


SHOOTABBY

Shoot me up!

02 NOVEMBRE 2024

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 impossible~~s~~ 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

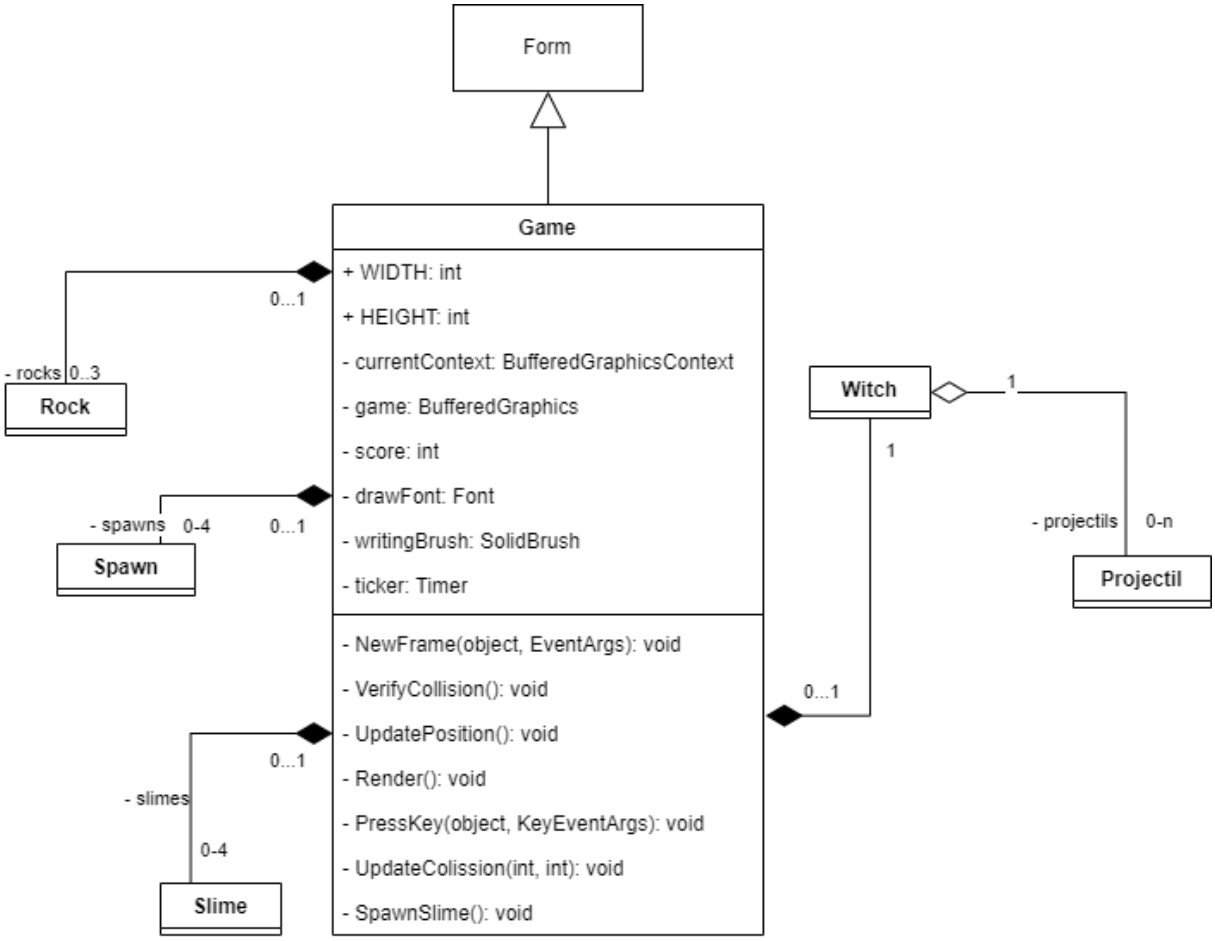

Figure 1 UML -Association

2.7.2 Diagramme de classe de l'héritage

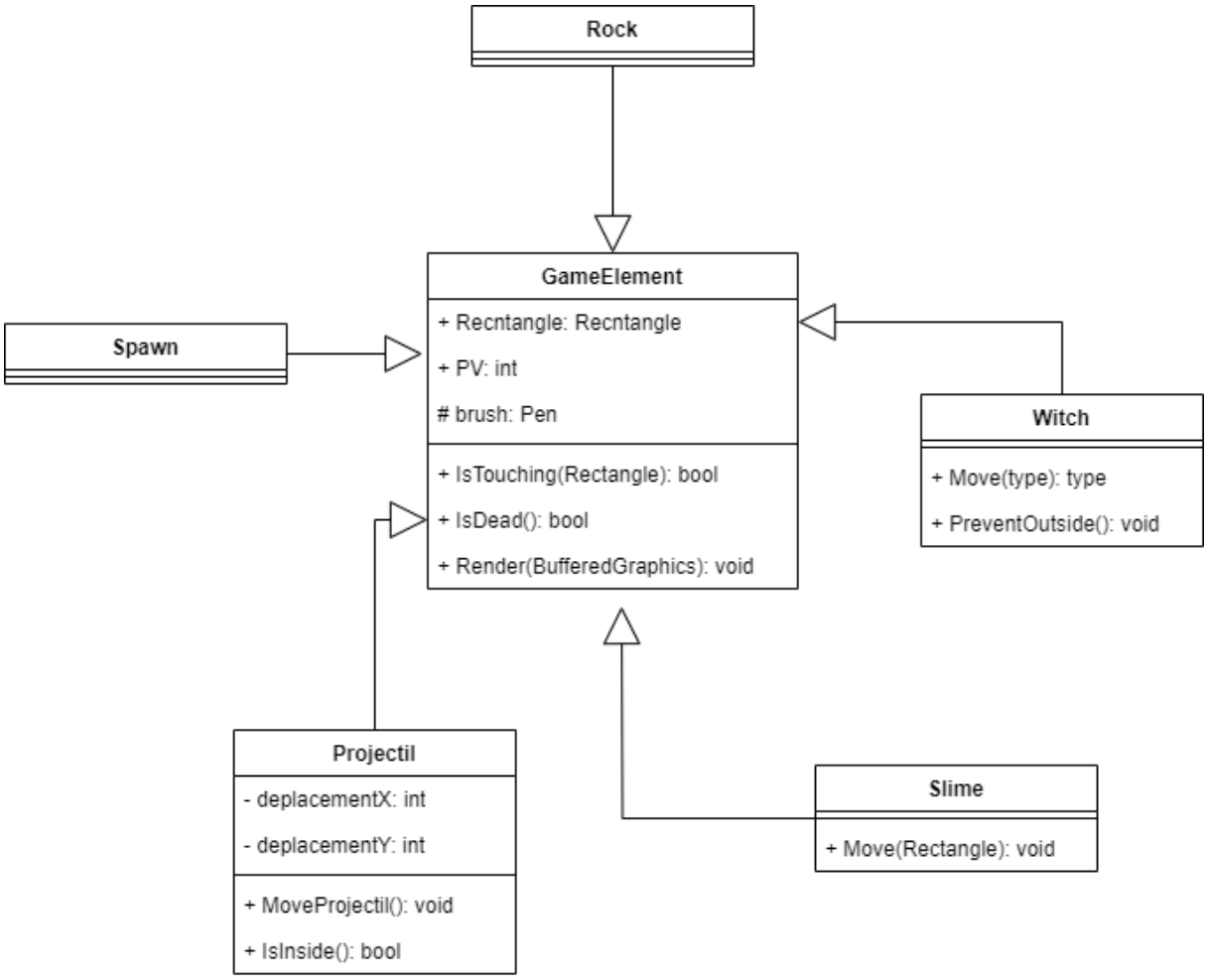

Figure 2 UML - Héritage

2.7.3 Diagramme d'état

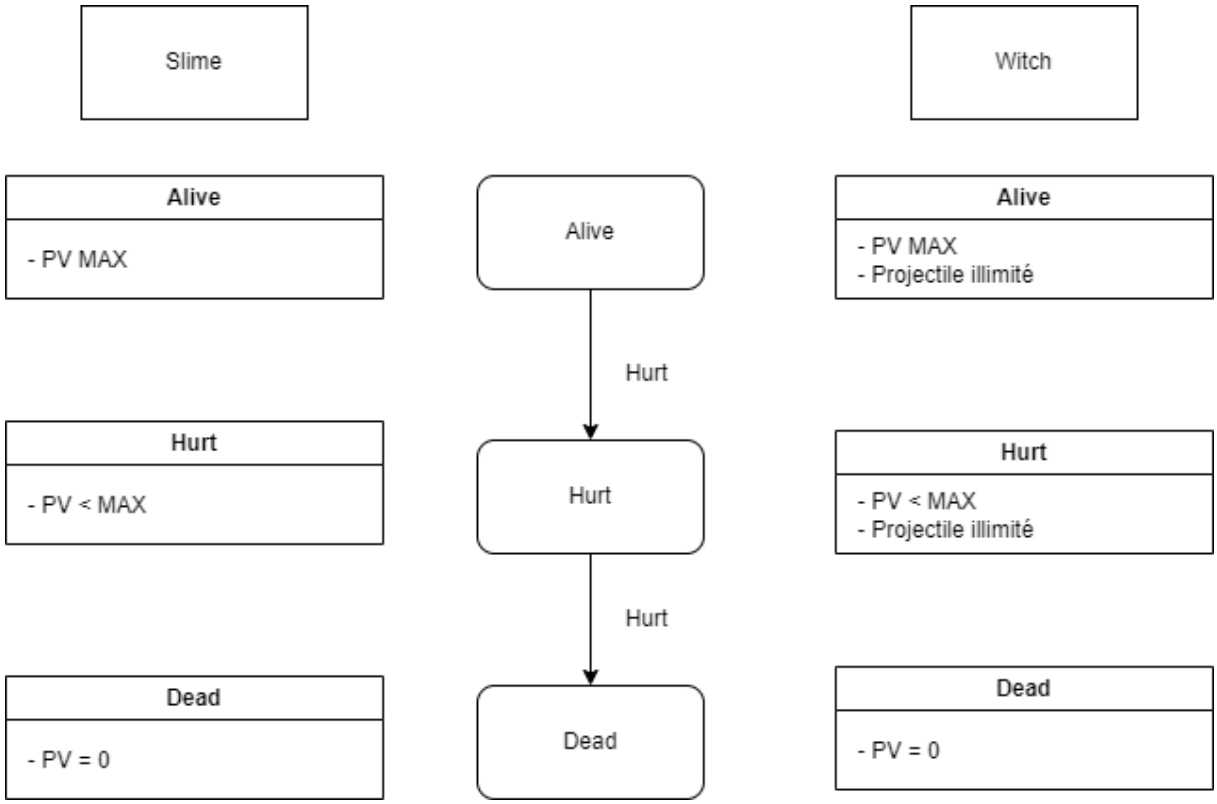

Figure 3 Diagramme Etat - Witch & Slime

2.8 Analyse fonctionnelle

2.8.1 Déplacement du joueur

En tant que joueur, je veux déplacer mon personnage

Tests d'acceptances :	
Déplacements du personnage	Quand j'appuie sur les touches directionnelles, mon personnage se déplace dans la direction de la touche appuyée
Collision 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
Collision aux buissons	Quand je déplace mon personnage et que ce dernier touche un buisson, 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)
Arme de base	Quand je lance le niveau, mon personnage a déjà une arme pour attaquer (voir maquette)
Décors rocher	Quand je lance le niveau, je vois 2 rocher sur l'écran (voir maquette)
Décors buissons	Quand je lance le niveau, je vois 3 buissons sur l'écran
Les ennemis archer apparaissent	Quand je lance le niveau, je vois 2 ennemis archer à l'écran
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

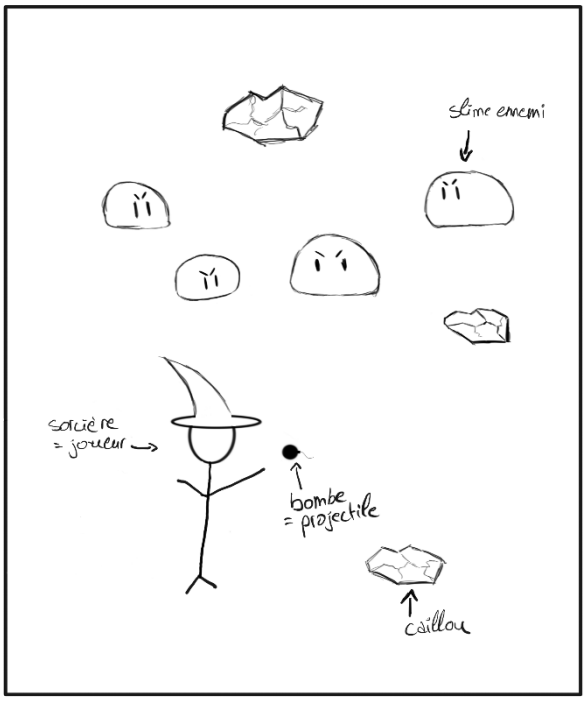

Figure 4 Maquette début de partie

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 un ennemi, un nouvel 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
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
Tirer sur un buisson	Quand je tire sur un buisson, le buisson est détruit et un nouveau apparait 10 secondes plus tard

2.8.5 Déplacement slime

En tant que joueur, je veux que les slimes 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 au rocher	Quand le slime se rapproche de moi et qu'il touche un rocher, il est bloqué
Collision des slimes	Quand les slimes se déplacent, ils ne peuvent pas se superposer

2.8.6 Apparition décors

En tant que joueur, je veux du décor à l'écran pour me cacher

Tests d'acceptance :	
Démarrage	Quand je lance le jeu depuis l'exécutable, il y a 3 rochers dans la zone de jeu.

2.8.7 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

2.8.8 Disparition ennemie

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

Tests d'acceptance :	
Disparition	Quand le slime a 0 de point de vie, il disparait de de la surface du jeu
Incrémentation	Quand un ennemi disparait, le score augmente

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. Je n'ai malheureusement pas fait tester mon jeu à d'autre personnes pour avoir leur retour.

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.

```
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

3.2 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 débloquée 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 des 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 à 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 décider que tout les slimes devait être éliminer avant que quatre nouveau slimes apparaissent.

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

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 au rocher	Quand le slime se rapproche de moi et qu'il touche un rocher, il est bloqué	OK
Collision des slimes	Quand les slimes se déplacent, ils ne peuvent pas se superposer	Ko

3.4.4 Apparition ennemie

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

3.4.5 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
Tirer sur un buisson	Quand je tire sur un buisson, le buisson est détruit et un nouveau apparait 10 secondes plus tard	Ko

3.4.6 Apparition décors

Démarrage	Quand je lance le jeu depuis l'exécutable, il y a 3 rochers dans la zone de jeu.	OK

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	Ko
Décors rocher	Quand je lance le niveau, je vois 3 rocher sur l'écran	OK
Décors buissons	Quand je lance le niveau, je vois 3 buissons sur l'écran	Ko
Les ennemis archer apparaissent	Quand je lance le niveau, je vois 2 ennemis archer à 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	Ko

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
Collision aux buissons	Quand je déplace mon personnage et que ce dernier touche un buisson, mon personnage s'arrête	Ko

3.5 Erreurs restantes

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 mentionnées 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 rend 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 approcher 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.

4 Conclusions

4.1 Objectifs atteints

J'ai pu effectuer les premières étapes du jeu qui permette de découvrir le but du jeu : tirer et tuer des ennemis.

Concernant le joueur, ce dernier peut se déplacer dans sa zone de jeu, ne pas sortir de cette zone de jeu et être bloqué par le décor. De plus, le joueur peut aussi tirer des projectiles qui s'arrêtent sur des ennemis, en dehors de la zone de jeu ou dans le décor. Ces projectiles sont supprimés de l'affichage une fois la cible atteins.
En ce qui concerne les ennemis, les slimes apparaissent uniquement dans leur zone d'apparition définie et poursuivent bien le joueur. Des dégâts sont cumulés sur le joueur lorsqu'il est touché par les slimes. De surcroît, des slimes réapparent en jeux quand il n'y en a plus sur la zone de jeu.

Pour terminer, chaque ennemi tué rapporte des points qui sont cumulés dans un score. De plus, la partie se freeze une fois que le joueur perd toute sa vie au contact des monstres.

4.2 Objectifs non-atteints

En premier lieu, le joueur peut tirer en continue sans s'arrêter et sans limite de tir ou de munition. Non seulement on peut dire qu'il manque là une gestion du temps entre chaque tir du joueur mais de plus la limitation des munitions n'a pas été implémenté. Etant donné qu'il manque là une gestion du temps entre chaque tir du joueur et chaque dégât pris par les ennemis, le joueur prend des dégâts en continue durant les collisions.

En deuxième lieu le slime, qui poursuit le joueur, n'a pas de collision avec les rochers. D'un autre coté, l'ennemi des coins de la zone de jeu n'ont pas été implémenté dans le jeu actuel. Par conséquence, aucun projectile n'est tiré en direction du joueur.
De même que l'implémentation des barricades, des objets bonus ; soin, arme alternative et munitions n'ont pas été créer sur la zone de jeu pour aider le joueur dans sa partie.

Pour conclure, aucun système de niveau n'est présent dans le jeu et aucune difficulté apparait en fonction du temps de jeu. Si bien que le jeu n'apporte ni difficulté ni variation dans son fonctionnement.

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 foreach. Dans le projet l'utilisation des foreach est cruciale pour les collisions et la mise a 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

Afin de pouvoir continuer le projet, un diagramme des classes contenant les informations sur les éléments manquants permettrait une plus grande facilité à organisé et implémenter le code. Grâce à cela, on pourrait ajouter les ennemi archer, ainsi que les objets récupérables au sol tel que les munitions et les potions de soin. De plus, la gestion du visuel et des sprites devra être adaptée pour mieux remplir la surface de la hit box définie. Mais il est aussi possible de réduire cette hitbox en lui donnant une autre forme qu'un rectangle voir de la personnalisée au sprite défini.
De plus, l'ajout d'un menu serait un plus pour le joueur, car le jeu se lance directement à l'ouverture du programme.

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
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
Attaque du joueur	06 oct. 2024	Assigner une touche pour la direction		0h24
		Créer un projectile		0h05
		Destruction du projectile hors map		0h15
		Déplacement du projectile		0h29
	22 oct. 2024	Disparait lors des collisions		0h18
		Retirer des points de vie au slime		0h18
Deplacement slime	06 oct. 2024	Déplacement des slimes		0h19
Disparition ennemie	22 oct. 2024	Un slime mort disparait de l'écran		0h22
Déplacement du joueur	02 oct. 2024	Colision aux bordures		1h00
	06 oct. 2024	Colision aux rochers		0h32
	09 oct. 2024	Déplacement du joueur		0h50

story	terminée_le	tâche	remarque	Durée [min]
Lancement du niveau	02 oct. 2024	Apparition du joueur au centre de l'écran		1h42
	01 nov. 2024	Affichage d'une barre de vie	Débloquée grâce à l'aide du professeur	0h25
		Vie complète pour le joueur		0h10
Mort de la sorcière	22 oct. 2024	les points de vie diminue à la colision d'un slime		0h12
	24 oct. 2024	Le jeu se bloque		0h08

story	terminée_le	tâche	remarque	Durée [min]
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 test 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
	29 oct. 2024	Diagramme de classe		0h57
		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
			Total	33h53