

Laporan Praktikum Godot 5

Muhammad Fauzan Lubis

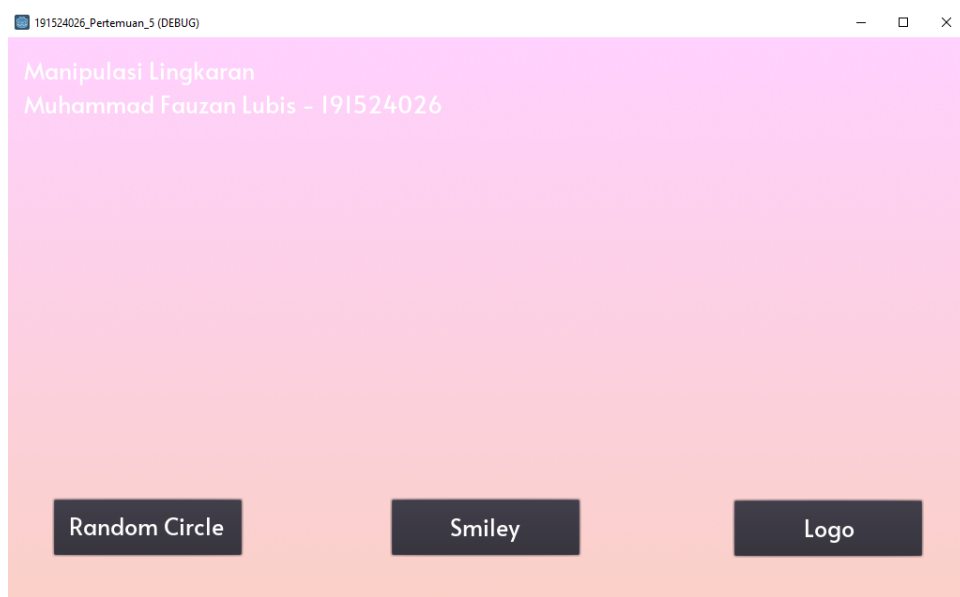
D4-2A

191524026

Rangkuman

Home

Pembuatan home sama saja dengan pembuatan menu pada sebelumnya. Nama dan nim, beserta judul, dan juga 3 tombol ke scene lainnya. Menu-nya terlihat seperti ini,



Source Code

```
extends Node2D

func _on_RandomCircleButton_pressed():
    get_tree().change_scene("res://random_circle/RandomCircle.tscn")

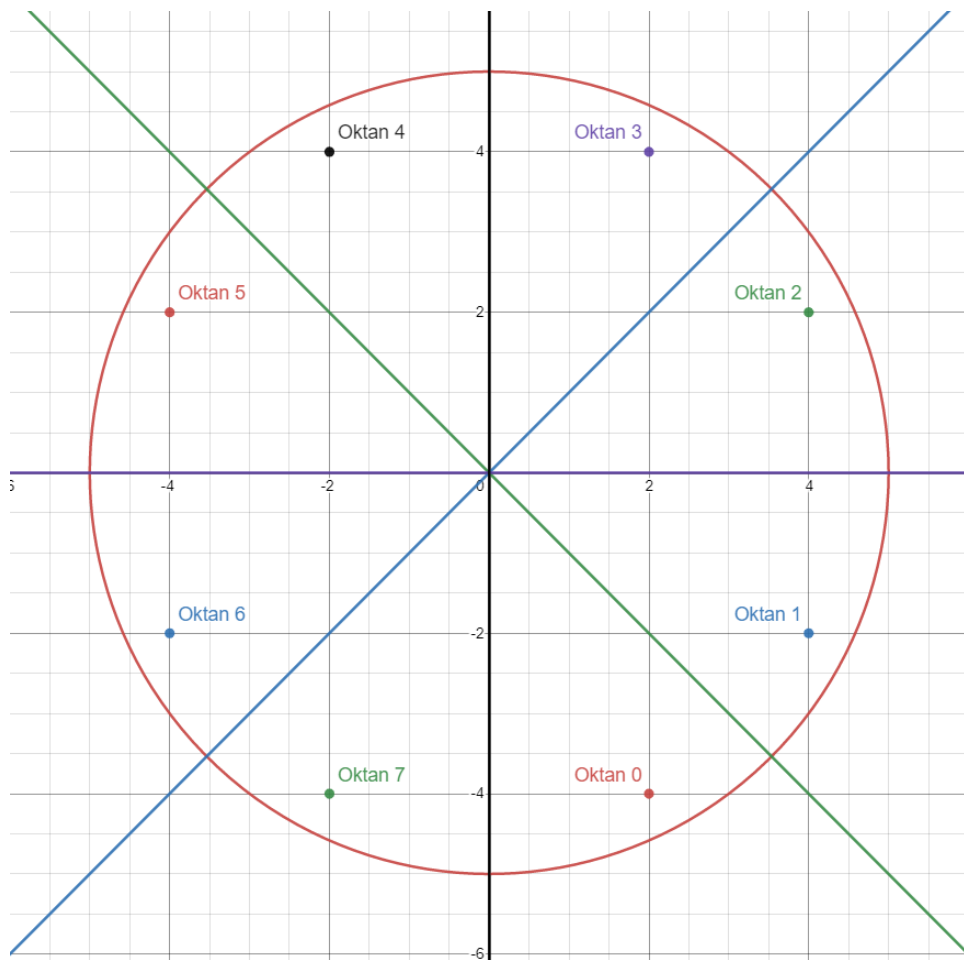
func _on_SmileyButton_pressed():
    get_tree().change_scene("res://smiley/Smiley.tscn")
```

```
func _on_Logo_pressed():  
    get_tree().change_scene("res://logo/Logo.tscn")
```

Lingkaran Random

Algoritma yang dipakai dalam membuat lingkaran di sini adalah midpoint algorithm. Algoritma ini hanya akan menghitung posisi dari salah satu oktan di lingkaran, dan melakukan transformasi ke oktan lainnya.

Dalam implementasi, karena melihat dengan kebutuhan ke depannya yang membutuhkan warna, dan apakah satu oktan akan di gambar atau tidak. Saya lakukan pembagian oktan dengan nomor tertentu, nomor yang digunakan terlihat seperti ini,



Jadi di dalam algoritma, setiap garis oktan dapat antara di gambar atau tidak, dan juga diwarnai apa. Karena konfigurasi yang banyak, dibentuklah suatu kelas tersendiri untuk konfigurasi penggambaran lingkaran,

```

class_name CircleParams

var octane_to_draw = []
var color_of_octane = PoolColorArray()

func _init(draw_octane: Array = [], octane_color: PoolColorArray = []):
    if draw_octane.size() < 8:
        initialize_octane_to_draw()
    else:
        octane_to_draw = draw_octane

    if octane_color.size() < 8:
        initialize_octane_color()
    else:
        color_of_octane = octane_color

func randomize_color():
    for i in range(8):
        randomize()
        color_of_octane[i] = Color(randf(), randf(), randf())

func initialize_octane_to_draw():
    for _i in range(8):
        octane_to_draw.append(true)


func initialize_octane_color():
    for _i in range(8):
        color_of_octane.append(Color.white)

```


Di sini terdapat array boolean untuk menentukan apakah oktan akan di gambar, misalkan oktan 1 akan digambar jika `octane_to_draw[1] == true`. Sama halnya dengan warna dari oktan 1 yang merupakan `color_of_octane[1]`.

Sementara untuk implementasi dari algoritmanya sendiri, karena cukup panjang dapat dilihat di link di bawah ini,

fauh45/KomGraf
 Computer Graphic of JTK 4th Semester . Contribute to fauh45/KomGraf development by creating an account on GitHub.
https://github.com/fauh45/KomGraf/blob/main/191524026_Pertemuan_5/Shapes.gd

fauh45/KomGraf
 Computer Graphic of JTK 4th Semester
 

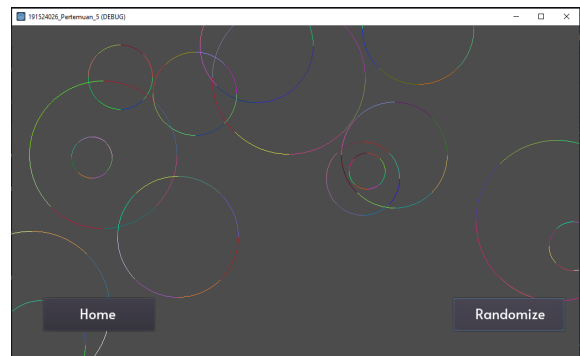
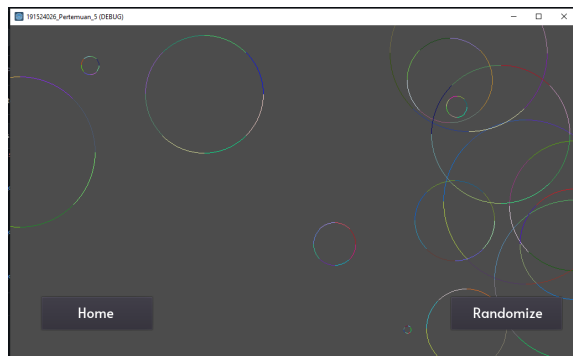
1 Contributors
 0 Issues
 0 Stars
 0 Forks



Fungsi yang digunakan dalam membuat lingkaran itu sendiri adalah `draw_circle_midpoint` dan juga `put_pixel_every_octane` yang akan menggambar

titik di semua oktan, atau kurang lebih akan melakukan translasi dari oktan yang awal. Di situ juga terlihat bahwa digunakan kelas `CircleParams` untuk menggambarkan oktan tertentu, dan juga warna tertentu.

Dalam membuat scene yang akan menampilkan lingkaran yang random itu sendiri, dibuat seperti scene yang lainnya. Tombol untuk kembali ke Home, dan tombol yang akan melakukan randomisasi lagi. Scene-nya sendiri terlihat seperti ini,



Source Code

```
extends Shapes

const NUM_CIRCLES = 15

func _draw():
    var viewport_size = get_viewport().size

    for _i in range(NUM_CIRCLES):
        randomize()

        var params = CircleParams.new()
        params.randomize_color()

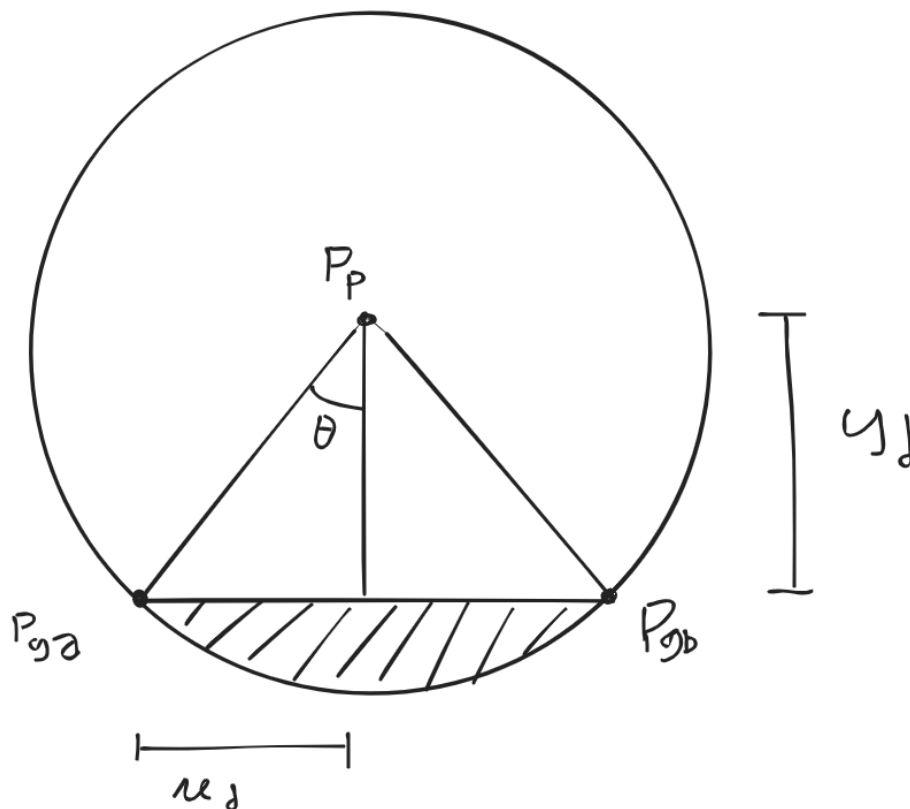
        draw_circle_midpoint(
            rand_range(5, 150),
            Vector2(rand_range(0, viewport_size.x), rand_range(0, viewport_size.y)),
            params
        )

func _on_HomeButton_pressed():
    get_tree().change_scene("res://menu/Menu.tscn")

func _on_Randomize_pressed():
    update()
```

Smiley

Smiley yang di gambar di sini menggunakan beberapa komponen dalam penggambarannya. Komponen pertama adalah mulutnya, mulut dari smiley ini merupakan lingkaran yang hanya 2×45 derajat. Jadi yang di gambarkan hanyalah oktan 7 dan 0.



Ditambah dengan garis dari ujung ke ujung, yaitu dari titik P_{ga} ke P_{gb} . Untuk mencari posisi dari kedua titik itu tersebut, dibutuhkan jarak horizontal dan vertikal dari titik $P_p(x_p, y_p)$. Diketahui bahwa $\theta = 45^\circ$ karena theta merupakan 1 oktan dari suatu lingkaran yang digambar, $\frac{1}{8}180^\circ = 45^\circ$. Dan r merupakan radius dari lingkaran tersebut. Maka bisa diketahui kedua jarak itu,

$$\cos 45^\circ = \frac{y_d}{r}$$

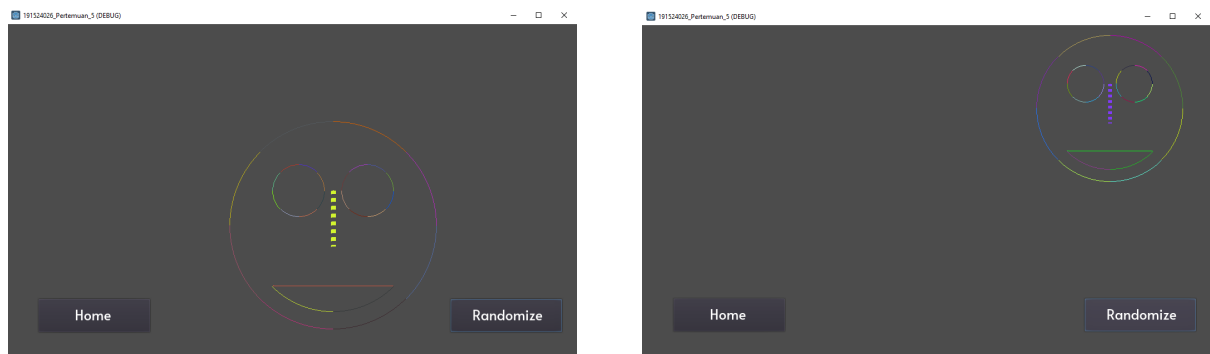
$$y_d = \frac{\sqrt{2}}{2}r$$

$$\sin 45^\circ = \frac{x_d}{r}$$

$$x_d = \frac{\sqrt{2}}{2}r$$

Jadi diketahui bahwa $y_d = x_d$. Jadi, $P_{ga} = (x_p - \frac{\sqrt{2}}{2}r, y_p + \frac{\sqrt{2}}{2}r)$ dan $P_{gb} = (x_p + \frac{\sqrt{2}}{2}r, y_p + \frac{\sqrt{2}}{2}r)$. Komponen lainnya dari smiley ini adalah seperti matanya, dan hidungnya hanya merupakan penggambaran biasa dengan margin tertentu yang sudah diatur.

Semua jarak-jarak, dan juga besaran dari komponen smiley ini di kalikan dengan suatu skala. Hal ini dilakukan agar dapat digunakan transformasi besar-kecilnya. Juga posisi di viewport di perhatikan. Smiley-nya itu sendiri terlihat seperti ini,



Source Code

```
extends Shapes

const DEG_30 = 0.7
const SMILEY_BASE_WIDTH = 120

func _draw():
    var viewport_size = get_viewport().size

    randomize()
    var scale: float = rand_range(0.2, 2)
    var width_padding = scale * SMILEY_BASE_WIDTH
    var center_point = Vector2(
        rand_range(0, viewport_size.x - width_padding),
        rand_range(0, viewport_size.y - width_padding)
    )

    draw_smiley(center_point, scale)

func draw_smiley(center_point: Vector2, scale: float):
    draw_smiley_mouth(center_point, scale)
    draw_smiley_outer_ring(center_point, scale)
    draw_smiley_eye_and_nose(center_point, scale)
```

```

func draw_smiley_eye_and_nose(center_point: Vector2, scale: float):
    var radius = 30 * scale

    var scaled_add = 25 * scale
    var x_seperation = radius / 2 + scaled_add
    var y = center_point.y - radius / 2 - scaled_add

    var params_1 = CircleParams.new()
    params_1.randomize_color()

    var params_2 = CircleParams.new()
    params_2.randomize_color()

    draw_circle_midpoint(radius, Vector2(center_point.x - x_seperation, y), params_1)
    draw_circle_midpoint(radius, Vector2(center_point.x + x_seperation, y), params_2)

    draw_line_custom(
        Vector2(center_point.x, y),
        Vector2(center_point.x, center_point.y + scaled_add),
        Color(randf(), randf(), randf()),
        5 * scale,
        true,
        5 * scale,
        5 * scale
    )

func draw_smiley_outer_ring(center_point: Vector2, scale: float):
    var params = CircleParams.new()
    params.randomize_color()

    draw_circle_midpoint(SMILEY_BASE_WIDTH * scale, center_point, params)

func draw_smiley_mouth(center_point: Vector2, scale: float):
    var params = config_draw_only([0, 7])
    params.randomize_color()

    var radius = 100 * scale

    var x_diff = DEG_30 * radius
    var y = center_point.y + (0.7 * radius)

    draw_circle_midpoint(radius, center_point, params)
    draw_line_custom(
        Vector2(center_point.x - x_diff, y),
        Vector2(center_point.x + x_diff, y),
        Color(randf(), randf(), randf())
    )

func _on_HomeButton_pressed():
    get_tree().change_scene("res://menu/Menu.tscn")

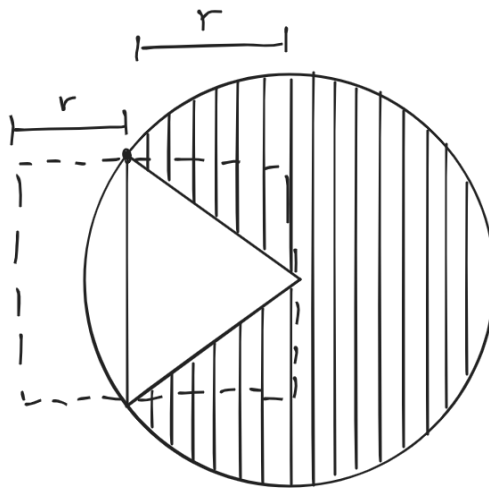
func _on_Randomize_pressed():
    update()

```

Logo

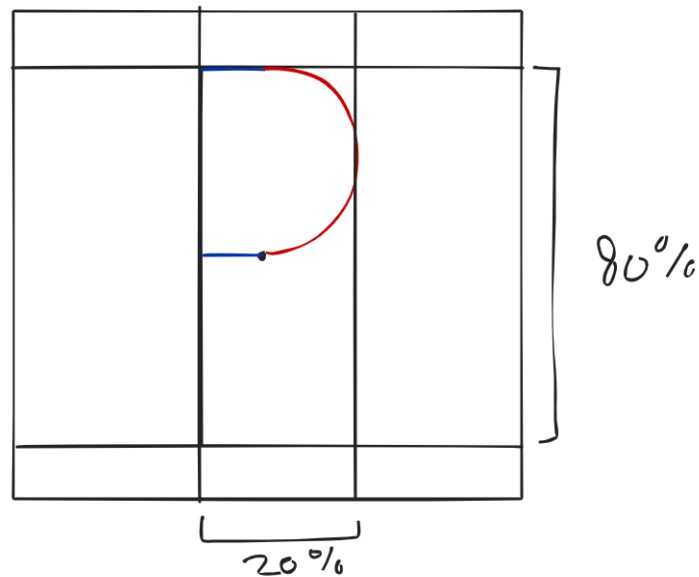
Logo yang saya pilih di sini adalah logo powerpoint yang modern, logonya seperti ini

Lalu, hal pertama yang digambarkan adalah "tubuh" dari logo itu sendiri. Yaitu lingkaran utama yang ada di di logo tersebut. Lingkaran itu sendiri akan digambarkan dengan paramter dimana oktan 5, dan 6 tidak akan digambarkan. Kurang lebih akan terlihat seperti ini,



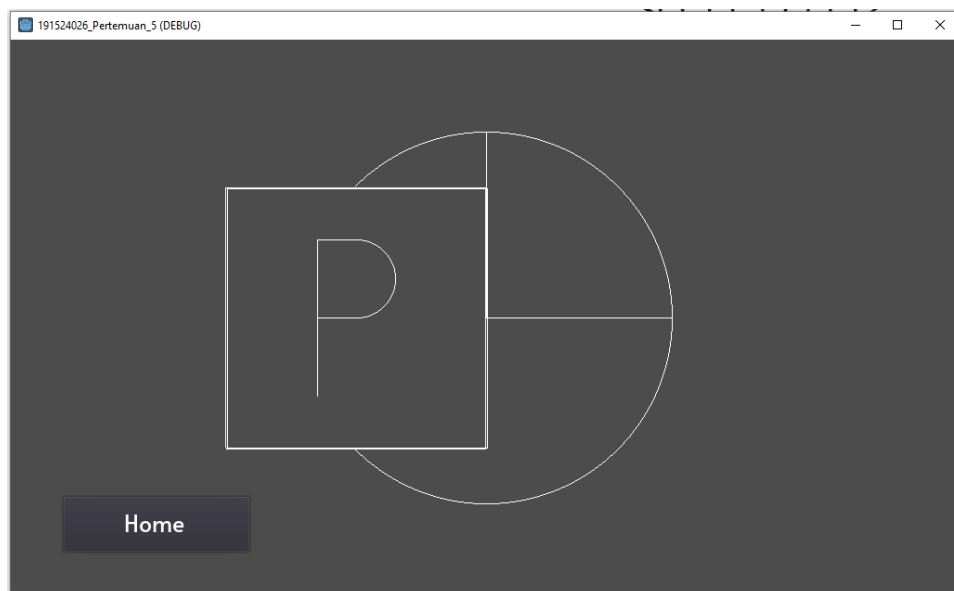
Lalu untuk menggambarkan kotak yang ada di logo, dibutuhkan titik kiri atas dari kotak itu sendiri. Titik itu dapat dicari dengan cara yang sama dengan mencari titik P_{ga} di bagian sebelumnya. Dari situ komponen x nya dapat dikurangi dengan r yaitu radius dari lingkaran. Panjang dari kotaknya sendiri adalah $2r$.

Lalu dalam menggambarkan P di dalam persegi, dibuat batasan dari persegi itu sendiri, di mana P dapat digambarkan.



Panjang dari P itu dibatasi dengan 20% dari panjang persegi, dan tingginya 80%. Lalu, bagian P yang merupakan semicircle sebenarnya terdapat 2 garis (garis biru di gambar) yang maju hingga titik tengah secara horizontal. Dari situ baru ditambahkan semicircle (berwarna merah di gambar).

Hasil akhirnya seperti di bawah ini,



Source Code

```
extends Shapes

const DEG_30 = 0.7
```

```

func _draw():
    var viewport_size = get_viewport().size
    var center_point = Vector2(viewport_size.x / 2, viewport_size.y / 2)

    var radius = 200
    var params = config_draw_only([4, 3, 2, 1, 0, 7])
    draw_circle_midpoint(radius, center_point, params)

    var diff = DEG_30 * radius
    var length = diff * 2
    var top_left_square = Vector2(center_point.x - length, center_point.y - diff)
    draw_square(length, top_left_square, Color.white)

    # boundary_fill(top_left_square, Color.white, Color(198, 66, 32))

    draw_line_custom(center_point, Vector2(center_point.x, center_point.y - radius), Color.white)
    draw_line_custom(center_point, Vector2(center_point.x + radius, center_point.y), Color.white)

    var square_center = Vector2(top_left_square.x + diff, top_left_square.y + diff)
    var p_width = 0.3 * length
    var p_height = p_width * 2

    var p_x = square_center.x - p_width / 2
    var p_y_delta = p_height / 2
    draw_line_custom(
        Vector2(p_x, square_center.y - p_y_delta),
        Vector2(p_x, square_center.y + p_y_delta),
        Color.white
    )

    draw_line_custom(Vector2(p_x, square_center.y), square_center, Color.white)
    draw_line_custom(
        Vector2(p_x, square_center.y - p_y_delta),
        Vector2(square_center.x, square_center.y - p_y_delta),
        Color.white
    )
    var half_circle_params = config_draw_only([3, 2, 1, 0])
    draw_circle_midpoint(
        p_y_delta / 2, Vector2(square_center.x, square_center.y - p_y_delta / 2), half_circle_params
    )

func _on_HomeButton_pressed():
    get_tree().change_scene("res://menu/Menu.tscn")

```

Di sana mungkin terlihat bahwa ada fungsi `boundary_fill` yang di-comment. Fungsi itu eksperimen saya dalam melakukan penggambaran warna yang ada seperti di dalam paint di windows. Source code nya sendiri terlihat seperti ini,

```

func boundary_fill(starting_position: Vector2, boundary_color: Color, fill_color: Color):
    var img = get_viewport().get_texture().get_data()

    var x = starting_position.x
    var y = starting_position.y

    img.lock()
    var curr_pixel_color = img.get_pixel(x, y)
    print("x: ", x, ", y: ", y, ", color: ", curr_pixel_color, ", boundary: ", boundary_color)

    if !curr_pixel_color.is_equal_approx(boundary_color):
        put_pixel(x, y, fill_color)
        print("x: ", x, ", y: ", y, ", color: ", curr_pixel_color, ", boundary: ", boundary_color)

        boundary_fill(Vector2(x + 1, y), boundary_color, fill_color)
        boundary_fill(Vector2(x - 1, y), boundary_color, fill_color)

        boundary_fill(Vector2(x, y + 1), boundary_color, fill_color)
        boundary_fill(Vector2(x, y - 1), boundary_color, fill_color)

```

Fungsi ini akan melakukan pengecekan warna yang ada di pixel sekarang, dengan cara mengambil screenshot dari viewportnya itu, lalu melihat di gambar warna apa yang ada di koordinat itu. Jika tidak sama dengan warna boundary yang ada di paramter, maka akan melanjutkan ke pixel di kiri, kanan, atas, dan bawahnya.

Namun saat dilakukan percobaan, hasilnya malah saja warnannya selalu sama di titik manapun yang di cek. Jadi, tidak berjalan, belum lagi algoritma ini sangat memberatkan programnya.

Lesson Learned

1. Walaupun namanya texture, tapi keluarannya suatu gambar (?)
2. Melakukan pewarnaan dengan flood fill, maupun boundary fill sangatlah lambat dan jika di gunakan cek yang benar, bisa jadi akan terus melakukan "penumpahan" warna.