



CS 101 - Problem Solving and Structured Programming

Lab 3 - Riding the MHC Balloon

Due: October 2/3/4/5

Pre-lab Preparation

Before coming to lab, you are expected to have:

- Read Bruce chapters 5 & 6
- Complete the table on page 4 of this handout

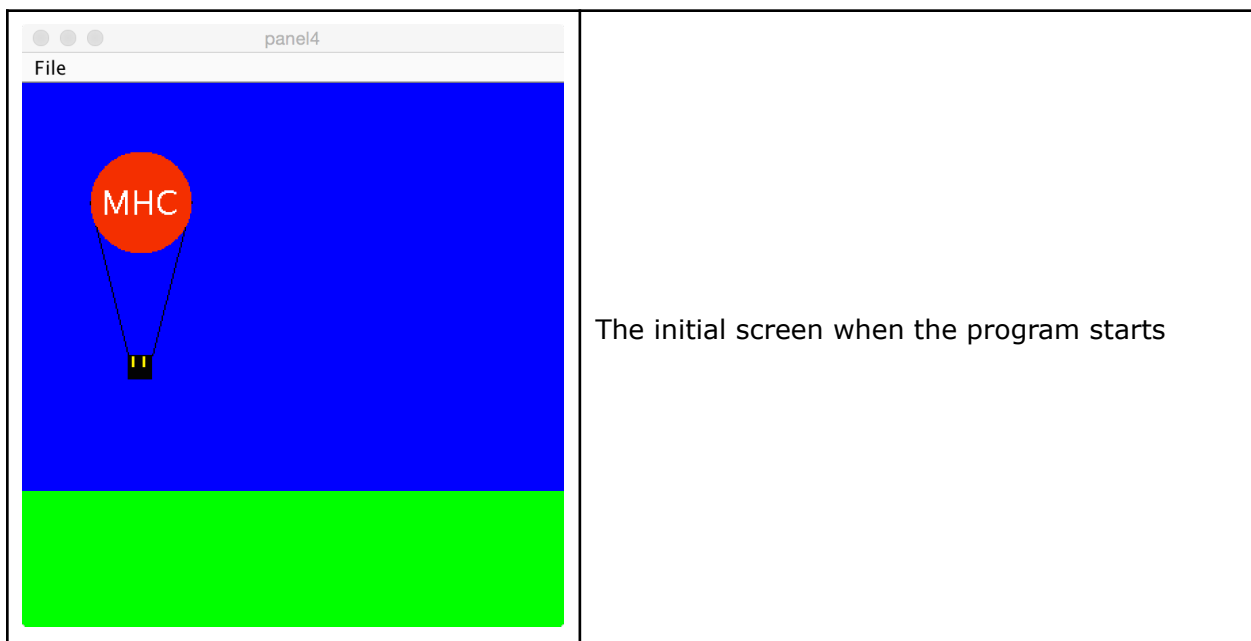
Goals:

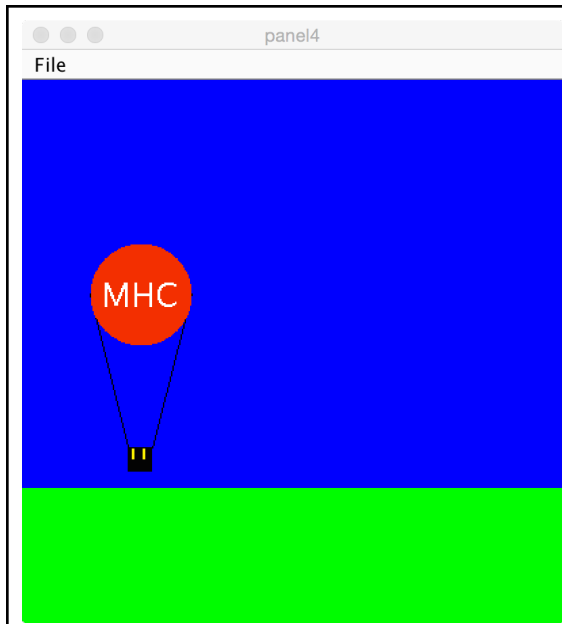
- To write a program with multiple classes
- To implement algorithms that use delegation
- To work with a random number generator

Introduction to the Assignment

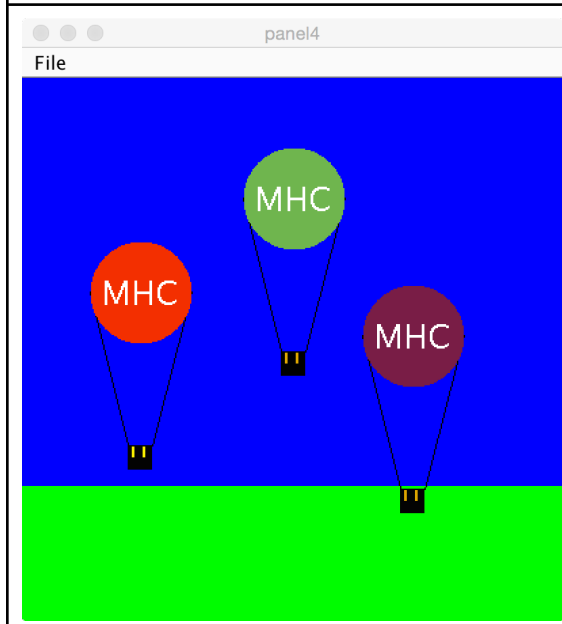
In this lab, you will create a drawing of a hot air balloon that the user can cause to rise and fall. The user will be able to drag the balloon vertically only. All horizontal mouse motion should be ignored. When the user clicks the mouse a new hot air balloon will be drawn, where the user clicks. The user will only be able to move the last balloon that is created. When the user moves the mouse out of the window, the scene will darken. When the user returns the mouse to the window, the scene will brighten.

Here are some snapshots from this program.

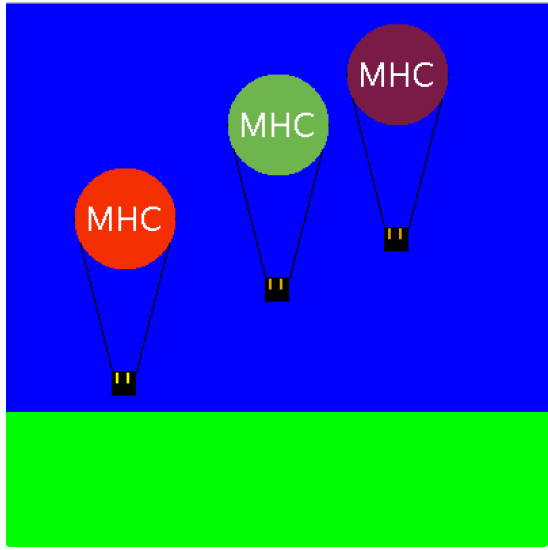
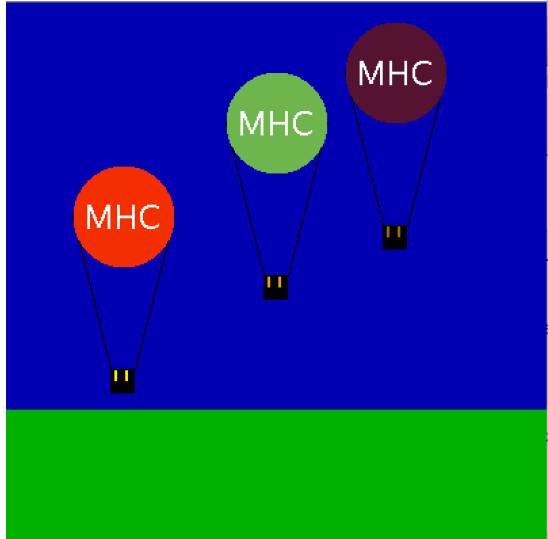




The user has dragged the balloon down. Notice that the balloon can only move up and down. It cannot move left and right.



Now the user has created multiple balloons by clicking at different places on the screen. Notice how the balloons have a random color.

	<p>The user will only be able to move the last balloon that was created. In this case, the one on the right.</p>
	<p>Extra credit: The user moved the mouse out of the window, causing the scene to get darker. The last balloon also gets darker.</p> <p>When the user moves the mouse back into the window, the scene should brighten, restoring the scene shown in the image above.</p>

Drawing a Hot Air Balloon

Before even thinking about writing this program, you should draw a hot air balloon by hand (or closely examine the images above). You will use your drawing to figure out how big the parts of the hot air balloon should be and where they should be located. As we saw with the stick figure in class, it is best to use constant values to define the size and location of just one of the shapes that makes up the hot air balloon. Then, define the sizes and locations of the other shapes relative to one that is already defined. For example, our stick figure looked like this:

We defined the size, left and top of the stick figure's head as constants and then calculated the sizes and locations of everything else relative to that. That way if we decide to make the stick figure smaller, for example, we can change the head's size and the sizes of everything else will be suitably scaled. As a reminder, here is what the Java code looks like that calculates the size and location of the body based on the head's size and location.

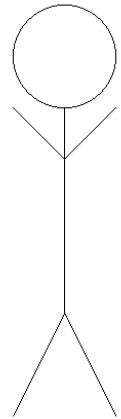
```

private static final int HEAD_LEFT = 50;
private static final int HEAD_TOP = 50;
private static final int HEAD_SIZE = 100;

private static final int BODY_LEFT = HEAD_LEFT + (HEAD_SIZE
/ 2);
private static final int BODY_TOP = HEAD_TOP + HEAD_SIZE;
private static final int BODY_SIZE = HEAD_SIZE * 2;

private static final int ARM_Y_OFFSET = HEAD_SIZE / 2;
private static final int ARM_BOTTOM = BODY_TOP + ARM_Y_OFF-
SET;

```



Before reading on, make a list of all the parts of the hot air balloon. Then pick a fixed size and location for one part and write equations you could use to calculate the sizes and locations of the other parts. Notice that an equation can use any constant that is defined earlier. For example, the ARM_BOTTOM of the stick figure shown above does not use any of the head constants. Use whatever seems most logical. The only constraint is that an equation can only use constants defined earlier in the class. Don't worry if you don't get it right the first time. In fact, it would be surprising if you did. I surely didn't! (It looked more like Picasso Balloon my first time!) Fill in the following table with this information.

Part	Width	Height	Left	Top

The Program Design

This project will have 2 classes: BalloonController and HotAirBalloon.

The BalloonController class

The BalloonController is the class that extends WindowController. It is responsible for defining the begin method and the event handlers.

- begin method - This method should draw the background consisting of the grass and sky. It should also construct one HotAirBalloon.

- `onMouseClicked` method - This method should construct a new `HotAirBalloon` where the user clicks.
- `onMousePress` / `onMouseDrag` / `onMouseRelease` - These methods should provide dragging code to allow the user to reposition the `HotAirBalloon`. Remember that the balloon should only go up and down. It should not move left and right at all.

The `BalloonController` class needs instance variables to hold the sky, grass, and `HotAirBalloon` so it can manipulate them. It also needs the usual instance variables to allow dragging.

The `HotAirBalloon` class

The `HotAirBalloon` is drawn using several shapes as you did with the train. You should define constants for the sizes of the shapes, relating those sizes with equations. Each time that you create a balloon, you should generate a random color for the balloon. The only required shapes are the balloon, the basket, and the lines connecting the two. (The MHC decoration and the sandbags are extra credit.)

You should create balloons where the user clicks, rather than always in the same location. As a result, you will need to pass the location information as parameters to the `HotAirBalloon` constructor. The location parameters should tell the constructor where the balloon part of the balloon should go and use equations to determine where to place the basket and lines to draw the rest of the balloon.

You will need instance variables to hold the parts of the balloon so that you can manipulate them later.

In addition to the constructor, the `HotAirBalloon` class must provide methods to:

- drag the balloon up and down. You will need two methods: `contains` and `move`. These will be called from `onMousePress` and `onMouseDrag`, respectively.

Documentation

There is documentation on all the `objectdraw` methods (also known as the API) available in Appendix B of the text book as well as on the course website in the left column, under Resources.

Writing the Program

Step 1: Creating the initial display

Start by writing the `begin` method in `BalloonController` to draw the sky and grass. Run your program.

Step 2: Constructing a hot air balloon

To draw a hot air balloon, you need to create a class for the hot air balloon. You will define a constructor **within the `HotAirBalloon` class** to draw the hot air balloon. The `HotAirBalloon` constructor is declared as follows:

```
public HotAirBalloon (DrawingCanvas balloonCanvas)
{

}
```

Before the constructor, you should place the declarations of the constants you will use to draw the hot air balloon. These constants should be defined based on the equations that you created on page 4 of this handout.

The statements within the HotAirBalloon constructor will be responsible for actually drawing a hot air balloon by creating shapes, as you did with the train in the first lab in the begin() method. Initially, you can draw the balloon at a fixed location. The balloon only requires the basket, lines and actual balloon. Adding sandbags (my son's idea!) or text on the balloon is extra credit. **Save that for last if you want to do it.**

Recall that when you create a shape, the last parameter indicates what "canvas" to put the shape on. For the last argument you should use the constructor parameter named balloonCanvas. Remember also to set the colors of the hot air balloon parts as you create them. You might start just drawing a couple of the hot air balloon's parts instead of trying to do it all at once.

You need to call the HotAirBalloon constructor from the begin method of the BalloonController class. The constructor call should appear after the construction of the two rectangles that make up the background. The constructor call should look like this:

```
new HotAirBalloon (canvas);
```

This tells Java to call the HotAirBalloon constructor passing the predefined canvas as the argument. Compile and run your program. If all is well, you should see a hot air balloon! Chances are, though, that things won't line up nicely or be the right size. If so, go back to BlueJ and improve the equations used to produce the sizes and locations of the parts and try again.

Once your program draws a hot air balloon, go back and check that all the equations work properly. To do this, change the value of the literal you used for your first constant to be half the value or twice the value. Think about what you expect to happen. Do you expect the hot air balloon to now be half its original size? Or closer to the left? Compile and run your program. If your equations are correct, the hot air balloon should scale or should be drawn nicely to a different location. If instead, the hot air balloon "comes apart", that is an indication that your equations are not right. Figure out which part(s) are in the wrong place or of the wrong size and fix the corresponding equations.

Step 3: Giving the balloon a random color

A balloon should have a random color. You will need a random number generator, ranging in value from 0 to 255 to generate each of the red, green, and blue values used in your color.

If you like, you can restrict the range used for any of the 3 components if you want to limit the colors produced. Remember that large numbers give lighter colors, while small numbers give darker colors.

Step 4: Dragging the HotAirBalloon

Dragging the HotAirBalloon requires similar code to what you wrote last lab to drag the parts of the snowman. In particular, notice that you will need to save the hot air balloon that you constructed in an instance variable in BalloonController (Hint: the balloon is of type HotAirBalloon - the name of the class we created to make balloons) so that you will be able to refer to it in the dragging code. In addition, you will need to decide if the user has selected the HotAirBalloon. This requires you to define a contains method in the HotAirBalloon class with the following signature:

```
public boolean contains (Location point)
```

To do the actual movement, you will need to provide a move method in the HotAirBalloon class. To keep the HotAirBalloon moving only vertically, you should ignore the x component of the mouse's motion when calculating how far to move the hot air balloon. As a result, the HotAirBal-

loon's move method only needs a vertical offset. The move method in HotAirBalloon should have this signature:

```
public void move (double dy)
```

Remember that to move the balloon, the move method that you define simply delegates to its parts, telling each of them to move just like dragging in the snowman lab.

Step 5: Creating additional hot air balloons

Next, you should modify your program so that the user can create additional balloons.

To do this, you need to provide an onMouseClick event handler in the **BalloonController class**. The event handler should call the HotAirBalloon constructor passing in the location where the new balloon should be created. This will require you to modify your HotAirBalloon constructor to include parameters for the top and left coordinates that should be used to determine where to draw the balloon.

Note that you will also need to **change the definition and the call** to the constructor that you already have to pass in top and left arguments to the constructor for the position of the initial balloon drawn. (Hint: these will be of type double)

When your program has more than one balloon, the user will only be able to drag the last balloon created. (Also, only the last balloon created will darken after you complete the extra credit. Hint: you should have an instance variable in the BalloonController to hold the last created balloon)

Extra credit: Darkening and lightening the scene

Java's Color class provides methods with the following signatures:

```
public Color darker()  
public Color brighter()
```

darker creates a new Color that is a darker version of the Color it is called on, while brighter returns a brighter color.

The objectdraw shapes provide a method:

```
public Color getColor ()
```

that allows you to determine the color of any of the shapes you have drawn.

Use these methods to darken the background and the last balloon created when the user moves the mouse out of the window.

When the user moves the mouse back into the window, brighten the background and the last balloon created.

Extra credit: Decorating the balloon

If you would like, you can add decorations to your balloon, such as the sand bags or text in the demo version of the program.

Grading

2	Pre-lab work
5	Step 1: Drawing the initial display
10	Step 2: Constructing a hot air balloon
10	Step 3: Giving the balloon a random color
10	Step 4: Dragging the HotAirBalloon
10	Step 5: Creating additional hot air balloons
10	Compiles without error
5	Comments
4	Constant declarations
4	Indentation
5	Variable naming
5	Variable scopes
5	If statements
5	Conditions
90	Total
3	Extra credit: darkening and lightening
2	Extra credit: decorating the balloon

Turning in Your Work

Your work will be automatically collected from your dev/cs101/Lab3 folder at the time that it is due. Please be sure your files are in the right place.

Over the course of the semester, you may have up to 5 late days total. Use them wisely! To request a late day, fill out the Google form on the course website so that we do not collect your assignment when it is due.