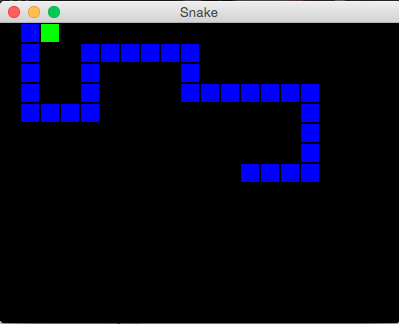
Snake

by David Melisso and Jin Kim

David: <http://djmcoder.com/>

Jin: <http://facebook.com/jin.kim338658>



Make the Game

If you haven’t already, make sure to download the Tetris/GridMonster skeleton off of Athena.

First, let’s think about how the game works. We have a game that contains a snake, and that snake moves around and picks up items.

Let’s make two classes for these two items: SnakeGame and Snake.

SnakeGame

First, SnakeGame. (Reminder: Make sure to put in a version and author!)

Make SnakeGame an ArrowListener; remember, we want the snake to move around. We also want SnakeGame to have some instance variables, such as the grid, the display, and whether the game is running.

Since SnakeGame is an ArrowListener, we need to put some required methods in the class. We will decide what is put in these later, but for now make the required methods from ArrowListener in the SnakeGame class.

We also need a play method. This method should have a while loop that stops when the game ends. Like in the Tetris play method, it should sleep for a certain amount of time and move after that amount of time, like this:

try

{

Thread.sleep(200);

//statements here

}

catch (InterruptedException e)

{

//ignore

}

Finally, create a constructor that sets all the instance variables to their proper state. (Refer to the Tetris class if you have difficulty with this).

Make the Snake

Now, we have to make the Snake class. Feel free to make supplementary methods as needed.

The Snake class should have three instance variables: The locations of its blocks, the grid that it is placed in, and the direction that it is going.

First, make a constructor with the grid as a parameter. Create/set instance variables to their proper state. Then make a for loop, where blocks are created and centered in the grid. Make sure to add them to the list of locations.

Let’s make some accessors for our Snake. Then, make a method to check if a Location is empty:

private boolean isEmpty(MyBoundedGrid<Block> eGrid, Location loc)

On the Subject of Directions

How will we store directions? There are numerous ways to do it, notably using a Direction class or storing the direction as a String and making the methods in the Snake class. You can decide which one you want to do, but make sure it has the following capability:

* Can determine the whether a direction is a valid direction (up, down, left, right).
* Can determine the opposite direction
* Snake can change its direction, but only if it is to the right or left of the original direction (changeDirection())

Translating

One more thing with Snake: Translating. When we translate the Snake, we don’t aren’t moving every block, but rather just the head block and everything follows. Or, you could insert a block at the start and remove a block from the end. Your choice.

The translate method should take in either a delta row and a delta column. In practice, the inputs will either be (±1, 0) or (0, ±1).

Remember to make translate() return a boolean (whether it was successful or not. Some examples of it being unsuccessful is if the block that it tries to translate to a Location that is already filled, or invalid.

Translating based off of Direction

What is the point of the translate() method if we don’t use it? We should make a method that, based off of the direction, translates the snake. This method should not take in any inputs and returns whether or not the method was successful. Let’s call this method translateOffDirection().

Finish play()

Make the play() method so that it in the //statements here

comment is replaced with this code:

if(snake.translateOffDirection())

{

display.showBlocks();

}

else

{

//snake dies here.

gameOver(false);

}