ПРЕОБРАЗОВАНИЕ СИСТЕМ КООРДИНАТ

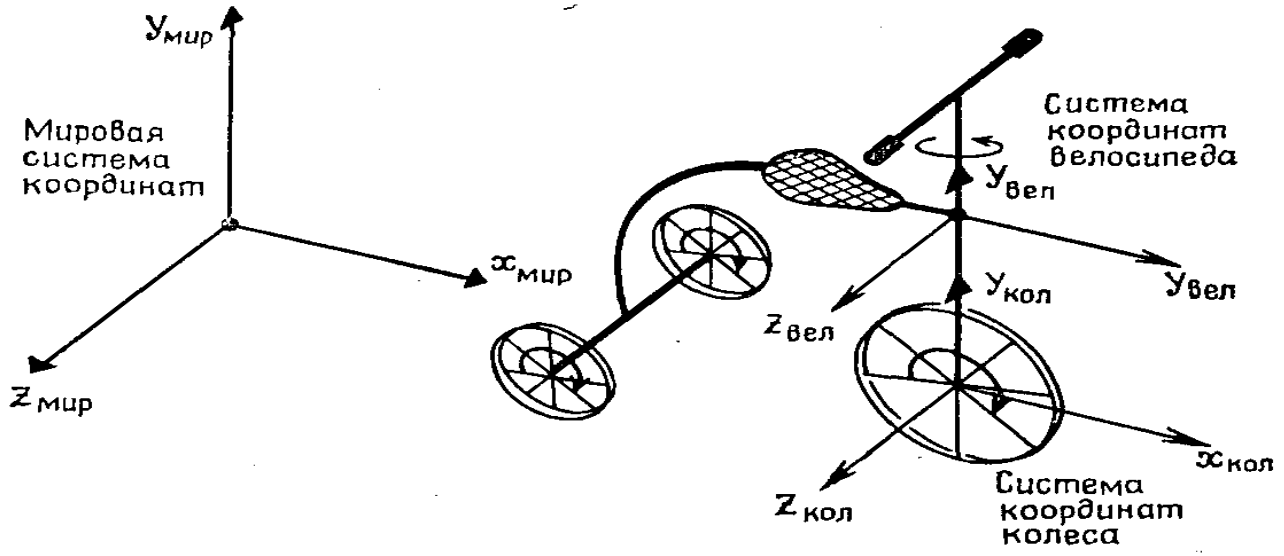
Можно рассматривать, что система координат остается неизменной, а сам объект преобразуется относительно начала координат до получения желаемого размера.

Другим эквивалентным способом описания преобразования является смена систем координат.

С математической точки зрения оба подхода идентичны.

Движение объектов можно рассматривать как движение в обратном направлении соответствующей системы координат.

ПРЕОБРАЗОВАНИЕ СИСТЕМ КООРДИНАТ



Введем три вида систем координат.

Первая из них – мировая система координат – задается осями *XmYmZm*. Мы размещаем ее в некоторой точке, и она остается неподвижной всегда.

Вторая – система координат наблюдателя. Эту систему назовем *XnYnZn*. Она определяет положение наблюдателя в пространстве и задает направление взгляда.

И третья – система координат объекта. Эти системы также могут перемещаться и изменять свое положение в пространстве относительно мировой системы координат.

ПРЕОБРАЗОВАНИЕ СИСТЕМ КООРДИНАТ

Координаты точек объектов задаются в системах координат объектов, каждая из которых, в свою очередь, привязана к мировой системе координат.

Система координат наблюдателя также перемещается относительно мировой системы координат.

Чтобы увидеть трехмерный объект на экране компьютера надо проделать следующие шаги:

1. Преобразовать координаты объекта, заданные в собственной системе координат, в мировые координаты.
2. Преобразовать координаты объекта, заданные уже в мировой системе координат, в систему координат наблюдателя.
3. Спроецировать полученные координаты на проекционную плоскость в системе координат наблюдателя.

ПРЕОБРАЗОВАНИЕ СИСТЕМ КООРДИНАТ

Чаще используют две системы координат.

Первая — мировые координаты, которые описывают истинное положение объектов в пространстве с заданной точностью.

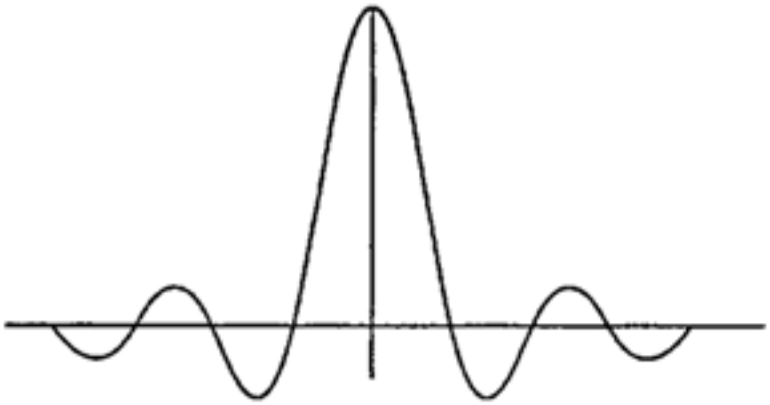
Вторая — система координат устройства отображения, в котором осуществляется вывод изображения объектов в заданной проекции, иногда их называют экранными координатами.

Основная задача — задать преобразования координат из мировых в экранные.

МИРОВЫЕ ОКНА И ПОРТ ПРОСМОТРА В OPENGL

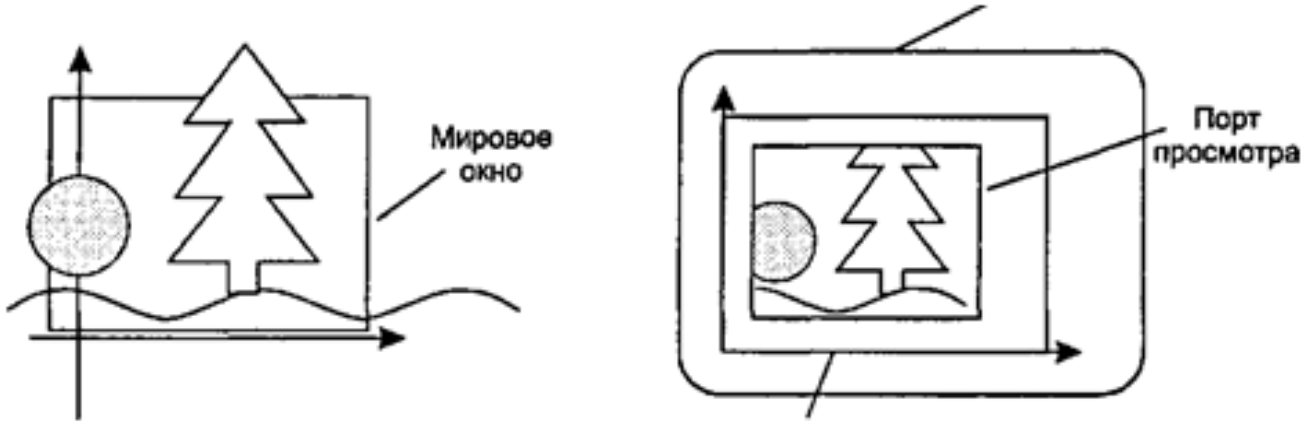
Пример

$$\operatorname{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$



```
void myDisplay(void)
{
    glBegin(GL_LINE_STRIP);
    for(GLfloat x = -4.0; x < 4.0; x += 0.1)
    {
    GLfloat y = sin(3.14159 * x) / (3.14159 * x);
        glVertex2f(x, y);
    }
glEnd();
glFlush();
}
```

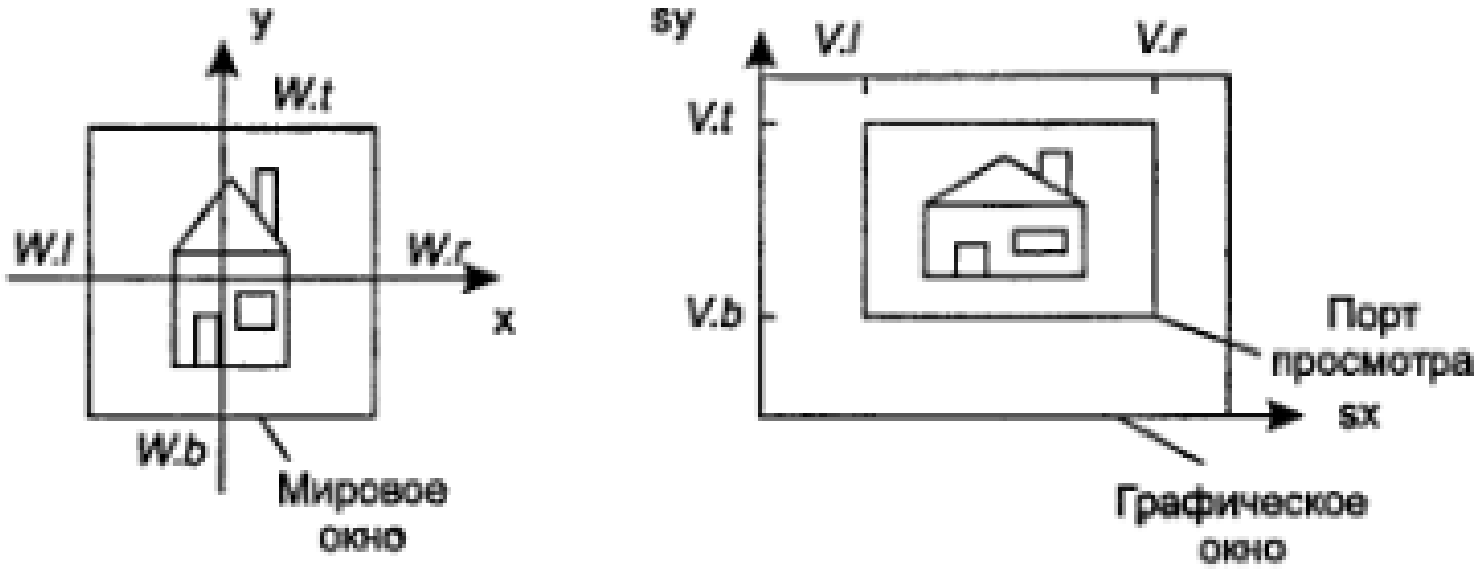
МИРОВЫЕ ОКНА И ПОРТ ПРОСМОТРА В OPENGL



а б

Мировое окно (а) и порт просмотра (б)

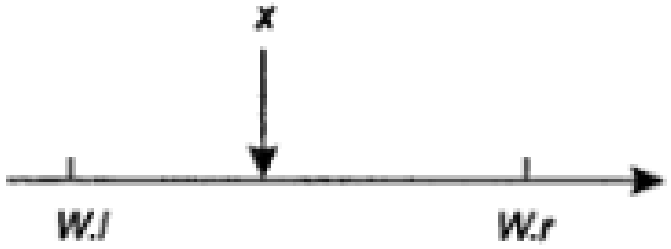
ПРЕОБРАЗОВАНИЕ ИЗ МИРОВОГО ОКНА В ПОРТ ПРОСМОТРА



ПРЕОБРАЗОВАНИЕ ИЗ МИРОВОГО ОКНА В ПОРТ ПРОСМОТРА

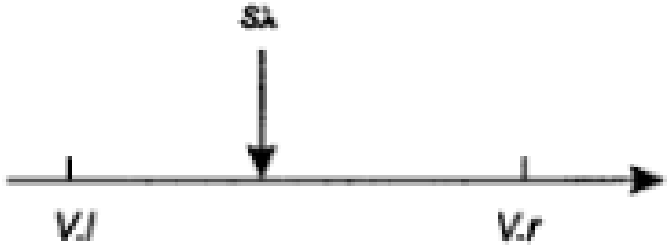
Пропорциональность в преобразовании

$$\frac{sx - V.l}{V.r - V.l} = \frac{x - W.l}{W.r - W.l};$$



$$(sx - V.l)(W.r - W.l) = (V.r - V.l)(x - W.l);$$

$$(sx - V.l) = \frac{V.r - V.l}{W.r - W.l} * (x - W.l);$$



$$sx = \frac{V.r - V.l}{W.r - W.l} * (x - W.l) + V.l;$$

$$sx = \frac{V.r - V.l}{W.r - W.l}x + \left(V.l - \frac{V.r - V.l}{W.r - W.l}W.l\right).$$

ПРЕОБРАЗОВАНИЕ ИЗ МИРОВОГО ОКНА В ПОРТ ПРОСМОТРА

Это преобразование можно использовать для любой точки (x, y) внутри или вне окна. Точки внутри окна преобразуются во внутренние точки порта просмотра, а точки вне окна — в точки вне порта просмотра.

Если x находится на левом краю окна ($x = W.l$), то sx находится на левом краю порта просмотра ($sx = V.l$).

Если x — на правом краю окна, то sx — на правом краю порта просмотра.

Если x составляет f-ю часть от ширины окна, то sx является f-й частью от ширины порта просмотра.

Если x находится вне окна слева от него ($x < W.l$), то sx также находится вне порта просмотра слева ($x < V.l$), и аналогично в случае, когда x вне окна справа от него.

МИРОВЫЕ ОКНА И ПОРТ ПРОСМОТРА В OPENGL

Мировое окно устанавливается с помощью функции `gluOrtho2D()`

`void gluOrtho2D(GLdouble left, GLdouble right, GLdouble buttom, GLdouble top);` устанавливает для окна левый нижний угол (left, bottom) и правый верхний угол (right, top),

Порт просмотра — функцией `glViewport().`

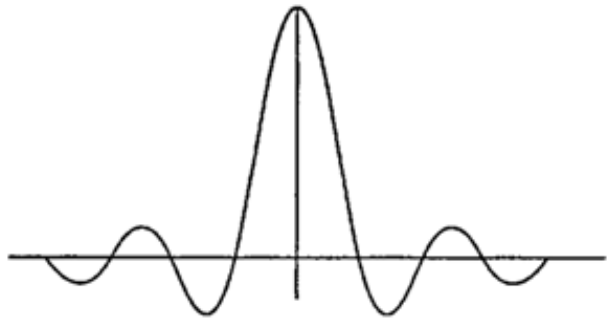
`void glViewport(GLint x, GLint y, GLint width, GLint height);` станавливает для порта просмотра нижний левый угол (x. y) и верхний правый угол (x + width, y + height).

По умолчанию порт просмотра является полным экранным окном: если ширина и высота экранного окна равны соответственно W и H, то порт просмотра по умолчанию имеет левый нижний угол (0, 0) и верхний правый угол (W, H).

МИРОВЫЕ ОКНА И ПОРТ ПРОСМОТРА В OPENGL

Пример

$$\operatorname{sinc}(x) = \frac{\operatorname{sinc}(\pi x)}{\pi x}$$



```
void myDisplay(void)
{
    glBegin(GL_LINE_STRIP);
    for(GLfloat x = -4.0; x < 4.0; x += 0.1)
    {
    GLfloat y = sin(3.14159 * x) / (3.14159 * x);
        glVertex2f(x, y);
    }
glEnd();
glFlush();
}
```

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-4.0, 4.0, 2.0, 1.0);// устанавливает окно
glViewport(0,  0,  640,  480);  //  устанавливает  порт
просмотра
```

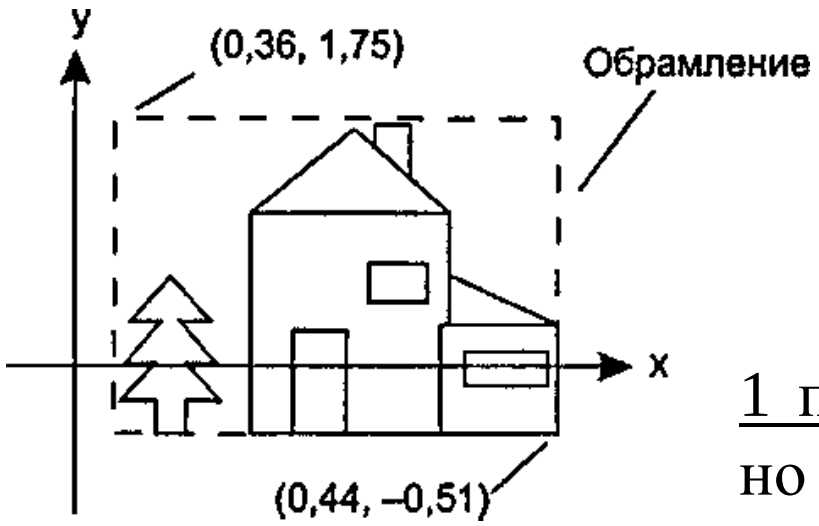
МИРОВЫЕ ОКНА И ПОРТ ПРОСМОТРА В OPENGL

```
    //  ... setWindow  ......
Void  setWindow(GLdouble  left,  GLdouble  right,  GLdouble
bottom, GLdouble top)
    {
            glMatrixMode(GL_PROJECTION);
            glLoadIdentity();
            gluOrtho2D(left, right, bottom, top);
    }
```

МИРОВЫЕ ОКНА И ПОРТ ПРОСМОТРА В OPENGL

```
    //……setViewport……..
    Void  setViewport(GLint  Left,  GLint  right,  GLint
bottom, GLint top)
    {
    glViewport(left, bottom, right - left, top - bottom);
    }
```

АВТОМАТИЧЕСКАЯ УСТАНОВКА ПОРТА И МИРОВОГО ОКНА

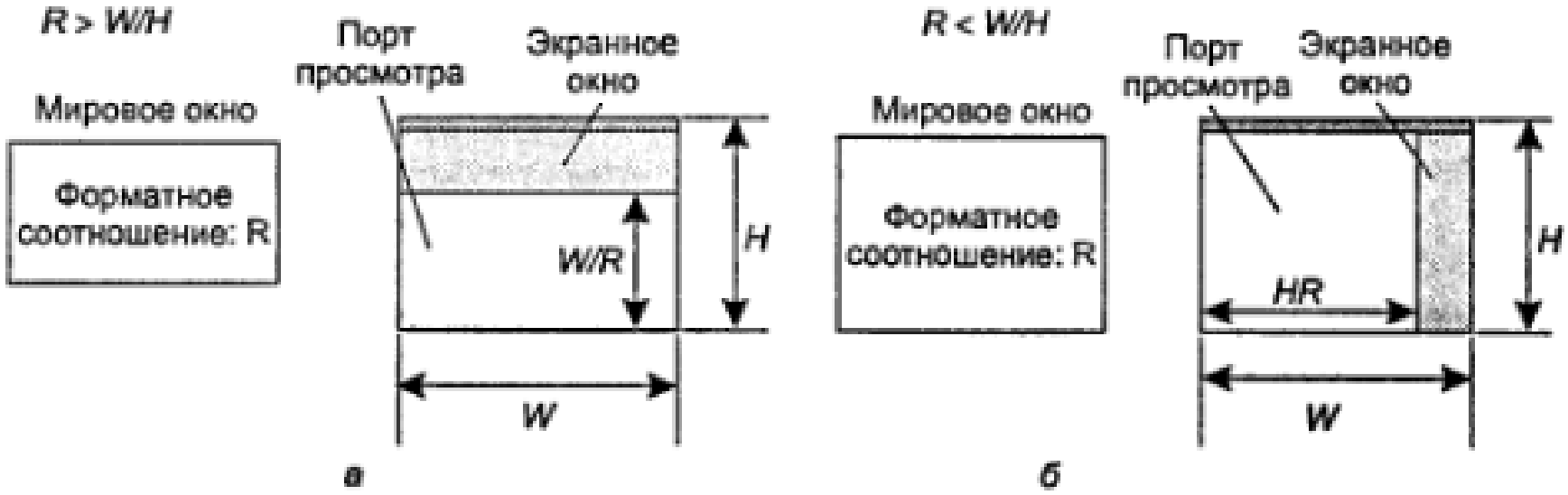


Экстент, или ограничивающий прямоугольник объекта — это выровненный прямоугольник, в точности покрывающий данный объект.

1 прогон. Выполнить подпрограмму рисования, но фактически не рисовать, а только вычислять экстент. Затем установить окно.

2 прогон. Снова выполнить подпрограмму, уже с рисованием.

АВТОМАТИЧЕСКАЯ УСТАНОВКА ПОРТА И МИРОВОГО ОКНА



setViewport(0, W, 0, W/R);

setViewport(0, H*R, 0, H);

АВТОМАТИЧЕСКАЯ УСТАНОВКА ПОРТА И МИРОВОГО ОКНА

Событие `resize` (изменение размеров).

Функция `glutReshapeFunc()` из инструментария OpenGL специфицирует функцию myReshape, вызываемую при возникновении данного события: `glutReshapeFunc(myReshape)`, т.е. задает функцию, вызываемую по событию resize.

Кроме того, эта зарегистрированная функция вызывается, когда окно открывается впервые, и должна иметь следующий прототип:

```
void myReshape(GLsizei W, GLsizei H);
```

При выполнении данной функции система автоматически передает в нее новую ширину и высоту того экранного окна, которое эта функция может затем использовать в своих вычислениях. (`GLsizei` является 32-битным целым.)

АВТОМАТИЧЕСКАЯ УСТАНОВКА ПОРТА И МИРОВОГО ОКНА

```
void myReshape(GLsizei W, GLsizei H)
{                    …
        if (R > W/H)
// используем форматное соотошение (глобального) окна – R
                setViewport(0, W, 0, W/R);
        else
                setViewport(0, H*R, 0, H);
}
```

ВЗАИМОДЕЙСТВИЕ С ПОМОЩЬЮ МЫШИ

`glutMouseFunc(myMouse)` — связывает myMouse() с событием, возникающим при нажатии или отпускании кнопки мыши;

```
void myMouse(int button, int state, int x, int y);
```

Параметр `button` должен принять одно из следующих значений: GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON (левая кнопка, средняя кнопка, правая кнопка). Параметр `state` должен быть равен GLUT_UP или GLUT_DOWN (вверх или вниз).

Значения x и y сообщают о положении мыши в момент события.

Но: величина x равна числу пикселей от левого края окна, а величина y равна числу пикселей вниз от верха окна.

ВЗАИМОДЕЙСТВИЕ С ПОМОЩЬЮ МЫШИ

```
void drawDot(GLint x, GLint y) // рисуем точку
{
 glBegin(GL_POINTS);
 glVertex2i(x,y);
 glEnd();
 glFlush();
}
```

ВЗАИМОДЕЙСТВИЕ С ПОМОЩЬЮ МЫШИ

```
    void Mouse (int button, int state, int x, int y) //
функция обратного вызова для событий нажатиия кнопок мыши
    {   if          (button==GLUT_LEFT_BUTTON          &&
state==GLUT_DOWN) // если нажата левая кнопка мыши
        {                                  drawDot((GLint)x,
(GLint)glutGet(GLUT_WINDOW_HEIGHT)-y); // рисуем точки
         } else
        {      if      (button==GLUT_RIGHT_BUTTON      &&
state==GLUT_DOWN){exit(-1);}}  //    если  нажата  правая
кнопка мыши то выход
    }
```

ВЗАИМОДЕЙСТВИЕ С ПОМОЩЬЮ МЫШИ

`glutMotionFunc(myMovedMouse)` — связывает `myMovedMouse()` с событием, возникающим при перемещении мыши, когда одна из ее кнопок нажата;

`void myMovedMouse(int x, int y);` значения x и y представляют собой позицию мыши в тот момент, когда рассматриваемое событие происходит.

ВЗАИМОДЕЙСТВИЕ С ПОМОЩЬЮ МЫШИ

```
void myMovedMouse(int mouseX, int mouseY)
{ GLint x = mouseX;
  GLint y = glutGet(GLUT_WINDOW_HEIGHT) - mouseY;
  GLint brushSize = 20;
  glRecti(x,y, x + brushSize, y + brushSize);
  glFlush();
}
```

ВЗАИМОДЕЙСТВИЕ С ПОМОЩЬЮ КЛАВИАТУРЫ

Функция обратного вызова `myKeyboard()` регистрируется с данным типом события посредством подпрограммы `glutKeyboardFunc(myKeyboard),` которая должна иметь следующий прототип:

```
void myKeyboard(unsingned char key, int x, int y);
```

Величина `key` определяется ASCII-кодом нажатой клавиши.

Величины x и y сообщают позицию мыши в момент возникновения события.

Функция `glutSpecialFunc(SpecialKey)` регистрирует функцию `SpecialKey()` которая должна иметь следующий прототип:

```
void SpecialKey (int key, int x, int y);
```

ВЗАИМОДЕЙСТВИЕ С ПОМОЩЬЮ КЛАВИАТУРЫ

```
    void Keyboard (unsigned char key, int x, int y) //
функция обратного вызова для событий при нажатии кнопок
клавиатуры
    {                                                  //
x,y текущие координаты мыши
    switch (key)
    {
    case 'p':
{ drawDot((GLint)x,(GLint)glutGet(GLUT_WINDOW_HEIGHT)-y);
break; }
    default : break;
    }
    }
```

ВЗАИМОДЕЙСТВИЕ С ПОМОЩЬЮ КЛАВИАТУРЫ

```
void SpecialKey (int key, int x, int y)
{ //функция для обработки специальных символов
    switch (key)
{
case GLUT_KEY_RIGHT:    {… break;}
case GLUT_KEY_LEFT:     {… break;}
case GLUT_KEY_UP:       {… break;}
…
default : break;
}
}
```