Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа № 1 по курсу «Компьютерная графика»

Студент: Е. А. Медведев
Преподаватель: Г. С. Филипов
Группа: М8О-301Б

Дата:
Оценка:
Подпись:

Москва, 2024

Тема лабораторной работы: Основы 2D-графикии трансформаций

1 Цель лабораторной работы

В данной лабораторной работе вам предстоит научиться работать с графическим API для отрисовки 2D-примитивов, освоить основные 2D-трансформации (перемещение, масштабирование, поворот) и изучить алгоритмы построения 2D-кривых.

2 Требования

Вы должны использовать C++ (OpenGL+SFML). Программа должна работать в реальном времени,обновляя изображение в цикле. Визуальный результат необходимо продемонстрировать на экране свозможностью управления через интерфейс.

3 Вариант 2: Отрисовка прямоугольника с трансформациями

Создайте программу, которая отрисовывает прямоугольник. Реализуйте возможность изменения положения, угла поворота и масштабирования прямоугольника. Управляйте трансформациями с помощью клавиатуры. Дополнительно: Добавьте возможность изменять цвет прямоугольника в зависимости от направления трансформации.

Описание работы программы

Программа создает графическое окно с использованием SFML и OpenGL, в котором отображается прямоугольник с возможностью управления его положением, масштабом и поворотом. Пользователь может:

- перемещать прямоугольник с помощью клавиш **W, A, S, D**;
- вращать прямоугольник с помощью клавиш **Q** (против часовой стрелки) и **E** (по часовой стрелке);
- изменять масштаб прямоугольника с помощью клавиш **Z** (уменьшение) и **X** (увеличение).

Цвет прямоугольника изменяется в зависимости от направления трансформации. В левом верхнем углу отображается текстовая информация о текущем положении, масштабе и угле поворота. Все трансформации выполняются в реальном времени, а графика обрабатывается через OpenGL с использованием ортографической проекции.

Код программы

```
#include <SFML/Graphics.hpp>
#include <SFML/OpenGL.hpp>
#include <GL/glu.h>
#include <sstream>

const int WINDOW_WIDTH = 800;
const int WINDOW_HEIGHT = 600;


void initOpenGL() {
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, WINDOW_WIDTH, WINDOW_HEIGHT, 0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}


void drawRectangle() {
    glBegin(GL_QUADS);
    glVertex2f(-50.f, -25.f);
    glVertex2f(50.f, -25.f);
    glVertex2f(50.f, 25.f);
```

```
24      glVertex2f(-50.f, 25.f);
25      glEnd();
26  }
27
28
29  sf::Color getColorByDirection(bool up, bool down, bool left, bool right, bool
        rotateLeft, bool rotateRight) {
30      if (up) return sf::Color(255, 182, 193);
31      if (down) return sf::Color(144, 238, 144);
32      if (left) return sf::Color(173, 216, 230);
33      if (right) return sf::Color(255, 255, 224);
34      if (rotateLeft) return sf::Color(255, 255, 120);
35      if (rotateRight) return sf::Color(120, 255, 100);
36      return sf::Color(211, 211, 211);
37  }
38
39
40  int main() {
41      sf::RenderWindow window(sf::VideoMode(WINDOW_WIDTH, WINDOW_HEIGHT), "Rectangle with
            Transformations", sf::Style::Default, sf::ContextSettings(24));
42      window.setFramerateLimit(60);
43
44      sf::Font font;
45      if (!font.loadFromFile("/usr/share/fonts/truetype/dejavu/DejaVuSans-Bold.ttf")) {
46          return -1;
47      }
48
49      sf::Text infoText;
50      infoText.setFont(font);
51      infoText.setCharacterSize(20);
52      infoText.setFillColor(sf::Color::White);
53      infoText.setPosition(10, 10);
54
55      initOpenGL();
56
57      float posX = 400.f, posY = 300.f;
58      float scale = 1.f;
59      float angle = 0.f;
60
61      while (window.isOpen()) {
62          sf::Event event;
63          while (window.pollEvent(event)) {
64              if (event.type == sf::Event::Closed) {
65                  window.close();
66              }
67          }
68
69          if (sf::Keyboard::isKeyPressed(sf::Keyboard::W)) posY -= 1.f;
70          if (sf::Keyboard::isKeyPressed(sf::Keyboard::S)) posY += 1.f;
```

```
71          if (sf::Keyboard::isKeyPressed(sf::Keyboard::A)) posX -= 1.f;
72          if (sf::Keyboard::isKeyPressed(sf::Keyboard::D)) posX += 1.f;
73
74          if (sf::Keyboard::isKeyPressed(sf::Keyboard::Q)) angle -= 1.f;
75          if (sf::Keyboard::isKeyPressed(sf::Keyboard::E)) angle += 1.f;
76
77          if (sf::Keyboard::isKeyPressed(sf::Keyboard::Z)) scale *= 0.99f;
78          if (sf::Keyboard::isKeyPressed(sf::Keyboard::X)) scale *= 1.01f;
79
80          sf::Color color = getColorByDirection(
81              sf::Keyboard::isKeyPressed(sf::Keyboard::W),
82              sf::Keyboard::isKeyPressed(sf::Keyboard::S),
83              sf::Keyboard::isKeyPressed(sf::Keyboard::A),
84              sf::Keyboard::isKeyPressed(sf::Keyboard::D),
85              sf::Keyboard::isKeyPressed(sf::Keyboard::Q),
86              sf::Keyboard::isKeyPressed(sf::Keyboard::E)
87          );
88
89          glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
90          glLoadIdentity();
91          glTranslatef(posX, posY, 0.f);
92          glRotatef(angle, 0.f, 0.f, 1.f);
93          glScalef(scale, scale, 1.f);
94
95          glColor3f(color.r / 255.f, color.g / 255.f, color.b / 255.f);
96          drawRectangle();
97
98          std::stringstream ss;
99          ss << "Position: (" << posX << ", " << posY << ")\n";
100         ss << "Scale: " << scale << "\n";
101         ss << "Rotation: " << angle << " degrees";
102         infoText.setString(ss.str());
103
104         window.pushGLStates();
105         window.draw(infoText);
106         window.popGLStates();
107
108         window.display();
109     }
110
111     return 0;
112 }
```

Примеры работы программы:

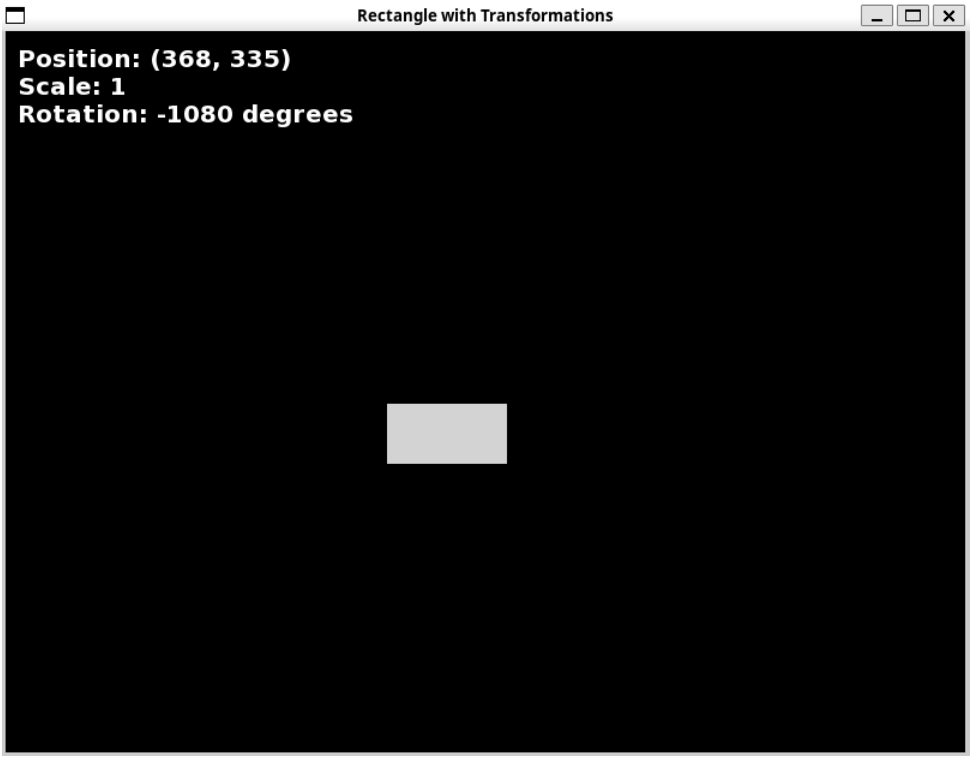

Рис. 1: прямоугольник

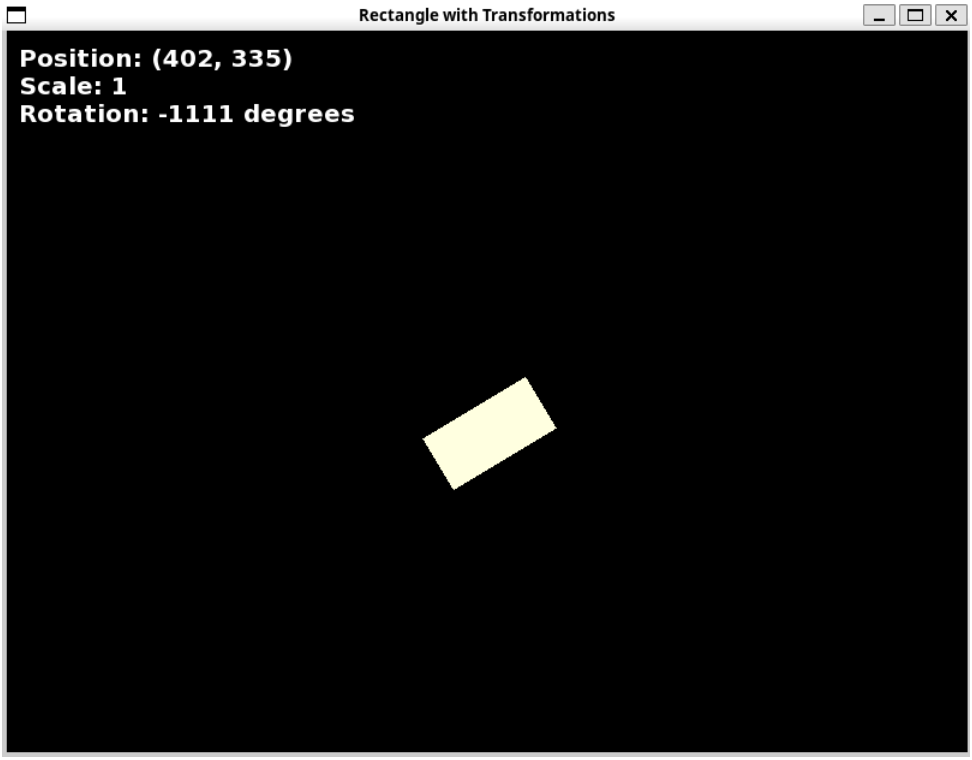

Рис. 2: прямоугольник в движении

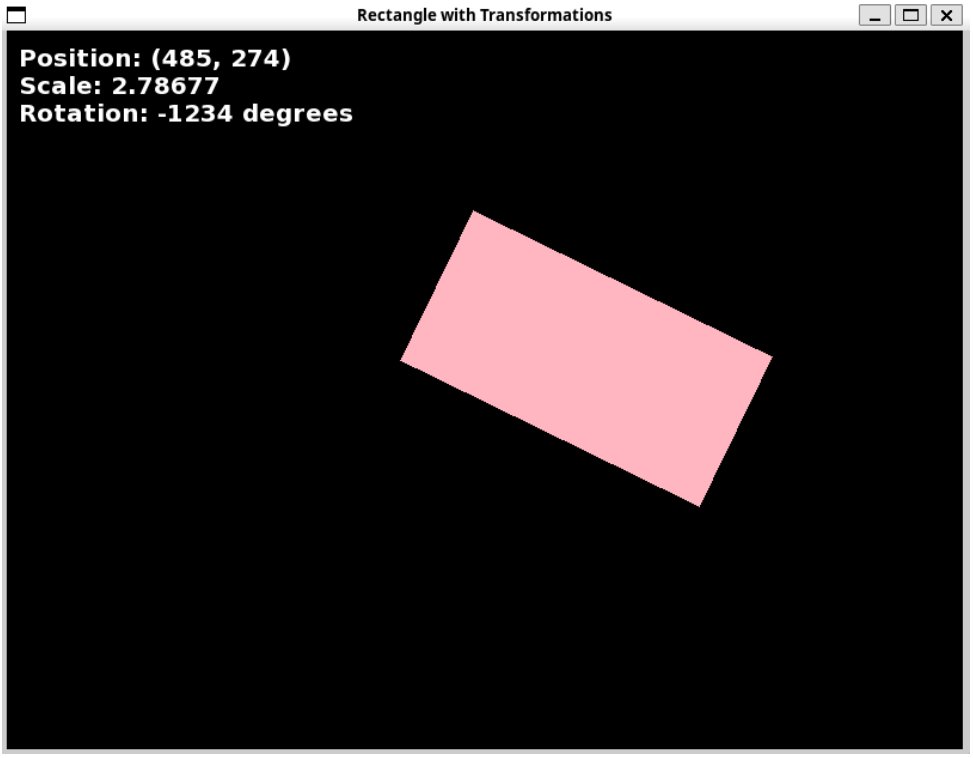

Рис. 3: прямоугольник в движении

Результаты работы программы

В результате работы программы был успешно реализован графический интерфейс для отображения прямоугольника с возможностью управления его положением, масштабом и углом поворота. Пользовательские действия обрабатывались корректно, а изменения отображались в реальном времени.

Основные функции программы

Программа предоставляет следующие функциональные возможности:

- **Перемещение:** Прямоугольник можно перемещать в пределах окна с помощью клавиш:
  - **W** — перемещение вверх.
  - **A** — перемещение влево.
  - **S** — перемещение вниз.
  - **D** — перемещение вправо.
- **Вращение:**
  - **Q** — вращение против часовой стрелки.
  - **E** — вращение по часовой стрелке.
- **Масштабирование:**
  - **Z** — уменьшение масштаба.
  - **X** — увеличение масштаба.
- **Цветовая индикация:** Цвет прямоугольника меняется в зависимости от направления трансформации:
  - Движение вверх — **розовый (255, 182, 193)**.
  - Движение вниз — **зелёный (144, 238, 144)**.
  - Движение влево — **голубой (173, 216, 230)**.
  - Движение вправо — **жёлтый (255, 255, 224)**.
  - Вращение — **оранжевый (255, 255, 120) или светло-зелёный (120, 255, 100)**.

Графический интерфейс

- Прямоугольник отрисован в окне размером 800×600 пикселей.
- В левом верхнем углу окна отображается текстовая информация:
  - Текущая позиция прямоугольника в формате (x, y).
  - Текущий масштаб.
  - Текущий угол поворота в градусах.
- Графика обработана через OpenGL с использованием ортографической проекции.

Пример работы программы

1. Изначальная позиция прямоугольника: $(400, 300)$, масштаб: 1.0, угол поворота: $0°$.
2. При нажатии клавиши **W**:
   - Позиция изменилась на $(400, 299)$.
   - Цвет прямоугольника стал розовым (`RGB(255, 182, 193)`).
3. При одновременном нажатии клавиш **A** и **Q**:
   - Позиция изменилась на $(399, 299)$.
   - Угол поворота изменился на $-1°$.
   - Цвет прямоугольника стал голубым (`RGB(173, 216, 230)`).
4. При нажатии клавиши **Z**:
   - Масштаб уменьшился до 0.99.
   - Цвет остался неизменным.
5. При нажатии клавиши **X**:
   - Масштаб увеличился до 1.0.
   - Цвет остался неизменным.

Тестирование работы программы

- **Тестирование трансформаций:** Все трансформации были протестированы. Они выполняются корректно и отображаются в реальном времени.
- **Проверка пользовательского ввода:** Одновременное нажатие нескольких клавиш обрабатывается правильно.
- **Производительность:** Программа стабильно работает при частоте 60 кадров в секунду.
- **Устойчивость:** Исключений и ошибок во время работы программы не обнаружено.

Выводы

Реализованная программа демонстрирует основные принципы работы с графикой в SFML и OpenGL. Она успешно выполняет поставленные задачи и обеспечивает интуитивное управление. Цветовая индикация и текстовая информация помогают пользователю отслеживать текущие параметры прямоугольника.