

# Glossaire

A

AND

porte logique ET 5, 9, 20, 24

Assets

dossier maître dans unity 20

B

Buffer

porte logique entrée = sortie 25

C

C#

Language de programmation 5, 24

E

EXE

programmes windows 27

F

FALSE

FAUX 24, 25

G

gameobjects

Objects du jeu 20

Gates

Portes logique 20

I

icescrum

logiciel de gestion de projet 7

N

NAND

porte logique NON-ET 25

NOR

Porte logique NON-OU 25

NOT

Porte logique NON 25

O

OR

Porte logique OU 9, 25

P

POO

Programmation orientée object 5

préfabs

game objects sauvegarder dans les assets 20

R

random

aléatoire 13

release

dernière version du programme 27

S

screenshot

Instantané, photo 27, 30

slider

glisseur 28, 30

sprites

Image pour les éléments du jeu 20

T

TRUE

VRAI 24, 25

U

UI

User interface, interface graphique 15, 20, 29

Unity

Moteur graphique de jeu 21

V

Visual Studio

Application pour programmer 5, 24

W

winrar

programme d'archivage de dossiers 27

X

XNOR

Porte logique NON-OU exclusif 25

XOR

Porte logique OU exclusif 5, 25

# Table des matières

[Glossaire 2](#_Toc104210294)

[Table des matières 4](#_Toc104210295)

[Analyse préliminaire 5](#_Toc104210296)

[Introduction 5](#_Toc104210297)

[Matériel à disposition 5](#_Toc104210298)

[Organisation 5](#_Toc104210299)

[Prérequis 5](#_Toc104210300)

[Objectifs 5](#_Toc104210301)

[Planification initialep 6](#_Toc104210302)

[Planification détaillée 6](#_Toc104210303)

[Analyse / Conception 10](#_Toc104210304)

[Analyse 10](#_Toc104210305)

[Uses cases et scénarios 10](#_Toc104210306)

[Concept 12](#_Toc104210307)

[Diagramme de flux 12](#_Toc104210308)

[Design de l’UI 16](#_Toc104210309)

[Stratégie de test 19](#_Toc104210310)

[Test Unitaire 19](#_Toc104210311)

[Tests d’acceptation 19](#_Toc104210312)

[Risques techniques 20](#_Toc104210313)

[Risque techniques 20](#_Toc104210314)

[Réalisation 20](#_Toc104210315)

[Dossier de réalisation 20](#_Toc104210316)

[Temps actuel 20](#_Toc104210317)

[Scripts 23](#_Toc104210318)

[Dossier 26](#_Toc104210319)

[Scripts 26](#_Toc104210320)

[Version de windows 26](#_Toc104210321)

[Version de Unity 27](#_Toc104210322)

[Version de visual studio 28](#_Toc104210323)

[Version du programme 28](#_Toc104210324)

[Matériel utilisé 28](#_Toc104210325)

[Librairies 28](#_Toc104210326)

[Description des tests effectués 29](#_Toc104210327)

[Tests d’acceptations 29](#_Toc104210328)

[Test Unitaire 30](#_Toc104210329)

[Test d’intégration 31](#_Toc104210330)

[Erreurs restantes 31](#_Toc104210331)

[Liste des documents fournis 32](#_Toc104210332)

[Conclusions 32](#_Toc104210333)

[Annexes 33](#_Toc104210334)

[Résumé du rapport du TPI / version succincte de la documentation 33](#_Toc104210335)

[Sources – Bibliographie 33](#_Toc104210336)

[Journal de travail 33](#_Toc104210337)

[Manuel d'Installation 33](#_Toc104210338)

[Manuel d'Utilisation 33](#_Toc104210339)

[Archives du projet 36](#_Toc104210340)

[Version de la documentation 36](#_Toc104210341)

# Analyse préliminaire

## Introduction

Eurêka est un 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

## Organisation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Nom | Prénom | E-mail | Téléphone |
| Candidat | Borcard | Jessy | Jessy.borcard@cpnv.ch | +41 78 912 32 41 |
| Chef de projet | Viret | Loïc | Loic.viret@cpnv.ch | +41 79 344 07 35 |
| Expert 1 | Gehrig | Romain | Romain.gehrig@cpnv.ch | +41 79 714 43 58 |
| Expert 2 | Lopes | Helder | Helder.costa@eduvaud.ch | +41 76 442 87 79 |

## 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 initialep

1 Planification initiale

La première partie étant la planification et l’analyse du projet, qui englobe, toute la partie de création de diagramme, uses cases et scénarios, et maquettes, elle s’étends sur une période d’une semaine et demi, après que la première partie, il y a la partie réalisation, là où tout le code est créé, l’aspect visuel, cette partie compte pour 2 semaine de travail, et ensuite, une partie documentation, qui prends 3 jours, à noter que la documentation du projet et mis à jour pendant la planification, et réalisation, la dernière partie n’est là que pour rentre la documentation plus jolie, et aussi pour tous ce qui est tests.

## Planification détaillée



2 Planification détaillée semaine 1

Le premier jour du TPI, ce consacre principalement à la création de la planification initiale, mais aussi avec une partie d’installation de l’environnement de travail, cela prend en compte, l’installation d’unity, visual studio, installation de packages nécessaires, qui s’étend au début de la matinée de mardi.

À partir de mardi jusqu’à jeudi, sans compter mercredi, une partie importante du projet, étant la création des uses cases et scénario, qui aideras par la suite l’implémentation des fonctionnalités demandées dans le cahier des charges.

La conception des diagrammes de flux est mise en place jeudi après-midi et toute la journée de vendredi.



3 Planification détaillée semaine 2

Le début de la deuxième semaine est consacrée pour la partie planification et un peu de l’implémentation, une partie, est donnée pour la conception de l’interface graphique, et de la création des sprints, stories, et tâches sur icescrum, mais aussi une partie d’implémentation, création de l’ui sur Unity, la création du système de théorie des portes logiques, qui affiche les portes logiques à l’écrans, qui prends en compte l’ui création du système de porte aléatoire, ou le puzzle créera un puzzle aléatoire afin que le joueur puisse le compléter.



4 Planification détaillée semaine 3

La troisième semaine, porte seulement sur de l’implémentation, en commençant par la création du puzzle aléatoire qui s’étends sur 2 jours complet, et ensuite, la partie de création de porte manuelle, c’est-à-dire, la possibilité de créer son propre puzzle, en choisissant ses portes logiques, et ensuite le compléter, qui s’étends sur tout le reste de la semaine.



5 Planification détaillée semaine 4

La quatrième semaine est coupée par le pont de l’ascension, seulement le lundi et mardi compte, le lundi, se porte sur la création du système de capture de puzzle, et mardi après-midi, la mise à jour de la documentation, qui prends en compte, l’amélioration de la qualité de la documentation, ajout d’images, de description, des choses pas documenté lors de l’implémentation, la conceptions et analyse.



6 Planification détaillée semaine 5

La cinquième semaine, est la dernière semaine de travail, qui prends en compte, les tests unitaire, validations des texte d’acceptation avec une partie de mise à jour de projet, le rendu du projet, étant le mardi 31 mai, il est prise en compte

# Analyse / Conception

## Analyse

### 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

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.



7 cas d'utilisation : puzzle aléatoire

#### 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.



8 cas d'utilisation : puzzle manuel

#### 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.



9 cas d'utilisation : résolution de puzzle

#### 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.



10 case d'utilisation : capture d'écran

#### 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.



11 cas d'utilisation : Théorie des porte logique

## Concept

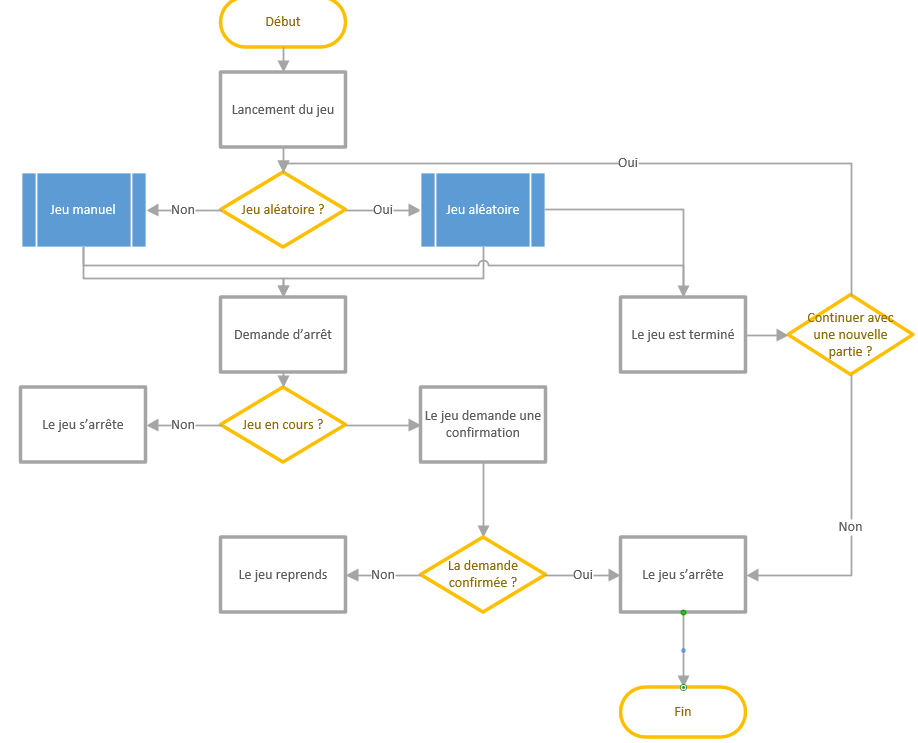
### Diagramme de flux

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, 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.

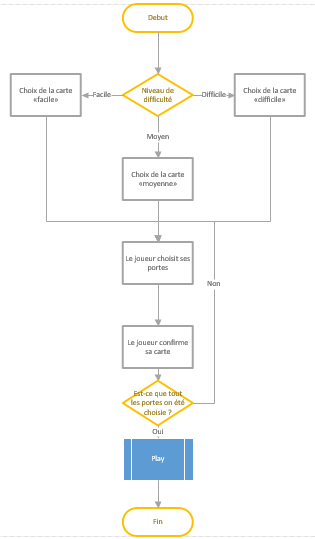
Il peut arriver que le joueur veut quitter le jeu, suivant le cas, si un puzzle est en cours, le jeu demande une confirmation au joueur. Au contraire, si aucun puzzle est en cours, le jeu se quitte sans demander de confirmations



12: Diagramme de flux global de l'application

#### Jeu Manuel

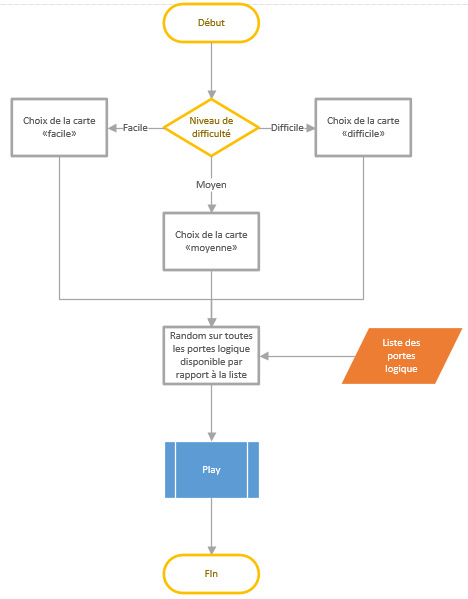
Si le joueur choisit un puzzle manuel, il doit aussi choisir la difficulté du jeu (un puzzle facile a 3 portes, moyen 9 portes et difficile 19 portes), les portes et fils sont déjà présent, le joueur peut juste choisir le type de portes par portes présente, si le joueur ne choisit pas tous ses portes,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



13 Diagramme du puzzle manuel

#### 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 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

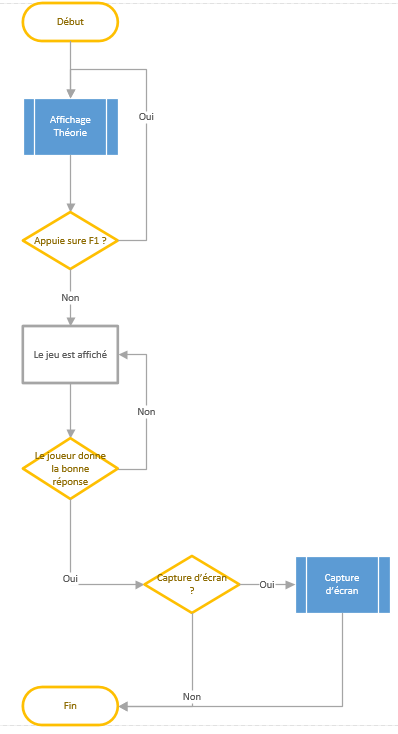


14 Diagramme du puzzle aléatoire

#### 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.



15 diagramme de la résolution de puzzle

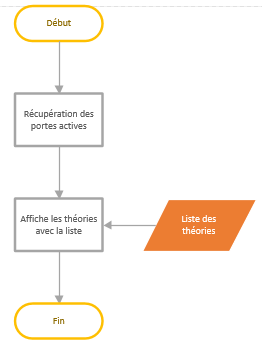
#### Capture d’écran

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

16 diagramme pour la théorie des portes logique



17 Diagramme pour la capture d'écran

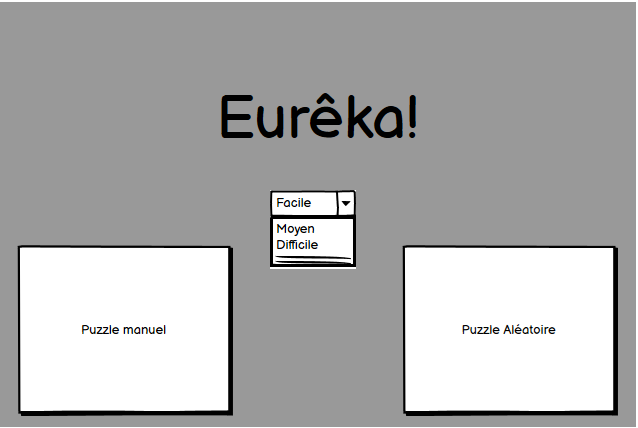
### Design de l’UI

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.

#### Menu principale/jeu suivant

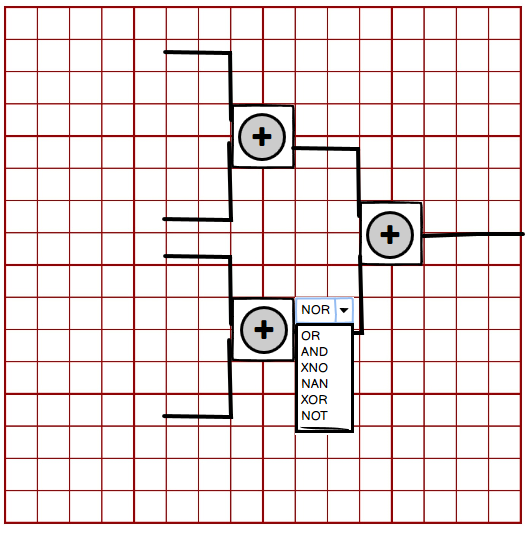
Cette « page » peut être utilisée pour le menu principale, mais aussi comme sélection de mode de jeu pour le prochain puzzle



18 maquette menu principale

#### Menu de création de puzzle

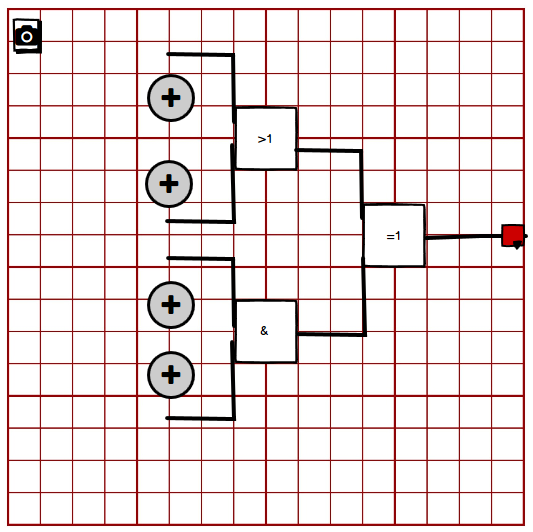
Le joueur aura la possibilité de créer son propre puzzle, les portes sont déjà présents, et le nombre de porte ne change que si le niveau de difficulté change, dans ce cas, le niveau de difficulté est facile, il peut, en cliquant sur la porte, changer de type de porte.



19 maquette : création de puzzle

#### Menu de résolution de puzzle

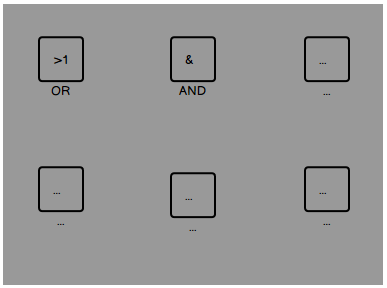
Après avoir terminé la création manuel d’un puzzle ou d’avoir choisi un puzzle aléatoire, le joueur devras compléter le puzzle, si le joueur choisit la bonne réponse, la sortie passe au vert, et le joueur à gagner



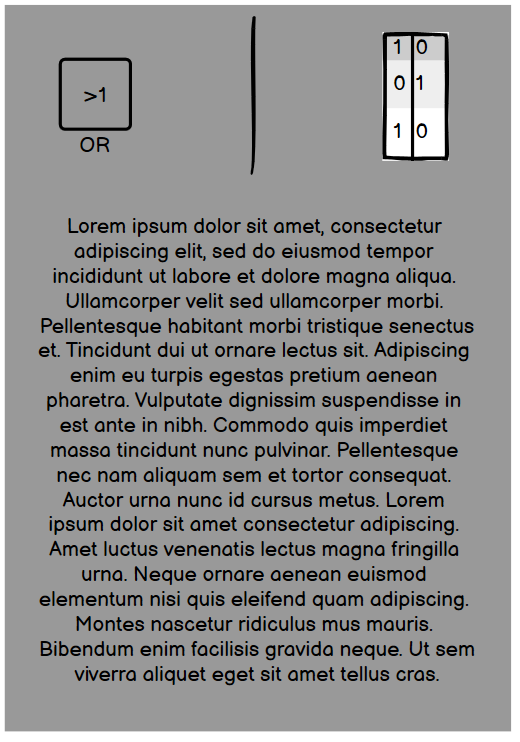
20 maquette : puzzle aléatoire

#### 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.



21 théorie : jeu d’icônes de porte logique



22 théorie détaillé des portes logique

## 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

### Risque techniques

Quelque chose dont je n’ai pas l’habitude de faire, du au faîte 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

#### Documentation

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

# Réalisation

## Dossier de réalisation

### Temps actuel





Comme pour la planification initiale, la matinée du lundi se porte sur la création de la planification initiale, l’installation de l’environnent de travail se fait plus tôt dans la matinée que vers la fin de journée, la création des uses cases et scénario se fait aussi plus tôt que prévu. Une période d’anglais n’était pas prise en compte durant la planification détaillée.

Pour l’après-midi, une partie est allouée pour la création de la documentation, et pour la création des diagrammes de flux.

Pour mardi, il y a beaucoup de documentation, en commençant par la continuation des diagrammes de flux, j’ai aussi un entretien pour technicien en développement, il y a aussi une petite partie pour la recherches des portes logiques, notamment les normes, et leurs tables de vérité.

J’ai aussi créé le projet sur iceScrum, les sprints, le planning des stories avec les tâches, pas de création de tests d’acceptation à ce moment, et la mise à jour de la documentation.

Pour jeudi, il y a un peu d’implémentation, création de l’UI de base, sur unity, sans coder. J’ai aussi beaucoup passé de temps dans le design de l’UI, en prenant en compte le système de théorie de portes, ajout de plusieurs tâches sur les stories dans icescrum pour mieux correspondre le projet.

Le vendredi, beaucoup de changement au niveau de la documentation, par exemple, changement des priorités sur les uses cases et scénario, ajout de descriptifs sur les images dans la documentation.

Comme pour tous les vendredis, suivant un sprint review avec mon chef de projet, je note les choses à changer.



Une toute petite partie le lundi dédié à la conception et analyse, sinon tout le reste de la semaine est de l’implémentation.

Commençant par la création des cartes avec différentes difficultés, recherches sur comment une fonction zoom sur un scroll view, prise d’anciens projet comme exemple, juste après, l’implémentation de la fonction zoom.

Début de la création de résolution de puzzle, en contrôlant les entrées et sorties de portes logique, ajout du système de reconnaissances des portes.

Pour mardi, retour sur les cartes pour adapter les nouveaux scripts au cartes, ajout d’autres porte logique, et des tests sur l’implémentation des scripts, UI jusqu’à présent, ajout dans la documentation de tous ce qui a été fait.

J’ai eu aussi des bugs par rapport à mes préfabs où certain préfabs se faisais écraser par d’autres, j’ai aussi fait la partie de création de puzzle, avec l’adaptation des portes pour la résolution pour que ça puisse être utiliser pour la création de puzzle.

Le jeudi, adaptation de l’UI pour les téléphones, ce qui est demandé dans le cahier des charges, après, avoir terminé la partie de création du puzzle, j’ai dû faire que le jeu passe en mode résolution de puzzle, j’ai donc un script qui gère cette partie.

Pour vendredi, création du système de théorie des portes, avec ajout des tests d’acceptation sur IceScrum, ajout d’éléments dans la documentation, tels que les versions des applications et de l’OS utilisé. Test de l’application en mode build, et commentage du code



Pratiquement la fin de l’implémentation, ajout de la dernière fonctionnalité le lundi : capture d’écrans des puzzles, création des tests unitaires.

Durant le mardi, beaucoup de documentation, ajout du glossaire, ajout des tests effectué, création du manuel d’installation et d’utilisation.

Le jeudi, beaucoup de temps passé sur l’amélioration de l’UI, c’est-à-dire, rendre les graphiques plus jolie.  
et l’après-midi, création du diagramme de classe, avec chaque classes, leur variables, et fonction.

Pour le vendredi, ajout du diagramme de classe dans la documentation, mise à jour de la documentation, rendu plus propre, et création du temps passé : différence entre ce qui a été planifié et ce que y a été fait.

### Scripts

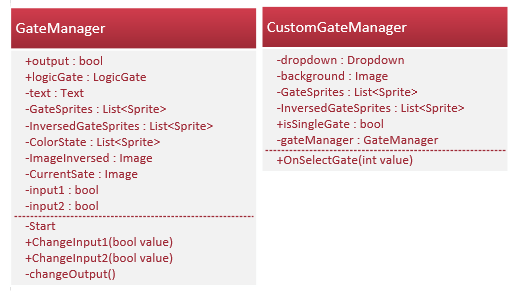
#### Classes



23 Diagramme de la classe LogicGate

* LogicGate : une classe contenant des types de porte logique

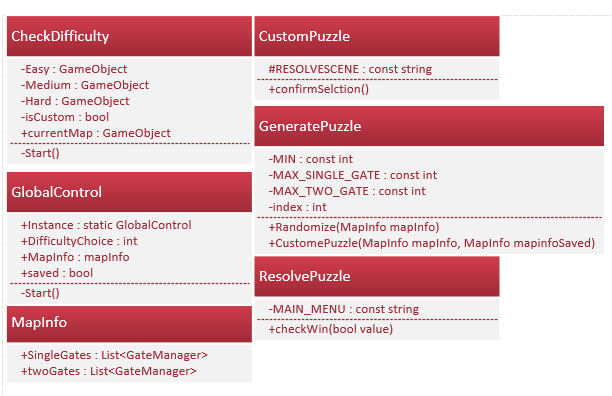
#### Gates



24 Diagramme des classe GateManager et CustomGateManager

* 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

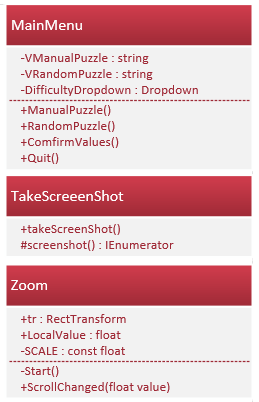
#### Puzzle



25 Diagramme des classe pour la gestion de 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

#### UI



26 driagramme des classe de l'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
* TakeScreenShot : permet la capture d’écran du puzzle actuel, et sauvegarde le sur le disque

### Dossier

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

#### 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

#### Scenes

* Répertoire contenant toutes les scènes du jeu

### Scripts

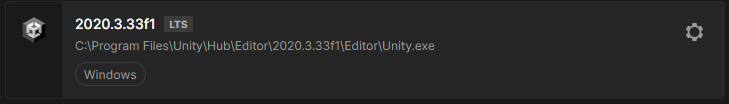
* Classes : répertoire contenant toutes les classes utilisé dans le jeu, notamment LogicGate
* Gates : répertoire avec les scripts gérant les portes logiques
* Puzzle : répertoire avec les scripts gérant les puzzles
* UI : répertoire contenant les scripts qui gère les UI

### Version de windows



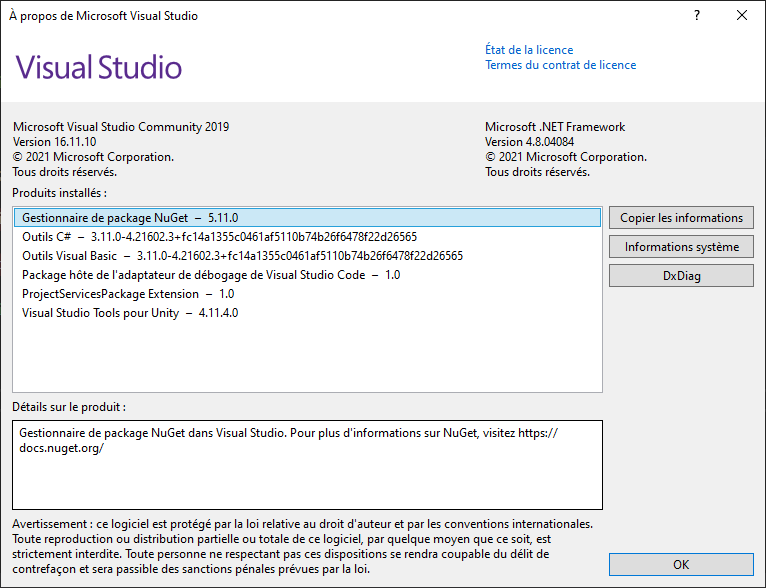
27 Version de windows

### Version de Unity



28 Version de Unity

### Version de visual studio



29 Version de Visual Studio

### Version du programme

2.00

### Matériel utilisé

#### Processeur

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

#### RAM

16.0 Go

#### OS

Windows 10 Éducation 21H2 x64

### Librairies

#### dans le projet

* 2D pixel perfect
* 2D Sprite
* JetBrains Reder Editor
* Test Framework
* TextMeshPro
* Timeline
* Unity UI
* Version Control
* Visual Studio Code Editor
* Visual Studio Editor

#### 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…

## Description des tests effectués

### Tests d’acceptations

#### 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

#### création du puzzle aléatoire

* 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

#### 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 as 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

#### Création du système des théorie

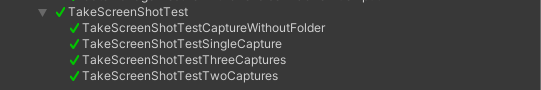
* Les boutons réagissent correctement
  + Les boutons affichent et caches les bonnes dalles

#### 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

### Test Unitaire

Le test unitaire se porte sur le script de la gestion de capture d’écran, c’est-à-dire, le script, qui gère les captures d’écrans, savoir si le fichier existe déjà ou pas, le test est réalisé en C# sur Visual Studio.



30 Test unitaire sur le script de capture d'écran

#### TakeScreenShot Test Capture without folder

Test sur le script de capture d’écran si le dossier des captures d’écran n’existe pas, le script recrée automatiquement le dossier s’il n’est pas présent.

#### TakeScreenShot Test Single Capture

Test sur le script de capture d’écran, avec une seule capture d’écran.

#### TakeScreenShot Test Three Captures

Test sur le script de capture d’écran, avec une trois captures d’écran, la dernière capture prise est sauvegardée.

#### TakeScreenShot Test Two Captures

Test sur le script de capture d’écran, avec une deux captures d’écran, la dernière capture prise est sauvegardée.

### C:\Users\jessy.borcard\AppData\Local\Microsoft\Windows\INetCache\Content.Word\UnitTests.PNGTest d’intégration

Le test d’intégration 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.

#### 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 spécifié, sont de base false

31 Test unitaires

#### GateManager Test Buffer with two true inputs

32 Test d'intégration sur les portes logiques

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

#### 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

#### Gatemanager test nor with one false inpute and one true input

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

#### GateManager Test Not with one true input

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

#### GateManager Test or with one true input

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

#### GateManager Test XNOR with two true inputs

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

#### 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

## Erreurs restantes

S'il reste encore des erreurs:

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

## Liste des documents fournis

* Diagrammes de flux
* Journal de travail
* Maquettes
* Uses case et scénario
* Résumé du projet
* Diagramme de classes

# 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

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)

## Journal de travail

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

## 

## Manuel d'Installation

Sous [tristepin222/Eureka: projet TPI (github.com)](https://github.com/tristepin222/Eureka), 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.

## 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.



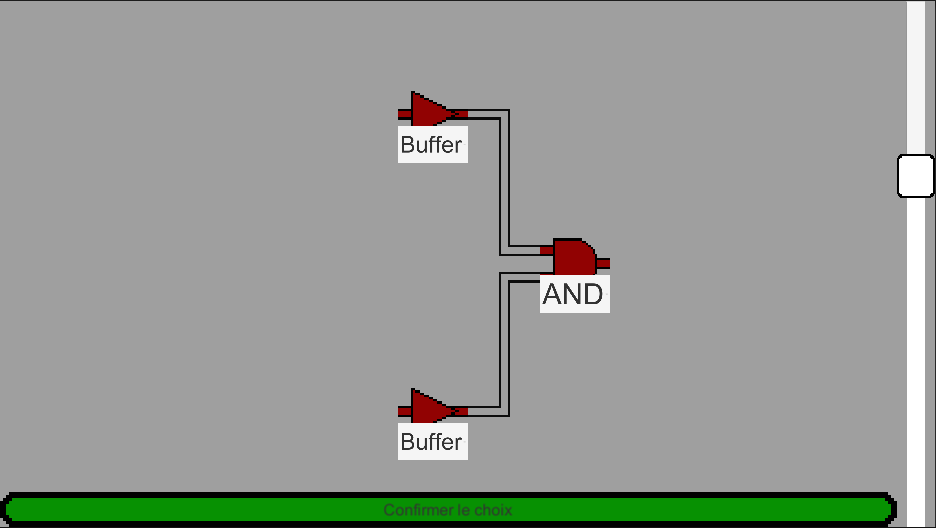
33 Menu principale

Le bouton « puzzle manuel », vous permets 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.

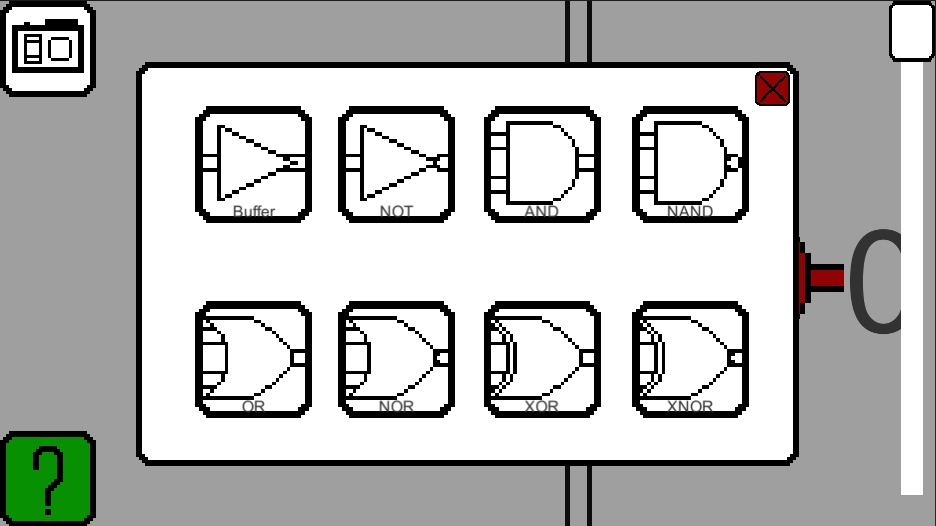
En cliquant sur le bouton rouge, le jeu se ferme



34 Système de création de puzzle

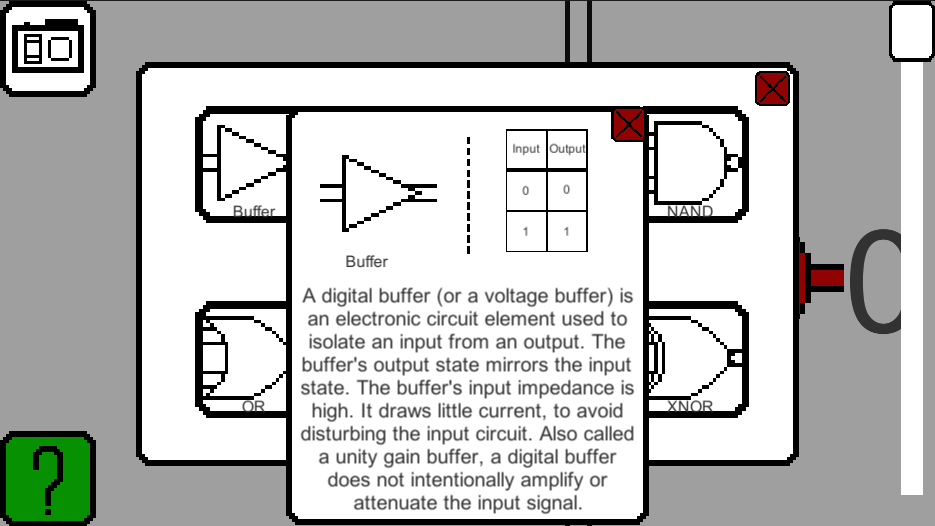
Les listes déroulantes en dessous des portes logiques, permettent de choisir la porte de votre choix, et en cliquant sur « confirmer le choix », votre choix est sauvegardé, est vous passez en mode résolution de puzzle.

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



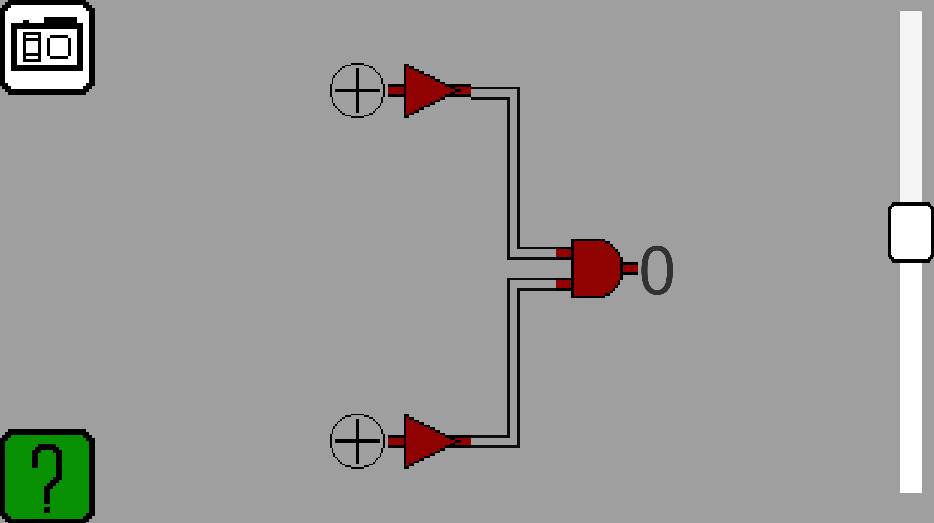
35 Boutons théorie des portes

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.



36 Théorie détaillée des portes logiques

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



37 Résolution de puzzle, aléatoire ou manuel

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 a 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.

## Archives du projet

### Version de la documentation

* 1.00
  + Canevas de base
* 2.00
* 3.00
* 4.00
* 5.00

Media, … dans une fourre en plastique

[1 Planification initiale 6](file:///D:\Eurêka!\Eureka\Documentation\Documentation5.00.docx#_Toc104208841)

[2 Planification détaillée semaine 1 6](#_Toc104208842)

[3 Planification détaillée semaine 2 7](#_Toc104208843)

[4 Planification détaillée semaine 3 7](#_Toc104208844)

[5 Planification détaillée semaine 4 8](#_Toc104208845)

[6 Planification détaillée semaine 5 8](#_Toc104208846)

[7 cas d'utilisation : puzzle aléatoire 9](#_Toc104208847)

[8 cas d'utilisation : puzzle manuel 10](#_Toc104208848)

[9 cas d'utilisation : résolution de puzzle 10](#_Toc104208849)

[10 case d'utilisation : capture d'écran 10](#_Toc104208850)

[11 cas d'utilisation : Théorie des porte logique 11](#_Toc104208851)

[12: Diagramme de flux global de l'application 11](#_Toc104208852)

[13 Diagramme du puzzle manuel 12](#_Toc104208853)

[14 Diagramme du puzzle aléatoire 13](#_Toc104208854)

[15 diagramme de la résolution de puzzle 14](#_Toc104208855)

[16 diagramme pour la théorie des portes logique 15](file:///D:\Eurêka!\Eureka\Documentation\Documentation5.00.docx#_Toc104208856)

[17 Diagramme pour la capture d'écran 15](#_Toc104208857)

[18 maquette menu principale 16](#_Toc104208858)

[19 maquette : création de puzzle 17](#_Toc104208859)

[20 maquette : puzzle aléatoire 17](#_Toc104208860)

[21 théorie : jeu d’icônes de porte logique 18](#_Toc104208861)

[22 théorie détaillé des portes logique 18](#_Toc104208862)

[23 Diagramme de la classe LogicGate 22](#_Toc104208863)

[24 Diagramme des classe GateManager et CustomGateManager 23](#_Toc104208864)

[25 Diagramme des classe pour la gestion de puzzle 24](#_Toc104208865)

[26 driagramme des classe de l'UI 25](#_Toc104208866)

[27 Version de windows 26](#_Toc104208867)

[28 Version de Unity 26](#_Toc104208868)

[29 Version de Visual Studio 27](#_Toc104208869)

[30 Test unitaires 30](file:///D:\Eurêka!\Eureka\Documentation\Documentation5.00.docx#_Toc104208870)

[31 Menu principale 33](#_Toc104208871)

[32 Système de création de puzzle 33](#_Toc104208872)

[33 Boutons théorie des portes 34](#_Toc104208873)

[34 Théorie détaillée des portes logiques 34](#_Toc104208874)

[35 Résolution de puzzle, aléatoire ou manuel 35](#_Toc104208875)