Dynami’K

Marin LUET, Joris TRIART, Paul DE MISCAULT, Paul WACQUET

Table des matières

Note d’intention

Concept

Dynami’K est un jeu platformer dans lequel deux joueurs s’affrontent sur le même appareil (smartphone, tablette ou ordinateur). Le but est d’avoir un score plus faible que son adversaire à la fin du temps de jeu. Pour cela, il faut faire attention à ne pas toucher les obstacles, qui augmentent le score du joueur.

Le joueur court sans arrêt comme dans le jeu Subway Surfer et les obstacles apparaissent de manière aléatoire. Le joueur peut les esquiver en sautant au-dessus d’eux de la même manière que dans Geometry Dash.

La musique influence le jeu. Lorsque des beats sont détectés (à l’aide d’algorithmes de détection), la vitesse du jeu est modifiée pour s'accélérer de manière légère mais notable. Le but est de faire de la musique un véritable acteur de l’expérience de jeu, à l’inverse de la plupart des jeux où la musique peut-être désactivée et n’est pas importante.

Public cible

Le jeu est tout public. Son fonctionnement simple et intuitif le rend abordable par n’importe qui. De plus, il sera adapté à une utilisation en toute circonstance grâce à une durée des parties réglable. Les deux joueurs pourront donc décider de leur temps de jeu.

Objectifs

Ce jeu se veut divertissant et facile à prendre en main. Pour ces raisons, une seule interactions sera autorisée : sauter. Cependant cette interaction simple provoquera beaucoup de réponses (haptique (vibrations), sonore (un son de saut) et visuelle (effets de particules et shake de l'écran)).

Nous voulons également rendre ce jeu plus immersif grâce à une ambiance originale et esthétique. Il s’agit de faire apprécier à l’utilisateur une harmonie entre la musique, son gameplay et ce qu’il voit.

Cahier des charges

Ressources médias (textes, sons, images, vidéos)

Nous avons utilisé des musiques chargées par défaut comme Elektronomia par exemple. Nous voulions des musiques assez dynamiques et électroniques, un peu dans le style des musiques de Geometry Dash.

Nous avons eu recours à quelques asset Unity pour nous faciliter la tâche et ne pas avoir à tout concevoir, dessiner depuis le début, ce qui nous aurait pris trop de temps. Nous avons utilisé :

- 2000+ Faces (https://assetstore.unity.com/packages/2d/characters/2000-faces-139263)
- Dark Theme UI (https://assetstore.unity.com/packages/2d/gui/dark-theme-ui-199010)
- Simple 2D Platformer Asset Pack (https://assetstore.unity.com/packages/2d/characters/simple-2d-platformer-assets-pack-188518)
- UI Samples (https://assetstore.unity.com/packages/essentials/ui-samples-25468
- UI Sfx (https://assetstore.unity.com/packages/audio/sound-fx/ui-sfx-36989)

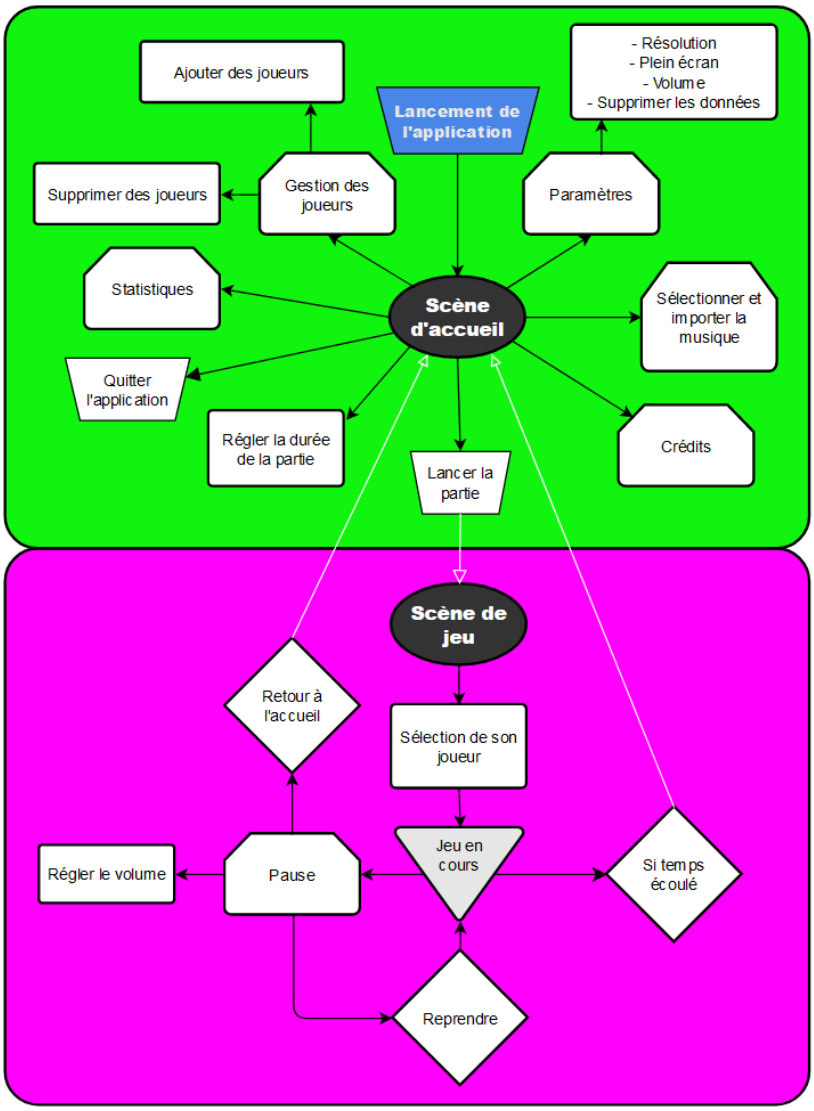
Nous n'avons pas utilisé de texte particulier, ni d'images ou de vidéo car nous n'en avions pas l'utilité et nous voulions garder une application simple et pas trop lourde.

Structuration et navigation

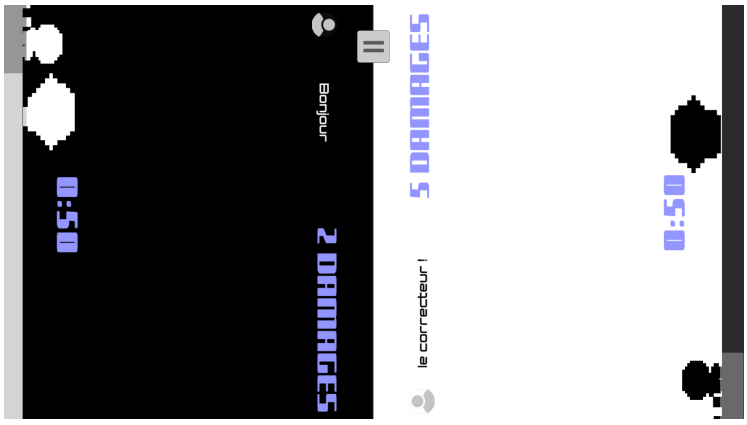
Notre jeu dispose de deux scènes (environnements séparés dont le but est de distinguer les différentes phases du jeu) :

- la scène d'accueil, sur laquelle arrive le joueur au lancement de l'application
- la scène de jeu, dans laquelle le jeu se déroule.

L'algorigramme ci-dessous représente les fenêtres qu'il est possible d'ouvrir et les actions que l'on peut faire sur ces scènes. Les liens entre les scènes sont représentés par des flèches blanches.



Scène de menu



Scène de jeu

La navigation se veut simple, intuitive et réversible car chaque fenêtre est accessible grâce un bouton et dispose d'un autre bouton pour revenir en arrière. Chaque bouton est prédictible car l'action qui lui est associée est soit écrite sur le bouton (bouton "Start" par exemple), soit représentée par une icône communément utilisée pour ce type d'action (bouton de paramètres ou de pause par exemple). Nous avons utilisé ces deux modes de représentations pour des raisons d'optimisation graphique et pour ne pas surcharger de texte l'écran alors qu'une icône plus compacte peut être aussi voire plus efficace qu'un mot. Chaque fenêtre dispose d'un titre en police *Gajraj One* et tous les autres textes sont écrits en police *Jupiter*. Nous trouvons ces polices dynamiques, adaptées au jeu et lisibles.

Formes et degrés d'interactivité

La première vraie forme d'interactivité est lors du démarrage du jeu, dans le menu principal sur lequel les joueurs interagissent en particulier pour choisir la musique et le temps de la partie avant de commencer de jouer. Il y a aussi d'autres boutons sur ce menu qui permettent d'atteindre d'autres menus :

- Le menu de paramètres dans lequel se trouve quelques options (résolution d'écran, volume…). Ce menu comporte également un bouton supplémentaire qui permet d'effacer les données du jeu. Pour éviter que l'utilisateur efface les données par erreur, il est nécessaire d'appuyer de nombreuses fois sur ce bouton afin qu'il fonctionne.
- Le menu de gestion de joueurs qui permet de gérer les joueurs enregistrés dans le jeu, avec pour chacun un nom et une icône de joueur arbitraire attribuée.
- Le menu de statistiques dans lequel sont stockés les scores de chaque joueur, le nombre de parties jouées et le temps de jeu.
- Le menu de crédits regroupe les développeurs qui ont créés le projet Dynami'K

Plusieurs menus déroulants sont à disposition, le temps du jeu se règle grâce à un slider. Chaque icône et bouton forment une animation et génèrent des bruitages distincts lorsqu'on clique dessus, ce qui permet de donner une sensation de fluidité dans la navigation.
Le bouton permettant d'ajouter des musiques depuis son ordinateur ouvre un explorateur de fichier intégré au jeu qui permet de parcourir les données de son appareil pour ajouter la musique.

Lors du lancement de la partie une fois le menu principal passé et que les joueurs se sont mis d'accord sur les paramètres de lancement, chaque joueur doit de son côté choisir depuis une liste déroulante de quelle joueur il s'agit afin que les scores obtenus lors de la partie puissent s'enregistrer au bon endroit. Après ce menu, le gameplay peut réellement démarrer.

Pour ce qui est des contrôles, nous les avons voulu les plus simples possible car rien ne nous frustre plus que d'installer un jeu et de ne pas comprendre comment on y joue. Pour cette raison, nous avons voulu nous limiter à une seule interaction, la plus intuitive et la plus basique : appuyer sur l'écran. Son effet est de faire sauter le personnage et cela suffit à instaurer un gameplay. Cependant, il y a quand même une courbe de progression car au début le joueur ne se rend pas compte qu'il peut rester appuyer (mais cela ne l'empêche pas de pouvoir jouer) puis quand il va s'en rendre compte, cela va lui générer un avantage par rapport à l'autre joueur et donc une montée d'excitation. Il en ira de même quand il découvrira qu'il peut sauter sur les monstres qui marchent puis sur ceux qui volent.

Pour pallier le manque de complexité dû au faible nombre de possibilités du joueur, il nous a été indispensable de mettre au cœur de notre projet l'interactivité. En effet, il faut qu' un simple appui sur l'écran provoque un maximum de feedback. Pour cela, nous avons ajouté des caméra shakes (discrets mais présents), des particules systèmes, des changements de couleurs et des retours haptiques très légers (seulement sur mobile).

Choix graphiques et d'interface

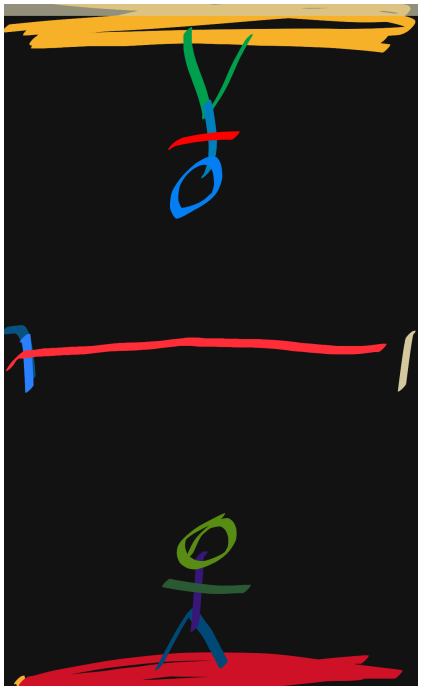
Nous avons choisi de faire un jeu avec un style pixel art noir et blanc à gros pixel car cela met en avant la simplicité du jeu. En effet, les jeux rétro avaient un stockage et des ressources limitées, ce qui en faisait des jeux simples (pas trop de sous-menus complexes) et légers (dans l'ordre des Mo ce qui est atteint car on est à 170Mo ce qui est très peu pour un jeu).

Grâce à ce style, nous espérons que cela amène les joueurs à faire l'effort de l'installer en cassant la barrière mentale du temps d'installation et de la place prise sur l'appareil.

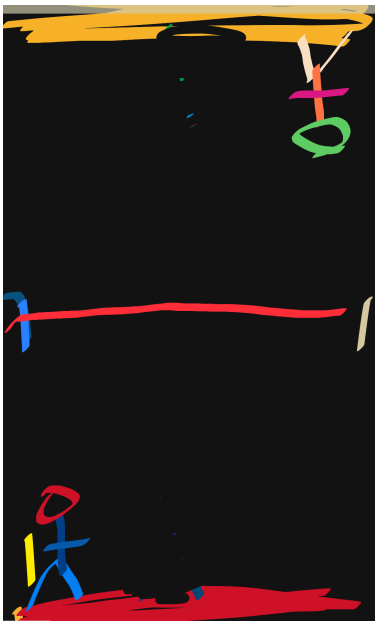
N'ayant pas toujours d'assets en pixel art (et même les assets en pixel art n'ont pas la même résolution), nous avons construit un système qui fait automatiquement le rendu en pixel art noir et blanc par défaut (parfois il se colore pour des effets).

Pour ce qui est de l'interface, nous avons voulu l'optimiser pour les appareils mobiles. Pour cela, nous avons soigneusement évité tout raccourci claviers et nous l'avons optimisé pour un usage par clic (donc aucun drag and drop etc). Nous avons eu le problème du format de l'écran (16:9 ou 21:9) car tous les téléphones n'ont pas le même format. Pour des raisons de simplicité avec la version pc et parce qu'un jeu en 16:9 sur un écran 21:9 rend mieux qu'un jeu en 21:9 sur un écran 16:9, nous avons décidé de partir sur le format 16:9. Il s'est aussi posé la question du sens dans lequel on affiche les deux joueurs. L'idéal sur pc aurait été que les deux soient comme ça (à gauche) :

et sur mobile cela aurait été ça (à droite) :



Notre cible première étant la version mobile, c'est la deuxième configuration que nous avons choisie. De plus, suite aux retours de notre première présentation, nous avons décalé le personnage à gauche afin de laisser un plus grand champ de vision au joueur comme ceci :



Choix techniques

La synchronisation du jeu avec la musique s'effectue par l'intermédiaire d'un algorithme de détection de beats.

Tout d'abord il nous faut récupérer le spectre audio. Nous récupérons toutes les données audio sous forme de spectre avec GetSpectrumData avec la méthode de Blackman (W[n] = 0.42 - (0.5 * Cos(n/N)) + (0.08 * COs(2.0 * n/N) avec un découpage du spectre en buffers de 512. Pour visualiser ce spectre, nous le découpons en 8 par la méthode des moyennes et nous assignons le résultat à la coordonnée y de 8 rectangles. Pour améliorer la visibilité de la hausse des signals, nous appliquons une exponentielle. De ce fait, un signal sonore un peu plus élevé que les autres (un instrument par exemple) se distinguera bien plus.

Pour ce qui est de la détection de beat, nous avons décidé de nous basé sur "l'énergie moyenne des basses de la musique". C'est-à-dire que nous prenons l'ensemble du spectre des basses (de 20Hz à 500Hz) puis nous faisons leur moyenne puis nous y appliquons une exponentielle pour la même raison. Et si cela dépasse une certaine valeur (appelée threshold et qui est réglable), alors nous considérons cela comme étant un beat.

L'algorithme est exploité dans unity à travers le composant AudioProcessor qui s'occupe d'analyser et de lancer les sources audios. Des ajustements des valeurs de sensibilité et d'échantillonnage audio ont ensuite dû être ajustés pour convenir à l'utilisation de l'algorithme dans un jeu avec la diminution de la sensibilité et de la taille d'échantillonnage pour ne pas détecter tous les beats (environnement de jeu beaucoup

trop chaotique sinon).

Ces beats sont ensuite utilisés pour altérer la vitesse du jeu, le spectre audio de l'AudioProcessor est également utilisé pour faire trembler le sol avec différentes intensités.

A chaque beat la vitesse de jeu alterne et passe d'une vitesse x1 à une vitesse x1.5, cela permet au joueur de ressentir les effets de la musique sans être trop déstabilisé par les changements de fréquence de la musique. Plusieurs options ont été testées mais finalement abandonnées :

- Le changement de vitesse de la musique pendant une durée définie à chaque beat, par exemple la vitesse passe à x3 pendant 1 seconde à chaque beat, le problème était que les changements de vitesse devait être brutaux, sinon on ne les remarquait pas, cela rendait le jeu beaucoup trop difficile et imprévisible, de plus en ajoutant une durée statique de changement de vitesse on perdait la synchronisation du jeu avec la musique.
- Un effet de glow (comme si on éclairait le jeu avec un projecteur) avait été rajouté à chaque beat pour faire briller l'écran à chaque beat, mais l'effet rendait le jeu trop illisible pour le joueur, il a donc été supprimé.
- Une utilisation du spectre audio en bandes avait également été envisagée comme décor pour le jeu, mais on perdait également en lisibilité, l'effet a donc été supprimé également et remplacé par la moyenne des basses et qui sert de sol. Les ennemis se mettent donc à "sauter" si il y a des beat

Pour fidéliser les joueurs dans la durée, nous avons mis en place une sauvegarde des données du jeu, pour que l'on n'ait pas l'impression de lancer le jeu pour la première fois à chaque lancement. Pour cela, nous devions pouvoir différencier les deux joueurs et les identifier pour leur attribuer des données. Le jeu permet donc de créer des joueurs, qui ont un nom et une icône personnalisés, et de les sélectionner au début de chaque partie. Il peut y avoir plus que deux joueurs. Les données récupérées sont le nombre de parties lancées et le meilleur score par minute. Ces statistiques sont visibles dans une fenêtre dédiée. Nous avons également sauvegardé une préférence de volume (le dernier volume sélectionné), pour ne pas forcer l'utilisateur à baisser le son à chaque partie par exemple, et une préférence de l'affichage ou non de l'écran d'explication au début de chaque partie. Pour réaliser le stockage en local de ces données, une base de données aurait été appropriée, mais compte tenu du faible volume de données et pour des raisons de simplicité, nous avons premièrement opté pour un simple fichier CSV. Le problème est que nous n'avons jamais réussi à faire fonctionner cette solution sur Android, nous

supposons principalement pour des raisons de permission et d'accès. Nous avons donc finalement basculé tout ce système sur les PlayerPrefs d'Unity. C'est une solution très pratique et simple à mettre en place et surtout, qui fonctionne sur Android.

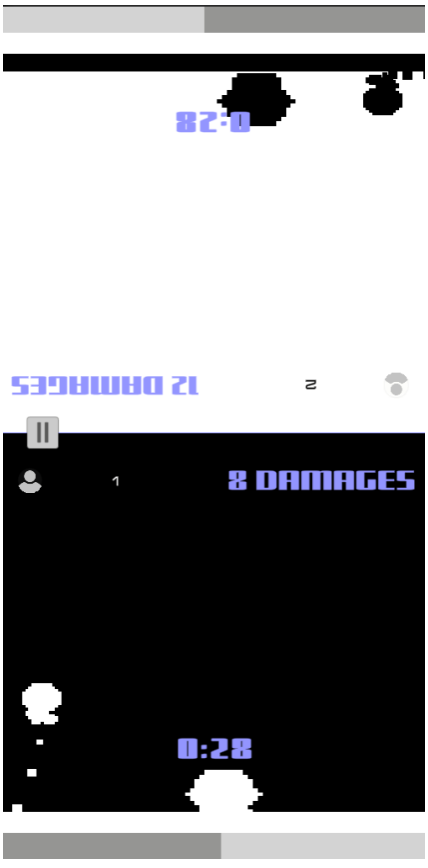
Afin de ne pas créer trop de données, il est possible de supprimer toutes ces données depuis l'application.

Scénario

"Dynami'K" est un jeu platformer compétitif conçu principalement pour mobile qui se joue sur un seul appareil pour deux joueurs. Le but du jeu est de terminer la partie avec un score inférieur à celui de son adversaire une fois le temps imparti écoulé. Pour y parvenir, les joueurs doivent éviter les obstacles qui apparaissent de manière aléatoire, car ceux-ci augmentent leur score.

La musique joue un rôle important dans le jeu, car elle influence l'expérience de jeu en temps réel. Des algorithmes de détection de beat sont utilisés pour modifier certains facteurs tel le rythme d'apparition des ennemis et obstacles, la vitesse du jeu et l'apparence de l'environnement. Notre priorité est de faire de la musique un élément clé de l'expérience de jeu, contrairement à la plupart des autres jeux où la musique est secondaire et peut tout aussi bien être désactivée.

Au début de la partie, les joueurs se mettent d'accord sur une durée qui déterminera le temps de jeu. Ensuite, les deux joueurs se partagent chacun une partie de l'écran face à face et contrôlent le personnage avec la touche de saut (actionable simplement en appuyant sur l'écran). Il y a des musiques par défaut qui déterminent l'expérience de jeu, mais il sera également possible de choisir ses propres musiques depuis son appareil qui affectent le jeu de la même façon.



Le jeu a comme intention d'être divertissant et facile à prendre en main. Pour ces raisons, une seule interactions sera autorisée : sauter. Cependant cette interaction simple provoquera beaucoup de réponses (haptique (vibrations), sonore (un son de saut) et visuelle (effets de particules et shake de l'écran)).

Conclusions

Marin LUET

Je suis très content de ce projet car nous nous étions donné les objectifs les plus simples possibles afin de pouvoir terminer le jeu et nous avons réussi au final à faire tous les objectifs secondaires que nous nous étions fixés. Il a fallu s'organiser autour du projet, assigner qui fait quoi, expliquer qu'est ce qu'on a fait sur le projet afin de ne pas créer de bugs quand on s'appuie sur le code d'un autre et enfin se stimuler en temps qu'équipe. En un mot : ce jeu a surpassé de très loin nos attentes et procure une réelle satisfaction et fluidité dans son gameplay et interface.

Joris TRIART

Ce projet a été assez chaotique pour moi dû à de nombreuses difficultés techniques, surtout liées à des problèmes matériels, 30 minutes de lancement du projet unity, crashs répétés, problèmes de gestions de versions. Malgré tous ces problèmes je suis content d'avoir pu apporter ma contribution au projet en implémentant l'algorithme de détection de la musique et les effets liés à la musique. Ce projet fut tout de même très enrichissant pour ce qui est de la partie conception est réflexion : Comment ajouter du challenge sans rendre le jeu aléatoire et frustrant pour le joueur ? Comment ajouter des effets liés à la musique sans rendre le l'expérience trop chaotique ? Cette étude de l'interactivité de la responsivité de l'application m'a fait découvrir de nouvelles problématiques liées au développement de jeux vidéo.

Paul DE MISCAULT

J'ai beaucoup aimé faire partie de l'aventure *Dynami'K* durant ce semestre. C'est avec une idée assez claire dans la tête que nous nous sommes engagés dans une quête formidable : *le projet de SI28*… Nous avons fait notre chemin à quatre et surmonté les difficultés les unes après les autres. Après avoir vaincu les monstres UNITY et ANDROID et triomphé du Temps, notre idée initiale s'est transformée pour devenir réalité. Ce

périple a été très enrichissant pour moi car j'ai découvert les coulisses de la création d'un jeu vidéo et la gestion de projet associée. Je n'avais jamais fait de projet numérique aussi important que Dynami'K, je suis content et fier du résultat, malgré quelques ajustements imprévus dus à des difficultés techniques. En somme, j'ai pris beaucoup de plaisir à participer à ce projet, qui m'a également apporté de nombreuses compétences.

Paul WACQUET

Ayant déjà créé quelques jeux avec Unity dans le passé, j'étais ravi de pouvoir contribuer à un nouveau projet avec cet environnement de développement. Notre groupe s'est impliqué dès le début du semestre et grâce à ce rythme cela nous a permis de réaliser un projet vraiment complet, malgré le temps limité à notre disposition. J'ai eu l'occasion de participer au développement de plusieurs fonctionnalités particulières telles que la gestion du gameplay multijoueur et la conception de l'interface (avec la possibilité de choisir ses musiques). Ce fut pour moi une très bonne expérience de pouvoir participer au développement d'un jeu vidéo dans un groupe de 4 personnes, car c'est différent de développer un jeu seul ou à 2 personnes. Nous avons pensé à un gameplay original et ça nous motivait à le réaliser car il ne s'agit pas d'un simple jeu 2D comme les autres mais nous avions ici une réelle idée d'intégration musicale. On voulait vraiment voir ce que ça allait donner et je suis content du résultat final.