Mono Kong

Rapport TPI et Documentation technique

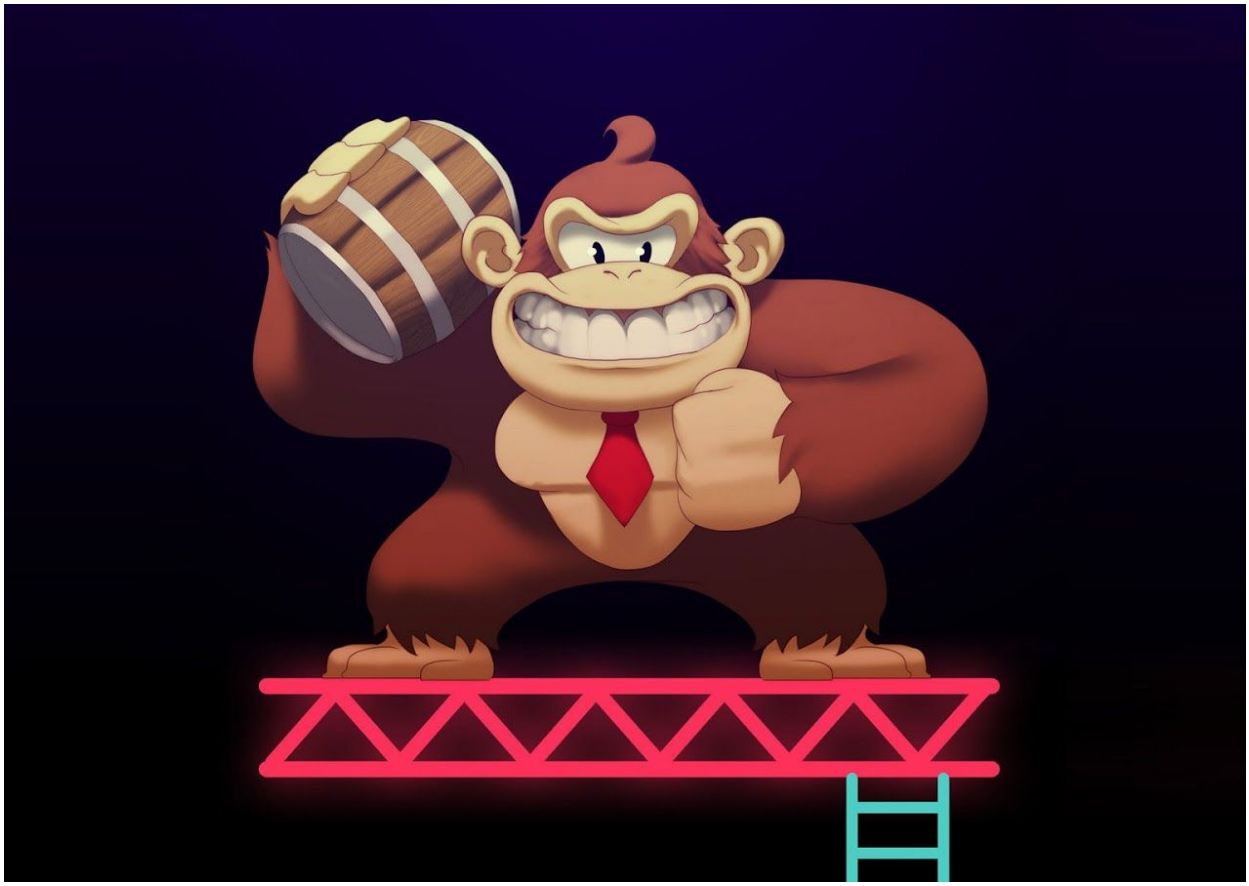

Fig. 0 Donkey kong

source: https://wallpaperaccess.com/donkey-kong

Travail pratique individuel (TPI)

Gama Tiago - 09 Juin 2020
Maître d'apprentissage : Stéphane Garchery

Table des versions

No. de version	Date	Auteur	Changements apportés
V1.0	09.06.2020	Gama Tiago tiago.gmfrr@eduge.ch	Version finale du document pour le rendu du TPI

Introduction

Ce document est un rapport des différents aspects de la conception du projet., le projet étant Donkey Kong en monogame, réalisé dans le cadre du TPI (Travail Pratique Individuel) durant la session de mai 2020, en télé-travail. Le but de ce travail est l'évaluation des compétences acquises pendant le CFC d'informaticien à l'école d'informatique du CFPT au Petit-Lancy.

Donkey Kong est un jeu vidéo d'arcade créé par Nintendo en 1981. C'est un des premiers jeux dans le genre plate-formes. Le but principal du jeu est d'esquiver les obstacles pour amener le personnage principal (Mario) à la princesse, qui a été capturé par Donkey Kong.

Rappel du cahier des charges

Les informations suivantes sont extraites du cahier de charges du TPI.

Objectifs

Durant 11 jours de travail, soit 88 heures, le but est de concevoir une réplique du jeu célèbre Donkey Kong en monogame 2D. Le jeu contiendra les animations de Kong et de la princesse. Le joueur pourra contrôler Mario et devra s'esquiver des tonneaux qui descendent le plateau de façon physique, les tonneaux sont lancés par Kong aléatoirement. Le joueur aura 3 vies pour atteindre la princesse, son score se basera sur le temps qu'il a nécessité pour atteindre la princesse.

Spécifications

Le jeu permettra de :

- Accéder à un menu d'accueil
- Entendre les musiques et bruitages liés au jeu
- Regarder les animations liés au jeu
- Gérer les meilleurs scores
- Contrôler le mouvement de Mario
- Avoir un système de vies / tentatives
- Avoir des obstacles en mouvement
- Créer des obstacles (tonneaux) de façon aléatoire
- Calculer le temps pris par le jour pour arriver à la princesse

Environnement

- Un poste de travail personnel avec windows 10 et deux écrans
- Visual studio comme IDE
- Outil de versionnage de code (Github)
- Outils bureautiques à choix pour les documents (Google Docs, MS Office)
- Outils de traitements d'images (Paint, Paint3D, Gimp)
- Outils de traitements de sons (https://online-audio-converter.com/, https://audiotrimmer.com/)
- Outil de création de maquettes (https://www.draw.io/)
- Outil de conversion du code source en pdf (https://www.overleaf.com/)

Organisation

Élève :

- Tiago Gama, **tiago.gmfrr@eduge.ch**

Maître d'apprentissage :

- Stéphane Garchery, **stephane.garchery@edu.ge.ch**

Experts :

- Philippe Bernard, **philippe.bernard@etat.ge.ch**
- Pierre Conrad, **pierre.conrad@etat.ge.ch**

Livrables

Pour la fin du TPI, le 9 juin 2020 :

Pour les experts et le formateur par email :

- Planning détaillé du projet
- Rapport de projet contenant le code source au format PDF
- Journal de bord
- Résumé du TPI (1 page A4)

Pour le formateur uniquement :

- L'accès au repository distant du projet
- Un readme explicitant l'installation du projet en local

Méthodologie

Pour planifier mon projet de TPI, je me suis basé sur la méthode en 6 étapes que j'ai appris durant les différentes modules du CFC.

Fig. 1 - Méthode en 6 étapes

Source: https://www.afci-ju.ch/fichiers/Planification-par-la-mthode-des-6-tapes-15.pdf

S'informer

La première étape pour la réalisation de ce projet a été la lecture de l'énoncé pour prendre connaissance des tâches à effectuer et les fonctionnalités majeures du projet. Ensuite, j'ai pris contacte avec mon formateur pour clarifier et spécifier certains points de l'énoncé pour assurer que j'avais bien compris le travail à réaliser.

Planifier

Dès la fin de la première étape, j'ai préparé un planning pour mieux comprendre les tâches à effectuer et combiens de temps j'avais pour chacune. Pour cela j'ai découpé mon travail sous forme de sous-tâches ou user stories qui sont stockées dans un product backlog. Cette méthode est fortement inspiré de la méthodologie Scrum, mais vu que le travail est individuel, le reste de la méthodologie n'as pas été nécessaire.

Pour mon planning, j'ai utilisé une feuille Excel, j'ai choisi cette méthode car elle était la méthode recommandé par mon formateur.

Décider

Durant l'implémentation des différentes fonctionnalités de mon travail, j'ai dû prendre des décisions sur comment procéder, les choix que je jugeais importants pour le projet sont mentionnés dans mon journal de bord.

Réaliser

Une fois que les décisions ont été prises, j'ai pu implémenter les différentes fonctionnalités du programme.

Contrôler

À la fin de l'implémentation de chaque fonctionnalité, je contrôlais si elle correspondait aux plan de testes que j'ai écris par rapport aux user stories. Le plan de testes mentionne la date où le test à été passé.

En chaque fin de journée j'ai vérifié intégralité de mon projet, pour m'assurer le fonctionnements de toutes les fonctionnalités déjà implémentées. Après l'implémentation de toutes les fonctionnalités, j'ai testé mon projet sur d'autres machines pour m'assurer qu'il n'y avait pas de problèmes de compatibilité.

Évaluer

La dernière étape de la méthode en six étapes est l'évaluation, cette étape a été une étape de réflexion pour avoir une rétrospective sur le déroulement du projet, les problèmes que j'ai rencontré et les solutions utilisés pour ces problèmes. Pour cela, j'ai fait référence à mon journal de bord.

Planification

Product backlog

Liste de fonctionnalités à implémenter dans le produit final.

Nom	**1. Menu d'accueil**
Description (user story)	En tant qu'utilisateur, je peux utiliser le menu d'accueil pour me rendre sur les différents endroits de l'application
Critère d'acceptation	1.1

Nom	**2. Page d'informations supplémentaires**
Description (user story)	En tant qu'utilisateur, je veux que la page d'informations supplémentaires affiche un tutoriel court et simple sur comment jouer le jeu et me permet de retourner sur le menu.
Critère d'acceptation	2.1 et 2.2

Nom	**3. Affichage des composants du jeu**
Description (user story)	En tant qu'utilisateur, je veux pouvoir voir tous objets du jeu, comme le terrain, kong, mario, etc.
Critère d'acceptation	3.1 à 3.6

Nom	**5. Gravité**
Description (user story)	En tant que développeur, je veux que les tous objets en mouvement du jeux subissent une gravité s'ils ne sont pas en collision avec le terrain.
Critère d'acceptation	5.1

Nom	**6. Mario**
Description (user story)	En tant qu'utilisateur, je veux pouvoir contrôler le mouvement de Mario (marcher, sauter ou monter une échelle).
Critère d'acceptation	6.1 à 6.3

Nom	**7. Saut de Mario**
Description (user story)	En tant que développeur, je veux que Mario puisse sauter seulement s'il est par terre.
Critère d'acceptation	7.1

Nom	**8. Echelles**
Description (user story)	En tant que développeur, je veux que la taille des échelles s’adapte au terrain et que Mario puisse seulement les utiliser s’il est au centre de l'échelle.
Critère d’acceptation	8.1 et 8.2

Nom	**9. Kong**
Description (user story)	En tant qu'utilisateur, je veux pouvoir voir Kong lancer un tonneaux.
Critère d’acceptation	9.1

Nom	**10. Tonneaux**
Description (user story)	En tant que développeur, je veux que les tonneaux descendent le terrain de façon physique et qu’ils soient retirés du jeu quand ils ne sont plus nécessaires.
Critère d’acceptation	10.1 et 10.2

Nom	**11. Animations**
Description (user story)	En tant qu'utilisateur, je veux que les objets soient animé.
Critère d'acceptation	11.1 à 11.4

Nom	**12. Collisions**
Description (user story)	En tant que développeur, je veux que les collisions entre les différents objets déclenchent des événements
Critère d'acceptation	12.1 et 12.2

Nom	**13. Système de vies**
Description (user story)	En tant que développeur, je veux que Mario aient trois tentatives pour arriver jusqu'à la Princesse.
Critère d'acceptation	13.1

Nom	**14. Score**
Description (user story)	En tant qu'utilisateur, je veux voir mon score et le comparer avec le highscore.
Critère d'acceptation	14.1 et 14.2

Nom	**15. Musiques et bruitages**
Description (user story)	En tant qu'utilisateur, je veux que les différentes pages de l'application (menu et jeu) aient une musique de fond, et que les objets de jeu aient des bruitages.
Critère d'acceptation	15.1 à 15.4

Planification prévue

Tâches à réaliser	Temps nécessaire	1er jour	2e jour	3e jour	4e jour	5e jour	6e jour	7e jour	8e jour	9e jour	10e jour	11e jour	Total
Lecture de l'énoncé	00:30	0:30											00:30
Planification	03:00	3:00											03:00
													00:00
Analyse													00:00
Imagination de la vue													00:00
Croquis	02:30	00:30	00:30	00:30	00:30	00:30							02:30
Recherche de sprites	02:30	01:00	00:30	00:30	00:30								02:30
Recherche de sons et bruitages	02:00		01:00	00:30	00:30								02:00
Recherche d'exemples de code pour les fonctionnalités	07:00		01:00		01:00	01:00	01:00	01:00	01:00	01:00			07:00
Réalisation													00:00
Menu d'accueil													00:00
Création de la vue	02:45	00:45	01:30	00:30									02:45
Changements d'etats à travers le menu	01:30		00:30	01:00									01:30
Test / Débuggage	01:30			00:30	01:00								01:30
Fenêtre de jeu													00:00
Création du terrain de jeu (briques et échelles)	03:30		01:30	02:00									03:30
Test / Débuggage	02:00			01:00	01:00								02:00
Fenêtre informations													00:00
Création de la vue	01:30				01:30								01:30
Affichage des informations du jeu	01:00				01:00								01:00
Fonctionnalités du jeu													00:00
Déplacement objets	03:30				00:30	02:00	01:00						03:30
Collisions	04:30					02:00	2:00	00:30					04:30
Inputs utilisateur	03:00						01:00	2:00					03:00
Génération alléatoire des obstacles (tonneaux)	03:00						0:30	1:30	01:00				03:00
Animations du jeux	04:00							1:40	02:20				04:00
Evenements de déclanchement de sons et bruitages	03:00								03:00				03:00
Test / Débuggage	07:00					02:00	2:20	1:10	0:30	01:00			07:00
Highscores													00:00
Sauvegarde des scores	02:30									01:30	01:00		02:30
Affichage des highscores	01:30									00:30	01:00		01:30
Test / Débuggage	02:00									00:30	1:30		02:00
													00:00
Documentation													00:00
Doc technique	13:15	01:45	01:00	01:00						03:20	3:30	02:40	13:15
Journal de bord	03:30	00:30	00:30	00:30	00:30	00:30	00:10	00:10	00:10	00:10	00:10	00:10	03:30
Doc utilisateur	03:00										00:50	2:10	03:00
Résumé	03:00											03:00	03:00
	88:00												
		8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00	88:00

Fig.2 Planification prévisionnel

Planification effective

Tâches à réaliser	Temps nécessaire	1er jour	2e jour	3e jour	4e jour	5e jour	6e jour	7e jour	8e jour	9e jour	10e jour	11e jour	Total
Lecture de l'énoncé	00:30	0:30											00:30
Planification	03:00	02:30	00:30										03:00
													00:00
Analyse													00:00
Imagination de la vue													00:00
Croquis	02:30		00:30		00:30	00:30							01:30
Recherche de sprites	02:30	02:00	02:00	01:30									05:30
Recherche de sons et bruitages	02:00		01:00	01:00	00:30			0:30	01:00	01:00			05:00
Recherche d'exemples de code pour les fonctionnalités	07:00		02:00	00:30	01:00		01:00	01:00		01:00			06:30
Réalisation													00:00
Menu d'accueil													00:00
Création de la vue	02:45		01:00	01:00									02:00
Changements d'etats à travers le menu	01:30		00:30	01:00									01:30
Test / Débuggage	01:30		0:20	00:15									00:35
Fenêtre de jeu													00:00
Création du terrain de jeu (briques et échelles)	03:30			00:45	0:50	0:30		1:00	1:00				04:05
Test / Débuggage	02:00				00:30	0:30							01:00
Fenêtre informations													00:00
Création de la vue	01:30				01:00								01:00
Affichage des informations du jeu	01:00				00:30								00:30
Fonctionnalités du jeu													00:00
Déplacement objets	03:30				00:30	01:00	00:30						02:00
Collisions	04:30				01:00	01:30	02:00		1:00	1:00			06:30
Inputs utilisateur	03:00					0:50	00:30						01:20
Génération alléatoire des obstacles (tonneaux)	03:00						0:45		02:00				02:45
Animations du jeux	04:00				00:30	02:00			01:00				03:30
Evenements de déclanchement de sons et bruitages	03:00							1:00	01:00				02:00
Test / Débuggage	07:00				01:00	01:00	03:00	4:00	0:45				09:45
Highscores													00:00
Sauvegarde des scores	02:30									02:00			02:00
Affichage des highscores	01:30									01:00			01:00
Test / Débuggage	02:00									01:00	01:00	01:00	03:00
													00:00
Documentation													00:00
Doc technique	13:15	02:30		01:50						00:45	4:00	04:00	13:05
Journal de bord	03:30	00:30	00:10	00:10	00:10	00:10	00:15	00:30	00:15	00:15			02:25
Doc utilisateur	03:00										3:00		03:00
Résumé	03:00											03:00	03:00
	88:00												
		8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00	88:00

Fig. 3 Planification effectif

Analyse Fonctionnelle

Fonctionnalités

Menu d'accueil

Cette fonctionnalité permettra à l'utilisateur de changer entre les plusieurs fenêtres de l'application

- Menu d'accueil
- Fenêtre de jeu
- Fenêtre informations

Musiques et bruitages

Cette fonctionnalité permettra à l'utilisateur d'écouter la musique et les sons du jeu Donkey Kong. Par exemple Kong et Mario on ses propres bruitages qui seront entendus avec leurs actions.

Animations

Cette fonctionnalité permettra à l'utilisateur de ne simplement regarder des images statiques mais au lieu de voir ses personnages préférés prendre vie. Par exemple Kong se déplacera pour prendre un tonneaux et ensuite le lancer.

Kong

Cette fonctionnalité permettra à l'utilisateur de voir le point d'apparition des tonneaux, car il sera Kong qui les lancera, chaque lancé sera accompagné d'une animation et d'un bruitage.

Mario

Cette fonctionnalité permettra à l'utilisateur d'avoir un personnage qu'il peut contrôler, son objectif est d'arriver à la princesse en esquivant les tonneaux. Il pourra sauter, marcher et monter des échelles.

Tonneaux

Cette fonctionnalité permettra à l'utilisateur d'avoir un obstacle pour difficulter l'arrivée de Mario à la princesse, ils seront lancés par Kong avec un interval aléatoire et ils auront de nouveau une chance aléatoire de descendre les échelles au lieu de suivre le chemin normal.

Score

Cette fonctionnalité permettra à l'utilisateur de s'évaluer, le système de points est basé sur le temps qu'il a pris pour arriver à la princesse.

Gestion des highscores

Cette fonctionnalité permettra à l'utilisateur de comparer ses scores avec ses camarades ou avec lui-même pour le motiver à améliorer son meilleur score.

Interfaces

Menu d'accueil

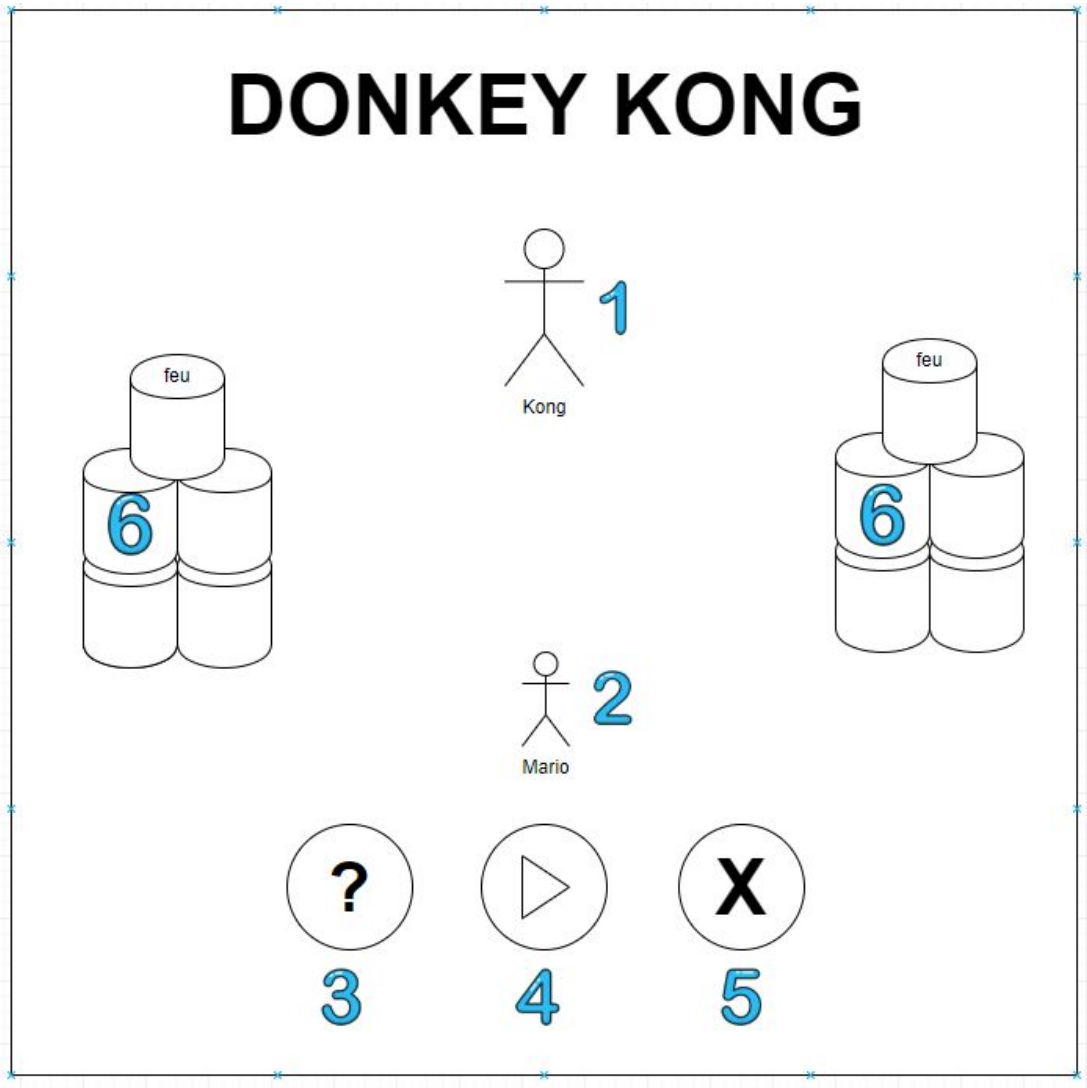

Fig. 4 - Maquette du menu d'accueil

1. Animation de kong pour animer le menu

2. L’utilisateur peut contrôler Mario (Joystick) avec le but de se déplacer vers un des boutons du menu, ensuite s’il est positionné sur un des boutons il pourra l’activer avec un des boutons de la borne

3. Permet à l’utilisateur d'ouvrir la page d’informations

4. Permet à l’utilisateur de commencer le jeu

5. Permet à l’utilisateur de fermer le jeu

6. Animation de tonneaux pour animer le menu

Fenêtre de jeu

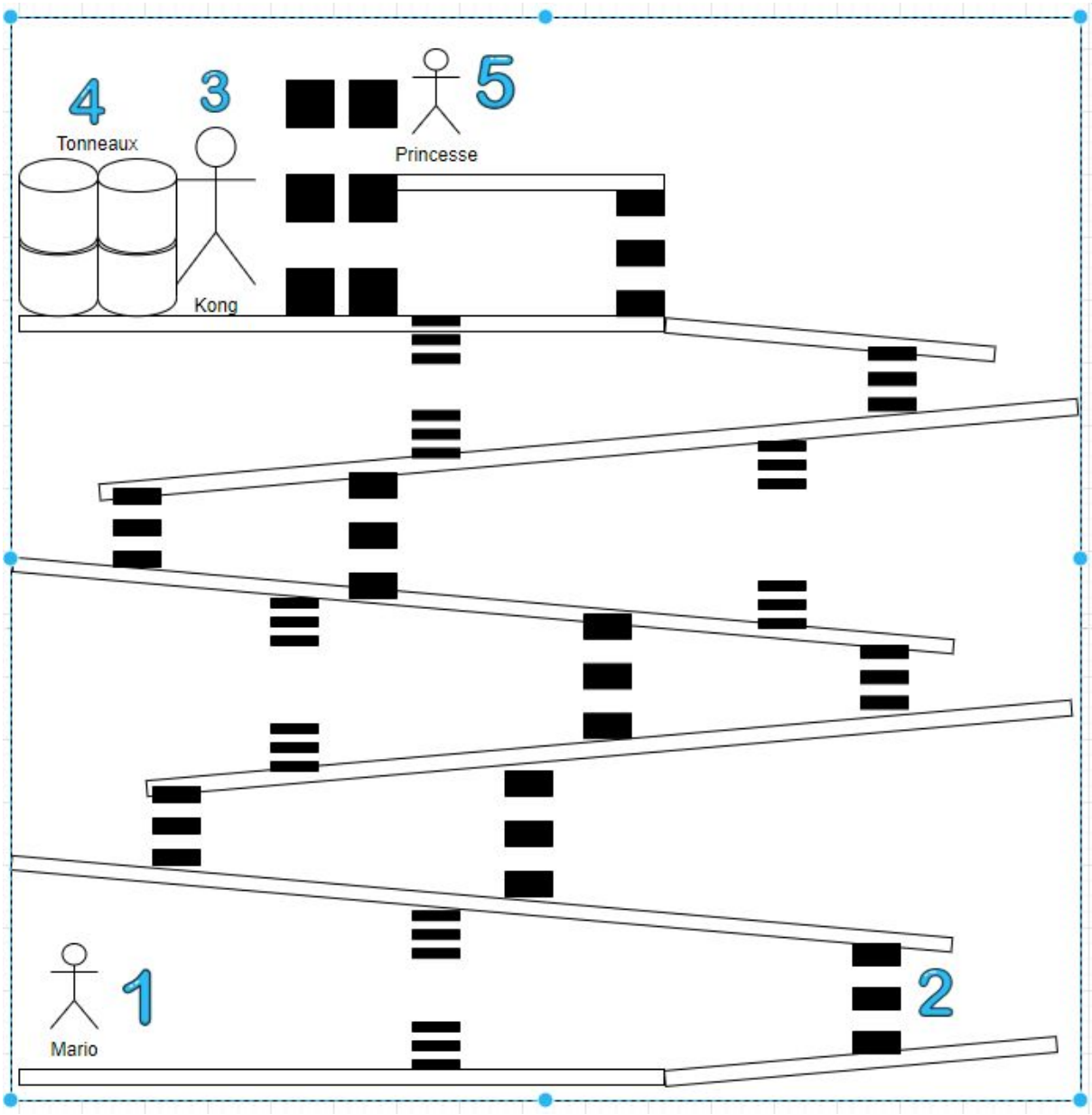

Fig. 5 - Maquette de la fenêtre de jeu

1. L’utilisateur doit amener Mario jusqu’à la princesse, il pourra se déplacer avec le joystick, sauter et monter les échelles. Il devra le faire en esquivant les tonneaux
2. Mario peut monter les échelles comme raccourci pour passer au prochain étage au lieu de sauter d’un étage à l’autre
3. Kong lancera des tonneaux avec un interval aléatoire
4. Après avoir été lancé par kong les tonneaux descendent de façon physique le plateau, ils auront une chance de descendre les échelles pour confondre le jouer.
5. Le but du jeu est de sauver la princesse, elle ne bougera pas et le jeu se termine quand Mario l’atteint

Fenêtre informations supplémentaires

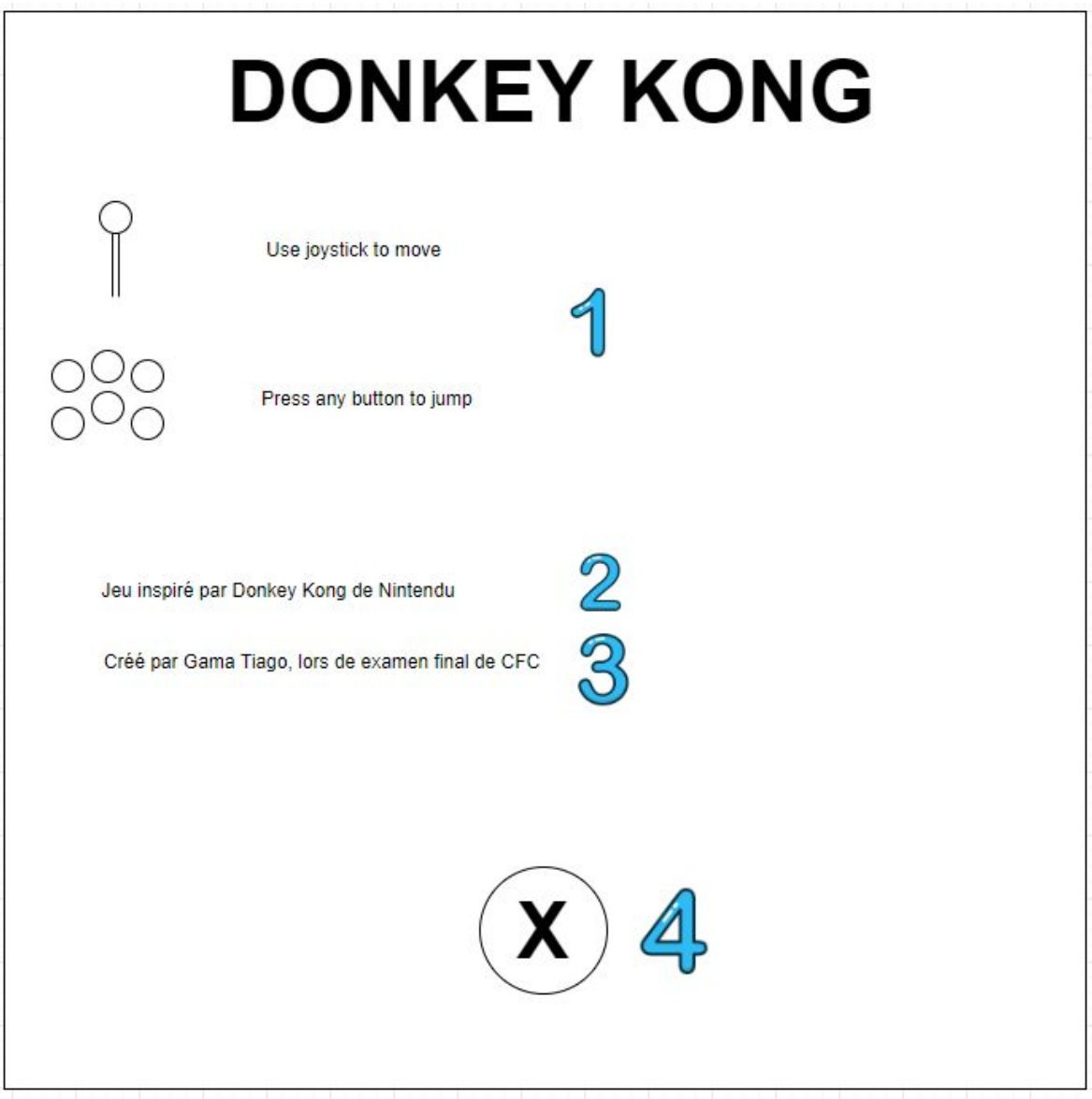

Fig. 6 - Maquette de la fenêtre d'informations supplémentaires

1. Permet à l'utilisateur de comprendre comment jouer le jeu
2. Permet à l'utilisateur de savoir d'où vient le jeu original
3. Permet à l'utilisateur de savoir qui a créé cette réplique
4. Permet à l'utilisateur de retourner vers le menu

Analyse Organique

Langage de développement

La totalité du programme a été réalisé en C#, qui est le langage d'apprentissage choisi par le CFPT.

C# est un langage de programmation orienté objet, qui a été conçu par Microsoft. Lors de ce projet j'ai utilisé la dernière version sortie c'est-à-dire la 8.0 (20 septembre 2019) avec Visual Studio 2019.

Fig. 7 - Logo de C#
source: https://gunnarpeipman.com/csharp-using-declarations/

Framework

Le framework utilisé pour le développement de ce projet s'appelle monogame, j'ai appris comment utiliser monogame lors de l'atelier applications en troisième année de mon CFC.

MonoGame est une implémentation Open Source du framework Microsoft XNA 4.

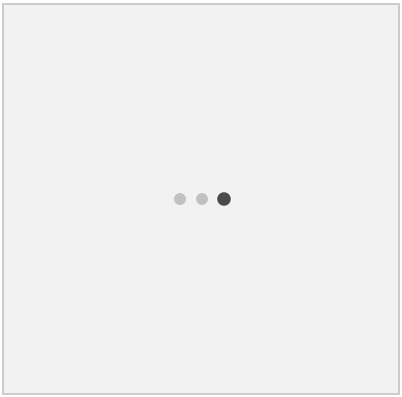

Fig. 8 - Logo de Monogame

source: https://commons.wikimedia.org/wiki/File:MonoGame_Logo.svg

Structure de données

Lors de l'implémentation des scores et highscores, j'ai dû décider comment le stockage de données allait avoir lieu. J'ai décidé de le faire avec une fichier XML oú l'application stocke un "Score" qui contient un nom d'un joueur (le jeu ne permet pas à son état d'introduire un nom et utilise donc "NONAME" mais cela est une des améliorations à penser) et la valeur de son score.

Ensuite, vu les nécessités du programme j'ai décidé de stocker seulement le meilleur score (highscore), mais une autre amélioration intéressante serait d'avoir une page oú on peut trouver, le TOP 10 des scores, et donc pour cela il faudra stocker les 10 meilleurs scores et non seulement le highscore.

Pour conclure, j'ai décidé d'utiliser un fichier XML et non une vrai base de données pour deux raisons majeures, la première étant que le programme n'as pas une structure complexe de données et a peu de besoins en termes de quantités de données à sauvegarder et la deuxième est ma familiarité avec l'utilisation de XML avec Monogame/C#, lors des plusieurs projets réalisés en monogame durant l'année scolaire j'ai utilisé ce type de structure de données et je suis content avec ses résultats.

```xml
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfScore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Score>
    <PlayerName>NONAME</PlayerName>
    <Value>0024</Value>
  </Score>
</ArrayOfScore>
```

Fig. 9 - Structure du fichier XML

Architecture du code

Arborescence des fichiers

- Le dossier Controls contient les objets permettant de contrôler une situation.
- Le dossier Game Components (composants de jeu) contient tous les objets du jeu.
- Le dossier Managers contient les gestionnaires du programme
- Le dossier Models contient les modèles utilisés pour faciliter les actions auxquelles leurs noms correspondent
- Le dossier Sprites contient les différents types d'objets qu'on peut avoir dans le programme
- Le dossier States contient les différents états de l'application

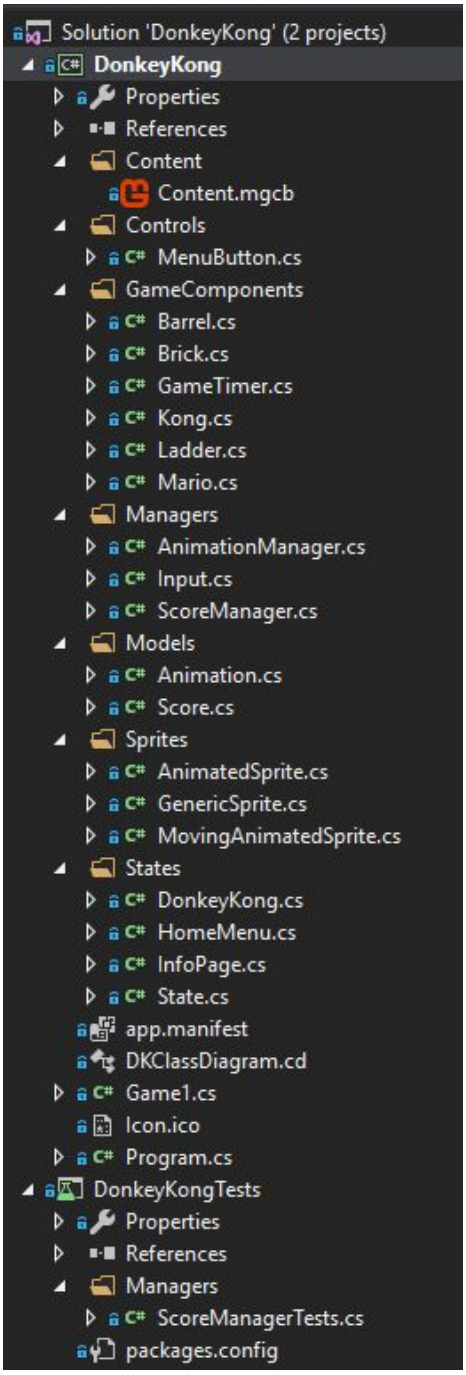

Fig. 10 - Arborescence du code

Diagramme de classes

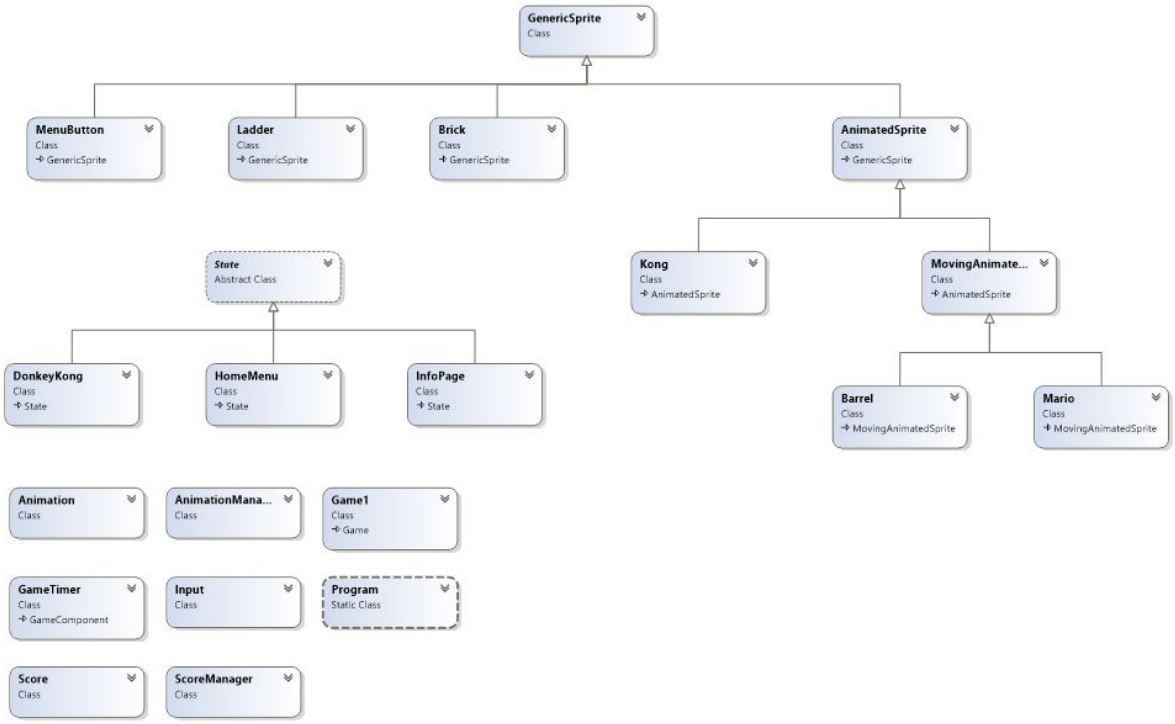

Fig. 11 - Diagramme des classes

Voici le diagramme de classes de mon programme, comme vous pouvez le constater les héritages font une grand partie de la structure de mes classes.

Explication des classes

Groupe d'États ou *States*

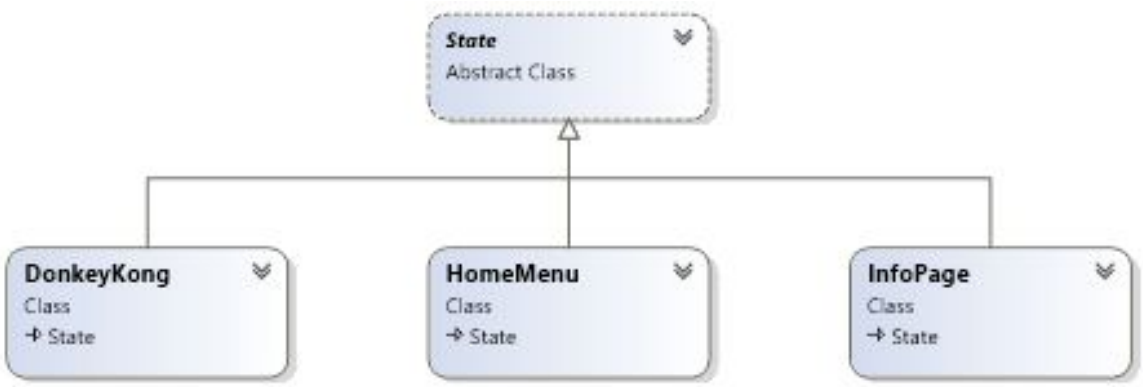

Fig. 12 - États du jeu

Voici la structure des états, comme vous pouvez le voir, State est la classe parente qui modélise un état.

State

Description :

Un état (state en anglais), représente une partie du programme, on utilise cette classe pour modéliser un état et faciliter sa création.

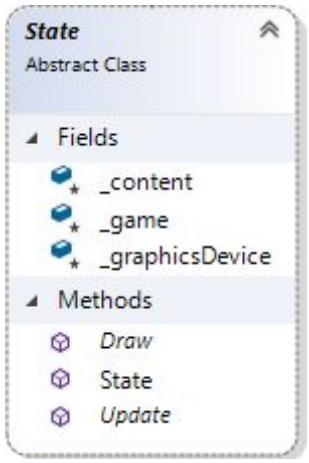

Fig. 13 - Classe State

Home Menu

Description :

Le menu d'accueil est la première page ou état qu'on rencontre quand on débarque sur le programme.

Il est le centre du projet, il nous permet d'avoir une transition entre les différents états de l'application.

Fonctionnalités :

Le bouton PlayButton, représenté graphiquement par une flèche vers la droite nous permet de lancer le jeu.

Le bouton InfoButton, représenté graphiquement par un point d'interrogation nous permet d'aller sur la page d'informations supplémentaires.

Le dernier bouton, le bouton ExitButton, représenté graphiquement par une croix, nous permet de quitter l'application.

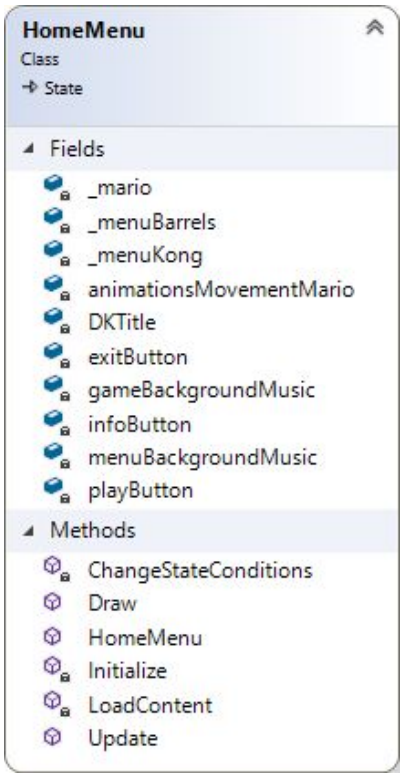

Fig. 14 - Classe HomeMenu

Info Page

Description :

La page d'informations est un How to play, pour les nouveaux joueurs, elle nous montrer ce qui se passe quand on utilise les différents inputs de l'arcade.

Fonctionnalités :

Le bouton GoBackButton, représenté graphiquement par une flèche vers la gauche nous permet de revenir sur le menu.

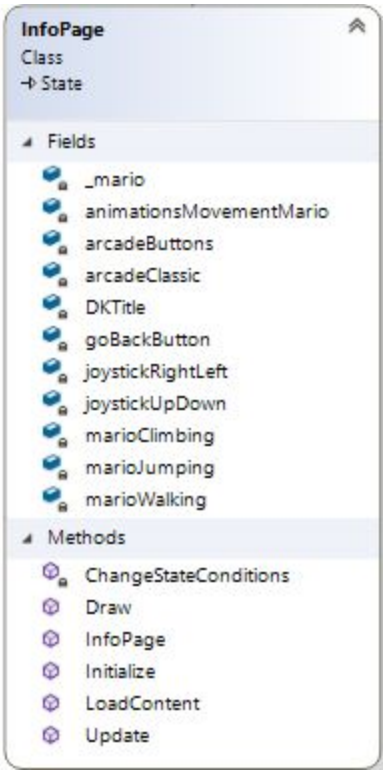

Fig. 15 - Classe InfoPage

Donkey Kong

Description :

L'état Donkey Kong est l'état de jeu, c'est ici qu'on pourra profiter de toutes les fonctionnalités du jeu.

Méthodes principales:

La génération du terrain nommée GroundLayoutSpawn est la méthode permettant de générer le terrain dynamiquement par rapport à la taille de l'écran.

La création des échelles nommée LadderSpawn est une méthode permettant de générer les échelles dynamiquement entre 2 hauteurs données, dans ce cas la différence d'hauteur entre les plateformes.

La logique de suppression des tonneaux nommée BarrelLogic est une méthode permettant de supprimer les tonneaux s'il ne sont plus nécessaires.

La condition de victoire nommée WinCondition est une classe permettant de savoir si Mario a atteint la Princesse.

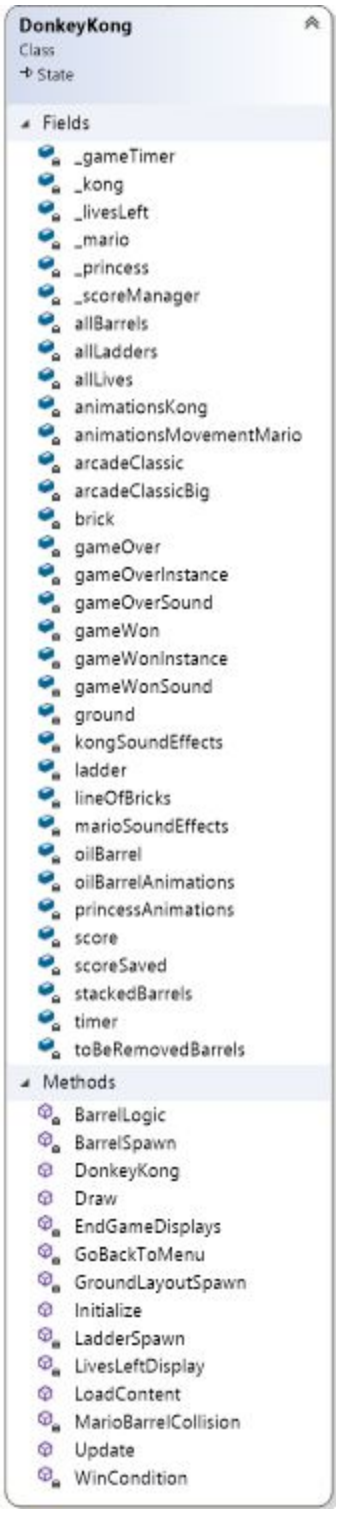

Fig. 16 - Classe DonkeyKong

Groupe de Sprites

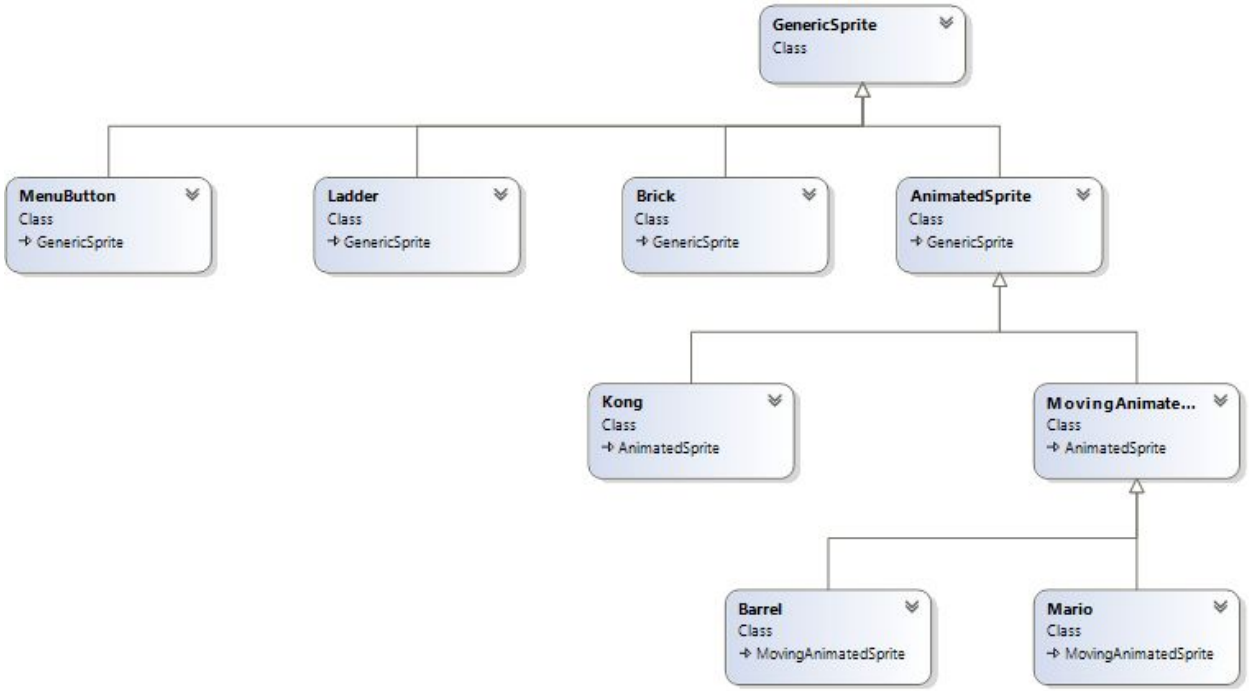

Fig. 17 - Sprites du jeu

Voici la structure des sprites, comme vous pouvez le voir, il y a trois classes principales qui sépare les différentes sprites, GenericSprite, AnimatedSprite et MovingAnimtedSprite.

Generic Sprite

Description :

Les sprites génériques ou GenericSprites sont des sprites comme son nom l'indique, génériques, elles possèdent les informations essentielles pour afficher et utiliser une sprite sur monogame.

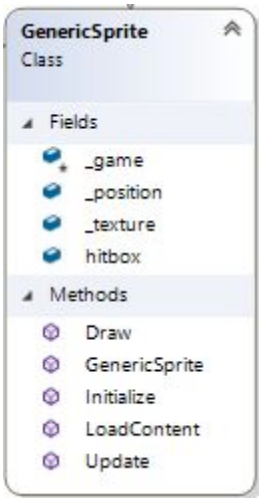

Fig. 18 - Classe GenericSprite

Menu Button

Description :

Le MenuButton est une sprite générique avec une collision spéciale, la méthode CollisionWithCollider affecte directement la variable booléenne collisionWithMario et si elle est égal à vrai, la couleur de l'image change pour simuler l'appuie d'un bouton.

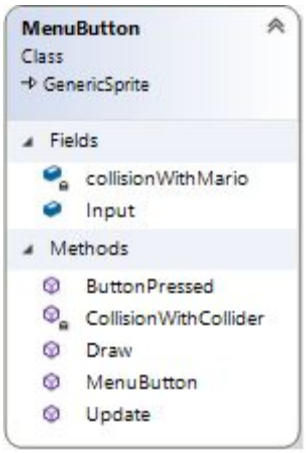

Fig. 19 - Classe MenuButton

Ladder

Description :

L'échelle ou Ladder est une sprite générique avec une collision et un Draw() spéciale, elle est constitué de petit morceaux d'échelle qui ensembles forment une échelle complète. La propriété NbSpritesInStack correspondant au nombre de petit morceaux de l'échelle.

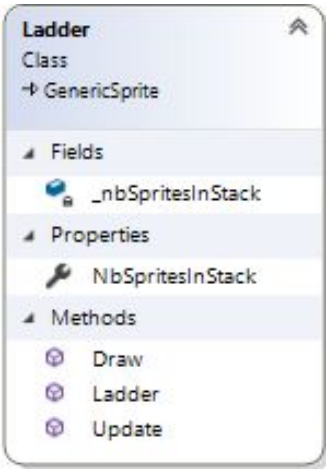

Fig. 20 - Classe Ladder

Brick

Description :

La brique ou Brick est une sprite générique normale avec une méthode de clonage qui retourne une copie superficielle de la brique.

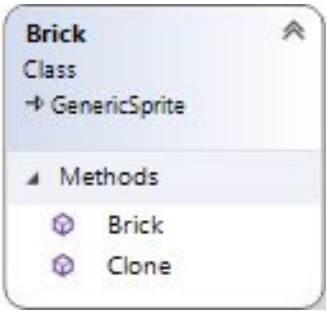

Fig. 21 - Classe Brick

Animated Sprite

Description :

Les sprites animés ou Animated Sprites sont des sprites comme son nom l'indique, animés, elles héritent toutes les propriétés d'une sprite générique et ajoutent la logique et variables nécessaires pour animer la texture d'une sprite.

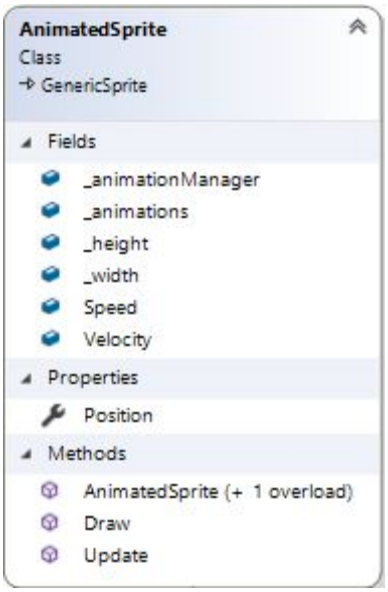

Fig. 22 - Classe Animated Sprite

Kong

Description :

La classe Kong représente le célèbre vilain du jeu Donkey Kong, elle contient deux animations et des bruitages pour elles, son but principale est de laisser le jeu savoir quand il peut créer un nouveau tonneau (Méthode CanSpawnBarrel).

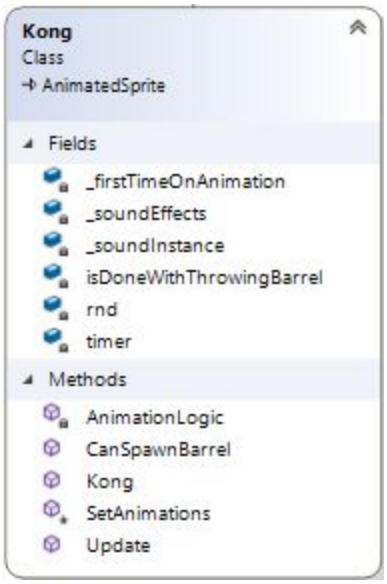

Fig. 23 - Classe Kong

Moving Animated Sprite

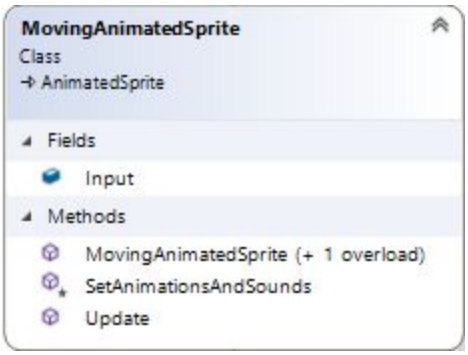

Fig. 24 - Classe Moving Animated Sprite

Description :
Les sprites animés qui bougent ou Moving Animated Sprite, sont des sprites animés qui peuvent se déplacer.

Barrel

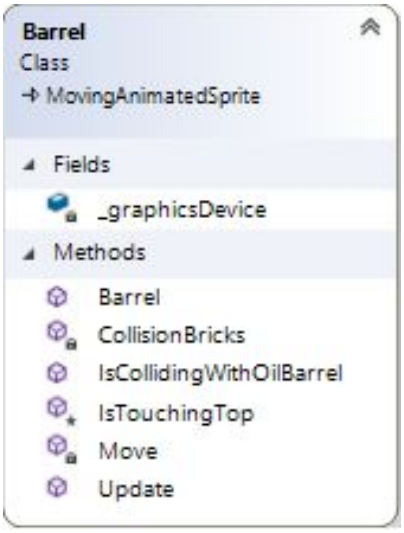

Fig. 25 - Classe Barrel

Description :
Le tonneau ou Barrel est une sprite animé bouge, il subit une gravité jusqu'à rencontrer le terrain (Méthode CollisionBricks) et s'il collisionne avec les barils de pétrole (OilBarrels) il sera supprimé.

Mario

Description :

La classe Mario représente le personnage principal du jeu. Mario est le seule objet contrôlé par le joueur. Le joueur a le droit de déplacer Mario horizontalement, verticalement s'il est en collision avec une échelle et saut s'il est par terre .

Méthodes principales:

La collision avec le sol ou GroundCollision vérifie si Mario est au dessus d'une brick et compense sa position Y si Mario est en train de monter une des plateformes inclinés, les autres collisions sont ignorés.

Le saut ou Jump crée une impulsion qui envois Mario vers le haut. Cette méthode peut seulement être utilisé si Mario est par terre.

La méthode "définir des animations et des sons" ou SetAnimationsAndSounds effectue les changements de l'animation et bruitages de Mario dépendant de la situation de Mario.

La collision avec les échelles ou LadderCollision permet de savoir si Mario est au centre d'une échelle.

La méthode "est Mario mort?" ou IsMarioDead, vérifie si Mario est en collision avec un des tonneaux.

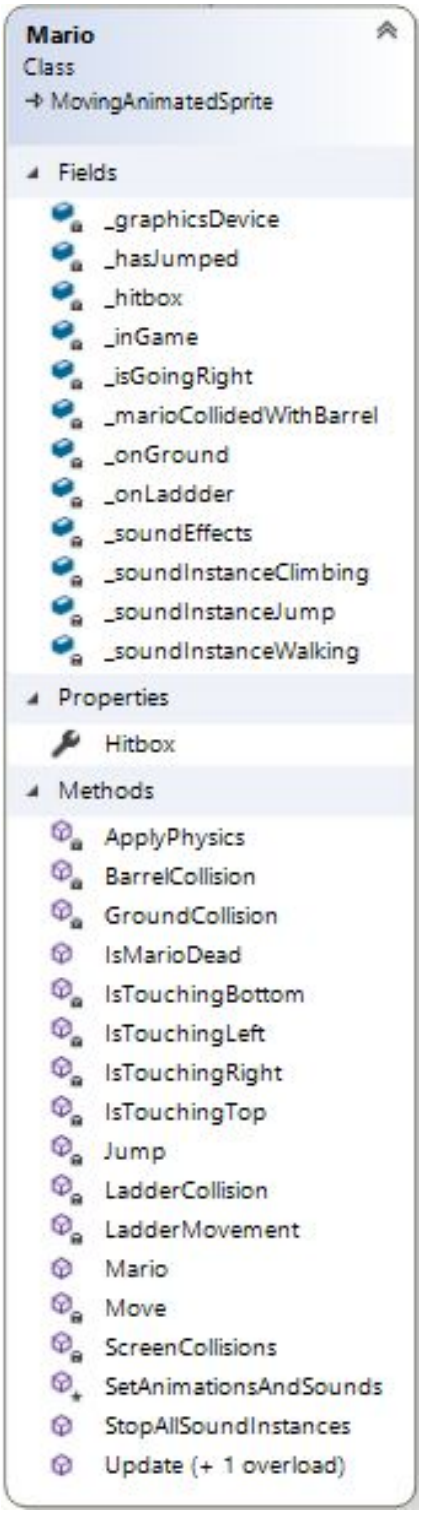

Fig. 26 - Classe Mario

Gestion des Animations

Animation

Description :

Une animation est un modèle pour les textures des objets animés, la classe sépare la texture en plusieurs morceaux et défini la vitesse de changement entre ces morceaux.

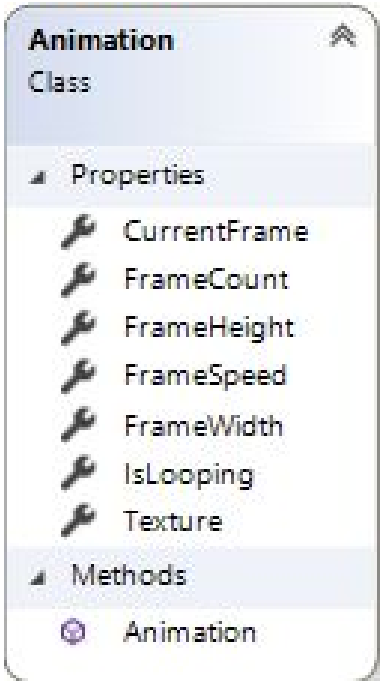

Fig. 27 - Classe Animation

Animation Manager

Description :

La classe de gestion d'animations ou AnimationManager contient toute la logique pour gérer une animation.

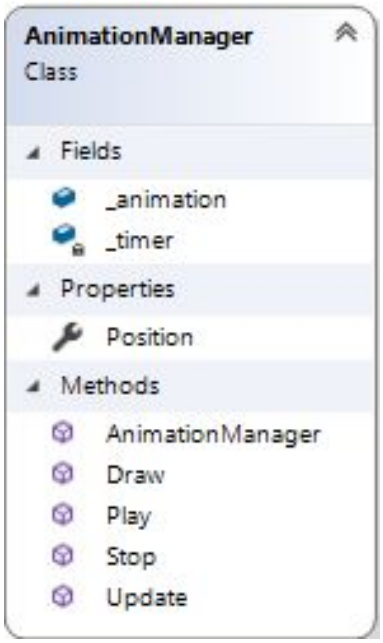

Fig. 28 - Classe Animation Manager

Gestion des Scores

Score

Description :
La classe score est un model utilisé comme structure de données pour la sauvegarde du highscore sur un fichier XML.

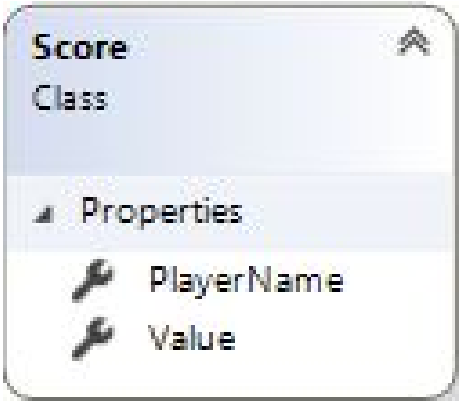

Fig. 29 - Classe Score

Score Manager

Description :
La classe de gestion de scores ou ScoreManeger contient toute la logique pour la sauvegarde et charge des données du fichier XML, elle gère aussi les Highscores.

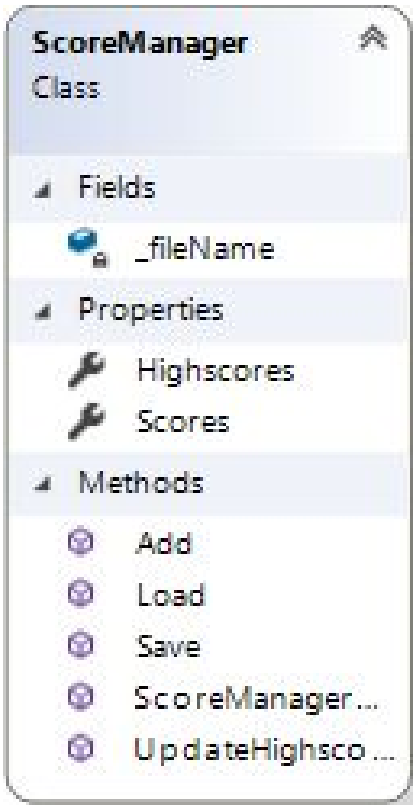

Fig. 30 - Classe Score Manager

Game Timer

Game Timer

Description :
Le minuteur de jeu ou GameTimer compte depuis combien de temps le joueur essaie d'atteindre la Princesse, cette valeur est utilisé pour le score.
Le minuteur possède des fonctionnalités de base comme start, stop et pause.

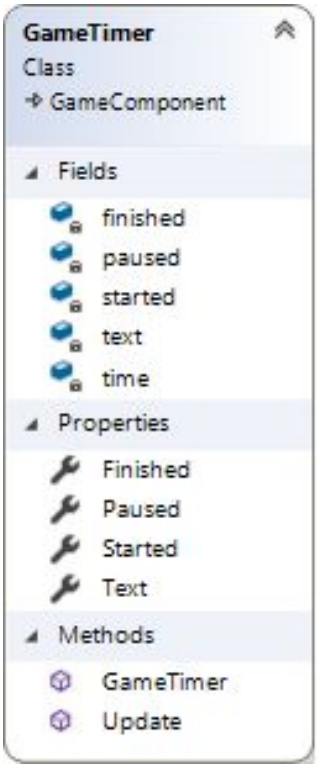

Fig. 31 - Classe Game Timer

Input

Input

Description :
La classe Input représente les entrées utilisateurs, elle est un modèle pour faciliter l'utilisation de Mario car il possède plusieurs touches pour ses actions.

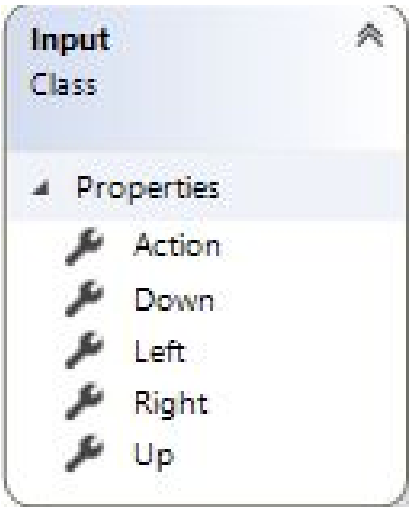

Fig. 32 - Classe Input

Périmètre de tests

Pour le projet Mono Kong, j'ai choisi d'écrire les protocoles de testes pour les fonctionnalités du programme et les interactions que l'utilisateur pourra avoir avec le programme, ces tests ont été effectués sur la machine de départ, donc un PC avec Windows 10 et deux écrans.

Scénarios de tests

Lors du premier jour du TPI, j'ai rédigé un product backlog qui m'as permis de créer des scénarios de tests, ces tests assurent que les fonctionnalités du product backlog fonctionnent parfaitement.

Les tests sont nommés, ils feront référence à l'user story qui leur correspond, ils ont aussi une description, un statut et une date de réalisation

Nom	**1.1 Transitions entre les différentes états de l'application**
User Story	1. Menu d'accueil
Description	Je suis un utilisateur qui débarque sur l'application, j'aimerais pouvoir lancer le jeu ou aller sur la page d'informations à travers le menu. Pour ce faire, je déplace Mario avec le joystick jusqu'à un des boutons du menu et ensuite j'appuie sur un des six boutons à côté du joystick.
Statut et date	Passé le 26 Mai 2020

Nom	**2.1 Affichage d'un tutoriel du jeu**
User Story	2. Page d'informations supplémentaires
Description	Je suis un utilisateur qui joue le jeu pour la première fois, je souhaite comprendre quelles touches je dois utiliser pour pouvoir jouer, je décide alors d'aller sur la page d'informations et je retrouve une explication de toutes les actions du jeu.
Statut et date	Passé le 5 Juin 2020

Nom	**2.2 Transition page d'informations -> Menu**
User Story	2. Page d'informations supplémentaires
Description	Je souhaite retourner sur le menu, pour le faire je déplace Mario avec le joystick jusqu'au bouton en bas à droite et j'appuie sur un des six boutons à côté du joystick.
Statut et date	Passé le 27 Mai 2020

Nom	**3.1 Affichage du terrain**
User Story	3. Affichage des composants du jeu
Description	Je joue le jeu, je peux voir le terrain.
Statut et date	Passé le 28 Mai 2020

Nom	**3.2 Affichage de Mario**
User Story	3. Affichage des composants du jeu
Description	Je joue le jeu, je peux voir Mario.
Statut et date	Passé le 28 Mai 2020

Nom	**3.3 Affichage des échelles**
User Story	3. Affichage des composants du jeu
Description	Je joue le jeu, je peux voir les échelles.
Statut et date	Passé le 4 Juin 2020

Nom	**3.4 Affichage de Kong**
User Story	3. Affichage des composants du jeu
Description	Je joue le jeu, je peux voir Kong.
Statut et date	Passé le 2 Juin 2020

Nom	**3.5 Affichage des tonneaux**
User Story	3. Affichage des composants du jeu
Description	Je joue le jeu, je peux voir les tonneaux.
Statut et date	Passé le 29 Mai 2020

Nom	**3.6 Affichage de la Princesse**
User Story	3. Affichage des composants du jeu
Description	Je joue le jeu, je peux voir la Princesse.
Statut et date	Passé le 2 Juin 2020

Nom	**4.1 Création dynamique du terrain**
User Story	4. Terrain de jeu
Description	La génération du terrain est dynamique et s’adapte avec la taille de l'écran pour être compatible avec plusieurs situations, par exemple le déploiement sur la borne et sur le pc.
Statut et date	Passé le 29 Mai 2020

Nom	**5.1 Gravité sur les objets en mouvement**
User Story	5. Gravité
Description	Lorsqu'un des tonneaux ou Mario se déplacent dans le jeu, il subira une force de gravité qui va le pousser vers le bas, jusqu'à qu'il collisionne avec le terrain.
Statut et date	Passé le 2 Juin 2020

Nom	**6.1 Déplacement de Mario**
User Story	6. Mario
Description	Je peux déplacer Mario avec le joystick, sur le menu et sur le jeu.
Statut et date	Passé le 28 Mai 2020

Nom	**6.2 Saut de Mario**
User Story	6. Mario
Description	Je joue le jeu, Mario saute si j'appuie sur un des six boutons à côté du joystick.
Statut et date	Passé le 28 Mai 2020

Nom	**6.3 Montée des échelles par Mario**
User Story	6. Mario
Description	Je joue le jeu, si j'ai déplacé Mario jusqu'au centre d'une échelle je peux utiliser le joystick verticalement pour faire Mario monter ou descendre une échelle.
Statut et date	Passé le 4 Juin 2020

Nom	**7.1 Conditions du saut de Mario**
User Story	7. Saut de Mario
Description	Mario ne peux pas sauter s'il est dans l'air, il peux seulement sauter s'il touche le terrain.
Statut et date	Passé le 2 Juin 2020

Nom	**8.1 Adaptation au terrain**
User Story	8. Echelles
Description	Les échelles s'adaptent au terrain dynamique et change de taille par rapport à l'espace entre chaque plateforme du terrain.
Statut et date	Passé le 3 Juin 2020

Nom	**8.2 Conditions pour permettre Mario de monter l'échelle**
User Story	8. Echelles
Description	Mario ne peux pas monter une échelle s'il n'est pas au centre de son hitbox.
Statut et date	Passé le 4 Juin 2020

Nom	**9.1 Lancement d'un tonneau par Kong**
User Story	9. Kong
Description	Je joue le jeu, je peux voir la création d'un tonneau au même temps que l'animation de Kong.
Statut et date	Passé le 4 Juin 2020

Nom	**10.1 Descente d’un tonneau**
User Story	10. Tonneaux
Description	Je joue le jeu, je peux voir les tonneaux descendre le terrain de façon physique.
Statut et date	Passé le 29 Mai 2020

Nom	**10.2 Suppression d’un tonneau**
User Story	10. Tonneaux
Description	Les tonneaux sont supprimés s’ils ne sont plus nécessaires.
Statut et date	Passé le 2 Juin 2020

Nom	**11.1 Animations de Mario**
User Story	11. Animations
Description	Les animations de Mario sont visibles (l’animation de marcher, sauter et monter une échelle).
Statut et date	Passé le 4 Juin 2020

Nom	**11.2 Animations de Kong**
User Story	11. Animations
Description	Les animations de Kong sont visibles (l’animation de lancer un tonneau et l’animation d'être inactif).
Statut et date	Passé le 2 Juin 2020

Nom	**11.3 Animations des tonneaux**
User Story	11. Animations
Description	L'animation déplacement d'un tonneau est visible.
Statut et date	Passé le 29 Mai 2020

Nom	**11.4 Animations de la Princesse**
User Story	11. Animations
Description	L'animation de la Princesse est visible.
Statut et date	Passé le 2 Juin 2020

Nom	**12.1 Collision entre Mario et un tonneau**
User Story	12. Collisions
Description	La collision de Mario avec un tonneau provoque la perte d'une vie et Mario retourne au départ du terrain.
Statut et date	Passé le 5 Juin 2020

Nom	**12.2 Collision entre Mario et la Princesse**
User Story	12. Collisions
Description	La collision de Mario avec la princesse provoque la victoire du jeu.
Statut et date	Passé le 5 Juin 2020

Nom	**13.1 Collision entre Mario et un tonneau**
User Story	13. Système de vies
Description	La perte de trois vies par Mario provoque la perte du jeu.
Statut et date	Passé le 5 Juin 2020

Nom	**14.1 Comptage du score**
User Story	14. Score
Description	Le score est l'équivalent du temps requis pour atteindre la Princesse.
Statut et date	Passé le 5 Juin 2020

Nom	**14.2 Affichage du score et du highscore**
User Story	14. Score
Description	Le score actuel et le highscore sont présents dans l'affichage du jeu.
Statut et date	Passé le 5 Juin 2020

Nom	**15.1 Musiques de fond**
User Story	15. Musiques et bruitages
Description	Le menu et le jeu ont une musique de fond (différente).
Statut et date	Passé le 5 Juin 2020

Nom	**15.2 Musiques de victoire ou défaite**
User Story	15. Musiques et bruitages
Description	A la fin du jeu une musique soit de victoire soit de défaite est joué.
Statut et date	Passé le 5 Juin 2020

Nom	**15.3 Bruitages de Mario**
User Story	15. Musiques et bruitages
Description	Mario fait du bruit quand il marche, saute ou monte une échelle.
Statut et date	Passé le 4 Juin 2020

Nom	**15.4 Bruitages de Kong**
User Story	15. Musiques et bruitages
Description	Kong fait un bruit quand il lance un tonneau.
Statut et date	Passé le 4 Juin 2020

Conclusion

Difficultés rencontrés

Tout au long du projet, j'ai rencontré des obstacles, certains plus bloquants que d'autres. Pour m'en sortir j'ai du prendre des décisions sur comment procéder.

Le meilleur exemple de ces obstacles a été la collision avec le terrain, j'ai fait des allers-retours avec quelle logique était la meilleure pour procéder, mes hypothèses de départ avaient des lacunes, je prenais un point de vue par "ligne" et je ne considérait pas l'impacte des lignes supérieures sur mon algorithme de collision, ensuite pour essayer de corriger les bugs amenés par ces lacunes, j'ai décidé d'implémenter un système de niveaux pour séparer ces "lignes" et traiter seulement la ligne nécessaire, pour cela je devrais implémenter des transitions et vérifications de niveau. Après prendre un peu de recul j'ai décidé que cela n'était pas la meilleur méthode pour résoudre les problèmes initiaux et faciliter l'implémentation des autres fonctionnalités. Après une discussion avec M.Russo, un de mes camarades classe, j'ai réussi à combler les lacunes avec des ajustements à l'algorithme initial.

Ensuite, j'ai rencontré des problèmes avec la collision des animations, ce qui m'as surpris car mon code semblait correcte, pour essayer de corriger ces erreurs j'ai testé plusieurs solutions, entre elles la collision par pixel, cette solution a résolu les problèmes mais elle demandait trop de ressources et ralentissait le programme et j'ai décidé donc de l'enlever. Plus tard dans le développement du projet j'ai trouvé que le problème était lié à une mauvaise préparation des sprites, ce qui m'as forcé à revérifier toutes les sprites.

Le dernier problème rencontré, qui mérite d'être mentionné, est la logique pour les échelles, en rétrospective l'implémentation des échelles me faisait "peur", en relisant mon journal de bord j'évitais l'implémentation des échelles et je les laissaient toujours pour le jour suivant.

Je m'inquiétais tellement des petits détails qui pouvaient survenir lors de l'implémentation des échelles, que je n'avais pas pris assez de recul afin de pouvoir penser à une manière propre et simple de les implémenter, lorsque je me suis aperçu que la date du rendu approchait, j'ai dû me mettre au travail qui a fini par être étonnamment facile.

Variantes de solution et choix

Lors de la réalisation de mon projet certaines décisions ont été prises sur comment implémenter les fonctionnalités du cahier des charges. Prenant un peu de recul sur l'ensemble du projet, je suis content de mes décisions considérant la taille du projet et le temps imposé pour sa complétion.

Améliorations possibles

- L'amélioration la plus évidente et l'ajout des autres niveaux de Donkey Kong, cela apporterait de la profondeur au jeu.
- Des obstacles qui impactent le départ du parcours, par exemple dans le jeu original une petit flemme est généré au début du premier de niveau.
- Une affichage du nom du joueur qui possède le meilleur score, la gestion des scores est déjà préparé pour cela, mais l'application n'a pas une interface pour l'entrée du nom d'un joueur.

Comparaison plannings

Dans l'ensemble du travail, je pense avoir été dans les temps. Concernant le planning prévisionnel, j'ai eu de la peine à prévoir à l'avance ce que j'allais faire mais après avoir comparé les deux je trouve que mes idées de départ étaients correctes mais je dois faire attention aux imprévus qui peuvent arriver et ne pas estimer que tout va se passer sans soucis.

Bilan personnel

La réalisation de ce projet a été une expérience très agréables, je me suis toujours demandé comment l'impacte d'avoir un délai précis comme dans le monde de travail pourrait affecter ma manière de travailler. Ce projet m'as permis de mieux comprendre mes points faibles et mes points forts concernant la préparation pour un projet de cette taille. Après comparer ce projet aux petits projets effectués à l'école où une analyse rapide suffit pour trouver la meilleur façon de coder le projet, je comprends beaucoup mieux l'importance d'une analyse et planification détaillé et structuré d'un projet.

Remerciements

Je tiens à remercier :

- M. Garchery pour son accompagnement durant mon TPI
- Christian Russo pour nos sessions de brainstorming, nous faisions tous les deux nos TPI en MonoGame ce qui nous a permis d'avoir des opinions intéressantes sur le travail de l'un de l'autre
- Daniel Carvalho, Elian Cruz, Thi-Kim Duyen Nguyen et Laszlo Dindeleux qui m'ont souvent donné leurs opinions sur l'esthétique de mon travail et la documentation.

Bibliographie

Lors du déroulement de mon projet, j'ai utilisé les ressources suivantes pour obtenir de l'aide technique :

- Les tutoriels de Oyyou sur github et youtube m'ont particulièrement aidés, ses tutoriels contiennent tous les conceptes pour la création d'un jeu Monogame en 2D, je me suis inspiré fortement sur eux
  → https://github.com/Oyyou/MonoGame_Tutorials
- Comment avoir du son dans le jeu
  → http://rbwhitaker.wikidot.com/playing-background-music
- Exemple de logique pour les échelles
  → https://www.reddit.com/r/godot/comments/d9mi3c/dev_blog_01_creating_ladders_in_godot/
- Logique pour les sons du jeu
  →https://community.monogame.net/t/how-do-you-make-monogame-play-a-sound-on-a-key-press-and-stop-as-soon-as-you-stop-pressing-that-key/9958/2
- Un compteur du temps de jeu
  → https://www.youtube.com/watch?v=-2FeSrYT1KE
- La taille du string dessiné avec SpriteFont
  →https://gamedev.stackexchange.com/questions/19438/how-do-i-get-the-height-of-an-xna-spritefont

Source de données

Les éléments du jeu → http://www.mariouniverse.com/sprites-nes-dk/

Le Joystick et les boutons d'arcade → https://www.pngfuel.com/free-png/nsvqq

Les bruitages de Mario → http://www.classicgaming.cc/classics/donkey-kong/sounds

Les musiques de fond →
https://downloads.khinsider.com/game-soundtracks/album/donkey-kong-arcade

Les boutons du menu → https://opengameart.org/content/game-ui-simple-outline-squares

Image de la page de garde → https://wallpaperaccess.com/donkey-kong

Glossaire

Nintendo	Entreprise multinationale japonaise
User story	Une description simple d'un besoin ou d'une attente exprimée par un utilisateur
Scrum	Cadre de développement de produits logiciels complexes
Fichier XML	Langage de balisage extensible en français, est un méta langage informatique de balisage générique qui est un sous-ensemble du Standard Generalized Markup Language (SGML)
Brique	Une brique est un élément de construction généralement en forme de parallélépipède rectangle constitué de terre argileuse crue
Tonneau	Un tonneau est un conteneur de révolution servant à conserver les liquides de consommation