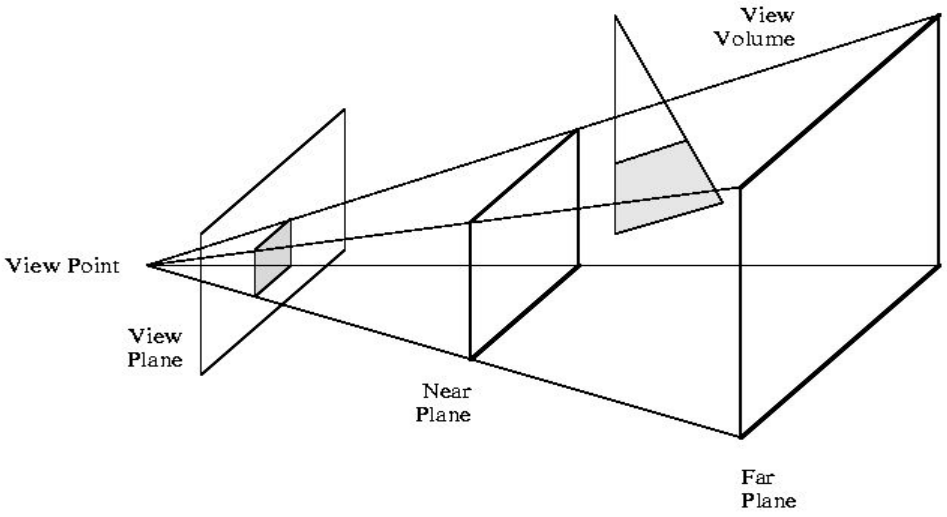
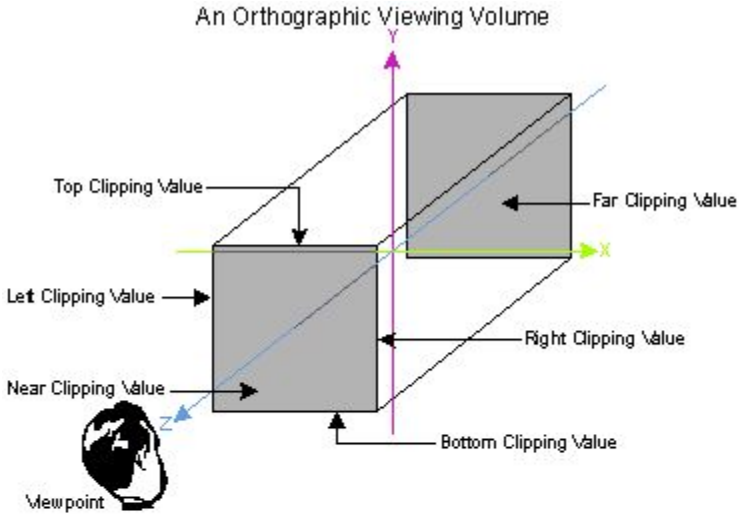
Компьютерная графика

Курс лекций

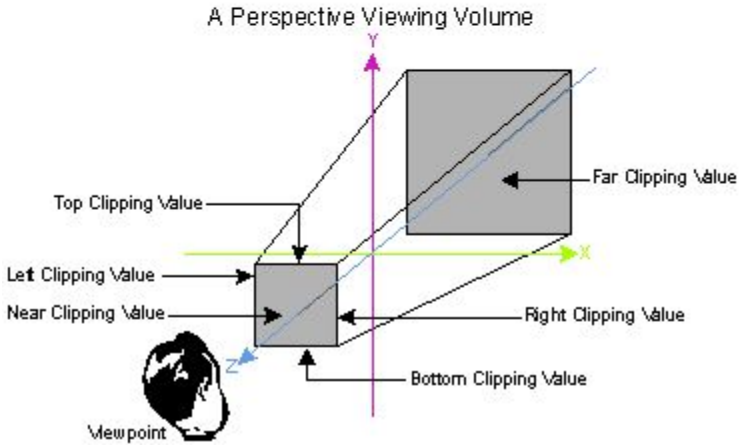
Тема №2. Программные средства машинной графики. Введение в интерактивную машинную графику

Формирование сцены и изображения

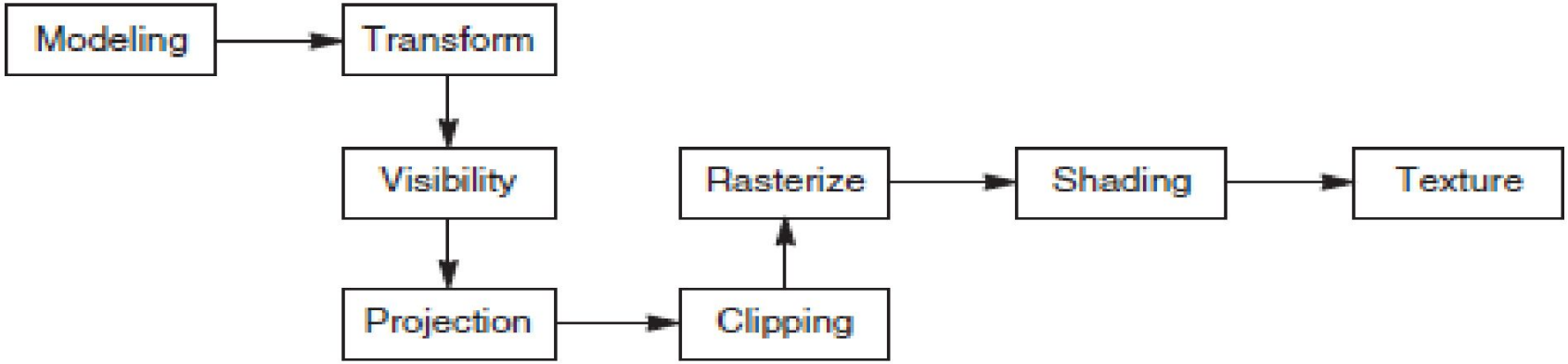




- положение и параметры объектов;
- положение и атрибуты наблюдателя (камеры);
- положение и параметры источников света;
- объем видимости, параметры проецирования и картинная плоскость;



Графический конвейер



- моделирование (modeling) – математическое описание объектов, всей сцены, источников света, с учётом расположения
- отображение (rendering):
  - преобразования (transformation) – задание местоположения;
  - определение видимости (visibility) – область видимости (field of view) + нелицевые поверхности отсечение (clipping);
  - проекция на картинную плоскость (projection);
  - растеризация (rasterization);
  - закраска (shading);
  - текстурирование (texturing).

Базовые алгоритмы машинной графики

- преобразование систем координат;
- удаление невидимых поверхностей;
- отсечение невидимых областей;
- отрисовка базовых графических примитивов (точек, прямых, ломаных и т.п.);
- заливка / штриховка (растровая развертка сплошных областей);

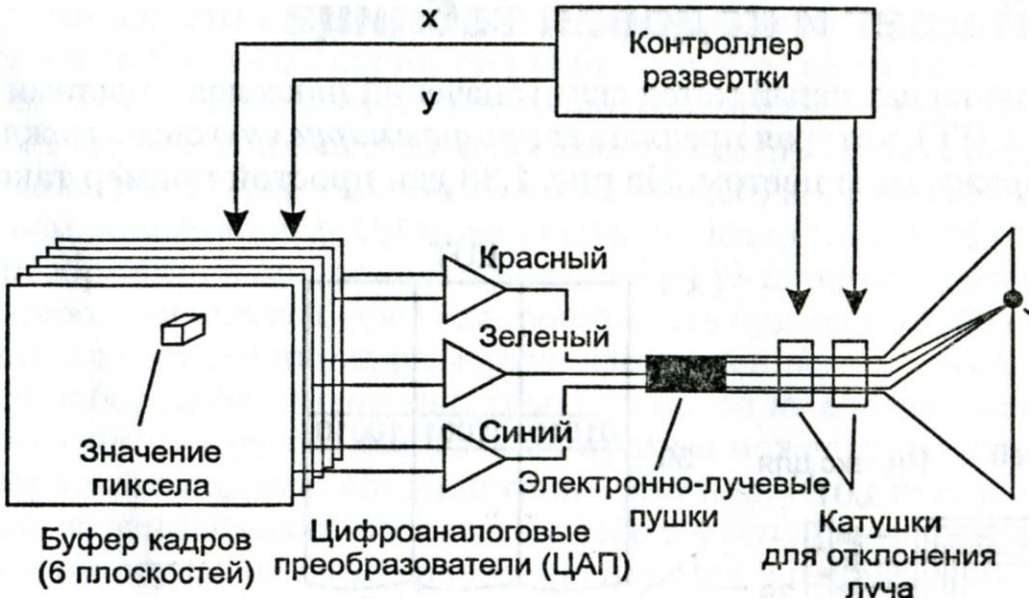
Средства повышения реалистичности изображения

- модели освещения (диффузное, направленное);
- отражение (диффузное, зеркальное) – свойства материала;
- построение теней;
- фактуры (текстуры) – нанесение некоторого изображения на поверхность или внесение возмущения;
- преломление;
- прозрачность (blending, комбинация цветов в различных режимах), цвет;
- движение, анимация.

Векторные графические системы: процедура регенерации

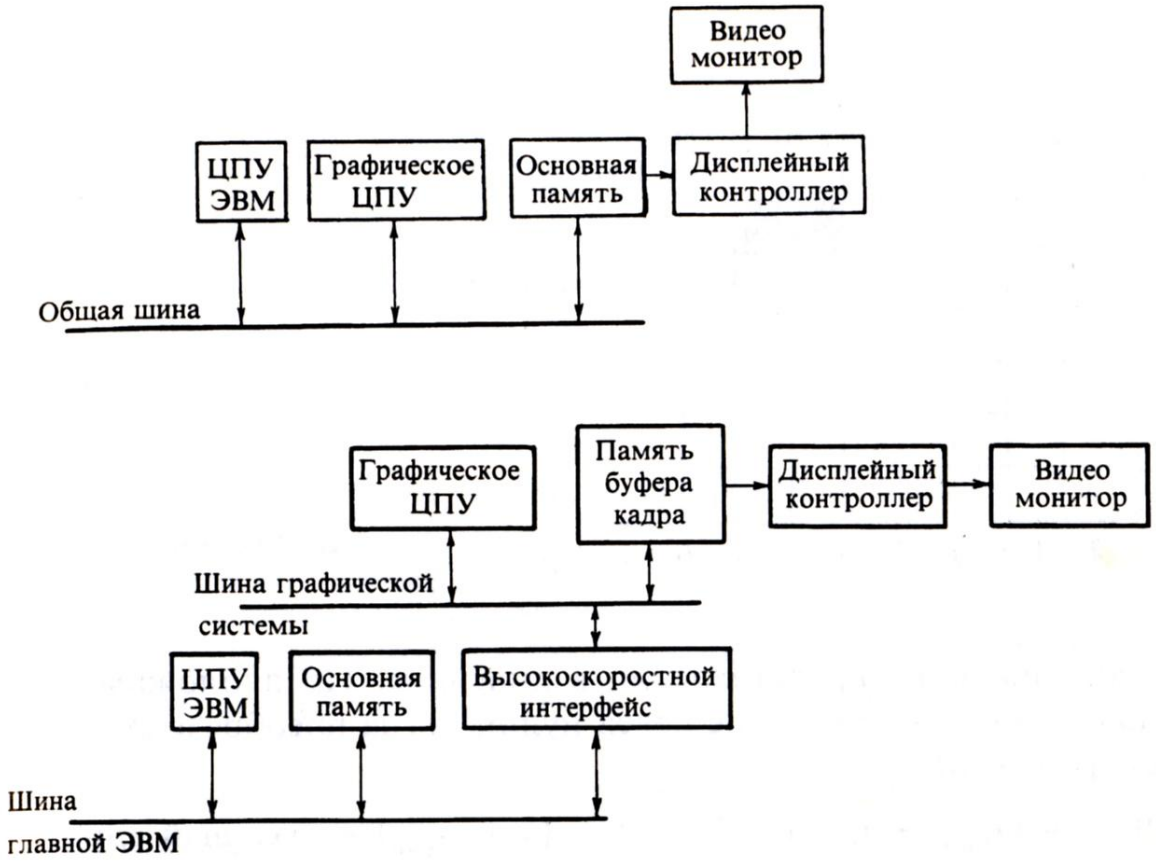
- *Принцип записи изображения*: произвольное сканирование луча.
- *Примитивы*: отрезки, литеры, кривые определенного типа.
- Скорость регенерации определяется количеством элементов в дисплейном файле.
- *Дисплейный файл* – данные, используемые для формирования изображения (набор команд дисплейного контроллера).
- *Геометрический процессор*: выполняет геометрические преобразования (поворот, перенос, масштабирование, проецирование, отсечение).
- *Основные ограничения*: невозможна заливка сплошных областей (заменяется штриховкой) и плавный переход цвета.

Растровые графические системы

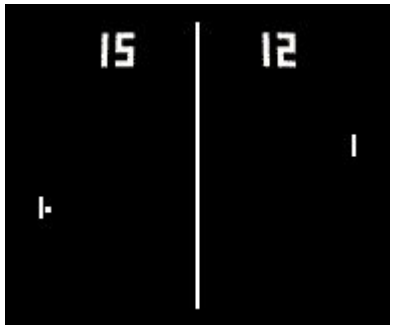


- *Принцип записи изображения*: построчное сканирование луча
- *Примитив*: точка (пиксель, pixel = picture element)
- *Видеоконтроллер* (дисплейный контроллер): выполняет процедуру регенерации, называемую *разверткой* (отображение буфера кадра)
- Необходима *растровая развертка* примитивов)
- *Буфер кадра* (буфер регенерации):
  - обеспечивает промежуточное хранение изображения (растеризованных графических примитивов)
  - основная характеристика: количество цветов (глубина, количество битовых плоскостей)

Архитектура растровых графических систем с буфером кадра



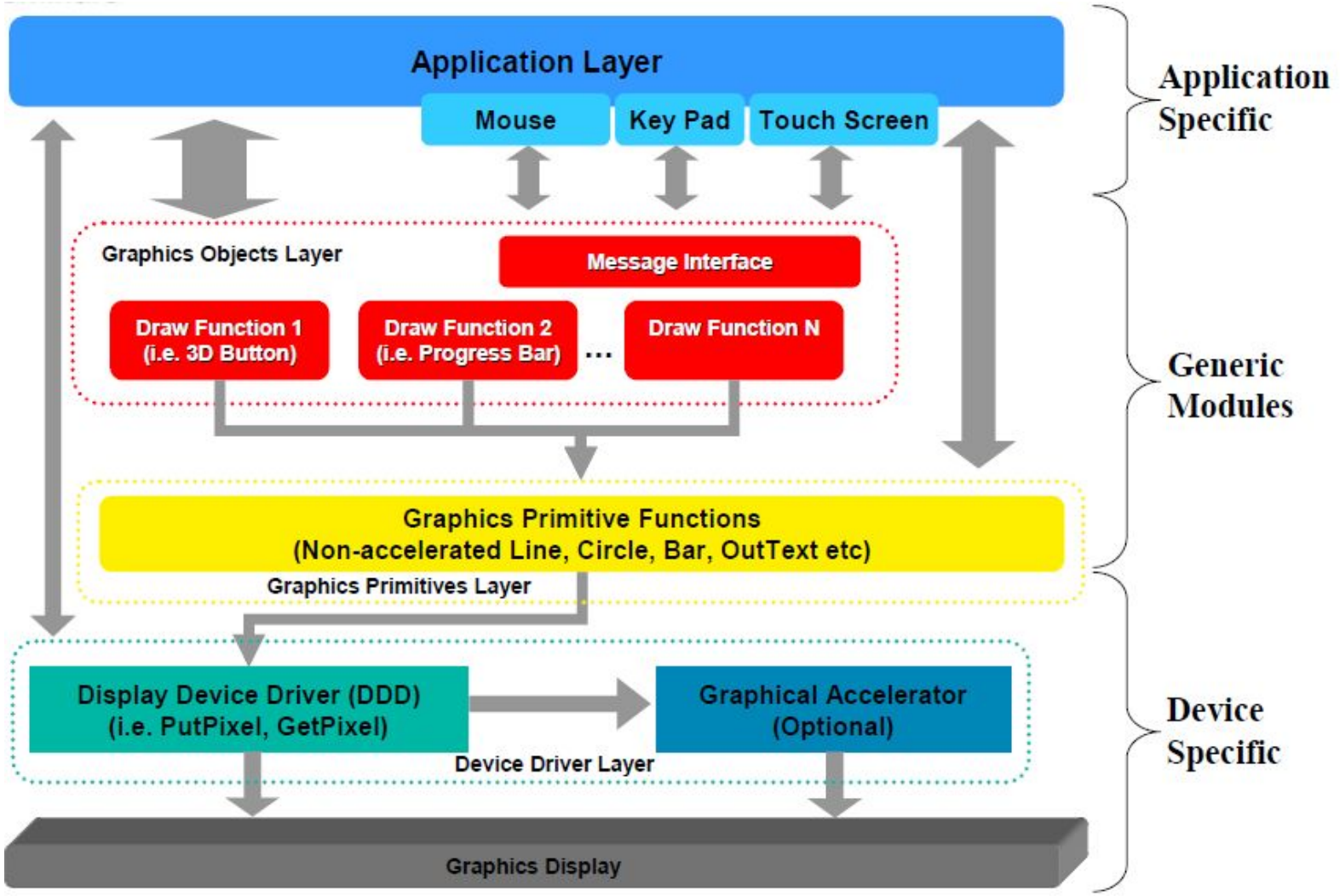
- ≈1960-е
- 1970г. 1K RAM
- 1974 z-буфер, Кэтмулл (Catmull)
- 1984 альфа-канал, Loren Carpenter
- 1982 VLSI-процессор (Geometry Engine, Silicon Graphics) → IRIS (Integrated Raster Imaging System)
- GPU(Geometry/Graphics Processing Unit): *n*VIDIA GeForce 256



Архитектура прикладной графической системы

- устройства ввода/вывода;
- операционная система;
- базовая графическая система;
- проблемно-ориентированный уровень;
- приложение

Архитектура прикладной графической системы: пример



Стандартизация прикладных графических систем

- *цель:*

  переносимость графических систем (получение одинакового визуального результата на различных платформах)

- *подход:*

  стандартизация интерфейса между графическим ядром системы (базовой графической системой), реализующим собственно графические функции, и моделирующей системой - проблемно-ориентированной прикладной программой, использующей функции графического ядра → API (Application Programming Interface)

- *требования к базовой графической системе:*
  - поддержка функций двумерной и трехмерной (2D/3D) графики;
  - аппаратная и платформенная независимость:
    - независимость от вычислительных систем;
    - независимость от языков программирования;
    - независимость от области применения;
    - независимость от графических устройств;
  - аппаратная реализация базовых функций;
  - стабильность (совместимость с разработанным ранее ПО).

Архитектура переносимой графической системы

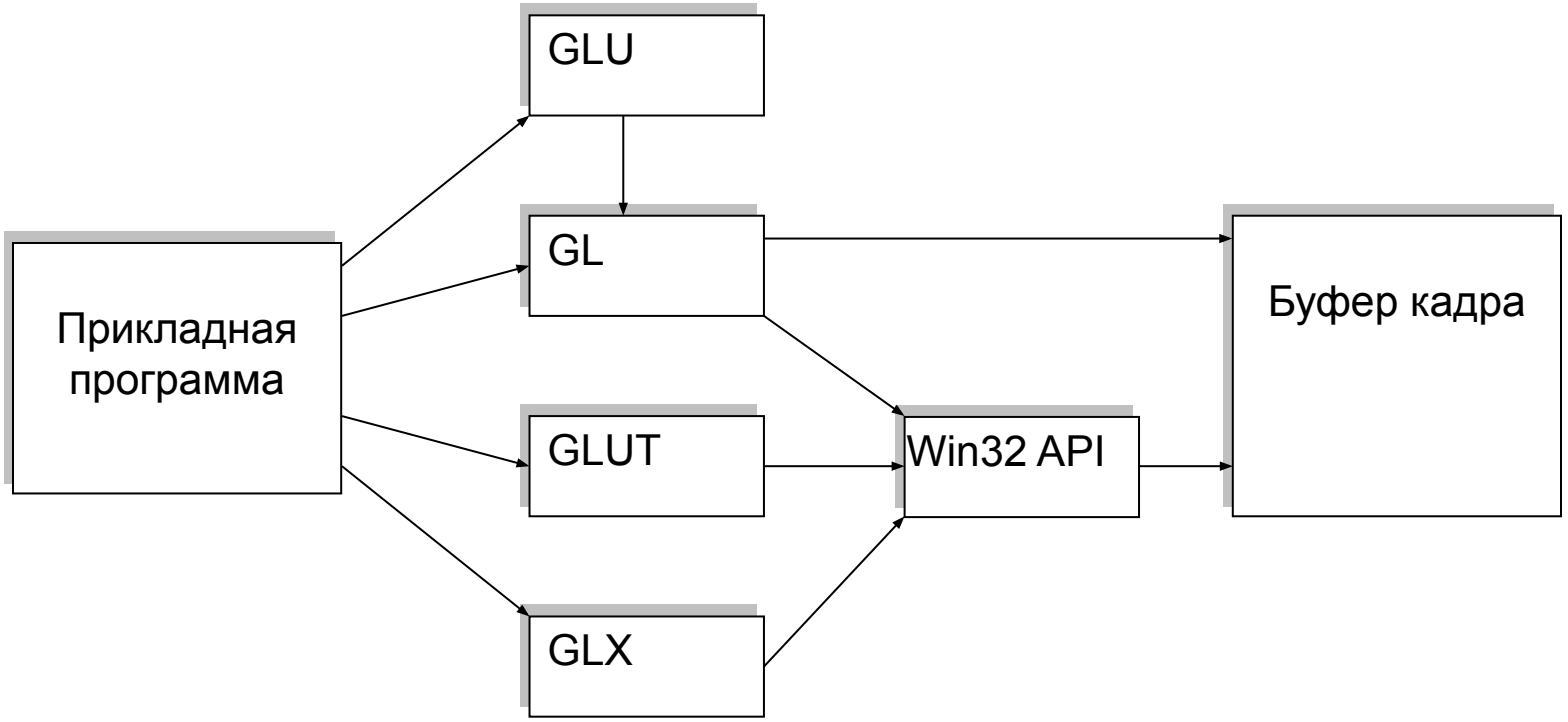
3 уровня стандартизации:

- приложение (данные);
- базовая графическая система (выбор базовых функций) – независимость от области применения;
- драйверы графических устройств (виртуализация графических устройств: абстракция возможностей устройств) – аппаратная независимость.

Классификация графических стандартов

- графические интерфейсы (наборы функций графических подсистем)
  - интерфейс виртуальных устройств:
    - CGI (Computer Graphics Interface);
  - интерфейс базовой графической системы:
    - GKS, GKS-3D (Graphical Kernel System);
    - PHIGS, PHIGS+ (Programmer's Hierarchical Interactive Graphics Standard);
    - IRIS GL (Integrated Raster Imaging System), OpenGL (Graphics Library – Silicon Graphics, 1992);
- графические протоколы (порядок и правила обмена информацией)
  - аппаратно-зависимые графические протоколы (команды графических устройств):
    - PCL (Printer Communication Language);
  - аппаратно-независимые графические протоколы (метафайлы) – процедурное описание изображения в функциях виртуального графического устройства:
    - DXF (Data eXchange Format);
    - WMF / EMF (Windows / Enhanced Metafile);
    - PostScript;
  - прикладные проблемно-ориентированные графические протоколы (САПР: IGES-Initial Graphics Exchange Specification)
  - растровые графические файлы: TIFF, GIF, PIC, PCX, BMP

Структура библиотеки OpenGL



1. GL (Graphics Library) – базовые функции графической библиотеки
2. GLU(GL Utility) – библиотека утилит (функции реализованы через базовые)
3. GLUT (GL Utility Toolkit), GLX, GLFW и т.д. – функции взаимодействия с оконной подсистемой и пользователем

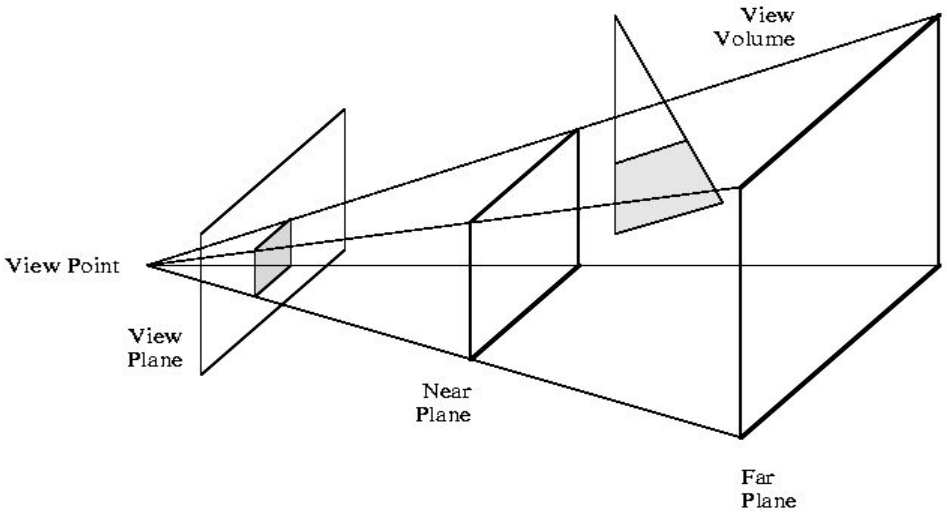
Организация библиотеки OpenGL

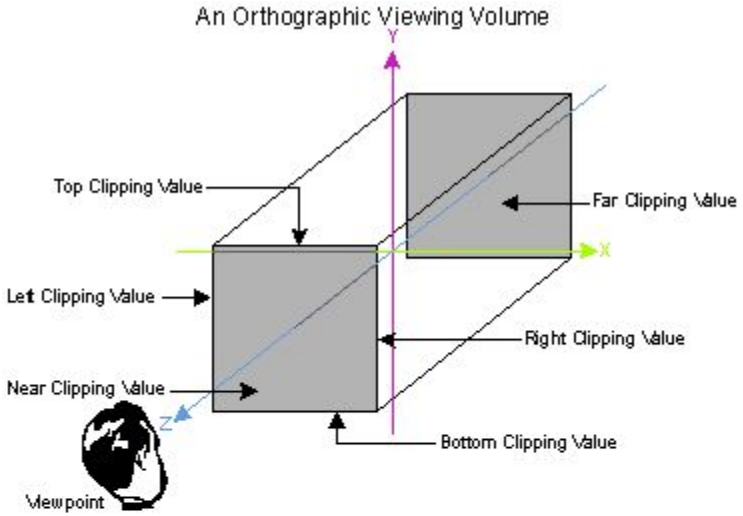
- *аппаратно-платформенная независимость*
  - библиотека не содержит никаких команд конфигурации буфера кадра или инициализации
- *процедурный механизм*
  - библиотека определяет операции двумерной и трехмерной графики (отсутствуют функции описания и моделирования сложных геометрических объектов)
- *модель выполнения «клиент-сервер»*
  - клиент: прикладное приложение, вырабатывает команды в форме вызова функций графической библиотеки;
  - сервер (локальный или удаленный): библиотека OpenGL, интерпретирует и обрабатывает команды

Организация библиотеки OpenGL: примитивы и команды

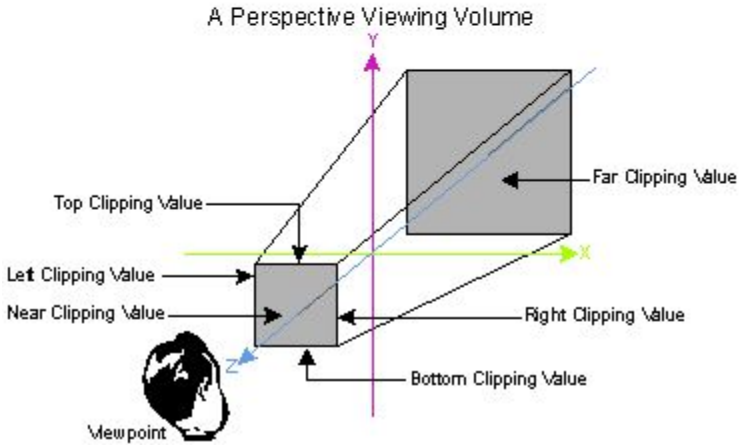
- *графическими примитивами* являются объекты, определяемые набором из одной или более *вершин* (точки, линии, многоугольники);
- *состояние* - набор внутренних переменных (*режимов*), определяющих параметры отображения *графических примитивов;* при этом режимы и их изменение независимы друг от друга;
- отображение каждой из вершин зависит от набора связанных с ней *атрибутов* (координаты, цвет, нормаль, текстурные координаты, флаги и т.п.), и не зависит от отображения других вершин (ИСКЛЮЧЕНИЕ: отсечение);
- реализован *графический конвейер:*
  - процедура отображения включает в себя несколько последовательных этапов обработки графических данных;
  - все *команды* выполняются исключительно в порядке их следования;

Формирование сцены и изображения

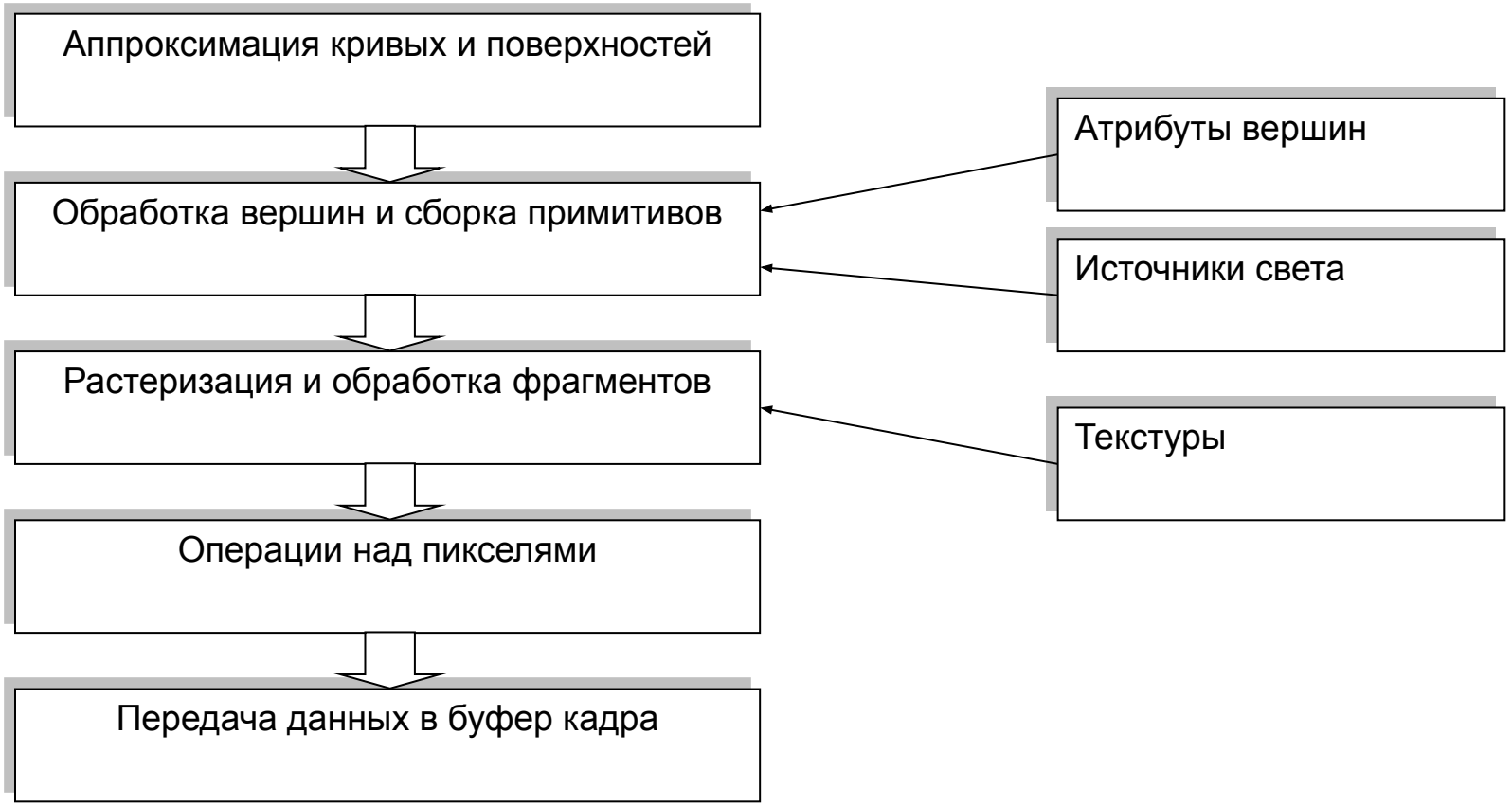




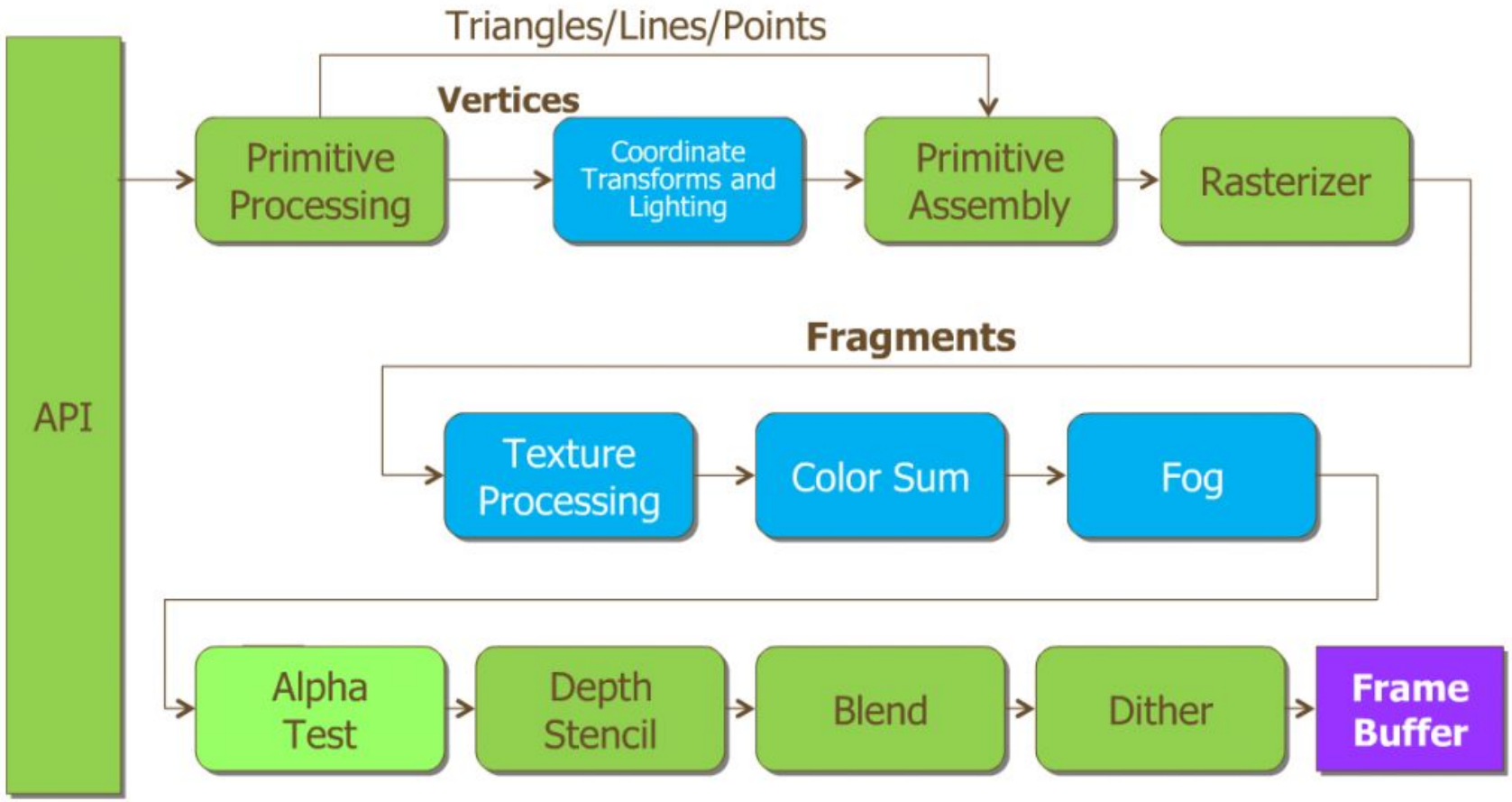
- положение и параметры объектов;
- положение и атрибуты наблюдателя (камеры);
- положение и параметры источников света;
- объем видимости, параметры проецирования и картинная плоскость;



Функционирование конвейера OpenGL



Объекты, обрабатываемые конвейером OpenGL



вершины (vertex) □ примитивы (primitive) □ фрагменты (fragment) □ пиксели (pixel)

Open GL и DirectX

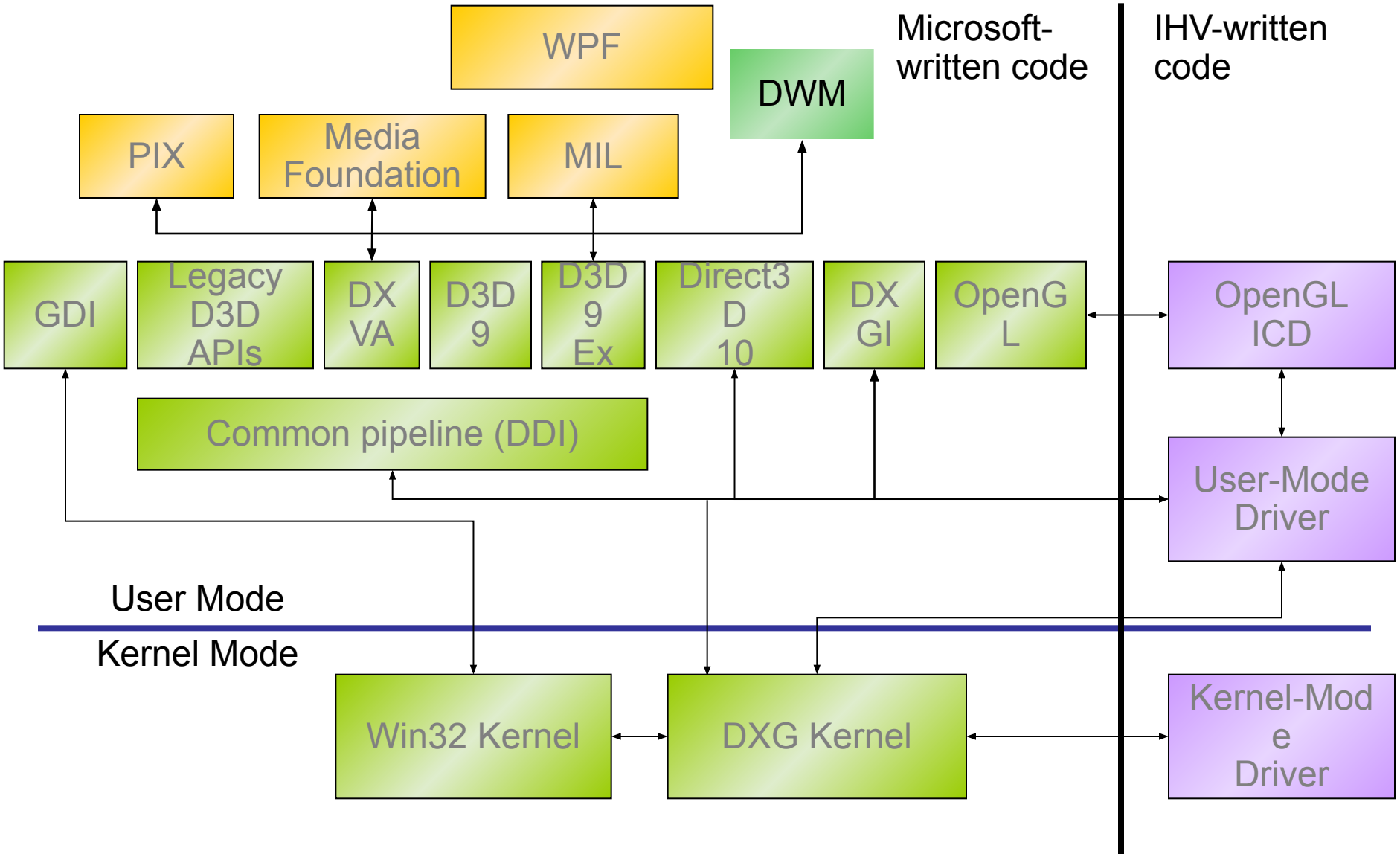
OpenGL

- открытые стандарты:
  - OpenGL(1.1-4.3);
  - GLSL(1.10-4.30);
  - OpenGL ES (1.0, 1.1, 2.0, 3.0);
- поддерживается производителями аппаратного обеспечения, а также позволяет использовать расширения;
- полная абстракция от платформы и ОС;
- первоначально ориентирован на профессиональное применение;
- ОС: Windows (OpenGL 1.1 в случае отсутствия драйверов), Linux, MacOS, iOS, Android, Symbian

DirectX

- собственный стандарт Microsoft (2.0-11.1);
- поддерживается производителями аппаратного обеспечения;
- обеспечивает взаимодействие с устройствами;
- первоначально ориентирован на применение разработчиками игр;
- ОС: Windows (преимущественно);

OpenGL и Windows



Нотация OpenGL

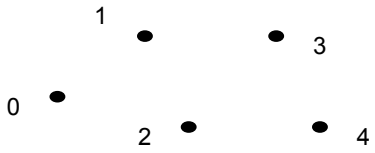
- *константы*
  - GL_*XXX*, GLU_*XXX*, GLUT_*XXX*
- *типы данных*
  - GL*typename (GLenum, GLboolean, GLbitfield, GLbyte, GLshort, GLint, GLsizei, GLubyte, GLushort, GLuint, GLfloat, GLdouble,GLvoid u m.д.)*
- *команды*
  - префикс: gl*Command,* glu*Command,* glut*Command*
  - суффикс:

    type **gl*Command*[1 2 3 4][b s i f d ub us ui][v]** (type1 *arg1*,…,typeN *argN*)
    - **[1 2 3 4] -** число аргументов команды
    - **[b s i f d ub us ui] -** тип аргумента
    - **[v] -** в качестве параметров функции используется указатель на массив значений

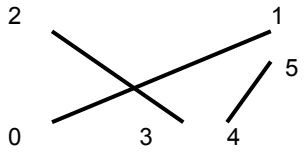
OpenGL: основы работы с графическими примитивами

- определение режимов:
  - включение режима
    glEnable (GLenum *mode*)
  - выключение режима
    glDisable (GLenum *mode*)
- очистка буфера кадра:
  glClearColor *(*clampf *r,* clampf *g,* clampf *b,* clampf *a)*
  glClear*(*bitfield *buf)* *// GL_COLOR_BUFFER_BIT*
- задание примитивов:
  glBegin (GLenum *mode*); // тип примитива
  glEnd (void);
- задание вершин и их атрибутов:
  - координаты (начало координат – в левом нижнем углу)**:**
    glVertex[2 3 4][s i f d] [v] (type [*]*coords*)
  - цвет:
    glColor[3 4][b s i f] [v] (GLtype [*]*components*)
- задание размера точки и ширины линии:
  glPointSize(GLfloat *size* **)**
  glLineWidth(GLfloat *size* **)**

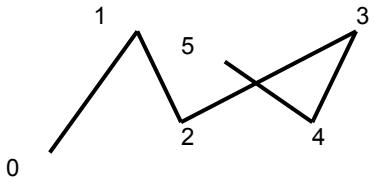
Основные графические примитивы OpenGL



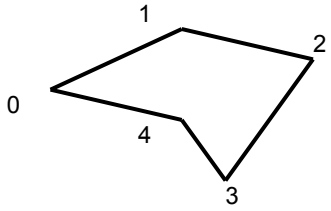
GL_POINTS



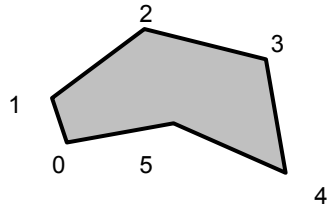
GL_LINES



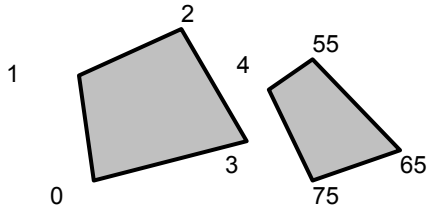
GL_LINE_STRIP



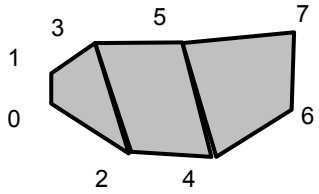
GL_LINE_LOOP



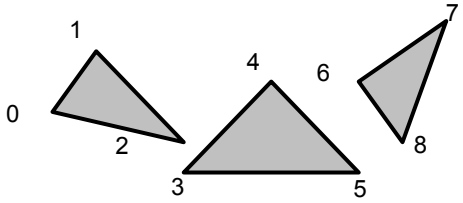
GL_POLYGON



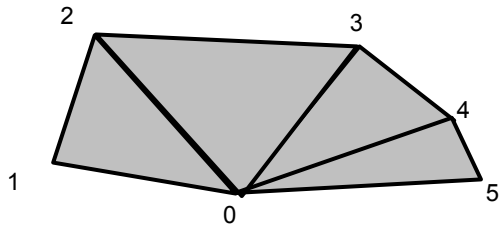
GL_QUADS



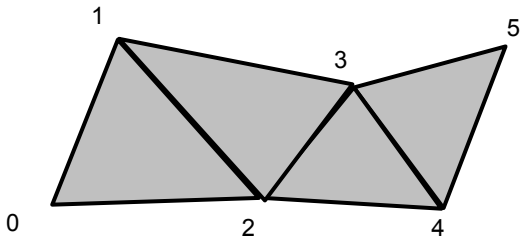
GL_QUAD_STRIP



GL_TRIANGLES



GL_TRIANGLE_FAN



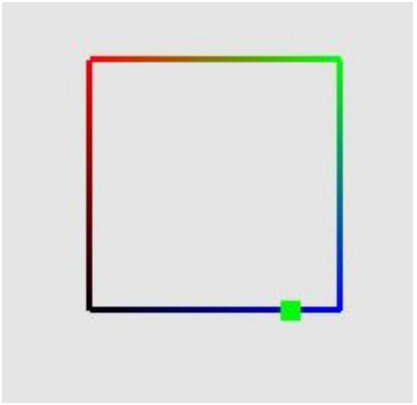
GL_TRIANGLE_STRIP

Примитивы OpenGL: пример

```
glPointSize(2.0);
glBegin(GL_POLYGON);
glColor3f(0,0,0);       glVertex2f(50,50);
glColor3f(1,0,0);       glVertex2f(50,175);
glColor3f(0,1,0);       glVertex2f(175,175);
glColor3f(0,0,1);       glVertex2f(175,50);
glEnd();

glLineWidth(3);
glBegin(GL_LINE_LOOP);
glColor3f(0,0,0);       glVertex2f(250,250);
glColor3f(1,0,0);       glVertex2f(250,375);
glColor3f(0,1,0);       glVertex2f(375,375);
glColor3f(0,0,1);       glVertex2f(375,250);
glEnd();

glPointSize(10.0);
glBegin(GL_POINTS);
glColor3f(0,1,0);
glVertex2f(350,250);
glEnd();
```



Организация буфера кадра в OpenGL

- буфер кадра (frame buffer), как результат преобразования фрагментов в отдельные пиксели, объединяет набор логических буферов:
  - буфер цвета (RGB + прозрачность): смешивание цветов;
  - буфер глубины (глубина – расстояние от наблюдателя до объекта) удаление невидимых линий и поверхностей;
  - буфер маски (stencil buffer): позволяет выводить только те пикселы изображения, которые удовлетворяют некоторому заданному условию (тесту маскирования) построение теней и отражений;
  - аккумулирующий буфер (буфер – накопитель, accumulation buffer): фактически, дополнительный буфер цвета с возможностью попиксельного накопления устранение ступенчатости;
- формирование стереоизображений: левый / правый буфер;
- двойная буферизация: рабочий / вспомогательный буфер;

Библиотека GL: основные группы функций

- *функции определения режимов OpenGL;*
- *функции описания примитивов* (определение объектов нижнего уровня иерархии - примитивов, которые способна отображать графическая подсистема);
- *функции задания атрибутов* (цвет, характеристики материала, текстуры, параметры освещения);
- *функции визуализации* (определение положения наблюдателя в виртуальном пространстве, параметров объектива камеры);
- *функции геометрических преобразований* (задание различных преобразований объектов – поворота, переноса, масштабирования);
- *функции описания источников света* (описание положения и параметров источников света, расположенных в трехмерной сцене).

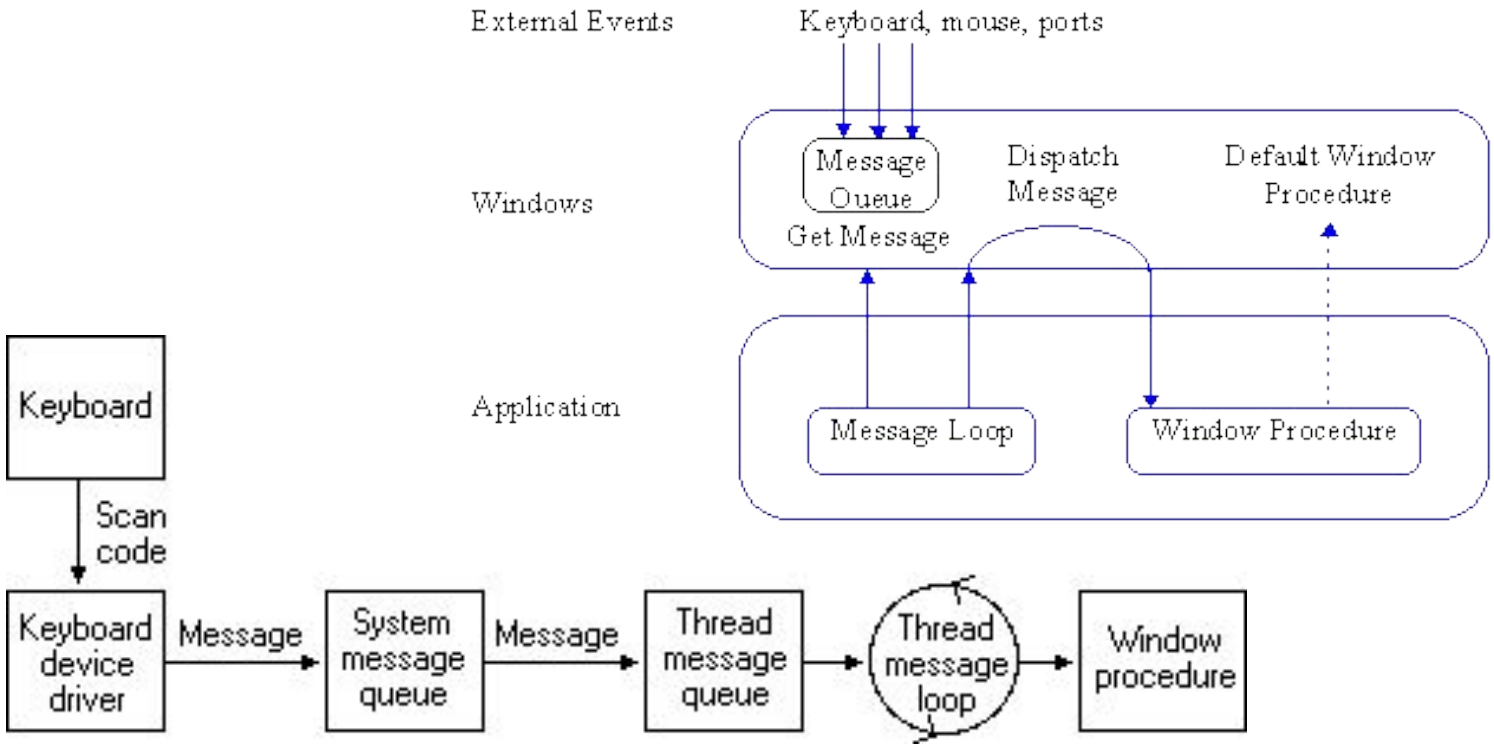
Библиотека GLU: основные группы функций

расширение библиотеки GL, реализация более сложных операций, но исключительно через вызовы функций библиотеки GL

- *функции преобразования координат* (определение параметров проецирования для некоторых проекций и положения наблюдателя);
- *функции работы с текстурами;*
- *функции триангуляции многоугольников* (polygon tesselation);
- *функции отображения элементарных геометрических фигур* (сфера, цилиндр, диск и т.п.);
- *функции аппроксимации и отображения кривых и поверхностей* (NURBS, Non-Uniform Rational B-Spline).

Библиотеки GLUT / GLFW etc. : философия

- изолируют особенности реализации оконной подсистемы и взаимодействия с пользователем необходим набор процедур инициализации;
- организуют / позволяют организовать собственный цикл обработки событий, реализуя концепцию «программы, управляемой событиями».



Библиотека GLFW: схема работы

– процедура инициализации и работы с библиотекой

```
if (!glfwInit()) exit(EXIT_FAILURE);

window = glfwCreateWindow(640, 480, "Simple example", NULL,
  NULL);

if (!window)
{
     glfwTerminate();
     exit(EXIT_FAILURE);
}

glfwMakeContextCurrent(window);
glfwSetKeyCallback(window, key_callback);
while (!glfwWindowShouldClose(window))
{
   …
   glfwSwapBuffers(window);
   glfwPollEvents();
}

glfwDestroyWindow(window);
glfwTerminate();

exit(EXIT_SUCCESS);
```

Библиотека GLFW: работа с окнами

- определение параметров контекста окна

```
void glfwWindowHint (int target, int hint )
void glfwDefaultWindowHints (void )
```

- создание окна

```
GLFWwindow* glfwCreateWindow (int width,
  int height, const char * title, GLFWmonitor
  * monitor, GLFWwindow * share )
```

- установка / определение текущего контекста

```
void glfwMakeContextCurrent (GLFWwindow * window)
GLFWwindow* glfwGetCurrentContext (void )
```

- переключение рабочего и фонового буферов

```
void glfwSwapBuffers (GLFWwindow * window)
void glfwSwapInterval (int interval)
```

- уничтожение окна

```
void glfwDestroyWindow (GLFWwindow * window)
```

Библиотека GLFW: обработка событий и состояние ввода

- обработка событий

```
void glfwPollEvents (void )
void glfwWaitEvents (void )
```

- состояние ввода

```
void glfwGetCursorPos (GLFWwindow * window, double * xpos,
  double * ypos )
void glfwSetCursorPos (GLFWwindow * window, double xpos,
  double ypos )
int glfwGetInputMode (GLFWwindow * window, int mode )
void glfwSetInputMode (GLFWwindow * window, int mode,
  int value )
int glfwGetKey (GLFWwindow * window, int key ) //
  GLFW_PRESS or GLFW_RELEASE
int glfwGetMouseButton (GLFWwindow * window, int button )
```

- сигнализация закрытия окна

```
int glfwWindowShouldClose (GLFWwindow * window)
void glfwSetWindowShouldClose (GLFWwindow * window,
  int value )
```

Библиотека GLFW: свойства окна

– определение / задание параметров контекста окна

```
int glfwGetWindowAttrib (GLFWwindow * window, int attrib )
void glfwGetFramebufferSize (GLFWwindow * window, int * width, int
  * height )

GLFWmonitor* glfwGetWindowMonitor (GLFWwindow * window)

void glfwGetWindowPos (GLFWwindow * window, int * xpos, int * ypos )
void glfwSetWindowPos (GLFWwindow * window, int xpos, int ypos )

void glfwGetWindowSize (GLFWwindow * window, int * width, int
  * height )
void glfwSetWindowSize (GLFWwindow * window, int width, int height )

void* glfwGetWindowUserPointer (GLFWwindow * window)
void glfwSetWindowUserPointer (GLFWwindow * window, void * pointer )

void glfwHideWindow (GLFWwindow * window)
void glfwShowWindow (GLFWwindow * window)

void glfwIconifyWindow (GLFWwindow * window)
void glfwRestoreWindow (GLFWwindow * window)

void glfwSetWindowTitle (GLFWwindow * window, const char * title )
```

Библиотека GLFW: функции обратного вызова

- оконные

```
GLFWframebuffersizefun glfwSetFramebufferSizeCallback (GLFWwindow * window,
  GLFWframebuffersizefun cbfun )
GLFWwindowfocusfun glfwSetWindowFocusCallback (GLFWwindow * window,
  GLFWwindowfocusfun cbfun )
GLFWwindowiconifyfun glfwSetWindowIconifyCallback (GLFWwindow * window,
  GLFWwindowiconifyfun cbfun )
GLFWwindowposfun glfwSetWindowPosCallback (GLFWwindow * window,
  GLFWwindowposfun cbfun )
GLFWwindowrefreshfun glfwSetWindowRefreshCallback (GLFWwindow * window,
  GLFWwindowrefreshfun cbfun )
GLFWwindowsizefun glfwSetWindowSizeCallback (GLFWwindow * window,
  GLFWwindowsizefun cbfun )
```

- ввод

```
GLFWcharfun glfwSetCharCallback (GLFWwindow * window, GLFWcharfun cbfun )
GLFWcursorenterfun glfwSetCursorEnterCallback (GLWwindow * window,
  GLFWcursorenterfun cbfun )
GLFWcursorposfun glfwSetCursorPosCallback (GLFWwindow * window,
  GLFWcursorposfun cbfun )
GLFWkeyfun glfwSetKeyCallback (GLFWwindow * window, GLFWkeyfun cbfun )
GLFWmousebuttonfun glfwSetMouseButtonCallback (GLFWwindow * window,
  GLFWmousebuttonfun cbfun )
GLFWscrollfun glfwSetScrollCallback (GLFWwindow * window, GLFWscrollfun cbfun )
```

- обработка ошибок

```
GLFWerrorfun glfwSetErrorCallback (GLFWerrorfun cbfun)
```

Вопросы к экзамену

- Стандартизация графических систем: цели, требования, иерархический подход
- Архитектура переносимой графической системы. Классификация стандартов. Стандарт OpenGL: структура библиотеки
- Основные графические примитивы OpenGL и их атрибуты
- Объекты, рассматриваемые на различных стадиях работы графического конвейера
- Организация буфера кадра в OpenGL. Создание стереоизображений
- Управление приложением OpenGL: использование библиотеки GLFW – сценарий работы, состояние, обработка событий и функции обратного вызова