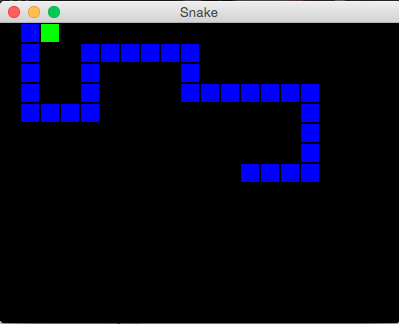
Snake

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.

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

Translating

One more thing with Snake