# Affichage des bordures dans le World

Modifiez le constructeur de **SnakeWorld** :

public SnakeWorld()

{

super(25, 20, 32);

addObject(new Border(), 0, 0);

}

Deuxième version du  constructeur de **SnakeWorld** :

public SnakeWorld()

{

super(25, 20, 32);

for (int x = 0; x < getWidth(); x ++ ) {

addObject(new Border(), x, 0);

addObject(new Border(), x, getHeight()-1);

}

for (int y = 1; y < getHeight()-1; y ++) {

addObject(new Border(), 0, y);

addObject(new Border(), getWidth()-1, y);

}

}

# Affichage de la tête de Bobby

A ajouter dans les **propriétés** de la classe **SnakeWorld**:

private LinkedList<SnakeBody> snake = new LinkedList<SnakeBody>();

A ajouter tout en haut de la classe **SnakeWorld**:

import java.util.\*;

A ajouter dans le **constructeur** de **SnakeWorld**:

SnakeBody body = new SnakeBody();

snake.add(body);

addObject(body, 2, 2);

# Déplacement de Bobby

A ajouter dans les **propriétés** de la classe **SnakeWorld**:

private int dx = 1;

private int dy = 0;

A ajouter dans la classe **SnakeWorld**:

public void act()

{

//on remplace l'image de la tête

SnakeBody head = snake.getLast();

head.setImage("tail.png");

//crée une nouvelle tête

SnakeBody newHead = new SnakeBody();

int newHeadX = head.getX()+dx;

int newHeadY = head.getY()+dy;

//ajoute la nouvelle tête à la liste et au world

addObject(newHead, newHeadX, newHeadY);

snake.add(newHead);

}

# Limiter la taille de la queue

A ajouter dans les **propriétés** de la classe **SnakeWorld**:

private int tailCounter = 5;

A ajouter dans la méthode **act()** de **SnakeWorld**:

if (tailCounter == 0) {

SnakeBody tail = snake.removeFirst();

removeObject(tail);

} else {

tailCounter--;

}

# Changement de direction

A ajouter dans la classe SnakeWorld :

private void changeDirection() {

if (Greenfoot.isKeyDown("left") && dx == 0 ) {

dx = -1;

dy = 0;

} else if (Greenfoot.isKeyDown("right") && dx == 0 ) {

dx = 1;

dy = 0;

} else if (Greenfoot.isKeyDown("down") && dy == 0 ) {

dx = 0;

dy = 1;

} else if (Greenfoot.isKeyDown("up") && dy == 0 ) {

dx = 0;

dy = -1;

}

}

A ajouter dans la méthode **act()** de **SnakeWorld** (au début de la méthode act() ):

changeDirection();

# Gérer les collisions

A ajouter dans les **propriétés** de la classe **SnakeWorld**:

private boolean dead = false;

A ajouter dans la méthode **act()** de **SnakeWorld**:

if (dead) {

return;

}

A ajouter dans la classe **SnakeWorld**:

public void dead() {

dead = true;

}

A ajouter dans la méthode **act()** de **SnakeWorld**(*attention à bien positionner ce code*) :

List<Block> blocks = getObjectsAt(newHeadX, newHeadY, Block.class);

for(Block block : blocks) {

block.collision(this);

}

A ajouter dans la classe **Block** :

public void collision(SnakeWorld world) {

world.dead();

}

# Ajouter une pomme

A ajouter dans le **constructeur** de **SnakeWorld**:

Apple apple = new Apple();

addObject(apple,

Greenfoot.getRandomNumber(getWidth()-2)+1,

Greenfoot.getRandomNumber(getHeight()-2)+1);

# Collision avec une pomme

A ajouter dans la classe **Apple** :

public void collision(SnakeWorld world) {

world.grow(2);

setLocation(

Greenfoot.getRandomNumber(getWorld().getWidth()-2)+1,

Greenfoot.getRandomNumber(getWorld().getHeight()-2)+1);

}

A ajouter dans la classe **SnakeWorld**:

public void grow(int i) {

tailCounter = tailCounter + i;

}

# Ajouter du son

A ajouter dans la méthode **collision** de la classe **Apple** :

Greenfoot.playSound("slurp.mp3");

A ajouter dans la méthode **collision** de la classe **Block** :

Greenfoot.playSound("dead.mp3");