**Description:**

This code is used to draw a cone-shaped ice cream with three scoops and a cherry on top using OpenGL. The key components of the code include:

**Ice Cream Cone (Custom Vertices):** The code creates the cone shape using vertices, and the vertices are defined manually to form the triangular shape of the cone, which is then filled with a brown color to represent the cone.

**First Scoop (Custom Vertices):** For the first scoop, which is the bottom-most, the code defines a set of vertices to form a shape, and that shape is filled with a light yellow color to simulate a vanilla-flavored ice cream scoop. The scoop is positioned above the cone.

**Second Scoop (Custom Vertices):** The second scoop, placed above the first, is similarly represented by a set of vertices. The code colors this scoop light pink to simulate a strawberry-flavored scoop. It is positioned above the first scoop and slightly overlaps the cone.

**Third Scoop (Custom Vertices):** The third scoop, placed above the second, is also represented by a set of vertices and is colored light green to represent a mint-flavored scoop. Like the second scoop, it overlaps the one below it.

**Cherry (Small Custom Vertices):** A small circle is created using vertices to form a compact circular shape, positioned above the third scoop to represent a cherry on top. It is colored red, typical for a cherry.

**Positioning:** The scoops and cherry are positioned using transformations (like translation) to stack them properly above the cone. Each scoop is drawn above the previous one, and the cherry is placed at the top.

**Colors:** The code uses OpenGL to specify different colors for each part of the ice cream—brown for the cone, light yellow for the first scoop, light pink for the second, light green for the third, and red for the cherry.

**Program and Results:**

// Fragment shader

const char\* fragmentShaderSource = "#version 330 core\n"

"out vec4 FragColor;\n"

"uniform vec4 triangleColor;\n" // Color uniform for each triangle fan

"void main()\n"

"{\n"

" FragColor = triangleColor;\n"

"}\n\0";

// Vertices

float vertices[] = {

… // For scoops and cherry

}

float vertices2[] = {

… // For cone

}

// Colors

glm::vec4 color1 = glm::vec4(1.0f, 0.0f, 0.0f, 1.0f); // Red

glm::vec4 color2 = glm::vec4(0.564f, 0.933f, 0.564f, 1.0f); // Light Green

…

// Draw scoops and cherry

glBindVertexArray(VAO1);

glDrawArrays(GL\_LINE\_STRIP, 0, 13); // Cherry

glDrawArrays(GL\_LINE\_STRIP, 14, 20); // Cherry

glDrawArrays(GL\_LINE\_STRIP, 34, 53); // 3rd Scoop

glDrawArrays(GL\_LINE\_STRIP, 87, 34); // 3rd Scoop

glDrawArrays(GL\_LINE\_STRIP, 121, 46); // 2nd Scoop

glDrawArrays(GL\_LINE\_STRIP, 167, 33); // 2nd Scoop

glDrawArrays(GL\_LINE\_STRIP, 200, 42); // 1st Scoop

glDrawArrays(GL\_LINE\_STRIP, 242, 36); // 1st Scoop

// Color Scoops and cherry

unsigned int colorLocation = glGetUniformLocation(shaderProgram, "triangleColor");

glUniform4fv(colorLocation, 1, glm::value\_ptr(color1));

glDrawArrays(GL\_TRIANGLE\_FAN, 14, 20); // Cherry: Red

glUniform4fv(colorLocation, 1, glm::value\_ptr(color2));

glDrawArrays(GL\_TRIANGLE\_FAN, 34, 53); // 3rd Scoop (Mint: Light green)

glUniform4fv(colorLocation, 1, glm::value\_ptr(color3));

glDrawArrays(GL\_TRIANGLE\_FAN, 87, 34); // 3rd Scoop (Mint: Light green)

glUniform4fv(colorLocation, 1, glm::value\_ptr(color4));

glDrawArrays(GL\_TRIANGLE\_FAN, 121, 46); // 2nd Scoop (Strawberry: Light pink)

glUniform4fv(colorLocation, 1, glm::value\_ptr(color5));

glDrawArrays(GL\_TRIANGLE\_FAN, 167, 33); // 2nd Scoop (Strawberry: Light pink)

glUniform4fv(colorLocation, 1, glm::value\_ptr(color6));

glDrawArrays(GL\_TRIANGLE\_FAN, 200, 42); // 1st Scoop (Vanilla: Light yellow)

glUniform4fv(colorLocation, 1, glm::value\_ptr(color7));

glDrawArrays(GL\_TRIANGLE\_FAN, 242, 36); // 1st Scoop (Vanilla: Light yellow)

// Draw and color cone

glBindVertexArray(VAO2);

glDrawArrays(GL\_LINE\_STRIP, 0, 17);

glUniform4fv(colorLocation, 1, glm::value\_ptr(color8));

glDrawArrays(GL\_TRIANGLE\_FAN, 0, 17);

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Fig 1: Cone ice cream | | |

// Scoops and cherry transformations

if (glfwGetKey(window, GLFW\_KEY\_R) == GLFW\_PRESS) rotateAngle += 0.05f; // Rotation

if (glfwGetKey(window, GLFW\_KEY\_UP) == GLFW\_PRESS) translate\_Y += 0.001f; // Translation (Up)

if (glfwGetKey(window, GLFW\_KEY\_DOWN) == GLFW\_PRESS) translate\_Y -= 0.001f; // Translation (Down)

if (glfwGetKey(window, GLFW\_KEY\_LEFT) == GLFW\_PRESS) translate\_X -= 0.001f; // Translation (Left)

if (glfwGetKey(window, GLFW\_KEY\_RIGHT) == GLFW\_PRESS) translate\_X += 0.001f; // Translation (Right)

if (glfwGetKey(window, GLFW\_KEY\_EQUAL) == GLFW\_PRESS) { scale\_X += 0.001f; scale\_Y += 0.001f; } // Increase scaling

if (glfwGetKey(window, GLFW\_KEY\_MINUS) == GLFW\_PRESS) { scale\_X -= 0.001f; scale\_Y -= 0.001f; } // Decrease scaling

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Fig 2: Translation, rotation and scaling of ice cream’s scoops and cherry | | |

// Cone transformations

if (glfwGetKey(window, GLFW\_KEY\_T) == GLFW\_PRESS)

rotateAngle2 += 0.05f; // Rotation

if (glfwGetKey(window, GLFW\_KEY\_KP\_8) == GLFW\_PRESS) translate\_Y2 += 0.001f; // Translation (Up)

if (glfwGetKey(window, GLFW\_KEY\_KP\_2) == GLFW\_PRESS) translate\_Y2 -= 0.001f; // Translation (Down)

if (glfwGetKey(window, GLFW\_KEY\_KP\_4) == GLFW\_PRESS) translate\_X2 -= 0.001f; // Translation (Left)

if (glfwGetKey(window, GLFW\_KEY\_KP\_6) == GLFW\_PRESS) translate\_X2 += 0.001f; // Translation (Right)

if (glfwGetKey(window, GLFW\_KEY\_KP\_ADD) == GLFW\_PRESS) { scale\_X2 += 0.001f; scale\_Y2 += 0.001f; } // Increase scaling

if (glfwGetKey(window, GLFW\_KEY\_KP\_SUBTRACT) == GLFW\_PRESS) { scale\_X2 -= 0.001f; scale\_Y2 -= 0.001f; } // Decrease scaling

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Fig 3: Translation, rotation and scaling of ice cream’s cone | | |