*2. Использование функции «Camera» в программе «OpenProcessing».*

2*.2. О функции.*

Для создания своего диплома я использовал программу «OpenProcessing». Если конкретнее, то самой нужной функцией в этой программе оказалась «Camera».

Прежде всего, стоит сказать, что эта камера используется только для 3D-объектов. Благодаря камере можно сделать имитацию движения и наблюдения за объектом, при этом не перемещая его.

Эта функция выглядит так:

«camera( X, Y, Z, centerX, centerY, centerZ, upX, upY, upZ)»

Значения:

1. X - позиция камеры на оси X;
2. Y - позиция камеры на оси Y;
3. Z - позиция камеры на оси Z;
4. centerX - координата X для центра объекта;
5. centerY - координата Y для центра объекта;
6. centerZ - координата Z для центра объекта;
7. upX - координата X точки обзора объекта;
8. upY - координата Y точки обзора объекта;
9. upZ - координата Z точки обзора объекта;

2.2. Использование sin и cos.

Если представить, что у нас не трехмерный объект, а двумерный, то для того, чтобы смотреть на него с разных ракурсов нам понадобятся синусы и косинусы, зависящие от перемещения курсора на экране. Вспомнив единичную окружность для синусов и косинусов, можно понять, как именно будут они изменяться в процессе перемещения. Вот программа, в которой представлено изменение синуса и косинуса для двумерного объекта на окружности:

int angle=0;

int radius = 100;

void setup() {

size (770, 770);}

void draw() {

background (0);

float x = cos(radians(angle)) \* radius;

float y = sin(radians(angle)) \* radius;

fill(234, 2, 2);

stroke(234, 2, 2);

text("Demonstration of sin and cos",

width/2 - textWidth("Demonstration of sin and cos")/2,

height / 2 -radius - 30);

strokeWeight(1);

line (width/2 - textWidth("Demonstration of sin and cos»)/2,

height / 2 -radius - 27,

width/2 + textWidth("Demonstration of sin and cos")/2-8,

height / 2 -radius - 27);

strokeWeight(1);

line (width/2, height / 2, width/2 + x, height/2 + y);

stroke(255, 2, 2);

strokeWeight(5);

point (width/2, height / 2);

stroke(2, 244, 222);

point (width/2 + x, height/2 + y);

stroke(2, 244, 2);

fill(2, 244, 2);

line (width/2, height / 2+ radius + 10, width/2 + x, height / 2 + radius + 10);

text("x = "+x, width/2, height / 2 + radius + 30);

stroke(30, 120, 222);

fill(30, 120, 222);

line (width/2+radius+10, height/2, width/2+radius+10, height/2+y);

text("y = "+y, width/2 + radius + 30, height / 2 + 0);

fill(234);

println(angle);

text ("angle = "+ angle, width/2-27, height / 2 - 10 );

fill(234);

text( "float x = cos(radians(angle)) \* radius;", width/2-227, height - 140 );

text( "float y = sin(radians(angle)) \* radius;", width/2-227, height - 120 );

angle++;

if (angle>360)

{angle=0;}

}

Эта программа высчитывает положение точки, на расстоянии заданного радиуса, в зависимости от синуса и косинуса. На экран выведены числовые значения каждого элемента.

С трехмерной моделью будет происходить тоже самое, только добавится третья ось Z. Будет использована разница между текущей позицией мыши и предыдущей, чтобы переделать значение в косинусы и синусы. Программа станет короче, так как нам уже не нужно будет выводить все на экран. В итоге программа для трехмерного объекта получается такой:

void setup() {

size(1250, 600, P3D);

}

void draw() {

background(1);

lights();

r = 0;

d = 0;

ySpeed = mouseY + pmouseY;

r = r + ySpeed;

xSpeed = mouseX + pmouseX;

d = d + xSpeed;

float camX = 700 \* cos(radians(d/9));

float camY = 700 \* cos(radians(r/9));

float camZ = 700 \* sin(radians(d/9));

camera(camX, camY, camZ, 0, 0, 0, 0, -1 ,0);

translate(0, 0, 0);

fill (255, 0,0);

box(200);

}