Основые параметры это: разрешение дисплея и указание на визуализацию окна OPENGL.

pygame.init()

display = (1600, 900)

screen = pygame.display.set\_mode(display, DOUBLEBUF | OPENGL)

Затем переключаемся на матрицу роекции и устанавливаем для нее параметры, такие как угол обзора, соотношение ширины дисплея на высоту, ближнюю и дальнюю плоскости отсечения.

glMatrixMode(GL\_PROJECTION)

gluPerspective(60, (display[0] / display[1]), 0.1, 6000.0)

Изменив матрицу перспективы на объектно-видовую устаннавливаем основные настройки для наего проекта:

glMatrixMode(GL\_MODELVIEW)

glEnable(GL\_LIGHTING) #включаем свет в нашем проекте

glLightModelf(GL\_LIGHT\_MODEL\_TWO\_SIDE, GL\_TRUE) # задаем двустороннюю модель освещения

glEnable(GL\_LIGHT0) #включаем первый источник света

glEnable(GL\_LIGHT1) #включаем второй источник света

glEnable(GL\_DEPTH\_TEST) #включаем тест глубины

glEnable(GL\_COLOR\_MATERIAL) #включаем материалы для “объектов”

glEnable(GL\_TEXTURE\_2D) #включаем возможность обрабатывать текстуры

glEnable(GL\_LIGHT2) #включаем второй источник света

gluLookAt(0, -90, 0, 0, -1, 0, 0, 0, 0.5) #устанавливаем начальное положение относительно началы точки рисования

viewMatrix = glGetFloatv(GL\_MODELVIEW\_MATRIX) #сохраняем в переменную нашу матрицу модели для дальнейшго использования

displayCenter = [screen.get\_size()[i] // 2 for i in range(2)] #сохраняем в переменную список из координат х и у центра окна

mouseMove = [] #инициализируем пустой список

pygame.mouse.set\_visible(False) #

Вызывая функцию BindTexture() мы производим загрузку текстур в список с помощью следующих функций:

def BindTexture():

for i in range(1, 14):

loadTexture('image/{}.jpg'.format(i)) #с помощью цикла мы передаем функции loadTexture путь к нашей текстуре

return texid

Основная функция загрузки текстур:

def loadTexture(file\_name): // принимает на вход путь к файлу

textureSurface = pygame.image.load(file\_name) // загрузка изображения, приведение к класссу Surface и сохранение в переменную

textureData = pygame.image.tostring(textureSurface, "RGBA", True) // перевод изображения в строковое представление

width = textureSurface.get\_width()//ширина изображения

height = textureSurface.get\_height()//высота изображения

texid = glGenTextures(1)//генерация порядкового номера текстуры

glBindTexture(GL\_TEXTURE\_2D, texid) //активируем текущую текстуру

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, width, height,

0, GL\_RGBA, GL\_UNSIGNED\_BYTE, textureData)// перевод из строкового представления обратно в битовый

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT)// устаавливаем модель поведения, если при установке коорднат текстур будет указано значеие больще 1

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_NEAREST)//устанавливаем модель поведения для фильтрации текстур

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST)

После процедуры инициализации нашего проекта, начинается основной цикл программы:

while True:

for event in pygame.event.get(): //отлавливание событий модулем pygame

if event.type == pygame.QUIT: //поведениепри закрытии окна программы

pygame.quit()

exit()

if event.type == pygame.KEYDOWN: //повдение при нажатии клавиш

if event.key == pygame.K\_ESCAPE or event.key == pygame.K\_RETURN:

pygame.quit()

exit()

if event.type == pygame.MOUSEMOTION: //повдение при перемещении курсора внутри окна

mouseMove = [event.pos[i] - displayCenter[i] for i in range(2)] //при изменении позиции курсора значения отклонения по оси х и у заносятся в список

pygame.mouse.set\_pos(displayCenter)// устанавливаем положение курсора в центре окна, для избежания вылета за границы окна программы

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT) //перед каждым новым кадром очищаем буфер цвета и глубины

glLoadIdentity()//загружаем единичную матрицу

keypress = pygame.key.get\_pressed()//присвоение переменной указателя на метод

glRotatef(mouseMove[0] \* 0.1, 0.0, 0.1, 0.0)// при изменениях координат курсора. Происходит поворот по оси х

glRotatef(mouseMove[1] \* 0.1, 0.1, 0.0, 0.0)// поворот по оси у

//ниже описаны события для обработки нажатия конкретных клавиш

if keypress[pygame.K\_t]://выключение источника света 0

glDisable(GL\_LIGHT0)

if keypress[pygame.K\_y]: //включение источника света 0

glEnable(GL\_LIGHT0)

if keypress[pygame.K\_1]: ]://выключение источника света 1

glDisable(GL\_LIGHT1)

if keypress[pygame.K\_2]: //включение источника света 1

glEnable(GL\_LIGHT1)

if keypress[pygame.K\_r]:// выключение текстур

glDisable(GL\_TEXTURE\_2D)

if keypress[pygame.K\_e]://включение текстур

glEnable(GL\_TEXTURE\_2D)

//обработка для перемещения по осям х и у

if keypress[pygame.K\_w]:

glTranslatef(0, 0, 0.8)

if keypress[pygame.K\_s]:

glTranslatef(0, 0, -0.8)

if keypress[pygame.K\_d]:

glTranslatef(-0.8, 0, 0)

if keypress[pygame.K\_a]:

glTranslatef(0.8, 0, 0)

glMultMatrixf(viewMatrix) // перемножаем нашу ммарицу со смещениями на нашу основную матрицу

viewMatrix = glGetFloatv(GL\_MODELVIEW\_MATRIX) //сохранение матрицы

Далее идет процесс описания цикла рисования :

glLightfv(GL\_LIGHT1, GL\_AMBIENT, [0.3, 0.3, 0.3])// установка интенсвности фонового излучения источника 0

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, [0.8, 0.8, 0.8])// установка интенсивности рассеенного источниасвета 0

glLightfv(GL\_LIGHT1, GL\_POSITION, [0.0, 0.0, 0.0, 0.0])// утановка позиции источника света 0, создание направленного источника света

sphere()//вызов функци рисования сферы

glTranslatef(0, 0, -20)//смещение

steve()//вызов функции рисования основой модельки

glTranslatef(-20, 0, 6.5)

steve()

glTranslatef(-5, -5, -12.0)

glTranslatef(17, -10, 22)

cube()// вызов функции рисования второй сферы

glTranslatef(0, 10, -10)

light()// вызов функии рисования куба(лампы)

//установка значений для второго источника света

glLightfv(GL\_LIGHT0, GL\_SPECULAR, [1, 0.7, 0])

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, [1, 0.7, 0])

glLightfv(GL\_LIGHT0, GL\_AMBIENT, [1, 0.7, 0])

//создание точечного источника света

glLightfv(GL\_LIGHT0, GL\_POSITION, [0.0, 0.0, 0.0, 1])

glLightf(GL\_LIGHT0, GL\_CONSTANT\_ATTENUATION, 0.001)//костанта затухания

glLightf(GL\_LIGHT0, GL\_LINEAR\_ATTENUATION, 0.3)// указание способа вычисления затухания и задания множителя(в нашем случае выбрана линейная модель)

glTranslatef(0, 0, 20)

sphere1()// рисование второй сферы

glTranslatef(0, 0, -1522)

ground()//отрисовка поверхности на которой стоят модельки

pygame.display.flip() //обновление экрана

pygame.time.wait(10) //установка таймера между обновлениями экрана

Описание функций рисования моделей:

def ground():

glColor3fv((1, 1, 1))//установка цвета для дальнейшего рисования

//установка свойств материала

glMaterialfv(GL\_FRONT, GL\_SPECULAR, (0.0, 0.0, 0.0))

glMaterialfv(GL\_FRONT, GL\_AMBIENT, (1.0, 1.0, 1.0))

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, (0.5, 0.5, 0.5))

//установка текстуры

glBindTexture(GL\_TEXTURE\_2D, 0+1)

//рисование сферы диаметром 1500, и кол. полигонов 100/100

gluSphere(qobj, 1500, 100, 100)

def light():

glColor3fv((1, 1, 1))

glMaterialfv(GL\_FRONT, GL\_AMBIENT, (0, 0, 0))

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, (0, 0, 0))

glMaterialfv(GL\_FRONT, GL\_SPECULAR, (0, 0, 0))

glBindTexture(GL\_TEXTURE\_2D, 12+1)

glBegin(GL\_QUADS)//начало процесса рисования, в качестве параметра указывается что мы будем рисовать с помощью прямоугольников

glTexCoord2f(0.0, 0.0)//установка координаты текстуры

glVertex3f(-2.0, -2.0, 2.0)//задание координаты

glTexCoord2f(1.0, 0.0)

glVertex3f(2.0, -2.0, 2.0)

glTexCoord2f(1.0, 1.0)

glVertex3f(2.0, 2.0, 2.0)

glTexCoord2f(0.0, 1.0)

glVertex3f(-2.0, 2.0, 2.0)

glTexCoord2f(1.0, 0.0)

glVertex3f(-2.0, -2.0, -2.0)

glTexCoord2f(1.0, 1.0)

glVertex3f(-2.0, 2.0, -2.0)

glTexCoord2f(0.0, 1.0)

glVertex3f(2.0, 2.0, -2.0)

glTexCoord2f(0.0, 0.0)

glVertex3f(2.0, -2.0, -2.0)

glTexCoord2f(0.0, 1.0)

glVertex3f(-2.0, 2.0, -2.0)

glTexCoord2f(0.0, 0.0)

glVertex3f(-2.0, 2.0, 2.0)

glTexCoord2f(1.0, 0.0)

glVertex3f(2.0, 2.0, 2.0)

glTexCoord2f(1.0, 1.0)

glVertex3f(2.0, 2.0, -2.0)

glTexCoord2f(1.0, 1.0)

glVertex3f(-2.0, -2.0, -2.0)

glTexCoord2f(0.0, 1.0)

glVertex3f(2.0, -2.0, -2.0)

glTexCoord2f(0.0, 0.0)

glVertex3f(2.0, -2.0, 2.0)

glTexCoord2f(1.0, 0.0)

glVertex3f(-2.0, -2.0, 2.0)

glTexCoord2f(1.0, 0.0)

glVertex3f(2.0, -2.0, -2.0)

glTexCoord2f(1.0, 1.0)

glVertex3f(2.0, 2.0, -2.0)

glTexCoord2f(0.0, 1.0)

glVertex3f(2.0, 2.0, 2.0)

glTexCoord2f(0.0, 0.0)

glVertex3f(2.0, -2.0, 2.0)

glTexCoord2f(0.0, 0.0)

glVertex3f(-2.0, -2.0, -2.0)

glTexCoord2f(1.0, 0.0)

glVertex3f(-2.0, -2.0, 2.0)

glTexCoord2f(1.0, 1.0)

glVertex3f(-2.0, 2.0, 2.0)

glTexCoord2f(0.0, 1.0)

glVertex3f(-2.0, 2.0, -2.0)

glEnd()//окончание рисования