UTM

UNIVERSITI TEKNOLOGI MALAYSIA

SECV2213 FUNDAMENTAL OF COMPUTER GRAPHICS (SECTION 01)

ASSIGNMENT 3

SUBMISSION DATE: 8 JUNE 2025

LECTURER NAME : TS. DR. GOH EG SU

GROUP NAME: TrioGraphiX	
GROUP MEMBERS	**MATRIC NUMBER**
NUR AINA BALQIS BINTI MOHAMAD ZAPARIN	A23CS0151
KRISTINE ELLE BENJAMIN	A23CS0095
NUR AMIERA ZULAIKHA BINTI HARDI	A23CS0153

Table of Contents

1.0 Introduction

For Assignment 3, we were tasked with creating a complete 3D scene using hierarchical modeling. Inspired by one of the cutest Pokemon characters, Jigglypuff, we decided to recreate it but with a twist, we added wings for a whimsical, fantasy touch. This report elaborates on how we structured our model hierarchically, detailing each part and how it connects in the overall design.

2.0 Objective

This project aimed to assess our ability to:

- Create a fully modeled character using basic 3D primitives.
- Apply hierarchical modeling techniques using glPushMatrix() and glPopMatrix().
- Plan and execute 3D transformations including translation, rotation, and scaling.
- Build an interactive 3D scene with lighting and mouse control.
- Present our creative and technical decisions in a clear, well-structured report.

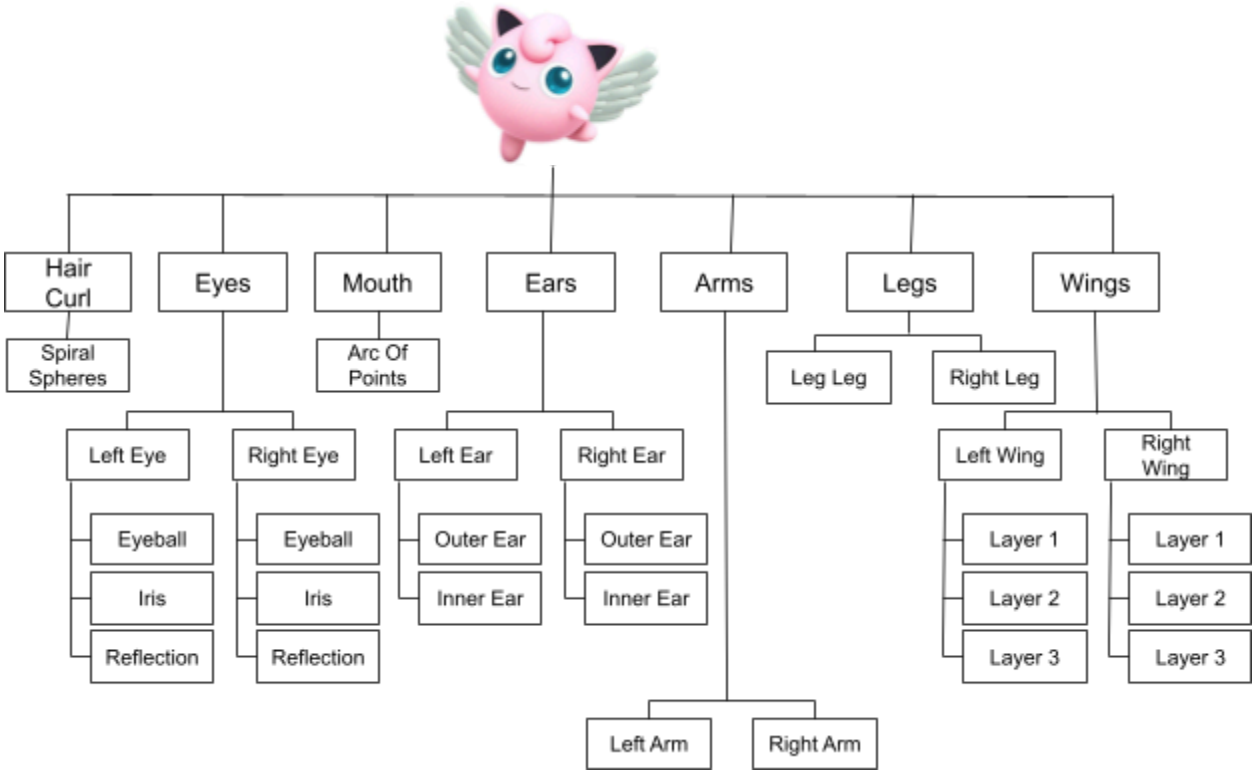
3.0 Overview of Hierarchical Modelling

Hierarchical modeling means organizing a 3D object into a tree structure. The root object, like Jigglypuff's body, has multiple child objects like eyes, limbs, ears, and each child can also have its own children like eye has iris and iris has reflection. This structure allows for relative transformations, if the parent moves, the children move with it.

We used:

- glPushMatrix() to save the current transformation state.
- glPopMatrix() to return to the previous state.
- Local transformations (translate, scale, rotate) inside each body part function.

4.0 Hierarchical Tree Diagram



5.0 Detailed Code and Component Explanation

In this section, we break down our Jigglypuff model into its components, explain their function and appearance, and provide technical insight into how each part was implemented using OpenGL primitives and transformations. Each component is structured hierarchically and relies on its own local transformation to remain modular and organized within the overall model.

5.1 Main Body

- Primitive: glutSolidSphere(0.8)
- Color: Light pink (glColor3f(1.0, 0.75, 0.8))
- Role: This acts as the root node of our hierarchical model. All other body parts are defined as child nodes and placed relative to this main sphere.

Code reference:

```
void drawJigglypuff() {
    glPushMatrix();
    glColor3f(1.0, 0.75, 0.8);
    drawSphere(0.8);

    drawHairCurl();
    drawEye(true);
    drawEye(false);
    drawMouth();

    drawEar(-0.45, 0.55, 0.0, true);
    drawEar(0.45, 0.55, 0.0, false);

    drawLimb(-0.45, -0.4, 0.55, true);
    drawLimb(0.45, -0.4, 0.55, true);
    drawLimb(-0.3, -0.7, 0.1, false);
    drawLimb(0.3, -0.7, 0.1, false);

    drawWing(true);
    drawWing(false);

    glPopMatrix();
}
```

5.2 Hair Curl

- Model: A spiral formed by multiple small spheres.

- Technique: Uses sinf() and cosf() to define positions in a 3D spiral pattern. The radius decreases over time to simulate a natural curl. The end of the curl is pulled forward and lowered to give it a sense of motion and gravity.

Code reference:

```
void drawHairCurl() {
    float angleStep = 0.15f;
    float radius = 0.18f;
    float baseRadius = 0.07f;
    float totalAngle = 7.85f; // approx 2.5 * π

    glColor3f(1.0f, 0.75f, 0.8f); // Light pink

    for (float angle = 0.0f; angle <= totalAngle; angle += angleStep) {
        float x = radius * cosf(angle);

        // Start with normal wave, then curve down at the end
        float y = 0.83f + 0.04f * sinf(angle * 1.5f);

        // Add extra forward pull toward the end of curl
        float z = radius * sinf(angle) + 0.03f * (angle / totalAngle); // +z to bring it toward forehead
        y -= 0.015f * (angle / totalAngle); // gently lower end of curl

        float taper = 1.0f - (angle / totalAngle);
        float currentRadius = baseRadius * taper;

        glPushMatrix();
        glTranslatef(x, y, z);
        glutSolidSphere(currentRadius, 20, 20);
        glPopMatrix();

        radius -= 0.003f; // spiral tightens
    }
}
```

5.3 Eyes (Left and Right)

- Each eye contains three subcomponents that are drawn hierarchically:
  Eyeball: A white, slightly flattened sphere (glScalef(1, 1, 0.8))
  Iris: A cyan-colored sphere, smaller and positioned just above the eyeball
  Reflection: A tiny white sphere to simulate light reflection, adding cuteness

- We use the drawEye(bool left) function to draw both eyes, changing the position depending on the side.

Code reference:

- drawEye() in drawJiggltpuff() function

```
drawEye(true); // Left Eye
drawEye(false); // Right Eye
```

true for left eye, false for right eye

- Inside drawEye() :

```
void drawEye(bool left) {
    glPushMatrix();
    glTranslatef(left ? -0.3f : 0.3f, 0.2f, 0.55f); // Closer to face - Eye position

    // Outer white eyeball
    glPushMatrix();
    glScalef(1.0f, 1.0f, 0.8f); // Slightly flattened
    glColor3f(1, 1, 1);
    drawSphere(0.26); // Eyeball
    glPopMatrix();

    // Blue iris
    glTranslatef(0, 0.03, 0.07);
    glPushMatrix();
    glScalef(1.0f, 1.0f, 0.8f);
    glColor3f(0.0f, 1.0f, 1.0f);
    drawSphere(0.20);
    glPopMatrix();

    // Inner white reflection
    glTranslatef(0.015f, 0.05f, 0.12);
    glPushMatrix();
    glScalef(1.0f, 1.0f, 0.6f);
    glColor3f(1, 1, 1);
    drawSphere(0.08);
    glPopMatrix();

    glPopMatrix();
}
```

5.4 Mouth

- Model: A curved line made up of individual points (GL_POINTS)

- Technique: Uses a loop from -120° to 120° to plot points in an arc using sinf() and cosf(). This results in a gentle, expressive smile fitting the character's personality.

Code reference:

```
void drawMouth() {
    glPushMatrix();
    glTranslatef(0.0, -0.18, 0.75);
    glColor3f(0, 0, 0);
    glPointSize(2.5);

    glBegin(GL_POINTS);
    for (float angle = -120.0f; angle <= 120.0f; angle += 1.0f) {
        float rad = angle * 3.14159265f / 180.0f;
        float x = 0.12f * cos(rad);
        float y = 0.13f * sin(rad);
        glVertex3f(x, y, 0.0f);
    }
    glEnd();

    glPopMatrix();
}
```

5.5 Ears

- Each ear has two parts:
  Outer Ear: A vertically stretched pink ellipsoid, created by scaling a sphere. It's tilted outward and slightly forward for a natural pose.
  Inner Ear: A smaller black ellipsoid, offset inward to create depth and shadow.
- We use rotation and positioning to give each ear a distinct angle, adding asymmetry and personality to the character.

Code reference:

- drawEar ()

```
void drawEar(float x, float y, float z, bool left) {
    glPushMatrix();
    glTranslatef(x, y, z);

    // Tilt the ear slightly based on left/right
    glRotatef(left ? -5.0f : 5.0f, 0.0f, 0.0f, 1.0f); // Tilt outward
    glRotatef(15.0f, 1.0f, 0.0f, 0.0f);                // Tilt forward


    // Outer ear (matches body color)
    glPushMatrix();
    glScalef(0.25f, 0.4f, 0.15f); // Flattened ellipse
    glColor3f(0.95f, 0.6f, 0.7f); // Jigglypuff pink (same as body)
    glutSolidSphere(1.0, 20, 40);
    glPopMatrix();

    // Inner ear (black)
    glPushMatrix();
    glTranslatef(0.0f, 0.09f, 0.12f); // Offset inward
    glScalef(0.15f, 0.25f, 0.04f);    // Smaller ellipse
    glColor3f(0.0f, 0.0f, 0.0f);      // Black inner ear
    glutSolidSphere(1.0, 20, 20);
    glPopMatrix();

    glPopMatrix();
}
```

- drawEar() in drawJigglypuff() function

```
drawEar(-0.45, 0.50, 0.0, true);  // Left ear
drawEar(0.45, 0.50, 0.0, false);  // Right ear
```

true for left ear, false for right ear

5.6 Arms and Legs

- Both arms and legs are handled by the same function: drawLimb(x, y, z, isArm).
  Arms: Smaller spheres, positioned near the top sides.
  Legs: Slightly larger and lower to ground the character.

- Each limb is a scaled sphere, stretched slightly in the y direction.

Code reference:

- drawLimb() in drawJigglypuff() function

```
drawLimb(-0.45, -0.4, 0.55, true); // Left arm
drawLimb(0.45, -0.4, 0.55, true); // Right arm
drawLimb(-0.3, -0.7, 0.1, false); // Left leg
drawLimb(0.3, -0.7, 0.1, false); // Right leg
```

  true for arm, false for leg

- drawLimb()

```
void drawLimb(float x, float y, float z, bool isArm) {
    glPushMatrix();
    glTranslatef(x, y, z);
    glScalef(0.15, isArm ? 0.15 : 0.2, 0.15);
    glColor3f(1.0, 0.75, 0.8);
    drawSphere(1.0);
    glPopMatrix();
}
```

5.7 Wings

This is our most creative feature and one of the most complex parts of the hierarchy.

- Each wing consists of 3 layers of 3 feathers.
- Layers are vertically and backward offset (layerYOffset, layerZOffset).
- Feathers are just scaled spheres that are positioned in a fanned-out layout.
- Left and right wings are mirrored using glScalef(-1.0f, 1.0f, 1.0f) for the left wing.

Code reference:

```
void drawWing(bool left) {
    glPushMatrix();
    // Position wing relative to body
    glTranslatef(left ? -0.85f : 0.85f, 0.2f, -0.3f);
    glScalef(left ? -1.0f : 1.0f, 1.0f, 1.0f);  // Mirror if left

    int layers = 3;
    int feathersPerLayer = 3;
    for (int layer = 0; layer < layers; ++layer) {
        float layerYOffset = 0.1f * layer;      // stack upward
        float layerZOffset = -0.05f * layer;    // backward for layering

        for (int i = 0; i < feathersPerLayer; ++i) {
            glPushMatrix();
            // Feather position across wing span
            float xOffset = 0.2f * i;

            // Slight upward tilt for wing shape
            float yOffset = layerYOffset + 0.05f * i;

            // Feather curvature (optional)
            float rotation = -10 + 5 * i;

            glTranslatef(xOffset, yOffset, layerZOffset);
            glRotatef(rotation, 0, 0, 1);  // Rotate feather to fan it

            // Longer feathers for outer layers
            float scaleX = 0.35f + 0.05f * layer;
            float scaleY = 0.12f;
            float scaleZ = 0.05f;

            glScalef(scaleX, scaleY, scaleZ);
            glColor3f(0.95f, 0.95f, 0.95f);
            drawSphere(1.0);
            glPopMatrix();
        }
    }
    glPopMatrix();
}
```

- This gave our model a fantasy aesthetic and fulfilled the assignment's creative requirement.

6.0 Lighting and Environment

- We initialized lighting in initLighting():
  Ambient Light: Soft background lighting.
  Diffuse Light: Stronger directional light.
  Light Position: Top-right front.

- We also used glEnable(GL_COLOR_MATERIAL) to let colors blend naturally under light.

```
void initLighting() {
    GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
    GLfloat light_ambient[] = { 0.2, 0.2, 0.2, 1.0 };
    GLfloat light_diffuse[] = { 0.8, 0.8, 0.8, 1.0 };

    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_COLOR_MATERIAL);
}
```

- The background color is sky blue (glClearColor(0.53, 0.81, 0.92, 1.0)), matching the cheerful mood.

```
void init() {
    glClearColor(0.53f, 0.81f, 0.92f, 1.0f); // Sky blue
    glEnable(GL_DEPTH_TEST);
}
```

7.0 Interaction and Camera Control

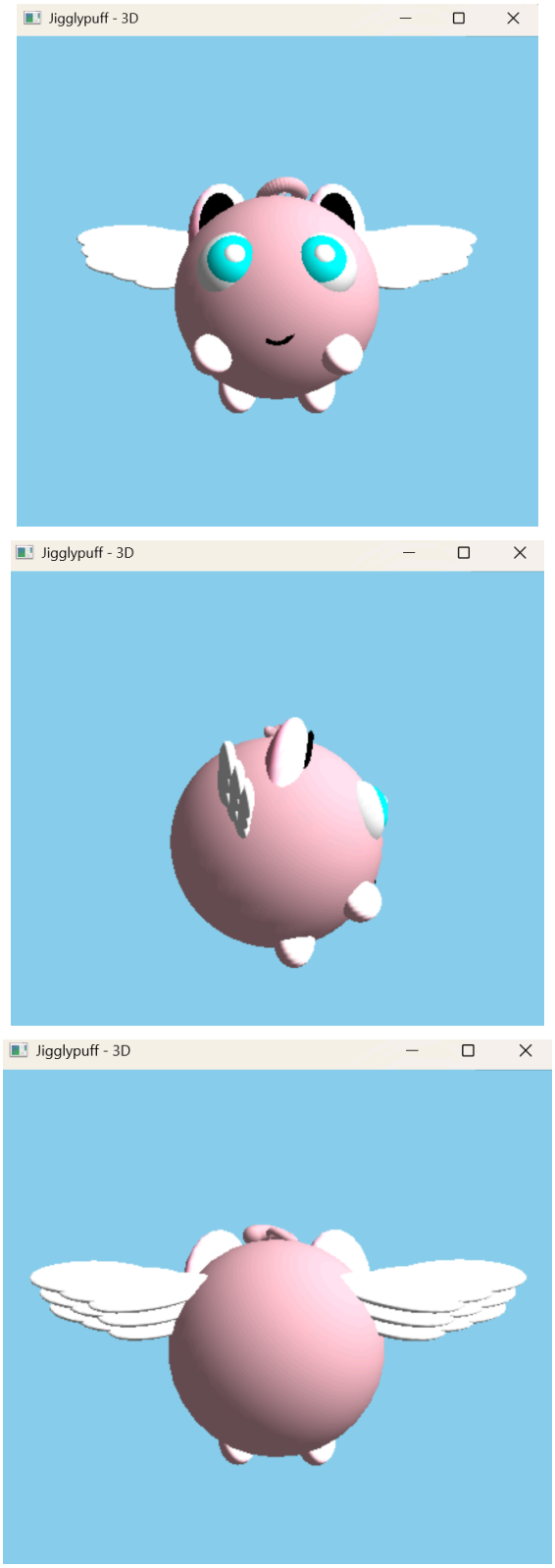
- We added mouse rotation using:
  glutMouseFunc() for mouse button events
  glutMotionFunc() for tracking motion while dragging

- This allows the user to rotate the character around both X and Y axes using mouse drag, offering a full 360° view of our model.

```
void mouseButton(int button, int state, int x, int y) {
    if (button == GLUT_LEFT_BUTTON) {
        if (state == GLUT_DOWN) {
            isDragging = true;
            lastX = x;
            lastY = y;
        }
        else {
            isDragging = false;
        }
    }
}

void mouseMotion(int x, int y) {
    if (isDragging) {
        rotY += (x - lastX) * 0.5f;
        rotX += (y - lastY) * 0.5f;
        lastX = x;
        lastY = y;
        glutPostRedisplay();
    }
}
```

8.0 Output



9.0 Conclusion

This project allowed us to put together everything we've learned so far from modeling and transformations to lighting and interactivity. Our final model of Jigglypuff with wings is not only visually expressive but also technically structured, using a correct and thoughtful hierarchical model.

We feel proud of the outcome, especially considering the effort and attention to detail that went into the curl, wings, and lighting. More than just an assignment, this was a fulfilling and creative learning experience.

Reference

Angel, E., & Shreiner, D. (2011). *Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL*. https://ci.nii.ac.jp/ncid/BB1012728X

Super Smash Bros Wiki. (2022). *Jigglypuff (SSB4)* [Image]. Fandom. https://static.wikia.nocookie.net/chick-hicks/images/a/a6/Jigglypuff_SSB4.png

The Pokémon Company International. (2024). *Jigglypuff - Pokémon character profile*. https://www.pokemon.com