


SHOOTABBY

Shoot me up!

10 JANVIER 2025

ABIGAËL PÉRISSET
Enseignant : Xavier Carrel

Table des matières

1 Analyse préliminaire

1.1 Introduction

Dans le cadre de ma formation d'informaticienne à L'ETML, chaque étudiant est amené à réaliser un jeu de tir en C# sur Windows. Ce travail s'étale sur un total de 80 périodes, elles-mêmes réparties sur le premier trimestre.

1.2 Objectifs

L'objectif de ce projet est de mettre en pratique nos acquis du module "ICT-320"à travers la création d'un jeu vidéo. Pour ce faire, nous devions programmer un jeu de tir du genre "Shoot'em up" et fournir un rapport détaillant les attentes et le résultat final.

Le code produit doit respecter les points suivants :
- Respect des normes imposée par l'établissement
- Un code clair et précis
- Un code optimisé grâce à une bonne utilisation des structures
- Des tests unitaires
- Un code commenté là où c'est nécessaire

Le résultat final du jeu doit comprendre les fonctionnalités suivantes :
- Un système de niveaux de jeu
- Le joueur
- Des ennemis
- Des obstacles

1.3 Gestion de projet

Pour la gestion de ce projet, j'ai opté pour la méthode "agile scrum". Cette technique consiste à planifier des « sprints », des éléments du projet complets, qui peuvent être d'une durée de quelques heures à plusieurs jours. Les sprints représentent une partie du programme à réaliser et ont l'avantage d'être très flexibles et de permettre, à tout moment, d'ajouter ou de retirer certaines fonctionnalités du projet. Pour ce projet, il a été décider de n'utiliser qu'un seul sprint pour la durée totale du projet. Ainsi la méthode agile est facile à mettre en place et offre une grande liberté dans le choix de l'avancement du projet.

Ainsi, je pouvais visualiser l'ensemble des tâches effectuées et des tâches à réaliser, ce qui m'a permis d'estimer le temps de travail nécessaire à chaque tâche afin de m'organiser.

Pour appliquer au mieux la méthode agile et l'utilisation des sprints, j'ai eu recours au site IceScrum

2 Gameplay

2.1 Généralités

Les différents aspects du gameplay du jeu ont été imposés par le client. Concrètement, le jeu demandé est un « Shoot'em up ». C'est un jeu de tir à la troisième personne en 2D, similaire aux classiques du genres que sont « Space Invader » ou encore « Asteroids ».

2.2 Le joueur

Le joueur possède un sprite unique en jeu. Il peut se déplacer dans quatre directions ; haut, gauche, bas et droite. Les touches utilisées sont respectivement W, A, S et D.

Au début d'une partie, le joueur possède 5 points de vie qu'il perd en entrant en collision avec un ennemi ou avec un projectile ennemi. Une fois les 5 points de vies perdus, le joueur meurt. Il peut mourir 3 fois avant que la partie ne soit terminée.

Le joueur peut tirer des projectiles dans les mêmes directions que les déplacements à l'aide des flèches directionnelles. Le joueur peut aussi ramasser une arme alternative qui modifie la manière de tirer du joueur. Contrairement à l'arme de base qui a des munitions infinies, cette dernière possède des munitions limitées, vingt. Ces dernières peuvent, elles aussi, être ramassées au sol.

2.3 Les ennemis

Il existe deux types d'ennemis ayant chacun leur propre sprite. Chaque ennemi possède 10 points de vie et disparait en mourant.

Le premier type d'ennemi peut se déplacer en poursuivant directement le joueur. Le deuxième type d'ennemi reste dans un coin du jeu pour tirer sur le joueur.

2.4 Les obstacles

Les deux types d'obstacles définis sont les rochers et les barricades. Tous deux peuvent bloquer les tirs (du joueur ou des ennemis). Ils sont aussi impossibles à traverser, autant pour le joueur que pour les ennemis. Les rochers sont indestructibles par le joueur tandis que les barricades sont détruites au troisième projectile intercepté.

2.5 Les objets ramassables

Lors d'une partie, le joueur peut ramasser des objets qui apparaissent aléatoirement sur le jeu :

- Le soin : redonne la totalité de ses points de vie au joueur

- L'arme alternative : apparaît avec des munitions limitées
- Les paques de munitions pour l'arme alternative

2.6 Les niveaux

Deux niveaux sont actuellement attendu pour le jeu.

Le premier niveau doit posséder plusieurs obstacles pour aider le jeu à se déplacer loin des ennemis. Il y aura davantage de rochers que de barricades. Les ennemis apparaissent par vague et chaque vague d'ennemi apparaît aléatoire sur le jeu au début du niveau et après que le joueur a tué tous les monstres.

Le deuxième niveau les monstres apparaissent toujours par vague du début à la fin. La première différence est que les monstres apparaissant toutes les 20 secondes ou si tous les monstres de la vague sont morts. La deuxième différence vient du nombre d'obstacle moins nombreux que sur le premier niveau et il n'y a que des barricades.

2.7 Concept

2.7.1 Diagramme de classe des associations

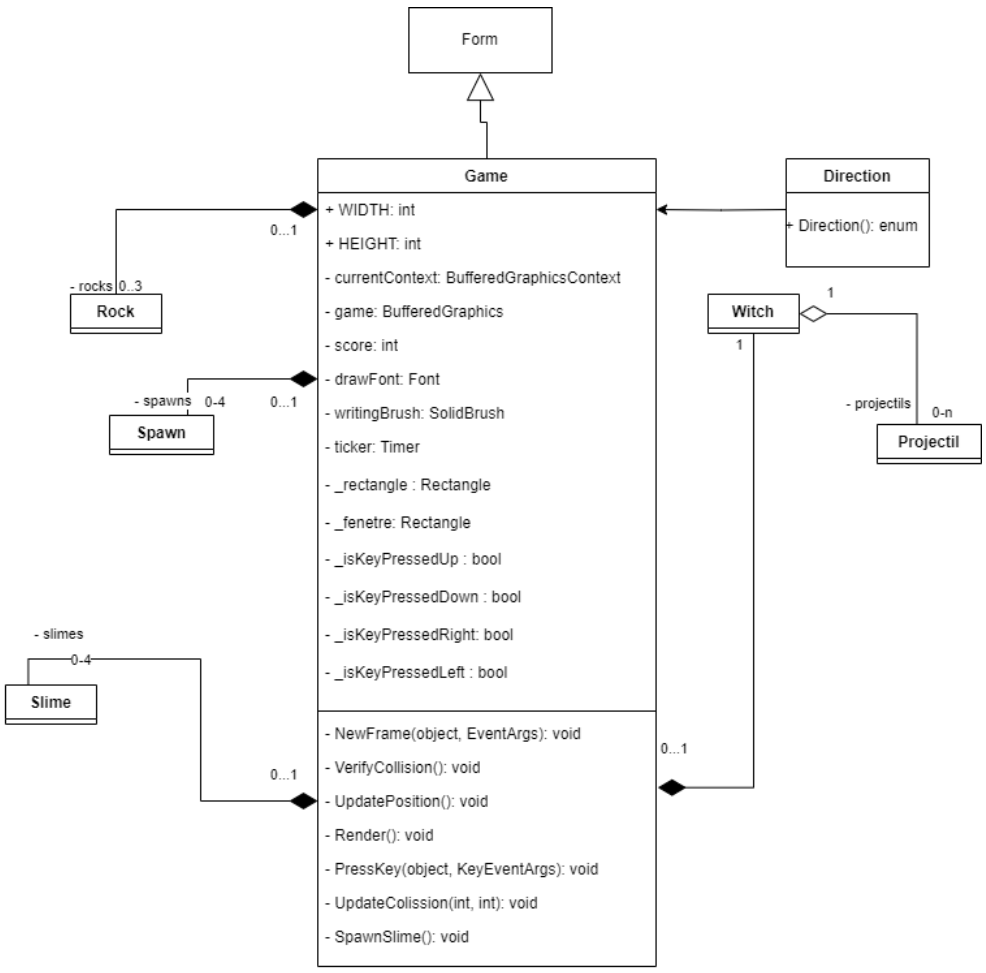

Figure 2 Diagramme UML des associations

2.7.2 Diagramme de classe de l'héritage

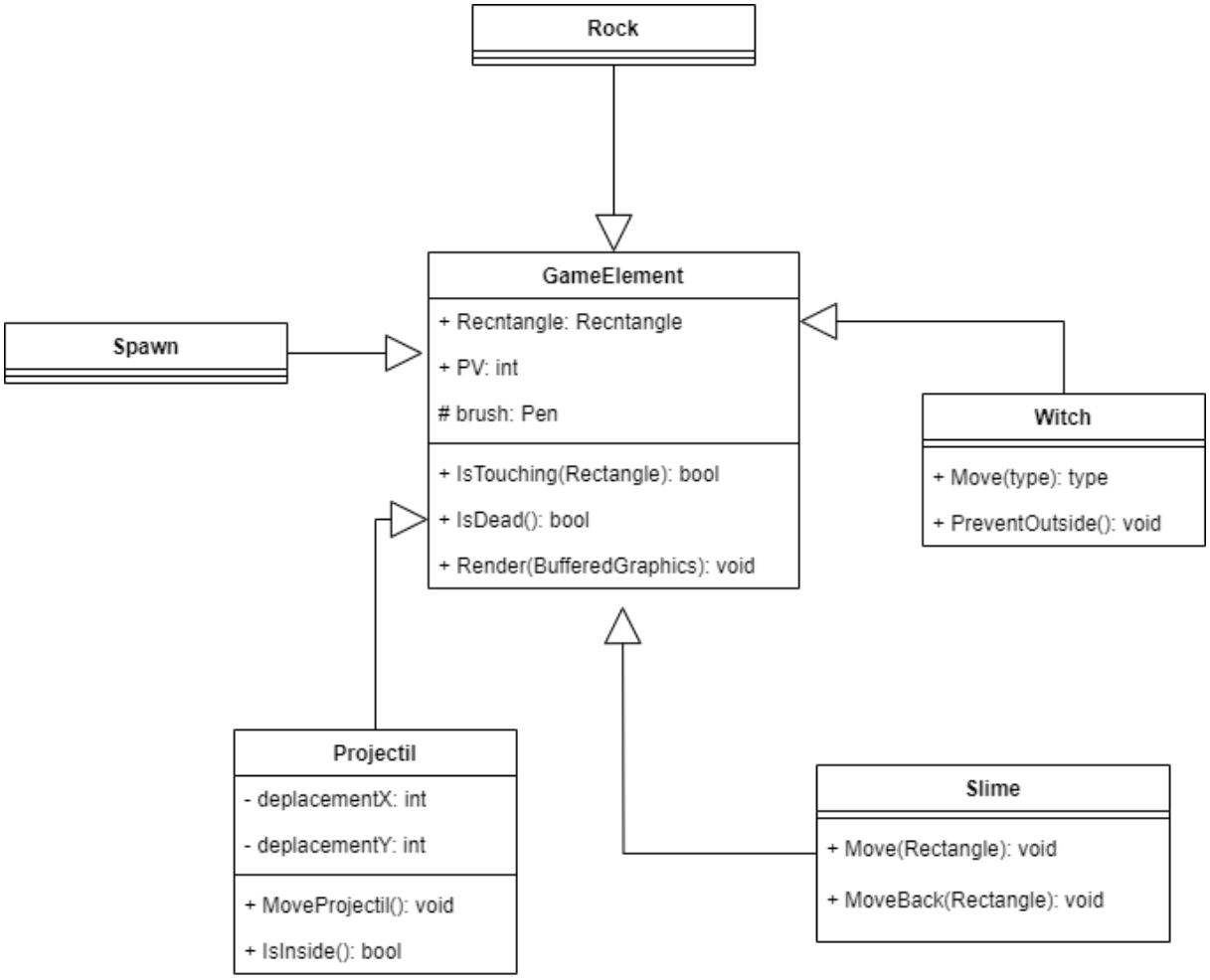

Figure 2 UML - Héritage

2.7.3 Diagramme d'état

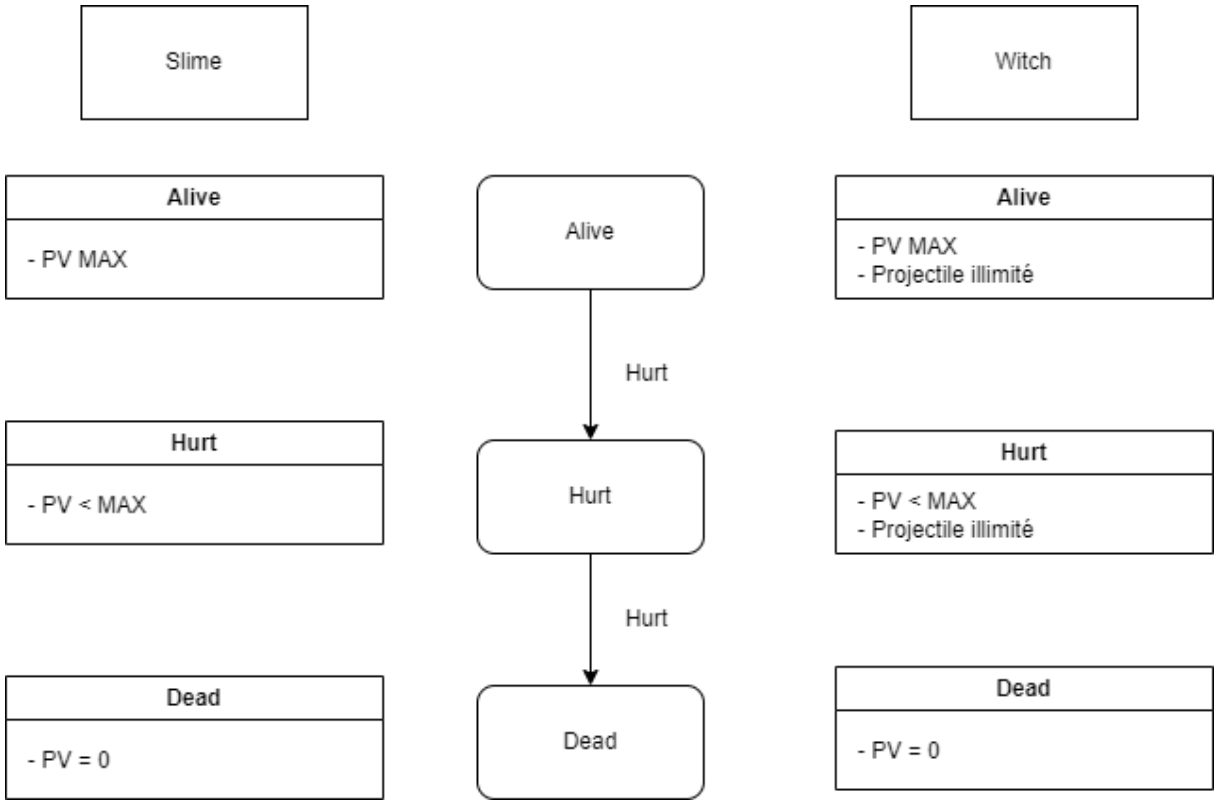

Figure 3 Diagramme Etat - Witch & Slime

2.8 Analyse fonctionnelle

2.8.1 Déplacement du joueur

En tant que joueur je veux déplacer mon personnage

Tests d'acceptance :	
Déplacements du personnage	Quand j'appuie sur les touches directionnelles, mon personnage se déplace dans la direction de la touche appuyée
Collisions aux bordures	Quand je déplace mon personnage et que ce dernier touche une bordure de l'écran, mon personnage s'arrête
Collision aux rocher	Quand je déplace mon personnage et que ce dernier touche un caillou, mon personnage s'arrête

2.8.2 Lancement du niveau

Quand je lance le niveau, mon personnage, les ennemis et les décors apparaissent.

Tests d'acceptance :	
Le joueur apparait	Quand je lance le niveau, mon personnage apparait au centre de l'écran (voir maquette)
Vie complète joueur	Quand je lance le niveau, la barre de vie de mon personnage est pleine (voir maquette)
Décors rocher	Quand je lance le niveau, je vois 3 rochers sur l'écran (voir maquette)
Les ennemis slimes apparaissent	Quand je lance le niveau, je vois 4 slimes à l'écran
Vie complète ennemi	Quand je lance le niveau, les ennemis ont leur barre de vie complète

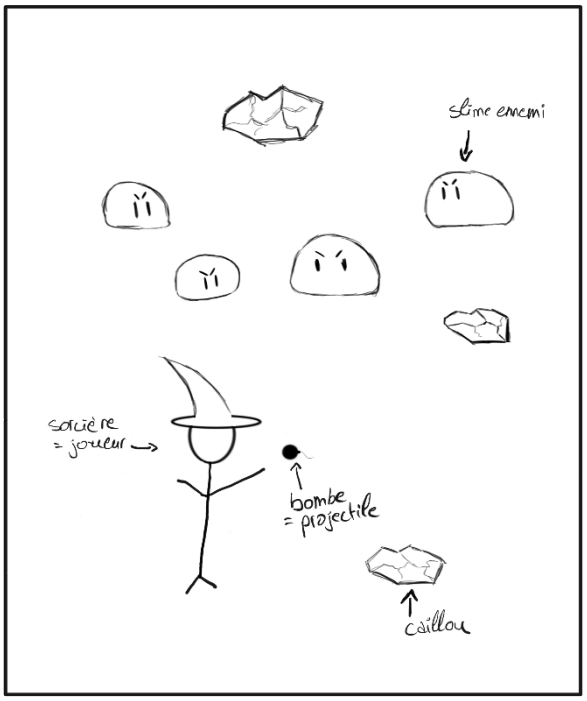

Figure 4 Maquette début de partie

2.8.3 Apparition ennemie

En tant que joueur je veux des ennemis à l'écran Pour jouer

Tests d'acceptance :	
Apparition ennemi	Quand je tue une vague d'ennemi, une nouvelle vague d'ennemi apparait
Vie au maximum	Quand un ennemi apparait, sa barre de vie est au maximum

2.8.4 Attaque du joueur

En tant que joueur je veux attaquer les ennemis Pour les tuer

Tests d'acceptance :	
Lancement des projectiles	Quand j'appuie sur les touches A, S, W, D, mon personnage tire dans la direction de la touche.
Distance projectile	Quand je tire mon projectile va en ligne droite jusqu'à la bordure de l'écran si aucun obstacle ou ennemi.
Tirer sur le rocher	Quand je tire sur un rocher mon projectile est arrêter
Cooldown	Je dois attendre 1 secondes avant de pouvoir tirer à nouveau
Rester appuyer	Quand je reste appuyer sur une touche de tire, un seul projectile dans la direction de la touche est tiré.

2.8.5 Déplacement slime

En tant que joueur je veux que les slime se rapproche de moi

Tests d'acceptance :	
Joueur immobile	Quand je suis immobile, le slime se rapproche de mon joueur
Joueur mobile	Quand je me déplace, le slime continue de se rapprocher de mon joueurs
Collision des slimes	Quand les slimes se déplacent, ils ne peuvent pas se superposer

2.8.6 Mort de la sorcière

Quand les points de vie de la sorcière sont à zéro, le jeu s'arrête

Tests d'acceptance :	
Arrêt du jeu	Quand la sorcière a 0 point de vie, le jeu s'arrête
Arrêt du tire	Quand la sorcière a 0 point de vie, appuyer sur les touches de tires ne provoquent pas de tire.
Arrêt du mouvement	Quand la sorcière a 0 point de vie, appuyer sur les touchent pour bouger la sorcière, rien ne se passe.

2.8.7 Disparition ennemie

Quand la vie d'un ennemi tombe à zéro, ce dernier disparait de l'écran

Tests d'acceptance :	
Disparition	Si le slime a 0 de point de vie, il disparait de l'écran

Incrémentation	Quand l'ennemi disparait, le score augmente

2.8.8 Collision des slimes avec les rochers

En tant que joueur, je veux que les slimes ne traversent pas les rochers pour les ralentir et les forcer à contourner l'obstacle.

	Tests d'acceptance :
Collision du slime	Quand un slime touche un rocher, il arrête d'avancer dans la direction du rocher.
Mouvement du slime	Quand un slime touche un rocher, il peut se déplacer dans les directions opposées au rocher si le joueur s'est déplacé.

2.8.9 Affichage des munitions

En tant que joueurs, je veux voir s'afficher le nombre de munitions présentent dans mon arme pour savoir quand le chargeur est vide.

	Tests d'acceptance :
Chargeur plein	Quand le joueur lance le jeu, les munitions affichées sont au maximum.
Diminutions munitions	Après un tir du joueur, le nombre de munitions affichées diminuent d'un.
Chargeur vide	Quand le joueur ne peut plus tirer, le nombre de munitions affichées est de 0.

2.9 Stratégie de test

Pour tester mon shoot'em up, j'ai effectué des tests directement en jeu. Je lançais l'exécutable et testais les fonctionnalités une à une.
Ainsi, je testais chaque nouveau bout de code implémenté. Je commençais par tester la fonction elle-même, puis je tentais de trouver les limites du code en cherchant les cas spécifiques liés à cette fonctionnalité dernièrement ajoutée à mon code.
Je n'ai donc pas utilisé la méthode de test unitaire sur ce projet. De plus, j'ai pu faire tester mon jeu à mes camarades de classe qui se sont amusé à me signaler les erreurs et bug.

3 Réalisation

3.1 Points de design spécifiques

3.1.1 Game.cs

La classe “Game” est la classe chargée de gérer tout le fonctionnement du jeu. Il gère l'affichage de tous les éléments du jeu, leurs déplacements mais aussi leurs collisions et leurs suppressions de l’écran. C’est aussi dans cette classe je gère les évènements au moment où le joueur appuie sur une touche pour se déplacer ou pour tirer dans une direction.
Pour permettre au joueur de tirer dans différentes directions, une fonction nommée « Fire » a été ajoutée à la classe Witch. Cette fonction utilise une liste de directions correspondant aux flèches directionnelles pour identifier la direction choisie par le joueur. Lorsqu'une touche est pressée, la direction correspondante est sélectionnée et un projectile est généré à partir de la position actuelle du joueur. Cette implémentation permet de centraliser la logique des tirs dans une classe unique qui se nomme « direction » présente dans « Direction.cs », simplifiant ainsi la maintenance et les extensions potentielles du code.
Grâce à cette approche, il est également possible d’ajouter des types de tirs supplémentaires ou des comportements spécifiques en fonction des conditions de jeu.

```
private void NewFrame(object sender, EventArgs e)
{
    if (!_witch.IsDead())
    {
        VerifyCollision();
        UpdatePosition();
        Render();
    }
}
```

3.1.2 GameElement.cs

La classe “GameElement” est la classe qui sert de représentation pour tous les objets qui peuvent apparaître sur le jeu. Dans le langage de programmation, « GameElement » est la classe parent de la sorcière, des projectiles, des slimes et des rochers.
Chaque objet est géré par un objet Rectangle pour l’affichage, la position et ses déplacements. Chaque objet possède aussi des points de vies pour gérer son affichage ou non.
Cette classe me permet aussi de gérer la collision et la fin de vie d’un objet avec les fonctions “IsTouching”, qui prend en paramètre un autre Rectangle, et “IsDead”.

3.1.3 Les méthodes de déplacement

Les classes Witch, Slime et Projectil ont leur propre fonction “Move”. En effet, bien que mon code puisse être similaire dans mes classes “Witch” et “Projectil” par exemple, les deux fonctions n’ont pas les mêmes paramètres. Il était préférable de ne pas avoir une fonction de déplacement en héritage car les objets qui ne bougent pas auraient hérité pour rien de cette fonction. C’est le cas de l’objet spawn qui ne contient et n’as pas besoin de mobilité. Mais utiliser une interface “IMovable” par exemple aurait pu être une solution.

```
public void Move(int deplacementX, int deplacementY)
{
    _rectangle.X += deplacementX;
    _rectangle.Y += deplacementY;
}
```

Figure 5Fonction Move Witch.cs

```
public void MoveProjectile()
{
    _rectangle.X += _deplacementX;
    _rectangle.Y += _deplacementY;
}
```

Figure 6 Function Move Projectil.cs

```
public void Move(Rectangle sorciere)
{
    if (_rectangle.X < sorciere.X)
    {
        _rectangle.X += 1;
    }
    else if (_rectangle.X > sorciere.X)
    {
        _rectangle.X += -1;
    }
    if (_rectangle.Y < sorciere.Y)
    {
        _rectangle.Y += 1;
    }
    else if (_rectangle.Y > sorciere.Y)
    {
        _rectangle.Y += -1;
    }
}
```

Figure 7 Function Move Slime.cs

3.1.4 Spawn.cs

Cet élément a été ajouté pour me faciliter la gestion de l'apparition des ennemis. N'arrivant pas à utiliser l'entièreté de la zone de jeu comme zone d'apparition, j'ai créé quatre zones plus petites. Ces zones ont une taille et une position fixe et apparaissent au début du jeu. Ainsi lors de la création des slimes, ces derniers sont initialisés à la position des spawn.

```
private void SpawnSlime()
{
    foreach (Spawn zone in _zones)
    {
        Slime slime = new Slime(zone.Rectangle.X, zone.Rectangle.Y);
        _slimes.Add(slime);
    }
}
```

Figure 8 Apparition des slimes dans une zone

3.1.5 Slime.cs

Une mécanique spécifique a été mise en place pour gérer le mouvement des slimes lorsqu'ils rencontrent des obstacles tels que des rochers. Par défaut, les slimes utilisent une fonction d'avancement qui leur permet de se déplacer en direction de la sorcière, leur cible principale. Cependant, lorsqu'un slime entre en collision avec un rocher, il active une fonction de recul qui le fait revenir à sa position précédente. Ce comportement évite que les slimes restent bloqués ou traversent les obstacles.

Ce cycle d'avancement et de recul se répète jusqu'à ce que le chemin vers la sorcière soit dégagé, permettant ainsi au slime de poursuivre sa cible en ligne droite sans obstacle. Cette approche garantit une dynamique réaliste des mouvements, tout en offrant un défi stratégique au joueur, qui peut utiliser les rochers pour ralentir ou détourner les slimes. Cette mécanique repose sur la détection de collisions, combinée à des mises à jour régulières de la position en fonction des coordonnées de la sorcière et de l'environnement.

En plus de leur interaction avec les obstacles, les slimes sont conçus pour ne pas se superposer les uns aux autres lorsqu'ils se déplacent. Cette mécanique permet aux slimes de maintenir une formation réaliste : ils peuvent se suivre en file indienne ou se déplacer côte à côte tout en poursuivant la sorcière. Cette absence de superposition est gérée par la vérification constante des positions des slimes, combinée à une logique de réajustement qui empêche leur chevauchement.

Les slimes ne conservent pas forcément un ordre fixe pendant leur déplacement. Ils peuvent, en fonction de leurs trajectoires respectives, se dépasser ou échanger leurs positions. Ce comportement apporte une variabilité naturelle aux mouvements des ennemis, rendant leur poursuite moins prévisible et plus dynamique pour le joueur. Ce système de gestion des mouvements collectifs est essentiel pour maintenir une fluidité visuelle et stratégique dans l'environnement du jeu.

3.2 Déroulement

En premier lieu, lors de la réalisation de ce projet, j'ai eu de la peine à créer et mettre en place me users stories. J'ai été débloqué grâce aux explications de l'enseignants. Or, j'ai rencontré une difficulté à mettre mes idées en place et à choisir quelles serait mes éléments de gameplay. Cette difficulté à mettre en place correctement des user stories et avoir une idée claire des éléments du jeu a rendu la partie développement plus longue que ce qu'elle aurait dû.

En deuxième lieu, j'ai rencontré une difficulté pour faire avancer mon personnage. C'est-à-dire que mon personnage se déplaçait en un seul mouvement sur l'ensemble de la zone de jeu. J'ai dû demander de l'aide à mon voisin de classe qu'elle était la bonne méthode pour récupérer l'information d'une touche pressée. Ainsi mon personnage se déplaçait une seule direction à la fin et je pouvais contrôler la distance.

D'autre part, j'ai commencé par réaliser une classe pour chaque élément de mon jeu. Si bien que j'ai réalisé de multiple fois la même partie de code et je l'ai implémenté dans chaque classe. C'est pourquoi j'ai perdu énormément de temps, lors de la création de mes classes. De plus, cette quantité de code à double, ma rendue confuse lorsque que je voulais mettre à jour la position de mes objets ou leur donner un comportement particulier.
Dès lors que mon enseignant m'a conseillé d'observé s'il était possible d'ajouter de l'héritage, le code a pu être alléger. À la suite de la création de l'héritage, il m'a semblé plus facile de distinguée mes éléments pour créer les boucles qui créer et détruisent mes objets en jeu.

En dernier lieux, je souhaitais qu'après chaque ennemi tuer, un nouvel ennemi apparaissent. Cependant, j'ai vite réalisé que je n'arrivais pas à définit la nouvelle apparition de l'ennemi. C'est-à-dire l'ennemi doit être suffisamment à distance du joueur qui peut se déplacer entre le temps de respawn de l'ennemi. De plus, l'ennemi ne devait pas spwan sur la ligne d'un projectile. C'est pourquoi, j'ai opté pour la solution de créer des zones de spawn pour l'apparition des ennemis. Mais le problème a été de savoir dans la quel des 4 zones de spawn l'ennemi allait apparaitre. C'est pourquoi j'ai décidé que tous les slimes devait être éliminer avant que quatre nouveau slimes apparaissent.

Durant les deuxième et troisième sprints, j'ai concentré mes efforts sur l'amélioration des mécaniques de jeu et l'optimisation du code. Tout d'abord, j'ai renforcé le système de collison entre les slimes, les rochers et la sorcière. Cela a nécessité une révision des fonctions de détection de collision et des hitboxes pour éliminer les comportements incohérents observés lors du premier sprint.
Ensuite, j'ai travaillé sur la logique de déplacement des slimes. J'ai ajouté des comportements plus dynamique, comme leur capacité à contourner les obstacles et à ajuster leur trajectoire en fonction de la position de la sorcière. Ce développement a également inclus la gestion des déplacements en groupe, avec l'évitement des superpositions et l'ajout d'une variabilité naturelle dans leurs positions.
Enfin, ces deux sprints ont permis d'apporter des améliorations visuelles et ergonomiques. Par exemple, j'ai affiné l'affichage des sprites et amélioré la gestion des zones de spwan des ennemis. Ces ajustements ont contribué à enrichir l'expérience de jeu et à rendre le code plus clair et plus simple à maintenir pour les futures itérations.

Lors de ces sprints, j'ai également travaillé sur la mécanique de tir de la sorcière. J'ai corrigé le problème où rester appuyé sur une touche permettait de tirer en continu. Désormais, chaque pression de touche génère un unique projectile. Cependant, je n'ai pas réussi à implémenter un système de cooldown entre les tirs. Cela signifie que si le joueur spamme la touche de tir, la cadence des projectiles reste anormalement élevée, ce qui rend le gameplay déséquilibré.
J'ai également introduit une gestion des munitions pour limiter le nombre de projectiles disponibles. Le joueur commence chaque partie avec un maximum de 10 munitions. À chaque tir, le stock diminue, empêchant le joueur de tirer une fois le nombre de projectiles épuisé. Une mécanique de régénération des munitions a été ajoutée : chaque fois qu'un slime est tué, les munitions du joueur sont automatiquement rechargées au maximum. Cela ajoute un élément stratégique au gameplay, en incitant le joueur à gérer ses tirs et à prioriser les cibles.

3.3 Mise en place de l'environnement de travail

Le projet peut être retrouvé sur un dépôt GitHub.
Le code est développé en C# avec .NET 8. C'est la librairie graphique WinForm qui est utilisé pour l'affichage du jeu.
Le développement est effectué à l'aide de Visual Studio 2022 et peut être exécuté sur Windows 10 et 11.

3.4 Description des tests effectués

3.4.1 Disparition ennemie

Disparition	Si le slime a 0 de point de vie, il disparait de l'écran	OK
Incrémentation	Quand l'ennemi disparait, le score augmente	OK

3.4.2 Mort de la sorcière

Arrêt du jeu	Quand la sorcière a 0 point de vie, le jeu s'arrête	OK
Arrêt du tire	Quand la sorcière a 0 point de vie, appuyer sur les touches de tires ne provoquent pas de tire.	OK
Arrêt du mouvement	Quand la sorcière a 0 point de vie, appuyer sur les touchent pour bouger la sorcière, rien ne se passe.	OK

3.4.3 Déplacement slime

Joueur immobile	Quand je suis immobile, le slime se rapproche de mon joueur	OK
Joueur mobile	Quand je me déplace, le slime continue de se rapprocher de mon joueurs	OK
Collision des slimes	Quand les slimes se déplacent, ils ne peuvent pas se superposer	OK

3.4.4 Collision des slimes avec les rochers

Collision du slime	Quand un slime touche un rocher, il arrête d'avancer dans la direction du rocher.	OK
Mouvement du slime	Quand un slime touche un rocher, il peut se déplacer dans les directions opposées au rocher si le joueur s'est déplacé.	OK

3.4.5 Apparition ennemie

Apparition ennemi	Quand je tue un ennemi, un nouvel ennemi apparait	OK
Vie au maximum	Quand un ennemi apparait, sa barre de vie est au maximum	OK

3.4.6 Attaque du joueur

Lancement des projectiles	Quand j'appuie sur les touches A, S, W, D, mon personnage tire dans la direction de la touche.	OK
Distance projectile	Quand je tire mon projectile va en ligne droite jusqu'à la bordure de l'écran	OK
Tirer sur le rocher	Quand je tire sur un rocher mon projectile est arrêter	OK

Cooldown	Je dois attendre 1 secondes avant de pouvoir tirer à nouveau	Ko

3.4.7 Lancement du niveau

Le joueur apparait	Quand je lance le niveau, mon personnage apparait au centre de l'écran	OK
Vie complète joueur	Quand je lance le niveau, la barre de vie de mon personnage est pleine	OK
Arme de base	Quand je lance le niveau, mon personnage a déjà une arme pour attaquer	OK
Décors rocher	Quand je lance le niveau, je vois 2 rochers sur l'écran	Ko
Les ennemis slims apparaissent	Quand je lance le niveau, je vois 4 slimes à l'écran	OK
Vie complète ennemi	Quand je lance le niveau, les ennemis ont leur barre de vie complète	OK

3.4.8 Déplacement du joueur

Déplacements du personnage	Quand j'appuie sur les touches directionnelles, mon personnage se déplace dans la direction de la touche appuyée	OK
Collision aux bordures	Quand je déplace mon personnage et que ce dernier touche une bordure de l'écran, mon personnage s'arrête	OK
Collision aux rocher	Quand je déplace mon personnage et que ce dernier touche un caillou, mon personnage s'arrête	OK

3.4.9 Affichage des munitions

Chargeur plein	Quand le joueur lance le jeu, les munitions affichées sont au maximum.	OK
Diminutions munitions	Après un tir du joueur, le nombre de munitions affichées diminuent d'un.	OK
Chargeur vide	Quand le joueur ne peut plus tirer, le nombre de munitions affichées est de 0.	OK

3.5 Erreurs restantes

3.5.1 Premier sprint

La plus grande fonctionnalité qui n'est pas encore implémentée est la gestion des collisions des slimes avec d'autres objets. Je parle ici des rochers, d'autres slimes ou du joueur.
Premièrement, n'existant aucun anti-collisionneur entre les slimes et les rochers, les ennemis peuvent passer au travers des cailloux comme s'ils n'existaient pas. Il n'y a donc aucun moyen de « feinter » les ennemis ou de simplement les ralentir avec des obstacles.
De plus, les slimes peuvent se superposer entre eux pour les mêmes raisons. Graphiquement, le joueur est donc incapable de voir combien d'ennemis il reste en jeu.
Le problème est le même avec la sorcière, respectivement le joueur. Les slimes peuvent superposer leur hitbox à celle de la sorcière. Cela a pour conséquence que la perte de points de vie occasionnée par le contact entre le joueur et un ennemi est extrêmement rapide.

Afin de régler toutes les erreurs mentionnée ci-dessus, il faut prévoir le moyen d'empêcher les hitbox des slimes de se superposer avec d'autres hitboxes. Techniquement, il suffit de bloquer une direction de déplacement si la prochaine coordonnée entre en collision avec une autre hitbox.

Deuxièmement, les tirs de projectiles du joueur sont uniquement liés à la pression d'une touche. Il n'y a donc pas de temps minimum entre chaque tir.
Au niveau du gameplay, cela signifie qu'il est possible de tirer à une cadence très élevée. Cela a un impact sur la difficulté du jeu, cela rends l'expérience trop simple.
Un temps de latence entre chaque tir est clairement envisageable, afin de ne pas trop faciliter le jeu. De plus, la sorcière n'est pas impactée par ses propres bombes. Elle peut se déplacer dessus sans se prendre de dégâts.

Finalement, c'est la différence entre les différentes hitboxes et leur spirite respectif qui pose un problème. En effet, toutes les hitboxes de chaque élément sont des rectangles. Cependant le spirite qui les représente est tout sauf rectangulaire.
Cela pose donc des soucis pour les déplacements, car il peut arriver que l'on ne puisse plus approché d'un rocher parce que les hitboxes du rocher et du joueur se touchent déjà, mais pas les sprites affichés. Il en va de même avec les projectiles.
Afin de régler ce problème technique, il faudrait adapter la taille et la forme de la hitbox au sprite correspondant, ou inversement.

3.5.2 Sprints suivants

Le cooldown de l'arme n'est toujours pas fonctionnel, ce qui est dû à une absence de mise en place du délai entre chaque tir.
Trois cailloux apparaissent au lieu de deux, en raison d'une confusion sur le nombre d'obstacles l Le cooldown de l'arme n'est toujours pas fonctionnel, ce qui est dû à une absence de mise en place du délai entre chaque tir.

Trois cailloux apparaissent au lieu de deux, en raison d'une confusion sur le nombre d'obstacles lors de la modification des tests d'acceptance et des user stories.
Les sprites de slime se superposent toujours à celui de la sorcière, en raison d'une gestion incorrecte des collisions visuelles.
La vie des slimes et de la sorcière diminue trop rapidement, à cause d'un paramétrage trop agressif de la perte de points de vie.
La barre de vie est une barre de progression et non segmentée, ce qui est dû à l'utilisation d'un mauvais type d'élément graphique pour la représentation de la vie lors de la modification des tests d'acceptance et des user stories.
Les sprites de slime se superposent toujours à celui de la sorcière, en raison d'une gestion incorrecte des collisions visuelles.
La vie des slimes et de la sorcière diminue trop rapidement, à cause d'un paramétrage trop agressif de la perte de points de vie.
La barre de vie est une barre de progression et non segmentée, ce qui est dû à l'utilisation d'un mauvais type d'élément graphique pour la représentation de la vie.

4 Conclusions

4.1 Objectifs atteints

Les premières étapes du jeu ont été mises en place, permettant de découvrir le but principal : tirer et tuer des ennemis.
Le joueur peut se déplacer dans une zone de jeu bien définie, sans sortir de cette zone et en étant bloqué par le décor. Il peut également tirer des projectiles qui interagissent avec les ennemis, le décor ou les bords de la zone, et ces projectiles disparaissent une fois leur cible atteinte.
Concernant les ennemis, les slimes apparaissent dans des zones spécifiques, poursuivent le joueur, et infligent des dégâts lorsqu'ils le touchent. Lorsque tous les slimes sont éliminer, de nouveaux apparaissent pour maintenir l'action.
De plus, chaque ennemi tué rapporte des points, et la partie se fige lorsque le joueur perd toute sa vie, mettant ainsi fin à la session de jeu.

4.2 Objectifs non-atteints

Cependant, certains problèmes reste à résoudre. Le cooldown de l'arme, qui est toujours absent, doit être ajouté pour éviter que la cadence des tirs ne devienne trop rapide et déséquilibrent le jeu. Il faut également corriger le nombre de cailloux qui apparaissent, ainsi que la gestion des zones de spawn des ennemis, afin d'éviter les superpositions. La superposition des sprites de slime et de la sorcière doit également être corrigée pour améliorer la lisibilité, et la gestion de la vie des personnages doit être ajustée pour offrir une difficulté plus progressive. Enfin, la barre de vie doit être remplacée par une version segmentée pour la rendre plus claire et intuitive.

4.3 Difficultés particulières

Pour commencer, je suis partie dans le développement en créant directement mes classes les uns après les autres sans réfléchir aux éléments communs. De plus, je me

suis concentré sur le fonctionnement individuel des classes. Par conséquent, j'ai éprouvé de la difficulté à regroupé grâce à l'héritage les éléments communs.
C'est pourquoi la conception le projet à prit du retard si bien que du code similaire a été implémenter plusieurs fois rendant la lecture des fonctionnalités plus difficiles. Ainsi j'ai eu du mal à préparer un diagramme de classe dès le départ et a organisé mes idées.
En deuxième lieu, j'ai remarqué que mes bases en programmation n'étaient pas solides. C'est-à-dire que je me suis retrouvée en difficulté à devoir utiliser les boucles for each. Dans le projet l'utilisation des for each est cruciale pour les collisions et la mise à jour des positions des objets. Par conséquent j'ai dû recommencer et utiliser plus de temps que prévus pour réussir à implémenter les fonctions dans Game.cs.

4.4 Suites possibles pour le projet

Pour les prochaines étapes, plusieurs améliorations sont possibles. Il est crucial de finaliser le cooldown de l'arme pour équilibrer les tirs et éviter que le jeu ne devienne trop facile. La gestion des collisions peut encore être améliorée pour que l'interaction entre les objets soit plus fluide. Il y a aussi des répétitions dans le code qu'il serait bon d'éliminer, ce qui simplifierait le projet et le rendrait plus facile à maintenir. Les zones de spawn des ennemis doivent être retravaillées pour garantir un placement plus logique et éviter les superpositions. Enfin, une optimisation des performances serait un plus, pour que le jeu reste fluide même quand il y a beaucoup d'éléments à l'écran.

5 Annexes

5.1 Journal de travail

Story	Terminée le	Tâche	Remarque	Durée [min]
Administratif	11 sept. 2024	Absence	Malade	3h00
	18 sept. 2024	Absence	Malade	3h00
Affichage des munitions	17 déc. 2024	Afficher le score		0h05
Apparition décors	02 oct. 2024	Création d'une liste de rocher		0h36
	06 oct. 2024	Affichage aléatoire sur la map		0h20
		Empêcher la superposition		0h48
Apparition ennemi	06 oct. 2024	Apparition des ennemis slimes		0h36
		Créer le slime		0h05
		Délimitation des zones de spawn		1h06
	24 oct. 2024	Vie complète à l'apparition		0h05
	01 nov. 2024	Affichage d'une barre de vie		0h16
	06 oct. 2024	Assigner une touche pour la direction		0h24
		Créer un projectile		0h05
		Destruction du projectile hors map		0h15
		Déplacement du projectile		0h29

Attaque du joueur	22 oct. 2024	Disparait lors des collisions		0h18
		Retirer des points de vie au slime		0h18
	17 déc. 2024	Limitation du nombre de projectile	Aider par l'enseignant pour ajouter une méthode Fire à la witch	1h00
		Rester appuyer sur une touche ne lance qu'un tir		0h26
		Tirer depuis le centre de la sorcière		0h23
Collision des slimes avec les rochers	03 déc. 2024	Collision slime avec le rocher	Débloquer grâce à l'aide de mon enseignant pour forcer les slimes à reculer lors de collisions avec le rocher	1h20
Urgente	28 août 2024	Introduction sur le Projet		0h45
		Mise en place repos Github		0h20
		Rédaction document concept gameplay		0h30
	04 sept. 2024	Améliorations des Users-storie		1h00
		Critique des Users-storie		0h10
		Démonstration Accepter les taches		0h10
		Rédaction premières Users-storie		0h55
	25 sept. 2024	Démonstration Exception	Présentation des exceptions et de leur mise en place	1h00
		Démonstration Test Unitaire	Démonstration de la mise en place des tests unitaire dans leur propre solution avec une dépendance à la solution ShootAbby	0h30
		Récupération csproj	Problème avec le framework qui se créait en double	0h35
	24 oct. 2024	Héritage des classes		1h13
		Héritage des classes dans View		0h54
		Diagramme de classe		0h57

	29 oct. 2024	Sprite objets		1h53
	01 nov. 2024	Démonstration release Github		0h09
		Présentation diagramme d'état		0h12
		Retour sur la documentation		0h26
		Documentation rapport		4h39
	02 nov. 2024	Diagramme d'état		0h14
	03 déc. 2024	Correction bug apparition des rocher		0h15
		Création des sprints		0h10
		Sprint review exemple test		0h09
	10 déc. 2024	Sprint rétrospective		0h15
		Sprint review sprint 2		0h15
Total				39h06