Week 7 - GUI and Game Programming

Graphical Tools for Ruby

Some common GUI libraries are:

- Gosu (only for Ruby and C)
- Tcl/Tk
- FOX (FXRuby)
- Open GL

GOSU - Examples of Some features

We are going to look at an example program that includes:

- Tileable images
- Cameras

Lets see the Captain Ruby example running (the code is attached below:)

CptnRuby.zip

NB: The version of Captain Ruby provided with the code for this lecture has been modified so as to be written in a more structured way (rather than Object Oriented). We look at this Structured version in the code snippets in the later slides.

Understanding the Example Code

We will look at the construction of the Captain Ruby example. The next slides will cover:

1. Splitting Arrays
2. Some new operators
3. Tileable images
4. Creating the game terrain map
5. Camera movement

Splitting Arrays

Run

RUBY

```ruby
def main
  array = ["Fred", "Sam", "Jill", "Jenny"]

  name1, name2, name3, name4 = *array

  puts "Name 1: " + name1
  puts "Name 2: " + name2
  puts "Name 3: " + name3
  puts "Name 4: " + name4
  puts "Array: " + array.to_s
  list = *array
  puts "List: " + list.to_s
end

```

Run

RUBY

```ruby
def main
  array = ["Fred", "Sam", "Jill", "Jenny"]

  name1, name2, name3, name4 = *array

  puts "Name 1: " + name1
  puts "Name 2: " + name2
  puts "Name 3: " + name3
  puts "Name 4: " + name4
  puts "Array: " + array.to_s
  list = *array
  puts "List: " + list.to_s
end

```

Digression - some other operators

%r() is a way to write a regular expression.

%w[foo, bar] is a shortcut for ["foo", "bar"].

%q() is a way to write a single-quoted string (and can be multi-line, which is useful)

%Q() gives a double-quoted string

%x() is a shell command

%i() gives an array of symbols (Ruby >= 2.0.0)

%s() turns foo into a symbol (:foo)

%i(a b c) # => [:a, :b, :c]

Run RUBY

```ruby
if "fred@mydomain.com".match(/\w{1,10}\@\w{1,10}\.\w{1,10}/)
    puts("Email address")
else
    puts("Not Email address")
end

if "fred@mydomain.com".match(%r{\w{1,10}\@\w{1,10}\.\w{1,10}})
    puts("Email address")
else
    puts("Not Email address")
end

puts %w{one, two, three}

```

Source: https://stackoverflow.com/questions/1274675/what-does-warray-mean

Tileable Images (Sprite Sheets)

Two sets used in Gosu “Captain Ruby” example:

These are split up using code like the following:

```
game_map.tile_set =
  Gosu::Image.load_tiles("media/tileset.png", 60, 60, :tileable => true)
```

```
player.standing, player.walk1, player.walk2, player.jump =
        Gosu::Image.load_tiles("media/cptn_ruby.png", 50, 50)
```

•Each call to load_tiles returns an array of tiled images. In the second case each tile is 50 x 50 pixels.

•Each element of the array contains a drawable Image.

i See 'Learn Game Programming with Ruby', Chapt 5.

Creating the game terrain I

The terrain looks as follows, with black squares, gem squares and blocks (with or without grass):

Creating the game terrain II

The terrain is drawn based on the following text file:

Creating the game terrain III

cptn_ruby_structured.rb

Run RUBY

```ruby

# create an array to process:

a = [1, 2, 3, 1, 2, 3]

# create a new array of the same size and fill based on contents of 1st

final = Array.new(a.length) do |x|
  case a[x]
    when 1
      'a'
    when 2
      'b'
    when 3
```

Run RUBY

```ruby
lines = ["one", "two", "three"]

puts lines[0][0, 1]
puts lines[0][0, 2]
puts lines[0][0, 3]

```

The following code maps the text into a 2D array:

Here is that section of code:

```ruby
# game_map functions and procedures
# converted from OOP to Structured
# Note: I change the name to GameMap as the Map here is NOT the same
# one as in the standard Ruby API, which could be confusing.

def setup_game_map(filename)
  game_map = GameMap.new

  # Load 60x60 tiles, 5px overlap in all four directions.

  game_map.tile_set = Gosu::Image.load_tiles("media/tileset.png", 60, 60

  gem_img = Gosu::Image.new("media/gem.png")
  game_map.gems = []
```

Creating the game terrain IV

The one or zero in the tile array is used as an index into the tileset to determine which terrain image () is drawn:

The gems are drawn rotating based on the current time in a wave cycle:

RUBY

```ruby
def draw_gem(gem)
  # Draw, slowly rotating
  gem.image.draw_rot(gem.x, gem.y, 0, 25 * Math.sin(Gosu.milliseconds /
end

```

Gems are removed once a collision is detected between the gem and the player:

Run

RUBY

```ruby
def collect_gems(player, gems)
  # Same as in the tutorial game.
  gems.reject! do |c|
    (c.x - player.x).abs < 50 and (c.y - player.y).abs < 50
  end
end
```

Creating the game terrain V

Thus at the top level we have:

Gosu – Cameras I

Source: 'Learn Game Programming with Ruby', pg 158

Gosu – Cameras II

Source: 'Learn Game Programming with Ruby', pg 159

Gosu – Cameras III

```ruby
def draw
  @sky.draw 0, 0, 0
  Gosu.translate(-@camera_x, -@camera_y) do
    draw_game_map(@game_map)
    draw_player(@cptn)
  end
end
```

`Gosu::translate()` will move the camera based on the offsets you provide.

> Source: 'Learn Game Programming with Ruby', pg 159

Gosu – Sounds I

Two options:

1. Sample – a short sound that is played – perhaps as part of a game
2. Song – a longer sound file that is played – eg: for the music player.

Lets see two examples: Food Hunter and Music Player

Gosu – Sounds II: Samples

Playing sounds: Two steps

1. In the food hunter task we use the following:

```
@yuk = Gosu::Sample.new("media/Yuk.wav")
@yum = Gosu::Sample.new("media/Yum.wav")
```

2. To play the sound we simply use the following code:

```
@yum.play
```

Gosu – Sounds III: Songs

From the Music Player Task:

```
@song = Gosu::Song.new(album.tracks[track].location)
@song.play(false)
```

But you also may want to use the following:

The TK Library

A GUI library for drawing widgets like text boxes, check boxes etc.

To install: gem install tk

A tutorial:

–https://www.tutorialspoint.com/ruby/ruby_tk_guide.htm

–For message boxes see: https://tkdocs.com/tutorial/windows.html

–See also: Pragmatic Programmers Guide

TK – Message Boxes

Lets see an example (tk_test1.rb):

tk_test1.rb

```
require 'tk'

# https://tkdocs.com/tutorial/windows.html

root = TkRoot.new
root.title = "Window"

filename = Tk::getOpenFile
Tk::messageBox :message => "File is" + filename

Tk.mainloop
```

This opens a Finder window and returns the filename selected. Others include:

```
filename = Tk::getOpenFile
filename = Tk::getSaveFile
dirname = Tk::chooseDirectory
```

Tk – Text Boxes

Example (tk_test2.rb):

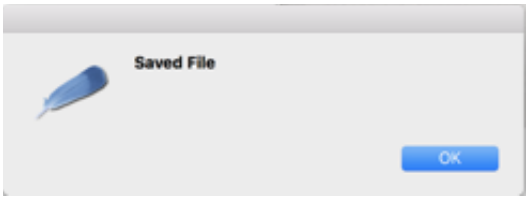
tk_test2.rb

Tk – Button

Example (tk_test3.rb):

tk_test3.rb

Includes a message box:



FXRuby

Example (texteditor.rb):

texteditor.rb

```
require 'fox16'
include Fox
```

```
app = FXApp.new
editor = TextEditor.new(app, "Simple Text Editor", 600, 400)
editor.add_menu_bar
app.create
app.run
```

This one highlights some keywords:

FXRuby II

FXRuby III

FXRuby IV

This is taken from the book: "FXRuby: Create Lean and Mean GUIs with Ruby”.

You could perhaps use FOX to write a Music Player:

See other FOX projects here:

http://fox-toolkit.org/projects.html

Open GL

Let us see the OpenGL site:

–https://www.opengl.org/about/

To install: gem install opengl

A (relatively simple) Tutorial:

–https://www.diatomenterprises.com/different-sides-of-ruby-development-opengl/

ENTIRELY OPTIONAL FOR THIS COURSE!

Open GL

OpenGL.zip

Example (opengl_integration.rb):

Food Hunter

Task 8.3D is a modification to a provided Food Hunter task which is explained in the video:

Resources.zip

RUBY

```ruby
# Encoding: UTF-8

require 'rubygems'
require 'gosu'

# Create some constants for the screen width and height

# The following determines which layers things are placed on on the scr
# background is the lowest layer (drawn over by other layers), user int

module ZOrder
  BACKGROUND, FOOD, PLAYER, UI = *0..3
end

```

Summary

We looked at what can use in the Gosu library for your custom program:

- Tiles or Sprite Sheets
- Cameras
- Sounds

We looked at other libraries you might also be interested in (optional):

- Tk
- OpenGL

The Distinction Food Hunter task was also looked at.