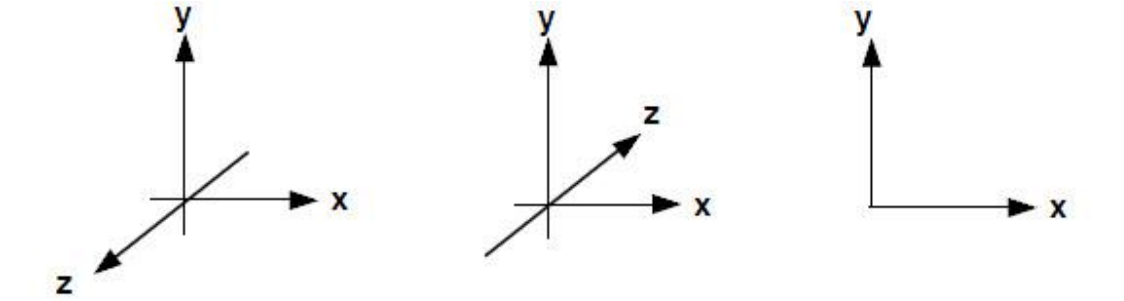
Лабораторная работа №4 «Модельные трансформации».

В OpenGL используются как основные три система координат: левосторонняя, правосторонняя и оконная. Первые две системы являются трехмерными и отличаются друг от друга направлением оси *z* : в правосторонней она направлена на наблюдателя, в левосторонней – в глубину экрана. Ось *x* направлена вправо относительно наблюдателя, ось *y* - вверх. Левосторонняя система используется для задания значений параметрами `gluPerspective()`, `glOrtho()`. Правосторонняя система координат используется во всех остальных случаях. Отображение трехмерной информации происходит в двумерную оконную систему координат.



а) правосторонняя система б) левосторонняя система в) оконная система

Существует три команды OpenGL для модельных преобразований: `glTranslate*()`, `glRotate*()` и `glScale*()`. Эти команды трансформируют объект (или координатную систему – в зависимости от того, как вы предпочитаете думать об этом), перенося, поворачивая, увеличивая, уменьшая или отражая его. Все эти команды эквивалентны созданию соответствующей матрицы переноса, поворота или масштабирования с последующим вызовом `glMultMatrix*()` с этой матрицей в качестве аргумента. Однако использование этих трех команд может быть быстрее, чем `glMultMatrix*()` – OpenGL автоматически вычисляет для вас нужные матрицы.

`void` **`glTranslate`**`{fd} (TYPE x, TYPE y, TYPE z);`

Умножает текущую матрицу на матрицу, передвигающую (переносящую) объект на расстояния *x*, *y*, *z*, переданные в качестве аргументов команды, по соответствующим осям (или перемещает локальную координатную систему на те же расстояния).

`void` **`glRotate`**`{fd} (TYPE angle, TYPE x, TYPE y, TYPE z);`

Умножает текущую матрицу на матрицу, которая поворачивает объект (или локальную координатную систему) в направлении против часовой стрелки вокруг луча из начала координат, проходящего через точку (*x*, *y*, *z*). Параметр *angle*задает угол поворота в градусах.

`void` **`glScale`**`{fd} (TYPE x, TYPE y, TYPE z);`

Умножает текущую матрицу на матрицу, которая растягивает, сжимает или отражает объект вдоль координатный осей. Каждая *x* -, *y*- и *z*- координата каждой

точки объекта будет умножена на соответствующий аргумент *x*, *y* или *z* команды `glScale*()`. При рассмотрении преобразования с точки зрения локальной координатной системы, оси этой системы растягиваются, сжимаются или отражаются с учетом факторов *x*, *y* и *z*, и ассоциированный с этот системой объект меняется вместе с ней. `glScale*()` – это единственная из трех команд модельных преобразований, изменяющая размер объекта: масштабирование с величинами более *1.0* растягивает объект, использование величин меньше *1.0* сжимает его.

Манипулирование матричными стеками

Видовая и проекционная матрицы является верхним элементом матричного стека. Матричный стек полезен для построения иерархических моделей, в которых сложные объекты конструируются при помощи простых. Поскольку преобразования сохраняются в матрицах, матричный стек предоставляет идеальный механизм для подобного рода запоминаний, переносов и отбрасываний. Все матричные операции (`glLoadMatrix()`, `glMultMatrix()`, `glLoadIdentity()` и команды, создающие специфические матрицы) работают с текущей матрицей, то есть с верхней матрицей стека. С помощью команд управления стеком можно управлять тем, какая матрица находится на вершине стека: `glPushMatrix()` копирует текущую матрицу и добавляет копию на вершину матричного стека, `glPopMatrix()` уничтожает верхнюю матрицу в стеке. Говоря проще, `glPushMatrix()` означает «запомнить, где мы находимся», а `glPopMatrix()` – «вернуться туда, где мы были».

```
void glPushMatrix (void);
```

Опускает все имеющиеся в текущем стеке матрицы на один уровень. То, какой стек является текущим, задается с помощью вызова `glMatrixMode()`. Верхняя матрица при этом копируется, таким образом, ее содержимое продублировано в верхней и второй сверху матрице стека. Если добавлено слишком много матриц, будет сгенерирована ошибка.

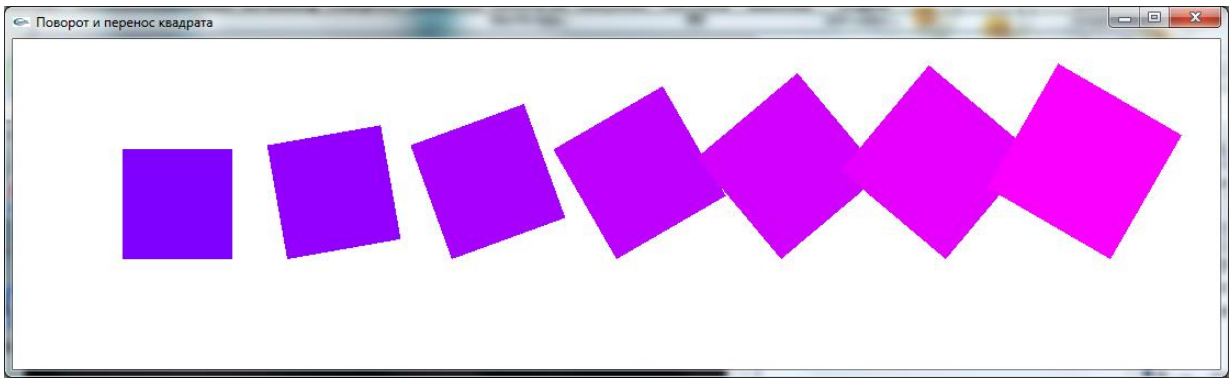
```
void glPopMatrix (void);
```

Выкидывает верхнюю матрицу из стека, тем самым, уничтожая ее содержимое. Верхней (и, как следствие, текущей) становится матрица, которая занимала второе сверху место в стеке. Текущий стек задается командой `glMatrixMode()`. Если стек содержит только одну матрицу, вызов `glPopMatrix()` сгенерирует ошибку.

Часть кода программы, реализующей три вида модельных преобразований.

```
void display(void)
{ int i;
  glClearColor(1.0,1.0,1.0,1.0);
  glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  for (i=0;i<=6;i++){
       glPushMatrix();
    glTranslatef(-4.5+i*1.5,0,0);
       glRotatef(i*10,0,0,1);
       glColor3f(0.5+i*0.08,0,1);
       glScalef(1+i*0.05,1+i*0.05,1+i*0.05);
       glBegin(GL_QUADS);
```

```
        glVertex2f(0,0);
        glVertex2f(0,1);
        glVertex2f(1,1);
        glVertex2f(1,0);
        glEnd();
        glPopMatrix();
 }
 glFinish();
}
```



Задание:

1. Написать программу, реализующую произвольное прямолинейное перемещение объекта в окне. При столкновении с границами окна, объект отскакивает и продолжает движение.
2. Модифицировать программу таки образом, чтобы объект во время движению вращался вокруг своей оси.

Таблица 1. Указания к лабораторной работе № 4

№ вар	Задание	№ вар	Задание
1	Равнобедренная трапеция	6	Правильный пятиугольник
2	Прямоугольный неравнобедренный треугольник	7	Ромб
3	Правильный шестиугольник	8	Параллелограмм
4	Ромб с меньшим углом 60°	9	Правильный восьмиугольник
5	Равнобедренный треугольник	10	Прямоугольник

3. Написать программу, которая в виде таблицы выводит на экран 6 объектов разного вида с разными скоростями вращения.