DATA VISUALIZATION: PIE CHARTS AND OPENGL

Programming assignment on fruit preference survey

TEAM

Ian Ndolo Mwau	SCT 211-0034/2022
Maureen Mukami	SCT 211-0052/2022
David Nzambuli	SCT 211-00682022
Pharis Kariuki	SCT 211-0033/2022

INTRODUCTION

- Objective:
  - Create a pie chart using OpenGL to visualize fruit preference data.
  - Demonstrate customization by applying fruit-resembling colors for each section.
  - Demonstrate grayscale conversion using OpenGL

PROBLEM 1 – PIE CHART FOR FRUIT PREFERENCES

Data Table:

Fruit	People
Orange	36
Banana	41
Kiwi Fruit	19
Mango	28
Grapes	30
Ovacado	16

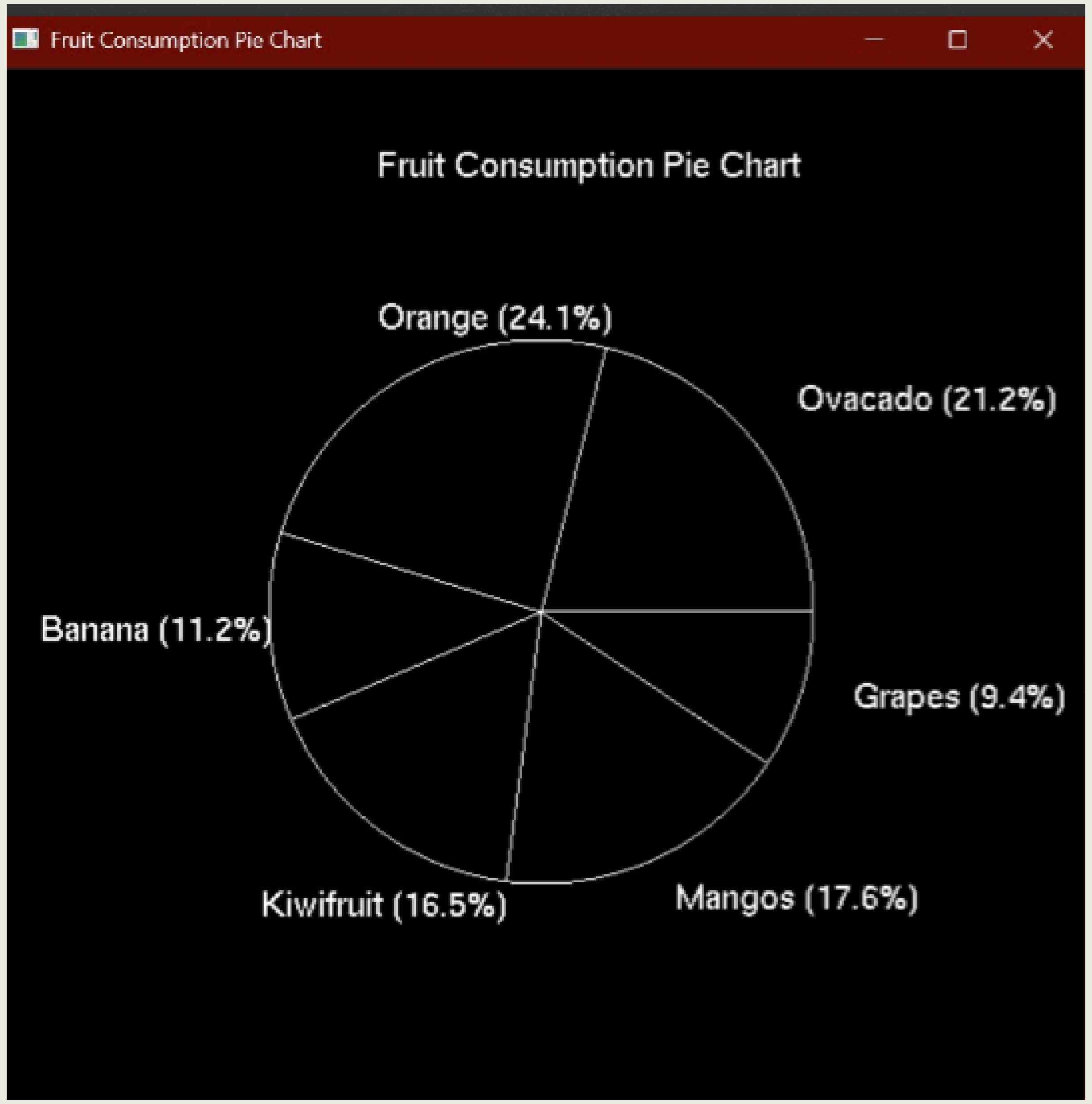
PROBLEM1:FRUIT PREFERENCES PIE CHART

Objective

- Draw a piechart with title "Fruit Preferences Survey"
- Calculate percentages from the data.
- Place labels outside each section.

FRUIT PIE CHART

Output



FRUIT PREFERENCES PIE CHART

Code Walkthrough

OVERVIEW & KEY COMPONENTS

Pie Chart Visualization in OpenGL

What the Code Does:

- Draws a black-and-white pie chart from fruit consumption data.
- Labels each slice with name + percentage outside the chart.

Key Components

1. Data Setup

```
float values[] = {36, 41, 19, 28, 30, 16}; // Percentagesconst char* labels[] =
{"Ovacado", "Orange", ...};
```

Pie Chart Visualization in OpenGL

2. OpenGL Functions
   - GL_TRIANGLE_FAN: Draws pie slices.
   - GL_LINES: White slice boundaries.
   - drawBitmapText(): Renders labels.
3. Smart Adjustments
   - Banana/Kiwifruit labels are repositioned to avoid clutter.

IMPLEMENTATION

1. Slice Calculation
    - ○ float sliceAngle = (values[i] / total) * 360.0f; // Convert % to angle

- Uses trigonometry (sin/cos) to plot points.

2. Label Positioning

    float midAngle = currentAngle + sliceAngle / 2.0f;

3. Rendering Pipeline

- display() → reshape() → main().
- Output: Clean pie chart with dynamic resizing.

C++ VS. PYTHON COMPARISON

1. Text Rendering

C++

```
// C++ (GLUT)  glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,
*c);
```

Python

```
# Python (PyOpenGL)
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, ord(ch))
```

2. Main Loop Structure

C++

```
int  main()  {glutInit(&argc,  argv);glutCreateWindow("Pie
Chart");glutDisplayFunc(display);  // Callback registration
}
```

Python

```
def main():
    glutInit(sys.argv)
    glutCreateWindow(b"Pie Chart")
    glutDisplayFunc(display)  # Same callback

if __name__ == "__main__":
    main()
```

3. Label Repositioning Logic

C++

```
// Adjust Banana label outward (extra space)if
(std::string(labels[i]) == "Banana") {
     labelRadius = radius + 0.42f;  // Push further out}//
Adjust Kiwifruit label leftwardif (std::string(labels[i]) ==
"Kiwifruit") {
    labelRadius = radius + 0.20f;  // Slightly out
    labelX -= 0.09f;              // Shift left}
```

Python

```
# Banana adjustmentif labels[i] == "Banana":
       labelRadius = radius + 0.42   # Push further out#
Kiwifruit adjustmentif labels[i] == "Kiwifruit":
    labelRadius = radius + 0.20  # Slightly out
    labelX -= 0.09               # Shift left
```

Key Takeaways

Same Output: Both versions produce identical visuals.

Syntax Differences:

- C++: Explicit types, manual string formatting.
- Python: Dynamic typing, f-strings.

Workflow: Python is quicker for prototyping; C++ offers more control.

PROBLEM 2 – FRUIT COLORED PIE CHART

Objective

- Redraw the pie chart so that each section corresponds to the color of the fruit represented when ripest.

FRUIT-COLORED PIE CHART

Color Selection Explanation

- Avocado: The greenish color (0.82, 0.81, 0.41) represents the distinctive flesh color of a ripe avocado.
- Orange: A vibrant orange shade (0.93, 0.55, 0.14) that mimics the citrus fruit's characteristic color.
- Banana: A bright yellow (1.0, 0.87, 0.35) selected to match a perfectly ripe banana skin
- Kiwifruit: Deeper green (0.43, 0.51, 0.04) representing the kiwi's flesh once cut open
- Mango: Orange-red shade (1.0, 0.51, 0.26) capturing the warm tones of a ripe mango
- Grapes: Rich purple (0.44, 0.18, 0.66) mimicking the deep color of dark grapes

FRUIT COLORED PIE CHART

Code Walkthrough

FRUIT-COLORED PIE CHART

Data Structure Setup

C++

```
// Data for the pie chart
static float values[] = {36.0f, 41.0f, 19.0f, 28.0f, 30.0f,
16.0f};
static const char *labels[] = {"Ovacado", "Orange",
"Banana", "Kiwifruit", "Mangos", "Grapes"};
static const int NUM_SLICES = sizeof(values) /
sizeof(values[0]);
```

Python

```
# Survey data
fruits = ['Avocado', 'Orange', 'Banana', 'Kiwifruit',
'Mango', 'Grapes']
people = [36, 41, 19, 28, 30, 16]
total = sum(people)
percentages = [round((count/total)*100, 1) for count in
people]
```

Color Array Definition

C++

```
// Fruit colors (R, G, B)
static float colors[][3] = { {0.34f, 0.51f, 0.01f}, //
Avocado
{1.0f, 0.65f, 0.0f}, // Orange
{1.0f, 0.87f, 0.35f}, // Banana
{0.65f, 0.89f, 0.18f}, // Kiwifruit
{1.0f, 0.51f, 0.26f}, // Mango
{0.44f, 0.18f, 0.66f} // Grapes;
```

Python

```
# Fruit colors (R, G, B)
fruit_colors = [
    (0.82, 0.81, 0.41),   # Avocado
    (0.93, 0.55, 0.14),   # Orange
    (1.0, 0.87, 0.35),    # Banana
    (0.43, 0.51, 0.04),   # Kiwifruit
    (1.0, 0.51, 0.26),    # Mango
    (0.44, 0.18, 0.66)    # Grapes
]
```

FRUIT-COLORED PIE CHART

Main Drawing Function

C++

```cpp
void display() {
  // Calculate percentages
  float total = 0.0f;
  for (int i = 0; i < NUM_SLICES; ++i) {
    total += values[I];
  }
  // Draw pie slices with colors
  for (int i = 0; i < NUM_SLICES; ++i){
    float sliceAngle = 360.0f * values[i] / total;
    // Draw filled slices using fruit colors
    glColor3f(colors[i][0], colors[i][1], colors[i][2]);
    glBegin(GL_TRIANGLE_FAN);
    // [Drawing code...]
    glEnd();
  }
}
```

Python

```python
def draw_pie_chart():
    # Draw pie slices
    for i in range(len(people)):
        angle_end = angle_start + 2 * math.pi * people[i] / total

        # Draw a pie slice with fruit colors
        glBegin(GL_TRIANGLE_FAN)
        glColor3f(*fruit_colors[i])
        glVertex2f(0, 0)  # Center
        # [Drawing code...]
        glEnd()
```

FRUIT-COLORED PIE CHART

Label Positioning Logic

C++

```cpp
// Calculate label position
float midAngle = currentAngle + sliceAngle / 2.0f;
float midRad = midAngle * PI / 180.0f;
float labelRadius = radius + 0.15f;

// Adjust specific label positions if needed
if (i == 2) labelRadius += 0.1f;  // Banana
if (i == 3) labelRadius += 0.05f; // Kiwi

float labelX = centerX + cos(midRad) * labelRadius;
float labelY = centerY + sin(midRad) * labelRadius;
```

Python

```python
# Calculate label position (on circumference)
mid_angle = (angle_start + angle_end) / 2
label_x = radius * math.cos(mid_angle)
label_y = radius * math.sin(mid_angle)

# Small offset to nudge labels outward
offset = 0.12
label_x += offset * math.cos(mid_angle)
label_y += offset * math.sin(mid_angle)
```

FRUIT COLORED PIE CHART

C++ vs Python Implementation

FRUIT COLORED PIE CHART

Similarities

- Both use OpenGL for rendering graphics
- Same core algorithm: calculate percentages, draw slices using triangle fans
- Similar color values for fruit-resembling colors
- Both position labels at midpoint of each slice angle

FRUIT COLORED PIE CHART

Differences

- **Syntax Style:**

C++ uses static arrays and explicit memory management

Python offers more concise data structures

- **Percentage Calculation:**

C++: Manual loop summing values

Python: Elegant sum(people) and list comprehension

- **Label Positioning:**

C++: Custom adjustments for specific slices (if (i == 2) labelRadius += 0.1f;)

Python: Consistent offset approach for all labels

- **Color Application:**

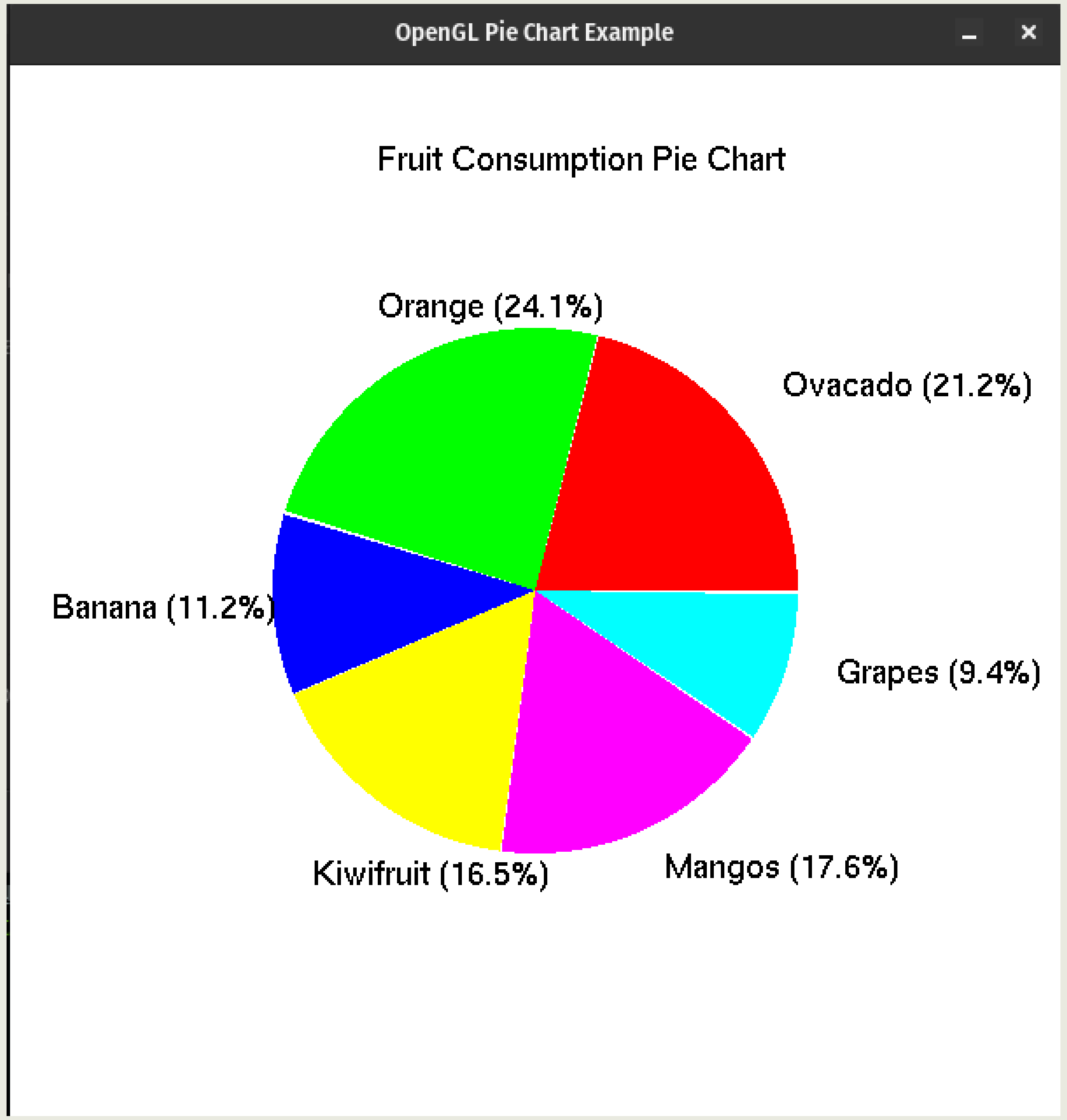
C++: glColor3f(colors[i][0], colors[i][1], colors[i][2])

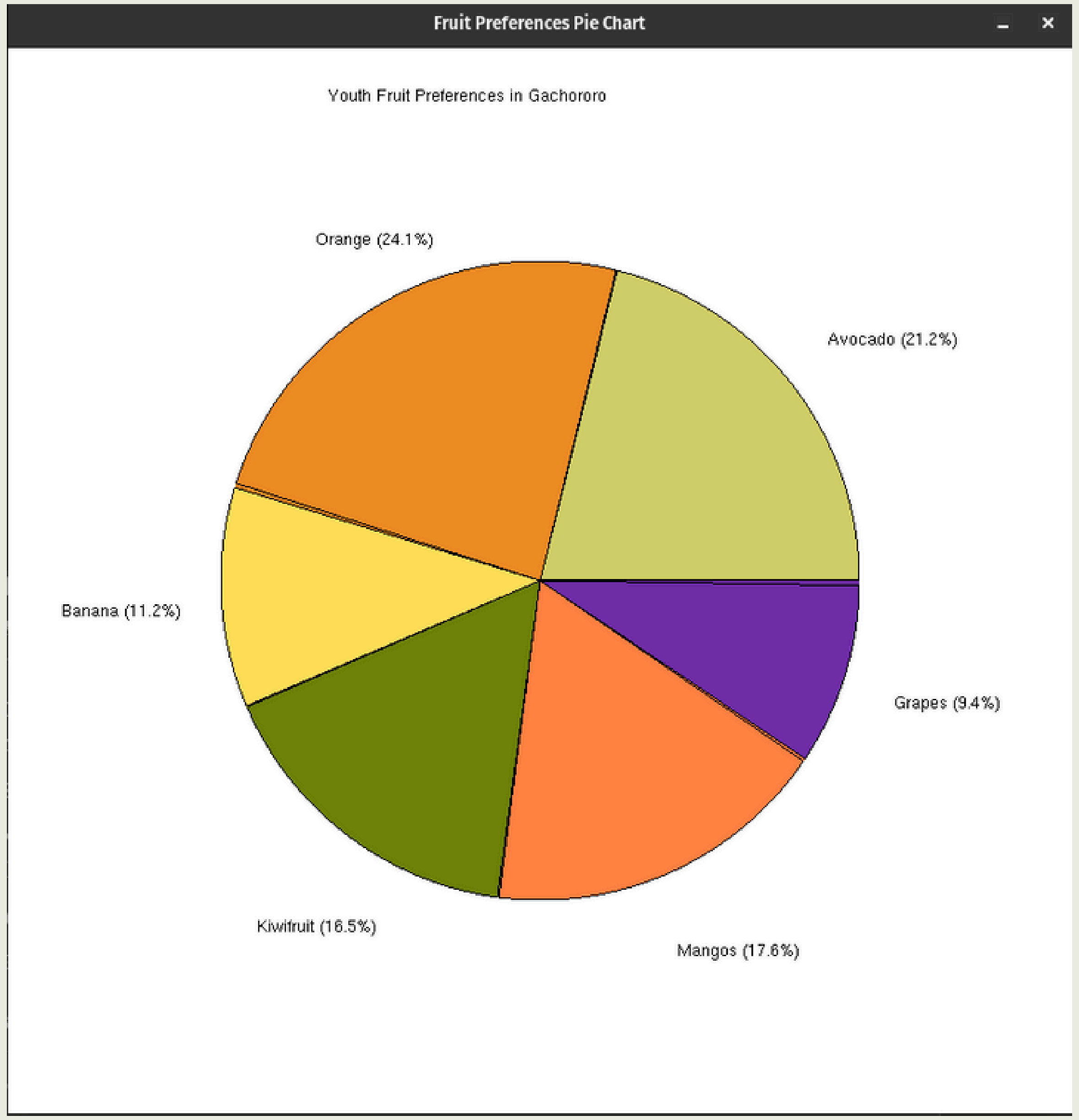
Python: Cleaner glColor3f(*fruit_colors[I])

FRUIT COLORED PIE CHART

Results & Comparison

FRUIT-COLORED PIE CHART





FRUIT-COLORED PIE CHART

Visual Changes and Additions

- Fruit-resembling colors create an intuitive association with data.
- Maintained sufficient contrast between adjacent sections.
- Adjusted label placement to prevent overlapping and improve readability.
- Added slice borders in the C++ version for better visual separation

FRUIT-COLORED PIE CHART

Challenges

Label Positioning Challenges

- Labels for adjacent small slices (Banana and Kiwifruit) were overlapping.

Solution

- Implemented custom offset adjustments for specific slices.

Color Selection Challenges

- Finding precise RGB values that closely match each fruit was difficult.
- "Some fruits (like avocado) have different colors inside and out.

Solution

- Researched RGB values representing each fruit's most recognizable color.
- Tested multiple color options to ensure sufficient contrast between adjacent sections.

PROBLEM 3 – GRAYSCALE BACKGROUND IN OPENGL

- **Objective:**

  Convert chart background to grayscale using OpenGL.

- **Approach:**

- Use OpenGL’s glColor3f to set grayscale values (R=G=B).

- Render the pie chart on a grayscale background.

Code Walk Through

1. Grayscale Colors Definition

C++

```
// Grayscale colors using luminance formula: 0.299*R +
0.587*G + 0.114*B
static float grayscaleColors[][3] = {
    {0.40f, 0.40f, 0.40f},  // Avocado
    {0.59f, 0.59f, 0.59f},  // Orange
    {0.89f, 0.89f, 0.89f},  // Banana
    {0.61f, 0.61f, 0.61f},  // Kiwifruit
    {0.77f, 0.77f, 0.77f},  // Mangos
    {0.21f, 0.21f, 0.21f}   // Grapes
};
```

Python

```
# Grayscale colors (R=G=B) using luminance-preserving
conversion
grayscale_colors = [
    [0.40, 0.40, 0.40],  # Avocado
    [0.59, 0.59, 0.59],  # Orange
    # ... other colors
]
```

2. Slice Coloring

C++

```
// Set grayscale color for this slice
glColor3fv(grayscaleColors[i]);  // Array pointeres
};
```

Python

```
# Set grayscale color
glColor3f(*grayscale_colors[i])  # List unpacking
```

Code Walk Through

3. Background Color Change

C++

```
// Set gray background
glClearColor(0.5f, 0.5f, 0.5f, 1.0f);
```

Python

```
# Set gray background (in main() or display())
glClearColor(0.5, 0.5, 0.5, 1.0)
```

Takeaways:

Same Visual Output:

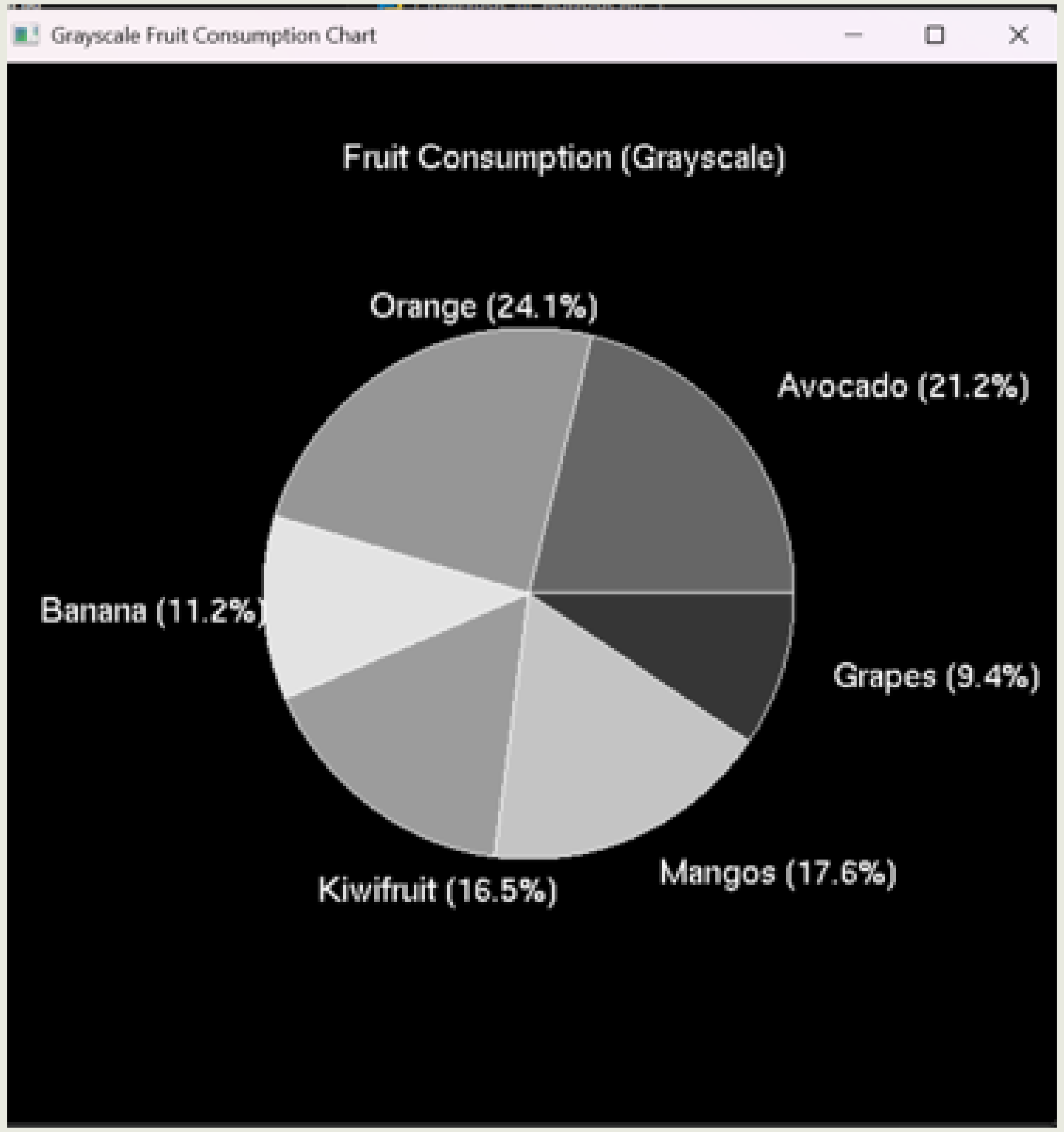
- Both versions produce identical grayscale rendering:
- Avocado = 0.4, Banana = 0.89, etc.

Maintenance:

- ○ Python version is easier to modify (e.g., adding new fruits)
- ○ C++ version offers better performance for complex visualizations

OUTPUT

Screenshot:OpenGL window with grayscale background and colored pie chart.



CONCLUSION

Summary of tasks achieved:

- Pie chart representation of fruit preferences with percentages and external labels.
- Color customization for realism.
- OpenGL grayscale implementation.

Q&A

Thank you.