计算机图形学理论和应用

讲　　授：董兰芳

研究方向：科学计算可视化(Computing and Visualization)

智能图像处理与分析

（Intelligent Image Processing and Analysis ）

计算机动画（Computer Animation ）

Email:lfdong@ustc.edu.cn

中国科学技术大学

视觉计算与可视化实验室

(Vision Computing and Visualization Laboratory)

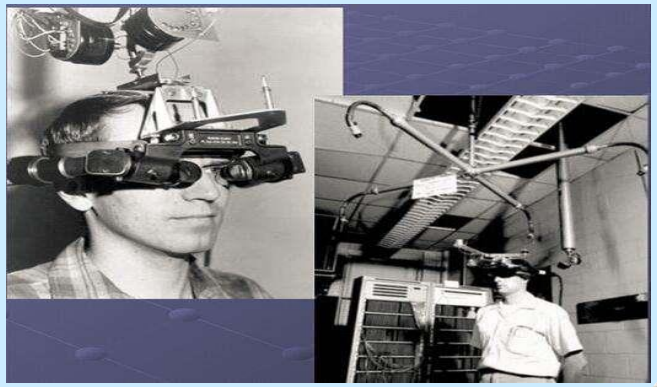
课程QQ群：　706514213

第一章 图形系统

- 计算机技术最重要的进展：

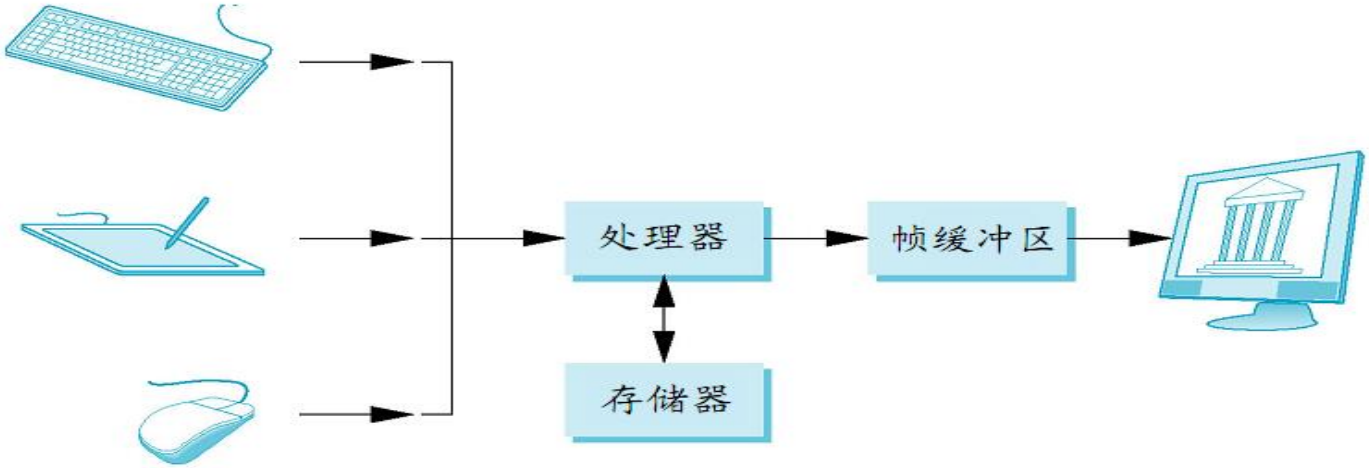
  用户能同计算机显示系统进行交互。

- Sketchpad项目：Ivan Sutherland 建立了刻画交互式计算机图形学的基本框架
  - 用户在显示器上看到一个对象。
  - 用户利用输入设备(光笔、鼠标、跟踪球等)点选该对象。
  - 对象发生了改变(移动、旋转、变形等) 。
  - 重复上述过程。

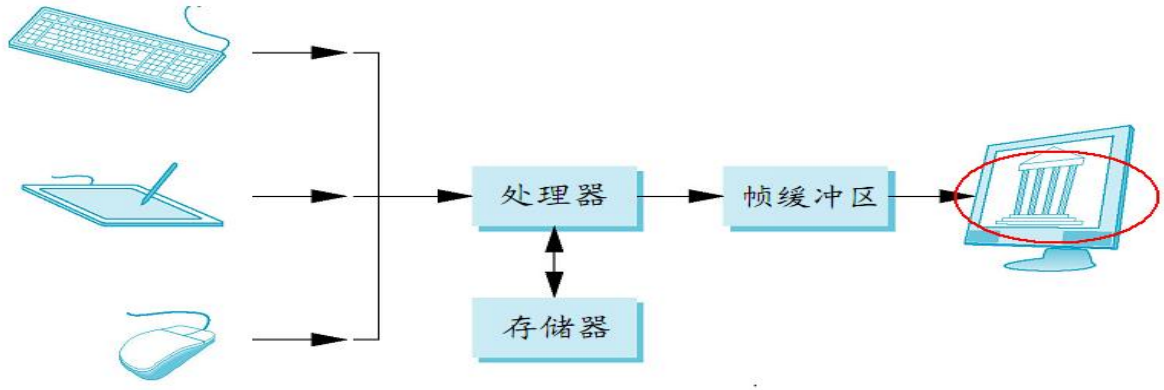


第一章 图形系统

❖ 图形系统组成



1.1 视频显示设备



1.1.1 刷新式CRT

1.1.2 光栅扫描显示器

1.1.3 随机扫描显示器

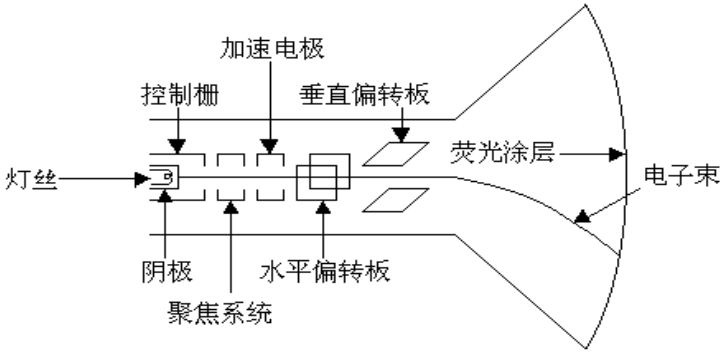
1.1.4 彩色CRT监视器

1.1.5 平板显示器

1.1.7 三维观察设备

1.1.8 立体感和虚拟现实

- 快速控制电子束反复重画屏幕
- CRT结构:



1.1 视频显示设备

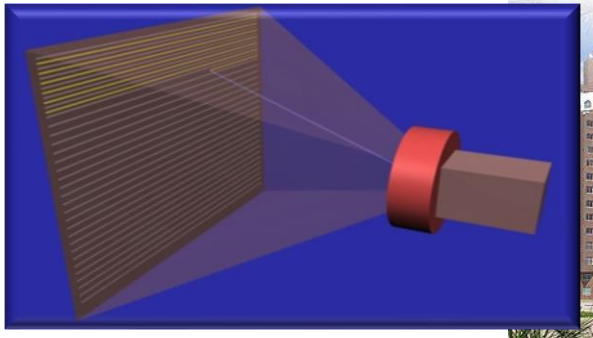
1.1.2 光栅扫描的显示系统

- **工作原理**

  - ✓ 电子束受偏转部件的控制，不断从左到右、从上到下将图像逐行逐点的扫描到显示屏上，通过控制电子束的强弱产生黑白、灰度或彩色的图像。
  - ✓ 整个显示屏面扫描线为N行，每一行又可以分为M个点。这样，整个屏幕为M*N的点阵.

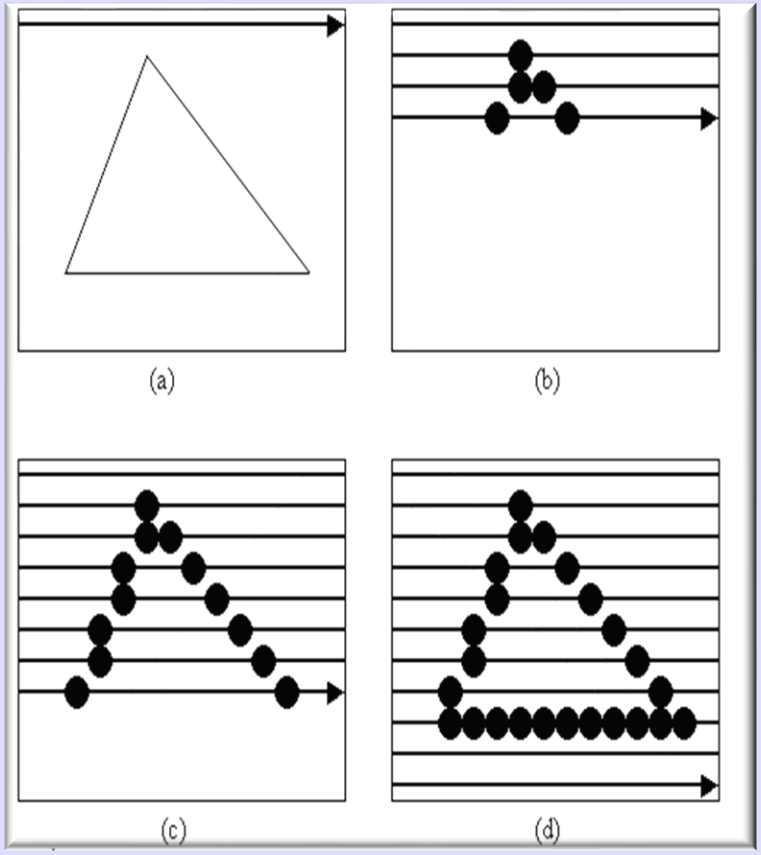
- **特点：光栅扫描**

- **基本概念:**
  - 扫描行
  - 帧：整个屏幕范围
  - 帧缓存：保存图形定义点的存储器，又称为刷新缓存。
  - 颜色缓存：刷新缓存用来存储屏幕颜色值，因此又称为颜色缓存。
  - 像素：屏幕上的一个点。
  - 缓存深度：每像素的位数,也称位平面
  - 位图：每像素一位的帧缓存。
  - 像素图：每像素多位的帧缓存



1.1 视频显示设备

- 绘图过程



- 光栅显示系统的特点
  - 优点：
    - ✓成本低
    - ✓易于绘制填充图形
    - ✓色彩丰富
    - ✓刷新频率一定，与图形的复杂程度无关
    - ✓易于修改图形
  - 缺点：
    - ✓需要扫描转换
    - ✓会产生混淆

1.1 视频显示设备

1.1.3 随机扫描显示器

- 特点：电子束可随意移动，只扫描荧屏上要显示的部分。
- 逻辑部件包括:刷新存储器(Refreshing Buffer)、显示处理器（DPU:Display Processing Unit）、 CRT

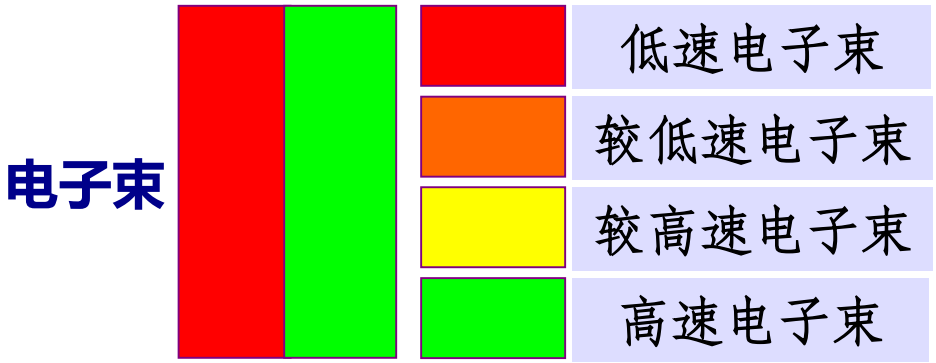
随机扫描显示器与光栅扫描显示器的比较

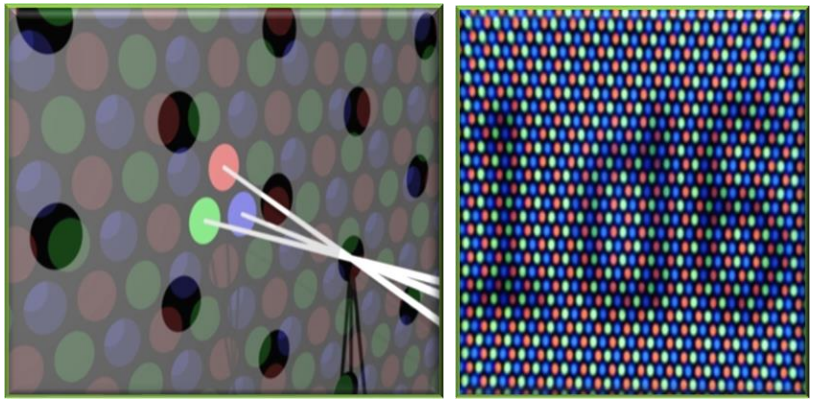
	随机扫描显示器	光栅扫描显示器
扫描方式	随机（完全按显示文件）	固定，一点接一点，一行接一行
显示信息	根据x，y的偏转方向	x，y是显示地址，z是显示信息
存储方式	指令序列	位图
刷新频率	限制图形复杂性	与图形复杂性无关
图形类型	点，线，字符	点，线，字符，面，实体
彩色与灰度	差	丰富
分辨率	略高（与刷新频率有关）	略低（刷新频率恒定）
动态交互能力	强	弱（扫描转换）
图形语义	接近（指令）	无（位图）

1.1 视频显示设备

1.1.4 彩色CRT监视器

- 产生彩色的常用方法：射线穿透法、影孔板法
- 射线穿透法
  - 原理：两层荧光涂层，红色光和绿色光两种发光物质，电子束轰击穿透荧光层的深浅，决定所产生的颜色





  - 应用：主要用于画线显示器
  - 优点：成本低
  - 缺点：只等产生有限几种颜色

如果每支电子枪发出的电子束的强度有256个等级，则显示器能同时显示256*256*256=16M种颜色，称为真彩系统

1.1 视频显示设备

1.1.5 平板显示器

- **平板显示器**代表一类比CRT能减小体积、减轻重量并节省功耗的视频设备。例如:液晶显示器等.

- 平板显示器分为:
  - 发射显示器
  - 非发射显示器
- 常见的平板显示器有:
  - 气体防电显示器
  - 薄膜光电显示器
  - 液晶显示器

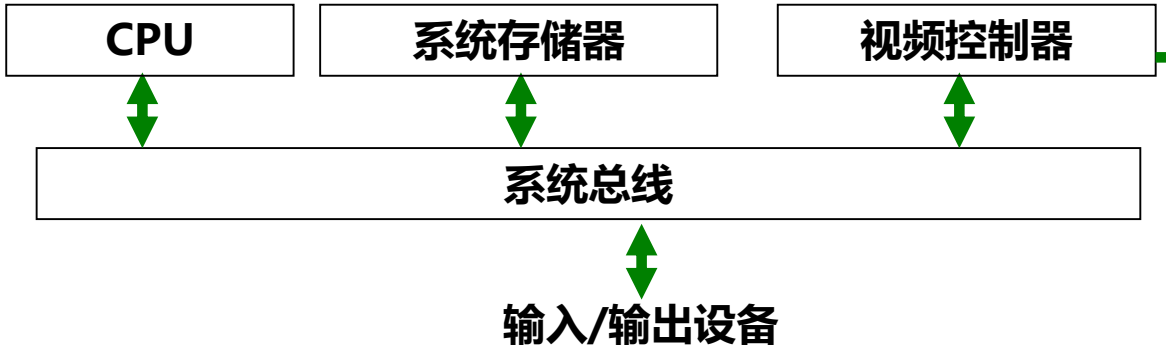
第一章 图形系统综述

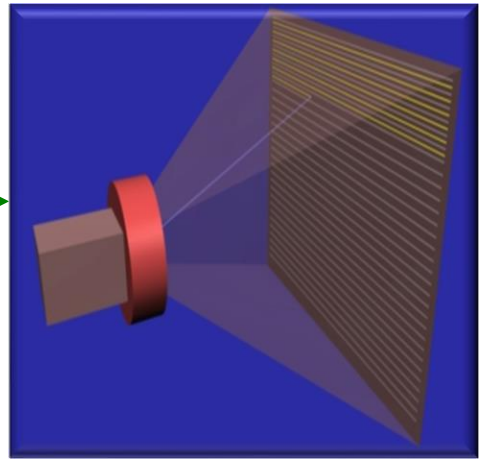
Contents

目 录

1.2 光栅扫描系统

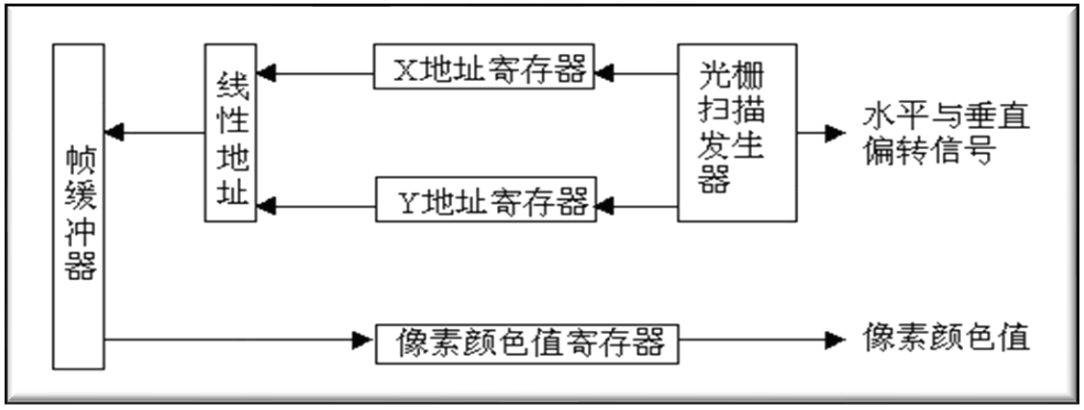
- 简单的光栅显示系统组成





显示器

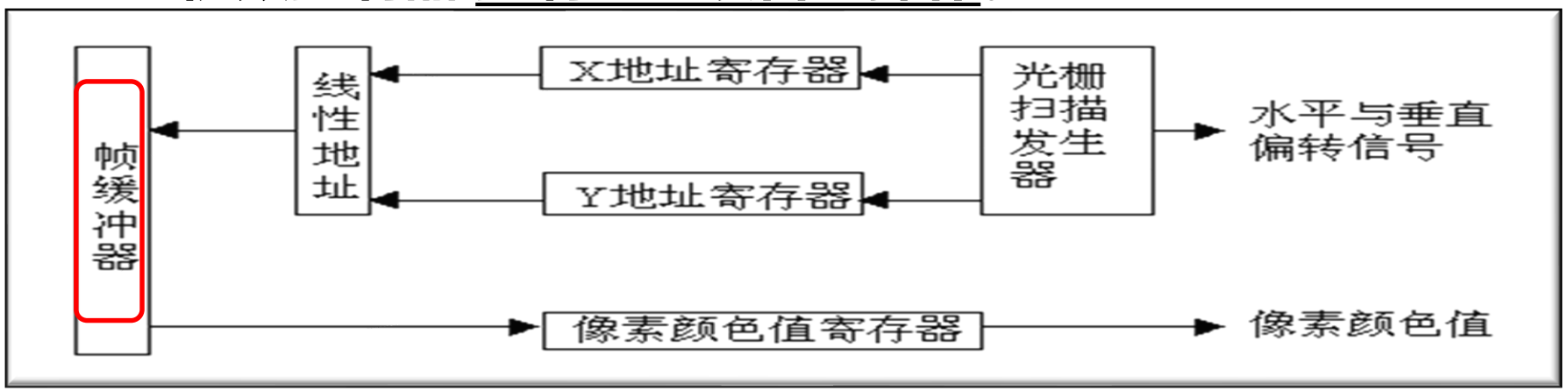
1.1.1 视频控制器:控制显示设备的操作。



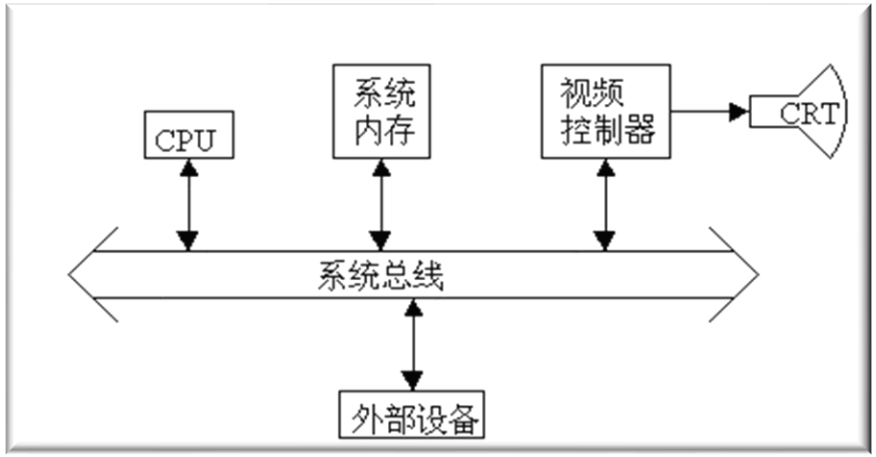
普通显卡=视频控制器+显存

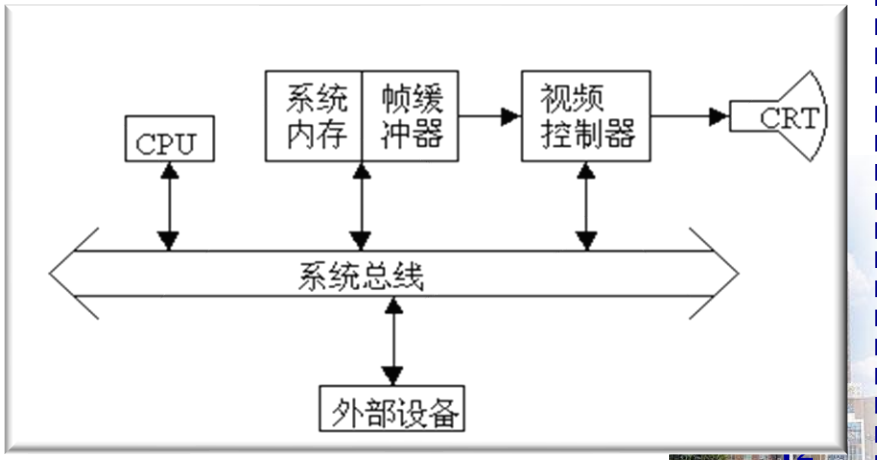
1.2 光栅扫描系统

1.1.1 视频控制器:控制显示设备的操作。



- 帧缓冲器

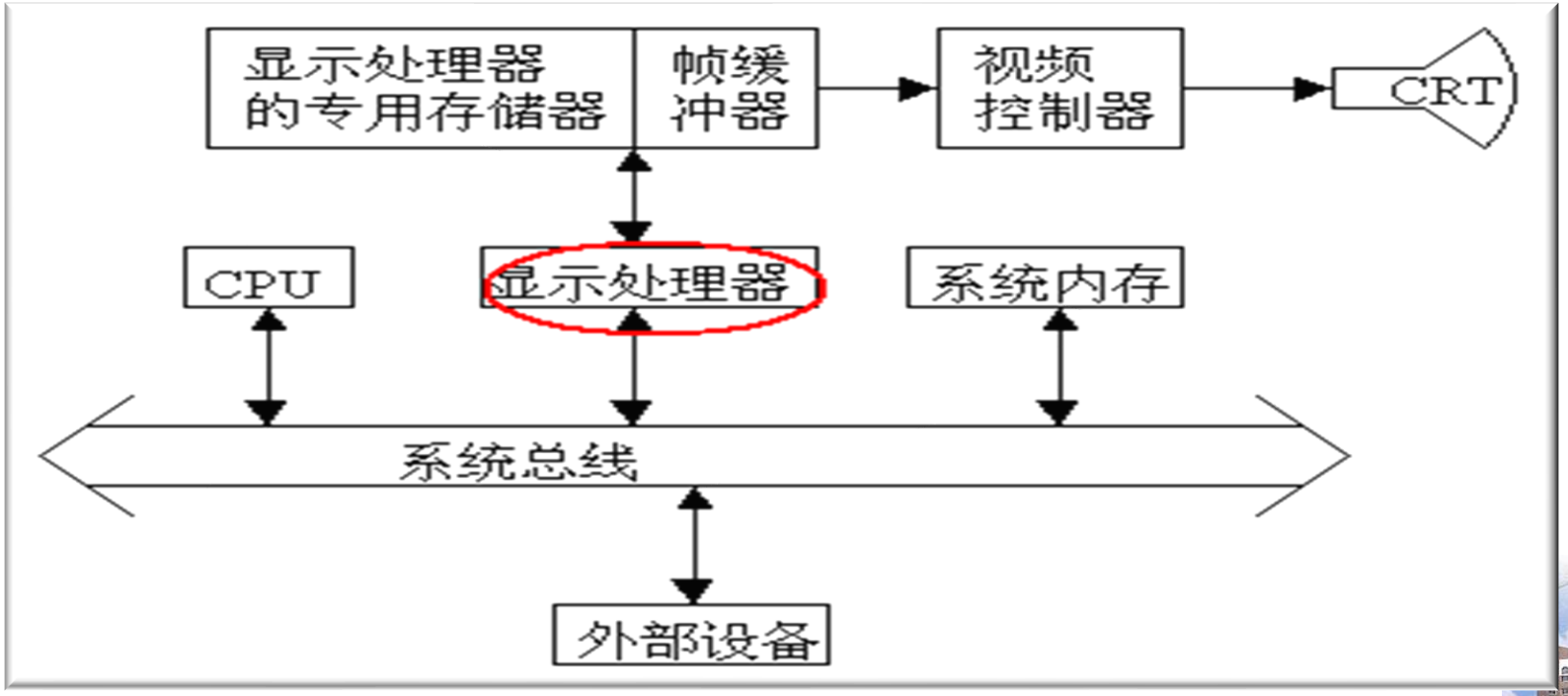




1.2 光栅扫描系统

1.1.2 光栅扫描显示器

- 显示处理器：让CPU从复杂的处理中解脱出来。



图形加速卡=视频控制器+帧缓存+显示处理器

第一章 图形系统综述

Contents

目 录

1.3 图形工作站和观察系统

- 双监视器系统

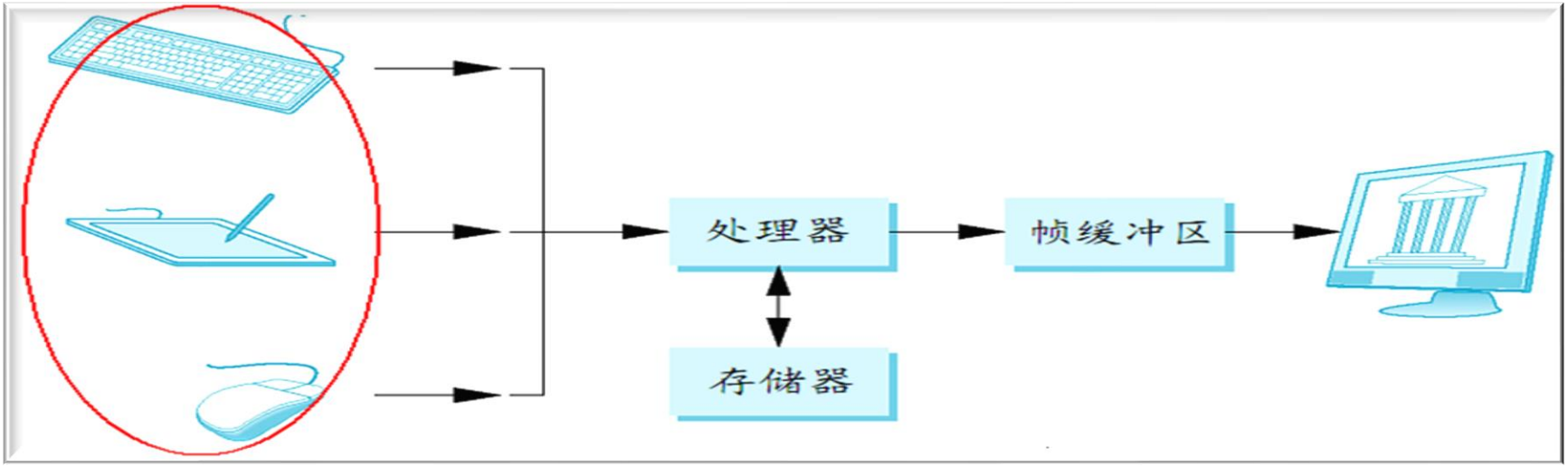
- 多屏幕系统 静态场景或动画

第一章 图形系统综述

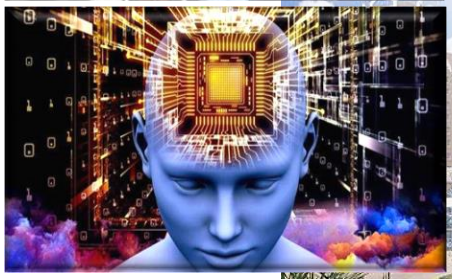
Contents

目 录

1.4 输入设备



- 二维：鼠标、坐标数字化仪、跟踪球、光笔、触摸屏、操纵杆、扫描仪……
- 三维：空间球、数据手套……
- 语言、表情、手势……
- 脑机接口



第一章 图形系统综述

Contents

目 录

1.5 硬拷贝设备

- 绘图仪
- 打印机
- ……

第一台激光打印机

第一台针式打印机

第一台复印机

打印复印一体机

第一章 图形系统综述

Contents

目 录

1.6 图形网络

多用户环境和计算机网络是目前许多图形应用的普遍特点。处理器、打印机、绘图仪和数据文件等许多资源都分布在网络中并为多个用户所共享。

组成:

图形服务器

客户

第一章 图形系统综述

Contents

目 录

第一章 图形系统综述

Contents

目 录

1.8 图形软件

Graphics Software

目 录

1.8 图形软件

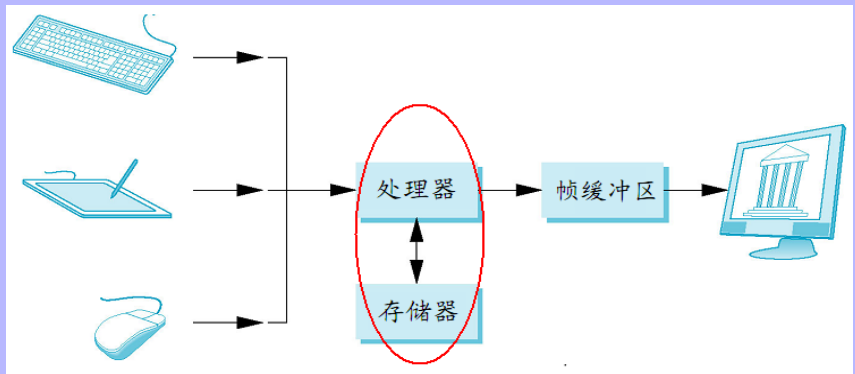
❖1.8.1 图形软件分类

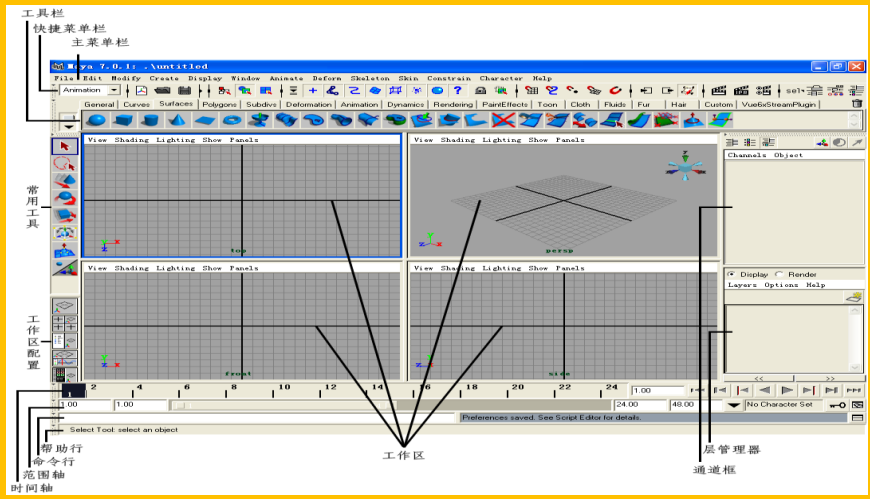
- 通用类：提供一个可用于高级程序语言的图形功能扩展集(比如OpenGL).

  基本功能：图元生成、属性设置（颜色，....）

  选择观察及实施变换等。

- 专用类：不关心图形操作过程（比如，CAD系统）





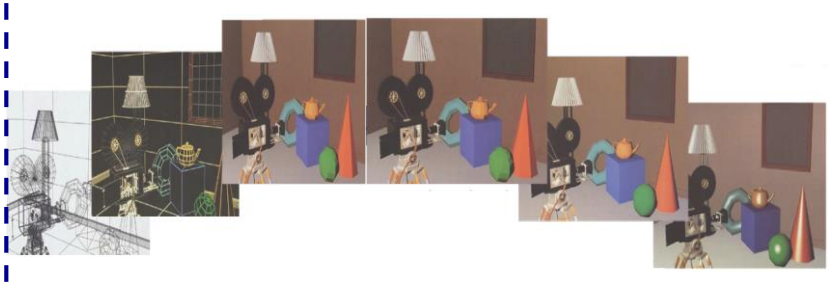
1.8 图形软件

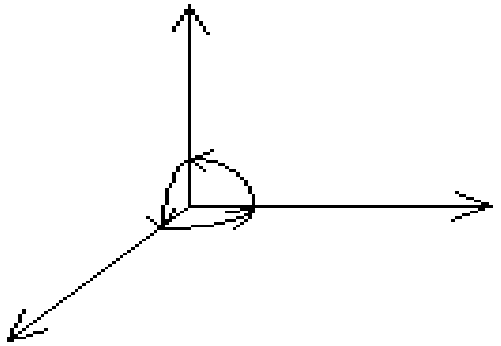
Graphics Software

目 录

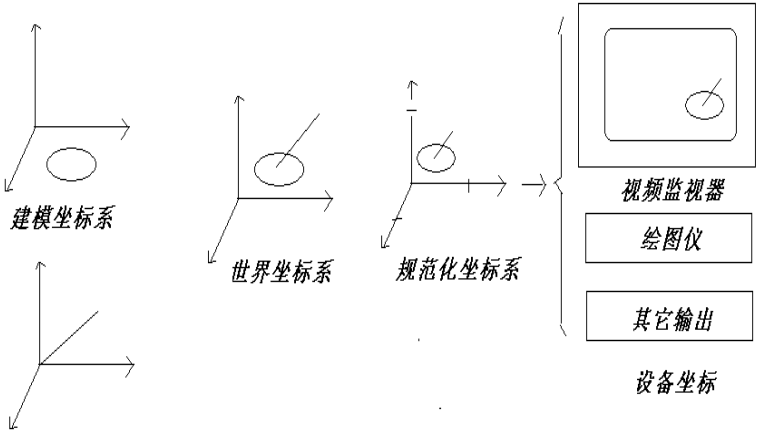
1.8 图形软件

❖1.8.2 坐标系分类





- 建模坐标系(局部坐标系或主坐标系)
- 世界坐标系
- 设备坐标系或屏幕坐标系



1.8 图形软件

Graphics Software

目 录

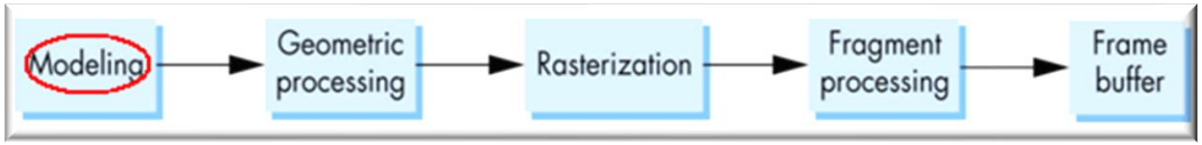
1.8 图形软件

❖1.8.3 图形系统的主要任务

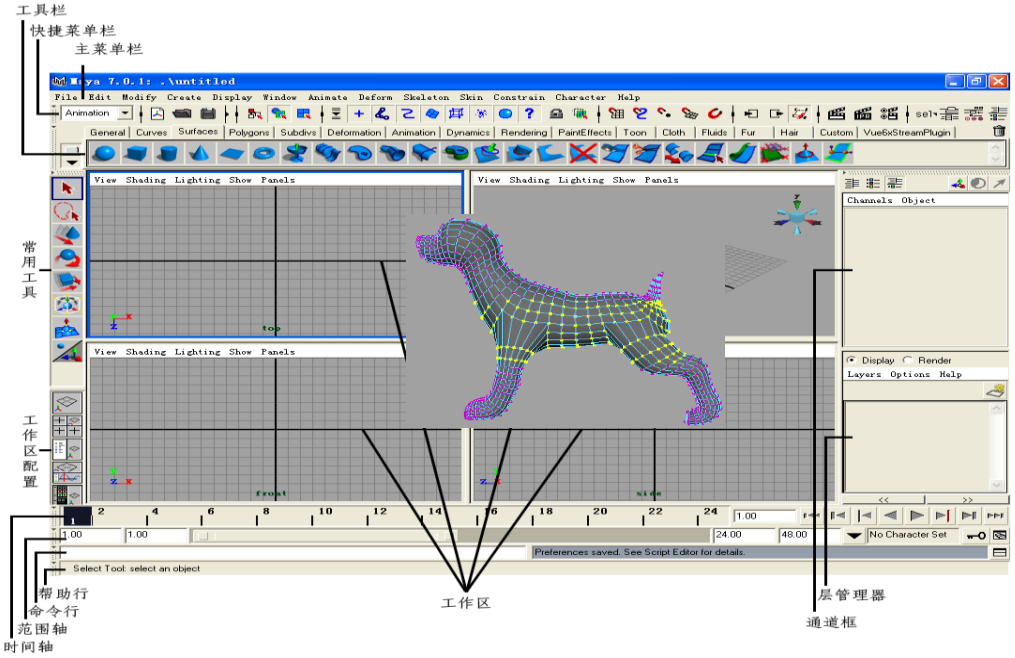
- 建模(Modeling)
- 几何处理(Geometric Processing)
- 光栅化(Rasterization)
- 片元处理(Fragment Processing)

1.8 图形软件

❖1.8.3 图形系统的主要任务

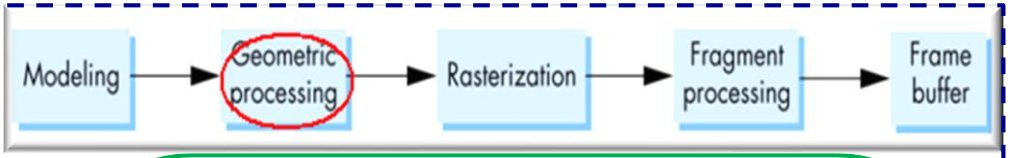


- 建模器输出几何对象的顶点数据集，它工作在对象坐标系。



1.8 图形软件

❖1.8.3 图形系统的主要任务



- 几何处理的对象是顶点，目标是确定显示在屏幕上的几何对象，并确定这些对象顶点的明暗值或颜色值投影。它包括步骤如下：
  - 投影
  - 图元装配
  - 裁剪
  - 明暗处理

投 影

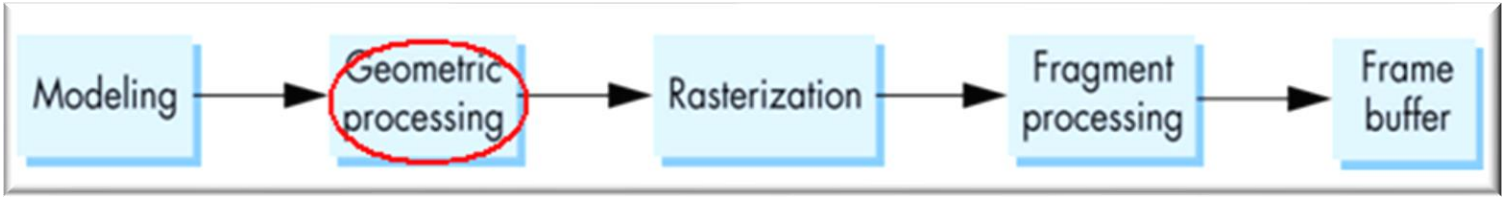
- 利用模型-视图变换把几何对象从对象坐标系变换到照相机坐标系或视点坐标系。
- 利用投影变换把顶点变换到规范化视景体内，顶点表示为裁剪坐标。

图元装配

- 变换是对顶点进行的，在进行后续操作需要组装成几何对象，称为图元装配。
- 裁剪是针对图元进行的。
- 光栅器不能对顶点单独进行处理。

1.8 图形软件

❖1.8.3 图形系统的主要任务



裁剪器

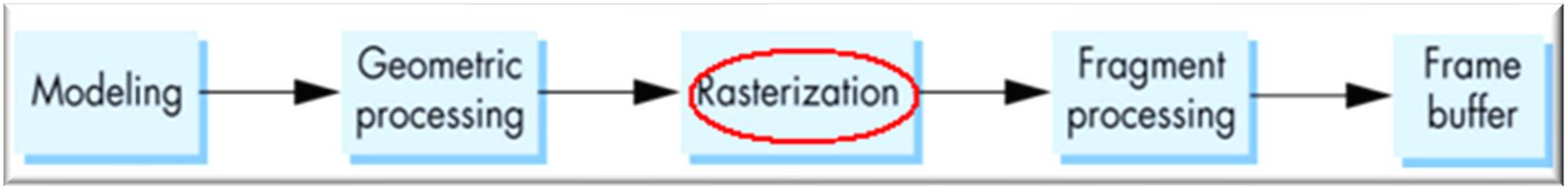
- 确定哪些图元或图元的哪些部分会被传送到光栅器，最终可能会显示在屏幕上只有在视景体内的对象。
- 部分在视景体内的图元，裁剪后生成新的图元。
- 没被裁剪掉的顶点仍表示为四维齐次坐标，通过透视除法转换为三维的规范化设备坐标。

明暗处理

- 为每个顶点赋颜色值，由当前绘制颜色决定。
- 开启光照时，根据改进的Phong模型计算得到。

1.8 图形软件

❖1.8.3 图形系统的主要任务

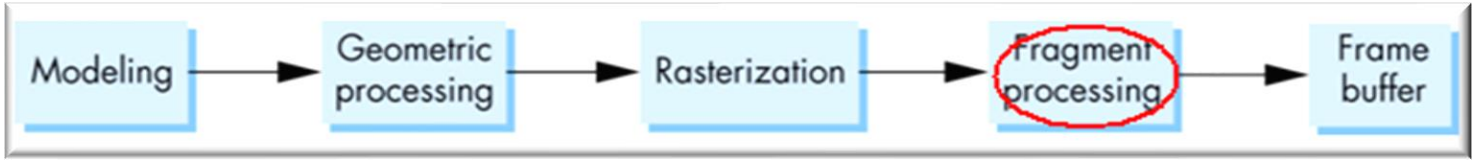


光栅化或扫描转换

- 从裁剪后的对象生成片段（准像素）。对于线段，确定哪些像素可用来近似表示顶点间的线段。
- 对于多边形，确定哪些像素位于多边形顶点定义的二维区域的内部
- 片段的颜色取决于系统颜色状态值或由光照模型计算顶点明暗值插值得到。

1.8 图形软件

❖1.8.3 图形系统的主要任务



片元处理

- 最简单的情形下，片段的颜色由光栅器赋值，并且该值就是片元所对应的帧缓冲区中像素的颜色值。
- 纹理贴图、融合、半透明效果，反走样。
- 隐藏面消除：确定可见对象的片元值可见对象指位于视景体内，且没有被其他更靠近照相机的不透明对象所遮挡的对象。

1.8 图形软件

Graphics Software

目 录

1.8 图形软件

1.8.4 图形软件提供的功能

软 件

- 输 入
- 输 出
- 属 性
- 变 换
- 观 察
- 控 制

图形标准

- 数据及文件格式标准
- 子程序界面标准

软件标准

- 面向图形设备的接口标准
- 面向应用软件的标准
- 面向图形应用系统数据模型及其文件格式

第一章 图形系统综述

Contents

目 录

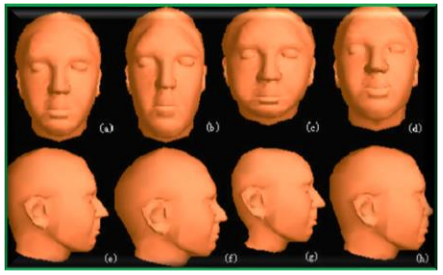
1.9 OpenGL简介

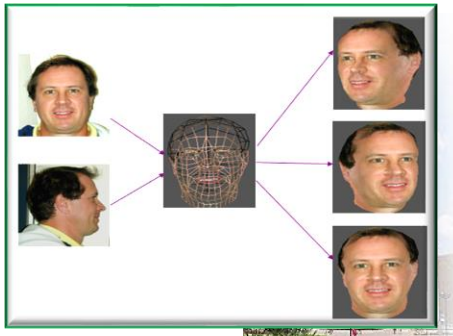
- **GL-图形程序库**
  - UNIX下运行
  - OpenGL—GL的微机版
  - 分类：基本图素；坐标变换；设置属性和显示方式；I/O 处理；真实图形显示。
- GL:Silicon Graphics(SGI) Geometry Engine（几何引擎）,硬件(VLSI)实现几何流水线，极大改良了图形工作站
  - 立即模式绘制
  - 可非常简单地设计出三维交互图形　应用程序

- OpenGL：SGI领导的OpenGL Architectural Review Board(OpenGL ARB)发布1.0版平台无关的API：
  - 易于使用
  - 与硬件非常贴近，从而可以充分发挥其性能
  - 着重在于渲染(rendering)
  - 没有提供窗口和输入接口，从而避免依赖于具体的窗口系统

1.9 OpenGL简介

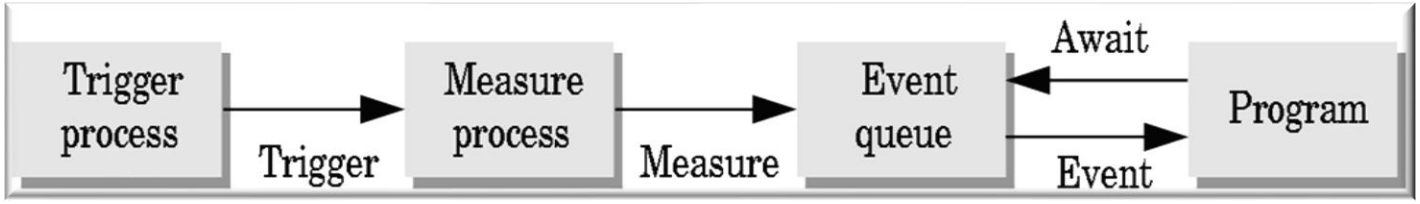
- OpenGL（Open Graphics Library）是个定义了一个跨编程语言、跨平台的编程接口的规格，是个专业的3D程序接口，是一个功能强大，调用方便的底层3D图形库
- OpenGL是个与硬件无关的软件接口，可以在不同的平台之间进行移植
- 这个接口由近二百五十个不同的函数调用组成，用来从简单的图元绘制复杂的三维景象。
- OpenGL常用于CAD、虚拟实境、电子游戏开发等。
- OpenGL是一个状态机(state machine),状态包括：持续性参数，颜色、线型、材质属性等。
  - OpenGL函数有两种类型生成图元（图元函数，如glVertex）如果图元可见，则被输出。
  - 图元的外观由状态控制。



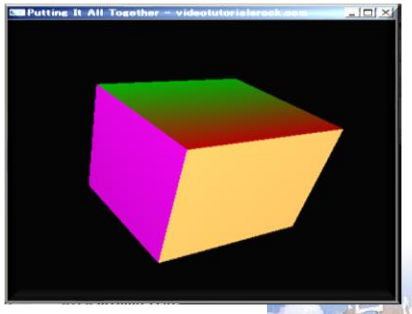


1.9 OpenGL简介

- 事件模式(Event Mode)：绝大部分系统具有多个输入设备，每个设备都可能被用户在任意时间触发。
- 每个触发生成一个事件，事件的测量值放到事件队列中，用户程序检查该队列，根据事件的类型采用相应的操作（回调函数）

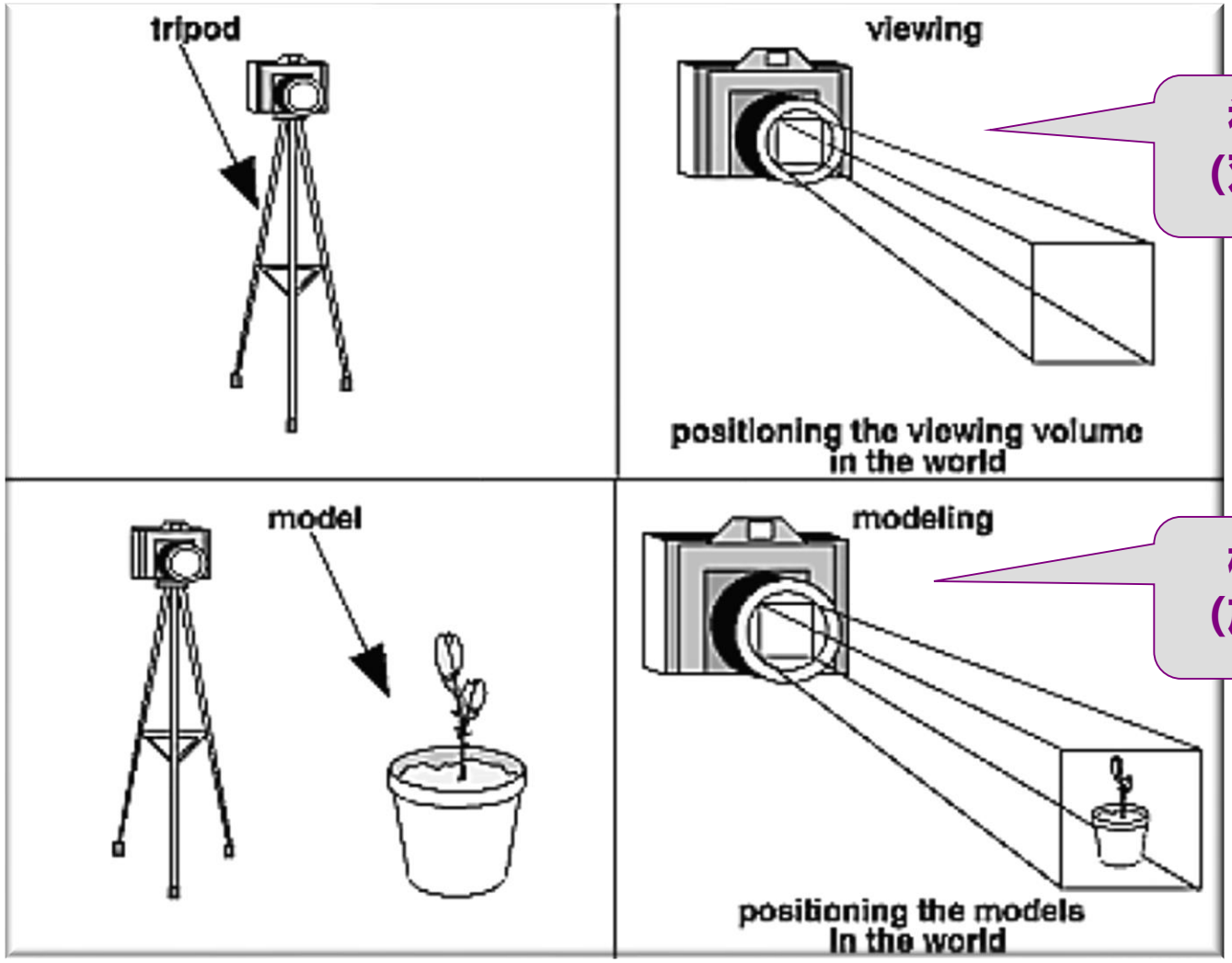


- 事件类型
  - 鼠标：点击一个或多个按钮，移动
  - 窗口：改变尺寸、重新显示、缩成图标
  - 键盘：按下或释放某个键



- 事件循环：在程序中定义一个显示回调函数，(display callback)，每个GLUT程序都必须有一个显示回调函数。
- 只要OpenGL确定显示内容要被刷新时，显示回调函数就会被调用
- main函数是以程序进入事件循环做为结束。

1.9 OpenGL简介



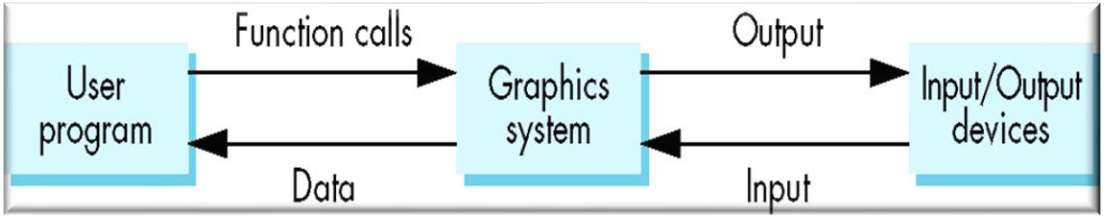
视点变换
（对准场景）

相反
过程

模型变换
（放置模型）

1.9 OpenGL简介

- 7类API的函数：
  - 图元函数
    - ✓ 线段
    - ✓ 多边形
  - 属性函数
  - 视图函数（或观察函数）
  - 变换函数
  - 控制函数
  - 查询函数
  - 输入及窗口函数（GLUT）
- 基本库的函数名以gl为前缀 glBegin,glClear,glCopyPixels,glPolygonMode
- 常量以GL开头 GL_2D,GL_RGB,GL_CCW,GL_POLYGON,GL_AMBIENT_AND_DIFFUSE
- 库中定义的专门的数据类型以GL开头 GLbyte,GLshort,Glint,GLfloat,GLdouble

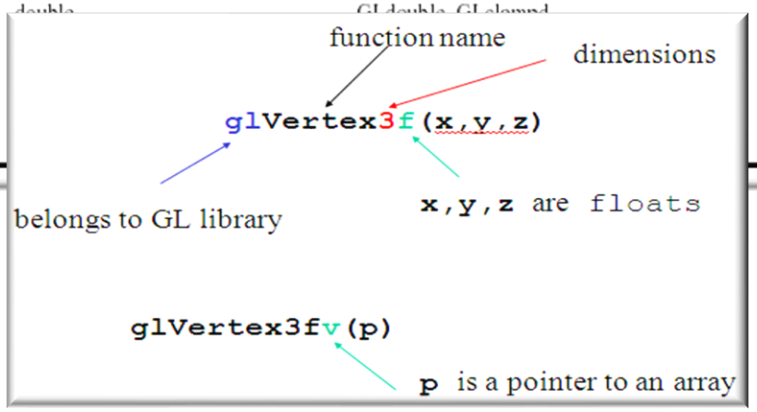


OpenGL基本语法

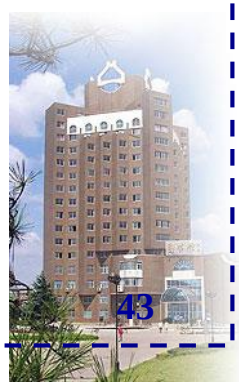
- 核心库(GL)：基本库。
- 实用函数库(GLU)：处理专门操作的附加库。
- 实用函数工具包(GLUT)：实用函数工具包。

表 3-1

字 符	C语言类型	OpenGL类型定义
b	signed char	GLbyte
s	short	GLshort
i	int	GLint
f	float	GLfloat, GLclampf
d	double	GLdouble, GLclampd
ub	unsigned char	GLubyte
us	unsigned short	GLushort
ui	unsigned int	GLuint



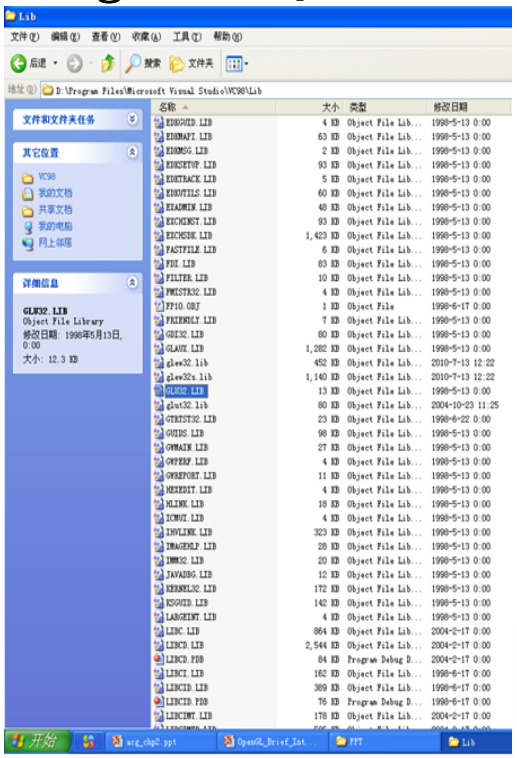

```
void glVertex3s(GLshort  x, GLshort y, GLshort  z);
void glVertex3i(GLint x, GLint y, GLint z);
void glVertex3f(GLfloat  x, GLfloat  y, GLfloat  z);
void glVertex3d(GLdouble  x, GLdouble  y, GLdouble  z);
```



OpenGL使用

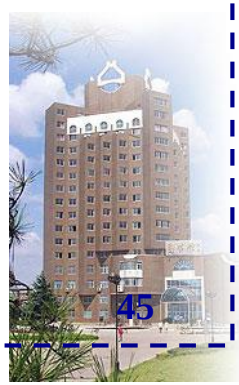
- 基于OpenGL标准开发的应用程序必须运行于32位Windows平台下，如WindowsNT或Windows95环境；而且运行时还需有动态链接库OpenGL31.DLL、Glu31.DLL，一般在…\WINDOWS\system32里面（注：window2000以上系统均 带有OPENGL31.DLL和glu31.dll)
- 一般来说，VC6和VS.NET里面就带有GL的基本库
  - VC6：…\Microsoft Visual Studio\VC98\Include\GL
  - VS.NET.2005：…\Microsoft Visual Studio8\VC\PlatformSDK\Include\gl
  - 一般是Gl.h，Glaux.h，Glu.h
  - Lib里也有：OpenGL31.lib，GLu31.lib，GLaux.lib
- 所以开发者在VC下可以使用
  - #include <gl\gl.h>　　// Header File For The OpenGL32 Library
  - #include <gl\glu.h>　　// Header File For The GLu32 Library

  #include <gl\glaux.h>　// Header File For The Glaux Library在没使用GLUT的情况下手动添加link）：
- 在你文件头加上
  - ✓ #include <gl\gl.h>
  - ✓ #include <gl\glu.h>
  - ✓ #include <gl\glaux.h>
  - 进入Project菜单，选Settings项，弹出 Settings 对话框，选Link项，在 Libraries 栏目中加入OpenGL库：opengl31.lib glu31.lib glaux.lib



OpenGL使用

- GLUT (OpenGL Utility Toolkit)
- 安装：在windows下通过 C/C++语言编写 GLUT 程序, 需要以下三个文件:
  - GLUT.H - 需要源代码中包含这个文件。通常情况下，这个文件应该放在系统的包含目录下的 GL 文件夹中。
  - GLUT.LIB (SGI windows版本) 以及 glut31.lib (微软版本) - 这个文件必须被连接到程序中, 确保它放在 LIB 目录中。
  - glut31.dll (Windows) 和 glut.dll (SGI Windows版本) - 根据所使用的OpenGL选择一个，如果正在使用微软公司的版本，那么必须选择glut31.dll。应该把DLL放置在系统文件夹中。
- 举例：在window XP下使用VC6安装GLUT
  - 下载glutdlls并解压
  - 复制glut31.dll和glut.dll到…\windows\system32
  - 复制glut.h到...\Microsoft Visual Studio\VC98\Include\GL
  - 复制glut31.lib和glut.lib到…\Microsoft Visual Studio\VC98\Lib



OpenGL使用

```
#include <GL/glut.h>
void init(void)
{
  glClearColor(1.0,1.0,1.0,0.0);
 glMatrixMode(GL_PROJECTION);
  gluOrtho2D(0.0,200.0,0.0,150.0)
}
```

```
void lineSegment(void)
{
     glClear(GL_COLOR_BUFFER_BI
     T);
  glColor3f(1.0,0.0,0.0);
  glBegin(GL_LINES);
    glVertex2i(180,15);
    glVertex2i(10,145);
  glEnd();  glFlush();}
```

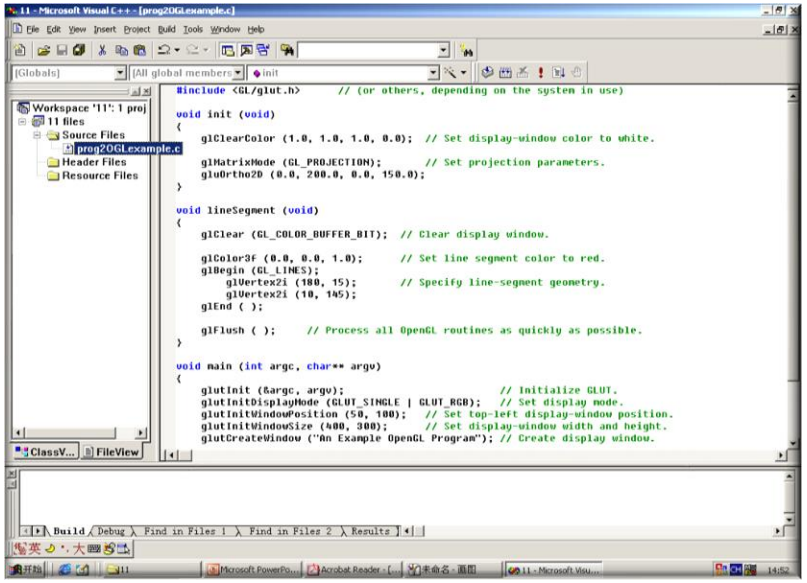
```
void main(int argc,char** argv)
{ glutInit(&argc,argv);

     glutInitDisplayMode(GLUT_SINGLE
     |GLUT_RGB);
  glutInitWindowPosition(50,100);
  glutInitWindowSize(400,300);
  glutCreateWindow("An Example
     OpenGL Program.");
  init();
  glutDisplayFunc(lineSegment);
  glutMainLoop();}
```

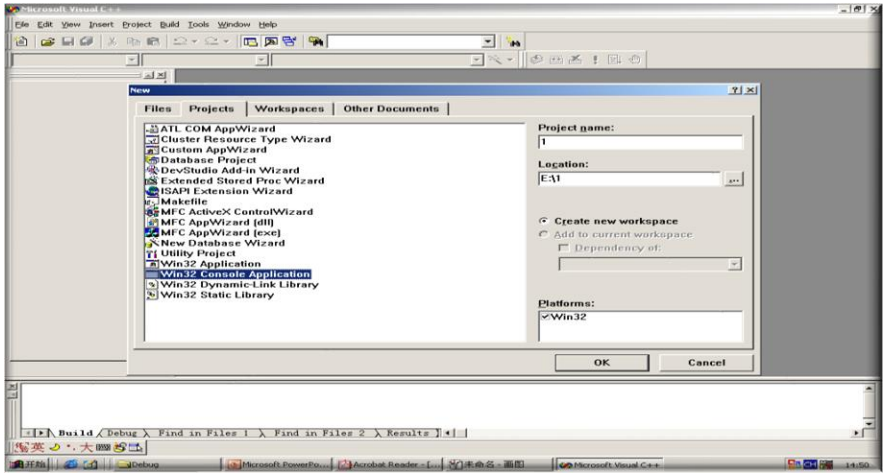
OpenGL使用

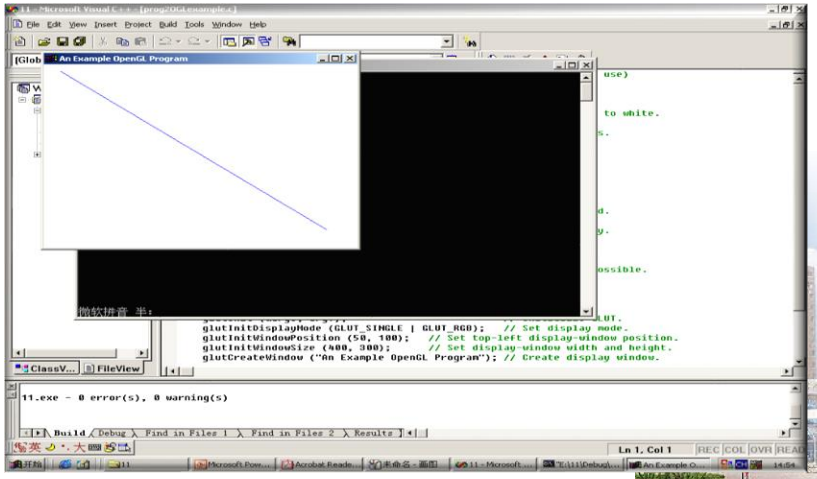
- glutDisplayFunc(lineSegment);

显示回调函数是程序员自己编写的关于显示窗口内容的函数，由glutDisplayFunc 作为显示窗口需要重新显示时引入的函数来注册

- 当一个窗口的图像层需要重新绘制时，GLUT将调用该窗口的的显示回调函数。

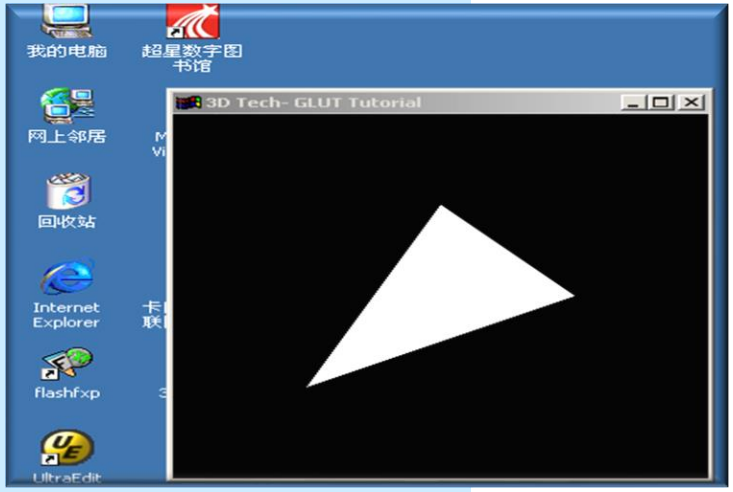






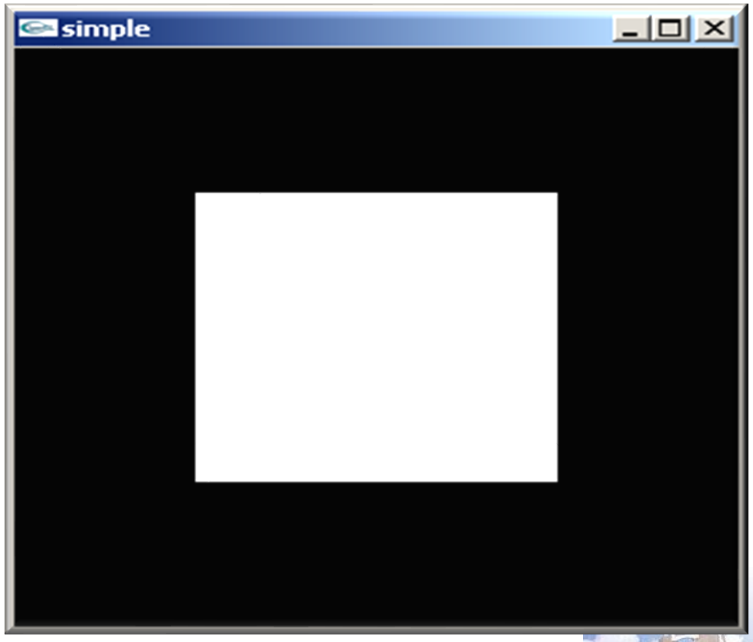
OpenGL使用

```
#include <GL/glut.h>
void renderScene(void) { //绘制一个简单的二维的三角形
      glClear(GL_COLOR_BUFFER_BIT);
      glBegin(GL_TRIANGLES);
      glVertex3f(-0.5,-0.5,0.0);
      glVertex3f(0.5,0.0,0.0);
      glVertex3f(0.0,0.5,0.0);
      glEnd();
      glFlush();
}
void main(int argc, char **argv) {
      glutInit(&argc, argv); //初始化glut
      glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
            //设置窗口的模式—深度缓存，单缓存，颜色模型
      glutInitWindowPosition(100,100); //设置窗口的位置
      glutInitWindowSize(320,320); //设置窗口的大小
      glutCreateWindow("3D Tech- GLUT Tutorial"); //创建窗口并赋予title
      glutDisplayFunc(renderScene);//调用renderScene把绘制传送到窗口
    glutMainLoop(); //进入循环等待
}
```



OpenGL使用

```
#include <GL/glut.h>
void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT);
        glBegin(GL_POLYGON);
            glVertex2f(-0.5, -0.5);
            glVertex2f(-0.5, 0.5);
            glVertex2f(0.5, 0.5);
            glVertex2f(0.5, -0.5);
        glEnd();
        glFlush();
}
int main(int argc, char** argv){
        glutCreateWindow("simple");
        glutDisplayFunc(mydisplay);
        glutMainLoop();}
```



OpenGL使用

- 通过定义 void glutReshapeFunc(void (*func)(int width, int height))来避免因窗口大小改变时图形的变形;
- 通过定义 void glutIdleFunc(void (*func)(void)); 使应用空闲时反复调用函数func
- 对于window的基本应用：mouse，keyboard，menu，sub window，font等glut里也提供了支持。

```
#include<GL/glut.h>
float angle = 0.0;
void changeSize(int w, int h) {
        if(h == 0)h = 1;
        float ratio = 1.0* w / h;
        glMatrixMode(GL_PROJECTIO
        N);
        glLoadIdentity();
        glViewport(0, 0, w, h);
        gluPerspective(45,ratio,1,1000);
        glMatrixMode(GL_MODELVIE
        W);
        glLoadIdentity();
        gluLookAt(0.0,0.0,5.0, 0.0,0.0,-1.0,
                0.0f,1.0f,0.0f);
}
```

```
void renderScene(void) {
        glClear(GL_COLOR_BUF
        FER_BIT |
        GL_DEPTH_BUFFER_BI
        T);
        glPushMatrix();
        glRotatef(angle,0.0,1.0,0.0);
        glBegin(GL_TRIANGLES);
        glVertex3f(-0.5,-0.5,0.0);
        glVertex3f(0.5,0.0,0.0);
        glVertex3f(0.0,0.5,0.0);
        glEnd();
        glPopMatrix();
        angle++;
        glutSwapBuffers();}
```

```
void main(int argc, char **argv) {
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DEPT
        H |             GLUT_DOUBLE |
        GLUT_RGBA);
        //设为双缓冲区,
         //平滑动画需要
        glutInitWindowPosition(100,100);
        glutInitWindowSize(320,320);
        glutCreateWindow("Lighthouse 3D
        - GLUT  Tutorial");
        glutDisplayFunc(renderScene);
        glutIdleFunc(renderScene);
       //register a callback function to be
        called when the application is idle
        glutReshapeFunc(changeSize);
        //Preparing the window for a
        reshape
        glutMainLoop();}
```

OpenGL参 考

- web：
  - OpenGL 官方主页：http://www.opengl.org
  - OpenGL 1.1 Reference：http://www.opengl.org/sdk/docs/man
  - GLUT：http://www.opengl.org/resources/libraries/glut/
  - NEHE的在线GL教程：http://nehe.gamedev.net
- specification：
  - The OpenGL Utility Toolkit (GLUT) Programming Interface (PDF)
  - OpenGL 1.1 specification (PDF)
- book：
  - OpenGL超级宝典（三版）（中文），人民邮电出版社
  - OpenGL编程指南（四版）（中文），人民邮电出版社
  - OpenGL Shading Language

总 结

- 1.1 视频显示设备
- 1.2 光栅扫描系统
- 1.3 图形工作站和观察系统
- 1.4 输入设备
- 1.5 硬拷贝设备
- 1.6 图形网络
- 1.7 因特网上的图形
- 1.8 图形软件
- 1.9 OpenGL简介

Contents

目 录

谢谢

欢迎交流！