计算机图形学（实验一）

课前须知（1）

- TA：高博文
  - Tel：15366539198
  - Email：bw_gao@zju.edu.cn

- 7个小作业+大作业（在**学在浙大**上提交**代码**+**实验报告pdf**，作业提交时间限定为布置后的两周内）
- 推荐平台（可选）：Windows，Microsoft Visual Studio 2017

课前须知（2）

1. 提交可运行的代码及完整的实验报告。可以录制运行视频，但仅作为辅助判定手段；
2. 上交作业请按照“学号-姓名-实验x”的文件夹/压缩包形式上传，实验报告命名同样为“学号-姓名-实验x-实验报告”，报告模版已另附；
3. 禁止抄袭代码。一经发现，严肃处理；
4. 迟交将酌情扣分；
5. 实验课按照课表在**线下展开**。课上将进行讲课，剩余时间用于编程，建议携带做作业用的电脑。

实验课内容

- 熟悉了解OpenGL的概念与配置
- 了解并逐步掌握渲染管线的基本知识
- 练习OpenGL的使用
- …

OpenGL(Open Graphics Library)

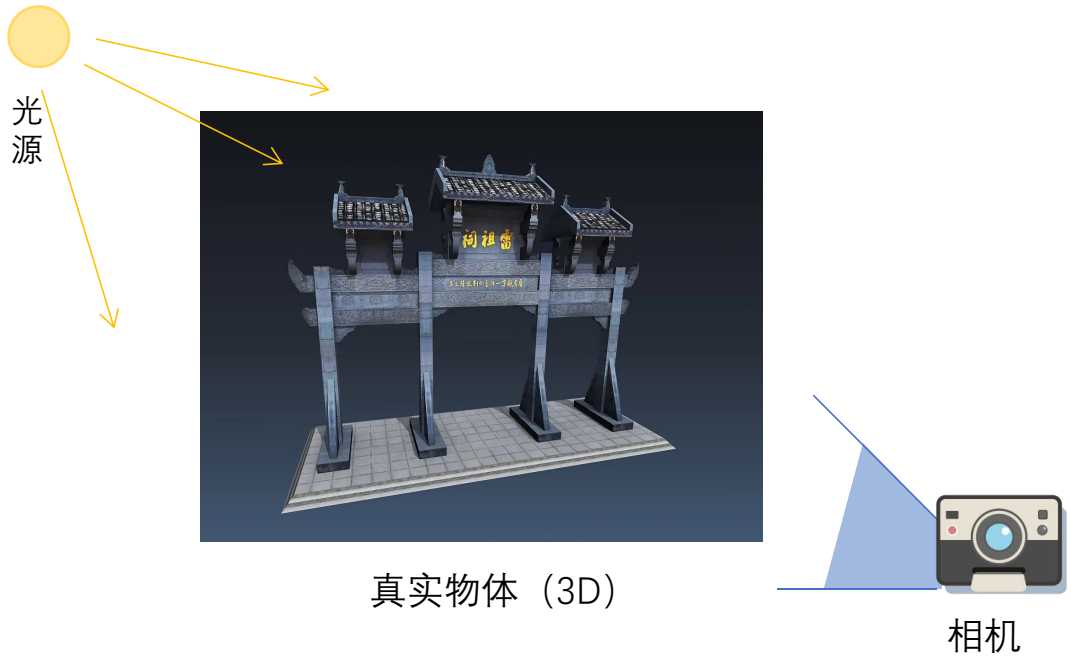
- 应用广泛的跨语言跨平台的图形接口
- 通过封装与显卡硬件的交互来实现跨平台的图形系统
- 1992年发布1.0，目前已更新至4.6

- 类似的还有DirectX、Vulkan、Metal等图形接口

https://www.opengl.org/

渲染管线

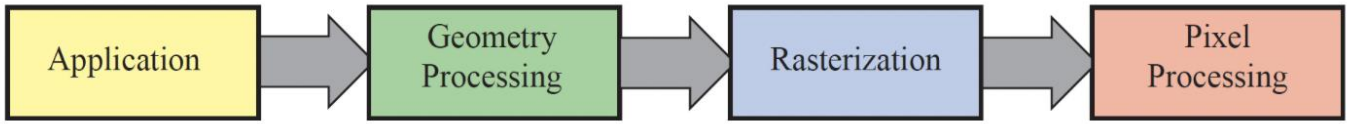
指显卡渲染图形的整个流程



工作内容（通俗理解，并不严谨）：

通过一系列计算，得到3D空间中的模型在2D“相片”上的样子，包括位置、颜色等等（如左图所示）这个过程中必须有光源、模型以及相机的存在。由于这是一个及其复杂的过程，于是会被分成很多小步骤。

下图表明的是一个粗粒度的渲染流程，仅为了便于理解以及划分大致流程。随着不断的迭代，渲染管线也发生了巨大的变化，并且在细节上有多种多样的划分方式，但其基本的流程是不变的。我们只需要把握住每一步的输入和输出即可，不需要纠结于每一步属于哪个阶段。



应用阶段：接收任何应用层面的信息（比如键盘交互、动画运动等），并输出最终得到的图元（如点、线、三角形）。此时不在GPU上执行，而是在CPU上计算。

几何处理：此时进入GPU进行处理。获得上一步输出的图元，对图元当中的信息（比如顶点位置、颜色等）进行处理（如调整、修改、增删、转换成投影坐标等）。输出处理后的图元。

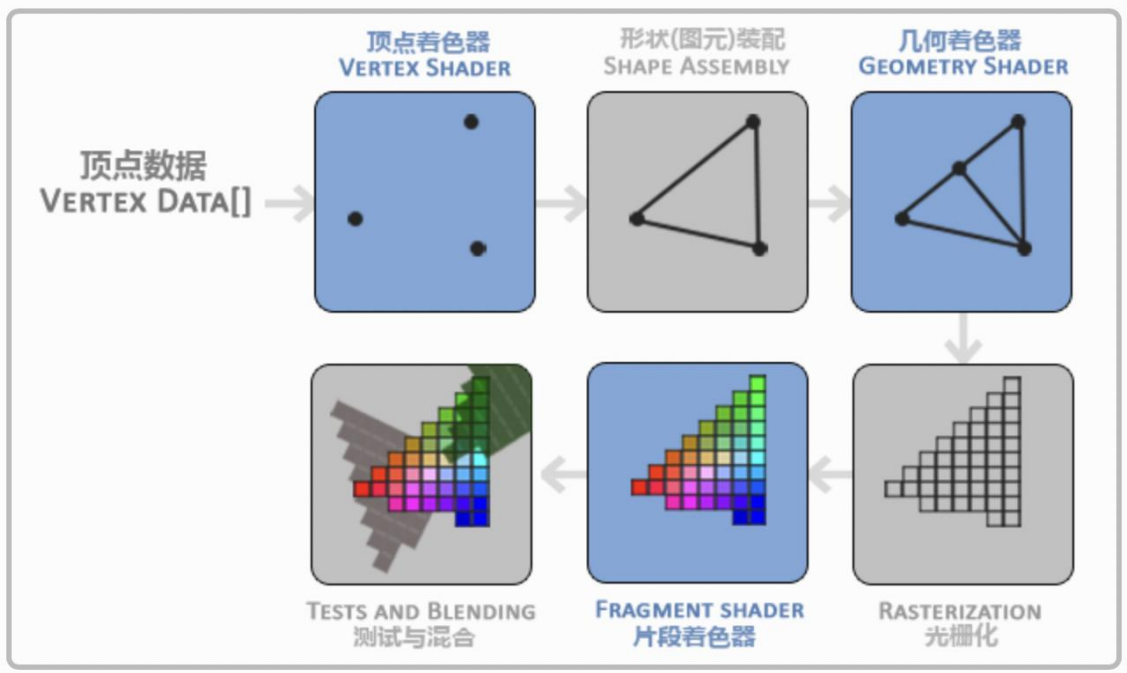
光栅化：获得上一步输出的图元，根据图元中包含的信息，将每个图元转换成屏幕像素的集合。输出像素。

像素处理：获得上一步输出的像素，根据像素中包含的信息，计算出该像素上该图元的颜色等信息，再通过一系列计算，获得最终融合的颜色。至此，所有的信息处理完毕，可以显示到屏幕上了。

渲染管线

指显卡渲染图形的整个流程

- 固定管线 OpenGL1.0-1.5
  - 效果简单，资源消耗低
  - 只需指定光源、物体模型坐标等即可实现渲染
  - 无法实现自定义的复杂效果
- 可编程管线 OpenGL2.0-4.6
  - 可以根据需要得到各种不同的效果
  - 需要使用着色器语言GLSL进行编程
  - GLSL版本之间不完美兼容



一种可编程的渲染管线的可视流程图（来自LearningOpenGL教程）其中蓝色的模块就是可编程的。
可着重观看里面的图像表示，即从顶点，到图元，到像素，到最终显示的变化，从而获得一个整体印象。

渲染管线

指显卡渲染图形的整个流程

- 固定管线 OpenGL1.0-1.5
  - 效果简单，资源消耗低
  - 只需指定光源、物体模型坐标等即可实现渲染
  - 无法实现自定义的复杂效果
- 可编程管线 OpenGL2.0-4.6
  - 可以根据需要得到各种不同的效果
  - 需要使用着色器语言GLSL进行编程
  - GLSL版本之间不完美兼容

就像一个完整的机器，程序员可以自由的操作机器上预留给自己的开关，通过“开”或“关”的组合，调用预置好的工作模型，达到自己预期的效果。
这个时候一般来说就是直接调用API，并对API里的参数进行更改。

渲染管线

指显卡渲染图形的整个流程

- 固定管线 OpenGL1.0-1.5
  - 效果简单，资源消耗低
  - 只需指定光源、物体模型坐标等即可实现渲染
  - 无法实现自定义的复杂效果
- 可编程管线 OpenGL2.0-4.6
  - 可以根据需要得到各种不同的效果
  - 需要使用着色器语言GLSL进行编程
  - GLSL版本之间不完美兼容

程序员获得了更多自由。不仅可以控制一些封装较好的开关接口，而且还能对指定的流程设计自己想要的开关和工作模型，即使用GLSL语言进行编程。
此时相当于程序员可以创建属于自己的API，自己决定要对顶点坐标、颜色等信息做何种操作，

渲染管线

指显卡渲染图形的整个流程

- 固定管线 OpenGL1.0-1.5
  - 效果简单，资源消耗低
  - 只需指定光源、物体模型坐标等即可实现渲染
  - 无法实现自定义的复杂效果
- 可编程管线 OpenGL2.0-4.6
  - 可以根据需要得到各种不同的效果
  - 需要使用着色器语言GLSL进行编程
  - GLSL版本之间不完美兼容

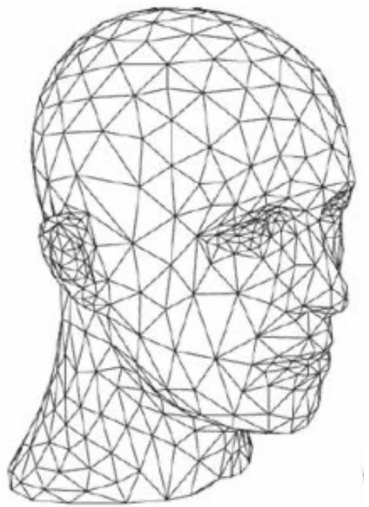
我们首先从简单的固定管线开始实践，学习OpenGL!

渲染管线中坐标转换的流程

- 当CPU将当前需要渲染的所有信息传入GPU后，GPU一定要做的事就是对图元信息进行坐标转换。
- 目标：将物体坐标投影到像平面上（但还不是最终显示！最终显示需要把像平面上的图像转换成屏幕图像）

- 局部坐标系
  转换到
- 世界坐标系
  转换到
- 相机坐标系
  投影到
- 像平面坐标系（二维，但仍是连续的）

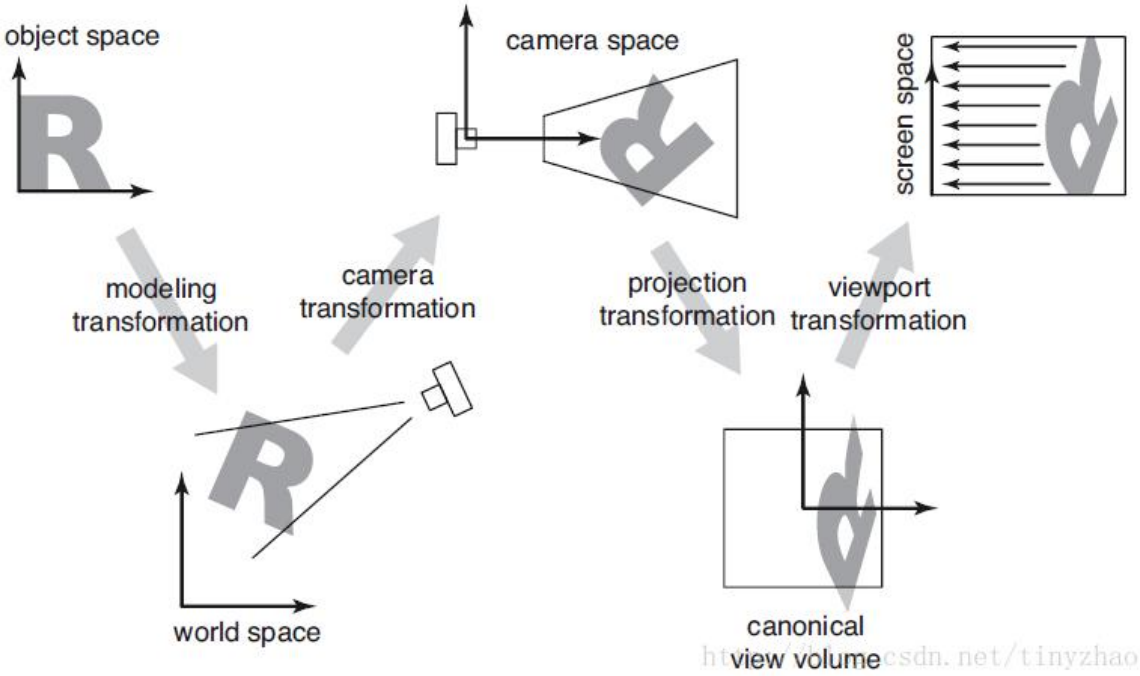
基本知识介绍：
这里的图元可以是点、线、三角形甚至是四边形。一般来说复杂模型都可划分成一群三角形图元的集合（如下图人像）。
构成图元的基本类就是顶点(Vertex)。点 = 1个Vertex; 线 = 2个Vertex; 三角形 = 3个Vertex。Vertex类中储存了位置信息、颜色信息、纹理贴图对应的uv位置信息等等。



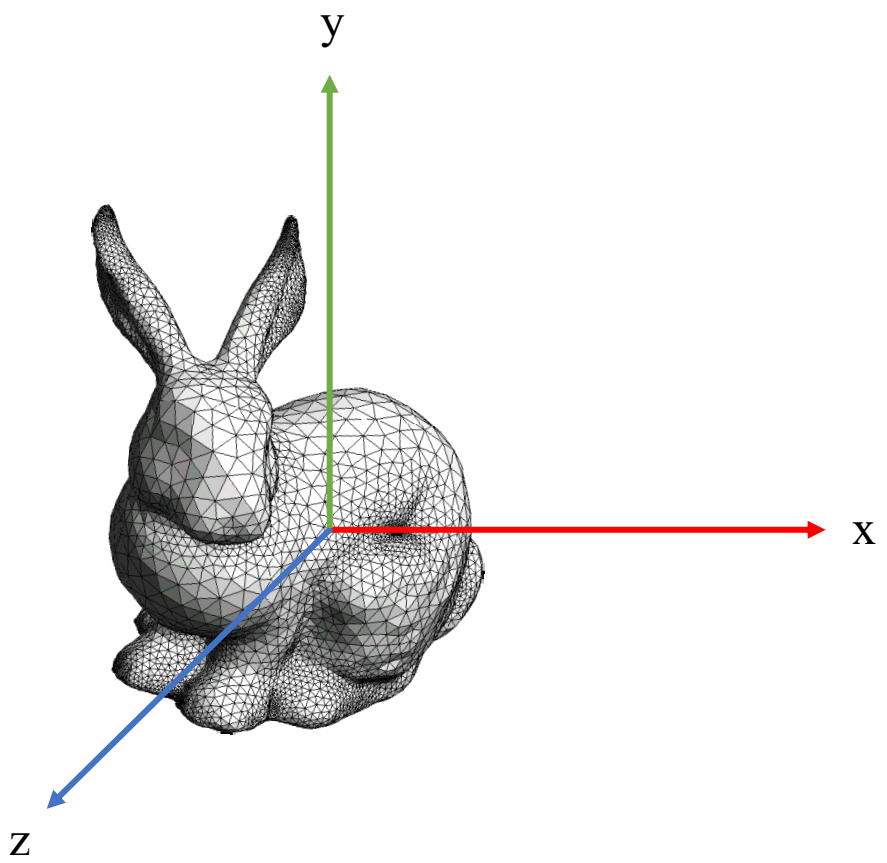
渲染管线中坐标转换的流程

- 目标：将物体的坐标投影到像平面上

- 局部坐标系/模型坐标系
  转换到
- 世界坐标系
  转换到
- 相机坐标系
  投影到
- 像平面坐标系（二维）

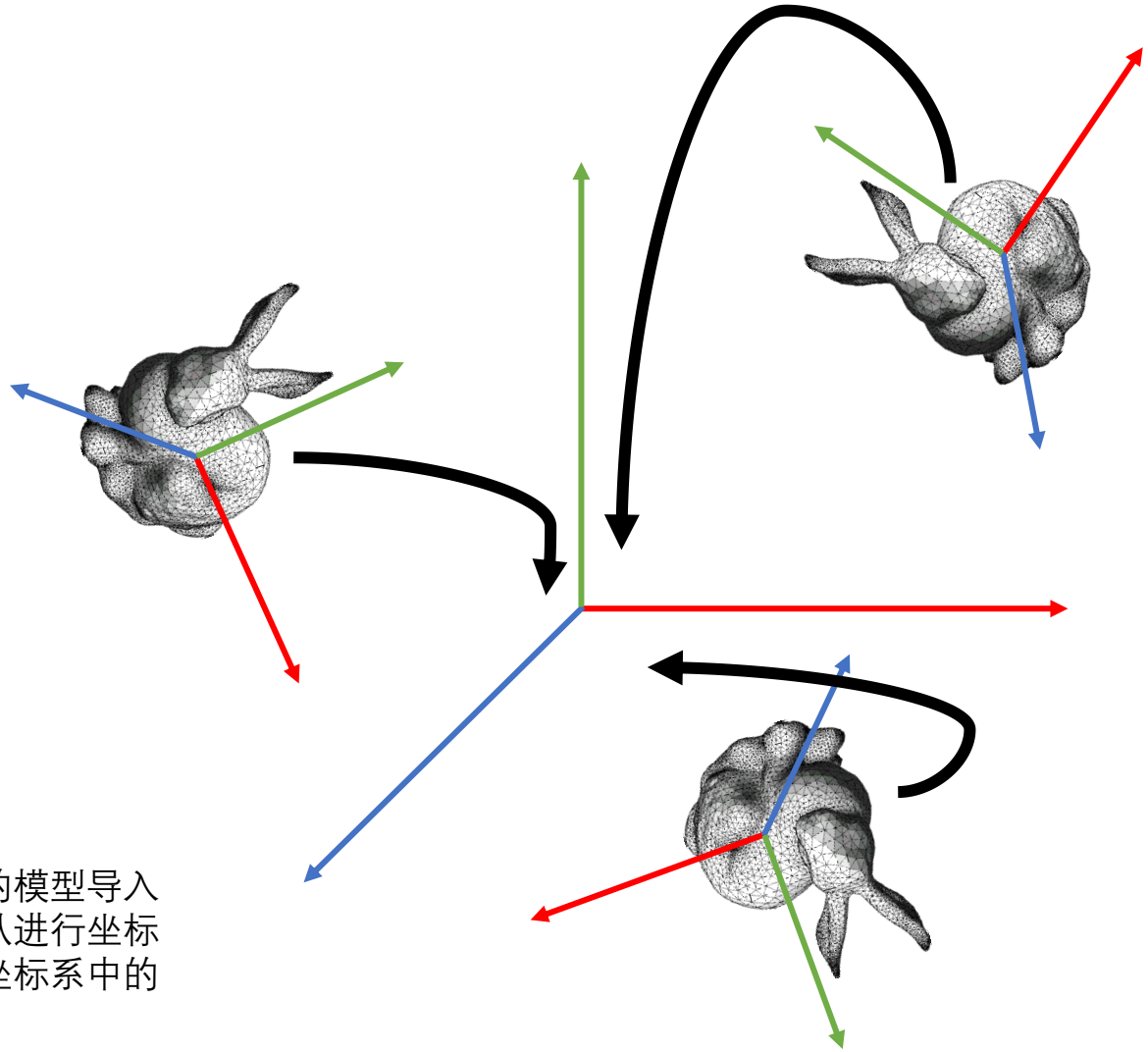


局部坐标系/模型坐标系



建模人员并不需要知道模型最终会被放在虚拟世界下的哪个位置，因此建模人员只需要按照自己的局部坐标系建立模型即可。

世界坐标系



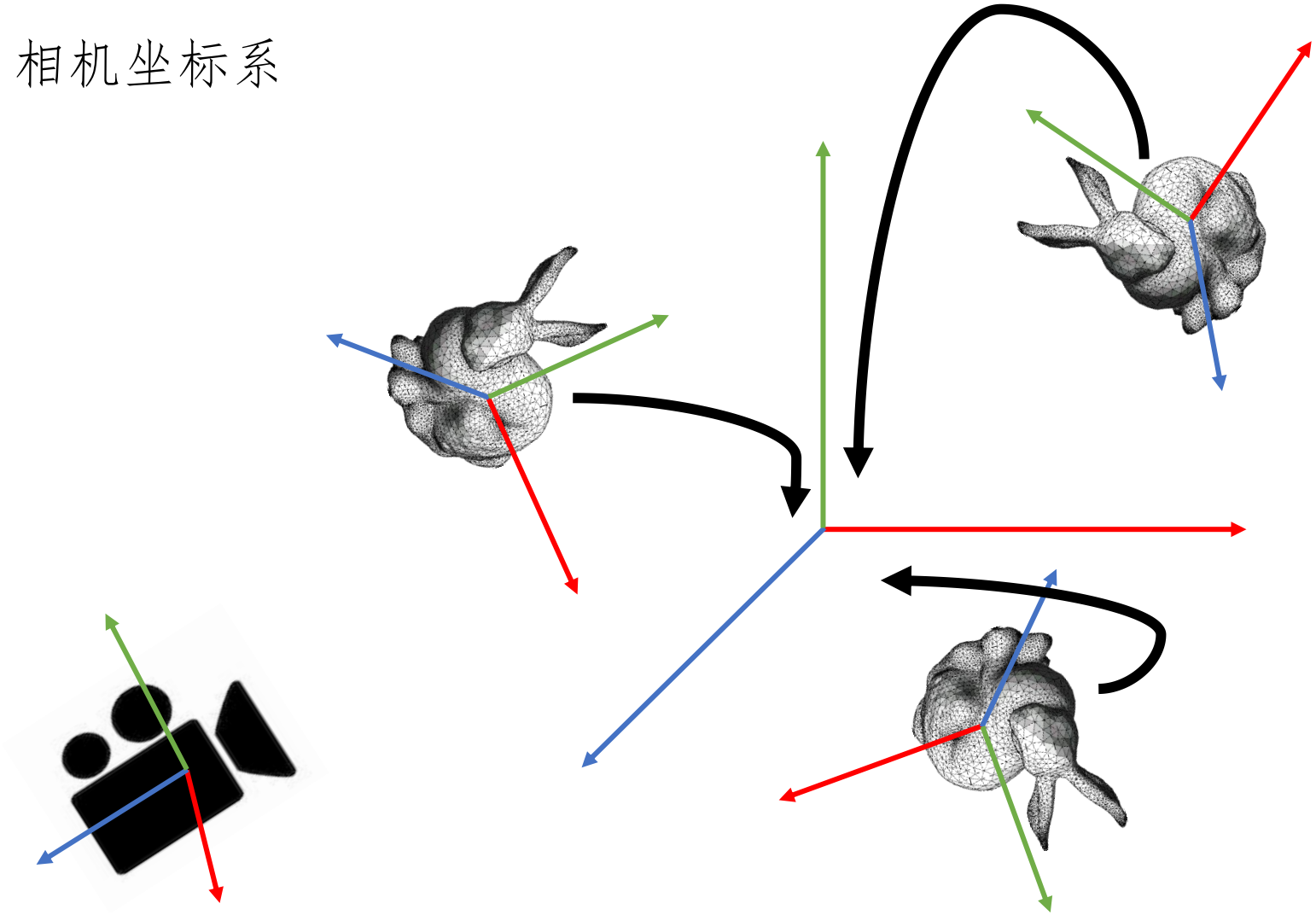
将根据局部坐标系创建好的模型导入虚拟世界后，需要对模型从进行坐标转换，以获得模型在世界坐标系中的位置

Model Coordinates

$[Model\ Matrix]$

World Coordinates

相机坐标系



获得模型的世界坐标后，通过根据相机所在位置，获得模型相对于相机坐标系的坐标。

Model Coordinates

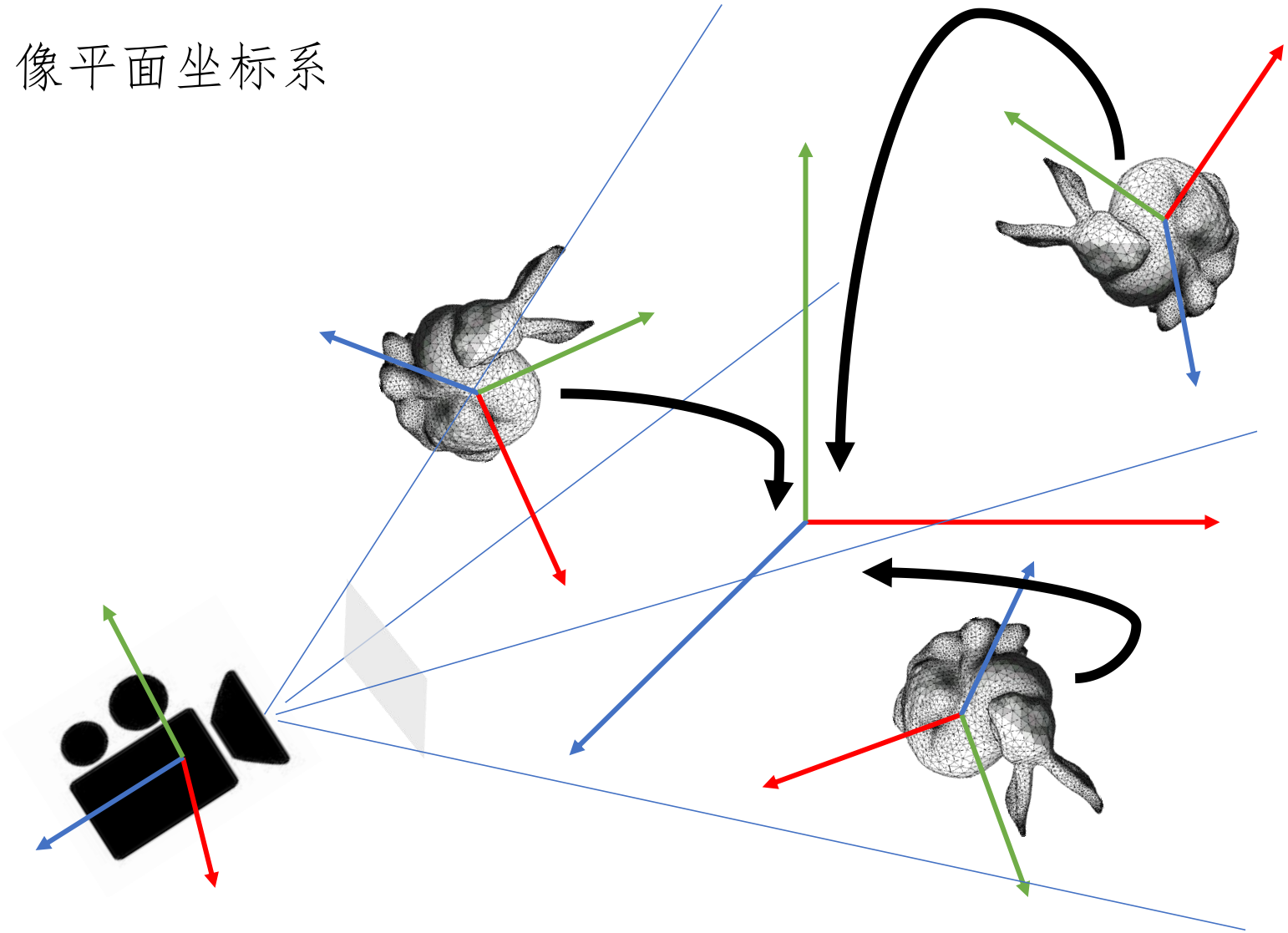
$[Model\ Matrix]$

World Coordinates

$[View\ Matrix]$

Camera Coordinates

像平面坐标系



获得相对于相机坐标系的坐标后，针对像平面进行投影，获得二维的坐标。

Model Coordinates

$[Model\ Matrix]$

World Coordinates

$[View\ Matrix]$

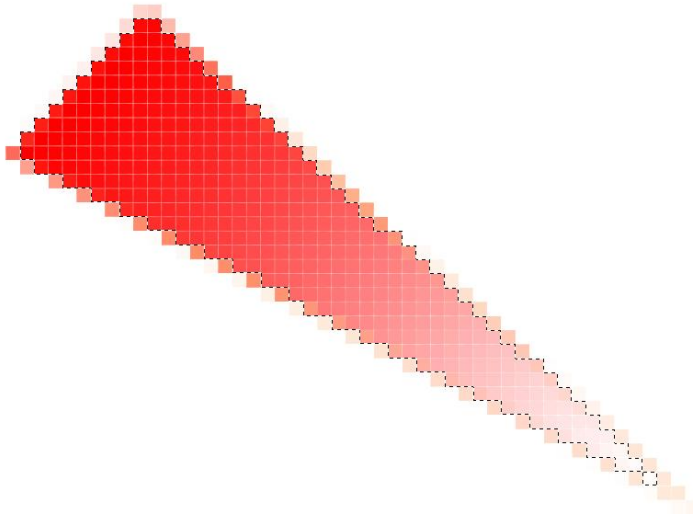
Camera Coordinates

$[Projection\ Matrix]$

Homogeneous Coordinates

渲染管线-光栅化和片段着色器

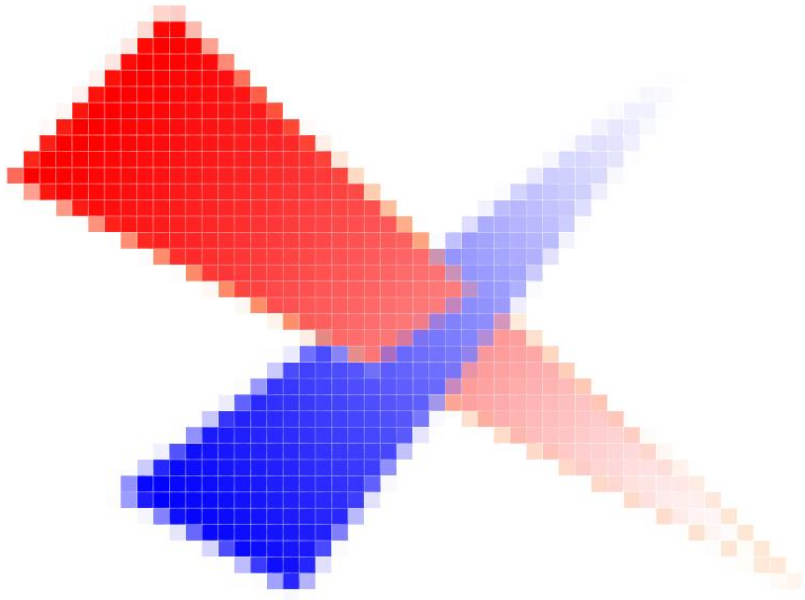
- 在上一步，我们获得了图元的二维坐标，此时坐标依然是在连续空间上的。在光栅化阶段，我们需要将连续的几何表示进行光栅化操作，以获得离散的像素。
- 补充：在光栅化之前，还会进行一些优化，如裁剪（clipping），背面剔除等，以减少计算。
- **光栅化时**，我们需要判定屏幕上哪些像素属于当前图元，哪些不属于。
- 由于我们在上一步为止，只有顶点（Vertex）的属性信息，因此需要通过插值的方式，获得当前属于该图元的像素点属性信息。
- 最终获得的离散化的、存有当前位置上所有属性信息的，我们称之为片段（fragment）
- 片段是显示的最小单位。片段也是像素的候选者。只有通过了后续的测试，片段才会成为最终显示的像素点。



- 接着在**片段着色器**上，我们对所有的片段进行颜色属性等的计算，可以实现光照、阴影、材质等效果。

渲染管线-测试混合阶段

- 深度测试
- 模板测试
- Alpha测试
- ……



- 通过测试的片段进行最终的混合；未通过的就舍弃。
- 实现遮挡、裁剪、半透明等效果

OpenGL配置-GLUT

- **背景：**

- OpenGL不是一种编程语言，而是一种API（Application Programming Interface，应用程序编程接口）。当我们说某个程序是基于OpenGL的或者说它是个OpenGL程序是，意思是说它是用某种编程语言如C或C＋＋编写的，其中调用了一个或多个OpenGL库函数。作为一种API，OpenGL遵循C语言的调用约定。
- OpenGL主要包括三个函数库，它们是核心库(gl)、实用函数库(glu)和编程辅助库(aux)。核心库中包含了OpenGL最基本的命令函数。核心库提供了一百多个函数，这些函数都以“gl”为前缀，用来建立各种各样的几何模型、进行坐标变换、产生光照效果、进行纹理映射、产生雾化效果等所有的二维和三维图形操作。实用函数库是比核心库更高一层的函数库，它提供四十多个函数，这些函数都以“glu”为前缀。由于OpenGL是一个图形标准，是独立于任何窗口系统或操作系统的，在OpenGL中没有提供窗口管理和消息事件响应的函数，也没有鼠标和键盘读取事件的功能，所以在编程辅助库提供了一些基本的窗口管理函数、事件处理函数和简单的事件函数。这类函数以“aux”作为前缀。值得一提的是，目前AUX编程辅助库已经很大程度上被GLUT库取代了。因此我们将在后续的学习中使用GLUT库。
- GLUT代表OpenGL应用工具包（OpenGL Utility Toolkit），是一个与窗口系统无关的工具包。它作为AUX库的功能更强的替代品，用于隐藏不同窗口系统API的复杂性。GLUT的函数前缀使用“glut”。
- GLUT的替代品包括SDL和GLFW。

OpenGL配置-GLUT

- OpenGL Utility Toolkit
- 封装了部分常用的功能，如建立窗口、连接窗口与渲染对象等等
- 开发者无需关心不同平台的窗口系统的不同
- 简单的处理鼠标键盘事件
- 直接绘制常用立体图形：长方体、球、犹他茶壶等

- 简单来说：GLUT就是帮你初始化gl，并且建好窗口等等，安装好GLUT就可以轻松使用OpenGL。
- 注：GLUT在许多年前就已经停止维护，并不支持很多新版本的OpenGL的API，因此仅作为实验课学习用。我们仅提供一个较容易学习的库作为图形学学习的起点，**不强制要求**必须使用GLUT。若你使用了其他的库，请在提交作业时告知硬件环境以及如何配置，保险起见，也请提供录屏。

OpenGL配置-GLUT使用

方法一：创建空项目->管理Nuget程序包->搜索并安装Nupengl程序包

OpenGL配置-GLUT使用

方法2：

1. 下载OpenGL

打开网址：https://www.opengl.org/resources/libraries/glut/glut_downloads.php

找到标题为 GLUT for Microsoft Windows 9X, ME, 2000, NT & XP users，下面有：If you want just the GLUT header file, the .LIB, and .DLL files all pre-compiled for Intel platforms……，点击 glutdlls37beta.zip 即可下载。

（注：为方便大家，已在压缩包内提供glutdlls37beta的内容，总共5个文件）

2. 配置OpenGL

解压glutdlls37beta.zip

然后找到vs2017安装的目录，路径为(.\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.26.28801\include), 创建一个名为gl的文件夹,并将解压到的glut.h文件复制其中。

再找到路径为 （.\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.26.28801\lib\x86 ，将解压到的glut.lib，glut32.lib复制其中。

最后把解压到的glut.dll和glut32.dll复制到C:\Windows\System32文件夹内（32位系统）或C:\Windows\SysWOW64(64位系统）。

HW

- 在模板基础上，绘制自己的学号后三位
- 如果后三位都一样（111，222，333，etc.），就向前移一位，绘制倒数4~2位
- 学在浙大上提交代码+实验报告pdf
- Bonus：学有余力的话，可以增加一些其他的图形，或者改变一些窗口参数，但必须有学号的绘制。

例图。
如果和自己的学号相同，请想办法让自己的作业变得和例图不同