**МИНОБРНАУКИ РОССИИ  
федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технологический университет «СТАНКИН»  
(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |  |
| --- | --- | --- |
| **Институт Информационных Систем и  Технологий** |  | **Кафедра Информационных Технологий и Вычислительных Систем** |

КУРСОВАЯ РАБОТА

|  |  |
| --- | --- |
| по дисциплине: | Компьютерная графика |
|  |  |
| на тему: | Тема курсовой работы |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| студента | *3* | курса | *бакалавриата* | группы | *ИДБ 19-02* |

|  |
| --- |
| **ЧАН ВУ КУЕТ** |

|  |  |
| --- | --- |
| направление: | 09.03.01 Информатика и вычислительная техника |
|  |  |
| специальность: | Программное обеспечение средств вычислительной техники и автоматизированных систем |

**ОСНОВНОЕ СОДЕРЖАНИЕ**

**Описание работы:**Игра про самолеты в трехмерном пространстве, управляемая пользователем. Есть много эффектов, которые комбинируются и улучшаются, чтобы сделать игру более яркой, интересной и более плавной.

**Язык и рабочая среда:** программное обеспечение OpenGL с языком C ++

**Модель задачи:**

**-**Самолетостроение

-Строительство многоэтажных домов

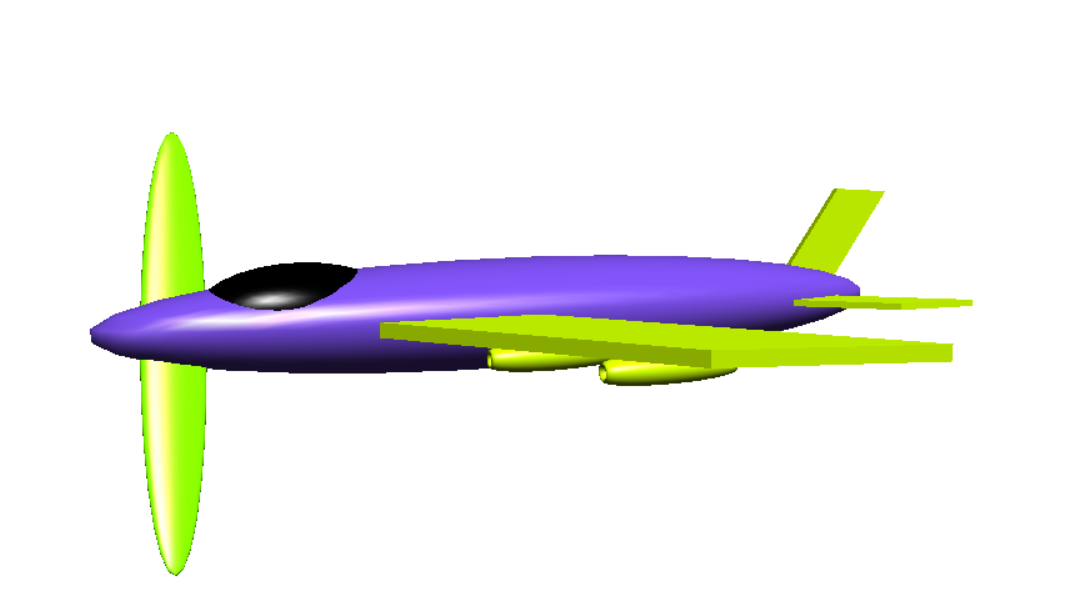
-Строительство солнечной батареи

-Экологическое строительство

-Комбинирование компонентов и добавление эффектов повышает качество игры.

**Процесс решения каждой конкретной модельной проблемы:**

**Самолетостроение**

****

**Рис 1 Модель самолета, который будет построен.**

**Задача состоит в том, чтобы собрать следующие детали:**

-Фюзеляж самолета

-Часть кабины самолета

- Крупнокрылые самолеты

- Двигатель под крылом

-Вентилятор самолета

-Хвостовой самолет

**Идеи деталей, пояснения и конкретный код:**

**Главная идея:**

-Использование встроенных функций (glutSolidSphere: рисование сплошной сферы, glutSolidCube: рисование сплошного куба, glutSolidTorus: рисование сплошного тора) в сочетании с принципами поворота и изменения размера объекта (glScalef, glRotatef).

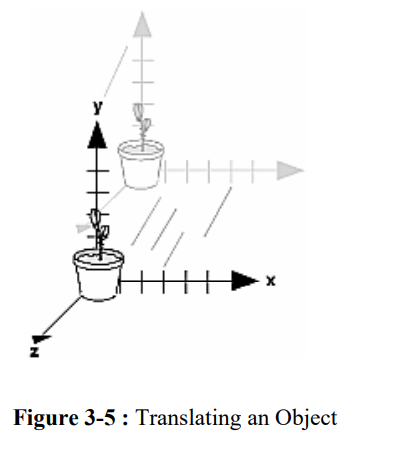
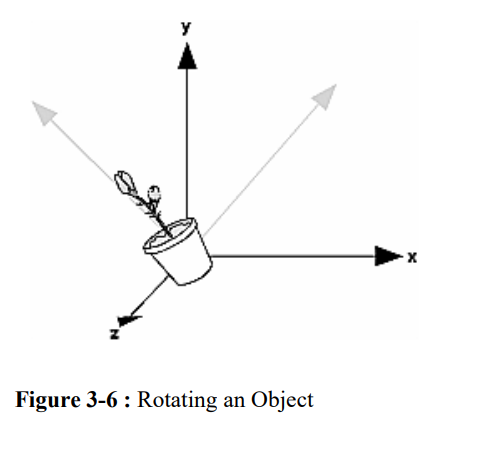
****

Рис 2 Общие правила изменения отображения объектов

**Конкретный код:**

\*/**Фюзеляж самолета** : Тело самолета (рис. 1) растягивается вдоль оси Ox, поэтому мы увеличиваем размер x и уменьшаем размер y, z для функции glScalef. Наконец, мы вызываем функцию glutSolidSphere, чтобы нарисовать сферу с радиусом 1, линиями долготы и широты 30. В сочетании с glScalef мы получаем фюзеляж.

glPushMatrix();

glTranslated(0, 0, 0);

glScaled(3, 0.4, 0.5);

glutSolidSphere(1, 30, 30);

glPopMatrix();

**\*/** **Часть кабины самолета:** Кабина самолета (рис. 1) расположена в верхней части фюзеляжа, поэтому сначала мы изменим значение x (x> 0) и значение y (y> 0) в функции glTranslatef. Затем, поскольку эта часть растянута вдоль оси Ox, мы увеличиваем размер x и уменьшаем размер y, z для функции glScalef. Наконец, мы вызываем функцию glutSolidSphere, чтобы нарисовать сферу с радиусом 0,45, линии долготы и широты равны 30. В сочетании с glScalef и glColor3f (черный) мы получаем кабину самолета.

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(1.7, 0.1, 0);

glScaled(1.5, 0.7, 0.8);

glutSolidSphere(0.45, 30, 30);

glPopMatrix();

**\*/** **Крупнокрылые самолеты (включая левую и правую) :** Поскольку часть крыла самолета (рис. 1) расположена по бокам фюзеляжа, мы сначала изменим значение z (z> 0 для правого крыла и z <0 для левого крыла) в функции glTranslatef. Затем, поскольку эта часть растягивается вдоль оси Oz, мы увеличиваем размер z и уменьшаем размер x, y для функции glScalef. Наконец, мы вызываем функцию glutSolidCube, чтобы нарисовать куб с радиусом 1. В сочетании с glScalef и glColor3f (зеленый) мы получаем кабину самолета.

**Примечание**. Далее, поскольку плоскость будет полностью вращаться на главном экране (меню), мы будем использовать glRotatef для поворота по вертикальной оси (Oy) с углом поворота -50 для правого крыла и 50 для левого крыла).

**правая**

glColor3d(0.8, 1, 0);

glPushMatrix();

glTranslated(0, 0, 1.2);

glRotated(-50, 0, 1, 0);

glScaled(0.7, 0.1, 3);

glRotated(25, 0, 1, 0);

glutSolidCube(1);

glPopMatrix();

**левая**

glColor3d(0.8, 1, 0);

glPushMatrix();

glTranslated(0, 0, -1.2);

glRotated(50, 0, 1, 0);

glScaled(0.7, 0.1, 3);

glRotated(-25, 0, 1, 0);

glutSolidCube(1);

glPopMatrix();

\*/ **Двигатель под крылом:** Поскольку двигательная часть самолета (рис. 1) расположена под крылом (2 стороны), сначала изменим значение x (x <0 для правого крыла и x> 0 для левого крыла) и цену. значение z (z> 0) в функции glTranslatef. Затем, поскольку эта часть растягивается вдоль оси Oz, мы увеличиваем размер z и уменьшаем размер x, y для функции glScalef. Наконец, мы вызываем glutSolidTorus, чтобы нарисовать круг с радиусом (внутри и снаружи), равным 0,5; количество радиальных линий и количество радиальных делений, равное 50. В сочетании с glScalef и glColor3f (зеленый) у нас есть машинная часть самолет.

**Примечание**: - Часть авиационного двигателя с каждой стороны построена в порядке от внутреннего двигателя до внешнего двигателя.

-В то же время, как и часть крыла, поскольку она также вращается на главном экране (меню), мы также вызываем функцию glRotatef с тем же параметром.

**Правая**

glColor3d(0.8, 1, 0);

glPushMatrix();

glTranslated(-0.3, -0.15, 1.5);

glRotated(90, 0, 1, 0);

glScaled(0.1, 0.1, 0.9);

glutSolidTorus(0.5, 0.5, 50, 50);

glPopMatrix();

glColor3d(0.8, 1, 0);

glPushMatrix();

glTranslated(0.2, -0.15, 0.9);

glRotated(90, 0, 1, 0);

glScaled(0.1, 0.1, 0.9);

glutSolidTorus(0.5, 0.5, 50, 50);

glPopMatrix();

**левая**

glColor3d(0.8, 1, 0);

glPushMatrix();

glTranslated(-0.3, -0.15, -1.5);

glRotated(90, 0, 1, 0);

glScaled(0.1, 0.1, 0.9);

glutSolidTorus(0.5, 0.5, 50, 50);

glPopMatrix();

glColor3d(0.8, 1, 0);

glPushMatrix();

glTranslated(0.2, -0.15, -0.9);

glRotated(90, 0, 1, 0);

glScaled(0.1, 0.1, 0.9);

glutSolidTorus(0.5, 0.5, 50, 50);

glPopMatrix();

\*/**Вентилятор самолета:** Поскольку вентилятор самолета (рис. 1) расположена в верхней части фюзеляжа, сначала мы изменим значение x (x> 0) в функции glTranslatef. Затем, поскольку деталь растягивается вдоль оси Oy, мы увеличиваем размер y и уменьшаем размер x, z для функции glScalef. Наконец, мы вызываем glutSolidSphere, чтобы нарисовать круг с радиусом (внутри и снаружи), равным 1; количество линий широты и долготы равно 50. В сочетании с glColor3f (желто-зеленый) мы получаем вентилятор самолета

**В случае только 1 вентилятора:**

glPushMatrix();

glRotated(10 \* a, 1, 0, 0);

fan(); //fan2() в случае 2 вентиляторов

glPopMatrix();

**В случае только 2 вентилятора:**

glPushMatrix();

glTranslated(0,0,0);

glPushMatrix();

glRotated(10\*a,1,0,0);

fan();

glPopMatrix();

glPushMatrix();

glRotated(-(10 \* a), 1, 0, 0);

fan2();

glPopMatrix();

glPopMatrix();

Где **функция fan():** (в случае только 1 вентилятора)

glColor3d(0.5, 1, 0);

glPushMatrix();

glTranslated(2.5, 0, 0);

glScaled(0.2, 1.5, 0.05);

glutSolidSphere(1, 30, 30);

glPopMatrix();

В случае 2 вентиляторов мы вызываем и функцию вентилятора, и функцию fan2:

Для случая 2 вентиляторов мы вызываем и функцию вентилятора, и функцию fan2. В этой функции fan2 мы будем вызывать функцию glRotatef с углом поворота, противоположным функции вентилятора.

**функция fan2():**

glColor3d(0.5, 1, 0);

glPushMatrix();

glTranslated(2.5, 0, 0);

glScaled(0.2, 1.5, 0.05);

glutSolidSphere(1, 30, 30);

glPopMatrix();

\*/**Хвостовой самолет** Поскольку хвостовая часть самолета (рисунок 1) расположена в верхней части фюзеляжа, мы сначала меняем значение x (x> 0) и значение z (z <0 для левого хвоста, z> 0 для правого хвоста. .); измените значение x, y для верхнего хвоста в glTranslatef. Затем, поскольку деталь растягивается по оси Oz (для правого и левого хвостов) и растягивается по оси Oy (для верхнего хвоста), мы увеличиваем размер z и уменьшаем размеры x, z для функции glScalef. Для левого и правые хвосты; увеличить размер y, уменьшить размер x, z в функции glScalef для верхнего хвоста. Наконец, мы вызываем glutSolidCube, чтобы нарисовать куб с радиусом 1. В сочетании с glColor3f (голубой) мы получаем хвост плоскости.

**Правый хвост**

glColor3d(0.8, 1, 0);

glPushMatrix();

glTranslated(0.4, 0, 1.5);

glRotated(-30, 0, 1, 0);

glScaled(0.7, 0.1, 3);

glRotated(10, 0, 1, 0);

glutSolidCube(1);

glPopMatrix();

**Левый хвост:**

glColor3d(0.8, 1, 0);

glPushMatrix();

glTranslated(0.4, 0, -1.5);

glRotated(30, 0, 1, 0);

glScaled(0.7, 0.1, 3);

glRotated(-10, 0, 1, 0);

glutSolidCube(1);

glPopMatrix();

glPopMatrix();

**Верхний хвост:**

glColor3d(0.8, 1, 0);

glPushMatrix();

glTranslated(-2.7, 0.5, 0);

glRotated(45, 0, 0, 1);

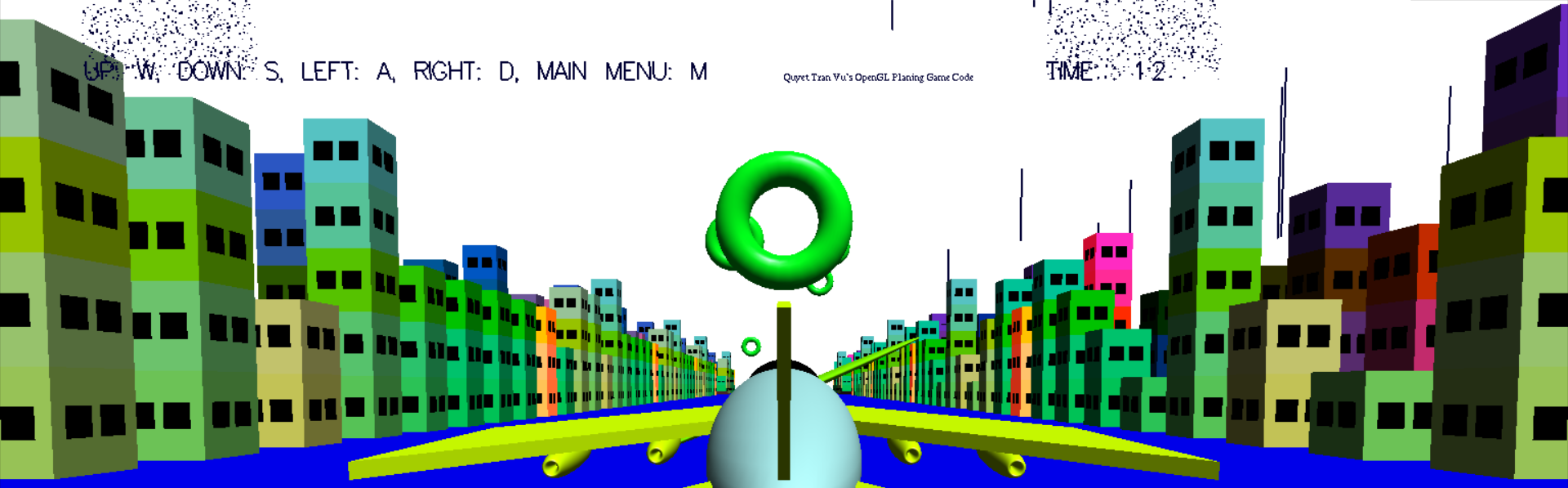
glScaled(0.8, 2, 0.1);

glRotated(-20, 0, 0, 1);// использование это, чтобы сделать наклон более красивым

glutSolidCube(0.5);

glPopMatrix();

**Строительство многоэтажных домов**

****

**Рис 3 Модель многоэтажных домов, который будет построен.**

**Задача состоит в том, чтобы собрать следующие детали:**

-Рисование высотных строительных блоков

-Объединение домов

**Идеи деталей, пояснения и конкретный код:**

**Главная идея:**

-Использование встроенные функции (glutSolidCube: рисование твердой формы в сочетании с принципами вращения и изменения размера объекта (glScalef, glRotatef). GlColor3f мы поместим в 3 массива r, g, b. Изменяя это значение, мы получат блоки разного цвета.

**Конкретный код:**

\*/ **Рисование высотных строительных блоков:** Блоки нарисованы в порядке от меньшего к большему. Сначала поместите мини-блок в корневую позицию (0,0, 0). Два блока средней высоты меняем значение x (x> 0 для правого блока и x <0 для левого блока); и значение z (z> 0 для предыдущего блока, z <0 для последнего) для двух самых высоких зданий позади в функции glTranslatef. Затем мы увеличиваем ширину (z) для двух блоков средней высоты и длину (x) для двух самых высоких зданий в функции glScalef. Наконец, мы вызываем функцию glSolidCube, чтобы нарисовать куб с параметром радиуса 1, и мы получаем 5 блоков.

void singleTolaHouse(int R, int G, int B)

{

glColor3d(r[R % 11], g[G % 11], b[B % 11]);

// **Детский блок**

glPushMatrix();

glTranslated(0, 0, 0);

glutSolidCube(1);

glPopMatrix();

// **Два блока средней-высокой**

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(0.2, 0, 0);

glScaled(0.3, 0.3, 1.001);

glutSolidCube(1);

glPopMatrix();

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(-0.2, 0, 0);

glScaled(0.3, 0.3, 1.001);

glutSolidCube(1);

glPopMatrix();

// **Два высоких блока позади**

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(0, 0, 0.2);

glScaled(1.001, 0.3, 0.3);

glutSolidCube(1);

glPopMatrix();

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(0, 0, -0.2);

glScaled(1.001, 0.3, 0.3);

glutSolidCube(1);

glPopMatrix();

}

**\*/Объединение домов:** Чтобы комбинировать блоки и отображать их несколько раз, мы будем использовать цикл с переменным параметром, который мы будем использовать для создания вертикального расстояния между блоками (параметр Oy функции glTranslatef). В конце концов, нам просто нужно вызвать вышеупомянутую встроенную функцию рисования блоков в цикле, и мы получим много блоков, отображаемых непрерывно с разными цветными блоками.

void house(int n, int R, int G)

{

for (int i = 0; i < n; i++)

{

glPushMatrix();

glTranslated(0, 0.8+i, 0);

singleTolaHouse(G, R, i);

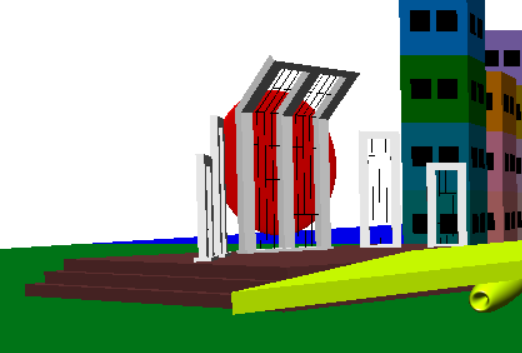
glPopMatrix();

}

}

Где функция singleTolaHouse (Рисование высотных строительных блоков-сверху)

**Строительство солнечной батареи**

****

**Рис 4 Модель блока солнечных батарей, который будет построен**

**Задача состоит в том, чтобы собрать следующие детали:**

- Нарисование постамент (по очереди снизу вверх)

- Строительство главного электрического столба

- Построение вспомогательные опоры, расположенные рядом с основной опорой.

- Нарисование круглую солнечную панель за электрическими столбами

**Идеи деталей, пояснения и конкретный код:**

**Главная идея:**

- Использование встроенных функций (glutSolidSphere: рисование сплошной сферы, glutSolidCube: рисование сплошного куба, glutSolidTorus: рисование сплошного тора) в сочетании с принципами поворота и изменения размера объекта (glScalef, glRotatef).

**Конкретный код:** Чтобы нарисовать аккумуляторную систему, мы будем рисовать снизу вверх (от коричневого постамента к полюсам и детализировать каждый провод, и, наконец, нарисуем круглую солнечную панель).

\*/**нарисование постамент (по очереди снизу вверх):** Поскольку коричневый пьедестал (рис. 4) состоит из трех вертикальных плоскостей, мы по очереди меняем значение y в glTranslatef. Затем, поскольку пьедестал простирается вдоль оси Oz и продолжается вдоль оси Ox, мы увеличиваем размеры x, z для функции glScalef. Наконец, мы вызываем glutSolidCube, чтобы нарисовать куб с радиусом 1. В сочетании с glColor3f (голубой) мы получаем коричневую основу.

glColor3d(0.4, 0.2, 0.2);// коричневый

glPushMatrix();

glTranslated(0, 1.55, 0);

glScaled(2, 0.05, 1.5);

glutSolidCube(1);

glPopMatrix();

glColor3d(0.4, 0.2, 0.2);

glPushMatrix();

glTranslated(0, 1.6, 0);

glScaled(1.9, 0.05, 1.4);

glutSolidCube(1);

glPopMatrix();

//glColor3d(0.4,0.2,0.2);

//glColor3d(1,0.8,0.7);

glColor3d(0.4, 0.2, 0.2);

glPushMatrix();

glTranslated(0, 1.65, 0);

glScaled(1.8, 0.05, 1.3);

glutSolidCube(1);

glPopMatrix();

\*/**Строительство главного электрического столба:**

В этой части мы по очереди построим:

1 / пьедестал солнечной панели

2 / Построение 3 основных полюса посередине

3 / Построение провода для этих 3 стержней

4 / Построение 3 диагональных опоры наверху

5 / Построение провода для этих 3 стержней

/// **пьедестал солнечной панели:** Поскольку подставка для солнца (рис. 4) плоская в основании трех полюсов питания, мы меняем значение y в glTranslatef. Затем, поскольку пьедестал простирается вдоль оси Oz и продолжается вдоль оси Ox, мы увеличиваем размеры x, z для функции glScalef. Наконец, мы вызываем glutSolidCube, чтобы нарисовать куб с радиусом 1. В сочетании с glColor3f (голубой) мы получаем лежак.

glColor3d(1, 1, 1);

glPushMatrix();

glTranslated(0, 1.68, -0.4);

glScaled(0.5, 0.02, 0.08);

glutSolidCube(1);

glPopMatrix();

///**Построение 3 основных полюса посередине и провода для этих 3 стержней:** В этой части мы также строим по очереди снизу вверх, применяя принципы увеличения и уменьшения размера, изменяя положение для получения соответствующего объекта.

**Примечание:** -Когда мы закончим построение вертикального провода для построения горизонтального провода, мы должны сбросить относительное положение, изменив значения x, z в функции glTranslatef.

-Когда мы закончим построение вертикального столба, чтобы построить диагональный столб выше, мы должны сбросить относительное положение, изменив значение x, y в функции glTranslatef. В то же время, чтобы отобразить диагональные полюса мощности, мы вызовем функцию glRotated для поворота на угол 45 градусов в направлении x перед началом рисования.

/// **Центральный полюс (главной солнечной панели)**

glPushMatrix();

glTranslated(0, 1.99, -0.4);

glScaled(0.06, 0.7, 0.04);

glutSolidCube(1);

glPopMatrix();

/// **Постоянный провод этой опоры (порядок рисования: 3 провода сверху, 3 провода снизу)**

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(0.07, 1.99, -0.4);

glScaled(0.003, 0.7, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.11, 1.99, -0.4);

glScaled(0.003, 0.7, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.15, 1.99, -0.4);

glScaled(0.003, 0.7, 0.003);

glutSolidCube(1);

glPopMatrix();

///

glPushMatrix();

glTranslated(-0.22, 0, 0);

glPushMatrix();

glTranslated(0.07, 1.99, -0.4);

glScaled(0.003, 0.7, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.11, 1.99, -0.4);

glScaled(0.003, 0.7, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.15, 1.99, -0.4);

glScaled(0.003, 0.7, 0.003);

glutSolidCube(1);

glPopMatrix();

glPopMatrix();

/// **Горизонтальный провод этого столба**

glPushMatrix();

glTranslated(2.2, 0, -0.1); // репозиция относительно стоячей проволоки

glScaled(4.2, 1, 1);

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(-0.528, 1.85, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 2.02, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 2.18, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

glColor3d(1, 1, 1);

glPopMatrix();

/// **Самый дальний провод этого полюса**

glColor3d(1, 1, 1);

glPushMatrix();

glTranslated(-0.22, 1.99, -0.4);

glScaled(0.06, 0.7, 0.04);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.22, 1.99, -0.4);

glScaled(0.06, 0.7, 0.04);

glutSolidCube(1);

glPopMatrix();

/// **Построение столбы по диагонали поверх солнечной панели**

glPushMatrix();

glTranslated(0, 0.743, -1.424); // репозиция относительно вертикальных столбов

glRotated(45, 1, 0, 0);

glPushMatrix();

glTranslated(0, 1.99, -0.4);

glScaled(0.06, 0.3, 0.04);

glutSolidCube(1);

glPopMatrix();

//glColor3d(1,1,1);

glPushMatrix();

glTranslated(-0.22, 1.99, -0.4);

glScaled(0.06, 0.3, 0.04);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.22, 1.99, -0.4);

glScaled(0.06, 0.3, 0.04);

glutSolidCube(1);

glPopMatrix();

// **Устройство горизонтального ограждения верхнего полюса мощности**

glPushMatrix();

glTranslated(0, 2.15, -0.4);

glScaled(0.5, 0.04, 0.04);

glutSolidCube(1);

glPopMatrix();

/// **Сделание вертикальные провода для столбов выше**

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(0.07, 1.99, -0.4);

glScaled(0.003, 0.277, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.11, 1.99, -0.4);

glScaled(0.003, 0.277, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.15, 1.99, -0.4);

glScaled(0.003, 0.277, 0.003);

glutSolidCube(1);

glPopMatrix();

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(-0.22, 0, 0);// сбросить положение проводов стоящих с другой стороны

glPushMatrix();

glTranslated(0.07, 1.99, -0.4);

glScaled(0.003, 0.277, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.11, 1.99, -0.4);

glScaled(0.003, 0.277, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.15, 1.99, -0.4);

glScaled(0.003, 0.277, 0.003);

glutSolidCube(1);

glPopMatrix();

glPopMatrix();

/// **Сделание горизонтальные провода для верхних панелей (включая 2 горизонтальных провода в центре).**

glPushMatrix();

glTranslated(2.2, 0, -0.1);

glScaled(4.2, 1, 1);

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(-0.528, 1.85, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 2, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

/\*glPushMatrix();

glTranslated(-0.528, 2.15, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();\*/

glColor3d(1, 1, 1);

glPopMatrix();

glPopMatrix();

/\* **Построение вспомогательные опоры, расположенные рядом с основной опорой:** В этом разделе мы будем строить в порядке поворота слева направо (2 полюса слева, затем 2 справа). Для каждого столба мы построим 2 вертикальных стержня, а затем 2 горизонтальных стержня. В то же время мы также построим вертикальные и горизонтальные провода для каждого полюса. Мы продолжим применять принципы изменения расстояния и размера для отображения соответствующего объекта.

**Примечание.** Поскольку все полюса, расположенные рядом с тремя основными полюсами, наклонены под углом 45 градусов, мы придем к функции glRotated, которая вращается вдоль оси Ox на угол 45 градусов.

/// **Построение первый левый столб (в порядке построения блока переднего края к нижнему горизонтальному блоку).**

glColor3d(1, 1, 1);

glPushMatrix();

glTranslated(0.1, 0, -0.4);

glRotated(45, 0, 1, 0);

glPushMatrix();

glTranslated(-0.605, 1.94, -0.3);

glScaled(0.045, 0.65, 0.03);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.45, 1.94, -0.3);

glScaled(0.045, 0.65, 0.03);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 2.258, -0.3);

glScaled(0.199, 0.04, 0.03);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 1.68, -0.3);

glScaled(0.199, 0.02, 0.06);

glutSolidCube(1);

glPopMatrix();

/// **Сделание постоянный провод для этого столба**

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(-0.64, -0.05, 0.1);

glScaled(1, 1.02, 1);

glPushMatrix();

glTranslated(0.078, 1.99, -0.4);

glScaled(0.003, 0.56, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.11, 1.99, -0.4);

glScaled(0.003, 0.56, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.145, 1.99, -0.4);

glScaled(0.003, 0.56, 0.003);

glutSolidCube(1);

glPopMatrix();

glPopMatrix();

/// **Горизонтальный провод этого столба**

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(-0.528, 1.85, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 2, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 2.15, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

glColor3d(1, 1, 1);

glPopMatrix();

/// **Построение второй левый полюс**

glPushMatrix();

glTranslated(0.65, 0, 0.3);

glRotated(-45, 0, 1, 0);

glPushMatrix();

glTranslated(-0.605, 1.94, -0.3);

glScaled(0.045, 0.65, 0.03);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.45, 1.94, -0.3);

glScaled(0.045, 0.65, 0.03);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 2.258, -0.3);

glScaled(0.199, 0.04, 0.03);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 1.68, -0.3);

glScaled(0.199, 0.02, 0.06);

glutSolidCube(1);

glPopMatrix();

/// **Сделание постоянный провод для этого столба**

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(-0.64, -0.05, 0.1);

glScaled(1, 1.02, 1);

glPushMatrix();

glTranslated(0.078, 1.99, -0.4);

glScaled(0.003, 0.56, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.11, 1.99, -0.4);

glScaled(0.003, 0.56, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.145, 1.99, -0.4);

glScaled(0.003, 0.56, 0.003);

glutSolidCube(1);

glPopMatrix();

glPopMatrix();

glColor3d(1, 1, 1);

/// **Сделание для этого столба горизонтальные провода.**

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(-0.528, 1.85, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 2, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 2.15, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

glColor3d(1, 1, 1);

glPopMatrix();

/// **Построение первый столб справа**

glPushMatrix();

/// pasher piller left 1

glTranslated(0.06, 0, 0.14);

glPushMatrix();

glTranslated(-0.2, 0, -0.31);

glRotated(45, 0, 1, 0);

glPushMatrix();

glTranslated(-0.605, 1.88, -0.3);

glScaled(0.045, 0.4, 0.03);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.45, 1.88, -0.3);

glScaled(0.045, 0.4, 0.03);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 2.08, -0.3);

glScaled(0.2, 0.04, 0.03);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 1.68, -0.3);

glScaled(0.199, 0.02, 0.06);

glutSolidCube(1);

glPopMatrix();

/// **Сделание постоянный провод для этого столба**

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(-0.641, 0.43, 0.1);

glScaled(1, 0.73, 1);

glPushMatrix();

glTranslated(0.078, 1.99, -0.4);

glScaled(0.003, 0.56, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.11, 1.99, -0.4);

glScaled(0.003, 0.56, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.145, 1.99, -0.4);

glScaled(0.003, 0.56, 0.003);

glutSolidCube(1);

glPopMatrix();

glPopMatrix();

/// **Сделание для этого столба горизонтальные провода.**

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(-0.528, 1.8, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 1.96, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

glColor3d(1, 1, 1);

///ROD

glPopMatrix();

/// **Построение второй правый столб**

glPushMatrix();

glTranslated(0.83, 0, 0.39);

glRotated(-45, 0, 1, 0);

glPushMatrix();

glTranslated(-0.605, 1.88, -0.3);

glScaled(0.045, 0.4, 0.03);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.45, 1.88, -0.3);

glScaled(0.045, 0.4, 0.03);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 2.1, -0.3);

glScaled(0.199, 0.04, 0.03);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 1.68, -0.3);

glScaled(0.199, 0.02, 0.06);

glutSolidCube(1);

glPopMatrix();

/// **Сделание для этого столба горизонтальные провода.**

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(-0.528, 1.8, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-0.528, 1.96, -0.3);

glScaled(0.1, 0.003, 0.003);

glutSolidCube(1);

glPopMatrix();

glColor3d(1, 1, 1);

/// **Сделание постоянный провод для этого столба**

glColor3d(0, 0, 0);

glPushMatrix();

glTranslated(-0.641, 0.43, 0.1);

glScaled(1, 0.73, 1);

glPushMatrix();

glTranslated(0.078, 1.99, -0.4);

glScaled(0.003, 0.56, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.11, 1.99, -0.4);

glScaled(0.003, 0.56, 0.003);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.145, 1.99, -0.4);

glScaled(0.003, 0.56, 0.003);

glutSolidCube(1);

glPopMatrix();

glPopMatrix();

glColor3d(1, 1, 1);

glPopMatrix();

glPopMatrix();

/// **Нарисование круглую солнечную панель за электрическими столбами:** Поскольку эта солнечная панель находится сзади (дальше всего) и на определенной высоте, мы меняем значения x и y в функции glTranslatef. Что касается размера, мы изменим значения x, y, z, чтобы они были меньше, чтобы получить относительно плоскую солнечную панель. Наконец, мы вызываем функцию gluSolidSphere, чтобы нарисовать сферу радиусом 1 и количеством линий широты и долготы, равным 50.

glColor3d(1, 0, 0);

glPushMatrix();

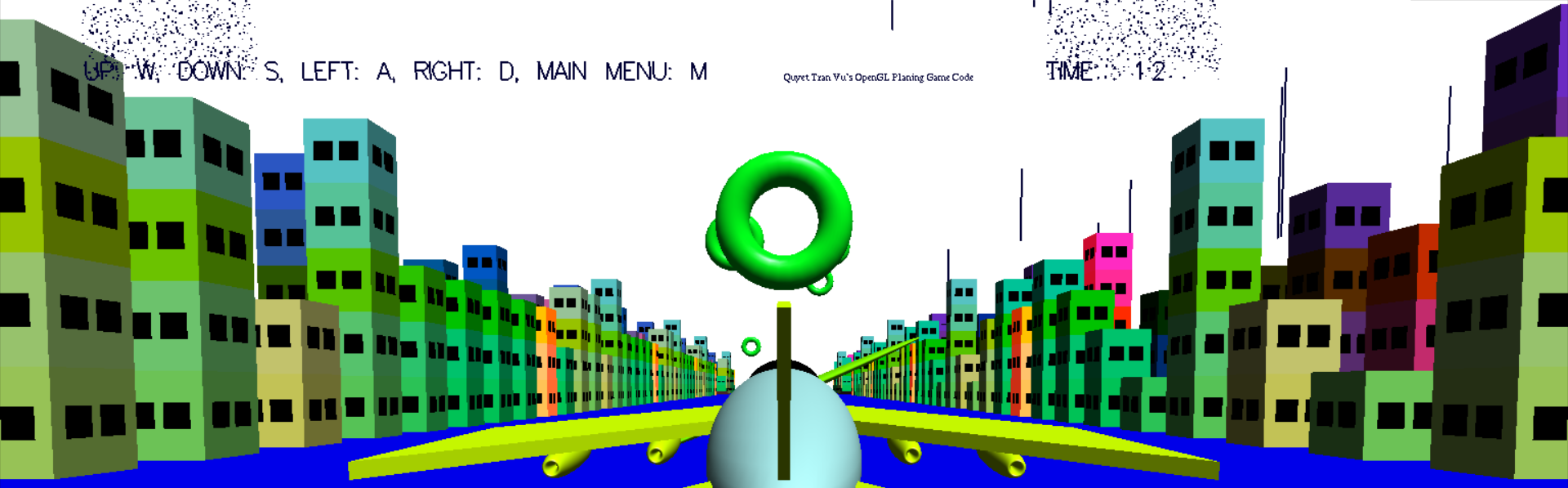
glTranslated(0, 2.1, -0.44);

glScaled(0.35, 0.35, 0.01);

glutSolidSphere(1, 50, 50);

glPopMatrix();

**Экологическое строительство**

****

**Рис 5 Изображение, показывающее игровую среду.**

**Задача состоит в том, чтобы собрать следующие детали:**

- Выполнение рисование 2 противоположных площадок, где находится постамент полюса (из которых эффект текстуры может быть добавлен позже).

-Выполнение чертежа среды

-Выполнение комбинации рисования среды (объединение компонентов вместе, но не включая другие эффекты)

**Идеи деталей, пояснения и конкретный код:**

**Главная идея:**

- Как и раньше с многоэтажным блоком, для непрерывного отображения объектов мы войдем в цикл с параметром EN\_SIZE (этот параметр указывает размер для определения земли (если увеличить, он будет работать) сцены замедленного движения)). При этом совместите принципы размера, чтобы получить желаемый объект.

**Конкретный код:**

\*/**Выполнение рисование 2 противоположных площадок, где находится постамент полюса (из которых эффект текстуры может быть добавлен позже):** В этой части мы дважды вызовем функцию для рисования блока солнечных элементов и электрического полюса. Поскольку они расположены напротив друг друга в направлении x, перед вызовом мы поместим соответствующие значения x в функцию glTranslatef. Как объяснялось выше, размер площадки будет изменяться в зависимости от значения переменной EN\_SIZE в функции glScaled.

void soheedMinarEnv() {

/// Земля, поверхность внизу

glColor3d(0, 0.5, 0.1);

glPushMatrix();

glTranslated(0, 0, 0);

glScaled(EN\_SIZE \* 2, 0.3, EN\_SIZE \* 2);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-8, -2.7, -5);

glRotated(65, 0, 1, 0);

glScaled(2, 2, 2);

drawShohidMinar();

glPopMatrix();

glPushMatrix();

glTranslated(8, -2.7, -5);

glRotated(-65, 0, 1, 0);

glScaled(2, 2, 2);

drawShohidMinar();

glPopMatrix();

}

Где Функция drawShohidMinar(): (**Строительство солнечной батареи сверху)**

\*/**Выполнение чертежа среды**

void environment(int n) {

/// **Призывая землю, поверхность ниже**

glPushMatrix();

glTranslated(0, 0, 0);

glScaled(EN\_SIZE \* 2, 0.3, EN\_SIZE \* 2);

glTexCoord2f(EN\_SIZE \* 2, 0.3);

glutSolidCube(1);

glPopMatrix();

// **Сделание зеленые круги**

glColor3d(0, 1, 0.1);

glPushMatrix();

glTranslated(torusPosX[n], torusPosY[n], 0);

glScaled(0.3, 0.3, 0.3);

glutSolidTorus(1, 3, 30, 30);

glPopMatrix();

// **Выполнение командный блок, чтобы вызвать цикл для домашнего дисплея. Здесь используется переменная массива tola: двумерный массив, содержащий соответствующие значения настройки для параметра при строительстве дома (function house, singletolaHouse)**

for (int i = -(EN\_SIZE / 2) + 1; i < (EN\_SIZE / 2); i += 2) {

for (int j = -(EN\_SIZE / 2) + 1; j < (EN\_SIZE / 2); j += 2) {

if (tola[i + (EN\_SIZE / 2) + 1][j + (EN\_SIZE / 2) + 1] != 0) {

glPushMatrix();

glTranslated(i, 0, j);

house(tola[i + (EN\_SIZE / 2) + 1][j + (EN\_SIZE / 2) + 1], i, j);

glPopMatrix();

}

else if (i >= -5 && i <= 5) {

}

else {

tola[i + (EN\_SIZE / 2) + 1][j + (EN\_SIZE / 2) + 1] = (rand() % 5) + 1;

glPushMatrix();

glTranslated(i, 0, j);

house(tola[i + (EN\_SIZE / 2) + 1][j + (EN\_SIZE / 2) + 1], i, j);

glPopMatrix();

}

}

}

}

\*/**Выполнение комбинации рисования среды (объединение компонентов вместе, но не включая другие эффекты)**

Значение переменных, используемых в функции объединения:

-rotX, rotY, rotZ: указанные значения вращаются в направлениях x, y, z плоскости

-tZ до tZ6: установка значений для переменных, работающих со скоростью, соответствующей каждому изменению среды с tZ

-tX, tY: расположение плоскости

-speed: установка значения скорости самолета.

- n: максимальное количество круговых перекрестков, отображаемых в 1 кадре среды.

Правило реализации: сначала мы вызываем начальную настройку плоскости (через значения rotX, rotY, rotZ). Затем, ограничивая значения tX, tY (если не ограничивать, плоскость может быть перевернута), мы соответственно вызываем функцию рисования окружающей среды (показываем количество витков, равное значению n) и случайным образом вводим функцию для рисования солнечных лучей. ячеек и электрических столбов.Наконец, мы изменяем значение tZ (значение, которое помогает устанавливать скорость каждый раз при изменении окружающей среды), соответствующее скорости окружающей среды.

void draw() {

double t = glutGet(GLUT\_ELAPSED\_TIME) / 1000.0;

double a = t \* 90.0;

TIME = t;

/// **Вызов самолет**

if (rotX > 11)rotX = 11;

if (rotX < -11)rotX = -11;

if (rotZ > 10)rotZ = 10;

if (rotZ < -15)rotZ = -15;

glPushMatrix();

glTranslated(0, 1, 0);

glRotated(90, 0, 1, 0);

glRotated(5, 0, 0, 1);

glRotated(rotX, 1, 0, 0);

glRotated(rotY, 0, 1, 0);

glRotated(rotZ, 0, 0, 1);

glScaled(0.4, 0.4, 0.4);

plane();

glPopMatrix();

///Gọi tới môi trường

if (tX >= 4.1)tX = 4.1;

if (tX <= -4.1)tX = -4.1;

if (tY > 0.1)tY = 0.1;

if (tY < -15)tY = -15;

glPushMatrix();

glTranslated(tX, tY, tZ);

environment(2);

glPopMatrix();

glPushMatrix();

glTranslated(tX, tY, tZ1);

soheedMinarEnv();

glPopMatrix();

glPushMatrix();

glTranslated(tX, tY, tZ2);

environment(3);

glPopMatrix();

glPushMatrix();

glTranslated(tX, tY, tZ3);

environment(1);

glPopMatrix();

glPushMatrix();

glTranslated(tX, tY, tZ4);

environment(5);

glPopMatrix();

glPushMatrix();

glTranslated(tX, tY, tZ5);

environment(4);

glPopMatrix();

glPushMatrix();

glTranslated(tX, tY, tZ6);

environment(2);

glPopMatrix();

**// Конечно, независимо от того, как быстро движется самолет, среда будет преобразовываться соответствующим образом.**

tZ += speed;

tZ1 += speed;

tZ2 += speed;

tZ3 += speed;

tZ4 += speed;

tZ5 += speed;

tZ6 += speed;

if (tZ >= 20)tZ = -110;

if (tZ1 >= 20)tZ1 = -110;

if (tZ2 >= 20)tZ2 = -110;

if (tZ3 >= 20)tZ3 = -110;

if (tZ4 >= 20)tZ4 = -110;

if (tZ5 >= 20)tZ5 = -110;

if (tZ6 >= 20)tZ6 = -110;

if (rotX > 0)rotX -= angleBackFrac;

if (rotX < 0)rotX += angleBackFrac;

if (rotY > 0)rotY -= angleBackFrac;

if (rotY < 0)rotY += angleBackFrac;

if (rotZ > 0)rotZ -= angleBackFrac;

if (rotZ < 0)rotZ += angleBackFrac;

speed += 0.0002;

if (speed >= 0.7)speed = 0.7;

}

**Комбинирование компонентов и добавление эффектов повышает качество игры.**

**Задача состоит в том, чтобы собрать следующие детали:**

-Изменить исходный код, заменить переменные, отредактировать несколько частей дисплея, которые неисправны или не работают.

-Текстовые эффекты (включая множество стилей).

- Эффекты фоновой музыки.

- Эффект динамического пола (за счет использования текстуры).

-Эффект дождя (погода и текст).

-Эффект регулировки обработки манипуляций с мышью

-Влияние формата кадра дисплея (функция resize: очень важна, если в программе отсутствует программа отображения белой рамки).

-Постоянный эффект изменения цвета самолета

-Показание все заявленные выше объекты

-Также используйте множество различных эффектов (включая свет, цвет, глубину и т. Д.), Объявленных через группу команд glEnable и glDisable.

**Идеи деталей, пояснения и конкретный код:**

**Главная идея:**

- В этом разделе, в зависимости от эффекта и цели использования, мы соответственно поместим этот эффект в то место, где инициализируется объект.

\*/**Изменить исходный код, заменить переменные, отредактировать несколькочастей дисплея, которые неисправны или не работают:** В процессе непрерывного изучения кода с редактированием старайтесь выбирать параметры, чтобы получить лучшую модель.

\*/**Текстовые эффекты (включая множество стилей**): Этот раздел включает 4 различных типа рисования текста:

-Рисование текста с использованием растрового изображения (объект в openGL, через который можно загрузить текстуру).

-Рисование текста светлыми линиями (включая две функции обводки и обводки2)

-Написание нечетные символы

**Принцип построения** здесь заключается в использовании функции рисования символов (glutStrokeCharacter и glutBitmapCharacter) для указания типа шрифта и отображаемого символа (символы здесь просматриваются, пока не встретится пробел)).

**Конкретный код:** В зависимости от цели и потребностей использования в этом разделе, соответственно, вызовите 1, 2, 3 или все 4 функции в функции комбинирования компонентов**.**

void drawBitmapText(const char \*str, float x, float y, float z)

{

const char \*c;

glRasterPos3f(x, y + 8, z);

for (c = str; \*c != '\0'; c++)

{

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_10, \*c);

}

}

void drawStrokeText(const char\* str, int x, int y, int z)

{

const char \*c;

glPushMatrix();

glTranslatef(x, y + 8, z);

glScalef(0.002f, 0.002f, z);

for (c = str; \*c != '\0'; c++)

{

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, \*c);

}

glPopMatrix();

}

void drawStrokeText2(const char\* str, int x, int y, int z)

{

const char \*c;

glPushMatrix();

glTranslatef(x, y + 8, z);

glScalef(0.005f, 0.005f, z);

for (c = str; \*c != '\0'; c++)

{

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, \*c);

}

glPopMatrix();

}

void drawStrokeChar(char c, float x, float y, float z)

{

glPushMatrix();

glTranslatef(x, y + 8, z);

glScalef(0.002f, 0.002f, z);

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, c);

glPopMatrix();

}

\*/**Эффекты фоновой музыки:** Чтобы иметь возможность добавлять фоновую музыку в игру в OpenGL, мы будем использовать функцию sndPlaySound с двумя параметрами (первый параметр - это название песни в формате BMP с примечанием, что оно должно находиться в том же каталоге, что и игра.; второй параметр определяет синхронизацию). Наконец, мы поместим эту функцию фоновой музыки в основную функцию.

**Конкретный код:**

sndPlaySound("[MP3DOWNLOAD.TO] Yiruma - River Flows in You-HQ.wav", SND\_ASYNC);

\*/**Эффект динамического пола (за счет использования текстуры).**

Чтобы иметь возможность создать эффект динамического пола, нам понадобятся две функции LoadBitmap и LoadTexture. Затем в функции среды (environment) мы устанавливаем функцию LoadGLTextures при инициализации земли, содержащей солнечные элементы и электрические столбы.

**Принцип работы:** Для функции AUX\_RGBImageRec параметром будет имя файла того типа текстуры, который мы хотим показать (здесь зеленая полоса). Затем эта функция сделает это, проанализирует растровое изображение и вернет данные растрового изображения (при успешном открытии файла).

**LoadGLTextures**: Со своей стороны, эта функция инициализирует объект начального указателя типа AUX\_RGBImageRec. Затем используйте эту переменную-указатель для загрузки данных в имя изображения. В случае успеха она вызовет функцию glTexParameteri для установки переменных текстуры, а затем через функцию glTexImage2D для установки размера отображаемого изображения (sizeX, sizeY: length и ширина области загруженной Текстурой)). В случае неудачи просто удалите данные этой переменной-указателя с помощью функции free.

**Конкретный код:**

GLuint texture; // управление текстурой

char texture\_name[100]={"brics.bmp"}; // сохранить имя файла текстуры

AUX\_RGBImageRec \*LoadBMP(char \*Filename)

{

FILE \*File = NULL;

if (!Filename)

return NULL;

fopen\_s(&File, Filename, "r");

if (File)

{

fclose(File);

return auxDIBImageLoadA((LPCSTR)Filename);

}

return NULL;

}

// Функция загрузки текстуры

bool LoadGLTextures()

{

int ret = false;

AUX\_RGBImageRec \*texture\_image = NULL;

if (texture\_image = LoadBMP(texture\_name))

{

glGenTextures(1, &texture); // Запустить процесс создания текстуры.

glBindTexture(GL\_TEXTURE\_2D, texture);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

// сопоставить данные растрового изображения с текстурой. glTexImage2D(GL\_TEXTURE\_2D, 0, 3, texture\_image->sizeX,

texture\_image->sizeY, 0, GL\_RGB,

GL\_UNSIGNED\_BYTE, texture\_image->data);

}

else

{

ret = false;

if (texture\_image)

{

if (texture\_image->data)

free(texture\_image->data);

free(texture\_image);

}

}

return ret;

}

\*/ **Эффект дождя (погода и текст).**

Чтобы нарисовать эффект дождя для игры, мы поместим этот код в файл RainEffect.h.

**Принцип построения** реализован путем создания массива дождя (дождя) типа структуры. Затем мы инициализируем капли дождя случайными свойствами (используя функцию rand). Затем, чтобы создать длинные капли дождя, мы соединим эти точки вместе. Наконец, с помощью переменной плотности дождя мы используем цикл для капель дождя.

**Примечание**: чтобы создать каплю дождя, мы должны учитывать значение y и соответствующее значение высоты.

**Конкретный код:**

#define RAINSIZE 1000 // переменная, определяющая плотность дождевых капель (чем она больше, тем четче выпадет количество дождя)

int winWidth = 1000, winHeight = 1000;

float rotationAngle = 0; // атрибут определяет угол падения дождя

// **создать структуру, содержащую атрибуты, относящиеся к капле дождя (координаты; движение; радиус; длина; угол падения; угол движения)**

struct drop {

float x = 400;

float y = 400;

float inc = 0.01;

float radius = 10;

float scale = 1.0;

float rotationAngle = 0;

float rotationInc = 1;

};

drop rain[RAINSIZE];

void initRain() {

for (int i = 0; i < RAINSIZE; i++) {

rain[i].x = rand() % winWidth;

rain[i].y = rand() % winHeight;

rain[i].inc = 1.5 + (float)(rand() % 100) / 1000.0;

rain[i].radius = (float)(rand() % 8);

rain[i].scale = (float)(rand() % 20000) / 1000.0;

rain[i].rotationAngle = (float)(rand() % 3000) / 1000.0;

rain[i].rotationInc = (float)(rand() % 100) / 1000.0;

if ((rand() % 100) > 1000) {

rain[i].rotationInc = -rain[i].rotationInc;

}

}

}

// **Нарисование дождь (соединение 2 точки прямой линией) (обратите внимание, что я соединю 2 точки с координатами (x, y) и точкой (x, y + радиус со второй точкой))**

void drawParticleShape(int i) {

glBegin(GL\_POINTS);

glVertex2d(rain[i].x, rain[i].y);

glEnd();

glBegin(GL\_LINES);

glVertex2d(rain[i].x, rain[i].y);

glVertex2d(rain[i].x, rain[i].y + rain[i].radius \* 2);

glEnd();

}

**// Реализовать создание капли**

void drawDrop(int i) {

glColor3f(0.0, 0.0, 1.0); //Синий

glLineWidth(2);

drawParticleShape(i);

rain[i].y -= rain[i].inc; // каждый раз, когда дождь падает до конца длины, я вычитаю значение движения и переназначаю исходное значение высоты для капли дождя

if (rain[i].y < 0) {

rain[i].y = winHeight;

}

}

**// Вызов функции дождя, используя цикл, связанный с плотностью дождя**

void drawRain() {

for (int i = 0; i < RAINSIZE; i++) {

drawDrop(i);

}

}

**\*/Эффект регулировки обработки манипуляций с мышью:** Чтобы позволить пользователю управлять мышью, мы должны объявить ключевую функцию и функцию glutKeyBoardFunc (в основной функции). Чтобы иметь возможность изменять, мы снова будем использовать переменные tX, tY (положение самолета) и rotX, rotY, rotZ (значения, указанные для поворота в направлении x, y, z самолета). Через клавиши, которые соответственно изменяют это значение, мы получим управление объектом. Функция glutPostReDisplay () сделает эту функцию повторяемой

Значение кнопок:

-W: двигаться вверх

-D: двигаться вправо

-S: двигаться вниз

-A: двигаться влево

-R: изменить на широкоугольный

-T: вернуться к старой точке зрения

-z: увеличить объект

-Z: увеличить (далеко) объект

**Конкретный код:**

static void key(unsigned char key, int x, int y)

{

float frac = 0.3;

float rotFrac = 1;

switch (key)

{

case 27:

case 'q':

exit(0);

break;

// два варианта поддержки, чтобы изменить угол обзора, чтобы изменить угол обзора

case 'r':

rot = true;

break;

case 't':

rot = false;

break;

// два случая изменения направления зрения вблизи

case 'z':

zoom += 0.05;

break;

case 'Z':

zoom -= 0.05;

// миграционный кейс для самолета

case 'w':

tY -= frac;

rotZ += rotFrac;

break;

case 's':

tY += frac;

rotZ -= rotFrac;

break;

case 'a':

tX += frac;

rotX -= rotFrac \* 3;

rotY += rotFrac / 2;

break;

case 'd':

tX -= frac;

rotX += rotFrac \* 3;

rotY -= rotFrac / 2;

break;

// два случая начинаются, игра заканчивается

case 'g':

START = true;

break;

case 'm':

START = false;

break;

}

glutPostRedisplay();

}

**\*/Влияние формата кадра дисплея (функция resize: очень важна, если в программе отсутствует программа отображения белой рамки):** Функция изменения размера играет очень важную роль в указании начальных размеров. Если мы этого не декларируем, мы получим только белый экран. Чтобы вызвать эту функцию в основной функции, мы должны вызвать glutReshapeFunc (изменение размера).

Смысл команд следующий: в режиме проекции (PROJECTION) мы отображаем фрейм (glViewPort) и указываем, как его отображать (glFrustum). Затем в режиме просмотра (MODELVIEW) мы будем использовать glLoadIdentity, чтобы применить вышеуказанные функции.

static void resize(int width, int height)

{

const float ar = (float)width / (float)height;

glViewport(0, 0, width, height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glFrustum(-ar, ar, -1.0, 1.0, 2.0, 1000.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

**\*/Постоянный эффект изменения цвета самолета :** Как упоминалось выше, переменная угла (установленное значение изменяет угол плоскости, когда мы нажимаем кнопки перемещения), поэтому мы изменим значение цвета, соответствующее измененному значению угла. Чтобы вызвать эту функцию для всей программы, мы будем использовать функцию glutTimerFunc (0, timer, 0) в основной функции**.** Функция glutPostReDisplay () сделает эту функцию повторяемой

void timer(int value)

{

angle += 3.0f;

color1 += 0.001f;

color2 += 0.003f;

color3 += color2;

if (color1 > 1.0)

color1 = 0;

if (color2 > 1.0)

color2 = 0;

if (color3 > 1.0)

color3 = 0;

glutPostRedisplay();

glutTimerFunc(16, timer, 0);

}

\*/**Показание все заявленные выше объекты:** Если в состоянии СТАРТ (запуск игры), отобразить текст, время, счет и вызвать функцию рисования объектов (функция рисования). Если нет, просто нарисуйте вводный текст и отобразите плоскость (функция плоскости).

static void display(void)

{

const double t = glutGet(GLUT\_ELAPSED\_TIME) / 1000.0;

double a = t \* 90.0;

double aa = a;

// Если снять этот блок, можно увидеть окрестности

if (!rot) {

a = 0;

}

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

gluLookAt(0.0, 4.5, 10.0,

0, 4, 0,

0, 1.0f, 0.0f);

if (START) {

glPushMatrix();

glTranslated(0, 0, 0);

glScaled(zoom, zoom, zoom);

glRotated(a, 0, 1, 0);

glBindTexture(GL\_TEXTURE\_2D, texture);

draw();

glPopMatrix();

drawStrokeText("UP: W, DOWN: S, LEFT: A, RIGHT: D, MAIN MENU: M", -8, 0.9, 0);

drawBitmapText("Quyet Tran Vu's OpenGL Planing Game Code", 0, 0, 0);

drawStrokeText("TIME : ", 3, 0, 0);

int mod, number = 0;

while (TIME) {

mod = TIME % 10;

number = number \* 10 + mod;

TIME /= 10;

}

// tmp - это значение, которое нужно установить для правильного времени приращения (оставьте пробел между числами)

float tmp = 0;

while (number) {

mod = number % 10;

// вызываем функцию для рисования числовых символов (= прошедшее время)

drawStrokeChar(mod + 48, 4 + tmp, 0, 0);

number /= 10;

tmp += 0.2;

}

}

else {

glPushMatrix();

glTranslated(0, 3, 0);

glRotated(aa, 0, 1, 0);

glScaled(1.5, 1.5, 1.5);

plane();

glPopMatrix();

drawStrokeText("Press G to Start", -1, -1, 0);

drawStrokeText2("Plane Game", -2, 0, 0);

}

glutSwapBuffers();

}

\*/**Также использование множества различных эффектов (включая свет, цвет, глубину и т. Д.), Объявленных через группу команд glEnable и glDisable.**

// Использование света

const GLfloat light\_ambient[] = { 0.0f, 0.0f, 0.0f, 1.0f };

const GLfloat light\_diffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };

const GLfloat light\_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };

const GLfloat light\_position[] = { 2.0f, 5.0f, 5.0f, 0.0f };

const GLfloat mat\_ambient[] = { 0.7f, 0.7f, 0.7f, 1.0f };

const GLfloat mat\_diffuse[] = { 0.8f, 0.8f, 0.8f, 1.0f };

const GLfloat mat\_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };

const GLfloat high\_shininess[] = { 100.0f };

**В основной функции(main) мы вызываем компоненты**

glClearColor(1, 1, 1, 1);

glEnable(GL\_CULL\_FACE); // разрешаем объекту смотреть вперед и назад (обратите внимание, если установлен режим GL\_FRONT\_AND\_BACK: полигоны не отображаются, только точки и линии)

glCullFace(GL\_BACK);

glEnable(GL\_TEXTURE\_2D);

glEnable(GL\_DEPTH\_TEST);

// Используя GL\_LESS, мы не увидим объект "ПУСТОЙ"

glDepthFunc(GL\_LESS);

glEnable(GL\_LIGHT0);

glEnable(GL\_NORMALIZE// делает векторы нормализованными после преобразования и до освещения.

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_LIGHTING);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, high\_shininess);

**Все содержимое кода и программы полностью отображается по ссылке на Github:**

Выполнил:

|  |  |  |
| --- | --- | --- |
| студент группы ИДБ-19-02 | (подпись) | Чан Ву Кует |
|  |  |  |

Принял:

|  |  |  |
| --- | --- | --- |
| ст. преподаватель  Гаврилов А. Г. | (оценка) | (подпись) |
|  |  |  |
|  |  | «15»Декабря 2021г. |