Технически университет - Варна

Проект
по
“Графични системи”

Студенти:
Даниел Гуцов , 21621649 , 1 група, СИТ, 3ти курс
и Божидар Иванов, 21621624, 2 група, СИТ, 3ти курс

Задание: Да се създаде приложение представящо работата на ветрогенератор, който се отдалечава.

Разработка:

Създава приложение в което има прозорец, който изобразява анимация на вятърна турбина на зелена поляна. Анимацията включва създаване на илюзия за въртене на перките и отдалечаване от турбината с цел използване на два типа интерполоция: ротация и мащабиране.

Програмата е написана на C++ във Visual Studio 2022 и е използвана библиотеката SFML(Simple and Fast Multimedia Library), използвана за олеснение на разработката на видео игри. Избрана е тази библиотека поради факта, че е често използвана, поддържана и наличието на лесно достъпни материали в интернет.

Библиотеката SFML позволява генератора да се изобрази като един обект, но за по-лесна манипулация са използвани няколко обекта: перки, стълб и генератор.

При рисуването на обектите е важно да се следи реда в който се прави, защото последно нарисуваният обект винаги ще покрива предходните. Пример: Ако на един прозорец нарисуваме първо кръг “А” с диаметър 20 пиксела с местоположение центъра на прозореца и след това кръг “Б” с диаметър 40 пиксела и при създаването на обектите е избран цвят за запълване, то кръга “А” няма да може да се види, защото “Б” го покрива.

За да се създаде илюзията за небе фонът е сменен на небесно синьо, а за да не изглежда, че лети в пространството е създаден обект поляна.

Няма входни параметри, всички нужни данни за изобразяването на обектите се задават в сорс кода.

Приложението се изпълнява от една програмата, която създава прозорец, нужните обекти и събития за осъществяването на анимацията използвайки следните команди:

Създаване на прозореца:

RenderWindow име_на_прозореца(sf::VideoMode(дължина в пиксели,височина в пиксели), “име на прозореца”)

Създаване обектите:
RectangleShape име_на_обекта;-квадратна поляна

RectangleShape plain;

При създаването на перките и стълба на турбината се използва команда за създаване на обект от полигони(триъгълници): **ConvexShape** име_на_обекта , следващата стъпка използва метода за задаване на броя нужни точки за построяване на полигоните име_на_обекта.**setPointCount**(брой на нужните точки) и след това се

задават координати на точките с име_на_обекта.**setPoint(**индекс, sf::Vector2f(координати по X, координати по Y)), като винаги се започва с индекс 0:

За стълба трапец:

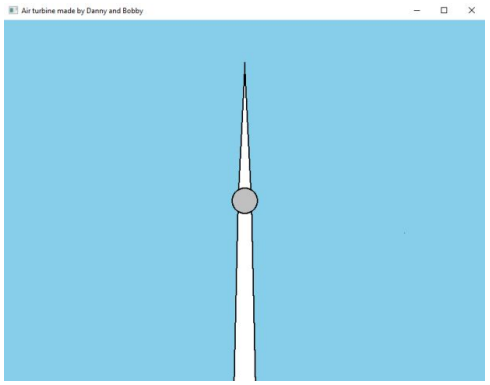
```
sf::ConvexShape trapezoid;

trapezoid.setPointCount(4);
trapezoid.setPoint(0, sf::Vector2f(-trapezoidBottomBase / 2, 0));
```

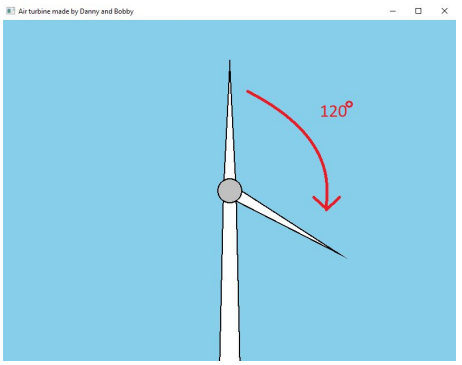
Всяка перка е триъгълник:

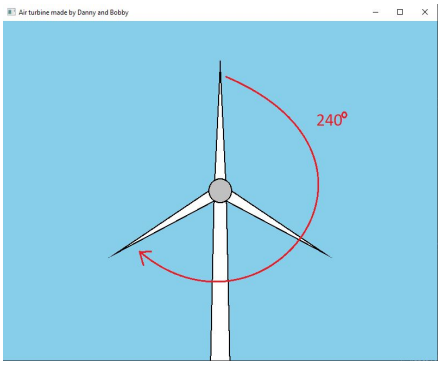
```
sf::ConvexShape triangles[numTriangles];

triangles[i].setPointCount(3);
triangles[i].setPoint(0, sf::Vector2f(-base / 2, -70));
```

Формула за намиране на ъгъла на ротация за конкретен триъгълник:
i=индекс на тригълника
Ъгъл на ротация=i * (360 / Брой на триъгълниците).
За да са равномерно поставени перките на турбината е използван следният цикъл:

```
const float base = 20.0f;
const float height = 6 * base;

for (int i = 0; i < numTriangles; i++) {
   triangles[i].setPointCount(3);
       triangles[i].setPoint(0, sf::Vector2f(-base / 2, height));
   triangles[i].setPoint(1, sf::Vector2f(base / 2, height));
   triangles[i].setPoint(2, sf::Vector2f(0, -height));
   triangles[i].setFillColor(sf::Color::White);
   triangles[i].setOutlineColor(sf::Color::Black);
       triangles[i].setOutlineThickness(2.0f);

   triangles[i].setOrigin(0, height);
   triangles[i].setPosition(centerX, centerY);

   float angle = i * (360.0f / numTriangles);

   triangles[i].rotate(angle);
}
```


Air turbine made by Danny and Bobby


Air turbine made by Danny and Bobby
120°


Air turbine made by Danny and Bobby
240°

Генераторът е изобразен с един кръг с командата
CircleShape име_на_обекта(радиус) :

```
sf::CircleShape circle(20);
```

Местоположение в даден прозорец се задава с метода **setPosition**(координати по X, координати по Y); като това се прави спрямо точката за транслации на даден обект(по подразбиране се пада точката в горния ляв ъгъл на всеки обект) .

Точката за транслации може да се промени с метода **setOrigin**(кординати по X, координати по Y)

Обектът се изпълва с цвят като използва метода **setFillColor**(sf::Color(192, 192, 192))-за задаване на конкретен цвят ИЛИ (sf::Color::Black/Blue/Yellow/…)

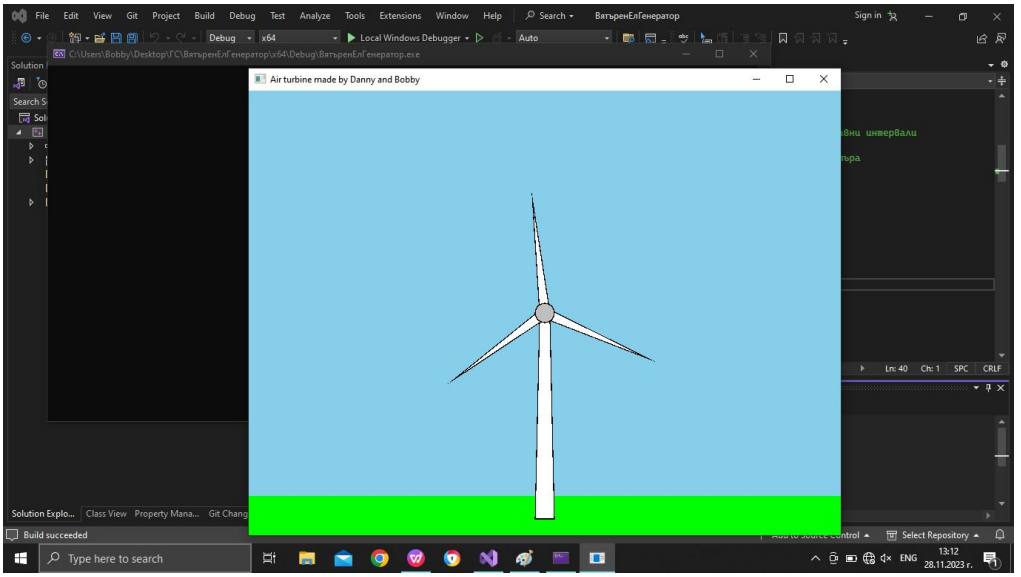
Обектът се очертава като използва метода **setOutlineColor**(sf::Color::Black)-за цвят на линията И setOutlineThickness(2.0f)-за дебелина

За да се види даден обект той трябва да се нарисува и прозорецът, върху който е нарисуван, трябва да го покаже.Примери:

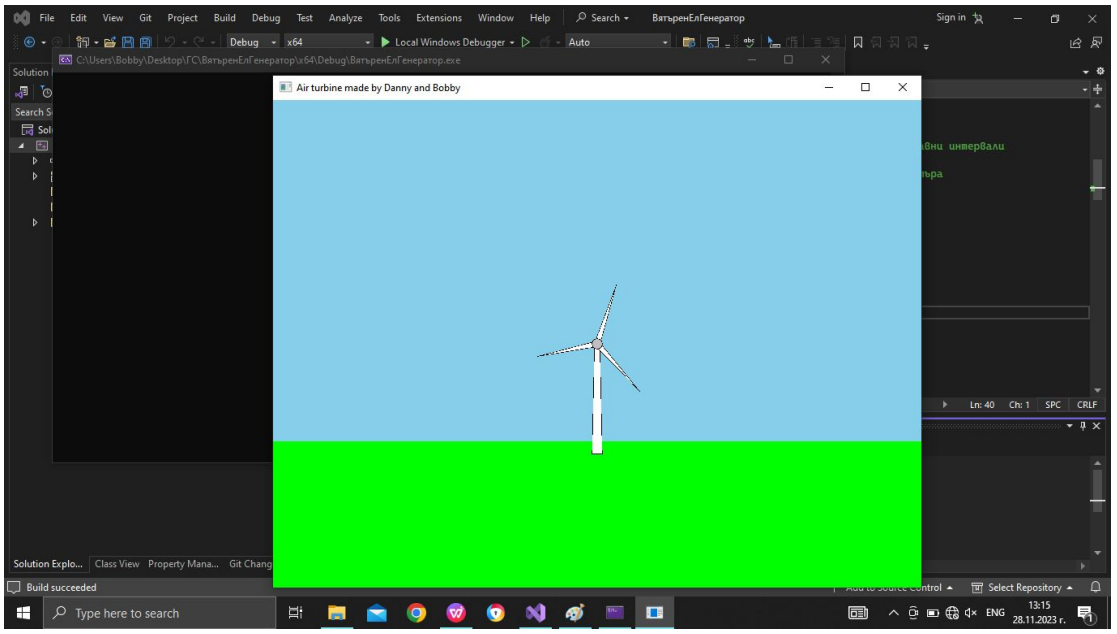
```
прозорец.draw(обект);
прозорец.display();
window.draw(circle);
window.display();
```

Тестване:

В началото на анимацията:



В края на анимацията:



Приложение сорс код

```
#include <SFML/Graphics.hpp>
#include <cmath>

int main()
{
   sf::RenderWindow window(sf::VideoMode(800, 600), "Air turbine made by Danny and Bobby");//създаване на празореца за анимация

   const int numTriangles = 3;//задаване броя на перките
   sf::ConvexShape triangles[numTriangles];//създаване на 3 еднакви триъгъкника

   const float centerX = window.getSize().x / 2.0f;//намиране на центъра на прозореца по Ox
   const float centerY = window.getSize().y / 2.0f;//намиране на центъра на прозореца по Oy
   const float base = 20.0f; // Дължина на основата на перките
   const float height = 6 * base; // Дължина (височината на триъгълниците) на перките

   for (int i = 0; i < numTriangles; i++) {
      triangles[i].setPointCount(3); // Три точки за триъгълник
      triangles[i].setPoint(0, sf::Vector2f(-base / 2, height)); // основа на перките
      triangles[i].setPoint(1, sf::Vector2f(base / 2, height)); //
      triangles[i].setPoint(2, sf::Vector2f(0, -height));//връх на перките
      triangles[i].setFillColor(sf::Color::White);    //цвят на перките
      triangles[i].setOutlineColor(sf::Color::Black); // цвят на линия за очертаване на перките
      triangles[i].setOutlineThickness(2.0f);         // дебелина на линията заочертаване

      triangles[i].setOrigin(0, height);// Точка за интерполация
      triangles[i].setPosition(centerX, centerY);//позиция на първия триъгълник

      float angle = i * (360.0f / numTriangles); // Разпределение на триъгълниците на равни интервали

      triangles[i].rotate(angle);// Разпределяне на триъгълниците равномерно около центъра
   }


   //създаване на поляната
   sf::RectangleShape plain;
   plain.setSize(sf::Vector2f(10000, 5000));
   plain.setFillColor(sf::Color::Green);
   plain.setOrigin(plain.getSize().x / 2, -400);
   plain.setPosition(centerX, centerY);

   // Създаване на стълба на генератора
   sf::ConvexShape trapezoid;
   const float trapezoidTopBase = 20.0f;
   const float trapezoidBottomBase = 2 * 20.0f;
   const float trapezoidHeight = 450;

   trapezoid.setPointCount(4);
   trapezoid.setPoint(0, sf::Vector2f(-trapezoidBottomBase / 2, 0));
   trapezoid.setPoint(1, sf::Vector2f(trapezoidBottomBase / 2, 0));
   trapezoid.setPoint(2, sf::Vector2f(trapezoidTopBase / 2, -trapezoidHeight));
   trapezoid.setPoint(3, sf::Vector2f(-trapezoidTopBase / 2, -trapezoidHeight));
   trapezoid.setFillColor(sf::Color::White);//цвят на стълба
   trapezoid.setOutlineColor(sf::Color::Black);//цвят на линия за очерататване
   trapezoid.setOutlineThickness(2.0f);//дебелина на линията за очертаване

   trapezoid.setOrigin(0, -trapezoidHeight); // Точка за интерполация
   trapezoid.setPosition(centerX, centerY); // Set position to center
```

```
    // Създаване на генератора/главата държаща перките
    sf::CircleShape circle(20); // задава се радиус при създаване на кръга
    circle.setFillColor(sf::Color(192, 192, 192));//цвят на кръга
    circle.setOutlineColor(sf::Color::Black);//цвят на линия за очерататване
    circle.setOutlineThickness(2.0f);//дебелина на линията за очертаване
    circle.setOrigin(20, 20); // Точка за интерполация
    circle.setPosition(centerX, centerY); // Поставете кръгчето в центъра на екрана

    float scaleFactor = 1.0f;

    while (window.isOpen())
    {
        sf::Event event;
        while (window.pollEvent(event))
        {
            if (event.type == sf::Event::Closed)
                window.close();
        }

        //рисуване на нов кадър
        window.draw(plain);
        window.draw(trapezoid);
        for (int i = 0; i < numTriangles; i++) {

            window.draw(triangles[i]);
        }
        window.draw(circle);
        window.display();

        //изчистване на прозорецас
        window.clear(sf::Color(135, 206, 235));

        // Смаляване на цялата картина
        scaleFactor -= 0.0001f; // скорост на мащабиране
        if (scaleFactor < 0.3f)
            scaleFactor = 0.3f; // задава лимит на смаляване


        // прилага мащабирането на върху всички елементи
        for (int i = 0; i < numTriangles; i++)
        triangles[i].setScale(scaleFactor, scaleFactor);


        plain.setScale(scaleFactor, scaleFactor);
        trapezoid.setScale(scaleFactor, scaleFactor);
        circle.setScale(scaleFactor, scaleFactor);

        const float rotationSpeed = 0.1f; // Скорост на въртене

        //Въртене на перките/ротация
        for (int i = 0; i < numTriangles; i++) {
            triangles[i].rotate(rotationSpeed);
        }
    }

    return 0;
}
```

Материали

Сайт на библиотеката:
https://www.sfml-dev.org/download/sfml/2.6.0/

Инсталация на библиотеката за Visual Studio:
https://www.sfml-dev.org/tutorials/2.6/start-vc.php

Материали за обучение:
https://www.sfml-dev.org/tutorials/2.6/

Видео материли за начинаещи:
https://www.youtube.com/watch?v=axIgxBQVBg0&list=PL21OsoBLPpMOO6zyVlxZ4S4hwkY_SLRW9