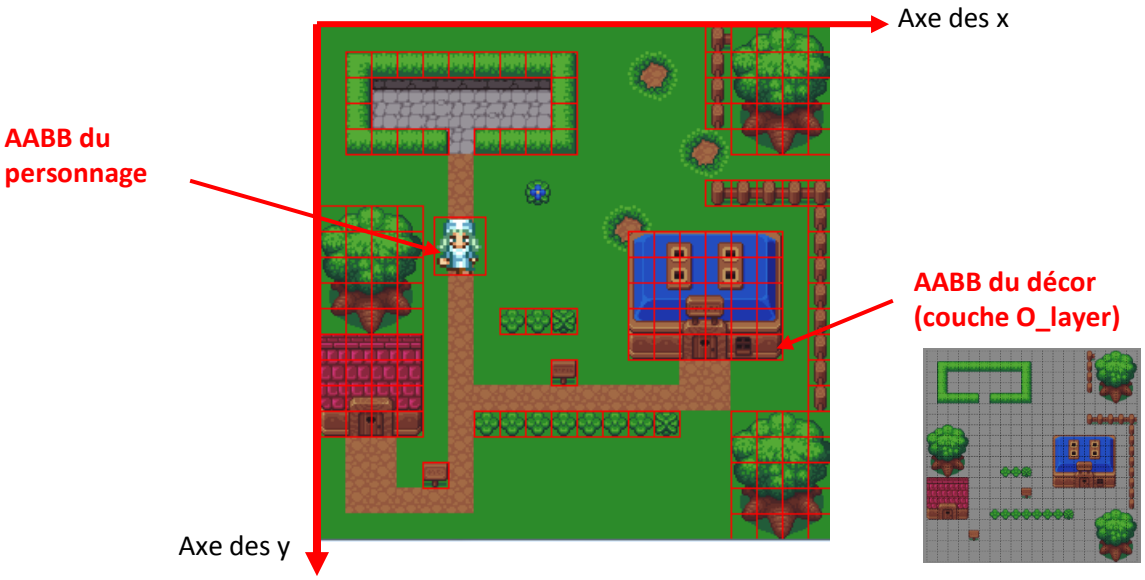
TP4 : Les collisions

Table des matières :

1. La théorie des collisions

Il existe de nombreux algorithmes pour gérer les collisions, nous allons étudier le plus simple, l'**AABB *Axis Aligned Bounding Box***. Comme son nom l'indique, il s'agit d'un rectangle dont les axes sont alignés avec le repère orthonormé de notre écran.



Une AABB est définie par ses coordonnées x,y (en haut à gauche), sa largeur W et sa hauteur H :

L'algorithme de test de collision entre deux AABB est le suivant :

```
Si (ABBB_decor.X==(AABB_perso.X+AABB_perso.W))     alors collision_droite = vrai
Si ((ABBB_decor.X+AABB_decor.X==AABB_perso.X)      alors collision_gauche = vrai
Si (ABBB_decor.Y==(AABB_perso.Y+AABB_perso.H))     alors collision_bas = vrai
Si ((ABBB_decor.Y+AABB_decor.H)==AABB_perso.Y)     alors collision_haut = vrai
```

2. Gestion des collisions

a) Identification des AABB

Travail n°1 :
A partir du programme Processing exemple1, codez les deux fonctions pour dessiner les AABB :

- la fonction ***void dessine_AABB_perso()*** qui affiche le rectangle délimitant l'AABB du personnage.
- la fonction ***void dessine_AABB_decor()*** qui affiche le rectangle délimitant les AABB du decor (la couche O_layer).

b) Recadrage de l'AABB

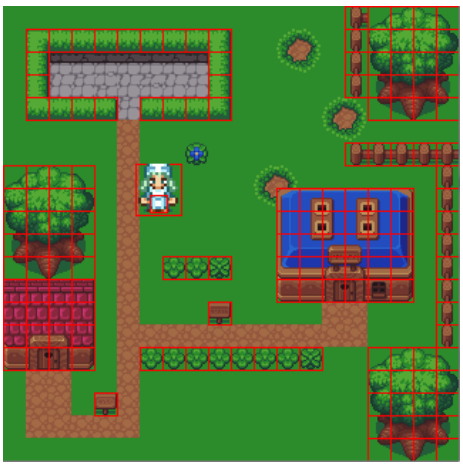
Pour permettre au personnage de passer au plus près des obstacles, il faut recadrer l'AABB du personnage :
La nouvelle AABB doit être légèrement inférieur à la taille d'une tuile de 16px. On prend donc une AABB de 12px/12px pour avoir une marge de 2px de chaque côté.

CoordX_AABB_perso1 = CoordX_perso1+((32-16)/2)+2
= CoordX_perso1+10

CoordY_AABB_perso1 = CoordY_perso1+(36-16)+2
= CoordY_perso1+22

Travail n°2 :
A partir du programme précédent, codez la fonction qui dessine la nouvelle AABB recadrée du personnage :

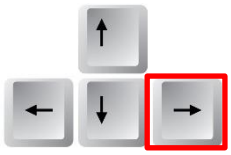
- la fonction ***void dessine_AABB_perso_tile()*** qui affiche le rectangle délimitant l'AABB du personnage.



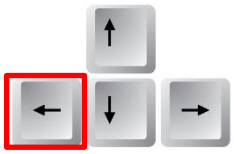
c) Gestion des collisions

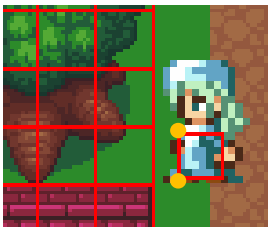
A présent, l'objectif est de localiser l'AABB du personnage par rapport à la grille de tuiles (20x20) qui composent la carte. Chaque tuile ayant un numéro allant de 0 à 400.
Il faut déterminer si la future position de l'AABB ne sera pas en collision avec un obstacle du décor (O_layer) :

⇨ Si on se déplace vers la droite, il faut tester le coin haut droit et le coin bas droit de l'AABB :



⇨ Si on se déplace vers la gauche, il faut tester le coin haut gauche et le coin bas gauche de l'AABB :

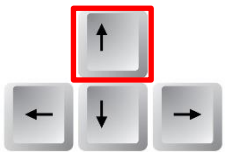


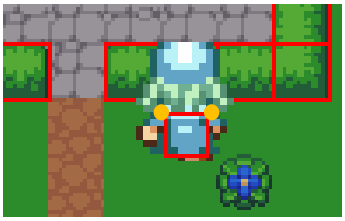


num_tuile_AABB_coin_HD

num_tuile_AABB_coin_BD

⇨ Si on se déplace vers le haut, il faut tester le coin bas gauche et le coin bas droit de l'AABB :

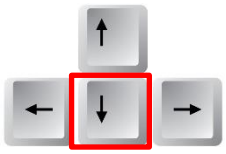


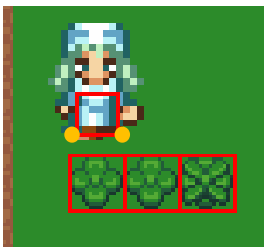


num_tuile_AABB_coin_HD

num_tuile_AABB_coin_HG

⇨ Si on se déplace vers le bas, il faut tester le coin haut gauche et le coin haut droit de l'AABB :





num_tuile_AABB_coin_BD

num_tuile_AABB_coin_BG

Travail n°3 :

A partir du travail précédent, réalisez le programme final :

Codez la fonction qui permet de localiser l'AABB du personnage en déterminant sur quelles tuiles les 4 coins de l'AABB se situent.

- Une fonction ***int localisation(int coordX, int coordY)*** qui renvoie le numéro de la tuile par rapport aux coordonnées du coin de l'AABB passées en paramètres.

Codez la fonction de test des collisions qui sera vérifiée à chaque déplacement sur les deux coins de l'AABB concernés (**exemple :** si on veut se déplacer vers la droite, on *test num_tuile_AABB_coin_HG+1* et on teste *num_tuile_AABB_coin_HG+1*) :

- Une fonction ***boolean test_collision(int num_tuile)*** qui renvoie « vrai » s'il y a collision.