9.3HD

COS10009

Introduction to programming

Thanh Minh

Student ID: 103809048

CODE EXPLANATION DOCUMENT

Swinburne University of Technology

---

1. **Program's Window and implement some features**

```
module ZOrder
  HIDDEN, BACKGROUND, SCENE, UI, TOP = *0..4
end
```

Figure 1: set up the ZOrder to arrange the layers for later uses in the program

Hidden will be at the back of the program when I want to hide content and create the effects so that the program can have a better looking.

I have created the constant HEIGHT(800) and WIDTH(1500) for the size of the window, and put it on the top of the file so that everyone can see clearly.

```
17  def initialize()
18   super(WIDTH,HEIGHT,false)
19   self.caption = "Spaceship Hunter"
20   @player = Player.new(self)
```

Figure 2: set up for the window

If I change false into true, the program's window will be played in full screen. I also create an array for Player for displaying ships in the Selecting page.

```
window = MyGame.new()
window.show()
```

Figure 3: End code of program

Without these 2 lines of code, the program cannot run so I must include it in every single Gosu program.

```
def update
  case @scene
  when :start
    update_start
  when :instruction
    update_instruction
  when :selecting
    update_selecting
  when :playing
    update_playing
  when :end
    update_end
  end
end
```

Figure 4: States and procedures of Update for program

```
def draw
  case @scene
    when :start
      draw_start
    when :instruction
      draw_instruction
    when :selecting
      draw_selecting
    when :playing
      draw_playing
    when :end
      draw_end
    end
  end
```

Figure 5: States and procedures of Draw for program

First I will have to set the @scene = :start so that the program will start at the entry page and will call all the procedures related to start such as draw_start, update_start, …

I must have each screen of the program, it should have different functions so that can increase the efficient of the code, make the program run smoother, faster. The code is also can be visualized easily.

2. Entry page

First, I need to set up in order to draw up the entry page for the program.

```
#--------------- start point --------------------
@scene = :start
@start_music = Gosu::Song.new('musics/start.wav')
@start_background = Gosu::Image.new('images/start.jpg')
@title = Gosu::Font.new(50)
@start_options = Gosu::Font.new(35)
@info_font = Gosu::Font.new(15)
@hover_options_1 = ZOrder::HIDDEN; @hover_options_2 = ZOrder::HIDDEN; @hover_options_3 = ZOrder::HIDDEN
@error_music = Gosu::Song.new('musics/error.wav')
@start_play_music = Gosu::Song.new('musics/start_play.wav')
@point = Gosu::Image.new('images/pointing.png')
```

Figure 6: Setup for the entry page

I need to set @scene = :start by default so that the program can start and draw the entry page first. I also nneed to import some sounds and musics into my program. I also define the font size for drawing title, description of the program so that the user can know what is on the screen.
For the @hover_options_1,2,3 , I will use it for creating effects while suffering through options so that the user can feel like the program is make with a lot of effort.
For two images that I have imported, start.jpg is for the background of the screen to decorate the screen. The other one is for pointing at the current option (there are some RPG using this function for their menu)

```
def draw_start
  @start_background.draw(0, 0, z = ZOrder::BACKGROUND)
  @start_music.play(false)
  @title.draw_text("Spaceship hunter", 650, 50, z = ZOrder::TOP, 1, 1, Gosu::Color::WHITE)
  @start_options.draw_text("Play \n\n\nInstructions \n\n\nExit
    ", 500, 300, z = ZOrder::TOP, 1, 1, Gosu::Color::WHITE)

  @point.draw(355,305,1, @hover_options_1)
  @point.draw(355,410,1, @hover_options_2)
  @point.draw(355,510,1, @hover_options_3)

  draw_quad(490,295, Gosu::Color::BLACK, 750, 295, Gosu::Color::BLACK, 750, 340, Gosu::Color::BLACK, 490, 340, Gosu::Color::BLACK, @hover_options_1)
  draw_quad(490,400, Gosu::Color::BLACK, 750, 400, Gosu::Color::BLACK, 750, 445, Gosu::Color::BLACK, 490, 445, Gosu::Color::BLACK, @hover_options_2)
  draw_quad(490,505, Gosu::Color::BLACK, 750, 505, Gosu::Color::BLACK, 750, 550, Gosu::Color::BLACK, 490, 550, Gosu::Color::BLACK, @hover_options_3)

end
```

Figure 7: draw_start procedure

After importing background image, I need to draw it so that it can display, but I will make the ZOrder of it is Background so that it won't hide other contents.

I have made the ZOrder of @point and @hover_options_1,2,3 are hidden so that @point and black rectangle won't display on the screen until their ZOrder are changed.

```
def update_start
  if (mouse_x > 489 && mouse_x < 756 && mouse_y > 294 && mouse_y < 341) or (button_down?(Gosu::KbUp) && @hover_options_2 == ZOrder::UI)
    @hover_options_1 = ZOrder::UI
    @hover_options_2 = ZOrder::HIDDEN
    @hover_options_3 = ZOrder::HIDDEN
  elsif (mouse_x > 489 && mouse_x < 756 && mouse_y > 399 && mouse_y < 446) or (button_down?(Gosu::KbUp) && @hover_options_3 == ZOrder::UI) or (button_down?
  (Gosu::KbDown) && @hover_options_1 == ZOrder::UI)
    @hover_options_1 = ZOrder::HIDDEN
    @hover_options_2 = ZOrder::UI
    @hover_options_3 = ZOrder::HIDDEN
    sleep(0.15)
  elsif (mouse_x > 489 && mouse_x < 756 && mouse_y > 504 && mouse_y < 556) or (button_down?(Gosu::KbDown) && @hover_options_2 == ZOrder::UI)
    @hover_options_1 = ZOrder::HIDDEN
    @hover_options_2 = ZOrder::HIDDEN
    @hover_options_3 = ZOrder::UI
  elsif (button_down?(Gosu::KbDown) && @hover_options_1 == ZOrder::HIDDEN && @hover_options_2 == ZOrder::HIDDEN && @hover_options_3 == ZOrder::HIDDEN)
    @hover_options_1 = ZOrder::UI
    @hover_options_2 = ZOrder::HIDDEN
    @hover_options_3 = ZOrder::HIDDEN
  elsif (button_down?(Gosu::KbUp) && @hover_options_1 == ZOrder::HIDDEN && @hover_options_2 == ZOrder::HIDDEN && @hover_options_3 == ZOrder::HIDDEN)
    @hover_options_1 = ZOrder::HIDDEN
    @hover_options_2 = ZOrder::HIDDEN
    @hover_options_3 = ZOrder::UI
  end

  if (button_down?(Gosu::KbUp) && @hover_options_1 == ZOrder::UI) or (button_down?(Gosu::KbDown) && @hover_options_3 == ZOrder::UI) or button_down?(Gosu::KbLeft) or
  button_down?(Gosu::KbRight)
    @error_music.play(true)
    sleep(0.05)
  end
end
```

Figure 8: update_start procedure

In figure 8, I have set up for the effects when suffering through options, the black box and the pointing rocket will be displayed at the current option like in figure 9

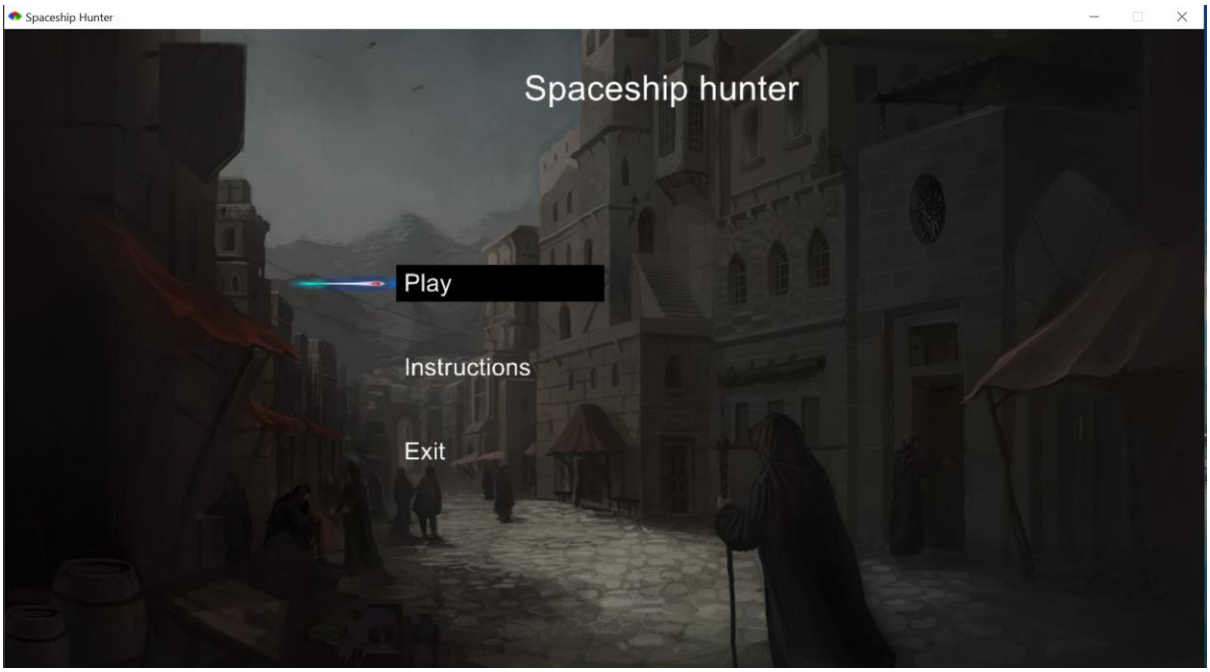

Figure 9: Entry page

```
def button_down_start(id)
  if (button_down?(Gosu::MsLeft) or button_down?(Gosu::KbReturn)) && @hover_options_1 == ZOrder::UI
    initialize_selecting
    @start_play_music.play(false)
    sleep(0.2)
  elsif (button_down?(Gosu::MsLeft) or button_down?(Gosu::KbReturn)) && @hover_options_2 ==
  ZOrder::UI
    initialize_instruction
  elsif (button_down?(Gosu::MsLeft) or button_down?(Gosu::KbReturn)) && @hover_options_3 ==
  ZOrder::UI
    close
  end

  end

end
```

Figure 10: button_down_start procedure

For this procedure, when I press the left mouse button or use Enter at the current option, it will direct me to the next page. When I choose the Play option, the start music will be played to announce me that I have entered the next stage of the game. And when I choose to close the program just choose the Exit option then the program will close immediately

3. Instruction page

```
#-------------- instruction -------------------------
@instruction_background = Gosu::Image.new('images/instruction.jpg')
@instruction_music = Gosu::Song.new('musics/instruction.wav')
@description = Gosu::Font.new(30)
```

Figure 11: set up for the instruction page

Before moving forward, I need to import music and background image for this page.

```
def draw_instruction
  @instruction_background.draw(0,0, ZOrder::BACKGROUND)
  @title.draw_text("          INSTRUCTION\n Created by Thanh Minh \n\n How to Play",
  500, 50, z = ZOrder::TOP, 1, 1, Gosu::Color::WHITE)
  @description.draw_text("Use keyboard to control:\n- ↑ or W to move up\n- ↓ or S to
  move down\n- → or D to move right\n- ← or A to move left\n- Space to
  Attack\n\n*REMEMBER: To avoid the enemy
    \nPress Q or Space to be back to the Menu",500,300, z = ZOrder::TOP, 1, 1,
    Gosu::Color::WHITE)
end
```

Figure 12: draw_instruction for the instruction page

For this procedure, mainly I draw the background image and the description for the user to read how to play this game. So that the user can understand more about the program.

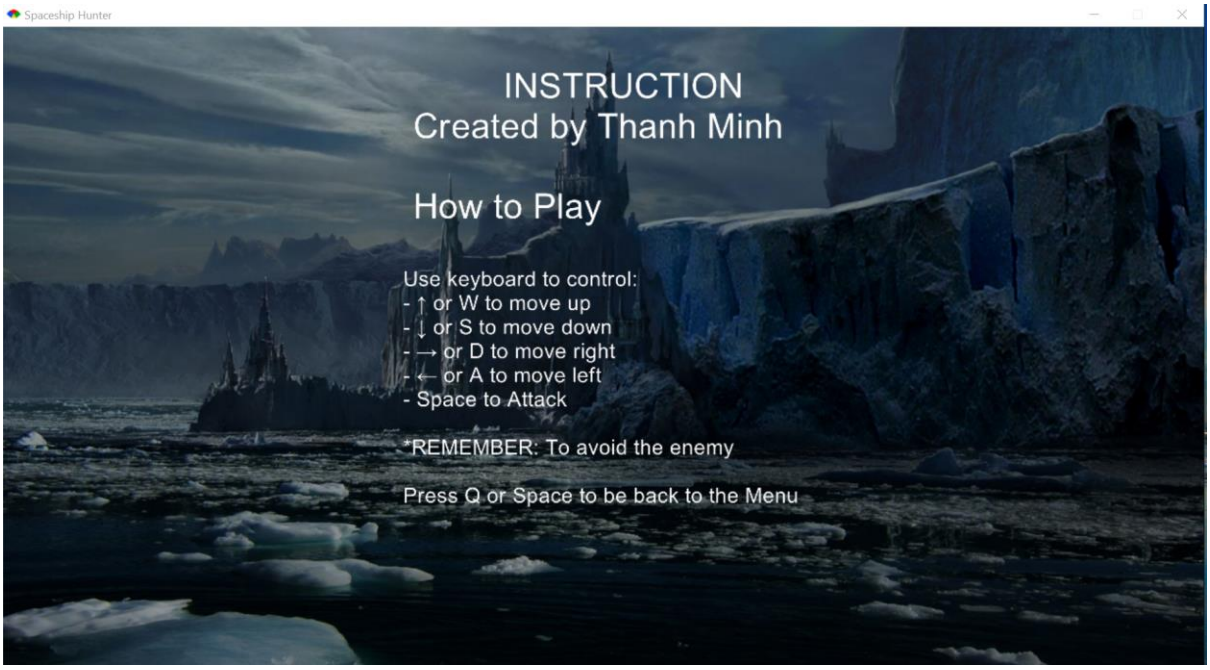

Figure 13: Instruction page

```
def update_instruction

end
```
Figure 14: Update_instruction for instruction page

This page is the easy page to be created among all the pages so in the update_instruction there is no code but if I have time to develop this project more, I will decorate more thing on this page and adding effects. Because there are some programs, games that in the instruction page, their page are so interactive and clearly show how the game work.

```
def button_down_instruction(id)
  if button_down?(Gosu::KbQ) or button_down?(Gosu::KbSpace)
    initialize_start
  end
end
```
Figure 15: button_down_instruction procedure for instruction page

When the user wants to exit and go back to the previous page, they can press Q or Space to go back.

4. **Selecting page**

At all the start of each page, I need to set up things for later use.

```
#--------------- selecting -------------------------
@selecting_music = Gosu::Song.new('musics/selecting.wav')
@select_char_1 = ZOrder::HIDDEN; @select_char_2 = ZOrder::HIDDEN
@select_color = 0xFF555555
@ship1_sel = Gosu::Image.new('images/ship1_select.png')
@ship2_sel = Gosu::Image.new('images/ship2_select.png')
```

Figure 16: Set up for Selecting page

After setting up, I also need to include the array for player for drawing it.

```
@player = Player.new(self)
```

Figure 17: Add array for player

After importing the media files, I need to draw out what I have set up above in the draw_selecting.

```
def draw_selecting
  @selecting_music.play(false)
  @title.draw_text("Selecting Your Player
    ", 550,50, z = ZOrder::TOP, 1, 1, Gosu::Color::WHITE)

    draw_quad(0,0, Gosu::Color::BLACK, WIDTH, 0, Gosu::Color::BLACK, WIDTH, HEIGHT,
    Gosu::Color::BLACK, 0, HEIGHT, Gosu::Color::BLACK, ZOrder::BACKGROUND)
    draw_quad(0,0, @select_color, WIDTH/2, 0, @select_color, WIDTH/2, 1500,
    @select_color, 0, 1500, @select_color, @select_char_1)
    draw_quad(WIDTH/2,0, @select_color, WIDTH, 0, @select_color, WIDTH, HEIGHT,
    @select_color, WIDTH/2, HEIGHT, @select_color, @select_char_2)

    @title.draw_text("Chaos Eater
      ", 250,300, z = ZOrder::TOP, 1, 1, Gosu::Color::WHITE)
    @title.draw_text("Ship Eater
      ", 1050,300, z = ZOrder::TOP, 1, 1, Gosu::Color::WHITE)
    @player.draw_select

  end
```

Figure 18: draw_selecting procedure for my program

For the first quadratic shape, I made its color Black and the ZOrder is Background and for the next two quadratic shapes I made it hidden so that when the user is choosing the player, the current option shape will be go up with the color of grey.

With the @player.draw_select, I will call the draw_select procedure for the array Player in another file so that the player can be drawn out.

```ruby
class Player
  def initialize(window)
    @x1 = 750/2
    @y1 = 650
    @x2 = 1175
    @y2 = 650
    @window = window
    @ship1 = Gosu::Image.new('images/ship1.png')
    @ship2 = Gosu::Image.new('images/ship2.png')

  end

  def draw_select
    @ship1.draw_rot(@x1 , @y1 , 1, 0)
    @ship2.draw_rot(@x2 , @y2, 1, 0)
  end
end
```

Figure 19: Array for Player

Above is the array of player I have created in another file to keep the main file shorter and clear.

```ruby
def update_selecting
  if mouse_x > 0 && (mouse_x < WIDTH/2) && mouse_y > 0 && mouse_y < 800
    @select_char_1 = ZOrder::BACKGROUND
    @select_char_2 = ZOrder::HIDDEN
  elsif (mouse_x > WIDTH/2) && mouse_x < WIDTH && mouse_y > 0 && mouse_y < 800
    @select_char_1 = ZOrder::HIDDEN
    @select_char_2 = ZOrder::BACKGROUND
  else
    @select_char_1 = ZOrder::HIDDEN
    @select_char_2 = ZOrder::HIDDEN
  end

end
```

Figure 20: update_selecting procedure

With those lines in the figure 20, the current option will be visualized for the user to see whenever they want to choose which player.

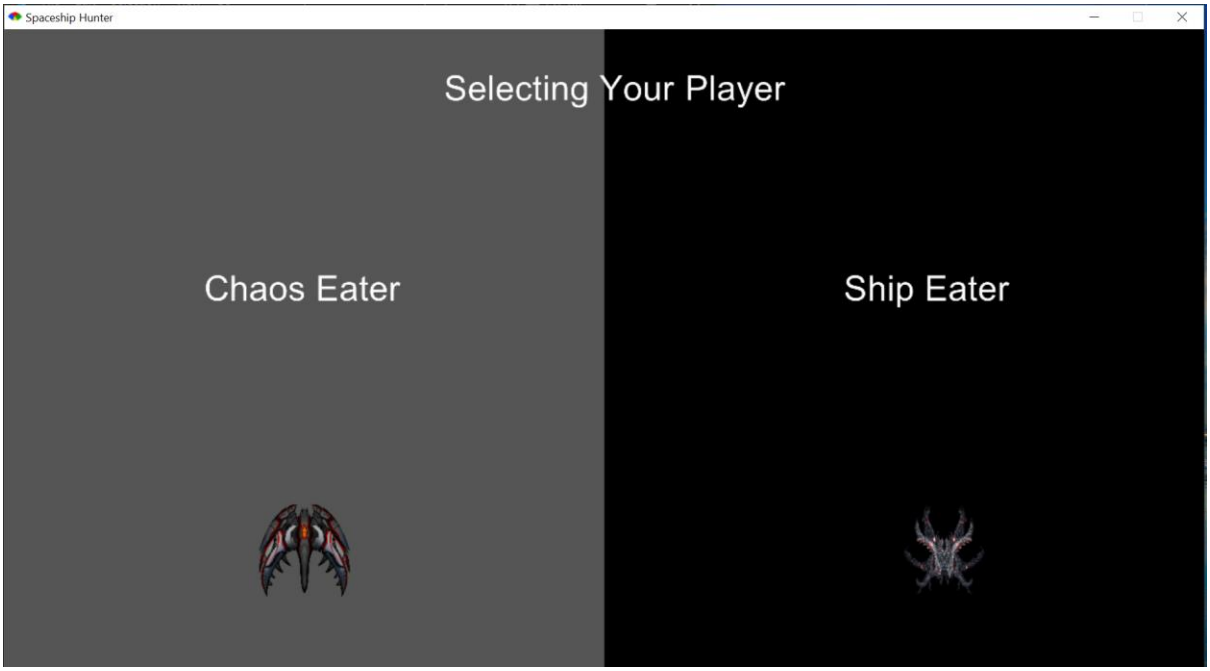

Figure 21: selecting page

The current option is chaos eater so the box on the left will be grey.

```
def button_down_selecting(id)
  if (button_down?(Gosu::KbReturn) or button_down?(Gosu::MsLeft)) && mouse_x >
  0 && mouse_x < WIDTH/2 && mouse_y > 0 && mouse_y < HEIGHT && @select_char_1
  == ZOrder::BACKGROUND
    initialize_playing
    @ship1_sel = Ship1.new(self)
    @ship = @ship1_sel
  elsif (button_down?(Gosu::KbReturn) or button_down?(Gosu::MsLeft)) &&
  mouse_x > WIDTH/2 && mouse_x < WIDTH && mouse_y > 0 && mouse_y < HEIGHT &&
  @select_char_1 == ZOrder::BACKGROUND
    initialize_playing
    @ship2_sel = Ship2.new(self)
    @ship = @ship2_sel
  elsif button_down?(Gosu::KbQ)
    initialize_start
  end
end
```

Figure 22: button_down_selecting procedure

With this procedure in figure 22, the program will know what the next step is when the user has chosen the player. The new array will be created depend on which ship the user choose and it will call exact what the user has chosen.

5. **Play and ending page**

   For these 2 stages, I am developing it so I will show it later on.