

**Groupe:**

**Dana A.**

**Solena D.**

**Inès Seguin**

**Terminale S**

**Spécialité ISN**





SUPER DINO

**Projet de jeu de plateforme sous Pygame**









**0- SOMMAIRE**

**I- Notre projet page 3**

**II- Répartition des tâches et démarche collaborative page 5**

**III- Réalisation personnelle page 6**

**IV – Problèmes, erreurs et résolutions page 9**

**V- Bilan et perspectives page 10**

**VI – Annexe page 11**

**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 dimensions 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’ère 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 qui nous intrigue beaucoup 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’extensions 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…)

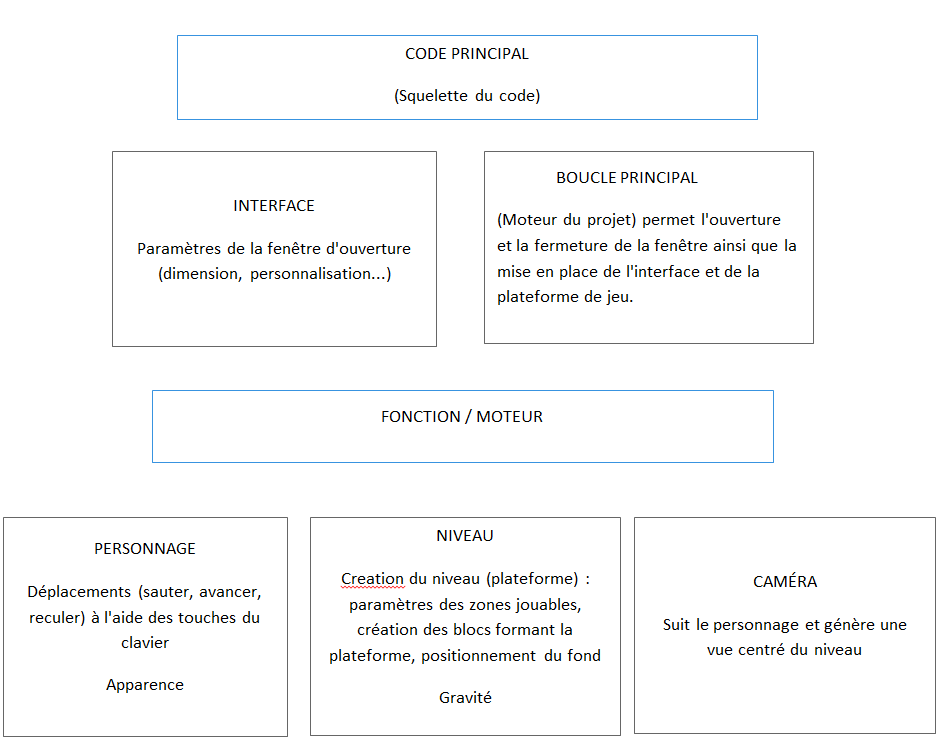
Cependant, au final, nous nous sommes heurtés à un problème : notre premier code ne marchait pas vraiment, nous n’arrivions pas à intégrer une gravité et une caméra suivant le joueur. Nous sommes donc allées chercher les réponses sur internet et sommes tombées sur un code « moteur » qui regroupait tous 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 :**



sur le personnage

(ensemble)

La gravité que subit le joueur

La collision entre le joueur et la plateforme

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

La répartition était en résumé ainsi :

|  |  |  |
| --- | --- | --- |
| **INES** | **SOLENA** | **DANA** |
| Interface du jeu (aka menu)  Appel d’un second code (Moteur\_SUPERDINO1) dans le code principal  Création de la plateforme dans la fonction | 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  Travail de la collision  Travail de la gravité (dans le moteur) | Mouvement des personnages (dans le moteur de SUPER DINO 1 et SUPER DINO 2)  Travail de la caméra |

**Mode de collaboration :**

Afin de communiquer et modifier plus facilement notre programme codé en langage 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 travaillé 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 nos travaux lors des moments où nous devions avancer et avons aussi utilisé Skype parfois.

**III- Réalisation personnelle**

Mon premier travail sur notre jeu vidéo a été tout d’abord d’apprendre les rudiments du fonctionnement d’un jeu vidéo. Nous nous lancions dans un projet qui nous était encore inconnu et il était nécessaire que l’on comprenne la mécanique générale d’un tel programme. Pour cela, je suis allée regarder ainsi que Solena et Dana des vidéos de personnes expliquant la création d’un jeu vidéo ainsi que des tutoriels sur OpenClassroom. Cette étape nous a permis de comprendre les principales fonctions et étapes nécessaires à la création de notre jeu. Nous avons pu, ainsi, nous répartir les différentes tâches.

Je me suis principalement occupé de la structure et du code principale pour la première version du jeu Super Dino 1.

Super Dino 1

1. **L’accueil du jeu :**

Les premières lignes de code de notre programme consistaient à réaliser des boucles while. En effet, le jeu est une grande boucle conditionnelle qui tant qu’elle est en marche, ses constantes possèdent la valeur 1 puis pour la quitter elles prennent la valeur 0. Ainsi, chaque partie de notre jeu, l’interface d’accueil, le gameplay (partie où l’on déplace le personnage) sont des boucles while dont les constantes ont toutes été initialisée à 1.

La première étape de la conception de notre code était d’afficher une fenêtre, celle qui allait contenir et afficher ensuite l’ensemble de notre projet. Nous avons eu recours, comme pour beaucoup de choses à Pygame et ses fonctions. En effet, l’extension pygame est un outil très utile et essentiel pour la conception d’un jeu sous python. Elle nous permet d’utiliser des fonctions préconçues afin de pouvoir avancer dans le code. La première fonction utilisée a été pour afficher une fenêtre dont la taille nous convenait. Cette étape a été très simple.

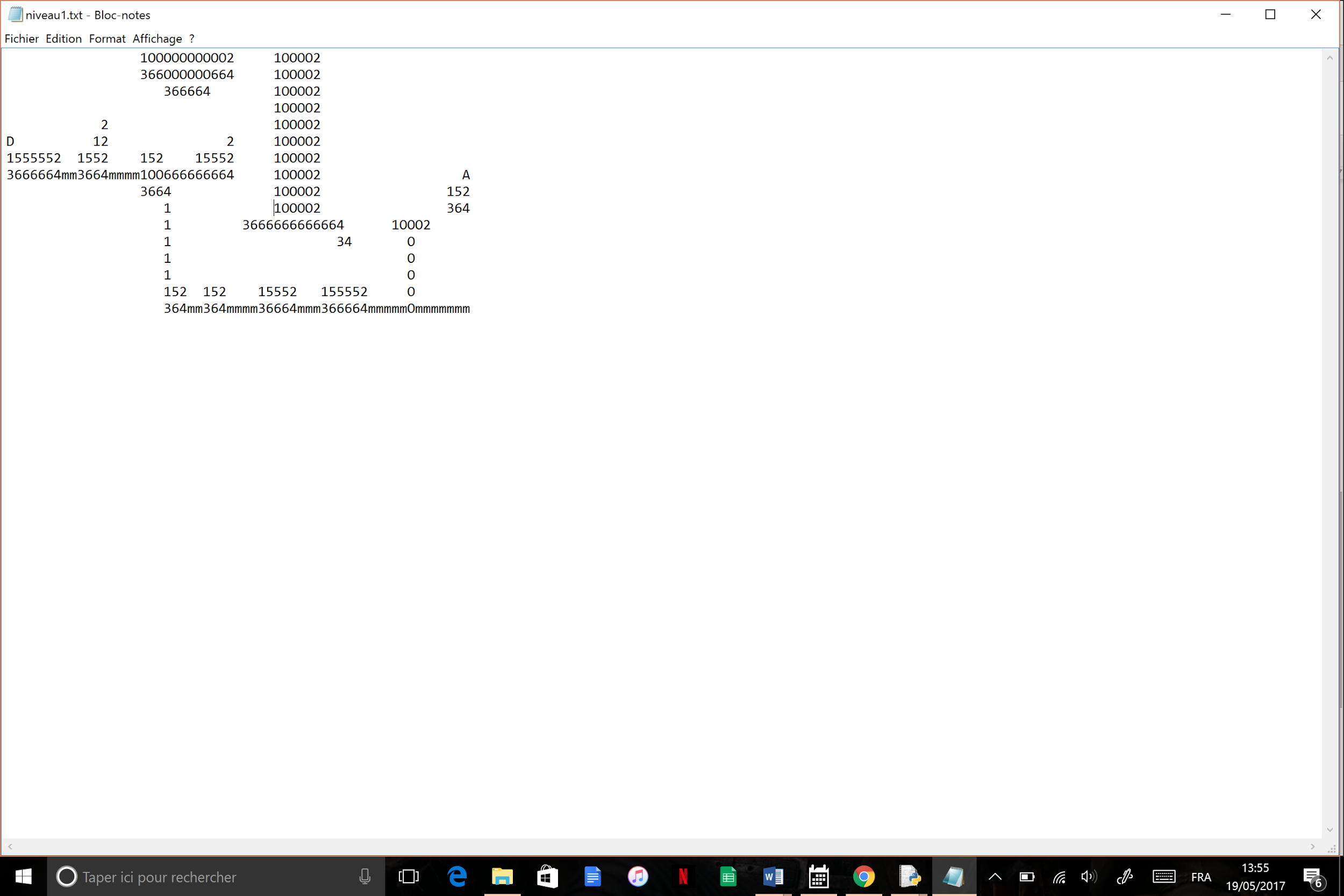
De même, il fallait pouvoir la fermer sans que le programme plante, nous avons alors créé une fonction « quitter() » qui permettait que lorsque l’on clique sur la croix en haut à droite de la fenêtre, les constantes du jeu sont toutes mises à 0 et le jeu se ferme avec la fenêtre.

L’étape suivante qui nous a posé nos premiers problèmes étaient d’afficher des choses à l’écran. Pour notre interface d’accueil, il faut afficher une image de fond. Au départ, elle était composée de plusieurs image, une image de fond, une image pour le bouton start et enfin le titre du jeu. Il était donc nécessaire de gérer plusieurs images en même temps.

Gérer plusieurs images pour ne pas qu’elles s’affichent sur les mêmes plans requiert l’utilisation des fonctions blit() et update() de pygame. Ces fonctions permettent respectivement de copier une image sur une autre et de la rafraîchir la fenêtre. Cependant, nous avons finalement décidé de fusionner toutes les images afin de n’avoir à en gérer qu’une seule. De ce fait, l’accueil est très simple et sobre et ne tient qu’en quelques lignes de codes. La seule interaction du joueur avec la fenêtre est que lorsqu’il appuie sur la barre espace, la boucle d’accueil passe à 0, se ferme et le niveau se génère et s’affiche.

Pour cela, j’ai utilisé une boucle if telle que SI le joueur appuie sur la touche espace, alors le niveau se lance.

1. **La création du niveau**
2. **Le fichier texte :**

Pour créer notre niveau, carte de jeu, il a fallu créer un fichier texte. Dans ce fichier texte, toutes les plateformes ont été codés par des lettres dans une zone délimitée de 58 sur 16 caractères. 

Les plateformes sur lequel se déplace le personnage sont codés par les numéros de 0 à 6 en fonction de l’image graphique utilisée, du sprite (un sprite est dans le jeu vidéo un élément graphique). Les zones marquées par un “m” représente les zones où si la position du personnage prend la position du “m” alors le jeu s’arrête et le joueur a perdu.

Pour faire lire ce fichier texte à notre programme, il a fallu créer une nouvelle Classe “Niveau”.

1. **La Classe Niveau :**

Faire lire un fichier texte à Python requiert l’utilisation des fonctions open() et read, tout en insérant l’adresse du fichier à lire.

Pour que Python puisse lire et traduire le fichier texte avec nos éléments graphiques, on a transformé notre fichier en tableau. Chaque ligne de texte et chaque caractère a été transformé en une case de notre tableau. Ainsi, grâce à une boucle for on se balade sur le fichier texte en créant le tableau et à chaque retour à la ligne “\n” on ajoute une ligne au tableau.

Une fois le tableau créé, on lit le tableau ligne par ligne de gauche à droite, on remplace chaque numéro (0 à 6) par l’image graphique correspondante avec la fonction blit() de pygame.

Remarque : Cette classe a été créé dans un fichier python séparé appelé “fonctions v1” car ceci permettait une meilleure lisibilité dans notre travail, d’avoir un code global le plus structuré possible et ainsi le plus clair possible. Ce fichier fonction est ensuite appelé au tout début de notre code afin d’utiliser les fonctions qu’ils renferment.

Afficher le niveau à l’écran :

Pour pouvoir afficher notre niveau à l’écran, il faut l’appeler dans le code principal du jeu.

Pour cela on crée une nouvelle boucle avec un constante à 1 qui tournera tant que le joueur n’a pas perdu et que le niveau est ouvert. Pour cela on définit le paramètre de notre Classe qui n’avait pour l’instant qu’un paramètre x appelée self.

Ainsi on appelle notre paramètre choix et on lui assigne le fichier texte de notre niveau. On remplace ensuite le paramètre self de notre classe par le paramètre choix : Niveau(choix). On va maintenant utiliser les fonctions de notre classe générer() qui correspond à la création du tableau puis afficher() qui correspond à la transformation des chiffres en sprite.

On rafraîchit notre fenêtre et le niveau s’affiche.

1. **Afficher le personnage dans le niveau :**

Maintenant que le niveau s’affiche à l’écran il faut afficher le personnage.

Notre personnage est également un sprite. C’est une image à laquelle on va attribuer des déplacements. Pour cela, on doit afficher l’image du dinosaure. Il faut ensuite créer un rectangle de la même dimension que l’image afin de la faire se déplacer sans découper le fond d’écran. Avec la fonction blit() on place ensuite l’image du perso dans ce rectangle.

**IV – Problèmes, erreurs et résolutions**

Au cours de la réalisation du code nous avons rencontré plusieurs problèmes, minimes et parfois plus important qui nous ont grandement ralenti dans la réalisation du projet.

Tout d’abord, aucune de nous n’avait jamais réalisé de jeu vidéo avant ce jour et il a fallu apprendre sur le tas en fonction de dont nous avions besoin.

Le premier problème a été d’installer Python et Pygame, Pygame étant une extension de Python il fallait les installer ensemble, dans le même dossier mais surtout que les deux soient compatibles. La première a été occupé à cette tâche car nous avons eu du mal à voir les différentes versions et surtout savoir laquelle nous serait le plus utile.

Au tout début de l’écrire du code nous avons beaucoup eu recourt à des tutoriels (lien en Annexe) pour comprendre les mécaniques du moteur du jeu. Ainsi on adaptait ce que nous apprenions avec nos images (principalement pour l’écran d’accueil).

Toutefois c’est au moment d’afficher le niveau que nous avons eu le plus de problème ainsi qu’avec les mouvements du personnage.

Pour afficher le niveau, après l’avoir créé en fichier texte nous n’arrivions pas à le faire lire par Python. C’était en réalité un problème d’extension du fichier et également d’adressage. Le fichier qui contenait du texte n’avait pas d’extension .txt et donc n’était pas reconnu par le système, de même que dans son adressage nous nous étions trompé de barre slash en mettant des « \ » au lieu de « / ».

Les plus gros problèmes ont été pour faire apparaître le personnage et le faire bouger. Nous ne savions pas comment faire, nous savions comment faire apparaître l’image mais non comment lui attribuer des mouvements. De plus, notre code pour les mouvements était beugué.

Afin de solutionné notre problème, nous avons dû rechercher dans d’autres codes similaires. C’est à ce moment où nous avons trouvé le moteur qui nous a finalement servi à réaliser une version supérieure de Super Dino.

En explorant différents codes et en les faisant lire à des personnes plus expertes que nous en informatique nous avons commencé à voir nos erreurs.

La première étape de débogage était de tester notre jeu ligne par ligne en ajoutant un timer avec une valeur très rapide (30ms) afin qu’il teste toutes les lignes de codes et que l’on sache exactement à quel moment de notre code le programme s’arrêtait de tourner. De cette façon nous avons pu voir toutes nos erreurs, notamment le fait que les mouvements du personnage ne pouvaient pas être attribué seulement à l’image mais à un carré créé sous celle-ci car dans le cas contraire plusieurs images se créaient. Un autre problème est que lorsque que l’on fait interagir une image avec d’autre comme c’est le cas ici (le personnage, les plateformes, le fond), il est nécessaire de réafficher constamment le niveau et de rafraîchir toutes les images à chaque mouvement du personnage.

**V- Bilan et perspectives**

Au moment de choisir notre projet d’ISN, Solena, Dana et moi-même avions toutes les trois penser à réaliser un jeu vidéo, cependant, aucune de nous trois avions réalisé la difficulté de ce travail surtout pour le style de jeu que nous avions décidé de réaliser. Si au début, la tâche ne semblait pas insurmontable mais requérait surtout une bonne connaissance de l’extension pygame, les problèmes ont commencé lorsqu’il a fallu créer notre personnage.

En classe, il était difficile d’avancer dans les 2 derniers mois car les cours consistaient principalement à rechercher des tutoriels, des aides afin de régler nos problèmes. Je pense que nous manquions aussi de travail personnel à la maison et peut-être d’un manque de dialogue entre les membres du groupe. Nous avons commencé que trop tard à réellement travailler notre code et nous avons été prises de court dans les dernières semaines. C’est une des raisons pour laquelle nous avons décidé de nous aider du moteur pour la création de Super Dino 2 afin de rendre projet fini et amélioré.

Cependant je ne voulais pas laisser tomber notre premier code et j’ai continué a tenté de le débeuguer avec l’aide de Dana sur les mouvements.

Finalement nous avons réussi à aboutir à deux jeux vidéo, bien que l’un bien moins évolué que l’autre, opérationnels.

Ce projet apporté, autant sur le plan des connaissances car j’ai appris beaucoup sur Python et notamment tout ce qui concerne la gestion des Classes et les fonctions, que sur le travail de groupe et le travail en complète autonomie et « self-learning ». Ce projet m’a obligé à apprendre par moi-même de nouvelles connaissances en un laps de temps très restreint et à pouvoir restituer ces connaissances le plus efficacement possible. Je pense avoir contribuer beaucoup au projet et même si travailler en groupe possède ses difficultés, nous avons su gérer les obstacles et trouver des solutions sans aucuns conflits au sein du groupe. De plus, l’ISN nous a aussi appris à collaborer avec les autres groupes, qui même s’ils avaient des projets complètement différents des notres, pouvaient nous apporter leur aide et nous permettre à tous d’avancer.

**VI – 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>

Vidéos utilisées :

<https://www.youtube.com/watch?v=bxibfL4TjWo&t=6s>

<https://www.youtube.com/watch?v=mTmJfWdZzbo>