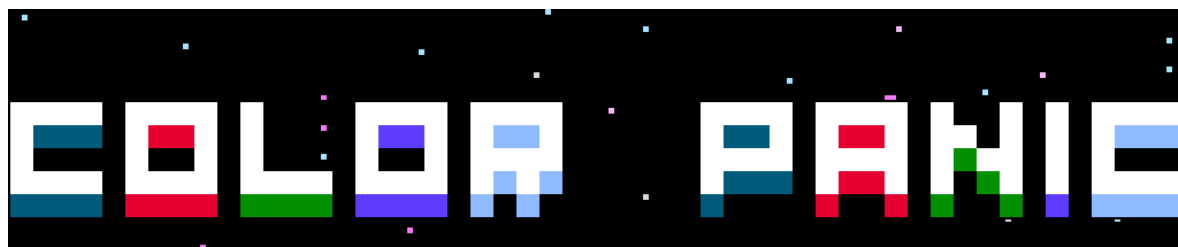


Rapport de projet



Projet de Borne d'arcade

Sommaire

Introduction	3
1. Objectif initial	
1.1 Pourquoi ce choix ?	4
1.2 Planning initial	4
1.3 Concept du jeu initial	5
2. Déroulement du projet	
2.1 Game Design	6
2.2 Graphismes du jeu	7
2.3 Levels design	8
2.4 Musique	10
2.5 Construction de la borne	10
2.6 Programmation	11
2.7 Sauvegarde sur le téléphone	12
3. Résultat et conclusions	
3.1 Planning finalement suivi	13
3.2 état final du projet	13
3.3 comment l'améliorer	13
4. photo de la borne et guide pour jouer	15

Introduction

Lors du projet d'électronique de PeiP 2 organisé par nos professeurs d'électroniques, nous avons dû trouver un projet à mener à bien en six mois ayant un rapport avec l'électronique et ayant la particularité de pouvoir avoir une communication sans fil avec un autre appareil électronique (dans notre cas nous avons choisi une communication SSH comme nous en parlerons plus tard dans la partie 2.7 Sauvegarde sur le téléphone). Ce projet nous a permis d'apprendre à concevoir et mener à bien un projet ce qui pourra se montrer utile dans nos futurs métiers. Après mûres réflexions nous avons choisi de concevoir une borne d'arcade, et aussi de créer notre propre jeu de A à Z qui tournera sur cette borne.

1 Objectif Initial

1.1 Pourquoi ce choix ?

Au départ on avait beaucoup de mal à trouver un sujet qui serait à la fois intéressant et faisable dans les temps,

On avait eu l'idée d'une lampe d'ambiance qui reproduirait la couleur de l'écran d'un ordinateur pour augmenter l'immersion dans les jeux ou les films mais finalement on a jugé ce projet trop simple et on a opté pour l'idée de M. Masson c'est-à-dire une borne d'arcade.

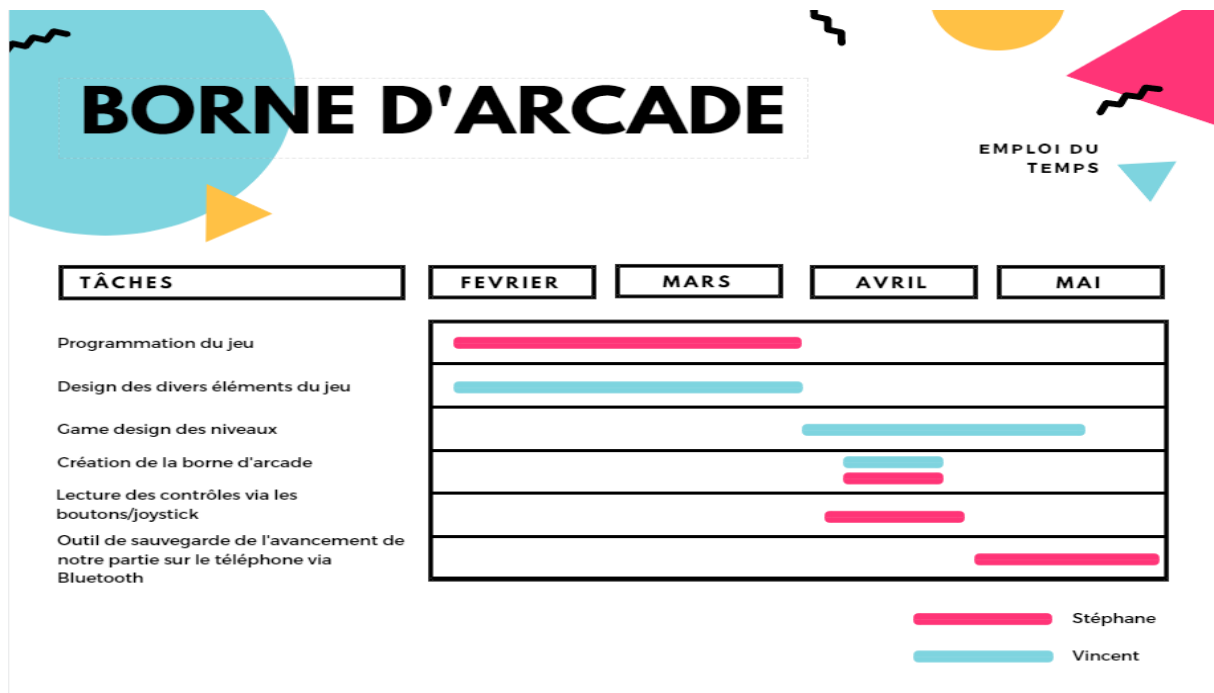
On s'est ensuite demander quels jeux mettre sur cette borne.

Et on s'est dit que tant qu'à faire une borne d'arcade, autant créer le jeu qui va avec de A à Z nous sommes donc partis dans cette idée de créer un jeu et la borne pour l'accueillir.

1.2 Planning Initial

Le planning que l'on avait établi à la base était trop stricte pour pouvoir être totalement respecté on avait prévu de finir le code du jeu début avril, plus tous les sprites et assets du jeu et commencer les niveau début avril et finir début mai . Puis courant mai créer la borne et installer tous les composants à

l'intérieur, les musiques quant à elles devaient être créées tout au long du temps imparti pour réaliser le projet.



Voici l'image du planning que l'on avait prévu début Février

Concept Initial

Lorsque l'on a commencé ce projet on est parti sur l'idée de créer un jeu plus dans le style arcade comme un MetalSlug, on avait créé un premier personnage qui ressemblait à un petit viking :



De plus on avait à la base prévue de coder le jeu sous java mais finalement après avoir mis environ 20 heures pour ouvrir une fenêtre et mettre des objets à l'intérieur, on a opté pour « Processing », la même chose nous prenait 10 min.

2 Déroulement du projet

2.1 Game Design

Pour le Game design, c'est-à-dire quels sont les éléments du jeu, on a choisi d'utiliser plusieurs différents bonus qui sont au nombre de 5, plus l'absence de bonus . Ainsi le personnage sans bonus peut se déplacer de gauche à droite et sauter et en prenant ces différents bonus il gagne différentes capacités, respectivement :

- Bonus de double saut : le personnage peut sauter 2 fois avant de devoir retoucher le sol pour ressauter.
- Bonus de téléportation : le personnage avance de quelque blocs et passe à travers les obstacles, de plus l'axe y du joueur est bloqué lors de la téléportation.

- Bonus de Dash : le personnage va avancer comme pour le TP mais il ne traverse pas les obstacles de plus il peut aller plus long qu'avec un simple TP.
- Bonus d'inversion de gravité : avec ce bonus le joueur ne peut plus sauter, il peut en revanche inverser la gravité.
- Bonus No clip : Ce bonus permet au joueur de passer à travers certain mur, en revanche ce bonus n'a pas pu être implémenté dans le jeu final, on en reparlera dans la partie programmation.


2.2 Graphismes du jeu

Pour les graphismes on a opté pour un style « retro pixel art », le

héros est un dérivé de Megaman :



Les blocks sont de trois type :

Les dalles : 

Les lignes : 

Les briques : 



La porte de fin de niveau :

Il y a aussi des piques qui tue le personnage au contact :



Un total de 215 images ou Gifs ont été créés pour le jeu.

Et 203 sont effectivement utilisés !

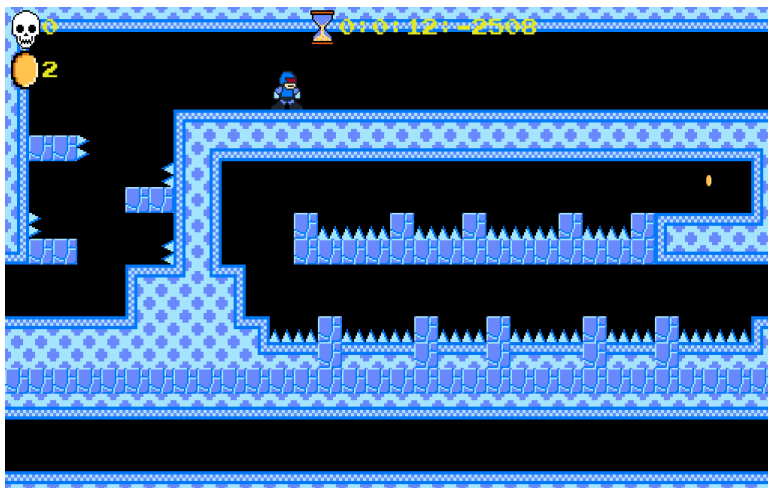


Image en jeu

2.3 Levels design

Le jeu a été découpé comme une matrice où chaque case fait du 8*16 pixels ce qui sur l'écran ayant une résolution de 1024*600 pixels soit 128*38 cases permettant de créer un éditeur de niveau plus simple à gérer car chaque case aura une valeur (un String) qui permettra de le différencier dans le programme permettant de lancer le jeu. Sur ces

cases différents objets seront amenés à interagir ensemble, les murs, les pics, les bonus, le héros, les blocs de fin de niveau et la porte de fin de niveau, un mur fait du 2*2 une case bonus est du 1*1, le héros a une taille de 6*3, les blocs de fin de niveaux on aussi une taille de 1*1 et finalement la porte fait une taille de 7*4. Pour placer n'importe lequel de ces blocs il faut cliquer dans la case en haut à gauche, de même pour effacer le bloc.



Dans l'ordre :

bonus de double saut, outil pour placer des blocs empty, soit effacer, bonus pour se téléporter, case qui quand activé permet de placer des blocs noClip, donc des blocs dans lesquels on peut passer si on active le bonus de la case suivante, soit le bonus noClip, cases nécessaires pour placer le héros, bonus permettant

de dash, cases pour placer la fin d'un niveau, cas pour placer des bonus inversant la gravité et enfin des cases pour augmenter le score.

Dans toutes ces cases à effet, seule les cases noClip et BnoClip ne seront pas utilisés, car la programmation du bonus n'a pas réussi à être finalisée dans le jeu.

Dans l'applet on peut aussi voir le nom de la case que l'on a sélectionné, sur l'image on a sélectionné le bloc empty, réinitialisé sert à effacer tous les blocs du niveau courant et enfin dans la case « nom du fichier : » on rentre le numéro du niveau sur lequel on souhaite travailler tandis que la case « niveau suivant : » sert à indiquer le numéro du niveau que l'on finit donc pour passer au niveau 2 on met end2 pour passer au niveau 69 on met end68.

Comme vu sur l'image on peut aussi choisir la musique du niveau, de plus il y a deux blocs permettant de finir le niveau, la door et le bloc end avant de les sélectionner il faut écrire dans la case « niveau suivant : » le numéro du niveau où l'on souhaite aller -1 (comme expliquer précédemment).

Quand on clique sur sauvegarder on enregistre dans le dossier data/levels/lvlsHitbox/ un fichier lvlJ.txt où J est le numéro du niveau, celui-ci contient $128 \times 38 + 1$ lignes où sur chaque ligne est écrit la valeur de la case associée (il s'agit d'un String i.e : BTP, empty, hero, spikeLeft...) puis on sauvegarde dans le dossier data/levels/lvlsPrint/ un fichier lvlJ.png où on prend une photo du niveau créé.

2.4 Musique

Pour les musique on avait prévu à la base de faire des musiques de type « retro chip tunes » comme dans les vieux jeux tels Mario ou Metroid. Mais finalement il s'est avéré plutôt difficile de crée ce type de musique, on a donc opté pour des musiques de type Retrowaves et Synthwaves qui étaient mieux maîtrisées, une image pour l'album des



musiques du jeu a aussi été créé :

2.5 Construction de la borne

La borne a été construite en bois contreplaqué de 1cm d'épaisseur.

Elle a été assemblée avec des vis puis on a bouché les trous avec de la pâte a bois, la borne a ensuite été poncé puis peinte en noire

On a ensuite placé le logo du jeu sur la borne, puis on a placé les composants à l'intérieur de la borne, les enceinte, les boutons, l'amplificateur, l'écran et la Raspberry pi, on a installé une multiprise dans la borne puis monté une porte à l'arrière pour pouvoir accéder à l'intérieur.

2.6 Programmation

Le programme du jeu est commenté pour comprendre le fonctionnement du code, il est disponible sous format PDF dans le dossier du compte rendu de la soutenance, pour ce qui est des bugs et problèmes que l'on a eu lors de la programmation, ils sont nombreux.

Coppé Vincent – Stéphane Viale G2

11

Tout d'abord un bonus n'a pas pu être programmé comme on l'aurait souhaité, on a donc décidé de l'abandonner, celui-ci est le bonusNoClip comme expliqué dans l'applet, les problèmes étaient dû au principe de déplacement dans le jeu, car celui-ci te « déplace » dans un tableau, les cases hero sont déplacé selon l'action de mouvement que l'on a fait, or si l'on se déplace dans un mur avec le bonusNoClip, ce mur doit être une case hero, mais rester aussi une case wall donc on la renommait (uniquement dans le tableau propre au code pas dans le fichier en permanence et à chaque mouvement) heroWallNoClip.. Et quand on était plus dans cette case la case redevenait wallNoClip, sauf que dans la version du bonus actuellement programmée les cases herowallNoClip se baladait en même temps que le héros la hitbox du héros s'agrandissait il y avait énormément de mauvaise interaction entre le bonus et l'environnement, et à cause d'autres bugs plus importants, on a décidé d'abandonner ce bonus.

Ensuite nous avons eu beaucoup de problème à cause de la Raspberry car celle-ci a des problèmes au niveau du chargement d'un son dans sa mémoire ce qui fait que, pour les musiques on les charge les unes après les autres ce qui ne pose pas de gros problèmes mais pour les SFX ils doivent être toujours chargé car si on a des freezes dus à des chargements en plein milieu d'un niveau ça rendrait l'expérience de jeu beaucoup moins agréable, ce qui fait qu'actuellement plusieurs sons sont retenus dans la même variable et selon si l'on est dans un menu ou non le son doit passer du son menu au son jeu, sauf que pour décharger un son on a trouvé que la méthode de la librairie « minim »(celle que l'on utilise pour le jeu) qui

supprime tous les sons donc on perd la musique d'ambiance et si on n'arrête pas le son, on a deux sons en même temps..

Au final pour corriger ce bug on a dut réduire le bit rate de nos sons, et supprimer le son du bonusDash et du bonusGravitySwap, sans oublier que dans les menus de pause il n'y a plus de sons lorsque l'on passe d'une case à l'autre du menu ou bien que l'on valide notre choix.

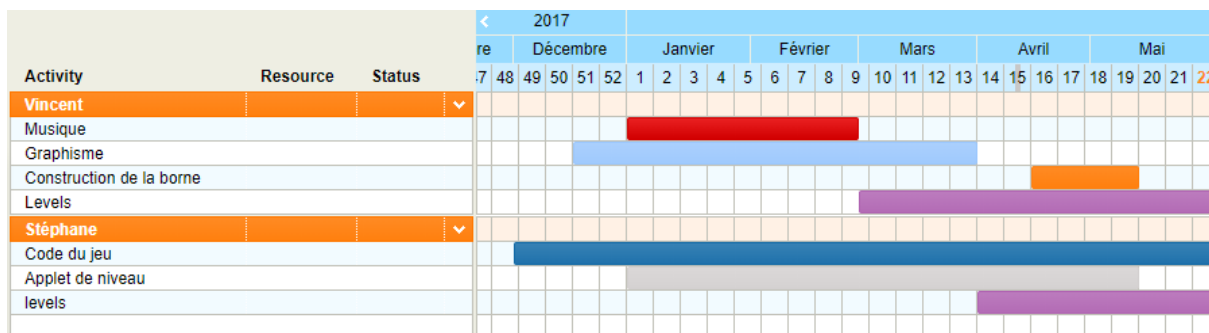
Le dernier bug qui lui est toujours présent, est un décalage sur les millisecondes de notre jeu, en effet, lors du calcul du temps, il se peut que plusieurs secondes passent en même temps (ou au contraire aucune), ce qui cause un décalage dans le calcul des millisecondes de notre jeu, ce bug n'a pas été corrigé car il dépend majoritairement du temps de boucle de la machine faisant tourner le jeu, et comme il est différent pour chaque ordinateur, on a abandonné de patch ce bug.

2.7 Sauvegardes sur le téléphones

Pour la sauvegarde sur le téléphone, nous avons décidé de laisser l'utilisateur gérer la majeure partie du fonctionnement de la sauvegarde, en effet, nous lui donnons l'identifiant et le mot de passe de la Raspberry avec le ligne de code qu'il doit insérer dans son terminal pour pouvoir soit copier le fichier où est sa sauvegarde ou charger sa sauvegarde.

3 Résultats et conclusion

3.1 Planning finalement suivi



Finalement les musiques ont pris moins de temps que prévu pour être créées, de plus le code du jeu et l'éditeur de niveau ont été bien plus compliqués que prévu à faire.

3.2 Etat final du projet

Le projet a été fini à 95%, tout ou presque a pu être implanté et créé dans les temps.

On a eu quelque problème avec notamment le bonus de no clip et la connexion au téléphone

3.3 comment l'améliorer

Notre projet peut être amélioré de bien des façons au niveau de l'optimisation du code, on aurait pu rajouter plus de niveau si on avait eu plus de temps, on aurait aussi pu ajouter d'autres éléments de jeu tels que des plateformes mobiles ou des ennemis qui suivent le joueur, sur la borne le seul problème est le lag de la Raspberry qui n'était pas assez puissante pour faire tourner le jeu vraiment de manière fluide.

4 Photos de la borne et comment jouer au jeu



Pour pouvoir jouer au jeu il suffit de télécharger la branche master du github comme ceci :

Elzebalth / GameProject

Watch 0 Star 0 Fork 2

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights

PeiP2 arduino

251 commits 2 branches 0 releases 2 contributors

Your recently pushed branches:

OldVersion (3 minutes ago) Compare & pull request

Branch: master New pull request Create new file Upload files Find file Clone or download

AnEdge fefe		Latest commit c372910 an hour ago
colorPanic	Final upload of the projet V27	an hour ago
Color Panic.pptx	uploading pptx powerpoint	6 hours ago
README.md	Update README.md	3 days ago
Rapport de projet.docx	Final upload of the projet V27	an hour ago
StéphaneVIALE.md	Update StéphaneVIALE.md	18 days ago
VincentCOPPÉ.md	Update VincentCOPPÉ.md	3 days ago

Ensuite il faut dézipper l'archive dans un dossier puis si vous êtes sur linux il faut ouvrir le dossier application.linux64 ou application.linux32 et exécuter le fichier ColorPanic

Si vous êtes sur Windows ouvrir le dossier application.windows64 ou application.windows32 et ouvrir le fichier ColorPanic.exe

Les touches dans le jeu sont :

Z, pour sauter (ou haut dans les menus) Q, pour aller vers la gauche D, pour aller vers la droite S, pour aller vers le bas (utile uniquement dans les menus) J, pour activer le bonus actif (Tp

,dash etc) et pour valider ces choix dans les menus et P pour mettre la pause en jeu et pour sortir du premier écran titre.

