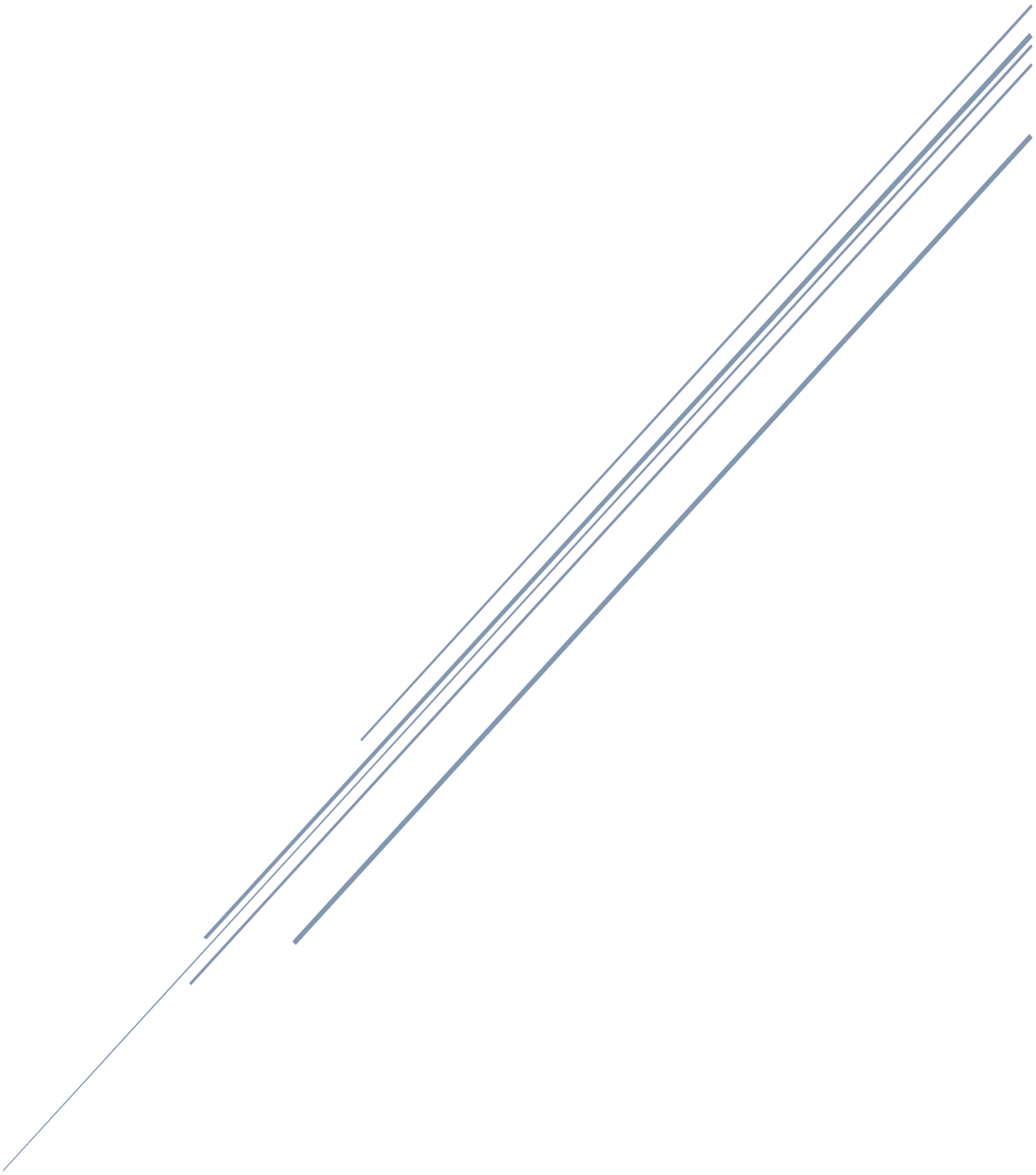
PROJET SPICY INVADER

UX, OO, DB



Ethan Schafstall

28.08.2023-03.11.2023

Table des matières

Expérience utilisateur (UX)

Introduction

L'UI et l'UX sont parmi les éléments les plus importants dans le développement d'un jeu ou de tout autre produit. Même avec les meilleures technologies, si l'utilisateur a du mal à comprendre, naviguer et a généralement une mauvaise expérience, alors tout le travail est une défaite pour le PO (Product Owner) et les développeurs. Il est donc important d'identifier les besoins des utilisateurs pour que le produit soit une réussite.

Analyse

L'Inclusivité/l'accessibilité

Dans le domaine des médias, il y a un problème d'inclusion de groupes marginalisées, toujours en temps moderne. Que ça soit le manque de représentation de certaines ethnies dans le cinéma. Le manque d'options pour navigation pour personnes avec handicapes. Les personnes de différentes zones géographiques, différentes langues, ainsi que plein d'autres.

L'Inclusivilité : Représentation des femmes

Dans le contexte de l'informatique et des jeux vidéo, c'est le manque de reconnaissance des femmes en tant que clientèle potentielle, et utilisateurs actifs des produits. L'idée qu'il y a qu'il y a seulement les males (ados surtout), et que les filles/femmes ne joue pas au jeux vidéo est un mythe qui percés malgré les études de haute éducation, et les études de marché faites dans les dernières+ 20 ans.
C'est pendant que phase de conception d'un produit ou, en tant que développer qu'il y a une responsabilité d'identifier les clients/utilisateurs des produits et de leur offrir une meilleure expérience.

L'accessibilité : Localisation

Dans un monde globalisé, Il y a une importance à fournir de différentes options (langue, monnaie, horaire, etc.). Dans le contexte de jeux, avoir plusieurs options de langues pour pouvoir naviguer les menus est nécessaire, mais si le jeu en lui-même a très peu d'éléments de lecture.

L'accessibilité : Déficience/Handicap visuelle

Pour toutes formes de “l’Entertainment“, surtout toutes media en rapport avec un écran, il faut avoir des adaptations pour personnes avec déficiences ou handicaps.

Dans le groupe de déficience/ et handicap visuelle, il y a :

- Personnes avec daltonisme
- Personnes âgées
- Personnes myopes ou hypermétropes
- Personnes avec TSA, ou autres troubles néologiques

Le choix d’une fonte qui soit facilement lisible a de différentes distances/tailles, avec choix de couleurs approprié.

Éco-conception

L'éco-conception est un concept très important en tant que développeur, car les applications vont toujours consommer des ressources énergétiques à chaque usage. Il est donc important d'optimiser au maximum et d'utiliser le plus de techniques pour éviter l'utilisation de ressources inutilement.

Pour un jeu, il y a plusieurs moyens :

- Le choix de la palette de couleurs pour le jeu/UI.
- Le nombre d'objets à l'écran pendant le gameplay.
- La fréquence de la connectivité à la base de données.

Autant d'autres peuvent avoir un meilleur impact écologique, avec le minimum de sacrifice pour l'expérience du joueur.

Innovation

Pour donner créer une meilleure expérience pour l’utilisateur c’est important d’être innovatif. L’implémentation de idées qui créer de nouvelles/améliore le produit ou service.

Pour les jeux il y a pleins de méthodes pour le rendre plus innovatif :

Le gameplay
Le design des personnages
Les options de personnalisation

Pour créer une meilleure expérience pour l'utilisateur, il est important d'être innovatif. L'implémentation d'idées qui créent de nouvelles améliorations pour le produit ou service.

Pour les jeux, il existe de nombreuses méthodes pour les rendre plus innovatifs :

- Le gameplay
- Le design des personnages
- Les options de personnalisation

Ce sont quelques-unes des méthodes où la créativité des développeurs joue un rôle clé, tout comme la capacité à repousser les limites de la technologie utiliser pour offrir des expériences uniques.

Conception

Persona

Une persona est censée être une représentation fiable et réaliste du public clé pour un produit, donc c'était l'opportunité d'utiliser des femmes.

Les deux personas crée ce sont des femmes, chacune avec une profession, un âge de plus de 25 ans, et une biographie qui ne correspond pas typiquement au stéréotype du jeune "gamer" masculin, mais qui est plus représentative du public actuel qui s'intéresse aux jeux.

Persona : Mia Rodriguez

Mia Rodriguez, est une jeune développeuse du Texas, mais avec des origines hispaniques. Elle est très bilingue en anglais et espagnol et est très proche de sa famille.

Avec Mme Rodriguez, il était important de se focaliser sur la notion de localisation et du partage. Elle utilise les jeux comme moyen de se connecter à sa jeune frangine, qui ne parle pas que l'espagnol, et veut qu'elle puisse en profiter autant que sa grande sœur.

Donc cette persona met en évidence l'importance de proposer différentes options linguistiques, pour que tout le monde puisse utiliser, comprendre et naviguer dans le jeu facilement. Elle montre aussi que toutes les personnes qui vont jouer au jeu n'ont pas forcément de l'expérience à jouer aux jeux vidéo, que ce soit des personnes âgées, des jeunes comme dans le cas de sa sœur, ou autres. Il est important de simplifier le plus possible pour le rendre accessible.

About

Name: Mia Rodriguez
Gender: Female
Occupation: Software Engineer
Location: Austin, Texas
Age: 28

Frustrations

- Overwhelming Complexity
- Difficulty options
- Bad translations

Criteria

- Language Options
- Accessibility
- Multiplayer

Biography

Mia Rodriguez, a 28-year-old software engineer living in sunny Austin. Texas. By day, she's a software developer, but when she's off the clock, she spends time with her family.

Mia's passion for gaming goes beyond the thrill of competition; it's a way for her to bond with her younger sibling. There's just one catch: her sibling primarily speaks Spanish, while Mia is fluent in both Spanish and English. This language barrier has sometimes stood in the way of their gaming adventures.

Mia's gaming frustrations have led her on a quest to find games that not only offer exciting gameplay but also bridge the language gap, allowing her and her sibling to fully enjoy gaming together. She seeks titles that provide multilingual support, boast accessibility for newcomers, and spare her the constant pestering of in-app purchases. showcase the power of multilingual gaming, bringing joy and connection to her family gaming sessions.

Persona : Mia Rodriguez

Persona : Sarah Mitchell

Sarah Mitchell est une conservatrice botanique âgée de 45 ans, originaire d'une petite ville en Oregon.

Mitchell représente la clientèle qui a grandi durant l'âge d'or des jeux, où les gens passaient leurs week-ends à l'arcade ou à jouer sur leur Atari 2600. Ceux qui ne s'identifient pas avec les jeux modernes, ou peut-être préfèrent juste d'autres options.

Cette persona met en évidence l'importance de la nostalgie pour de nombreuses personnes qui recherchent les mêmes expériences qu'elles ont eues pendant leur enfance : des temps plus simples pour les jeux en ce qui concerne les graphismes, le gameplay, la musique et le choix des jeux. Pas trop difficiles, mais un défi correct.

About

Name: Sarah Mitchell
Gender: Female
Occupation: Botanical Conservator
Location: Serenity Falls, Oregon
Age: 45

Frustrations

- Unfair difficulty spikes
- Lack of variety in gameplay
- Unresponsive menus

Criteria

- Nostalgia factor
- Retro graphics
- Balanced challenge

Biography

Sarah Mitchell, a 45-year-old botanical conservator hailing from a quiet suburban town in the Pacific Northwest. While her day-to-day work revolves around preserving and studying plant life, her true passion lies in retro-style gaming. Sarah's childhood was spent in front of a CRT TV, mastering classic retro games like Donkey Kong.

Today, she seeks to relive those nostalgic moments through modern gaming experiences, particularly console games reminiscent of the classics. She may not be looking for an intricate story or immersive gameplay; instead, she craves balanced challenges and games that capture the essence of the retro era.

Persona : Sarah Mitchell

Palette Couleur

Pour la palette couleur c'était important de prendre en compte les personnes avec troubles visuelle, ainsi que l'éco-conception.

Donc deux palettes ont été créer pour la maquette. Une version normale, et une palette pour aider les personnes trouble visuelle, en particulier le daltonisme.

Les deux palettes on sont sur le côté sombre, qu'évité la fatigue des yeux pendent de périodes de gameplay, et pour utiliser moins de ressources énergétiques. Avec les couleurs qui servira pour mettre en valeur/évidence certaines éléments du jeux/menu.

Primary color:	#5821CA #5821CA	#4211A9 #4211A9	#37137F #37137F	#280D5F #280D5F	#130531 #130531
Secondary color (1):	#9815C7 #9815C7	#7A06A4 #7A06A4	#5E0C7B #5E0C7B	#46075C #46075C	#23022F #23022F
Secondary color (2):	#EEFD11 #EEFD11	#E5F400 #E5F400	#ADB70A #ADB70A	#818906 #818906	#424700 #424700
Complement color:	#FFDF11 #FFDF11	#F7D600 #F7D600	#BAA20B #BAA20B	#887906 #887906	#473E00 #473E00

Palette Couleur default

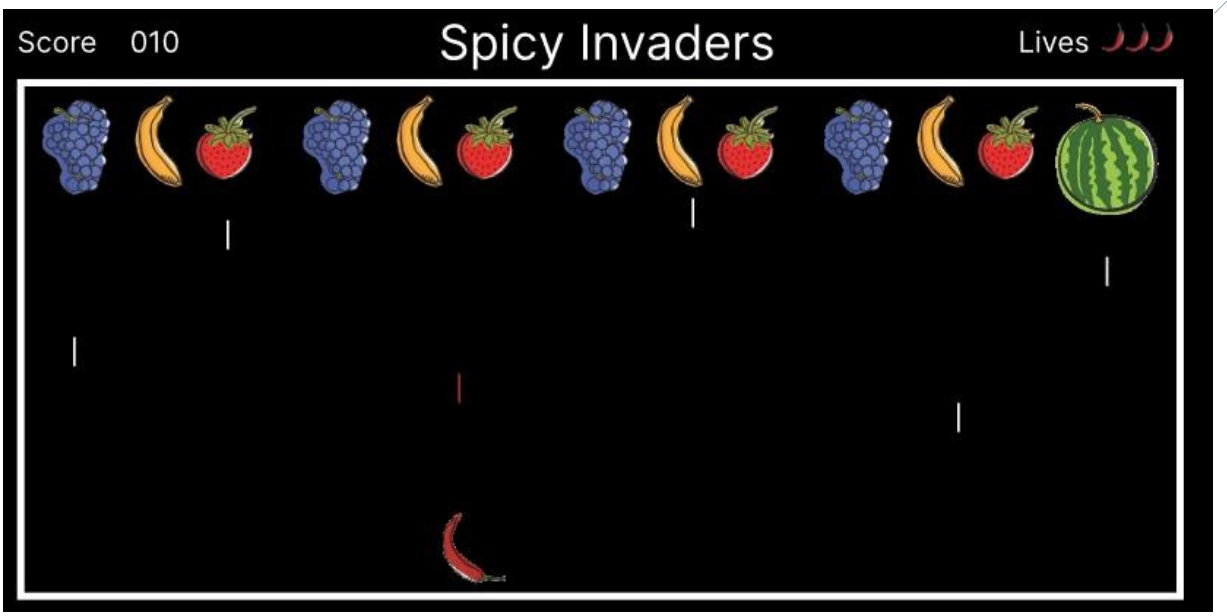
Primary color:	#356A92 #356A92	#1A5888 #1A5888	#0E4671 #0E4671	#05375D #05375D	#022845 #022845
Secondary color (1):	#42469E #42469E	#262B93 #262B93	#181D7A #181D7A	#0D1165 #0D1165	#06094A #06094A
Secondary color (2):	#E3BA48 #E3BA48	#D4A41C #D4A41C	#AF840B #AF840B	#916B00 #916B00	#6B4F00 #6B4F00
Complement color:	#E3A148 #E3A148	#D4851C #D4851C	#AF690B #AF690B	#915300 #915300	#6B3D00 #6B3D00

Palette Couleur alternatif

Maquette : Basse Fidélité

Pour la maquette basse fidélité, c'était important de montrer la version la plus simple de quoi ressemblerai le jeu.
Dans le thème des jeux retro, le coloré du fond est noir, avec très peu de couleurs.
Le menu est très, avec un minimum d'options.

Maquette basse fidélité : Menu principale



Maquette basse fidélité : Gameplay

Dans la version basse fidélité il y a le titre du jeu, les ennemies, le personnage du joueur (le piment), et d'autres informations utile pendants que l'utilisateur joue.

La décision de choisi de fruits pour toutes personnages à plusieurs raisons.

La visibilité : Les couleurs des fruits donne un bon contraste sur le fond noir, et avec chaque fruit qui a une forme diffèrent ça reste facile à les différencier.

L'accessibilité : Le faite d'utiliser des fruits comme ennemies au lieu d'alien, aide à pouvoir pour attirer un public plus jeune, ou peut-être les personnes qui n'aime pas forcement la violence. Des fruits qui explose n'embête personne, généralement.

L'innovation : Utiliser des fruits comme personnages dans ce genre de jeu, est hors du commun, donc ça rend le jeu unique a un certain dégrée.

Maquette : Haut Fidélité

Pour la maquette haut fidélité, les deux palettes couleurs en été utiliser pour rendre les menus, l'écran de jeu plus intéressant visuellement, et de créer un fond d'écran plus dans le thème de l'espace.

Maquette haut fidélité : Menu principale, thème principale, Anglais

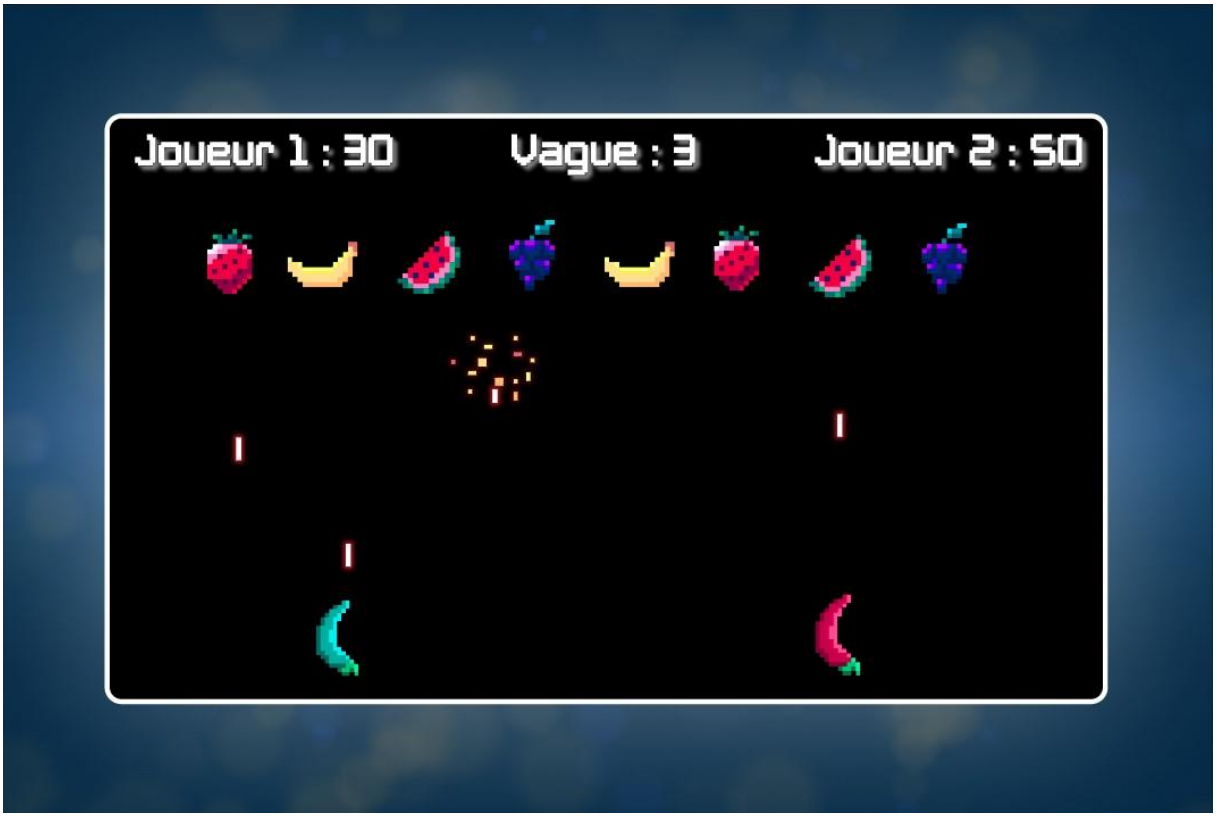
Maquette haut fidélité : Menu principale, thème alternative, Anglais

Dans la maquette haut fidélité c'était important de montrer l'usage des deux palettes sur le thème du jeu, et surtout le gameplay :

Maquette haut fidélité : Gameplay Singleplayer, thème principale, Anglais

Maquette haut fidélité : Gameplay Singleplayer, thème alternative, Anglais

Maquette haut fidélité : Gameplay multiplayer, thème principale, Anglais



Maquette haut fidélité : Gameplay multiplayer, thème alternative, Français

Maquette haut fidélité : Tableaux des scores, thème principale, Anglais

Pour démontrer la partie d'interactivité du jeu/programme, voici le Lien vers le prototype avec menu interactive.

La maquette haut fidélité démontrés les différentes façons de personnaliser sont expérience de jeu, avec les diverses options de jeu :

Maquette haut fidélité : Menu options, thème principale, Anglais

Maquette haut fidélité : Menu sélection langue, thème alternative, Français

Le choix de changer la langue du jeu pour que toutes utilisateurs des différentes régions puisse comprendre, et avoir une meilleure expérience.

Maquette haut fidélité : Menu sélection difficulté, thème principal, Français

Maquette haut fidélité : Menu changement graphismes, thème principal, Anglais

Maquette haut fidélité : Menu selection theme couleur, thème alternative, Francais

Maquette haut fidélité : Menu sélection style ennemies, thème principal, Anglais

Une option pour changer le style des ennemies du jeu, pour les personnes nostalgiques.

Évaluation

Les tests d'acceptance ont été faite dans icescrum durant la planification des projets. Il y a 4 user stories avec une totale de 11 taches, soit 3~ taches par user story. Les tests concernent les différents éléments de la conception du projet :

- Personas
- Palette couler
- Maquette basse fidélité
- Maquette haut fidélité

Create 2 personas

About	_*Given*_ I'm in the persona folder _*When*_ I open a persona card _*Then*_ I should see about info	???
Frustrations and Criteria	_*Given*_ I'm in the persona folder _*When*_ I open a persona card _*Then*_ I should see frustrations and criteria info	???
Biography	_*Given*_ I'm in the persona folder _*When*_ I open a persona card _*Then*_ I should see a biography	???
2 personas	_*Given*_ I'm in the persona folder _*When*_ I look at the persona files _*Then*_ I should have files associated to both of the personas	???

Color palette

Mock-up colors	_*Given*_ I open the mock-up file _*When*_ I look at the mock-up _*Then*_ I should see the different colors from the main color palette.	???
Color blind colors	_*Given*_ I'm in my mock-up _*When*_ I tick the color blind option _*Then*_ the mock-up colors should switch to the secondary color palette.	???

low fidelity mock-up

title screen	_*Given*_ I open the low fidelity mock-up _*When*_ I'm on the title screen _*Then*_ I should see the game title, as well as some art/a logo.	???

menu	_*Given*_ I open the low fidelity mock-up _*When*_ I'm on the menu screen _*Then*_ I should see a menu prompt and different menu options.	???
gameplay screen	_*Given*_ I open the low fidelity mock-up _*When*_ I'm on the gameplay screen _*Then*_ I should see a game assets and game info.	???

(high fidelity model) Interactive menu

Return to previous menu screen	_*Given*_ in the menu _*When*_ I click the return option _*Then*_ I should return to the previous menu	???
Interactive menu	_*Given*_ On a menu _*When*_ click on an option _*Then*_ the screen and options should change to the appropriate menu.	???

Le projet s'est globalement bien déroulé avec les différentes phases, et les travaux rendus correspondent aux critères du cahier des charges. La phase analytique a permis l'identification de potentiels problèmes, besoins, et améliorations qui pourraient être abordés.

La phase de conception a ensuite permis la mise en œuvre des différentes solutions et choix de design. Le choix des personas, de la palette de couleurs, des maquettes, etc. Tout cet ensemble a servi à créer de la valeur pour le plus grand nombre d'utilisateurs. Mais aussi à promouvoir les idées d'accessibilité et d'inclusivité

Programmation orientée objet (OO)

Introduction

La partie programmation orientée objet du projet comprends tout le code et fonctionnalité, du jeu, avec sa complexité, c'est nécessaire d'utiliser le classes et objets.
Comme la majorité des jeux, il utilise un design pattern de MVC (Model-View-Controller), attribué l'un de trois responsabilités à chaque classe.

Le Model est tous qui concerne les données et la logique de notre jeu, dont les classes d'ennemies, du vaisseau joueur, etc.

Le View est pour toute affichage graphique de l'interface et des modèles.

Le Controller c'est pour gérer toutes logique concernant les actions effectuées par l'utilisateur

Analyse fonctionnelle

Move the player ship horizontally

As a player I want to move the player ship horizontally In order to avoid enemy projectiles	
Tests d'acceptance:	
Move left	_*Given*_ during gameplay _*When*_ I press leftarrow or a key _*Then*_ my ship moves left
move right	_*Given*_ during gameplay _*When*_ I press rightarrow or d key _*Then*_ my ship moves right

implement vector based movement.

As a développer I want to implement vector based movement. In order to streamline movement based classes.	
Tests d'acceptance:	
Creating an instance of the Vector class	Given: The Vector class is available. When: I create a new Vector instance with values (3, 4). Then: The Vector instance should have x and y values set to 3 and 4, respectively.
Adding two vectors together.	Given: Two Vector instances are available: Vector A with values (2, 3) and Vector B with values (1, -1). When: I add Vector A and Vector B together. Then: The result should be a new Vector instance with values (3, 2).
Updating a class to include Vector-based movement.	Given: A class named "MoveableEntity" exists, representing a game object. When: I update the "MoveableEntity" class to include Vector-based movement by adding a "velocity" property of type Vector. Then: The "MoveableEntity" class should now have a "velocity" attribut, allowing objects to move using vectors.
Moving a game object using Vector-based movement.	Given: A "MoveableEntity" instance with position (5, 5) and a "velocity" of (2, 2). When: I update the game state to move the object according to its velocity and move direction for one time step. Then: The "MoveableEntity" should have a new position of (7-/+2, 7-/+2).

Projectile enemy interation

As a user I want projectile enemy interation In order to make the gameplay interactive	
Tests d'acceptance:	
Player projectile hits Enemy	_*Given*_ I'm playing the game _*When*_ one of the player's projectiles hit an enemy _*Then*_ I get a visual confirmation that the enemy was hit
Player projectile destroys Enemy	_*Given*_ I'm playing the game _*When*_ one of the player's projectiles hit an enemy with lower healthpoints than projectile damage _*Then*_ I get a visual confirmation that the enemy was destroyed
Enemy projectile destroys Player	_*Given*_ I'm playing the game _*When*_ one of the Enemy's projectiles hit the player ship _*Then*_ I should get a visual confirmation that the player ship was destroyed
Remove destroyed Enemies	_*Given*_ I'm playing the game _*When*_ after I've destroyed an enemy _*Then*_ the destroyed enemy does not appear on the screen

Create UML

As a développer I want to a UML diagram In order to understand/visualize my programs classes and relations.	
Tests d'acceptance:	
UML conventions	_*Given*_ The file docs/invaders.drawio (ou quelque soit le nom de votre fichier) _*When*_ I open it _*Then*_ the diagram is displayed, the UML conventions are respected
Contains all classes, attributs, and functions.	*Given*_ The file docs/invaders.drawio (ou quelque soit le nom de votre fichier) _*When*_ I open it _*Then*_ all the classes, their attributs and functions from the projet solution are in the UML

Analyse technique

Diagramme de classe

Dans le projet C# il y a 21 classes, 5 enum, et 1 interface, tous reparti entre 5 namespace.
Tous les diagrammes des différentes class se trouve dans le dossier **Doc\TechnicalAnalyst\ClassDiagrams** du projet sous forme d'image png, ou directement dans la solution C#.

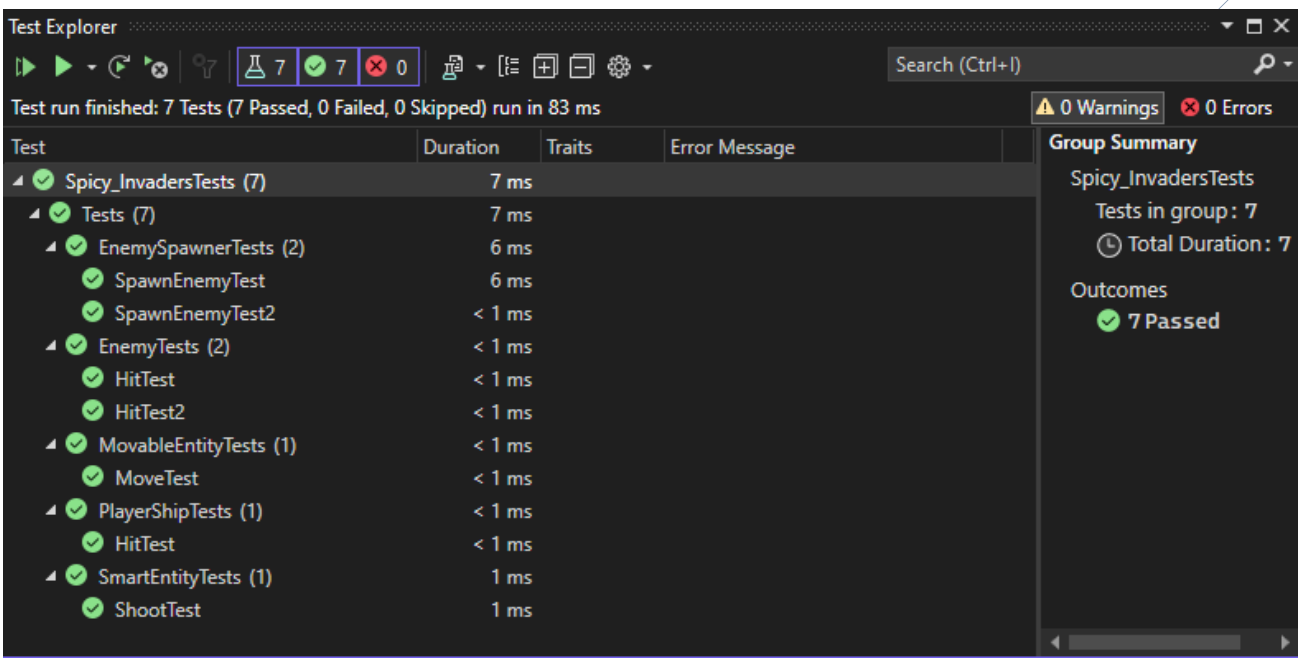
Explications

Les détails sur chaque classe, leurs méthodes, prophéties, et autres on était documenté avec **docfx**.
Toutes fichiers créer avec **docfx** trouve dans le dossier **Doc\TechnicalAnalyst\ClassExplanations** du projet sous forme PDF, dans les dossiers portant le même nom que leur namespace.

Tests Unitaire

Nom du test	Class	Method	Explication	Condition
SpawnEnemyTest	GameEngine	SpawnEnemy	Pour tester la method SpawnEnemy qui creer un enemy et l'ajoute dans la variable de class List Enemy.	Si le nombre de objets (ennemies) dans la liste correspond au expected count int.
SpawnEnemyTest2	GameEngine	SpawnEnemy	Pour tester la method SpawnEnemy qui créer un enemy Melon et l'ajoute dans la variable de class List Enemy.	Si le first et last enemy du list correspond / ou ne correspond pas au type d'enemy melon
ShootTest	SmartEntity	Shoot	Test si quand un SmartEntity spawn un projectile via la Method shoot(), si la position	Si la position x, position y, et la travel direction sont les

			x/y depart du projectile, et sa direction sont correcte.	mêmes que le SmartEntity qui l'a envoyé.
HitTest	Enemy	Hit	Test si un enemy est touche par un projectile, si ces points de vie descends.	Si la nombre des points de vie de l'ennemie touche est égale à leur base health – le nombre de dmg du projectile qui la touche
HitTest2	Enemy	Hit	Test si an enemy n'a plus de points de vie, si n'est pas Alive, et si le niveau d'explosion a augmenter	Si après que l'ennemie ses fais toucher par le projectile si la propriété IsAlive est faut, est le explosionlevel est de 1
MoveTest	MovableEntity	Move	Test si la position d'un MovableEntity change selon se velocity, et traveldirection, quand on appelle la method move.	Si la nouvelle position de l'entity correspond et la valeur correcte du addition/substruction (selon sa direction) de sa position et velocity.
HitTest	PlayerShip	Hit	Test les propriétés de classe du playership, après d'avoir été toucher par un projectile.	Si les points de vie descendent, et les bools IsHit et IsAlive sont vrai ou faux après avoir été toucher, selon le projectile qui la toucher.



ChatGPT

Dans le contexte de ce projet **ChatGPT** a été utiliser à plusieurs stades de développement.

Aux premiers stades du projet, dans la phase de conception du programme, les classes, etc. **ChatGPT** est utiliser pour mieux comprendre les fonctionnalités, et interactions de l'héritage en C#. Les classes **SmartEntity**, **MoveableEntity**, **Projectile**, ce sont exemplaires de l'utilisation correct d'héritage en C#. Ces structures dans le code ont été conçu, en parti, grâce aux explications, et exemplaires fournis par **ChatGPT.**

Durant les stades de développement/programmation du jeu, **ChatGPT** est utiliser pour le renommage de classes, variables, méthodes, pour rendre le code plus compréhensible sans le besoin de commentaires pour chaque ligne de code.
Pendant cette phase de développement **ChatGPT** est aussi utiliser dans l'optimisation de code, pour avoir une meilleure performance, ainsi pour éviter de la redondance.

Conclusion

Généralement le projet c'est bien passer. Mais il y a eu un manque d'organisation de ma parte. Dû à pas avoir bien comprise certaines notions de github/icescrum au début du projet, et d'avoir créer un jeu trop complexe qui ma pris plus de temps à développer.

Base de données (DB)

Introduction

La fonctionnalité de SQL et l'utilisation d'une base de données dans ce jeu servent à offrir une meilleure expérience pour l'utilisateur, ainsi que de pouvoir récolter des données pertinentes. Ce chapitre comprend la partie technique de la base de données du jeu, les différentes requêtes pour la récupération de données, la gestion des utilisateurs/rôles, les index, ainsi que la protection de notre base de données via les sauvegardes et les restaurations.

Gestions des utilisateurs

La gestion des utilisateurs et des rôles est une étape importante pour définir et limité l'accès de certains donnés.

Administrateur de jeu

Création du rôle **r_admin** avec le command **CREATE**.

```
CREATE ROLE IF NOT EXISTS r_admin;
```

Donne les privilèges au rôle **r_admin** avec command **GRANT**. **WITH GRANT OPTION** et ***.*** pour donner tous les privilèges au rôle, donc créer un rôle "super user"

```
GRANT ALL PRIVILEGES ON *.* TO r_admin WITH GRANT OPTION;
```

La création d'un user "admin" avec **CREATE USER**, nommée admin1

```
CREATE USER 'admin1'@'localhost' IDENTIFIED BY 'password';
```

L'ajoute du user admin1 au rôle **r_admin** avec la commande **GRANT ROLE**.

```
GRANT ROLE r_admin TO 'admin1'@'localhost';
```

Joueur

Création du rôle **r_player** avec le command **CREATE**.

```
GRANT ROLE r_player TO 'player1'@'localhost';
```

Donne les privilèges au rôle r_player avec command **GRANT SELECT ON db_space_invaders.t_arme** car le joueur n'a que le droit de récupérer des données sur les armes. **GRANT INSERT, SELECT ON db_space_invaders.t_commade** car le jour à le droit

d'insérer, et récupérer des donnes dans la table commande. Pour que le joueur puisse voir les infos sur les armes, leurs prix, etc. Et ensuite effectuer des commandes.

```
GRANT SELECT ON db_space_invaders.t_arme TO r_player;
GRANT INSERT, SELECT ON db_space_invaders.t_commande TO r_player;
```

La création d'un user "joueur" avec **CREATE USER**, nommée player1

```
CREATE USER 'player1'@'localhost' IDENTIFIED BY 'password';
```

L'ajoute du user player1 au rôle **r_player** avec la commande **GRANT ROLE**.

```
GRANT ROLE r_player TO 'player1'@'localhost';
```

Gestionnaire de la boutique

Création du rôle **r_shopkeeper** avec le command **CREATE**.

```
CREATE ROLE IF NOT EXISTS r_player;
```

Donne les privilèges au rôle **r_shopkeeper** avec command **GRANT SELECT ON db_space_invaders.t_joueur** car le shopkeeper n'a que le droit de récupérer des données sur les joueurs. **GRANT ALTER, SELECT, DELETE ON db_space_invaders.t_arme** pour pouvoir ajouter de nouvelles armes/modifier les prix. Et **GRANT SELECT ON db_space_invaders.t_commade** pour voire toutes commandes faites.

```
GRANT SELECT ON db_space_invaders.t_joueur TO r_shopkeeper;
GRANT ALTER, SELECT, DELETE ON db_space_invaders.t_arme TO r_shopkeeper;
GRANT SELECT ON db_space_invaders.t_commande TO r_shopkeeper;
```

La création d'un user "shopkeeper" avec **CREATE USER**, nommée shopkeeper1

```
CREATE USER 'shopkeeper1'@'localhost' IDENTIFIED BY 'password';
```

L'ajoute du user shopkeeper1 au rôle **r_shopkeeper** avec la commande **GRANT ROLE**.

```
GRANT ROLE r_shopkeeper TO 'shopkeeper1'@'localhost';
```

Requêtes de sélection

Les requêtes de select **SQL** servent à récupérer des donnés depuis la base de données du jeu, pour ensuite afficher/ou en servir pour l'UX, ainsi que le fonctionnement du jeu.

Requête n°1

Cette requête récupère toutes les colonnes de **t_joueur**, les ordonner parce leur nombre de points en décroissant, et limitée à cinq.
Donc elle montre les top cinq joueurs parce nombre de point décroissant va être afficher.

```
SELECT *
FROM t_joueur
ORDER BY jouNombrePoints DESC
LIMIT 5;
```

Requête n°2

Cette requête récupère le **MAX**, **MIN**, et **AVG** du tableaux **t_arme**.
Donc elle récupère l'arme qui coute le plus cher, le moins cher, et le prix moyenne d'une arme.

```
SELECT MAX(armPrix) AS PrixMaximum, MIN(armPrix) AS PrixMinimum, AVG(armPrix) AS
PrixMoyen
FROM t_arme;
```

Requête n°3

Cette requête récupère le **COUNT** de **idCommande** et le **fkJoueur** du tableaux **t_idCommande**, groupé par le **fkJoueur**, et ordonner par le **COUNT** de **idCommande DESC.**
Donc elle récupère le id de chaque joueur et le nombre de commandes qu'ils ont effecteur, ordonner par le **COUNT DESC** pour les organiser par du joueur avec les plus au jouer avec le moins de commandes.

```
SELECT COUNT(idCommande) AS NombreCommandes, fkJoueur AS idJoueur
FROM t_commande
GROUP BY fkJoueur
ORDER BY NombreCommandes DESC;
```

Requête n°4

Cette requête récupère le **COUNT idCommande** et le **fkJoueur** du tableaux **t_commande**, groupé par le **fkJoueur**, et groupée par le fkJoueur **HAVING COUNT** de **idCommande** plus que 2.
Seulement les joueurs qui ont passé plus que 2 commandes vont être afficher.

```sql
SELECT fkJoueur AS idJoueur,
COUNT(idCommande) AS NombreCommandes
FROM t_commande
GROUP BY fkJoueur
HAVING NombreCommandes > 1;
```

Requête n°5

Cette requête récupère le **jouPseudo** de **t_joueur**, le **armNom** de **t_arme**, et **comNumeroCommade** de **t_joueur**. Des **JOIN** sont utiliser pour faire les liaisons entre les différents tableaux à travers de id et fk.
Cette requête affiche les pseudos et les armes qu'ils ont acheté dans chaque commande.

```sql
SELECT jouPseudo, armNom
FROM t_joueur
JOIN t_commande ON fkJoueur = idJoueur
JOIN t_detail_commande ON fkCommande = idCommande
JOIN t_arme ON idArme = fkArme;
```

Requête n°6

Cette requête récupère le **idJoueur** de **t_joueur**, le **SUM armPrix * detQuantiteCommande** de **t_arme** et **t_detail_commande.** Des **JOIN** sont utiliser pour faire les liaisons entre les différents tableaux à travers de id et fk. C'est groupé par le **idJoueur**, ordonnée par **SUM armPrix * detQuantiteCommande DESC**, et **LIMIT** 10.
Cette requête affiche les identifient des joueurs et le total qu'ils ont dépensé, avec seulement le top 10 qui ont dépensé le plus en ordre décroissant.

```sql
SELECT idJoueur, SUM(armPrix*detQuantiteCommande) AS TotalDepense
FROM t_joueur
JOIN t_commande ON fkJoueur = idJoueur
JOIN t_detail_commande ON fkCommande = idCommande
JOIN t_arme ON fkArme = idArme
GROUP BY idJoueur
ORDER BY SUM(armPrix*detQuantiteCommande) DESC
LIMIT 10;
```

Requête n°7

Cette requête récupère **idJoueur** de **t_joueur**, et le **idCommande** de **t_commande**. **t_commande** est relier à **t_joueur** via un **LEFT JOIN**.
Cette requête affiche tous les joueurs, et leurs commandes, ainsi que ceux qui n'ont pas passé de commandes.

```sql
SELECT idJoueur, idCommande
FROM t_joueur
LEFT JOIN t_commande ON fkJoueur = idJoueur;
```

Requête n°8

Cette requête récupère toutes les commandes par joueur,
Cette requête affiche chaque colonne de **t_commande** et la Pseudo de joueur qui là effectuer.

```sql
SELECT t_commande.*, jouPseudo
FROM t_commande
JOIN t_joueur ON fkJoueur = idJoueur;
```

Requête n°9

Cette requête va récupérer la **SUM** (totalité) des armes acheter par chaque joueur.
Il faut utiliser des **LEFT JOIN** pour faire les jointures, pour quoi même les joueurs qui n'on pas acheter d'armes sont afficher dans la requête.

```sql
SELECT idJoueur, SUM(detQuantiteCommande) AS nbTotalArmes
FROM t_joueur
LEFT JOIN t_commande ON fkJoueur = idJoueur
LEFT JOIN t_detail_commande ON fkCommande = idCommande
GROUP BY  idJoueur;
```

Requête n°10

Cette requête va récupérer le idJoueur de chaque joueur qui a acheté plus de 3 armes.
Le **HAVING** est une condition pour quand le **GROUP BY** est utilisé. Le **COUNT** va compter le nombre de **fkArmes** associé a chaque joueur. **Le DISTINCT** est pour que chaque **fkArmes** soit unique.

```sql
SELECT idJoueur
FROM t_joueur
JOIN t_commande ON fkJoueur = idJoueur
JOIN t_detail_commande ON fkCommande = idCommande
GROUP BY idJoueur
HAVING COUNT(DISTINCT fkArme) > 3;
```

Création des index

3.5.1.

MySQL va automatiquement créer des indexes sur les colonnes uniques, les clés primaires, et les clés étrangères.

3.5.2.

Un index va permettre d'optimiser le requête SELECT pour les effectuer plus, en utilisant un B+ tree structure.

3.5.3.

Dans le contexte de la structure de base de données, si le jamais avant un système de commandes et de boutique.
Ça aurait été pertinent de mettre un index sur le champ 'comDate' de la table 't_commande' car lorsqu'une requête vise à recouper des données en fonction de ce champ (avant une certaine date, entre deux dates spécifiques), cela peut prendre du temps. Par exemple, lorsqu'on veut récupérer toutes les commandes effectuées avant le 31 décembre 2023, MySQL devra parcourir tous les jours de tous les mois avant d'atteindre la date du 31 décembre 2023. Ce n'est pas très efficace et cela prendrait plus de temps que nécessaire.

Backup/Restore

Backup

Cette commande mysqldump, qui va être utiliser dans une CMD, va servir à créer une sauvegarde de notre base de donnes, sa structure, et ses données. Le -B est utiliser pour que le fichier backup contient un DROP nom_du_db et USE nom_du_db avant et après le CREATE nom_du_db. Sinon l'exportation du dump, ne marcherais pas sans avoir déjà créer la DB avant.

```
mysqldump -uroot -proot -B db_space_invaders > db_space_invaders_backup.sql
```

Restore

Cette commande serre à créer une nouvelle base de données avec structure et donnes à partir d'un fichier de backup.sql créer avec la commande mysqldump. La commande doit être exécuter depuis le répertoire, où se trouve le fichier backup, sinon le lien du fichier dump dois être mis dans la commande.

```
mysql -uroot -proot < C:\Users\username\..\db_space_invaders_backup.sql
mysql -uroot -proot < db_space_invaders_backup.sql
```