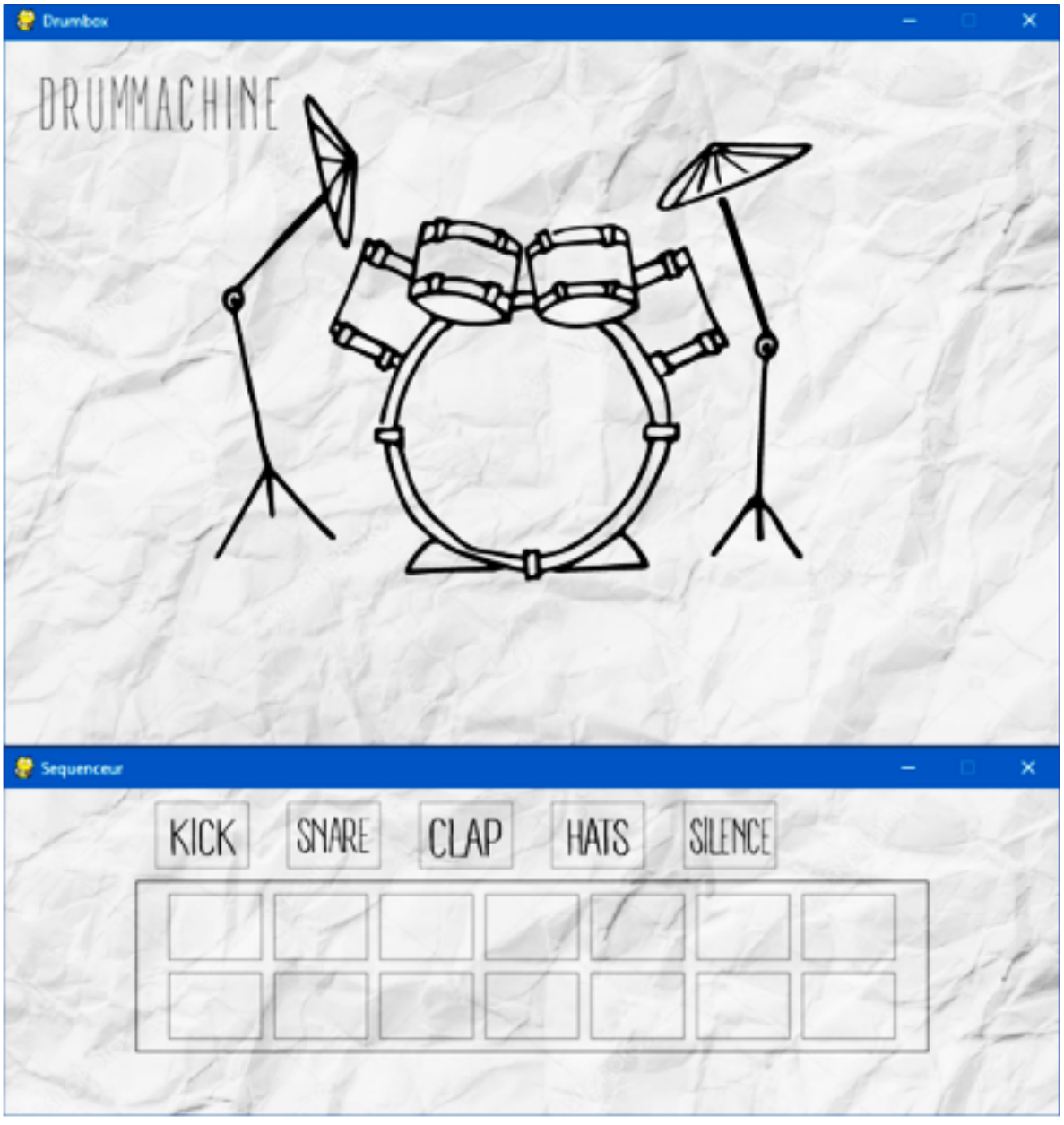
PROJET ISN : DRUMBOX



ROSINA Léo, CALLEWAERT Jules, MARTIN-LOUMEAU Adèle
Terminales S

Sommaire

I/ Présentation du projet :

La DrumBox est le projet que nous avons réalisé durant cette année de Terminale dans le cadre de notre enseignement de spécialité ISN. Nous y avons rencontré de nombreuses difficultés et consacré une grande partie de notre temps, et nous sommes fiers du résultat. Notre DrumBox est une batterie virtuelle à laquelle on peut jouer en pressant les touches du clavier de notre ordinateur, et qui peut lire une partition (liste d'instruments) que l'on écrit nous-même en utilisant la souris.

A) Pourquoi ce projet ?

Lors de la création de notre équipe en janvier, l'idée d'un projet autour de la musique nous trottait déjà en tête. En effet, nous sommes tous trois de grands passionnés de musique, et en présence de ce fort point commun, le projet d'une drumbox nous est rapidement apparu comme étant un choix judicieux.

Aujourd'hui, la musique est très liée au numérique et l'industrie musicale utilise beaucoup l'outil informatique, et sa maîtrise peut être un atout dans ce secteur.

De plus, chacun de nous trois ayant déjà observé ou même pratiqué de la MAO (Musique Assistée par Ordinateur), il était très enrichissant pour nous de comprendre les rouages derrière les logiciels de MAO. Nous voulions découvrir comment la musique peut être codée avec Python, et nous voulions explorer, non seulement la façon dont on associe une action sur l'ordinateur (clavier ou souris) à un son, mais également celle dont on crée un son à part entière, à l'aide de principes de physique comme les fréquences.

Ainsi, la problématique de notre projet est : de quelles différentes manières peut-on générer et contrôler un son grâce à l'outil informatique et au langage python?

C) Cahier des charges :

Objectifs du projet:

Même si nous avions appris depuis le début de l'année à utiliser Python, nous savions que nous aurions beaucoup à apprendre sur la gestion du son dans ce langage. Mais il est vrai que nous nous lancions quelque peu dans l'inconnu, c'est pourquoi nous avons beaucoup changé d'avis au cours du projet (suppression et ajout de fonctionnalités).

Nous avions été légèrement ambitieux au début : nous voulions pouvoir enregistrer les morceaux que l'on créait. Il est apparu très vite que cela demandait des connaissances trop poussées et que nous manquions d'informations, et qu'il était plus judicieux de se concentrer sur d'autres fonctionnalités. Finalement nos concertations dans le groupe ainsi qu'avec notre professeur nous ont tournés vers deux nouvelles fonctionnalités très intéressantes : le clavier virtuel et le séquenceur. Le séquenceur est une fonctionnalité qui nous a poussé à mieux comprendre le fonctionnement du son, mais surtout nous a permis de découvrir la gestion des événements de souris. Le clavier virtuel nous a appris à coder des sons à partir de données numériques.

Notre programme consiste en une drumbox, un clavier virtuel ainsi qu'un séquenceur, et nous accédons à ces fonctions par un menu principal.

DrumBox :

- Quatre sons de batterie (grosse caisse, caisse claire, charleston, cymbale crash)
- Image d'une batterie dont chaque instrument change de couleur lorsqu'on appuie sur la touche qui lui est attribué
- Utilisation des touches A, Z, Q, S.

Clavier virtuel :

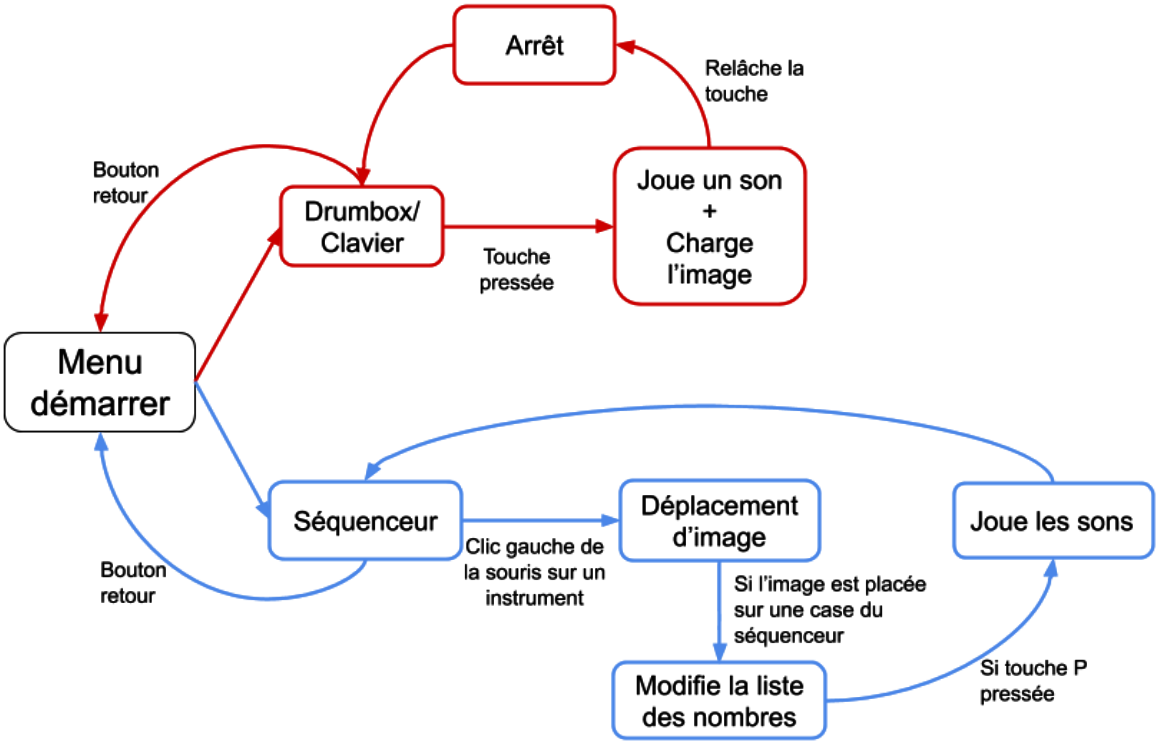
- Notes de clavier qui dépendent d'une fréquence (de DO à LA)
- Affectation de chaque touche à une note, de F à L
- Image de clavier dont chaque touche change de couleur lorsqu'on appuie sur la touche qui lui est attribuée.
- Matériel : Adobe Illustrator (design), modules : numpy et sounddevice

Séquenceur :

- 5 cases avec sur chacune un instrument de la batterie + une case "silence"
- 14 "boîtes" où placer les cases
- affectation de la touche "p" au séquenceur qui joue la suite de cases en fonction de l'ordre dans lequel on les a placées dans les boîtes.

Voici le matériel et les sources que nous avons utilisé:
Plateforme de partage : github (facilite la communication grâce au tableau de bord et le partage de documents avec les "branches")
Logiciel : Thonny (simple d'utilisation et assez performant pour des projets d'une longueur comme le nôtre, possède beaucoup de modules python et permet d'en installer très facilement)
Bibliothèques python : Pygame, Sounddevice, Numpy, Time

STRUCTURE DU PROGRAMME :



Voici ci dessus le schéma représentant l'algorithme de notre programme. Il est donc représenté en deux parties, la drumbox/clavier en rouge et le séquenceur en bleu. Nous n'avons pas réussi à coder le bouton retour, néanmoins nous avons décidé de le laisser sur le schéma car cela correspond à ce que nous voulions faire à l'origine.
On peut voir que l'algorithme comporte des boucles, il n'y a donc pas de "fin" au programme.

POSITIONNEMENT PAR RAPPORT À D'AUTRES SOLUTIONS DÉJÀ EXISTANTES:

Il existe déjà de nombreux logiciels qui permettent de jouer de la musique (le plus connu est "garageband", mais il y a aussi des logiciels de MAO comme Fl Studio, Logic Pro). Ils sont très développés et performants, et possèdent une multitude d'instruments,

d'effets sonores, et des fonctionnalités comme l'enregistrement. Sans oser prétendre faire face à ces logiciels, notre projet s'en démarque tout de même premièrement car il est plus spécialisé (batterie et clavier) et peut servir à des utilisateurs qui ne recherchent que ces fonctions. De plus, les logiciels de MAO sont assez compliqués à utiliser, alors que notre projet est très simple à manier ! Enfin, notre démarche de création du son par différents moyens est originale et constitue un atout par rapport aux autres logiciels qui n'utilisent souvent que des sons enregistrés ou que des données numériques.

B) Répartition des tâches :

JOURNAL DE BORD DU PROJET ET RÉPARTITION DES TÂCHES :

Dates	Léo	Adèle	Jules
10/01/2019	Invention du nom de l'équipe; Début d'idées sur le projet: appuyer sur les touches de clavier afin de produire un son de batterie/percussion; Donner à l'utilisateur la possibilité de s'enregistrer et de rejouer la piste précédemment enregistrée.		
17/01/2019	Recherche des sons de batteries; Imagination du profil qu'aura la fenêtre; Début d'apprentissage d'outils sur Python afin de jouer des sons		
24/01/2019	Design de la batterie avec des images.png	Recherches sur la possibilité d'enregistrer des sons que l'on produit nous-même	Apprentissage de la commande liée à la pression et au relâchement d'une touche
31/01/2019	Apprentissage de des tkinter puis abandon de cette bibliothèque car la gestion des événements est compliquée	Apprentissage sur Pygame (avec Pyaudio et le package Numpy) de comment enregistrer un son à partir d'un fichier	Mini programme de test afin de jouer des sons lorsque l'on appuie sur une touche avec des problèmes
7/02/2019	Début du code de la Drumbox à l'aide du petit programme de Jules	Abandon de l'enregistrement. Exploration du package Numpy. Début de la création artificielle de sons sur des fréquences différentes.	Réussite du mini programme, celui-ci joue des sons enregistrés dans le fichier lorsqu'on appuie sur une touche. Début du codage de la Drumbox.
Du 7/02 au 14/03	Durant les vacances nous avons chacun avancé sur notre partie du programme		

14/03/2019	Finition du code de la drumbox	Association de touches à une fréquence pour jouer des notes de clavier.	Finition du code de la drumbox
4/04/2019	Entretien avec Mr Jules, il nous fait constater que notre programme ne présente pas tellement de difficultés, et nous propose l'idée d'une fonctionnalité de séquenceur.		
18/04/2019	Début de la réflexion autour du séquenceur		
22/04/2019	Début de la programmation du séquenceur. Création de l'interface graphique à l'aide d'Adobe Illustrator. Découverte de la gestion des évènements d'une souris.	Début de la programmation du séquenceur. Découverte de la gestion des évènements d'une souris.	Début de la programmation du séquenceur. Découverte de la gestion des évènements d'une souris.
23/04/2019	Fonctionnalité de glisser/déposer codée, finitions de l'interface graphique du séquenceur.	Elaboration du menu principal (design des touches + élaboration des événements de la souris correspondant aux touches du menu).	Suite de la programmation du séquenceur avec Léo.
26/04/2019	Changement du design de la drumbox. Fin de la programmation du séquenceur avec la fonction d'analyse de la liste de nombres jouant. Insertion du code du séquenceur dans le menu principal avec la drumbox	Fusion des codes du menu principal et de celui de la drumbox.	Rangement et organisation du dossier github.

Démarche collaborative:

Nous avons bien su collaborer, et une vrai dimension de confiance et d'entraide s'est installée. Léo étant très doué en design, Jules qui s'occupait des sons enregistrés et Adèle de la partie mathématique et physique du projet, nous avons pu échanger nos connaissances et partager nos compétences. Nous avons utilisé Github mais aussi Google Drive et Messenger pour faire des points réguliers sur l'avancée du projet.

III/ PARTIE PERSONNELLE :

J'ai contribué à ce projet en travaillant à la fois sur une partie du code de la drumbox et sur le code du séquenceur. J'ai également réalisé le design du programme.
La partie sur laquelle j'ai passé le plus clair de mon temps est la partie séquenceur. En effet, c'était un code bien plus compliqué à coder que celui de la drumbox, car la partie graphique ne se résumait pas simplement à des changements d'image, mais à des déplacements d'images. De plus, il fallait que j'apprenne à gérer les événements de la souris (clic gauche par exemple), cela étant plus compliqué que des événements des touches. J'ai utilisé le module Pygame pour le séquenceur, ainsi que le module Time, qui comme son nom l'indique permet de gérer tout ce qui a un rapport avec le temps.

A) Analyse du programme

Voici une partie du programme du séquenceur que j'ai codé :

```
337 def MainSequenceur():
338     global selected_kick, selected_snare, selected_clap, selected_hats, selected_silence
339     ChangementImgSequenceur()
340 # boucle déplacement d'image pour le KICK
341     if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1 and kick_position.collidepoint(event.pos):
342         selected_kick = 1
343 
344     elif event.type == pygame.MOUSEBUTTONUP and event.button == 1 and selected_kick == 1:
345         posx[0],posy[0] = pygame.mouse.get_pos()
346         BoutonRelacher(posx[0],posy[0], ImgKick_Sequenceur)
347         selected_kick = 0
348         Detection_Sequenceur(kickson, ImgKick_Sequenceur, posx[0], posy[0])
349 
350     elif event.type == MOUSEMOTION and selected_kick == 1:
351         SourisMouvement(ImgKick_Sequenceur, kick_position)
```

J'ai choisi cette partie du programme car elle me permet d'illustrer un problème que j'ai rencontré tout au long du projet, lié aux fonctions. De plus, c'est une partie de la fonction principale du séquenceur. La partie "# boucle déplacement d'image pour le KICK" se répète pour chaque instrument, le SNARE (caisse claire), le CLAP (applaudissement), le HAT (charleston), et le SILENCE. Il est donc judicieux de ne s'intéresser qu'à cette partie.
J'ai réalisé cette fonction à l'aide de plusieurs tutoriels qui traitaient de la gestion des événements de souris. En compilant les codes et en rajustant pour que cela fonctionne

avec notre programme, j'ai finalement aboutit à cette fonction qui peut néanmoins être amélioré.

Je vais donc détailler la fonction :

`-global` : ce mot-clé de python permet de lire et de modifier une variable globale (ici selected_kick, selected_snare, etc) dans une fonction. Je détaille plus précisément le rôle de cette fonction python dans la partie Problèmes.

`-selected_instrument` (ex: `selected_kick`) = cette variable est égale à zéro pour chaque instrument, et devient égale à un lorsque l'instrument est sélectionné par la souris. Cela permet de détecter lorsqu'un instrument et sélectionné.

`-ChangementImgSequenceur()` est une fonction qui définit l'interface graphique du séquenceur.

`-event.type` est une variable pygame qui détecte l'événement réalisé par l'utilisateur.

`-pygame.MOUSEBUTTONDOWN` est la variable correspondant à l'événement bouton souris pressé.

`-event.button == 1` correspond à l'événement clic gauche de la souris.

`-kick_position.collidepoint(event.pos)` -> `kick_position` est la variable associée à l'image du kick et `.collidepoint(event.pos)` est une fonction de pygame qui permet de détecter les coordonnées exactes de l'endroit où l'on clique sur l'image avec la souris.

`-pygame.MOUSEBUTTONUP` est la variable correspondant à l'événement : bouton souris relâché.

`-posx[0], posy[0] = pygame.mouse.get_pos()` -> `posx` et `posy` sont les coordonnées de l'image, qui ici grâce à `pygame.mouse.get_pos()`, prennent les valeurs des coordonnées du curseur de la souris lorsque le bouton gauche est relâché.

`-BoutonRelacher(posx[0], posy[0], ImgKick_Sequenceur)` est une fonction qui déplace l'image de l'instrument qui a été sélectionné, vers l'endroit où se trouve le curseur de la souris lorsque le bouton gauche est relâché. Les arguments de la fonction sont donc la position de la souris ainsi que l'image de l'instrument.

`-Detection_Sequenceur(kickson, ImgKick_Sequenceur, posx[0], posy[0])` -> fonction qui modifie la liste de nombres qui constitue le séquenceur. `kickson` est la variable qui correspond au nombre du kick. Chaque instrument à un nombre qui lui est associé (kick = 1, snare = 2, etc...). La liste est ensuite lue et décodée puis joue les sons, par une autre fonction, la fonction *SequenceurBoutonPressé()*.

-MOUSEMOTION est la variable qui correspond à l'événement mouvement de la souris.

-SourisMouvement(ImgKick_Sequenceur, kick_position) est la fonction qui "aimante" l'image sélectionnée au curseur de la souris.

Je vais la détailler :

```
290
291 def SourisMouvement(instrument, instrument_position):
292     r = screen.blit(screen, instrument_position, instrument_position)
293     instrument_position.move_ip(event.rel) #a chaque fois il y a mouvement de souris,
294     pygame.display.update((r,screen.blit(instrument, instrument_position)))
295
```

-r est une variable qui prend pour valeur l'image et la position de l'image

-instrument_position.move_ip(event.rel) -> instrument_position correspond à la position de l'image de l'instrument, .move_ip permet de déplacer l'image. ip correspond à "in place" est permet de remplacer les images précédentes. Si nous avions seulement mis .move, l'image se serait déplacer mais en se dupliquant. Enfin, (event.rel) sont les coordonnées du curseur de la souris.

-pygame.display.update((r,screen.blit(instrument,instrument_position)) met à jour l'image lors du déplacement.

B) Problèmes rencontrés

Ainsi, comme je l'expliquai au dessus, la création de cette fonction MainSequenceur() s'est avéré très compliqué. Un des problèmes majeurs venait de mes variables selected_instrument. Je rencontrai le problème suivant :
UnboundLocalError: local variable 'selected_kick' referenced before assignment

En ajoutant les variables qui posaient problème à la suite de global, j'ai résolu le problème.

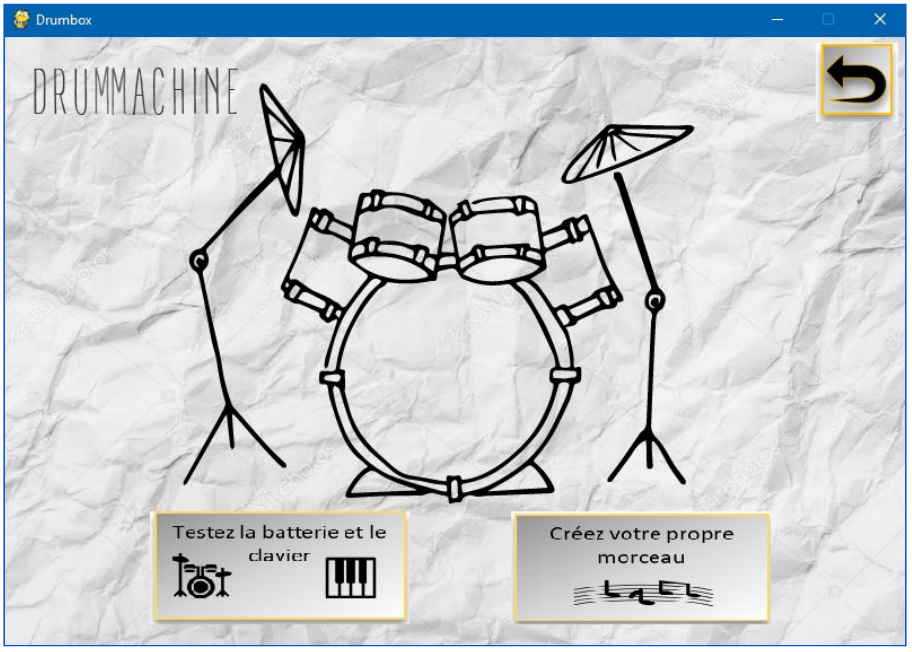
Cependant un problème que je n'ai pu résoudre fait aussi parti de cette fonction. Et c'est encore un problème de fonction. En effet, on peut remarquer que la partie boucle déplacement d'image pour le kick est quasiment la même pour chaque instrument. Mr Jules m'avait signalé qu'une fonction pourrait simplifier ce code. Je me suis donc mis à la création d'une fonction comportant toutes les variables nécessaires, sans variables locales. Après plusieurs essais, corrections de bugs et d'erreurs, je n'avais plus d'erreur et le programme se lançait.

Néanmoins, on ne pouvait plus déplacer les images. J'ai donc entrepris de recommencer une nouvelle fois mais ce fut un échec. Par manque de temps je n'ai pas réessayé encore une fois mais je sais pertinemment que cette fonction peut être réalisée.

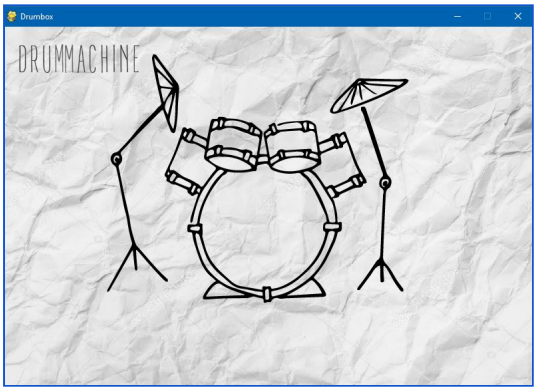
Un autre problème rencontré bien auparavant est un problème lié au son. En effet, lorsque l'on a commencé à coder la drumbox, on a vite dû utiliser des échantillons de sons pour les instruments. Or aucun ne fonctionnait. Le problème venait du fait que les échantillons étaient codés en 32 bits alors que pygame n'accepte pas de sons dépassant 16 bits. J'ai donc converti les échantillons de 32 bits en 16 bits avec le logiciel gratuit Audacity. Le problème fut résolu.

C) Intégration et validation

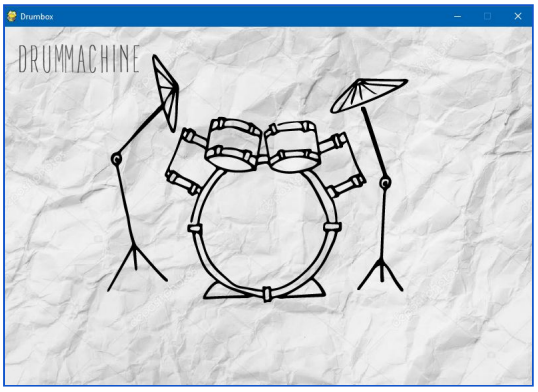
Page d'accueil :



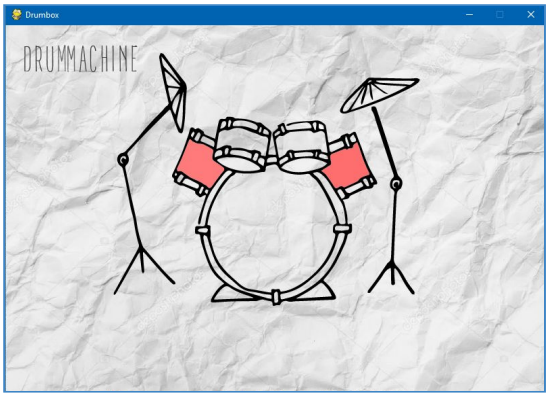
Quand on appuie sur le bouton "Testez la batterie et clavier" on lance la drumbox :



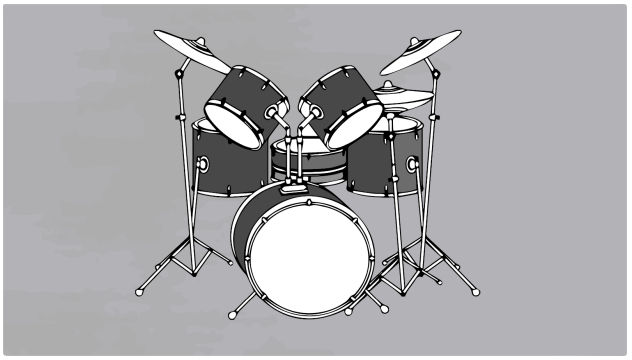
Quand on appuie sur une des touches de F à L



Quand on appuie sur une des touches entre A,Z,Q et S (sur l'image c'est A qui est pressée)

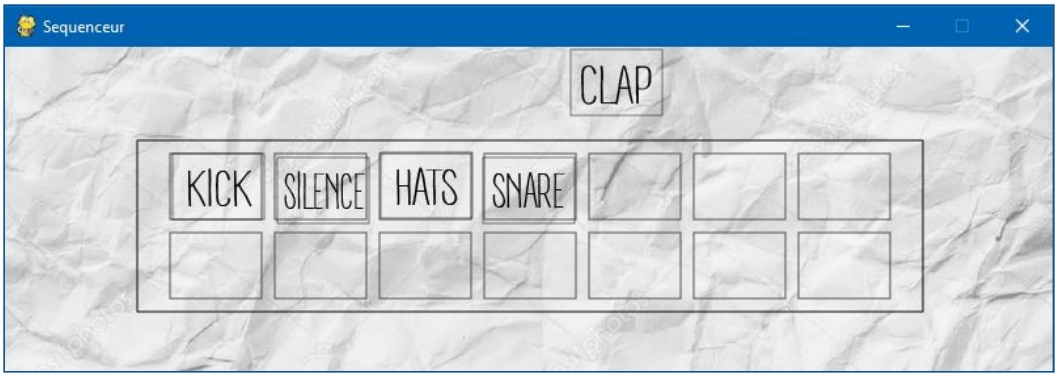


Mon travail sur la partie Drumbox concerne essentiellement la partie graphique et l'animation. En effet, cela étant ma tâche d'origine, j'ai très vite entrepris de faire un design original et assez épuré. Voici l'ancien design que j'ai grandement retravaillé pour qu'il s'intègre mieux avec notre projet, car nous voulions un design plus enfantin et plus original.



Quand on appuie sur le bouton "Créez votre propre morceau" on lance le séquenceur :

↓ On déplace les rectangles instruments sur le séquenceur



Pour la partie séquenceur j'ai également réalisé la partie graphique, en plus du code. Le séquenceur s'intègre très bien au programme complet et donne ainsi un intérêt supplémentaire à notre programme. J'ai utilisé la même police et le même arrière plan pour donner plus de cohérence à notre programme final.

Ainsi notre programme correspond en partie à notre cahier des charges. Les différents sons de la drumbox fonctionnent lorsque l'on presse les touches A,Z,Q et S, il y a quatre sons d'instrument. Cependant, nous n'avons pu mettre la fonctionnalité qui permet de choisir ses propres sons. La fonction d'enregistrement est malheureusement tombée à l'eau également car cela est très compliqué à coder avec Python. C'est souvent avec C++ que sont codés les logiciels de musique (voire d'autres langages encore).
Au niveau du séquenceur, il répond au cahier des charges, il y a bien 14 cases qui jouent les sons associés aux instruments.

D/ Perspectives d'amélioration:

Lacunes et problèmes à améliorer:

- fonctions, cases séquenceur, retour?
- Séquenceur : le majeur problème que nous devrions régler est le fait que lorsque l'on relâche la souris, les cases des instruments ne restent pas exactement dans leur boîte. Nous pourrions également améliorer le séquenceur en faisant en sorte de pouvoir placer nos instruments plusieurs fois dans les boîtes car pour l'instant, il ne sont utilisables qu'une seule fois. Nous pourrions ensuite ajouter le clavier au séquenceur et augmenter le nombre de boîtes pour créer des morceaux plus longs et plus complexes.

Ajout de fonctions:

- Fonction d'enregistrement: ceci était à l'origine une fonction à la base de notre projet, mais suite à la complexité de cette fonction et au fait que nous avons privilégié les nouvelles idées de clavier virtuel et se séquenceur, nous l'avons rapidement abandonnée. Il aurait été cependant très intéressant de l'inclure à notre projet, dans l'optique de pouvoir enregistrer des morceaux à la batterie et au clavier, et avec en musique de fond la musique du séquenceur. L'enregistrement semble être possible grâce à la bibliothèque "sounddevice" et à la fonction "sounddevice.rec".

E/ Bilan personnel

Ce projet a été très enrichissant pour moi pour différentes raisons.

Tout d'abord il m'a permis de découvrir ce que c'était que la programmation. En effet, avant d'entamer le projet, on ne codait que de très petits programmes et le passage à un programme d'une autre ampleur change considérablement la manière de programmer. Par exemple, sur un petit programme, une erreur se retrouve facilement, il n'y a pas non plus besoin de fonctions. Notre programme faisant plusieurs centaines de lignes, il a fallu réfléchir aux moyens pour l'optimiser et sans fonction c'est impossible. C'est un exercice très intéressant car il nous pousse à être le plus synthétique possible.

D'autre part, ce projet m'a permis de prendre mes responsabilités et d'apprendre à travailler en équipe. En effet, le projet étant tout de même ambitieux, il était impossible de le réaliser seul. De plus, étant nous quatre des novices dans la matière, nous partions sur une même base et avons appris ensemble. Il y avait donc un esprit d'entraide et de partage au sein de notre équipe, ce qui nous a permis d'avancer à grand pas. D'un point de vu personnel, c'est très intéressant d'apprendre à plusieurs et de confronter les points de vu.

J'ai également appris à prendre du recul pour résoudre les problèmes. En effet, face à chaque problème j'imaginais des solutions qui fonctionnaient pour les petits problèmes. Pour les gros problèmes, je trouvais des solutions plusieurs jours après, quand je n'y pensais plus trop.

Ainsi, ce projet a été très enrichissant pour moi. Il m'a permis de mieux comprendre le fonctionnement de logiciels que j'utilise quotidiennement, et m'a permis de développer un esprit d'équipe.

Léo ROSINA