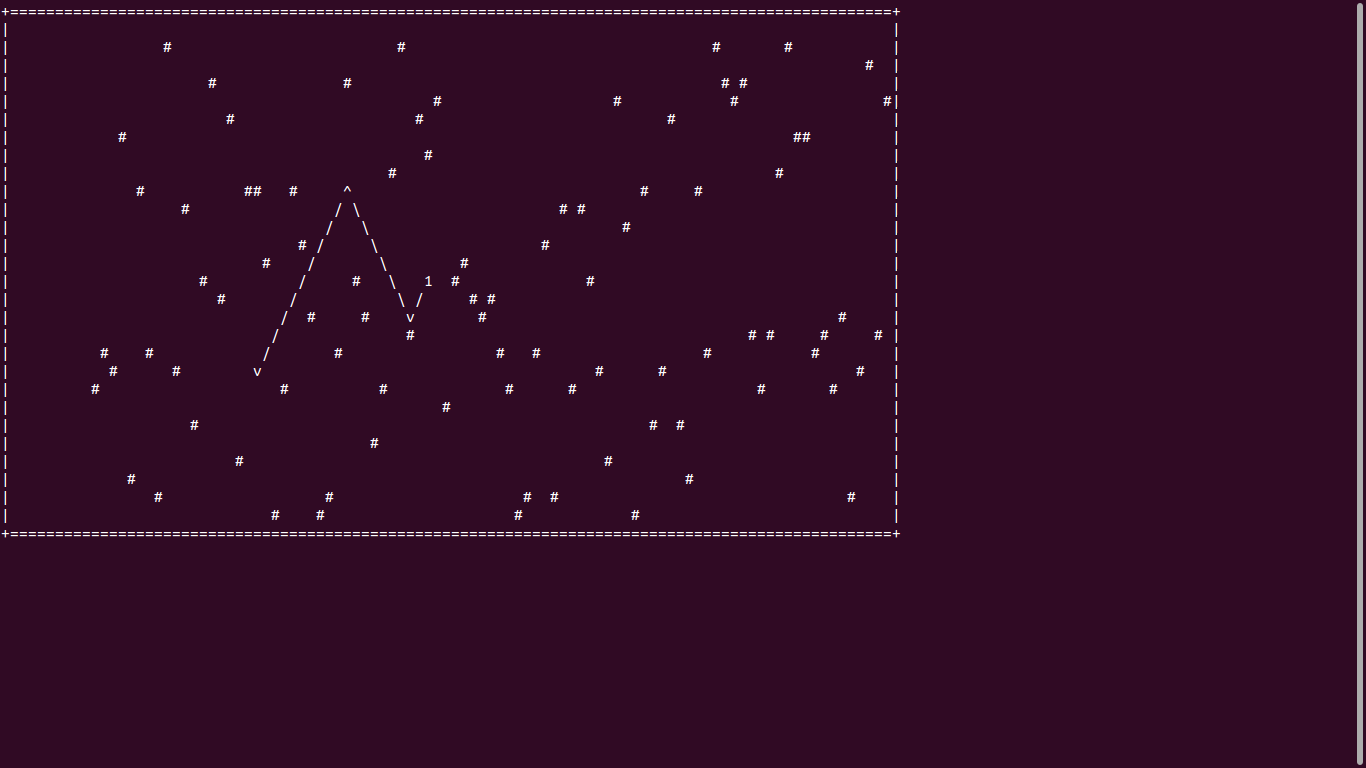
Présentation du jeu :

Dans notre jeu Diagonal Rush, le joueur incarne un serpent se déplaçant dans une carte remplie, ou pas, d’obstacles à éviter. Celui-ci ne peut s’arrêter et ne peut bouger qu’en diagonal, dans toutes les directions.



Une fois arrivé à une des extrémités horizontales de la carte, il repart dans le sens inverse. Attention, le plafond, le sol, et beaucoup d’autres objets sont, eux, mortels ! Le but principal du jeu est donc de survivre le plus longtemps sans toucher le haut ou bas de la carte, tout en évitant les obstacles et sa propre queue.

Le programme du jeu a donc été divisé en plusieurs parties avec chacune son rôle précis : la carte, l’affichage et le déplacement, les informations de blocs et la boucle principale de jeu .  
Tout d’abord pour ce qui est de la carte, on a décidé d’utiliser une grille à deux dimensions qu’on utiliserait comme des coordonnées.

Pour ce qui est de personnage, on le sépare en deux parties, la tête que l’on contrôle et la queue qui suit le même chemin que la tête

L’affichage sera géré à l’aide Ncurses, une bibliothèque qui donne l’accès à une interface plus interactive.

De plus, nous avons rendu chaque partie du code indépendante, pour avoir un code plus lisible, et que cela soit plus facile de régler tout problème rencontré.

l

l

l

l

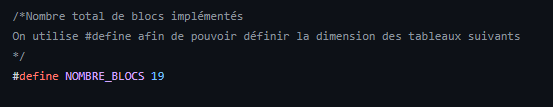
l

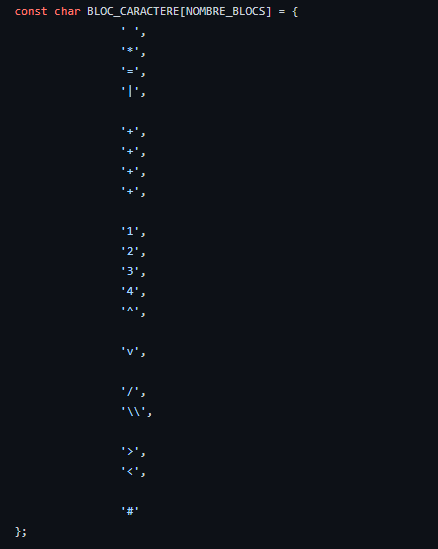
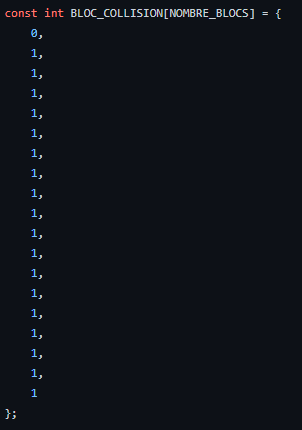
Mais on rencontre alors plusieurs difficultés : Comment afficher en temps réel des objets définis seulement par des nombres dans une grille ? De plus, comment permettre à la queue du personnage de se déplacer, tout en devant toujours suivre l’ancienne position de la tête ?

La carte étant un tableau d’entiers à deux dimensions, nous devions trouver un moyen d’afficher, à l’aide de caractères spéciaux, la carte avec tous ses composants : c’est-à-dire le personnage, les obstacles, ou autres. De plus celle-ci doit changer perpétuellement avec les déplacements du personnage : il faut donc pouvoir la modifier en temps réel et l’actualiser.

Pour l’affichage du jeu, chaque bloc associe à son numéro un caractère spécial. Il fallait donc trouver un moyen d’associer ces numéros à leurs caractères. Face à cela, nous avions pensé à utiliser des ‘switch‘ afin d’associer à chaque valeur un caractère. Cependant, à mesure que nous aurions amélioré le jeu, de nouveau bloc avec d’autres fonctions auraient vu le jour, ce qui aurait augmenté de plus en plus la taille du switch. C’est pour cela que nous avons décidé d’utiliser un tableau de caractère, où l’indice de chaque case du tableau correspond au numéro d’un bloc, et en valeur correspond le caractère de ce bloc.

De la même manière que pour les caractères, nous avons fait un tableau d’entiers, compris de nombres entre 0 et 1, correspondants à la collision éventuelle du bloc ( 0 = le personnage ne peut pas entrer en collision avec ce bloc, 1 sinon).

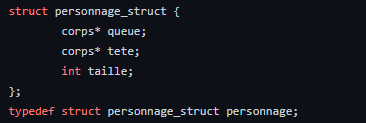


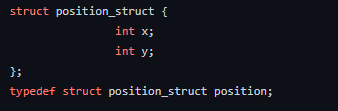
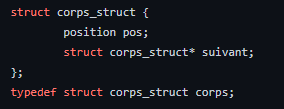
Pour l’exemple le caractère le bloc de numéro 3 a pour caractère ‘|’, correspond à une barre horizontale et possède une collision.

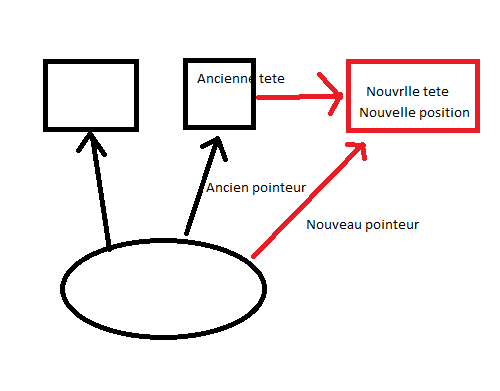
II]

Une fois l’affichage terminé, nous nous sommes penchés sur le déplacement du personnage et en particulier de la queue de celui-ci. Le problème était de garder en mémoire le trajet de la tête, nous ne pouvions pas faire un tableau étant donnée que la queue est de taille variable et, dans le cas où le personnage augmente de taille considérablement, la taille du personnage aurait pu dépasser la taille du tableau. Un tableau ne permettant qu’une allocation dans la pile, c'est-à-dire une allocation de mémoire statique, nous avons donc décidé de faire une liste chainée car celle-ci permet d’avoir une allocation de mémoire dynamique de façon à pouvoir agrandir la queue si besoin. On a donc créé une structure ‘personnage’ avec deux champs de pointeurs, la première pointant vers la tête et la seconde vers la queue. A cela on ajoute un champ d’entier comprenant la taille du personnage, c’est-à-dire le nombre de ‘corps’ entre la tête et la queue.



Chaque maillon (corps) formant le personnage garde en mémoire les coordonnées d’une ancienne position de la tête, ainsi qu’un pointeur vers le corps suivant.





L’ouverture]

Pour le moment les obstacles dans la carte sont créés de façon aléatoire, et la partie est bel et bien perdu au contact de ceux-ci. Malheureusement, la taille de la carte ne dépend pas de la taille de la fenêtre. Il serait intéressant de rajouter des niveaux avec des structures tel que des rampes ou de pouvoir modifier la difficulté. Nous souhaiterions aussi améliorer les graphismes en implémentant l’encodage Unicode plutôt que Ascii. Et pour plus de diversité, ajouter différents bonus, ou blocs spéciaux aux différents effets comme augmenter la taille ou la vitesse, changer de direction ou la carte et pour finir créer un système de score.