Dana AFAGHPOUR Groupe :

Terminale S Soléna DUBUS

Spécialité ISN Inès SEGUIN

****

****

****

****

**I- Notre projet** (p3 à 5)

1. Introduction
2. Notre Scénario
3. Pourquoi ce projet ? Pourquoi un jeu vidéo ?
4. But et objectifs, et découpage du jeu
5. Sources d’informations

**II- Répartition des tâches et démarches collaboratives** (p6 à 7)

**III- Réalisation personnelle** (p7 à 13)

A) Mouvements et commandes d’actions

B) La Caméra

**I- NOTRE PROJET**

1. **Introduction**

Pour notre projet, nous avons décidé de créer un jeu vidéo de plateforme sous forme d’un « pixel game » (jeu formé de pixels de grandes tailles, « visibles »). En s’inspirant notamment des pères fondateurs du jeu vidéo, nous voulions produire un jeu similaire à Mario Bros. Ainsi donc, notre jeu serait donc visible en 2 dimension et permettrait au personnage de sauter, d’avancer, reculer et d’interagir avec son environnement (collision, etc.)

1. **Notre scénario**

Un beau jour, des archéologues en plein milieu de leur travail découvrirent un œuf gigantesque, et fossilisé. Plein de curiosité, ils décidèrent de le confier à un ensemble de chercheurs de la compagnie « LEDA ». Durant un des tests, les scientifiques réussirent à redonner vie à l’être au sein de l’œuf, qui s’avérait être un dinosaure. Par l’ensemble de ces tests, le dinosaure était devenu intelligent (et… radioactif). Son but, dès sa naissance, était de retrouver sa liberté, et donc de revenir à l’aire Jurassique afin de retrouver sa famille. Pour cela, le dinosaure entreprit de s’échapper du laboratoire et de prendre une machine à remonter dans le temps (créé par le laboratoire). C’est ainsi que commença l’aventure de… SUPER DINO !

1. **Pourquoi ce projet ? Pourquoi un jeu vidéo ?**

En choisissant la spécialité ISN, une partie du groupe avait d’ores et déjà envie de créer un jeu vidéo (de tout type possible). On trouvait que c’était un projet à la fois créatif et ambitieux (c’est pour cela que l’on trouvait l’idée intéressante). Qui plus est, le jeu vidéo est un domaine très intriguant car il est fréquent de se demander lorsque l’on joue chez nous comment un jeu peut-il être fait et conçu.

1. **But et objectifs, et découpage du jeu**

Premièrement, notre but est de créer un jeu vidéo de A à Z à l’aide de Python (un langage informatique) et Pygame (bibliothèque d’extension de Python). Plus précisément, nous avions divisé notre projet en de nombreuses étapes, même si nous nous sommes rendu compte au fur et à mesure que la tâche était trop importante.

Ainsi donc, notre but originel était de produire une interface d’accueil, proposant l’option de jouer suite à une commande. Ensuite, nous voulions de cette interface accéder à notre jeu vidéo, composé donc d’une plateforme au sein duquel un personnage pourrait se déplacer et sauter selon les intentions du joueur. Si le joueur tombait, il mourrait et tombait sur une image « Game Over ». S’il finit le niveau, il tombe sur un écran « Bravo ! »

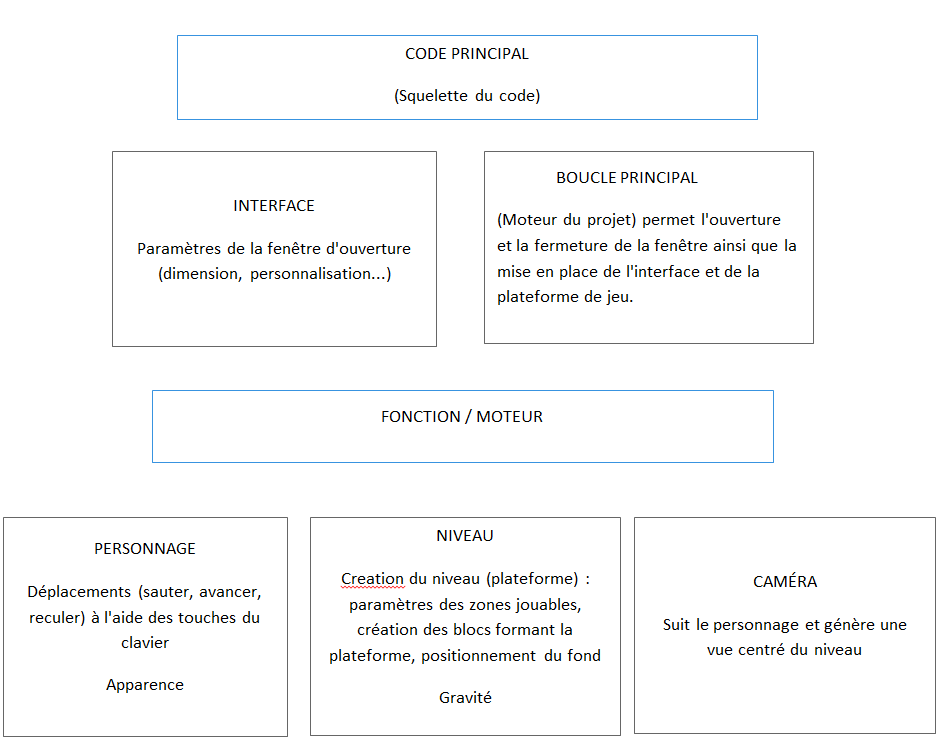
Afin de rendre notre code plus lisible, nous avons voulu diviser le programme avec l’utilisation d’un code principal (squelette du code) qui aurait recours à un autre code « source » qui contiendrait l’ensemble des informations importantes (informations sur les personnages, le décor du niveau, les différents blocs de plateformes, etc.)

Cependant, au final, nous nous sommes heurtés à un problème : notre premier code ne marchait pas vraiment, nous n’arrivions pas à intégrer une gravité et une caméra suivant le joueur. Nous sommes donc allés chercher les réponses sur internet et sommes tombés sur un code « moteur » qui regroupait tous ce dont l’on avait besoin. Ainsi, en s’inspirant de ce code, nous avons obtenu une seconde version de notre jeu, qui intègre pratiquement tout ce que l’on souhaitait, mais qui se retrouve à être très différent de notre premier. Afin de présenter l’ensemble de notre travail, nous avons donc décidé de montrer et expliquer notre première version du jeu, puis ensuite la seconde. Les deux jeux se divisent de la façon suivante :

|  |  |
| --- | --- |
| **SUPER DINO 1 (incomplet)** | **SUPER DINO 2** |
| I- Interface menu du jeu   1. Traitement de l’image sur pygame 2. Chargement des fonctions et classes du jeu 3. Notion d’évènements   II- Le jeu   1. Le fond et les plateformes (construction du niveau) 2. Le Joueur 3. Les mouvements | I- Interface menu du jeu   1. Traitement de l’image sur pygame 2. Chargement des fonctions et classes du jeu 3. Notion d’évènements   II- Le jeu   1. La notion d’Entité 2. Le fond et les plateformes 3. Le Joueur 4. Les mouvements 5. La collision 6. La gravité 7. La Caméra |

1. **Sources d’informations**

Pour nous aider à appréhender la bibliothèque Pygame (que nous n’avions jamais étudié), nous nous somme aidés du site Open Classroom, proposant un tutorial sur les bases de pygame (et python). De plus, nous avons lu différents projets similaires de pixel games fait par d’autres créateurs afin de mieux comprendre la logique et la forme requise pour un jeu vidéo. Il était aussi important à travers cette étape de se familiariser avec les fonctions importantes de pygame dans la création d’un jeu de plateforme (ou même un jeu tout court).

Structure globale du projet :

**II- Répartition des tâches et démarche collaborative :**

La répartition était en résumé ainsi :

|  |  |  |
| --- | --- | --- |
| **INES** | **SOLENA** | **DANA** |
| Menu du jeu  Appel d’un second code (appelé pour notre code « fonction » ou moteur) dans le code principal  Création du niveau dans SUPER DINO 1  Attribution des plateformes dans le moteur de SUPER DINO 1 | Aspect graphique (dessin du décor, de l’interface, de l’échantillon des sprites des personnages et de la plateforme)  Création de la plateforme dans le moteur SUPER DINO 2  Travail de la collision  Travail de la gravité (dans le moteur) | Mouvement des personnagesdans le moteur de SUPER DINO 1 et SUPER DINO 2 (attribution des évènements au joueur)  Travail de la caméra |

Voici nos taches résumés en fonction des semaines :

|  |  |
| --- | --- |
| Nombre de séances et date | Taches |
| 13/01/17 | - Installation de Pygame.  - Cependant, problèmes au niveau de l’installation, pygame n’étant seulement compatible qu’avec Python 3.2 (ou ses versions ultérieurs). Installer une nouvelle version de python (plus vieille) a engendré de nombreux problèmes selon les ordinateurs, mais Ines a réussi finalement à télécharger la bonne version ainsi que Pygame, et parvint à ouvrir l’interface de Pygame.  - Soléna et Dana regardent les bases de pygame sur openclassroom. |
| 20/01/17 | - Ines : absente.  - Création du scénario et découpage du jeu et des étapes à sa création. Prises de décisions sur l’histoire, gameplay, but du niveau (atteindre la machine à remonter le temps).  - Lecture du dernier chapitre du tutorial sur Open Classroom « comment coder un jeu sur pygame. » |
| 27/01/17 | - Lecture du tutorial pygame sur la fonction à utiliser pour créer l’interface et la fermer.  - Définition des dimensions de l’interface de l’écran d’accueil et du jeu. |

Nous avons par la suite mis en commun les codes que nous avions crée pour le jeu, en asseyant de les assembler et s’entraidant tout au long du projet.

**Mode de collaboration :**

Afin de communiquer et modifier plus facilement notre programme codé sous le langage de python et avec l’aide de pygame, nous avons utilisé la plateforme Github qui permet d’interagir avec d’autres créateurs en cas de problèmes, et facilite l’ensemble de l’organisation du codage (car nous y avions tous accès simultanément).

Afin de finaliser notre projet, nous avons travaillés en dehors des heures de cours, fait de nombreuses recherches de notre côté afin d’avancer le projet. De plus, afin de mieux travailler, nous nous sommes vus le weekend pour mettre en commun lors des moments où nous étions trop en retard, et nous avons aussi fait des mises en communs de nombreuses fois pas Skype.

**III- Réalisation personnelle**

Notre projet fur divisé en différentes parties réparties entre chaque membre du groupe. Dans un premier temps, nous avions crée la base du code (le code Super Dino 1), dont Inès c’était occupée de l’interface, Soléna de l ‘aspect graphique et moi des déplacements du personnage et de celui-ci en général. Cependant nous n’arrivions pas à faire marcher le jeu, les images ne s’affichaient pas, le personnage n’arrivait pas à bouger. Nous avons donc décidé de nous tourner vers internet pour mieux comprendre nos problèmes et avons trouvé un moteur pour notre jeu que nous avons par la suite réarranger et complètement approprier. Il s’agit là du deuxième code (Super Dino 2).

Pour ma part j’ai participé à la création des deux jeux à certains niveaux de différences. J’ai débuté le mouvement dans le premier jeu (Super Dino 1) mais les tentatives échouaient les unes après les autres. C’est alors que nous avons trouvé le moteur du second jeu (Super Dino 2) et que j’ai pu finaliser les mouvements. La caméra a été entièrement réalisée dans le second jeu. Je vais donc expliquer les principes de la caméra dans le second code et ceux des mouvements de personnage dans le premier code et dans le second. L’interface et le reste du code est très bien expliquer par Soléna et Inès dans leur dossier.

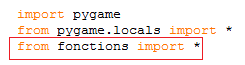
**A- Les mouvements et commandes d’actions**

1. **Mouvements : Super Dino 1**

Je vais en premier lieu vous expliquer l’action ou l’effet voulu par les commandes d’actions. Premièrement, aller à droite : on veut que notre personnage se déplace à droite sur la plateforme en utilisant la flèche droite. Pour cela on attribut à la flèche droite une commande, celle de déplacer notre personnage que l’on aura au préalable définit et de le faire bouger à droite d’un certain nombre de pixel à chaque tentative déplacement à droite (flèche droite).

De même pour celui de gauche, en assignant la flèche gauche du clavier à ce mouvement. Cependant se fut un peu différent avec le saut. En effet, on veut que le personnage puisse sauter (d’une certaine hauteur) et atterrir sur la plateforme d’où il est parti pour faire son mouvement. Pour cela on assigne à la touche flèche du haut cette commande, le personnage monte d’un certain nombre de pixel et redescend mais n’ayant pas pris en compte la « gravité », le personnage de fera que monter et descendre au lieu de « sauter » et « retomber ». Ainsi on pourra voir dans la suite (l’explication du code dans le second jeu) que cette notion de « gravité » permet de crée un mouvement plus fluide et ainsi permettre une meilleur expérience de jeu.

Dans le premier jeu, les commandes de mouvements sont réparties entre le fichier Code Principale et le fichier fonction qui est appelé par le code principale.



Cette division des deux a été faite afin pouvoir mieux apprécier le code, sa lisibilité et compréhension de celui-ci.

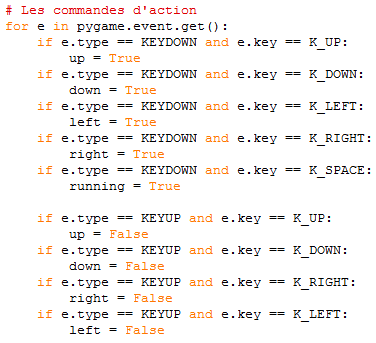
Au niveau du fichier principal, on définit les touches du clavier, assigné à chacun des mouvements possible du jeu : « up » (sauter), « down » (descendre), « left » (aller à gauche), « right » (aller à droite), « running » (courir).

Dans un premier lieu on va dire que toute ces commandes crée sont fausses « False ».



Au début du jeu, le personnage ne peut rien faire, on le voit juste arriver sur la plateforme et quand il y est les commandes s’active et deviennent « True ».

Ainsi en descendant dans le code, on crée les commandes d’actions : les touches du clavier sont assignées à ce moment précis. On utilise la fonction  « e.type == KEYDOWN ». Les touches sont :



Flèche du haut

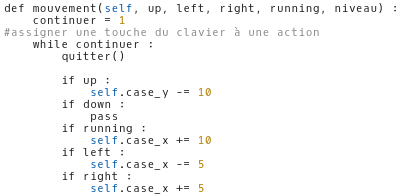
Flèche du bas

Flèche de gauche

Flèche de droite

Barre espace

Les mouvements sont crée en temps que « variable » que l’on défini dans le fichier « fonctions » :



Ici la définition des mouvements est simple, si le joueur appuie sur une des flèches utilisées dans le code, alors le personnage avance, recule ou saute d’un certain nombre de pixels. On utilise la fonction (Une fonction est une suite d'instructions que l'on peut appeler avec un nom) .uptade() pour attribuer au « player » les mouvements et les paramètres de niveau :

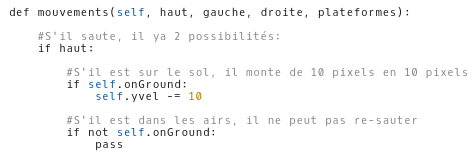


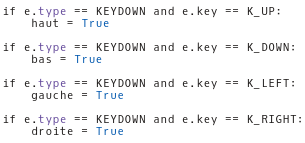
Le joueur peut ainsi faire bouger le personnage de droite à gauche et de haut en bas. Cependant, ce code manque de beaucoup de fonctionnalités qui sont indispensables à une bonne expérience de jeu (par exemple la collision avec une plateforme ou un bloc), c’est pourquoi nous avons crée un deuxième code pour le même jeu afin de rendre compte de notre envie de créer un « produit fini et complet ».

1. **Mouvements : Super Dino 2**

Dans ce second code les mouvements sont définis comme une méthode dans la classe « Joueur » qui se trouve dans le fichier « Moteur » appelé par le « Code Principal Super Dino 2 ».

Ainsi les mouvements et leur assignations se trouve dans la boucle principale du Moteur. On crée une boucle et on utilise la fonction de la bibliothèque pygame : pygame.key est un module que l’on retrouve dans la boucle e qui met en place dans pygame.event.get(). Ainsi pour attribuer un mouvement à une touche du clavier on utilise l’alphabet ASCII notamment les KeyASCII qui permettent directement de sélectionner une touche du clavier. Ainsi si on presse une touche du clavier (KEYDOWN) et une touche en particulier (ex : K\_UP, il s’agit de la flèche du haut) alors l’action devient « vrai » (la variable « haut » s’active).





Cet enchainement d’événements est le même pour tout les déplacements.

Dans les mouvements, j’ai eu des problèmes au niveau de la « gravité ». En effet, je n’arrivais pas à trouver un moyen pour faire tomber notre personnage avec une vitesse. Ainsi j’ai compris en m’inspirant du moteur de base et de nombreuses sources internet qu’il fallait ralentir la vitesse de saut au fur et à mesure que l’on monte d’où :

****

Ici on augmente le yvel de 0.3 (la position (0,0) se trouvant à gauche en haut, pour monter on doit utiliser des « - » en plus de la valeur) donc cela réduit la vitesse de saut. J’avais tenté différente façon de sauter mais ne savais pas que l’abscisse était inversée. De plus mes fonctions d’incrémentation au personnage (joueur) ne marchaient pas car nous n’avions à ce moment pas encore crée de classe « joueur » ou d’instance à laquelle je puisse y attacher les mouvements, il ne s’agissait que de l’image du personnage. Ici pour attribuer les mouvements au personnage on utilise :



**B – la caméra**

Je vais maintenant expliquer le système de la caméra et plus particulièrement la classe « Caméra » (Une classe regroupe des fonctions et des attributs qui permettent de définir un objet) que l’on retrouve dans notre code du second jeu (Super Dino 2). La classe Camera ne se trouve pas dans la boucle principale, seul l’application de celle-ci au joueur et à « l’ensemble » des éléments (les éléments visible du niveau). On définit « ensemble » comme étant une instance de la classe pygame.sprite.Group() qui permet de regrouper et gérer de multiples objets (ou instances).





Il s’agit ici de pouvoir suivre le personnage dans toute la plateforme en mettant au milieu de l’écran, la zone du niveau affichée à l’écran correspondant à la zone située autour du joueur. La plateforme (représenté par les cubes sur lesquels le personnage est posé, elle a été défini dans « main() », notre fonction moteur du jeu, et plus précisément dans la liste NIVEAU qui contient tout les blocs de plateforme) est plus grande que la taille de la fenêtre d’affichage qui est de 640\*480 pixels. Pour que l’on puisse voir le joueur dans tout le niveau, la classe Camera qui est représenté en rectangle (il s’agit là d’un sprite) placé autour de notre personnage

Dans un premier temps on initialise la classe caméra en lui donnant ses paramètres :



Puis on va crée le rectangle Camera, c’est lui que nous pourront « coller » au personnage par la suite. la caméra prend les coordonnées du coin supérieur gauche et s’étend selon la largeur et la longueur qui sont des valeurs que nous avons défini auparavant. La classe Caméra est composé de différentes méthodes que l’on va lui appliquer :



Dans la fonction (méthode) « appliquer » on va bouger le rectangle de la caméra pour le placé devant la fenêtre d’affichage (0, 0, 640, 480) qui sera appliqué à tout les sprites de la plateforme (niveau) selon les déplacements du personnage et donc du joueur.

La fonction mouvements de cette classe permet de recalculer la position de la caméra par rapport à la cible (à notre personnage) pour que celle-ci soit centrée.



Ainsi pour finaliser la caméra on lui donne des limites : en utilisant des « if » (on utilise un système de conditions), on définit les côtés que le rectangle de la caméra ne peut dépasser : le côté gauche, droit, haut et bas correspondant aux cotés du niveau (plateforme) et non de la fenêtre d’affichage. Y et X prenne les nouvelles valeurs qui ont été calculé avant par « self.rectangle\_caméra ».



Ainsi la caméra est un élément complexe qui permet de pouvoir jouer à notre jeu car sans cela, il aurait été impossible de finir le jeu.

**Bilan et perspective :**

Au final, ce codage de jeu vidéo a été très intéressant et bénéfique. D’ores et déjà, travailler sur un projet créatif auquel nous avons tous choisis et créés ensemble m’a beaucoup motivée et m’a amené à me dépasser. Les deux codes que nous avons crée nous montre bien l’aspect multivisionnel que nous permet la programmation. De plus permettre à d’autre une expérience de jeu de qualité nous a permis de comprendre la valeur de la programmation et de toutes les bibliothèques comme pygame qui, sans lesquels, nous aurions mis soit beaucoup plus de temps pour un résultat moins bien ou n’aurions pas réussi.

**IV – Annexe**

Lien Github :

https://github.com/ecolelasource92/Super-Dino

Sources d’informations utilisées :

<https://jeux.developpez.com/tutoriels/python/sdl/Pygame/introduction-a-pygame/>

Pygame documentation :

<http://www.pygame.org/docs/>

Apprendre à programmer avec Python :

<https://python.developpez.com/cours/TutoSwinnen/>

Apprenez à programmer en Python :

<https://openclassrooms.com/courses/apprenez-a-programmer-en-python>