Génie Logiciel Orienté Objet : Gaudrophone

Livrable 2

présenté à

Jonathan Gaudreault

par

Équipe — GLOrious

matricule	*nom*	*signature*
111 152 584	Benjamin Matte-Jean	
111 152 547	Édouard Carré	
111 048 727	Marc-Olivier Tourigny	
111 153 499	Olivier Gamache	

Université Laval

17 octobre 2017

Historique des versions		
version	*date*	*description*
0	11 septembre 2017	Création du document
0.5	20 septembre 2017	Amalgame des différents fichiers déjà écrits
1.0	25 septembre 2017	Mise à jour finale du livrable 1
1.1	1er octobre 2017	Mise en place du format du livrable 2
1.2	8 octobre 2017	Ajout du diagramme de classe conceptuelle
1.3	12 octobre 2017	Modification du diagramme de classe conceptuelle avec les package
1.7	12 octobre 2017	Ajout des diagrammes de séquence
1.8	13 octobre 2017	Ajout du Gantt
2.0	17 octobre 2017	Remise finale du livrable 2

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Diagramme de classe conceptuelle

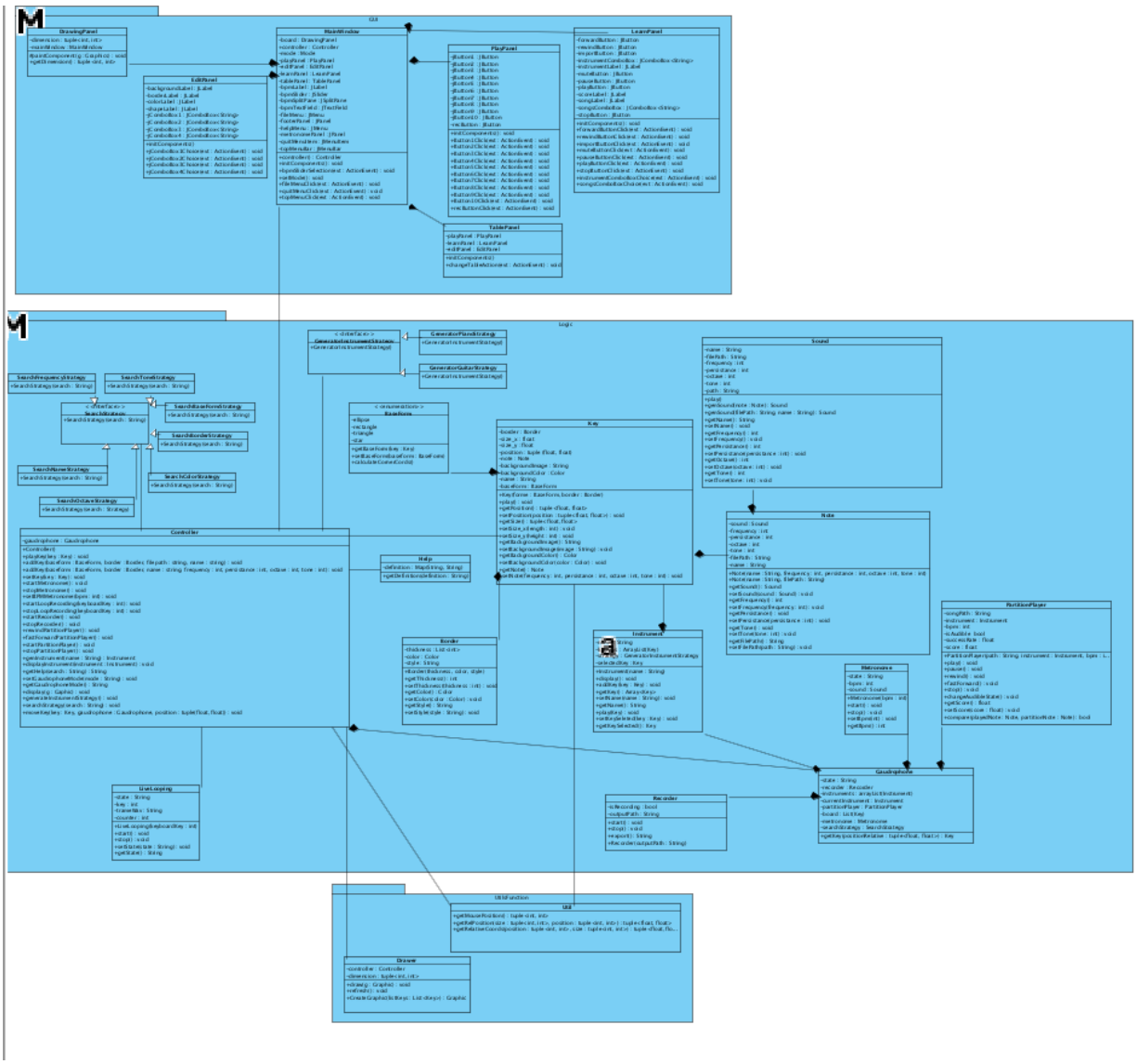


FIGURE 1.1 – Diagramme de classes

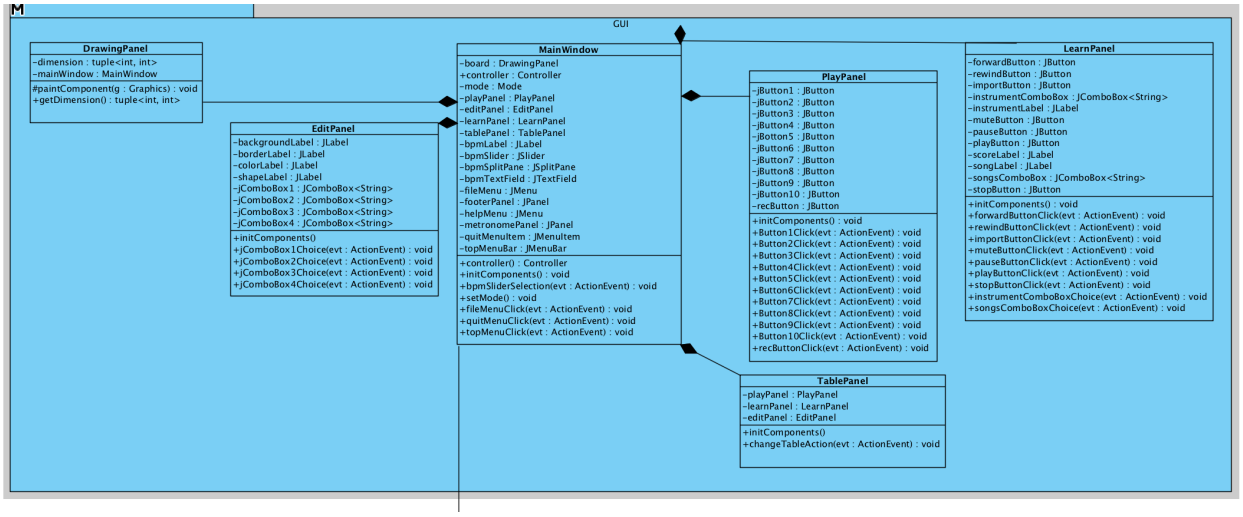


FIGURE 1.2 – Package de GUI

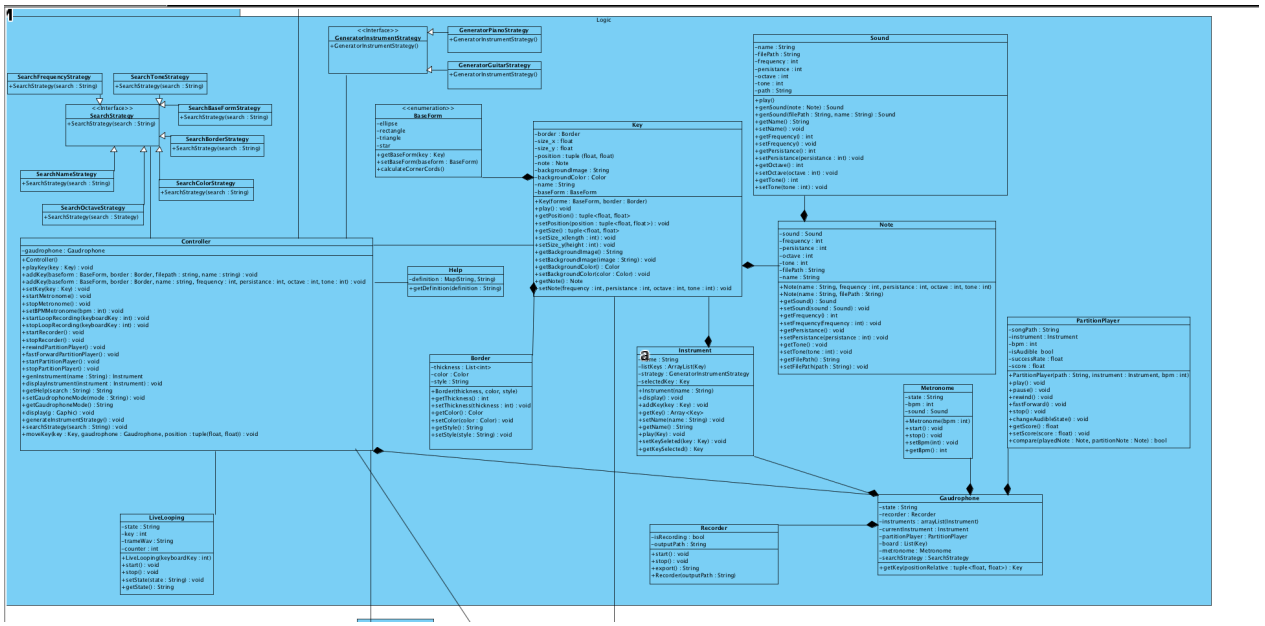


FIGURE 1.3 – Package de logique

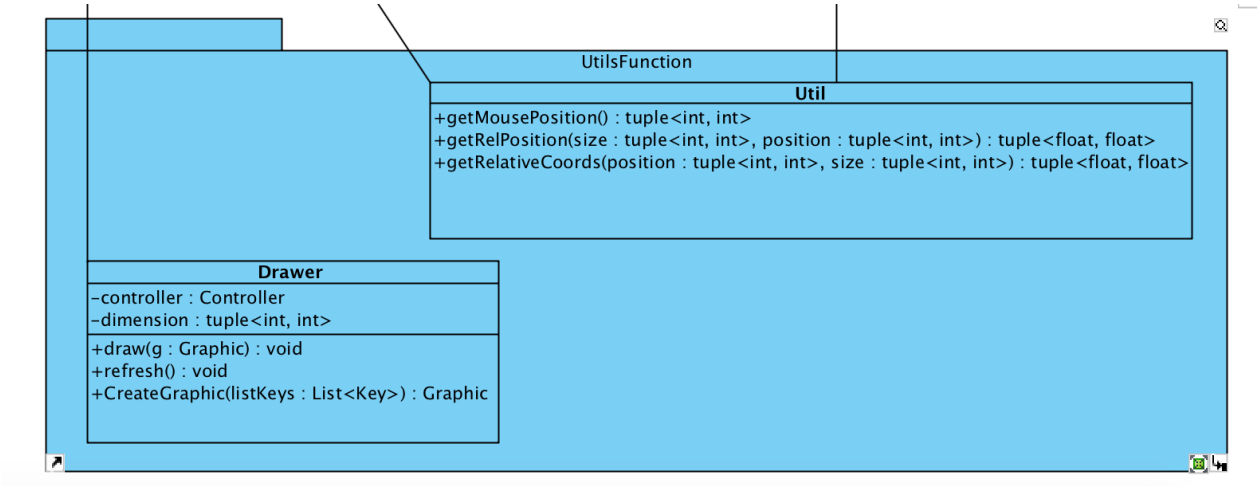


FIGURE 1.4 – Package de UtilsFunction

1.1 BaseForm : Logic

La classe BaseForm contient un constructeur par défaut qui permet d'instancier un nouvel objet BaseForm. Elle contient les attributs ellipse,rectangle,triangle et star qui sont des objets de type Form. Ces formes sont déjà programmés et contenu dans le dictionnaire. On peut sélectionner la forme et ensuite l'associer à un objet Key et ensuite la méthode setBaseForm retourne la forme associé à l'objet. La méthode calculateCornerCoords permet de calculer les coordonnées de la forme et cela permet d'avoir la position de la formes à chacun de ses points sur le board. Les sommets des formes sont transformés en un tuple avec les coordonnées et leur sont ensuite attribués. Pour ce qui est du cercle, son centre est saisi et attribué comme étant son seul sommet. C'est comme cela qu'on calcule les coordonnées des différentes formes.

1.2 Controller : Logic

La classe controller permet de gérer toutes les interactions entre l'interface graphique et la couche logic de l'application. Elle contient des méthodes pour récupérer les commandes effectuées par l'affichage graphique pour faire la gestion de l'affichage des objets de type Key.

Elle contient également des méthodes pour démarrer et arrêter le métronome, en plus de modifier son rythme directement via l'interface graphique.

Elle contient également des méthodes permettant de commencer, d'enregistrer et finalement arrêter le bouclage en direct (live looping). Ces trois méthodes prennent un entrée sous forme d'une touche de clavier appuyée pour associer la boucle à une touche et pour arrêter la boucle voulue.

Elle contient les méthodes startPartitionPlayer et stopPartitionPlayer qui permettent de faire la lecture d'un fichier audio dans l'application (ou bien de l'arrêter complètement). La pièce sera donc jouée par l'instrument couramment chargé. Les méthodes rewindPartitionPlayer

et fastForwardPartitionPlayer permettent de reculer ou d'avancer la lecture du fichier audio joué par la classe Partition Player dans l'interface selon la demande de l'usager.

Cette classe peut également générer un instrument à partir de la méthode generateInstrument qui prend en paramètre une string qui est ensuite convertie en l'instrument désiré parmi ceux pré-programmés. Ceci fait appel aux patrons stratégies.

La méthode displayInstrument permet d'afficher l'instrument dans la zone d'affichage de l'application et elle prend en paramètre un objet Instrument qui peut être soit généré ou créé.

La classe contient également la méthode getHelp qui prend en paramètre une string. Une fenêtre d'aide s'affichera et filtrera les diverses rubriques selon ce que l'usager a entré dans un champ de recherche d'aide pour essayer de lui donner l'information qu'il recherche.

Elle permet d'aller chercher et d'attribuer un mode à l'objet Gaudrophone grâce aux méthodes getGaudrophoneMode qui sélectionne le mode désiré et ce mode peut ensuite être attribué au Gaudrophone par la méthode setGaudrophoneMode. On peut ainsi naviguer dans l'interface entre les mode de l'application.

Elle permet également d'afficher des objets de type Graphic sur l'interface.

Elle permet également d'utiliser le patron stratégie. En effet, cette méthode permettra, selon le switch case, de générer l'instrument désiré selon une sélection d'instruments déjà présent et celui que l'utilisateur désire générer. Cette méthode s'occupera de générer l'instrument désiré de l'utilisateur.

Finalement, la méthode searchStrategy permet de choisir la stratégie à prendre pour rechercher quelque chose, soit selon n'importe quel attribut d'une key, d'une note et d'un sound.

1.3 Gaudrophone : Logic

La classe Gaudrophone contient un constructeur par défaut qui permet de créer un nouvel objet Gaudrophone. Elle contient également plusieurs attributs qui sont propres à chaque objet. Elle contient les attributs state qui est une string et qui indique le mode en cours (Jouer, Apprendre ou Créer), le recorder qui est un objet Recorder et qui permet d'enregistrer des boucles. De plus, elle possède l'attribut instruments qui est une liste des instruments pré-programmés. Elle contient également currentInstrument qui est un objet Instrument qui correspond à l'objet affiché l'application. Elle contient également l'attribut partitionPlayer qui est un objet de type PartitionPlayer qui permet de jouer les sons. Elle contient aussi l'attribut board qui est une liste d'objet Key et qui sont reliés à un instrument dans un double Array, donc un tableau dans tableau pour former une matrice. Cette matrice est de grosseur 2000 par 1000 et pour chaque nombre, un pointeur à la position relative d'une key est stocké à ces coordonnées pour chaque coordonnée entière de la matrice. Ce board obligera via le contrôleur au DrawingPanel d'avoir des noeuds pour placer les keys et celles-ci pourront occuper un nombre arbitraire en int de noeuds, mais ne pourra pas dépasser ces noeuds. Finalement, elle possède l'attribut metronome qui est un objet Metronome qui permet d'avoir le rythme désiré avec un objet Sound qui lui est propre. Elle contient également un attribut de type SearchStrategy qui va permettre la recherche de touche dans le mode Apprendre. La méthode

getKey qui prend un Tuple de float, soit une position relative, remet la key de son attribut Board qui est sous le clique de l'utilisateur à cette position.

1.4 MainWindow : GUI

La classe MainWindow permet l'affichage de l'application. Elle possède les attributs suivants : un board de type DrawingPanel, un controller de type Controller, un mode de type Mode, un panel de chaque type(play,edit,learn), un objet de type Label, un objet de type Slider, un objet de type JSplitPane ainsi qu'un objet de type JTextField qui sont reliés au métronome de l'application. Elle possède également, un JMenu pour chaque bouton du menu principal, un autre objet de type JPanel pour les onglets en haut de page, un JMenu pour le menu d'aide, un JPanel pour le métronome, un JMenuItem pour quitter l'application et un JMenuBar pour afficher le menu de l'application. Elee possède les méthodes une méthode qui permet de créer un objet de type Controller et une méthode initComponents qui permet d'initialiser les composantes. Le bpmSliderSelection est un ActionEvent qui réagi au déplacement du JSlider. Il est possible d'attribuer le mode grâce à la méthode setMode. Chaque boutton de la barre du menu principal ainsi que le x rouge possède un ActionEvent qui s'occupe de faire le traitement lorsqu'un clic se produit.

1.5 DrawingPanel : GUI

La classe contient un attribut de type MainWindow ainsi que des dimensions qui sont définis par un tuple de float. Elle possède une méthode protégée qui est les composantes de l'interface principal ainsi qu'une méthode privée pour sélectionner les dimensions et elle retourne un tuple de int.

1.6 EditPanel : GUI

La classe contient plusieurs attributs. Elle est composé d'un fond, une bordure, une couleur et une forme qui sont tous de type JLabel. De plus, elle est également composé de quatre JComboMix qui sont formés de liste de string et qui défile les différents éléments. Elle possède également les méthodes initComponents qui permet d'initialiser l'EditPanel. Également, chaque JComboBox possède une méthode qui permet de traiter les différents évènements reliés à un clic.

1.7 LearnPanel : GUI

La classe contient des boutons pour faire avancer ou reculer la piste courante. Elle contient des boutons pour arrêter complètement, mettre sur pause et partir la piste. Elle contient divers label pour afficher le nom de l'instrument, le nom de la piste et le score. Elle contient

également un combo box présentant les divers instruments et un autre comboBox montrant quelles chansons sont associées à cet instrument. Finalement, elle possède diverses méthodes pour faire la sélection des différents comboBox et des différents boutons énumérés ci-haut.

1.8 PlayPanel : GUI

La classe contient les 10 boutons (le 0 sur le clavier est le bouton 10) de live looping différents, ainsi qu'un bouton pour commencer l'enregistrement du live looping. Elle contient également des méthodes permettant de capter les clics sur les divers boutons pour savoir lequel est pressé.

1.9 TablePanel : GUI

La classe contient une méthode initComponents qui permet d'initialiser le TablePanel ainsi qu'une autre méthode qui permet d'interchanger l'attribut qui est affiché. Cette classe possède un attribut de type PlayPanel, un de type LearnPanel ainsi qu'un de type editPanel.

1.10 Key : Logic

Cette classe a les attributs suivants : un objet de type Bordure(border), des coordonnées en x et en y, tout deux de type float, une position qui sera déterminé par les coordonnées en x et en y, un objet Note(note), une image de fond ainsi que sa couleur (backgroundImage et backgroundColor), une string qui représente le nom et une forme de base qui est un type d'objet BaseForm.

La classe Key est une classe très importante de l'application. Elle permet de créer un objet Key et ce constructeur prend les attributs baseForm et border. Elle contient les méthodes setSize qui prenne la position en x et en y de l'objet et ce sont des paramètres de type int. Elle permet également de sélectionner et de déterminer la couleur de fond de l'image(getBackgroundImage, setBackgroundImage(image : String)). La méthode set prend en paramètre une string qui sera ensuite converti en un objet couleur. Elle permet également d'associer à l'objet une note par la méthode getNote.

1.11 Border : Logic

La classe Border possède un constructeur qui prend en paramètre un int contenu dans une liste qui représente l'épaisseur(thickness), une couleur qui est un objet Couleur ainsi qu'un style qui est une string. On peut sélectionner et associer les différents valeurs aux attributs grâce aux méthodes get et set de chaque attribut. Notons que cette liste, via son index, indique quel côté est affecté par cette bordure. Par exemple, disons que le côté central d'un carré (une

ligne au centre) doit avoir une bordure, alors au cinquième index, une thickness sera donnée pour qu'une ligne apparaisse au centre du bouton.

1.12 Note : Logic

La classe Note possède les attributs suivants : un objet de type Sound, une fréquence, une persistence,un octave, un timbre qui sont tous des int ainsi qu'un chemin de fichier de type string. Cette classe a deux constructeurs tout dépendemment ce qu'on lui passe en paramètre. On peut soit lui passer un chemin vers un fichier audio (.wav) et un objet sera construit. Sinon, on peut la créer manuellement en lui passant différents paramètres (fréquence et cetera). Par la suite, on peut sélectionner et attribuer des valeurs aux différents attributs. Ces notes peuvent être affectées à plus d'une touche, bien sûr. Prenons par exemple une guitare : une corde jouée à vide peut avoir exactement la même note qu'une autre corde jouée quelques frettes plus bas. Cette note est exactement la même et aura comme attribut Sound le même, car une corde vibre selon sa longueur et son épaisseur : les sons de deux cordes jouées selon des frettes différentes seront identiques si la guitare est bien accordée.

1.13 Sound : Logic

La classe Sound est instanciée avec un constructeur par défaut. Puis, pour associer un son à l'objet Sound créé, on utilise la méthode genSound prenant soit un name et un chemin fichier(sous forme de string), soit une frequency, un tone, un octave, une persistance et un name en argument. Avec la première option, le fichier audio pointé sera directement lié à l'attribut path. Avec la deuxième option, le son sera construit avec les différentes propriétés physiques. Par la suite, un fichier audio sera créé, puis exporté dans un fichier audio et ce fichier sera associé à l'attribut path. La classe possède de nombreuses méthodes pour aller chercher et ajuster (get/set) les divers paramètres énumérés. Cependant, la méthode la plus importante est la méthode play(). Celle-ci se chargera d'aller lire le fichier audio associé au son dans l'attribut path lors qu'une touche est activée. Notons que dans tous les cas, un fichier est créé pour qu'il soit joué et que plusieurs instances de clip Java soient jouées en même temps.

1.14 Instrument : Logic

La classe Instrument contient un constructeur qui prend en argument son nom de type String, une liste d'objet de type Key qui sera d'abord vite à laquelle les notes créées par l'utilisateur seront ajoutées ainsi qu'une stratégie qui est un objet GeneratorInstrumentStrategy pour être généré automatiquement par l'application. Elle contient une méthode display qui affiche l'instrument dans la zone d'affichage de l'application. On peut également jouer de l'instrument grâce à la méthode play qui joue d'un objet Key. On peut également sélectionner et attribuer des valeurs aux différents attributs avec les méthodes get et set.

1.15 Metronome : Logic

La classe Métronome contient un constructeur qui prend comme paramètre un int qui représente le bpm(beep per minute). Il possède les attributs state qui est une string qui indique l'état du métronome(on or off), un int qui représente le bpm(beep per minute) et un objet Sound qui permet de produire le son inhérent du métronome. Les méthodes start et stop permettent de déterminer l'état du métronome et de pouvoir le partir et l'arrêter à la guise de l'utilisateur. On peut également passer en argument différentes valeurs pour l'attribut bpm et l'attribuer au start grâce aux méthodes get et set.

1.16 Util : UtilsFunction

La classe Util perment de récupérer la position de la souris lorsqu'un clic occure. En effet, elle permet d'obtenir un tuple de deux valeurs de type int qui indique la position du clic. De plus, elle permet de la faire la conversion de la position absolue des touches et des clic vers les positions relatives grâce à la méthode getRelPosition qui prend une position qui est un tuple d'int et retourne un tuple de float.

1.17 LiveLooping : Logic

Cette classe fait la gestion du live looping. Dans le constructeur, on retrouve seulement un int, qui correspond à une touche du clavier que l'utilisateur aura choisit. Lorsque l'utilisateur appuie sur cette touche, le constructeur est appelé et on stocke la touche du clavier dans l'attribut key. À ce moment, l'attribut statique counter s'incrémente et l'état state est mis à recording. Ensuite, l'objet Recorder est instantié et celui commence à enregistrer ce qui est joué par l'usager. Quand l'usager réappuie sur la même touche, l'enregistrement s'arrête. Un fichier audio est créé à un chemin d'accès défini par l'application. Ce chemin est stocké dans l'attribut tramWav. L'attribut state est ensuite mis à playing. À ce moment, la méthode play est appelée. Celle-ci ira lire en boucle le fichier audio créé par le constructeur. Ce processus s'arrêtera seulement lorsque l'usager appuiera une troisière fois sur la touche associée à cet objet. Ceci apppele la méthode stop, qui met fin à la lecture en boucle du fichier audio et ira ensuite le supprimer. On décrémente ensuite l'attribut counter.

1.18 Drawer : UtilsFunction

La classe Drawer sert à faire la gestion de l'affichage de l'instrument courant. Elle dispose de sa propre instance de l'objet Controller pour capter les divers évènements effectués effectués sur l'interface pour respecter l'application du contrôleur de Larman. Elle dispose également des dimensions de l'affichage de l'instrument. Ceci est essentiel pour l'appel de la méthode draw. Celle-ci reçoit en paramètre un objet Graphic (ex : une touche à afficher). Celle-ci se charge d'en faire l'affichage dans l'application et doit donc connaître la position relative des

différents éléments impliqués. Ceci sera donc géré par la gestion d'évènements gérés par le contrôleur. La méthode refresh sert à refaire l'affichage des différents éléments graphiques, dans le cadre d'un réajustement de la fenêtre par exemple.

1.19 Recorder : Logic

Cette classe fait la gestion de l'enregistrement de fichier audio. Dans son constructeur, on retrouve un chemin fichier où sera enregistré le fichier audio produit. Ceci sera stocké dans l'attribut outputPath. Ensuite, pour enregistrer, on appelle la méthode start. Ceci met l'attribut isRecording à true. L'enregistrement s'arrêtera lorsqu'un évènement quelconque appellera la méthode stop. Ensuite, la méthode export sera appelée et celle-ci se chargera de créer et d'exporter le fichier audio au chemin spécifié et finalement isRecording sera mis à false.

1.20 PartitionPlayer : Logic

Cette classe fait la gestion d'un lecteur de fichier audio. Dans son constructeur, on retrouve l'assignation des divers attributs, soit la vitesse de lecture (bpm), le fichier audio qui devra être lu (songPath) et l'instrument qui sera utilisé (instrument). Ces paramètres seront bien entendu choisis par l'utilisateur dans l'interface graphique. La méthode start démarre la lecture de la pièce musicale pointée par songPath. La méthode pause met la lecture sur pause. La méthode stop arrête complètement la lecture et sert à réinitialiser les divers attributs pour pouvoir lire un fichier audio différent ou changer d'instrument. Les méthodes fastForward() et rewind() servent à naviguer dans le fichier audio pour rejoindre des parties de la chanson plus rapidement selon le désir de l'utilisateur. Ces méthodes sont exécutés suite à l'interception de commande par le contrôleur. L'attribut isAudible sert à définir si l'utilisateur entendra ce que le PartitionPlayer joue ou si celui-ci sera en sourdine. On peut le modifier avec la méthode changeAudibleState() qui changera de façon binaire cet attribut. L'attribut score sert à donner un rendu final de la performance acoustique de l'usager. S'il joue les notes similairement au même moment (ceci sera géré par la méthode compare()), son score sera ajusté avec setScore(). On aura également accès au score courant avec getScore() pour faire un affichage visuel par exemple.

1.21 Help : Logic

Cette classe possède une map qui met ensemble une définition et un objet de l'application. Chaque objet est enregistré selon son nom en string et il possède une définition, en string aussi, qui explique ce à quoi elle sert. Elle pourra être utilisée avec la méthode getDefinition qui affichera via le contrôleur à l'écran dans une fenêtre chacune des définitions pour aider le musicien à utiliser l'application.

1.22 GeneratorInstrumentStrategy : Logic

Cette interface sert bien sûr à générer des instruments selon un algorithme. Toutes les stratégies implémentent cette interface en surchageant la méthode associée au dit algorithme. Cette interface est donc utilisée par le contrôleur pour qu'il soit facile d'ajouter des générateurs différents tout en limitant la duplication de code, mais surtout le fait que dans le futur, il n'y ait pas des blocs de code aux mauvais endroits si jamais quelqu'un venait à vouloir en ajouter. L'instrument devra donc implémenter une stratégie qui a comme type cette interface. Ensuite, lorsqu'un objet générateur de guitare sera créé, par exemple, il faudra que cette stratégie soit d'abord celle utilisée par l'instrument et c'est via cette interface que ces changements de stratégie seront faits.

1.23 GeneratorGuitarStrategy : Logic

Cette classe surcharge la méthode de génération en implantant l'algorithme qui sert à générer une guitare. Il sera expliqué plus loin, mais faisons un survol. L'algorithme est une boucle for qui, à l'aide de l'index, crée des touches. Celles-ci varieront selon plusieurs attributs : l'index sert à modifier la fréquence et la position pour avoir un multiple qui pourra modifier la fréquence fondamentale selon la corde et la frette et, justement, l'emplacement de cette dite frette qui sont en train d'être générés à cet index de la boucle. Ainsi, le générateur créera des touches une à une avec une note associée à sa position sur la guitare, indiquée par l'index, puis générera le son associé à cette corde sur la frette. Pour imiter un manche de guitare, une position selon une hauteur qui correspond à la corde et une largeur qui correspond à la frette sera calculé via l'index. Le timbre de l'instrument sera celui d'une guitare, bien entendu.

1.24 GeneratorPianoStrategy : Logic

Cette classe surcharge aussi la méthode de génération de son parent. Elle agit exactement de la même manière que la génération d'une guitare, mais en utilisant l'index pour modifier l'emplacement sur la longueur, plutôt que sur la hauteur et la largeur, comme la guitare pour imiter un manche. Le clavier créé aura des notes selon l'index qui modifiera aussi le son selon son emplacement sur ce clavier. Le timbre, cette fois, sera celui d'un piano.

1.25 SearchStrategy : Logic

Cette fois-ci, cette interface est utilisée pour chercher des touches lorsque le Gaudrophone est dans le mode Apprendre. L'interface ajoute encore une fois une méthode vide de Search qui sera surchargée par les classes enfant. Chacun de ses enfants pourra être utilisé via le Gaudrophone qui implémente la stratégie qui change selon ce qui est recherché. Même si cette méthode ne sera pas implémentée ici, expliquons-la, car la seule différence sera ce qui est recherché, mais ce sera autrement presque identique. Lors d'une recherche, les touches

qui correspondent aux critères de recherche implémentés par les différents algorithmes de stratégie verront leur couleur modifiée de sorte à être affichée en surbrillance.

1.26 SearchFrequencyStrategy : Logic

Cette classe surcharge la méthode search de son parent pour que, via le contrôleur, dans la barre de recherche, il sera possible de chercher les touches d'un instrument via l'attribut frequency de ses notes et les affichera en surbrillance.

1.27 SearchToneStrategy : Logic

Cette classe surcharge la méthode search de son parent pour que, via le contrôleur, dans la barre de recherche, il sera possible de chercher les touches d'un instrument via l'attribut tone de ses notes et les affichera en surbrillance.

1.28 SearchBorderStrategy : Logic

Cette classe surcharge la méthode search de son parent pour que, via le contrôleur, dans la barre de recherche, il sera possible de chercher les touches d'un instrument via l'attribut border de ses Keys et les affichera en surbrillance.

1.29 SearchBaseFormStrategy : Logic

Cette classe surcharge la méthode search de son parent pour que, via le contrôleur, dans la barre de recherche, il sera possible de chercher les touches d'un instrument via l'attribut BaseForm de ses keys et les affichera en surbrillance.

1.30 SearchColorStrategy : Logic

Cette classe surcharge la méthode search de son parent pour que, via le contrôleur, dans la barre de recherche, il sera possible de chercher les touches d'un instrument via l'attribut Color de ses keys et les affichera en surbrillance.

1.31 SearchOctaveStrategy : Logic

Cette classe surcharge la méthode search de son parent pour que, via le contrôleur, dans la barre de recherche, il sera possible de chercher les touches d'un instrument via l'attribut octave de ses notes et les affichera en surbrillance.

1.32 SearchNameStrategy : Logic

Cette classe surcharge la méthode search de son parent pour que, via le contrôleur, dans la barre de recherche, il sera possible de chercher les touches d'un instrument via l'attribut Name de ses notes et des keys et les affichera en surbrillance.

Chapitre 2

Diagramme de package

2.1 Texte explicatif du diagramme de package

Le diagramme de package est divisé en trois parties différentes. La première partie (GUI) contient les classes de l'affichage graphique. Celles-ci gèrent les différents fonctions d'affichage. Ces classes passent par les méthodes du contrôleur pour communiquer et intéragir avec la couche logic. La couche logic contient de son côté toutes les classes servant à l'instantiation des objets nécessaires pour créer un instrument et l'utiliser. Elle contient également les objets servant à faire les diverses manipulations sur les fichiers audio, tel la lecture, l'import et l'export de ceux-ci. On y retrouve également le contrôleur qui est utilisé pour communiquer avec la couche GUI. Le troisième package, UtilsFunction, contient diverses fonctions utilitaires qui ne sont pas directement liés à la construction ou l'utilisation d'un instrument.

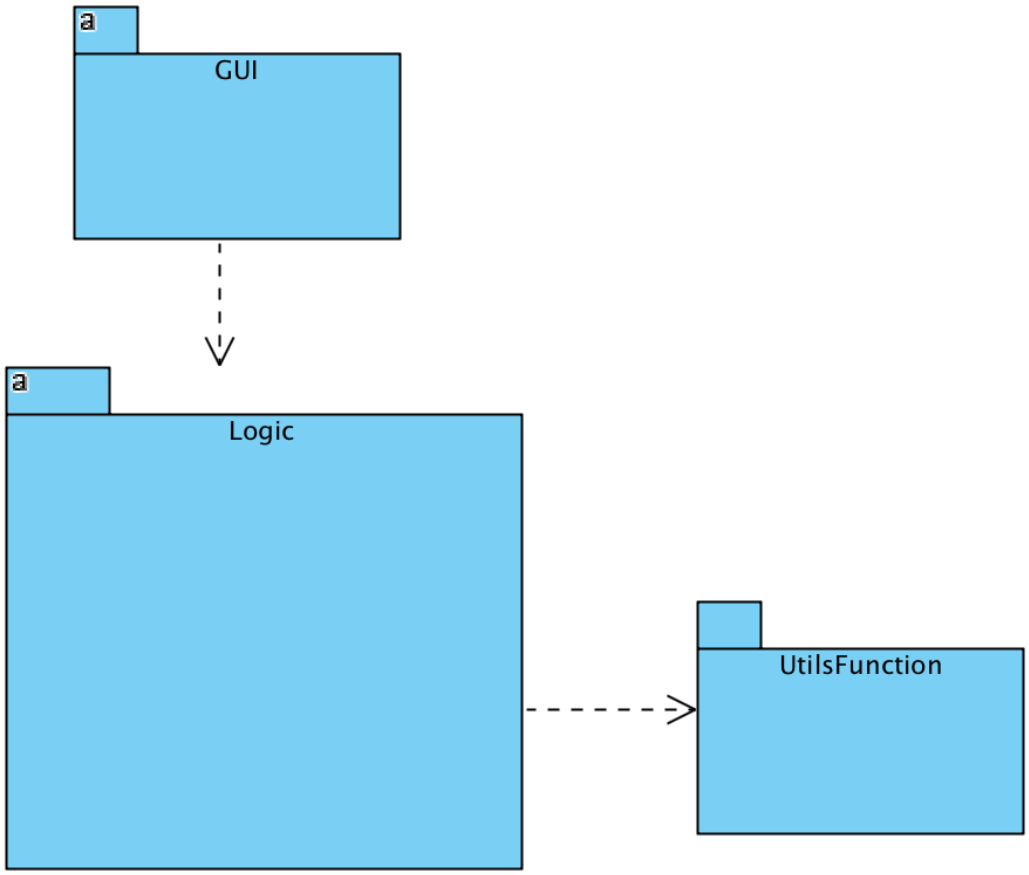


FIGURE 2.1 – Diagramme de package

Chapitre 3

Diagramme de séquence

3.1 Conversion valeur en pixel vers valeur relative

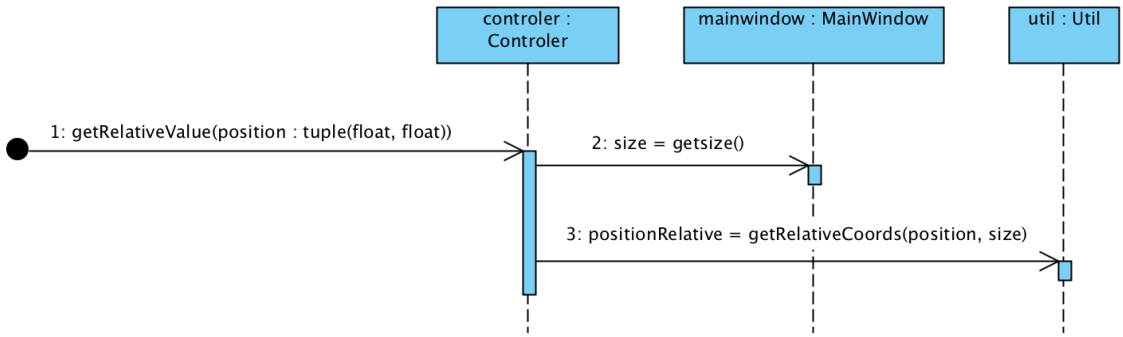


FIGURE 3.1 – Diagramme séquence : 3.1

L'application reçoit d'abord la commande getRelativeValue en prenant comme argument un tuple de float. Par la suite, le contrôleur appelle la fonction getRelativeCoords du package Util en passant en argument le tuple <int, int> correspondant à la position d'un objet quelconque (ie : touche, bouton), ainsi que la grosseur actuelle de la fenêtre, sous forme d'un tuple <int, int>. Dans cette méthode, on obtiendra la position relative en divisant la position en x de l'objet par la longueur de la fenêtre et la position en y par la hauteur de la fenêtre. On retournera ensuite ces valeurs sous forme d'un tuple <float, float> représantant la position relative de l'objet.

3.2 Vérifier l'emplacement d'un clique

FIGURE 3.2 – Diagramme séquence : 3.2

Tout d'abord, le contrôleur reçoit un clic de souris et la méthode click prend en paramètre les coordonées du clic dans la fenêtre d'affichage de l'instrument. Par la suite, si une touche existe au point où le clic a eu lieu, la fonction getRelativeBoard retourne un couple de point en coordonnées relatives. Par la suite, on sélectionne la clé de la touche avec la méthode getKey qui prend la relativePosition pour déterminer laquelle des touches a été sélectionné par le clic de souris. Si les coordonnées du clic de souris ne correspondent à aucune touche, le clic ne fera rien et il ne se produirera rien tant que le clic ne se fera pas sur une coordonnée qui est associé à une touche.

3.3 Ajouter une touche

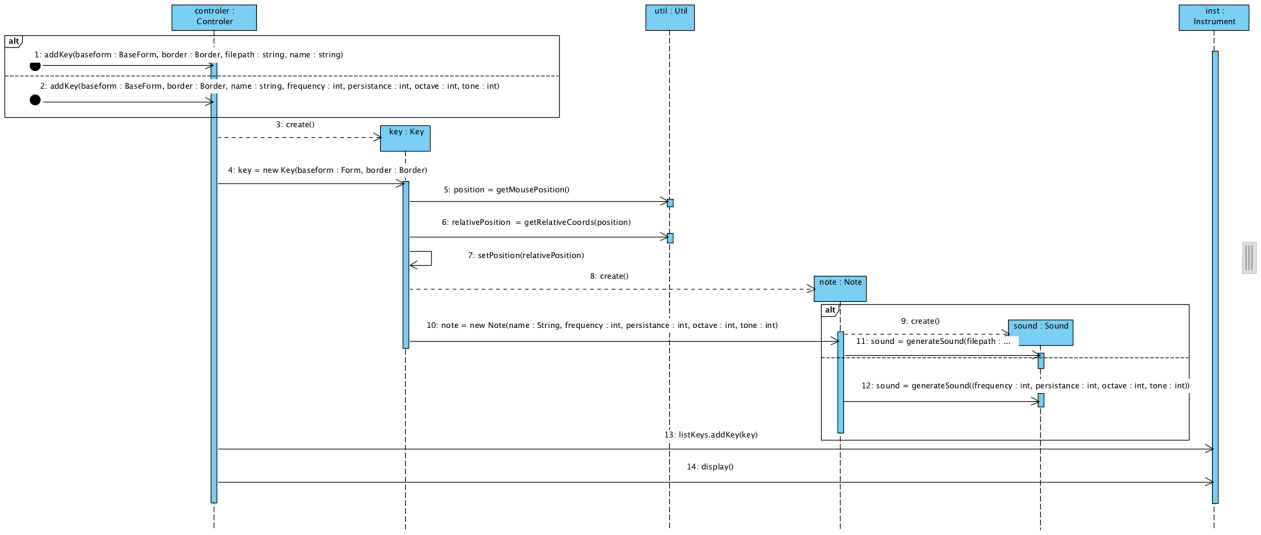


FIGURE 3.3 – Diagramme séquence : 3.3

On commence par envoyer au contrôleur la commande AddKey en lui passant comme arguments baseform : BaseForm, border : Border, filePath : string, name : string ou bien baseform : BaseForm, border : Border, frequency : int, persistance : int, octave : int et tone : int. Par la suite, le contrôleur appelle le constructeur de Key en lui passant baseform : BaseForm et border : Border comme arguments. Ensuite, l'usager doit choisir l'emplacement de l'objet Note en cliquant dans l'interface du Gaudrophone. Une méthode setPosition de Key récupère les coordonnées du clic et associe ces coordonnées à la postion en x et y de l'objet Key.

Par la suite, un objet Sound sera associé à l'objet Note créé avec l'appel de la méthode generateSound. Si l'usager a passé l'argument filePath, un objet Sound sera créé avec la méthode generateSound qui ira importer directement le son associé au fichier pointé. Si l'usager a passé frequency, persistance, octave, name et tone du son au moment d'appeler AddKey, l'appel de generateSound va plutôt construire le son en assemblant ces propriétés physiques. Finalement, l'objet Key sera ajouté à la liste de Key de l'instrument présentement affiché dans le Gaudrophone.

3.4 Déplacer une touche

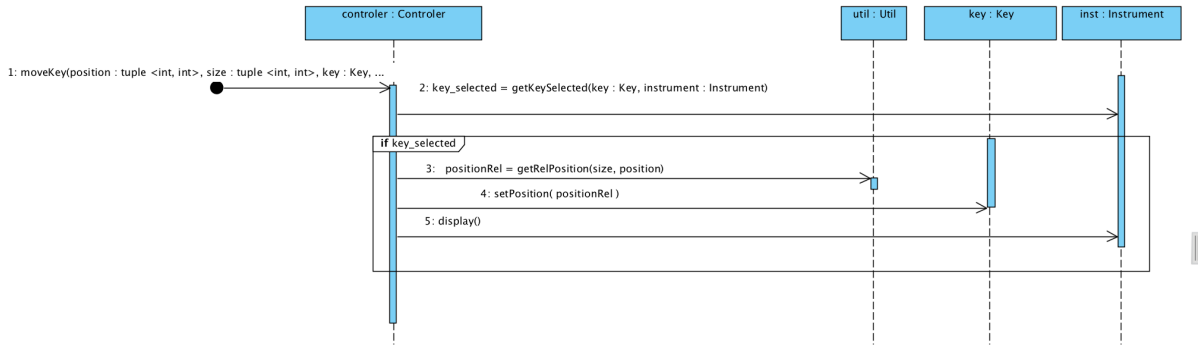


FIGURE 3.4 – Diagramme séquence : 3.4

Premièrement, l'utilisateur, en sélectionnant la touche et en la déplacant sur la zone d'affichage, envoie un signal au contrôleur et celui-ci appelle la méthode moveKey est appelée et elle prend en paramètre une touche (key), un plateau (board) et une paire contenant la nouvelle position *(newPosition)*. Par la suite, on vérifie que la position sélectionné contient un objet key avec la méthode checkKeySelected qui prend en paramètre key : Key et board : Board. Si cette méthode retourne vrai, on crée la position relative de la touche à l'aide de la méthode getPositionRelative qui prend *newPosition : tuple (float,float)* et ensuite on attribue cette position à la key. Finalement, le contrôleur affiche la touche à sa nouvelle position avec la méthode *display()*.

3.5 Cliquer sur une touche pour produire un son

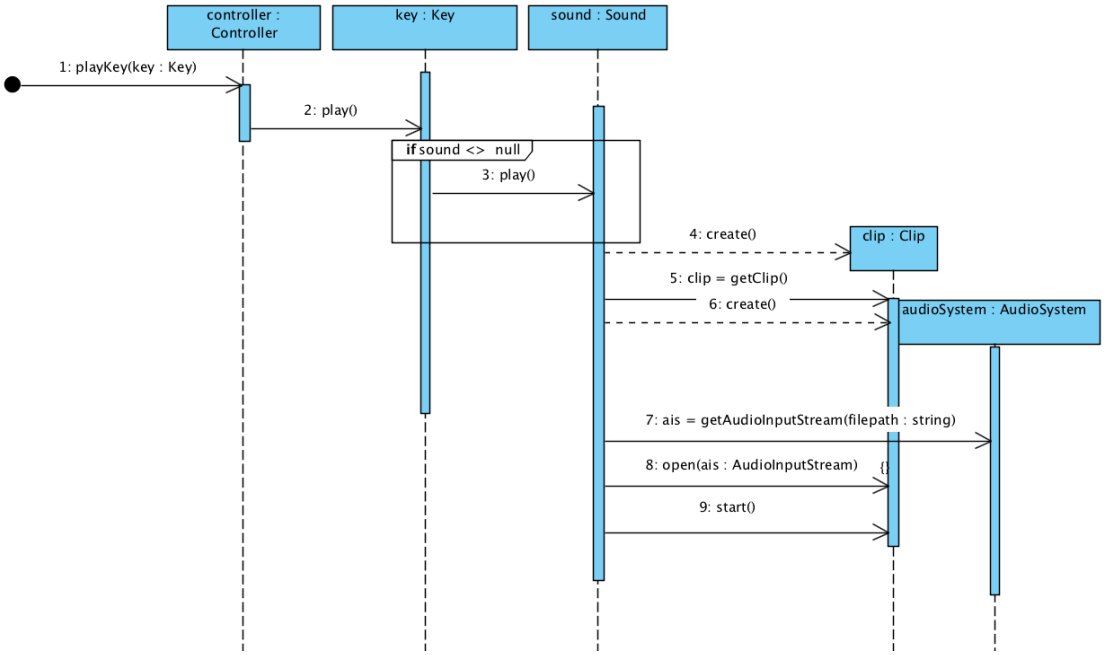


FIGURE 3.5 – Diagramme séquence : 3.5

Lorsque la méthode du contrôleur *play(Key)* est appelée, le contrôleur appelle la méthode *play()* de la touche en question. Si un son est défini pour cette touche, la touche appelle la méthode *play()* sur le son. Pour faire jouer un son, la méthode *play()* du son doit créer un clip audio et un flux audio avec des méthodes du module AudioSystem (javax.sound.sampled.AudioSystem). Le flux est ouvert avec un fichier audio, puis le clip est joué par la méthode *start()*.

3.6 Afficher l'instrument

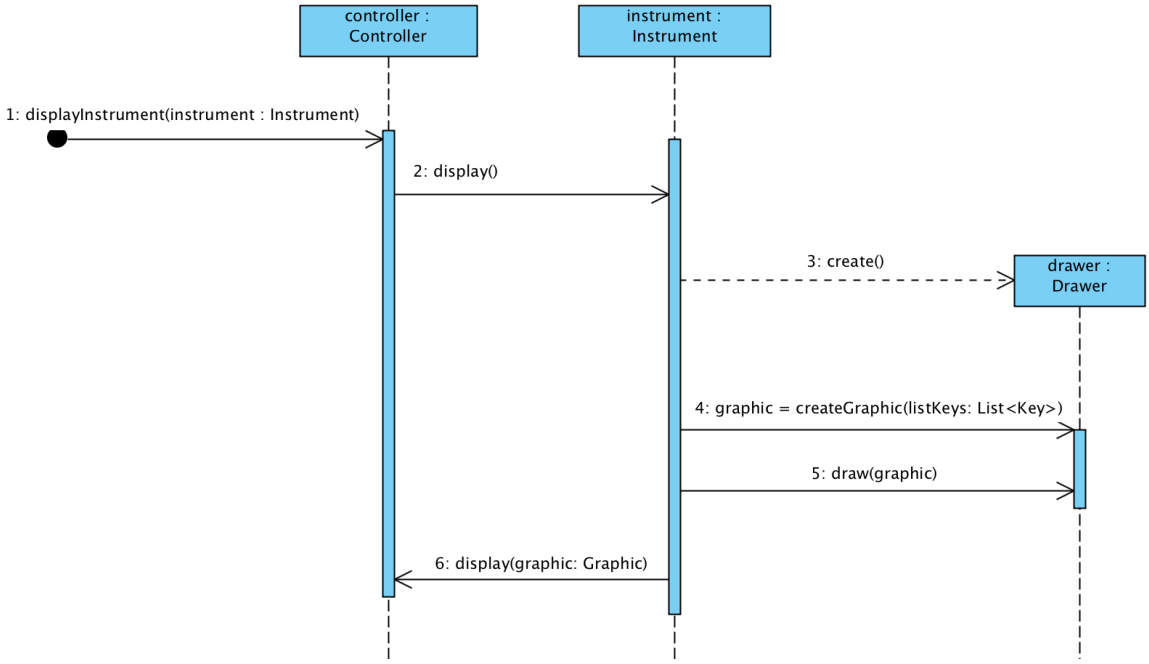


FIGURE 3.6 – Diagramme séquence : 3.6

Premièrement, le controller reçoit un signal qui lui dit d'afficher l'instrument. Le controller appelle ensuite la méthode display de la classe instrument. La classe instrument crée un nouvel instrument et crée du même coup un nouveau Drawer. Par la suite, la classe Drawer crée des objets Graphics qui serviront à leur affiche pour chacune des touches de la liste de l'instrument. Par la suite, l'instrument est affiché à l'écran dans la zone réservée à cet effet par la méthode display. Bien sûr, une boucle for est utilisée sur la liste de keys membre de l'instrument pour que chacune d'entre elles fassent parti de l'objet Graphic de l'instrument passé au GUI. Notons donc qu'une touche ne peut pas être affichée séparément, seul l'instrument peut être affiché pour éviter certaines erreurs telles qu'une note par-dessus une autre et cetera.

3.7 Générer le gabarit guitare

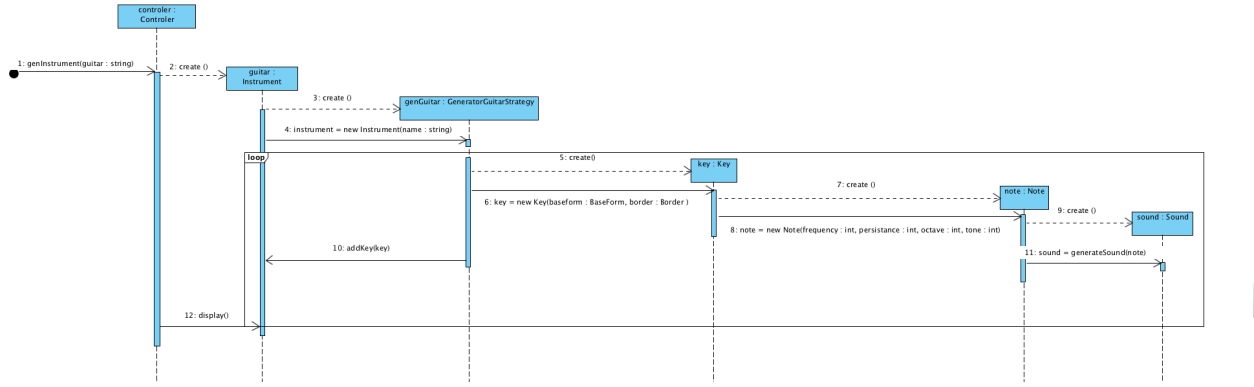


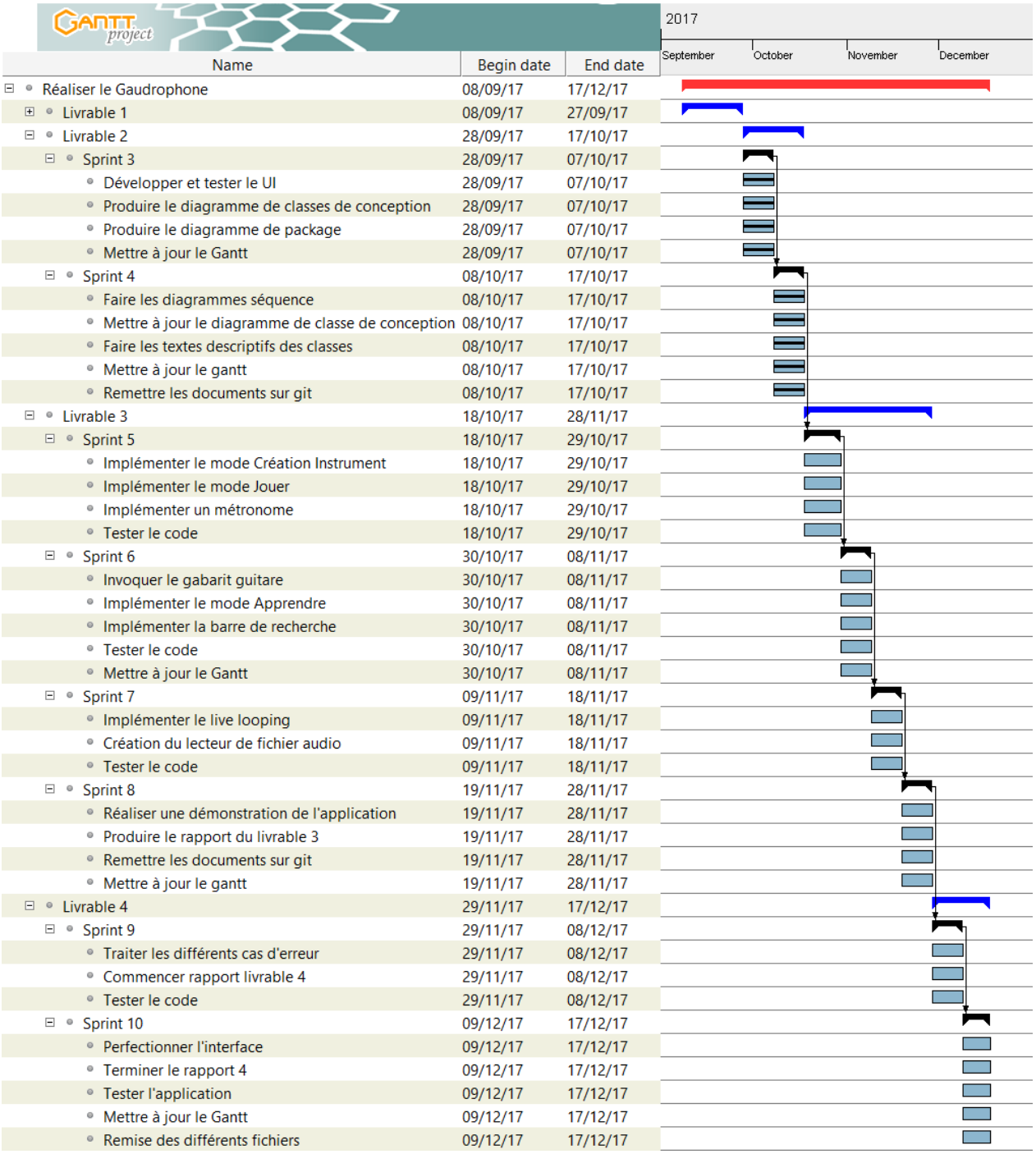
FIGURE 3.7 – Diagramme séquence : 3.7

Lorsque le contrôleur reçoit la commande generateInstrument avec comme argument la string guitar, il va appeler la méthode create de la classe Instrument. Ensuite, la stratégie GenerateGuitar sera appelée en raison de la string guitar passée dans le generateInstrument. Par la suite, la stratégie prendra une note fondamentale d'une guitare et appellera le constructeur de note en boucle en appliquant des variations selon l'index de la boucle. Ceci aura pour effet de générer des notes avec des propriétés physiques différentes qui seront retournées sous la forme d'une liste de notes de l'instrument. Finalement, le controleur recevra la commande de faire l'affichage (display) de l'instrument généré.

Chapitre 4

Planification

4.1 Diagramme de Gantt



Name	Begin date	End date
Réaliser le Gaudrophone	08/09/17	17/12/17
Livrable 1	08/09/17	27/09/17
Livrable 2	28/09/17	17/10/17
Sprint 3	28/09/17	07/10/17
Développer et tester le UI	28/09/17	07/10/17
Produire le diagramme de classes de conception	28/09/17	07/10/17
Produire le diagramme de package	28/09/17	07/10/17
Mettre à jour le Gantt	28/09/17	07/10/17
Sprint 4	08/10/17	17/10/17
Faire les diagrammes séquence	08/10/17	17/10/17
Mettre à jour le diagramme de classe de conception	08/10/17	17/10/17
Faire les textes descriptifs des classes	08/10/17	17/10/17
Mettre à jour le gantt	08/10/17	17/10/17
Remettre les documents sur git	08/10/17	17/10/17
Livrable 3	18/10/17	28/11/17
Sprint 5	18/10/17	29/10/17
Implémenter le mode Création Instrument	18/10/17	29/10/17
Implémenter le mode Jouer	18/10/17	29/10/17
Implémenter un métronome	18/10/17	29/10/17
Tester le code	18/10/17	29/10/17
Sprint 6	30/10/17	08/11/17
Invoquer le gabarit guitare	30/10/17	08/11/17
Implémenter le mode Apprendre	30/10/17	08/11/17
Implémenter la barre de recherche	30/10/17	08/11/17
Tester le code	30/10/17	08/11/17
Mettre à jour le Gantt	30/10/17	08/11/17
Sprint 7	09/11/17	18/11/17
Implémenter le live looping	09/11/17	18/11/17
Création du lecteur de fichier audio	09/11/17	18/11/17
Tester le code	09/11/17	18/11/17
Sprint 8	19/11/17	28/11/17
Réaliser une démonstration de l'application	19/11/17	28/11/17
Produire le rapport du livrable 3	19/11/17	28/11/17
Remettre les documents sur git	19/11/17	28/11/17
Mettre à jour le gantt	19/11/17	28/11/17
Livrable 4	29/11/17	17/12/17
Sprint 9	29/11/17	08/12/17
Traiter les différents cas d'erreur	29/11/17	08/12/17
Commencer rapport livrable 4	29/11/17	08/12/17
Tester le code	29/11/17	08/12/17
Sprint 10	09/12/17	17/12/17
Perfectionner l'interface	09/12/17	17/12/17
Terminer le rapport 4	09/12/17	17/12/17
Tester l'application	09/12/17	17/12/17
Mettre à jour le Gantt	09/12/17	17/12/17
Remise des différents fichiers	09/12/17	17/12/17

FIGURE 4.1 – Diagramme de Gantt

Chapitre 5

Contribution des membres

Le diagramme de classe de conception a été réalisé par toute l'équipe. Chaque membre de l'équipe a contribué à la réalisation de chaque diagramme de séquence. Tout le monde a aidé à créer l'interface graphique de l'application. Le diagramme de package a été fait par l'ensemble des membres. Le Gantt a été mis à jour par Benjamin, Olivier et Édouard. La mise à jour du livrable 1 a été fait par l'ensemble de l'équipe. Le document Latex a été rédigé par Olivier et Benjamin.

Bibliographie

[1] Larousse. *Dictionnaire Français*, [En ligne]. http://www.larousse.fr/dictionnaires/francais/octave (Page consultée le 11 septembre 2017).

[2] Larousse. *Dictionnaire Français*, [En ligne]. http://www.larousse.fr/dictionnaires/francais/note (Page consultée le 11 septembre 2017)

[3] Larousse. *Dictionnaire Français*, [En ligne]. http://www.larousse.fr/dictionnaires/francais/temps (Page consultée le 11 septembre 2017)

[4] Larousse. *Dictionnaire Français*, [En ligne]. http://www.larousse.fr/dictionnaires/francais/timbre (Page consultée le 11 septembre 2017)

[5] Larousse. *Dictionnaire Français*, [En ligne]. http://www.larousse.fr/dictionnaires/francais/frequence (Page consultée le 11 septembre 2017)

[6] Larousse. *Dictionnaire Français*, [En ligne]. http://www.larousse.fr/dictionnaires/francais/persistance (Page consultée le 11 septembre 2017)

[7] Larousse. *Dictionnaire Français*, [En ligne]. http://www.larousse.fr/dictionnaires/francais/alteration (Page consultée le 11 septembre 2017)

[8] Larousse. *Dictionnaire Français*, [En ligne]. http://www.larousse.fr/dictionnaires/francais/gabarit (Page consultée le 11 septembre 2017)

[9] Larousse. *Dictionnaire Français*, [En ligne]. http://www.larousse.fr/dictionnaires/francais/rythme (Page consultée le 11 septembre 2017)

[10] Larousse. *Dictionnaire Français*, [En ligne]. http://www.larousse.fr/dictionnaires/francais/partition (Page consultée le 11 septembre 2017)

[11] Larousse. *Dictionnaire Français*, [En ligne]. http://www.larousse.fr/dictionnaires/francais/tonalite (Page consultée le 11 septembre 2017)

[12] Larousse. *Dictionnaire Français*, [En ligne]. http://www.larousse.fr/dictionnaires/francais/gamme (Page consultée le 11 septembre 2017)

Annexe A

Énoncé de vision

À la racine de tout être humain somnole un artiste qui ne demande qu'à s'exprimer. L'informatique aura su au fil du temps devenir une alliée indétrônable et voilà qu'une fois de plus, elle surprendra avec une nouvelle application révolutionnaire. Le Gaudrophone permet donc à un usager de réveiller sa fibre musicale et de s'improviser maître dans l'art. Il sera pour lui le meilleur instrument de musique qui soit : il pourra à la fois le créer et apprendre à en jouer, puis partager au monde son talent.

Chaque son que l'instrument produit peut être créé par l'usager et il est même possible d'y intégrer les caractéristiques sonores d'un instrument existant. En plus d'une vaste personnalisation visuelle lors de sa création, sa sonorité peut être ajustée à la vision de celui qui en joue. Les sons peuvent être modulés à la guise de l'artiste : les critères vont de la fréquence à la tonalité en passant par la persistance. Également, l'option d'importer sa propre bibliothèque sonore n'est pas oubliée, sans compter les instruments préprogrammés. Ainsi, le néophyte pourra s'amuser à concevoir de l'instrument de son rêve en jouissant d'une liberté totale.

Pour laisser libre cours à l'inspiration musicale, le Gaudrophone permet une liberté artistique sans précédent. En effet, un mode libre intégrant des méthodes prisées par des célébrités populaires telles que le Live looping est offerte. Jouer d'un instrument virtuel n'aura jamais été aussi agréable et intuitif. Pour garder le rythme, un métronome omniprésent est en plus offert. Son activation relève toutefois de la discrétion de l'usager et celui-ci pourra régler la cadence selon son besoin.

Pour suivre les modes en vogue, le projet sera doté d'une interface d'entraînement. Pensons au monde du jeu vidéo qui aura su tenter tout un chacun à plonger dans l'univers musical : ici, le but est le même. Un mode de jeu stimulant et coloré permettra à un utilisateur de peaufiner ou de débuter sa maîtrise de l'instrument qu'il aura choisi ou créé. Apprendre à jouer d'un l'instrument avec le Gaudrophone signifie voir les notes d'une partition s'afficher à l'écran. Au rythme de la musique, l'artiste en devenir devra faire résonner son instrument comme dicté par l'application. En parfait synchronisme avec la partition, il devra appuyer sur les touches pour apprendre cette chanson. Afin de reproduire le plus fidèlement possible la chanson choisie, il pourra également visionner son instrument la jouer, puis le répéter à son tour.

Ce projet d'envergure s'échelonnera sur une durée de quatre mois. Malgré son caractère

purement académique, le but de l'équipe est de fournir un artéfact de qualité qui saura montrer le savoir et la persévérance de quatre jeunes étudiants débutant leur baccalauréat dans le cadre du cours Génie Logiciel Orienté Objet. Ce quatuor du programme éponyme offrira une application digne des préceptes de l'Université Laval.

Annexe B

Modèle du domaine

B.1 Diagramme du modèle du domaine

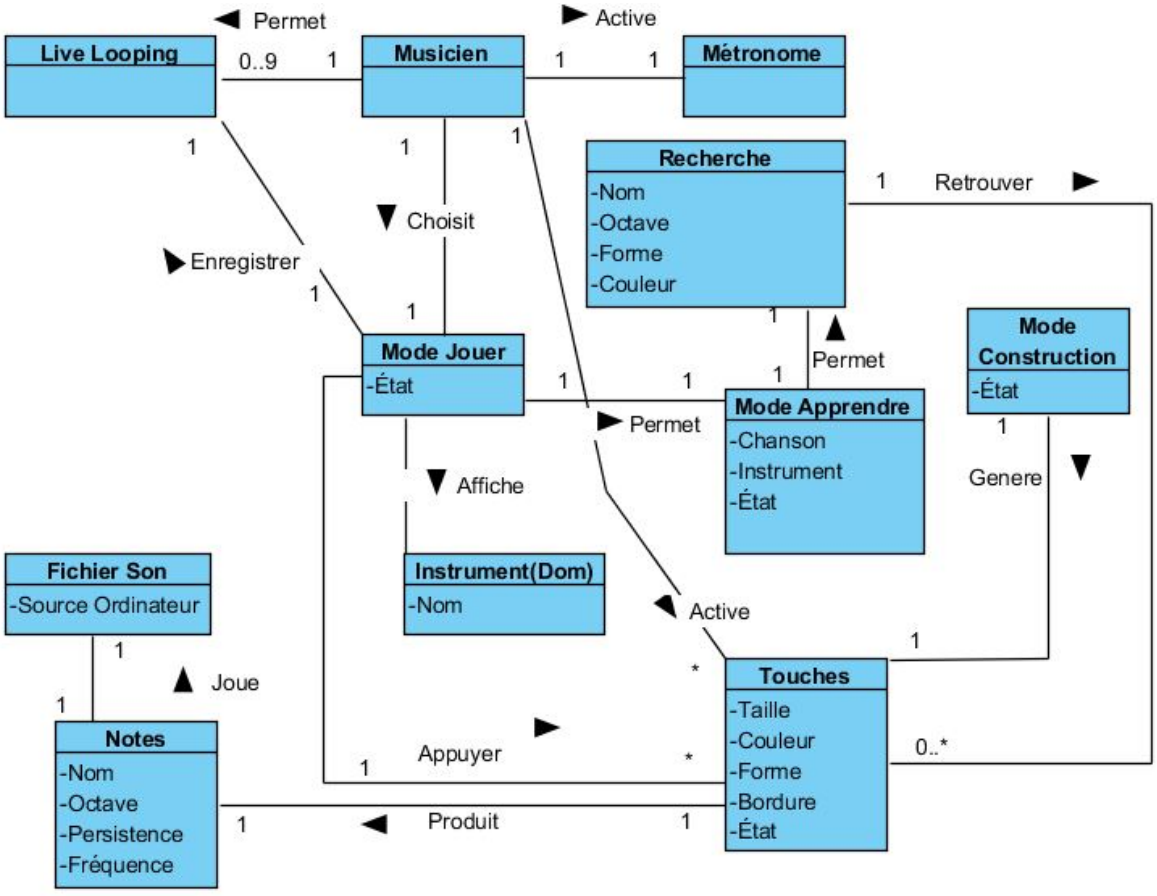


FIGURE B.1 – Diagramme du modèle du domaine

Annexe C

Glossaire

Tableau C.1 – Glossaire

Terme	Définition
Octave [1]	Un octave est composé de sept notes (do, ré, mi, fa, sol, la, si, do). Quand on monte d'un octave, la fréquence d'une note particulière double.
Note [2]	Une note est une fréquence de son particulière qui sonne bien à l'oreille (do, ré, mi, fa, sol, la, si, do). De plus, une note donne aussi sa durée en temps.
Temps [3]	C'est un découpage du rythme. Par exemple, une noire peut être suivi d'une blanche ou d'une croche.
Timbre [4]	Qualité unique du son indépendante de l'intensité et de la hauteur. Le timbre est spécifique à chaque instrument et de la voix qui l'émet. Dans le cadre de ce projet, c'est le nombre de fréquences harmoniques dans le son.
Fréquence [5]	C'est le nombre de vibrations par seconde d'une onde. Son unité est le Hertz.
Fréquence harmonique	C'est le nombre de vibrations par seconde d'une onde fondamentale. Son unité est le Hertz.
Persistance [6]	Une fois que l'on arrête de jouer une certaine note, elle représente la durée de temps qu'elle est entendue.
Altération [7]	Changement d'une note d'un demi-ton. Si on monte d'un demi-ton, cela représente un dièse et si on descend d'un demi-ton, cela représente un bémol.
Gabarits [8]	Ensemble de sons que l'on peut importer d'un seul coup.
Rythme [9]	Cela est la vitesse de la chanson.
Partition [10]	Représentation visuelle d'une chanson. Elle contient la tonalité, la fréquence, l'enchaînement et la durée de chaque note.
Tonalité [11]	Un ton est un saut qu'on fait pour passer d'une note à une autre.
Gamme [12]	Succession ordonnée des différents degrés d'une tonalité.
Live looping	Jouer un segment de sons à répétition dans le but de faire une chanson. Exemple : rythme de batterie, morceau de guitare, puis chanter par-dessus.
Touche	Un bouton de l'interface du Gaudrophone qui permet de jouer un son.

Annexe D

Modèle des cas d'utilisation

D.1 Diagramme du modèle des cas d'utilisation

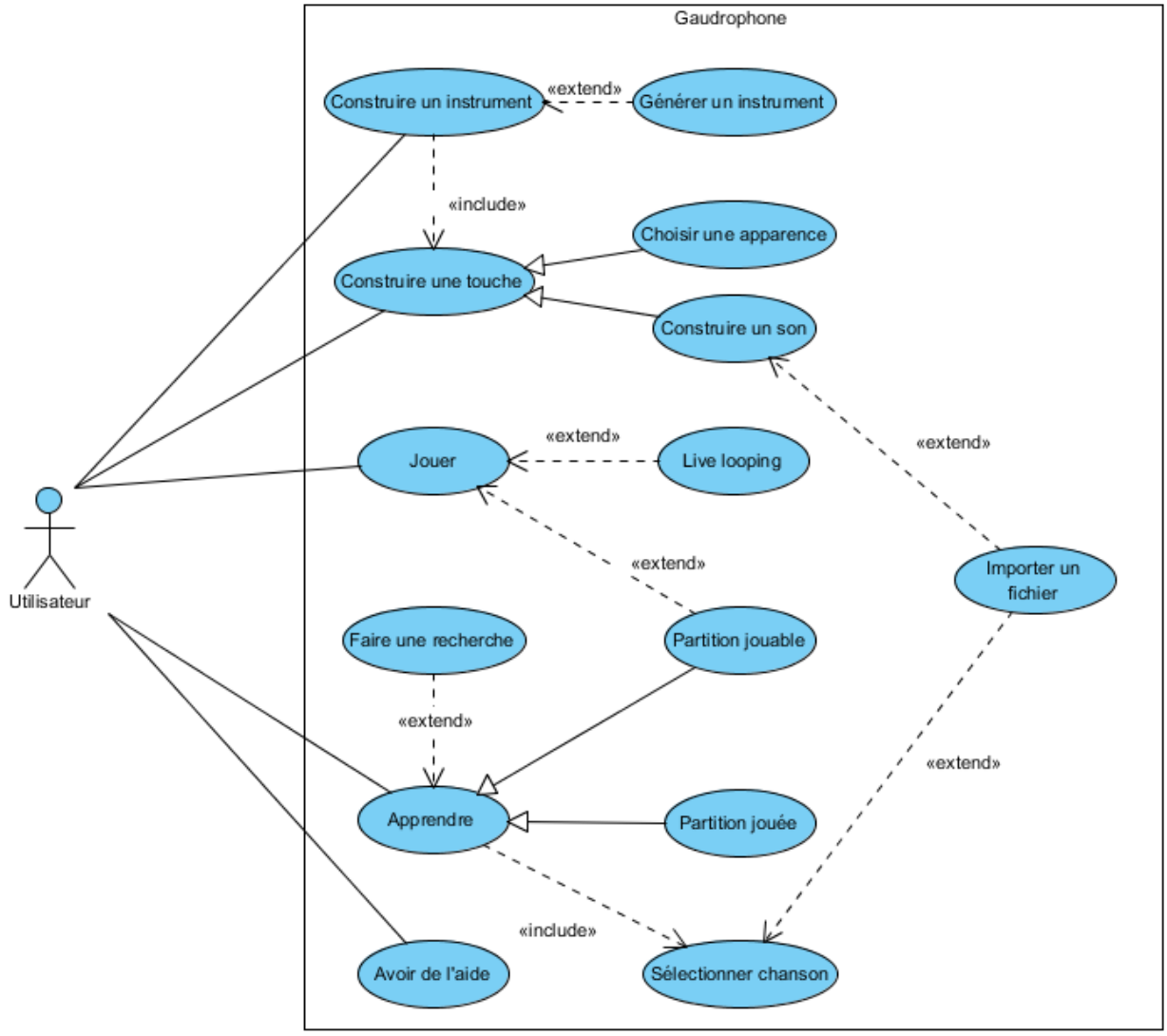


FIGURE D.1 – Diagramme du modèle des cas d'utilisation