cisc3665, fall 2011 / prof sklar
Assignment I: Introduction to Game Design and the Processing Environment

Instructions

This is the assignment for unit I, Introduction to Game Design and the Processing Environment.
It is worth 7 points (i.e., 7% of your term grade).

The assignment is due on **September 25** (electronic submission). *Instructions for electronic submissions will be posted on the class web page.*

The assignment has three parts:

- Part 1 is a written component, based on the reading distributed on the first day of class. This part must be submitted in a **plain text** or **PDF** document.
- Parts 2 and 3 are programming parts. These must be completed using the Processing environment (see labI.1). These parts must be submitted as **zip** files containing the Processing folder in which you created your sketch.

1 Written component *(2 points)*

Read the chapter entitled **Introduction to Game Design** (by Kevin Oxland), which was handed out in the first class. Answer the following questions. Submit your answers in **plain text** or **PDF** format.

1.1 Oxland defines games as having *rules* and *boundaries*. He discusses these in the context of draughts (checkers) and Tetris. Give an example of a computer game that you like to play and explain the rules and boundaries of that game. Note that some computer games have quite complex sets of rules—don't pick one of those games to describe here. Select something that you can explain easily, something that you could imagine writing yourself.
Also, don't just copy-and-paste rules from a game web site. Explain the rules *in your own words*.
(1 point this part)

1.2 Oxland also discusses *feedback* and the user's *interface*. For the same computer game you described above, explain how the user interface works (e.g., what controls the user has) and describe how the game provides feedback based on what the user does.
(1 point this part)

2 Programming component: Images and Pixels *(2 points)*

2.1 Go through the Processing tutorial on **Images and Pixels**:
`http://www.processing.org/learning/pixels/`

2.2 Visit the NSKY web site: `http://nskyc.com/`

2.3 Create a program in Processing that loads an image and computes the average color over all the pixels in the image. Create a second image where all the pixels are set to the average color (like on NSKY!). Have the program display the original image first, and then when the user presses any key or clicks the mouse,

show the averaged image. Have the program toggle between the original image and the averaged image. *(1 point this part)*

2.4 Modify your program so that it can behave differently based on different inputs by the user. For example, if the user presses **A** or **a**, show the averaged image; if the user presses **B** or **b**, show the original image tinted blue; if the user presses **G** or **g**, show the original image tinted green; if the user presses **R** or **r**, show the original image tinted red; if the user presses **Y** or **y**, show the original image tinted yellow; and if the user presses **O** or **o**, show the original image again.
HINT: use the `tint()` function.
(1 point this part)

3 Programming component: Drawing Curves *(3 points)*

3.1 Go through the Processing tutorial on **Drawing Curves**:
`http://www.processing.org/learning/curves/`

3.2 Create a program in Processing that randomly selects coordinates for 4 points within the size of the Processing display window. Draw a filled circle around each point with a radius of at least 5 pixels. Each circle should be a different color. Draw a **spline curve** between two of the points.
HINT: use the `size()` function, the `width` and `height` variables, and the `random()` function.
(1 point this part)

3.3 Modify your program in two ways: (1) when the mouse hovers over any point, it changes color (e.g., turns yellow); and (2) when the user presses any key, the program toggles between drawing a black curved line on a white background and drawing a white curved line on a black background.
(1 point this part)

3.4 Modify your program so that the black-curve-on-white draws a spline curve, and the white-curve-on-black draws a **Bézier curve**. Make sure that the endpoints of both curves are the same. For example, if the first point you draw is green and the second point is red, then those should be the starting and ending points of both the spline curve and the Bézier curve.
(1 point this part)