РЕКОМЕНДАЦИИ ДЛЯ ВЫПОЛНЕНИЯ ЗАДАНИЯ 9

Построение проволочного трехмерного образа

Будем строить проволочное трехмерное изображение, заданное набором треугольников во входном файле. Для выполнения задания потребуется процедура отсечения многоугольников и структуры данных реализованные при выполнении 7-го и 8-го заданий. Целесообразней для выполнения настоящего задания взять копию проекта, полученного в результате выполнения 7-го задания.

Работа приложения

Разрабатываемое решение должно выводить проволочный образ трехмерной сцены в прямоугольнике на экране.

Управление изображением осуществляется горячими клавишами:

P — Переключение типа проекции в образе: ортогональная/перспективная проекция;

A/D — поворот точки наблюдения относительно точки центра сцены влево/вправо на угол α;

W/S — поворот точки наблюдения относительно точки центра сцены вверх/вниз на угол α;

Q/E — поворот окна наблюдения против часовой стрелки/по часовой стрелке на угол α;

Z/X — увеличение/уменьшение расстояния от наблюдателя до окна наблюдения;

C/V — увеличение/уменьшение угла вертикального обзора через окно наблюдения на 1 градус;

O/L — увеличение/уменьшение угла α в 1.1 раза;

ESC — переход к первоначальному изображению (сброс параметров).

Взгляд наблюдателя в приложении всегда направлен в центр сцены. Центр сцены находится всегда в одной точке (задается во входном файле).

Изменения в `Transform.h` и `Transform.cpp`

При выполнении задания придется иметь дело как с двумерными так и с трехмерными преобразованиями.

Прежде всего добавим в проект структуры данных для трехмерных объектов. Для этого в файле `Transform.h` опишите структуры `point3D` (трехмерная точка), `polygon3D` (трехмерный многоугольник/треугольник), `vec3D` (трехмерная точка в однородных координатах), `mat3D` (матрица трехмерного преобразования) как копии двумерных структур с добавлением дополнительной размерности.

Дополнительно к имеющимся процедурам в файле `Transform.h` должны быть объявлены, а в файле `Transform.cpp` описаны процедуры для трехмерных преобразований:

```
1  void times (mat3D a, mat3D b, mat3D &c);
2  void timesMatVec (mat3D a, vec3D b, vec3D &c);
3  void set (mat3D a, mat3D &b);
4
5  void point2vec (point3D a, vec3D &b);
6  void vec2point (vec3D a, point3D &b);
7
8  void makeHomogenVec (float x, float y, float z, vec3D &c);
9
10 void unit (mat3D &a);
11 void move (float Tx, float Ty, float Tz, mat3D &c);
12 void rotate (point3D n, float phi, mat3D &c);
13 void scale (float Sx, float Sy, float Sz, mat3D &c);
```

Здесь все процедуры подобны их двумерным аналогам (с учетом добавленной размерности), за исключением процедуры `rotate`. Процедура `rotate` должна инициализировать матрицу `c` — матрицу вращения относительно радиус-вектора точки `n` на угол `phi`. Вычисление элементов матрицы должно производиться с использованием формулы Родригеса (см. пункты 2.7.3 и 2.7.5 теоретического материала).

Для реализации перехода к системе координат наблюдателя необходимо описать процедуру

```
void LookAt (point3D eye, point3D center, point3D up, mat3D &c);
```

формирующую матрицу соответствующего преобразования `c`, при условии, что `eye` — точка наблюдения, `center` — точка, на которую должен быть направлен вектор наблюдения, `up` — вектор, указывающий направление вверх (см. пункт 3.2 теоретического материала).

Для формирования матриц проекций необходимо реализовать процедуры

```
1  void Ortho (float Vx, float Vy, float near, float far, mat3D &c);
2  void Frustrum (float Vx, float Vy, float near, float far, mat3D &c);
3  void Perspective (float fovy, float aspect, float near, float far, mat3D &c);
```

где `near` — расстояние от наблюдателя до окна наблюдения, `far` — расстояние от наблюдателя до самой отдаленной по оси Oz точки трехмерной сцены, `Vx` и `Vy` — размеры окна наблюдения, `fovy` — максимальный вертикальный угол обзора через окно наблюдения и `aspect` — соотношение сторон окна наблюдения (отношение горизонтальной стороны к вертикальной) (см. пункты 3.4, 3.6 теоретического материала).

Наконец, для перехода из трехмерной системы координат к двумерной, опишите процедуру

```
1 void set (point3D a, point &b);
```

в которой двумерная точка получается из трехмерной за счет отбрасывания третьей координаты.

Для реализации вышеперечисленных процедур могут понадобиться вспомогательные процедуры, например, процедура умножения элементов матрицы на число, процедура векторного произведения, процедура нормализации вектора и т. п.

Параметры приложения

Образ трехмерной сцены должен выводиться в прямоугольнике на экране.

Для хранения трехмерной сцены понадобится организация списка треугольников.

Кроме набора треугольников необходимо хранить параметры трехмерной сцены:

eye — первоначальные координаты точки наблюдения в мировой системе координат;

center — первоначальные координаты центра сцены (точки в направлении которой смотрит наблюдатель);

near — расстояние от наблюдателя до окна наблюдения;

far — расстояние от наблюдателя до самой отдаленной по оси Oz точки трехмерной сцены;

$fovy$ — вертикальный угол обзора через окно наблюдения;

up — вектор, указывающий направление вверх от наблюдателя;

prOrtho — параметр, указывающий какой тип проекции должен строиться: если значение `true`, то должна строиться ортогональная проекция, в противном случае — перспективная.

В ходе работы приложения некоторые из этих параметров будут изменяться, поэтому для них необходимо хранить первоначальное и текущее значение.

Формат входного файла

Предполагаем, что во входном файле могут встречаться (кроме пустых строк и строк с комментариями) строки команд следующего вида:

`triangle x1 y1 z1 x2 y2 z2 x3 y3 z3` — задается треугольник, в котором числа `xi`, `yi`, `zi` — координаты `i`-ой вершины треугольника;

`color R G B` — команда установки цвета линий последующих треугольников, в которой `R`, `G`, `B` — целочисленные значения от 0 до 255 — количество красной, зеленой и синей составляющей заданного цвета, соответственно. По умолчанию, цвет треугольников черный;

`lookat eyex eyey eyez centerx centery centerz upx upy upz` — команда установки параметров системы координат наблюдателя, где `eyex`, `eyey`, `eyez` — координаты точки наблюдения, `centerx`, `centery`, `centerz` — координаты точки центра сцены, `upx`, `upy`, `upz` — элементы вектора, указывающего направление вверх;

`screen fovy near far` — установка параметров окна наблюдения и трехмерной сцены, где `near` — расстояние от наблюдателя до окна наблюдения, `far` — расстояние от наблюдателя до самой отдаленной по оси Oz точки трехмерной сцены, `fovy` — максимальный вертикальный угол обзора через окно наблюдения (в градусах).

Организация работы приложения

В ходе работы приложения должны меняться положение точки наблюдения (координаты точки eye), вектор направления вверх (вектор up) и параметры окна наблюдения $fovy$ и $near$. Поэтому следует иметь рабочие параметры eye', up', $fovy'$, $near'$. Кроме этого нам понадобится параметр $aspect'$ равный соотношению сторон прямоугольника на экране (W_x/W_y).

Значения eye' и $center'$ и up' будем вычислять (и хранить) в рабочей системе координат. Для перехода в эту систему при загрузке файла сформируем матрицу T общего преобразования точек как результат $T = LookAt(center, eye, up)$. В результате такого преобразования направление вверх совпадет с направлением оси Oy, а точка eye будет располагаться на оси Oz. Инициализируем точку eye' как eye после преобразования T (результат произведения матрицы T с однородными координатами точки eye), вектор up' вектором $(0, 1, 0)$, точку $center'$ точкой $(0, 0, 0)$.

Теперь вращения точки eye' вокруг точки $center$ сводится к вращениям вокруг осей координат. При каждом повороте точки eye вокруг точки $center$ будем изменять (вращать) рабочую систему координат, чтобы eye' и up' не меняли своих значений.

Процедура отрисовки

Процедура отрисовки образа будет заключаться в следующем:

1. Вычислить матрицу преобразования $V = LookAt(eye', center', up')$, для перехода из рабочей системы координат в систему координат наблюдателя;
2. Вычислить матрицу преобразования проекции U, равную $Perspective(fovy', aspect', near', far)$ или $Ortho(V_x, V_y, near', far)$, в зависимости от типа выбранной проекции (значения V_x и V_y вычисляются из значений $near$ и $fovy$);
3. Вычислить результат произведения матриц $R = U \cdot V \cdot T$;
4. Вычислить матрицу преобразования кадрирования $F = frame(2, 2, -1, -1, W_x, W_y, W_{cx}, W_{cy})$;
5. Для каждого многоугольника P сформировать многоугольник P':
   (a) Для каждой точки A многоугольника P
      i. вычислить координаты A' после преобразования R;
      ii. вычислить координаты A'' — координаты точки A' после перехода в двумерную систему координат;
      iii. Вычислить координаты A''' — координаты точки A'' после преобразования F;
      iv. Поместить A''' в список многоугольника P';
   (b) Провести отсечение многоугольника P' относительно прямоугольника на экране с помощью алгоритма, реализованного при выполнении задания 7;
   (c) Если получен результат отсечения, начертить его цветом многоугольника P.

Обработка горячих клавиш

Обработка горячих клавиш должна проводиться в следующем порядке:

- при повороте влево/вправо формируется матрица R_1 вращения относительно вектора up' (вектор $(0, 1, 0)$) на угол α, после чего вычисляется новое значение $T = R_1 \cdot T$;
- при повороте вверх/вниз формируется матрица вращения R_2 относительно оси Ox (вектор $(1, 0, 0)$) на угол α, после чего после чего вычисляется новое значение $T = R_2 \cdot T$;

- при повороте окна наблюдения формируется матрица вращения R_3 относительно оси Oz (вектор $(0, 0, 1)$) на угол α, после чего вычисляется новое значение $T = R_3 \cdot T$;
- параметр $near'$ может принимать только положительные значения;
- параметр α по умолчанию должен равняться 5 градусам.