**Rapport de projet Agar.io**

**BONNOT Manon - CLOCHARD Baptiste**

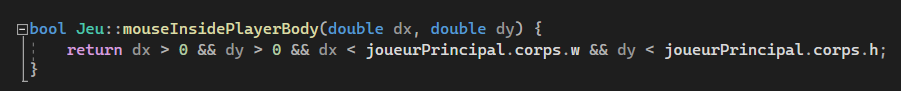
**Introduction**

Lors de la première séance, nous avons travaillé chacun de notre côté sur la base du jeu, à savoir les déplacements aléatoires des ennemis. À la fin de la séance, nous avons décidé de prendre le projet de Baptiste comme base pour l’ajout des extensions. Nous avons décidé de choisir comme extensions le « Mode immobile » et le « Laché de lest ». Vu que nous avions encore du temps après ces deux ajouts, nous avons ajouté l’extension « Affichage cercle ».

**Ajout 1 : Mode immobile**

Cette extension permet au joueur de rester immobile s’il rapproche sa souris du centre de son personnage.

Dans Jeu.cpp, nous avons ajouté une fonction permettant de détecter si le joueur à positionné sa souris à l’intérieur du sprite de son personnage.



Nous avons aussi ajouté une méthode stop() dans Joueur.cpp qui met la vitesse en x et en y à 0. Cependant, lorsque nous avons mis la valeur à 0, nous avons eu un problème. Le jeu s’affichait en blanc et plus aucune action n’était possible. Nous pensions que cela était dû à une division par 0, mais lorsque nous avons tester de « catch » l’erreur, aucune exception n’était levée. Nous avons donc trouvé comme alternative de mettre une vitesse quasi inexistante pour stopper le joueur, l’arrondi se faisant au pixel près, le joueur est donc immobile.

Une image contenant texte

Description générée automatiquement

Ensuite dans la méthode tick() dans Jeu.cpp, nous avons ajouté une condition pour lancer la fonction stop() si la fonction mouseInsidePlayerBody() renvoie true, sinon pour effectuer un déplacement classique.

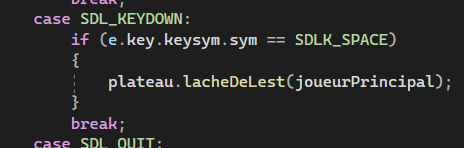
Une image contenant texte

Description générée automatiquement

**Ajout 2 : "Lâché de lest"**

Le but de cette extension est de permettre au joueur de fuir lorsqu’un ennemi trop dangereux s’approche de lui. Pour cela, le joueur appuie sur la touche « ESPACE », ce qui divise sa taille en deux, augmente sa vitesse et créer un ennemi généré à partir de la taille qu’il a perdue.

Lorsque la touche « ESPACE » est utilisée, dans Jeu.cpp, la méthode lacheDeLest() de la classe Plateau.cpp est appelée.



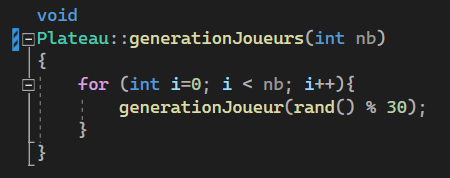
La méthode va vérifier que le joueur à une taille suffisante afin de l’empêcher d’utiliser la fonctionnalité à l’infini. Nous avons décidé par empirisme que le joueur doit avoir une taille supérieure à 25. Si le joueur respecte la condition, les méthodes suivantes sont appelées :

Une image contenant texte

Description générée automatiquement

* joueur.diviseTaille() va diviser la hauteur et la largeur du joueur en 2. Ce dernier étant un cercle, « w » et « h » sont égaux.
* generationJoueur(taille\_joueur) va créer un nouveau joueur IA de la taille souhaitée. Nous avons modifié la méthode présente dans le code, à savoir generationJoueurs(nombre\_de\_joueur), pour que cette dernière ne contienne plus qu’une boucle for qui appelle la méthode generationJoueur(taille\_joueur) autant de fois qu’il y a de joueurs IA à créer :

Nombre de joueur à créer : 1 seul dans le cas de « lacheDeLest() »

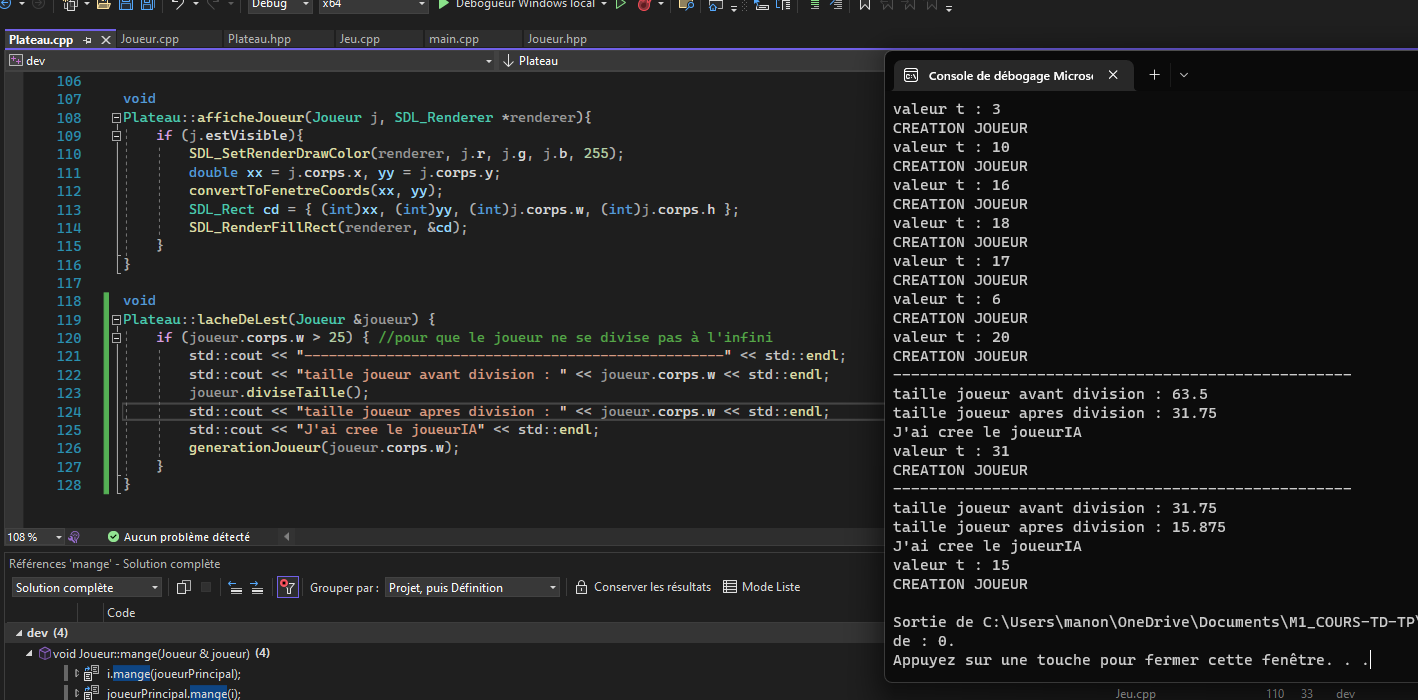
****

Taille du joueur à créer : taille du joueur humain/2 dans le cas de « lacheDeLest() »

**Une image contenant texte

Description générée automatiquement**

Ces modifications nous permettent de prendre en compte uniquement le nombre dejoueurs à créer en début de partie lors de la génération des ennemis, la taille étant définie aléatoirement, et de prendre en compte une taille bien précise lors de l’utilisation du lacheDeLest().



2

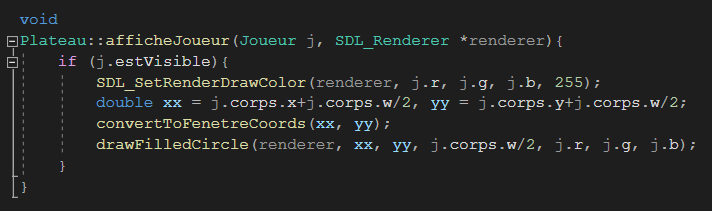
1

Création des joueurs IA en début de partie

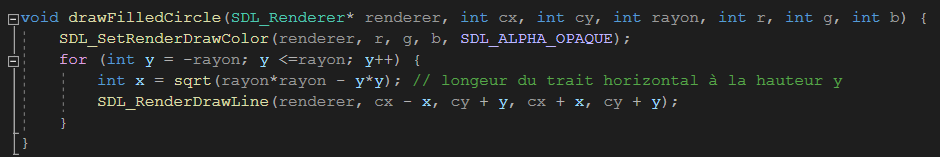
Figure 1 : Double utilisation de lacheDeLest() avant d'atteindre une taille trop petite

**Ajout bonus : "Affichage cercle"**

Comme dis précédemment, nous souhaitions compléter notre travail par un ajout bonus. Nous avons donc décidé de changer le type d’affichage afin de nous rapprocher de celui du jeu original.

Nous avons donc changé l’affichage des joueurs dans la méthode afficheJoueur() :

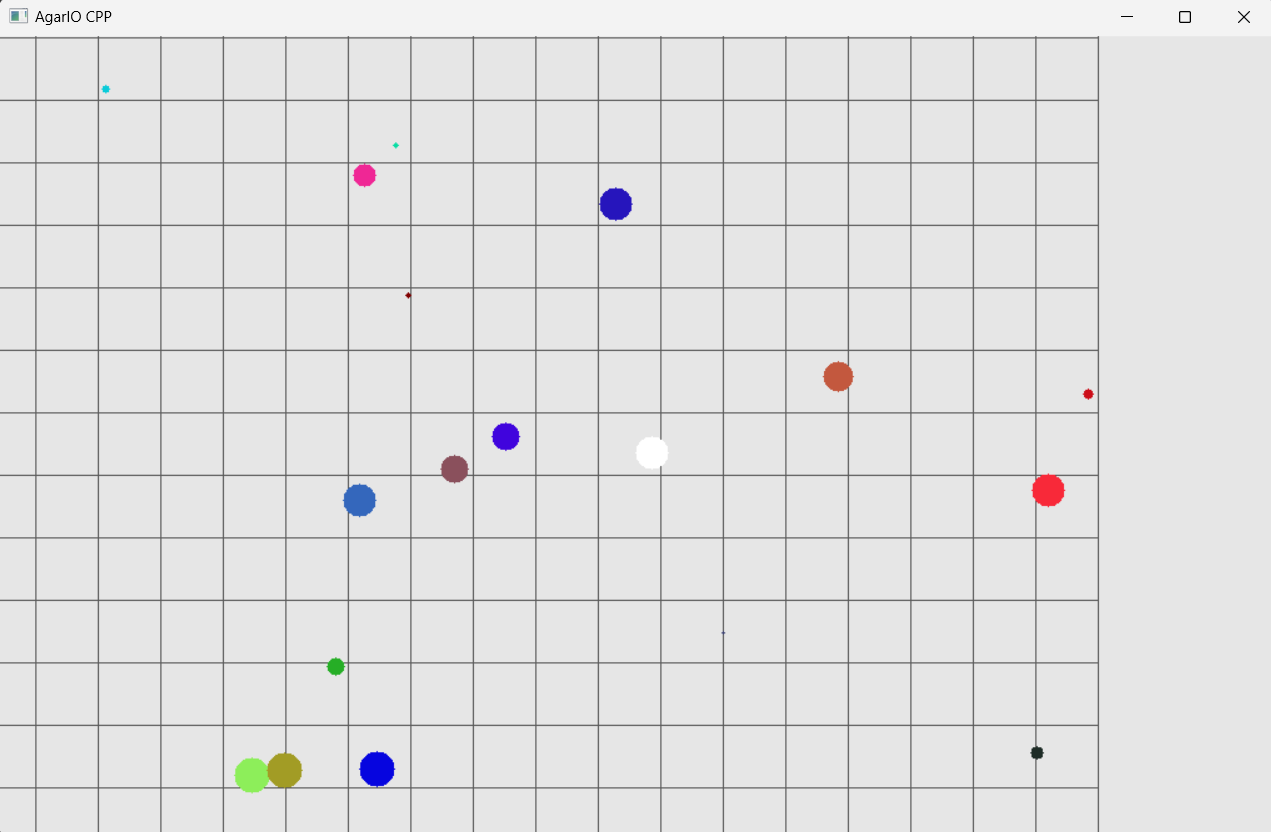
Nous calculons d’abord le rayon du cercle avant de convertir les valeurs en coordonnées et de dessiner le cercle grâce à notre méthode drawFilledCircle(). Cette dernière nous permet de dessiner les cercles représentant les joueurs ligne par ligne en fonction du rayon de ces derniers.



Ensuite nous avons tenté de modifier les collisions de sorte que la boite de collision de chaque joueur corresponde au nouvel affichage. Cependant nous avons rencontré un problème : les collisions circulaires étaient détectées trop tôt sur un des côtés des sprites et trop tard sur l’autre côté. Nous n’avons pas réussi à comprendre l’origine de ce bug, c’est pourquoi nous avons décidé de conserver les boites de collisions carrées.

Une image contenant texte

Description générée automatiquement



**Conclusion**

Malgré les quelques petites difficultés rencontrées, nous avons réussi à trouver des solutions afin d’implémenter les différentes extensions que nous avions choisies. Notre projet est disponible sur notre Github : <https://github.com/DRUM0ND/agario_m1>.