EURÊKA !

Table des matières

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.

1 ANALYSE PRÉLIMINAIRE

1.1 INTRODUCTION

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.

1.2 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

1.3 PRÉREQUIS

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

1.4 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 :

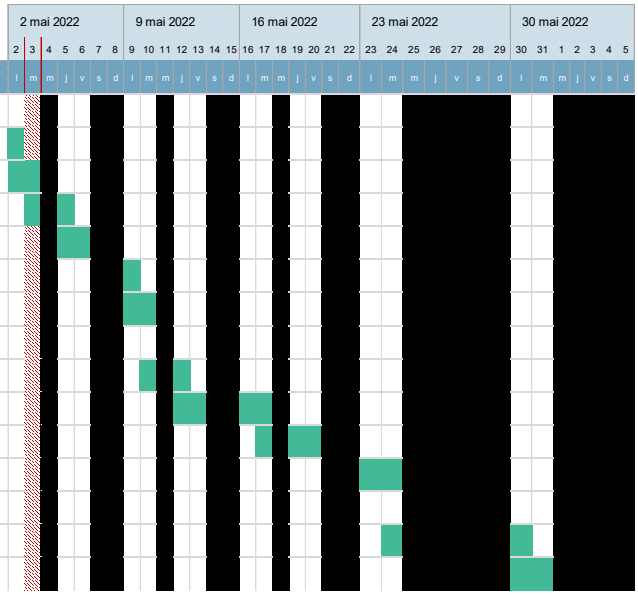
- Résolution de puzzle
  Le but étant que le joueur puisse résoudre le puzzle en devinant la sortie(résultat) du puzzle (Exemple : une porte AND avec en entrée 1 et 0, donne 0 comme sortie)
  Puzzle Aléatoire
  Création Aléatoire des puzzles, chaque porte est générée aléatoirement, un nombre fixe de 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

1.5 PLANIFICATION INITIALE

Eurêka!

lun, 5.2.2022

1



TÂCHE	DÉBUT	FIN
Planification - Analyse		
Planification Initatiale	2.5.22	2.5.22
Création de l'environment de travail	2.5.22	3.5.22
Conception des uses cases et scénario	3.5.22	5.5.22
Conception des diagrammes de flux	5.5.22	8.5.22
Conception de l'interface graphique	8.5.22	9.5.22
Création des sprints, tâches, tests sur IceScrum	9.5.22	10.5.22
Réalisation		
Réalisiation de l'interface graphique avec la théorie des portes	10.5.22	12.5.22
Réalisation du système de portes aléatoires	12.5.22	17.5.22
Réalisation du système de portes manuelle	17.5.22	22.5.22
Réalisation du système de capture du puzzle	22.5.22	24.5.22
Documentation et Tests		
Mise à jour du rapport de projet	24.5.22	30.5.22
Création des test unitaire, validation des tests d'acceptation	30.5.22	31.5.22

1.6 PLANIFICATION DÉTAILLÉE

Heure	lundi, 2 mai 2022	mardi, 3 mai 2022	mercredi, 4 mai 2022	jeudi, 5 mai 2022	vendredi, 6 mai 2022
08:00	Planification Initiale	Installation de l'environment de travail			Conception des diagrammes de flux
08:45					
08:50		Conception des uses cases scénario		Conception des uses cases scénario	
09:35					
09:50					
10:35					
10:40					
11:25					
11:30					
12:15					
13:30	Planification Initiale	Conception des uses cases scénario		Conception des diagrammes de flux	Conception des diagrammes de flux
14:15					
14:20					
15:05					
15:20					
16:05					
16:10	Installation de l'environment de travail				
16:55					

Heure	lundi, 9 mai 2022	mardi, 10 mai 2022	mercredi, 11 mai 2022	jeudi, 12 mai 2022	samedi, 0 janvier 1900
08:00	Conception de l'interface graphique	création des sprints, tâches, tests sur ice scrum			Réalisation du système de porte aléatoire
08:45					
08:50				Réalisation de l'interface graphique avec la théorie des portes	
09:35					
09:50					
10:35					
10:40					
11:25					
11:30					
12:15					
13:30	création des sprints, tâches, tests sur ice scrum	Réalisation de l'interface graphique avec la théorie des portes		Réalisation du système de porte aléatoire	Réalisation du système de porte aléatoire
14:15					
14:20					
15:05					
15:20					
16:05					
16:10					
16:55					

Heure	lundi, 16 mai 2022	mardi, 17 mai 2022	mercredi, 18 mai 2022	jeudi, 19 mai 2022	vendredi, 20 mai 2022
08:00	Réalisation du système de porte aléatoire	Réalisation du système de porte aléatoire			Réalisation du système de portes manuelle
08:45					
08:50				Réalisation du système de portes manuelle	
09:35					
09:50					
10:35					
10:40					
11:25					
11:30					
12:15					
13:30	Réalisation du système de porte aléatoire	Réalisation du système de portes manuelle		Réalisation du système de portes manuelle	Réalisation du système de portes manuelle
14:15					
14:20					
15:05					
15:20					
16:05					
16:10					
16:55					

Heure	lundi, 23 mai 2022	mardi, 24 mai 2022	mercredi, 25 mai 2022	jeudi, 26 mai 2022	vendredi, 27 mai 2022
08:00	Réalisation du système de capture de puzzle	Réalisation du système de capture de puzzle			
08:45					
08:50					
09:35					
09:50					
10:35					
10:40					
11:25					
11:30					
12:15					
13:30	Réalisation du système de capture de puzzle	Mise à jour du rapport de projet			
14:15					
14:20					
15:05					
15:20					
16:05					
16:10					
16:55					

Heure	lundi, 30 mai 2022	mardi, 31 mai 2022	mercredi, 1 juin 2022	jeudi, 2 juin 2022	vendredi, 3 juin 2022
08:00	Mise à jour du rapport de projet	Création des tests unitaire, validation des tests d'acceptation			
08:45					
08:50					
09:35					
09:50					
10:35					
10:40					
11:25					
11:30					
12:15					
13:30	Création des tests unitaire, validation des tests d'acceptation	Création des tests unitaire, validation des tests d'acceptation			
14:15					
14:20					
15:05					
15:20					
16:05					
16:10					
16:55					

2 ANALYSE / CONCEPTION

2.1 ANALYSE

2.1.1 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

2.1.1.1 PUZZLE ALÉATOIRE

Un puzzle aléatoire est généré lors du lancement d'une partie en tant que puzzle aléatoire.
Si le jeu est en échec, c'est-à-dire que le résultat attendu, n'est pas trouvé par le joueur, le jeu ne peut pas se terminer, ce use case et scénario ne gère pas la résolution de puzzle. Â l'inverse, si le résultat attendu, est trouvé par le joueur, le jeu est considéré comme trouvé, et un nouveau puzzle est généré, sauf si le joueur décide de quitter le jeu ou de passer sur un puzzle manuel.

Si le joueur décide de fermer le jeu quand un puzzle est en cours, le jeu demande une confirmation, si le joueur décide quand même de quitter le jeu, le puzzle ne seras pas sauvegarder, et un nouveau puzzle seras généré au prochain lancement du jeu, s'il n'y a pas de jeu en cours, le jeu se ferme sans donner de confirmation.

en tant que	Joueur	
Je veux	Avoir une puzzle aléatoire à chaque partie	
Pour	finir le puzzle	
Priorité	M	
Action	Condition	Réaction
Lancement du jeu	Le joueur choisi un puzzle aléatoire	Un puzzle aléatoire est généré
La partie est en échec		la partie ne peux pas se terminer
La partie est gagnée		la partie est terminée
La partie courante est terminée		Un puzzle aléatoire est généré
Fermeture du jeu	la partie est terminée	le jeu ce ferme
Fermeture du jeu	une partie est en cours	le jeu demande une comfirmation avant de ce fermer

1 cas d'utilisation : puzzle aléatoire

2.1.1.2 PUZZLE MANUEL

Un puzzle manuel est généré lors du lancement de la partie, les ports sont vides mais déjà présente, les liaisons sont aussi déjà présente.
Le joueur peux donc choisir le type de portes (porte AND, OR, par exemple).
Après avoir confirmé son choix, le puzzle passe en mode résolution de puzzle, le joueur peux donc compléter son propre puzzle.
Une partie peux être considéré comme en échec si le joueur ne choisit pas toutes les portes logique (laisse une vide par exemple) et le jeu peux être considéré comme terminé, si toutes les portes ont été choisies et le joueur aille confirmer son puzzle. Comme pour les puzzles aléatoires, le jeu demande une confirmation, si une partie est en cours, et ne demande pas si le jeu est déjà terminé.

Je veux	Avoir une puzzle manuel à chaque partie	
Pour	finir le puzzle	
Priorité	M	

Action	Condition	Réaction
Lancement du jeu	Le joueur choisi un Manuel	Un puzzle Manuel est généré
Le joueur choisit une porte de la liste		la porte est placé dans le jeu
Le joueur choisit une porte de la liste	il choisit la même porte	rien ne change
La partie est en échec		la partie ne peux pas se terminer
La partie courante est terminée		Un puzzle Manuel est généré
Fermeture du jeu	la partie est terminée	le jeu ce ferme
Fermeture du jeu	une partie est en cours	le jeu demande une comfirmation avant de ce fermer

2 cas d'utilisation : puzzle manuel

2.1.1.3 RÉSOLUTION DE PUZZLE

C'est la partie la plus importante du jeu, c'est ce qui gère toute la partie de résolution des puzzles, aléatoire et manuel, le résultat que le joueur doit trouver, et soit 1 ou 0 (true ou false), il faut donc vérifier si la valeur est dans cette fourchette de données ou que la valeur entrée est bien du décimal, si ce n'est pas le cas, le résultat est considéré comme faux, et un erreur est du coup affichée, si le joueur ne rentre aucun résultat rien ne se passe, si le joueur trouve le bon résultat, le jeu se termine.

Nom	RésolutionPuzzle-01	
en tant que	Joueur	
Je veux	Avoir une puzzle	
Pour	finir le puzzle	
Priorité	H	

Action	Condition	Réaction
Le joueur entre une résultat	le résultat est faux	une erreur est affichée
Le joueur entre une résultat	le résultat est juste	le jeu se termine
Le joueur n'entre pas résultat		rien ne se passe
Le joueur entre une résultat	la valeur est supérieur à un	le résultat est considéré comme faux
Le joueur entre une résultat	la valeur est inférieur à un	le résultat est considéré comme faux
Le joueur entre une résultat	la valeur n'est pas en décimal	le résultat est considéré comme faux

3 cas d'utilisation : résolution de puzzle

2.1.1.4 CAPTURE D'ÉCRAN

Le système de capture d'écran, permet la sauvegarde en photo du puzzle actuel, le jeu ne peux pas relire l'image pour charger une partie, cela est juste une photo.
Si la théorie des portes logique, ou le menu démarrée est affiché, le bouton pour prendre en capture le jeu, n'est ni actif, ni visible, à l'inverse on peut y accéder sans soucis.
Si le jeu ne peut pas avoir accès au fichier de sauvegarde d'image pour quelconques raisons, le jeu retourne une erreur.

Nom	CaptureD'écran-01	
en tant que	Joueur	
Je veux	pouvoir avoir en image le puzzle	
Pour	l'observer	
Priorité	S	

Action	Condition	Réaction
Le joueur prends en photo le puzzle		la photo est sauvergardée dans le disque
Le joueur prends en photo le puzzle	les résultat de la partie est affiché	le bouton n'est pas visible ni actif
Le joueur prends en photo le puzzle	le chemin d'accès est inaccessible	un message d'erreur est retourné au joueur

4 case d'utilisation : capture d'écran

2.1.1.5 THÉORIES DES PORTES LOGIQUES

A chaque lancement de partie, soit en lançant un puzzle aléatoire, ou après la création d'un puzzle manuel, le jeu affiche un jeu d'icône de portes logiques, et en appuyant sur une icône, un descriptif détaillé, leur icone, et une table de vérité sera possible d'afficher. Il est aussi possible de réafficher cette fenêtre en appuyant sur F1.

Nom	ThéoriePortLogiques-01	
en tant que	Joueur	
Je veux	Avoir afficher sur comment marche les portes logique	
Pour	connaître/être sur des fonctionnement des portes logique	
Priorité	S	
Action	Condition	Réaction
Lancement d'une partie		la fonctionnement sur les porte logique courante est affichée
en apuyant sur F1		la fonctionnement sur les porte logique courante est affichée

5 cas d'utilisation : Théorie des porte logique

2.2 CONCEPT

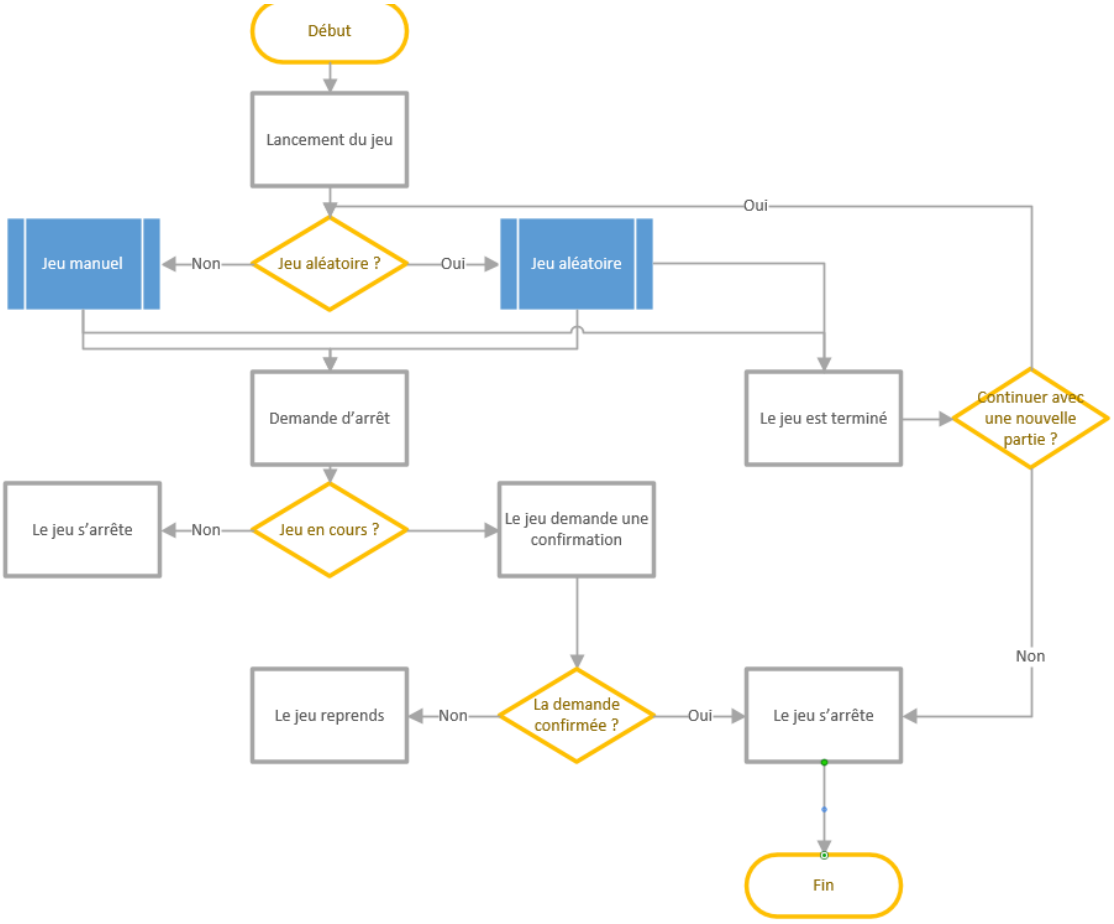
2.2.1 DIAGRAMME DE FLUX

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

2.2.1.1 MAIN

Le code main, est le script maitre qui gère l'appelle des autres fonctions, et gère aussi la fermeture du programme, le jeu commence toujours par demander si le joueur veut commencer par un puzzle manuel ou un puzzle aléatoire, et appelle la fonction qui gère le type de puzzle choisit. Le joueur peut choisir de continuer ou pas après chaque partie, si le joueur choisit de continuer, le jeu repose la question au joueur sur quel type de puzzle il veut, s'il ne veut pas, le jeu se ferme.

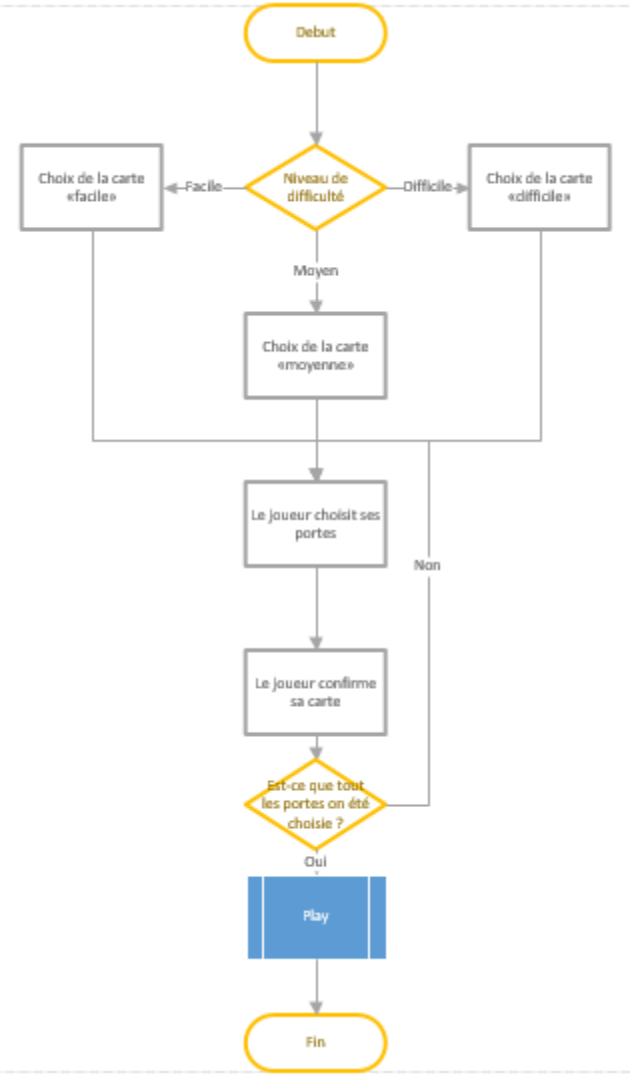
Il peut arriver que le joueur veut quitter le jeu, suivant le cas, si un puzzle est en cours, le jeu demande une confirmation au joueur. Au contraire, si aucun puzzle est en cours, le jeu se quitte sans demander de confirmations



6: Diagramme de flux global de l'application

2.2.1.2 JEU MANUEL

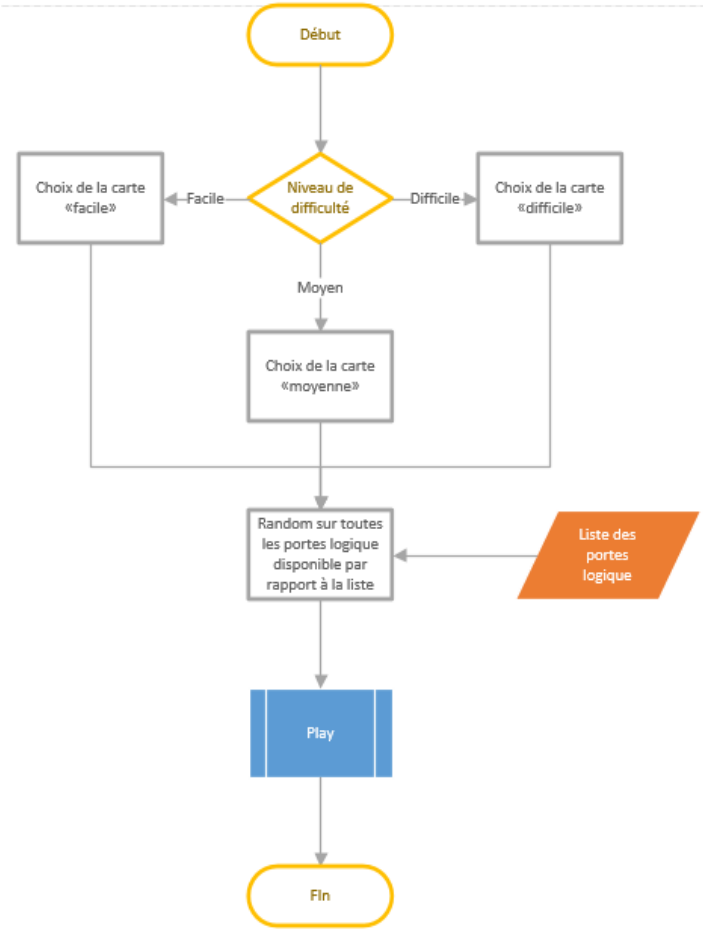
Si le joueur choisit un puzzle manuel, il doit aussi choisir la difficulté du jeu (un puzzle facile a 3 portes, moyen 9 portes et difficile 19 portes), les portes et fils sont déjà présent, le joueur peut juste choisir le type de portes par portes présente, si le joueur ne choisit pas tous ses portes, le jeu ne continue pas, à l'inverse s'il choisit toute ces portes, le jeu continue et appelle la fonction play.



7 Diagramme du puzzle manuel

2.2.1.3 JEU ALÉATOIRE

Comme pour le puzzle manuel, le puzzle aléatoire commence par un choix de niveau de difficulté, avec le même nombre de portes, le jeu après sélection de la difficulté prends toute les portes du puzzle, et la liste de porte disponible et fait un random sur toutes les portes, et ensuite le joueur peu enfin participer au puzzle

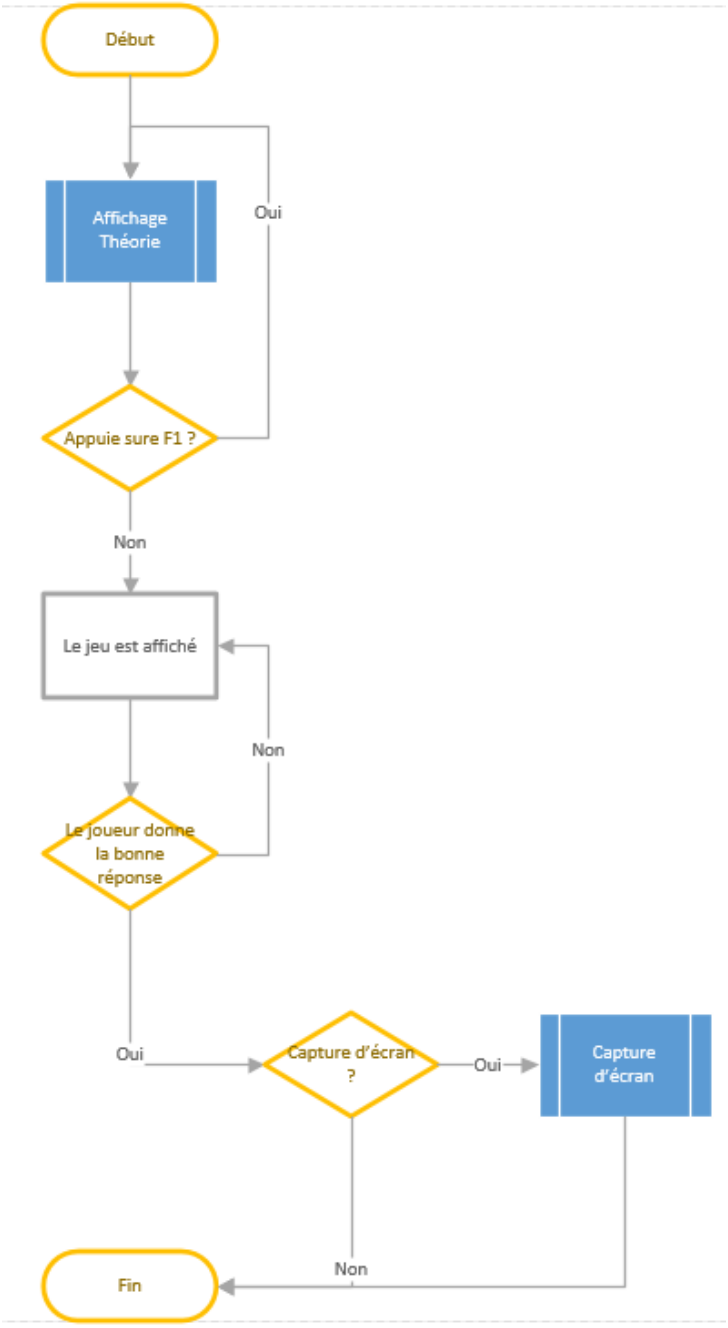


8 Diagramme du puzzle aléatoire

2.2.1.4 PLAY

Au début de chaque partie, la théorie sur chaque porte présente sur le puzzle courant est affiché, et peux être réafficher en appuyant sur F1.

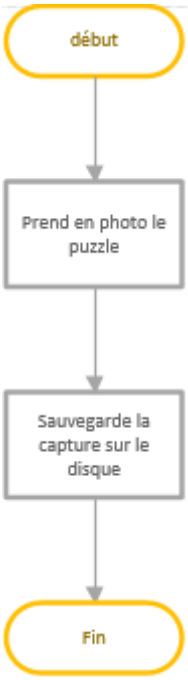
Après, cela, le jeu est affiché, et tant que le joueur n'a pas trouvé la bonne réponse, le joueur ne pourras pas passer à autre chose, s'il trouve la bonne réponse, il peut prendre en photo le puzzle, et le jeu est ensuite considéré comme terminé et retourne au main.



9 diagramme de la résolution de puzzle

2.2.1.5 CAPTURE D'ÉCRAN

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



11 Diagramme pour la capture d'écran

2.2.1.6 THÉORIE DES PORTES LOGIQUE

Permet l'affichage de la théorie des port logiques

10 diagramme pour la théorie des portes logique

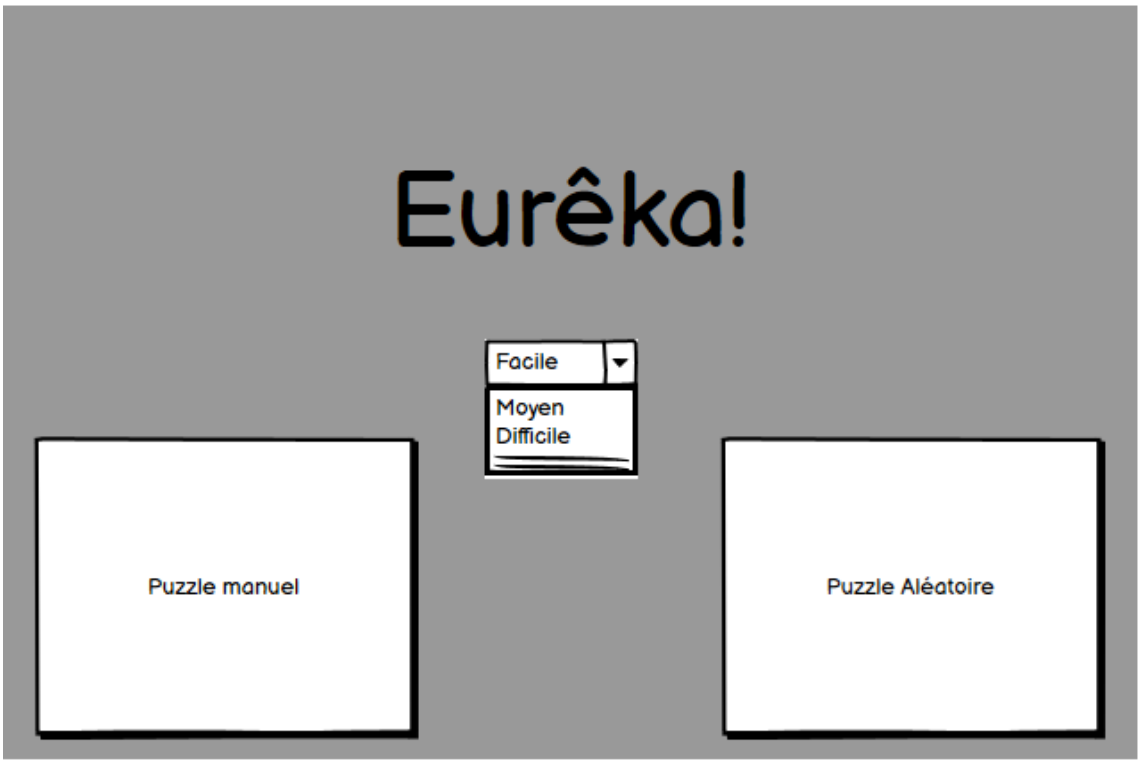
2.2.2 DESIGN DE L'UI

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.

2.2.2.1 MENU PRINCIPALE/JEU SUIVANT

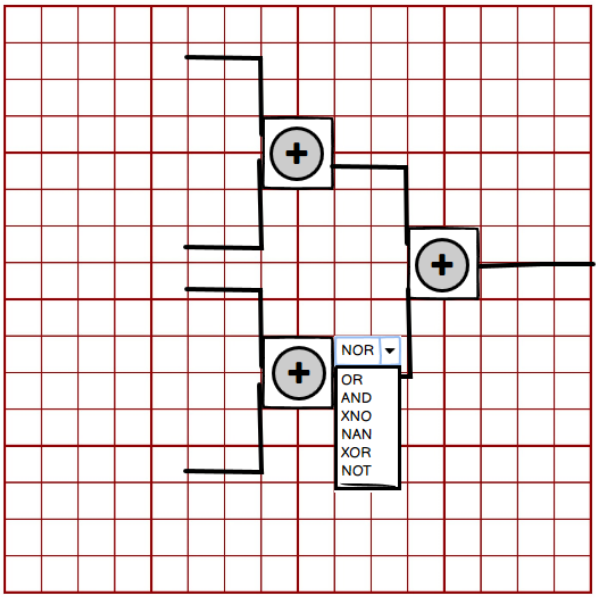
Cette « page » peut être utilisée pour le menu principale, mais aussi comme sélection de mode de jeu pour le prochain puzzle



12 maquette menu principale

2.2.2.2 MENU DE CRÉATION DE PUZZLE

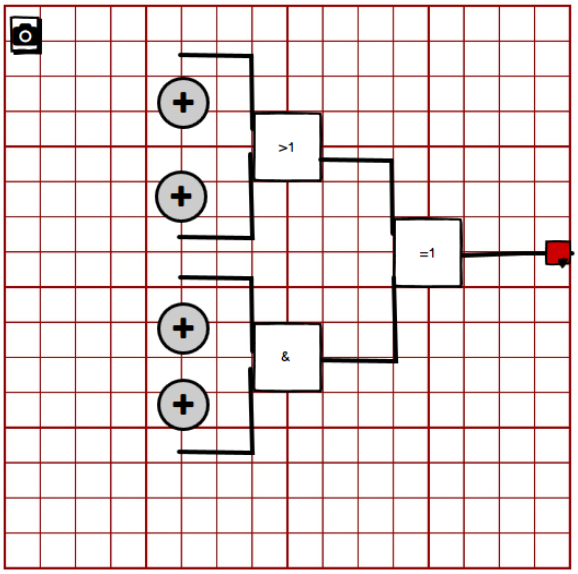
Le joueur aura la possibilité de créer son propre puzzle, les portes sont déjà présents, et le nombre de porte ne change que si le niveau de difficulté change, dans ce cas, le niveau de difficulté est facile, il peut, en cliquant sur la porte, changer de type de porte.



13 maquette : création de puzzle

2.2.2.3 MENU DE RÉSOLUTION DE PUZZLE

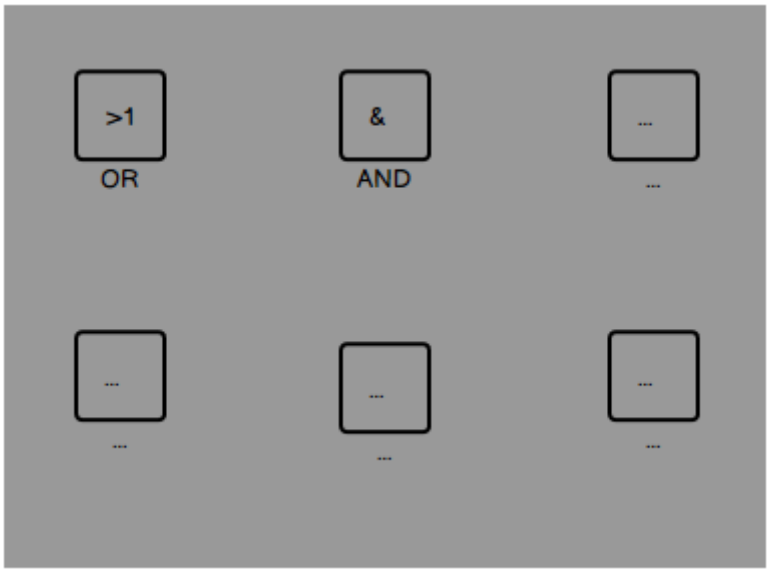
Après avoir terminé la création manuel d'un puzzle ou d'avoir choisi un puzzle aléatoire, le joueur devras compléter le puzzle, si le joueur choisit la bonne réponse, la sortie passe au vert, et le joueur à gagner



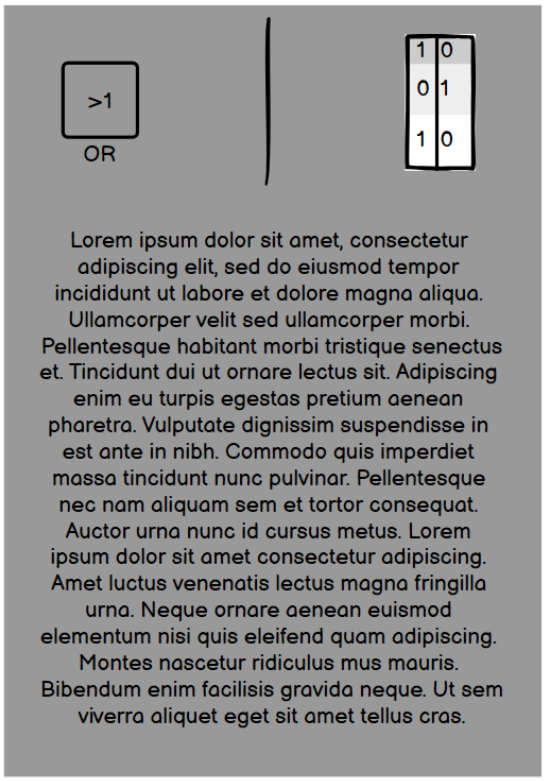
14 maquette : puzzle aléatoire

2.2.2.4 THÉORIE DES PORTES LOGIQUES

Un choix de portes logique seras affiché, en début de partie et en appuyant sur F1, en appuyant sur les différentes icônes de la première fenêtre, il est possible d'avoir un descriptif détaillé de la porte logique.



15 théorie : jeu d'icônes de porte logique



16 théorie détaillé des portes logique

2.3 STRATÉGIE DE TEST

2.3.1 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.

2.3.2 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

2.4 RISQUES TECHNIQUES

-
- risques techniques (complexité, manque de compétences, …).
-
- Décrire aussi quelles solutions ont été appliquées pour réduire les risques (priorités, formation, actions, …).
-

3 RÉALISATION

3.1 SCRIPTS

3.1.1 LOGICGATE

3.1.1.1 VARIABLES

public enum LogicGateType
public LogicGateType type ;

3.1.2 GATEMANAGER

3.1.2.1 VARIABLES

```
public bool output;
[SerializeField] public LogicGate logicGate;
[SerializeField] private Text text;
[SerializeField] private List<Sprite> GateSprites = new List<Sprite>();
[SerializeField] private List<Sprite> InversedGateSprites = new List<Sprite>();
[SerializeField] private List<Sprite> ColorState = new List<Sprite>();
[SerializeField] private Image ImageInversed;
[SerializeField] private Image CurrentState;
private bool input1;
private bool input2;
private Image Image;
```

3.1.2.2 FONCTIONS

private void Start()
public void ChangeInput1(bool value)
public void ChangeInput2(bool value)
private void changeOuput()

3.1.2.3 EVENTS

public ValueChangedEvent onValueChanged

3.1.3 CUSTOMGATEMANAGER

3.2 DOSSIER DE RÉALISATION

3.2.1 SCRIPTS

3.2.1.1 CLASSES

- LogicGate : une classe contenant des types de porte logique

3.2.1.2 GATES

- GateManager : le script gérant une porte logique, chaque porte logique à se script, il gère entrées et sorties
- CustomGateManager : le script gérant la création des portes logique dans la création de puzzle

3.2.1.3 PUZZLE

- CheckDifficulty : script qui prends en compte la difficulté choisit par le joueur, et choisis la bonne carte
- GlobalControl : script qui permet de garder en mémoire des variables, pour qu'elles ne soient pas perdue au changement de scène
- MapInfo : script qui donne les infos, de toutes les portes logique courante pour la difficulté choisie
- GeneratePuzzle : récupère toutes les portes logiques de la difficulté choisie et change toutes les port (sauf pour moyen et difficile ou la dernière porte est de tout de façon un AND), si le joueur a créer un puzzle auparavant, il génère le puzzle par rapport au choix du joueur
- ResolvePuzzle : Script qui gère la complétion du puzzle, si la dernière porte est en « Vert », le jeu retourne au menu principale, et il est possible de rechoisir un nouveau puzzle.
- CustomPuzzle : gère la confirmation du choix du joueur pour la création du puzzle, et sauvegarde le tout dans GlobalControl

3.2.1.4 UI

- MainMenu : script qui gère le menu démarré, notamment le changement de scène et la difficulté
- Zoom : script qui permet de zoomer sur le puzzle

3.2.2 DOSSIER

- Assets : répertoire principal, englobent tous les autres dossiers, fichier et script

3.2.2.1 RESSOURCES

- GameObjects : répertoire contenant diffèrent gameobjects utile pour le projet.
- Prefabs : répertoire contenant les préfabs utilisé pour le projet
- Sprites : répertoire contenant tous les sprites du jeu

3.2.2.2 SCENES

- Répertoire contenant toutes les scènes du jeu

3.2.3 SCRIPTS

- Classes : répertoire contenant toutes les classes utilisé dans le jeu, notamment LogicGate
- Gates : répertoire avec les scripts gérant les portes logiques
- Puzzle : répertoire avec les scripts gérant les puzzles
- UI : répertoire contenant les scripts qui gère les UI

3.2.4 VERSION DE WINDOWS

3.2.5 VERSION DE UNITY

- 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…

3.3 DESCRIPTION DES TESTS EFFECTUÉS

3.3.1 TESTS D'ACCEPTATIONS

3.3.1.1 RÉSOLUTION DU PUZZLE

- Les portes logiques fonctionnent correctement
  - Quand le joueur change l'entrée des portes, la sortie des portes logique réagisse correctement
- Les entrées changent quand je joueur veux

- Si le joueur change l'entrée, le changement est pris en compte et affecte les portes logiques
- Le système détecte quand le puzzle est terminé
  - Quand le puzzle est réussi, et que la dernière porte logique est en vert, le jeu se termine correctement

3.3.1.2 CRÉATION DU PUZZLE

- Le système change aléatoirement toutes les portes
  - Le système génère aléatoire les portes logiques, en prenant en compte les portes à 2 entrées et 1 entrée
- Le système génère aléatoirement à chaque fois
  - Le système génère aléatoirement, et ne régénère jamais le même puzzle

3.3.1.3 CRÉATION DU SYSTÈME DES THÉORIE

- Les boutons réagissent correctement
  - Les boutons affichent et caches les bonnes dalles

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

3.4 ERREURS RESTANTES

S'il reste encore des erreurs:

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

3.5 LISTE DES DOCUMENTS FOURNIS

- Diagrammes de flux
- Journal de travail
- Maquettes
- Uses case et scénario

4 CONCLUSIONS

- Développez en tous cas les points suivants:
-
- Objectifs atteints / non-atteints
- Points positifs / négatifs
- Difficultés particulières
- Suites possibles pour le projet (évolutions & améliorations)

5 ANNEXES

5.1 RÉSUMÉ DU RAPPORT DU TPI / VERSION SUCCINCTE DE LA DOCUMENTATION

5.2 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)

5.3 JOURNAL DE TRAVAIL

Date	Durée	Activité	Remarques

5.4

5.5 MANUEL D'INSTALLATION

5.6 MANUEL D'UTILISATION

5.7 ARCHIVES DU PROJET

Media, … dans une fourre en plastique