

**Rapport de projet tuteuré (2016/2017)**

« Créer un jeu alliant hardware et software afin d’apprendre la programmation Arduino de manière ludique »

**Tutrice : Amélie Cordier**

Adrien Techer

Valentin Guevara

Jérémy Duval

Fanny Velien

# Introduction

## Contexte & Tuteur

## Présentation du projet

## Organisation

### GitHub

### Responsabilités des membres

## Droits

### Contenus multimédias et code non crées

### Licence

## Préconception

# Technologies utilisées

## Matériel Electronique

### Arduino

### Grove

## La librairie LibGDX

## La librairie RXTX

# Mécaniques du jeu

## Organisation du code et logique

### Design Pattern

## Menus et changement d’écran

## Génération du monde

### Génération du Terrain

### Génération des objets

#### Types d’objets, interactions et utilité

#### Processus de création et méthode

## Les entités vivantes

### Créatures mobiles

#### Génération

#### Différents types et caractéristiques

### Joueur

#### Génération

#### Caractéristiques

## Arduino

### Interactions

## Système d’apprentissage

### Cours

### Défis

# Contenu Multimédia

## Gestion dans le Jeu

## Graphismes

### Menus et Typographie

### Textures

## Musiques & Sons

### Musiques

### SFX

# Les obstacles rencontrés

## Organisation

## Intégration Continu

## Débogage

# Futur Développement

## Interactions avec Arduino & Défis

## Objectifs de Jeu

## Univers

# Bilans Personnels

## Jérémy

## Valentin

## Fanny

## Adrien

# Conclusion

## Bilan Général

## Remerciements

# Annexes

## Diagrammes UML

# Introduction

## Contexte & Tuteur

## Présentation du projet

## Organisation

### GitHub

### Responsabilités des membres

## Droits

### Contenus multimédias et code non crées

### Licence

## Préconception

# Technologies utilisées

## La librairie LibGDX

## La librairie RXTX

Afin de permettre un interfacage efficace entre notre carte Arduino et le programme JAVA, nous avions besoin d’une librairie capable d’intéragir directement avec les composants. A ce moment précis, après de nombreuses recherches sur Internet et notamment sur le site internet officiel d’Arduino, nous avons trouvé un ensemble de fonctions permettant de réaliser l’intégralité de ces actions.

Cette librairie se nomme « RXTXComm.jar » .

<http://playground.arduino.cc/Interfacing/Java>

Elle nous permet premièrement d’interroger les ports séries(cf…) à travers des méthodes déjà écrites mais également d’envoyer des informations ou en recevoir plus facilement.

L’utilisation de ces librairies s’est avérée essentiel pour nous permettre un développement optimisé et plus rapide.

Un des nombreux avantages de cette bibliothèque logicielle était son universalité, elle est compatible avec tous systèmes d’exploitation et est capable de trouver les ports de communication de tous les SE. De plus, de par sa célébrité, bon nombre d’exemples existent en ligne et nous ont permis d’être plus efficace dans notre phase de développement.

# Mécaniques du jeu

## Organisation du code et logique

### Design Pattern

## Menus et changement d’écran

## Génération du monde

### Génération du Terrain

### Génération des objets

#### Types d’objets, interactions et utilité

#### Processus de création et méthode

## Les entités vivantes

### Créatures mobiles

#### Génération

#### Différents types et caractéristiques

### Joueur

#### Génération

#### Caractéristiques

## Arduino

1. Relation Software-Hardware
2. La partie programmation

Figure 1-Logo Arduino

1. Grove
2. Amélioration : La création d’un parseur

*Pourquoi Arduino ?*

L’utilisation de la technologie Arduino était une nécessité. Afin de familiariser l’utilisateur avec les concepts initiaux de la programmation de composants électroniques, nous étions à la recherche d’un système adaptable et ludique. En tenant compte de ces attentes, nous avons entamé une recherche de périphériques facilement programmables et compatibles avec la carte Genuino Uno que nous allions utiliser pour le projet.

Ce choix était, avant tout, une décision commune. Durant notre parcours pré-universitaire, nous avons tous côtoyé à un moment ou un autre le monde merveilleux d’Arduino. Le projet tutoré du semestre 1 était pour la plupart des membres du groupe une première approche au sein de l’IUT à cet environnement électronique que nous voulions absolument découvrir en profondeur.

Jérémy et Valentin avaient déjà travaillé avec cette technologie sur des projets pré-baccalauréat et avaient par conséquent une première expérience et certains réflexes de programmation qui se sont avérés forts utiles lors de la phase de développement de la partie électronique. Pour toutes ces raisons, Arduino nous semblait un choix judicieux et plus particulièrement une carte : Genuino 101. Elle a pour avantage d’intégrer nativement un gyroscope mais aussi une possibilité de connexion Bluetooth. Pour son côté pratique et son aspect « plug-and-play », ce microcontroleur s’est imposé comme une évidence.

*Relation Software-Hardware*

Nous étions guidés par cet objectif de transmettre la connaissance nécessaire au joueur afin de s’amuser comme nous l’avons fait avec ces différents outils Arduino. Nous n’avions pas pour ambition de le porter à un haut niveau mais de lui permettre d’acquérir un niveau débutant suffisant pour commencer à tester les différents composants.

Pour parvenir à cet objectif-là, notre plan d’action était le suivant :

* Donner la possibilité de lire des cours
* Appliquer l’apprentissage à travers des questions en jeu
* Les bonnes réponses permettront de débloquer des parties du jeu

La finalité étant de pouvoir demander à l’utilisateur de saisir un petit programme qui permettrait par exemple d’allumer et éteindre une LED.

A travers la démonstration, nous avons permis un premier contact entre ces deux univers. Effectivement, le joueur doit pouvoir répondre à une question portant sur un programme Arduino afin de débloquer l’inversion de gravité et lui permettre de terminer le premier niveau.

Notre second objectif était de développer l’expérience en jeu pour surprendre l’utilisateur et lui montrer que les possibilités apportées par ce microcontroleur sont infinies. Nous avons alors mis en place une liaison en continue avec les périphériques Arduino qui permettent de :

* Tenir informé de la vie de Genuini[[1]](#footnote-1)
* Afficher sur l’écran LED l’écran actif
* Permettre une interaction immédiate entre le monde physique et logiciel

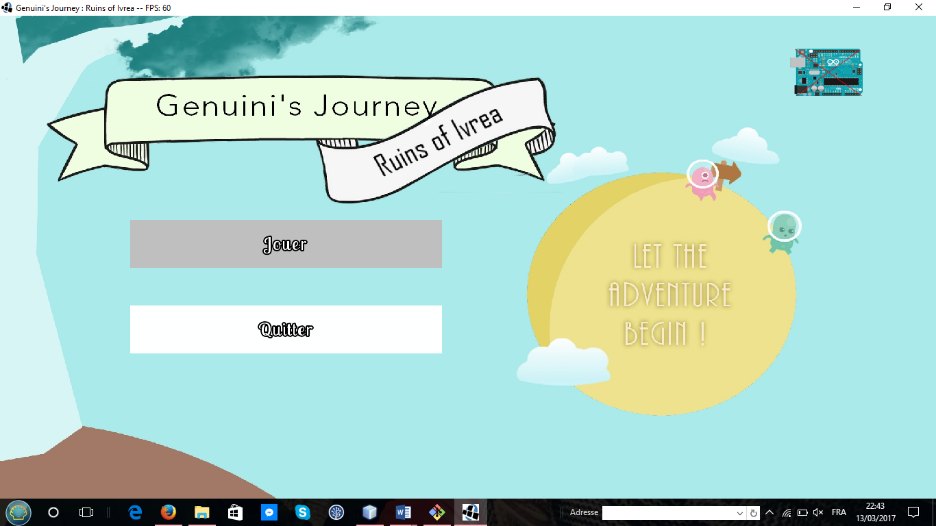
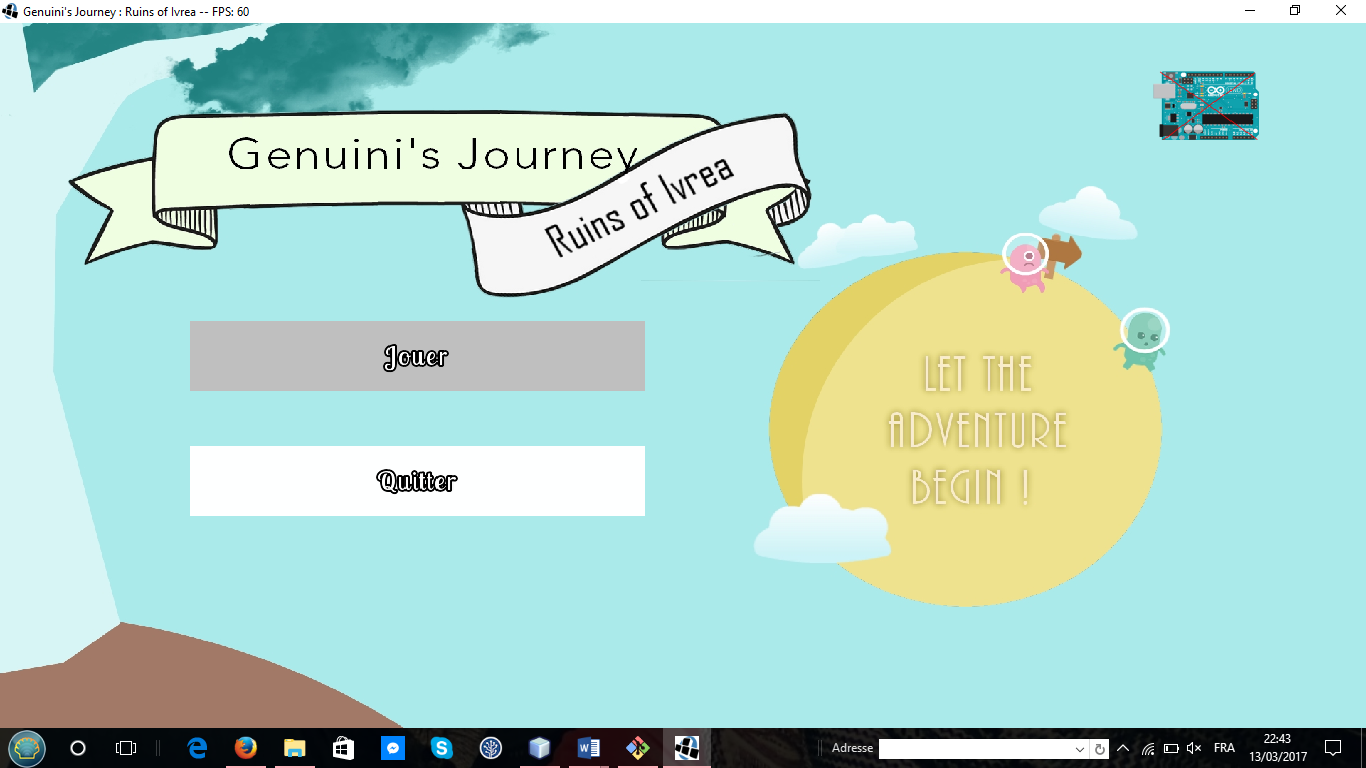
Le jeu devait pouvoir être également utilisable sans connexion entre les composants et l’ordinateur, nous avons alors implémenté la détection des périphériques et les actions qui seront effectuées en conséquence.

Figure 2-Liaison Arduino-Jeu non détectée

*Partie programmation*

Comme expliqué précédemment, l’environnement Arduino était un univers à approfondir pour certains et à découvrir pour d’autres. L’objectif premier était une collaboration sur toutes les parties du projet par tous les membres du groupe. Le développement Hardware n’a donc pas dérogé à la règle et a interessé de par son aspect ludique toutes les parties.

Le langage de programmation utilisé dans l’univers Arduino est le C/C++. Ayant déjà suivi des cours sur ce langage, la manière de développer n’a donc pas été une surprise pour nous.

Pour une réalisation optimale, nous avons travaillé sur l’IDE du constructeur nommé Arduino IDE. Son apparence ne diffère guère de nombreux autres environnements de développement, il a pour avantage de mettre en avant une interface claire et épuré permettant une compréhension rapide des différentes fonctionnalités du logiciel.

Son principal inconvénient reste la non présence de coloration syntaxique. A ce problème, une solution existe. Il est possible de coder sur un environnement différent appelé SublimeText qui lui prend en compte de nombreuses aides aux développeurs. Un plugin présent dans sa librairie de contenu additionnel permet à l’utilisateur de compiler et téléverser son programme par l’intermédiaire de ce logiciel à son microcontroleur.

<http://eskimon.fr/2224-arduino-mini-tuto-utiliser-sublime-text-ide>

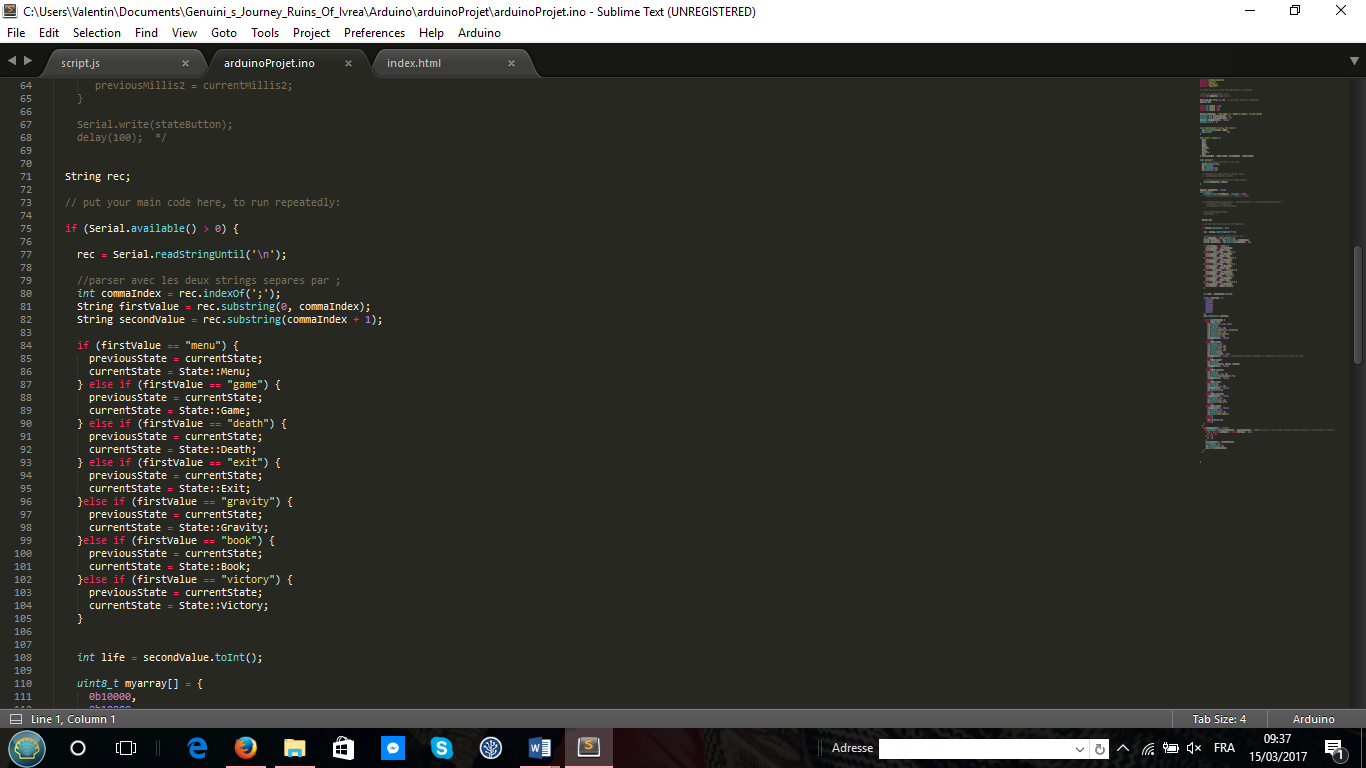
Le point essentiel du développement était la réussite de l’interfacage entre notre code Arduino et le jeu exécuté sur un ordinateur. Il s’est alors posé la question de la manière d’obtenir un résultat satisfaisant entre ces deux appareils.

Un port permettant la liaison entre le microcontroleur et le PC existe, il se nomme « Port COM ». La connexion série permet une relation d’entrée-sortie entre les deux appareils aux extrémités de la liaison. C’est une norme disponible sur tous les périphériques actuels et est utilisable via les connectiques USB. Une machine comporte plusieurs ports COM mais en attribue un seul pour le périphérique désiré.

*« Notre programme devait pouvoir faire face à tous types de situations. »*

Une de ces difficultés était la détection de la connexion entre le module Arduino et l’ordinateur. Pour parer ce problème, nous avons donc parcouru un tableau contenant tous les ports COM de la machine et avons vérifié si le microcontroleur était présent. La suite allait de soi, si le module est connecté nous le préparons pour recevoir et envoyer des données sinon le jeu informe l’utilisateur de la non détection du périphérique.

Par conséquent, afin de transmettre des informations de vie du personnage ou bien d’écran actif, nous utilisons une méthode : « InstanceDeLArduino.write(« infos ») » qui envoie sur le port série une séquence de bits correspondant à l’action que le microcontroleur va devoir effectuer. Nous devions rendre possible l’envoi de plusieurs informations en simultanée à l’Arduino. Malheureusement, la liaison série ne peut contenir qu’une seule suite de données, une chaîne de caractères. Il a donc fallu diviser notre chaîne à l’aide de point virgule afin que le programme côté microcontroleur puisse l’interpréter et effectuer les actions en conséquence.

Du côté Hardware, une énumération des écrans ainsi qu’une liste des différentes instructions permettent au code Arduino d’entretenir un contact réel avec le gameplay. Pour traiter les informations reçues, nous avons séparé la chaîne de caractère afin de pouvoir par exemple récupérer la valeur de la vie du personnage et ensuite l’écran actif du jeu.

Afin d’illustrer notre propos, soumettons un exemple concret qui est l’affichage de l’écran de mort. Une fois la vie valant 0 la partie logicielle s’occupe, de changer d’écran en indiquant au joueur la fin de la partie mais également, de lancer une musique particulière. A ce moment précis, un petit écran LED relié à la carte Arduino inscrit un message de mort et change de couleur pour passer en rouge afin d’augmenter le côté dramatique de la situation.

Finalement, de la même manière que la programmation JAVA, le développement Arduino a constitué une part importante dans le processus de réalisation du projet. Nous avons dû nous coordonner afin de parvenir à effectuer toutes les actions voulues.

Pour conclure, le projet à travers un univers Arduino s’est avéré être une excellente idée et ce microcontroleur a su convaincre tous les membres du groupe.

*Grove*

Afin de rendre les branchements électroniques plus faciles pour l’utilisateur, nous cherchions des composants additionnels compatibles avec notre solution Arduino et obéissant à ce principe que nous avions fixé : « plug-and-play ». Après de multiples recherches, nous avons découvert l’univers Grove. Il s’agit de modules offrant la possibilité d’accroitre les capacités de la carte en y ajoutant pour l’exemple des capteurs ou bien des écrans. L’avantage de ces outils est la librairie prête à l’utilisation fournie par l’entreprise SeedStudio comportant toutes les fonctions nécessaires à l’exploitation complète de ces modules.

<http://wiki.seeed.cc/Grove_System/>

Afin de proposer au joueur une multitude d’interaction avec le milieu Hardware, nous avons opté pour un pack comprenant les capteurs que nous souhaitions puis nous avons fait l’acquisition également d’une bargraphe LED pour permettre un affichage sur une Interface Homme-Machine de la vie du personnage.



Voici une liste exhaustive des différents modules présents dans ce pack :

* Base shield : A connecter à l’Arduino pour rendre disponible Grove
* LCD RGB BlackLight : Ecran Led pour l’écriture de textes
* Sound sensor : Capte le son ambiant
* Touch sensor : Petite interface tactile
* Temperature sensor : Retranscrit la température extèrieure à l’Arduino
* LED : Une Led de chaque couleur
* Button : Un bouton poussoir pour affecter des actions spécifiques

<http://wiki.seeed.cc/Grove_Starter_kit_for_Arduino_101/>

*Finalement, Grove offre de nombreuses possibilités d’interactions ludiques et immédiates entre les programmes Hardware et Software.*

*Amélioration : La création d’un parseur*

Nous avions un idéal dans ce projet, mettre en œuvre un parseur du côté Arduino.

Parseur : « Parser » signifie effectuer une action spécifique adéquate à partir d’un texte reçu dans un langage pré-établi.

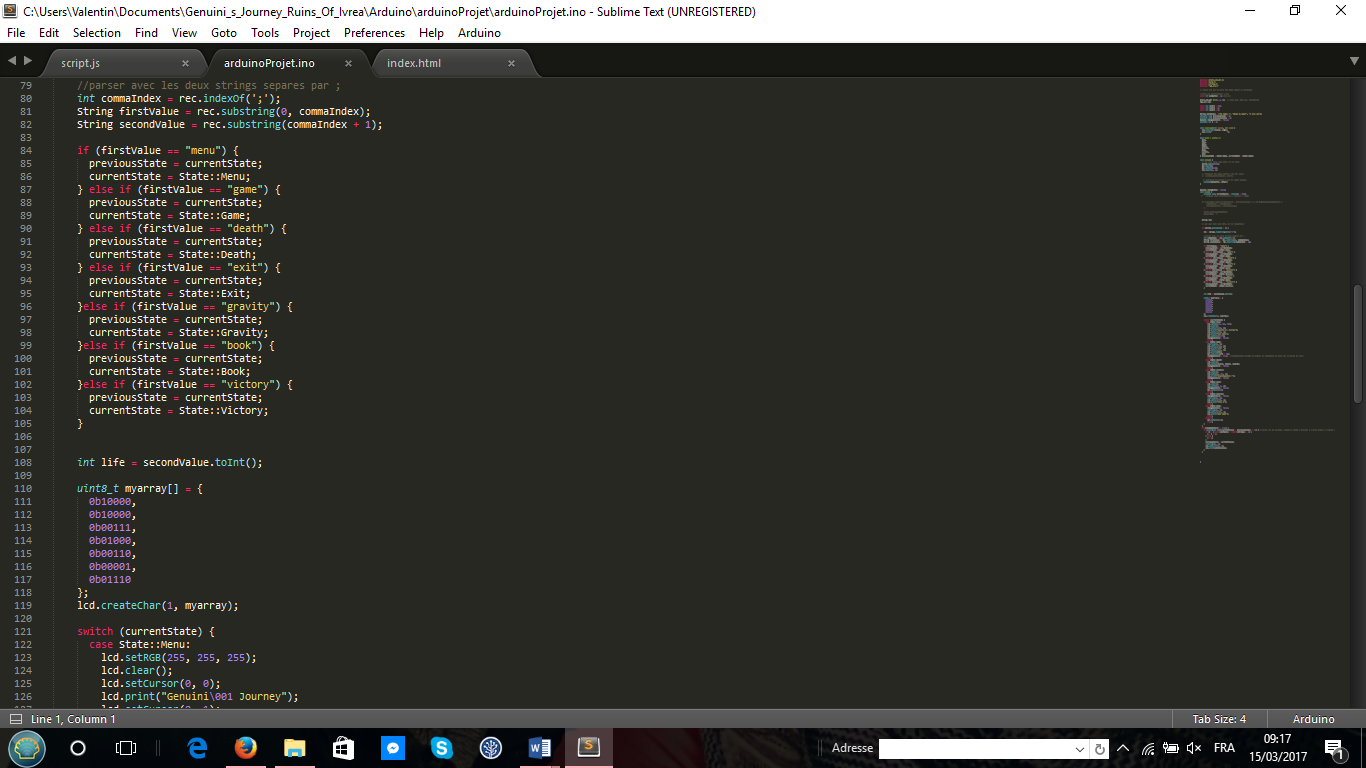
Par la suite, dans la continuité de notre développement, nous mettrons en place dans notre programme Arduino un système permettant la compréhension du langage C/C++ écrit dans le jeu par le microcontroleur. Afin de créer les premières activités d’apprentissage de l’utilisateur, nous avons commencé par vérifier si une ligne saisie par le joueur correspond à la ligne attendue pour compléter le niveau. De cette manière, notre réalisation est en adéquation avec le cahier des charges mais avant tout avec l’univers que nous voulions créer.

Figure 3-Conditions liées à la réception des données

*Afin de conclure pour la partie Arduino, nous pouvons tous affirmer avoir apprécié la conception, la programmation mais également l’interaction que nous a procuré cette liaison entre une partie logicielle et une partie Hardware. Dans un futur même proche, nous aimerions travailler dans un secteur proche de celui-ci. Et si jamais cela ne s’avérait pas être le cas, nous expérimenterons de nouveau ce lien à travers des projets personnels qui permettront de maitriser de plus en plus le monde Arduino.*

*De plus, nous aimerions poursuivre ce jeu par la suite après la soutenance.*

## Système d’apprentissage

### Cours

### Défis

Les défis sont le point clef du jeu.

Ils nous permettent de tester nos connaissances sur les cours, vu précédemment par le joueur ; ce sont eux qui vont permettre l’apprentissage d’Arduino.

Le principe est simple : le joueur se retrouve confronté à un problème dans le jeu. Il est alors poussé à ouvrir le grimoire dans lequel l’attend un défi. Celui-ci, une fois résolu va permettre l’accessibilité à une nouvelle fonctionnalité qui, par incidence, relancera la progression du joueur au sein du niveau.

Il y a deux types de défi.

Le premier consiste en une question avec plusieurs propositions de réponse (comme dans le premier niveau). C’est le plus simple pour le joueur mais aussi pour la programmation.

Le second consiste en une question avec un champ texte servant à entrer une ou plusieurs lignes de codes (comme dans le deuxième niveau). L’utilisateur va donc devoir entrer de vraies instructions, comme s’il programmait.

Ces deux solutions ont été construite au fur et à mesure de notre compréhension de LibGDX, de la communication entre Java et Arduino et de notre capacité à structurer et factoriser le code.

En effet, la question avec proposition de réponse est simple à mettre en œuvre. Elle consiste à créer plusieurs boutons et à tester la valeur de celui sur lequel cliquera l’utilisateur.

Le second type de question est plus complexe du fait qu’elle nécessite l’analyse du code saisi par le joueur. Nous comparons donc une chaine de caractère correspondant à la solution à celle contenant la réponse de l’utilisateur. Nous pourrions aussi aller plus loin et créer un parseur Arduino, mais ce serait peu rentable au niveau du temps pour l’instant.

# Contenu Multimédia

## Gestion dans le Jeu

La gestion du jeu se fait de manière simple et efficace.

La classe « MainGame » créée par la classe main (située dans DesktopLauncher) charge en mémoire toutes les musiques, sons et images en mémoire. Le programme ne créé donc les sons qu’une seule fois, au démarrage, évitant des pertes conséquentes de temps -contrairement au cas où nous les chargerions à chaque appel d’un écran-.

« MainGame » se sert de la classe « Content » (du package genuini.game) afin de gérer les musiques, sons et images. Celle-ci permet d’ajouter, de modifier ou de supprimer les différentes listes (sous forme de HashMap). Elle stocke la donnée à laquelle elle attribue une clef correspondant soit à une clef passée en paramètre, soit au nom de cette donéée (par défaut).

Pour utiliser une musique (ou autres), il n’y a plus qu’à appeler un getter avec en paramètre le nom de celle-ci (la clef) au chargement de l’écran correspondant.

## Graphismes

### Menus et Typographie

### Textures

## Musiques & Sons

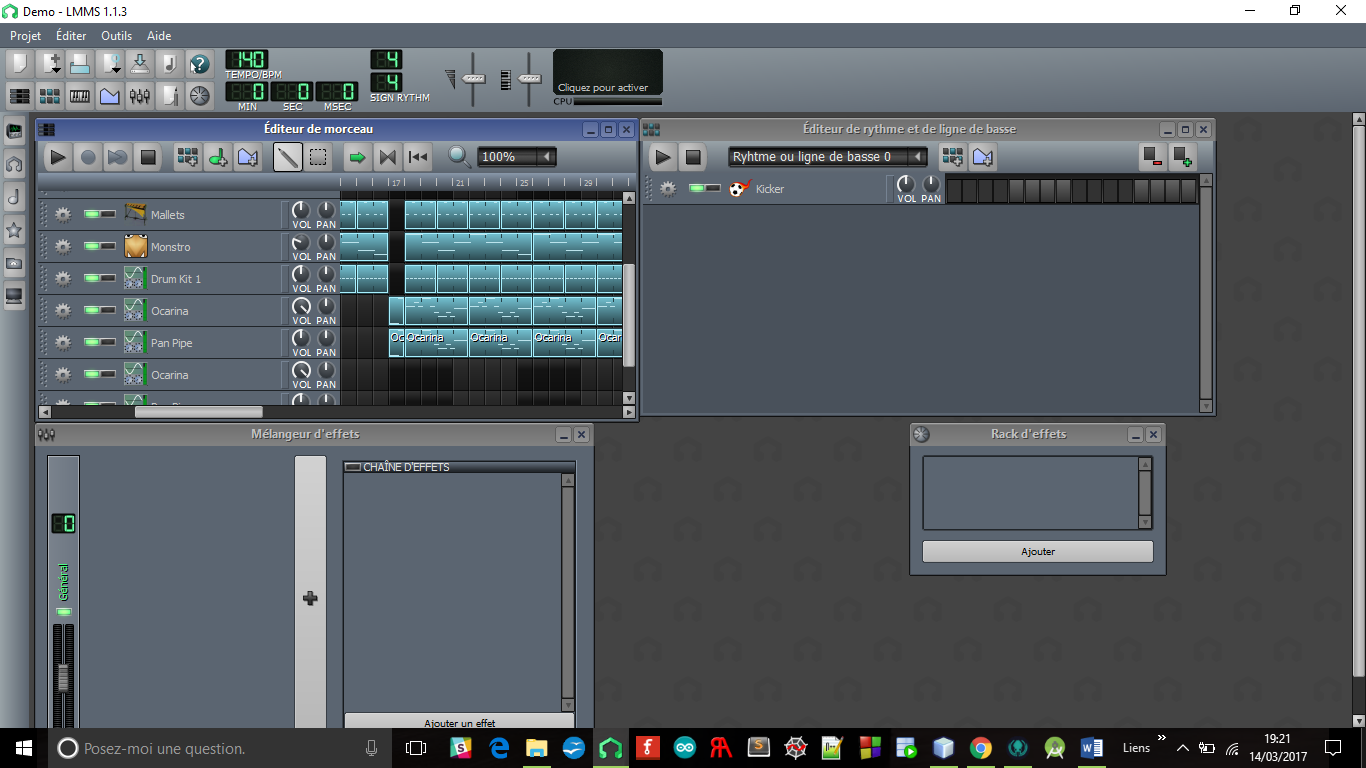
### Musiques

Les musiques sont –à l’instar des graphismes- un point clef d’un jeu vidéo, comme l’explique

Paul-Victor Vettes, alias PV Nova dans ses vidéos (cliquer [ici](https://www.youtube.com/watch?v=7xU3B4uIyaY) et [là](https://www.youtube.com/watch?v=gTC95xzsZmU)). Ceux-ci vont influencer directement sur l’ambiance du jeu et l’état d’esprit du joueur. Comme disait Richard Wagner « La musique commence là où s’arrête le pouvoir des mots ». Ce n’est donc pas une partie négligeable. La musique peut nous permettre de plonger le joueur dans le jeu et donc de le rentre plus réceptif et ouvert à l’apprentissage.

Nous avons dans un premier temps penser à enregistrer nous même les musiques, ayant de nombreux instruments au sein de notre groupe, de types variés. Faute de temps et ayant considéré le code du jeu prioritaire, nous avons donc décider d’abandonner en partie l’idée.

En effet, chaque musique du jeu a été créer à manuellement, à l’aide du logiciel de son LMMS. Ce logiciel comporte un nombre important de sons préenregistrés, une interface simple d’utilisation et intuitive. De plus il a l’avantage d’être gratuit et open source.



Ces musiques sont gérées de manière simple dans le code, à l’aide de LibGDX(cf. 4.1).

### SFX

Les sons et bruitages sont primordiaux dans un jeu vidéo. Ils sont au moins aussi importants que les musiques.

Les sons de notre jeu ont été réalisés à la voix pour des raisons d’efficacité.

Ils ont été enregistrés à l’aide d’un micro et du logiciel Audacity en raison de sa facilité d’utilisation et des connaissance préalable de certain de nos membres.

METTRE UN SCREEN

La gestion des sons dans le programme se fait à l’instar des musiques et des images ; à l’aide des classes « MAinGame » et « Content » (cf. 4.1).

# Les obstacles rencontrés

## Organisation

## Intégration Continu

## Débogage

# Futur Développement

## Interactions avec Arduino & Défis

Lors de multiples réflexions autour du jeu, nous avons pensé à de nombreuses interactions avec Arduino et à de nombreux défis.

Les possibilités sont infinies, Arduino -étant open source- se construit sans cesse. Pour chaque nouveau composant, nous avons la possibilité de développer un nouveau niveau, toujours plus élaboré et par incidence, un nouveau défi.

Voici quelques idées d’interaction :

* Un potentiomètre pourrait nous permettre de régler l’intensité d’un sort et de l’utilisation de point de magie.
* Un potentiomètre pourrait nous permettre de régler la densité de certains objets, de certaines plateformes afin de pouvoir passer à travers ou monter dessus.
* Un potentiomètre nous permettrait de régler l’intensité lumineuse afin de faire apparaitre ou disparaitre des objets ou des ennemis.
* Un capteur photoélectrique pourrait nous permettre de faire la même chose suivant la luminosité qu’il détecte.
* Un capteur d’humidité pourrait faire pleuvoir sur une zone et, par exemple, y faire pousser un arbre afin d’accéder à une plateforme trop haute ; ou de la même manière, assécher une zone afin de créer un passage là où un arbre nous bloquait.
* Un moteur (continu, servomoteur, pas à pas) pourrait nous permettre d’actionner certains mécanismes bloqués dans le jeu comme une manivelle.
* Un gyroscope pourrait nous permettre de faire déplacer certaines choses ou même le personnages en inclinant la carte Arduino.
* Un télémètre nous permettrait de faire soulever le personnage ou des objets (comme des plateformes) en soulevant la carte Arduino.
* Un écran tactile ou plus simplement un écran LCD avec quelques boutons nous permettrait de faire jouer une deuxième personne qui pourrait aider la première dans le jeu, ou de déclencher des mini défis sur ces écrans.
* Un module wifi, Bluetooth, XBee pourrait permettre la communication entre deux cartes et ainsi ouvrir de multitude de possibilité. Le personnage pourrait pas exemple passer entre deux écrans.

Pour chaque type d’interaction il nous faudra des défis. Comme expliqué dans la partie correspondante (cf. 3.6.2) il y a deux familles de défis.

Les défis avec réponse proposés serait plus présent au début de chaque cours afin de tester des connaissances plus théoriques ou n’étant pas directement du code (par exemple une question sur les signaux analogiques et numériques).

Ceux-ci présente de nombreuses possibilités que nous pourrons évaluer à chaque nouveau cours créé.

Les défis dans lesquels il faut programmer offre autant de possibilités voire plus.

En effet, pour chaque cours, les possibilités sont extrêmement grandes, bien qu’il faille faire attention à ne pas être trop répétitif.

Nous avions prévu que ce type de défi devienne de plus en plus complexe et élaborer afin de vraiment amener l’utilisateur vers l’apprentissage.

Cela commencerait donc par une question demandant simplement une ligne de code, comme déclarer une variable, initialiser une broche, allumer une led… Pour finir sur des défis où le joueur devra saisir un programme entier ; en entrant lui-même le squelette du programme et des fonctions.

## Objectifs de Jeu

## Univers

# Bilans Personnels

## Jérémy

## Valentin

Lors de la première réunion, nous cherchions une idée commune, une ambition qui pourrait nous apporter l’envie de programmer en continue durant une année. Nous nous étions tous retrouvés sur la possibilité de jouer et d’apprendre dans un même temps. N’y a-t-il pas de meilleur moyen d’assimiler de nouvelles notions ?

Depuis mes débuts dans le milieu de l’informatique, la programmation de composants électroniques s’est imposée à moi, et plus particulièrement les systèmes Arduino ainsi que leurs dérivés. Notre but était de faire ressentir aux utilisateurs ces sensations que nous-mêmes avons éprouvées et éprouvons encore.

Je retire de cette réalisation de projet un sentiment positif. Dès la phase de conception, nous étions tous les quatres motivés pour parvenir à arriver au bout de cet objectif que nous nous étions fixé. Dans toute la continuité du projet, nous nous sommes heurtés à de nombreux problèmes (Travis, Bug de librairie…etc). J’ai pû tirer des enseignements de tous ces soucis et je saurai désormais réagir efficacement à ce type de situation dans un contexte professionnel.

Le semestre 3 a été, pour nous, le début de la phase de développement. J’avais hâte de programmer dans le langage JAVA. J’ai pour objectif dans un futur proche d’en faire mon langage de programmation quotidien, j’apprécie son côté haut niveau et le développement orienté objet. Son utilisation dans ce projet a donc été une excellente nouvelle d’un point de vue personnel.

Il est maintenant l’heure des conclusions. En ce qui me concerne, j’éprouve certains regrets, j’ai plutôt eu l’habitude de coder dans la précipitation. Sans paraitre arrogant et en tenant compte de l’avis de mes collègues, il me semble avoir bien avancé dans mes parties tout de même. Si je devais revenir en arrière, je programmerais dans la durée en étalant le travail. Mais, est-ce que cela aurait été possible ? Nous ne le serons pas. Toutefois, je suis fier de ce que j’ai réussi à accomplir et de ce que NOUS avons réussi à réaliser.

Pour conclure, cette aventure a été une réussite collective bien plus que personnelle. J’ai adoré travailler, coder, jouer aux côtés de mes trois compagnons et pouvoir communiquer avec notre tutrice.

Ce projet ne pouvait pas mieux correspondre à ce que j’espérais d’un travail en commun et j’ai énormément appris.

Je remercie donc mes collègues ainsi que Mme Cordier mais aussi toutes les personnes qui nous ont permis de mener à bien ce projet.

## Fanny

## Adrien

# Conclusion

## Bilan Général

## Remerciements

# Annexes

## Diagrammes UML

1. Genuini : Personnage principal du jeu [↑](#footnote-ref-1)