

```

/**
 * Graphics Pgm 3 for Prajun Trital
 *
 * EXTRA CREDIT
 * - Added dust particles that cross the screen horizontally.
 * - The diamond gets displaced by 4 units due to the wind effect.
 *
 * ARCHITECTURE
 * GLUT event-driven generation of a canvas with a diamond, and a landing zone.
 * Canvas is produced via the display event handler, which is in `displayCallback`, which calls the series of
 * display lists to draw text on the screen using `GLUT_BITMAP`, draws the diamond using the `drawDiamond`
 * function, and draws the landing zone using `drawLandingZone` function. The `timerFunc` updates the
 * position of the snowflakes and uses 50 ms as the time interval to achieve an animation running at
 * approximately 20 Frame per second(FPS). The `keyboardCallback` detects the keyboard press to start the diamond's
 * fall, moves the diamond, and activates the wind effect. The `initDisplayLists` initiates seven displayLists
 * to record a set of drawing commands for diamond, landing zone, red line, and four different messages.
 * The `initDisplayLists` is called in the main function. The `isPointInsideLandingDip` function determines
 * whether the diamond's tip is within the landing zone dip. The diamond falls with an initial velocity of zero, and
 * its vertical displacement is calculated using the equation that includes the initial
 * distance plus one-half times gravity multiplied by the square of time.
 *
 * EXTRA CREDIT ARCHITECTURE
 * The `timerFunc` adds dust particle into the array called `dustParticles`, updates the position of it and
 * also removes the dust particles. The `displayCallback` loops over the dustParticles array to draw
 * each dust particle, and the `keyboardCallback` toggles the wind effect by enabling or disabling a boolean variable
 * `wind_enabled`, which is used in the displayCallback to displace the diamond.
 */

#include <GL/glew.h>
#include <GL/freeglut.h>
#include <stdio.h>
#include "OpenGL445Setup-2025.h"
#include <cstring>
#include <vector>

#define canvas_Width 800
#define canvas_Height 600

// Display list index to draw a diamond.
GLuint diamondList;

// Display list index to draw bottom red line.
GLuint bottomRedLineList;

// Display list index to draw the landing zone.
GLuint landingZoneList;

// Display list index for the "Fuel" label.
GLuint fuelLabellList;

// Display list index for the "Fuel" label.
GLuint youWinLabellList;

// Display list index for "Press 'W' to enable the wind effect" message
GLuint enableWindEffectMessageList;

// Display list index for "Press 'D' to disable the wind effect" message
GLuint disableWindEffectMessageList;

// Flag to show the wind toggle message
bool showWindMessage = false;

// Timer to control message display duration
float windMessageTimer = 0.0f;

// Starting fuel value.
int fuel = 200;

// Diamond's vertical position.
float diamond_y = 575.0f;

// Diamond's horizontal position.
float diamond_x = 400.0f;

// Starting position (used when simulation begins)
float diamond_initial_y = 575.0f;

```

```

bool simulation_started = false;

// Time (in seconds) since the drop started
float simulation_time = 0.0f;

// Current gravity acceleration (ft/s^2)
float gravity = 0.0f;

// Gravity constants (in ft/s^2; negative means downward acceleration)
const float MOON_GRAVITY = -5.31f; // Moon's gravity
const float IO_GRAVITY = -5.9f; // Io's gravity constant

// Struct to represent a dust particle
struct DustParticle {
    float x;
    float y;
};

// Stores all active dust particles
std::vector<DustParticle> dustParticles;

// Accumulates time for new dust particle creation
float dustTimer = 0.0f;

// Frame interval in milliseconds for 20 fps (1000ms / 20 = 50ms)
const int frame_interval = 50;

// Wind effect is initially disabled
bool wind_enabled = false;

/**
 * Draws an octahedral diamond with a size of 25 units
 */
void drawDiamond() {
    // For UAH's blue color with hex code #0077C8
    // Hex Breakdown:
    // Red: 00 (0 in decimal)
    // Green: 77 (119 in decimal)
    // Blue: C8 (200 in decimal)

    // Converting to normalized RGB (dividing by 255):
    // Red: 0 / 255 = 0.0
    // Green: 119 / 255 ≈ 0.467
    // Blue: 200 / 255 ≈ 0.784
    glColor3f(0.0f, 0.467f, 0.784f);

    glPushMatrix();
    // Scaling the octahedron so that each edge is about 25 units long.
    glScalef(25.0f, 25.0f, 25.0f);
    glutWireOctahedron();
    glPopMatrix();
}

/**
 * Draws a bottom red line that extends the axis.
 */
void drawBottomRedLine()
{
    glColor3f(1.0f, 0.0f, 0.0f);
    glBegin(GL_LINES);
        glVertex3f(0.0f, 7.0f, -50.0f);
        glVertex3f(canvas_Width, 7.0f, -50.0f);
    glEnd();
}

/**
 * Draws a landing zone on the bottom left.
 */
void drawLandingZone() {
    // 10 units = 10% of 600 = 60
    // 40 units = 40% of 800 = 320
    glColor3f(0.0f, 0.467f, 0.784f);

    glBegin(GL_LINE_LOOP);
        // Bottom edge of the rectangle
        glVertex3f(10.0f, 0.0f, -50.0f);
        glVertex3f(270.0f, 0.0f, -50.0f);

```

```

// Right side of the rectangle at y=32
glVertex3f(270.0f, 32.0f, -50.0f);

// Right edge of the dip (inner top)
glVertex3f(165.0f, 32.0f, -50.0f);

// Triangular dip bottom at y=7 (same as the red line at y=7)
glVertex3f(140.0f, 7.0f, -50.0f);

// Left edge of the dip (inner top)
glVertex3f(115.0f, 32.0f, -50.0f);

// Left side of the rectangle at y=32
glVertex3f(10.0f, 32.5f, -50.0f);
glEnd();
}

/**
 * Function to decide if the tip of the diamond is inside the landing zone
 * dip or not. Returns true if the tip is inside the landing zone.
 *
 * @param pointX x-coordinate of the tip of the diamond
 * @param pointY y-coordinate of the tip of the diamond
 */
bool isPointInsideLandingDip(float pointX, float pointY) {
    // Landing dip defined by vertices: (115,32), (165,32), (140,7)
    if (pointY < 7.0f || pointY > 32.0f) {
        return false;
    }
    // The total vertical distance is 32 - 7 = 25.
    float t = (32.0f - pointY) / 25.0f; // Goes from 0 at y=32 to 1 at y=7

    float leftBoundary = 115.0f + 25.0f * t; // Moves from 115 to 140
    float rightBoundary = 165.0f - 25.0f * t; // Moves from 165 to 140

    return (pointX >= leftBoundary && pointX <= rightBoundary);
}

/**
 * Method to init the DisplayLists.
 * Used in main function.
 */
void initDisplayLists() {
    // Display list for diamond
    diamondList = glGenLists(1);
    glNewList(diamondList, GL_COMPILE);
    drawDiamond();
    glEndList();

    // Display list for bottom red line
    bottomRedLineList = glGenLists(2);
    glNewList(bottomRedLineList, GL_COMPILE);
    drawBottomRedLine();
    glEndList();

    // Display list for landing zone
    landingZoneList = glGenLists(3);
    glNewList(landingZoneList, GL_COMPILE);
    drawLandingZone();
    glEndList();

    // Fuel message display list
    fuelLabelList = glGenLists(4);
    glNewList(fuelLabelList, GL_COMPILE);
    const char* fuelLabel = "Fuel";
    for (size_t i = 0; i < strlen(fuelLabel); i++) {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, fuelLabel[i]);
    }
    glEndList();

    // You win message display list
    youWinLabelList = glGenLists(5);
    glNewList(youWinLabelList, GL_COMPILE);
    const char* youWinLabel = "YOU WIN";
    for (size_t i = 0; i < strlen(youWinLabel); i++) {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, youWinLabel[i]);
    }
    glEndList();
}

```

```

    }
    glEndList();

    // Wind enable message display list
    enableWindEffectMessageList = glGenLists(6);
    glNewList(enableWindEffectMessageList, GL_COMPILE);
    const char* windEnableMsg = "Press 'W' to enable wind effect";
    for (size_t i = 0; i < strlen(windEnableMsg); i++) {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, windEnableMsg[i]);
    }
    glEndList();

    // Wind disable message display list
    disableWindEffectMessageList = glGenLists(7);
    glNewList(disableWindEffectMessageList, GL_COMPILE);
    const char* windDisableMsg = "Press 'D' to disable wind effect";
    for (size_t i = 0; i < strlen(windDisableMsg); i++) {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, windDisableMsg[i]);
    }
    glEndList();
}

/**
 * The display callback sets up the display and renders the initial scene with a diamond and a landing
 * zone.
 * It executes the series of display lists to render the texts, diamond and landing zone.
 * The background is cleared to yellow, and the objects are rendered in UAH's Blue.
 */
void displayCallback() {
    glClearColor(1.0f, 1.0f, 0.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);

    // Resets the modelview matrix to ensure transformations start fresh each frame.
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    // Draw the diamond
    glPushMatrix();
    glTranslatef(diamond_x, diamond_y, -50.0f);
    glCallList(diamondList);
    glPopMatrix();

    // Draw the bottom Red line
    glCallList(bottomRedLineList);

    // Draw the Landing Zone
    glCallList(landingZoneList);

    // Draw dust particles (each as a 3-unit long horizontal gray line)
    glColor3f(0.5f, 0.5f, 0.5f); // Gray color
    glBegin(GL_LINES);
    for (size_t i = 0; i < dustParticles.size(); i++) {
        glVertex3f(dustParticles[i].x, dustParticles[i].y, -50.0f);
        glVertex3f(dustParticles[i].x + 3.0f, dustParticles[i].y, -50.0f);
    }
    glEnd();

    glColor3f(0.0f, 0.0f, 0.0f);

    // Draw the Fuel text
    glRasterPos2i(740, 570);
    glCallList(fuelLabelList);

    // Draw the numeric fuel value
    glRasterPos2i(740, 550);
    char fuelStr[20];
    sprintf(fuelStr, sizeof(fuelStr), "%d", fuel);
    for (size_t i = 0; i < strlen(fuelStr); i++) {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, fuelStr[i]);
    }

    // Draw the YOU WIN text
    if (isPointInsideLandingDip(diamond_x, diamond_y - 25.0f)) {
        glRasterPos2i((canvas_Width / 2) - 50, canvas_Height / 2);
        glCallList(youWinLabelList);
    }
}

```

```

// Display wind toggle message at center for 1.5 seconds
if (showWindMessage) {
    glRasterPos2i((canvas_Width / 2) - 117, 310);
    glCallList(enableWindEffectMessageList);

    glRasterPos2i((canvas_Width / 2) - 122, 290);
    glCallList(disableWindEffectMessageList);
}

glFlush();
}

/**
 * Timer callback to update the diamond's position.
 * It moves the diamond and triggers a redisplay.
 */
void timerFunc(int value) {
    float dt = frame_interval / 1000.0f; // Convert frame interval to seconds
    dustTimer += dt;

    // Update wind message timer
    if (showWindMessage) {
        windMessageTimer += dt;
        if (windMessageTimer >= 1.5f) {
            showWindMessage = false;
            windMessageTimer = 0.0f;
        }
    }

    if (simulation_started) {
        simulation_time += dt;
        // Compute the new vertical position using:  $p(t) = p_0 + 0.5 * a * t^2$ .
        // Since the diamond falls from the rest, the initial velocity is 0
        diamond_y = diamond_initial_y + 0.5f * gravity * (simulation_time * simulation_time);

        // Apply wind effect: displace right by 4 units if between 200 and 300 units above ground
        if (wind_enabled && diamond_y >= 200.0f && diamond_y <= 300.0f) {
            diamond_x += 4.0f;
        }

        float tip_x = diamond_x;
        float tip_y = diamond_y - 25.0f;

        // Landing dip triangle vertices: A = (115, 50), B = (165, 50), C = (140, 25)
        if (isPointInsideLandingDip(tip_x, tip_y)) {
            // Snap the diamond into perfect alignment inside the dip:
            diamond_x = 140.0f; // Center horizontally with the dip.
            diamond_y = 32.0f; // So that the lower tip (diamond_y - 25) becomes 7, matching the dip's bottom.
            simulation_started = false;
            simulation_time = 0.0f;
        }

        // Check if tip touches the red line (y=7).
        if (tip_y <= 7.0f) {
            diamond_y = 32.0f; // 7 + 25
            simulation_started = false;
            simulation_time = 0.0f;
        }

        // Create a new dust particle every 600 milliseconds
        if (dustTimer >= 0.6f) {
            DustParticle newParticle;
            newParticle.x = 0.0f;
            newParticle.y = 200.0f + (rand() % 101); // Random y between 200 and 300
            dustParticles.push_back(newParticle);
            dustTimer -= 0.6f;
        }

        // Update dust particles: move right by 4 units
        for (size_t i = 0; i < dustParticles.size(); ++i) {
            dustParticles[i].x += 4.0f;
        }

        // Remove particles that are off-screen (x >= 800)
        for (int i = dustParticles.size() - 1; i >= 0; --i) {
            if (dustParticles[i].x >= 800.0f) {
                dustParticles.erase(dustParticles.begin() + i);
            }
        }
    }
}

```

```

    }
}

glutPostRedisplay();
glutTimerFunc(frame_interval, timerFunc, 1);
}

/**
 * Handles keyboard input to control the animation and movement of the diamond.
 *
 * Controls:
 * - Press 'M' to apply the Moon's gravity.
 * - Press 'I' to apply Io's gravity.
 * - Press 'H' to move the diamond left.
 * - Press 'J' to move the diamond right.
 * - Press 'U' to move the diamond up.
 * - Press 'W' to enable the wind effect.
 * - Press 'D' to disable the wind effect.
 */
void keyboardCallback(unsigned char key, int x, int y) {
    if (!simulation_started) {
        if (key == 'm' || key == 'M') {
            gravity = MOON_GRAVITY;
            simulation_started = true;
            simulation_time = 0.0f;
            diamond_initial_y = diamond_y;
            showWindMessage = true;
            windMessageTimer = 0.0f;
        }
        if (key == 'i' || key == 'I') {
            gravity = IO_GRAVITY;
            simulation_started = true;
            simulation_time = 0.0f;
            diamond_initial_y = diamond_y;
            showWindMessage = true;
            windMessageTimer = 0.0f;
        }
    }

    // Only move if the simulation has started
    if (simulation_started) {
        if (key == 'h' || key == 'H') {
            diamond_x -= 4.0f; // Moves left 4 units
        }
        if (key == 'j' || key == 'J') {
            diamond_x += 4.0f; // Moves right 4 units
        }
        if (key == 'u' || key == 'U') {
            if (fuel > 0) {
                diamond_y += 5.0f;
                diamond_initial_y += 5.0f; // For continuous gravity simulation.
                fuel -= 5;
            }
        }
    }

    // Toggle wind effect with 'W' (enable) and 'D' (disable)
    if (key == 'w' || key == 'W') {
        wind_enabled = true;
    }

    if (key == 'd' || key == 'D') {
        wind_enabled = false;
    }
}

char canvas_Name[] = "Diamond Drop";

int main(int argc, char ** argv) {
    glutInit(&argc, argv);
    my_setup(canvas_Width, canvas_Height, canvas_Name);

    glutDisplayFunc(displayCallback);
    glutKeyboardFunc(keyboardCallback);

    // Create the display list for the diamond.

```

```
initDisplayLists();

// Set up the timer function for 20 fps.
glutTimerFunc(frame_interval, timerFunc, 1);

glutMainLoop();
return 0;
}
```