Introduction to Programming

Week 7, Graphical and Game Programming

Graphical Tools For Ruby

- Some common GUI libraries are:
  - Gosu (only for Ruby and C)
  - Tcl/Tk
  - Open GL

Gosu – some features

- Tileable images
- Cameras
  - Lets see the Captain Ruby example running.

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.

First: Splitting Arrays

```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
```

1. But *avoid* the above in the TUTORIAL and PASS tasks for this unit.

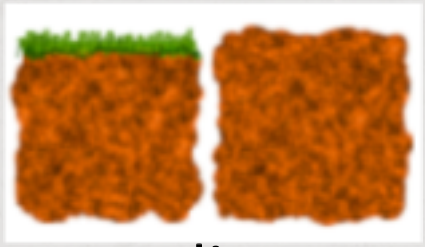
```ruby
def main
  array = ["Fred", "Sam", "Jill", "Jenny"]
  name1 = array[0]
  name2 = array[1]
  name3 = array[2]
  name4 = array[3]

  puts "Name 1: " + name1
  puts "Name 2: " + name2
  puts "Name 3: " + name3
  puts "Name 4: " + name4
  puts "Array: " + array.to_s
end
```

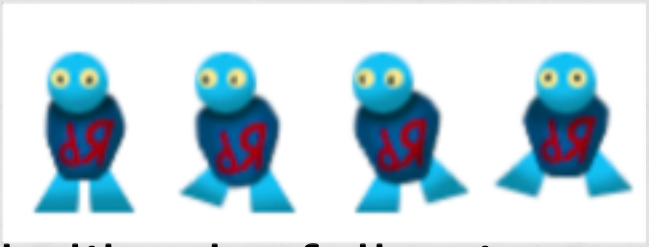
2. Instead do it like this

```
MacBook-Pro:Code mmitchell$ ruby array_split.rb
Name 1: Fred
Name 2: Sam
Name 3: Jill
Name 4: Jenny
Array: ["Fred", "Sam", "Jill", "Jenny"]
List: ["Fred", "Sam", "Jill", "Jenny"]
MacBook-Pro:Code mmitchell$
```

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)
```

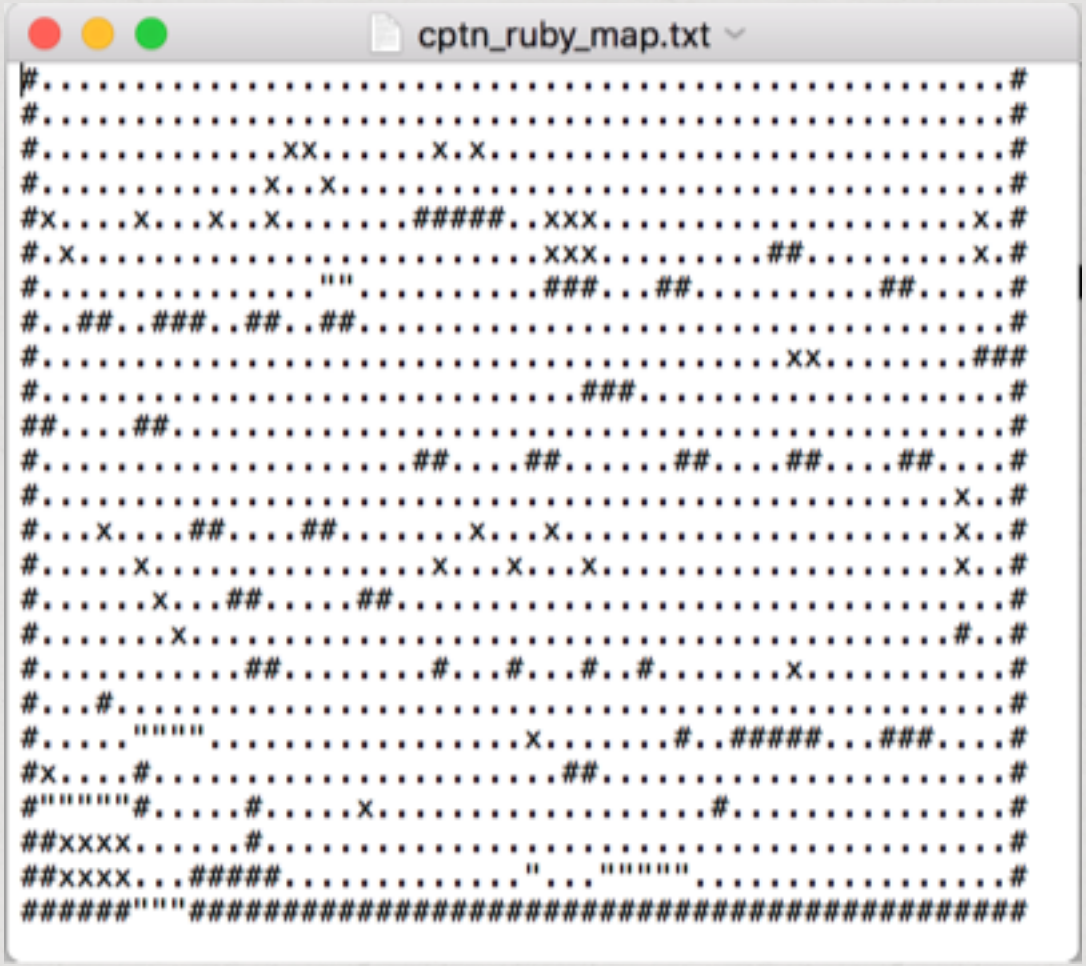
- Each call to load_tiles returns an array of tiled images. In the second case each tile is 50 x 50 pixels.
- Each element of the array contains a drawable Image.
- See *Learn Game Programming with Ruby*, Chapt 5.

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

- This is drawn based on the following text file:



Creating the game terrain III

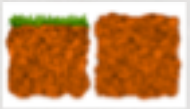
The following code maps the text into a 2D array:

```
lines = File.readlines(filename).map { |line| line.chomp }
game_map.height = lines.size
game_map.width = lines[0].size
game_map.tiles = Array.new(game_map.width) do |x|
  Array.new(game_map.height) do |y|
    case lines[y][x, 1]
    when '"'
      Tiles::Grass
    when '#'
      Tiles::Earth
    when 'x'
      game_map.gems.push(setup_gem(gem_img, x * 50 + 25, y * 50 + 25))
      nil
    else
      nil
    end
  end
end
```

```
module Tiles
  Grass = 0
  Earth = 1
end
```

Each array location will contain either 1, 0, or nil using an Enumeration (See left)

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:

```
def draw_game_map(game_map)
  # Very primitive drawing function:
  # Draws all the tiles, some off-screen, some on-screen.
  game_map.height.times do |y|
    game_map.width.times do |x|
      tile = game_map.tiles[x][y]
      if tile
        # Draw the tile with an offset (tile images have some overlap)
        # Scrolling is implemented here just as in the game objects.
        game_map.tile_set[tile].draw(x * 50 - 5, y * 50 - 5, 0)
      end
    end
  end
  game_map.gems.each { |c| draw_gem(c) }
end
```

Creating the game terrain V

Thus at the top level we have:

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

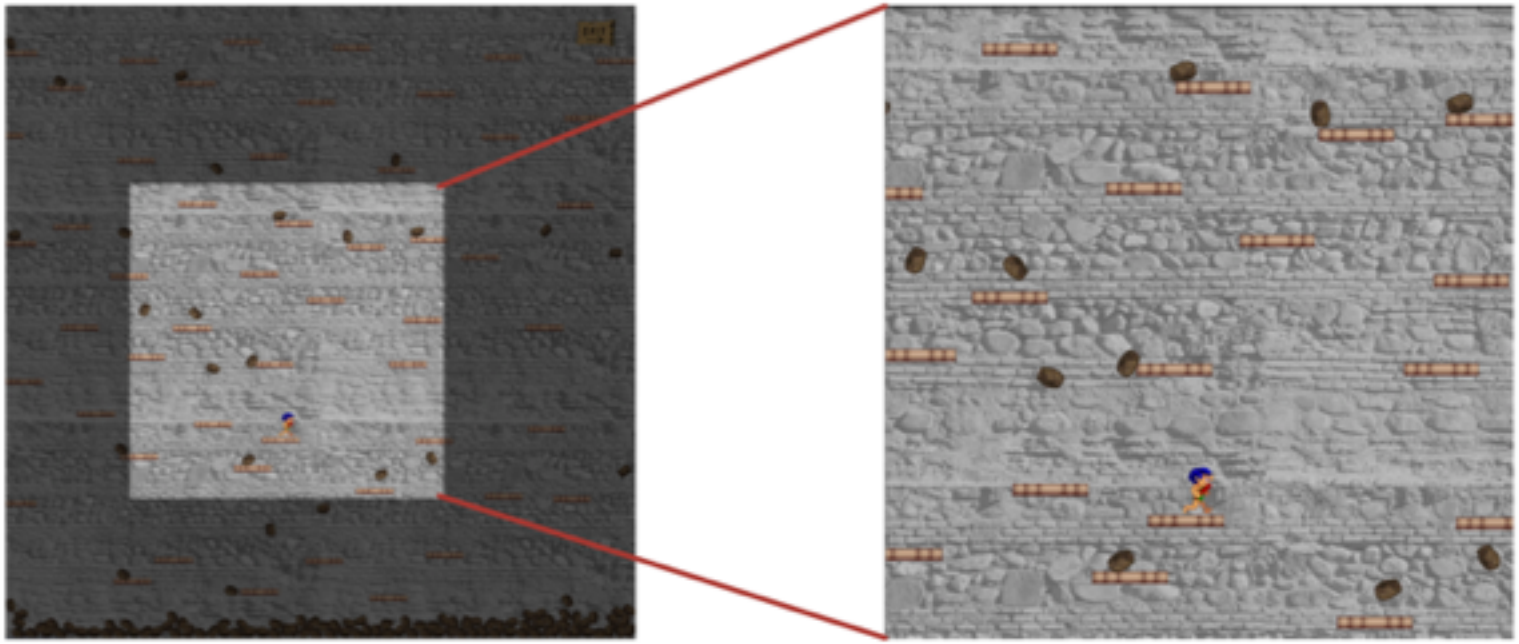
Draw the black squares

We will look at this next.

Draw the terrain tiles and gems

Draw the player

Gosu – Cameras I

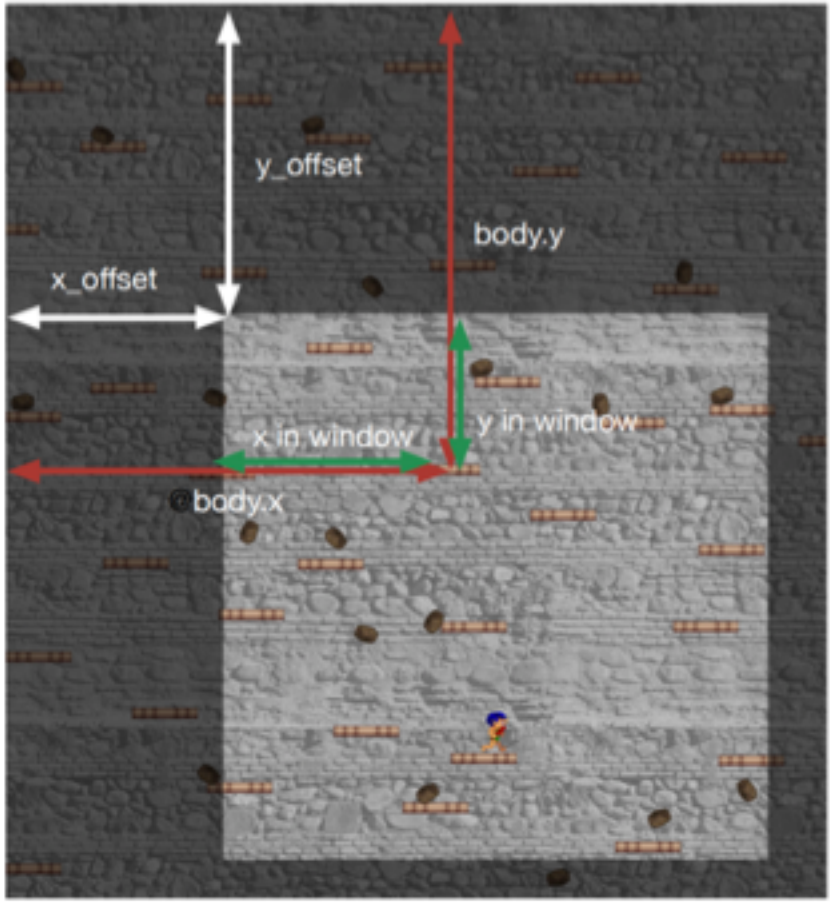


All the game objects are in the physics space.

The camera draws only some of the game window, based on Chip's location.

Source: *Learn Game Programming with Ruby* pg 158

Gosu – Cameras II



Calculating the position of an object in the window.

x = body.x - x_offset

y = body.y - y_offset

Source: *Learn Game Programming with Ruby* pg 159

Gosu – Cameras III

```
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:
  - Sample – a short sound that is played – perhaps as part of a game
  - 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
  - In the food hunter task we use the following:

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

  - 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:

#pause ⇒ void

Pauses playback of the song.

#paused? ⇒ true, false

Returns true if this song is the current song and playback is paused.

#play(looping = false) ⇒ void

Starts or resumes playback of the song.

#playing? ⇒ true, false

Whether the song is playing.

#stop ⇒ void

Stops playback if this song is the current song.

TK

- 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):

```
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
```

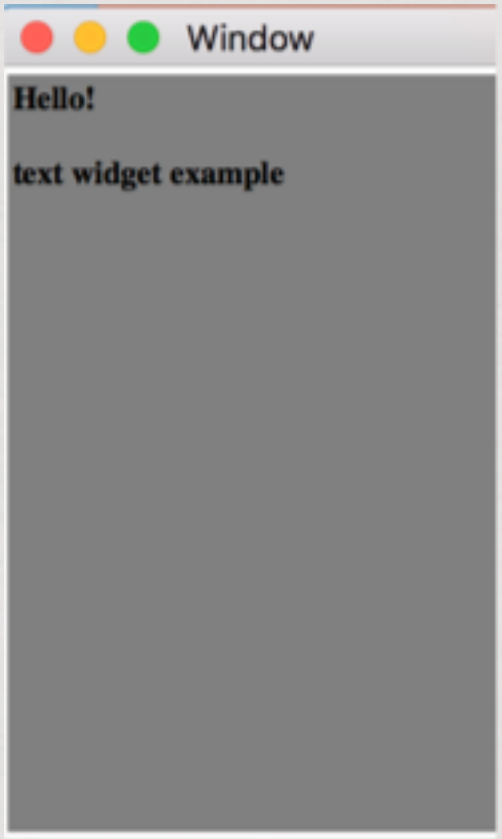
This open a Finder window and returns the filename selected. Others include:

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

Tk – Text Boxes

Example (tk_test2.rb):

```ruby
require 'tk'

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

text = TkText.new(root) do
   width 30
   height 20
   borderwidth 1
   background "gray"
   font TkFont.new('times 12 bold')
   grid('row'=>0, 'column'=>0)
end
text.insert 'end', "Hello!\n\ntext widget example"

Tk.mainloop
```

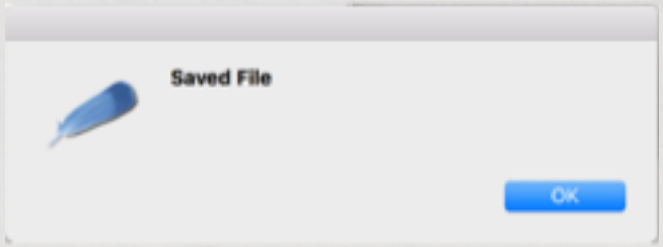


Tk – Button

Example (tk_test3.rb):

```
btn_OK = TkButton.new(root) do
   text "Save File"
   borderwidth 5
   state "normal"
   cursor "watch"
   font TkFont.new('times 20 bold')
   foreground  "red"
   activebackground "blue"
   relief      "groove"
   command (proc {Tk::messageBox :message => 'Saved File'})
   grid('row'=>2, 'column'=>0)
end
```

Includes a message box:



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

Example (opengl_integration.rb):

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