Course : Diploma in Infocomm & Media Engineering (EGDF21)

Module : Programming Methodologies and Practices (EGL145)

Topic : 14 – Getting Starting with your Project

Objectives : This topic guides you to get started with programming for your Project. Your Project is worth 20% of your grade.

Rubrics:

Your project will be graded based on the following rubrics:

|  |  |
| --- | --- |
| Criteria | Marks |
| Name your mp3 file to credit the artist | 5 |
| Number of unique patterns implemented in your dance sequence | 15 |
| Complexity of the patterns implemented | 35 |
| How colourful are the patterns | 20 |
| How well your patterns are synchronised with the music | 15 |
| Java codes are properly formatted with comments | 10 |
| Total | 100 |

**Exercise A**

1. Download Music.zip from Blackboard (<http://learn.nyp.edu.sg>) and import into your Eclipse workspace.
2. In the Music Project, take a look of the contents in the source file called YourCode.java **inside pkg**. Notice that there is a Java method as below:

**public** **class** YourCode

{

**public** **static** **void** Blink()

{

ColorDisplay.*setPixelColor*(5, 10, "RED");

ColorDisplay.*pause*(30);

ColorDisplay.*setPixelOFF*(5, 10);

}

**public** **static** **void** onBeat(**int** c)

{

System.***out***.println("At beat : " + c);

// Replace the below with your own code to dance

*Blink*(); //This is just for testing your mp3 that it responds to the beats.

}

}

1. Run the project and select your mp3. Make sure that the blinking pixels is blinking according to the beat in your music. Otherwise, select another mp3 with stronger beat.
2. In YourCode.java, the method void onBeat(int c) will be called whenever a beat is detected. The argument c represents the count number of the beat, starting from Beat 0 being the first beat of your music.
3. You will only need to write your code inside the onBeat method for this assignment. You do not need to modify code in any of the other java files.

e) Ignore the following messages when you see them in the console :

==== JavaSound Minim Error ====

==== Don't know the ID3 code TXXX

**Exercise B (Understanding the Methods in ColorDisplay Library)**

1. Try out **EVERY ONE** of the usage examples in YourCode.java.
2. **DO NOT SKIP** any of the examples. They are there to assist you in understanding every single method that is available to you for your Music Project

|  |  |  |
| --- | --- | --- |
| **Method** | **void ColorDisplay.setPixelColor(int r, int c, String g)** | |
| **Arguments** | **r :** This represents the row number of the Pixel to turn on  **c :** This represents the column number of the Pixel to turn on  **g**: This represents the color that you want the Pixel to be. Possible colors are "RED", "GREEN", "BLUE", "CYAN", "ORANGE", "GRAY" | |
| **Description** | Set the pixel represented at position (r,c) to color g | |
| **Usage Examples** | public static void onBeat(int c)  {  int i=3, j=8;  String k = "ORANGE";  ColorDisplay.setPixelColor(i, j, k);  } |  |

|  |  |
| --- | --- |
| **Method** | **void ColorDisplay.setPixelOFF(int r, int c)** |
| **Arguments** | **r :** This represents the row number of the Pixel to turn off  **c :** This represents the column number of the Pixel to turn off  Notice that setting the Pixel to color "GRAY" has the same effect as setting the Pixel OFF. |
| **Description** | Set the pixel represented at position (r,c) to gray background |
| **Usage Example** | public static void onBeat(int c)  {  int i=3, j=8;  String k = "ORANGE";  ColorDisplay.setPixelColor(i, j, k);  ColorDisplay.pause(100);  ColorDisplay.setPixelOFF(i, j);  } |

|  |  |
| --- | --- |
| **Method** | **void ColorDisplay.pause(int t)** |
| **Arguments** | **t**  This represents the time (in milliseconds) |
| **Description** | The screen will pause for duration of t milliseconds. This is to control the speed of animation on the screen. |
| **Usage Example** | public static void onBeat(int c)  {  int i=3, j=8;  String k = "ORANGE";  ColorDisplay.setPixelColor(i, j, k);  ColorDisplay.pause(100); // pause for 100ms  ColorDisplay.setPixelOFF(i, j);  } |

|  |  |
| --- | --- |
| **Method** | **void ColorDisplay.setScreenOFF()** |
| **Arguments** | **Nil** |
| **Description** | This will turn off all the Pixels on ColorDisplay. |
| **Usage Example** | public static void onBeat(int c)  {  String k = "RED";  for (c = 3; c<= 12; c++)  {  ColorDisplay.setPixelColor(4, c, k);  }  ColorDisplay.pause(100); // pause for 100ms  ColorDisplay.setScreenOFF();  } |

**Exercise C (Blinking dot)**

1. This exercises teaches you how to implement a pixel that blinks at every beat
2. Define method BlinkDot(int row, int col, String color) as

{

* turn on a pixel at (row, col) with color
* pause for 50ms
* turn off pixel at (row, col)

}

1. In the onBeat method, you can then call BlinkDot method on every beat :

**public** **static** **void** onBeat(**int** c)

{

*BlinkDot*(5, 8, "RED");

}

1. Now, let's have 2 alternating dots blinking ! We can have a blue dot that blinks on every odd beat, and a red dot that blinks on every even beat. We can therefore make use of the argument c (which is the beat counter). Below is the pseudo code :

**public** **static** **void** onBeat(**int** c)

{

if (c is odd ?)

* *Call BlinkDot*(8, 15, "BLUE");

otherwise

* *Call BlinkDot*(8, 5, "RED");

}

1. How about having 3 rotating dots blinking and so on ? We can make use of % operator !

**public** **static** **void** onBeat(**int** c)

{

if (c % 3 == 0)

* *Call BlinkDot*(8, 15, "BLUE");

otherwise if (c % 3 == 1)

* *Call BlinkDot*(4, 10, "GREEN");

otherwise

* *Call BlinkDot*(8, 5, "RED");

}

**Exercise D (Blinking line)**

1. This exercises teaches you how to implement a blinking line as shown in <https://youtu.be/Lid0bGikDkk>
2. The strategy to implement this is:

Define a method DrawVertLineAtCol(int col)

{

* Draw a vertical line of height 5 at column col by SetPixelColor for every pixels along the line
* Pause for 100 ms
* Draw a vertical line of height 5 at column col by SetPixelOFF for every pixels along the line. This will wipe off the line.

}

1. On the onBeat method, call the DrawVertLineAtCol method at each beat.
2. Refer to Topic 13 Lesson 1 where you coded a vertical line of height 6 at any column col for MonoDisplay.

**public** **static** **void** DrawVertLineAtCol(**int** col)

{

**for** (**int** i=0; i <= 5; i++)

{

MonoDisplay.*setPixelON*(i, col);

}

}

1. You can now change the arguments for the method to include height and color for use with ColorDisplay :

**public** **static** **void** DrawVertLineAtCol(**int** col, int ht, String color)

{

**for** (**int** i=0; i < ht; i++)

{

ColorDisplay.*setPixelColor*(i, col, color);

}

}

1. Add another pause before wiping off the vertical line :

**public** **static** **void** DrawVertLineAtCol(**int** col, int ht, String color)

{

**for** (**int** i=0; i<ht; i++)

{

ColorDisplay.*setPixelColor*(i, col, color);

}

ColorDisplay.pause(150);

**for** (**int** i=0; i<ht; i++)

{

ColorDisplay.*setPixelOFF*(i, col);

}

}

1. Now call this method from the main method :

**public** **static** **void** onBeat(**int** c)

{

*DrawVertLineAtCol*(5,8,"RED");

}

**Exercise E (Dancing line)**

1. The blinking line in Exercise 4 is always drawn at column 5.
2. By making use of the beat count c in onBeat method, we can draw the line at a different column on every beat.
3. However, as there are only 20 columns (0-19) on ColorDisplay, your line will go off the screen beyond 20 beats.
4. One solution is to use have beat count % 20, so that the column always stay between 0 to 19:

**public** **static** **void** onBeat(**int** c)

{

*DrawVertLineAtCol*(c % 20 ,8,"RED");

}

1. Now it is your job to think of how to change the height and color of the line with every beat.

**Exercise F (Shooting line)**

1. Refer to the following code in the previous exercise.

**public** **static** **void** DrawVertLineAtCol(**int** col, int ht, String color)

{

**for** (**int** i=0; i<ht; i++)

{

ColorDisplay.*setPixelColor*(i, col, color);

}

ColorDisplay.pause(100);

**for** (**int** i=0; i<ht; i++)

{

ColorDisplay.*setPixelOFF*(i, col);

}

}

1. You can add small delays to slow down the drawing and wiping of the line for better animated effects !

**public** **static** **void** DrawVertLineAtCol(**int** col, **int** ht, String color)

{

**for** (**int** i=0; i<ht; i++)

{

ColorDisplay.*setPixelColor*(i, col, color);

ColorDisplay.*pause*(10);

}

ColorDisplay.*pause*(100);

**for** (**int** i=0; i<ht; i++)

{

ColorDisplay.*setPixelOFF*(i, col);

ColorDisplay.*pause*(10);

}

}

1. Now instead of painting the lines upwards, you can paint the line downwards by changing the For loop to count downwards !

**Exercise G (Different patterns at different beats)**

1. Assume that you want to draw a different pattern at different sections of your song as follows:

* Between beat count 0 to beat count 15
  + Blink a red dot (5,10) at every beat
* Between beat count 16 to beat count 40
  + Blink a cyan line column 5 with height 8 at every beat
* From beat count 41 onwards
  + Blink a green line column (c % 20) with height (c % 10) at every beat

1. Try the following code:

**public** **class** YourCode

{

**public** **static** **void** BlinkDot(**int** row, **int** col, String color)

{

ColorDisplay.*setPixelColor*(row, col, color);

ColorDisplay.*pause*(50);

ColorDisplay.*setPixelOFF*(row, col);

}

**public** **static** **void** DrawVertLineAtCol(**int** col, **int** ht, String color)

{

**for** (**int** i = 0; i < ht; i++)

{

ColorDisplay.*setPixelColor*(i, col, color);

}

ColorDisplay.*pause*(150);

**for** (**int** i = 0; i < ht; i++)

{

ColorDisplay.*setPixelOFF*(i, col);

}

}

**public** **static** **void** onBeat(**int** c)

{

System.***out***.println("At beat : " + c);

**if** (c >= 0 && c <= 15)

{

*BlinkDot*(5, 10, "RED");

} **else** **if** (c > 15 && c <= 40)

{

*DrawVertLineAtCol*(5, 8, "CYAN");

} **else** **if** (c > 40)

{

*DrawVertLineAtCol*(c % 20, c % 10, "GREEN");

}

}

}

**Exercise H (Other ideas)**

* Instead of drawing vertical lines, you can draw horizontal or sloping lines.
* Drawing multiple horizontal lines and consecutive rows will result in rectangles.
* Make use of arguments so that your methods can draw lines or shapes of any sizes
* Adapt from the patterns you have drawn in Topic 13.
* Make sure you draw within column 0-19 and row 0-9. Otherwise you will encounter red error messages in the Console when you try to address a Pixel outside of the display.