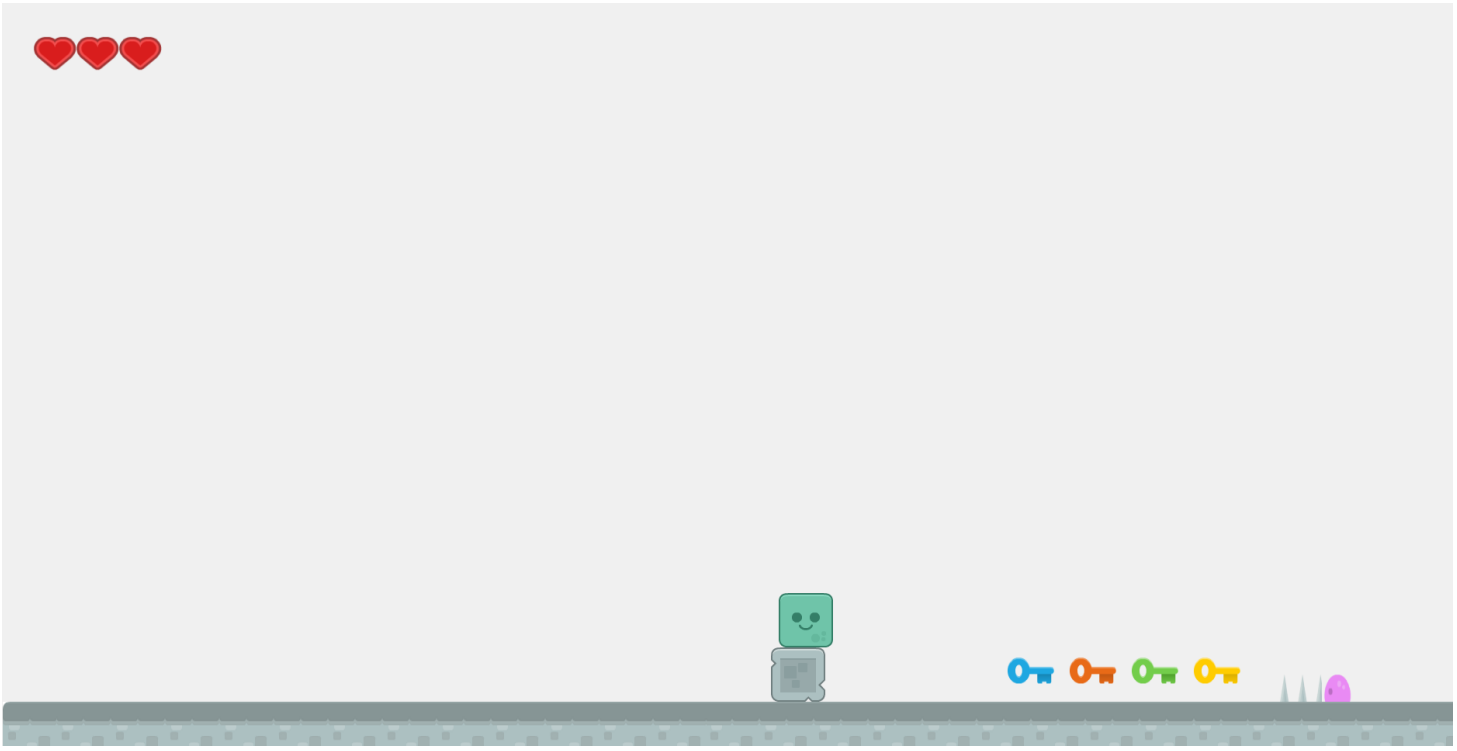
Unlock the door



Duvoisin Guillaume
guillaume.duvoisin@cpnv.ch

Mbayo Guilain
guilain.mbayo@cpnv.ch

SI-CA2a

28.08.2019

Table des matières

1 Introduction

1.1 Cadre, description et motivation

Ce projet s'inscrit dans le cours « Projet en binôme » de la seconde année FPA du CPNV.
Il a pour but la création d'un jeu en 2D (type mario bross) créé en langage C#.
L'utilisateur aura le control d'un personnage pouvant se déplacer de gauche à droite, sauter et devra trouver le moyen d'ouvrir des portes menant aux niveaux suivants.

1.2 Organisation

Elève 1 :

Nom : Duvoisin — Prénom : Guillaume
E-mail : Guillaume.DUVOISIN@cpnv.ch — Tél : 079 443 56 17

Elève 2 :

Nom : Mbayo — Prénom : Guilain
E-mail : Guilain.MBAYO@cpnv.ch — Tél : 079 896 17 45

Client du projet :

Nom : Andolfatto — Prénom : Frédérique
E-mail : Frederique.ANDOLFATTO@cpnv.ch — Tél : 077 206 66 45

Chef de projet :

Nom : Virret — Prénom : Loïc
E-mail : Loïc.VIRRET@cpnv.ch — Tél : -

Répartition générale :

	Eleve 1	*Eleve 2*
Partie administration	*X*	
Partie programmation principale		*X*
Partie programmation menu et level design	*X*	
Maintenance Planning		*X*
Tests du code	*X*	*X*

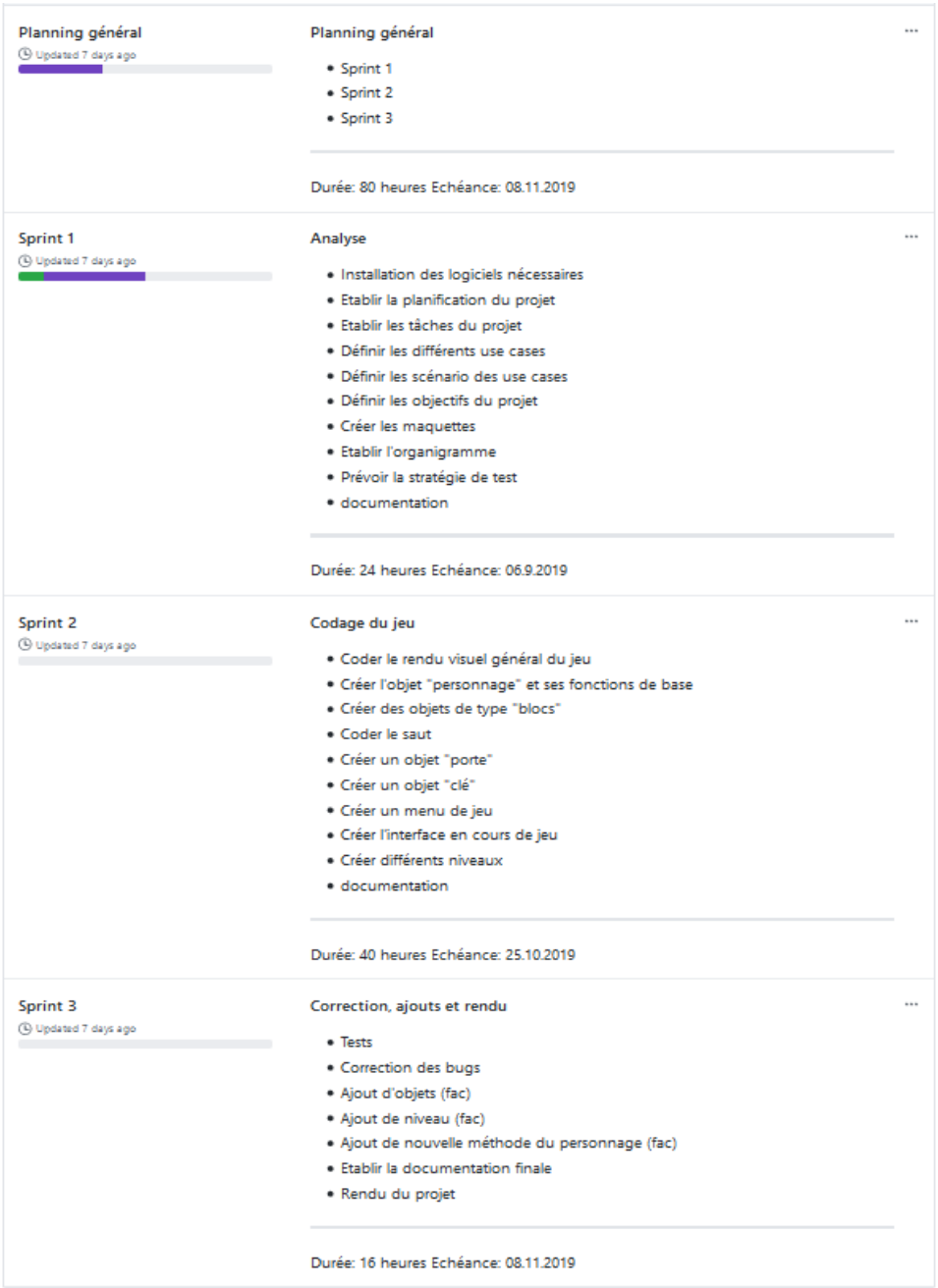
Chaque élève réalisera ses tâches, puis une mise en commun et une validation par l'autre élève sera effectuée. Au début de chaque jour de travail, les élèves se réunissent afin de répartir les tâches du jour et de constater l'état d'avancement du projet.
Note : le journal de travail ne sera pas réaliser, conformément au consignes reçues en cours.

1.3 Objectifs

Nous voulons réaliser un dossier complet et précis du projet de groupe d'ici fin octobre 2019 en nous servant du canevas fourni en cours.
Nous voulons également réaliser l'application « Unlock the doors » selon les contraintes fixées dans le cahier des charges et dans ce document d'ici au 8 novembre 2019.

1.4 Planification initiale

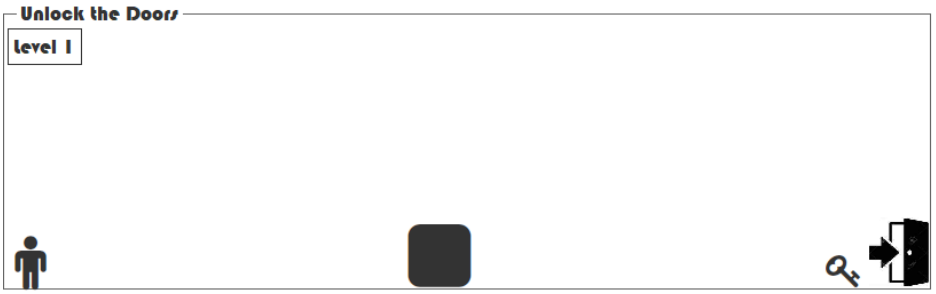
Le projet sera planifié selon la méthode AGILE.

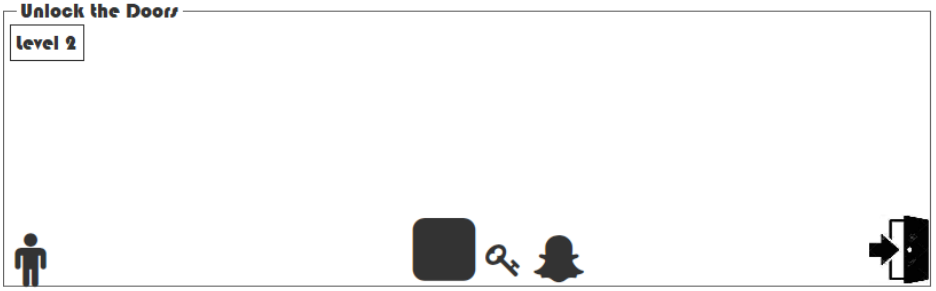


Lien vers le github : https://github.com/GuilDuvoisin/Jeu-en-2d

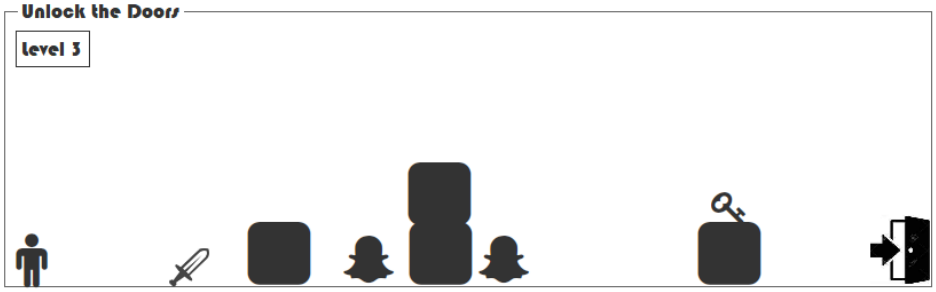
2 Analyse

2.1 Maquettes

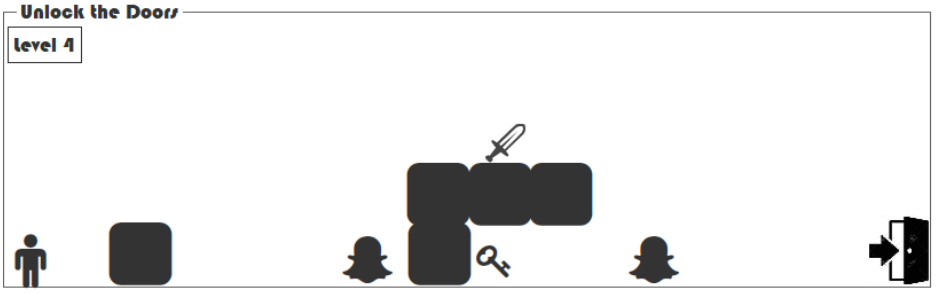


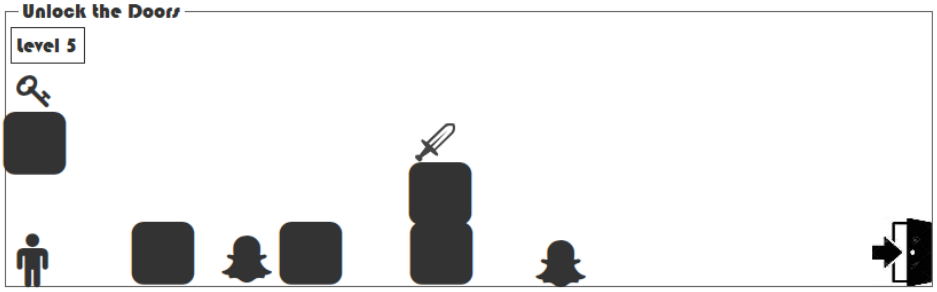


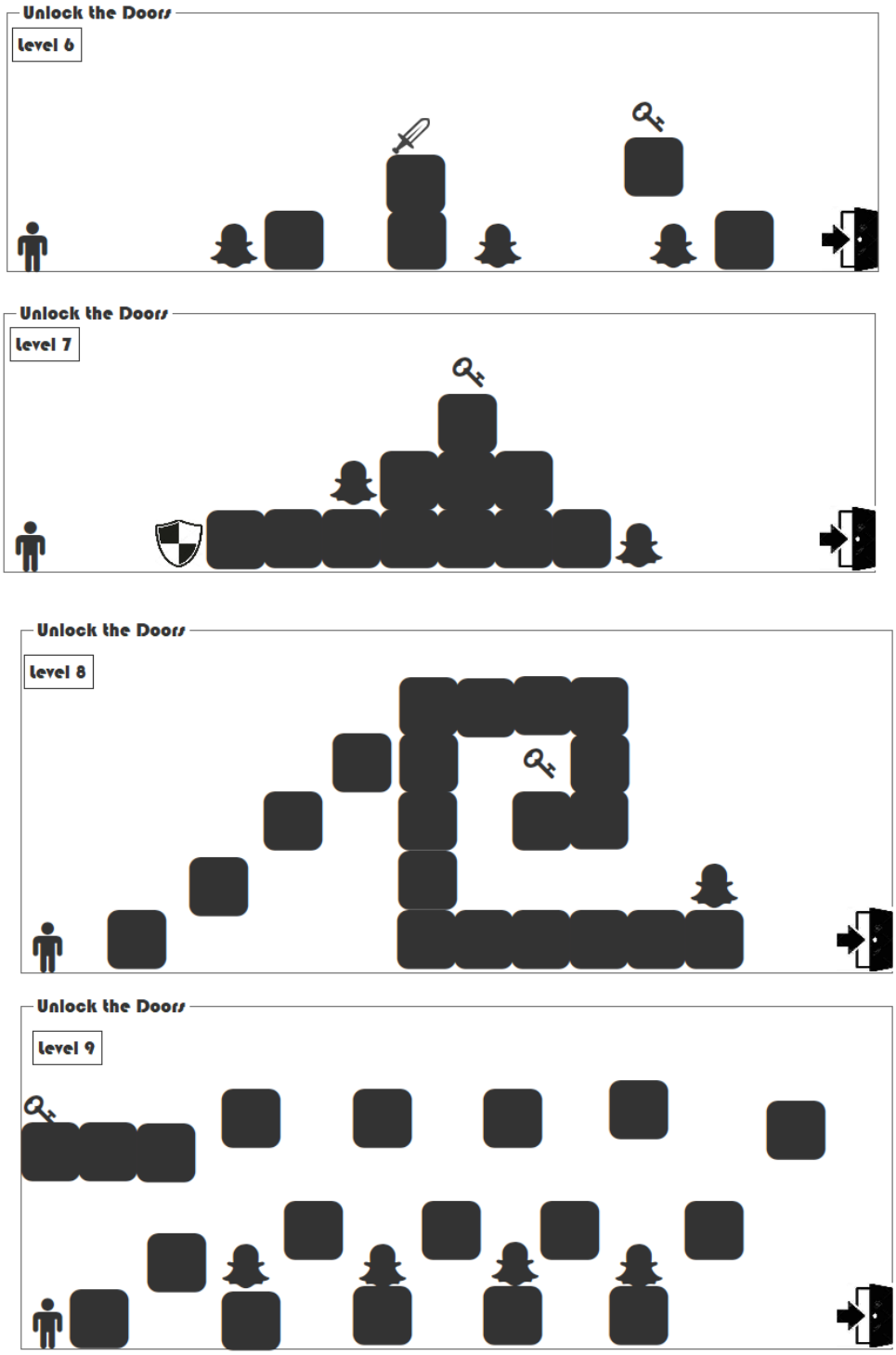
les enemis se deplacent . si le joueur entre en contact avec. il perds un point de vie. Pour se déplacer il faut utiliser les touches. W pour sauter. A pour se déplacer a gache. D pour se déplacer a droit et la touche. E deux fois pour ouvir la porte et passer au niveau suivant. Pour passer au niveau suivant il faut obligatoirement récupres la clé



le joueur peut obtenir differents objets. une épée qui tu les enemis un bouclier qui protège le joueur des enenmis.






Unlock the Doors
Level 6
Unlock the Doors
Level 7
Unlock the Doors
Level 8
Unlock the Doors
Level 9

2.2 Use cases et scénarios

2.2.1 Afficher l'aide

1.1 Lire les règles du jeu
1.2 Afficher les commandes de jeu

2.2.2 Sélectionner le niveau

2.1 Lancer le premier niveau par défaut
2.2 Choisir un niveau plus spécifique

2.2.3 Jouer

3.1 Récupérer un objet
3.2 Passer au niveau suivant
3.3 Afficher l'inventaire
3.4 Tuer un ennemi
3.5 Faire mourir le personnage
3.6 Finir le jeu

2.1.1 Afficher l'aide

1.1 Lire les règles du jeu

Action	Condition particulière	Réaction
J'ouvre l'application		Le menu du jeu s'affiche
Je clique sur « Comment jouer »		Le menu d'aide s'affiche
Je clique sur « Règles du jeu »		Les règles du jeu s'affichent

1.2 Afficher les commandes du jeu

Action	Condition particulière	Réaction
J'ouvre l'application		Le menu du jeu s'affiche
Je clique sur « Comment jouer »		Le menu d'aide s'affiche
Je clique sur « Afficher les commandes »		Les commandes du jeu s'affichent

2.1.2 Sélectionner le niveau

2.1 Lancer le premier niveau par défaut

Action	Condition particulière	Réaction
J'ouvre l'application		Le menu du jeu s'affiche
Je clique sur « Jouer »		Le Premier niveau est lancé

2.2 Choisir un niveau spécifique

Action	Condition particulière	Réaction
J'ouvre l'application		Le menu du jeu s'affiche
Je clique sur « Sélectionner niveau »		Une zone de texte vide s'affiche
J'écris le code du niveau auquel je veux accéder		Le niveau choisi s'affiche
	Le code est erroné	Un message d'erreur s'affiche et la zone de texte est vidée

2.1.3 Jouer

Pour toute la partie « jouer », le tableau commencera au lancement du niveau.

3.1 Récupérer un objet

Action	Condition particulière	Réaction
Je clique sur « Jouer »		Le Premier niveau est lancé
J'utilise les flèches pour me déplacer jusque à l'objet		Le personnage se déplace jusque devant l'objet
Je presse sur la touche d'action		L'objet est récupéré et placé dans l'inventaire
	Je suis trop loin de l'objet	Rien ne se passe

3.2 Passer au niveau suivant

Action	Condition particulière	Réaction
Je clique sur « Jouer »		Le Premier niveau est lancé
J'utilise les flèches pour me déplacer jusque à la porte		Le personnage se déplace jusque devant la porte
Je presse sur la touche d'action		La porte s'ouvre, le personnage y entre et passe au niveau suivant
	Je n'ai pas remplis les conditions requises pour passer au niveau suivant	La porte ne s'ouvre pas

3.3 Afficher l'inventaire

Action	Condition particulière	Réaction
Je clique sur « Jouer »		Le Premier niveau est lancé
Je clique sur la touche « inventaire » (par défaut la touche « i »)		Le Jeu se met en pause et l'inventaire est affiché

3.4 Tuer un ennemi

Action	Condition particulière	Réaction
Je clique sur « Jouer »		Le Premier niveau est lancé
J'utilise les flèches pour me déplacer jusque à l'ennemi		Le personnage se déplace jusque devant l'ennemi
Je presse sur la touche de saut pour retomber sur la tête de l'ennemi		L'ennemi perds des points de vie/est tué
	L'ennemi est immunisé	Rien ne se passe/je perds des points de vie

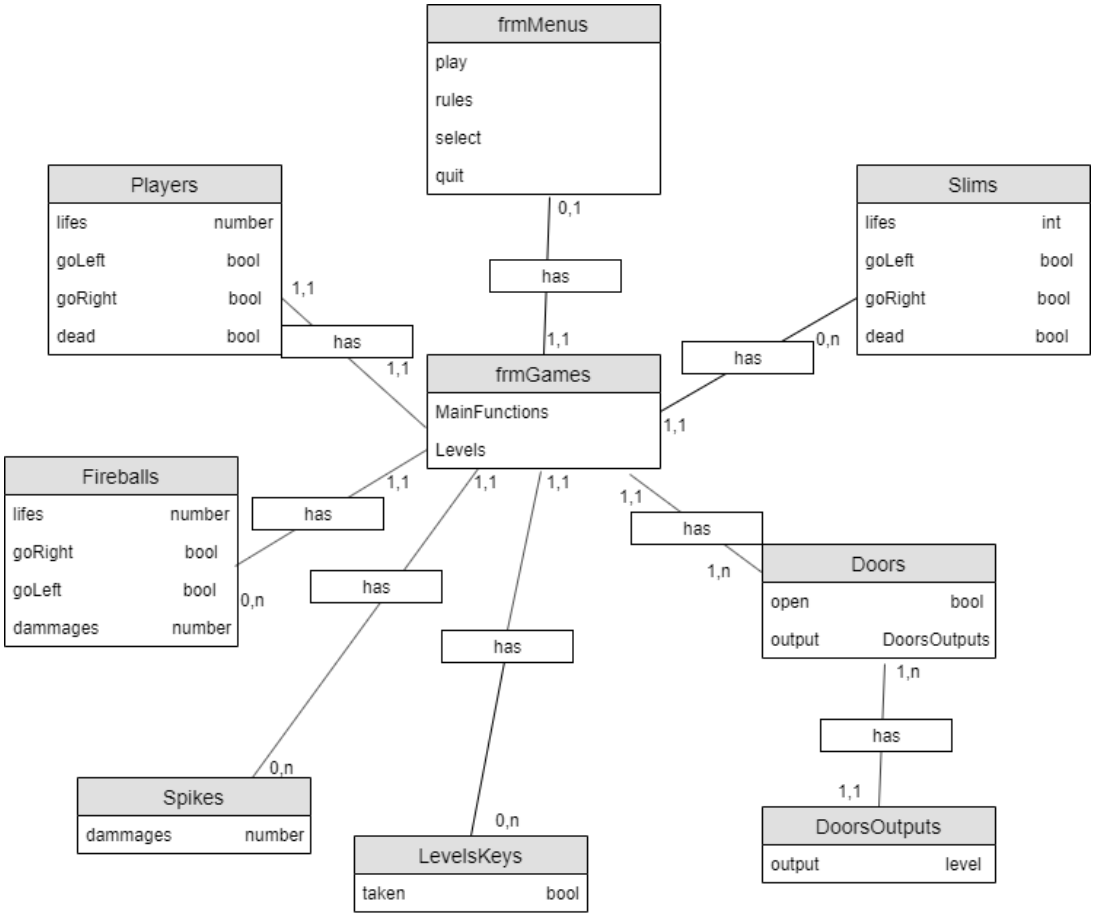
3.5 Faire mourir le personnage

Action	Condition particulière	Réaction
Je clique sur « Jouer »		Le Premier niveau est lancé
Je reçois des dégâts d'un ennemi ou d'un objet dangereux		Le personnage perd des points de vie
	Les points de vie du personnage sont inférieurs aux dégâts reçus	Le personnage meurt. Un écran de transition est affiché et le joueur peut choisir de recommencer le niveau ou de revenir au menu principal

3.6 Finir le jeu

Action	Condition particulière	Réaction
Je clique sur « Jouer »		Le Premier niveau est lancé
Je réussis tous les niveaux		Un écran avec un message de félicitation s'affiche.
J'attends quelques secondes		Un écran avec les crédits s'affiche
J'attends quelques secondes		Le menu principal s'affiche

2.3 Modèle Conceptuel de Données

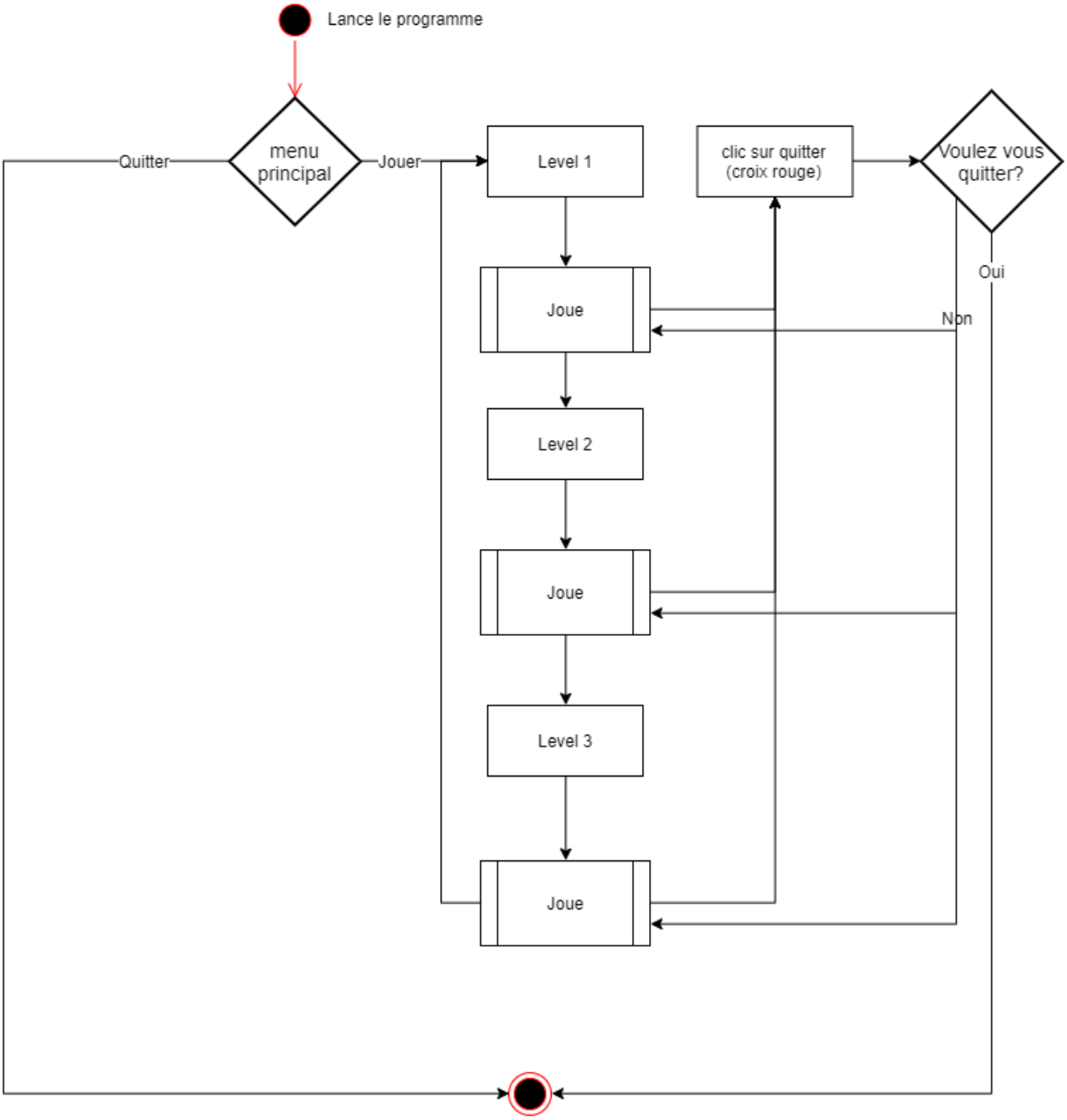


2.4 Stratégie de test

Pour le développement de « Unlock the Door », l'IDE sera Visual Studio sûr l'OS Windows 10. Nous effectuerons des tests fonctionnels régulièrement durant la conception du projet.
Nous utiliserons également un test unitaire afin de contrôler une fonction de base du joueur.
Un ou plusieurs camarades et un ou plusieurs membres de ma famille effectuerons quelques tests fonctionnels, de robustesse et de performance sur différents ordinateurs.

3 Implémentation

3.1 Vue d'ensemble



3.2 Choix techniques

Matériel HW : Ordinateurs du CPNV
OS : Windows 10
Logiciel : Visual Studio
Framework : .NET (Windows form)
Langage : C#

Ces choix ont été fait car il s'agit de l'environnement utilisé durant les cours de c#. Le C# a été choisi afin de consolider les bases acquises en cours, et le framework afin de ne pas avoir à gérer l'affichage de la fenêtre.

3.3 Points techniques spécifiques

3.3.1 Test unitaire

Le test unitaire contrôle la fonction MoveUp de la classe Players.

- Le premier test vérifie le résultat de l'inertie du joueur lorsque le joueur saute et ne touche pas le sol.
- Le second test vérifie le résultat de l'inertie du joueur lorsque le joueur ne saute pas et ne touche pas le sol.
- Le troisième test vérifie le résultat de la vitesse de saut du joueur lorsque le joueur ne saute pas et ne touche pas le sol.
- Le dernier test vérifie le résultat de la vitesse de saut du joueur lorsque le joueur ne saute pas et touche le sol.

La fonction MoveUp fonctionne ainsi :

```
Si le joueur saute :
        Vitesse de saut = - inertie
        Si l'inertie > 4 * vitesse du joueur :
                Inertie -= 1
Si le joueur ne saute pas :
        S'il ne touche pas le sol :
                Si son inertie > 0 :
                        Inertie -= 1
                Vitesse de saut = - inertie
                Si l'inertie > 4 * vitesse du joueur :
                        Inertie -= 1
        Sinon, sa vitesse de saut = 0
```

3.3.2 Interaction avec les objets

Les objets sont tous des enfants de la classe PictureBox.
Chaque objet est déclaré et instancié dans le formulaire frmGame. Toutes les instanciations sont réalisées au niveau de la création du niveau (dans les fonctions Level), puis toutes les interactions avec le jeu (collisions avec le joueur ou avec d'autres objets, déplacement, mort, etc…) sont codées dans différentes fonctions du frmGame (faisant appel aux fonction interne de l'objet), puis utilisées à l'intérieur de

plusieurs BackgroundWorkers. Ces BackgroundWorkers regroupent 4 thèmes de fonctions différentes :

- Celles gérants les déplacements et les collisions principales. (bw_move)
- Celles gérants des actions secondaires, comme les collisions entre le joueur et les piques. (bw_action)
- Celles gérants des comptes à rebours pour les animations et pour la mort du joueur. (bw_delay)
- Celles gérants l'affichage des animations. (bw_animation)

3.3.3 Gestion des collisions

Les collisions sont gérées à l'intérieur d'un BackgroundWorker. Ce choix a été fait afin que les calculs effectués dans le jeu soient séparés sur différents threads et évite ainsi certains ralentissements au niveau de l'affichage qui aurait amoindrit la qualité de l'expérience utilisateur. Les quatre BackgroundWorkers utilisés sont décrits plus en détails dans le chapitre précédent.

3.3.4 Architecture du code

Le code est construit en utilisant la programmation orientée objet (POO).
Les attributs de chaque objet sont privés, puis ceux auxquels on doit pouvoir accéder depuis d'autre fichier possède une propriété donnant l'accès en get et/ou en set selon les besoins. Les autres ne sont accessibles que dans leurs classes.
L'architecture étant construite autour d'un fichier principal (frmGame), les fonctions contenues dans ce dernier sont privées, tandis que la plupart de celles contenues dans d'autres fichier sont publiques afin d'être accessibles au frmGame.
La fonction OnFormClosing (déclenchée juste avant que le formulaire ne se ferme) a été surchargée dans frmGame afin qu'une fenêtre pop-up demande à l'utilisateur s'il veut vraiment quitter le jeu.
Cette surcharge met le jeu en pause le temps que l'utilisateur fasse son choix, puis, en fonction de ce choix relance le jeu ou ferme le formulaire.

3.3.5 Interface utilisateur

L'interface a été conçue afin de montrer toutes les informations utiles au joueur. Ceci inclut :

- Ses vies
- Les clés ramassées
- L'écran de jeu

De la place en dessus de l'écran de jeu a été laissée en prévision d'ajout de fonctionnalités supplémentaires.
Un accès à l'aide en cours de jeu était prévue, mais n'aura finalement pas été implémentée, toujours pour manque de temps. L'utilisateur devra donc se reposer sur le mode d'emploi qui lui sera fourni en annexe du jeu.

Lors du lancement de l'application, le jeu est automatiquement ouvert en plein écran, ce qui permet une meilleure immersion du joueur dans le niveau.
La première interface à laquelle est confrontée le joueur est celle du menu. Deux boutons sont grisés car non implémentés pour l'instant, mais ils sont à disposition pour une éventuelle version 2.0. Le joueur a la possibilité de jouer et de se retrouver au niveau un (déterminé dans la fonction Menu de frmGame) ou de quitter, ce qui ferme l'application sans passer par la pop-up demandant si l'on veut réellement quitter.
Le rendu visuel est affiché dans un windows form. Rétrospectivement, ce choix n'est pas optimal pour la création d'un jeu vidéo. En effet, le fait d'utiliser un formulaire déjà fait et des PictureBox au lieu de générer soi-même une fenêtre et de dessiner ses images dedans a entraîné certains problèmes de performance au niveau de l'affichage ainsi que des contraintes au niveau du code qui m'ont forcé à chercher des solutions alternatives et parfois un peu « bricolées ».
Ce choix semblait pertinent au début car nous avions déjà travailler sur les windows form en cours, mais si le projet était à refaire, nous ne nous dirigerions pas sur cette option.

3.3.6 Gestion du travail

La gestion du travail en équipe a été quelque-peut compliquée durant ce projet. Cela s'explique par les problèmes de santés d'un des membres du groupe, provoquant plusieurs absences durant le projet, puis une absence définitive à la fin.
La répartition prévue au départ aurait été moitié moitié pour la documentation et un tiers/deux tiers pour la programmation. Le déséquilibre prévu dans la programmation était dû à des différence d'affinité avec ce thème.
Le résultat réel a été bien plus déséquilibré, ce qui nous a forcé à revoir à la baisse les objectifs que nous nous étions fixés afin de pouvoir tout de même rendre un travail que nous espérons correct.

3.4 Livraisons

Rendu final : 08.11.19 23 :59

Livrable :

- Rapport
- Mode d'emploi
- Documentation DoxyGen
- Cahier des charges
- Code
- Executable (Unlock_the_Door.exe)

4 Tests

4.1 Tests effectués

Scénario	Développeur
1.1 Lire les règles du jeu	Non implémenté pour manque de temps
1.2 Afficher les commandes	Non implémenté pour manque de temps
2.1 Lancer le premier niveau	Fonctionne
2.2 Choisir un niveau spécifique	Non implémenté pour manque de temps
3.1 Récupérer un objet	Fonctionne(le récupère et le place dans la zone d'inventaire, pas besoin d'appuyer sur la touche d'action)
3.2 Passer au niveau suivant	Fonctionne
3.3 Afficher l'inventaire	Fonctionne (pas de touche d'inventaire)
3.4 Tuer un ennemi	Fonctionne (avec les boules de feu)
3.5 Faire mourir le joueur	Fonctionne
3.6 Finir le jeu	Non implémenté pour manque de temps
Le joueur interagit avec les éléments du jeu	Fonctionne
Le joueur peu tirer des boules de feu	Fonctionne
La porte ne s'ouvre que lorsque la bonne clé est dans l'inventaire	Fonctionne
Une pop-up apparaît lorsque l'on veut quitter le jeu	Fonctionne

4.2 Erreurs restantes

Les erreurs restantes sont le résultat d'un manque de temps à la fin du projet, dû aux problèmes de santé d'un membre du groupe.
Les boules de feu ont une durée de vie différentes en fonction de l'emplacement du joueur, mais le temps ne nous a pas permis de corriger cette erreur.

5 Conclusions

Les objectifs de ce projet sont partiellement atteints. En effet, Le jeu est fonctionnel et la documentation est réalisée, en revanche les fonctionnalités prévues au début ne n'ont pas toutes été implémentées.

Ces différences entre le résultat attendu et le résultat réel est entre autre dû aux problèmes de santé de l'un des membres du groupe, qui ont conduit à de multiples absences ainsi qu'à un abandon deux semaines avant le rendu du projet. Les ambitions initiales ont donc dû être revues à la baisse, et certaines idées abandonnées.
D'autre différences ont comme source des discussions entre le groupe, le client et le chef de projet. Par exemple, il n'y a pas besoin d'appuyer sur la touche d'action pour ramasser les clés car le client a dit au groupes que ça n'était pas nécessaire.
Enfin, certaines idées ont été modifiée en cours de route par le groupe, par souci de simplicité d'utilisation. Par exemple, comme les seuls objets stockés dans l'inventaire sont les clés, il nous a parût plus logique et intuitif de les afficher directement en haut de l'écran au lieu de devoir utiliser une touche spéciale pour afficher un inventaire. Cela n'aurait fait que donner plus de touches à mémoriser à l'utilisateur sans pour autant améliorer son expérience de jeu.

Nous pouvons retirer à la fois du positif et du négatif de ce projet. Pour commencer par le négatif nous avons :

- Les problèmes de santés inattendus qui nous ont pénalisés au niveau du nombre d'heures de travail effectifs.
- Les connaissances en programmation qui sont arrivées au fur et à mesure du projet, ce qui a donc empêché de construire une structure de code optimale.

Et en positif, nous avons :

- Les connaissances acquises grâce à ce projet.
- Un résultat fonctionnel gratifiant.

La répartition des groupes est efficace dans le sens qu'il équilibre les capacités de chaque groupe de la classe, en revanche il génère peut-être également des attentes différentes à l'intérieur des groupes, ce qui pousse (dans certains cas) l'un ou l'autre de membres à l'insatisfaction et place donc un déséquilibre dans la répartition du travail.

Si ce projet devait avoir une suite, la priorité serait de remodeler la structure de base du code afin de le rendre plus propre, plus compréhensible et mieux optimisé. Il faudrait ensuite rajouter les fonctionnalités qui était prévue à la base. Une fois ces deux tâches effectuées, le projet serait au stade qui aurait été souhaité pour la version 1.0.
Il serait ensuite possible de se concentrer sur l'ajout d'autres ennemis (ennemis volants par exemple), d'autres objets utilisables (tels que des trampolines, des armes, de l'eau, des décors, etc…), du son, et surtout d'autres niveaux plus longs et élaborés !

6 Annexes

6.1 Documents annexes

- Mode d'emploi
- Documentation DoxyGen

6.2 Sources – Bibliographie

- *https://social.msdn.microsoft.com/Forums/vstudio/en-US/9c85d1f3-66a1-42be-9a6e-175fc3c6e739/remove-picture-box-controls?forum=csharpgeneral*
- *https://openclassrooms.com*
- https://docs.microsoft.com
- *M. Loïc Virret*