ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

UNIVERSITÉ DU QUÉBEC

rapport technique

PRÉSENTÉ À L’ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

DANS LE CADRE DU PROJET DE FIN D’ÉTUDES

<titre DU DOCUMENT>

par

<nom de famille, Prénom>

<code permanent>

PPRÉSENTÉ À

<PrénoM ET nom de famille DU PROFESSEUR)>

montréal, le <date>

**REMERCIEMENTS**

Thaieasyelec - <http://www.thaieasyelec.net/>

Merci à toute l’équipe de Thaieasyelec qui n’ont pas hésité à m’envoyer une nouvelle plateforme de développement par poste express de Thaïlande lorsque la mienne s’est avérée défectueuse. Sans eux, le prototype comme présenté dans ce rapport n’aurait pas vue le jour.

**<titre du rapport TECHNIQUE>**

<NOM, Prénom>

**RÉSUMÉ**

<Texte interligne simple>

**TABLE DES MATIÈRES**

Page

[INTRODUCTION 2](#_Toc185065390)

[CHAPITRE 1 <titre> 2](#_Toc185065391)

[1.1 <Titre> 2](#_Toc185065392)

[CHAPITRE 2 <titre> 2](#_Toc185065393)

[CHAPITRE 3 <Titre> 2](#_Toc185065394)

[3.1 <Titre> 2](#_Toc185065395)

[CHAPITRE 4 <titre> 2](#_Toc185065396)

[4.1 <Titre> 2](#_Toc185065397)

[CHAPITRE 5 <titre> 2](#_Toc185065398)

[5.1 <Titre> 2](#_Toc185065399)

[CHAPITRE 6 <titre> 2](#_Toc185065400)

[6.1 <Titre> 2](#_Toc185065401)

[CHAPITRE 7 <titre> 2](#_Toc185065402)

[7.1 <Titre> 2](#_Toc185065403)

[CHAPITRE 8 <titre> 2](#_Toc185065404)

[8.1 <Titre> 2](#_Toc185065405)

[CHAPITRE 9 <titre> 2](#_Toc185065406)

[9.1 <Titre> 2](#_Toc185065407)

[CHAPITRE 10 <titre> 2](#_Toc185065408)

[10.1 <Titre> 2](#_Toc185065409)

[CONCLUSION 2](#_Toc185065410)

[RECOMMANDATIONS <S’il y a lieu> 2](#_Toc185065411)

[ANNEXE I <titre> 2](#_Toc185065412)

[ANNEXE II <titre> 2](#_Toc185065413)

[ANNEXE III <titre> 2](#_Toc185065414)

[ANNEXE IV <titre> 2](#_Toc185065415)

[ANNEXE V <titre> 2](#_Toc185065416)

[ANNEXE VI <titre> 2](#_Toc185065417)

[ANNEXE VII <titre> 2](#_Toc185065418)

[ANNEXE VIII <titre> 2](#_Toc185065419)

[ANNEXE IX <titre> 2](#_Toc185065420)

[ANNEXE X <titre> 2](#_Toc185065421)

[APPENDICES <S’il y a lieu> 2](#_Toc185065422)

[LISTE DE RÉFÉRENCES 2](#_Toc185065423)

[BIBLIOGRAPHIE 2](#_Toc185065424)

**LISTE DES TABLEAUX**

Page

[Tableau 1.1 Titre](#_Toc150853242) x

**LISTE DES FIGURES**

Page

[Figure 1.1 Test](#_Toc150853834) x

**LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES**

<Texte interligne simple>

**LISTE DES SYMBOLES ET UNITÉS DE MESURE**

<Texte interligne simple>

INTRODUCTION

<Texte interligne 1 1/2>

# Analyse de la problèmatique

## Définition du besoin

Si vous voulez développer un jeu vidéo, quelques possibilités s’offrent à vous. Vous pouvez bien sûr le faire sur ordinateur, mais il est souvent plus intéressant de jouer dans le confort de son salon. Dans ce cas, vous pouvez vous tourner vers les dernières consoles de jeux qui offrent souvent la possibilité de faire son propre jeu. Cependant dans le dernier cas, vous êtes limité aux fonctionnalités que le fabricant veut bien vous donner.

Il vous reste donc les consoles ouvertes. Ces consoles sont faites dans le but de fournir une plateforme de base pour le développement de jeux vidéo tout en permettant à l’utilisateur de modifier tout le logiciel s’exécutant sur celle-ci. De plus, dans la plupart des cas, ce genre de plateforme offre les plans du matériel afin de permette à quiconque ayant les habiletés nécessaires, de modifier la plateforme. Bref, tous les pouvoirs sont donnés à l’utilisateur. Ceci est très intéressant, mais les choix disponibles présentement comportent plusieurs lacunes.

Premièrement, les solutions existantes n’utilisent pas de contrôleur graphique matériel. Ceci limite grandement les performances, car il est très difficile pour le logiciel de faire le rafraichissement d’un écran. Comme un grand pourcentage du temps du processeur est utilisé pour la gestion de l’écran, il ne reste que peu de temps pour la gestion du jeu lui-même qui est une tâche assez difficile en soi.

Encore du coté du matériel, les solutions existantes on souvent trop peu de mémoire disponible ce qui rend impossible l’utilisation d’image de haute qualité visuelle à l’intérieur du jeu.

Une autre lacune des consoles ouvertes est le logiciel fourni. En effet, celui-ci est souvent très optimisé dû au manque de performance du microcontrôleur ce qui le rend souvent difficile à comprendre. De plus, la plupart de temps, aucun moteur de jeux n’est disponible ce qui rend la programmation de jeux assez ardus. Pour finir, le jeu et le noyau sont souvent indissociables.

Du côté de l’interface utilisateur, les consoles ouvertes actuelles utilisent souvent de simples contrôleurs désuets ce qui limite l’envergure des jeux développés.

## Définition du système

Vue de haut niveau, le système répondant aux besoins d’écrit plus haut ressemble à ceci.

Nous avons donc une entré utilisateur et deux sorties soit l’audio et la vidéo.

Bien que sous cette forme le système semble simple, plusieurs choix s’offre à nous pour la réalisation d’une tel machine. Nous aurons donc un choix à faire et le cheminement est exposé au chapitre suivant.

## Risques et opportunités

# Conceptualisation de la solution

## Contraintes

Comme nous l’avons vue au chapitre précédent, nous avons besoin d’un système ne comportant que quelque entré/sortie. Ceci laisse une grande marge de manœuvre quant au matériel et logiciel utilisé pour la réalisation. Nous devons donc préciser quelques contraintes afin de réduire les possibilités. Suite à une longue réflexion, nous en sommes arrivés à la liste de contraintes suivante.

Table - Lise de contraintes

|  |
| --- |
| Utilisation de manette de jeux récente |
| Capacité plus grande ou égale au Super Nintendo |
| Utilisation d’un moteur de jeux 2D pour les jeux de type « side scroller » |
| Coût de production plus petit ou égale à 50$ |

La contrainte la plus importante est le coût de réalisation du prototype matériel. Ce dernier ne devrait, idéalement, pas dépasser 50$. Il faudra donc être minutieux dans le choix des composants utilisés, sans trop influencer la qualité du produit final. Une courbe d’apprentissage élevée est à prévoir pour le choix des composants tel le type de mémoire externe et le microcontrôleur idéal à utiliser pour notre projet. Aussi, la réalisation du circuit-imprimé à l’aide du logiciel Altium, la conception efficace d’un circuit imprimé afin de minimiser le bruit qui pourrait être causé par les signaux à haute fréquences et l’évaluation des différents types de signaux vidéos disponibles à la sortie de la console sont des tâches importantes qui demandent beaucoup de temps. Il faudra donc se limiter à ce qu’on peut faire dans le temps alloué.

Cependant, du côté logiciel nous serons limités à la puissance de calcul du microprocesseur choisi ce qui ne sera sûrement pas très élevé. De plus, comme nous voulons produire du code facilement réutilisable, nous nous limiterons aux langages de programmation orientés objet. Ce langage devra aussi être assez populaire afin de simplifier l’apprentissage.

## Calculs préliminaires

Suite aux barèmes établies dans la section précédente, nous devons faire quelque calcule pour avoir une meilleur idée de la taille des composent requis.

Pour ce faire, nous devons dès maintenant préciser ce que nous voulons dire exactement par capacité supérieur ou égale au Super Nintendo. Premièrement, cette console possède les caractéristiques suivantes.

Table - Caractéristique du Super Nintendo

|  |  |
| --- | --- |
| Caractéristiques | Valeurs |
| Architecture | 16 bits |
| Fréquence | 3.58 Mhz (valeur effective max) |
| Résolution | 256x224 (plus utilisé) |
| Couleurs | 16 bits |
| Image par seconde | 10 (valeur moyenne) |
| Espace pour le jeu | 6 MB (max) |
| Nombre de couches d’affichage | 2-3 |

À partir de ces valeurs, nous avons fait une série de choix qui seraient des valeurs acceptables pour notre console.

Table - Caractéristique de la console du projet

|  |  |
| --- | --- |
| Caractéristiques | Valeurs |
| Architecture | 32 bits |
| Résolution | 480x272 |
| Couleurs | 24 bits |
| Image par seconde | 10 |
| Nombre de couches d’affichage | 2 |

Nous faisons ces choix pour plusieurs raisons. Premièrement, les architectures 32 bits sont très populaires de nos jours. Deuxièmement, les images de 24 bits par pixel sont un format très répandu et 10 images par seconde, en moyenne, est un taux de rafraichissement encore utilisé aujourd’hui et assez facile à atteindre. Nous nous limiterons à deux couches d’affichage, car la plupart des jeux n’utilisent que 2 couches. Pour finir, la résolution choisie est un bon compromis entre la haute résolution et la résolution du Super Nintendo. C’est cette résolution qui est normalement utilisée pour les écrans 4.3 pouces, très populaire chez les consoles portables.

Avec ces informations, nous pouvons estimer la puissance de calcul du processeur dont nous aurons besoin.



Le processeur devra donc être capable de faire en moyenne 2.6 millions de transferts par seconde pour l’affichage. Ceci peut être assez contraignant, mais il sera toujours possible de faire des choix matériels afin d’alléger la tâche comme avec l’utilisation de canaux DMA.

D’autre calcules sont nécessaires notamment au niveau de la taille de la mémoire requise. La mémoire contiendra le code du noyau ainsi que celui du moteur de jeux et du jeu. Elle contiendra aussi les images et le son. Compte tenue que le code ne devrait pas occuper plus de 200 KB, nous pouvons considérer l’espace utilisé par le code négligeable par rapport aux images et aux sons.

À ce stade nous ne savons pas quel type de fichier nous allons utiliser mais prenons le pire cas en considérant des formats non compressés.

Pour les images nous voulons 24 bits couleur, une résolution de 480x272 et deux couches d’affichage. Prenons tout de même 32 bits par couleur car il est plus facile de faire 1 transfert de 32 bits que 3 transferts de 8 bits ou 1 transfert de 16 bits et un de 8 bits.



Bien sûr ceci n’est qu’une approximation et en réalité nous utiliserons probablement plus. Pour plus de sureté, mettons nous une marge de 50%. Ceci nous donne 2 Méga-octets utilisé pour les images. Notez que nous n’avons pas calculé la taille de l’entête du fichier ici mais ceci est négligeable.

Pour rester dans l’affichage, nous aurons aussi besoins de mémoire pour le contrôleur LCD. Le calcules est pratiquement le même que le précédant. Cependant ici ce qui nous intéresse c’est le nombre de tampons d’affichage et non le nombre de couche. L’utilisation de 2 tampons est standard et fonctionnera très bien pour notre application.



Ici cette utilisation est fixe donc nous n’avons pas besoin de marge de sécurité. Tant que la résolution reste 480x272, la mémoire utilisé par le contrôleur LCD est exactement celle calculé.

Du coté de l’audio maintenant, considérons le standard Wave 16 bits par échantillons, 44.1 kilo-échantillons par secondes. Le Super Nintendo qui est notre console de référence utilisait 32 kHz et ceci sera probablement assez pour nos besoins mais prenons tout de même le pire cas. Commençons par traiter un fichier de 3 minutes qui serait parfait pour une musique d’arrière plan.



Comme nous le voyons, la musique non compressé occupe rapidement beaucoup d’espace. De plus, nous aurons aussi des effets sonores mais ceux-ci serait probablement d’une duré de quelque milliseconde à une seconde et donc beaucoup plus petit. Comme nous pouvons utiliser un type de fichier sonore de moins longue duré en augmentant la fréquence de répétition et/ou diminuer la qualité, nous avons tout de même beaucoup de façon de réduire la taille utilisé par l’audio. Nous pouvons donc considérer le 15 Mo comme le maximum sans utiliser de marge de sécurité.

À la lumière de ces calcules, la mémoire devrait être d’une taille de 17 Mo

## Définition des barèmes menant à la prise de décision

Du côté matériel, le choix du microcontrôleur est principalement basé sur sa rapidité, son coût et les périphériques disponibles sur celui-ci. Notons qu’ici nous avons préconisé le choix d’un microcontrôleur comme composant central au lieu d’un FPGA étant donné sa simplicité d’utilisation et de programmation. On voudra donc un microcontrôleur assez puissant pour afficher des images à une fréquence de 10Hz et une résolution de 480x272 pixels, ce qui demande le plus grand temps de traitement comparé aux autres opérations. Le microcontrôleur choisi devra donc idéalement inclure la gestion interne des signaux vidéo. On voudra aussi qu’il puisse gérer au minimum un périphérique USB afin d’utiliser différents type de contrôleurs de jeux communs et récents. Finalement, il devra pouvoir communiquer facilement avec une composante de mémoire externe afin d’y stocker les différentes images, les sons, le moteur de jeux et le code du jeu lui-même.

Pour cette mémoire externe, nous avons le choix entre le type DRAM et SRAM. La mémoire de type DRAM est souvent moins dispendieuse mais est volatile et doit donc être rafraîchie fréquemment, ce qui complexifie son utilisation. Par contre, les circuits récents incluent souvent le rafraîchissement automatique. Une mémoire de type SRAM est plus coûteuse mais possède une bande passante plus élevée étant donnée l’absence de condensateur, contrairement à la DRAM. La SRAM est aussi de plus grande dimension que la DRAM pour une même taille mémoire, étant donné le plus grand nombre de transistors utilisés. Ici, le coût étant un facteur important et étant donné la rapidité relativement élevée, pour notre projet, des mémoires DRAM synchrone (SDRAM), nous nous pencherons donc intuitivement vers le choix d’une mémoire de ce dernier type. N’ayant pas de connaissances poussées dans le domaine, et le temps limité alloué au projet, nous n’utiliserons que les principales caractéristiques, dont nous jugeons pour le moment nécessaire, afin de faire un choix le plus éclairé possible. Les trois principales caractéristiques à tenir en compte seront la bande passante, la taille et le temps de latence, qui est défini comme le temps d’accès à la mémoire en lecture et mesuré en nombre de coup d’horloge. Comme vu dans les calculs préliminaires, nous aurons besoin d’une mémoire d’une taille égale ou supérieure à 17MO avec un BUS de 32bits. Nous préconiserons un temps de latence de 2 coups d’horloges afin d’optimiser le temps de chargement des données.

Pour ce qui est de la sortie du signal vidéo de la console le choix se fait principalement entre les type de connections standards retrouvées sur les écrans modernes. Ici, on parle des connections de type S-Vidéo, VGA ou composite.

Le signal audio devrait idéalement

## Évaluation des solutions possibles

Premièrement, du côté du choix du microcontrôleur, nous avons plusieurs options disponibles. Ci-dessous, une liste de quelques microcontrôleurs qui ont été analysé suivant le barème fixé.

|  |  |
| --- | --- |
| **Composante** | **Caractéristiques principales** |
| LPC2478 |          Vieux cœur ARM7 |
|          Pas de division hardware |
|          72 Mhz |
|          Contrôleur LCD |
|          Contrôleur SDRAM |
|          Contrôleur USB host |
|          Boitier TQFP pouvant être soudé facilement |
| STM32F103 |          Nouveau cœur Cortex-M3 |
|          Division hardware |
|          72 Mhz |
|          15$ |
|          Pas d’USB host. Besoin d’un contrôleur USB externe (10$) |
|          Contrôleur SRAM seulement |
|          Pas de contrôleur LCD. Possibilité d’en ajouter un hardware externe sur le bus de mémoire externe. |
|          Nous possédons déjà du code fonctionnel |
|          Boitier TQFP pouvant être soudé facilement |
| LPC1768 |          Nouveau cœur Cortex-M3 |
|          100 Mhz |
|          10$ |
|          USB host |
|          Pas de mémoire externe |
|          Pas de contrôleur LCD |
|          Boitier TQFP pouvant être soudé facilement |
| AT91SAM9260 |          Vieux cœur ARM9¸ |
|          ~200 MHz |
|          15$ |
|          Possède de la cache |
|          Pas de contrôleur LCD |
|          Boitier PQFP pouvant être soudé facilement |
| OMAP3550 |          Microcontrôleur très performant, L1 et L2 cache, grand pipeline, division et opération point flottant matériel. |
|          600 MHz |
|          50$ |
|          Co-processeur graphique |
|          Affichage HDMI 720p |
|          Le beagleboard contient déjà la sortie audio/vidéo donc possiblement pas de développement matériel. |
|          Pas de code bas niveau ainsi que de fichier de définition pour les registres interne fournis par la compagnie |
|          Boitier BGA ne pouvant pas être soudé facilement |

Deuxièment, les choix disponibles pour la mémoire externe sont plutôt nombreux. Ci-dessous, une liste de quelques modèles que nous avons sélectionnés.

## Prise de décision en fonction des barèmes établis

<Texte>

# Architecture matériel

## <Titre>

<Texte>

# architecture logiciel

## <Titre>

<Texte>

# réalisation matériel

## <Titre>

<Texte>

# Réalisation logiciel

## <Titre>

<Texte>

# <titre>

## <Titre>

<Texte>

# <titre>

## <Titre>

<Texte>

# <titre>

## <Titre>

<Texte>

# <titre>

## <Titre>

<Texte>

CONCLUSION

<Texte interligne 1 1/2>

RECOMMANDATIONS <S’il y a lieu>

<Texte interligne 1 1/2>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

APPENDICES <S’il y a lieu>

<Texte>

LISTE DE RÉFÉRENCES

<Texte>

BIBLIOGRAPHIE

<Texte – Style Références bibliographiques déjà activé>