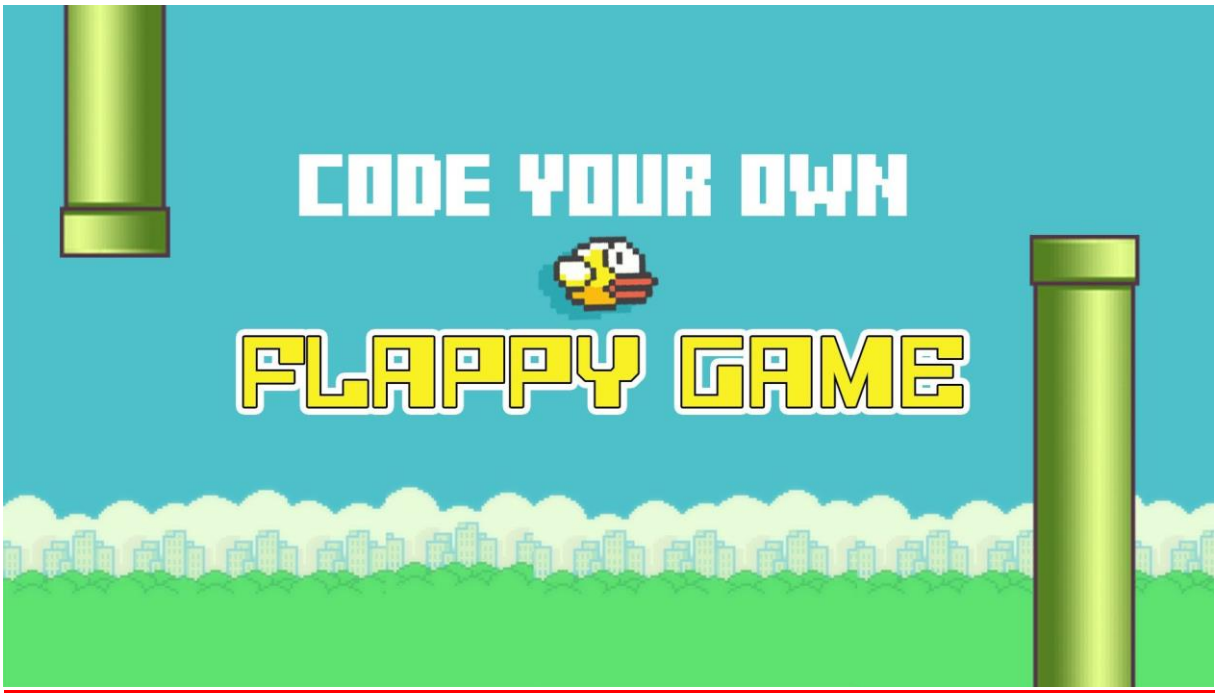
Matthieu Cherrier 731

Compte-rendu du projet Flappy Bird



I – Synopsis du projet

Flappy Bird est un jeu vidéo créé par Nguyễn Hà Đông, au Viêt Nam. Publiée par « .GEARS Studios » en mai 2013, cette application est devenue en janvier 2014 l'application la plus téléchargée de l'App Store. Ce jeu est souvent qualifié de frustrant voire d'ennuyeux, car le seul but est de franchir des hordes de tuyaux qui se ressemblent tous.

Suite à de nombreuses controverses, l'application est retirée de l'App Store et de Google Play en février 2014. En effet, de nombreux graphismes tels que les tuyaux et le sol ressemblaient particulièrement à ceux de l'univers des jeux de Nintendo (Super Mario World notamment). De plus, certaines rumeurs évoquaient l'existence, au bout d'un certain score, d'un « boss » ressemblant particulièrement à Mario, jetant des tomates.

Le principe du jeu est particulièrement simple. On incarne un oiseau qui semble se déplacer de gauche à droite (en réalité, il s'agit d'une illusion d'optique, seuls les tuyaux se déplacent). Le but est d'obtenir le plus gros score possible, tout en passant entre une horde de tuyaux, sachant que chaque rangée de tuyaux surmontée est récompensée par un point. Si l'oiseau touche le sol, ou un tuyau, il tombe par terre et la partie se termine. Comme il se joue sur smartphones, il suffit de taper sur l'écran pour effectuer un saut (qui est d'ailleurs la seule action possible dans ce jeu).

II – Le but poursuivi, et les motivations

Nous avons choisi de programmer ce jeu, car il s'agit d'un des jeux les plus joués sur smartphones auquel nous avons joué tous les deux. De plus, nous l'avons particulièrement apprécié, ce qui nous a d'autant plus encouragés à le choisir. Nous avons cherché à comprendre son fonctionnement, et pour ce faire, nous avons décidé de le recréer avec nos propres moyens. Au cours de son élaboration, nous avons pu acquérir de nouvelles connaissances sur le langage Python et la bibliothèque Pygame, ce qui nous a d'autant plus motivé lors de la programmation.

Mon principal objectif lors de ce projet, était de programmer un jeu qui me plaisait, et qui serait amusant à jouer (bien qu'il se révèle plus dur que le jeu original). De plus, le jeu m'a paru intéressant à programmer, car il m'a permis de découvrir comment se faisait un jeu, en passant de la conception des graphismes aux collisions, tout en passant par les déplacements des diverses images.

III – Déroulement du projet et répartition du travail

Afin d'échanger nos résultats, nous nous sommes envoyés des mails, vus en salle informatique ou en cours d'ISN, et nous avons eu recours à la Dropbox (plateforme permettant le partage et le stockage de fichiers) fournie par Mr.Junier.

Nous nous sommes répartis les tâches de la façon suivante :

Moi (Matthieu) :

- Graphismes (flappies, tuyaux, fond, sol)
- déplacements (oiseau, tuyaux, sols),
- collisions (flappy, tuyaux, sol),
- compteur de tuyaux (et affichage du score)

Vincent :

- Son
- Menu (highscores, bouton PLAY, bouton CREDITS)

Concernant les périodes de programmation, je me suis occupé des graphismes (flappy, tuyaux, fond, sol), des collisions (flappy avec les tuyaux et le sol) ainsi que des déplacements (flappy, tuyaux, sol) du 15 Février 2015 au 29 Mars 2015. Je me suis également chargé, du 30 Mars 2015 au 5 Avril 2015 de la conception des Flappies (de différentes couleurs), du compteur de tuyaux (conception de la variable de score ainsi que de son affichage). En résumé, je me suis chargé de la première partie du projet, tandis que Vincent s'est occupé de la deuxième partie quand j'ai fini la mienne (menu, son et refonte de l'algorithme pour réduire le nombre de lignes).

IV – Présentation des outils et techniques adoptées

Dans le but de mener à bien ce projet d'ISN, nous avons utilisé le langage Python, la bibliothèque Pygame ainsi que le logiciel Photofiltre. Nous avons choisi d'utiliser Python, car nous l'avons étudié tout au long de l'année (et c'est le seul langage que je connaisse), ainsi que Pygame pour la manipulation graphique (nous aurions pu utiliser le module « Tkinter », mais nous étions déjà quelque peu familiarisés à Pygame grâce à un DM).

Python est un langage de programmation créé par Guido Van Rossum en 1990. Il a la particularité de présenter une syntaxe assez simple à utiliser. Nous étudions ce langage en ISN, et ce, depuis le début de l'année 2014. Les langages de programmation comme Python servent à programmer des algorithmes. Un algorithme, en effectuant une suite d'actions bien précises, permet d'aboutir à un certain résultat (comme une recette de cuisine par exemple). Ils permettent par exemple de chercher la première valeur d'une suite, ou encore, de faire des additions.

Python présente également la possibilité d'être utilisé avec des « bibliothèques » ou « modules ». Ce sont des plateformes permettant de « rajouter du contenu » sur un langage. Parmi les bibliothèques de Python utilisées au sein du code, il y a :

- « random » qui permet d'utiliser des fonctions génératrices de nombres aléatoires. Elle a servi à la génération d'un Flappy aléatoire, ainsi qu'au changement d'altitude des tuyaux.
- « pygame » le module que nous avons utilisé, qui permet la création de jeux en 2D. Cette bibliothèque nous a permis d'utiliser de nombreuses fonctions concernant l'affichage graphique.
- « sys » qui permet de récupérer des arguments.
- « os » qui permet d'utiliser des fichiers externes, quel que soit le système d'exploitation.

De plus, nous avons utilisé le logiciel Photofiltre pour faire les différents graphismes. Il existe deux versions, dont l'une d'entre elle est téléchargeable gratuitement. Ce logiciel est très complet, et nous a largement servi à réaliser les graphismes de notre jeu. Son principal avantage consiste à définir les dimensions de l'image que l'on veut créer. Ainsi, cela nous a permis à créer des objets de largeur égale (Flappy et les tuyaux présentent une largeur de 50 pixels).

Pour programmer, nous avons utilisé le logiciel Pyzo, un logiciel qui permet de coder des scripts en Python. Ce logiciel présente de nombreux avantages, comme la sélection de la langue, la possibilité d'ouvrir plusieurs fichiers à la fois, etc…

V – Les difficultés rencontrées au cours de la réalisation de ce projet

Tout d'abord, il nous a fallu choisir le sujet. Afin de partir sur de bonnes bases, nous avons décidé de choisir un jeu qui nous plairait, ou, du moins, que nous connaissions tous les deux. Nous avons hésité entre de nombreux jeux comme le Monopoly, le jeu de la barre, mais notre choix s'est finalement porté sur Flappy Bird.

Peu après, j'ai eu des difficultés à installer Pyzo ainsi que le module Pygame sur mon ordinateur personnel. J'ai essayé de les télécharger plusieurs fois, mais rien à faire, Pyzo ne parvenait pas à trouver cette bibliothèque. J'ai donc dû faire appel à l'aide de Mr.Junier, qui a vérifié l'installation en classe. Il ne s'agissait que d'une erreur de dossier, une erreur qui a tout de même perduré deux semaines.

Ensuite, afin de programmer le projet Flappy Bird, nous avions de nombreuses contraintes relatives au jeu qu'il fallait respecter :

- Faire des graphismes le plus proche possible de ceux du jeu.
- Mettre en place des déplacements pour les flappies, les tuyaux et le sol.
- Prendre en compte les collisions, entre le Flappy et les tuyaux/sol.
- Compter les tuyaux passés, et afficher le score.
- Emettre un son lorsque le score augmente.
- Mettre en place un menu avec un bouton PLAY pour jouer et CREDITS pour voir nos noms.

De plus, nous ne nous sommes pas suffisamment bien répartis les tâches. En effet, Vincent, qui s'occupait du menu, ne pouvait guère avancer tant que le jeu (dont j'étais le responsable) n'était pas prêt. Bien que j'aie tenté de développer cette partie le plus vite possible, il m'a tout de même fallu presque deux mois pour coder le jeu ainsi que les graphismes. Cela signifie que pendant deux mois, Vincent n'a pas pu faire grand-chose.

VI – Algorithmique du code

Parmi les contraintes citées précédemment, nous en avons résolu la grande majorité. Voici les diverses solutions algorithmiques apportées aux problèmes (exprimées en français) :

- Afin de gérer les déplacements, nous avons attribué des coordonnées d'abscisse et d'ordonnée à des images. Ces coordonnées correspondent au coin supérieur gauche de l'image. En posant un changement de valeur de ces coordonnées dans une boucle, on répète indéfiniment ce changement. Concrètement, cela signifie par exemple que le Flappy continuera de descendre tant qu'il ne touchera pas le sol.
- Pour gérer les collisions, on attribue des coordonnées d'abscisse et d'ordonnée à des images. Ces coordonnées correspondent au coin supérieur gauche de l'image. On attribue donc un couple de coordonnées au Flappy (coin supérieur gauche et coin inférieur droit) et aux tuyaux (altitude et côté droit). Or, si l'un des couple de coordonnées du Flappy devient égal

voire supérieur (ou inférieur selon les cas) au couple d'un tuyau, cela signifie, graphiquement parlant, que l'oiseau et l'obstacle se sont rencontrés.

- Afin compter les tuyaux, on crée une variable « score » qui augmentera au fur et à mesure. A chaque fois que l'abscisse d'un tuyau atteint une certaine valeur (la même valeur que l'abscisse du Flappy), on incrémente cette variable de 1.
- Pour émettre un son à chaque fois que le score augmente, on crée une variable « son » qui correspond à un fichier .wax externe. Dès que l'abscisse d'un tuyau atteint une certaine valeur (la même valeur que l'abscisse du Flappy), l'algorithme joue ce son.
- Afin de mettre en place un menu, on crée une fonction « menuprincipal() » qui comportera plusieurs instructions. Dans ce menu, on a la possibilité de jouer en appuyant sur Y, de quitter en appuyant sur N, et de voir les crédits en cliquant sur C (qui fait appel à une image externe). Grâce à la fonction « pygame.event.get() », le jeu peut prendre en compte les entrées de l'utilisateur. De ce fait, lorsque le joueur clique sur Y, je jeu se lance, s'il clique sur N, la fenêtre du jeu s'éclipse et dans le cas où il appuierait sur C, l'image des crédits défilera de bas en haut, et ce menu réapparaîtra.

Au sein de ce programme, nous avons utilisé de très nombreuses structures de données.

Tout d'abord, les variables. Ce sont des mots ou des lettres qui stockent une information précise (un nombre, une liste, etc…). Parmi les diverses variables utilisées, il y a par exemple « x », « y » qui représentent les coordonnées supérieures gauche du Flappy ; il y a également « img » qui, au sein des fonctions de chargement d'image, permet de stocker une image grâce à l'instruction « =pygame.image.load(fichier_image) » écrite juste après.

Ensuite, nous avons utilisé des listes. Ecrites sous la forme de mots/lettres, elles permettent de stocker plusieurs informations à la fois. D'après le code, il y a par exemple la liste « flappies_x » qui stocke l'abscisse des flappies du menu (qui sont, au départ, toutes égales à 0). Ensuite, la liste « flappies_x_final » contient les abscisses finales de ces flappies.

Après cela, nous avons également utilisé des fonctions. Comme en mathématiques, en donnant un argument, les fonctions permettent de le transformer et de ressortir cet argument modifié. Si l'on prend par exemple la fonction « charger_image() », on prend en argument une image externe au code (un tuyau par exemple) et on retourne cette image au sein de la variable « img ». Il existe également la fonction « menuprincipal() » qui gère tout l'affichage du menu, ainsi que les actions de l'utilisateur, mais également la fonction « jeu() » qui gère les coordonnées, les blits, etc..

De plus, nous avons utilisé une classe. Même principe que les listes, sauf que les classes présentent la possibilité de contenir des fonctions. Dans notre code, la classe « CoupleTuyau » contient les fonctions relatives à l'altitude des tuyaux, les collisions avec Flappy, mais aussi des variables telles que l'écartement entre les tuyaux ou même leur largeur.

VII – Analyse de la réalisation et prolongement possible

Au cours de ce projet, nous avons essuyé de nombreuses difficultés. Tout d'abord, il nous a fallu un certain temps pour produire les graphismes de ce jeu, car, nous voulions que ce soient des graphismes faits personnellement. Ensuite, nous avons eu des problèmes à propos des variables. Elles étaient particulièrement redondantes, ce qui nous obligeait à refaire régulièrement des schémas pour ne pas s'embrouiller. De plus, comme le code était particulièrement redondant, l'algorithme faisait plus de 500 lignes. De nombreuses rubriques ont été réduites, on est, désormais, passé en dessous de 500 lignes.

Le cahier des charges, lui a été presque totalement complété. Il ne manquait que la musique, qui ne pouvait être jouée car elle entrait en conflit avec le bip des scores. Nous devions donc faire un choix entre la musique, et le bip, qui lui, était présent dans le jeu original. Nous avons donc choisi de garder le bip, afin de rester le plus fidèle possible au jeu originel.

Afin de prolonger le projet, nous avions plusieurs idées.

Parmi elles, celle de jouer la musique et les bips de score en même temps, avec des volumes différents par exemple.

Nous pensions aussi à rafistoler les graphismes, en rendant le fond moins triste, voire en le faisant bouger (on aurait blitté le fond, puis les graphismes bougeant, puis les autres objets juste après).

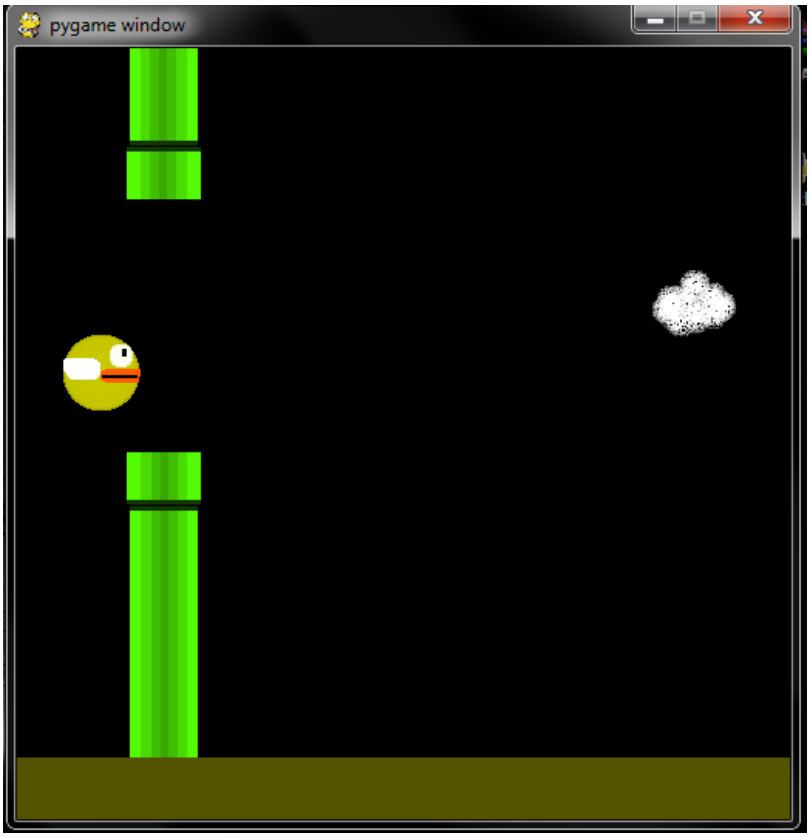
On pensait également à faire bouger les ailes du Flappy, en blittant une image d'aile orientée vers le bas régulièrement.

De plus, nous avons proposé de mettre un bouton « Personnalisation », qui permettrait au joueur de sélectionner la couleur de son Flappy, des tuyaux, du fond en fonction du score maximal de la liste. A partir d'un score de 20 par exemple, on rendrait possible l'utilisation du Flappy Gris.

J'ai également pensé à créer un deuxième mode de jeu, en plus du mode classique consistant à éviter les tuyaux. Dans ce mode de jeu, l'utilisateur incarnerait toujours le Flappy, mais il aurait eu à esquiver des flappies « kamikazes » tombant du ciel (diagonalement de droite à gauche). Toutefois, les actions du joueur restent limitées, car il ne peut se déplacer qu'avec des sauts. Cela rend de ce fait la création de ce mode de jeu impossible.

Et enfin, bien que l'on n'en ait jamais parlé, faute de temps, j'avais pensé à modifier les Highscores, de façon à ce qu'ils affichent des noms en face des scores, qu'ils soient ordonnés (du plus grand au plus petit) ou encore, qu'on mette tout en haut dans une police différente la moyenne de tous ces scores.

VIII – Annexes

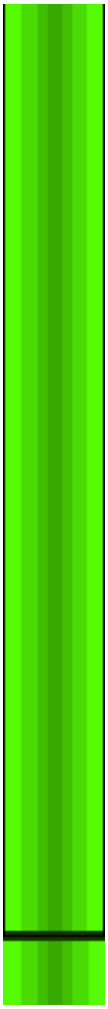


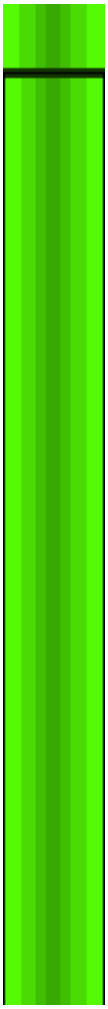
Capture d'écran représentant notre projet à ses débuts. Il n'y avait qu'une rangée de tuyaux, et le Flappy ne craignait pas les collisions.

Photo montrant Nguyễn Hà Đông, le concepteur du Flappy Bird originel.

Image montrant le jeu en deux parties. La première partie est ce qui précède le jeu, une sorte de pause précédent la partie. La deuxième, montre l'oiseau en action.

Voici les images faites à partir de Photofiltre des tuyaux, du sol ainsi que des Flappy Birds.





```
def menuprincipal():
    ecran = pygame.display.set_mode((500, 500))
    ecran.fill((38, 196, 236))
    ecran_rect = ecran.get_rect()
    fond = pygame.Surface((500, 500))
    fond.fill((38, 196, 236))

    ## Chargement des images du menu principal et dÃ©finition de leurs coordonnÃ©es

    titre = charger_image('Titre-Flappy.png')
    playbutton = charger_image('Bouton-Play.png')
    image_flappies = (charger_image('50x50bleu.png'), charger_image('50x50gris.png'),
                      charger_image('50x50jaune.png'), charger_image('50x50rouge.png'), charger_image('50x50vert.png'))
    credits = charger_image('credits-titre.png')

    ## Animation du menu

    chrono = pygame.time.Clock()

    faire_anim_debut(ecran, chrono, fond, titre, credits, playbutton, image_flappies, True)
    while True:
        for event in pygame.event.get():
            if event.type == QUIT:  # Croix rouge
                pygame.display.quit()
            elif event.type == KEYDOWN:
                if event.key == K_y:
                    score = jeu()
                    ecrire_score(score)
                    faire_anim_debut(ecran, chrono, fond, titre, credits, playbutton, image_flappies,True)
                elif event.key == K_n:
                    pygame.display.quit()
                elif event.key == K_c:
                    charger_image_credits(ecran, chrono, fond)
                    faire_anim_debut(ecran, chrono, fond, titre, credits, playbutton, image_flappies, False)
```

Capture d'écran montrant la fonction menuprincipal(), qui gère les animations, les évènements de l'utilisateur (s'il appuie sur Y ou N, etc…) …