МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (ФГБОУ ВО «КубГУ»)**

**Факультет математики и компьютерных наук**

**Кафедра вычислительной математики и информатики**

**КУРСОВАЯ РАБОТА**

**МОДЕЛИРОВАНИЕ ДИНАМИЧЕСОГО ИЗОБРАЖЕНИЯ  
 «СОЛНЕЧНАЯ СИСТЕМА»**

Работу выполнил М. М. Луговой

(подпись)

Направление подготовки 02.03.01 Математика и компьютерные науки курс 2

Научный руководитель

ст. преподаватель И. В. Сухан

(подпись, дата)

Нормоконтролер

ассистент А.А.Лахтина

(подпись, дата)

Краснодар 2021

**СОДЕРЖАНИЕ**

Введение 3

1. Анализ условия задачи и разработка алгоритма решения 5
2. Программная формализация задачи 6
3. Составление контрольных примеров для отладки программы 11

Заключение 14

Список использованных источников 15

**Введение**

Поставлена задача: построить модель солнечной системы, включающую в себя основные объекты: солнце и планеты, с приблизительным сохранением масштаба.

Цель: разработать программу, решающую поставленную задачу.

Для реализации задачи будет использован С++, как наиболее подходящий язык программирования. В ходе работы будут проведены следующие действия:

1. Анализ и подбор графической библиотеки.

2. Изучение возможностей подобранной библиотеки и её применение.

3. Разработка алгоритма решения.

Требования к программе:

1. Соответствие масштаба планет солнечной системы и расстояний между ними.

2. Вывод результата работы программы на экран.

1. **Анализ условия задачи и разработка алгоритма решения**

По условию, задание предполагает написанный код, который моделирует поведение солнечной системы. Для того, чтобы достичь реалистичной визуализации и анимации модели солнечной системы, была выбрана библиотека OpenGL, позволяющая работать с 3D-пространством и фигурами.

Модель солнечной системы должна содержать не только планеты и солнце, но и спутники. Планеты, солнце и спутники будут изображены сферами с нанесением цвета сверху. Созданием сфер будет заниматься функция gluSphere(), которая также находится в библиотеке OpenGL. Для лучшей наглядности были добавлены орбиты планет и спутников.

Расположением камеры на экране занимается функция gluLookAt. Эта функция принимает 9 значений: первые 3 — координаты позиции глаза наблюдателя, вторые 3 — координаты точки, располагающейся в центре экрана, третьи 3 — направление вектора, задающего поворот сцены. Также было принято решение, что анимацией, камерой и некоторыми компонентами солнечной системы можно будет управлять с клавиатуры.

Скорость вращения планет вокруг солнца относительно других планет должна соблюдаться. Для реализации вращения в OpenGL имеется команда **glRotate\*(). Эта команда** производит поворот объекта против часовой стрелки на угол angle вокруг вектора (x,y,z).

1. **Программная формализация задачи**

Прежде чем приступить к выполнению программы, нужно выбрать среду программирования, где будет проходить реализация самой задачи.

Язык программирования C++, подходит для реализации этой задачи лучше всего. Для C++ доступно огромное множество библиотек. Используя C++ мы можем реализовать все, что захотим.

— Заголовочный файл «iostream» служит для того, чтобы предоставить средства ввода-вывода для стандартной консоли. Все его возможности размещены в пространстве имен *std*.

#include <iostream>

— Кроссплатформенная библиотека на С++ допускает упрощение запросов и загрузку расширений OpenGL.

#include <GL/glew.h>

— Заголовочный файл «freeglut.h» позволяет пользователю создавать окна, предоставляющие контекст OpenGL на широком спектре платформ, и управлять ими, а также взаимодействовать с мышью, клавиатурой и джойстиком.

#include <GL/freeglut.h>

— Любая программа в С++ должна содержать главную функцию main ( ). В функции main(int argc, char\*\* argv) объявляются все функции.

int main(int argc, char\*\* argv)

— Следующая функция выводит на экран инструкцию о том, чем можно управлять в анимации.

intructions()

— Каждая программа OpenGL, использующая GLUT, должна начинаться с инициализации GLUT-машины.

glutInit(&argc, argv)

— Ниже представлен блок кода создания окна вывода с определенными параметрами и названием.

glutInitContextVersion(4, 2);

glutInitContextProfile(GLUT\_COMPATIBILITY\_PROFILE);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

glutInitWindowSize(700, 700);

glutInitWindowPosition(500, 0);

glutCreateWindow("Solar System");

glutDisplayFunc(drawScene);

glutReshapeFunc(resize);

— Функции с описанным алгоритмом позволяют пользователю упралять программой, используя клавиатуру и мышку.

glutMouseFunc(mouseControl);

glutKeyboardFunc(keyInput);

glutMouseWheelFunc(mouseWheel);

— В класс Planet передаются параметры, которые будут присвоены планетам. В функциях drawSmallOrbit(void) и drawMoon(void) программа рисует спутники планетам и орбиты.

class Planet

{

public:

float radius, distance, orbit, orbitSpeed, axisTilt, axisAni;

Planet(float \_radius, float \_distance, float \_orbit, float \_orbitSpeed, float \_axisTilt, float \_axisAni)

{

radius = \_radius;

distance = \_distance;

orbit = \_orbit;

orbitSpeed = \_orbitSpeed;

axisTilt = \_axisTilt;

axisAni = \_axisAni;

}

void drawSmallOrbit(void)

{

glPushMatrix();

glColor3ub(255, 255, 255);

glRotatef(90.0, 1.0, 0.0, 0.0);

glutWireTorus(0.001, distance, 100.0, 100.0);

glPopMatrix();

}

void drawMoon(void)

{

GLUquadricObj\* quadric;

quadric = gluNewQuadric();

glPushMatrix();

glColor3f(0.3, 0.3, 0.3);

glRotatef(orbit, 0.0, 1.0, 0.0);

glTranslatef(distance, 0.0, 0.0);

gluSphere(quadric, radius, 20.0, 20.0);

glPopMatrix();

}

}

— Передача в «class Planet» параметров планет и спутников осуществляется следующим образом.

Planet sun(5.0, 0, 0, 0, 0, 0);

…

Planet tri(.36, 3.2, 0, 3.40, 0, 0);

— Задаем глобальные переменные, которые отвечают за анимацию, переключающие орбиты планет и спутников, приближение камеры, угол камеры.

int isAnimate = 0;

int bigOrbitActive = 1;

int smallOrbitActive = 1;

int moonsActive = 1;

int changeCamera = 0;

int zoom = 50;

Функция образует орбиты планет для лучшей наглядности их движения.

void orbitalTrails(void)

{

glPushMatrix();

glColor3ub(255, 255, 255);

glTranslatef(0.0, 0.0, 0.0);

glRotatef(90.0, 1.0, 0.0, 0.0);

glutWireTorus(0.001, mer.distance, 100.0, 100.0);

…

}

— Ниже приведен блок кода главной функции, который моделирует планеты, и устанавливает 3 варианта расположения камер. Местоположение камер было задано через координаты.

void drawScene(void)

{

…

//Sun

glColor3f(1.0, 0.8, 0.3);

glPushMatrix();

glRotatef(sun.orbit, 0.0, 1.0, 0.0);

glTranslatef(sun.distance, 0.0, 0.0);

glPushMatrix();

glRotatef(sun.axisTilt, 1.0, 0.0, 0.0);

glRotatef(sun.axisAni, 0.0, 1.0, 0.0);

glRotatef(90.0, 1.0, 0.0, 0.0);

gluSphere(quadric, sun.radius, 20.0, 20.0);

glPopMatrix();

glPopMatrix();

…

if (changeCamera == 0)gluLookAt(0.0, zoom, 50.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

}

— Функция изменяет размеры сцены, с учетом того, что солнечная система в сцене действительно отображается.

void resize(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glFrustum(-5.0, 5.0, -5.0, 5.0, 5.0, 200.0);

glMatrixMode(GL\_MODELVIEW);

}

— Блок кода отвечает за анимацию и скорость, которая будет увеличивать значения каждый раз на заданную величину. В библиотеке OpenGL за это отвечает функция **glRotate\*(). Эта команда** производит поворот объекта против часовой стрелки на угол angle (измеряется в градусах) вокруг вектора (x,y,z).

void animate(int n)

{

if (isAnimate)

{

mer.orbit += mer.orbitSpeed;

…

tri.orbit += tri.orbitSpeed;

if (mer, ven, ear, mar, jup, sat, ura, nep, plu, lun, pho, dei, eur, gan, cal, tit, nix, puc, tri.orbit > 360.0)

{

mer, ven, ear, mar, jup, sat, ura, nep, plu, lun, pho, dei, eur, gan, cal, tit, nix, puc, tri.orbit -= 360.0;

}

mer.axisAni += 10.0;

…

plu.axisAni += 10.0;

if (mer, ven, ear, mar, jup, sat, ura, nep, plu.axisAni > 360.0)

{

mer, ven, ear, mar, jup, sat, ura, nep, plu.axisAni -= 360.0;

}

glutPostRedisplay();

glutTimerFunc(30, animate, 1);

}

}

1. **Составление контрольных примеров для отладки программы**

Пример: При запуске программы появляется изображение солнечной системы. Все планеты выстроены в одну линию. Это изображено на рисунке 1.

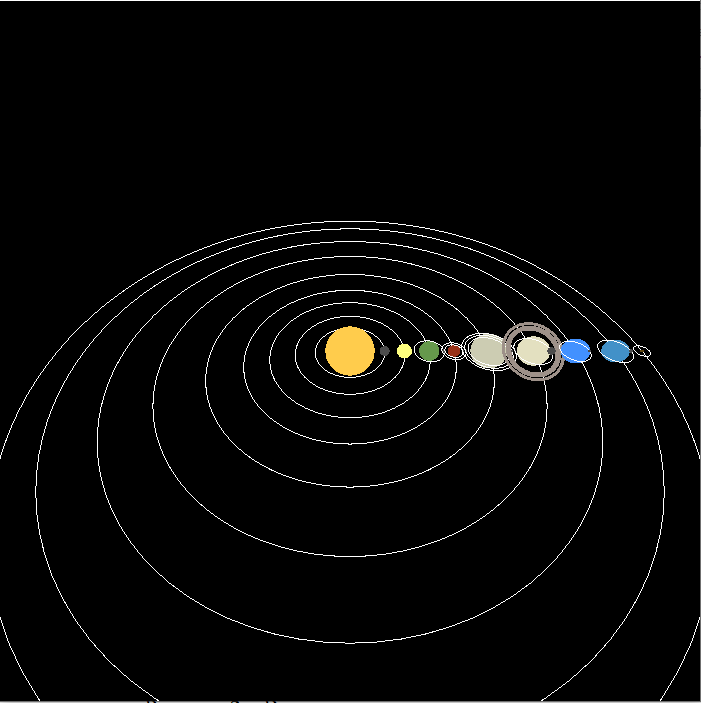


Рисунок 1 — Общий вид окна при запуске анимации

На рисунке 2 и рисунке 3 показаны виды камер, между которыми можно переключаться.

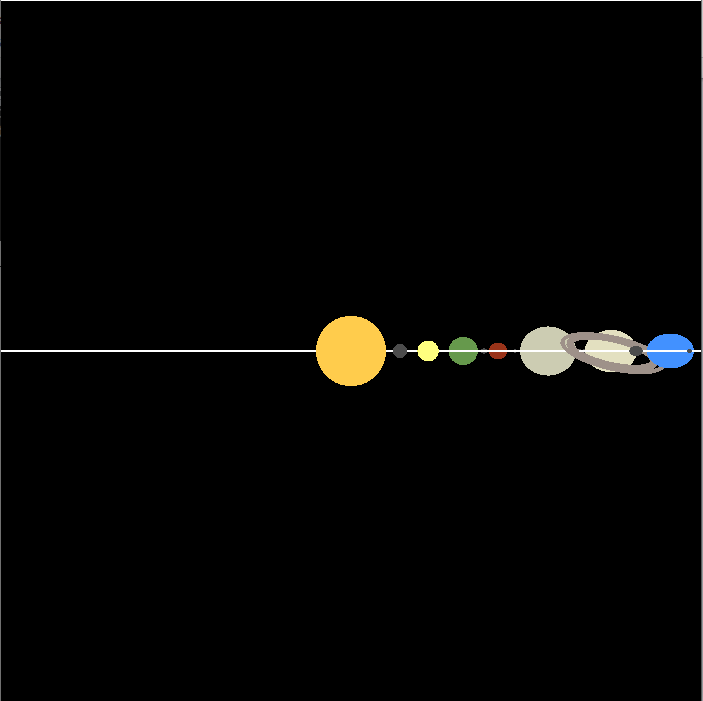


Рисунок 2 — Вид камеры №2

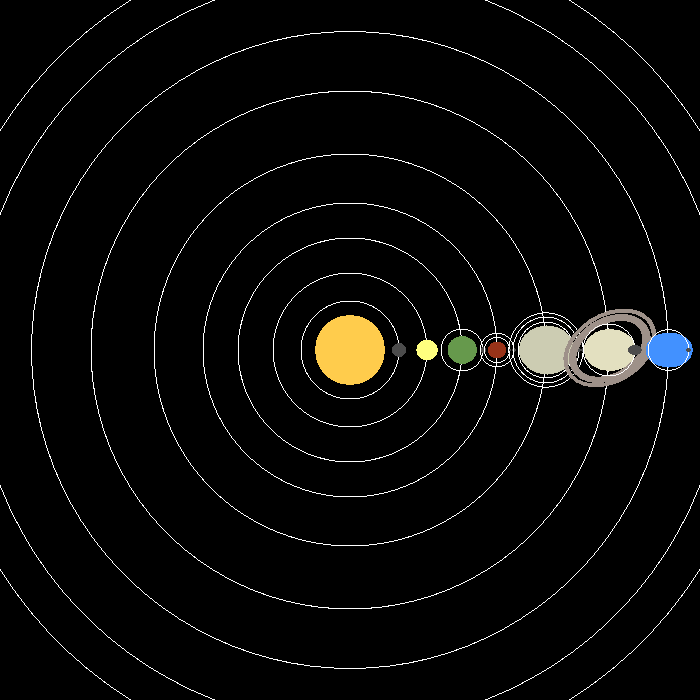


Рисунок 3 — Вид камеры №3

На рисунке 4 изображена солнечная система с отключенными орбитами.

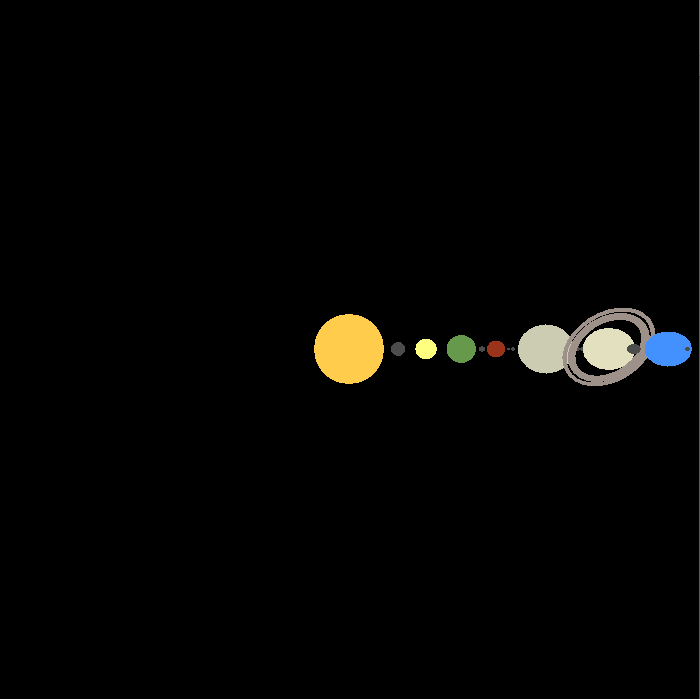


Рисунок 4 — Вид без орбит

**Заключение**

В ходе работы выполнены следующие задачи:

1. Проведен анализ условия задачи и разработаны этапы её реализации.

2. Описан алгоритм действий на языке программирования С++

3. Продемонстрированы примеры работы программы на изображениях.

Также можно доработать программу, добавив:

1. Текстуры планет.

2. Дополнительная анимация звезд, комет и тд.

3. Светотени.

Вкупе с вышеуказанными пунктами можно реализовать более доработанную 3D-модель солнечной системы. Проработанные текстуры и светотени позволят реализовать объёмные объекты. Дополнительные элементы в виде вспомогательных небесных светил образуют законченную композицию солнечной системы.

**Список использованных источников**

1. Программирование GLUT: окна и анимация. http://www.nl.linuxfocus.org/Russian/January1998/article16.html
2. Уроки программирования OpenGL на С++. <https://www.youtube.com/watch?v=Qmc4f3bwnYk>
3. Народный учебник по OpenGL. <http://pmg.org.ru/nehe/nehe04.htm>