SWINBURNE UNIVERSITY OF TECHNOLOGY

Introduction to Programming

Week 4, Topic 2: GUI Control Flow.

GUI Control Flow

- In this topic we look at the Ruby Gosu cycle.
  1. Looping
  2. What the methods do
  3. Call backs

Looping

- We have been using **main** as our starting point for programs.
- To loop, we would write a **while** or **repeat** or similar loop within main. Eg:

```
def main
  finished = false
  begin
    choice = read_integer_in_range("Please enter your choice:", 1, 3)
    case choice
    when 1
      maintain_albums
    else
      puts 'Please select again'
    end
  end until finished
end
```

Gosu cycle methods

- In Gosu, we don’t need to write the main loop, it is implemented for us.
- Instead of putting our code in main, we put it in any of:
  - draw()
  - update()
  - or one of the callback methods (covered later)

Gosu cycle I

- In Gosu, we don't need to write the main loop, it is implemented for us.
- Instead of putting our code in main, we put it in any of:
  - draw()
  - update()
  - or one of the callback methods
- Most of your 'work' will go in **update()**.
- **update()** and **draw()** are called in turn in a loop that continues until you stop the program.

Gosu cycle II

```ruby
require 'gosu'

class GameWindow < Gosu::Window

  def initialize
    super 200, 135, false
    self.caption = "Gosu Cycle Example"
  end

  # Put any work you want done in update
  # This is a procedure i.e the return value is 'undefined'
  def update
  end

  # the following method is called when you press a mouse
  # button or key
  def button_down(id)
  end

  # Draw (or Redraw) the window
  def draw
  end
end
```

Gosu Call Backs

- **button_down()** is a call-back method. i.e it is called by Gosu when it detects that you have pressed a button or key.
- There is also the following, which you tend to leave unchanged:

```
def needs_cursor?; true; end
```

button_down()

- In **button_down()** you can evaluate the variable id to see what button or key was pressed.
- You also have access to two variables **mouse_x** and **mouse_y** that indicate at what position the mouse button was clicked:

```
def button_down(id)
  case id
    when Gosu::MsLeft
      @locs = [mouse_x, mouse_y]

    end
end
```

- These variables and the keys are defined in the Gosu documentation.

Alternative to button_down()

- You can also access which keys were pressed in **update()** (from the Shape Moving task):

```
def update
  if button_down?(Gosu::KbRight)
    if @shape_x != (WIDTH - SHAPE_DIM)
      @shape_x += 3
    end
  end
  if button_down?(Gosu::KbLeft) && (@shape_x != 0)
      @shape_x -= 3
  end
end
```

Lights Example

- Let’s go through building a GUI program in Ruby.
- See the code for this lecture.

This Week's Tasks:

- File Handling
- Gosu cycle
- Shape moving
- Maze Creation (HD Only) - demo.