|  |
| --- |
| Eurêka ! |

Table des matières

[1 Analyse préliminaire 3](#_Toc102485436)

[1.1 Introduction 3](#_Toc102485437)

[1.2 Objectifs 3](#_Toc102485438)

[1.3 Planification initiale 5](#_Toc102485439)

[2 Analyse / Conception 6](#_Toc102485440)

[2.1 Concept 6](#_Toc102485441)

[2.1.1 Uses cases et scénarios 6](#_Toc102485442)

[2.1.2 Diagramme de flux 7](#_Toc102485443)

[2.2 Stratégie de test 11](#_Toc102485444)

[2.2.1 Test Unitaire 11](#_Toc102485445)

[2.2.2 Tests d’acceptation 11](#_Toc102485446)

[2.3 Risques techniques 11](#_Toc102485447)

[2.4 Planification 11](#_Toc102485448)

[2.5 Dossier de conception 11](#_Toc102485449)

[3 Réalisation 12](#_Toc102485450)

[3.1 Dossier de réalisation 12](#_Toc102485451)

[3.2 Description des tests effectués 12](#_Toc102485452)

[3.3 Erreurs restantes 12](#_Toc102485453)

[3.4 Liste des documents fournis 12](#_Toc102485454)

[4 Conclusions 13](#_Toc102485455)

[5 Annexes 14](#_Toc102485456)

[5.1 Résumé du rapport du TPI / version succincte de la documentation 14](#_Toc102485457)

[5.2 Sources – Bibliographie 14](#_Toc102485458)

[5.3 Journal de travail 14](#_Toc102485459)

[5.4 Manuel d'Installation 14](#_Toc102485460)

[5.5 Manuel d'Utilisation 14](#_Toc102485461)

[5.6 Archives du projet 14](#_Toc102485462)

*NOTE L’INTENTION DES UTILISATEURS DE CE CANEVAS:  
Toutes les parties en italiques sont là pour aider à comprendre ce qu’il faut mettre dans cette partie du document. Elles n’ont donc aucune raison d’être dans le document final.*

*De plus, en fonction du type de projet, il est tout à fait possible que certains chapitres ou paragraphes n’aient aucun sens. Dans ce cas il est recommandé de les retirer du document pour éviter de l’alourdir inutilement.*

# 

# Analyse préliminaire

## *Introduction*

*Ce chapitre décrit brièvement le projet, le cadre dans lequel il est réalisé, les raisons de ce choix et ce qu'il peut apporter à l'élève ou à l'école. Il n'est pas nécessaire de rentrer dans les détails (ceux-ci seront abordés plus loin) mais cela doit être aussi clair et complet que possible (idées de solutions). Ce chapitre contient également l'inventaire et la description des travaux qui auraient déjà été effectués pour ce projet.*

*Ces éléments peuvent être repris des spécifications de départ.*

Eurêka est une serious game de résolution de puzzle, créer avec le langage C# et le moteur graphique unity, il a pour but de proposer au joueur la possibilité de résoudre des puzzles sous la forme d’un schéma avec des portes logiques généré aléatoirement, le joueur devra deviner le résultat de tous le résultat afin de réussir le puzzle, plusieurs niveaux de difficulté s’offre à lui, il peut aussi créer son propre puzzle.  
Il a aussi la possibilité de prendre en photo son puzzle, afin de le sauvegarder sur le disque.

## Matériel à disposition

Liste de matériel physique et de logiciel mis à disposition

* 1 PC du CPNV
* Unity 2020
* Visual Studio 2020
* Visual Studio Code
* Suite office

## Prérequis

* Formation de base du CPNV
* Connaissances en POO
* Maitrise de Unity

## Objectifs

*Ce chapitre énumère les objectifs du projet. L'atteinte ou non de ceux-ci devra pouvoir être contrôlée à la fin du projet. Les objectifs pourront éventuellement être revus après l'analyse.*

*Ces éléments peuvent être repris des spécifications de départ.*

Les objectifs du projet étant :

* Résolution de puzzle
  + Le but étant que le joueur puisse résoudre le puzzle en devinant la sortie(résultat) du puzzle (Exemple : une porte AND avec en entrée 1 et 0, donne 0 comme sortie)
* Puzzle Aléatoire
  + Création Aléatoire des puzzles, chaque porte est générée aléatoirement, un nombre fixe de porte est présent, qui change en fonction du niveau de difficulté choisi par le joueur : en mode facile il y’a 3 portes logique, en mode moyen, il y’a 9 portes logique, en mode difficile, il y’a 19 portes logique.
* Puzzle manuel
  + Création manuel des puzzles, avec la possibilité de choisir les logiques portes dans le jeu, comme pour le puzzle aléatoire, les portes sont déjà prédéfinie par la difficulté, ils ne peuvent être déplacée, par contre le
  + joueur peut choisir quelle
  + type de porte (AND, XOR, etc…), et par la suite, de complété le puzzle par la suite, comme un puzzle aléatoire
* Capture de puzzle
  + Permet de prendre en photo le puzzle courant, qui est sauvegarder dans sur le disque
* Théorie des portes logiques
  + Affiche sur l’écran comment les portes logique courante sur le puzzle courant du comment fonctionne, leur entrée, sortie, avec une table de vérité

## 

## *Planification initiale*



# Analyse / Conception

## Concept

*Le concept complet avec toutes ses annexes:*

*Par exemple :*

* *Multimédia: carte de site, maquettes papier, story board préliminaire, …*
* *Bases de données: interfaces graphiques, modèle conceptuel.*
* *Programmation: interfaces graphiques, maquettes, analyse fonctionnelle…*
* *…*

### Uses cases et scénarios

Afin de comprendre comment, chaque fonctionnalité doit être programmé, et aussi d’avoir un scénario type pour gérer les exceptions j’ai fait des uses cases et scénario, afin de pouvoir bien se mettre d’accord sur ce qu’il faut faire

#### Puzzle aléatoire



#### Puzzle Manuel



#### Résolution de puzzle



#### Capture d’écran



#### Théories des portes logiques

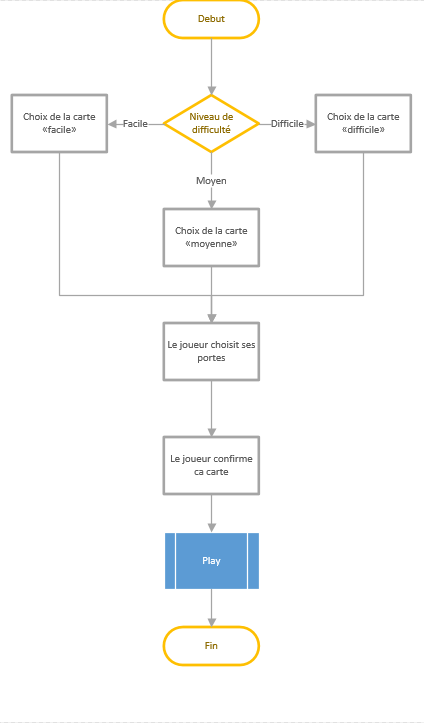
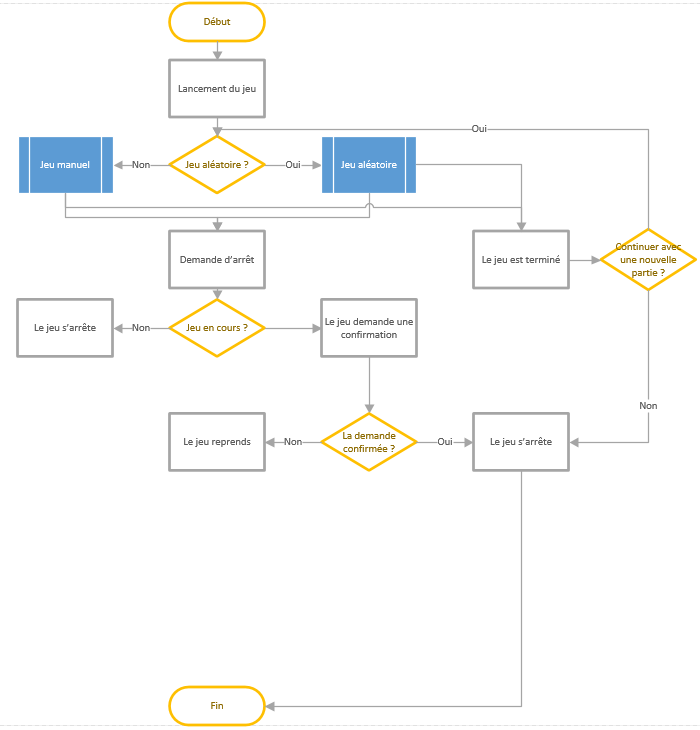


### Diagramme de flux

Pour avoir une idée plus précise du comment une fonctionnalité doit être implémentée, j’ai fait plusieurs diagrammes de flux, dont on peut voir comment les fonctions interagisse entre elles

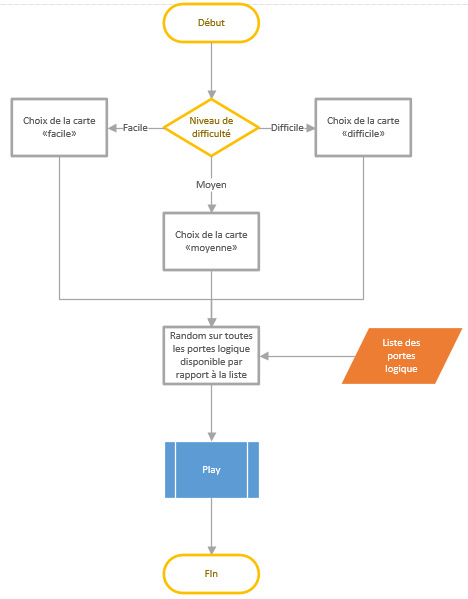
#### Main

Le code main, est le script maitre qui gère l’appelle des autres fonctions, et gère aussi la fermeture du programme



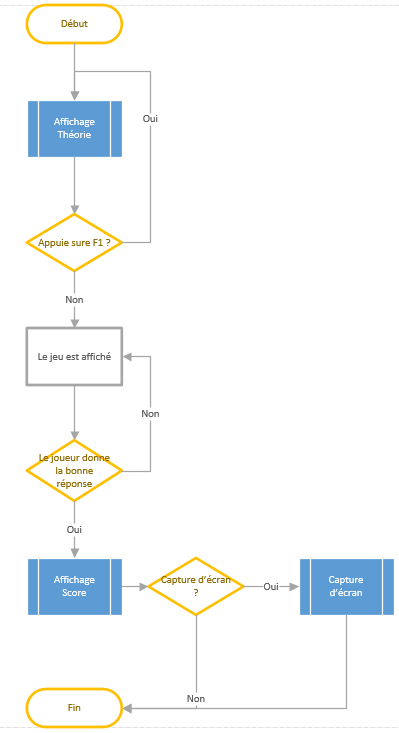
#### Jeu Manuel

#### Jeu aléatoire



#### Play

Play, est la fonction qui gère la résolution du puzzle, et si le joueur trouve la bonne sortie, le jeu est considéré comme terminé

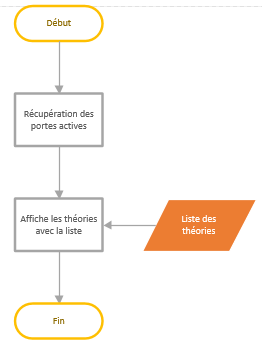


#### Capture d’écran

Permet la sauvegarde du puzzle courant par le biais d’une photo de ce dernier

#### Théorie des portes logique

Permet l’affichage de la théorie des port logiques courante



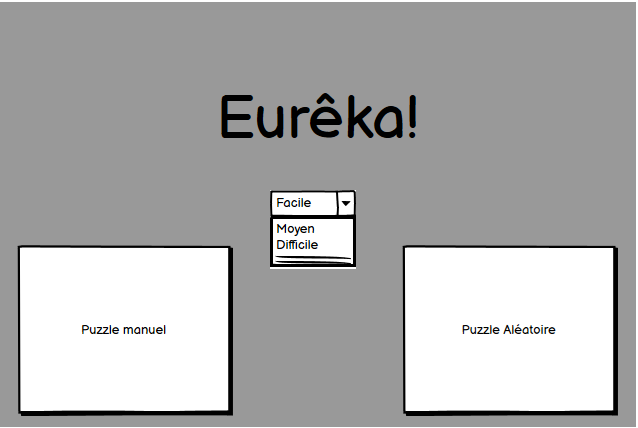
### Design de l’UI

Afin d’avoir une idée de comment l’UI du serious game, va ressembler, il est important d’avoir un design à quoi on peut s’attacher pour plus tard designer l’UI du jeu.

L’UI est pensé pour être accessible par PC et téléphone, les boutons sont grands, facile a voir.

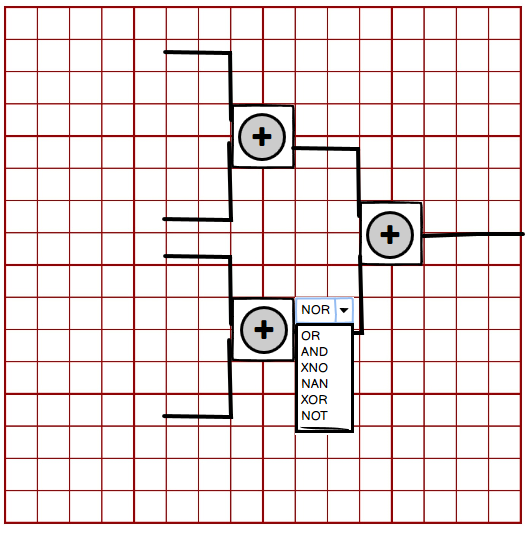
#### Menu principale/jeu suivant

Cette « page » peut être utilisée pour le menu principale, mais aussi comme sélection de mode de jeu pour le prochain puzzle



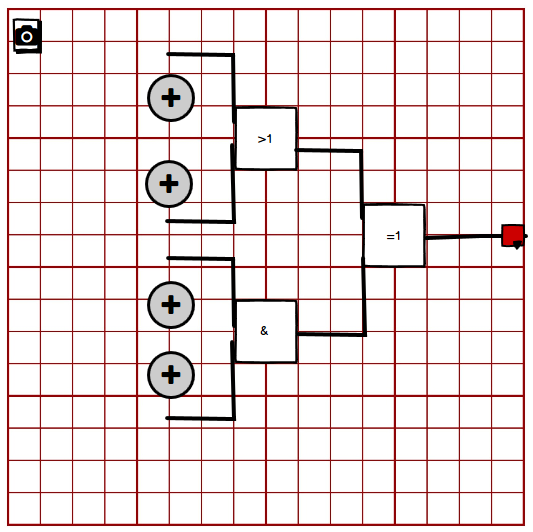
#### Menu de création de puzzle

Le joueur aura la possibilité de créer son propre puzzle, les portes sont déjà présents, et le nombre de porte ne change que si le niveau de difficulté change, dans ce cas, le niveau de difficulté est facile, il peut, en cliquant sur la porte, changer de type de porte.



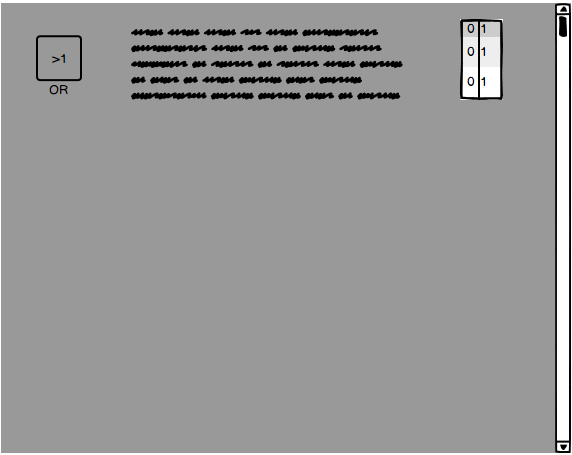
#### Menu de résolution de puzzle

Après avoir terminé la création manuel d’un puzzle ou d’avoir choisi un puzzle aléatoire, le joueur devras compléter le puzzle, si le joueur choisit la bonne réponse, la sortie passe au vert, et le joueur à gagner



#### Théorie des portes logiques

Une liste des portes logique, le symbole, en plusieurs normes, avec la description et table de vérité



## Stratégie de test

### Test Unitaire

Un test Unitaire seras effectué sur la fonctionnalité de la résolution de puzzle, afin de bien vérifier le bon fonctionnement de cette fonctionnalité, avec le module de tests proposer par visual studio.

### Tests d’acceptation

Plusieurs tests d’acceptation seront mis sur IceScrum, je demanderais à plusieurs camarades de classe et membres de ma famille pour tester le produit, et de vérifier que toutes les tests d’acceptation passent

## Risques techniques

* *risques techniques (complexité, manque de compétences, …).*

*Décrire aussi quelles solutions ont été appliquées pour réduire les risques (priorités, formation, actions, …).*

## Planification

*Révision de la planification initiale du projet :*

* *planning indiquant les dates de début et de fin du projet ainsi que le découpage connu des diverses phases.*
* *partage des tâches en cas de travail à plusieurs.*

*Il s’agit en principe de la planification* ***définitive du projet****. Elle peut être ensuite affinée (découpage des tâches). Si les délais doivent être ensuite modifiés, le responsable de projet doit être avisé, et les raisons doivent être expliquées dans l’historique.*

## *Dossier de conception*

*Fournir tous les document de conception:*

* *le choix du matériel HW*
* *le choix des systèmes d'exploitation pour la réalisation et l'utilisation*
* *le choix des outils logiciels pour la réalisation et l'utilisation*
* *site web: réaliser les maquettes avec un logiciel, décrire toutes les animations sur papier, définir les mots-clés, choisir une formule d'hébergement, définir la méthode de mise à jour, …*
* *bases de données: décrire le modèle relationnel, le contenu détaillé des tables (caractéristiques de chaque champs) et les requêtes.*
* *programmation et scripts: organigramme, architecture du programme, découpage modulaire, entrées-sorties des modules, pseudo-code / structogramme…*

***Le dossier de conception devrait permettre de sous-traiter la réalisation du projet !***

# Réalisation

## Dossier de réalisation

*Décrire la réalisation "physique" de votre projet*

* *les répertoires où le logiciel est installé*
* *la liste de tous les fichiers et une rapide description de leur contenu (des noms qui parlent !)*
* *les versions des systèmes d'exploitation et des outils logiciels*
* *la description exacte du matériel*
* *le numéro de version de votre produit !*
* *programmation et scripts: librairies externes, dictionnaire des données, reconstruction du logiciel - cible à partir des sources.*

*NOTE : Evitez d’inclure les listings des sources, à moins que vous ne désiriez en expliquer une partie vous paraissant importante. Dans ce cas n’incluez que cette partie…*

## *Description des tests effectués*

*Pour chaque partie testée de votre projet, il faut décrire:*

* *les conditions exactes de chaque test*
* *les preuves de test (papier ou fichier)*
* *tests sans preuve: fournir au moins une description*

## *Erreurs restantes*

*S'il reste encore des erreurs:*

* *Description détaillée*
* *Conséquences sur l'utilisation du produit*
* *Actions envisagées ou possibles*

## *Liste des documents fournis*

*Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions*

* *le rapport de projet*
* *le manuel d'Installation (en annexe)*
* *le manuel d'Utilisation avec des exemples graphiques (en annexe)*
* *autres…*

# Conclusions

*Développez en tous cas les points suivants:*

* *Objectifs atteints / non-atteints*
* *Points positifs / négatifs*
* *Difficultés particulières*
* *Suites possibles pour le projet (évolutions & améliorations)*

# Annexes

## *Résumé du rapport du TPI / version succincte de la documentation*

## *Sources – Bibliographie*

*Liste des livres utilisés (Titre, auteur, date), des sites Internet (URL) consultés, des articles (Revue, date, titre, auteur)… Et de toutes les aides externes (noms)*

## *Journal de travail*

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Durée** | **Activité** | **Remarques** |
|  |  |  |  |
|  |  |  |  |

## 

## *Manuel d'Installation*

## *Manuel d'Utilisation*

## *Archives du projet*

*Media, … dans une fourre en plastique*