Genuini's Journey

Ruins of Ivrea

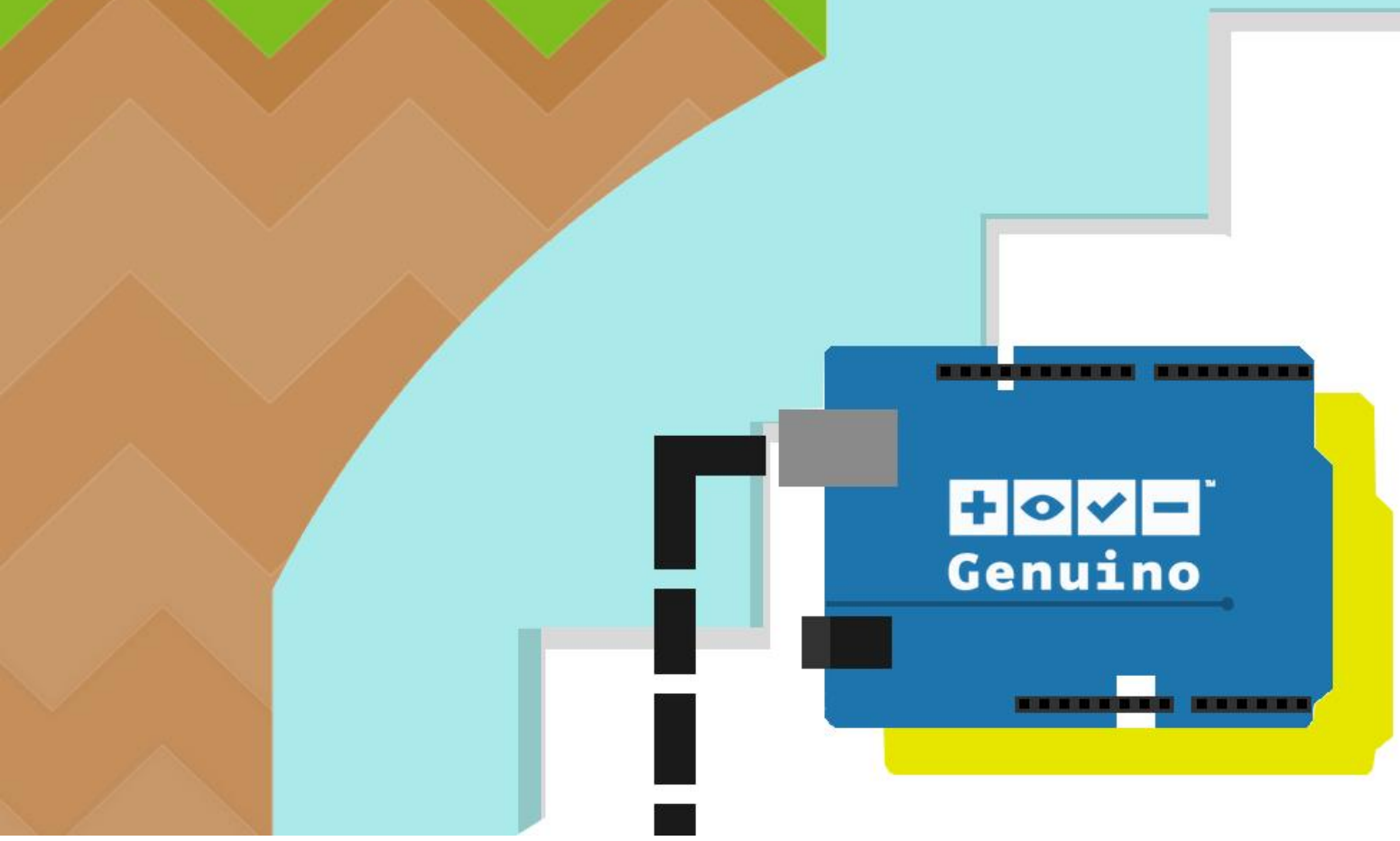
Tutrice : Amélie Cordier

Adrien Techer

Valentin Guevara

Jérémy Duval

Fanny Velien



IUT Lyon 1

l'excellence technologique

Rapport de projet tuteuré (2016/2017)

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

TABLE DES MATIERES

1 INTRODUCTION

Ceci est le rapport de notre projet tuteuré. Il a été réalisé entre janvier 2015 et mars 2017 sous la direction d'Amélie Cordier.

Toute la conception et la rédaction s'est faite en Français et le développement en Anglais.

Un glossaire définissant les termes techniques se situe en annexe de ce rapport.

1.1 CONTEXTE & TUTEUR

Malgré une image parfois controversée, le jeu vidéo est un divertissement extrêmement populaire. Réunissant de très nombreux aspects techniques et artistiques, sa richesse lui a permis de s'approprier peu à peu une nouvelle place : celui de support d'apprentissage. Les jeux vidéo ayant cette vocation ont été popularisés sous l'appellation de Serious Game (ou « Jeu sérieux » en français).

Novatrice, la transmission de connaissances via un Serious Game permet d'allier l'amusement et le plaisir du jeu à l'acquisition de connaissances nouvelles. Malgré le fait que le support ait encore des choses à prouver (car il est transgressif), nous avons choisi de réaliser un tel projet afin de s'inscrire dans une démarche de démocratisation de ce type de jeu. Puisque nous sommes tous les quatre étudiants en Informatique ainsi qu'amateurs de jeux vidéo et de créations originales, il s'agissait pour nous d'un challenge à la convergence de nos intérêts personnels.

Nous souhaitions avant tout développer l'intérêt pour les nouvelles technologies de l'informatique et de l'électronique. Ayant déjà expérimenté la programmation Arduino au lycée et connaissant les possibilités offertes par le microcontrôleur, nous l'avons logiquement choisi.

L'idée était de ne pas nécessairement viser les amateurs de jeux vidéo mais également les néophytes curieux de programmation et de réussir à créer l'interaction. La prétention du projet n'est pas d'offrir une formation complète mais une initiation avancée d'une forme nouvelle.

Lorsque nous avons défini les grandes lignes du projet, nous nous sommes naturellement renseignés auprès de Mme. Cordier avant de lui soumettre le concept. C'est une professeure qui nous avait déjà témoigné un intérêt fort pour la robotique et Arduino et nous souhaitions fortement pouvoir travailler avec elle.

1.2 PRESENTATION DU PROJET

Le projet consiste à développer un jeu vidéo en 2D afin d'enseigner les bases de la programmation Arduino. Le jeu est découpé en différents niveaux qui confronteraient le joueur face à des défis de programmation qu'il faut obligatoirement résoudre pour débloquer de nouvelles fonctionnalités et compléter la partie.

Nous avons imaginé un concept de level design où seule l'interaction entre matériel et logiciel permettrait au joueur d'avancer, la carte Arduino étant en permanence connectée. Cela consiste à intégrer peu à peu de nouveaux composants (comme un potentiomètre, des LEDs…) et à transférer des données virtuelles dans le monde physique, par exemple en modélisant la barre de vie grâce à un bar LED. L'intégration de modules complémentaires passe par la lecture d'un petit cours, leurs déclarations, et l'écriture de quelques lignes afin de les rendre fonctionnel dans le jeu. L'idée est de laisser peu à peu la main à l'utilisateur dans l'écriture de programmes de plus en plus complexes.

1.3 ORGANISATION

1.3.1 GitHub

Figure 1-Logo Github

Dans le but d'optimiser nos phases de développement, nous avons mis en place un CVS[1]. Il nous a notamment permis de travailler chacun de notre côté et de pouvoir mettre en commun les parties de code au moment où celles-ci ont été nécessaires.

https://github.com/Jeremy-Duval/Genuini_s_Journey_Ruins_Of_Ivrea

1.3.2 Responsabilités des membres

- Adrien : Mécanique du jeu
- Jérémy : Cours et défis
- Fanny : Graphismes
- Valentin : Arduino

[1] CVS : Système de gestion de versions

1.4 DROITS

1.4.1 Contenus multimédias et code non créés

Initialement, nous considérions que la réalisation des sprites permettrait un design plus proche de l'idée que nous avions du jeu.

Après quelques recherches en ligne, nous avons découvert un pack contenant la totalité des éléments graphiques nécessaires à la création de l'univers Genuini. De plus, il s'agit de ressources libres de droit appartenant au domaine public.

Lien du pack graphique : http://opengameart.org/content/platformer-art-complete-pack-often-updated

1.4.2 Licence

Genuini's Journey : Ruins of Ivrea est un logiciel libre sous licence CeCILL-C.

Le dépôt contenant l'intégralité des codes sources, mais également les différents médias, est déclaré sous licence par l'intermédiaire d'un fichier se situant à sa racine.

Vous pouvez trouver la licence à cette adresse :

https://github.com/Jeremy-Duval/Genuini_s_Journey_Ruins_Of_Ivrea/blob/master/LICENSE.md

1.5 PRECONCEPTION

Au semestre 2, durant la phase de préconception, nous avons décidé unanimement de créer des diagrammes UML (voir annexe), une représentation de notre planification (GANTT) et également un cahier des charges et pour terminer un GDD (Game Design Document).

Parmi tous ces rendus, un seul nous a réellement servi : il s'agit du GDD. Nous nous sommes rendu compte qu'il serait très difficile de suivre la préconception que nous avions établie, pour développer un jeu vidéo.

Le GDD nous plongeait avant même le début de la programmation dans le monde de Genuini.

2 LIBRAIRIES

2.1 LA LIBRAIRIE LIBGDX

Figure 2 - Logo de LibGDX

LibGDX est une bibliothèque libre de droit et multiplateforme basée sur OpenGL et utilisable sous Java, Android et HTML5.

Nous l'avons choisie car elle est performante, en constante évolution et documentée. Effectivement, de nombreux témoignages d'utilisation de cette bibliothèque logicielle sur le Web nous ont encouragés à en apprendre un peu plus.

De par sa réputation et sa documentation, nous n'avons pas regretté ce choix qui s'est avéré être un allié de taille dans la création des graphismes du jeu.

2.2 LA LIBRAIRIE RXTX

Afin de permettre un interfaçage efficace entre notre carte Arduino et le programme JAVA, nous avions besoin d'une librairie capable d'interagir directement avec les composants. A ce moment précis, après de nombreuses recherches sur Internet et notamment sur le site web 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

Figure 3-Librairie

Elle nous permet d'abord d'interroger les ports séries (cf. 4.3) à travers des méthodes déjà écrites. Par la suite envoyer des informations ou en recevoir plus facilement.

L'utilisation de ces librairies s'est avérée essentielle 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 elle 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 efficaces dans notre phase de développement.

3 MECANIQUES DU JEU

3.1 ORGANISATION DU CODE ET LOGIQUE

Naïvement d'aucuns pourraient penser que le code d'un jeu vidéo s'articule de la même manière qu'un programme informatique classique. Or, la création d'un jeu vidéo dans son intégralité requiert l'intégration de composants multimédias, la génération d'un monde virtuel avec en son sein des créatures et décors interactifs, des menus, la gestion d'interactions utilisateurs, etc.

L'organisation du code joue donc un rôle primordial dans la capacité des membres de l'équipe à construire le jeu efficacement et sans être tourmenté par les éventuelles erreurs de régression (cf. **Erreur ! Source du renvoi introuvable.**).

Nous avions conscience de l'enjeu de la clarté de code, d'autant que nous allions être amenés à réutiliser le code des autres. Nonobstant ce fait, nous nous sommes vite aperçus que nous n'avions aucune idée de comment organiser le code pour notre jeu. Nous ne l'avons compris qu'après avoir recréé la base (classes essentielles qui permettent chacune des mécaniques du jeu) trois fois et après avoir factorisé, épuré et simplifié le code à chaque étape de création.

La description suivante fait état de notre compréhension sur cette articulation issue des spécificités de LibGDX, de Java et de nos connaissances.

3.1.1 Design Pattern

Un projet LibGDX avec Gradle sépare de manière inhérente le code lié au jeu, du code lié au lancement de celui-ci. Le lancement peut ainsi s'adapter aux mobiles et aux différents systèmes opérationnels sur lequel il est lancé.

Dans notre cas, le lancement se fait donc sous Windows à partir de la classe *DesktopLauncher*.

Cette dernière lance la classe principale du jeu qui initialise le contenu (cf. **Erreur ! Source du renvoi introuvable.**) et le premier écran (i.e. menu principal).

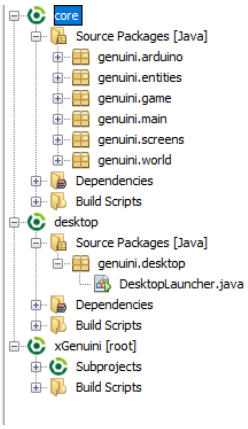


Figure 4 - Packages

Les noms de packages résument assez bien la séparation logique comme nous allons le voir ci-dessous avec la traduction de la JavaDoc correspondante.

Package Arduino

Ce package regroupe toutes les classes faisant le lien avec Arduino.

Package Entities

Ce package contient tout le code associé aux éléments du monde qui agissent sur celui-ci ou sur le joueur.

Package Game

Ce package contient tous les gestionnaires pour charger le contenu, modifier les écrans et créer les caméras, les skins (textures appliquées aux boutons), le texte, etc….

Package Main

Ce package contient uniquement la classe principale du jeu qui s'occupe de créer l'écran de démarrage, de charger le contenu en appelant le gestionnaire de contenu et de mettre à jour l'écran de jeu en cours de rendu.

Package Screens

Ce package contient tous les écrans utilisés lors du jeu comme le menu principal, l'écran de chargement, l'écran de jeu, etc.

3.1.2 Logique du flux de données

A chaque fois qu'un écran est chargé, les boutons, les écouteurs (détecteurs d'entrée utilisateurs), le texte, la musique sont créés.

Le jeu est rafraichi grâce à la méthode *render()* native de LibGDX. Cette méthode est appelée par la classe principale (*MainGame)* 60 fois par seconde. Elle dessine toutes les images/textures et appelle la méthode *update()* de chaque élément ayant besoin d'être mis à jour.

Cette dernière fait passer le monde virtuel (cf. 0) au stade suivant calculant ainsi toutes les valeurs des objets subissant la physique du jeu. C'est également en son sein que le traitement d'éventuelles actions venant de l'utilisateur sont traitées.

Sauvegarde des données

Les données relatives au jeu comme la position, la vie du joueur, son état d'avancement sont stockées dans un fichier XML. Le gestionnaire de préférences permet d'accéder et de modifier ces données.

3.2 MENUS ET CHANGEMENT D'ECRAN

Tous les écrans sont des extensions de notre classe abstraite d'écran qui elle-même est une implémentation de l'interface *Screen* de LibGDX. C'est aussi une extension de la classe *Stage* qui gère le placement des boutons et du texte fixe ainsi que leurs actions en fonction de l'entrée utilisateur. C'est donc au travers de la scène (*Stage)* que nous avons effectué la création et le placement des boutons des menus.

Le changement d'écran est natif de libGDX ; nous avons juste eu à implémenter la suppression de l'ancien écran. De plus, nous avons voulu créer un « Splash-screen » ou écran de chargement. Pour cela nous avons créé un processus en arrière-plan (thread) en chargeant l'écran de jeu alors que l'écran de chargement boucle sur une animation.

3.3 LE MONDE VIRTUEL

3.3.1 Physique

LibGDX intègre le moteur physique Box2D qui permet de simuler une physique réaliste au sein d'un monde 2D.

Le monde virtuel où évolue le joueur est composé de « corps » qui sont des objets géométriques localisés ayant de multiples propriétés. Leur forme et leur densité permettent au moteur de calculer leur masse et donc la manière dont ils se meuvent au sein du monde vis-à-vis de la valeur du champ gravitationnel choisi. Les déplacements des corps se font au moyen d'impulsion linéaire au niveau des centres de masses des objets.

Ils possèdent également un coefficient de friction et une résistance à l'air qui permet d'avoir un rendu plus réaliste et des attributs permettant de les identifier et de décrire leur solidité relative aux autres corps.

Ces propriétés sont fixées lors de la génération du monde.

3.3.2 Génération du monde

Le monde est généré à partir d'une carte de tuiles (« tiled map » en Anglais). Cette carte qui correspond à un « niveau » du jeu, a été dessinée en posant différentes tuiles (chaque tuile étant une image) et « objets de carte ». En fait, ces objets sont des indications pour le gestionnaire du monde comme nous le verrons par la suite.

La génération du monde est gérée exclusivement par la classe *WorldManager*. Cette séparation permet de limiter la taille de la classe *GameScreen* pour que, dans la mesure du possible, elle ne serve que de nœud central aux composants du monde.

Le gestionnaire commence son activité par charger la carte que lui fournit l'écran de jeu au travers du gestionnaire de préférences. Après l'avoir chargé, il récupère chaque couche (« layer » en Anglais) puis créé les corps correspondants. Cette étape est détaillée ci-dessous.

3.3.2.1 Terrain

Le terrain du monde est composé exclusivement de tuiles. Un corps est généré à partir de la position et de la forme de la tuile.

On a donc une génération de la forme : *Tuile → Corps* , le corps se superpose donc au dessin de la tuile.

3.3.2.2 Objets

Les couches objets permettent une génération de la forme *Objet de carte → Entité (Corps + Sprite).*

N.B : La méthode créant chacune des types d'objets parcourt les objets de la carte de la couche correspondante.

Décors interactifs

En l'état, nous avons implémenté trois types de décors interactifs :

- Tourelles pouvant envoyer des boules de feu à une vitesse et dans une direction variable.
- Ressort faisant rebondir le joueur
- Bouton lié à un objet et qui permet de l'activer/désactiver (ex : désactivation d'une tourelle)

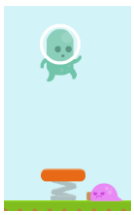


Figure 5 – Ressort

Figure 6 - Bouton

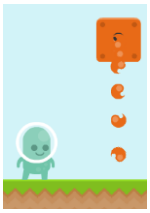


Figure 7 - Tourelle

Points de générations de créatures

Ce sont des lieux non-dessinés qui servent à placer une zone d'apparition de créatures.

Points d'accès

Ce sont des lieux non-dessinés qui servent à accéder à d'autres niveaux/cartes ou qui fixent le lieu d'apparition du joueur.

3.3.3 Gestion des contacts

Box 2D permet l'implémentation d'une interface gérant les contacts. Lorsque deux corps entrent en collision, le gestionnaire de contact effectue un traitement en deux temps : au début de la collision et lorsque les deux objets se séparent. En fonction des données dites « utilisateurs » des corps, une procédure est appliquée à celui-ci comme une impulsion linéaire ou la modification des dîtes données « utilisateurs ». Un exemple de cette modification est le marquage du corps afin de signifier au gestionnaire de monde de les détruire.

3.3.4 Les entités vivantes

Les entités vivantes sont animées.

En fait, les images associées aux différentes créatures et personnages sont indexées et compressées dans un fichier (atlas de textures).

La manipulation de ce dernier permet d'accéder aux images de manière chronologique. Ainsi, une procédure utilisant un compteur de temps permet de récupérer l'image correspondant à l'état dans lequel se trouve l'entité. Les états comme « marche » nécessitant une animation, renvoient l'image correspondant au temps écoulé.

3.3.4.1 Créatures mobiles

Les créatures sont générées par l'objet correspondant à leur zone d'apparition. Elles ont un comportement interactif avec le joueur en fonction de l'état dans lequel elles se trouvent.

Nous en avons pour le moment implémentées deux :

Les slimes (littéralement « substance visqueuse ») glissent sur le terrain et attaquent le joueur à partir d'une certaine distance.

Les escargots sont des créatures endormies s'ils sont trop loin du joueur et ils passent à l'attaque lorsque le joueur est proche.

3.3.4.2 Genuini

Le joueur possède des caractéristiques qui sont sauvegardées à savoir sa position et sa vie. Ses mouvements sont déterminés par l'entrée de l'utilisateur (touches ZQSD). Il a également des capacités spécifiques. Nous n'avons pour le moment qu'implémenté la capacité de lancer des boules de feu.

4 ARDUINO

4.1 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 101 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 tuteuré 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 et particulièrement la carte Genuino 101, nous semblait un choix judicieux. 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 microcontrôleur s'est imposé comme une évidence.

4.2 RELATION SOFTWARE-HARDWARE

Figure 5 - Genuino 101

Nous étions guidés par cet objectif de transmettre la connaissance nécessaire au joueur afin qu'il s'amuse comme nous l'avions 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 bagage suffisant pour commencer à tester les différents composants.

Pour parvenir à cet objectif, 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 gameplay

La finalité est 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 microcontrôleur 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[2]
- Afficher sur l'écran LED l'écran actif
- Permettre une interaction immédiate entre le monde physique et logiciel

Le jeu devant pouvoir être 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 6-Liaison Arduino-Jeu non détectée

4.3 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 matériel n'a donc pas dérogé à la règle et a intéressé 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.

[2] Genuini : Personnage principal du jeu

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 d'ailleurs pour avantage de mettre en avant une interface claire et épurée permettant une compréhension rapide des différentes fonctionnalités du logiciel.

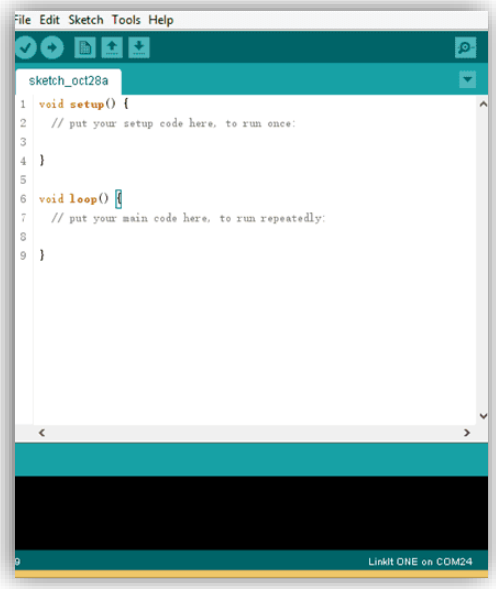


Figure 7-Interface Arduino IDE

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 bibliothèque de contenu additionnel permet à l'utilisateur de compiler et téléverser son programme par l'intermédiaire de ce logiciel à son microcontrôleur.

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

Le point essentiel du développement était la réussite de l'interfaçage 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 microcontrôleur 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 qui est utilisable via les connectiques USB. Une machine comporte plusieurs ports COM, mais elle 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 microcontrôleur y 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.

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

Du côté matériel, 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.

```
int commaIndex = rec.indexOf(';');
String firstValue = rec.substring(0, commaIndex);
String secondValue = rec.substring(commaIndex + 1);
```

Afin d'illustrer notre propos, nous 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 cet univers s'est avéré être une excellente idée et ce microcontrôleur a su convaincre tous les membres du groupe.

4.4 GROVE

Afin de rendre les branchements électroniques plus faciles pour l'utilisateur, nous cherchions des composants additionnels compatibles avec notre solution Arduino et qui obéissaient à ce principe que nous avions fixé : « plug-and-play ». Après de multiples recherches, nous avons découvert Grove. Il s'agit de modules offrant la possibilité d'accroitre les capacités de la carte en y ajoutant par 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 matériel, 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.

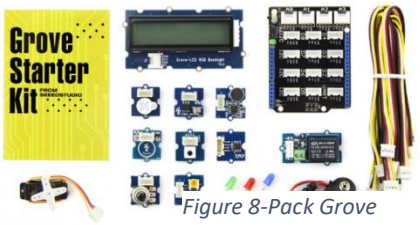


Figure 8-Pack Grove

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 Arduino et le gameplay.

4.5 AMELIORATION : LA CREATION D'UN PARSEUR

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

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 microcontrôleur. 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.

```
if (firstValue == "menu") {
  previousState = currentState;
  currentState = State::Menu;
} else if (firstValue == "game") {
  previousState = currentState;
  currentState = State::Game;
```

Figure 9-Conditions liées à la réception des données

Pour conclure, 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 matérielle. Dans un futur même proche, nous aimerions travailler dans un secteur cotôyant 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.

5 SYSTEME D'APPRENTISSAGE

5.1.1 Cours

Le joueur dispose d'un accès permanent aux cours qui sont consignés dans le grimoire. Ils portent sur des notions élémentaires d'électronique et d'informatique et permettent de comprendre le fonctionnement des différents composants ainsi que les fondamentaux de la programmation.

5.1.2 Défis

Les défis sont le point clef du jeu.

Ils permettent de tester les connaissances théoriques des cours (vus précédemment par le joueur) et de les mettre en pratique.

Le principe est simple : le joueur se retrouve confronté à un problème dans le jeu qui interrompt sa progression. Il est alors poussé à ouvrir le grimoire dans lequel l'attend un défi. Une fois ce dernier résolu, une nouvelle fonctionnalité sera débloquée, ce qui lui permettra de surmonter les difficultés.

On distingue deux types de défi. Premièrement, une question de cours avec plusieurs propositions de réponse (c'est le cas du premier niveau). C'est le plus simple pour le joueur, mais aussi pour la programmation. Il permet l'acquisition des connaissances basiques.

Pour le second type, le joueur doit saisir une ou plusieurs lignes de code dans un champ texte (c'est le cas du deuxième niveau).

Ces deux solutions ont été construites 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. Le second type de question est plus complexe du fait qu'il nécessite l'analyse du code saisi par le joueur.

6 CONTENU MULTIMEDIA

6.1 GESTION DANS LE JEU

La classe *MainGame* créée par la classe main (située dans *DesktopLauncher*) stocke toutes les musiques, les sons et les images en mémoire. Le programme instancie 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, les sons et les 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 qui correspond soit à une clef passée en paramètre soit au nom de cette donnée (par défaut).

Pour utiliser une musique, un son ou une texture, il n'y a plus qu'à appeler un getter avec en paramètre le nom de la clef au chargement de l'écran correspondant.

6.2 GRAPHISMES

6.2.1 Généralités

Le choix de réaliser un jeu vidéo en tant que projet tuteuré va bien au-delà de la programmation pure. Nous attachions une importance particulière à développer une identité visuelle propre. Néanmoins, l'ampleur de la tâche s'est révélée très importante puisque nous souhaitions réellement fournir des graphismes nets, presque professionnels et n'étions pas idéalement formés ou équipés pour. Nous avons donc utilisé des spritesheets (« feuilles de sprites », fichier contenant le nécessaire à la création d'une map ou partie de map) déjà existantes que ce soit pour le personnage principal ou les maps. Celles-ci proviennent du même auteur afin de garder une cohérence graphique : Kenney.

Par la suite nous avons essayé de faire le design d'un premier personnage grâce à Inkscape. Beaucoup d'idées laissées sans suite puisqu'il était alors nécessaire de produire des maps et plateformes cohérentes, dans un style de dessin similaire, alors que nous avions en parallèle beaucoup de défis de programmation que ce soit vis-à-vis du projet ou bien de la charge de travail personnelle à l'IUT. Nous avons donc donné priorité au fait de créer un contenu graphique que l'on pouvait exploiter directement avec le code sans perdre de temps.

En termes de création, nous n'avons également pas disposé des outils nécessaires jusqu'à la fin du projet (S4). Désormais : Krita et Tablette Huion. Nous avons réalisé avec ces outils un premier design d'écran de chargement, de victoire et de défaite et comptons étendre l'utilisation de design personnalisés. Encore une fois, l'importance d'une homogénéité de la partie graphique est primordiale et la réalisation de graphismes « sophistiqués » est une tâche qui, n'entrant pas directement dans le cadre de notre formation de développeur, a été quelque peu réservée en bonus.

Figure 13-Interieur du grimoire

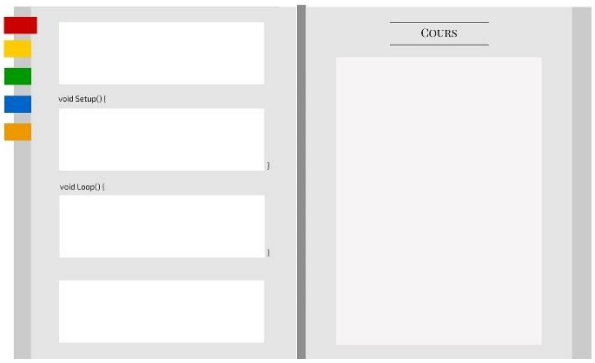


Figure 104-Zone de saisie des codes

***Exemple : l'apparence du grimoire**. Nous avons préféré reprendre la maquette présentée au S2 que d'utiliser un autre skin que nous avions trouvé suite à nos recherches. Moins flagrant sur la miniature, la figure à gauche a un style beaucoup plus réaliste et antique, ce qui cassait avec le reste du jeu. Au contraire, le second grimoire est simple, sans ombres.*

Néanmoins, les choix des sets ont été faits naturellement. S'agissant d'un Serious Game tout public, il nous fallait un design simple et efficace qui puisse plaire au plus grand nombre. De manière analogue pour les polices d'écriture, nous recherchions quelque chose de doux voir d'enfantin pour prendre le contrepied de l'image généralement véhiculée autour de l'électronique et de l'informatique, traditionnellement associée avec une certaine hostilité au grand public, très technique, très austère. Notre but étant d'ouvrir les gens à la programmation Arduino, nous voulions une image épurée et accueillante pour que tout le monde puisse s'y retrouver sans préjugés particuliers.

6.2.2 Choix des polices

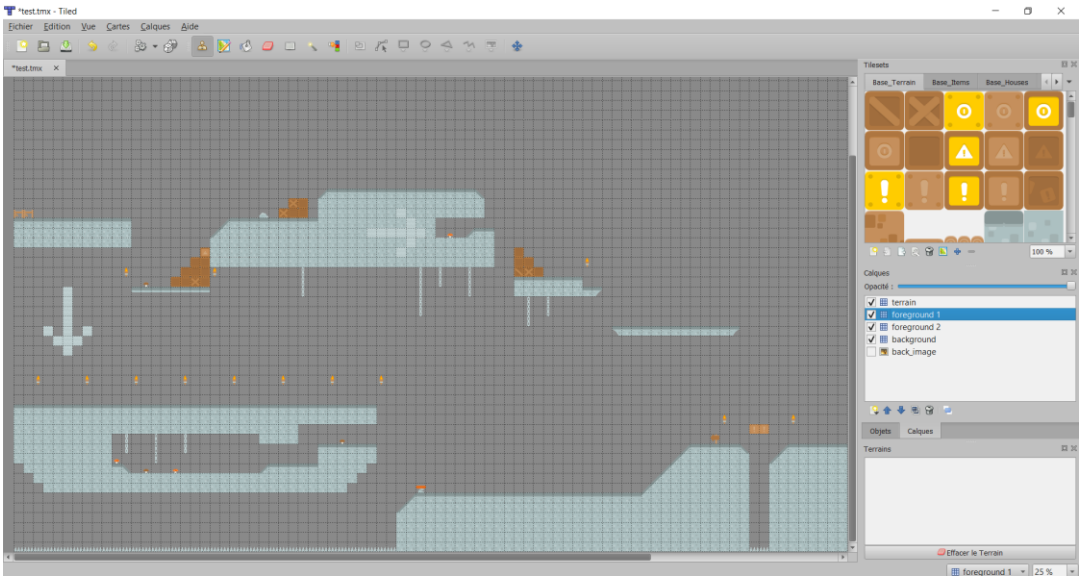
Grundshrift : police utilisée dans le milieu scolaire en Allemagne, semblable à l'écriture manuscrite. Elle nous a plu de par sa simplicité et sa lisibilité et correspond à notre volonté didactique.

LobsterTwo : police sous licence SIL Open Font License.

6.2.3 Outils

Figure 11-Logo du logiciel Tiled

Il s'agit d'un éditeur de cartes (ou maps) libre de droit qui permet de réaliser les différents niveaux en y positionnant les différents éléments (plateformes, décors, etc. …). Il repose sur un système de couches, ou « layers ». C'est un principe simple qui s'explique naturellement : si le level design était abordé comme une illustration, il serait alors impossible de distinguer les différents éléments : comment reconnaître et appliquer des comportements particuliers aux divers éléments ? Reconnaître ce qui se passe devant, derrière, au-travers de notre personnage ?



***Interface de Tiled**. On remarque : le quadrillage de la map, les tilesets sur la droite ainsi que les layouts (calques)*

La carte est découpée en carrés de 70px par 70px, appelés Tiles. On peut définir la taille de la carte en précisant le nombre de Tiles qu'elle comprend, en longueur et en largeur. Puis, on travaille avec les différentes couches afin de créer différents plans et différentes profondeurs. En général, on créer une base composée de trois couches : le background, le foreground et le terrain (où le personnage se déplace) auxquelles on peut ajouter des variations.

Il suffit ensuite de remplir la carte à l'aide du quadrillage et de Tileset (sets d'éléments graphiques). La prise en main du logiciel est relativement simple, la difficulté réside dans la conception du niveau lui-même afin de lui donner un vrai sens sans se contenter d'aligner les blocs.

Figure 12-Logo du logiciel Krita

Un logiciel de dessin libre (GNU General Public License version 2 et plus).

Plus intuitif que The Gimp et infiniment moins complexe que Photoshop, Krita dispose de tous les outils idéaux à la réalisation de dessins à niveau amateur ou pro. Nous l'avons peu utilisé pour le moment.

6.3 Musiques & Sons

6.3.1 Musiques

Les musiques sont –à l'instar des graphismes- un point clef d'un jeu vidéo, celles-ci vont influencer directement sur l'ambiance du jeu et l'état d'esprit du joueur. Ce n'est donc pas une partie négligeable. La musique peut nous permettre de plonger le joueur dans le jeu et donc de le rendre plus réceptif et ouvert à l'apprentissage.

Nous avons dans un premier temps pensé à enregistrer nous-mêmes les musiques ayant de nombreux instruments au sein de notre groupe. Faute de temps et ayant considéré le code du jeu prioritaire, nous avons donc décidé d'abandonner en partie l'idée.

En effet, chaque musique du jeu a été créée manuellement, à l'aide d'un logiciel 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.

Figure 13-interface du logiciel LMMS

Ces musiques sont gérées de manière simple dans le code, à l'aide de LibGDX (cf. **Erreur ! Source du renvoi introuvable.**).

6.3.2 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 connaissances préalables de certains de nos membres.

La gestion des sons dans le programme s'effectue à l'aide des classes *MainGame* et *Content* (cf. 4.1).

7 LES OBSTACLES RENCONTRES

7.1 ORGANISATION

Durant la phase d'attribution des tâches, nous nous étions fixés des objectifs par personne avec comme ambition que chacun participe tout de même à toutes les différentes parties du projet. Nous avions raisonné en termes de travail à accomplir et non en créneaux propices à la mise en commun des réalisations. L'absence de temps réservé dans notre planning hebdomadaire nous a handicapé pour l'organisation de réunion informative ou bien récapitulative.

Nous avons donc souffert d'un manque de relation directe entre les participants. La distance nous séparant les uns des autres n'a également pas aidé à coder simultanément durant des périodes propices comme les vacances.

7.2 INTEGRATION CONTINUE

Figure 14-Logo de Travis CI

Sur les conseils de Mme. Cordier, nous voulions développer notre jeu dans une démarche d'intégration continue. Notre objectif était simple, créer des parties de programme testant une unité de code et par la suite implémenter les différentes fonctions. Le logiciel devait pouvoir être essayé sur le système développant l'application mais également les autres systèmes d'exploitation.

Pour parvenir à avoir un jeu universel, nous avons utilisé le site internet Travis-CI qui permet de créer de nombreuses machines virtuelles représentant les différents SE puis de lancer des tests unitaires et vérifier s'ils réussissent ou échouent.

https://travis-ci.org/

Nous nous attendions à ce que tout se passe convenablement lorsque Travis nous informa de nombreuses erreurs. Au premier abord, le problème nous semblait non compréhensible puis dans un second temps, nous avons compris que la problématique majeure résidait dans le fait que les machines virtuelles émulées par Travis ne parvenaient pas à installer la librairie « RXTX » (cf. 2.2).

De fait, toutes les lignes de code contenant un appel à une fonction de cette bibliothèque logicielle occasionnaient des erreurs.

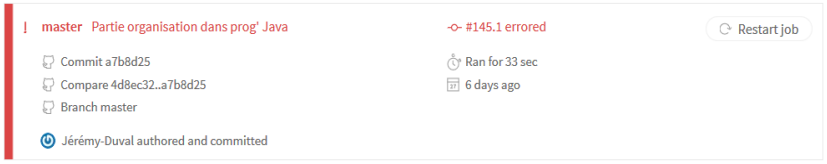


Figure 15 - Message d'erreur de Travis

Pour palier à ce problème, nous avons tenté de scripter l'installation de RTXT sur les machines virtuelles mais cela échoua. Une solution alternative aurait pu être d'inclure « RXTX » au projet via un gestionnaire de dépôt distant Maven.

https://mvnrepository.com/artifact/org.rxtx/rxtx/2.2pre2

De cette manière, les machines de Travis n'auraient plus à essayer d'installer la librairie qui se trouvait en local sur l'ordinateur. Malheureusement, du fait de la découverte de cette solution tardivement, nous n'avons pas exploité l'intégration continue à son maximum.

7.3 DEBOGAGE

Une fois la programmation avancée à un stade un peu plus conséquent, il était désormais temps de passer d'un écran à un autre comme par exemple jouer sur la plateforme principale puis par la suite afficher l'écran de mort. Durant un long moment, aucun problème n'avait surgi puis lors d'une victoire notre jeu s'est arrêté brutalement. Dans les informations de l'erreur était inscrit ALC_CLEANUP.

Après de nombreuses recherches auprès de personnes ayant été confrontées au même souci et ayant tentées de déboguer, nous en avons conclu qu'il s'agissait d'un problème de rendu graphique lié à la synchronisation.

A la suite d'une factorisation intense du code, nous avons pu bien comprendre les multiples origines de ce message d'erreur.

En effet, le monde virtuel lors de sa mise à jour bloque ses composants. Or de nombreuses tentatives d'accès à des objets bloqués étaient effectuées par des composants du jeu. Il fallut donc bien séparer la mise à jour du monde et celle des autres composants du jeu. De plus, pour régler ces problèmes, une gestion directe de la destruction des corps devait être implémentée.

Bien que ce bug soit à présent très rare, une des sources de celui-ci est inhérent à l'intégration de Box 2D à LibGDX et nous pensons donc ne pas pouvoir le résoudre.

8 FUTUR DEVELOPPEMENT

8.1 INTERACTIONS AVEC ARDUINO & DEFIS

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, il 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.

- Grâce à un potentiomètre, nous pourrions régler la densité de certains objets et 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 donnerait la possibilité de faire la même chose suivant la luminosité qu'il détecte.

- Grâce à un capteur d'humidité, nous pourrions faire pleuvoir sur une zone et 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) nous permettrait d'actionner certains mécanismes bloqués dans le jeu comme une manivelle.

- Un gyroscope nous donnerait la possibilité de faire déplacer certains contenus dans le jeu 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 ajouterait la possibilité de faire jouer une deuxième personne qui pourrait aider la première dans le jeu en déclenchant des mini défis sur ces écrans.

- Un module wifi ou XBee entraineraient la communication entre deux cartes et ouvriraient ainsi des multitudes de possibilités. Le personnage aurait par exemple la possibilité de passer entre deux écrans.

Pour chaque type d'interaction, il nous faudra des défis. Comme expliqué dans la partie correspondante (cf. **Erreur ! Source du renvoi introuvable.**), il y a deux familles de défis.

Les défis avec réponses proposées seraient plus présents au début de chaque cours afin de tester des connaissances théoriques. Ils présentent de nombreuses possibilités que nous pourrons mettre à profit pour développer l'apprentissage du joueur.

Les défis dans lesquels il faut programmer offrent plus de résultats. Nous avions prévu que ce type de défi devienne de plus en plus complexe et élaboré.

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

8.2 UNIVERS

Nous comptons ajouter au monde plus de créatures qui auraient des comportements amicaux ou hostiles.

Nous souhaitons également peupler l'univers du jeu avec des personnages non-joueurs qui dialogueraient avec le joueur pour lui fournir des indications sur l'univers ou les objectifs et lui proposer des quêtes. Des formes d'échanges d'objets ou de services pourraient avoir lieux.

Pour les quêtes, il sera nécessaire d'implémenter des objets pouvant être collectés comme des pièces ou des champignons.

Elles constitueront la toile de fond pour le scénario qui n'a pas été implémenté jusqu'alors pour les besoins de la démonstration.

Concernant le terrain, nous pensons rendre plus vivant le décor grâce à des couches d'images se déplaçant relativement aux autres et à la caméra donnant ainsi un effet de profondeur.

De plus, il faudrait que la physique du jeu ait des comportements localisés en créant par exemple des niveaux avec des objets subissant le champ gravitationnel différemment. Nous souhaitons également ajouter des éléments de terrain mobiles tel que des murs ou des plateformes.

Le joueur, quant à lui, aurait la possibilité de lancer de nombreux sorts.

9 BILANS PERSONNELS

9.1 JEREMY

Je pense que ce projet m'a beaucoup apporté et j'ai particulièrement aimé l'idée de pouvoir aider des personnes pour l'apprentissage d'Arduino.

Sur le plan technique, cela m'a permis de continuer à côtoyer l'univers d'Arduino que je connaissais depuis le lycée et d'être dans la continuité de mon projet du premier semestre (un poster sur Arduino).

Ensuite, j'ai vraiment pu me familiariser avec Java et apprendre à construire un projet dans ce langage : organiser les packages et les classes, constater l'importance des commentaires et de la javadoc dans un travail de groupe… Nous avons aussi dû apprendre à maitriser un nouvelle bibliothèque (libGDX) afin de mener à bien ce projet. Cela m'a permis de constater à quel point la recherche et l'apprentissage étaient des points important dans un projet.

Enfin, j'ai apprécié de pouvoir me renseigner sur les licences libres, d'en appliquer une à notre projet et de travailler sous forme d'open-source. Cela permet de contribuer aux valeurs de partage, de liberté et de donner une dimension plus humaine à notre projet.

Sur le plan humain, cela m'a permis de constater que tout n'était pas toujours facile. Il faut apprendre à persévérer avec la fatigue et les difficultés.

J'ai tout de même passé de bons moments. Construire un projet pouvant perdurer, et ce, dans la bonne entente est vraiment agréable. Avec une tutrice à l'écoute et des amis en coéquipiers, cela ne pouvait que fonctionner.

9.2 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. Notre but était de faire ressentir aux utilisateurs ces sensations que nous-mêmes avons éprouvées et éprouvons encore concernant le monde d'Arduino.

Je retire de cette réalisation de projet un sentiment positif. Dès la phase de conception, nous étions tous les quatre motivés pour 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 pu 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 réalisation. J'avais hâte de programmer dans le langage JAVA. J'ai pour objectif dans un futur proche d'en faire mon langage de développement quotidien, j'apprécie son côté haut niveau et qu'il soit orienté objet. Son utilisation dans ce projet a donc été une excellente nouvelle d'un point de vue personnel.

En ce qui me concerne, j'éprouve certains regrets car j'ai plutôt codé dans la précipitation. Si je devais revenir en arrière, je programmerais dans la durée en étalant le travail. Mais, nous ne pouvons pas être persuadé que cette méthodologie aurait porté ses fruits. Toutefois, je suis satisfait 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 et 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.

9.3 FANNY

Technique

Personnellement, je ne suis pas une grande adepte de la programmation en Java et je ne peux pas dire que cette situation ait beaucoup évolué. J'ai eu quelques soucis avec Git (conflits avec le dépôt, problèmes de merge) que j'ai parfois pu gérer, mais que j'ai aussi souvent négligé par nécessité en lui préférant d'autres outils. Néanmoins, j'ai souvent subi l'absence de CMS et j'ai commencé à l'utiliser en parallèle personnellement afin de mieux comprendre sans mettre en péril notre dépôt et toutes ses branches…

Nous avons mis en place différents outils de gestion de projets intéressants dans un premier temps (au S2, début de S3) que nous avons délaissés : la façon de programmer un jeu était très différente de tout ce qu'on a pu voir et expérimenter et l'adaptation a été corsée. Malgré tout, je pense pouvoir affirmer que nous avons su surmonter les difficultés que nous avons rencontrées.

Humain :

Les échanges autour du projet ont toujours été très constructifs surtout durant la phase de conception. En pratique, j'ai subi une importante baisse de motivation durant certaines étapes du projet. J'ai eu beaucoup de hauts et de bas avec les différents outils (sans aucun doute liés à une mauvaise utilisation) ce qui m'a rendu la tâche fastidieuse. Les projets tuteurés sont vraiment un élément important du DUT et je trouve cela dommage qu'on ne nous réserve pas de créneaux sur l'emploi du temps pour le travailler en autonomie (excepté le jeudi après-midi, plutôt réservé aux activités sportives…).

De fait, j'ai constaté une certaine difficulté pour se coordonner, d'arriver à se voir en dehors des cours afin de travailler en équipe. C'est une problématique récurrente des travaux de groupe, et bien que je n'y ai pas été étrangère, cela m'a permis de constater son importance.

En dépit de ces difficultés, j'ai adoré présenter le projet aux visiteurs durant les Journées de l'Enseignement Supérieur. Malheureusement, je n'ai pu y prendre part qu'une journée et j'aurais aimé étendre l'expérience. Il était intéressant de voir la réaction et le retour des gens sur le jeu et de s'essayer à expliquer le plus simplement possible le concept qui n'est pas toujours évident à appréhender.

Pour finir, je dirais que le point le plus enrichissant du projet sur le plan humain est le fait que l'on se retrouve à travailler avec des gens ayant des profils très différents en termes de compétences et d'attitude, d'implication, de gestion du stress et du travail.

9.4 ADRIEN

Ce projet tuteuré est le projet le plus conséquent que j'ai eu à réaliser. De fait j'ai pu observer en moi et sur mes équipiers de nombreux processus psychologiques et d'épreuves intellectuelles lié au travail sur le moyen/long terme.

Outre l'intérêt très fort pour la réalisation du projet cela rendit cette expérience d'autant plus passionnante.

Construire une vision

Rassemblés autour de valeurs que nous voulions transmettre à travers la réalisation du projet, nous avons beaucoup partagé afin de cadrer nos idées. Même avec cadre commun, nous pouvions voir que la vision du projet divergeait suivant les personnes et qu'il était en conséquence, compliqué pour nous, de prendre des initiatives personnelles sans qu'il y ait une personne dirigeant le projet. En fait, nous avons observé que les différentes visions du projet n'ont convergé qu'à partir du moment où le jeu prit forme. Nous pouvons en conclure que ni la préconception ni le brainstorming, n'ont permis aux membres de s'aligner. Cela bouscule à juste égard, notre conception idéaliste du déroulement d'un projet et d'une organisation totalement horizontale.

Idées et adaptabilité

Néanmoins, la phase de conception a largement contribué à l'apparition d'idées, nous n'étions ainsi jamais à cours de choses à réaliser (nous en avons d'ailleurs encore pour deux ans de développement). Au long du projet, nous avons pu remarquer lesquelles nous abandonnions à cause de difficultés techniques non-prévues, du temps nécessaire ou de l'incohérence avec la vision. A donc été éprouvé notre capacité à jongler entre nos envies, nos attentes et nos capacités. Cela m'a permis de comprendre qu'il fallait bien souvent, lorsqu'on ne maîtrise pas un processus de création, essayer puis réfléchir ensuite.

Motivation

Cette façon de faire permet de concrétiser rapidement les idées et ainsi de raviver l'implication dans le projet. L'ampleur de la tâche au vu de nos compétence étant grande, il est frustrant de ne pas avoir pu y travailler plus. Je souhaite donc vivement continuer le développement pour qu'il atteigne un stade intéressant pour une audience plus large que celle de l'IUT. Bien que l'organisation et la répartition dans le temps du travail était sous-optimales, j'ai quand même pu trouver la motivation nécessaire grâce à mes co-équipiers et aux contraintes de rendu, afin de fournir le travail nécessaire à une réalisation intéressante.

De plus, c'est une source de confiance en soi de savoir que je suis capable de travailler des centaines d'heures sur un projet et de rester patient et confiant face à une difficulté qui au premier abord parait irrésoluble.

Je suis donc très content de me conforter dans l'idée que le travail lorsqu'il trouve un sens à mes yeux, peut-être pour moi, une source de satisfaction qui se transforme en plaisir lorsqu'il est collaboratif et partagé.

10 CONCLUSION

10.1 BILAN GENERAL

Le projet tuteuré a été pour nous une aventure très enrichissante et surprenante. Elle a concerné des technologies et milieux extérieurs à nos enseignements conventionnels à l'IUT : ceux du jeu vidéo. Néanmoins, nous avons utilisé beaucoup d'outils et de méthodes que nous avions acquis et maîtrisé grâce aux cours. Cela nous a permis de sortir des sentiers battus tout en disposant d'une bonne base.

Il s'agissait également de la première fois où nous explorions les travaux de groupe à cette échelle. Nous avons pu réellement constater les écarts entre les estimations et la réalité. En général, les travaux pratiques et différents projets à l'IUT ne nous offraient que peu de résistance et nous parvenions toujours à nos objectifs à moindre effort. Le projet tuteuré nous a sorti de notre zone de confort et nous a beaucoup stimulé.

Nous entretenons désormais le désir de poursuivre le projet dans les années à venir avec beaucoup d'optimisme. Il nous accompagnerait alors dans notre progression en informatique et nous pourrions intégrer de plus en plus d'outils et ainsi parvenir à une bonne maîtrise de tout ce dont nous espérions. Pour certains d'entre nous, le projet a même permis de créer ou de confirmer des vocations professionnelles dans des secteurs tels que la programmation Java ou la robotique.

10.2 REMERCIEMENTS

Nous tenons à remercier chaleureusement notre tutrice Amélie Cordier pour nous avoir accompagné dans ce projet. Son implication dans ses divers enseignements a été une grande source de motivation pour nous.

Nous remercions également le département Informatique pour son soutien au travers de l'achat de matériel, ce qui a rendu ce projet possible. Enfin, nous pensons à tous ceux qui ont témoigné de l'intérêt envers le projet et qui ont su nous conseiller et nous soutenir.

11 ANNEXE

11.1 GLOSSAIRE D'INFORMATIQUE

Bibliothèque (logicielle) : ensemble de fonctions permettant un développement simplifié

Classe abstraite (interface) : permet un formalisme des classes l'implémentant

Coloration syntaxique : morceaux de code colorés permettant une identification des méthodes ou variables par le développeur.

CVS (Système de gestion de version) : procédé offrant une hiérarchisation du développement et une collaboration entre tous les membres.

Débogage : fait de trouver puis de réparer une erreur de programmation.

Design Pattern : organisation du code pour le rendre lisible et réutilisable.

Factorisation (fait de factoriser du code) : principe consistant à créer de nombreuses classes afin d'obtenir des fonctions pour chaque action nécessaire.

Gradle : outil de compilation JAVA.

IHM (Interface Homme-Machine) : interface permettant une interaction entre le monde réel et virtuel.

Intégration continue : vérification régulière de compatibilité d'une partie ou de l'entièreté d'un code.

JAVA : langage de programmation orienté objet.

JavaDoc : documentation permettant une compréhension facilitée pour des personnes reprenant le programme.

Librairie : synonyme de *Bibliothèque (logicielle)*.

Libre de droit : réutilisation de la ressource non soumise à une règlementation spécifique

Microcontrôleur : périphérique matériel offrant de nombreuses possibilités d'interconnexion.

Package : équivalent d'un dossier contenant des classes JAVA.

Parseur : analyseur syntaxique permettant une compréhension et un découpage par le programme d'une partie de texte.

Système d'exploitation : ensemble de logiciels chargés d'assurer la liaison entre les ressources matérielles, l'utilisateur et les applications.

UML : standard de représentation des diagrammes de conception.

11.2 GLOSSAIRE DE JEU-VIDEO

Caméra : symboliquement l'œil du joueur. C'est en déplaçant cette caméra que l'utilisateur pourra voir les différentes zones de la carte.

Gameplay : scénario d'un jeu.

GDD (Game Design Document) : document présentant l'univers d'un jeu ainsi que son scénario.

Level/Game Design : création de l'univers du jeu, de ses éléments et mécanique de manière théorique.

Mobs : créatures virtuelles interagissant avec le joueur.

Moteur physique : module simulant des forces physiques dans le monde virtuel.

Plug-and-Play : fait de brancher et jouer immédiatement.

PNJ (Personnage Non Joueur) : personnage fictif pouvant interagir ou non avec le joueur. Il peut être ami, neutre ou ennemi.

Serious Game : apprendre une ou plusieurs notions en conservant un aspect ludique.

Splash Screen : écran invitant l'utilisateur à patienter durant un chargement.

Sprites : image en deux dimensions qui peut être déplacée par rapport au fond de l'écran.

Texture : surface d'une ressource graphique ou composante d'un rendu au sens de programmation.

Tile : littéralement tuile. Elément d'une tiled map.

Tiled Map : Carte sous forme de grille, composée de tiles/tuiles symbolisant un carreau. Peut comporter plusieurs couches de tiles afin de définir les décors et les obstacles.

11.3 DIAGRAMMES UML

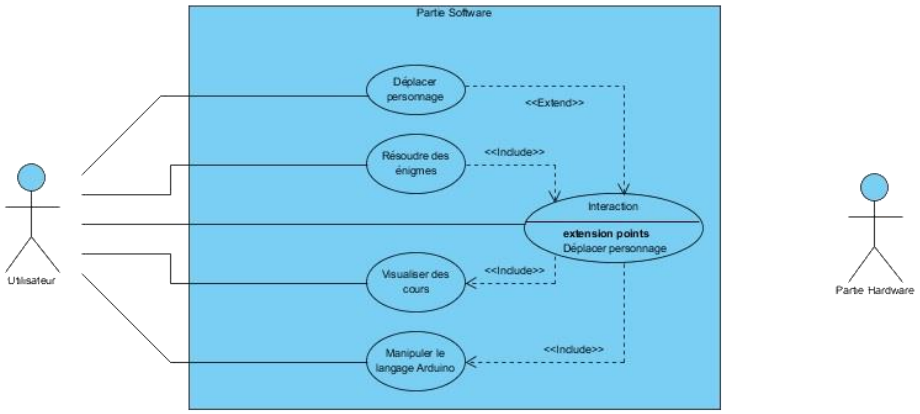


Figure 16 - Diagramme de cas d'utilisation logiciel

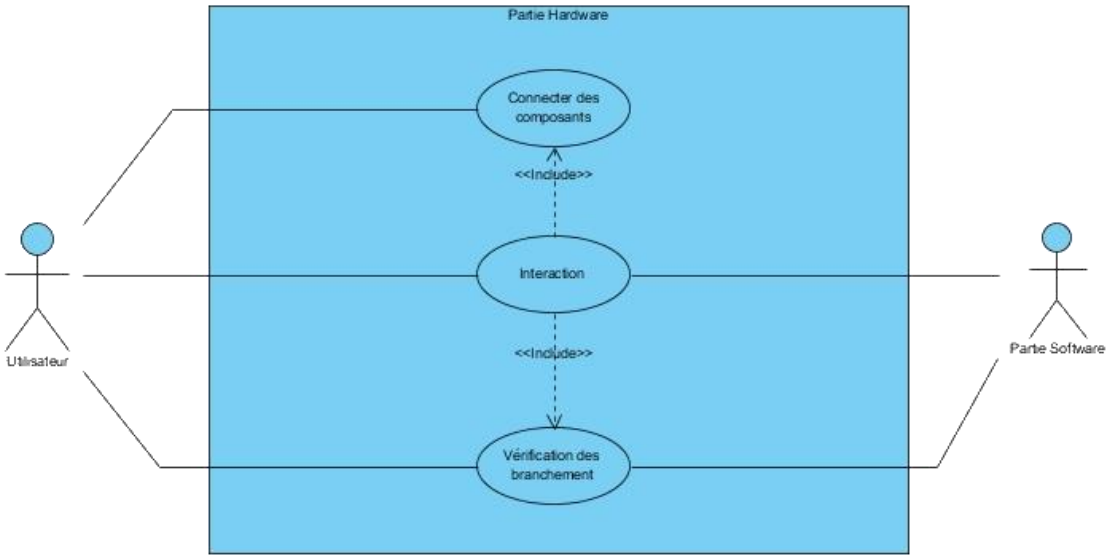


Figure 17 - Diagramme de cas d'utilisation matériel

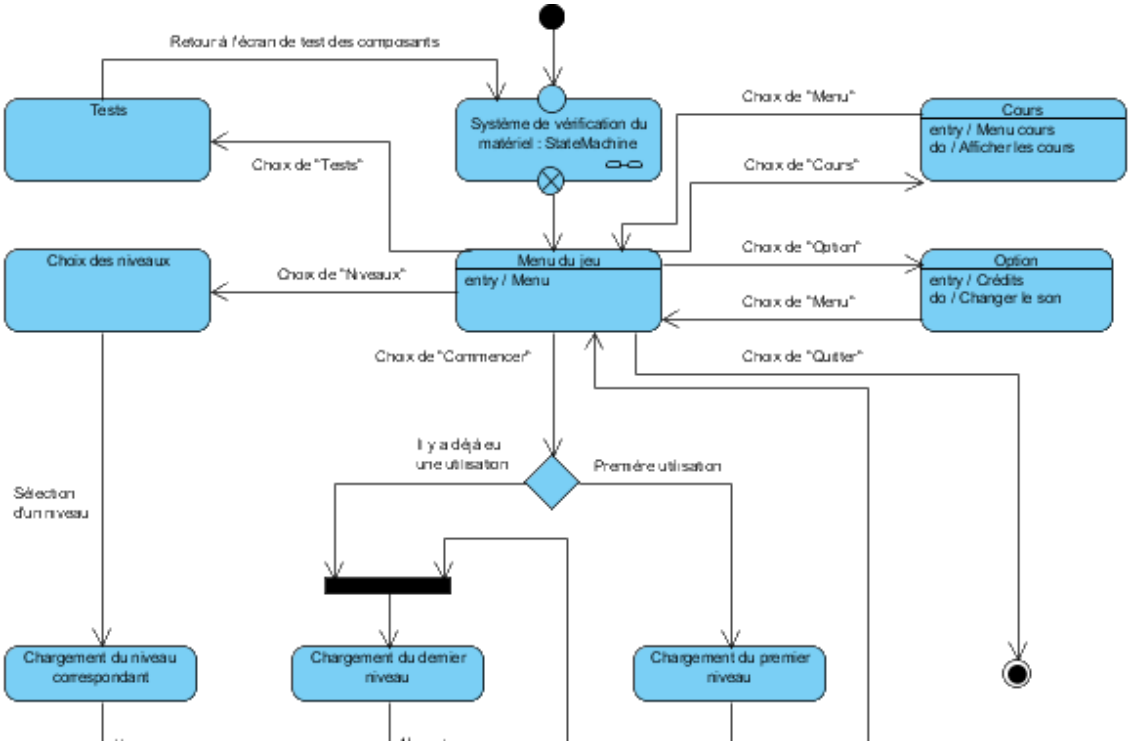


Figure 18- Diagramme d'états transition 1

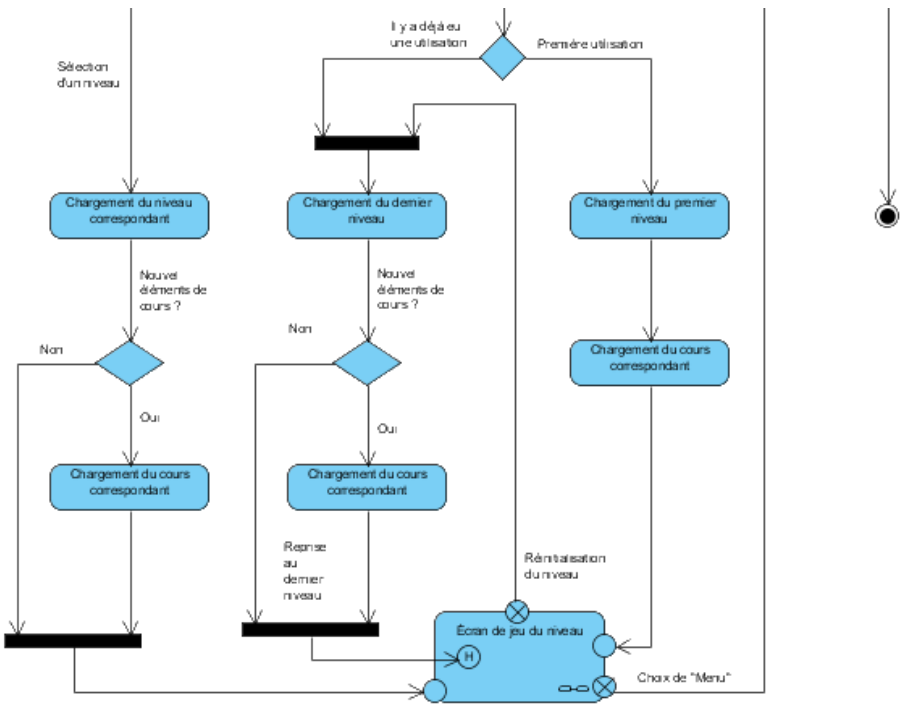


Figure 19 - Diagramme d'états transition 2