

# Glossaire

A

AND

porte logique ET 6, 10, 26, 31

Assets

dossier maître dans unity 26

B

Buffer

porte logique entrée = sortie 31

C

C#

Language de programmation 6, 30, 31

E

EXE

programmes windows 37

F

FALSE

FAUX 31

G

gameobjects

Objects du jeu 26

Gates

Portes logique 26

I

icescrum

logiciel de gestion de projet 8

N

NAND

porte logique NON-ET 31

NOR

Porte logique NON-OU 31

NOT

Porte logique NON 31

O

OR

Porte logique OU 10, 31

P

packages

Packets, modules 8, 34

POO

Programmation orientée object 6

préfabs

game objects sauvegarder dans les assets 26

R

random

aléatoire 14

release

dernière version du programme 37

S

screenshot

Instantané, photo 39

slider

glisseur 38, 39

sprites

Image pour les éléments du jeu 26

T

TRUE

VRAI 31

U

ui

User interface, interface graphique 8, 34, 35, 36

UI

User interface, interface graphique 16, 26, 38

unity

Moteur graphique de jeu 6, 8, 21, 24, 32, 33, 34, 35, 37, 41

Unity

Moteur graphique de jeu 4, 6, 8, 20, 27, 28, 29, 33, 34, 41

V

Visual Studio

Application pour programmer 6, 30, 31

W

winrar

programme d'archivage de dossiers 37

X

XNOR

Porte logique NON-OU exclusif 31

XOR

Porte logique OU exclusif 6, 31

# Table des matières

[Glossaire 2](#_Toc104293385)

[Table des matières 4](#_Toc104293386)

[Analyse préliminaire 6](#_Toc104293387)

[Introduction 6](#_Toc104293388)

[Matériel à disposition 6](#_Toc104293389)

[Organisation 6](#_Toc104293390)

[Prérequis 6](#_Toc104293391)

[Objectifs 6](#_Toc104293392)

[Planification initialep 7](#_Toc104293393)

[Planification détaillée 8](#_Toc104293394)

[Analyse / Conception 10](#_Toc104293395)

[Analyse 10](#_Toc104293396)

[Uses cases et scénarios 10](#_Toc104293397)

[Concept 12](#_Toc104293398)

[Diagramme de flux 12](#_Toc104293399)

[Design de l’UI 16](#_Toc104293400)

[Stratégie de test 19](#_Toc104293401)

[Test Unitaire 19](#_Toc104293402)

[Test d’intégration 19](#_Toc104293403)

[Tests d’acceptation 20](#_Toc104293404)

[Risques techniques 20](#_Toc104293405)

[Risque techniques 20](#_Toc104293406)

[Réalisation 20](#_Toc104293407)

[Dossier de réalisation 20](#_Toc104293408)

[Temps actuel 20](#_Toc104293409)

[Scripts 24](#_Toc104293410)

[Dossier 27](#_Toc104293411)

[Scripts 27](#_Toc104293412)

[Version de windows 28](#_Toc104293413)

[Version de Unity 28](#_Toc104293414)

[Version de visual studio 29](#_Toc104293415)

[Version du programme 29](#_Toc104293416)

[Matériel utilisé 29](#_Toc104293417)

[Librairies 29](#_Toc104293418)

[Description des tests effectués 30](#_Toc104293419)

[Tests d’acceptations 30](#_Toc104293420)

[Test Unitaire 31](#_Toc104293421)

[Test d’intégration 32](#_Toc104293422)

[Erreurs restantes 32](#_Toc104293423)

[Problèmes de build 32](#_Toc104293424)

[Liste des documents fournis 32](#_Toc104293425)

[Conclusions 33](#_Toc104293426)

[Objectifs atteint 33](#_Toc104293427)

[Points positifs / négatifs 33](#_Toc104293428)

[Difficultés particulières 33](#_Toc104293429)

[suite possible 33](#_Toc104293430)

[Annexes 34](#_Toc104293431)

[Résumé du rapport du TPI / version succincte de la documentation 34](#_Toc104293432)

[Situation de départ 34](#_Toc104293433)

[Mise en oeuvre 34](#_Toc104293434)

[résultats 34](#_Toc104293435)

[Sources – Bibliographie 34](#_Toc104293436)

[Journal de travail 34](#_Toc104293437)

[Manuel d'Installation 38](#_Toc104293438)

[Manuel d'Utilisation 38](#_Toc104293439)

[Archives du projet 41](#_Toc104293440)

[Version de la documentation 41](#_Toc104293441)

[Version du jeu 41](#_Toc104293442)

[Table des illustrations 42](#_Toc104293443)

# 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 initiale

1 Planification initiale

2 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



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.



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.



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.



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.



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.



8 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 présentes.

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.



9 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.



10 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.



11 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.



12 cas d'utilisation : Théorie des porte logique

## Concept

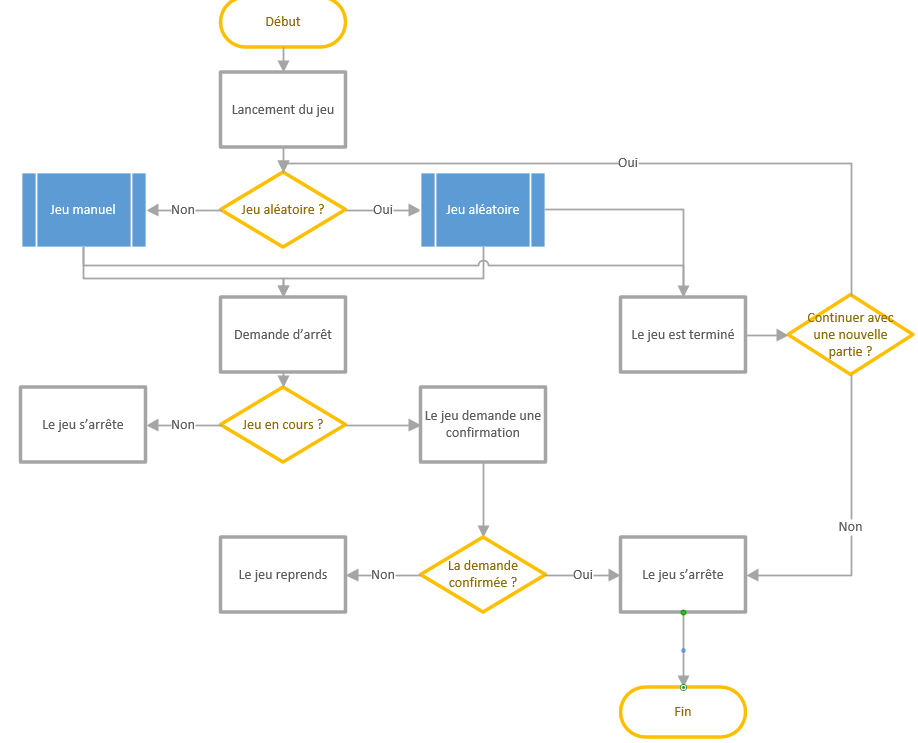
### Diagramme de flux

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.

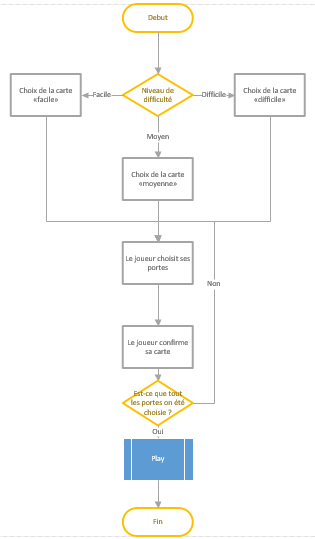
Il peut arriver que le joueur veut quitter le jeu, suivant le cas, si un puzzle est en cours, le jeu demande une confirmation au joueur. Au contraire, si aucun puzzle est en cours, le jeu se quitte sans demander de confirmations



13: Diagramme de flux global de l'application

#### Jeu Manuel

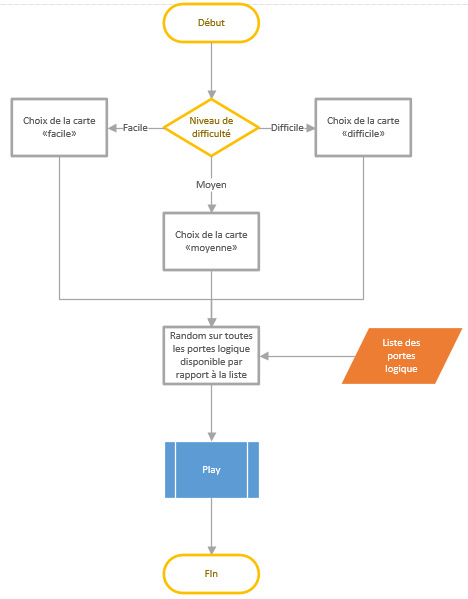
Si le joueur choisit un puzzle manuel, il doit aussi choisir la difficulté du jeu (un puzzle facile a 3 portes, moyen 9 portes et difficile 19 portes), les portes et fils sont déjà présent, le joueur peut juste choisir le type de portes par portes présente, si le joueur ne choisit pas tous ses portes,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



14 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

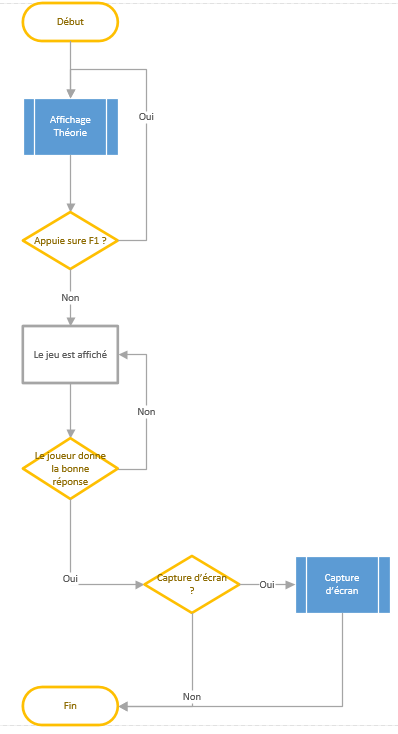


15 Diagramme du puzzle aléatoire

#### 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.



16 diagramme de la résolution de puzzle

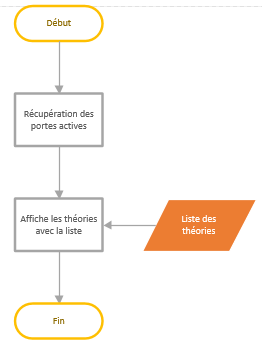
#### Capture d’écran

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

17 diagramme pour la théorie des portes logique



18 Diagramme pour la capture d'écran

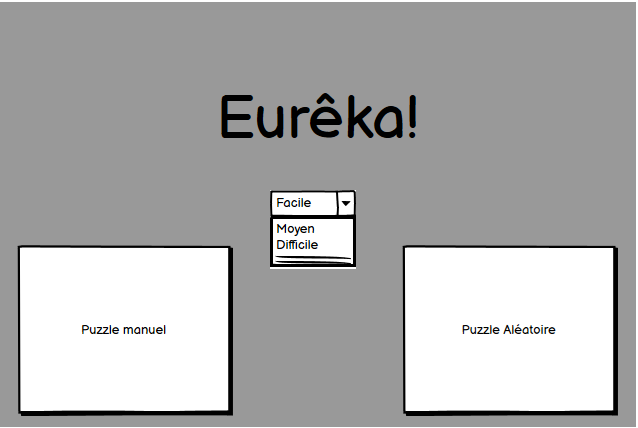
### Design de l’UI

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

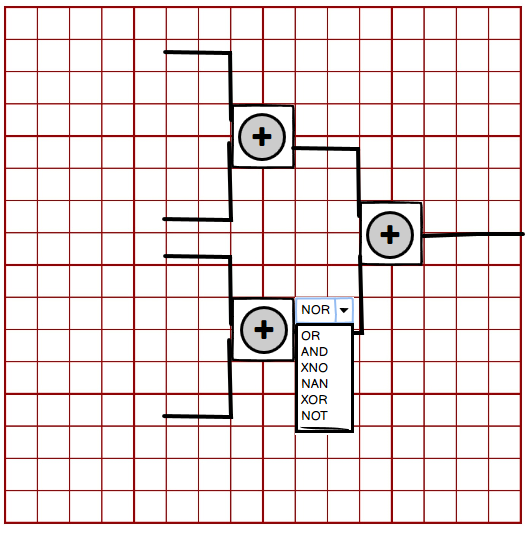
Cette « page » peut être utilisée pour le menu principale, mais aussi comme sélection de mode de jeu pour le prochain puzzle



19 maquette menu principale

#### Menu de création de puzzle

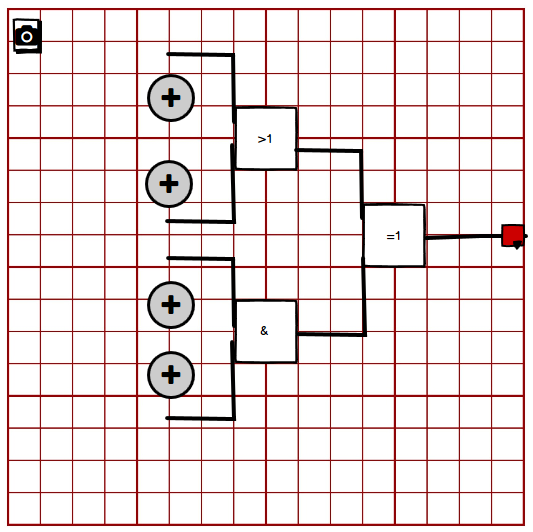
Le joueur aura la possibilité de créer son propre puzzle, les portes sont déjà présents, et le nombre de porte ne change que si le niveau de difficulté change, dans ce cas, le niveau de difficulté est facile, il peut, en cliquant sur la porte, changer de type de porte.



20 maquette : création de puzzle

#### Menu de résolution de puzzle

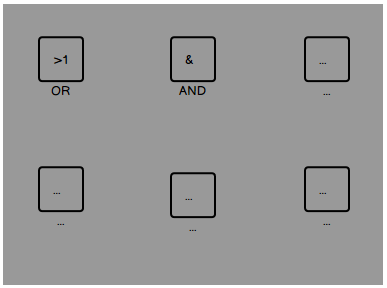
Après avoir terminé la création manuel d’un puzzle ou d’avoir choisi un puzzle aléatoire, le joueur devras compléter le puzzle, si le joueur choisit la bonne réponse, la sortie passe au vert, et le joueur à gagner



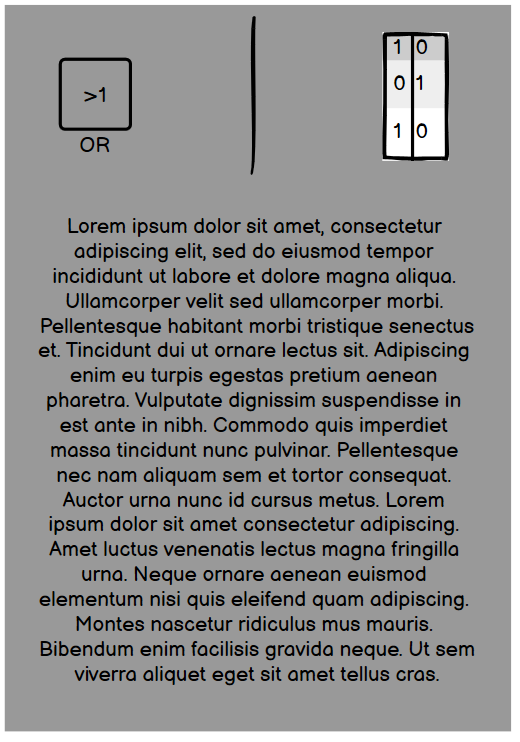
21 maquette : puzzle aléatoire

#### 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.



22 théorie : jeu d’icônes de porte logique



23 théorie détaillé des portes logique

## Stratégie de test

La stratégie de test contient principalement les tests, unitaire, intégration et d’acceptation, mais aussi des tests de fonctionnement sur plusieurs type d’environnement différents

### 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.

### Test d’intégration

Ce n’est pas une chose demandée dans le cahier des charges, mais pour la vérification du bon fonctionnement du jeu, un test d’intégration sera créé

### 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 de mon projet de la bonne manière

# Réalisation

## Dossier de réalisation

### Convention de nommage

Du fait que je fais du C#, et que je travaille sur unity, j’utilise les conventions de nommage C# étant : Pascal Case.

C’est une convention de nommage dont je connais bien, et que j’utilise dans tous mes projets de code, c’est lisible et facile de faire la différence entre fonctions, interfaces, variables et classes

### Temps actuel





Semaine 1, 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.



Semaine 2, temps actuel

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



Semaine 3, temps actuel

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.

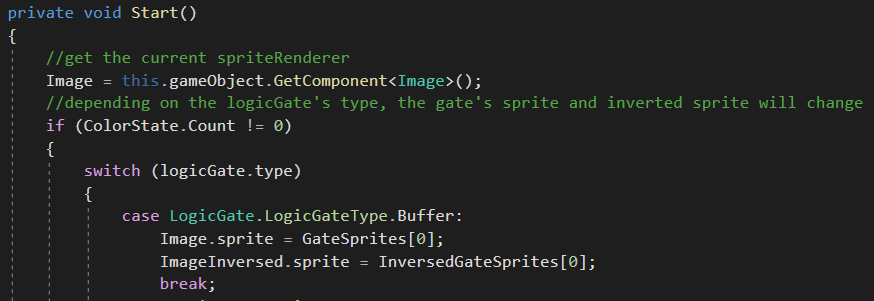
Le jeudi, beaucoup de temps passé sur l’amélioration de l’UI, c’est-à-dire, rendre les graphiques plus jolie.  
et l’après-midi, création du diagramme de classe, avec chaque classes, leur variables, et fonction.

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

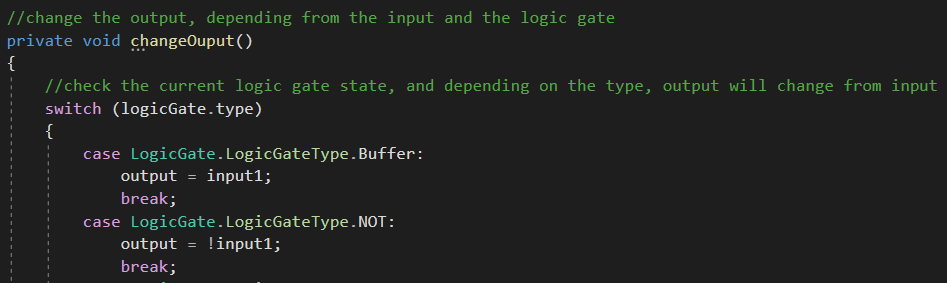
#### GateManager

Le script de GateManager, est le script principal qui gère tout l’aspect de gestion d’entrées et sorties en fonction des types de portes logique actuel. Il comporte plusieurs fonctions, plusieurs listes, et un event.



partie du code dans la fonction start

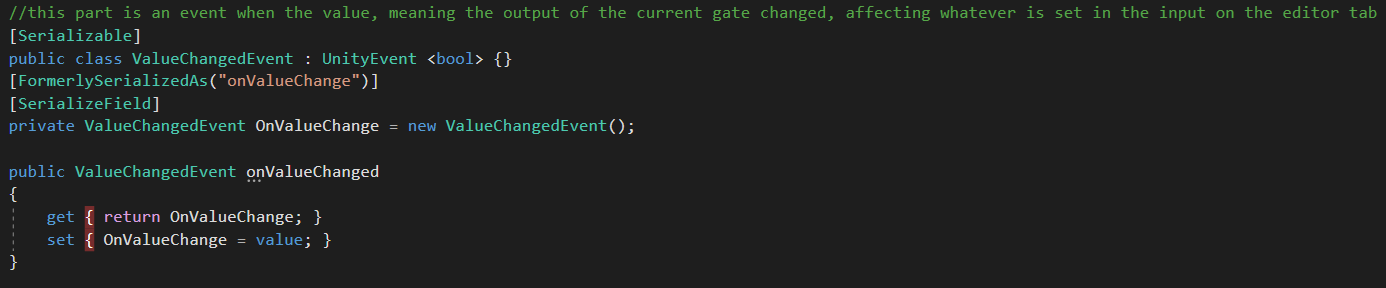
Le script commence par initialiser son sprite, et son sprite inversé, le sprite inversé permet l’affiche rouge et vert par rapport à la forme de la porte logique, tout ça, en fonctionne du type de la porte logique, dans le code si dessus, si une porte buffer est choisie, il va récupérer la première entrée des listes.



fonction qui change l'output de la porte logique

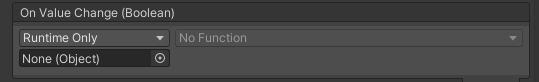
C’est la fonction cœur de se script, la fonction qui, vas par rapport au entrée données, soit par l’utilisateur, soit par une autre porte logique, vas changer sa sortie, chaque porte logique as un fonctionnement différent, donc, la sortie va changer par rapport à la porte logique, dans le code si dessus, la sortie de la porte buffer, est égale à la première entrée, si la deuxième entrée est changée, à la place, elle ne changera rien de la porte logique.

Même fonctionne pour la porte NOT, qui inverse simplement l’entrée.



event qui permet de changer une autre porte logique

Une autre partie du même script dont je suis fière, qui est un event.



l'event visuel dans unity

Voici comment l’event ressemble au niveau d’unity, il est donc possible de glisser une autre porte logique dans le champ objet, et de choisir la fonction pour changer l’input de l’autre porte logique, cela permet donc d’avoir des portes logiques à la chaine, et si, nous mettions 5 porte logique à la chaine, et changerions l’entrée de la première porte logique, toutes les autres portes seraient impactée par le changement, si, le type de porte le permet, bien évidemment

#### Classes



Diagramme de la classe LogicGate

* LogicGate : une classe contenant des types de porte logique

#### Gates

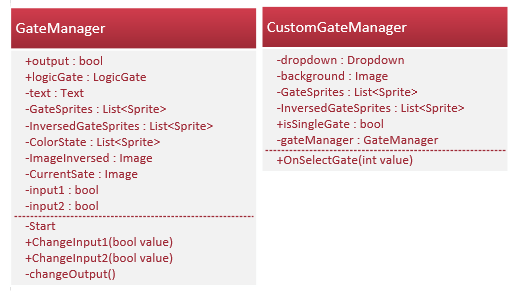


Diagramme des classe GateManager et CustomGateManager

* 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

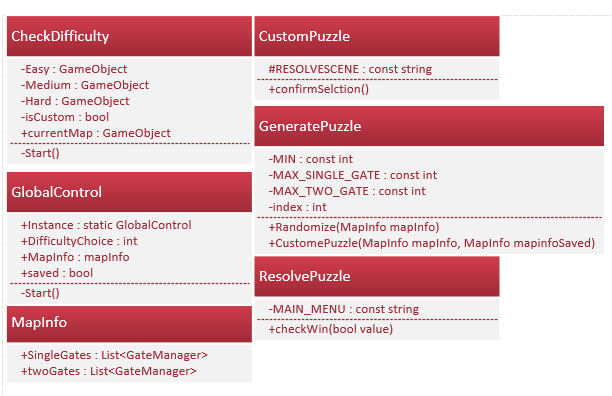
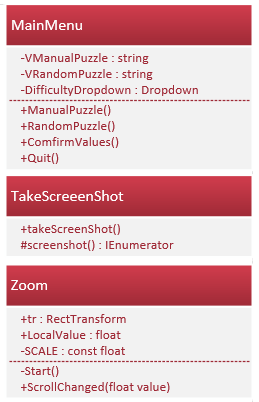


Diagramme des classe pour la gestion de puzzle

* 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



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

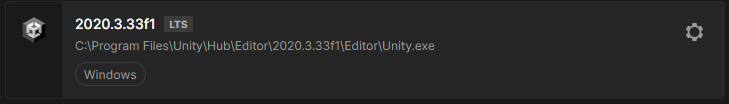
* Classes : répertoire contenant toutes les classes utilisé dans le jeu, notamment LogicGate
* Gates : répertoire avec les scripts gérant les portes logiques
* Puzzle : répertoire avec les scripts gérant les puzzles
* UI : répertoire contenant les scripts qui gère les UI

### Version de windows



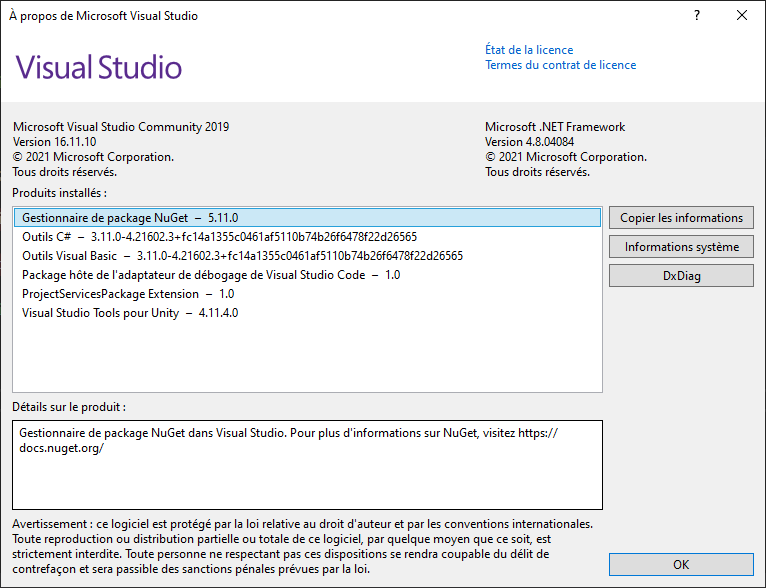
Version de windows

### Version de Unity



Version de Unity

### Version de visual studio



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

## 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

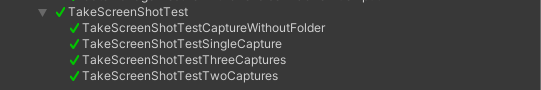
* Les boutons réagissent correctement
  + Les boutons affichent et caches les bonnes dalles

#### 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.



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

39 Test d'intégration sur les portes logiques

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

40 Test unitaires

#### 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

#### 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

## Erreurs restantes

### Problèmes de build

J’ai eu beaucoup de problèmes pour build avec ma solution de tests, effectivement il est impossible d’avoir un build fonctionnel avec ma solution de test dans le projet, je suis forcé d’enlever la solution pour pouvoir avoir un build opérationnel.

## Liste des documents fournis

* Diagrammes de flux
* Journal de travail
* Maquettes
* Uses case et scénario
* Diagramme de classes
* Le jeu : [tristepin222/Eureka: projet TPI (github.com)](https://github.com/tristepin222/Eureka)
* IceScrum : [IceScrum : Eurêka!](https://icescrum.cpnv.ch/p/EUREKA)

# Conclusions

### Objectifs atteint

Tous les points mentionné dans le cahier des charges, et dans la planification initiale/détaillée ont été implémenté dans le projet

### Points positifs / négatifs

#### point positifs

Le cahier des charges laisse beaucoup de liberté dans la conception, analyse et implémentation, les seuls points demandés étant une idée vague des fonctions à implémenter, m’a permis d’utiliser mon imagination, ce que j’ai beaucoup apprécié, la communication entre, les experts, chef de projet, était aussi très cordiale.

#### Point négatifs

Le seul point négatif dont je vois, étant le fait que le visuel adapté au PC et téléphone était un point technique à évaluer, je ne comprends pas pourquoi c’est de la technique, si cela n’as pas grand-chose à voir avec la formation CFC, même si on a eu de la conception graphique durant la formation, c’est clairement pas un élément principal, de plus, il n’est pas mentionné dans le cahier des charges si un build PC et un build téléphone doit être créer, il est aussi très difficile de faire des tests sur des téléphones

### Difficultés particulières

J’ai eu beaucoup de peine avec mes tests unitaire et d’intégration, je ne sais toujours pas pourquoi, mais il est impossible pour d’avoir ma solution de tests et d’avoir un build fonctionnel, je supprime ma solution à chaque fois que je veux build, et je la remets quand j’ai besoin de push sur github, j’ai eu aussi quelques problèmes avec les références, et les tests qui ne voulais pas passer, unity me disant que je ne pouvais pas utiliser la mot clé « new » avec « MonoBehaviour », pourtant je n’utilisais pas le mot clé « new »

### suite possible

J’espère, après avoir rendu le TPI, de pouvoir continuer un peu sur le jeu, et de rajouter des fonctions plus avancé, comme la création des cartes, ajout de porte logique, la sauvegardes des portes logiques, pouvoir relier plusieurs portes logiques.

À la fin j’aimerais pouvoir simuler des simples puces, qu’on peut trouver sur le marché.

# Annexes

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

### Situation de départ

Le projet a débuté le 2 mai 2022, avec la prise de connaissance du CdC, j’ai, après avoir lu et approuvé ce dernier avec mon premier expert, commencé à créer la planification initiale et la planification détaillée la conception et analyse de projet, en créant des uses cases scénarios, stories, diagramme de flux et maquette pour toute la partie UI, j’ai dans cette partie fait la création du menu principale.

### Mise en oeuvre

Après avoir terminé toute la partie de conceptualisation et analyse du projet, il était temps de passer à l’implémentation des fonctionnalités demandée, j’ai donc coder, en premier lieu le script pour gérer les entrées, et sorties des portes logiques, ce qui est en élément essentiel du projet, après je suis passé sur la création du système de génération de puzzle, ensuite la création du système de création de puzzle custom, l’affichage des théories des portes, et le système de capture d’écran, tout ça a été documenté durant toute la partie d’implémentation.

### résultats

J’ai pour, confirmer tout mon implémentation créer des tests, qui m’as permis de savoir si ce que j’ai fait, fonctionne dans le sens que je voulais ou pas, j’ai donc créer un test unitaire et un test d’intégration, le test unitaire se porte sur le système de capture d’écran, et le test d’intégration se porte sur le système « GateManager », ou je teste que les entrées que je mets, me donne la sortie appropriée la porte logique, étant donné que la table de vérité de chaque porte logique est inchangeable, il est très difficile de fausser un test, tous les tests effectué ont passé, après avoir terminé la partie de test, j’ai créé un document ou je note ce que j’ai fait durant le projet, comme pour la planification détaillée, mais ce que j’ai vraiment fait, pour comparer la planification et l’implémentation, et pour terminer, les finitions de la documentation pour le rendu,

## Sources – Bibliographie

game, Let's Make A. "Unity Unit Testing Basics Tutorial (C#, NUnit)." *Let's Make A game.* 05 24, 2022. https://letsmakeagame.net/unity-unit-testing-basics-tutorial/#:~:text=Unit%20Tests%20are%20a%20way%20to%20be%20more,has%20a%20new%20name%20%E2%80%93%20%E2%80%9CUnity%20Test%20Framework%E2%80%9D.

Microsoft. "C# Coding Conventions." *Microsoft Docs.* 05 24, 2022. https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions.

tutorialspoint. "Logic gates." *tutorialspoint.* 05 16, 2022. https://www.tutorialspoint.com/computer\_logical\_organization/logic\_gates.htm.

Uccello, Anthony. "Introduction To Unity Unit Testing." *raywenderlich.* 05 24, 2022. https://www.raywenderlich.com/9454-introduction-to-unity-unit-testing.

Unity. "Screenshot, but from a certain camera." *Unity Forum.* 05 16, 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.* 05 16, 2022. https://docs.unity3d.com/ScriptReference/Application-persistentDataPath.html.

—. "Unity - ScreenCapture.CaptureScreenshot." *Unity Documentation.* 05 16, 2022. https://docs.unity3d.com/ScriptReference/ScreenCapture.CaptureScreenshot.html.

## Journal de travail

|  |  |
| --- | --- |
| **00:55** | Création de la planification initiale |
| **00:35** | Installation de unity 2020 |
| **01:00** | création des uses case et scénario (Puzzle aléatoire et puzzle manuelle) |
| **01:00** | Création du rapport de projet |
| **00:35** | création des uses case et scénario (Résolution de puzzle, capture d'écran, théorie des porte logiques) |
| **00:20** | mise à jour du rapport de projet |
| **00:50** | Création des diagrammes de flux (code main) |
| **00:03** | Rendu de la planification initiale |
| **00:12** | Création des diagrammes de flux (code pour puzzle aléatoire) |
| **00:05** | Création du projet de base sur unity |
| **00:20** | Création du dépôt git + first commit |
| **01:10** | Création des diagrammes de flux (code pour puzzle aléatoire, puzzle manuel, capture d'écran, affichage théorie, affichage scores) |
| **00:05** | installation de baslamiq |
| **00:35** | Entretien pour technicien dévloppement |
| **00:40** | Création du design de l'UI (puzzle aléatoire et puzzle manuel) |
| **00:15** | Recherches sur les normes des porte logiques (IEC) |
| **00:45** | Création du design de l'UI (main, prise de screenshot) |
| **00:04** | Création du design de l'UI (Théorie) |
| **00:44** | mise à jour du rapport de projet |
| **00:23** | Création du projet sur IceScrum + mise en place des sprints et création des stories |
| **00:15** | mise à jour du rapport de projet |
| **01:20** | mise à jour du rapport de projet |
| **00:05** | Envoie aux experts et chef de projets la documentation courante (rapport de projet et le journal de travail) |
| **00:30** | intégration des diagrammes de flux, théorie des porte, capture d'écran, schéma du design de l'UI |
| **00:03** | mise à jour du design pour intégrer le niveau de difficulté |
| **00:12** | intégration du schéma du design de l'UI, changement du menu principale, ajout des schémas des puzzle aléatoire et manuel |
| **01:30** | intégration du schéma du design de l'UI, ajout des schémas des puzzle aléatoire et manuel, théorie des portes + changement du schéma des porte logique |
| **00:30** | ajout des tâches sur icescrum pour toutes les stories |
| **00:24** | création de sprites pour le jeu (porte logiques, câbles, effets) |
| **00:25** | ajout d'une stories pour l'ui dans ice scrum, avec des tâches |
| **00:10** | création de l'ui du menu démarré sur Unity |
| **00:07** | installation de packages pour unity 2D |
| **00:10** | création de l'ui du menu démarré sur Unity |
| **00:13** | ajout de tâches sur ice scrum pour l'ui, création de plusieurs tâches pour ajout de scènes, inégration de boutons |
| **00:08** | Importation des Sprites |
| **00:06** | recherches sur les conventions de nommage sur c# |
| **00:05** | création du script main menu, qui gère les boutons, et la liste déroulante du menu démarré |
| **00:11** | modifications des uses cases et scénario, plus changement des prioritée |
| **00:09** | modification des diagrammes de flux |
| **00:46** | ajout de descriptif aux images sur les uses cases et scénario, diagrammes de flux |
| **00:08** | modification des diagrammes de flux |
| **00:04** | modification du journal de travail pour prendre en compte l'avancement de tâche pas terminée, et ajout du numéro de tâches |
| **00:17** | ajout de descriptif aux images sur les uses cases et scénario, diagrammes de flux et maquettes |
| **00:21** | ajout de descriptif aux images sur les diagrammes de flux et maquettes |
| **00:14** | ajout de descriptif aux stories et tâches sur iceScrum |
| **00:23** | Aide à Arthur bourgue |
| **00:12** | ajout de descriptifs aux tâches des stories sur icescrum |
| **00:54** | Sprint review avec M. Viret |
| **00:19** | ajout de descriptions sur certaines tâches qui en manque, ajout de la durée des tâches, changement du format de cellule dans le journal de travail |
| **00:41** | ajout de légendes, changement, modifications du style, mise en forme |
| **00:16** | Aide à Arthur bourgue |
| **00:20** | Changement du des maquette, afin que la théorie des portes logique soit plus facile à lire sur téléphone |
| **00:11** | intégration des nouvelles maquettes dans la documentation, mise en page plus correcte |
| **00:13** | suppression de la storie sur l'UI, intégration des tâches sur les autres stories, création d'une storie Documentation |
| **01:10** | création des cartes, facile, moyen, ajout de UI pour résolution de puzzle, adaption des sprites |
| **00:27** | Recherches sur comment ajouter une fonction zoom sur un scroll view |
| **00:24** | intégration d'une fonction zoom sur le puzzle |
| **02:28** | création du système de résolution de puzzle, partie contrôle d'entrée sortie, avec choix de niveau |
| **00:47** | ajout du système de reconnaissances des portes |
| **00:44** | Adaptation des autres cartes pour le nouveau système |
| **00:14** | Ajout des autres portes logique |
| **00:34** | Tests des implémentation des ports, vérification que les portes réagisse bien par rapport à la table de véritée |
| **00:12** | Tests des implémentation des ports, vérification que les portes réagisse bien par rapport à la table de véritée |
| **00:19** | Ajout de couleur rouge, vert pour différiencer sortie et entrée |
| **00:14** | Certaines portes ne s'affiche pas correctement |
| **00:15** | Création du sytème de vérification du score |
| **01:18** | Création du système de puzzle aléatoire |
| **01:02** | Ajout du listing des scripts et dossier dans la documentation |
| **00:33** | correction de bug au niveau des préfabs |
| **01:23** | Implémentation de la partie création des puzzles, adaptation de l'UI, changement des portes quand le joueur choisit une porte |
| **00:08** | push sur github, envoie du mail aux experts et chef de projet, contrôle de la documentations |
| **00:20** | Adaptation de l'ui de la création de puzzle pour les téléphones |
| **00:27** | création du script qui contrôle toutes les portes logiques et passe en mode résolution |
| **02:23** | création du script qui contrôle toutes les portes logiques et passe en mode résolution |
| **01:35** | création du sytème de théorie |
| **00:20** | création du sytème de théorie |
| **00:15** | Ajout des tests d'acceptation sur icescrum |
| **00:24** | Ajout des version de windows et unity, plus ajout des descriptions des nouveaux script : GeneratePuzzle, customGateManager, customPuzzle |
| **00:18** | ajout du bouton pour quitter le jeu, ajout d'icones pour le jeu |
| **00:16** | Commentage du code |
| **00:11** | Commentage du code |
| **00:19** | Ajout des tests d'acceptation éffectué dans le code |
| **00:20** | ajout des splash screen et écran d'acceuil |
| **00:23** | ajout des variables, fonctions, events des scripts dans la documentation |
| **00:22** | Sprint review avec M. Viret |
| **00:25** | Création de la planification détaillée |
| **00:19** | Aide à Valentin Zingg |
| **00:57** | Création de la planification détaillée |
| **00:16** | ajout des stories de documentation sur chaque sprint sur icescrum |
| **00:12** | Tests globale des fonctionnalitée en mode build |
| **00:05** | Envoie aux experts et chef de projets la documentation courante (rapport de projet et le journal de travail) |
| **00:20** | Ajout de la fonction prise en photo le puzzle |
| **00:40** | Recherches sur ScreenCapture.CaptureScreenshot, son fonctionnement |
| **00:27** | Ajout de la fonction prise en photo le puzzle |
| **00:08** | Ajout des lien, sources utilisé durant le projet |
| **00:11** | Ajout des lien, sources utilisé durant le projet |
| **00:39** | Ajout des tests d'acceptation pour le système de capture d'écran, ajout des tests dans la documentation, et ajout des version de visual studio |
| **00:39** | Création du test unitaire sur GateManager (gestion de la porte logique, entrée et sortie) |
| **01:23** | Création du test unitaire sur GateManager (gestion de la porte logique, entrée et sortie) |
| **00:11** | Ajout d'une story tests dans iceScrum |
| **00:18** | ajout de releases sur Github |
| **00:34** | Aide à bastien |
| **00:38** | ajout du test unitaire dans la documentation |
| **00:35** | ajout d'entrée dans le glossaire |
| **00:30** | ajout du reste du test unitaire dans la documentation |
| **00:30** | Tests du jeu de bastien fardel |
| **00:14** | Ajout des risques techniques, tests et documentation |
| **00:26** | Aide à Dylan berney |
| **00:24** | Mise en page de la documentation |
| **00:34** | Commentage du code du test unitaire |
| **00:33** | création du manuelle d'installation |
| **00:09** | création du manuelle d'utilisation |
| **00:25** | création du manuelle d'utilisation |
| **00:25** | Problème de variables non assignée |
| **01:06** | création du manuelle d'utilisation |
| **01:08** | ajout de releases sur Github |
| **00:22** | poffinage de l'ui |
| **00:45** | poffinage de l'ui |
| **01:07** | poffinage de l'ui |
| **00:57** | création du diagramme de classe |
| **01:30** | création du diagramme de classe |
| **00:23** | Ajout du diagramme de classe, ajout de classe non documentée |
| **01:57** | Mise à jour de la documentation, ajout de titres, entêtes, pied de pages, changement d'image plus à jour |
| **00:40** | Sprint review avec M. Viret |
| **00:55** | Mise à jour de la documentation, ajout de titres, entêtes, pied de pages, changement d'image plus à jour |
| **00:20** | Création du temps passé (comme la planification détaillée mais pour l'implémentation) |
| **01:35** | Création du temps passé (comme la planification détaillée mais pour l'implémentation) |
| **00:55** | Création du temps passé (comme la planification détaillée mais pour l'implémentation) |
| **00:40** | Finilasitation de la documentation, ajout du temps passé courant |
| **00:35** | Finilasitation de la documentation, ajout du temps passé courant, commentage |
| **01:00** | création de tests unitaire sur unity |
| **00:53** | création de tests unitaire sur unity |
| **00:42** | ajout du test unitaire dans la documentation, ajout des informations du candidat, experts |
| **01:25** | Problème de préfabs, hiérarchie perdue |
| **01:32** | ajout de la conclusion, résumé, scripts que j'ai bien aimé faire / et que j'ai voulu montrer |
| **00:31** | ajout de la conclusion, résumé, scripts que j'ai bien aimé faire / et que j'ai voulu montrer |
| **00:20** | changement des commentaires dans le code |
| **00:39** | ajout du journal de travail dans la documentation, ajout de noms dans le glossaire |
| **00:55** | adaptation de la documentation pour impression |
| **00:17** | adaptation de la documentation pour impression |
| **00:45** | Ajout des conventions de nommage, + impressions des feuilles |

Le journal de travail est abrégé au temps passé par tâche, et la tâche elle-même, la version complète du journal de travail est disponible en annexe

## 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.



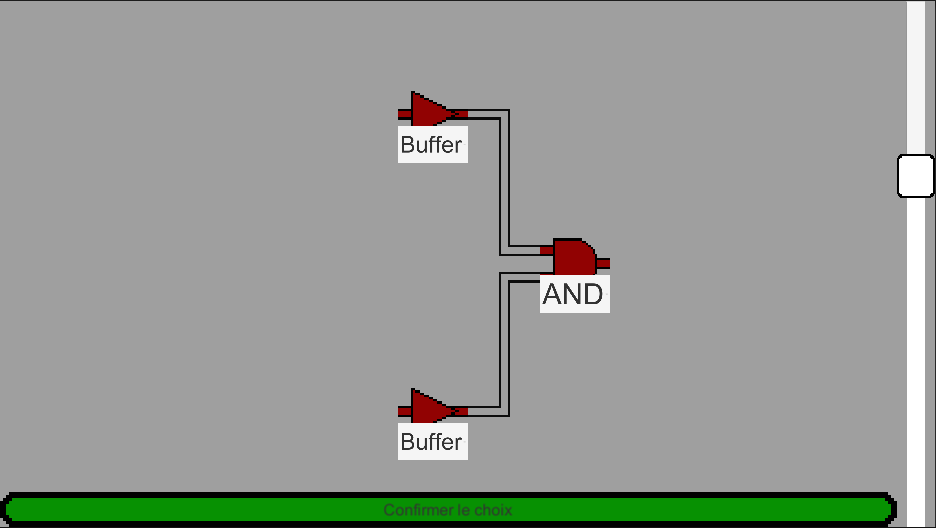
Menu principale

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.

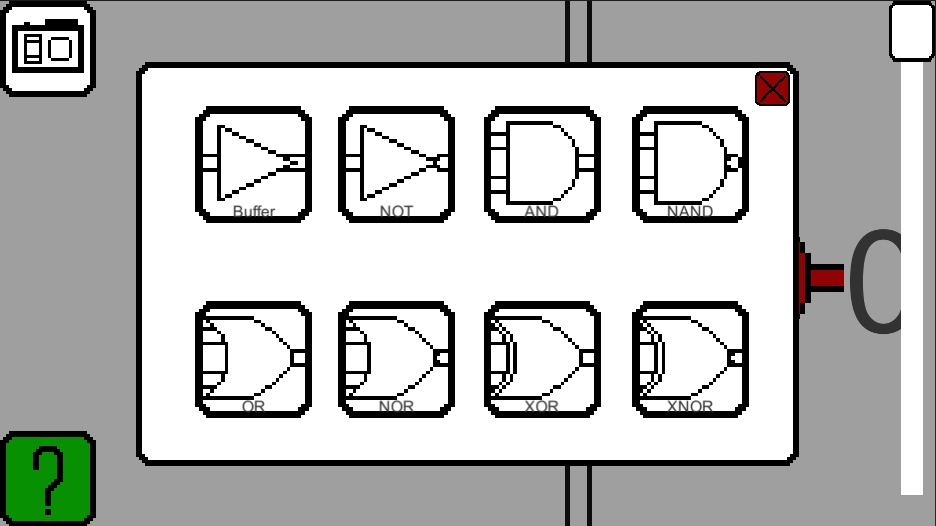
En cliquant sur le bouton rouge, le jeu se ferme



Système de création de puzzle

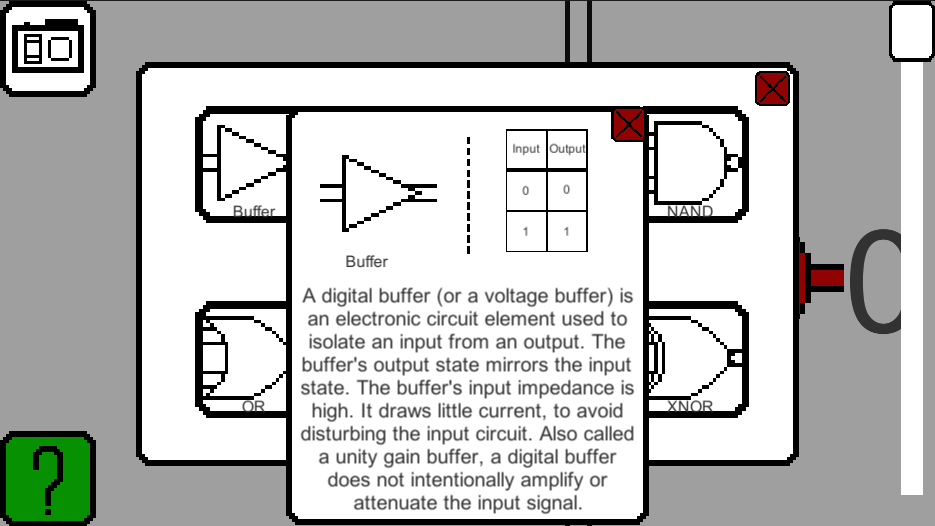
Les listes déroulantes en dessous des portes logiques, permettent de choisir la porte de votre choix, et en cliquant sur « confirmer le choix », votre choix est sauvegardé, est vous passez en mode résolution de puzzle.

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



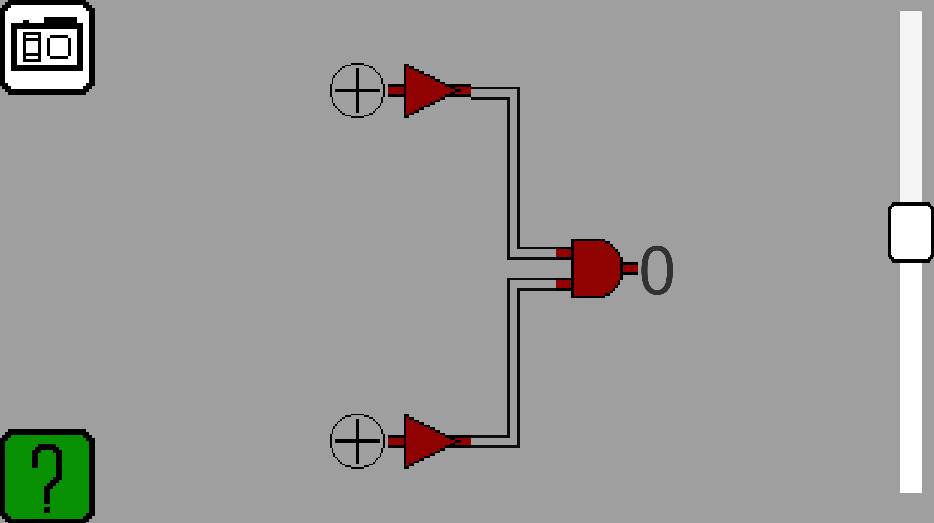
Boutons théorie des portes

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.



Théorie détaillée des portes logiques

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



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

Les anciennes versions de la documentation son disponible en format numérique, comme pour les anciennes versions du jeu lui même

* 1.00
  + Canevas de base
* 2.00
* 3.00
* 4.00
* 5.00
  + Version final de la documentation

### Version du jeu

* 1.00
  + Système de résolution de puzzles
  + Système de génération de puzzle aléatoire
  + Système de création de puzzle
* 2.00
  + Ajout du système de capture d’écran
  + Ajout des tests unitaire
* 3.00
  + Résolution de problèmes avec la création de puzzles
* 4.00
  + Le système de zoom va plus loin

# Table des illustrations

[1 Planification initiale 7](file:///D:\Eurêka!\Eureka\Documentation\Documentation5.00.docx#_Toc104282553)

[1 Planification initiale 7](file:///D:\Eurêka!\Eureka\Documentation\Documentation5.00.docx#_Toc104282554)

[2 Planification détaillée semaine 1 8](#_Toc104282555)

[3 Planification détaillée semaine 2 8](#_Toc104282556)

[4 Planification détaillée semaine 3 9](#_Toc104282557)

[5 Planification détaillée semaine 4 9](#_Toc104282558)

[6 Planification détaillée semaine 5 9](#_Toc104282559)

[7 cas d'utilisation : puzzle aléatoire 10](#_Toc104282560)

[8 cas d'utilisation : puzzle manuel 11](#_Toc104282561)

[9 cas d'utilisation : résolution de puzzle 11](#_Toc104282562)

[10 case d'utilisation : capture d'écran 11](#_Toc104282563)

[11 cas d'utilisation : Théorie des porte logique 12](#_Toc104282564)

[12: Diagramme de flux global de l'application 12](#_Toc104282565)

[13 Diagramme du puzzle manuel 13](#_Toc104282566)

[14 Diagramme du puzzle aléatoire 14](#_Toc104282567)

[15 diagramme de la résolution de puzzle 15](#_Toc104282568)

[16 diagramme pour la théorie des portes logique 16](file:///D:\Eurêka!\Eureka\Documentation\Documentation5.00.docx#_Toc104282569)

[17 Diagramme pour la capture d'écran 16](#_Toc104282570)

[18 maquette menu principale 17](#_Toc104282571)

[19 maquette : création de puzzle 18](#_Toc104282572)

[20 maquette : puzzle aléatoire 18](#_Toc104282573)

[21 théorie : jeu d’icônes de porte logique 19](#_Toc104282574)

[22 théorie détaillé des portes logique 19](#_Toc104282575)

[23 Semaine 1, temps actuel 22](#_Toc104282576)

[24 Semaine 2, temps actuel 23](#_Toc104282577)

[25 Semaine 3, temps actuel 24](#_Toc104282578)

[26 partie du code dans la fonction start 24](#_Toc104282579)

[27 fonction qu change l'output de la porte logique 25](#_Toc104282580)

[28 event qui permet de changer une autre porte logique 25](#_Toc104282581)

[29 l'event visuel dans unity 25](#_Toc104282582)

[30 Diagramme de la classe LogicGate 25](#_Toc104282583)

[31 Diagramme des classe GateManager et CustomGateManager 26](#_Toc104282584)

[32 Diagramme des classe pour la gestion de puzzle 26](#_Toc104282585)

[33 driagramme des classe de l'UI 27](#_Toc104282586)

[34 Version de windows 28](#_Toc104282587)

[35 Version de Unity 28](#_Toc104282588)

[36 Version de Visual Studio 29](#_Toc104282589)

[37 Test unitaire sur le script de capture d'écran 31](#_Toc104282590)

[38 Test unitaires 32](file:///D:\Eurêka!\Eureka\Documentation\Documentation5.00.docx#_Toc104282591)

[39 Test d'intégration sur les portes logiques 31](file:///D:\Eurêka!\Eureka\Documentation\Documentation5.00.docx#_Toc104282592)

[40 Menu principale 39](#_Toc104282593)

[41 Système de création de puzzle 39](#_Toc104282594)

[42 Boutons théorie des portes 40](#_Toc104282595)

[43 Théorie détaillée des portes logiques 40](#_Toc104282596)

[44 Résolution de puzzle, aléatoire ou manuel 41](#_Toc104282597)