Компьютерная графика

лекция 3

Аффи́нное преобразование — отображение $f : \mathbb{R}^n \to \mathbb{R}^n$, которое можно записать в виде

$$f(x) = M \cdot x + v,$$

где M — обратимая матрица и $v \in \mathbb{R}^n$.

Иначе говоря, преобразование называется аффинным, если его можно получить следующим образом:

1. Выбрать «новый» базис пространства с «новым» началом координат v;
2. Каждой точке x пространства поставить в соответствие точку $f(x)$, имеющую те же координаты относительно «новой» системы координат, что и x в «старой».

Преобразование плоскости называется **аффинным**, если

оно взаимно однозначно;

образом любой прямой является прямая.

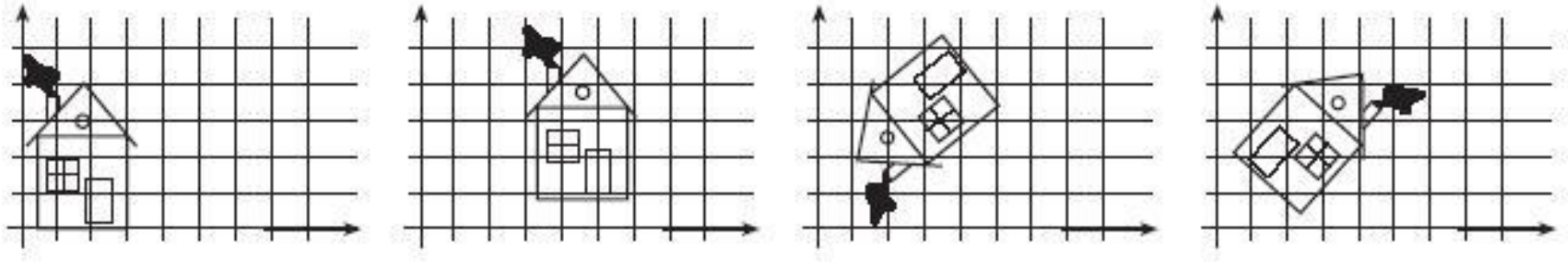
Преобразование называется **взаимно однозначным**, если

разные точки переходят в разные;

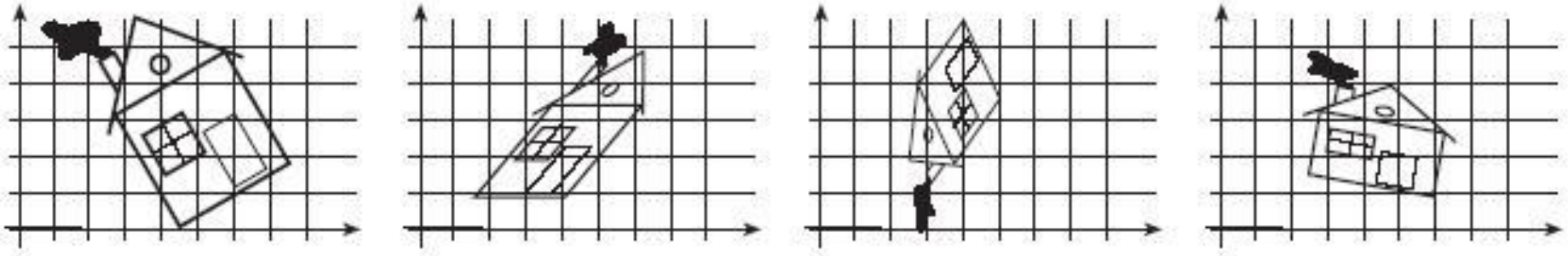
в каждую точку переходит какая-то точка.

Однородные координаты

Однородные координаты — координаты, обладающие тем свойством, что определяемый ими объект не меняется при умножении всех координат на одно и то же число.



Примеры движений.



Примеры аффинных преобразований

Простейшие преобразования

Сжатие/растяжение

Поворот

Параллельный перенос

Отражение

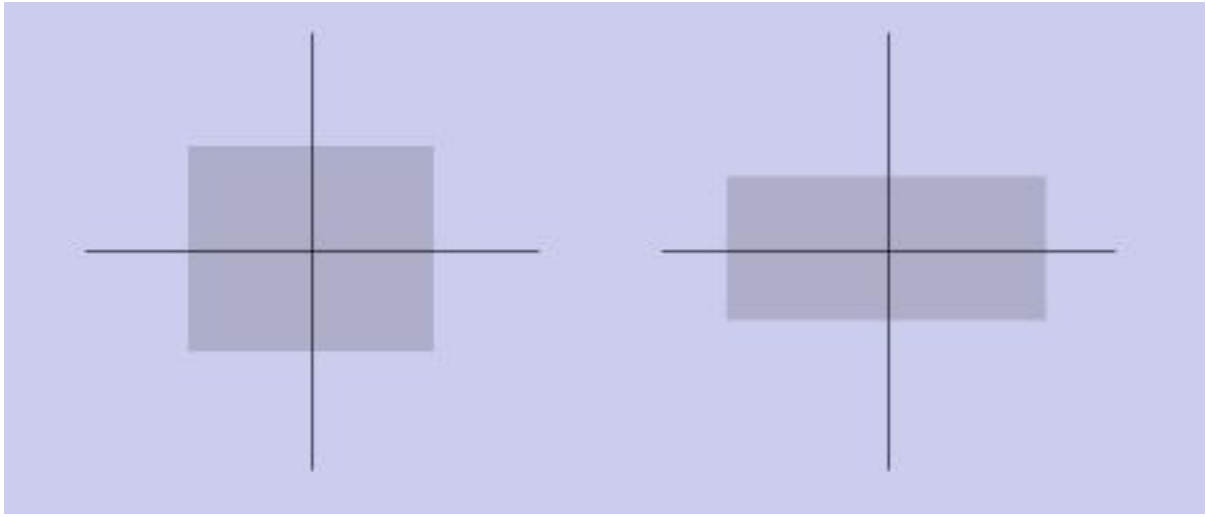
1) каждое из приведенный преобразований имеет простой и наглядный геометрический смысл

2) любое преобразование всегда можно представить как последовательное исполнение простейших преобразований

Сжатие/растяжение

Это преобразование умножает соответствующие координаты точек на коэффициенты масштабирования по осям: (x, y) -> (ax * x, ay * y).
Матрица преобразования :

$$\begin{bmatrix} ax & 0 & 0 \\ 0 & ay & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



ax – растяжение по оси x,

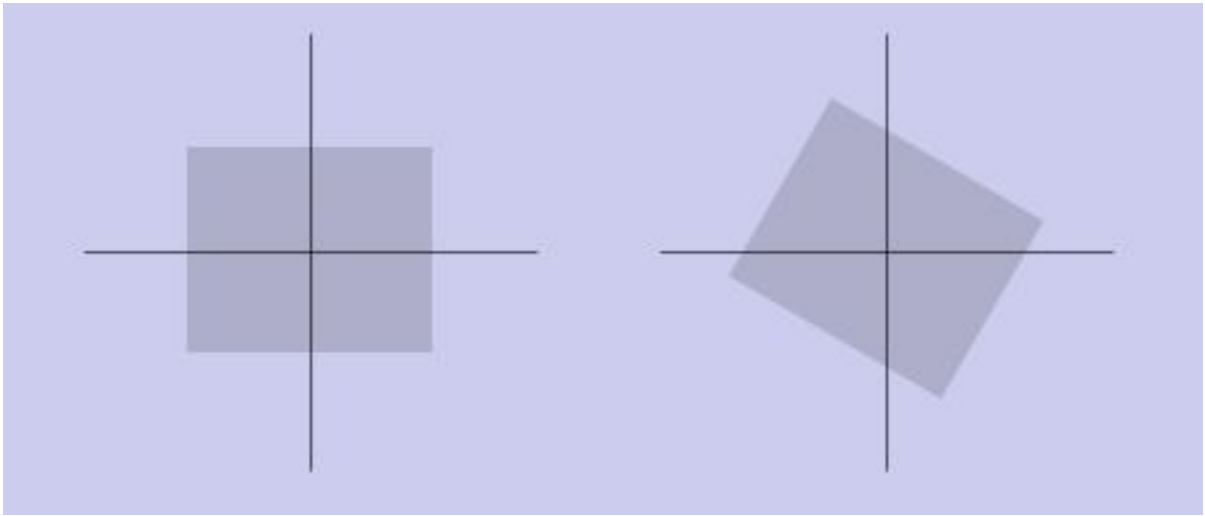
ay – растяжение по оси y.

Поворот

Матрица преобразования :

$$\begin{bmatrix} \cos(a) & \sin(a) & 0 \\ -\sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

a – угол поворота



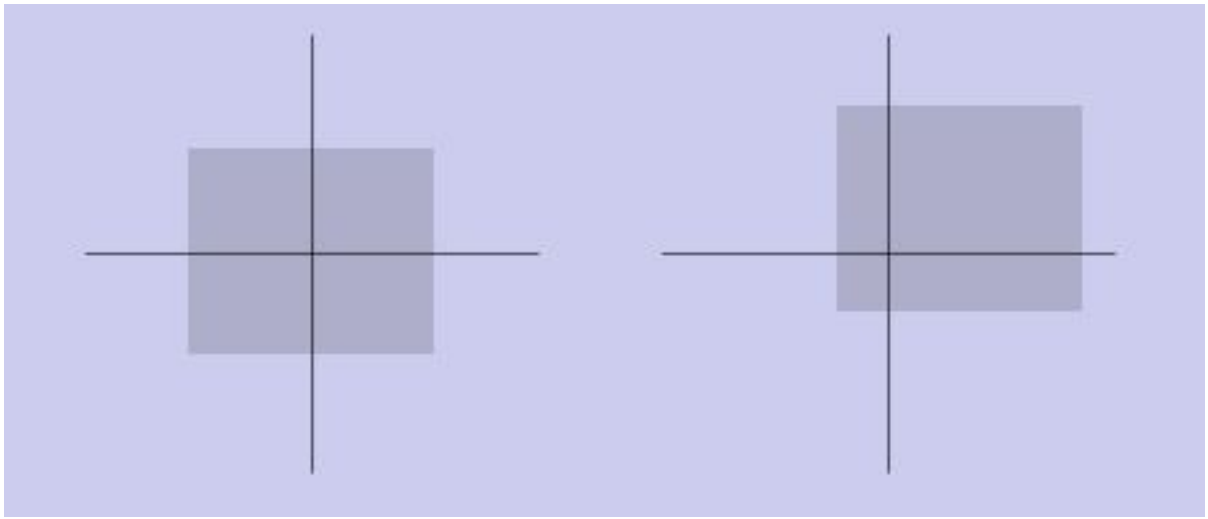
Параллельный перенос

Исходный вектор (x, y) переходит в

(x + dx, y + dy).

Матрица преобразования:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{bmatrix}$$



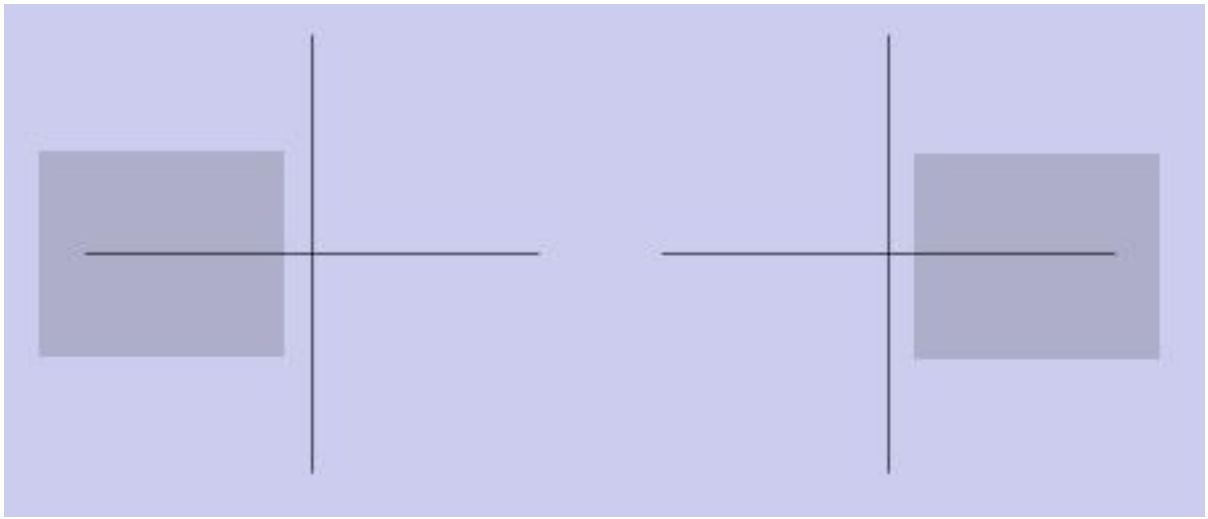
Отражение

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

относительно оси x

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

относительно оси y



Общий вид аффинного преобразования

f(x) = x * R + t, где R – обратимая матрица 2x2,
a t – произвольный вектор.

Матрица преобразования:

$$\begin{bmatrix} R_{1,1} & R_{1,2} & 0 \\ R_{2,1} & R_{2,2} & 0 \\ tx & ty & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$x' = x \cdot R_{1,1} + y \cdot R_{2,1} + tx$$

$$y' = x \cdot R_{1,2} + y \cdot R_{2,2} + ty$$

glRotate

Функция **glRotate** поворачивает текущую матрицу на заданный угол вокруг заданного вектора. Имеет следующие формы:

glRotated (GLdouble angle, GLdouble x, GLdouble y, double z)

glRotatef (GLfloat angle, GLfloat x, GLdouble y, GLfloat z)

angle - угол поворота

x, y, z – вектор, задающий ось поворота

Примеры вызова функции glRotate

glRotated(45, 1, 0, 0) – поворот на 45 градусов против часовой стрелки относительно оси x.

glRotated(90, 0, 0, -1) – поворот на 90 градусов по часовой стрелке относительно оси z.

В функции **glRotate** вычисляется матрица поворота:

$$\begin{pmatrix} & & & 0 \\ & R & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

выполняющая поворот на угол *angle* против часовой стрелки вокруг вектора, направление которого задано точкой (*x*, *y*, *z*). Эта матрица вычисляется следующим образом:

glRotate*$(a, 1, 0, 0)$:
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos a & -\sin a & 0 \\ 0 & \sin a & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

glRotate*$(a, 0, 1, 0)$:
$$\begin{bmatrix} \cos a & 0 & \sin a & 0 \\ 0 & 1 & 0 & 0 \\ -\sin a & 0 & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

glRotate*$(a, 0, 0, 1)$:
$$\begin{bmatrix} \cos a & -\sin a & 0 & 0 \\ \sin a & \cos a & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

glTranslate

Функция **glTranslate** вносит дополнительное смещение в текущую матрицу. Имеет следующие формы:

glTranslated(GLdouble x, GLdouble y, GLdouble z)

glTranslatef(GLfloat x, GLfloat y, GLfloat z)

x, y, z - координаты вектора сдвига

Функция **glTranslate** выполняет сдвиг текущей матрицы на вектор (*x*, *y*, *z*). Этот вектор сдвига используется для составления матрицы сдвига:

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

glScale

Функция **glScale** масштабирует текущую матрицу. Имеет следующие формы:

glScaled(GLdouble x, GLdouble y, GLdouble z)

glScalef(GLfloat x, GLfloat y, GLfloat z)

x, y, z - Коэффициенты масштабирования по соответствующим осям

В функции **glScale** вычисляется матрица масштабирования:

$$\begin{pmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Текущая матрица умножается на матрицу преобразования, а затем результат этого произведения записывается в текущую матрицу. Таким образом, если M — текущая матрица, а R — матрица преобразования, то матрица M будет заменена на M*R.

Если режим текущей матрицы GL_MODELVIEW , то все примитивы, нарисованные после вызова **glScale, glRotate, glTranslated**, будут соответствующим образом масштабированы

glLoadIdentity

Функция **glLoadIdentity** заменяет текущую матрицу на единичную.

glLoadIdentity ()

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

glPushMatrix, glPopMatrix

Функции **glPushMatrix** и **glPopMatrix** предназначены для помещения матриц в стек и извлечения из него.

glPushMatrix( )

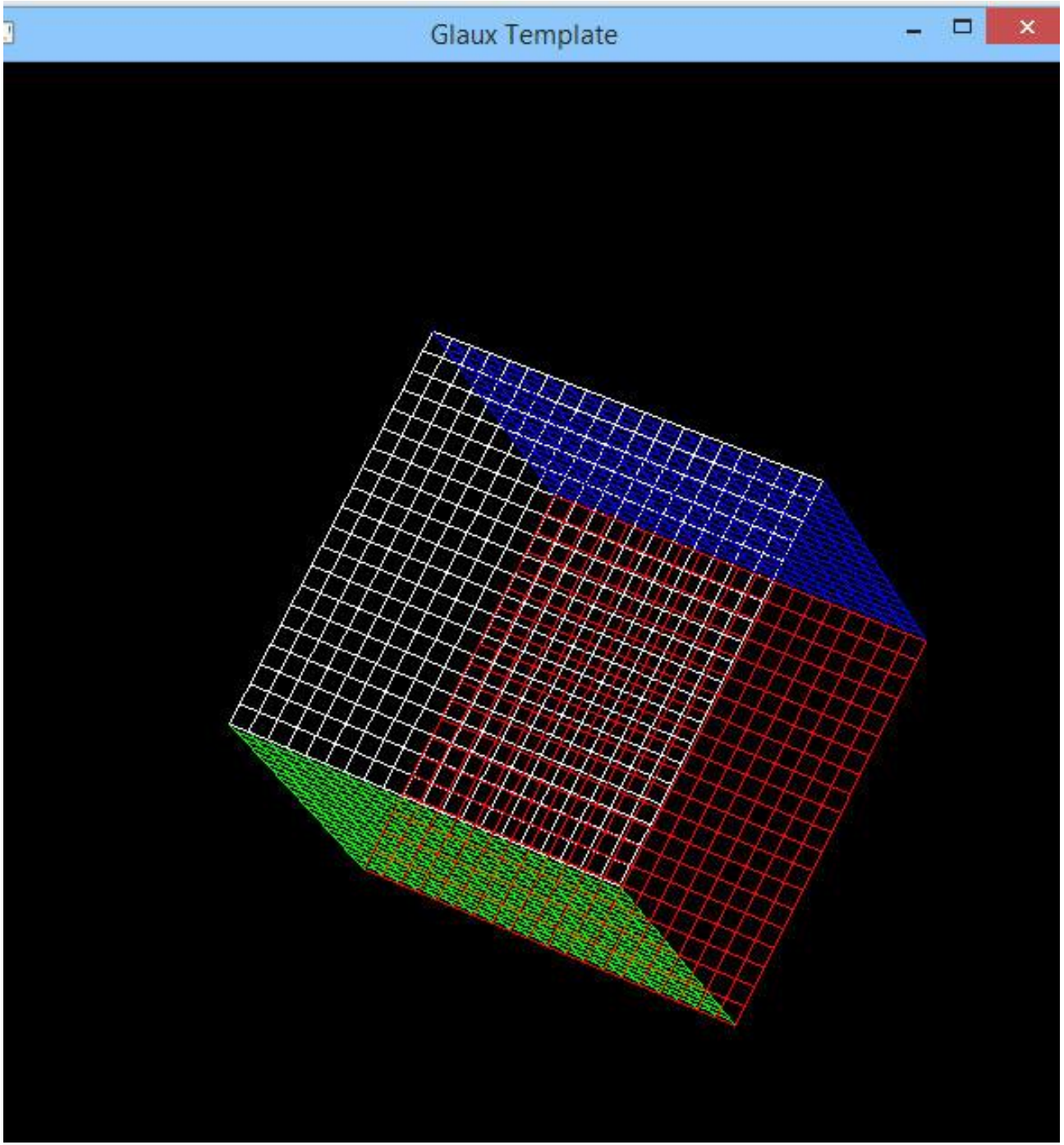
glPopMatrix( )

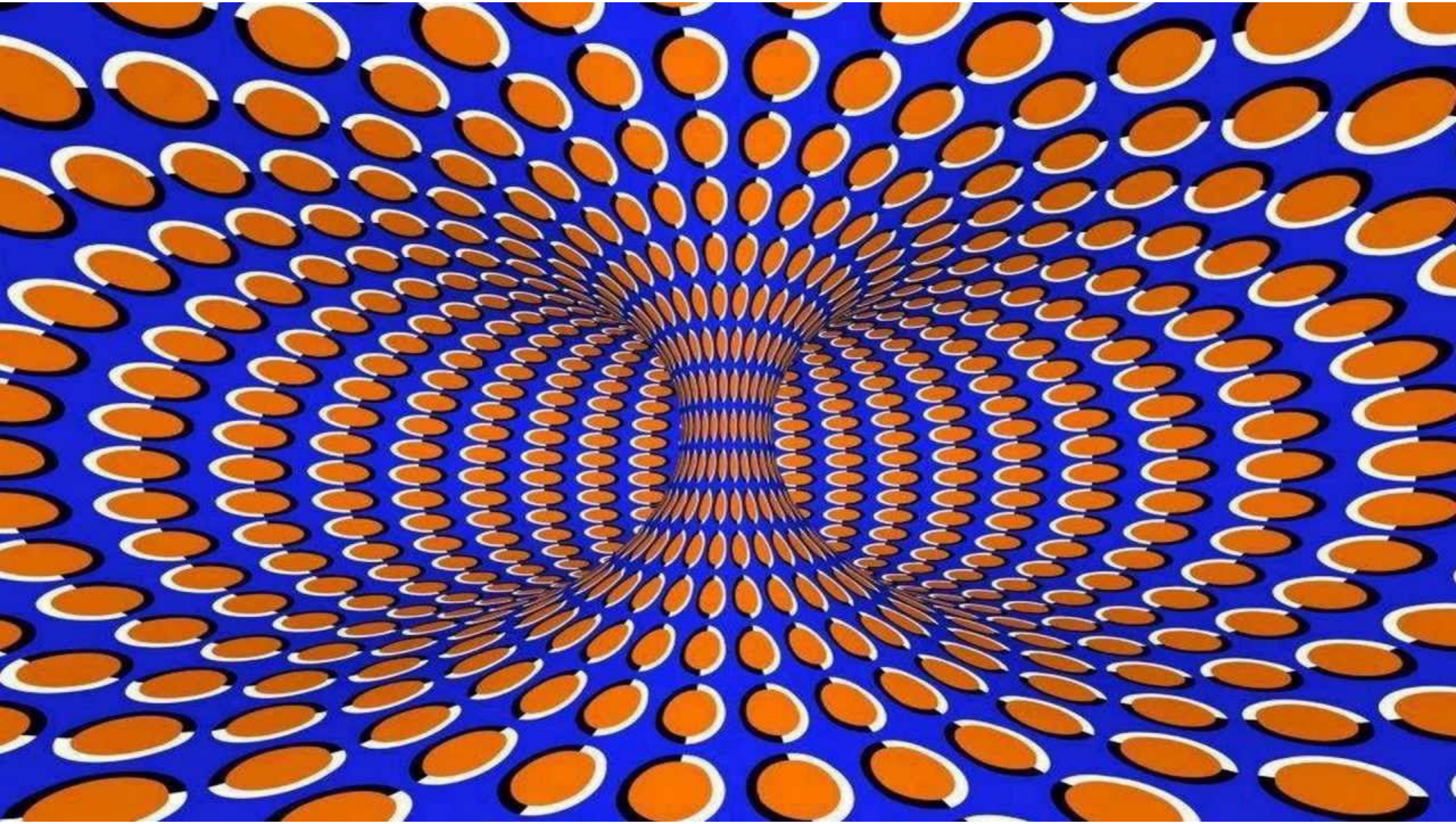
Примеры преобразований

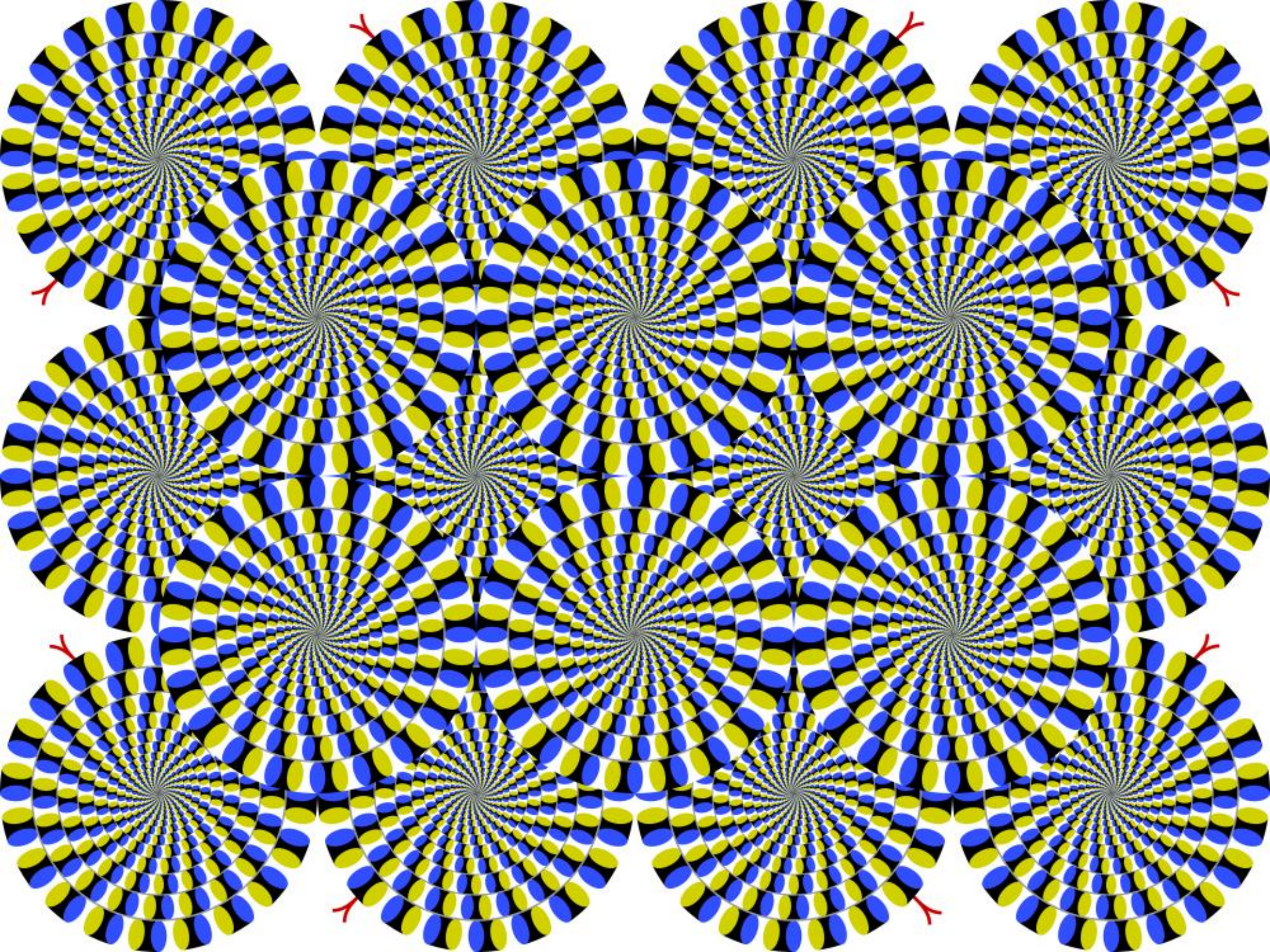
- void DrawGrid(GLdouble r, GLdouble g, GLdouble b)
- {
- GLdouble dx = 1;
- for (int i = -10; i<=10; i++)
- DrawLine(i*dx, -10, 0, i*dx, 10, 0, r, g, b);
- }

Примеры преобразований

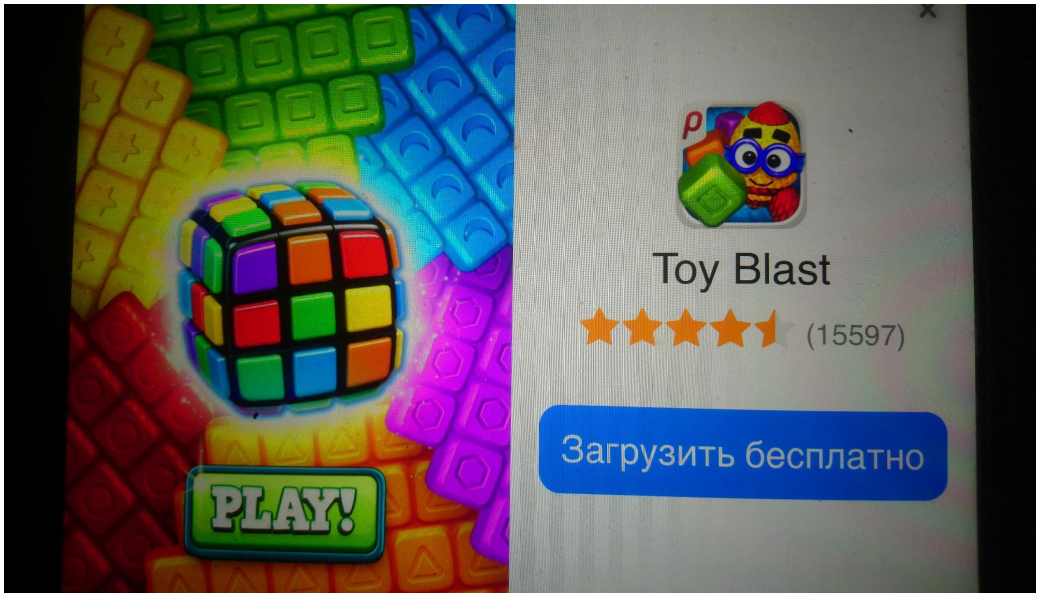
```
void Draw()
{       glTranslated(0, 0, -30);
      glRotated(GetTickCount()/10.f , 0.1, 0.2, 0.3);
      glTranslated(0, 0, -10); DrawGrid(1, 1, 1);
glRotated(90, 0, 0, 1); DrawGrid(1, 1, 1);

   glPushMatrix(); glTranslated(0, 0, -20);
   DrawGrid(1, 0, 0); glRotated(90, 0, 0, 1);
      DrawGrid(1, 0, 0);

   glPopMatrix(); glTranslated(-10, 0, -10);
   glRotated(90, 0, 1, 0); DrawGrid(0, 1, 0);
      glRotated(90, 0, 0, 1); DrawGrid(0, 1, 0);

   glTranslated(0, 0, 20); DrawGrid(0, 0, 1);
glRotated(90, 0, 0, 1); DrawGrid(0, 0, 1);
}
```


Glaux Template





AppStore 09.2018



GooglePlay 02.2021

Равномерное движение

Равномерное движение — механическое движение, при котором тело за любые равные отрезки времени проходит одинаковое расстояние.

Равномерное движение материальной точки — это движение, при котором величина скорости точки остаётся неизменной.

Расстояние, пройденное точкой за время t, задаётся в этом случае формулой L=vt.

Функция GetTickCount()

function GetTickCount: Longint;

Возвращает время, прошедшее с момента запуска системы в миллисекундах.

float x = GetTickCount()/1000.f;

x — дробное время в секундах.

Равномерное движение

Необходимые переменные:

```
float xfrom = 0;
float xto = 10;
float v = 2;
int startTime = 0;
```

Равномерное движение

```
DrawRect(xfrom - 1.2, 0, 0.4, 10);
DrawRect(xto + 1.2, 0, 0.4, 10);
float x = 0, y = 0;

if (startTime==0) startTime = GetTickCount();

float dt = (GetTickCount() - startTime)/1000.f;

x = xfrom + v*dt;
if (x>xto) startTime = GetTickCount();

DrawRound(x, y, 1);
```

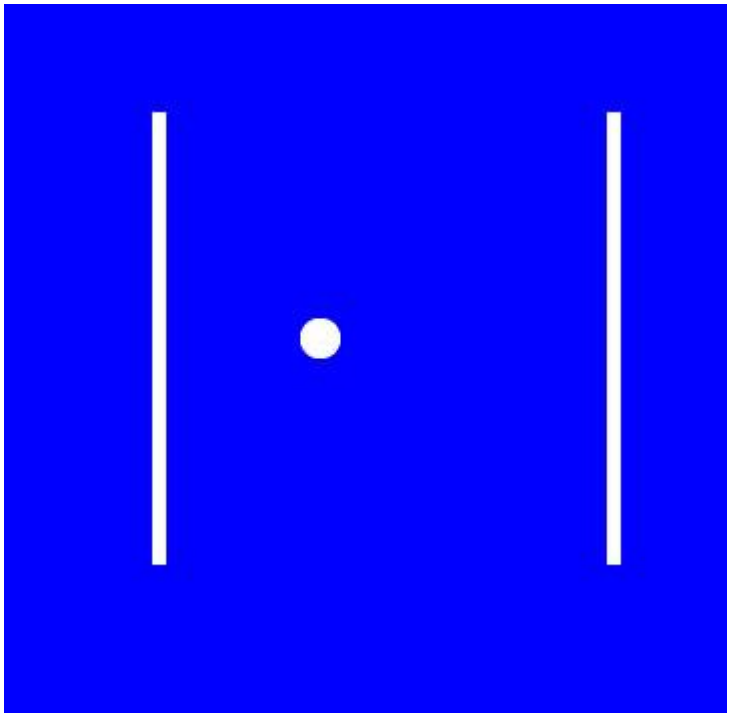
Равномерное движение

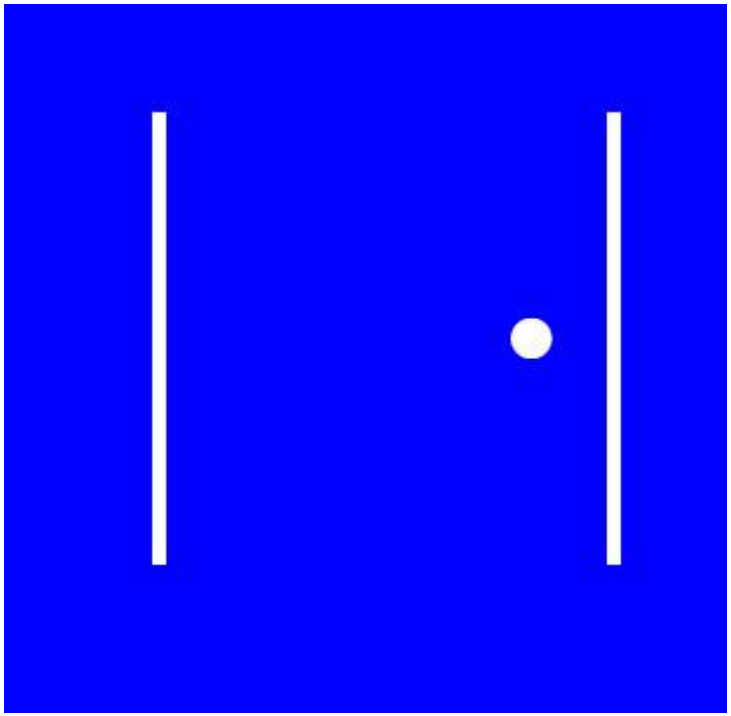
Равномерное движение

```
DrawRect(xfrom - 1.2, 0, 0.4, 10);
DrawRect(xto + 1.2, 0, 0.4, 10);
float x = 0, y = 0;

if (startTime==0) startTime = GetTickCount();

float dt = (GetTickCount() - startTime)/1000.f;

x = xfrom + v*dt;
if (((x>xto) &&(xto>xfrom)) || ((x<xto) &&(xto<xfrom)))
 {
  v = -v;  float xtemp = xto;  xto = xfrom; xfrom = xtemp;
  startTime = GetTickCount();
 }
DrawRound(x, y, 1);
```

Равномерное движение

Равномерное движение





Равномерное движение в плоскости

Необходимые переменные:

```
float xfrom = 0;
float xto = 10;
float vx = 5;
float yfrom = 0;
float yto = 10;
float vy = 3;
int startTimex = 0;
int startTimey = 0;
```

Равномерное движение в плоскости

```
DrawRect(0-1.2, 5, 0.4, 12.8);
DrawRect(10+1.2, 5, 0.4, 12.8);

DrawRect(5, 0-1.2, 12, 0.4);
DrawRect(5, 10+1.2, 12, 0.4);
float x, y = 0;

if (startTimex==0) startTimex = GetTickCount();
if (startTimey==0) startTimey = GetTickCount();

float dtx = (GetTickCount() - startTimex)/1000.f;
float dty = (GetTickCount() - startTimey)/1000.f;
```

Равномерное движение в плоскости

```
x = xfrom + vx*dtx;
y = yfrom + vy*dty;

if (((x>xto) &&(xto>xfrom)) || ((x<xto) &&(xto<xfrom)))
{
    vx = -vx;
    float xtemp = xto;
    xto = xfrom;
    xfrom = xtemp;

    startTimex = GetTickCount();
}
```

Равномерное движение в плоскости

```
if (((y>yto) &&(yto>yfrom)) || ((y<yto) &&(yto<yfrom)))
 {
     vy = -vy;
     float ytemp = yto;
     yto = yfrom;
     yfrom = ytemp;

     startTimey = GetTickCount();
 }

DrawRound(x, y, 1);
```

Равномерное движение в плоскости

Равномерное движение по эллипсу

Необходимые переменные:

```
float vr = 90;
int startTime = 0;
float rx = 10;
float ry = 5;
```

```
float dt = (GetTickCount() - startTime)/1000.f;

float x = rx * sin(dt*vr/60.f);
float y = ry * cos(dt*vr/60.f);

DrawRound(x, y, 0.5);
```

Равномерное движение по эллипсу

Равномерное движение по криволинейной траектории

```
float dt = (GetTickCount() - startTime)/1000.f;

float x = rx*sin(dt*vx/0.7);
float y = ry*cos(dt*vy/1.2);

DrawRound(x, y, 0.5);
```

Равномерное движение по криволинейной траектории

Равномерное движение по криволинейной траектории

если убрать очистку цвета

glClear(/*GL_COLOR_BUFFER_BIT |*/ GL_DEPTH_BUFFER_BIT);

Поворот и изменение размеров

```
float scale = (sin(GetTickCount()/1000.f) + 2)/2.f;

 glScaled(scale, scale, 1);
glRotated(GetTickCount()/10.f, 0, 0, 1);

DrawOneKube();
```

Поворот и изменение размеров

Пример движения планет

```
DrawRound(vertex(0, 0, 0), 1, clRed);

float angle = GetTickCount() / 100.f;
float r = 10.0;
float x =  r*sin(angle*M_PI / 180);
float y =  r*cos(angle*M_PI / 180);
glPushMatrix();
glTranslated(x, y, 0);
DrawRound(vertex(0, 0, 0), 1, clBlue);
```

Пример движения планет

```
    angle = GetTickCount() / 30.f;
    r = 3.0;
    x = r*sin(angle*M_PI / 180);  y = r*cos(angle*M_PI / 180);
    glTranslated(x, y, 0);
    DrawRound(vertex(0, 0, 0), 0.5, clGrey);
glPopMatrix();

    angle = GetTickCount() / 70.f;
    r = 7.0;
    x = r*sin(angle*M_PI / 180);  y = r*cos(angle*M_PI / 180);
    glTranslated(x, y, 0);
    DrawRound(vertex(0, 0, 0), 0.8, clGreen);
```

Пример движения планет