EXPERIMENT NUMBER: 4
DATE: 30/08/2022, Wednesday

GRAPHICAL FRONT END DEVELOPMENT USING PROCESSING IDE ...

* **AIM:**
To develop graphical front end using prodessing IDE.

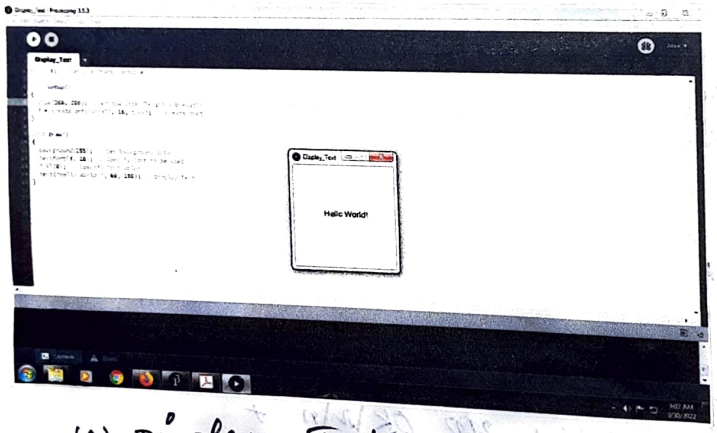
* **CODE (PROCESSING IDE):**

(a) Display Text.

```
PFont f; // Declare PFont variable

void setup()
{
  size(200, 200); // output window size (length x breadth)
  f = createFont("Arial", 16, true); // Create font
}

void draw()
{
  background(255); // set background color
  textFont(f, 16); // Specify font to be used
  fill(0); // Specify font color
  text("Hello, World!", 60, 100); // Display Text
}
```

(b) Slider control using ControlP5 Library.

```
import controlP5.*; // Import controlP5 library
ControlP5 cp5; // controlP5 object
```



(a) Display Test

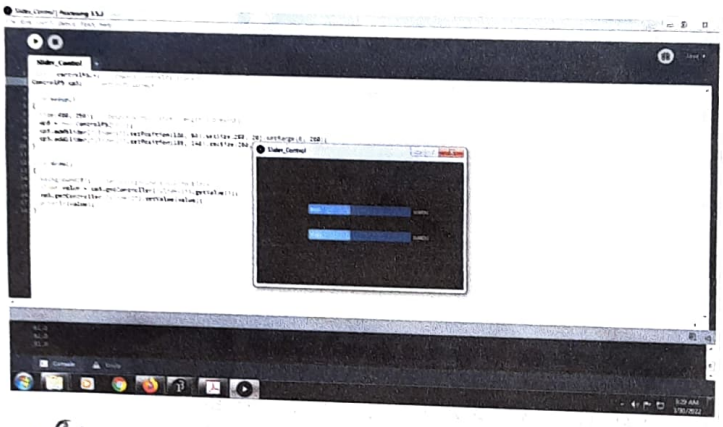
"Hello World!"

- 200 x 200 window size
- Arial Font
- Font size : 16

```
void setup ()
{
  size (400, 250); // output window size (length x breadth)
  cp5 = new ControlP5 (this);
  cp5.addSlider ("slider1"). setPosition (100, 90). setSize (200, 20).
    setRange (0, 200);
  cp5. addSlider ("slider2") . setPosition (100, 140). setSize (200, 20).
    setRange (0, 200);
}


void draw ()
{
  background (0); // Set background color to black
  float value = cp5. getController ("slider1"). getValue ();;
  cp5. getController ("slider 2"). setValue (value);
  println (value);
}
```

(c) Graphical Front End Development using ControlP5 library.

```
import controlP5. *; // Import controlP5 library
ControlP5 controlP5; // ControlP5 object
PFont f; // declare PFont variable

void setup ()
{
  size (360, 260); // output window size (length x breadth)
  smooth ();
  f = createFont ("Arial", 16, true); // create Font
  controlP5 = new ControlP5 (this);
  // Description : A bang controller triggers an event when pressed
  // Parameters : name, x, y, width, height
  controlP5. addBang ("bang1", 10, 10, 20, 20);
```

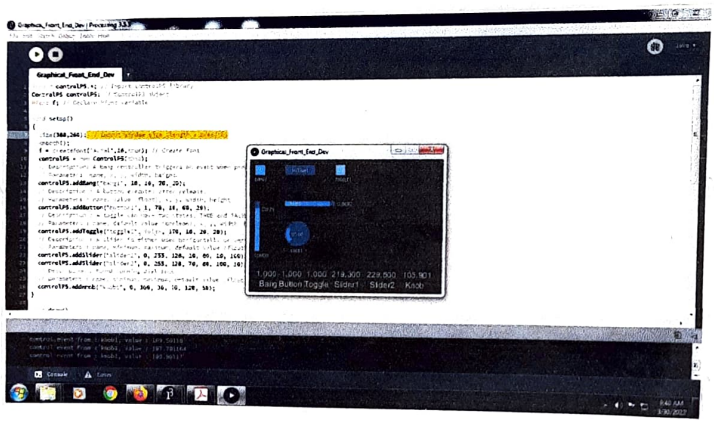


(b) Slider control using controlP5 library

- 400 x 250 window size
- 2 x sliders (Horizontal)
- Black Background
- Print slider value

```
// Description: A button executes after release.
// Parameters: name, value (float), x, y, width, height
controlP5.addButton("button1", 1, 70, 10, 60, 20);

// Description: A toggle can have two states, TRUE and FALSE
   where TRUE has the value 1 and FALSE has the value 0.
// Parameters: name, default value (boolean), x, y,
   width, height
controlP5.addToggle("toggle1", false, 170, 10, 20, 20);

// Description: A slider is either used horizontally or vertically.
// Parameters: name, minimum, maximum, default value (float),
   x, y, width, height
controlP5.addSlider("slider1", 0, 255, 128, 10, 80, 10, 100);
  // Width is bigger, you get a horizontal slider.
controlP5.addSlider("seider2", 0, 255, 128, 70, 80, 100, 10);
  // Height is bigger, you get a vertical slider.

// Description: Round turning dial knob.
// Parameters: name, minimum, maximum, default value
   (float), x, y, diameter
controlP5.addKnob("knob1", 0, 360, 30, 70, 120, 50);
}


void draw()
{
  background(0); // Set background color to black
  textFont(f, 16); // specify font to be used
  fill(255); // specify font color

  text("Bang", 20, 250);
  text("Button", 60, 250);
  text("Toggle", 110, 250);
```

(1) Graphical Front end Development using controlP5 library



- 380 x 260 Window size
- Arial Font
- Font size : 16
- objects : Bang, Button, Toggle, Vertical slider, Horizontal slider, Knob
- Black Background

```
  text ("slider 1", 170, 250);
  text ("slider 2", 240, 250);
  text ("knob", 310, 250);

  text (controlP5.getController ("bang1").getValue(), 10, 230);
  text (controlP5.getController ("button1").getValue(), 60, 230);
  text (controlP5.getController ("toggle1").getValue(), 110, 230);
  text (controlP5.getController ("slider1").getValue(), 160, 230);
  text (controlP5.getController ("slider2").getValue(), 230, 230);
  text (controlP5.getController ("knob1").getValue(), 300, 230);
}

void controlEvent (ControlEvent theEvent)
{
  /* events triggered by controllers are automatically forwarded
  to the controlEvent method. By checking the name of a
  controller, one can distinguish which of the controllers
  has been changed. */

  /* check if the event is from a controller, otherwise you'll
  get an error when clicking other interface elements like radio
  button that doesn't support the controller() methods. */

  if (theEvent.isController())
  {
    print ("control event from : " + theEvent.getController().
    getName());
    println (", value: " + theEvent.getController().getValue());
  }
}
```

(d) Display Image -

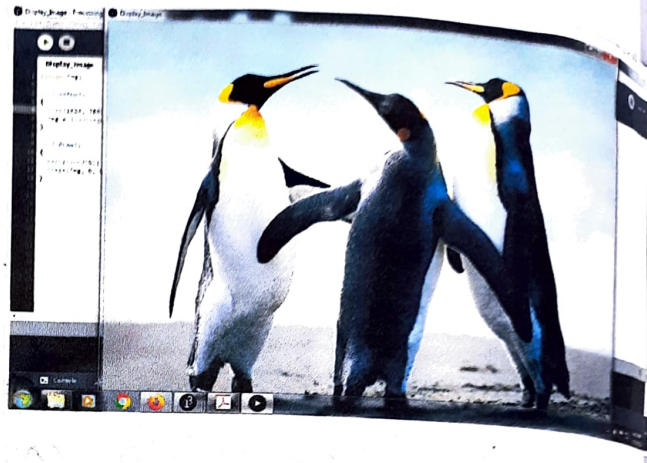
```
PImage img;

void setup()
{
  size(1024, 768); // output window size (length x breadth)
  img = loadImage("Penguins.jpg"); // make a new instance
   of a PImage by loading an image file.
}

void draw()
{
  background(0); // set background color to black
  image(img, 0, 0, width, height); // The image() function
    displays the image at a location, in this case the point (0,0)
}
```

(e) Display Image at Half of its Size -

```
PImage img;

void setup()
{
  size(1024, 768); // output window size (length x breadth)
  img = loadImage("Penguins.jpg"); // make a new instance
   of a PImage by loading an image file.
}

void draw()
{
  background(0); // set background color to black.
  image(img, 0, 0, width/2, height); // Displays the image at
    point (0,0) at half of its size.
```

(d) Display Image

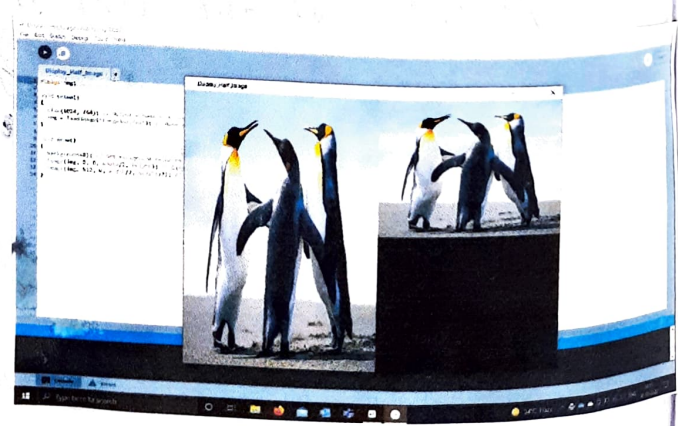
Original Image
size : 1024 x 768 pixels.



Displayed Image.
- Window size: 1024 x 768
- Black Background
- Point : (0, 0)

(e) Display Image at Half of its size

Original Image
size : 1024 x 768 pixels



Displayed Image
- Window size: 1024 x 768
- Black Background
- First point : 10, 0, w/2, h)
- Second point: 1512, 0, w/2, h/2)

```
    image(img, 512, 0, width/2, height/2); // The image()
    function displays the image at a location, in this case
    the point (512, 0);
}
```

(f) Display Image using Pixel Array -

```
PImage img;

void setup()
{
  size(1024, 768); // Output window size (length x breadth)
  img = loadImage("Tulips.jpg");
}

void draw()
{
  loadPixels(); // Call loadPixels() on the PImage to read
                its pixels.
  img.loadPixels();
  for(int x=0; x< width; x++)
  {
    for(int y=0; y< height; y++)
    {
      int loc = x + y * width;

      // The functions red(), green(), and blue() pull out the
        three color components from a pixel:
      float r = red(img.pixels[loc]);
      float g = green(img.pixels[loc]);
      float b = blue(img.pixels[loc]);

      pixels[loc] = color(r, g, b); // Set the digital pixel to the
        image pixel
```

(f) Display Image using Pixel Array

Original Image
Size: 1024 x 768 pixels

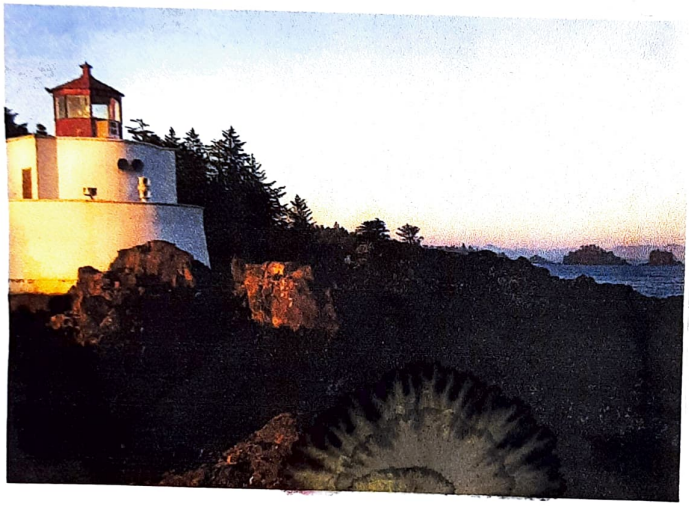
Displayed Image
window size: 1024 x 768

```
    }
  }
  updatePixels();
}
```

(9) Threshold Image using Pixel Array -

```
PImage source; // source image
PImage destination; // Destination image

void setup()
{
  size(1024, 768); // output window size (length x breadth)
  source = loadImage("Lighthouse.jpg");
  destination = createImage(source.width, source.height, RGB);
}

void draw()
{
  float threshold = 121;
  source.loadPixels();
  destination.loadPixels();
  for(int x=0; x < source.width; x++)
  {
    for(int y=0; y < source.height; y++)
    {
      int loc = x + y * source.width;

      // Test the brightness against the threshold:
      if(brightness(source.pixels[loc]) > threshold)
      {
        destination.pixels[loc] = color(255); // white
      }
```

(g) Threshold Image using Pixel Array



original Image
size: 1024 x 768 pixels

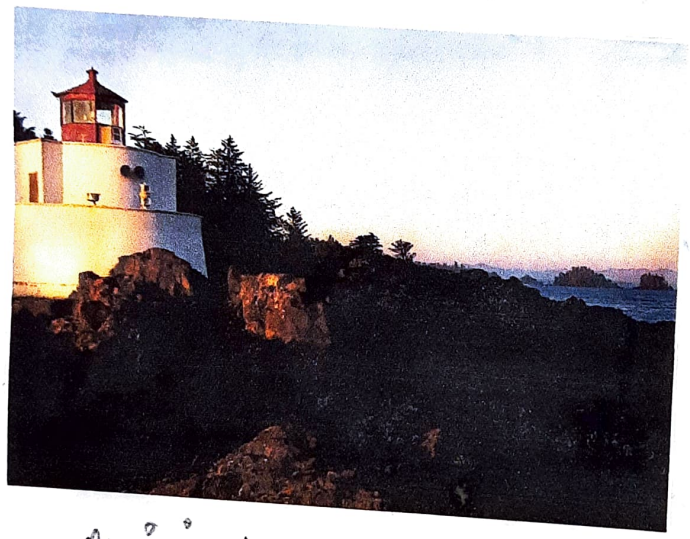
Threshold Image
- Window size: 1024 x 768
- Threshold Value: 127
- Destination point: (0, 0)

```
        else
        {
          destination.pixels[loc] = color(0); // Black
        }
      }
    }

    destination.updatePixels(); // Update the pixels in destination
    image(destination, 0, 0); // Display the destination
}
```

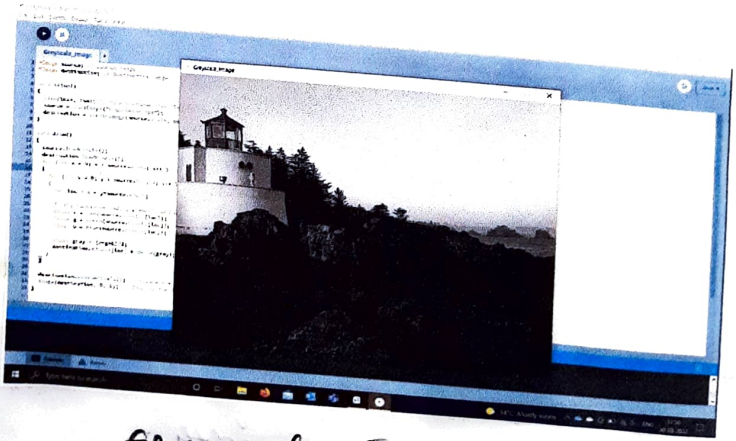
(2) Greyscale Image using Pixel Array -

```
PImage source; // Source image
PImage destination; // Destination image

void setup()
{
  size(1024, 768); // Output window size (length x breadth)
  source = loadImage("lighthouse.jpg");
  destination = createImage(source.width, source.height, RGB);
}

void draw()
{
  source.loadPixels();
  destination.loadPixels();
  for(int x=0; x < source.width; x++)
  {
    for(int y=0; y < source.height; y++)
    {
      int loc = x + y * source.width;
```

(A) Greyscale Image using Pixel Array



Original Image
Size: 1024 x 738 pixels



Greyscale Image
- Window size: 1024 x 768
- Gray = (Red + Green + Blue)/3
- Destination Point: (0,0)

```
      // The functions red(), green(), and blue() pull out
         the three components from a pixel:
      float r = red(source.pixels[loc]);
      float g = green(source.pixels[loc]);
      float b = blue(source.pixels[loc]);

      float gray = (r+g+b)/3;
      destination.pixels[loc] = color(gray);
    }
  }

  destination.updatePixels(); // Update the pixels in destination
  image(destination, 0, 0); // Display the destination
}
```

+ INFERENCE:

Analyzed and developed graphical front end using processing IDE and all simulation results were verified successfully.