Projet de Baccalauréat : « Run.»

Jeu sur téléphone mobile

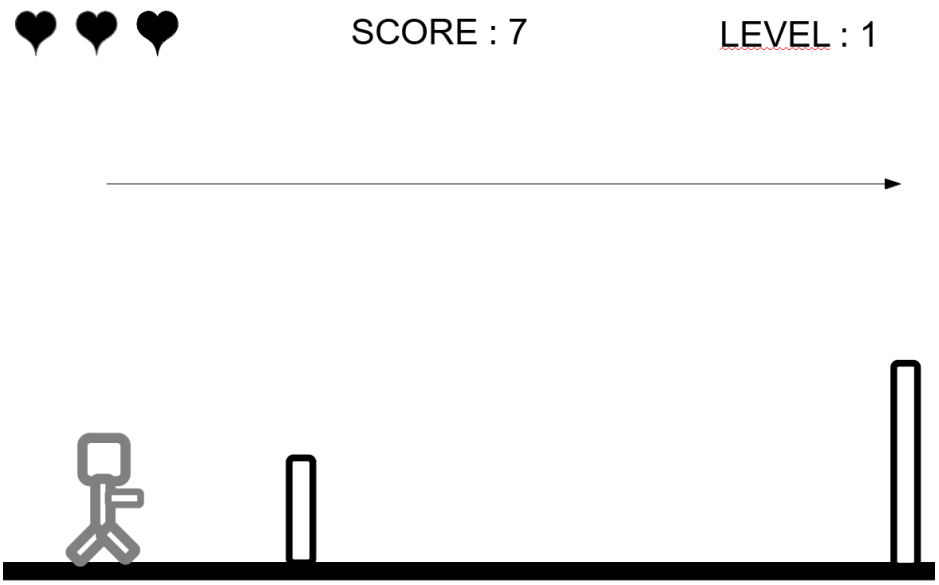
Valentin Foucault

Année 2016-2017

Mme COULON

Introduction

L'idée de réaliser un jeu sur mobile est née dans ma tête en Septembre 2016, lorsque les cours d'ISN venaient tout juste de débuter et que Madame Coulon nous avait alors demandé de commencer à réfléchir à notre projet pour le baccalauréat. Au fur et à mesure de l'année mon idée s'est précisée et c'est en Février 2017 que j'ai soumis à Madame Coulon l'idée de réaliser un jeu mobile nommé « Run. », dont le but serait de faire un maximum de score possible en évitant une série d'obstacles générés aléatoirement. Madame Coulon a accepté notre projet, en posant comme condition que le langage de développement devait être Python 2. J'ai alors entrepris de réaliser une première maquette du jeu ainsi qu'un cahier des charges léger (Disponible en annexe) afin d'avoir un fil conducteur auquel je pourrais me référer durant toute la durée du projet.



Première maquette du jeu

C'est donc en Févier 2017 que j'ai débuté le projet. La première étape fut alors de trouver une librairie Python ainsi qu'un éditeur de code Python adapté à mes besoins. Pour la bibliothèque Python le challenge était de taille : Il fallait en effet trouver une bibliothèque adaptée à la création de jeux 2Ds fluides et qui permettait d'exporter le jeu sur mobile. Après plusieurs heures de recherche je suis parvenu à trouver la bibliothèque « Kivy », qui est selon moi parfaitement adaptée à nos besoins. Elle permet en effet de créer des jeux multiplate-forme grâce à un support Linux, Windows, Mac OS, Android et iOS. De plus le moteur graphique de Kivy est basé sur la technologie graphique OpenGL, me garantissant ainsi que le jeu allait pouvoir être fluide (ce qui est d'une importance capitale pour un jeu tel que « Run.»).

Il a fallu ensuite trouver un éditeur de code efficace qui allait pouvoir m'aider dans le développement du jeu. Après encore une fois quelques heures de recherche j'ai fini par choisir le logiciel « PyCharm ». Celui-ci m'a été d'une grande aide dans ce projet car il facilite la gestion de plusieurs fichiers .py et intègre des fonctionnalités telles que l'auto-complétation ou le « refractor », qui permet de changer le nom d'une variable à tous les endroits où celle-ci est utilisée en quelques clics.

Règles du jeu

Le jeu a été conçu pour que les règles soient simples et compréhensibles rapidement par l'utilisateur. Le personnage court automatiquement. Plus le personnage avance et évite des obstacles, plus le joueur gagne des points.

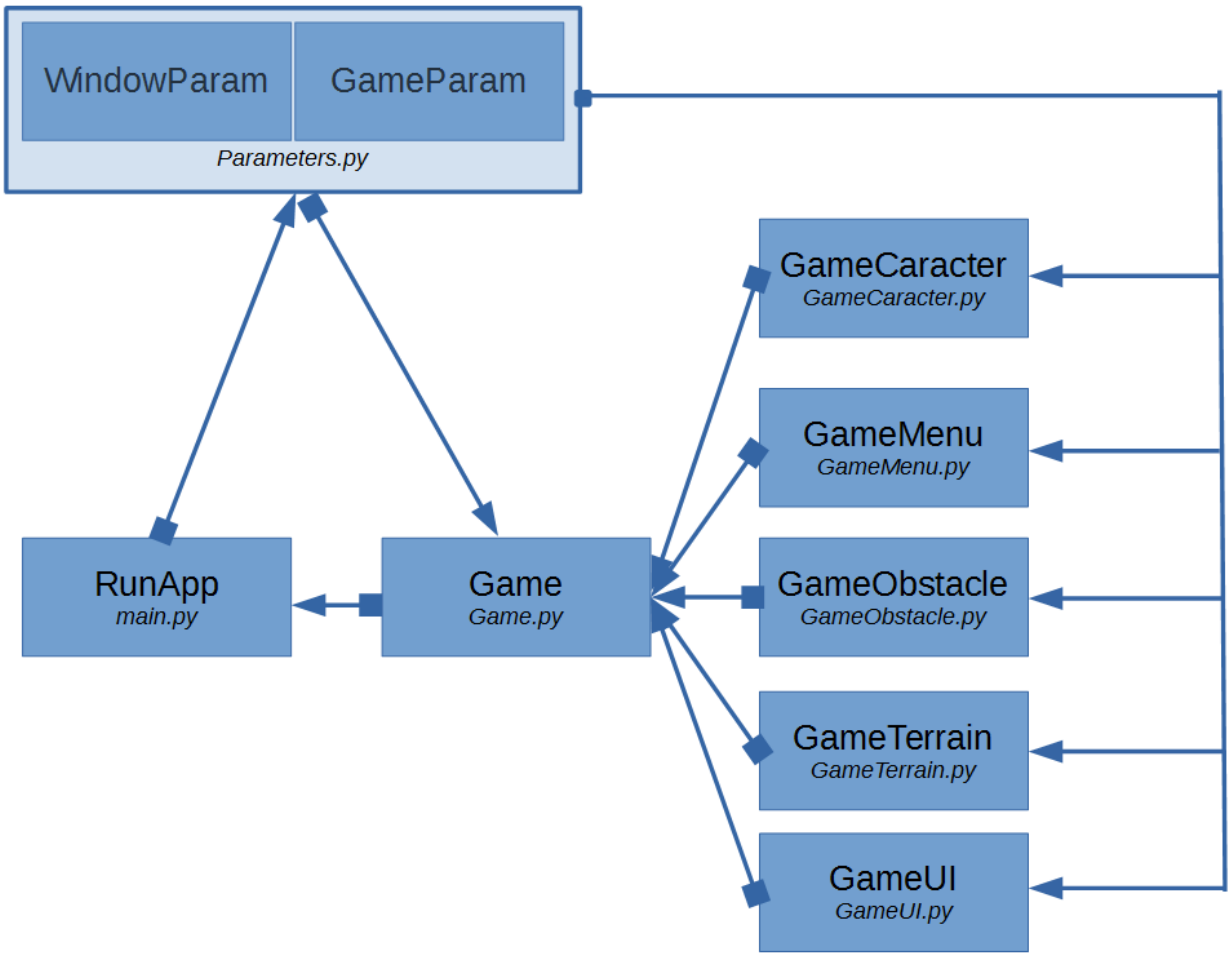
Celles-ci sont définies dans les fonctions **rules_management(self)** et **collision_management(self)**, situées dans la classe **Game.** Ces fonctions sont appelées à chaque image du jeu pour gérer les règles du jeu.

```
def rules_management(self):
```

- **À chaque nouvelle image, le score du joueur augmente de 1 point :**
  ```
  self.player_score += 1
  ```
- **À chaque fois que le score est un multiple de 150 :**
  ```
  if self.player_score % 150 == 0 :
  ```
  - **Le niveau du joueur augmente, et l'interface graphique est mise à jour en conséquence**
    ```
    self.level += 1
    self.ui.update_level(self.level)
    ```
  - **La vitesse de l'obstacle augmente selon une fonction croissante**
    ```
    self.obstacle.speed = ((32 - 250 / (float(self.level) + 10.5)) * WindowParam.window_x) / 1920
    ```
  - **Si le niveau est supérieur à 60, on génère et fait apparaître un texte qui s'affichera aléatoirement sur l'écran pour déstabiliser le joueur**
    ```
    if self.level >= 60 :
        self.ui.generate_label_omg()
        self.ui.label_oh_my_god.opacity = 1
    ```
- **Si l'obstacle sort de l'écran, celui-ci est régénéré :**
  ```
  if self.obstacle.pos_x <= -self.obstacle.size_x:
      self.obstacle.generate_obstacle()
  ```
- **Si la vie du joueur tombe à zéro :**
  ```
  if self.player_life <= 0:
  ```
  - **Le menu est réactivé**
    ```
    self.menu.label_you_lost.opacity = 1
    self.menu.menu_activated = True
    ```
  - **Si le score atteint par le joueur est supérieur au meilleur score, le meilleur score est mis à jour et le fichier de sauvegarde du meilleur score est adapté en conséquence**
    ```
    if self.player_score > int(self.player_best_score):
        self.player_best_score = self.player_score
        self.file_score = open("Ressources/Save/best_score.sc", "w")
        self.file_score.write(str(self.player_best_score))
        self.file_score.close()
    ```
- **Si l'obstacle touche le joueur : le joueur perd une vie, l'interface graphique est mise à jour, un son d'explosion est joué, l'obstacle est régénéré (cette partie est définie dans la fonction collision_management(self)), détaillée plus en détail dans l'analyse fonctionnelle**

Analyse fonctionnelle

Dès le départ j'ai pris la décision d'utiliser la puissance de la programmation orientée objet (POO) de Python afin d'organiser notre code plus facilement et de le rendre plus rapide. Le jeu est ainsi réparti en 9 classes qui « s'imbriquent entre elles » dans 8 fichiers différents, comme présenté sur le schéma ci-dessous :



Ainsi la classe RunApp fait appel à la classe Game, qui elle-même fait appel aux classes GameCaracter, GameMenu, GameObstacle, GameTerrain et GameUI. Les classes WindowParam et GameParam, situées dans le fichier Parameters.py sont elles des classes utilisées par toutes les classes, et sont alimentées par des paramètres provenant de la classe RunApp.

- **RunApp :** Classe appelée au lancement du programme. S'occupe d'ouvrir une fenêtre et d'y afficher le jeu
- **Game :** Lie tous les composants du jeu entre eux
- **GameCaracter :** Permet la gestion du personnage
- **GameMenu :** Permet la gestion du menu du jeu
- **GameObstacle :** Permet la gestion de l'obstacle du jeu
- **GameTerrain :** Permet la gestion du terrain (fond d'écran et sol) du jeu
- **GameUI** : Permet la gestion de l'interface utilisateur du jeu (Score, Niveau, Meilleur score…)
- **GameParam :** Permet d'obtenir diverses informations à propos du jeu (version, auteurs, nom…)
- **WindowParam** : Permet d'obtenir diverses informations à propos de la fenêtre du jeu (résolution, titre…)

Le code source faisant plus de 700 lignes, l'expliquer entièrement serait bien trop long pour ce document. Néanmoins je peux vous présenter l'algorithme de deux fonctions intéressantes du jeu.

Tout d'abord il faut comprendre que la fonction **update()** de la classe **Game** est au centre du programme. En effet c'est cette fonction qui va être appelée 60 fois par seconde afin de mettre à jour jeu en fonction des actions du joueur. Cette fonction est chargée de faire appel à d'autres fonctions qui vont gérer la collision, l'animation, les règles du jeu, la gestion du menu de départ ou encore du menu pause. Je vais ici expliquer deux fonctions qui servent à la fonction **update().**

La fonction **collision_management** :

*Cette fonction est appelée à chaque image du jeu par la fonction **update().** Son but est de détecter si l'obstacle rentre en collision avec le personnage. Si une collision est détectée, le joueur perdra alors une vie.*

Algorithme :

Soit la fonction **collision_management(self)** :

SI la fonction **collide_widget(obstacle)** de l'objet **widget_caracter** de l'objet **caracter** renvoie True, ALORS :

- La variable **player_life** diminue de 1
- La fonction **update_life(vie),** de l'objet **ui** met à jour l'interface graphique en fonction de la variable **player_life**
- La fonction **play(),** de l'objet **sound_explosion** de l'objet **ui** joue un son correspondant à une explosion
- La fonction **generate_obstacle()**, de l'objet **obstacle** re-génère un obstacle avec une nouvelle position, une nouvelle taille et un nouveau type.

Code Source :

```
def collision_management(self):
    if self.caracter.widget_caracter.collide_widget(self.obstacle.widget_obstacle) == True :
        self.player_life -= 1
        self.ui.update_life(self.player_life)
        self.ui.sound_explosion.play()
        self.obstacle.generate_obstacle()
```

La fonction **generate_obstacle** :

Cette fonction est appelée à chaque fois qu'il est nécessaire de re-générer un nouvel obstacle (par exemple lorsque le personnage heurte un obstacle). Elle utilise d'autres fonction qui ont été écrites par moi-même et qui permettent de générer un obstacle avec un nouveau type, une nouvelle position et une nouvelle taille.

Algorithme :

Soit la fonction **collision_management(self)** :

- Générer une nouvelle position aléatoire l'obstacle à l'aide de la fonction **generate_pos**
- Générer une nouvelle taille aléatoire de l'obstacle à l'aide de la fonction **generate_size**
- Générer un nouveau type d'obstacle aléatoire à l'aide de la fonction **generate_type**

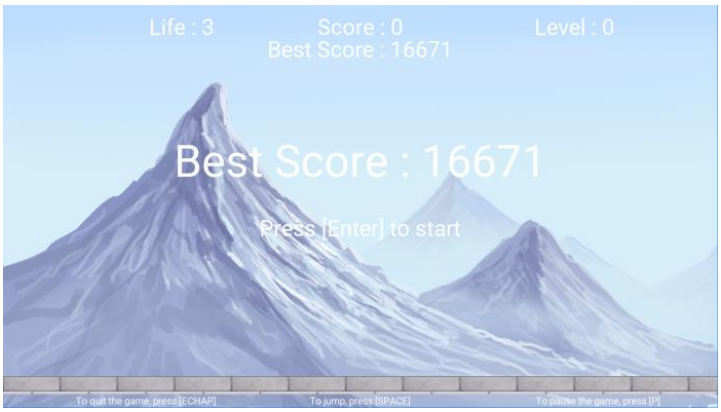
Code Source :

```
def generate_obstacle(self):
    self.generate_pos()
    self.generate_size()
    self.generate_type()
```

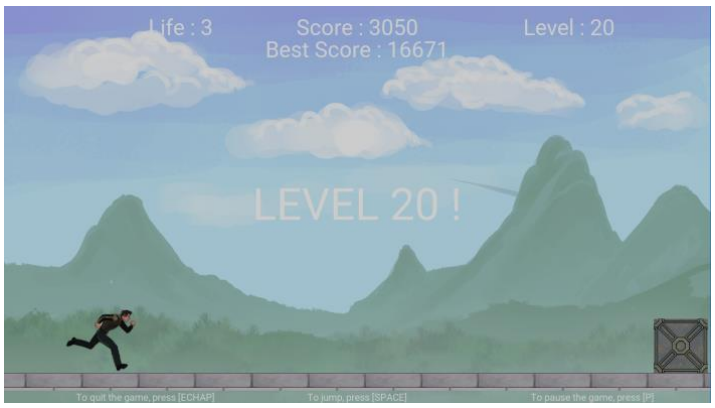
De plus, voici les principales fonctionnalités du jeu :

- Le jeu est tout d'abord traduit entièrement en anglais afin de pouvoir être distribué partout dans le monde, a été codé de façon à ce qu'il soit multiplate-forme (Compatible avec Android, Windows et Linux pour le moment), et est capable de s'adapter à un maximum de résolutions d'écran différentes.

- Lorsque vous lancez le jeu un Menu vous accueille en vous montrant votre meilleur score et en vous invitant à appuyer sur la touche Entrée pour lancer le jeu. Une musique de fond se met également en marche :

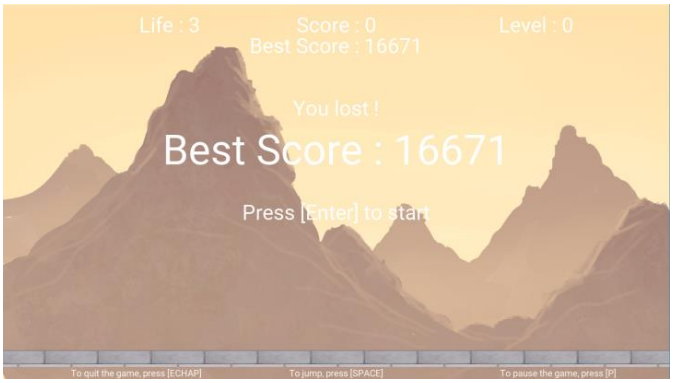


- En appuyant sur Entrée le jeu se lance. Vous devez alors appuyer sur la touche Espace pour sauter et éviter les obstacles. Les obstacles arrivent de plus en plus au vite au fur et à mesure que vous montez de niveau. Le personnage est animé, le sol bouge, ce qui donne une impression que le personnage court véritablement. Tous les 10 niveaux le paysage du jeu peut changer aléatoirement, et une inscription « LEVEL X » indique au joueur sa progression actuelle :



- En appuyant sur la touche P vous pouvez à tout moment activer le mode Pause du jeu. Le jeu s'arrête temporairement et affiche sa version actuelle en haut à gauche. Réappuyer sur la touche P vous sortira du mode Pause :

- Si le personnage perd toutes ses vies, le menu se réaffiche et si le score est meilleur que le meilleur score, le meilleur score est mis à jour sur le disque dur. Appuyer sur la touche Echap permet à tout moment de quitter le jeu :



Travail en groupe

Lors du lancement du projet nous nous étions mis d'accord sur la répartition des tâches suivantes :

- Valentin : Je serais chargé du « management » du projet et de la première structure du programme. Je serais également chargé de réaliser l'exécutable Windows et Android
- Eugène : Il serait chargé de développer certaines fonctionnalités du jeu et de réaliser la partie graphique du jeu

Cette répartition des tâches impliquait un travail régulier et assez intense afin de pouvoir sortir un jeu fluide, avec des fonctionnalités intéressantes et attractives. Nous nous étions mis d'accord sur le projet à réaliser et sur le fait que la quantité de travail devait être la même pour tous les deux.

Malheureusement, malgré mes demandes répétées à Eugène de travailler plus régulièrement sur le projet la quantité de travail n'a au final pas été répartie équitablement, et j'ai dû parfois travailler sur des éléments que Eugène devait réaliser, voire reprendre une partie de son travail.

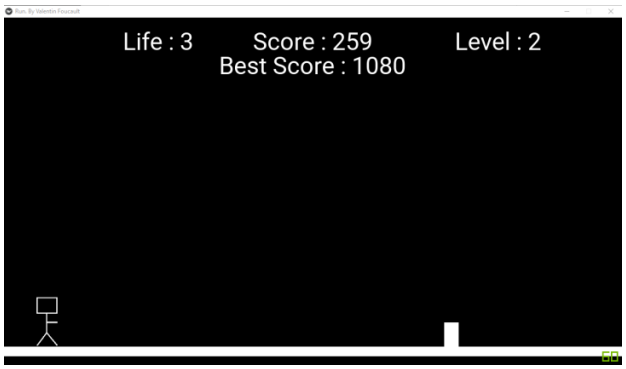
Afin de pouvoir travailler en même temps à distance sur un même fichier Python nous avions, au début du projet, commencé à utiliser un site web nommé « Floobits ». Cependant nous l'avons rapidement abandonné car son inconvénient était qu'il était obligatoire de rendre le code public pour pouvoir l'utiliser gratuitement : https://floobits.com

Difficultés rencontrées et évolution du projet

Bien évidemment ce projet ne s'est pas fait sans peine et nous avons été confrontés à quelques difficultés qui ont pu être surmontées par de nombreuses heures de recherche et d'essais :

Adapter le projet sur mobile a par exemple représenté un certain challenge. En effet un téléphone portable présente de nombreuses contraintes qui doivent être prises en compte pour pouvoir créer un programme. Il faut ainsi prendre en compte que la taille de l'écran peut varier d'un téléphone à un autre, que chaque utilisateur n'a pas forcément les mêmes caractéristiques physiques (il peut avoir de petits ou de gros doigts), ou encore qu'un téléphone n'a pas forcément la même puissance qu'un ordinateur, ce qui implique que j'ai dû alléger le code source et les images utilisées afin d'obtenir un jeu fluide.

Dès le lancement du projet je me suis rapidement mis à développer une première version très basique du jeu mais dont la structure du code (avec une classe par Widget : Personnage, Obstacle, Terrain, Interface Utilisateur) allait nous permettre de l'améliorer au cours du temps :

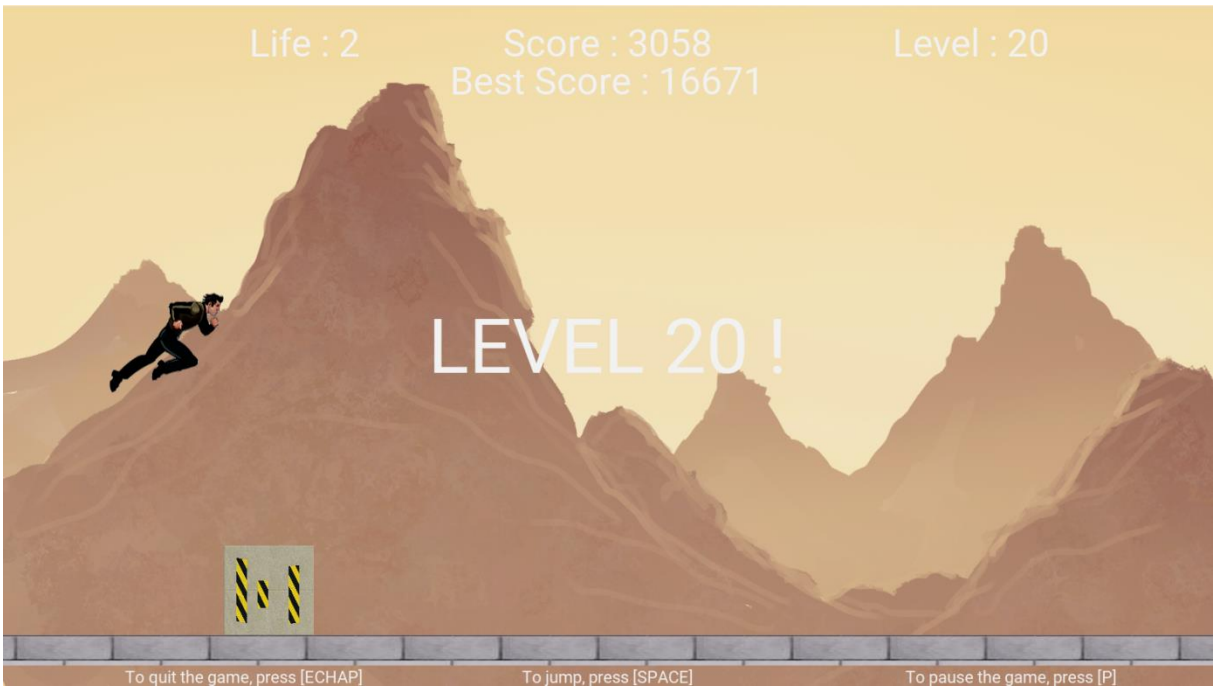


Une des premières versions du projet, le 22 Mars 2017

Par la suite de nombreux ajouts ont été effectués, tels qu'une meilleure gestion du clavier, du son, des textures, et un menu pause a été ajouté :

Une autre version plus évoluée le 24 Avril 2017

Enfin des ajouts esthétiques et des améliorations pour la gestion des niveaux ont été effectués. Le jeu est ensuite rentré dans une période de tests intenses afin de s'assurer qu'il soit correctement jouable sur téléphone et sur ordinateur, puis la version 1.0 est officiellement sortie :



Le jeu aujourd'hui (Version 1.0)

Bien évidemment après tant d'efforts le développement du jeu va se poursuivre après le baccalauréat, dans l'optique de pouvoir le distribuer au grand public. Voici certaines des améliorations qui sont prévues :

- Créer une icône pour le jeu
- Mettre des cœurs à la place d'un nombre pour représenter les vies du joueur
- Réaliser un cryptage du fichier de sauvegarde du meilleur score (ce qui évitera de pouvoir tricher en modifiant manuellement ce fichier)
- Ajouter un bouton « pause » sur la version Android du jeu
- Réaliser des améliorations sonores (meilleurs bruitages)
- Ajouter un bouton pour couper le son
- Ajouter une gestion de plusieurs obstacles sur une même fenêtre
- Ajouter la possibilité de partager et de comparer son meilleur score par internet avec les autres joueurs du jeu
- Ajouter une compatibilité avec l'environnement Apple : iOS et MacOS

Conclusion

En conclusion je peux affirmer que ce projet m'a beaucoup apporté à la fois sur le plan personnel, relationnel, mais aussi sur le plan des compétences techniques. Ces capacités acquises me serviront sans aucun doute pour mes études supérieures. J'ai en effet candidaté dans de nombreuses universités anglaises et canadiennes dans le but de faire un Bachelor en « Computer Science » l'année prochaine.

- Sur le plan personnel j'ai pu confirmer ma volonté de poursuivre des études en Informatique pour l'année prochaine, et j'ai également pu me rendre compte du « danger » que peut représenter le fait de vouloir être trop perfectionniste : je me suis en effet surpris à plusieurs reprises de passer des soirées entières à travailler sur mon projet de bac, ce qui aurait pu m'empêcher de travailler d'autres matières

- Sur le plan relationnel j'ai pu développer mes capacités de « Management » de projet, et j'ai également eu l'opportunité de voir l'importance de bien choisir la personne avec qui travailler sur un projet afin de s'assurer que la charge de travail soit équitablement répartie.

- Sur le plan technique j'ai pu apprendre le langage Python ainsi que la bibliothèque « Kivy ». Cet apprentissage pourra ainsi me permettre de réaliser des projets personnels ou des projets universitaires dans le cadre de mes futures études supérieures.

En définitive je pense qu'avoir pris la spécialité ISN pour mon année de Terminale a été un très bon choix car étant passionné d'informatique, cette spécialité m'a permis de m'épanouir dans mes études en réalisant un projet intéressant et qui me sera utile pour mes études supérieures.

Annexes

Exécutables du jeu

Exécutable Windows pour tester le jeu : Afin que le jury puisse tester le jeu, un exécutable Windows a été créé. Celui-ci est disponible sur une clé USB qui a été remise à Madame Coulon. Pour jouer au jeu, double-cliquez simplement sur le fichier .bat « Jouer ». Celui-ci se chargera de lancer automatiquement le .exe qui est situé dans le dossier « Run ».

Jouer	12/05/2017 23:02	Fichier de comma...	1 Ko

Version sur téléphone mobile du jeu : Le jeu fonctionne parfaitement sur mobile cependant pour des raisons de droits d'auteur relatifs notamment à la musique utilisée dans le jeu, nous ne pouvons pas pour le moment le distribuer au jury par l'intermédiaire du Google Play Store. La version mobile du jeu sera donc présentée au jury le jour de l'oral du baccalauréat.

Code source du jeu

Inclure les **735 lignes de code** qui composent le jeu au sein de ce fichier Word semble assez irréaliste. C'est la raison pour laquelle le code source du jeu ainsi que les différentes ressources utilisées pour faire le jeu (sons, images, fichiers de sauvegarde) ont été remises à Madame Coulon sur la même clé USB qui contient l'exécutable du jeu.

Ressources graphiques et auditives utilisées pour créer le jeu

Musique de fond : « Run » - Par le groupe AWOLNATION
Bruitages : obtenus à partir d'un pack de bruitages disponible ici :
https://mega.nz/#!QVBmjSKb!q99Cu2EeF6xNbt2MxbJNs17JJeTjT3Qt-y_32OzZOcQ
Obstacles, Image de fond : issus du site https://opengameart.org/

Carnet de Bord

La semaine du Vendredi 3 Mars au Vendredi 10 Mars

- Début du projet, découverte de « Floobits », un site web qui nous permettra de travailler nos lignes de code ensemble et de voir les changements apportés par chacun en direct.
- Découverte de la bibliothèque Kivy de Python, qui nous permet de créer des jeux fluides et surtout d'exporter le jeu sur téléphone mobile pour pouvoir y jouer sur un téléphone mobile.
- Création d'un cahier des charges par Valentin pour notre jeu intitulé Run. Nous avons décidé de créer un jeu facile d'utilisation mais attractif. Il y aurait un personnage en bas à gauche de l'écran qui devra éviter des obstacles générés aléatoirement. Nous avons aussi décidé de créer un système de score et de niveau, ainsi qu'un système de vie, qui augmente la difficulté en fonction du temps. Le jeu sera aussi équipé d'une musique (peut-être de notre composition) qui accélérera au cours du temps. En fonction de l'avancement du projet, nous pourrons ajouter un système de bonus et peut-être d'ennemis (obstacles plus développés).

La semaine du Vendredi 10 Mars au Vendredi 17 Mars

Début de l'écriture du code principale du projet Run.
Valentin a :

- Créé un personnage pour le moment immobile
- Codé la génération aléatoire d'obstacle.
- Créé le système de score, de niveau et de vie du personnage
- Importé une musique qui augmente son volume
- Amélioré la structure du code

Eugène a :

- Ajouté la gestion clavier : quand le joueur appuie sur la barre d'espace le personnage saute.

La semaine du Vendredi 17 Mars au Vendredi 24 Mars

- Abandon de « Floobits ». Son utilisation était trop complexe et ne permettait pas de cacher le code du jeu.
- Création d'une accélération pour la montée (le joueur monte de plus en plus lentement)
- Création de la gravité (plus le joueur reste longtemps dans les airs plus la gravité augmente donc il descend encore plus vite)
- Création d'un système de meilleur score qui enregistre dans un fichier texte notre meilleur score pour le conserver
- Portage sur Android du jeu : des problèmes rencontrés pour la sauvegarde du meilleur score et surtout que le clavier s'affiche pendant le jeu, ce qui empêche de joueur correctement. De plus le texte apparaît en trop petit sur l'écran d'un téléphone portable.

La semaine du Vendredi 24 Mars au Vendredi 31 Mars

- Travail sur la « hit box » de l'obstacle et du joueur
- Travail sur la version Android du jeu, encore quelques bugs à corriger

Les semaines du Vendredi 31 Mars au Vendredi 21 avril

- Amélioration de l'aspect du jeu (nouveau fond d'écran, images d'obstacles)
- Amélioration de la structure du code
- Portage linux de l'adaptabilité écran
- Modification de la gravité

La semaine du Vendredi 21 avril au Vendredi 28 avril

- Création d'un menu qui s'affiche au début et à la fin d'une partie
- Création de raccourci sur le jeu pour permettre au joueur de mettre pause, de quitter le jeu ou de recommencer à n'importe quel moment.
- Amélioration de la vitesse et meilleure gestion de la difficulté

La semaine du Vendredi 28 avril au Vendredi 5 mai

- Meilleure gestion et organisation de la totalité du code
- Création d'un bouton pour pouvoir démarrer le jeu avec une souris et pour le portage Android, qui sera au final abandonné
- Création d'un exécutable du jeu
- Développement de l'animation du personnage

La semaine du Vendredi 5 Mai au Vendredi 12 Mai

- Amélioration des graphismes sur l'ensemble du jeu
- Finalisation de l'animation du personnage

La semaine du Vendredi 12 mai au Vendredi 19 mai

- Finalisation du projet, tests de la difficulté du jeu, sortie de la version 1.0
- Création d'une animation du terrain

Cahier des charges initial

Un personnage court et doit éviter des obstacles qui se présentent sur son chemin. Sur l'écran le personnage reste fixe seul les obstacles défilent. À chaque fois que le personnage évite un obstacle il gagne un point. Si le personnage heurte un obstacle, il perd une vie. Le jeu s'arrête lorsque le personnage a perdu toutes ses vies.

Un système de niveaux est en place, le plus le niveau augmente, plus le jeu devient difficile. A chaque changement de niveau, il se passe :

- Une augmentation de la vitesse d'arrivée des obstacles
- Une augmentation de la taille des obstacles, de leur type et de leur espacement,
- Un changement de couleur des obstacles et de l'arrière-plan
- Une augmentation de la vitesse de la musique

Le jeu devra être au moins compatible sur Windows et Android. Une compatibilité Linux, Mac OS et iOS pourra être envisagée plus tard