

EURÊKA !

1 TABLE DES MATIÈRES

1	Table des matières	2
2	Analyse préliminaire	3
2.1	Introduction	3
2.2	Matériel à disposition	3
2.3	Prérequis	3
2.4	Objectifs	3
2.5	Planification initiale.....	4
2.6	Planification détaillée.....	4
3	Analyse / Conception	6
3.1	Analyse	6
3.1.1	Uses cases et scénarios	6
3.2	Concept	8
3.2.1	Diagramme de flux	8
3.2.2	Design de l'UI	12
3.3	Stratégie de test	15
3.3.1	Test Unitaire	15
3.3.2	Tests d'acceptation	15
3.4	Risques techniques.....	16
3.4.1	Risque techniques	16
4	Réalisation	16
4.1	Dossier de réalisation	16
4.1.1	Scripts.....	16
4.1.2	Dossier.....	16
4.1.3	Scripts.....	17
4.1.4	Version de windows	17
4.1.5	Version de Unity.....	17
4.1.6	Version de visual studio	18
4.1.7	Version du programme	18
4.1.8	Matériel utilisé	18
4.1.9	Librairies	18
4.2	Description des tests effectués	20
4.2.1	Tests d'acceptations.....	20
4.2.2	Test unitaire	20
4.3	Erreurs restantes.....	21
4.4	Liste des documents fournis	21
5	Conclusions	22
6	Annexes.....	23
6.1	Résumé du rapport du TPI / version succincte de la documentation	23
6.2	Sources – Bibliographie	23
6.3	Journal de travail.....	23
6.4	23
6.5	Manuel d'Installation	23
6.6	Manuel d'Utilisation	23
6.7	Archives du projet	26
7	Glossaire.....	27

2 ANALYSE PRÉLIMINAIRE

2.1 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 générées 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.

2.2 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

2.3 PRÉREQUIS

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

2.4 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 compléter 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

2.5 PLANIFICATION INITIALE

Eurêka!

Eureka!

1

lun, 5.2.2022

2 mai 2022

9 mai 2022

16 mai 2022

23 mai 2022

30 mai 2022

23 mai 2022

30 mai 2022

TÂCHE

DEBUT

FIN

l m j v s d

l m j v s d

l m j v s d

l m j v s d

l m j v s d

Planification - Analyse

Planification Initiale

Création de l'environnement de travail

Conception des uses cases et scénario

Conception des diagrammes de flux

Conception de l'interface graphique

Création des sprints, tâches, tests sur IceScrum

Réalisation

Réalisation de l'interface graphique avec la théorie des portes

Réalisation du système de portes aléatoires

Réalisation du système de portes manuelle

Réalisation du système de capture du puzzle

Documentation et Tests

Mise à jour du rapport de projet

Création des test unitaire, validation des tests d'acceptation

2.5.22

2.5.22

2.5.22

3.5.22

5.5.22

8.5.22

9.5.22

10.5.22

10.5.22

12.5.22

17.5.22

22.5.22

24.5.22

24.5.22

30.5.22

31.5.22

2.6 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'environnement 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					
16:55	Installation de l'environnement de travail				

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						
09:35				Réalisation de l'interface graphique avec la théorie des portes		Réalisation du système de porte aléatoire
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	
14:15						
14:20						
15:05				Réalisation du système de porte aléatoire		
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								
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			
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	Création des tests unitaire, validation des tests d'acceptation	Création des tests unitaire, validation des tests d'acceptation						
13:30								
14:15								
14:20								
15:05								
15:20								
16:05								
16:10								
16:55								

3 ANALYSE / CONCEPTION

3.1 ANALYSE

3.1.1 USES CASES ET SCÉNARIOS

Afin de comprendre comment, chaque fonctionnalité doit être programmée, 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

3.1.1.1 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. À 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 sera pas sauvegardé, et un nouveau puzzle sera 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 un puzzle aléatoire à chaque partie	
Pour	finir le puzzle	
Priorité	M	
Action	Condition	Réaction
Lancement du jeu	Le joueur choisit un puzzle aléatoire	Un puzzle aléatoire est généré
La partie est en échec		la partie ne peut 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 se ferme
Fermeture du jeu	une partie est en cours	le jeu demande une confirmation avant de se fermer

1 cas d'utilisation : puzzle aléatoire

3.1.1.2 PUZZLE MANUEL

Un puzzle manuel est généré lors du lancement de la partie, les ports sont vides mais déjà présents, les liaisons sont aussi déjà présentes.

Le joueur peut 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 peut donc compléter son propre puzzle.

Une partie peut être considérée comme en échec si le joueur ne choisit pas toutes les portes logiques (laisse une vide par exemple) et le jeu peut 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 confirmation avant de ce fermer

2 cas d'utilisation : puzzle manuel

3.1.1.3 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

3.1.1.4 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 sauvegardée 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

3.1.1.5 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 appuyant sur F1		la fonctionnement sur les porte logique courante est affichée

5 cas d'utilisation : Théorie des porte logique

3.2 CONCEPT

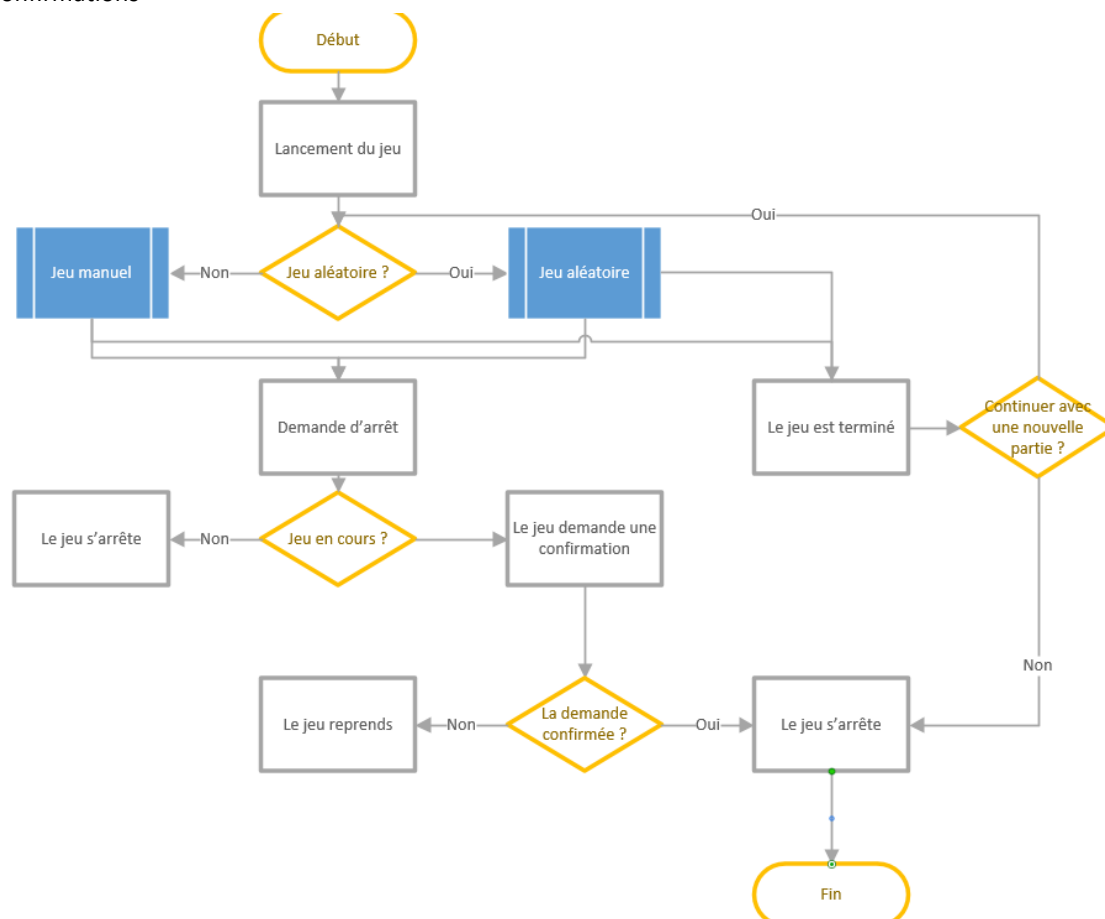
3.2.1 DIAGRAMME DE FLUX

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 interagissent entre elles

3.2.1.1 MAIN

Le code main, est le script maître 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 choisi. 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.

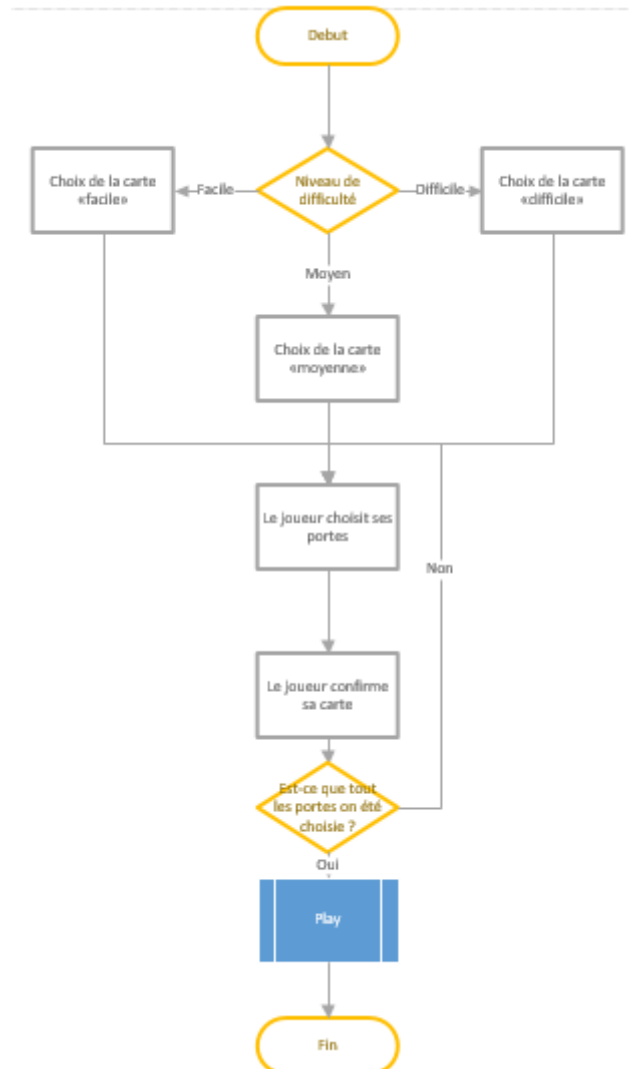
Il peut arriver que le joueur veut quitter le jeu, suivant le cas, si un puzzle est en cours, le jeu demande une confirmation au joueur. Au contraire, si aucun puzzle est en cours, le jeu se quitte sans demander de confirmations



6: Diagramme de flux global de l'application

3.2.1.2 JEU MANUEL

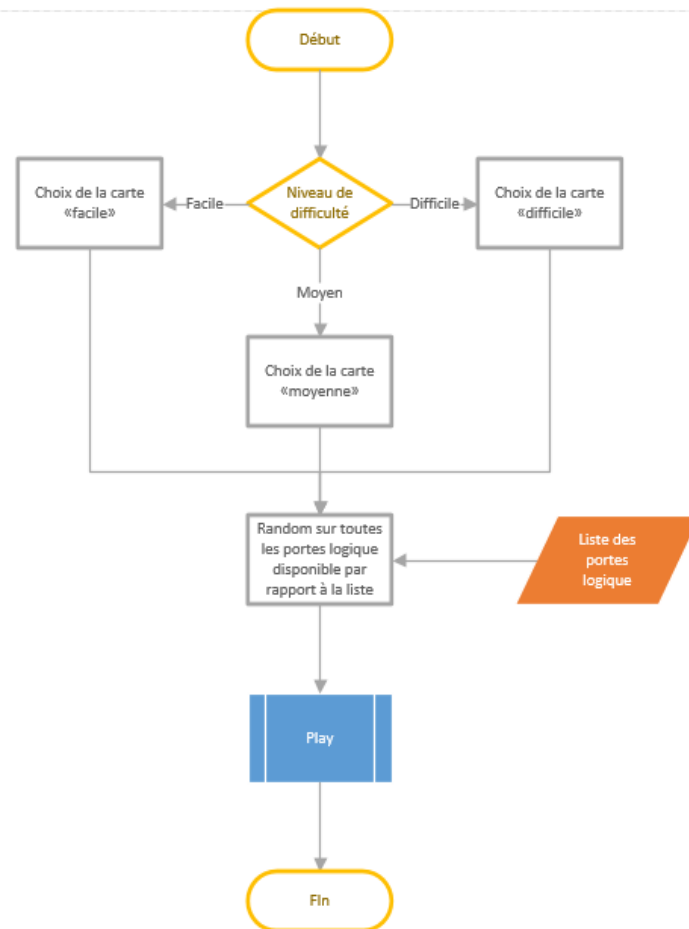
Si le joueur choisit un puzzle manuel, il doit aussi choisir la difficulté du jeu (un puzzle facile a 3 portes, moyen 9 portes et difficile 19 portes), les portes et fils sont déjà présent, le joueur peut juste choisir le type de portes par portes présente, si le joueur ne choisit pas tous ses portes, 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

3.2.1.3 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 prend toutes les portes du puzzle, et la liste de porte disponible et fait un random sur toutes les portes, et ensuite le joueur peut enfin participer au puzzle

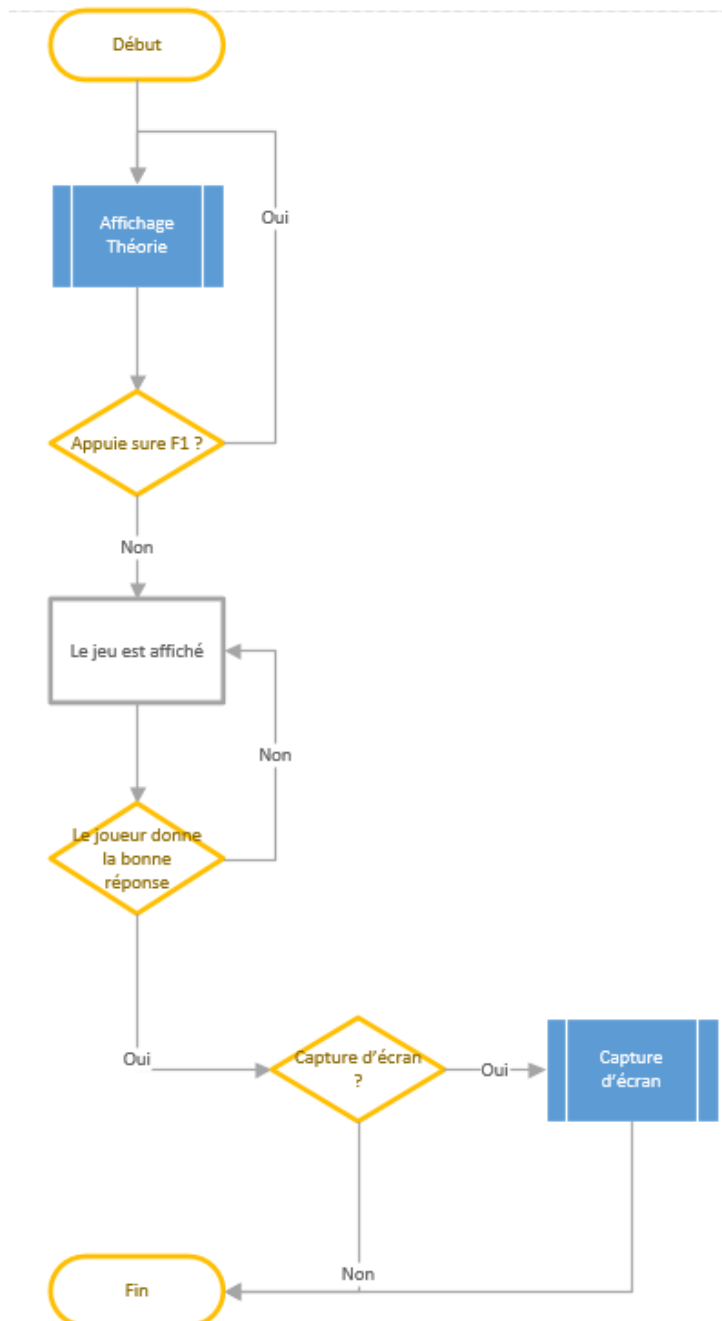


8 Diagramme du puzzle aléatoire

3.2.1.4 PLAY

Au début de chaque partie, la théorie sur chaque porte présente sur le puzzle courant est affichée, et peut être réaffichée en appuyant sur F1.

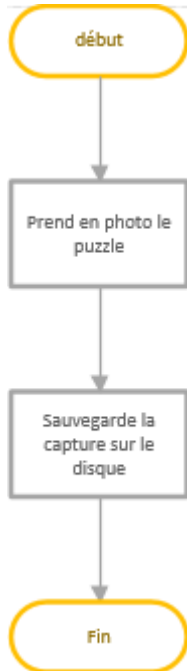
Après, cela, le jeu est affiché, et tant que le joueur n'a pas trouvé la bonne réponse, le joueur ne pourra pas passer à autre chose, s'il trouve la bonne réponse, il peut prendre en photo le puzzle, et le jeu est ensuite considéré comme terminé et retourne au main.



9 diagramme de la résolution de puzzle

3.2.1.5 CAPTURE D'ÉCRAN

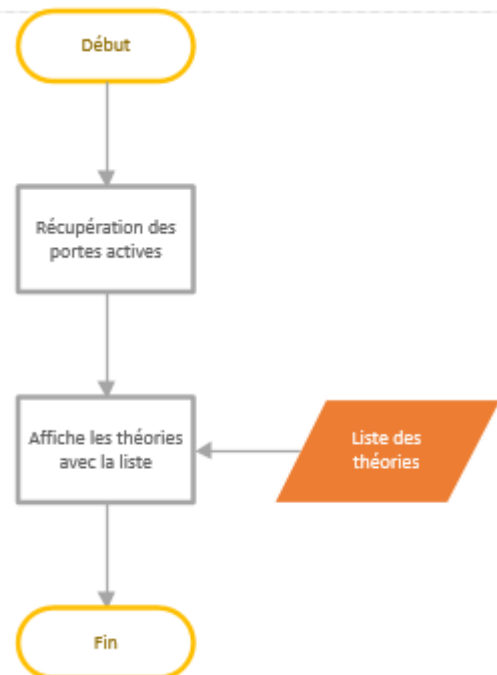
Permet la sauvegarde du puzzle courant par le biais d'une photo de ce dernier



11 Diagramme pour la capture d'écran

3.2.1.6 THÉORIE DES PORTES LOGIQUES

Permet l'affichage de la théorie des port logiques



10 diagramme pour la théorie des portes logique

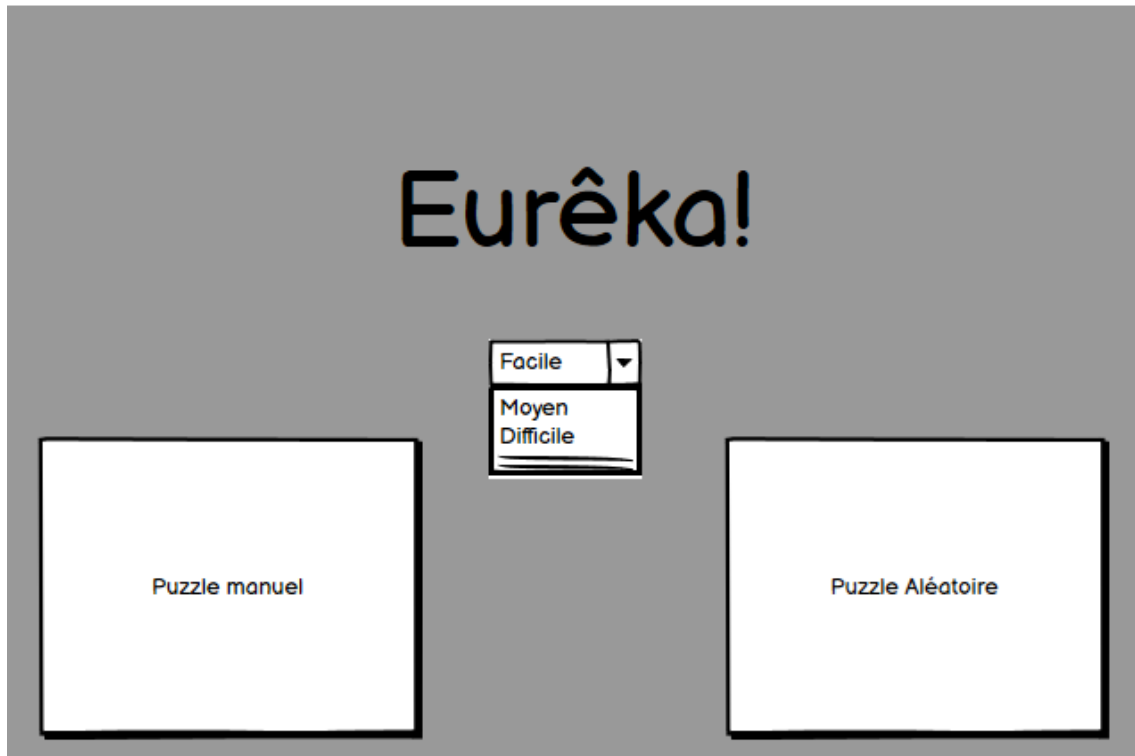
3.2.2 DESIGN DE L'UI

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 à voir.

3.2.2.1 MENU PRINCIPALE/JEU SUIVANT

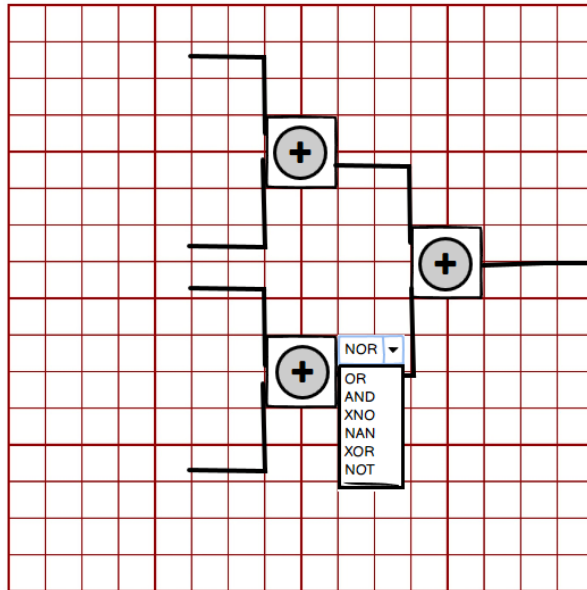
Cette « page » peut être utilisée pour le menu principale, mais aussi comme sélection de mode de jeu pour le prochain puzzle



12 maquette menu principale

3.2.2.2 MENU DE CRÉATION DE PUZZLE

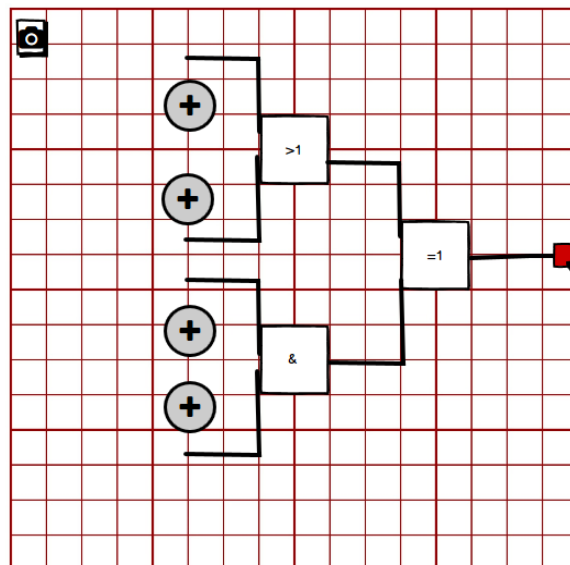
Le joueur aura la possibilité de créer son propre puzzle, les portes sont déjà présents, et le nombre de porte ne change que si le niveau de difficulté change, dans ce cas, le niveau de difficulté est facile, il peut, en cliquant sur la porte, changer de type de porte.



13 maquette : création de puzzle

3.2.2.3 MENU DE RÉOLUTION DE PUZZLE

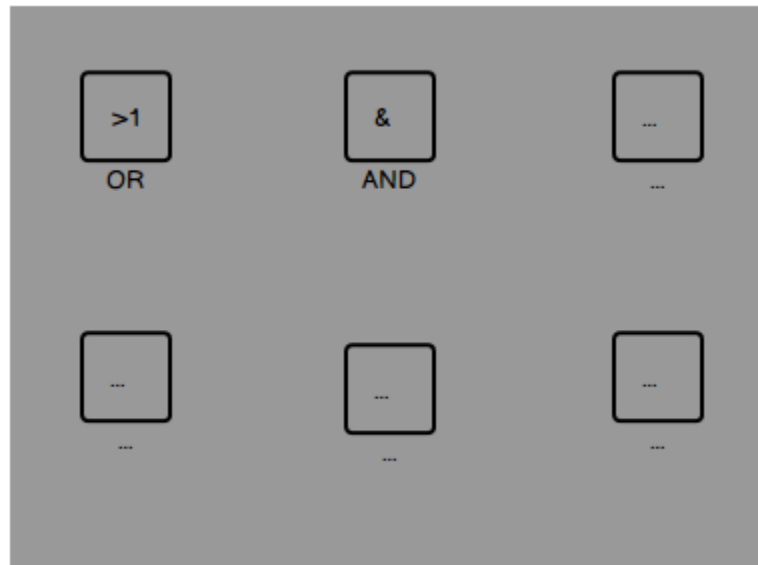
Après avoir terminé la création manuel d'un puzzle ou d'avoir choisi un puzzle aléatoire, le joueur devra compléter le puzzle, si le joueur choisit la bonne réponse, la sortie passe au vert, et le joueur à gagner



14 maquette : puzzle aléatoire

3.2.2.4 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

OR

1	0
0	1
1	0

Lorem ipsum dolor sit amet, consectetur
 adipiscing elit, sed do eiusmod tempor
 incididunt ut labore et dolore magna aliqua.
 Ullamcorper velit sed ullamcorper morbi.
 Pellentesque habitant morbi tristique senectus
 et. Tincidunt dui ut ornare lectus sit. Adipiscing
 enim eu turpis egestas pretium aenean
 pharetra. Vulputate dignissim suspendisse in
 est ante in nibh. Commodó quis imperdiet
 massa tincidunt nunc pulvinar. Pellentesque
 nec nam aliquam sem et tortor consequat.
 Auctor urna nunc id cursus metus. Lorem
 ipsum dolor sit amet consectetur adipiscing.
 Amet luctus venenatis lectus magna fringilla
 urna. Neque ornare aenean euismod
 elementum nisi quis eleifend quam adipiscing.
 Montes nascetur ridiculus mus mauris.
 Bibendum enim facilisis gravida neque. Ut sem
 viverra aliquet eget sit amet tellus cras.

16 théorie détaillé des portes logique

3.3 STRATÉGIE DE TEST

3.3.1 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.

3.3.2 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

3.4 RISQUES TECHNIQUES

3.4.1 RISQUE TECHNIQUES

Quelque chose dont je n'ai pas l'habitude de faire, du au faite que j'ai très rarement pratiqué, étant les tests unitaires en C#, est j'en n'ai jamais fait avec tout l'infrastructure d'Unity

3.4.1.1 DOCUMENTATION

Un risque qui reste majeur pour moi, étant savoir comment documenter toute la partie implémentation mon projet de la bonne manière

4 RÉALISATION

4.1 DOSSIER DE RÉALISATION

4.1.1 SCRIPTS

4.1.1.1 CLASSES

- LogicGate : une classe contenant des types de porte logique

4.1.1.2 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

4.1.1.3 PUZZLE

- CheckDifficulty : script qui prends en compte la difficulté choisit par le joueur, et choisit 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

4.1.1.4 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

4.1.2 DOSSIER

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

4.1.2.1 RESSOURCES

- GameObjects : répertoire contenant différent gameobjects utile pour le projet.
- Prefabs : répertoire contenant les préfabs utilisé pour le projet

- Sprites : répertoire contenant tous les sprites du jeu

4.1.2.2 SCENES

- Répertoire contenant toutes les scènes du jeu

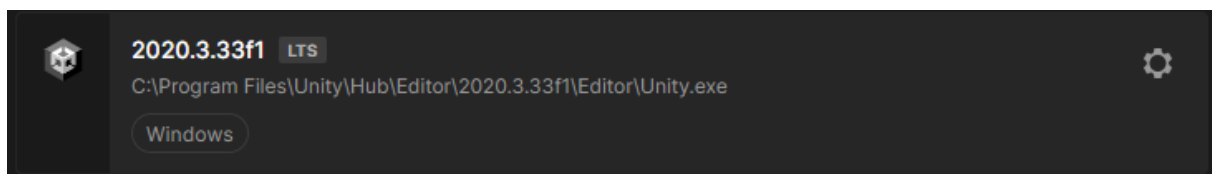
4.1.3 SCRIPTS

- 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

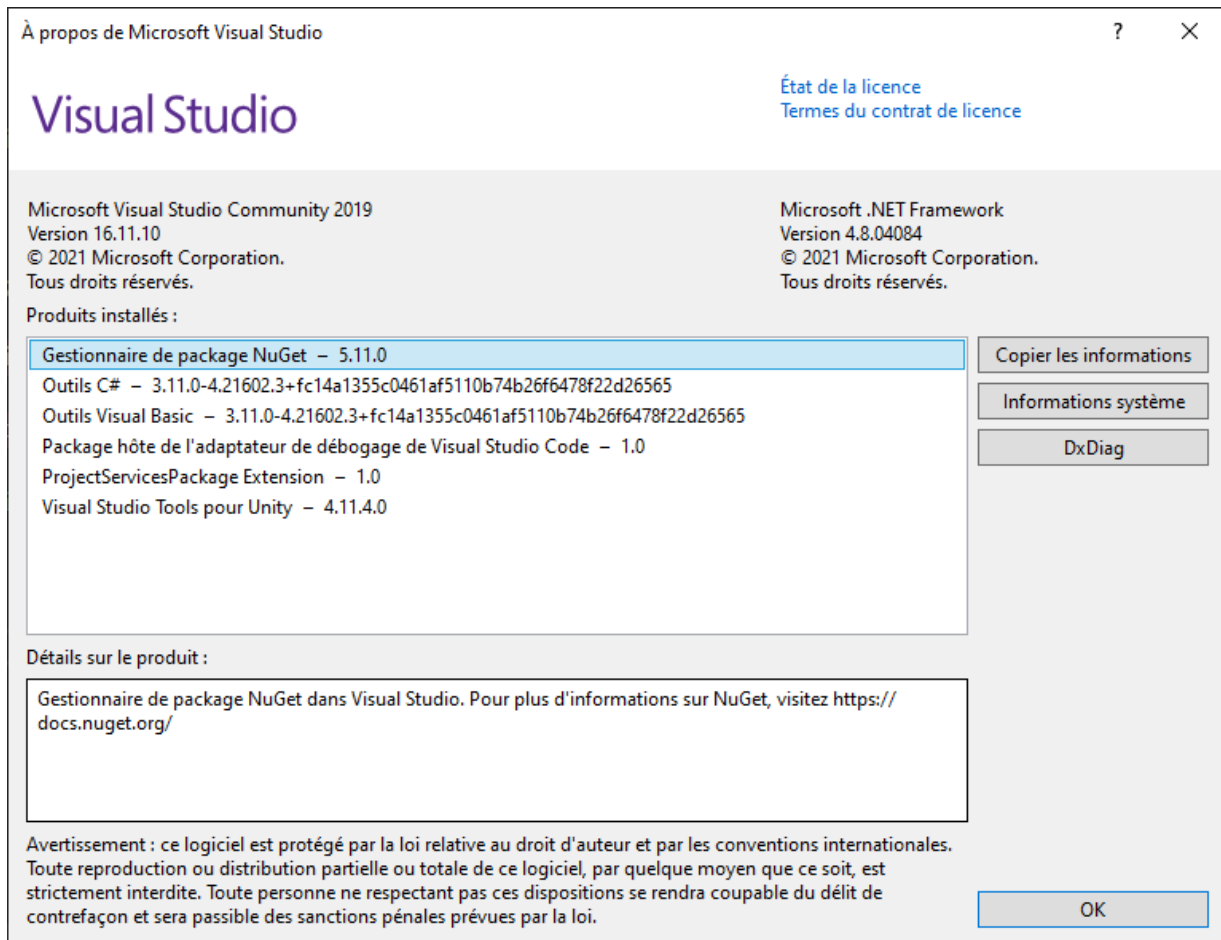
4.1.4 VERSION DE WINDOWS



4.1.5 VERSION DE UNITY



4.1.6 VERSION DE VISUAL STUDIO



4.1.7 VERSION DU PROGRAMME

2.00

4.1.8 MATÉRIEL UTILISÉ

4.1.8.1 PROCESSEUR

Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz 3.41 GHz

4.1.8.2 RAM

16.0 Go

4.1.8.3 OS

Windows 10 Éducation 21H2 x64

4.1.9 LIBRAIRIES

4.1.9.1 DANS LE PROJET

- 2D pixel perfect
- 2D Sprite

- JetBrains Rider Editor
- Test Framework
- TextMeshPro
- Timeline
- Unity UI
- Version Control
- Visual Studio Code Editor
- Visual Studio Editor

4.1.9.2 LIBRAIRIES BUIL-IN

- AI
- Android JNI
- Animation
- Asset Bundle
- Audio
- Cloth
- Director
- Image Conversion
- IMGUI
- JSONSerialize
- Particle System
- Physics
- Physics 2D
- Screen Capture
- Subsystems
- Terrain
- Terrain Physics
- Tilemap
- UI
- UIElements
- UIElements Native
- Umbra
- Unity Analytics
- Unity Web Request
- Unity Web Request Asset Bundle
- Unity Web Request Audio
- Unity Web Request Texture
- Unity Web Request WWW
- Vehicles
- Video
- VR
- Wind
- XR
- 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...

4.2 DESCRIPTION DES TESTS EFFECTUÉS

4.2.1 TESTS D'ACCEPTATIONS

4.2.1.1 RÉOLUTION 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 le joueur veut
 - 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

4.2.1.2 CRÉATION DU PUZZLE ALÉATOIRE

- Le système change aléatoirement toutes les portes
 - Le système génère aléatoirement 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

4.2.1.3 CRÉATION DU PUZZLE CUSTOMISÉ

- Quand le joueur choisit sa porte logique la porte change
 - Le visuel change, et le système prend en compte
- Quand le joueur confirme son choix le système prend en compte le choix
 - Quand le joueur a terminé de créer son puzzle, et confirme son choix, le système passe en mode création, avec les portes logique à la bonne place, qui fonctionne correctement

4.2.1.4 CRÉATION DU SYSTÈME DES THÉORIE

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

4.2.1.5 SYSTÈME DE CAPTURE D'ÉCRAN

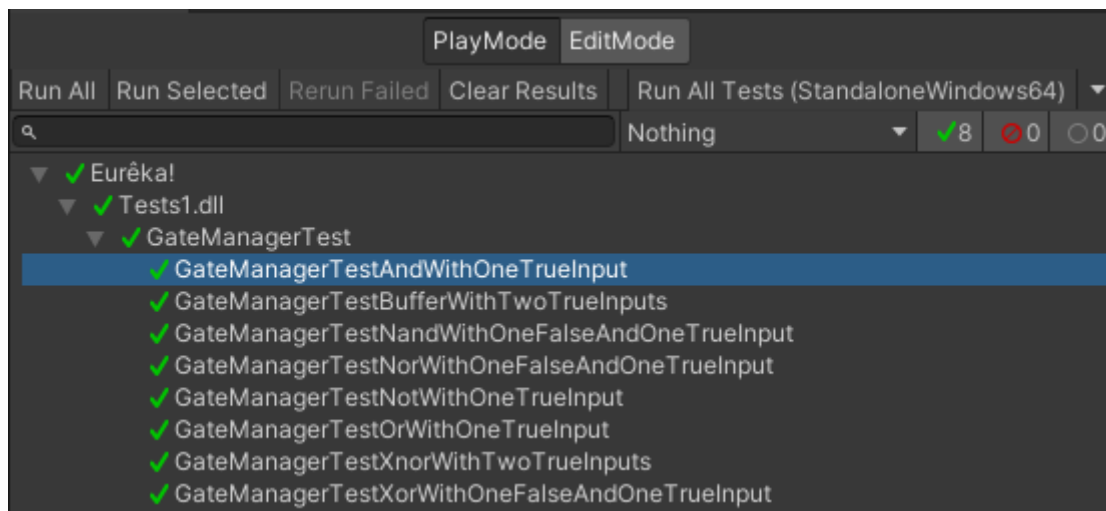
- Quand le bouton est pressé une photo est prise
 - Une photo est prise au moment du pressage du bouton, et le l'image est bien capturée
- Si pas de dossier existe, un dossier est créé
 - Si le joueur prend une photo et le dossier de photo n'est pas présent, le dossier est créé

4.2.2 TEST UNITAIRE

Le test unitaire se porte sur le script de la gestion de porte logique, c'est-à-dire, le script, qui gère les entrées, et sortie des portes, le test est réalisé en C# sur Visual Studio.

4.2.2.1 GATEMANAGER TEST AND WITH ONE TRUE INPUTS

Test sur une porte AND avec une entrée TRUE, et qui retourne un FALSE, les entrées qui ne sont pas spécifiées, sont de base false



4.2.2.2 GATEMANAGER TEST BUFFER WITH TWO TRUE INPUTS

Test sur une porte Buffer, avec deux entrées TRUE, et qui retourne un TRUE, les portes logiques avec une seule entrée, ne prennent en considération seulement que la première entrée

4.2.2.3 GATEMANAGER TEST NAND WITH ONE FALSE AND ONE TRUE INPUT

Test sur une porte NAND, avec une entrée TRUE et une entrée FALSE, et qui retourne un FALSE

4.2.2.4 GATEMANAGER TEST NOR WITH ONE FALSE INPUT AND ONE TRUE INPUT

Test sur une porte NOR, avec une entrée FALSE et une entrée TRUE, qui retourne un FALSE

4.2.2.5 GATEMANAGER TEST NOT WITH ONE TRUE INPUT

Test sur une porte NOT, avec une entrée TRUE, qui retourne un false

4.2.2.6 GATEMANAGER TEST OR WITH ONE TRUE INPUT

Test sur une porte OR, avec une entrée TRUE, qui retourne un TRUE

4.2.2.7 GATEMANAGER TEST XNOR WITH TWO TRUE INPUTS

Test sur une porte logique XNOR, avec deux entrée TRUE, qui retourne un TRUE

4.2.2.8 GATEMANAGER TEST XOR WITH ONE FASLE AND ONE TRUE INPUT

Test sur une porte logique XOR, avec une entrée TRUE, et une entrée FALSE, qui retourne un TRUE

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

4.3 ERREURS RESTANTES

S'il reste encore des erreurs:

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

4.4 LISTE DES DOCUMENTS FOURNIS

- Diagrammes de flux

- Journal de travail
- Maquettes
- Uses case et scénario
- Résumé du projet

5 CONCLUSIONS

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

6 ANNEXES

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

6.2 SOURCES – BIBLIOGRAPHIE

- tutorialspoint. «Logic gates.» *tutorialspoint.* 16 05 2022.
https://www.tutorialspoint.com/computer_logical_organization/logic_gates.htm.
- Unity. «Screenshot, but from a certain camera.» *Unity Forum.* 16 05 2022.
<https://forum.unity.com/threads/screenshot-but-from-a-certain-camera.69645/#:~:text=%20%20%201%20Disable%20all%20activated%20cameras,is%20done%2C%20you%20can%20switch%20cameras...%20More%20>.
- . «Unity - Application.persistentDataPath.» *Unity documentation.* 16 05 2022.
<https://docs.unity3d.com/ScriptReference/Application-persistentDataPath.html>.
- . «Unity - ScreenCapture.CaptureScreenshot.» *Unity Documentation.* 16 05 2022.
<https://docs.unity3d.com/ScriptReference/ScreenCapture.CaptureScreenshot.html>.

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)

6.3 JOURNAL DE TRAVAIL

Date	Durée	Activité	Remarques

6.4

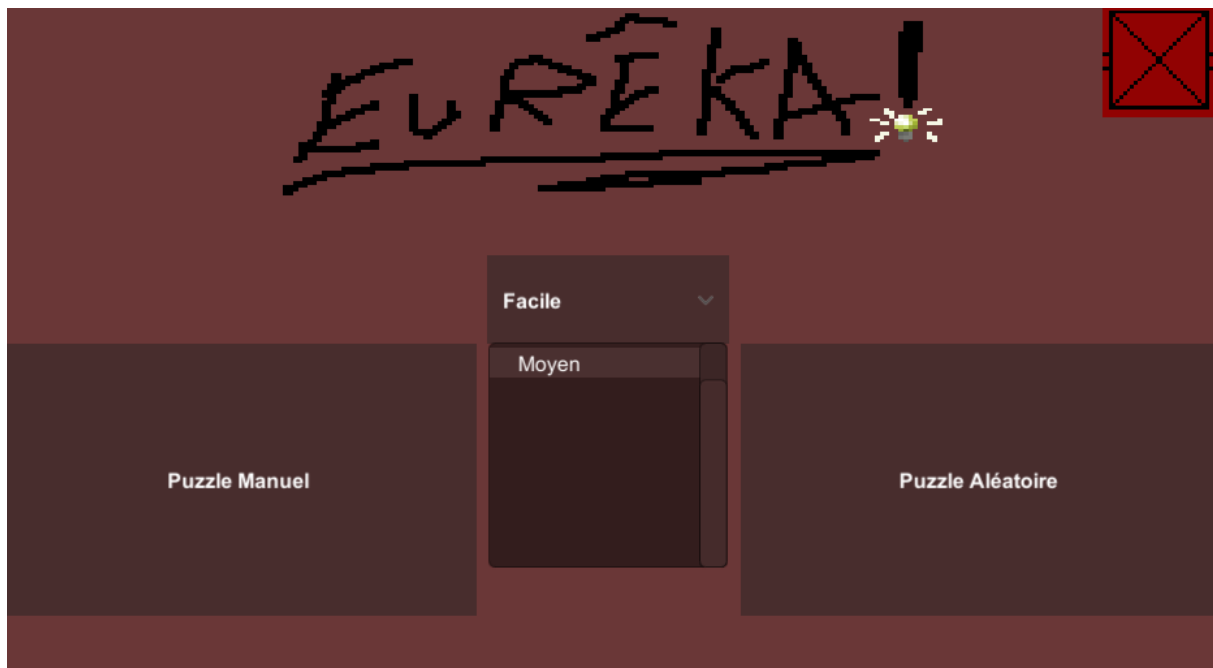
6.5 MANUEL D'INSTALLATION

Sous [tristepin222/Eureka: projet TPI \(github.com\)](https://github.com/tristepin222/Eureka-projet-TPI), installez la dernière release disponible, à faire attention : le programme est compatible seulement sur les version Windows 10.

Après avoir télécharger le jeu, feuillez le désarchiver, avec un programme comme winrar, et simplement lancer le jeu, en cliquant sur l'icône se terminant par EXE.

6.6 MANUEL D'UTILISATION

Après avoir, installer le jeu, et d'avoir lancé le jeu pour la première fois, vous trouverez sur le menu principal, ou plusieurs choix se présente à vous, le choix de difficulté, puzzle manuel ou puzzle aléatoire, le choix de difficulté est par défaut, facile.

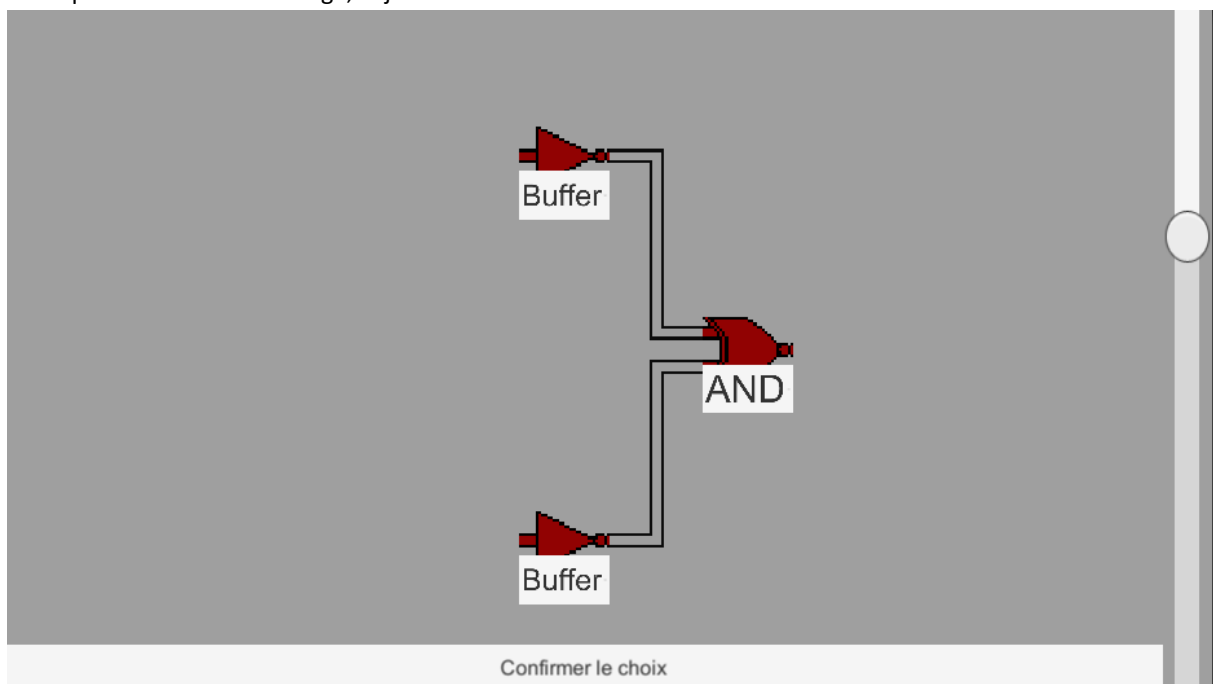


Le bouton « puzzle manuel », vous permet de créer votre propre puzzle.

La liste déroulante, vous permet de choisir la difficulté.

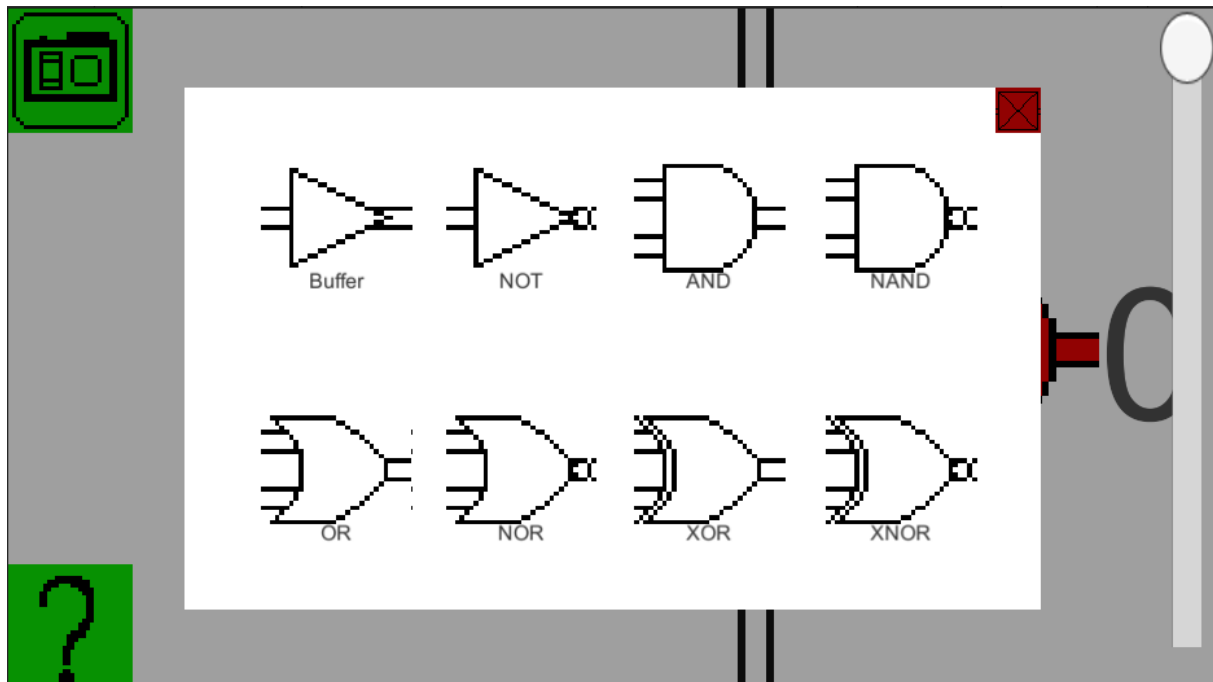
Le bouton « puzzle » aléatoire, vous permet de générer un puzzle aléatoirement.

En cliquant sur le bouton rouge, le jeu se ferme

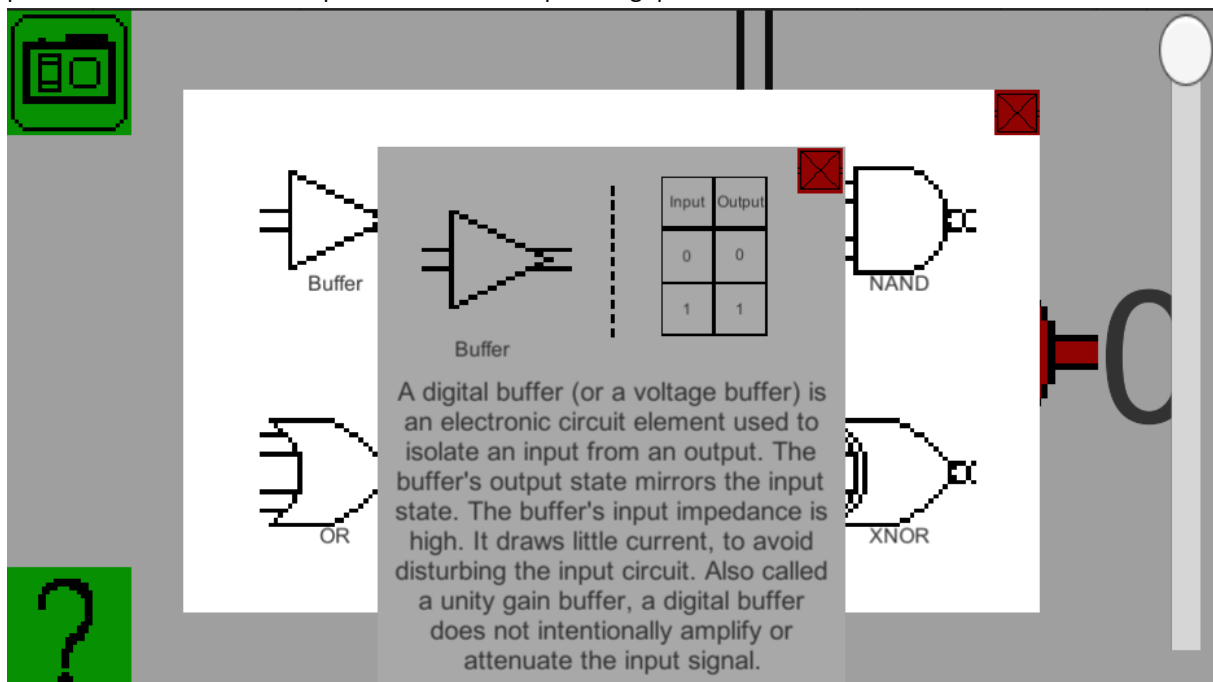


Les listes déroulantes en dessous des portes logiques, permettent de choisir la porte de votre choix, et en cliquant sur « confirmer le choix », votre choix est sauvegardé, et vous passez en mode résolution de puzzle.

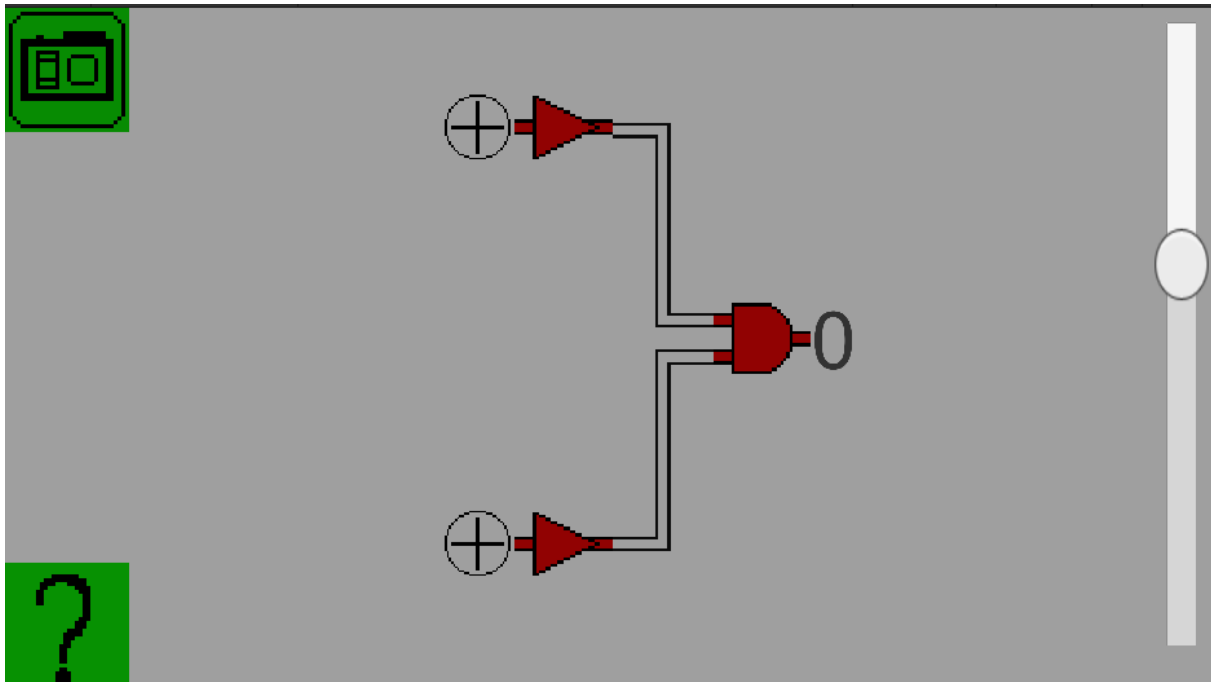
Il y a aussi possibilité de zoomer et dézoomer, avec le slider à droite.



Vous êtes présenté par une UI, avec plusieurs portes logiques, en appuyant sur n'importe qu'elle bouton, il est possible d'afficher une description détaillée de la porte logique.



Il est possible de fermer ces fenêtres en appuyant sur la croix rouge



En appuyant sur le point d'interrogation, il est possible de réafficher la page avec les portes logique et ses descriptions.

En appuyant sur le bouton avec l'appareil photo, il est possible de prendre un screenshot du puzzle courant, avec les boutons à droite des portes logique, il est possible de changer l'entrée, si la sortie finale est en verte, le puzzle est gagné, et vous retournerez au menu principale. Il y a aussi possibilité de zoomer et dézoomer, avec le slider à droite.

6.7 ARCHIVES DU PROJET

6.7.1 VERSION DE LA DOCUMENTATION

Media, ... dans une fourre en plastique

7 GLOSSAIRE

A

AND
 porte logique ET 3, 6, 16, 21
 Assets
 dossier maître dans unity..... 16

B

Buffer
 porte logique entrée = sortie 21

C

C#
 Language de programmation..... 3, 20

E

EXE
 programmes windows 23

F

FALSE
 FAUX 21

G

gameobjects
 Objects du jeu 17
 Gates
 Portes logique 17

N

NAND
 porte logique NON-ET 21
 NOR
 Porte logique NON-OU..... 21
 NOT
 Porte logique NON 21

O

OR
 Porte logique OU 6, 21

P

POO
 Programmation orientée object 3

R

random
 aléatoire 10
 release
 dernière version du programme..... 23

S

screenshot
 Instantané, photo 23, 26
 slider
 glisseur..... 24, 26

T

TRUE
 VRAI 21

U

UI
 User interface, interface graphique.....12, 16, 17, 25
 Unity
 Moteur graphique de jeu..... 17

V

Visual Studio
 Application pour programmer 3, 20

W

winrar
programme d'archivage de dossiers 23

X

XNOR
Porte logique NON-OU exclusif 21
XOR
Porte logique OU exclusif 3, 21