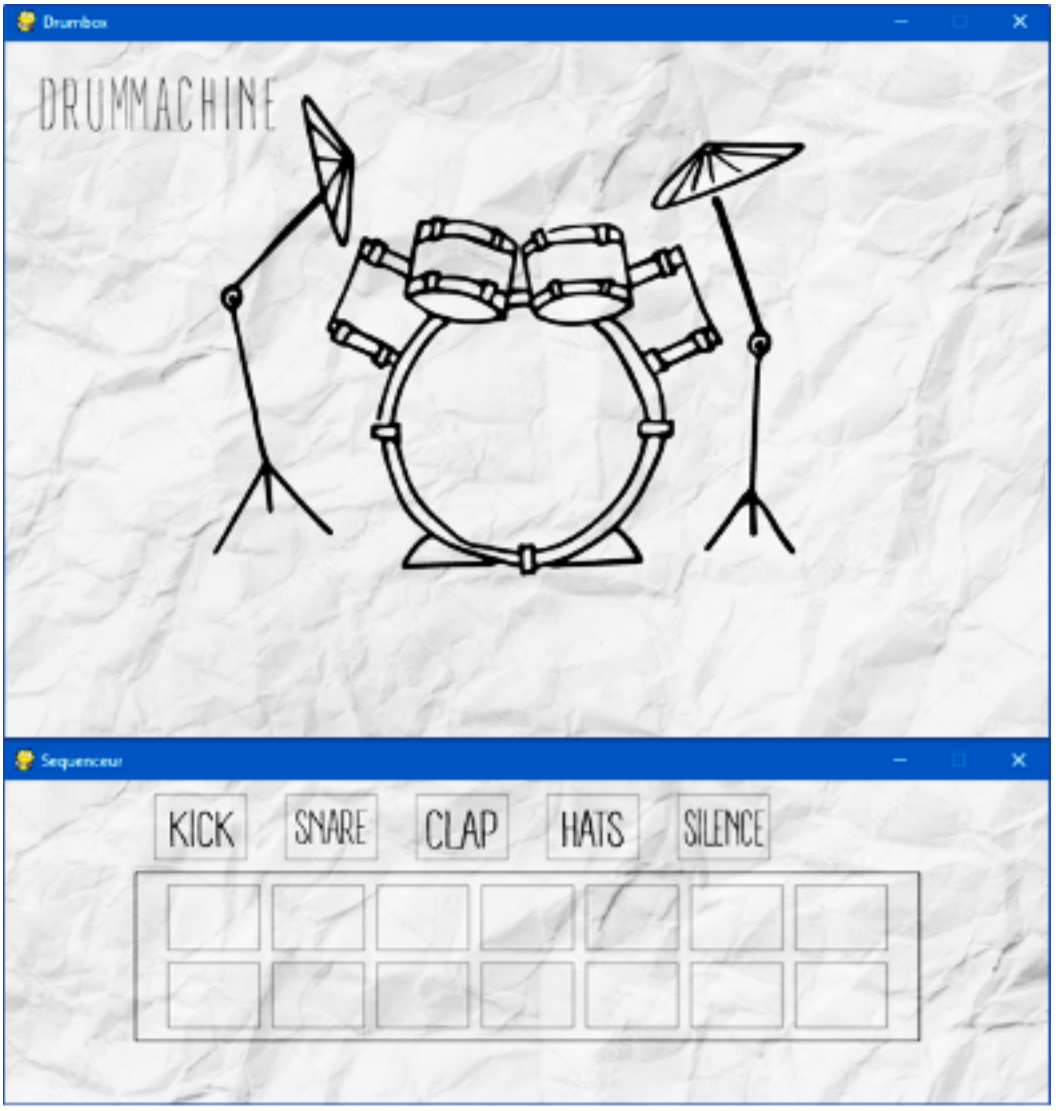
PROJET ISN : DRUMBOX



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?

b) 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 :

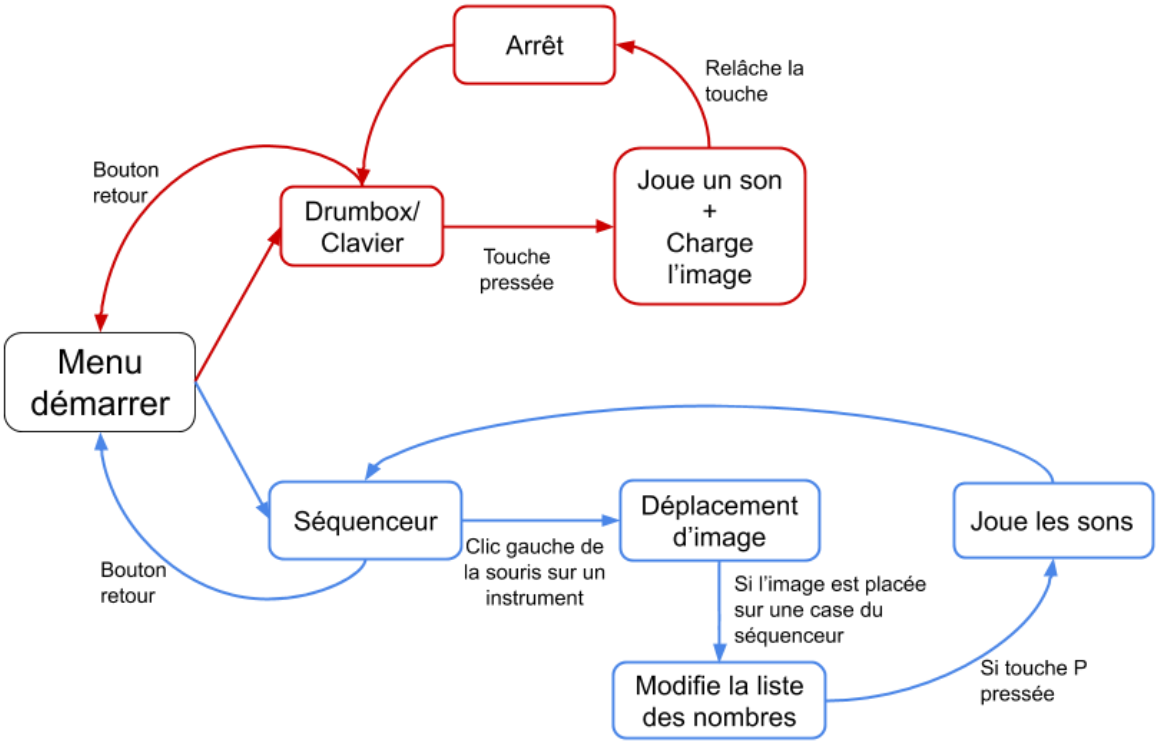
- Notes de clavier qui dépendent d'une fréquence (de DO à LA)
- Affectation de chaque touche à une note, de F à L
- Image de clavier dont chaque touche change de couleur lorsqu'on appuie sur la touche qui lui est attribuée.
- Matériel : Adobe Illustrator (design), modules : numpy et sounddevice

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 notre, possède beaucoup de bibliothèques python)
Bibliothèques python : Pygame, Sounddevice, Numpy, Time

STRUCTURE DU PROGRAMME :



POSITIONNEMENT PAR RAPPORT À D'AUTRES SOLUTIONS DÉJÀ EXISTANTES:

Il existe déjà de nombreux logiciels qui permette de jouer de la musique (le plus connu est "garage band", mais il y a aussi des logiciels de MAO comme Pro Tools, Sonar). 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	Début de la programmation du	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.	séquenceur. Découverte de la gestion des évènements d'une souris.	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, avec 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.

II/ Réalisation Technique personnelle:

A)Réalisation personnelle

Il y a deux parties principales du projet que j'ai réalisé : le clavier et le menu principal et son interaction avec le reste des fenêtres du projet (j'ai aussi participé au design du projet, aux événements souris du séquenceur ainsi qu'à l'élaboration de certaines fonctions comme KEYDOWN et KEYUP).

Clavier virtuel :

Mon rôle était à l'origine de m'occuper de la fonction enregistrement de la drumbox, qui a finalement été remplacée par le clavier artificiel, qui nous a paru plus intéressant. Lors de mes recherches sur la fonction enregistrement, je suis tombée sur deux bibliothèques python très intéressantes : numpy et pygame. J'ai découvert qu'elles permettaient de coder des sons à partir de données numériques (fréquence d'échantillonnage, fréquence des sons, durée, amplitude...). Je me suis dit qu'il serait intéressant d'intégrer cette dimension mathématique et physique à notre projet, et j'ai alors eu l'idée d'y intégrer un clavier virtuel. Après discussion, même si mes collaborateur n'étaient pas très motivés au départ, car l'on s'éloignait du projet initial, ils ont reconnu que ce concept serait intéressant et novateur.

Je me suis aidée de deux bibliothèques python : Numpy et Sounddevice. Numpy est très performant dans le traitement des données mathématiques, c'est pourquoi je l'ai utilisé pour des calculs précis comme ceux des sinus (à l'aide de la fonction np.sin) ou de π (grâce à np.pi).

Sounddevice est utile pour le traitement du son en direct: conversion des données numériques en sons, jouer en playback, enregistrer...

Je me suis aidée des principes de la numérisation du son. Le son est une énergie qui se propage sous forme de vibrations dans un milieu compressible, et comme tout phénomène vibratoire, il peut être analysé comme un signal qui varie dans le temps. On peut le ramener à des fonctions sinusoïdales, appelées signaux analogiques et qui ont comme caractéristiques la fréquence et l'amplitude, auxquelles j'ai assigné des valeurs :

```
190 def NoteClavier(frequence):#fonction qui traite les données numériques pour créer un son. Variable : fréquence du son.
191
192     # assignation de valeurs aux caractéristiques du signal analogique
193     amplitude = 10000
194     time = 1.0 # on définit le temps d'une note (1 seconde)
```

- *frequence* (variable) en Hz. Selon la valeur de la variable, le clavier va jouer une note différente.

- *amplitude* (10000). Elle correspond au volume sonore (plus précisément, il s'agit de l'expression de la pression de l'air). Elle se mesure souvent en dB. Cependant, j'ignore pourquoi mais le programme ne sort pas de son lorsqu'elle est en dB (comme lorsque je teste la valeur 60 pour 60 dB). J'ai donc testé de nombreuses valeurs et c'est la valeur de 10000 qui correspond au meilleur niveau sonore.
- *time* : durée du son (1 seconde).

```
196     # élaboration du signal numérique
197     echantillonnage = np.arange(0, time, 1/44100) # définition de la fréquence d'echantillonage
198     onde = amplitude * np.sin(2 * np.pi * frequence * echantillonnage)# création du signal numérique de formule : onde = amplitude*sin(2*pi*fréquence*échantillonage)
199
200     ond_onde = np.array(onde, dtype=np.int16) # conversion en format onde que l'on code sur 16 bits
201     sd.play(ond_onde, blocking=True) # on convertit le signal numérique en son
```

→ A partir du signal analogique, il est possible de le convertir en signal numérique. C'est ce signal numérique qui est lu par l'ordinateur et qui lui permet ensuite de reproduire un son réel.

- echantillonnage (qui correspond à la fréquence d'échantillonnage): J'ai choisi de lui attribuer une valeur de 44100 Hz car la qualité du son en est meilleure, car d'après Shannon, plus la fréquence d'échantillonnage est élevée, plus elle est fidèle au réel (ici, nous avons une qualité CD).
  La fonction "numpy.arange" a pour but de créer une liste avec des intervalles. La première valeur est 0, et le temps est la dernière valeur. La valeur 1/44100 est la période (l'inverse de la fréquence), et elle représente le pas entre les différentes valeurs de la liste. C'est donc cette fonction qui code pour la fréquence d'échantillonnage.

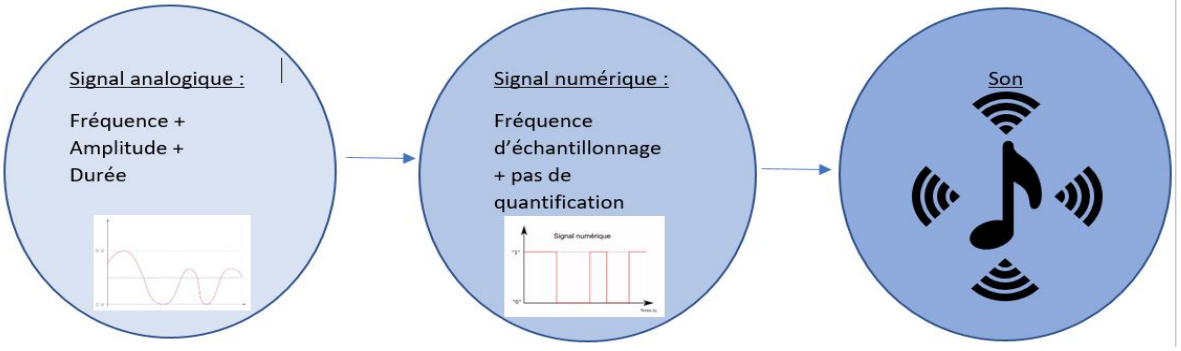
- onde (je lui ai donné ce nom car il s'agit d'une fonction sinusoïdale) : elle définit le signal numérique.
  Elle est définie par la fonction onde (fréquence d'échantillonnage) = amplitude * sin (2 * pi * fréquence * fréquence d'échantillonnage). Pour cette formule, je me suis inspirée du site https://www.scivision.co/playing-sounds-from-numpy-arrays-in-python/, qui cependant n'explicitait pas les variables de la fonction (il n'y avait que des valeurs sans explication). J'ai donc effectué de nombreux tests en changeant les valeurs pour comprendre comment était structurée cette formule. Je pense qu'outre la fréquence d'échantillonnage, elle utilise le pas de quantification (tension mesurée à chaque point d'échantillonnage), qui dépend de l'amplitude et de la fréquence du signal analogique qui sont dans la formule.

  - ond_onde : la fonction np.dtype16 (qui pourrait être traduit par "type de data sur 16 bits") a pour but de pouvoir coder le pas de quantification sur 16 bits, donc sur des entiers allant de -32768 à 32767. Cela permet au signal numérique d'être plus précis et donc fidèle au signal analogique.

  - sd.play(ond_onde, blocking = True) : permet de convertir les données numériques en son, c'est à dire en vibration de l'air par le biais de l'émetteur de l'ordinateur.

J'ai ensuite inséré cette fonction dans d'autres fonctions telles que KEYDOWN() qui associe une touche de l'ordinateur à une note du clavier virtuel.

Voici le schéma global de cette ma fonction noteclavier(frequence).



Ce schéma n'est pas similaire aux schémas habituels de numérisation du son puisqu'il convertit directement les signaux numériques en sons, contrairement à d'habitude où il doit "re-passer" par les signaux analogiques, c'est pourquoi cette fonction est intéressante et originale. J'ai eu de nombreux problèmes pour la créer, notamment car, ayant commencé par des programmes plus simples n'impliquant pas la variable "ond_onde", j'obtenais un son grésillant et insupportable. Je suis consciente que le son final n'est pas très agréable non plus, mais c'est le plus audible que j'ai pu faire ! Je n'ai pas beaucoup persévéré sur ce point car je voulais me pencher sur des nouvelles parties.

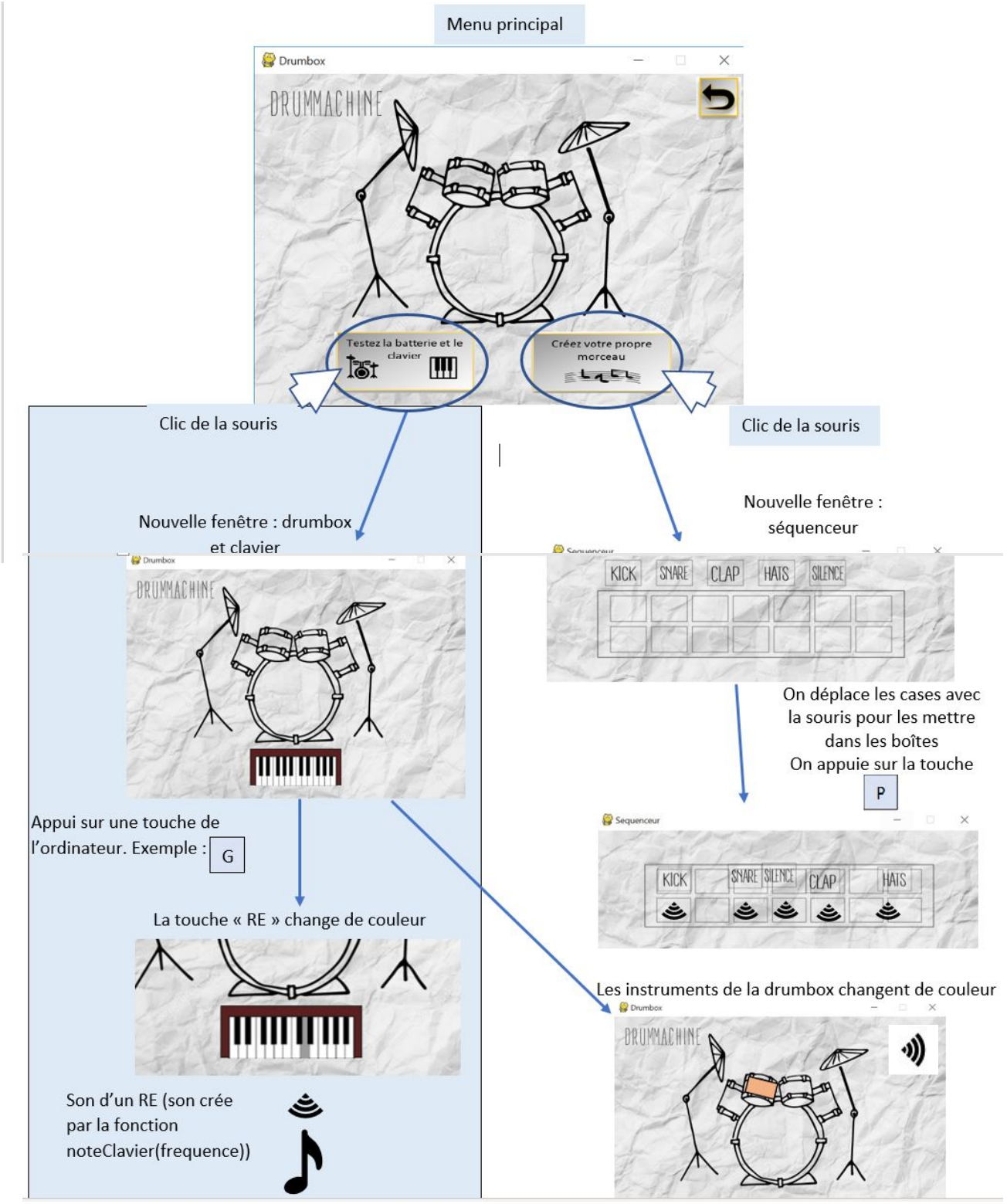
Menu principal:

Voici une des parties du menu que j'ai codée. Au début du code, j'ai affiché la page du menu principal (l.28 à 40). Dans le code ci-dessous, j'ai fait en sorte que quand la souris est cliquée sur les cases "testez la batterie et le clavier" du menu, la fenêtre de la drumbox et du clavier s'affichent.

```
397 continuer = 1
398 while continuer:
399     for event in pygame.event.get():
400 
401         if event.type == QUIT:
402             continuer = 0
403 
404         if event.type == MOUSEBUTTONDOWN and event.button == 1:
405             x1 = event.pos[0]
406             y1 = event.pos[1]
407 
408 
409 #affichage de la fenêtre de la drumbox quand est pressée la case "testez la batterie et le clavier")
410             if 130 <= x1 <= 360 and 420 <= y1 <= 520:
411                 fenetre_drumbox = pygame.display.set_mode(res)#on ouvre une nouvelle fenêtre
412                 drumbox_image = pygame.image.load("Images/DrumMachine.png")
413                 fenetre_drumbox.blit(drumbox_image, [0, 0])
414                 fenetre_drumbox.blit(clavier_image, [0, 70])
415                 pygame.display.flip()
416                 launched = True
417                 while launched:
418                     for event in pygame.event.get():
419                         if event.type == pygame.QUIT:   #Fermeture de la fenêtre lorsque l'on appuie sur la croix rouge
420                             launched = False
421                         Batterie_clavier()
```

Elle est beaucoup basée sur les événements de la souris (sur pygame : `event.get()`).
l.401 : `event.type == quit` → lorsqu'on appuie sur la croix, on quitte le programme.
`event.type == MOUSEBUTTONDOWN and event.button == 1` → action de faire un clic gauche sur la souris
Les variables x1 et y1 sont définies par les fonctions `event.pos[]` → coordonnées de la souris lorsqu'on on clique sur l'écran.
Dans la condition `if 130 <= x1 <= 360 and 420 <= y1 <= 520` , les valeurs numériques correspondent aux coordonnées de la touche "testez la batterie et le clavier". Ainsi, lorsque la souris est cliquée dans la zone de la touche "testez la batterie et le clavier", la fenêtre de la drumbox et du clavier s'affiche.
J'ai ensuite fait s'afficher cette fenêtre (l.411 à 415).
l.421 : `BatterieClavier()` → importation de la fonction principale de la drumbox et du clavier

B)Résultat final:



Voici le résultat final de notre projet. Les parties que j'ai programmées sont surlignées en bleu.

La partie clavier répond au cahier des charges car le programme du clavier marche convenablement (lorsque l'on appuie sur une des touches entre F et G, la note s'allume et fait un son, ce qui est le but de la fonction). Cependant, j'ai réalisé plusieurs tests qui montrent des lacunes de ce code : lorsque l'on appuie simultanément sur deux touches, les notes sont jouée une à la fois, et pas en même temps comme dans un vrai piano. Cela vient sûrement du fait que j'ai définit un temps invariable (1 seconde) pour le son, et je n'ai pas réussi à coder la fonction de sorte que la durée du son varie selon la durée de l'appui sur la touche.
Au final, je suis tout de même contente de ma fonction noteClavier(frequence) qui code pour le clavier. Elle m'a pris beaucoup de temps pour un résultat en apparence banal, mais elle cache des aspects beaucoup plus complexes et techniques qu'elle en a l'air, et a nécessité de nombreuses réflexions et recherches sur la numérisation du son.

Le menu est également satisfaisant car il renvoie efficacement aux autres fenêtres du programme, dès lors que l'on clique sur les touches du menu. Même lorsqu'on test le fait de cliquer aux bornes des coordonnées des touches du menu, les fenêtres des autres parties du programme s'ouvrent. La croix rouge fonctionne également. Cependant, il manque un élément important : le retour au menu principal. (voir III/ A/ Perspectives d'amélioration))

III/ Réflexions post-projet:

A) Perspectives d'amélioration:

Lacunes et problèmes à améliorer:

- *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.*

- *Menu : il manque un élément important : le retour, qui pourrait être présent sur chaque fenêtre du projet et qui permettrait d'accéder au menu principal sans avoir à recommencer le programme. Nous avons songé à insérer la fonction break pour revenir au début de la boucle while. Cependant, nous ne sommes pas*

parvenus à le coder car de nombreuses tentatives ont été vaines et nous avons préféré nous concentrer sur les autres finitions.

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".

B) Bilan personnel sur les apports du projet

Ainsi, même si ce projet a été quelquefois complexe et qu'il pourrait être amélioré, je pense que nous sommes toutefois tous satisfaits et fiers de notre projet. Pour ma part, il s'agissait d'une première dans le domaine du codage, et jamais je n'aurais pensé parvenir individuellement à effectuer un tel projet. Il y a eu de nombreux moments de découragement dus à des bugs permanents, d'agacement contre des fonctions qui ne marchaient pas et de nombreuses nuits presques blanches, mais également beaucoup d'épisodes de rires avec mes coéquipiers, d'excitation et de jubilation lorsque nous réussissons enfin à débloquer des problèmes ou à faire marcher des fonctions. Il y a régné un fort esprit d'entraide basé sur la communication et sur le partage de connaissances et n'avons pas cessé de nous motiver mutuellement.
En outre, il nous a permis de nous ouvrir au travail d'équipe (communication, collaboration, compromis...) et à la gestion de projet (planning à respecter, connaissances à accumuler et à utiliser...). Ceci pourra nous être très utile dans l'avenir, en particulier pour moi qui souhaite me tourner plus tard vers la gestion de projets. J'ai aussi trouvé intéressante la pédagogie du "self-learning" instaurée par notre professeur, car elle nous a permis de nous forcer à faire des recherches par nous-mêmes et à être autonomes. Je pense qu'avec le recul, nous sommes tous surpris de nous-même et de ce que nous sommes parvenu à faire après quelques mois d'ISN. Notre projet fut ainsi une expérience très enrichissante, tant du point de vue informatique que du point de vue de l'épanouissement personnel.