Nithin S
221IT085

IT351 Lab Assignment 1.1

Code

```
import turtle
import customtkinter as ctk
import random

class ModernSpiralApp:
    def __init__(self):
        ctk.set_appearance_mode("dark")
        ctk.set_default_color_theme("blue")

        self.root = ctk.CTk()
        self.root.title("Spiral Creator")
        self.root.geometry("1200x800")


        self.main_container = ctk.CTkFrame(self.root)
        self.main_container.pack(fill="both", expand=True, padx=20, pady=20)

        self.canvas_frame = ctk.CTkFrame(self.main_container)
        self.canvas_frame.pack(side="left", fill="both", expand=True, padx=(0, 10))
        self.drawing = False
        self.colors = ['#FF0000', '#00FF00', '#0000FF', '#FFA500', '#800080', '#008080',
              '#FF1493', '#FFD700', '#00FFFF', '#FF6347', '#9400D3', '#ADFF2F',
              '#FF00FF', '#FF4500', '#DAA520', '#008B8B']


        self.canvas = turtle.Canvas(self.canvas_frame, width=800, height=800)
        self.canvas.pack(fill="both", expand=True)
        self.root.update_idletasks()

        self.screen = turtle.TurtleScreen(self.canvas)
        self.screen.bgcolor("black")  # Changed to black
        self.original_center_x = self.canvas.winfo_width() // 2 - 150
        self.original_center_y = self.canvas.winfo_height() // 2 - 390
        self.center_x = self.original_center_x
        self.center_y = self.original_center_y
        self.current_scale = 1.0
```

```
        self.t = turtle.RawTurtle(self.screen)


        self.t.speed(0)
        self.t.pensize(2)
        self.t.hideturtle()
        self.create_starfield()
        self.center_turtle()

        self.control_panel = ctk.CTkFrame(self.main_container, width=300)
        self.control_panel.pack(side="right", fill="y", padx=(10, 0))

        self.title_label = ctk.CTkLabel(
            self.control_panel,
            text="Spiral Creator",
            font=ctk.CTkFont(size=28, weight="bold"),
            text_color="#E8E8E8"
        )
        self.title_label.pack(pady=30)

        self.create_buttons()
        self.root.bind("<Configure>", self.on_resize)


    def create_buttons(self):
        # Main button container with more padding and transparent background
        button_frame = ctk.CTkFrame(self.control_panel, fg_color="transparent")
        button_frame.pack(fill="x", padx=30, pady=80)

        # Define modern button styles with wider buttons and transparent background
        primary_button_style = {
            "height": 50,
            "width": 200,
            "corner_radius": 8,
            "font": ctk.CTkFont(size=15, weight="bold"),
            "fg_color": "#ab552f",
            "hover_color": "#964a29",  # Slightly darker shade for hover
            "border_width": 2,
            "border_color": "#ab552f",
            "text_color": "white"
        }

        danger_button_style = {
            "height": 50,
            "width": 200,
            "corner_radius": 8,
            "font": ctk.CTkFont(size=15, weight="bold"),
            "fg_color": "#ab552f",  # Changed to transparent
```

```
    "hover_color": "#964a29",
    "border_width": 2,
    "border_color": "#ab552f",
    "text_color": "#FFFFFF"
}

# Category labels
spiral_label = ctk.CTkLabel(
    button_frame,
    text="SPIRAL CONTROLS",
    font=ctk.CTkFont(size=14, weight="bold"),
    text_color="#FFFFFF"
)
spiral_label.pack(pady=(0, 10))

# Create a container for spiral buttons with transparent background
spiral_buttons_frame = ctk.CTkFrame(button_frame, fg_color="transparent")
spiral_buttons_frame.pack(fill="x", pady=(0, 20))

# Rest of the code remains the same, just added transparent background to all frames
ctk.CTkButton(
    spiral_buttons_frame,
    text="⟳ Spiral Right",
    command=lambda: self.start_drawing(self.create_multi_spiral_clockwise),
    **primary_button_style
).pack(pady=8)

ctk.CTkButton(
    spiral_buttons_frame,
    text="⟲ Spiral Left",
    command=lambda: self.start_drawing(self.create_multi_spiral_counterclockwise),
    **primary_button_style
).pack(pady=8)

zoom_label = ctk.CTkLabel(
    button_frame,
    text="ZOOM CONTROLS",
    font=ctk.CTkFont(size=14, weight="bold"),
    text_color="#FFFFFF"
)
zoom_label.pack(pady=(10, 10))

zoom_buttons_frame = ctk.CTkFrame(button_frame, fg_color="transparent")
zoom_buttons_frame.pack(fill="x", pady=(0, 20))

ctk.CTkButton(
    zoom_buttons_frame,
    text="⊕ Zoom In",
```

```
        command=lambda: self.start_drawing(self.create_zoom_in_spiral),
        **primary_button_style
    ).pack(pady=8)

    ctk.CTkButton(
        zoom_buttons_frame,
        text="⊖ Zoom Out",
        command=lambda: self.start_drawing(self.create_zoom_out_spiral),
        **primary_button_style
    ).pack(pady=8)




    utility_label = ctk.CTkLabel(
        button_frame,
        text="UTILITIES",
        font=ctk.CTkFont(size=14, weight="bold"),
        text_color="#FFFFFF"
    )
    utility_label.pack(pady=(10, 10))

    utility_buttons_frame = ctk.CTkFrame(button_frame, fg_color="transparent")
    utility_buttons_frame.pack(fill="x")

    ctk.CTkButton(
        spiral_buttons_frame,
        text="■ Stop",
        command=self.stop_drawing,
        **danger_button_style
    ).pack(pady=8)

    ctk.CTkButton(
        utility_buttons_frame,
        text="✗ Clear Canvas",
        command=self.clear,
        **danger_button_style
    ).pack(pady=8)

    ctk.CTkButton(
        utility_buttons_frame,
        text="✕ Exit",
        command=self.exit_program,
        **danger_button_style
    ).pack(pady=8)

# Rest of the methods remain the same
def center_turtle(self):
```

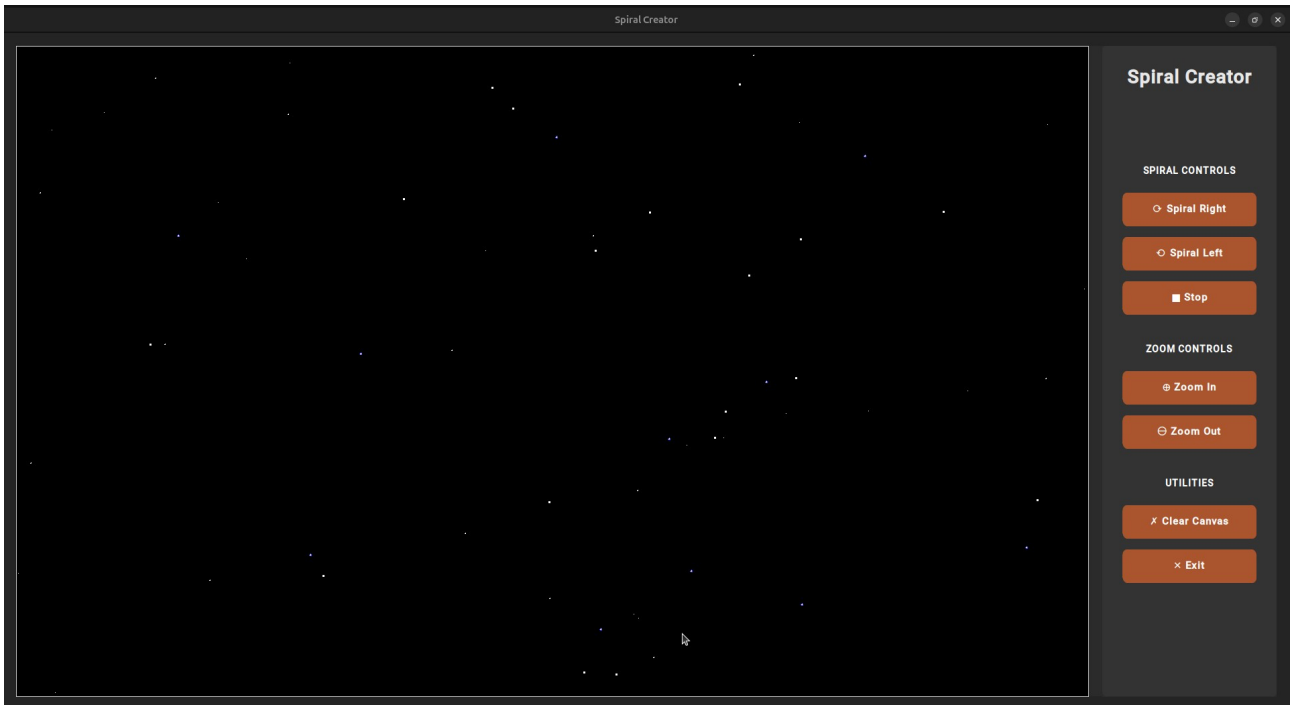
```
    self.t.penup()
    self.t.goto(self.original_center_x, self.original_center_y)
    self.t.setheading(0)
    self.t.pendown()

def stop_drawing(self):
    self.drawing = False

def on_resize(self, event):
    if event.widget == self.root:
        # Update original center coordinates
        self.original_center_x = self.canvas.winfo_width() // 2 - 150
        self.original_center_y = self.canvas.winfo_height() // 2 - 390
        self.center_x = self.original_center_x
        self.center_y = self.original_center_y
        self.center_turtle()



def create_starfield(self):
    """Creates a starfield effect on the canvas instantly"""
    # Turn off animation temporarily
    self.screen.tracer(0)

    original_color = self.t.pencolor()
    original_size = self.t.pensize()

    self.t.penup()
    self.t.color("white")

    # Create basic stars quickly
    for _ in range(400):
        x = random.randint(-2000, 2000)
        y = random.randint(-2000, 2000)
        self.t.goto(x, y)
        self.t.dot(random.randint(1, 3))  # Random size between 1-3

    # Add a few brighter stars
    for _ in range(25):
        x = random.randint(-800, 1200)
        y = random.randint(-800, 800)
        self.t.goto(x, y)
        self.t.color("#4444FF")
        self.t.dot(4)
        self.t.color("white")
        self.t.dot(2)

    # Restore original state
```

```
        self.t.pensize(original_size)
        self.t.color(original_color)
        self.t.goto(0, 0)
        self.t.pendown()

        # Turn animation back on and update
        self.screen.tracer(1)
        self.screen.update()

    def on_resize(self, event):
        self.center_turtle()

    def start_drawing(self, func):
        self.drawing = True
        func()
        self.drawing = False

    def create_multi_spiral_clockwise(self):
        # Ensure we start from center
        self.center_turtle()
        self.t.pensize(2)
        # Reset the canvas scaling first
        self.canvas.scale("all", self.center_x, self.center_y, 1.0, 1.0)
        self.canvas.configure(scrollregion=self.canvas.bbox("all"))

        angle = 89  # Changed angle for a different pattern
        step_increment = 5  # Smaller step increment for a tighter spiral

        for x in range(512):
            if not self.drawing:
                break
            self.t.color(self.colors[x % len(self.colors)])
            self.t.forward(10 + step_increment * x)  # Start with a smaller step and increase
gradually
            self.t.right(angle)
            self.root.update()

    def create_multi_spiral_counterclockwise(self):
        # Ensure we start from center
        self.center_turtle()
        self.t.pensize(2)
        # Reset the canvas scaling first
        self.canvas.scale("all", self.center_x, self.center_y, 1.0, 1.0)
        self.canvas.configure(scrollregion=self.canvas.bbox("all"))

        angle = 89  # Changed angle for a different pattern
        step_increment = 5  # Smaller step increment for a tighter spiral
```

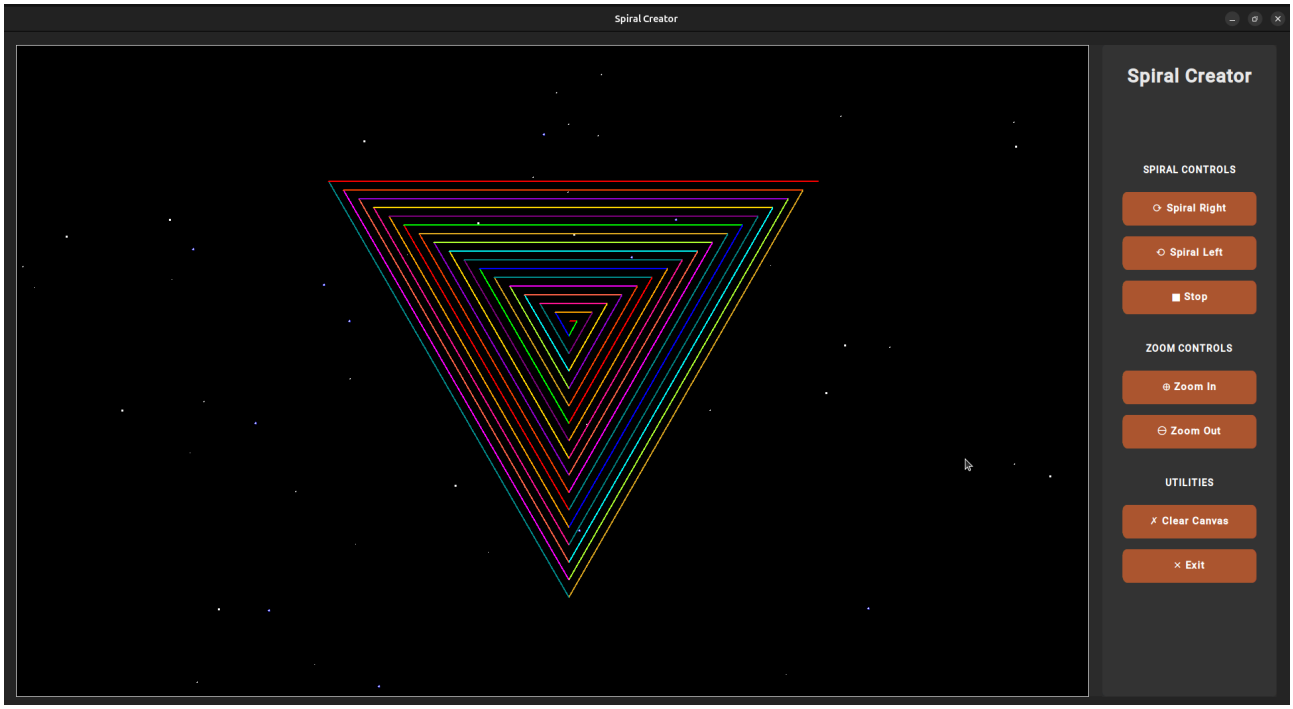
```
        for x in range(512):
            if not self.drawing:
                break
            self.t.color(self.colors[x % len(self.colors)])
            self.t.forward(10 + step_increment * x)  # Start with a smaller step and increase
gradually
            self.t.left(angle)
            self.root.update()

    def create_zoom_in_spiral(self):
        """Zoom in while keeping the spiral centered."""
        self.zoom(1.2)

    def create_zoom_out_spiral(self):
        """Zoom out while keeping the spiral centered."""
        self.zoom(0.8)

    def zoom(self, scale_factor):
        """Zoom function that maintains the custom center point"""
        # Scale around the custom center point
        self.current_scale *= scale_factor

        # Scale around the original center point
        self.canvas.scale("all", self.original_center_x, self.original_center_y, scale_factor,
scale_factor)

        # Update current center coordinates based on scale
        self.center_x = self.original_center_x
        self.center_y = self.original_center_y

        # Update scroll region to ensure full content visibility
        bbox = self.canvas.bbox("all")
        if bbox:
            padding = 50
            self.canvas.configure(scrollregion=(
                bbox[0] - padding,
                bbox[1] - padding,
                bbox[2] + padding,
                bbox[3] + padding
            ))

        # Maintain turtle position relative to center
        self.t.penup()
        current_x, current_y = self.t.position()
        scaled_x = (current_x - self.original_center_x) * scale_factor + self.original_center_x
        scaled_y = (current_y - self.original_center_y) * scale_factor + self.original_center_y
        self.t.goto(scaled_x, scaled_y)
        self.t.pendown()
```

```
    def adjust_scroll_region(self):
        """Adjusts the canvas scroll region after zooming to keep everything visible"""
        self.canvas.configure(scrollregion=self.canvas.bbox("all"))

    def clear(self):
        self.drawing = False
        self.t.penup()
        self.t.clear()

        # Reset scale to original
        if self.current_scale != 1.0:
            self.canvas.scale("all", self.original_center_x, self.original_center_y,
1/self.current_scale, 1/self.current_scale)
            self.current_scale = 1.0

        # Reset center coordinates to original values
        self.center_x = self.original_center_x
        self.center_y = self.original_center_y

        # Create new starfield and center turtle
        self.create_starfield()
        self.center_turtle()
        self.t.pensize(2)
        self.t.pendown()

    def exit_program(self):
        self.root.destroy()

    def run(self):
        self.root.mainloop()

if __name__ == "__main__":
    app = ModernSpiralApp()
    app.run()
```

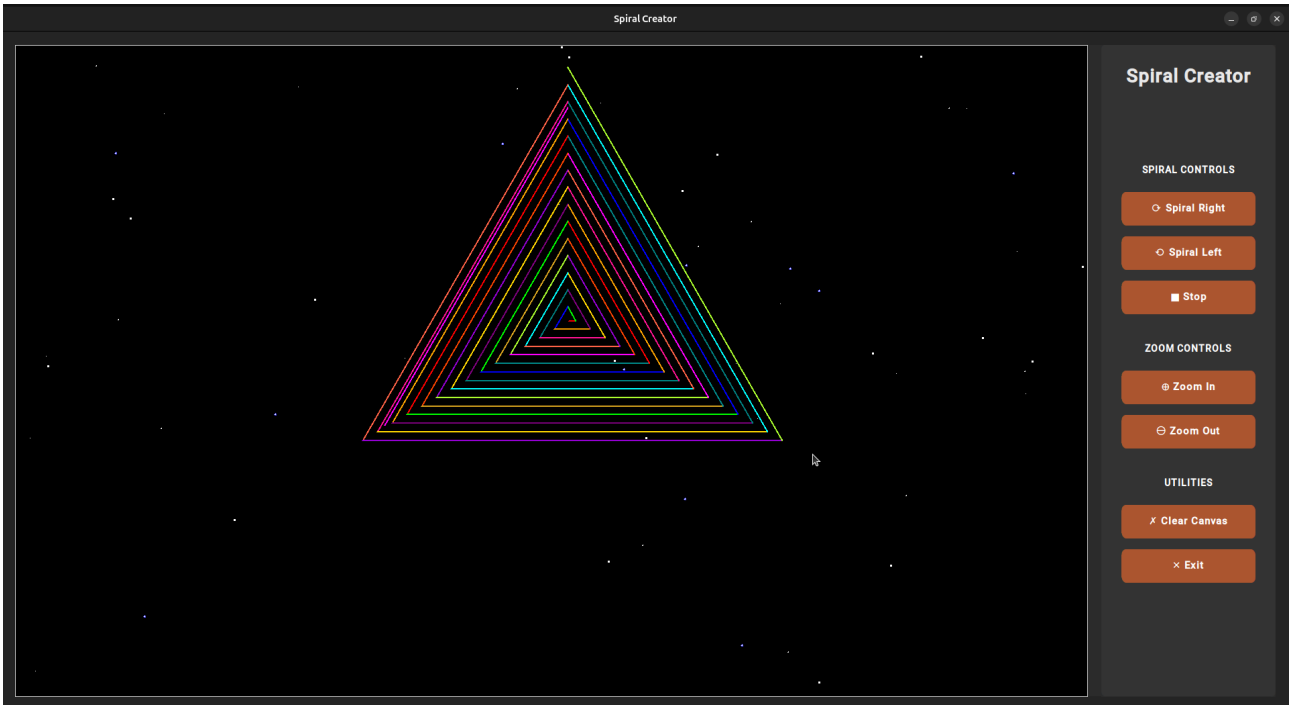
OUTPUT



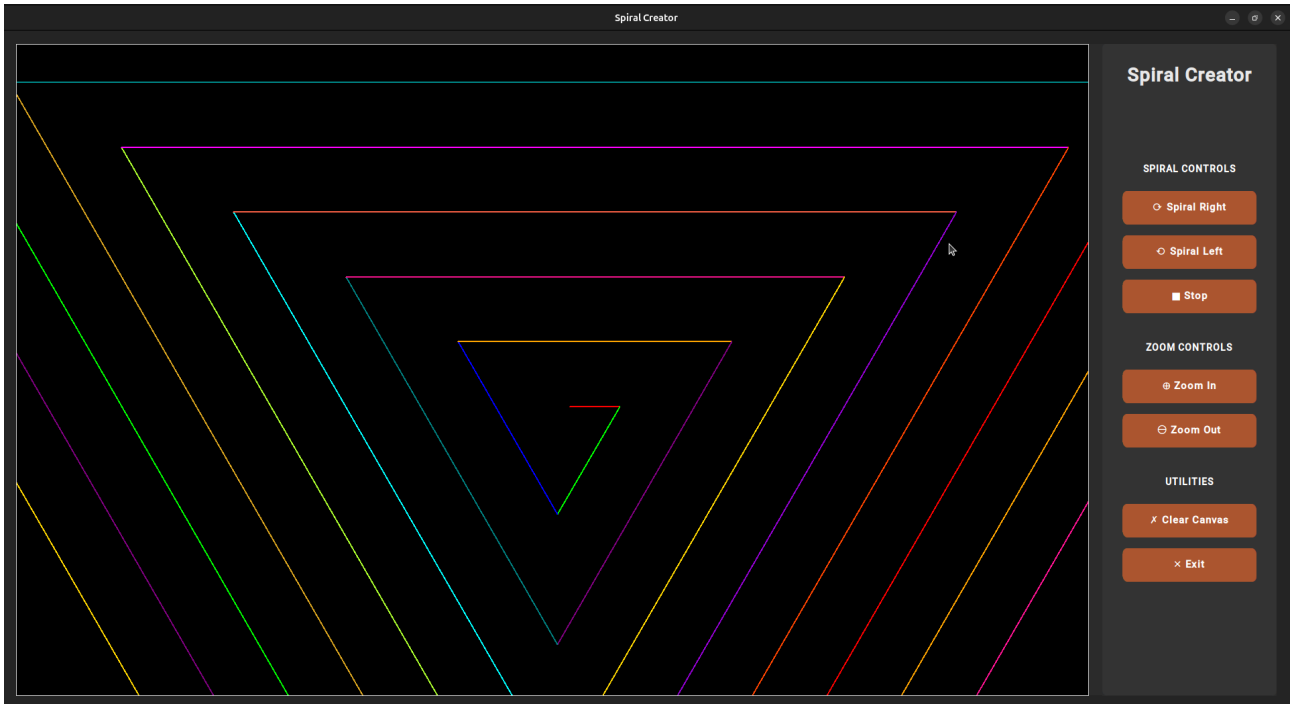
1) Spiral Right



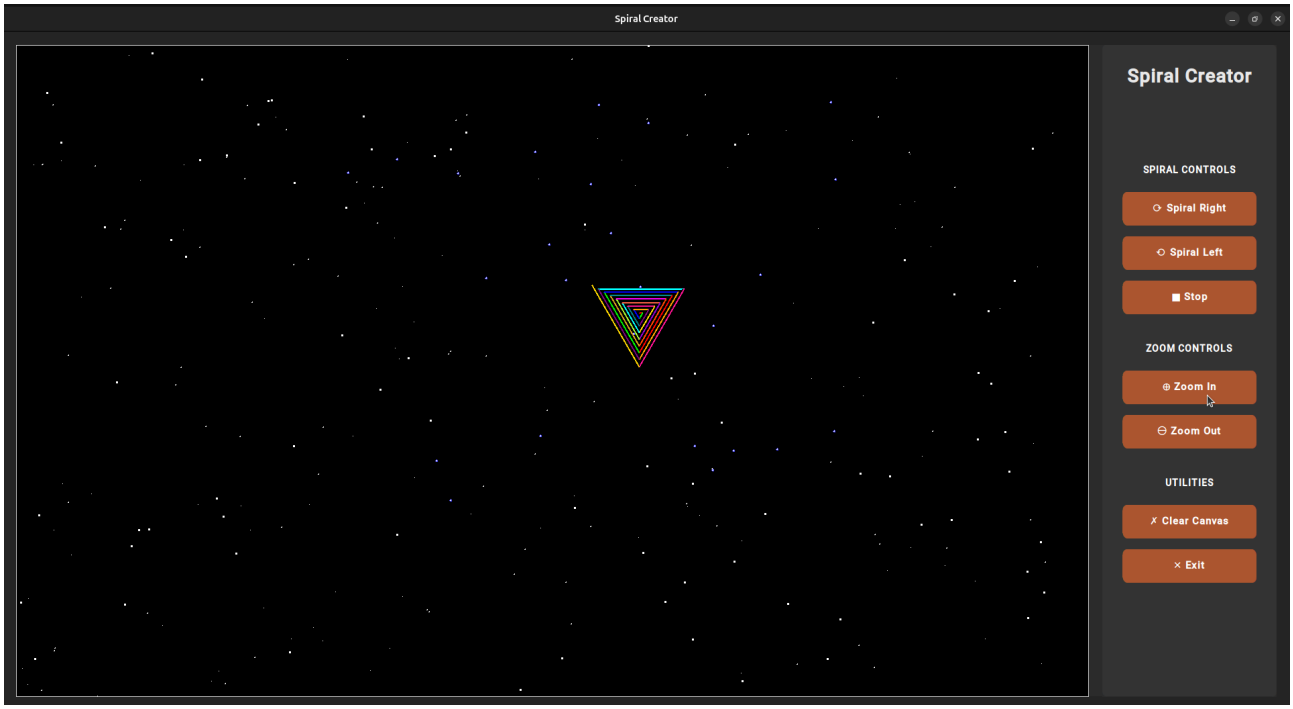
2) Spiral Left



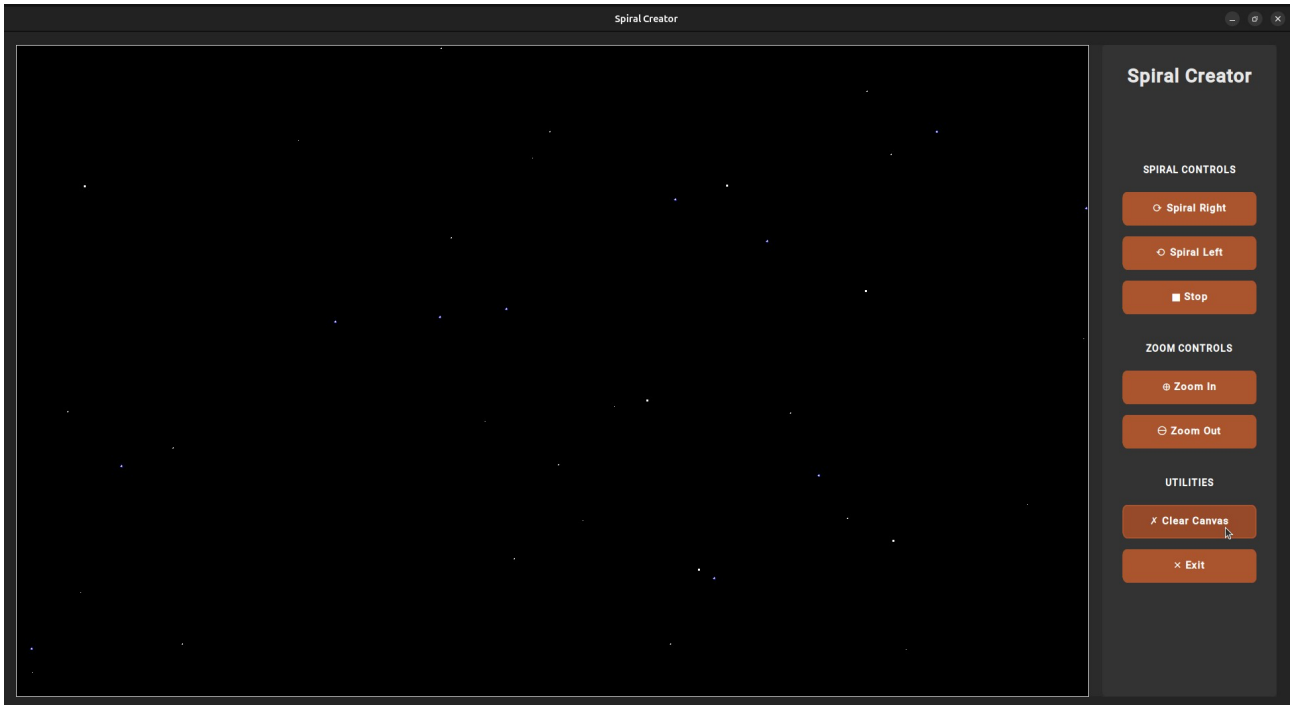
3) Zoom In



4) Zoom Out



5) Clear



Answer the following questions

i. Whether interactive design was comfortable?

Yes, the design is comfortable for users, featuring simple and intuitive methods with minimal complexity and clear contrast.

The menu was easy to navigate using mouse clicks, with clearly labeled options.

A graphical user interface (GUI) improved usability compared to a command-line approach.

Buttons were grouped into categories: Spiral Controls, Zoom Controls, and Utilities, making them easier to use.

Emojis were added next to buttons to visually represent their functions. The design is intuitive, featuring colorful spirals for an engaging experience.

The width of spiral colors changes when zooming in and out but remains constant when spiraling left or right, improving representation.

A hover effect was added to buttons for better visibility.

Clear contrast enhances readability.

Minimal buttons ensure an intuitive UI.

Stars were added to the background for an aesthetically pleasing design.

Stop button to stop the spiral at any size.

ii. What challenges you faced when selecting menu?

Ensuring menu items were easily clickable without overlapping the drawing area.

Making sure the turtle responds correctly to the selected button.

Arranging the buttons in an intuitive layout.

Selecting appropriate contrast for buttons and the canvas.

Making the spirals colorful while maintaining clear distinctions between options.

Differentiating between zoom and spiral functions, particularly through varying spiral widths.

iii. What challenges you faced when selecting next choice in menu?

Maintaining continuity of the spiral drawing when switching options; for example, ensuring zooming scales the spiral correctly without redrawing it from scratch.

Handling rapid menu selections without lag or unexpected behavior.
Ensuring smooth transitions between different drawing modes.

Preventing selected options from interfering with previous drawings.

Indicating which menu option is currently active.

iv. Whether the design was continuous, means the menus can be selected repeatedly until you press Exit.

Yes, the design supports continuous interaction, allowing users to select actions repeatedly until they choose to exit.

The canvas resets with new inputs and options until the exit button is pressed.

v. In what way you may improve the design of menu selection

Adding a visual cue (such as highlighting the active button or changing its color) to indicate the current selection.

Implementing an undo button or action history to allow users to reverse their previous selections.

Adding tooltips or short descriptions that appear when hovering over buttons to help users understand each option before selecting it.

Improving menu organization by introducing collapsible sections or subcategories to reduce visual clutter while maintaining ease of access.