МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №3

по дисциплине: Компьютерная графика

тема: «Аффинные преобразования на плоскости»

Выполнил: ст. группы ПВ-223

Дмитриев Андрей Александрович

Проверил:

Осипов Олег Васильевич

Белгород 2024 г.

**Цель работы:** получение навыков выполнения аффинных преобразований на плоскости и создание графического приложения на языке C++ для создания простейшей анимации.

**Порядок выполнения работы**

1. Разработать алгоритм и составить программу для построения на экране изображения в соответствии с номером варианта. В качестве исходных данных взять указанные в таблице №1.

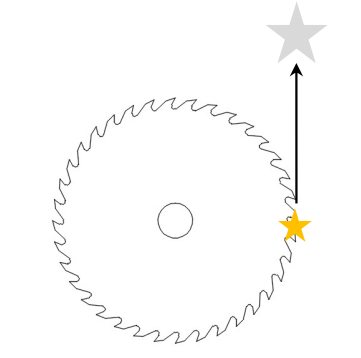
**Требования к программе**

1. Разработать модуль для выполнения аффинных преобразований на плоскости с помощью матриц. В модуле должны быть реализованы перегруженные операции действия с матрицами (умножение), с векторами и матрицами (умножение вектора-строки на матрицу), конструкторы различных матриц (переноса, масштабирования, переноса, отражения).
2. Раскрасить (залить) примитивы (круги, многоугольники и др.) по собственному усмотрению.

**Содержание отчѐта**

1. Название темы.
2. Цель работы.
3. Постановка задачи.
4. Вывод необходимых геометрических формул для построения изображения и расчѐта цвета. Также указать, какие матрицы используются и в какой последовательности они умножаются для реализации анимации.
5. Текст программы для рисования фигур.
6. Результат работы программы (снимки экрана).
7. Выводы о проделанной работе.

Вариант 2:



Ход работы:

Пила состоит из 3-х частей: внутреннего круга, внешнего круга и зубьев.

Зубья состоят из 2-х треугольников, точки выбраны путём отклонения от центра мировых координат и размножены тем же отклонением. Вращение задаётся увеличением global\_angle.

Листинг кода:

void drawSaw(  
 Frame &frame,  
 Matrix &WS,  
 float xC, float yC,  
 float radius,  
 float maxCoord,  
 float globalAngle  
) {  
 globalAngle = globalAngle \* 180.0f / M\_PI\*2;  
  
 float maxAngle = 360.0f;  
 float rotateAngle = 12.0f;  
 float angle = 0.0f;  
 while (angle < maxAngle) {  
 // точки для рисования зубьев  
 Vector p5 = **{**0.0f, maxCoord, 1.0f**}**;  
 p5 = p5

\* Matrix::rotation(10 + globalAngle + angle)\*WS;  
  
 Vector p4 = **{**0.0f, maxCoord - 1, 1.0f**}**;  
 p4 = p4

\* Matrix::rotation(358 + globalAngle + angle)\*WS;  
  
 Vector p3 = **{**0.0f, maxCoord - 2, 1.0f**}**;  
 p3 = p3

\* Matrix::rotation(3 + globalAngle + angle)\*WS;  
  
 Vector p2 = **{**0.0f, maxCoord - 3, 1.0f**}**;  
 p2 = p2

\* Matrix::rotation(10 + globalAngle + angle)\*WS;  
  
 Vector p1 = **{**0.0f, maxCoord - 3, 1.0f**}**;  
 p1 = p1

\* Matrix::rotation(350 + globalAngle + angle)\*WS;  
  
 // рисуем зубья  
 frame.Triangle(  
 p5.x() + xC, p5.y() + yC,  
 p4.x() + xC, p4.y() + yC,  
 p3.x() + xC, p3.y() + yC,  
 sawColor  
 );  
 frame.Triangle(  
 p4.x() + xC, p4.y() + yC,  
 p2.x() + xC, p2.y() + yC,  
 p1.x() + xC, p1.y() + yC,  
 sawColor  
 );  
  
 // отклоняем угол для рисования следующего зуба  
 angle += rotateAngle;  
 }  
 // рисуем внешний круг  
 frame.Circle(  
 xC, yC,  
 radius \* 0.84,  
 sawColor  
 );  
 // рисуем внутренний круг  
 frame.Circle(  
 xC, yC,  
 radius \* 0.3,  
 sawCenterColor  
 );  
}

Искры сделаны из фигуры «Звезда», правая сторона фигуры выложена на мировой координатной плоскости и отражена по горизонтали. Перед тем как позиционировать звезду координатная плоскость сначала рандомно поворачивается, затем удаляется на расстояние, зависящее от глобального поворота угла. После искра поворачивается в зависимости от глобального угла и увеличивается. Чтобы сделать затухание используем барицентрическую интерполяцию, будет брать один цвет из палитры и применять на фигуру.

Пример работы программы:

int currH = 0;  
float currAngle = 0;  
  
void drawStars(  
 Frame &frame,  
 Matrix WS,  
 float xC, float yC,  
 float maxCoord,  
 float globalAngle  
) {  
 globalAngle = globalAngle \* 180.0f / M\_PI;  
  
 // получаем переменную для просчёта удаления искр  
 int h = (int) globalAngle % (int) (maxCoord \* 3);  
 // условие для повторного воспроизведения  
 if (currH > h) currAngle = rand() % 40 - 10;  
 currH = h;  
  
 // изменяем положение координатной плоскости  
 Matrix toss =  
 Matrix::rotation(currAngle)  
 \* Matrix::transform(0, h)  
 \* Matrix::rotation(globalAngle)  
 \* Matrix::scale(h \* 0.025f);  
  
 // применяем матрицу на точки для построения звезды  
 Vector p1 = **{**0.0f, maxCoord \* 3 / 4, 1.0f**}**;  
 p1 = p1 \* toss \* WS;  
 Vector p2 = **{**0.0f, maxCoord \* 1 / 4 - 1.0f, 1.0f**}**;  
 p2 = p2 \* toss \* WS;  
 Vector p3 = **{**0.0f, -maxCoord \* 1 / 4 - 1.4f, 1.0f**}**;  
 p3 = p3 \* toss \* WS;  
 Vector p4 = **{**maxCoord / 2, -maxCoord \* 3 / 4, 1.0f**}**;  
 Vector p4\_ = **{**-maxCoord / 2, -maxCoord \* 3 / 4, 1.0f**}**;  
 p4 = p4 \* toss \* WS;  
 p4\_ = p4\_ \* toss \* WS;  
 Vector p5 = **{**maxCoord \* 3 / 4, maxCoord \* 1 / 4 - 1.0f, 1.0f**}**;  
 Vector p5\_ = **{**-maxCoord \* 3 / 4, maxCoord \* 1 / 4 - 1.0f, 1.0f**}**;  
 p5 = p5 \* toss \* WS;  
 p5\_ = p5\_ \* toss \* WS;  
  
 // интерополируем цвета  
 BarycentricInterpolator interpolator(  
 xC, yC + 10,  
 0, 0,  
 frame.width - 1, 0,  
 {255, 255, 0},  
 {180, 180, 180},  
 {180, 180, 180}  
 );  
 // получаем цвет из интерполятора  
 COLOR starColor = interpolator.color(p3.x() + xC, p3.y() + yC);  
  
 // рисуем звезду  
 frame.Triangle(  
 p5.x() + xC, p5.y() + yC,  
 p2.x() + xC, p2.y() + yC,  
 p3.x() + xC, p3.y() + yC,  
 starColor  
 );  
 frame.Triangle(  
 p4.x() + xC, p4.y() + yC,  
 p3.x() + xC, p3.y() + yC,  
 p1.x() + xC, p1.y() + yC,  
 starColor  
 );  
 frame.Triangle(  
 p5\_.x() + xC, p5\_.y() + yC,  
 p2.x() + xC, p2.y() + yC,  
 p3.x() + xC, p3.y() + yC,  
 starColor  
 );  
 frame.Triangle(  
 p4\_.x() + xC, p4\_.y() + yC,  
 p3.x() + xC, p3.y() + yC,  
 p1.x() + xC, p1.y() + yC,  
 starColor  
 );  
}

Код полной отрисовки выглядит следующим образом:

void Draw(Frame &frame) {  
 int W = frame.width, H = frame.height;  
  
 // Размер рисунка возьмём меньше (7 / 8), чтобы он не

// касался границ экрана  
 float a = 7.0 / 8.0 \* ((W < H) ? W - 1 : H - 1) / 2;  
  
 Assets::drawChessBg(frame);  
  
 Matrix WS = Matrix::WorldToScreen(  
 W / 2 - a / 2, H / 2 - a / 2,  
 W / 2 + a / 2, H / 2 + a / 2,  
 -15, -15, 15, 15  
 );  
  
 float globalAngle = global\_angle;  
  
 drawSaw(  
 frame,  
 WS,  
 W / 2, H / 2,  
 a / 2,  
 15,  
 -globalAngle  
 );  
  
 drawStars(  
 frame,  
 WS,  
 W \* 5 / 8.2, H / 2,  
 15,  
 globalAngle  
 );  
 }

**Вывод:** в ходе работы получены навыки выполнения аффинных преобразований на плоскости и создания графического приложения на языке C++ для создания простейшей анимации.