Travail de maturité

Création d'un jeu vidéo avec le logiciel GDevelop 4

Antoine LACOUR 402
David DANIEL 401

Collège Claparède

Novembre 2018

Résumé

L’objectif de ce travail de maturité a été la réalisation d’un jeu vidéo à l’aide d’un logiciel de développement de jeux. Le logiciel que nous avons utilisé s’appelle « GDevelop 4 » et propose une interface graphique qui simplifie le codage du jeu. Cette interface est intuitive mais demande tout de même une logique de programmation et l'appropriation des fonctionnalités spécifiques du logiciel. L’apprentissage peut donc prendre du temps mais une fois les connaissances assimilées, la partie créative dans l'élaboration d'un jeu vidéo est très riche.

Dans ce travail, nous avons tenté d'expliquer les bases et la logique de GDevelop 4 en illustrant les explications par des exemples concrets, afin que la prise en main du logiciel soit simplifiée pour toute personne souhaitant l’utiliser. Nous proposons une explication progressive et logique des différents concepts nécessaires à l’utilisation du logiciel. La maîtrise du logiciel est ainsi grandement facilitée par l’acquisition de ces concepts de base.

Table des matières :

Introduction et motivations

L’objectif de ce travail de maturité était la création d’un jeu vidéo à l’aide du logiciel GDevelop 4. Le document qui suit décrit l’utilisation de ce logiciel et s’adresse à toute personne voulant débuter la création d’un jeu vidéo avec GDevelop 4.

L’idée de la création d’un jeu vidéo est venue de notre intérêt et plaisir à jouer aux jeux vidéo. Nous apprécions en effet tous les deux les jeux vidéo et nous avons beaucoup joué à des *Rogue Like*, le même type de jeu que nous avons développé. Bien que nous appréciions ces jeux et que nous étions tous deux intéressés par l’informatique, nous n’avions jamais créé notre propre jeu, par manque de compétences. En fin de deuxième année, nous avions choisi l'OC informatique et avons eu l'idée de développer un projet dans cette branche.

La création d’un jeu vidéo n’a pas été notre première idée. Effectivement, nous voulions initialement construire une borne d’arcade, qui aurait fonctionné grâce à un Raspberry Pi. Nous nous sommes par la suite rendu compte que le projet, bien que complexe, n’était pas suffisamment conséquent pour un travail de maturité à deux. L’idée de créer parallèlement un jeu pour cette borne d’arcade nous est ensuite venue, mais cette idée fut rapidement écartée, car il était trop complexe de développer deux projets à la fois. Nous avons donc finalement décidé de créer un jeu vidéo. Nous avions également tous deux un intérêt pour l'informatique. Le TM nous permettait donc d'allier deux intérêts communs, tout en développant un projet relativement libre.

Nous n’avions à ce moment aucune connaissance en codage, ce qui nous a poussé à utiliser un logiciel simplifiant le développement. Notre premier choix fut Game Maker, mais ce logiciel étant coûteux, nous avons cherché des logiciels équivalents, puis avons choisi GDevelop, car il est gratuit et semblait complet.

Nous avons choisi de développer un jeu de type *Rogue Like* (un joueur qui se déplace dans un donjon et qui affronte des ennemis) car, comme indiqué précédemment, nous apprécions ce type de jeu et y avons beaucoup joué. Nous avions donc des idées de mécaniques et plusieurs inspirations. Ce type de jeu présentait un autre avantage : sa simplicité. Ces jeux sont en effet en deux dimensions, ce qui est grandement avantageux et nous semblait relativement aisé à développer. Nous avons ensuite réalisé la véritable complexité de ces jeux et avons revu nos ambitions initiales à la baisse.

Présentation de GDevelop 4

GDevelop (aussi appelé Game Develop) est un logiciel de création de jeux vidéo gratuit, disponible sur internet. Il peut être téléchargé sur Windows, Linux et Mac. Ce logiciel a été créé en 2008 et a continué à évoluer jusqu'en 2016 par mises à jour régulières. En 2016, les mises à jour s'arrêtent et le développement d'une nouvelle version du logiciel est lancée : GDevelop 5. Depuis peu, GDevelop 5 a été officialisée. GDevelop 4, la version que nous avons utilisée est cependant toujours disponible.

Le logiciel a été programmé en C++. Il permet de développer des jeux en 2 dimensions ainsi que des jeux en 3 dimensions. Les jeux 3D sont rudimentaires, ceux en 2D, en revanche, peuvent être assez élaborés. Le logiciel a été pensé pour apprendre le développement d'un jeu, la programmation se fait donc par le biais d'une interface graphique simple mais complexe à maîtriser dans les détails.

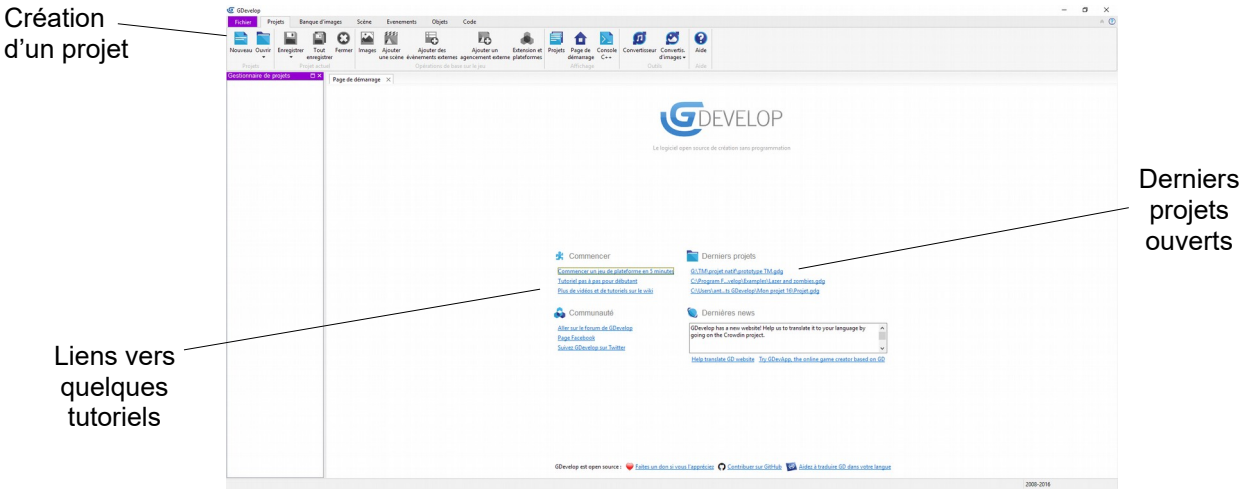
Deux plateformes de création de jeu sont disponibles. La plateforme "HTML5" est dédiée aux jeux sur page Web, sur Android et iOS. La plateforme "Natif" est utilisées pour les jeux sur Windows et Linux. La plateforme "HTML5" est plus limitée en termes d'extensions que la plateforme "Natif". Par exemple la plateforme "HTML5" ne permet pas de gérer les lumières alors que la plateforme "Natif" le permet.

Téléchargement de GDevelop 4

Le logiciel est disponible sur le site gdevelop-app.com sous la section download. Il peut être téléchargé sur Windows, Linux et Mac. Le téléchargement est gratuit et rapide.

Création d’un nouveau jeu

Pour créer un jeu avec GDevelop, il faut tout d’abord ouvrir le logiciel. Une fois le logiciel lancé, la page de démarrage de GDevelop s’ouvre. Cette page comprend beaucoup d’éléments. Au centre de la page se trouvent différentes sections de liens dont ceux permettant d’ouvrir des pages menant à des tutoriels pour débutants ou menant au site internet dédié à la communauté de GDevelop. L’historique des derniers projets ouverts est affiché à gauche de ces quelques liens.



Page d'accueil du logiciel GDevelop 4

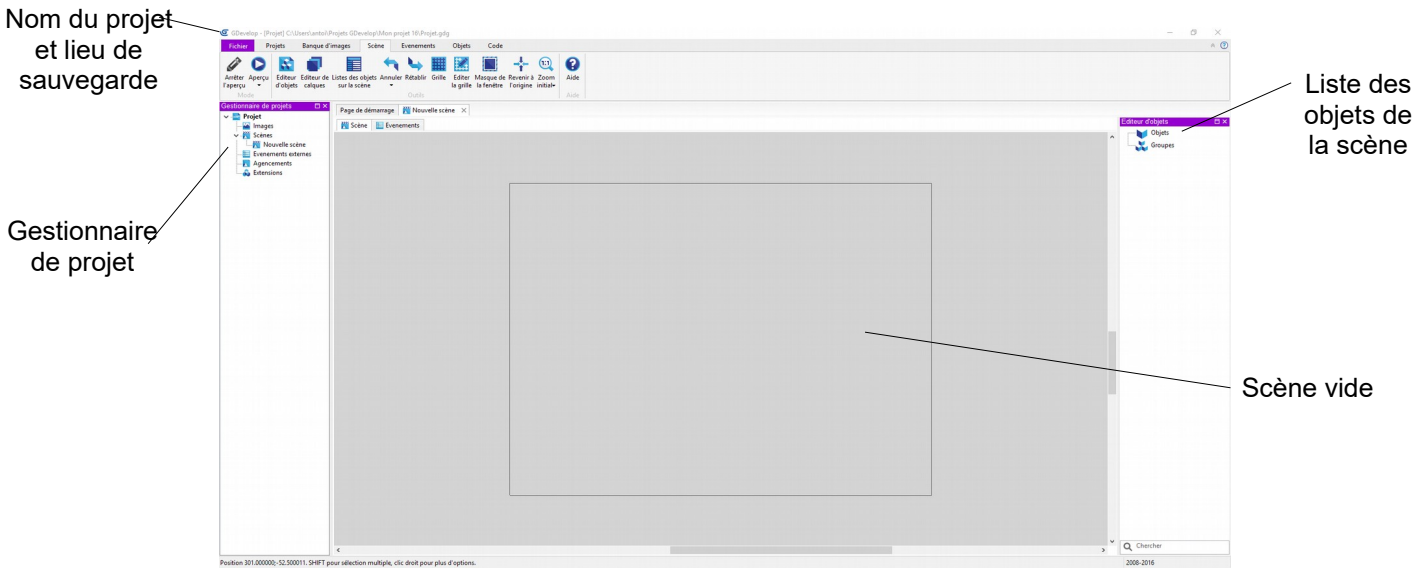
Le terme projet est une façon de dire "jeu". Ainsi, pour créer un nouveau jeu il faut cliquer en haut à gauche sur l'icône "Nouveau". Une nouvelle fenêtre apparaîtra et nous avons ici le choix entre les deux catégories de plateformes à savoir "HTML5" et "Natif". Une fois la plateforme choisie, un modèle de jeu est proposé. Les modèles sont des jeux pré créés que l’on peut modifier ou dont on peut s’inspirer pour faire son propre jeu. Ils diffèrent selon la plateforme choisie. Il est également possible de choisir un projet vierge afin de démarrer son jeu à partir de rien. On choisit son modèle en cliquant sur l’icône lui correspondant, il en est est de même pour la création d’un projet vierge. Chaque jeu, ou projet, est constitué d’une multitude de scènes qu’il faudra assembler afin de constituer notre jeu.

Création d’une scène

Une fois le jeu créé, nous pouvons faire une scène. Une scène désigne un environnement de jeu. Elle peut représenter une salle, un menu de jeu ou tout autre chose. Il est important de noter que la caméra peut se déplacer à l’intérieur de la scène, permettant ainsi de montrer ou de cacher certains endroits, ou encore de se déplacer dans une grande salle.

Pour créer la scène, il faut faire un clic-droit sur « Nouvelle scène » dans le gestionnaire de projets, sur la gauche de l’écran, puis cliquer sur « Ajouter une scène ». Il est possible ensuite de renommer cette scène, à nouveau grâce à un clic-droit, puis à « Renommer ». Pour sélectionner une scène, il suffit de double-cliquer sur la scène voulue. La scène apparaît alors au centre de l’écran. Elle est pour au départ vide. Le fond est gris et les bords sont délimités par un cadre noir. Il est ensuite possible d’ajouter un fond à la scène, en créant un objet pour cela.

À gauche se trouve le gestionnaire de projet, qui regroupe les différentes scènes du jeu, les événements externes, les agencements et les extensions. À droite, la liste des objets est affichée, qui est pour, au départ, vide.



Fenêtre s'ouvrant à la création d'une salle

Les objets

Tous les éléments qui peuvent être ajoutés à une scène sont appelés "objets". Ils concernent : le fond d'une salle, les murs, le joueur, des ennemies, des projectiles, une plateforme, des obstacles, des portes, des particules, etc. Il faut toutefois distinguer les musiques et les sons, qui sont gérés indépendamment.

L'ensemble de ces éléments à implémenter dans une salle sont donc des objets. Comme on peut le remarquer à la vue de tous les exemples cités précédemment, les possibilités sont nombreuses.

Pour ajouter des objets à une scène nous avons deux choix, soit cliquer sur l'onglet "Objets" présent en haut de la page puis on clique sur "Ajouter un objet", soit on fait un clic droit sur "Objets" sur la droite de la page dans l'onglet "Éditeur d'objets" puis, encore une fois, on clique sur "ajouter un objet". On a ensuite le choix entre plusieurs types d'objets : les "sprites" ("sprite" est le nom donné aux images animées), les objets "texte", les particules (qui sont définies ci-dessous), etc. De base, seuls les "sprites" sont débloqués. Les autres sont grisés, ils devront être activés dans les extensions avant de pouvoir être utilisés.

Les "sprites" sont des objets auxquels peuvent être ajoutés des animations permanentes ou activées par une condition. Ils sont utilisés pour les personnages, mais aussi pour des portes, des fonds de salle, etc. Ce sont les objets les plus importants.

Comme le terme l'indique, les objets "texte" permettent d'afficher du texte à l'écran, avec lequel il est possible d'interagir, par exemple en cliquant dessus en jeu pour amener à une nouvelle scène (très utile pour créer différents menus de jeu).

Les particules sont des effets plus ou moins animés qui apparaissent, puis disparaissent en fonction d'une condition (nous reviendrons sur les conditions par la suite) comme une collision avec un autre objet. Par exemple, un projectile lancé par le joueur atteint une cible et à l'impact, de petites particules apparaissent au point dudit impact.

Une fois l'ajout de l'objet effectué (en cliquant sur le type d'objet qui nous intéresse) nous pouvons le nommer comme souhaité, par exemple "Player" pour l'objet correspondant au joueur. Il est important de nommer l'objet car par la suite nous allons devoir utiliser le nom de l'objet dans des conditions. Il faut donc faire attention à l'emploi de lettres en majuscules qui pourraient ne pas être reconnues plus tard. Le logiciel fait en effet la distinction entre les lettres majuscules et les minuscules.

Un objet peut être modifié de différentes façons en faisant un clic droit sur l’objet concerné et en choisissant le type de modification que l’on veut apporter. On peut lui assigner une image, modifier ses animations, ajouter un comportement particulier (comme suivre un autre objet par exemple) ou encore modifier ses caractéristiques. Les animations et les images seront développés plus en détail par la suite.

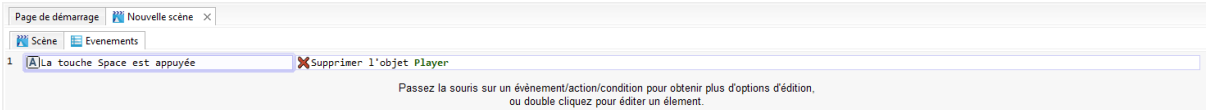
Les caractéristiques d’un objet peuvent être modifiées en cliquant sur l’objet et en sélectionnant "Autres propriétés". Une petite fenêtre apparaîtra sur laquelle nous pouvons cliquer sur "Éditer" afin de modifier tout ce qui touche à l'apparence de l’objet, sur "Variables" pour modifier les variables correspondant à l’objet (sa vie si c’est un ennemi ou un joueur par exemple) ou encore sur "Ajouter un comportement" pour lui assigner un comportement, par exemple en faire un objet contrôlable, un joueur. Dans la fenêtre des comportements, une brève description est disponible permettant de comprendre l'intérêt de chaque comportement.

Conditions, actions et événements

Les événements sont ce que l'on pourrait le plus associer à du codage sur GDevelop. Ils vont nous servir à coder toutes les interactions du jeu. Ils correspondent en codage à des tests "if". Pour créer un nouvel événement de scène il faut se rendre dans l'onglet "Evénements" de la scène correspondante et cliquer sur "Ajouter un événement" en haut à gauche de l'écran.

Un événement est structuré en deux parties : à gauche se trouvent les conditions et à droite les actions. Une action aura lieu uniquement si toutes les conditions définies sont remplies. Pour choisir une condition ou une action, il faut cliquer sur "Ajouter une condition/action" sur l'événement qui nous intéresse. Une fenêtre va s'ouvrir dans laquelle sont triées les différentes conditions/actions par catégorie et par sous catégories.

Prenons un exemple : ci-dessous, un événement a été créé. Il comporte une action et un événement. La condition (à gauche) est la pression sur la touche espace. L'action (à droite) est la suppression de l'objet player. Donc, durant le jeu, si la touche espace est activée, l'objet "Player" disparaîtra de l'écran. Notons que si plusieurs conditions sont définies, l'événement ne sera activé que si toutes les conditions sont validées. Si plusieurs actions sont définies, elles seront toutes réalisées en même temps lorsque les conditions sont validées.



Exemple d'un événement comprenant une condition et une action

Distinguons maintenant les deux types d'événements présents dans GDevelop, à savoir les événements liés à une scène et les événements externes. Les événements de scène ne sont valables que quand la scène correspondante est active. Ils codent les différentes interactions valables uniquement dans la scène. Les événements externes, quant à eux, ne correspondent à aucune scène directement mais peuvent être implémentés dans plusieurs d'entre elles grâce à des liens.

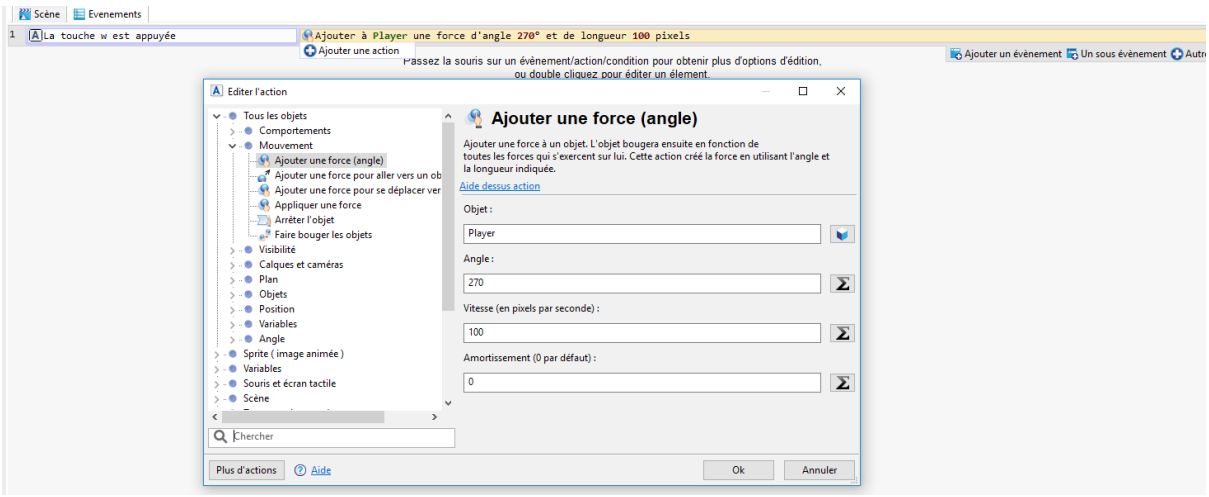
Les événements externes se trouvent dans l'onglet "Gestionnaire de projet" sur la droite de l'écran sous "événements externes". Les pages d'événements externes sont similaires aux pages d'événements de salles. Elles sont aussi composées de conditions et d'actions. Ces événements externes permettent de stocker des bouts de code qui sont utilisés dans plusieurs salles. Par exemple, le code servant à déplacer le personnage principal peut être stocké dans un événement externe, car il revient dans toutes les salles (ou presque). Pour cela, il faut d'abord créer la page d'événements externes, puis créer un lien de la salle voulue à la page d'événements

externes voulue.

Les événements ressemblent beaucoup à des lignes de codes dans la manière où ils sont agencés. On peut par ailleurs ajouter des sous-événements qui dépendent d'autres événements. On distingue un sous-événement, car il est un petit peu décalé sur la droite par rapport à l'événement que l'on pourrait qualifier de principal. Un sous-événement ne sera déclenché que si l'événement principal correspondant est actif et que sa ou ses conditions sont remplies.

Des commentaires peuvent également être ajoutés dans les pages d'événements. Ils permettent par exemple de décrire les événements de la scène. Ils sont particulièrement utiles si le jeu est ouvert par une autre personne que le créateur qui cherche à comprendre les mécaniques du jeu.

Par exemple : pour coder le fait que lorsque l'on appuie sur "W" le joueur se déplace vers le haut il faut, pour la condition, se rendre sous "clavier" puis cliquer sur "touche pressée" puis entrer la touche "w" dans la partie droite de la fenêtre. Ensuite pour l'action il faut se rendre sous "Tous les objets", puis sous "mouvements" et cliquer sur "ajouter une force (angle)", puis entrer sur la partie droite de la fenêtre le nom de l'objet correspondant à votre joueur (par ex : "player"), entrer l'angle de la force (ici 270° pour aller en haut, bizarrement 90° va vers le bas) et la vitesse du joueur (en pixels par seconde). L'amortissement doit être laissé à 0.



Exemple d'une action permettant un déplacement

Il existe beaucoup de conditions et d'actions. Par exemple pour coder la mort d'un objet quelconque (comme le joueur ou un boss), il faut mettre en condition « variable "Points de Vie" <= 0 » et en action « supprimer l'objet "boss" » ou « changer pour la scène "défaite" ». La condition implique l'utilisation d'une variable, une notion nécessaire en codage, qui va être expliquée dans le paragraphe suivant.

Les variables

Les variables permettent de stocker des informations. Une variable, à laquelle on assigne un nom, stocke une valeur numérique, qui peut être appelée dans le programme et qui peut être modifiée. Par exemple, une variable nommée *Life* pourrait être utilisée pour stocker la quantité de vie du joueur. Elle serait diminuée chaque fois que le joueur subit des dégâts et lorsqu'elle atteint 0, le jeu s'arrêterait.

Il existe trois types de variables : les variables globales, les variables de scène et les variables d'objets.

Les variables globales sont accessibles depuis n'importe quelle scène du jeu. Elles permettent de contenir des informations utiles dans plusieurs ou dans toutes les scènes. Par exemple, une variable globale peut être utilisée pour le nombre de dégâts qu'inflige le joueur aux ennemis. Cette variable devrait être utilisée dans toutes les scènes de combat et pourrait évoluer au cours du jeu, si le joueur améliore ses capacités. Pour créer une variable globale, il faut effectuer un clic droit sur le nom du jeu dans le "gestionnaire de projet" (à gauche de l'écran), puis cliquer sur "modifier les variables globales". Une fenêtre s'ouvre ensuite et affiche la liste des variables globales utilisées dans le jeu, ainsi que leurs valeurs initiales (leur valeur attribuée au lancement du jeu). Il est possible d'ajouter et de supprimer des variables, ainsi que de modifier leurs valeurs dans cette fenêtre.

Les variables de scène sont spécifiques à une scène et ne peuvent pas être appelées ou modifiées depuis une autre scène. Elles pourraient par exemple être utilisées pour stocker la vie d'un ennemi qui n'apparaîtrait que dans une scène. Pour créer une variable de scène, il faut effectuer un clic droit sur la scène voulue dans le gestionnaire de projet, puis cliquer sur "modifier les variables initiales". Il est alors possible d'ajouter, de supprimer et de modifier des variables.

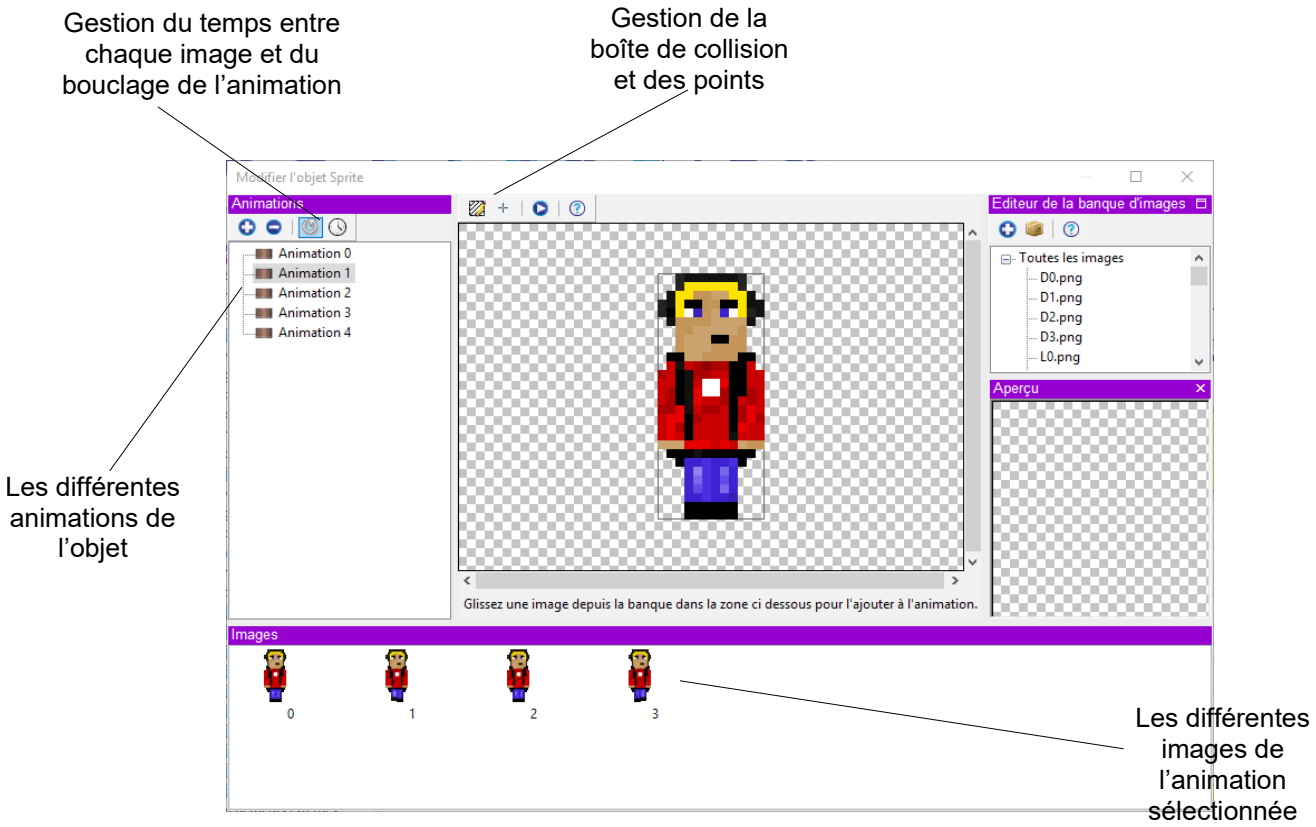
Les variables d'objets sont, quant à elles, spécifiques à un seul objet, mais peuvent être utilisées dans plusieurs scènes si l'objet est global. Elles sont plus difficiles à utiliser et moins intuitives. Il n'est pas possible de voir une liste de variables d'objets comme pour les deux autres types de variables. Elles doivent être directement ajoutées ou modifiées depuis les événements. Elles peuvent tout de même être très utiles, par exemple si une salle contient plusieurs ennemis similaires. Les variables d'objets permettraient de gérer individuellement la variable *Life* de chaque ennemi, au lieu d'avoir plusieurs variables de scène *Life0, Life1, Life2, Life3,* etc.

Les images

L’aspect graphique d’un jeu est important pour le rendre esthétique, clair et attractif. Nous n’aborderons que brièvement la création des graphismes en présentant le logiciel que nous avons utilisé. Nous nous concentrerons sur la gestion des images dans le logiciel GDevelop.
GDevelop utilise des images au format jpg et png. Il permet également de décomposer des animations au format gif en images individuelles de format png grâce au bouton Décomposeur d’images, sous l’onglet Projet. Toutes les images sont regroupées dans une banque d’image, qui permet de les visualiser facilement. Il est également possible d’y ajouter directement de nouveaux graphismes en les faisant glisser dans cette banque,

Les images sont principalement utilisées pour donner une texture aux objets, que ce soit une animation ou une image fixe. Pour cela, nous utilisons des objets sprite. Ils permettent notamment de gérer l’animation d’un personnage qui se déplace, une explosion ou encore le fond d’une salle.

Double cliquer sur un objet Sprite permet d’ouvrir un menu d’édition de l’objet. Il est ensuite possible de définir plusieurs animations pour cet objet ; les animations sont numérotées. Chaque animation peut contenir une seule ou plusieurs images, qu’on ajoute au bas de la fenêtre. Il est important de mettre les images dans l’ordre, pour que l’animation fonctionne bien. Il est ensuite possible de définir le temps entre chaque image, ce qui permet de créer des animations plus ou moins rapides. Cela peut, par exemple, être utile pour distinguer un personnage qui court et un personnage qui marche. Un second paramètre peut être modifié : il est possible soit de boucler l’animation, par exemple pour un personnage qui marche sans s’arrêter, soit la faire tourner une seule fois, par exemple pour une explosion ou une porte qui s’ouvre. Si l’animation n’est pas bouclée, elle s’arrête sur la dernière image, qui reste affichée.



Fenêtre permettant de gérer les animations d'un objet

Différents points peuvent être définis pour un objet, comme la tête, le centre ou le bord de l'objet. Ils permettent par exemple de faire apparaître des projectiles à un endroit précis de l'image (par ex. au bout d'un canon) pour rendre le jeu plus réaliste et esthétique.

Finalement, il est possible de modifier le masque de collision de l'objet, plus couramment appelé en anglais *hit box*. Ce masque, toujours invisible en jeu, sera utile lors de la condition de collision entre deux objets. Si deux masques de collision entrent en contact, la collision est validée. Par défaut, le masque est un rectangle aux dimensions de l'image, mais il peut être modifié, en un polygone dont les coordonnées des sommets de la zone de contact sont définissables.

Le logiciel de création que nous avons utilisé s'appelle Piskel. Il peut être utilisé en ligne sur piskelapp.com ou être téléchargé gratuitement sur le même site. Il permet de créer des images et animations en pixel art (des images pixelisées) très simplement. Il permet d'exporter les images au format png ou les animations au format gif. Toutes les images de notre jeu ont été créées sur ce logiciel.

Sons et Musiques

Les sons et les musiques apportent un petit plus au jeu et le rendent plus attrayant et vivant. Les sons et les musiques ne sont pas utilisés de la même façon. Les sons sont en fait des bruitages, des effets sonores qui vont appuyer une action ou une interaction dans le jeu et permettre une meilleure lisibilité du jeu. On peut par exemple accentuer un dégât asséné à un objet par un bruitage, ce qui rend le fait de s'être fait toucher ou d'avoir touché plus évident pour le joueur. La musique, en revanche permet de créer une ambiance et ajouter une dimension émotionnelle au jeu. Une musique inquiétante peut par exemple donner un sentiment d'insécurité au joueur et le rendre plus à même de faire des erreurs. Bien évidemment il n'est pas recommandé de mettre une musique qui ne correspond pas au contexte. Par exemple, pour un boss de fin de niveau, il sera préférable d'utiliser une musique rythmée et attrayante pour motiver le joueur.

Des formats de fichier spécifiques sont nécessaires pour ajouter des sons et des musiques. GDevelop utilise des fichiers ".wav" pour les sons et des fichiers ".ogg" pour les musiques. Il est donc préférables de télécharger les éléments qui nous intéressent directement en ".wav" ou en ".ogg". Les développeurs GDevelop ont toutefois mis à disposition un convertisseur de fichiers directement dans le programme. Il se trouve dans l'onglet "Projet" en haut de l'écran sous "Convertisseur" (celui avec les deux notes de musique). Il faut toutefois avoir téléchargé "LAME.exe" ainsi que "OGGENC.exe" au préalable pour que cela fonctionne. Il reste malgré tout la possibilité de convertir les fichiers sur des sites internet libres. Nous avons pour notre part utilisé le site audioconverto.com, qui permet notamment de convertir des fichiers mp3 en fichiers ogg.

Pour implémenter des sons et des musiques dans le jeu, il suffit de créer un nouvel événement avec la condition voulue, par exemple "au lancement de la scène" pour une musique ou encore "lorsque l'objet [...] est en collision avec l'objet [...]" pour un bruitage. Ensuite pour l'action il faut se rendre sous la section "Audio" et sélectionner ce qui nous intéresse. Il est nécessaire de comprendre la notion de canal.

Un canal est exactement comme un canal radio. Il s'agit d'une "fréquence" ou plutôt d'un numéro de canal sur lequel va être diffusé le son ou la musique. Les canaux sont utiles pour passer d'un son (ou d'une musique) à l'autre sans avoir à mettre le morceau précédent en pause. On peut donc superposer les sons sur différents canaux pour avoir plusieurs bruitages en même temps. Par exemple, le canal 0 correspond au bruit d'un vaisseau spatial auquel on superpose les bruits d'artillerie laser sur le canal 1.

Les extensions

Les extensions sont des fonctionnalités déjà intégrées à GDevelop, qui permettent d'éviter d'écrire un morceau de code compliqué. Ces extensions permettent de gérer de nouveaux objets, de nouvelles conditions, des actions, des événements ou ajoutent des comportements. Ces extensions sont par défaut désactivées, elles doivent être choisies dans l'onglet "Extensions" du gestionnaire de projet. Certaines extensions ne sont disponibles que pour les jeux natifs et ne peuvent pas être utilisées pour des jeux HTML5.

Nous ne décrirons pas toutes les extensions car elles sont trop nombreuses. Nous nous concentrons sur les extensions "Mouvement vu du dessus", "Objet Texte", "Comportement chemin" et "Comportement Destruction à la sortie de l'écran", car elles sont très utiles et permettent de comprendre plus facilement les autres extensions.

Tout d'abord, l'extension "Mouvement vu du dessus" permet de déplacer un objet sur l'écran, ce qui est très utile, surtout dans les jeux en deux dimensions. Par défaut, l'objet se déplace sur quatre directions, sur les axes X et Y, mais il est possible de le faire se déplacer dans huit directions, soit en diagonale également. Cette extension présente un avantage dans les déplacements en diagonale par rapport à appliquer des forces sur le personnage : si on veut déplacer un objet en diagonale en mettant deux forces perpendiculaires. Les forces s'additionneraient comme deux vecteurs perpendiculaires et la vitesse résultante serait finalement égale à environ 1,41 fois la vitesse normale. La conséquence serait que l'objet irait plus vite en diagonale que lorsqu'il avance sur les axes X et Y. L'extension permet donc de déplacer l'objet en huit directions tout en gardant une vitesse constante, qui peut être choisie et modifiée en cours de jeu.

Deuxièmement, l'extension "Objet Texte" est également utile. Elle permet, comme son nom l'indique, d'afficher du texte à l'écran. Les objets « texte » sont conçus avec "Créer un objet", puis en sélectionnant "Objet texte", si l'extension a été activée précédemment. Un texte peut alors être écrit dans l'objet. La police d'écriture, la taille de police et la couleur du texte peuvent être choisies dans la fenêtre de paramètres de l'objet. Les objets textes peuvent ensuite être gérés comme les sprites dans les événements.

Ensuite, l'extension "Comportement chemin" est pratique et permet de simplifier considérablement le code. Ce comportement permet de prédéfinir les chemins qu'un objet va emprunter. Ces chemins sont composés d'une suite de segments. Plusieurs chemins peuvent être définis pour un même objet. Ils seront ensuite choisis dans les événements. La vitesse de déplacement le long du chemin peut ensuite être

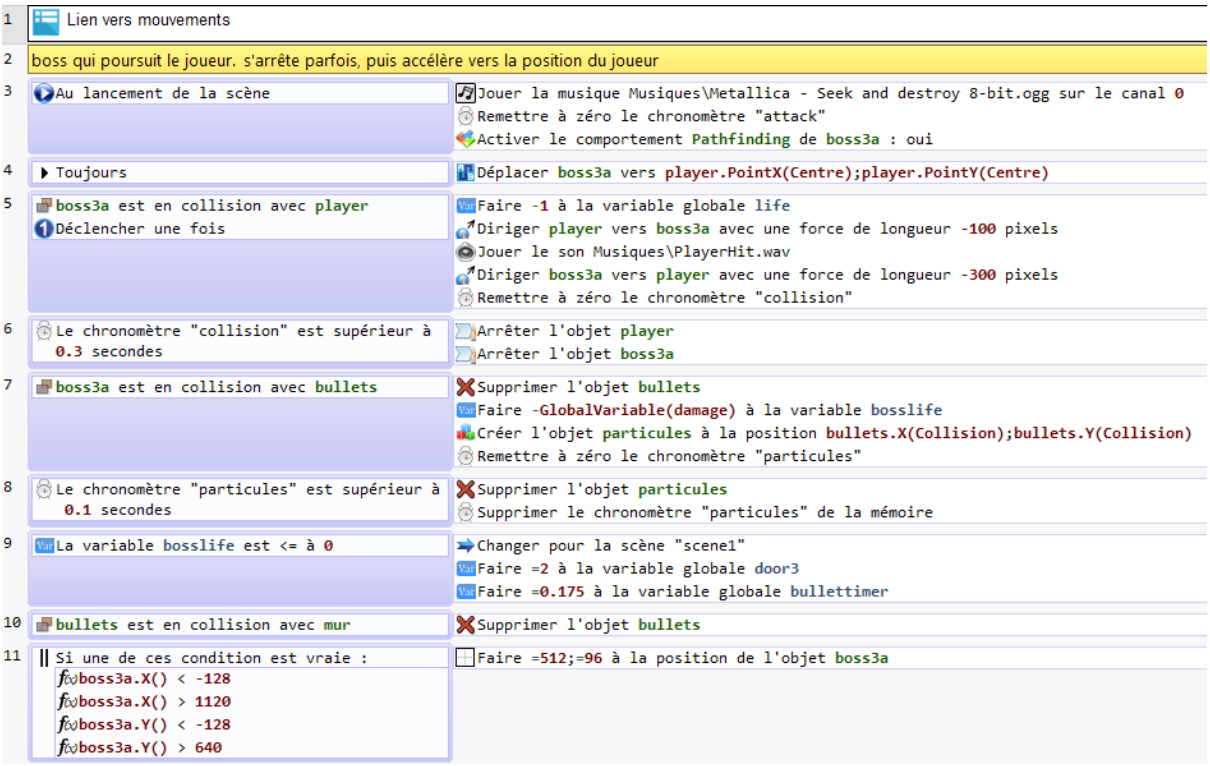
déterminée. Cet événement permet de déplacer des objets sur des chemins compliqués, ce qui serait complexe à gérer avec des chronomètres et des forces. L'objet parcourt en boucle un chemin qu'on lui a assigné, ce qui permet de rendre le jeu plus complexe et intéressant. Une cible qui se déplace est en effet plus difficile à toucher qu'un ennemi immobile.

Finalement, l'extension "Comportement Destruction à la sortie de l'écran" est peut-être la plus simple à comprendre, mais est non négligeable. Elle permet de supprimer des objets qui ne sont plus visibles à l'écran. Pour l'appliquer à un objet, il suffit d'aller dans les propriétés de cet objet et d'activer le comportement. Cette extension peut notamment être utilisée pour des projectiles, qui seraient nombreux et qui ne percuteraient pas d'ennemis. Le comportement permettrait de les supprimer quand ils ne sont plus visibles et donc inutiles. De plus, l'ordinateur a ainsi moins d'objets à gérer en même temps.

Exemple de salle

Afin de rendre plus concret les concepts expliqués précédemment, prenons un exemple concret du code de notre jeu. Il s'agit du boss 3, qui est un œil avec des pattes qui poursuit le joueur en s'arrêtant et en accélérant .

Le commentaire se fera ligne par ligne. Le numéro de chaque ligne est écrit sur la partie gauche de l'image.



Exemple de code, page 1

1. Lien amenant aux événements externes, permettant les déplacements du joueur.

2. Commentaire décrivant le code qui suit.

3. Actions à effectuer au lancement de la scène : lancer la musique, mettre le timer « attack » (utilisé pour les attaques) à zéro et activer le comportant de recherche de chemin pour le boss.

4. Déplace l'ennemi vers le joueur.

5. À chaque fois que le boss et joueur se touchent, le joueur perd un point de vie et un son indiquant la perte de vie est lancé. Il n'est activé qu'une fois, sinon le joueur prendrait des dégâts en continu. Le joueur est également poussé par le boss

(force négative appliquée au joueur en direction du boss) pour l'empêcher de rester sur le boss. Le boss est également repoussé par le joueur de la même manière. On remet le timer « collision » à zéro.

6. Le timer « collision » arrête le joueur qui a été repoussée par le boss et inversement à la ligne 5, pour arrêter qu'ils se repoussent.

7. Quand un projectile tiré par le joueur touche le boss, le projectile est supprimé et le boss perd un nombre de points de vie égal à la variable gérant les dégâts qu'inflige le joueur aux ennemis. Une particule indiquant que le tir à fait mouche est créée à l'endroit de la collision. On remet le timer « particules » à zéro.

8. Le timer « particules » (démarré à la ligne 7) indique combien de temps la particule reste à l'écran (ici 0.1 secondes). Passé ce délai, la particule est supprimée. Le chronomètre « particules » est supprimé.

9. Indique ce qu'il faut faire quand le boss n'a plus de point de vie, c'est-à-dire à sa mort. On revient à la scène 1, qui correspond au menu du jeu. On augmente la cadence de tir du joueur, qui est un bonus donné à la fin de la salle, puis on ferme la porte correspondant au boss que l'on vient de battre.

10. Quand un tir rate l'ennemi et touche un mur, le projectile est détruit.

11. Au cas où le boss sortirait de l'écran par erreur (Quand ses coordonnées sont en dehors du cadre de l'écran), le boss est remis dedans.

Exemple de code, page 2

12. Commentaire décrivant l'attaque du boss : il se déplace vers le joueur lentement, puis, après 3 secondes, il s'arrête. Il accélère ensuite brusquement vers le joueur, s'arrête et accélère à nouveau. Il recommence ensuite à se déplacer lentement. L'attaque se joue en boucle.

13. Lorsque la variable « attack » vaut 1, le boss avance rapidement vers la position (X ;Y). Lorsque la variable vaut 0, rien ne se passe. La variable n'a aucune autre valeur possible.

14. Après 3 secondes écoulées au chronomètre « attack », le boss s'arrête et une animation indiquant l'attaque est lancée.

15. À 3.4 secondes, la position du joueur (X ;Y) est mémorisée dans deux variables X et Y.

16. À 3.5 secondes, la variable « attack » est fixée à 1 : Le boss avance rapidement vers la position (X ;Y) définie à la ligne 15. L'animation normale (0) est remise au boss.

17. À 4.0 secondes, l'attaque est arrêtée, le boss s'arrête.

18. Toujours à 4.0 secondes, le boss est arrêté (ligne inutile car il l'était déjà). L'animation d'attaque est relancée.

19. À 4.4 secondes, la position du joueur est à nouveau mémorisée comme à la ligne 15.

20. Similaire à la ligne 16, mais à 4.5 secondes.

21. Similaire à la ligne 17, mais à 5.0 secondes.

22. À 5.1 secondes, le chronomètre « attack » est remis à 0, l'attaque reprend donc depuis la ligne 14. Le boss recommence à avancer vers le joueur.

Bilan personnel

Le travail de maturité que nous avons réalisé nous a permis d'acquérir de nouvelles compétences, de développer notre intérêt pour l'informatique et de prendre du recul par rapport aux jeux vidéo. Ce travail nous a permis d'apprendre à nous organiser et à réaliser un projet conséquent sur le long terme.

En décidant de créer un jeu, nous avions en tête ceux auxquels nous jouions, notamment le jeu *The Binding of Isaac*. Nous ne rendions pas compte de la difficulté de la création et du temps nécessaire à cela. Nous imaginions donc créer un jeu bien plus complexe mais nous avons par la suite dû revoir nos ambitions à la baisse, en ayant tout de même un résultat satisfaisant à la fin.

Nous avons passé les trois premiers mois à découvrir le logiciel, en visualisant des vidéos tutoriels, en ouvrant des petits jeux disponibles dans le logiciel et en testant diverses fonctionnalités de ce-dernier. Nous avons également créé des petits jeux simplistes (notamment une copie du jeu « Pong ») pour se familiariser avec le logiciel.

Après ces trois mois, nous avons enfin commencé le jeu à proprement parler, en commençant par le personnage principal et le code nécessaire pour son déplacement. Nous avons ensuite créé une salle « test » d'un ennemi simplifié, qui a ensuite évolué et qui est devenue une des salles du jeu.

Au début, nous utilisions des textures déjà implémentées ou trouvées gratuitement sur des sites. Nous avons ensuite commencé à créer nos propres textures, ce qui fut plus long et complexe que prévu. Comme mentionné précédemment, nous avons utilisé le logiciel « Piskel », avec lequel nous avons eu des problèmes (notamment avec le format des images utilisé). Grâce à cela, nous avons été familiarisés avec les différents formats d'image et avec le *pixel-art*.

Par ailleurs, la réalisation d'un projet aussi conséquent sur une année nous a appris à avoir une certaine rigueur au niveau de la gestion du temps et de notre organisation. Travailler en binôme était stimulant et par moments plus complexe. Ce travail est gratifiant et a favorisé une grande autonomie.

Bibliographie

Sites web – tous valides en octobre 2018

gdevelop-app
https://gdevelop-app.com/

piskelapp
https://www.piskelapp.com/

Tutoriel pour débutant pour GDevelop 4
http://wiki.compilgames.net/doku.php/gdevelop/tutorials/beginnertutorial2

Différents tutoriels pour GDevelop 4
http://wiki.compilgames.net/doku.php/gdevelop/tutorials

Forum GDevelop 4
http://forum.compilgames.net/viewforum.php?f=17

audioconverto
http://www.audioconverto.com/fr/

Vidéos

CREER UN JEU VIDEO – VATSUG. *[GDEVELOP] Jeu de plateforme 01 – CUJV* [Vidéo en ligne]. Youtube, 17/11/2014 [Consulté en décembre 2017].
https://www.youtube.com/watch?v=8XLZSQtwFgY

FLORIAN RIVAL. *Make a HTML5 Platformer game with GDevelop* [Vidéo en ligne]. Youtube, 02/04/2014 [Consulté en décembre 2017].
https://www.youtube.com/watch?v=5jdOR-NAiSA

Musiques utilisées dans le jeu

0. Nom de la vidéo youtube pour la musique en 8 bit (si il y a)
lien youtube
Musique originale : *Titre*, **Artiste** Album date

1. *Aerials [8 Bit Cover Tribute to System of a Down] - 8 Bit Universe*
https://youtu.be/hm5Vb680tt0
Original : *Aerials*, **System of a Down**, Toxicity 2001

2. *Sober [8 Bit Tribute to Tool] - 8 Bit Universe*
https://youtu.be/u_R9DTsKZWs
Original : *Sober*, **Tool**, Undertow 1993

3. *Metallica - The four Horsemen 8-bit*
https://youtu.be/I8IfGeJ1syM
Original : *The Four Horsemen*, **Metallica**, Kill 'Em All 1983

4. *Metallica - Seek and destroy 8-bit*
https://youtu.be/pVP-gaSkILs
Original : *Seek and Destroy*, **Metallica**, Kill 'Em All 1983

5. *The Doomed [8 Bit Tribute to A Perfect Circle] - 8 Bit Universe*
https://youtu.be/jwbgEwfMAxc
Original : *The Doomed*, **A Perfect Circle**, Eat the Elephant 2018

6. *Toxicity [8 Bit Cover Tribute to System of a Down] - 8 Bit Universe*
https://youtu.be/hdt2ZhmzVm8
Original : *Toxicity*, **System of a Down**, Toxicity 2001

7. *Au revoir*, **Igorrr**, Savage Sinusoid 2017

8. Mastodon Blood & Thunder 8-bit
https://youtu.be/qMEcCloGWac
Original : *Blood and Thunder*, **Mastodon**, Leviathan 2004

9. Metallica - Eye Of The Beholder 8-Bit
https://www.youtube.com/watch?v=KDkC2083AAM
Original : *Eye Of The Beholder,* **Metallica,** ...And Justice for All 1988

Bruitages utilisés en jeu

freesound.org
https://freesound.org/people/timgormly/sounds/

Certains sons tirés de ce site ont été modifiés avec le logiciel Audacity, qui peut être téléchargé ci-contre : https://www.audacityteam.org/

Remerciements

Nous tenons tout d’abord à remercier vivement M. Bernard Gisin, notre maître accompagnant et enseignant d’informatique, qui nous a aidés et soutenus tout au long de notre projet. Nous le remercions de son implication dans le projet et pour le temps qu’il y a consacré.

Nous remercions notre camarade Thomas Lecomte, qui nous a aidé à créer plusieurs graphismes importants de notre jeu, dont le personnage principal. Il nous a également apporté plusieurs idées quant à l’esthétisme et l’univers du jeu.

Nous remercions enfin les testeurs de notre jeu qui nous ont permis de mettre en évidence des problèmes tant de gameplay, d'équilibrage ou tout simplement de code.