CPT205 2D Modelling Project – Birthday card

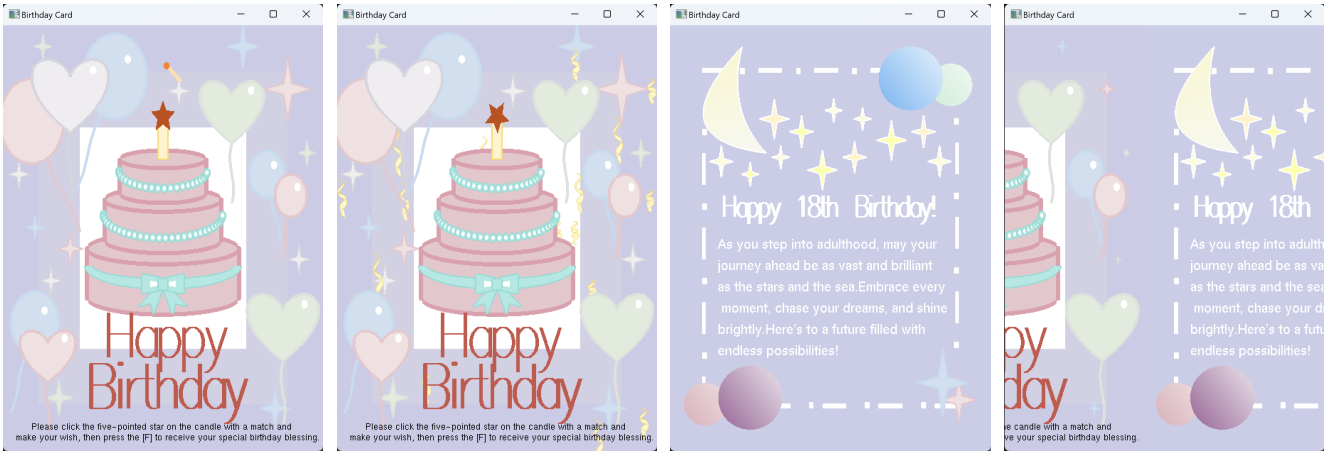
Rui Sang 2251576
Information and Computing Science

October 29, 2024

1 Introduction

1.1 Theme and Effects of the Greeting Card

This project presents a 2D dynamic birthday card themed around an "18th Birthday Blessing," developed using OpenGL and the freeglut library. The card conveys best wishes to an 18-year-old stepping into adulthood, hoping their future shines as brightly as the stars, full of endless possibilities.Sized at 450×600 pixels, the card adopts a soft, minimalist style. The cover features classic birthday elements such as a cake, candles, stars, balloons, and ribbons. The inner page focuses on the theme with celestial elements like stars and the moon, along with a birthday message. The card's effects are shown in Figure 1, illustrating the cover, inner page, and the transition between them.



(a) Cover Initialization (b) Cover After Interaction (c) Inner Page Interface (d) Cover to Inner

Figure 1: Screenshots of the Greeting Card Effects

This report details the design approach, code structure, element rendering, and animation implementation, while also providing instructions on how users can interact with the card.

1.2 Code Structure and Logic

The code structure is divided into several parts:

1) **Global Variables and Initialization of Dynamic Elements**: This section defines the global variables controlling animations (e.g., star scaling, ribbon speed, star rotation) and initializes dynamic elements for smooth animations.

2) **Element Rendering**: It renders basic shapes like rectangles and circles, which are then combined into more complex elements, such as the cake, candles, balloons, ribbons, stars, and the moon.

3) **Rendering Layers**: The rendering is done in layers, with the cover displaying the background, cake, and text, while the inner page shows celestial elements and a birthday message. These layers are combined into the final scene.

4) **Animation Updates**: This part updates dynamic elements like star scaling, ribbon movement, and candle star rotation, ensuring fluid animations.

5) **Interaction**: Handles user interactions—clicking the candle's star triggers rotation and ribbon animation, while pressing the 'F' key transitions from the cover to the inner page.

6) **Timers and Main Function**: Timers control animation updates, and the main function initializes the window, registers events, and starts the rendering process.

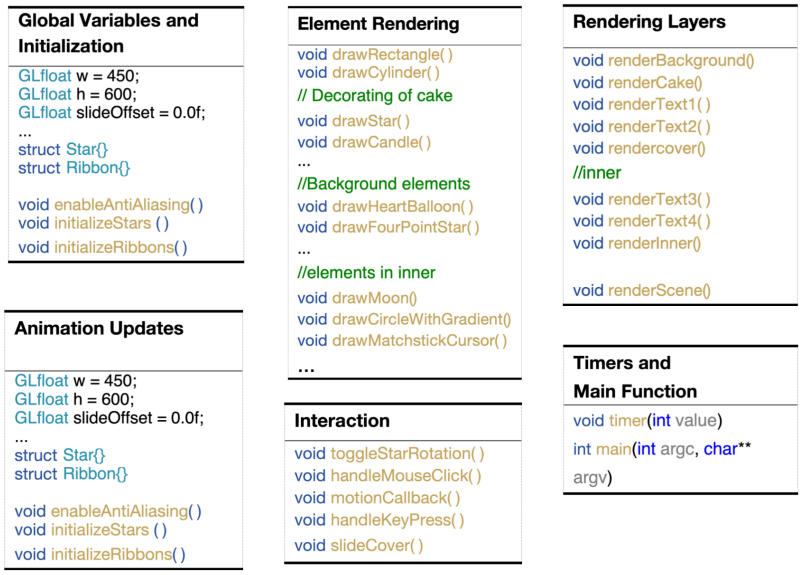


Figure 2: code structure and functions

This modular structure ensures clarity, ease of management, and enhances the code's readability and scalability.

2 Content Design

2.1 Cover Design

2.1.1 Static Elements Design

A. Background: To avoid a monotonous background, three layers of rectangles with different colors are drawn using `'glBegin(GL_QUADS)'`, which are stacked sequentially to create a gradient effect, adding depth to the scene.

B. Two Types of Balloons: Two balloon shapes are heart-shaped and oval-shaped balloons:

1. **Oval Balloons:** The ovals are approximated by drawing polygons using `"glBegin(GL_POLYGON)"`. The x and y coordinates of the oval are generated with `cos()` and `sin()` functions, and the width and height parameters control the major and minor axes of the oval.

2. **Heart Balloons:** The heart shape is generated using a parametric equation. Specifically, `"pow(sin(angle), 3)"` controls the X-coordinate, and `cosf(angle)` shapes the Y-coordinate, forming the characteristic heart contour.

Figure 3: Two Types of balloons

To enhance the refinement of the balloons, a thicker border is added using `"glBegin(GL_LINE_LOOP)"` to outline the balloon edges.

Balloon Strings: The strings are drawn using a **Bézier curve** to simulate a floating effect. The Bézier curve formula is:

$$P(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3$$

```
float lineStartX = centerX;
float lineStartY = centerY - height * 0.6f;
glColor3fv(borderColor);
glLineWidth(4.8f);
glBegin(GL_LINE_STRIP);
for (float t = 0.0; t <= 1.0; t += 0.02) {
    float x = (1 - t) * (1 - t) * (1 - t) * lineStartX + 3 * (1 - t) * (1 - t) * t * (lineStartX - width * 0.1f)
        + 3 * (1 - t) * t * t * (lineStartX + width * 0.1f) + t * t * t * lineStartX;
    float y = (1 - t) * (1 - t) * (1 - t) * (lineStartY)+3 * (1 - t) * (1 - t) * t * (lineStartY - height * 0.4f)
        + 3 * (1 - t) * t * t * (lineStartY - height * 0.6f) + t * t * t * (lineStartY - height * 1.0f);
    glVertex2f(x, y);
}
glEnd();
```

Figure 4: Specific code implementation of Bézier curve

```
glBegin(GL_LINE_LOOP);
for (int i = 0; i <= numSegments; i++) {
    float angle = 2.0f * 3.14159f * i / numSegments;
    float x = width * 0.5f * cos(angle);
    float y = height * 0.5f * sin(angle);

    float rotatedX = centerX + x * cos(angleRad) - y * sin(angleRad);
    float rotatedY = centerY + x * sin(angleRad) + y * cos(angleRad);

    glVertex2f(rotatedX, rotatedY);
}
glEnd();
```

Figure 5: code of Rotation

where P_0 to P_3 are control points, generating a smooth curve for the balloon strings.

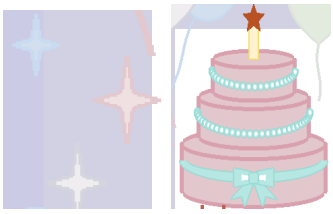
To add variety to the background, I implemented a rotation transformation for the balloons in different positions. This is achieved by applying a rotation matrix after converting the rotation angle to radians:

$$x' = x \cdot \cos(\theta) - y \cdot \sin(\theta)$$

$$y' = x \cdot \sin(\theta) + y \cdot \cos(\theta)$$

The angle of rotation is controlled by the variable `angleDeg`, allowing the balloons to tilt at various angles. Different colors and sizes further diversify the balloon background, creating a visually rich scene.

C. Four-point Star: The four-point star is drawn using `"glBegin(GL_TRIANGLE_FAN)"` to create the main body. First, the center point of the star is drawn, followed by the outer vertices, forming a continuous fan of triangles. The positions of the vertices are calculated using `cosf(angle)` and `sinf(angle)`, with larger radii for the main points and smaller radii for the inner points, ensuring clear sharp edges and indentations. The inner region is filled with a smooth color gradient, while the darker edges enhance the star's 3D effect and layering.



(a) Four stars (b) Cake

Figure 6: The elements in cover

D. Birthday Cake: The birthday cake design focuses on creating a sense of dimensionality. A combination of rectangles and ellipses is used to create the multi-layered cake, with a candle placed on top to enhance realism. The sides of the cake are drawn using `"glBegin(GL_QUADS)"`, with the top and bottom of the rectangle representing the upper and lower edges of each layer.

The top of the cake is drawn as an ellipse using `"glBegin(GL_POLYGON)"`, approximating the elliptical shape with multiple vertices. By adjusting the number of segments with `numSegments`, the smoothness of the ellipse can be controlled. The angle `theta` is iterated over to compute the positions of each vertex, ensuring the ellipse is fully drawn.

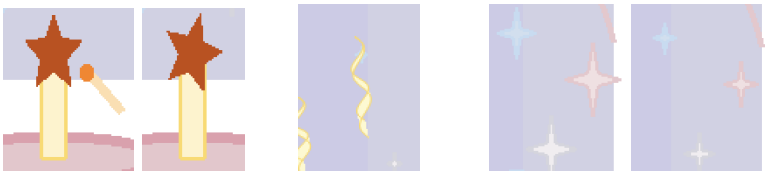
The necklace decoration around the cake is simulated by arranging a series of small circles along a curve. The positions of these circles are determined using `cos()` and `sin()` functions, arranging them along an arc to simulate a necklace wrapping around the cake. The bow decoration is created using simple polygons, mirrored symmetrically for a balanced appearance. The lines are thickened to give the bow a more pronounced 3D shape.

2.1.2 Dynamic Elements Design

The cover includes two main dynamic elements:

A. Rotating Star: The five-point star located at the top of the candle simulates the candle's flame. The star's rotation is triggered by a mouse click. Using `"glPushMatrix"` and `"glRotatef"`, the star rotates around the z-axis, creating a fixed-point rotation effect. The rotation is continuously updated by a timer, generating smooth animation with each frame.

B. Falling Ribbons (Hidden Element): The ribbons are drawn using `"glBegin(GL_QUAD_STRIP)"` and are animated to simulate a wave motion with the sine function `sin(angle)`. The ribbons are decorative elements



(a) Rotation (b) Translation (c) Scaling

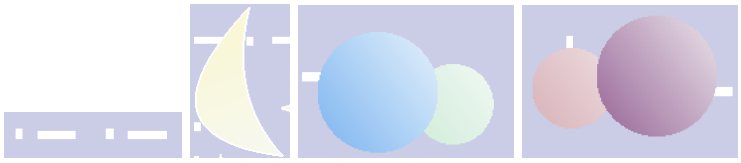
Figure 7: Dynamic Elements

hidden initially and are triggered by the star's rotation, causing them to gradually fall from the top of the screen. The vertical position of the ribbons is controlled by the function `"updateRibbonsPosition()"`, which continuously decreases the y-coordinate to simulate falling. Once a ribbon reaches the bottom of the screen, its position is reset to the top, creating a looping animation that complements the rotating star, enhancing the festive atmosphere.

C. Scaling Stars: The scaling effect of the stars is controlled by setting the maximum and minimum scaling values (`maxStarScale` and `minStarScale`), ensuring the stars do not become too large or too small. A scaling speed (`starScaleSpeed`) is defined to make the scaling effect appear smooth and natural.

2.2 Inner Page Design

2.2.1 Static Elements Design



(a) Border (b) Moon (c) Gradient Planets

Figure 8: Static Elements

A. Background and Border: To maintain stylistic consistency, the inner page background is filled with a light purple color. Since the inner page emphasizes textual content, a combination of long rectangles and small squares is used to create a dashed border effect, giving the inner content a framed appearance that enhances its aesthetic appeal.

B. Moon and Stars: The `"drawMoon()"` function renders a crescent moon using Bézier curves for smooth, natural curvature. The moon's vertices are dynamically calculated to ensure a fluid shape, adding a romantic touch to the page. Twinkling stars further enhance the page's dynamism, creating a dreamy night sky effect that makes the inner page more engaging.

C. Gradient Planets: The planets are drawn as circles with gradient colors to simulate lighting and volume, giving them a three-dimensional appearance. The use of gradient start and end colors adds realism to the planets.

To ensure the planets' positions and sizes remain proportional across different screen resolutions, the window's aspect ratio (width w and height h) is considered during rendering. The `"getAspectRatio()"` function dynamically adjusts the drawing parameters, ensuring the planets' X and Y coordinates are properly adapted to the window's dimensions without distortion.

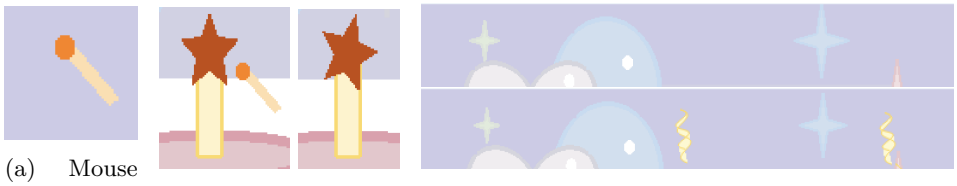
2.2.2 Dynamic Elements Design

Figure 9: Scaling Stars

The primary dynamic element on the inner page is the scaling effect of the stars. These stars continuously grow and shrink, simulating a "breathing" effect as the page loads. This effect is implemented using a timer, which updates the scaling factor of the stars every 16 milliseconds, creating a smooth and dynamic visual effect.

3 Interaction Design

3.1 Mouse Interaction Design

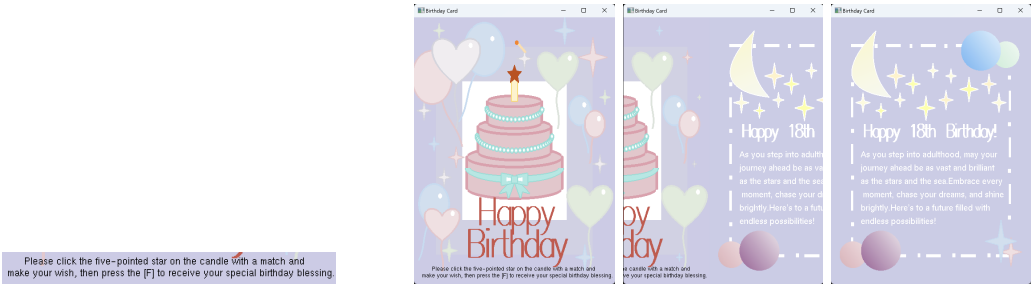


(a) Mouse cursor (b) Rotation of stars (c) The appearance of hidden elements

Figure 10: Mouse interaction

The mouse cursor on the cover page has been customized to resemble a matchstick icon, which moves with the cursor's position. Following the instructions on the cover, the user can use the matchstick to click on the five-pointed star at the top of the candle, simulating a candle-lighting scene. This triggers the star's rotation animation, symbolizing the burning flame of the candle, allowing the user to silently make a wish. Simultaneously, hidden ribbons on the cover page are activated, falling from the top of the screen, enhancing the festive atmosphere and making the animation more engaging. The custom cursor is implemented using `"glutSetCursor(GLUT_CURSOR_NONE)"`, and after the user clicks the star, the cursor reverts to the default arrow style.

3.2 Keyboard Interaction Design



(a) Instruction (b) Animation

Figure 11: Keyboard interaction

While on the cover page, the user can press the 'F' key to trigger the page transition effect. Upon pressing the key, the cover slides off to the right, and the inner page smoothly slides in with a similar animation. The transition between pages is achieved by adjusting the horizontal displacement of the cover and inner page, and the smoothness of the animation is maintained using a timer.

Page Transition Effect: The page transition is initiated by detecting keyboard events using the `"glutKeyboardFunc(handleKeyPress)"` function, which listens for keypresses. When the 'F' key is pressed, a flag is set, triggering the page transition.

Animation Implementation: The sliding effect is achieved by controlling the horizontal movement of the cover and inner pages. A timer (e.g., `"glutTimerFunc"`) is used to ensure the smoothness of the animation, allowing the cover to slide out to the right as the inner page slides in. This ensures a seamless transition, enhancing the user's visual experience.

4 Conclusion

This project successfully designed and implemented a 2D dynamic birthday card themed "18th Birthday Wishes" using OpenGL and freeglut. With a multi-layer gradient background, heart-shaped balloons, and four-pointed stars, it features dynamic elements like a rotating star and falling ribbons triggered by user interaction. Custom mouse cursors and intuitive keyboard controls add to the interactive experience. The clear code structure allows for easy extension, with future improvements aimed at enhancing dynamic effects, user interaction, and interface design.