

Dec 18, 17 14:46

API.java

Page 1/2

```

/**
 * Pour représenter le contenu d'une cellule
 * dans l'ordre: terre calcaire, terre siliceuse, roche et eau
 */
enum Ground {CHALK, CLAY, STONE, WATER};

/**
 * Pour représenter des zones géographiques simulables
 */
interface Simulable {

    public abstract boolean begin(Window window, FileSystem fileSystem);

    public abstract void update(float deltaTime);

}

/**
 * Terrains dont on veut simuler l'évolution
 */
class Terrain implements Simulable {
    Window window;
    FileSystem fileSystem;

    // terrain rectangulaire composé de cellules
    private Cell[][] cells;
    // nom du fichier indiquant le contenu initial de chaque cellule
    private String initialMap;

    // initialise cells à partir du contenu du fichier de nom initialMap
    private void readFromFile() {
        //...
    }

    // initialise le nom du fichier de configuration
    public Terrain(String initialMap) {
        //...
    }

    // initialise le tableau cells au moyen de readFromFile
    @Override
    public boolean begin(Window window, FileSystem fileSystem) {

        //...
        return true;
    }

    // appelle la méthode de dessin de chaque cellule
    public void draw() {
        //...
    }

    // appelle update sur chaque cellule
    @Override
    public void update(float deltaTime) {
        //...
    }

    //choisit une cellule dans un rayon autour d'une position
    // si la cellule est libre y plante l'arbre
    // sinon ne fait rien
    public void plantInRadius(Tree tree, Vector position, double radius) {
        //...
    }

```

Dec 18, 17 14:46

API.java

Page 2/2

```

    }

    // choisit une cellule au hasard
    // si la cellule est libre y plante l'arbre
    // sinon ne fait rien
    public void plantInRandomCell(Tree tree) {
        //...
    }
}

/**
 * Programme principal
 */

public class Program {

    public static final float MAX_DELTA_TIME = 0.1f;

    public static void main(String[] args) {

        // Define file system
        FileSystem fileSystem = new FolderFileSystem(new ResourceFileSystem(DefaultFileSystem.INSTANCE));

        // Initialise la fenêtre de simulation
        Window window = new SwingWindow("Play", fileSystem);
        try {

            // Create une zone géographique à simuler
            Simulable land = new Terrain("toundra.map");
            if (land.begin(window, fileSystem)) {

                // Utilise l'horloge système pour suivre l'évolution du temps
                long before;
                long now = System.nanoTime();

                // S'exécute jusqu'à ce que l'utilisateur essaie de fermer la f
                while (!window.isCloseRequested()) {

                    // calcule l'intervalle de temps
                    //...
                    //Laisse la simulation s'exécuter
                    land.update(deltaTime);

                    // met à jour les affichages
                    window.update();

                }
            }
        } finally {

            // libère les ressources
            window.dispose();

        }
    }
}

```