**Génération Procédurale**

# Note : ICI PLATEFORME = Ensembles de blocs ET BLOC c’est un objet Plateforme

Pour les objets Plateformes le Xmax correspond à la largeur et Ymax correspond à la hauteur. Dsl Ahmed je modifierai ça aussi.

Table des matières

[Note : ICI PLATEFORME = Ensembles de blocs ET BLOC c’est un objet Plateforme 1](#_Toc37949610)

[Introduction : 1](#_Toc37949611)

[Définir les positions que le joueur peut atteindre : 2](#_Toc37949612)

[Importation des images et création des types de plateformes : 2](#_Toc37949613)

[Idée : 2](#_Toc37949614)

[Mise En Place : 3](#_Toc37949615)

[Création des objets plateformes : 4](#_Toc37949616)

[Idée : 4](#_Toc37949617)

[Mise en Place : 4](#_Toc37949618)

[Définir la position de chaque block : 4](#_Toc37949619)

[Idée 4](#_Toc37949620)

[Mise en Place 5](#_Toc37949621)

# Introduction :

Tout d’abord, le programme contient 5 classes permettant de placer aléatoirement des plateformes dans une fenêtre.

Map : Classe qui utilise toutes les autres classes pour réaliser la génération Procédurale. Le but de cette classe est de créer une liste de blocs tous placés aléatoirement. Cette liste permettra ensuite d’afficher chaque bloc au bon endroit.

CoordonneePoint : Classe qui permet de créer un objet possédant 2 attributs, x = abscisse et y = ordonnée. Cette classe permet de définir des points où l’on peut placer la plateforme.

Plateforme : Permet de créer des blocs possédants une position (xMin, yMin), une largeur et hauteur et une image.

Tableau aléatoire : classe permettant de créer un tableau rempli avec des chiffres aléatoire.

LectureFichierTexte : Classe qui permet de lire un fichier texte et de remplir un tableau avec les informations lues. Il existe un fichier texte par type de plateforme.

L’idée est d’assembler plusieurs blocs formants ainsi une plateforme. Ces différentes plateformes seront ensuite placées de manière aléatoire à des endroits accessibles par le joueur.

# Définir les positions que le joueur peut atteindre :

Pour cela, j’ai pensé à deux possibilités.

**La première :**

A partir de l’équation de mouvement du personnage, récupérer les positions qu’il peut atteindre. Je ne crois pas que cette méthode soit possible car on n’a pas d’équation de mouvement.

**Deuxième :**

Aller dans le jeu, puis faire sauter le personnage tout en notant dans un fichier texte la position du joueur à intervalle régulier. Refaire la manip avec des sauts différents. Enfin lire ce fichier texte et remplir un tableau avec. Je n’ai pas encore créé ce fichier texte. La méthode qui devra s’occuper de ça s’appelle calculPositionPlateforme dans la classe Map. On a 3 tableaux correspondant à 3 types de sauts. Ces tableaux sont de dimensions 2 mais cela ne sert à rien, je vais les changer en dimension 1. Ils possèdent n objets de type CoordonnéePoint. J’ai initialisé ces objets CoordonnéePoint avec différentes positions (x,y) mais plus tard, celles-ci seront initialisées avec le document texte.

Ces 3 tableaux sont ensuite réunis dans un autre tableau ‘tabPositionPossible’ pour pouvoir l’utiliser facilement. De même, je changerai ce tableau en dimension 2.

# Importation des images et création des types de plateformes :

## Idée :

On a un fichier texte pour chaque type de plateforme. Ils contiennent des caractères (A, B…) correspondant aux noms des images. La position de ces caractères dans le fichier (ligne 2, colonne 3) correspond aussi à la position des blocs dans la plateforme.

On charge donc les bonnes images qui sont ensuite placées dans une liste à la bonne place.

De plus, on définie le nombre de ligne et de colonne de chaque type de plateforme ainsi que la taille des blocs qui composent la plateforme. Pour l’instant dans une même plateforme tous les blocs ont la même largeur et hauteur, je changerai ça.

Pour créer un nouveau type de plateforme, il faut juste crée un fichier texte et le remplir avec les caractères correspondant aux images voulus et définir dans le programme le nombre de ligne, colonne et la taille des blocs.

## Mise En Place :

Les méthodes qui s’occupent de ça sont creationListImage, AffecterDimensionPlateforme et importerImages ainsi que la classe LectureFichierTexte.

Tout d’abord, dans créationListImage j’utilise un switch correspondant à chaque type de plateforme créée. Pour l’instant il y en a 3. De plus je pense que le switch n’est pas obligatoire, j’essaierai de le supprimer.

Pour chaque type de plateforme je définie le nombre de colonne « tabInfoPlateforme[i][0] » et le nombre de lignes « tabInfoPlateforme[i][1] » puis la largeur et hauteur des blocs qui constituent la plateforme. Ça aussi, je le changerai pour pouvoir utiliser des blocs de différentes tailles dans une même plateforme.

Ensuite dans affecterDimensionPlateforme, on ajoute dans listDimensionAllPlateforme la hauteur et la largeur de chaque blocs constituants chaque type de plateforme. Cette méthode est pour l’instant inutile puisque tous les blocs de chaque plateforme possèdent les mêmes dimensions. Mais je transformerai tout ça pour pouvoir utiliser des blocs de différentes tailles.

Enfin, on importe les images.

Pour cela on utilise la classe LectureFichierTexte qui va lire un fichier correspondant au type de plateforme d’indice i. Ce fichier texte est constitué de caractères correspondant aux noms des images à importer. Ces caractères sont insérés dans un tableau qui peut être appelé à partir des objets tabTypePlateforme[i]. « Le caractère ‘.’ Correspond une image null. »

On utilise ensuite la méthode importerImages.

Pour chaque type de plateforme d’indice i, on crée une listImagePlateforme qui contiendra dans l’ordre les images de chaque blocs (ligne 1, 2 ,3 … puis colonne 1 ,2, 3…).

Le chemin pour accéder à l’image voulu est défini avec tabTypePlateforme[i].tabPlateforme[H][L] correspondant aux caractères du fichier texte. C’est pourquoi les noms des images sont « A, B, C, D… ».

On ajoute ces listes dans listAllImagePlateforme qui contient les images de tous les types de plateformes.

# Création des objets plateformes :

## Idée :

On va créer un tableau aléatoire avec la classe TableauAleatoire. Celui-ci permettra d’attribuer à chaque plateforme un type de plateforme aléatoire.

Ensuite pour chaque plateforme on va créer des blocs (blocs = objets plateformes) avec une largeur, hauteur et image en fonction du type de plateforme sélectionné aléatoirement. Chaque type de plateforme a déjà été défini plus haut.

Ici, on ne s’occupe pas de la position des plateformes (xMin = 0, yMin = 0).

## Mise en Place :

On utilise 3 boucles for pour parcourir les 3 listes.

L’indice i corresponds à chaque plateforme.

Pour chaque plateforme on a donc un typeDePlateforme, un nombre de ligne et de colonne. Ces plateformes sont placées dans listOPlateforme qui contient toutes les plateformes du jeu.

On crée ensuite une liste pour chaque j qui correspond à chaque colonne de la plateforme.

L’indice m correspond à chaque bloc de chaque colonne j. Pour chaque bloc, on a une largeur, hauteur et image avec lesquelles on crée des objets Plateformes.

# Définir la position de chaque block :

## Idée

Le premier bloc du jeu est placé manuellement.

Ensuite, on place les blocs colonne par colonne.

Une fois tous les blocs de la plateforme placées, on passe à la deuxième.

Le premier bloc de la deuxième plateforme est placé à une position aléatoire qui peut être atteinte par le joueur. Puis, on place le reste des blocs colonnes par colonnes.

Ainsi de suite.

## Mise en Place

On parcourt les 3 listes avec 3 boucles for.

L’indice i correspond à chaque plateforme. Le J, à chaque colonne de la plateforme i. Et le m, à chaque bloc de la colonne j.

A partir de la deuxième plateforme (i ! = 0), on place le 1er bloc aléatoirement à l’aide du tabPositionPossible[0][objetPatternPositionPlateforme.tabPattern[i]][0].x

tabPositionPossible[0][objetPatternPositionPlateforme.tabPattern[i]][0].y

On les place plus ou moins loin en fonction du nombre de colonne de la plateforme précédente (listOPlateforme.get(i-1).size() <= 2).

On place ensuite le premier bloc de la deuxième colonne (car le premier bloc de la 1ère colonne est déjà placé) avec j != 0.

Pour placer ces blocs colonnes, on ajoute la largeur du bloc d’à côté.

**if**(j != 0) {

listOPlateforme.get(i).get(j).get(0).xMin = listOPlateforme.get(i).get(j-1).get(0).xMin + listOPlateforme.get(i).get(j-1).get(0).xMax;

listOPlateforme.get(i).get(j).get(0).yMin = listOPlateforme.get(i).get(j-1).get(0).yMin;

}

Enfin, on place le deuxième bloc de la j colonne puis le 3éme bloc de la j colonne…. Avec l’indice m. Ces blocs sont placés en ajoutant la hauteur du bloc d’au-dessus.

**for**(**int** m = 1;m < listOPlateforme.get(i).get(j).size();m++) {

listOPlateforme.get(i).get(j).get(m).xMin = listOPlateforme.get(i).get(j).get(m-1).xMin;

listOPlateforme.get(i).get(j).get(m).yMin = listOPlateforme.get(i).get(j).get(m-1).yMin+ listOPlateforme.get(i).get(j).get(m-1).yMax;

}

Tous les blocs sont placés au bon endroit. On peut donc maintenant créer un objet Map qui possède l’attribut listOPlateforme pour l’affichage.

J’arrangerai le code pour créer aussi une liste de plateforme de type G.