Unit 3 – 2D Arrays

Media Computation Project

Part 1 – Create a Greenfoot Project

1. Create a new Scenario in Greenfoot.
2. Rename your World Class "`Canvas`"

Part 2 – Finding a Picture

1. Do a Google Image Search for a picture of your choice of a decent size. If you add 1024 x 768 to your search, you can find some standard picture sizes that might be good.

   Example: snowman 1024 768.

   A lot of websites will be blocked, but find one you can load the picture from, then right click the picture and save it in the images folder for your Scenario. Make sure not to save the image directly from Google's thumbnails – it won't be the right size.

Part 3 – Getting Started

1. Create two field variables:
   a. A `static final String` that holds the filename for the image you selected.
   b. A `static final GreenfootImage` that builds a GreenfootImage for the selected image file.
2. Write a method called `reset` that sets the background image to the original file.
3. Write a constructor that uses the parent's constructor, setting the width and height to that of the selected image and calls the reset method.
4. Write a method called `getPixels()` that converts any `GreenfootImage` to a 2D Array. This can be accomplished by:
   a. creating a 2D Array that has *width* x *height* elements,
   b. looping through this 2D Array,
   c. and storing the image at each (x, y) pixel in the corresponding slot in the 2D Array.
5. Overload the `getPixels()` method so that it defaults to creating this array for the background image.
6. Write a method called `drawPixels()` that takes a 2D Array and converts it to an image. This can be accomplished by:
   a. creating an image that has a width equivalent to the number of columns in the 2D array and a height equivalent to the number of rows in the 2D array,
   b. looping through the 2D Array,
   c. setting the `Color` at (column, row) in the image to the `Color` stored in the 2D array.

Part 4 – Picture Modification Methods

Write the following methods:

7. `randomizeOneColor()`
    a. **Create a random `Color` and set the entire screen to that `Color`.**
8. `static()`
    a. **Set every pixel on the screen to a different random `Color`.**
9. `grayScale()`
    a. **Convert each pixel to its gray equivalent.**
10. `invert()`
    a. **Change every pixel to its opposite `Color` value.**

        For example (0, 0, 0) -> (255, 255, 255) and (3, 200, 50) -> (252, 55, 205)
11. `mirrorTopToBottom()`
    a. **Reflect the top half of the pixels to the bottom half of the picture.**
12. `mirrorBottomToTop()`
    a. **Reflect the bottom half of the pixels to the top half of the picture.**
13. `mirrorLeftToRight()`
    a. **Reflect the left half of the pixels to the right half of the picture.**
14. `mirrorRightToLeft()`
    a. **Reflect the right half of the pixels to the left half of the picture.**
15. `flipHorizontal()`
    a. **Swap each pixel with its horizontal opposite.**
16. `flipVertical()`
    a. **Swap each pixel with its vertical opposite.**
17. `detectEdges(int threshold)`
    a. **Compare two pixels at (r, c) and (r, c + 1) by using the distance formula on the `Color` values themselves. If the distance is greater than the threshold, then make the pixel turn white, otherwise turn it black.**