

float dot(vec4 v1, vec4 v2) // Функции скалярного произведения:

{  
vec4 tmp = v1 \* v2; // вычисляем произведения соответствующих координат  
return tmp.x + tmp.y + tmp.z + tmp.a; // и возвращаем их сумму  
}  
  
float dot(vec3 v1, vec3 v2) {  
vec3 tmp = v1 \* v2; // вычисляем произведения соответствующих координат  
return tmp.x + tmp.y + tmp.z; // и возвращаем их сумму  
}  
  
При выполнении операции v1 \* v2 вызывается операция перегрузки оператора, соответственно для четырехмерного/двумерного вектора. Она вычисляет результат покоординатного произведения векторов  
const vec4 operator\*(const vec4& v) {  
return vec4(\*this) \*= v; // делаем временную копию текущего объекта,  
// которую домножаем на данный вектор,  
// и возвращаем ее как результат  
  
}  
const vec2 operator\*(const vec2& v) {  
return vec2(\*this) \*= v;  
}  
В этой операции вызывается еще один перегруженный оператор, который сохраняет соответствующие координаты и возвращает полученный объект объект  
vec4& operator\*=(const vec4& v) {  
x \*= v.x;  
y \*= v.y;  
z \*= v.z;  
a \*= v.a;  
return \*this;  
  
}  
vec2& operator\*=(const vec2& v) {  
x \*= v.x;  
y \*= v.y;  
return \*this;  
}

https://sun7-6.userapi.com/Z9MB7uP8EOo0XDmnANC8XVAzzGm3xrFmVjM7Hw/mnU08bqZmqI.jpg

mat4 translate(float Tx, float Ty, float Tz) {

mat4 \*res = new mat4(1.f); // создали единичную матрицу

(\*res)[0][3] = Tx; // поменяли

(\*res)[1][3] = Ty; // значения в последнем столбце

(\*res)[2][3] = Tz; //

return \*res;

}

mat4 scale(float Sx, float Sy, float Sz) {

mat4 \*res = new mat4(1.f); // создали единичную матрицу

(\*res)[0][0] = Sx; // поменяли

(\*res)[1][1] = Sy; // значения на главной диагонали

(\*res)[2][2] = Sz; //

Ниже представлен фрагмен кода, по нажатию которого на клавишу “A” происходит сдвиг графика вправо на 1 пиксель

Keys::A: T = translate(-V\_work.x / Wx, 0.f) \* T; // сдвиг графика вправо на один пиксель

Ниже представлен фрагмент кода, где представлено масштабирование на форме

case Keys::Z: 2

T = translate(-centerX, -centerY, -centerZ) \* T; // перенос начала координат в центр

T = scale(1.1, 1.1, 1.1) \* T; // масштабирование относительно начала координат

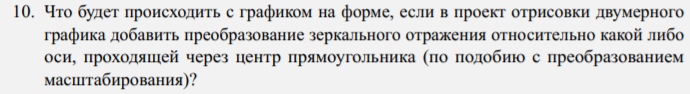
T = translate(centerX, centerY, centerZ) \* T; // возврат позиции начала координат

График будет зеркально отображаться относительно центра нашей формы

case Keys::M:  
  
T = translate(-centerX, -centerY) \* T; // перенос начала координат в центр  
  
T = mirrorX(T); // масштабирование относительно начала координат  
  
T = translate(centerX, centerY) \* T; // возврат позиции начала координат  
  
break;

https://sun9-61.userapi.com/c205820/v205820182/ee965/glcUprCMjEk.jpg

https://i.imgur.com/kYmv0rR.png

tmpEnd – это временная переменная для сохранения в нее конечного значения отрезка.

В некоторых строках цикла требуется исходное значение координат точки end — до отсечения. Поэтому, перед добавленным условным оператором сохраним значение end во временной переменной tmpEnd.

https://sun9-5.userapi.com/c857036/v857036820/10082f/11ztTri9pEE.jpg

Что произойдет с проектом отрисовки трехмерного графика, если в нем снять ограничения значений переменных Wx\_part и Wy\_part ?  
Если убрать верхнюю границу – сетка выходит за очерченную область и график не помещается, становится слишком большой, если убрать нижнюю границу – сетка превратиться в линию и график исчезнет.