, il atterrit à quelques mètres de l’entrée de la forteresse de l’iceberg glacé. Alors que le pingouin tente d’entrer sur l’iceberg, il se voit bloquer par une grosse loutre et un portail. Celle-ci lui explique qu’elle est la gardienne du portail et est chargée de la garder fermée à tout envahisseur. Cependant, elle trouve qu’elle n’est pas assez bien payée pour faire ce travail. La loutre propose de fermer les yeux et laisser le pingouin passer si celui-ci lui donne du poisson. En effet, vu son poids, elle ne peut plus aller pêcher.

C’est ainsi que le joueur commence son aventure à la quête de la banane sacrée.

# Conclusion

Notre cahier des charges fut assez gourmant en fonctionnalités. Mais au final, nous avons obtenu un jeu fonctionnel, certaines fonctionnalités n’ont pas été implémentées par manque de temps et certains mécanismes du jeu ont évolués lors du développement vers des solutions plus mûres et réfléchies.

Les tâches indispensables pour le fonctionnement du jeu ont été priorisées, les mécanismes les plus importants du jeu ont été implémentés et sont fonctionnels. Nous avons aujourd’hui un jeu jouable pour les premiers niveaux. Il sera également facile par la suite de générer et ajouter de nouveaux niveaux grâce à notre population automatique des cartes de jeu.

De nombreux problèmes de compatibilité inter-plateformes sont apparus durant le développement, mais des solutions ont été trouvées et au final l’application est compatible Mac et Windows. On peut avancer que la diversité des OS lors du développement a été profitable et a obligé de se poser des questions de compatibilités plus avancées que lors d’un développement sur une seule plateforme.

Au fil de l’avancement dans le projet de nouvelles et meilleures solutions ont été trouvées aux fonctionnalités précédentes, cependant la plupart des anciennes solutions déjà implémentées n’ont pas pu être implémentées par manque de temps. On peut donc remarquer que lors du développement du projet, les solutions apportées sont de plus en plus propres et mûres.

Par exemple, l’héritage de la classe « Player » dans la classe «P\_Penguin » n’était pas indispensable, les méthodes et les éléments (QGraphicsRectItem) de « P\_Penguin » auraient pu être implémentées dans « Player » comme nous l’avons fait plus tard avec « Ennemi ».

Une solution moins gourmande en mémoire et CPU aurait également pu être trouvée pour les champs de détection des ennemis.

L’espoir de pouvoir continuer ce projet anime l’équipe de Banana Rocket, implémenter les fonctionnalités qui ont été mises en suspend et finir les niveaux manquants.

# Conception

## Planning et suivi

La forge Bitbucket avec la technologie GIT a été utilisée. Nos commits sont datés et considérés comme le journal de bord le l’équipe, ils sont commentés et contiennent les modifications et ajouts.

La méthodologie de développement en cascade s’est imposée au début du projet, cependant il s’est avérée que nous travaillions plus avec une méthodologie agile sans pour autant le faire exprès (avant de prendre connaissance de cette méthode lors du cours de Génie Logiciel).

La structure de notre projet était en effet composée d’un « Product Backlog » avec toutes les fonctionnalités du cahier des charges. D’un « Sprint Backlog » qu’on se donnait tous lundi pendant les heures du projet. Les « sprints » en eux même qui duraient une semaine, du lundi au lundi suivant. Et chaque début de semaine on avait une itération du jeu avec de nouvelles fonctionnalités.

La répartition du travail a aussi été adaptée à chaque sprint, il est rare qu’une personne ait été la seule à travailler sur une classe, les cas où ça s’est produit ont été par manque de temps.

Au final le travail selon les classes a été reparti ainsi :

|  |  |
| --- | --- |
| **Nom** | **Classes / Autres** |
| Margaux | Gameboard, S\_ViewTransition, Profil, M\_MenuStart, Object, Level, W\_Object, S\_Dialog, W\_Dialog, W\_Life, ItemPopulation, Enigmes, Design |
| Romain | Player, Surface, Gameboard, S\_ViewTransition, P\_Penguin, M\_MenuStart, M\_Pause, Level, MainGame, Images, ItemsPopulation, Icons, Licenses, Maps, Enigmes, Deploy, Installer |
| Steve | Surface, Gameboard, B\_Movable, B\_Wall, S\_Snow, P\_Penguin, B\_Water, Ennemi, S\_Ice, E\_Renard, E\_Loup, Enigmes, Design |

Le travaille a été reparti équitablement, tout le monde à pu toucher à tout, et l’expérience acquise lors de ce projet a été profitable à toute l’équipe, ce qui est le plus important.

Pour voire la répartition des tâches selon les fonctionnalités et non selon les classes, se référer au digramme de Gantt joint en annexe.