Programming Project 1

Guidelines

A programming project is a laboratory that you complete on your own, without the help of others. It is a form of take-home exam. You may consult your text, your notes, your lab work, our on-line examples, and the web pages associated with the course web page, but no other sources for code are permitted. You are encouraged to reuse the code from your labs or our class examples, however. While you may discuss these problems with the course instructors, you may not discuss them with the TAs, tutors, classmates, friends, etc. The use of any outside help or sources is a violation of the Honor Code.

This project is due by 9am on Monday, October 17. No late work will be accepted.

The projects will be graded out of 100 points, with roughly half the points given for correctness and half the points given for style. A more detailed breakdown appears at the end of this document. Thus, even imperfect programs can receive close to full credit, but poor design and style can lead to quite low final scores.

Program 1: Seeing Red

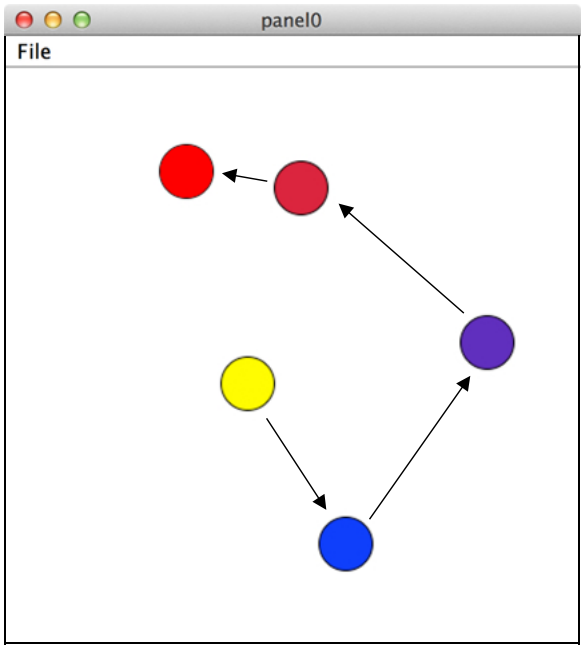
Your first program is a "hide-and-seek" game, AngryBird style. There is a bird looking for a pig. But the pig is hiding. The player's goal is to guide the bird to the pig. Fortunately, that job is made a little easier because the bird changes color from blue to red as it gets closer and closer to the pig. The player wins when the bird is placed on top of the pig's hiding place. See the handouts page for a demonstration of this game.

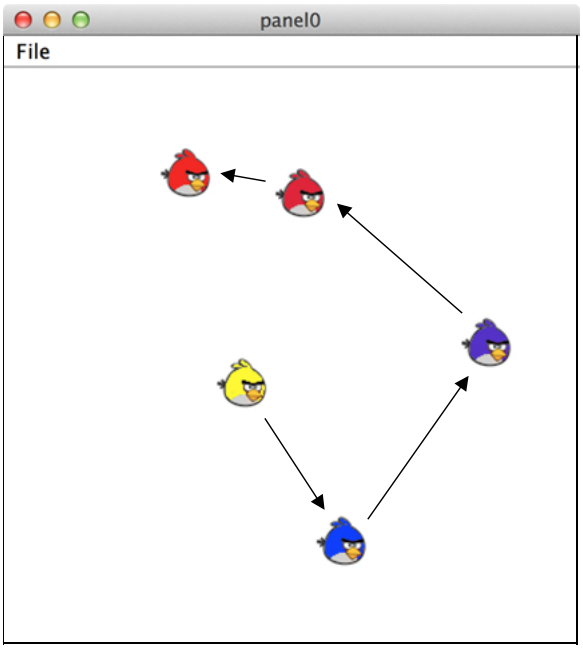
In your program, the hidden pig is represented as a point on the program's window picked at random by the computer. The program won't draw anything on the screen at that point. That would make it too easy to find. Instead, the program's window will be completely blank except...

The program will also draw a small bird in search of the hidden point. The bird is originally yellow in color. The player will drag this bird around the window using the mouse (as in the laundry program, the basketball programs, etc.). As the bird is dragged, the program will vary the color of this bird to give the player clues about where the hidden point is. When the circle is close to the hidden point it should be bright red. When it is far away, the bird should be blue. In between, the red should fade from red through various shades of purple until it finally becomes blue.

Each time the player releases the mouse, the program should check to see if the hidden point is within a short distance of the mouse, say 10 or 15 pixels. If it is, the program should turn the bird yellow again and display a pig centered in the canvas to indicate the player's success. Be sure the pig is *behind* the bird so it doesn't hide it. The program should also then immediately pick a new random point so the player can try again. The pig should disappear once the user presses on the bird again.

We provide bird and pig images for you to use. To get started, however, *ignore the bird image*. Instead, simply use a `FilledOval` to represent the bird, and add a `FramedOval` around it to create a border, just as you did for the swatch in the laundry lab. We'll see how to transform this border into the "colorful" bird below... *The oval should be exactly 35 pixels in diameter*. The image on the left below illustrates what happens as the user drags the bird around the screen and onto the hidden point for the version with only a filled oval and border. The image on the right shows the same sequence of events with the bird.





There are two hints you may find useful. First, there is an `objectdraw` method that you will want to use for this program. It is named `distanceTo` and it is an accessor method associated with `Location`s. If the names `point1` and `point2` are associated with `Location`s, then

```
point1.distanceTo(point2)
```

will compute and return the distance (measured in pixels) between the two points.

You will want to use the distance between your hidden point and the mouse location to determine the color of the oval displayed on the screen. Unfortunately, when you make a new `Color`, the numbers given for the amount of red, green and blue must be integers and `distanceTo` returns a `double`. Our second hint is how to turn a `double` into an `int`. It's simple. You just type "`(int)`" in front of any expression that describes a `double`. Java will drop the fractional part of the `double` to produce an `int`. For example,

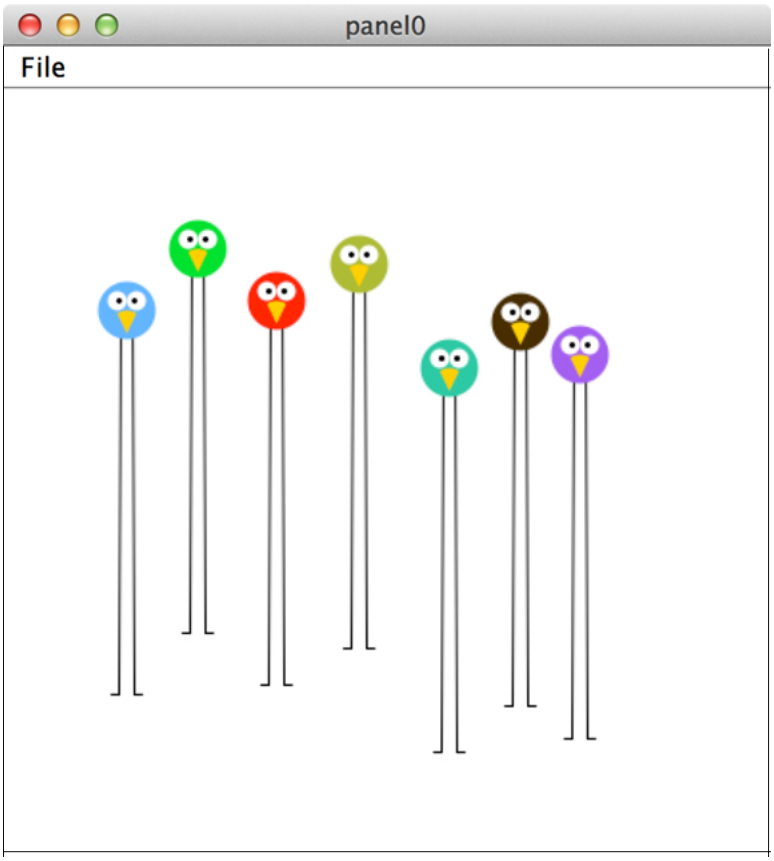
```
(int) point1.distanceTo(point2)
```

will produce an integer approximating the distance between two points. Once you have computed this distance, you can use it to create a linear progression from red to blue: if the distance is 0 you can create a pure red color; if the distance is 25, you can create a color with a red component of 230 and blue component of 25; if the distance is 50, you can create a color with a red component of 205 and blue component of 50; and so on. Any distance greater than or equal to 255 can be represented as pure blue.

Give it the Bird. Once you have the program working as described above, it's time to add the bird. This requires only one small change: replace the `FramedOval` border around your circle with a `VisibleImage` created from the "TransparentBird.png" image. We created this bird image using one special feature of most image encodings (such as JPEG, GIF, PNG). Namely, we made the pixels for the bird's body be transparent so that those pixels take on the color of whatever is layered below it on the canvas. Thus if you put the bird image over your filled oval, your bird will change colors just as you would like!

Program 2: Irate Ibises

Some birds grow long wings. Others grow long beaks. And a few grow long legs. Ibises are one such long-legged bird. This program illustrates that particular characteristic of ibises, objectdraw-fashion:



The canvas initially starts empty. When you click on the canvas, an ibis starts growing at that point. Initially it will be just a short bird with a body, beak, and legs. The bird's body may be of a color of your choosing. As you drag the mouse around, it grows. When it reaches its full height, the bird stops growing and eyes appear. The ibis won't grown any more, but if you click on the eyes after the bird has reached full size, its body should change to have a random color.

When you are happy with the ibis's color, you can grow another one by clicking somewhere else in the window. When the mouse moves out of and then back into the window, the scene resets itself to be empty.

Program Design. Your program should be divided into two classes: a window controller called `IrateIbises` and an `Ibis` class. We have actually provided a complete `IrateIbises` class in the starter folder. You should not modify our `IrateIbises` class in any way. Instead, you have to implement the `Ibis` class so that it works with our `IrateIbises`. In particular, the `Ibis` constructor should expect three parameters: the Location of the center of the bird's body, a double specifying the maximum height of the `Ibis`, and the canvas. The `Ibis` class should define the following methods used by the controller to implement the functionality described above:

- `public void changeColor()`: sets the color of the ibis body to a random color.
- `public boolean eyesContain(Location point)`: returns true if either of the eyes contains the point.
- `public void grow()`: makes the ibis grow a bit, or makes the eyes appear when it reaches its full height.

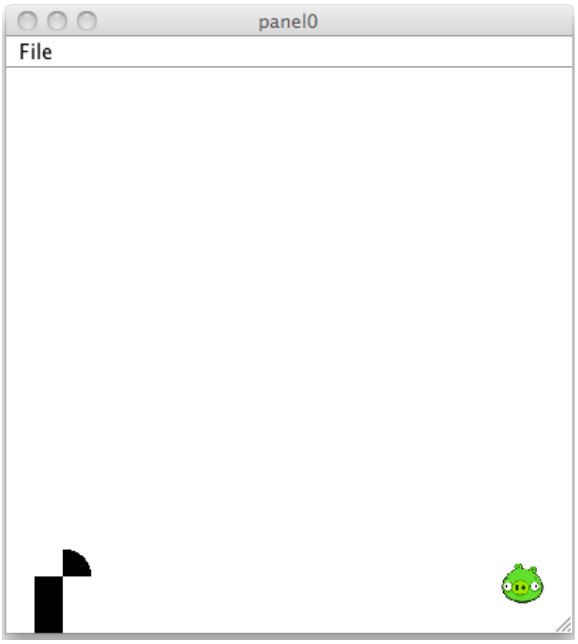
The ibis will only grow when the mouse moves, so `Ibis` should not extend `ActiveObject`.

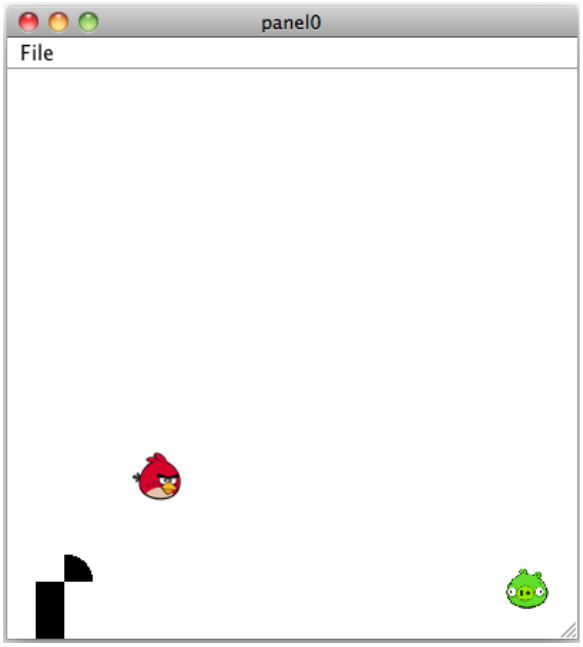
Ham It Up. Your bird need not look identical to ours — you may make the size, shape, and beak/eyes look however you like, and also feel free to add other graphic items to the `Ibis` class to make the birds more attractive.

Program 3: Angry Birds

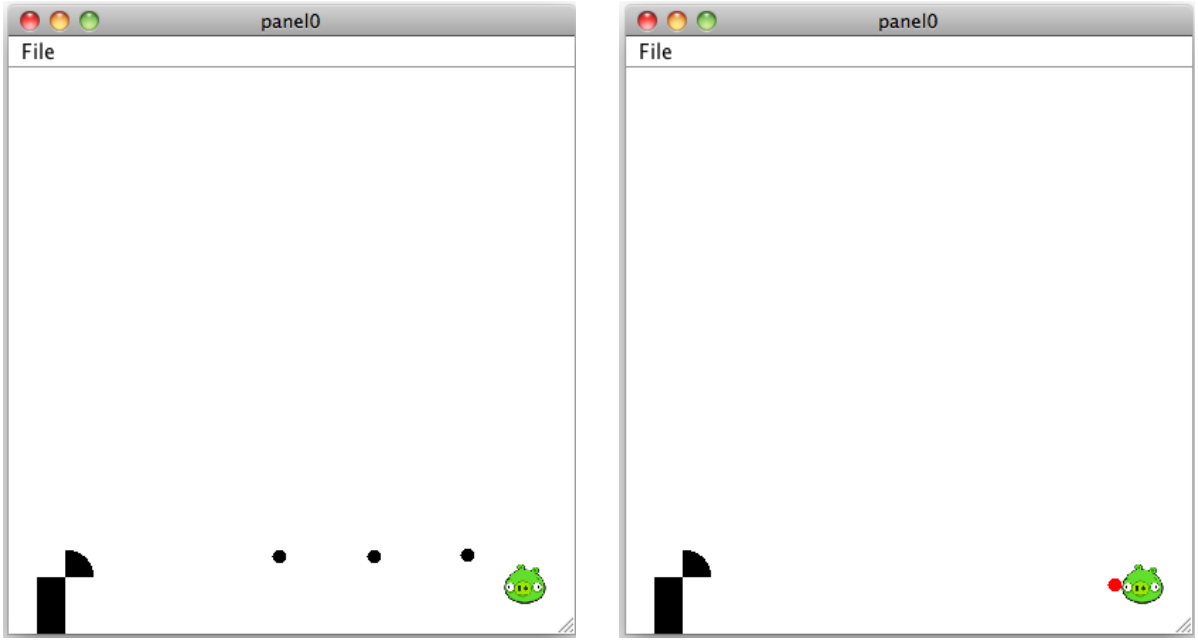
The goal of your final program is to write a simple version of the Angry Birds game.

As shown in the image on the left below, the program begins with a pig (the target) at the lower right of the canvas. At the other side of the canvas is an "exploding angry bird" launcher. If the user clicks in the wedge-shaped part of the launcher, an exploding angry bird is launched, as shown in the image on the right.





The bird travels along a parabolic path. As it descends, it "explodes" into three "eggs" that fall diagonally to the left, straight down, and to the right, respectively. Once the bird has exploded, it no longer appears on the canvas. If any of the eggs hits the target (i.e., if any egg overlaps the pig `VisibleImage`), then the egg remains on the canvas and turns red. In the image on the left below, we show the screen after the bird has exploded and on the right we show the final outcome, in which the target has been hit by the rightmost egg.



Program Design. Your program should be divided into three classes: a window controller called `AngryBirdController`, a class that `extends ActiveObject` called `ExplodingBird`, and another `ActiveObject` class called `AngryEgg`. We have provided the outlines of these classes in the starter folder. The starter folder also contains various bird and pig images for you to use.

The Window Controller. The provided window controller code already constructs the launcher. As you will see, it is one fourth of a circle of radius 20. When the user clicks in the wedge, the point on which they clicked should determine the starting velocity of the bird. The horizontal velocity is the difference between the click point's x value and the x value of the location from which the wedge visibly emerges (which is defined as the `LAUNCH_LOC` constant in the window controller). Similarly for the vertical velocity. In order to make the average starting velocities approximately 2 pixels per move, we suggest you scale them by dividing each by 5.

When you construct an `ExplodingBird`, you should tell it the starting location from which it should emerge as well as its starting velocities in the x and y directions. (Hint: you will likely need to provide other parameters as well.)

Our target is placed at location (350, 350), and we've elected to scale the pig to have width and height 30. Your placement and target size need not match these exactly.

Exploding Birds. There are many possible ways to make your exploding bird follow a nice arc-shaped path. We suggest the following. As the bird takes each step, adjust its velocity in the y direction by adding a small amount (we use 0.03) to the previous y velocity. This will cause the bird to appear to slow down as it reaches maximum altitude. It will also eventually change the y velocity from a negative value to a positive value, causing the bird to arc downward.

Our bird moves for a fixed number of steps before it explodes. The fixed number of steps we chose is one-third the width of the canvas. Feel free to experiment with other values.

When the bird explodes, it should be removed from the canvas, but not before "laying" three `AngryEgg`s. Our eggs emerge from the bottom center of the bird and move with x and y velocities of -1 and 1; 0 and 1; 1 and 1, respectively. When you construct an `AngryEgg`, you should pass it parameters to describe where it should be drawn and how quickly it should descend. (Hint: Here, too, you will likely find that you need to pass additional parameters.)

Angry Eggs. Don't forget that as an `AngryEgg` falls, it should check whether it has hit the target or not. If so, it should stop and turn red. If an egg drops off the visible part of the window, it should be removed from the canvas.

Pigging Out. The project folder for this program contains images for several individual pigs and birds, as well as larger "contact sheets" containing many more. You may use any that you like for this program (or the others). To use an image from the contact sheets, simply open the sheet in Preview, click on the "Rectangular Selection" button in the tool bar, select the piece you would like to use, and then select Copy from the Edit menu. Once you have copied the selection, choose "New From Clipboard" from the File menu and save the new image into your project folder. You can also use Preview to change image sizes, and so on. Feel free to edit these images any way you like, or find your own!

Submitting Your Work

Once you have saved your work in BlueJ, please perform the following steps to submit your assignment. *Please do not submit three separate folders. Instead, place the folders for all three of your complete programs into one folder, make sure that your name appears in the title of the main folder and each of the subfolders, and then place the main folder in our dropoff folder* with the following steps:

- First, return to the Finder. You can do this by clicking on the smiling Macintosh icon in your dock.
- From the "Go" menu at the top of the screen, select "Connect to Server...".
- For the server address, type in "afp://Guest@fuji" and click "Connect".
- A selection box should appear. Select "Courses" and click "Ok".
- You should now see a Finder window with a "cs134" folder. Open this folder.
- You should now see the drop-off folders for the three lab sections. Drag your "Project1LastNames" folder into the appropriate lab section folder. When you do this, the Mac will warn you that you will not be able to look at this folder. That is fine. Just click "OK".
- Log off of the computer before you leave.

Grading Guidelines

Points will be assigned roughly as follows:

Style (16 pts per program)

- Proper use of boolean conditions
- Proper use of ifs/whiles
- Proper use of variables (instance/local, public/private)
- Descriptive comments
- Good names
- Good use of constants
- Appropriate formatting (indenting, white space, etc.)
- Parameters used appropriately

Correctness (16 pts per program)

- Seeing Red

  Initial layout and color
  Can drag object
  Color changes properly
  Pig shown and color changes on success
  Chooses new random location for pig

- Irate Ibises

  Bird created correctly on press
  Bird grows on drag
  Eyes open when bird reaches full height
  Bird color changes correctly

- Angry Birds

  Target placed correctly at start
  Bird launched in response to a click in the launcher
  Bird launched with correct velocity
  Bird moves along a path that follows an arc
  Bird "explodes" and launches three angry eggs
  Angry eggs move along appropriate paths
  "Hit" detected if an angry egg hits the target

Miscellaneous (4 pts)

Extra Credit (up to 6 pts) While not required, feel free to experiment and add embellishments to your programs once you have the basic features working. You may receive a small amount of extra credit for any extensions.