|  |
| --- |
| ETML |
| Projet Space Invaders |
| Space Invaders Pixels Old Classic Retro Game Poster | Space invaders ... |

|  |
| --- |
| Auteur : Tiago Rodrigues Sousa  Chef de projet : Mathieu Meylan / Xavier Carrel / Aurélie Curchod  Version : 1  11/09/2023 |

Table des matières

[1 POO 3](#_Toc148105912)

[1.1 Introduction 3](#_Toc148105913)

[1.2 Analyse fonctionnelle 3](#_Toc148105914)

[1.2.1 Player mouvement 3](#_Toc148105915)

[1.2.2 Ennemy mouvement 3](#_Toc148105916)

[1.2.3 Player shoot 4](#_Toc148105917)

[1.2.4 Change the program structure 4](#_Toc148105918)

[1.2.5 make menu 4](#_Toc148105919)

[1.2.6 Score display 5](#_Toc148105920)

[1.2.7 infinit apparation enemies 5](#_Toc148105921)

[1.2.8 limit ammo 5](#_Toc148105922)

[1.2.9 Finish game 5](#_Toc148105923)

[1.2.10 dataBase connection 6](#_Toc148105924)

[1.3 Analyse technique 6](#_Toc148105925)

[1.3.1 Diagramme de classe 6](#_Toc148105926)

[1.3.2 Explication 8](#_Toc148105927)

[1.4 Test unitaire 12](#_Toc148105928)

[1.4.1 Test déplacement vertical de la balle 12](#_Toc148105929)

[1.5 Conclusion 12](#_Toc148105930)

[2 UX 13](#_Toc148105931)

[2.1 Introduction 13](#_Toc148105932)

[2.2 Analyse 13](#_Toc148105933)

[2.2.1 Conception centrée utilisateur 13](#_Toc148105934)

[2.2.2 Palette graphique 14](#_Toc148105935)

[2.2.3 Eco-conception 14](#_Toc148105936)

[2.2.4 Accessibilité 14](#_Toc148105937)

[2.3 Conception 15](#_Toc148105938)

[2.4 Evaluation/Test 15](#_Toc148105939)

[2.5 Conclusion 15](#_Toc148105940)

[3 DB 16](#_Toc148105941)

[3.1 Importer les données et le schéma de base de données 16](#_Toc148105942)

[3.2 Gestion des utilisateurs 17](#_Toc148105943)

[3.3 Requêtes de sélection 19](#_Toc148105944)

[3.3.1 Explication quelques commandes 19](#_Toc148105945)

[3.3.2 Requêtes 20](#_Toc148105946)

[3.4 Index 24](#_Toc148105947)

[3.5 Sauvegarde/Restore 24](#_Toc148105948)

[3.5.1 Sauvegarde 24](#_Toc148105949)

[3.5.2 Restaurer 25](#_Toc148105950)

# POO

## Introduction

Lors du projet space invaders, il a été demandé de faire un jeu comme space invaders en programmation orienté objet. Cette section va expliquer la partie code du projet. Le cahier des charges exigeant que le programme puisse faire au minium :

* Un vaisseau avec possibilité de tir et déplacement
* 10 ennemis qui descendent sur l’axe vertical

Il a également été demandé d’utiliser IceScrum pour la gestion de projet.

## Analyse fonctionnelle

### Player mouvement

|  |
| --- |
| As a player I want to go left and right with a player |
| Tests d’acceptance :   |  |  | | --- | --- | | Go to the left | In the program when i press "a" the player go to the left | | Go to the right | In the program when i press "d" the player go to the right | | Border left | In the program when i press "a" and the player is on the border left the player stop mouvement | | Border right | In the program when i press "d" and the player is on the border right the player doesn’t move | |

### Ennemy mouvement

|  |
| --- |
| As a player I want to see my enemy move |
| Tests d'acceptance:   |  |  | | --- | --- | | Ennemie show | When the game start Ennemies are displayed (see maquette 1) | | Move to the down | When enemies are in the border left or right enemies move down (see maquette 3) | | Move to the left or right | If enemies are the border right or left and go down ennemies move to the left or right | | Move right at the start | when game start ennemies go to the right (see maquette 2) | |

### Player shoot

|  |
| --- |
| As a player I want to can shoot with de player In order to kill enemies |
| Tests d'acceptance:   |  |  | | --- | --- | | player shoot | In the program when the player press "space" a ammo is shot | | If ammo don't touch enemies | In the program If ammo reaches the border top ammo disappear | | If ammo touch enemies | In the program if ammo touch enemies ammo and enemies disappear | | ammo move | in the program with an ammo shot the ammo moves by itself straight up | |

### Change the program structure

|  |
| --- |
| As a programmer I want to change the program structure to separate the different elements of the program |
| Tests d'acceptance:   |  |  | | --- | --- | | Program separate | In the structure of the program when i see the structure the program is separate in 4 project (model, display, console, storage) | |

### make menu

|  |
| --- |
| As a player I want a menu In order to navigate in game |
| Tests d'acceptance:   |  |  | | --- | --- | | display menu | In the desktop when i start the program he display a menu (see maquette 1) | | go down | In the main menu when i press down arrow Cursor go down | | go up | In the main menu when i press up arrow Cursor go up | | border top/down | In the main menu when cursor is in border top/down and i press up/down arrow Cursor don't move | | go highscore/rules/option | In the main menu when cursor is in highscore/rules/option and press Enter he display a highscore/rules/option page | | quit | In the main menu when cursor is in quit and press Enter He quit the program | | play | In the main menu when cursor is in play and press Enter He start the game | |

### Score display

|  |
| --- |
| As a player I want to see score when i play |
| Tests d'acceptance:   |  |  | | --- | --- | | display score | In the game when the game starts the score is displayed and it's equal at 0 (see maquette 1) | | add score | In the game when player kill an enemie the score add 10 points | |

### infinit apparation enemies

|  |
| --- |
| As a player I want to more enemies appear In order to add more dificult |
| Tests d'acceptance:   |  |  | | --- | --- | | new enemies appear | In the game Every 8 seconds between 5 and 10 enemies appear randomly at the top middle of screen | | all enemies go down | In the game when ONE enemies touch the border left or right ALL enemies go down | | change direction | In the game when enemies go down they change a direction at opposite | |

### limit ammo

|  |
| --- |
| As a player I want to limit a number of ammo In order to have more dificult and not spam |
| Tests d'acceptance:   |  |  | | --- | --- | | display number ammo | In the program When game starts he display a number ammo under the score | | lose ammo | In the game when the player shoot the number of ammo goes down by 1 | | give ammo | In the game when the player kills 5 enemies the number of ammo goes up by 3 | | run out of ammo | In the game with 1 ammo left when the player shoots and does not earn new ammos the player loses game | |

### Finish game

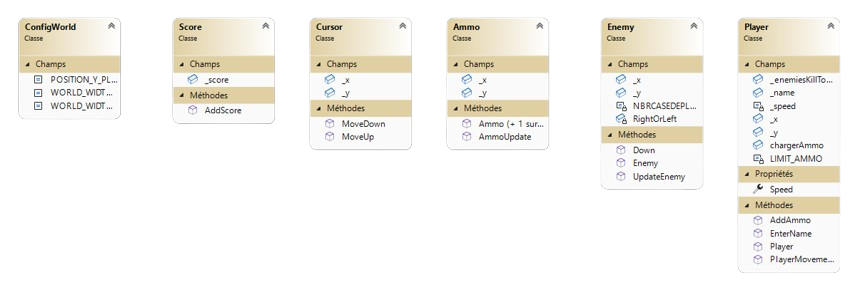
|  |
| --- |
| As a player I want to have end in the game |
| Tests d’acceptance :   |  |  | | --- | --- | | Lose game | In the main program if the enemies are on the same line as the player the console is cleared and he displays a game over screen (see maquette 1) | | Press Enter | In the game over screen When player press enter This returns to the main menu | | Enter a name | In the game over screen When player enter a random name The name is displayed at the screen next to "Enter a name" | |

### dataBase connection

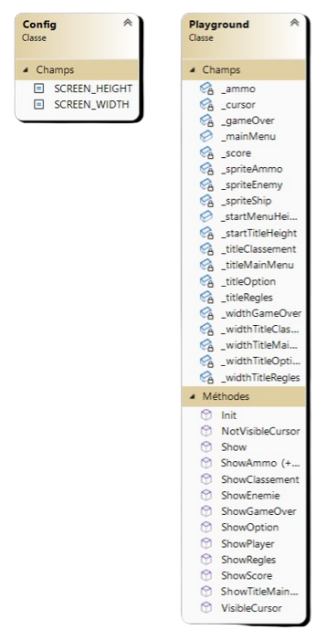
|  |
| --- |
| As a devlopper I want to do the dataBase connection |
| Tests d'acceptance:   |  |  | | --- | --- | | open connection | In the main progam when the highscore page is loaded the database connection is open | | Select scores | In the main progam when the highscore page is loaded a select query is made to get all games data: - the player name - player Score The data is sorted on time ascending, only the first 5 players are selected | | Insert player score | In the main program when the player lose the game her score is save in the database | | close connection | In the main progam When the highscore page is unloaded the database connection is closed | |

## Analyse technique

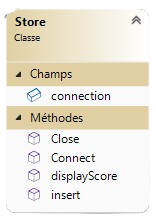
### Diagramme de classe

Model 

Display



Store



### Explication

#### Model.Ammo Class Reference

Public Member Functions

* **Ammo** (int x)

*Constructeur lorsque le joueur tire la balle prend la même position que le joueur.*

* **Ammo** ()

*Constructeur pour mettre des balles dans le chargeur du joueur.*

* void **AmmoUpdate** ()

*met à jour les balles*

Public Attributes

* int **\_x**
* int **\_y** = ConfigWorld.POSITION\_Y\_PLAYER\_FOR\_START

Constructor & Destructor Documentation

Model.Ammo.Ammo (int x)

Constructeur lorsque le joueur tire la balle prend la même position que le joueur.

Parameters

|  |  |
| --- | --- |
| *x* |  |

Display.Config Class Reference

Static Public Attributes

* const int **SCREEN\_HEIGHT** = 40
* const int **SCREEN\_WIDTH** = 150

#### Model.ConfigWorld Class Reference

Static Public Attributes

* const int **WORLD\_WIDTH\_LEFT** = 25
* const int **WORLD\_WIDTH\_RIGHT** = 125
* const int **POSITION\_Y\_PLAYER\_FOR\_START** = 30

#### Model.Cursor Class Reference

Public Member Functions

* void **MoveDown** ()

*Met à jour la position verticale lorsqu'il descend*

* void **MoveUp** ()

*Met à jour la position verticale lorsqu'il monte*

Public Attributes

* int **\_x**
* int **\_y** = 0

#### Model.Enemy Class Reference

Public Member Functions

* **Enemy** (int x)

*Constructeur pour décider ou commence l'ennemie sur la ligne horizontal*

* void **UpdateEnemy** ()

*Va déplacer l'ennemie à gauche ou à droite d'un certain nombre de case*

* void **Down** ()

*Fais descendre l'ennemie de 2 case (hauteur de l'ennemi) et va changer la direction x de l'ennemie, c'est à dire s'il va à gaucher ou à droite*

Public Attributes

* int **\_x**
* int **\_y**

Constructor & Destructor Documentation

Model.Enemy.Enemy (int x)

Constructeur pour décider de la hauteur des ennemies.

Parameters

|  |  |
| --- | --- |
| *x* |  |

#### Model.Player Class Reference

Public Member Functions

* **Player** ()

*Constructeur qui va mettre le nombre maximal de munitions dans le chargeur du joueur*

* void **AddAmmo** (int numberAmmo)

*Va ajouter des balles dans le chargeur*

* void **PlayerMovementUpdate** (int move)

*Va mettre à joueur la position du joueur*

* void **EnterName** (string pseudo)

*le joueur met un pseudo*

Public Attributes

* int **\_x** = 75
* int **\_y** = ConfigWorld.POSITION\_Y\_PLAYER\_FOR\_START
* int **\_enemiesKillToGiveAmmo** = 0
* string **\_name** = ""
* List< **Ammo** > **chargerAmmo** = new List<**Ammo**>()

Properties

* int **Speed** [get]

Member Function Documentation

void Model.Player.AddAmmo (int numberAmmo)

Va ajouter des balles dans le chargeur

Parameters

|  |  |
| --- | --- |
| *numberAmmo* |  |

void Model.Player.EnterName (string pseudo)

Le joueur met un pseudo

Parameters

|  |  |
| --- | --- |
| *pseudo* |  |

void Model.Player.PlayerMovementUpdate (int move)

Va mettre à joueur la position du joueur

Parameters

|  |  |
| --- | --- |
| *move* |  |

#### Model.Score Class Reference

Public Member Functions

* void **AddScore** ()

*Rajoute 10 de score à chaque ennemi tué*

Public Attributes

* int **\_scor****e**

#### Storage.Store Class Reference

Public Member Functions

* void **Connect** ()

*Tente de se connecter à une base de données MySQL.*

* void **Close** ()

*Ferme la connexion à la base de données.*

* void **displayScore** (int widthWorld, int height)

*Va chercher les pdseudos et les scores des 5 meilleurs joueurs à la base de données.*

* void **insert** (**Player** player, **Score** score, int widthWorld, int height)

*Va insérer un score avec le pseudo dans la base de données.*

Public Attributes

* MySqlConnection **connection**

Member Function Documentation

void Storage.Store.displayScore (int widthWorld, int height)

Va chercher les pdseudos et les scores des 5 meilleurs joueurs à la base de données.

Parameters

|  |  |
| --- | --- |
| *widthWorld* |  |
| *height* |  |

void Storage.Store.insert (Player player, Score score, int widthWorld, int height)

Va insérer un score avec le pseudo dans la base de données.

Parameters

|  |  |
| --- | --- |
| *player* |  |
| *score* |  |

## Test unitaire

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nom test | Classe | Méthode testée | Description | Condition réussite |
| AddAmmoTest | Player | AddAmmo | Ajoute des balles dans le chargeur du joueur | Nombre de balle ajoutée = nombre de balle dans le chargeur |
| PlayerMovement  UpdateTest | Player | PlayerMovement  Update | Le joueur se déplace d’un vers la droite ou la gauche | X du joueur augment ou descend d’un |
| DownTest | Enemy | Down | L’ennemi descend de 2 qui correspond à sa taille | Y de l’ennemi augmente de 2 |
| UpdateEnemyTest | Enemy | UpdateEnemy | L’ennemi se déplace d’un vers la droite ou la gauche | X de l’ennemi augment ou descend d’un |
| AddScoreTest | Score | AddScore | Ajoute 10 au score | Si le score augmente de 10 |
| AmmoUpdateTest | Ammo | AmmoUpdate | Test le déplacement vertical de la balle | Si la balle se déplace de -1 à la vertical |

### Test déplacement vertical de la balle

Le test « AmmoUpdateTest » va tester si la méthode « ammoUpdate » déplace correctement la balle. Pour ça on stock la valeur de Y et on utilise la méthode « ammoUpdate » pour la déplacer vers le haut d’une case. Ensuite on compare l’ancienne et la nouvelle valeur et on regarde si la nouvelle valeur à moins un par rapport à la nouvelle.

## Conclusion

Ce projet m’a permis d’approfondir mes connaissances en POO. Le projet a rempli tous les critère dans le cahier des charges. Certaine chose pourrait être améliorer notamment en ce qui concerne la modificabilité de certaine valeur et l’indépendance de certaine classe. Aussi il y a la menue option et le menu règles qui peuvent être implémenter.

L’utilisation d’IceScrum a été compliqué au début car nous n’avons pas commencer tout de suite à l’utiliser. Puis le fait d’avoir un seul Scrum master pour tous les élèves et assez compliqué car techniquement nous ne pouvions pas commencer tant que l’user story n’avait pas été validé.

Le fait que le projet soit un jeu a été assez motivant. Le projet en général était assez amusant et instructive.

# UX

## Introduction

Lors du projet space invaders, un menu pour le jeu a été demandé. Il devait les contraintes suivantes :

* Option permettant de jouer solo ou de jouer en multijoueur
* Plusieurs palettes graphiques
* Plusieurs designs pour les ennemis
* Une page highscore
* Le choix de gameplay

## Analyse

### Conception centrée utilisateur

La conception c’est centré sur 2 catégories de joueur, le joueur occasionnel et le joueur régulier. 2 persona ont été créés en leur donnant les informations suivantes :

* Informations personnelles
* Description/Graphique
* Ce qu’il aime
* Ce qu’il déteste

Les persona ont été générer grâce à chatGPT. Les voici :





Ce choix a été fait pour avoir 2 types de joueur qui sont opposé dans leur manière de consommer et donc de toucher un maximum de personnes.

### Palette graphique

Concernant les couleurs les couleurs choisies, elles s’inspirent du vrai jeu vidéo avec un fond noir et un texte blanc tout en rajoutant en fond, un background avec des étoiles qui rappelle l’espace. Il y a également du vert lorsque on sélectionne quelque chose dans le menu. Cela rajoute un côté rétro au jeu. Il y également un thème clair qui va inverser les couleurs sauf le vert. Voici le code hexadécimal les couleurs utilisés : noir = #000000, blanc = #FFFFFF, vert = #3EBD02.

### Eco-conception

Pour l’éco-conception, plusieurs choses ont été fait en se basant sur les 115 règles de bonne conception :

* Un thème sombre a été fait pour limiter la consommation d’énergie des écrans
* Peu d’animation ont été intégré pour limiter la consommation
* Le parcours de l’utilisateur à été optimiser pour éviter une perte de temps dans les menus

### Accessibilité

Pour l’accessibilité, il y a plusieurs choses. Le jeu est disponible en français et en anglais, le but étant de toucher un public plus large en le rendant disponible aux anglophones.

Il est possible de configurer les touches pour toutes les personnes ayant des problèmes d’articulation aux mains. Cela leur permet de soulager leur main en se mettant dans une configuration adaptée.

Le jeu est entièrement disponible au clavier.

Il y a un thème clair qui permet aux personnes ayant du mal à voir avec le thème sombre de pouvoir un autre thème qui permet de mieux voir.

Il y a une possibilité d’enlever le fond qui peut gêner la vision du jeu.

Il y a plusieurs difficultés permettant à des gens moins expérimentés de quand même pouvoir jouer sans frustration. Cela permet également au gens voulant du défi de ne pas quitter le jeu car il est trop facile.

## Conception

Lors de la conception, deux designs ont été fait. Un qui va être intégrer et l’autre qui ne sera pas intégrer.

Pour le menu qui a été intégrer, des contraintes techniques dut au fait que le programme est fait en mode console ont dut être pris compte. Pour cette raison, le menu est très simpliste et n’ai que composé d’un fond noir et du texte blanc. Seulement un menu principal, de règles, d’options et de configurations des touches ont été désignée. L’inspiration principale vient surtout des ancien jeux arcades (Pac-Man, Space Invaders).

Pour le menu qui ne sera pas intégrer, les couleurs choisies reste fidèles au vrai jeu avec un fond noir du texte blanc et le curseur du joueur en vert. Il existe un thème clair pour ceux qui préfèrent. Cela donne un côté rétro au jeu. Le design du menu principal a été inspiré par un jeu (Iconoclasts), cela donne un côté moderne au jeu. Le reste des menus des options a été conçu pour être simple mais efficace avec des options pour améliorer le confort de jeu.

## Evaluation/Test

Voici les tests d’utilisabilité :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Menu** | **Description** | **Précondition** | **Étape** | **Donnée** | **Résultat attendu** |
| Classement | Le but est de voir le classement des 5 meilleurs joueurs local. | Avoir  l’application installé | Dans le menu,  cliquez sur classement | Aucun | Classement des 5  meilleurs joueurs |
| Lancer une  partie en ligne | Le but est de lancer  une partie en ligne | Avoir  l’application installé,  être  connecté  à internet | Dans le menu,  cliquez sur Jouer  Cliquez sur 2P  Cliquez sur En ligne | Aucun | Une page avec marqué  “Recherche d’un autre joueur” |
| Sélectionner un vaisseau | Le but est de sélectionner  un des vaisseau débloqué | Avoir  l’application installé | Dans le menu,  cliquez sur Jouer  Cliquez sur 1P  Cliquez sur Facile  Cliquez sur un vaisseau débloqué | Aucun | Une image 3D du  vaisseau sélectionné avec ses stats et son nom |
| Changer la langue | Le but est de changer la langue | Avoir  l’application installé | Dans le menu,  cliquez sur option  Cliquez sur  changer la langue | Aucun | Cela va changer le jeu en une autre langue (FR -> ENG, ENG -> FR) |
| Changer de thème | Le but est de changer de thème | Avoir  l’application installé | Dans le menu,  cliquez sur option  Cliquez sur  thème | Aucun | Cela va le thème sombre en clair et vice versa |

## Conclusion

Ce projet a permis d’utiliser la théorie dans un cas concret. Pour ma part j’ai eu beaucoup de mal pour la partie design. Pour le reste cela était assez divertissant de faire une maquette sur un jeu. Toute les demandes dans le cahier des charges ont été respectées. Si je devais refaire le projet, j’aurais essayé de faire la maquette d’en un design plus démarqué et moins simpliste/minimaliste. La partie prototype était assez amusante.

# DB

## Importer les données et le schéma de base de données

Lancez le terminal (conseil : allez dans l’endroit où il y a la base de données est dans le chemin d’accès mettez cmd) :

Une image contenant texte, ligne, logiciel, capture d’écran

Description générée automatiquement

Puis mettez la commande suivante :

Une image contenant texte, capture d’écran, carte de visite, Police

Description générée automatiquement

Docker = met en lien avec l’application docker

exec = exécuter

-i = identifiant docker

Id\_docker = identifiant du container

mysql = met en lien avec MySQL

-u = utilisateur MySQL

-p = mot de passe

root = nom et mot de passe admin dans MySQL par défaut

< = importer vers MySQL

db\_space\_inavaders.sql = chemin d’accès à la base de données

(PS : Pour copier l’identifiant docker, il suffit d’aller dans docker, puis le menu des containers et de cliquer sur l’icône pour copier)

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Normalement cela devrait avoir implémenter la base de données sur le serveur mysql.

## Gestion des utilisateurs

Tout d’abord nous allons créer 3 utilisateurs : Alice (administratrice de jeu), Bob (joueur), Charly (gestionnaire de la boutique)

Pour créer les utilisateurs, nous allons utiliser la commande ci-dessous :

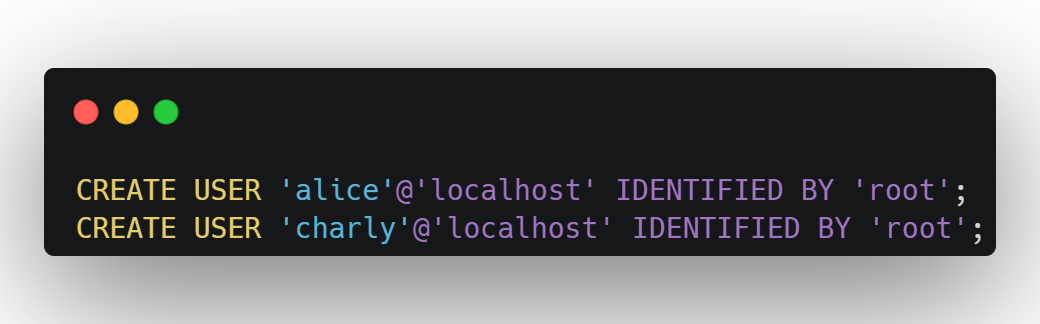


‘bob’ = nom de l’utilisateur

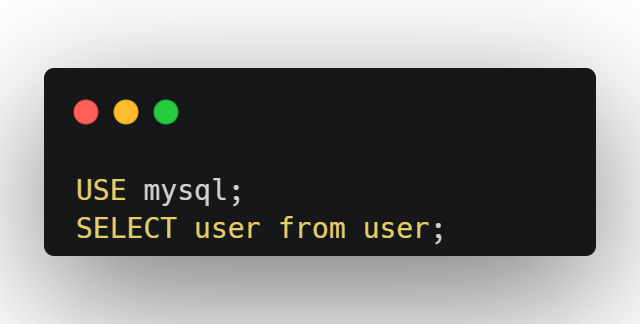
‘localhost’ = L’host depuis lequel il a le droit de se connecter

‘root’ = mot de passe

Je vais le faire pour les 2 autres utilisateurs.



On va vérifier si les utilisateurs ont été créer, pour cela il faudra sélectionner la base de données MySQL puis faire une requête.



Normalement vous devriez voir les différents nom des utilisateurs.

Maintenant nous allons créer les rôles pour les différents besoin (administratrice de jeu, joueur, gestionnaire de la boutique). Le but d’un rôle est de pouvoir donner ce rôle à beaucoup des utilisateurs sans avoir à redonner des rôles à chaque fois.

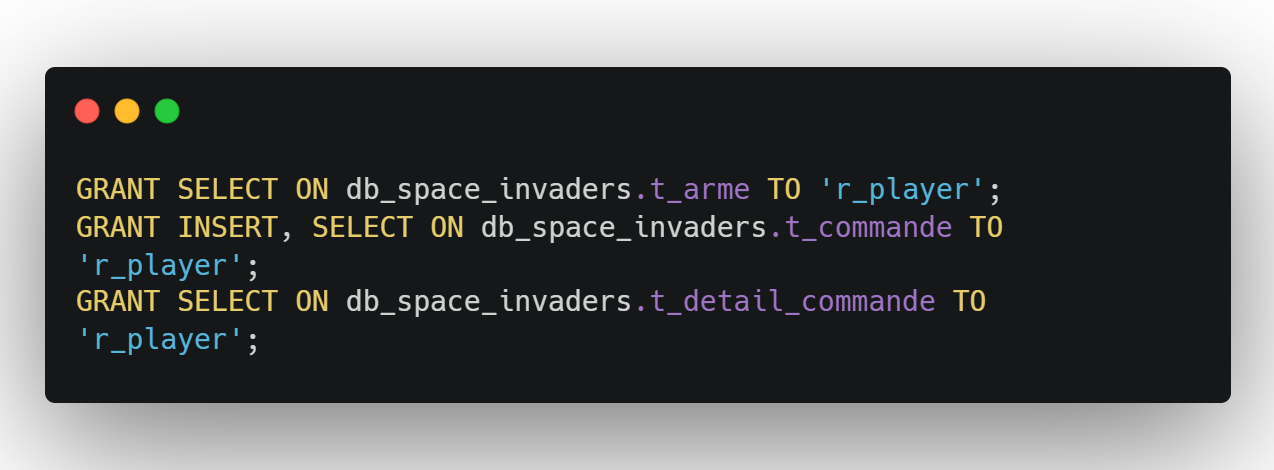
Donc pour créer un rôle, il suffit de faire la commande ci-dessous :



‘r\_player’ = nom du rôle

Ensuite on va donner aux rôles des privilèges, pour ça nous allons faire la commande suivante :





(Ici on va rajouter un droit sur la table joueur car il en a besoin pour créer une commande) Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Explication de la requête :

GRANT = indique qu’on va donner quelque chose

SELECT, ALTER, DELETE, … = les permissions données

ON db\_space\_invaders.t\_joueur = base de donnée.table (\* = tout)

TO ‘r\_shopkeeper’ = indiquer qui est concerné par cette commande (par exemple : r\_shopkeeper)

Maintenant on va donner les rôles aux utilisateurs, pour ça on va utiliser la commande suivante :



Cela va donner les rôles au différents utilisateurs (à noter qu’on peut rajouter plusieurs utilisateurs à la fois).

## Requêtes de sélection

### Explication quelques commandes

SELECT = permet de sélectionner des colonnes

FROM = indique depuis quelle table on fait le select

WHERE = permet de mettre une condition aux données

ORDER BY = permet de mettre en ordre les résultats (par défaut croissant mais peut être décroissant grâce au mot DESC)

LIMIT = permet de mettre un nombre défini de résultat

MAX = sélectionne le plus grand

MIN = sélectionne le plus petit

AVG = Fais la moyenne

AS = permet de renommer une méta donnée (si on l’utilise, on doit utilisé le surnom si la méta donnée réapparait)

COUNT = compte le nombre de données lorsqu’il y a un group by

GROUP BY = permet de regroupé par ligne qui ont la même donnée

HAVING = permet de mettre une condition au regroupement, ne peut être utilisé qu’après des GROUP BY

INNER JOIN (ou JOIN) = permet de joindre 2 tables grâce aux clés primaires (id) et aux clés étrangères (fk) qu’on va regrouper avec le ON et puis regroupe uniquement ceux qui sont en commun

LEFT JOIN = permet de joindre 2 tables grâce aux clés primaires (id) et aux clés étrangères (fk) qu’on va regrouper avec le ON et puis regroupe tout ce qui sont en commun puis toutes les informations de la table qui est sur le FROM même s’il n’est pas sur l’autre table

RIGHT JOIN = la même chose que LEFT JOIN mais au lieu de mettre toutes les informations de la table qui est sur le FROM, ce sera la table avec laquelle on va relier (après le RIGHT JOIN) qui aura toutes les informations même s’il n’apparaît pas sur l’autre table

SUM() = va faire la somme de tous les chiffres

DISTINCT = permet d’éviter les doublons

### Requêtes

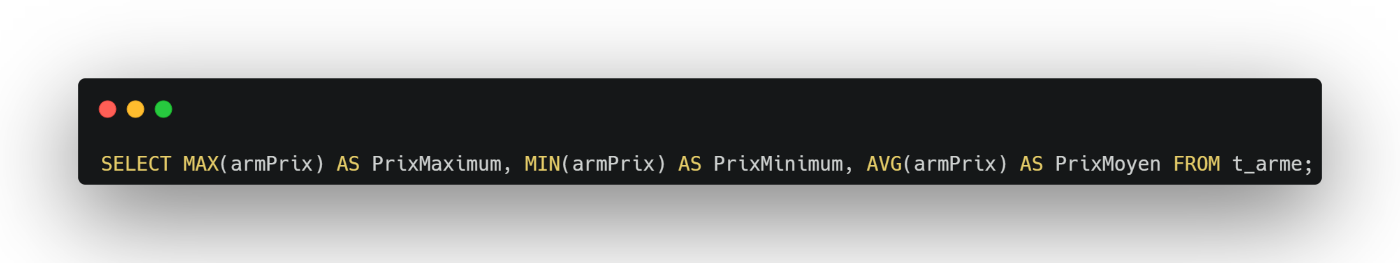
Voici les différentes requêtes :

Requête n°1 :



Cela va tout (\*) sélectionner [**SELECT**] de la table *t\_joueur* (FROM), par ordre (ORDER BY) décroissant (DESC) par rapport au nombre de point et se limitant (LIMIT) au 5 premier résultat.

Requête n°2 :



Cela va sélectionner l’arme la plus chère (MAX()) et la renommer PrixMaximum grâce au « *AS* », l’arme la moins chère (MIN()) et la renommer PrixMinimum et le prix moyen des armes (AVG()) et la renommer PrixMoyen depuis la table t\_arme.

Requête n°3 :

Cela va sélectionner l’id des joueurs la renomment idJoueur, le nombre de commande (COUNT()) et le renommer NombreCommande depuis la table t\_commande. Puis les grouper (group by) par l’id des joueurs et les ordonner par le nombre de commande décroissant.

Requête n°4 :

Cela va sélectionner l’id des joueurs la renomment idJoueur, le nombre de commande et le renommer NombreCommande depuis la table t\_commande. Puis les grouper par l’id des joueurs qui ont plus de 2 commandes (HAVING).

Requête n°5 :

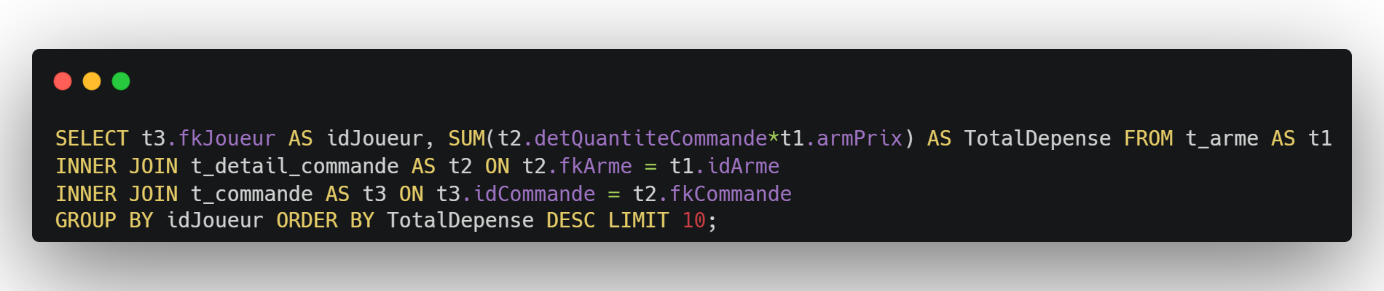
Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Cela va sélectionner le nom du pseudo (table : t\_joueur), l’id de la commande (table : t\_commande) et le nom de l’arme (table : t\_arme). Puis on va joindre la table t\_joueur et la table t\_commande via l’id du joueur (INNER JOIN), on va joindre la table t\_detail\_commande et la table t\_commande via l’id des commandes et on va joindre la table t\_detail\_ commande et la table t\_arme via l’id de l’arme.

L’INNER JOIN a été choisi car il regroupe uniquement ce qui est en commun car certaines personnes n’ont pas acheté d’arme.

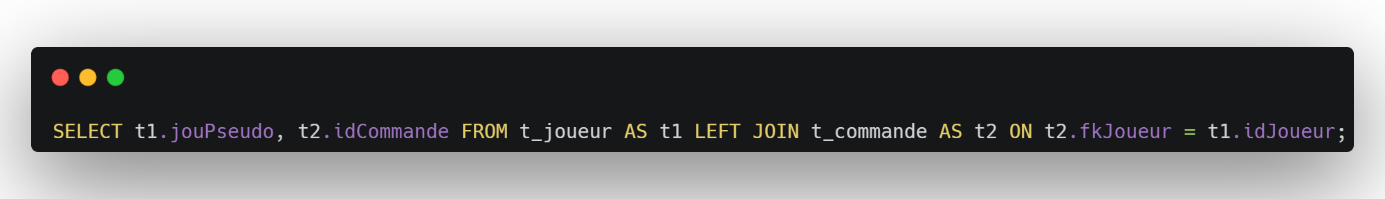
Requête n°6 :



Cela va sélectionner l’id des joueurs, la somme de tous les calculs prix de l’arme \* la quantité, tout ça regroupé par joueur. On va joindre t\_detail\_commande à t\_arme puis on va regrouper t\_commande avec t\_detail\_commande pour avoir accès à l’id des joueurs et au nombre de quantité de commande.

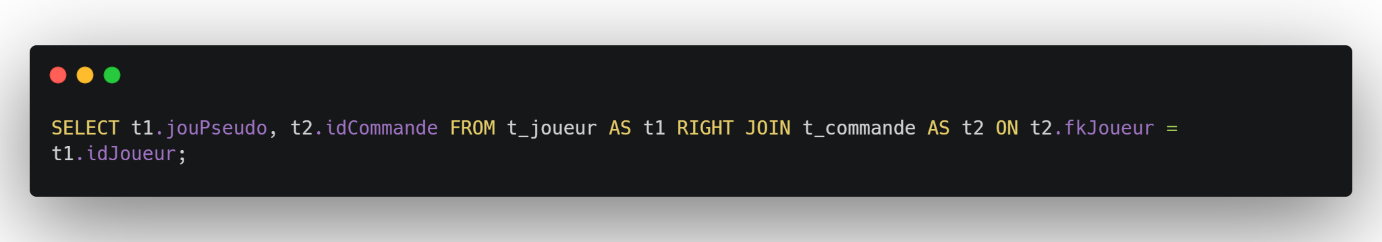
On va regrouper le tout par joueur, mettre en ordre décroissant par le total dépensé et ne mettre que les 10 premières réponses.

Requête n°7 :



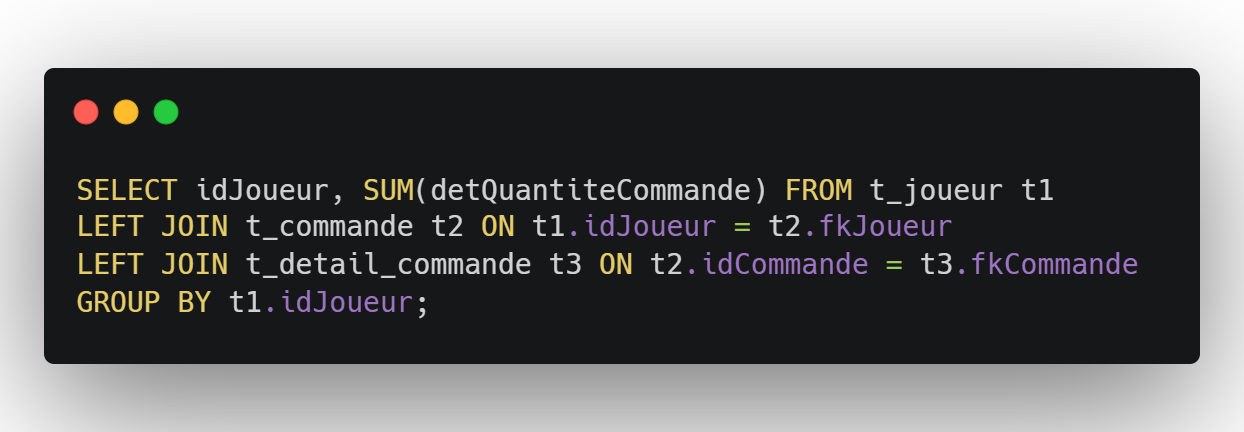
On va sélectionner le pseudo des joueurs et leur commande. On va relier les joueurs à leur commande via l’id des joueurs. Ici on fait un LEFT JOIN pour avoir accès à tous les joueurs même s’ils n’ont pas passées de commandes.

Requête n°8 :



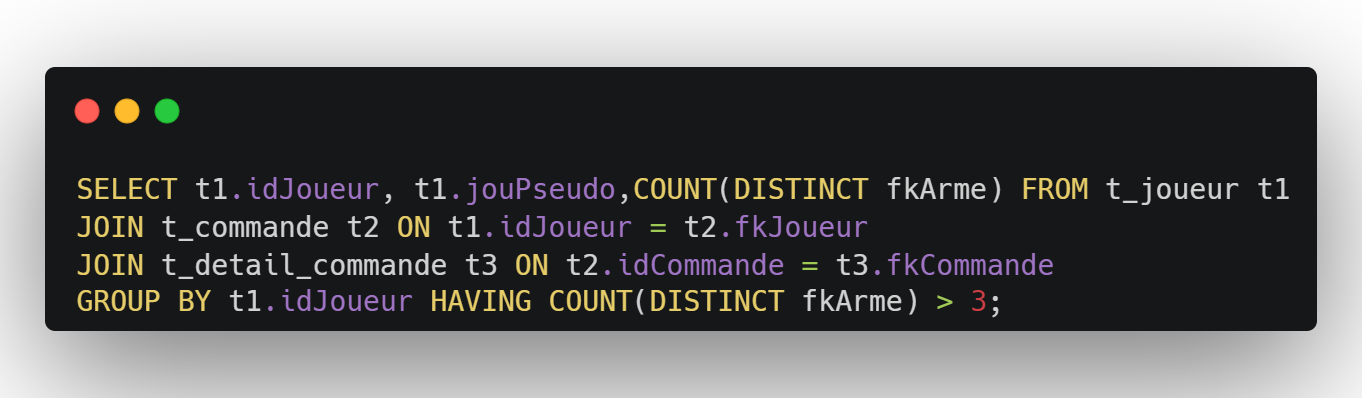
On va sélectionner le pseudo des joueurs et leur commande. On va relier les joueurs à leur commande via l’id des joueurs. Ici on fait un RIGHT JOIN pour avoir accès à toutes les commandes même si le joueur n’existe plus.

Requête n°9 :



On va sélectionner l’id des joueurs, la somme (SUM()) de ce qu’ils ont commandé. Puis on va joindre la table des joueurs et des commandes pour savoir quel joueur à fait quelle commande puis la table des commandes avec celle des détails des commandes pour avoir la quantité de chaque commande. Le tout regrouper par joueur. On utilise des LEFT JOIN pour avoir accès à tous les joueurs même s’ils n’ont rien commandé.

Requête n°10 :



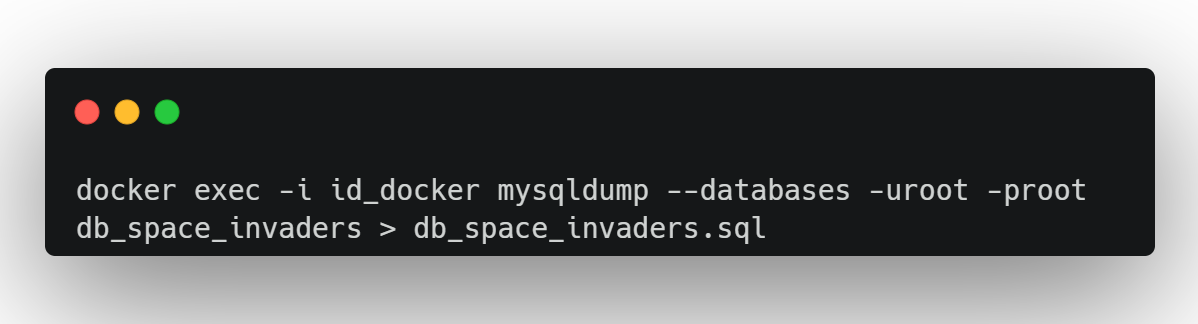
On va sélectionner l’id, le pseudo et on va compter le nombre d’arme différents dans la liste de commande de chaque joueur. Le DISTINCT permet d’éviter que 2 commandes avec la même arme fait par le même joueur ne soit compter 2 fois. Puis on va joindre la table des joueurs et des commandes pour savoir quel joueur à fait quelle commande puis la table des commandes avec celle des détails des commandes pour savoir quelle arme est commandé. Le tout regrouper par joueur et on va trier par le nombre d’arme différente que chaque joueur a commandé, on va uniquement garder ceux qui ont commandé plus de 3 armes différentes.

## Index

1. Les 3 index qui sont créés automatiquement sont pour les colonnes id (clé primaire), fk (clé étrangère), et les colonnes qui sont uniques. Ils sont créés automatiquement car les id et les fk sont souvent utilisés pour les requêtes se qui permet de les accélérer. Pareil pour les champs uniques.
2. Le but d’un index est d’accélérer les requêtes selectes en accélérant la navigation dans les colonnes. Pour cela il crée un arbre B-Tree. Cela permet d’aller plus vite dans les requêtes selectes mais en contrepartie cela va prendre plus de mémoire et va rendre les insertions, modifications et les suppressions plus lentes.
3. Toutes les tables sont souvent mises à jour sauf la table arme. Cela pourrait être intéressant de mettre un index composite dans l’ordre suivant : armNom, armPrix, armDescription, armForce et armNombreCoups. Cela serait intéressant dans le cas où on veut souvent savoir des informations sur les armes, par exemple pour voir l’inventaire ou bien s’il y a un shop ce qui nécessiterait de souvent faire un select. L’index composite serait utile car lorsque on voit une arme on affiche souvent les informations la concernant et ce ne que rarement qu’on update la table.

## Sauvegarde/Restore

### Sauvegarde

Voici la commande pour faire une sauvegarde : 

docker exec -i id\_docker = cette partie sert à utiliser le docker où il y a la base de données (remplacé id\_docker par l’id de votre docker

mysqldump = mysqldump est un utilitaire permettant de sauvegarder et de restaurer des bases de données MySQL

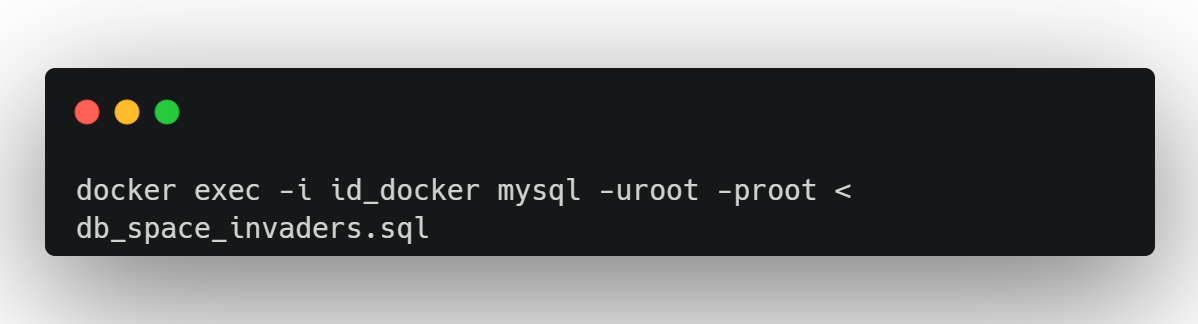
--databases = va rajouter dans le fichier un code qui crée la base de données, cela évite d’avoir un problème lors de la restauration car il faut lui donner une base de données ou la créer

-uroot -proot = permet de se connecter à la base de données (-u = utilisateur, -p = mot de passe)

db\_space\_invaders = nom de la base de données qu’on aimerait sauvegarder

> db\_space\_invaders.sql = nom du fichier dans le lequel on va stocker la base de données

### Restaurer

Voici la commande pour restaurer une base de données : 

docker exec -i id\_docker = cette partie sert à utiliser le docker où il y a la base de données (remplacé id\_docker par l’id de votre docker)

mysql = va mettre un lien avec MySQL

-uroot -proot = permet de se connecter à la base de données (-u = utilisateur, -p = mot de passe)

< db\_space\_invaders.sql = chemin d’accès vers la sauvegarde