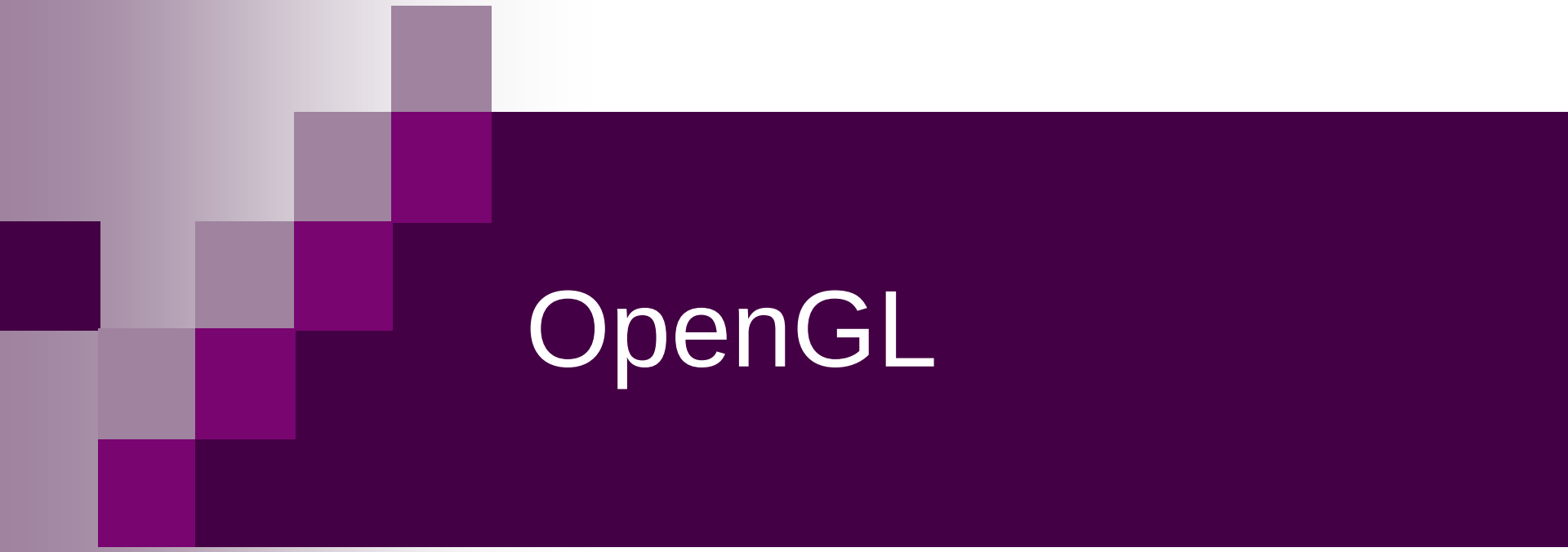
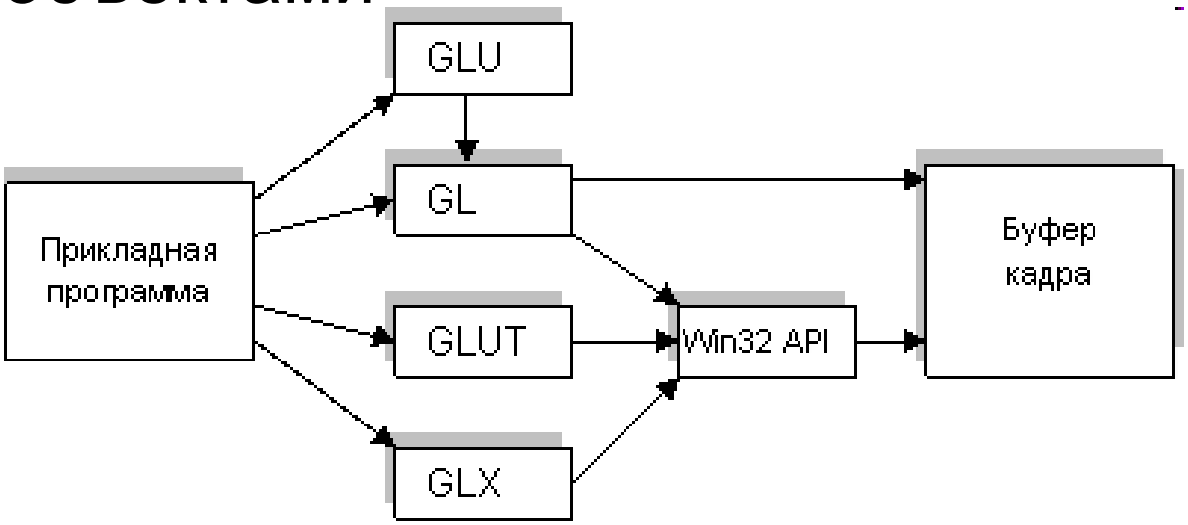
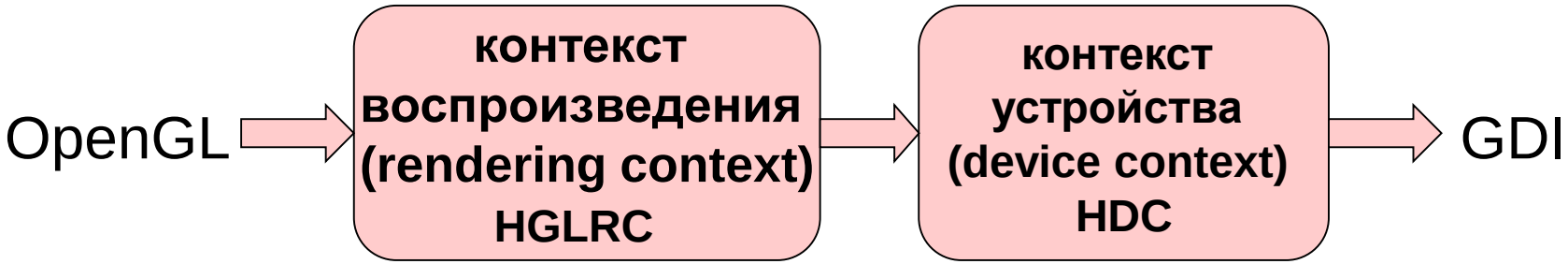

OpenGL

# OpenGL в Windows

- opengl32.dll – набор базовых функций
- glu32.dll – Надстройка позволяющая работать с glu объектами



# GLU, GLUT и GLX

- **Ф-ции GLU:** переключение между экранными и мировыми координатами, создание текстур, расширенные функции трансформации для установки точек обзора и более простого управления камерой, функции для рисования: сферы, цилиндра, конуса, диска
- **OpenGL Utility Toolkit (GLUT)** - библиотека утилит, которая отвечает за системный уровень операций В/В при работе с ОС.
- **Ф-ции GLUT:** создание окна, управление окном, мониторинг за вводом с клавиатуры и событий мыши, рисование примитивов: куб, сфера, чайник.
- **GLX** обеспечивает работу OpenGL в Unix подобных системах

# Функции для начала работы

- **ChoosePixelFormat(HDC, const PIXELFORMATDESCRIPTOR)** – функция выбирающая наиболее подходящий формат исходя из информации, переданной в полях структуры PIXELFORMATDESCRIPTOR
- **SetPixelFormat(HDC hDC, inl pixelFormat, const PIXELFORMATDESCRIPTOR)** - устанавливает формат пикселя в контексте устройства
- **wglCreateContext(HDC hDC) и BOOL wglMakeCurrent(HDC hDC, HGLRC hGLRC)** – создание контекста воспроизведения
- **wglDeleteContext(HGLRC hGLRC)** – удаление контекста вопроизведения

# Создание контекста воспроизведения

**Обработчик onCreate:**

```
 SetDCPixelFormat(Canvas.Handle);
 hrc := wglCreateContext(Canvas.Handle);


procedure SetDCPixelFormat (hdc : HDC);
var
 pfd : TPixelFormatDescriptor;
 nPixelFormat : Integer;
begin
 FillChar (pfd, SizeOf (pfd), 0);
 nPixelFormat := ChoosePixelFormat (hdc, @pfd);
 SetPixelFormat (hdc, nPixelFormat, @pfd);
end;
```

# Формат команд

- Все команды (процедуры и функции) OpenGL начинаются с префикса **gl**, а все константы – с префикса **GL** (GLbyte, GLfloat, и т.д.)
- Некоторые команды OpenGL оканчиваются на букву v

```
glColor3f(1.0, 1.0, 1.0);
```

и

```
GLfloat color[] = {1.0, 1.0, 1.0};
glColor3fv(color);
```

эквивалентны

# Буферы OpenGL

- **буфер изображения** (фреймбуфер);
- ***z*-буфер**;
- **аккумулирующий буфер** (можно сохранять визуализированное изображение, применяя при этом попиксельно специальные операции);
- **буфер трафарета** (при выводе пикселей в буфер кадра иногда возникает необходимость выводить не все пиксели, а только некоторое подмножество, т.е. наложить трафарет (маску) на изображение. Для этого OpenGL предоставляет так называемый буфер маски (stencil buffer)).

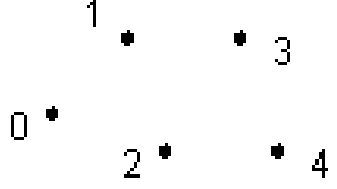
# Использование буферов для рисования

- **glClear(GLbitfield mask)** – очистка буферов
- **glClearColor(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha)** – цвет очистки
- **glClear** - очищает одновременно все заданные буферы
- **glColor{3 4}{b s i f d ub us ui}[v](TYPE red, …)** - для задания цвета объекта
- **glFlush()** - немедленное рисование
- **glEnable()** - удаление невидимых поверхностей методом z-буфера

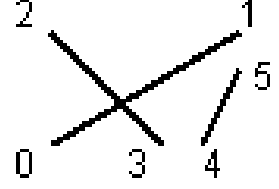
# Задание графических примитивов :

- Все геометрические объекты в OpenGL задаются посредством вершин
- OpenGL работает с однородными координатами (x, y, z, w)
- glVertex{2 3 4}{s i f d}[v](TYPE x, …) – задание вершин
  - glVertex3f(2.3, 1.5, 0.2);
  - GLdouble vect[] = {1.0, 2.0, 3.0, 4.0};
- glBegin() и glEnd() - выделение набора вершин, определяющих этот объект

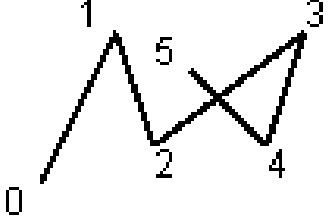
# Параметры glBegin



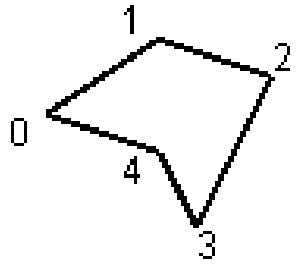

GL_POINTS



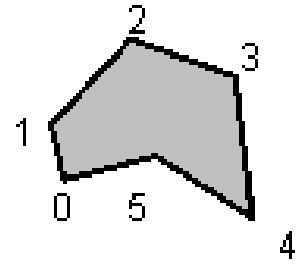

GL_LINES



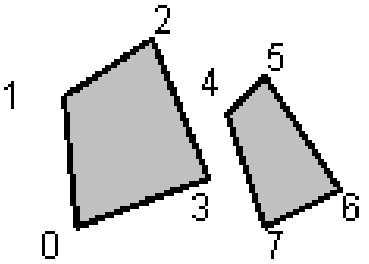

GL_LINE_STRIP



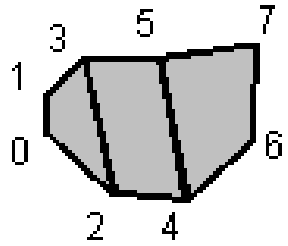

GL_LINE_LOOP



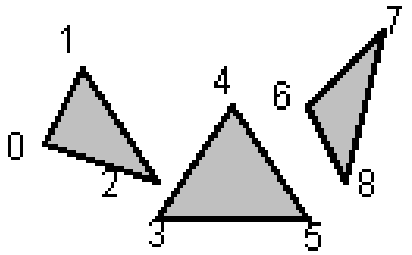

GL_POLYGON



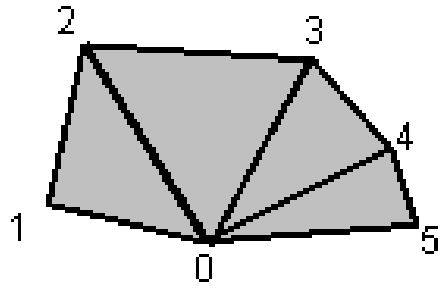

GL_QUADS



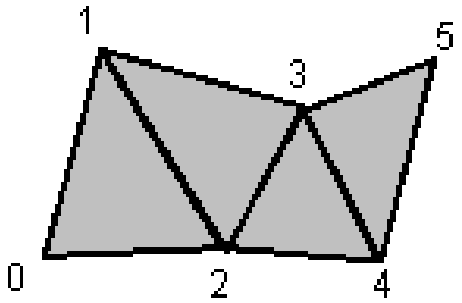

GL_QUAD_STRIP




GL_TRIANGLES




GL_TRIANGLE_FAN




GL_TRIANGLE_STRIP

# Пример - задание окружности:

```
glBegin(GL_LINE_LOOP);
for (int i = 0; I < N; i++)
{
float angle = 2 * M_PI * i / N;
glVertex2f(cos(angle), sin(angle));
}
glEnd();
```

# Рисование точек, линий и многоугольников:

- **glPointSize**(GLfloat size) – задание размеров точки
- **glLineWidth**(GLfloatwidth) - задание ширины линии
- **glPolygonMode**(GLenum face, GLenum mode) – рисование многоугольника
- **glNormal3**{b s i d f}(TYPE nx, TYPE ny, TYPE nz), glNormal3{b s I d f}v(const TYPE * v) – задание вектора нормали

# Матрицы OpenGL

- **матрица моделирования** (model view)- служит для задания положения объекта и его ориентации
- **матрица проецирования** - отвечает за выбранный способ проецирования
- **Матрица текстур**

- **glMatrixMode()** параметры : GL_MODELVIEW, GL_PROJECTION, или GL_TEXTURE
- **glLoadIdentity()**
- **glPushMatrix(), glPopMatrix()** – для помещения и извлечения матрицы из стека

# Преобразования в пространстве

- **glMatrixMode**(GLenum mode) – задание текущей матрицы; параметр mode может принимать значения GL_MODELVIEW, GL_TEXTURE или GL_PROJECTION
- **glTranslate**{f d}(TYPE x, TYPE y, TYPE z) – перенос
- **glRotate**{f d}(TYPE angle, TYPE x, TYPE y, TYPE z) – поворот
- **glScale**{f d}(TYPE x, TYPE y, TYPE z) - масштабирование

# Получение перспективных проекций

- **glFrustrum**(GLdouble left, GLdoubte right, GLdouble bottom, GLdouble top, GLdouble near, GLdoubte far)
- **gluPerspective**(GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar)
- **gluLookAt** (GLdouble eyeX, GLdouble eyeY, GLdouble eyeZ, GIdouble centerX, GLdouble centerY, GLdouble centerZ, GLdouble upX, GLdouble upY, GLdouble up2)

# Получение ортогр-их проекций

- **glOrtho**(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far) – параллельное проецирование
- **glViewport**(GLint x, GLint y, GLsizei width, GLsizei height) - задание области в окне, в которую будет помещено получаемое изображение

# Закрашивание:

- glShadeModel(GLenum mode), где параметр mode принимает значение GL_SMOOTH или GL_FLAT

# Полупрозрачность, α-канал:

- gEnable(GL_BLEND) - разрешение режима прозрачности и смешения цветов

# Освещение (1):

Пользователь может определить до восьми источников света и их свойства, такие, как цвет, положение и направление

- gllight{i f}[v](GLenum light, GLenum pname, TYPE param)
- glEnable(GL_LIGHTING) – разрешение использования источников света
- glEnable(GL_LIGHT0) - применение соответствующего источника

# Освещение (2):

- gtLightModel{i f} [v] (…)- глобальное фоновое освещение
- glMaterial{i f}[v](GLenum face, GLenum pname, TYPE param) – свойства материала

# Наложение текстуры (1):

- glEnable(GL_TEXTURE1D) или glEnable(GL_TEXTURE_2D) – одно- или двумерное текстурирование
- glTexImage2D(GLenum target, GLint level, GLint component, GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const GLvoid *pixels) - задание двумерной текстуры
- Тексель - подходящий элемент текстуры
- glTexParameteri(GL_TEXTURE_2D, GLenum p1, GLenum p2) – выбор текселя

# Наложение текстуры (2):

- Координаты текстуры, её преобразование
  - glMatrixMode(GL_TEXTURE)
  - glRotatef(...}
  - glMatrixMode(GL_MODELV1EW)
- glGenTextures(GLsizei n, GLuint *textures) – генерация имени текстур
- glBindTexture(GLenum target, GLuint texture) – выбор активной текстуры
- glTexCoord{1 2 3 4}{s i f d}[v](TYPE coord, ...) – задание координат текстуры