1.1.4 Classe Descente

La **classe Descente** est un **thread** responsable de **simuler la gravité** dans le jeu. Son objectif principal est de faire en sorte que **Mario tombe naturellement** lorsqu'il n'est pas sur une surface solide.

Elle fonctionne en appliquant une **descente automatique** à Mario en fonction de la gravité du jeu, définie par la constante `CONSTANTS.GRAVITY`.

Si **Mario n'est pas sur une plateforme solide**, alors **il doit descendre** jusqu'à atteindre le sol.

Ce comportement est géré par **la boucle infinie du thread** qui :

1. Vérifie **toutes les 16 millisecondes** si Mario doit tomber.
2. Si Mario n'est **pas encore sur le sol**, sa position y est **augmentée progressivement** (ce qui le fait descendre).
3. Si Mario **atteint ou dépasse** le sol, il est **immobilisé** sur la surface.

Intégration avec d'autres classes

--- Interaction avec Collision

La **classe Collision** interagit avec **Descente** pour empêcher Mario de traverser les plateformes solides :

- **Si Mario atterrit sur une plateforme**, `allowedToFallDown` est mis à **false**, ce qui bloque la chute.
- **Si Mario quitte une plateforme**, `allowedToFallDown` est mis à **true**, ce qui réactive la gravité.

Extrait du code de **Collision** :

```
if (gp.tm.tiles[point1].collision || gp.tm.tiles[point2].collision)
{
    threadDescente.allowedToFallDown = false;  // Mario est sur une
plateforme
} else {
    threadDescente.allowedToFallDown = true;   // Mario peut tomber
}
```

Améliorations possibles

Ajouter une accélération de la gravité (comme dans les jeux classiques) pour que Mario ne descende pas à vitesse constante

1.1.4 CLASSE COLLISION

La classe **Collision** est le **thread** le plus important du jeu. Son rôle est de détecter, toutes les **16 millisecondes**, s'il y a eu collision entre Mario et un objet du jeu.

Les objets avec lesquels Mario est susceptible d'entrer en collision sont :

- **Les ennemis** (tortues, Goombas, etc.)
- **Les briques du jeu**, y compris :
    - Les grands tuyaux verts
    - Les plateformes de terre
    - Les blocs contenant des récompenses

Déterminer si Mario est entré en collision avec un ennemi est à la fois **simple et complexe**.
En effet, la classe **Mario** hérite de la classe `GameCharacter`, qui elle-même hérite de la classe `Rectangle`.

########## (montrer le diagramme de classes ici)

Grâce à `Rectangle`, un rectangle invisible est modélisé autour de Mario et des ennemis avec des dimensions prédéfinies.

Pour simplifier, nous avons défini : `x = 8`, `y = 8`, et `width = height = 8`. (Ces valeurs sont essentielles pour **simuler des collisions pseudo-réalistes**. Une explication plus détaillée est donnée ci-dessous.)

En utilisant la fonction `intersects()`, il est possible de détecter si ces deux rectangles se chevauchent, signalant ainsi une collision.

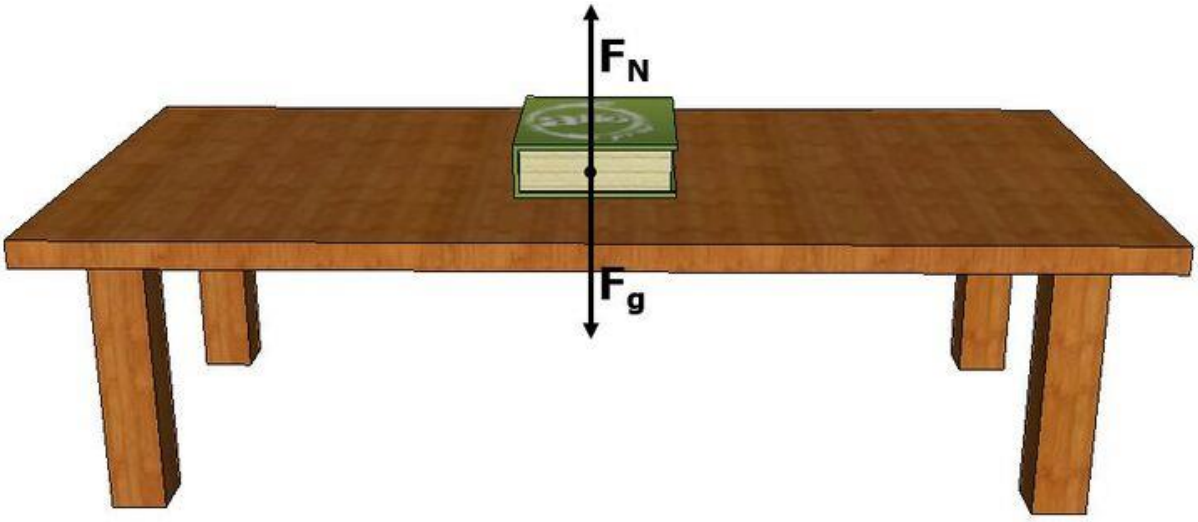
Toutefois, cette détection nécessite **un ajustement précis des coordonnées** des deux entités du jeu (Mario et l'ennemi potentiel).

->>> explication aprés avec mengtong .

Par ailleurs, cette classe vérifie en permanence les collisions avec la partie bas de ‘sa cellule‘ par ce que si jamais il saute et rebondit sur une tuile qui est “ solide “

On ne doit pas traverser la tuile, en physique ceci se traduit

Avec la force normale :



Dans notre cas , on ne va pas créer un thread pour **la force normale** mais plutot mettre à 0 la force de la gravité de la classe : Descente, plus précisément, on bloque le code du thread !

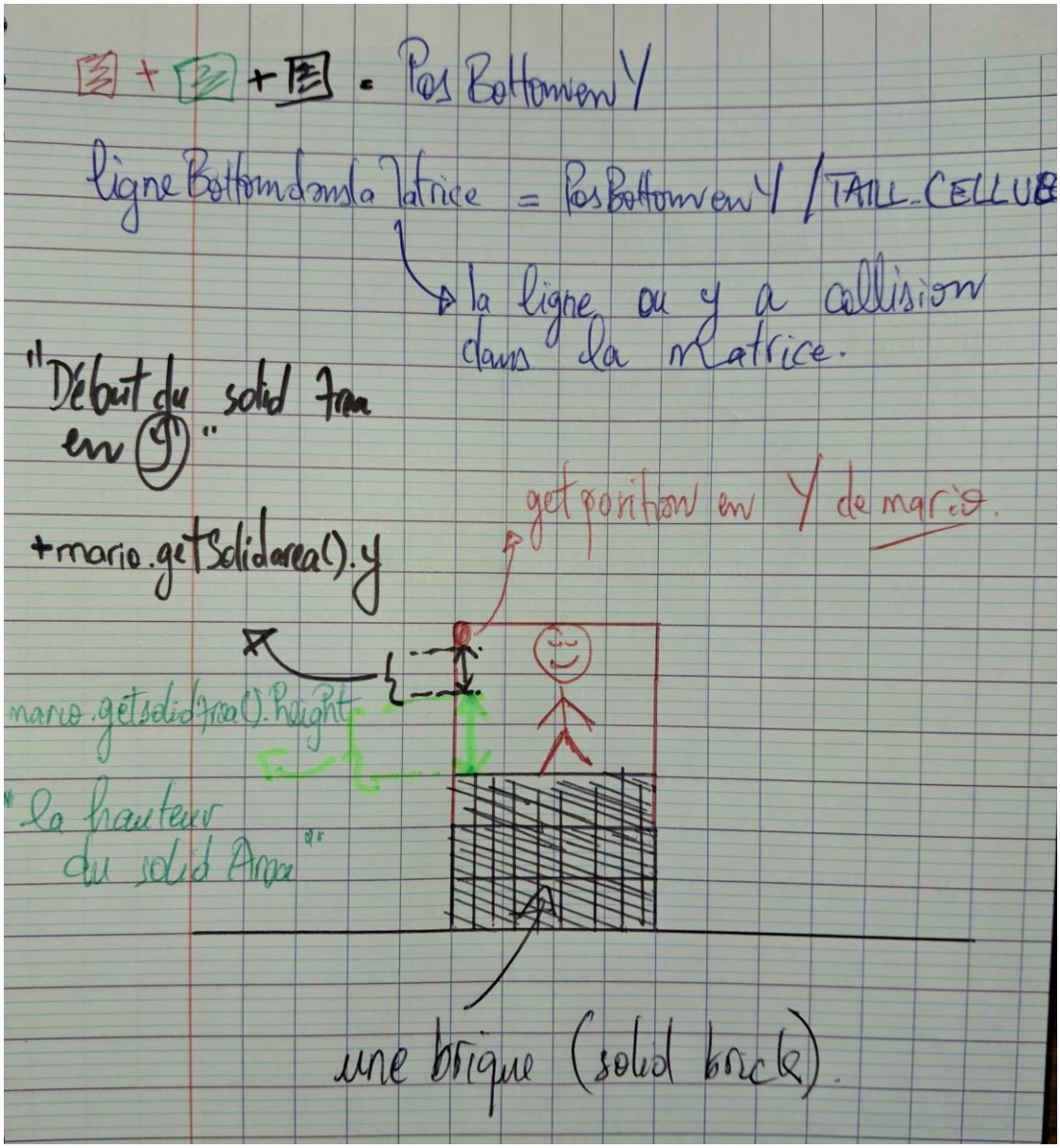
Voici le code :

```
if (gp.tm.tiles[point1].collision || gp.tm.tiles[point2].collision) {
                    System.out.println("point1 = " + point1 + " point2 = " +
point2);
                    // si c'est une brique de type 3 ( puit , riviere , ... )
alors mario meurt
                    if (point1 == 3 || point2 == 3) {
                        // mario meurt
                        threadDescente.setSol(CONSTANTS.LE_SOL * 2);
                        jumpingThread.notjumping();
                        mario.noMoving();

                        System.out.println("waaaaaaaaaaaa333333");

                    } else if (!sur_brick) {
                        mario.position.y = (ligneBottomdanslaMatrice) *
CONSTANTS.TAILLE_CELLULE;
                        sur_brick = true;
```

```
                        // bloquer la descente !
                        threadDescente.allowedToFallDown = false;
// comme mario est entrain de “ jump “ , on doit aussi arreter l’action de
saut .
                        jumpingThread.notjumping();
                    } else {
                        // System.out.println("je suis sur une brick”);

                    }
                } else {
    // comme on verifie ça en permanance , je verifie que mario est sorti de
la zone de collision et doit donc “ tomber par terre “
                    // // debloquer la descente
                    threadDescente.allowedToFallDown = true;
                    sur_brick = false;
                }
```

Le point 1 et point 2 sont les 2 points qui correspondent à une valeur dans ma matrice du jeu, obtenu avec les calculs montés dans l’image suivante:

j’explique le point 1 (idem pour le point 2) .



Donc avec ligneBottomdanslaMatrice , j'obtiens la ligne ou se trouve mario dans la matrice (remarquons que je n'ai jamais intéragit avec la vue ici, je n'utilise que des variable du modele)

Point 1 = matrice[ligneBottomdanslaMatrice][colonnegauchedanslamatrice]

La colonnegauchedanslamatrice on l'obtient de la maniere suivante :

```
int posLeftenX = mario.getPosition().x + mario.getSolidArea().x;
int colonneLeftdanslaMatrice = posLeftenX / CONSTANTS.TAILLE_CELLULE;
```

Donc le point 1, pourrait maintenant avoir la valeur 2,

Je vérifie si la tuile du type 2 est solide ou pas, si oui, je fais ce que j'ai dit avant.

La question qui se pose est donc la suivante pourquoi utiliser getsolidArea().x et getsolidArea().y ?

C'est pour permettre à Mario de rentrer en collision avec les tuiles pour de vrai (to have a real effect) , on doit voir Mario foncer dedans dans une brique, au lieu d'une collision comme ça :

 On préfère une collision comme ça 

Idem pour les collisions avec les ennemies.(surtout avec les ennemies)