Faculty of Science, Engineering and Technology

Introduction to Programming

Credit Task 3.3: Shape Drawing

Overview

Create a program that calls procedures to draw a picture to a window.

Purpose:	Create a graphical program to explore the use of procedures and to understand that instructions are run in sequence.
Task:	Create your own Shape Drawing program to draw a picture. Submit to Canvas when complete.
Time:	This task should be started in your third lab class and submitted for feedback before the start of week 5.
Resources:	▪ Sobkowicz, M 2015 *Learn Game Programming with Ruby: Bring Your Ideas to Life with Gosu*, The Pragmatic Programmer, Chapter 3 ▪ Here is a short video on Gosu. ▪ Gosu documentation

Submission Details

You must submit the following files to Canvas:

- Picture Drawing source code
- Screenshot of the Window showing your picture.

Make sure that your task has the following in your submission:

- Your picture has at least 4 shapes, and is something other than the examples.
- Code layout - match the example for indentation and use of case.
- The code must run and the screenshot show it working on your machine.
- The program must show a picture of some kind, bonus points for some creativity.

SWINBURNE UNIVERSITY OF TECHNOLOGY

Instructions

For this task you will create a program that draws a scene using primitive shapes (triangles, rectangles, and circles).

The goal of this exercise is to learn a little about how to create a program using the library.

Gosu is a development environment that makes it easy to create programs that use graphics, sounds, animations, networking, and other aspects relevant to creating small interactive games.

1. Download the **Credit Task 3.3 code** from the Resources for this task. This contains some example code you can use to get started.
2. Extract the zip file to your code directory (e.g. Documents/Code)
3. Modify either of the programs so as to produce your desired drawing.
   - Use the following site to select colours for the circle (which uses RGB values):

     https://www.rapidtables.com/web/color/RGB_Color.html

     Or

     Use the Gosu colour constants:

     https://www.rubydoc.info/github/gosu/gosu/master/Gosu/Color

     eg: a Red circle with a radius of 50 pixels would be produced by the two statements:

```
img = Gosu::Image.new(Circle.new(50))
img.draw(200, 200, ZOrder::TOP, 0.5, 1.0, Gosu::Color::RED)
```

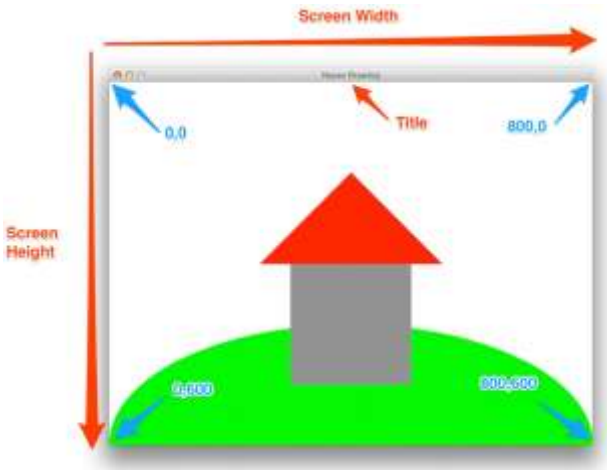
     Or you could use the HEX values:

```
img.draw(300, 50, ZOrder::TOP, 1.0, 1.0, 0xff_ff0000)
```

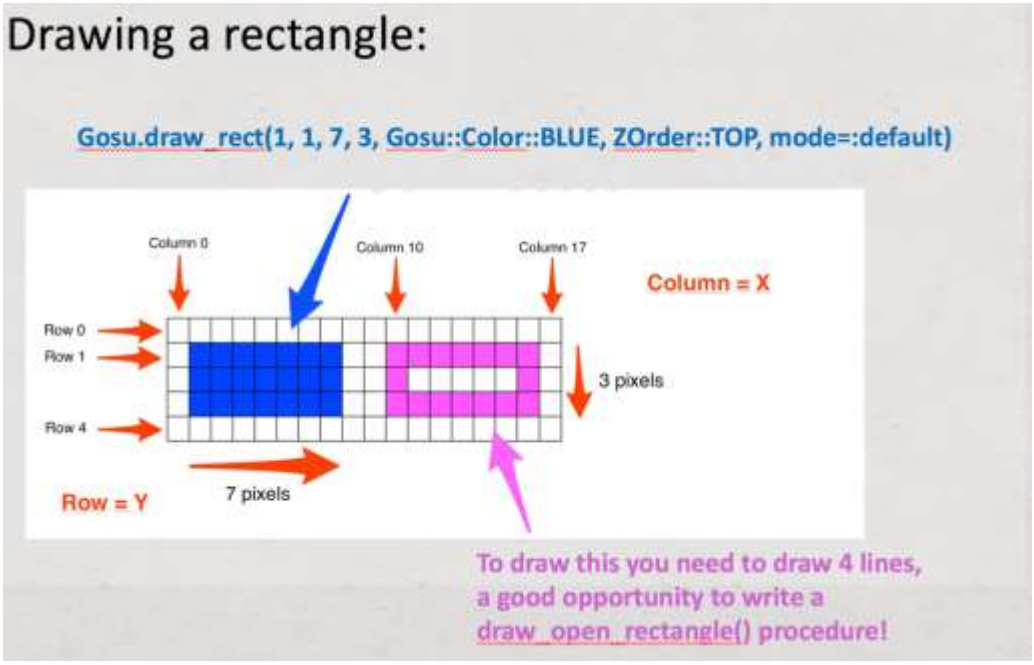
     See here to work out HEX values:

     https://www.binaryhexconverter.com/decimal-to-hex-converter

     The co-ordinate system works as follows:



Example:



4. Compile and run your program.

Now that the Task is complete you can submit it for assessment, which will help prepare it for your portfolio.

1. Use **Skitch** (or your preferred screenshot program) to take a screenshot of the Terminal, as this is one of the things you will need to submit.
2. Save the document and backup your work to multiple locations!
   - Once you get things working you **do not** want to lose them.
   - Work on your computer's storage device most of the time... but backup your work when you finish each task.
   - Use **Dropbox** or a similar online storage provider, as well as other locations.
   - Canvas is not a Backup of your work, so make sure you keep a copy!
   - A USB key and portable hard drives are good secondary backups... but can be lost/ damaged (do not rely upon them).
3. Login to Canvas, and locate Pass Task 3.3
4. Change the status of the task to **Ready To Mark**
5. Upload your completed My Functions code and the screenshot.
6. If you check back later Canvas will have prepared these as PDFs for your tutor to assess.

You now have another of your first portfolio pieces. This will help demonstrate your learning from the unit.

End of Task