Московский Авиационный Институт
(национальный исследовательский университет)
Факультет прикладной математики

Компьютерная графика
Отчет по лабораторной работе №6

Бондарева Елена
Группа М8О-305Б-21

Преподаватель: Симкин О.В.

Москва, 2023

1. Тема: Создание шейдерных анимационных эффектов в OpenGL 2.1

2. Цель работы: Для слоя парабалоида обеспечить выполнение следующего шейдерного эффекта.

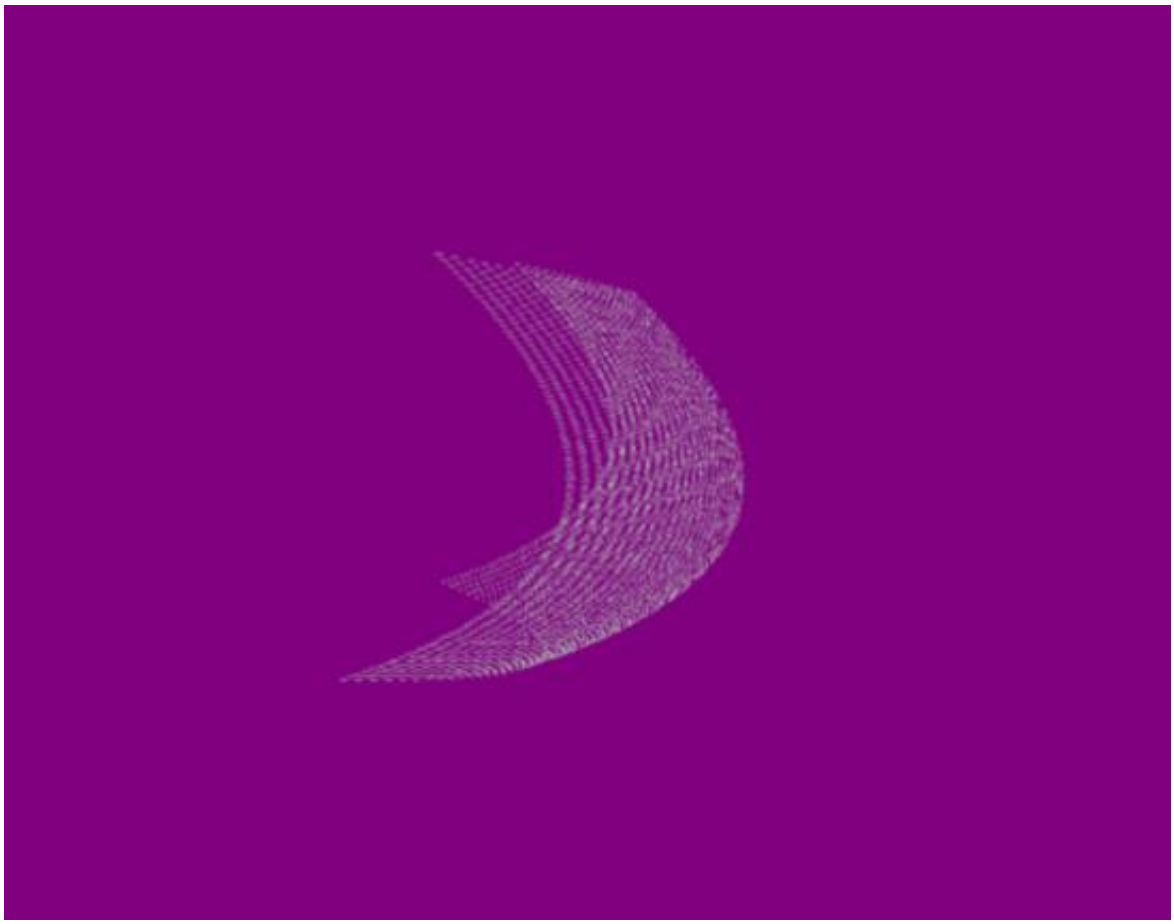
3. Задание (вариант №18) Анимация. Вращение относительно оси OZ

Скорость вращения меняется по синусоиде.

4. Идея, метод, алгоритм решение задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные

спецификации с пред- и постусловиями)

5. Результат работы



6. Распечатка протокола

```
import pygame
from pygame.locals import *
from OpenGL.GL import *
from OpenGL.GLU import *
import math

def draw_frustum(bottom_radius, top_radius, height, sides):
    angle_step = 360.0 / sides

    glBegin(GL_TRIANGLE_STRIP)
    for i in range(sides + 1):
        angle = i * angle_step
        x1 = bottom_radius * math.cos(math.radians(angle))
        y1 = bottom_radius * math.sin(math.radians(angle))
        x2 = top_radius * math.cos(math.radians(angle))
        y2 = top_radius * math.sin(math.radians(angle))
        glVertex3f(x1, y1, -height / 2.0)
        glVertex3f(x2, y2, height / 2.0)
    glEnd()

    glBegin(GL_TRIANGLE_FAN)
    glVertex3f(0, 0, -height / 2.0)
    for i in range(sides + 1):
        angle = i * angle_step
        x = bottom_radius * math.cos(math.radians(angle))
        y = bottom_radius * math.sin(math.radians(angle))
        glVertex3f(x, y, -height / 2.0)
    glEnd()

    glBegin(GL_TRIANGLE_FAN)
    glVertex3f(0, 0, height / 2.0)
```

```
    for i in range(sides + 1):
        angle = i * angle_step
        x = top_radius * math.cos(math.radians(angle))
        y = top_radius * math.sin(math.radians(angle))
        glVertex3f(x, y, height / 2.0)
    glEnd()

def main():
    pygame.init()
    display = (1280, 720)
    pygame.display.set_mode(display, DOUBLEBUF | OPENGL)

    # Используем glFrustum вместо gluPerspective
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    aspect_ratio = display[0] / display[1]
    glFrustum(-0.1 * aspect_ratio, 0.1 * aspect_ratio, -0.1, 0.1, 0.1, 50.0)
    glMatrixMode(GL_MODELVIEW)

    glTranslatef(0.0, 0.0, -5)
    glEnable(GL_DEPTH_TEST)
    glEnable(GL_LIGHTING)
    glEnable(GL_LIGHT0)

    glLightfv(GL_LIGHT0, GL_POSITION, [0, 1, 0, 3])
    glLightfv(GL_LIGHT0, GL_DIFFUSE, [1.0, 1.0, 1.0, 1.0])

    glMaterialfv(GL_FRONT, GL_DIFFUSE, [1.0, 0.0, 0.0, 1.0])

    clock = pygame.time.Clock()

    def draw_scaled_frustum(scale_x, scale_y, scale_z):
```

```
        glPushMatrix()
        glScalef(scale_x, scale_y, scale_z)
        draw_frustum(1, 0.5, 2, 32)
        glPopMatrix()

    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                return

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glLoadIdentity()
        glTranslatef(0.0, 0.0, -5)
        glRotatef(pygame.time.get_ticks() * 0.1, 3, 1, 1)

        draw_scaled_frustum(1.0, 1.0, 1.0)

        pygame.display.flip()
        clock.tick(60)

if __name__ == "__main__":
    main()
```