Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

软光栅渲染器

计算机图形学课程设计报告

1650940 江晓湖

1651058 王毅诚

1751984 王舸飞

软件工程学院

同济大学

指导老师

贾金原

2020 年 1 月 12 号

摘要

项目思路、计划、工作量描述

Contents

List of Figures

List of Tables

1

项目介绍

1.1 项目综述

我们的项目是实现一个软光栅渲染器。实现的基本功能：顶点变换、裁剪、线框画线附加功能：有键盘鼠标交互的相机旋转移动、扫描线填充、三维模型读取、纹理贴图、点光源光照、天空盒子

1.1.1 ...

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti

sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

1.1.2 ...

1.2 开发环境

1.2.1 ...

Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus sodales, augue est scelerisque sapien, venenatis congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.

Etiam ac leo a risus tristique nonummy. Donec dignissim tincidunt nulla. Vestibulum rhoncus molestie odio. Sed lobortis, justo et pretium lobortis, mauris turpis condimentum augue, nec ultricies nibh arcu pretium enim. Nunc purus neque, placerat id, imperdiet sed, pellentesque nec, nisl. Vestibulum imperdiet neque non sem accumsan laoreet. In hac habitasse platea dictumst. Etiam condimentum facilisis libero. Suspendisse in elit quis nisl aliquam dapibus. Pellentesque auctor sapien. Sed egestas sapien nec lectus. Pellentesque vel dui vel neque bibendum viverra. Aliquam porttitor nisl nec pede. Proin mattis libero vel turpis. Donec rutrum mauris et libero. Proin euismod porta felis. Nam lobortis, metus quis elementum commodo,

nunc lectus elementum mauris, eget vulputate ligula tellus eu neque. Vivamus eu dolor.

Nulla in ipsum. Praesent eros nulla, congue vitae, euismod ut, commodo a, wisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nonummy magna non leo. Sed felis erat, ullamcorper in, dictum non, ultricies ut, lectus. Proin vel arcu a odio lobortis euismod. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Proin ut est. Aliquam odio. Pellentesque massa turpis, cursus eu, euismod nec, tempor congue, nulla. Duis viverra gravida mauris. Cras tincidunt. Curabitur eros ligula, varius ut, pulvinar in, cursus faucibus, augue.

1.2.2 ...

1.2.3 ...

1.3 第三方依赖库

1.3.1 ...

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetuer quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim,

tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.

Donec et nisl id sapien blandit mattis. Aenean dictum odio sit amet risus. Morbi purus. Nulla a est sit amet purus venenatis iaculis. Vivamus viverra purus vel magna. Donec in justo sed odio malesuada dapibus. Nunc ultrices aliquam nunc. Vivamus facilisis pellentesque velit. Nulla nunc velit, vulputate dapibus, vulputate id, mattis ac, justo. Nam mattis elit dapibus purus. Quisque enim risus, congue non, elementum ut, mattis quis, sem. Quisque elit.

Maecenas non massa. Vestibulum pharetra nulla at lorem. Duis quis quam id lacus dapibus interdum. Nulla lorem. Donec ut ante quis dolor bibendum condimentum. Etiam egestas tortor vitae lacus. Praesent cursus. Mauris bibendum pede at elit. Morbi et felis a lectus interdum facilisis. Sed suscipit gravida turpis. Nulla at lectus. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Praesent nonummy luctus nibh. Proin turpis nunc, congue eu, egestas ut, fringilla at, tellus. In hac habitasse platea dictumst.

Vivamus eu tellus sed tellus consequat suscipit. Nam orci orci, malesuada id, gravida nec, ultricies vitae, erat. Donec risus turpis, luctus sit amet, interdum quis, porta sed, ipsum. Suspendisse condimentum, tortor at egestas posuere, neque metus tempor orci, et tincidunt urna nunc a purus. Sed facilisis blandit tellus. Nunc risus sem, suscipit nec, eleifend quis, cursus quis, libero. Curabitur et dolor. Sed vitae sem. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Maecenas ante. Duis ullamcorper enim. Donec tristique enim eu leo. Nullam molestie elit eu dolor. Nullam bibendum, turpis vitae tristique gravida, quam sapien tempor lectus, quis pretium tellus purus ac quam. Nulla facilisi.

Duis aliquet dui in est. Donec eget est. Nunc lectus odio, varius at, fermentum in, accumsan non, enim. Aliquam erat volutpat. Proin sit amet nulla ut eros consectetuer cursus. Phasellus dapibus aliquam justo. Nunc laoreet. Donec consequat placerat magna. Duis pretium tincidunt justo. Sed sollicitudin vestibulum quam. Nam quis ligula. Vivamus at metus. Etiam imperdiet imperdiet pede. Aenean turpis. Fusce augue velit, scelerisque sollicitudin, dictum vitae, tempor et, pede. Donec wisi sapien, feugiat in, fermentum ut, sollicitudin adipiscing, metus.

Donec vel nibh ut felis consectetuer laoreet. Donec pede. Sed id quam id wisi laoreet suscipit. Nulla lectus dolor, aliquam ac, fringilla eget, mollis ut, orci. In pellentesque justo in ligula. Maecenas turpis. Donec eleifend leo at felis tincidunt consequat. Aenean turpis metus, malesuada sed, condimentum sit amet, auctor a, wisi. Pellentesque sapien elit, bibendum ac, posuere et, congue eu, felis. Vestibulum mattis libero quis metus scelerisque ultrices. Sed purus.

Donec molestie, magna ut luctus ultrices, tellus arcu nonummy velit, sit amet pulvinar elit justo et mauris. In pede. Maecenas euismod elit eu erat. Aliquam augue wisi, facilisis congue, suscipit in, adipiscing et, ante. In justo. Cras lobortis neque ac ipsum. Nunc fermentum massa at ante. Donec orci tortor, egestas sit amet, ultrices eget, venenatis eget, mi. Maecenas vehicula leo semper est. Mauris vel metus. Aliquam erat volutpat. In rhoncus sapien ac tellus. Pellentesque ligula.

Cras dapibus, augue quis scelerisque ultricies, felis dolor placerat sem, id porta velit odio eu elit. Aenean interdum nibh sed wisi. Praesent sollicitudin vulputate dui. Praesent iaculis viverra augue. Quisque in libero. Aenean gravida lorem vitae sem ullamcorper cursus. Nunc adipiscing rutrum ante. Nunc ipsum massa, faucibus sit amet, viverra vel, elementum semper, orci. Cras eros sem, vulputate et, tincidunt id, ultrices eget, magna. Nulla varius ornare odio. Donec accumsan mauris sit amet augue. Sed ligula lacus, laoreet non, aliquam sit amet, iaculis tempor, lorem. Suspendisse eros. Nam porta, leo sed congue tempor, felis est ultrices eros, id mattis velit felis non metus. Curabitur vitae elit non mauris varius pretium. Aenean lacus sem, tincidunt ut, consequat quis, porta vitae, turpis. Nullam laoreet fermentum urna. Proin iaculis lectus.

Sed mattis, erat sit amet gravida malesuada, elit augue egestas diam, tempus scelerisque nunc nisl vitae libero. Sed consequat feugiat massa. Nunc porta, eros in eleifend varius, erat leo rutrum dui, non convallis lectus orci ut nibh. Sed lorem massa, nonummy quis, egestas id, condimentum at, nisl. Maecenas at nibh. Aliquam et augue at nunc pellentesque ullamcorper. Duis nisl nibh, laoreet suscipit, convallis ut, rutrum id, enim. Phasellus odio. Nulla nulla elit, molestie non, scelerisque at, vestibulum eu, nulla. Ut odio nisl, facilisis id, mollis et, scelerisque nec, enim. Aenean sem leo, pellentesque sit amet, scelerisque sit amet, vehicula pellentesque, sapien.

1.3.2 ...

1.3.3 ...

2

基础功能

2.1 顶点变换

在软光栅渲染器中，顶点变换的基本任务是把空间三维顶点转换成屏幕上的二维坐标。这个过程涉及坐标变换的流水线，如图所示：

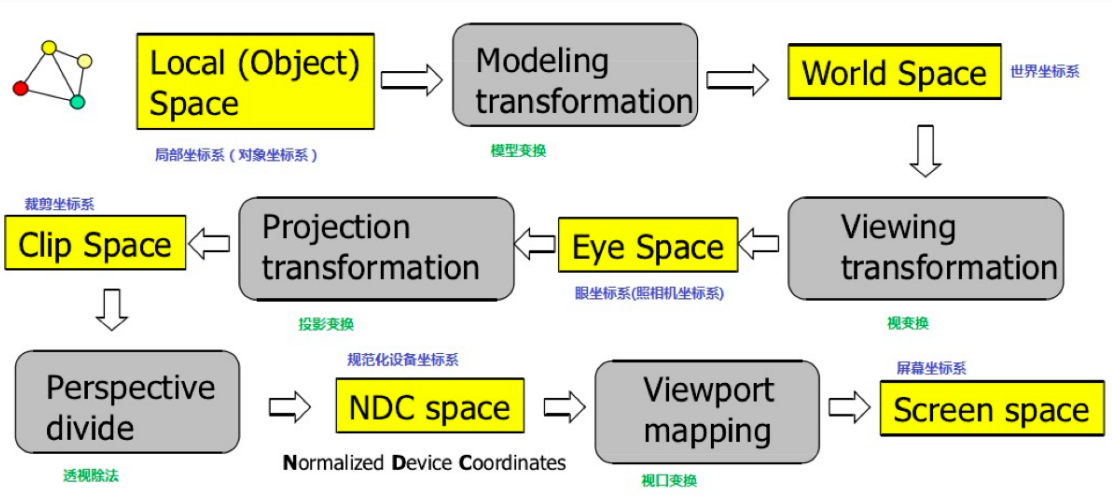

Figure 2.1: 软光栅渲染器坐标处理流水线

为了充分理解流水线各段，需要对几个坐标系进行理解。

2.1.1 局部坐标系

每个物体在被创建时，都会有自己的局部坐标系，一般是以物体的几何中心为原点，那么物体的每个顶点相对于几何中心都会有一个相对坐标。这个坐标系就被称为物体的局部坐标系 (Local Object Space)。该坐标系不是固定的，每个物体都有他们独立的坐标系，且仅对该对象适用。在物体移动或改变方向时，其局部坐标系也将随着移动或改变方向。

2.1.2 世界坐标系

因为每个物体都有自己的独立局部坐标系，如果要对多个物体观察，我们需要将它们放置到同一个绝对坐标系下，这个坐标系就叫世界坐标系 (World Space)。世界坐标系始终是固定不变的。

2.1.3 相机坐标系

当我们在世界坐标系中对物体进行观察时，就会有一个观察原点，对应一个相机坐标系 (Eye Space)，相机坐标系是和观察者密切相关的坐标系。相机坐标系和

屏幕坐标系相似，差别在于相机坐标系在三维空间中，而屏幕坐标系在二维平面里。为了方便处理，相机坐标系的原点和世界坐标系的原点是重合的，世界坐标系的 z 轴从相机的视角中心穿过。如图所示：

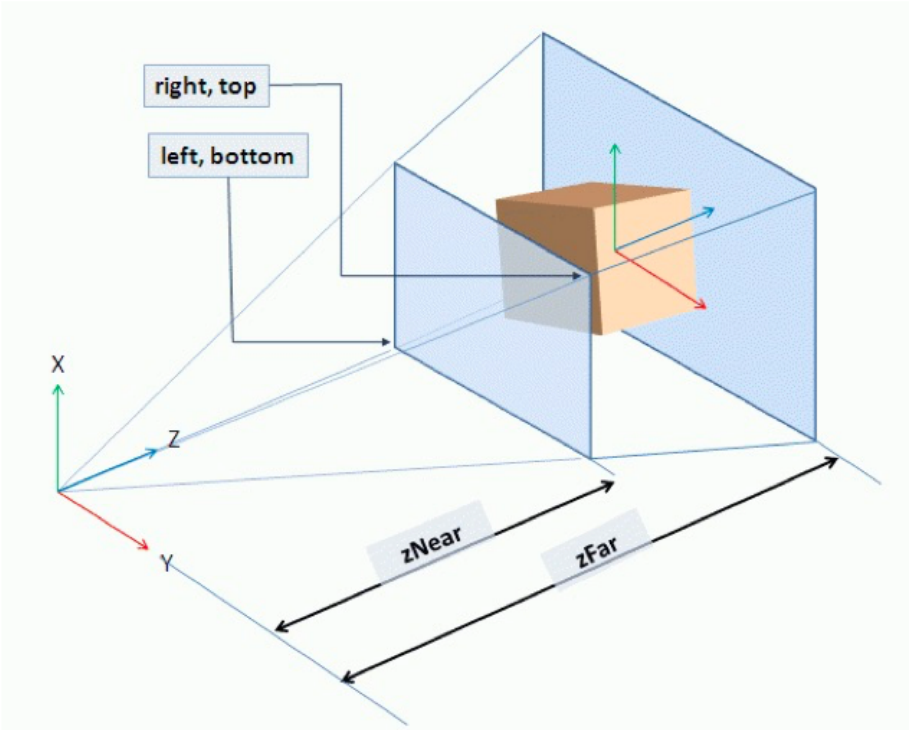

Figure 2.2: 相机坐标系的原点与世界坐标系的原点重合

2.1.4 裁剪坐标系

要了解为何要有裁剪坐标系，我们先得了解视锥体的概念：

视锥体

在图2.2中，相机有近视面 zNear 和远视面 zFar，只有处在近视面和远视面中间的物体才能被相机所观测到，近视面和远视面共同将相机的视角切分成了一个锥体，这个锥体就叫做视锥体。

正如我们前面所提，只有在视锥体中的物体才能被观测到，那么对物体处于视锥体外的部分自然要进行裁剪，但是在不规则的视锥体内进行裁剪是一件非常困难的事，所以人们将相机坐标投影到裁剪坐标系 (Clip Space)，裁剪坐标空间是一个立方体，这就很容易进行裁剪。

2.1.5 规范化设备坐标系

为了消除底层物理特性对裁剪的干扰，通常会把裁剪空间变换成一个规范化立方体，这个立方体的 x,y,z 坐标范围都是 [-1,1]。这个坐标系被称为规范化设备坐标系 (NDC space)。

2.1.6 屏幕坐标系

通常将屏幕上的设备坐标称为屏幕坐标。设备坐标又称为物理坐标，是指输出设备上的坐标。设备坐标用对象距离窗口左上角的水平距离和垂直距离来指定对象的位置，是以像素为单位来表示的，设备坐标的 X 轴向右为正，Y 轴向下为正，坐标原点位于窗口的左上角。

了解了各个坐标系后，就能对各个坐标系的变换有更深的认识。

2.1.7 模型变换

把物体从世界坐标系变换到局部坐标系称为模型变换。模型变换设计到的变换通常是很简单的平移变换。在我们的项目中味了进行简化，在创建物体时就给予了物体一个绝对坐标。三维空间中的平移变换矩阵如下：

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{pmatrix} \tag{2.1}$$

2.1.8 视变换

把物体从世界坐标变换到相机空间称为视变换，如前面所提，视变换主要是把物体通过平移和旋转变换到视锥体中，所以用到的变换主要是平移和旋转变换，平移变换矩阵为等式2.1，旋转变换矩阵为：

- 绕 Y 轴旋转

$$\begin{pmatrix} cos\theta & 0 & -sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

- 绕 X 轴旋转

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & sin\theta & 0 \\ 0 & -sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

- 绕 Z 轴旋转

$$\begin{pmatrix} cos\theta & sin\theta & 0 & 0 \\ -sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.4}$$

2.1.9 投影变换

从相机坐标系到裁剪坐标系的变换称为投影变换。投影变换矩阵的推导如下：

先观察视锥体的一个截面，如图所示：

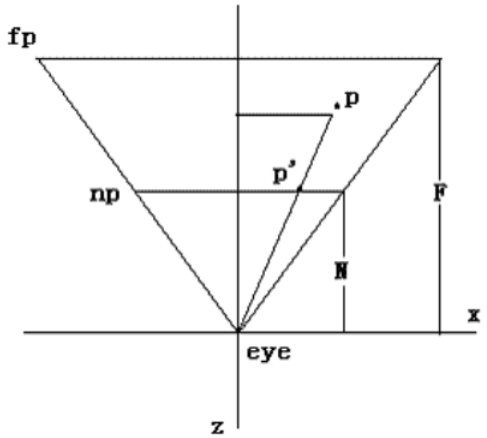

Figure 2.3: 视锥体截面

图中 N 是观察原点到近视面的距离，F 是观察原点到远视面的距离，$P(x, y, z)$ 为视锥体中的点，$P'(x', y', z')$ 是投影之后的点。这里 z 轴指向视锥体的反方向，指向正方向的推导过程与之类似。投影面可以选择任何平行于近视面的平面，为了简化分析，这里我们选择近视面作为投影平面。于是有 $z' = -N$，通过相似三角形容易得到以下式子：

$$\begin{aligned} \frac{x}{x'} = \frac{z}{z'} &= \frac{z}{-N} \\ x' &= -N\frac{x}{z} \\ y' &= -N\frac{y}{z} \end{aligned} \tag{2.5}$$

这样 P' 可以写为:

$$P' = (-N\frac{x}{z}, -N\frac{y}{z}, -N) \tag{2.6}$$

正如我们前面提到的，在不规则视锥体中进行裁剪是一件非常麻烦的事情，因此需要一个规范化的裁剪空间 (CVV)，假定裁剪空间中 $x, y, z \in [-1, 1]$，因此我们构造 z 的一个映射:

$$z^{'} = -\frac{az + b}{z} \tag{2.7}$$

使得式子 Eq.2.7 在 $z = -N$ 时的值为-1，而在 $z = -F$ 时的值为 1。于是投影变换可以暂时写为:

$$\begin{pmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} -Nx/z \\ -Ny/z \\ -(az+b)/z \\ 1 \end{pmatrix} \tag{2.8}$$

由待定系数法，有：

$$-\frac{az+b}{z}=\begin{cases}-1, & if \quad z=-N\\ 1, & if \quad z=-F\end{cases} \tag{2.9}$$

可以计算出系数 a,b:

$$\begin{aligned}a&=-\frac{F+N}{F-N}\\ b&=-\frac{2FN}{F-N}\end{aligned} \tag{2.10}$$

代入式子 Eq.2.8这样就可以得到透视投影矩阵的第一个版本：

$$\begin{pmatrix}N & 0 & 0 & 0\\ 0 & N & 0 & 0\\ 0 & 0 & -\frac{F+N}{F-N} & \frac{-2FN}{F-N}\\ 0 & 0 & -1 & 0\end{pmatrix} \tag{2.11}$$

虽然 z 方向满足 CVV 的要求了，但是 x，y 方向仍没有被限制在 $[-1,1]$ 之间，我们将投影平面的左边界值记为 $left$，右边界值记为 $right$，即 $[left,right]$，同理

对上下边界有 $[bottom, top]$。要把 $-Nx/z \in [left, right]$ 和 $-Ny/z \in [bottom, top]$ 映射到 $[-1, 1]$ 中，需要用到线性插值：

$$\begin{cases} \frac{-Nx/z-left}{right-left} = \frac{x-(-1)}{1-(-1)} \\ \frac{-Ny/z-bottom}{top-bottom} = \frac{y-(-1)}{1-(-1)} \end{cases} \tag{2.12}$$

可以解得：

$$\begin{cases} x = \frac{-2Nx/z}{right-left} - \frac{right+left}{right-left} \\ y = \frac{-2Ny/z}{top-bottom} - \frac{top+bottom}{top-bottom} \end{cases} \tag{2.13}$$

于是，P' 可以再写为：

$$P' = \begin{pmatrix} \frac{-2Nx/z}{right-left} - \frac{right+left}{right-left} \\ \frac{-2Ny/z}{top-bottom} - \frac{top+bottom}{top-bottom} \\ -(az+b)/z \\ 1 \end{pmatrix} \tag{2.14}$$

可以改写成：

$$P^{'} = \begin{pmatrix} \frac{2Nx}{right-left} + \frac{right+left}{right-left}z \\ \frac{2Ny}{top-bottom} + \frac{top+bottom}{top-bottom}z \\ az + b \\ -z \end{pmatrix} \tag{2.15}$$

设投影矩阵为 M，则运用待定系数法有：

$$M \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{2Nx}{right-left} + \frac{right+left}{right-left}z \\ \frac{2Ny}{top-bottom} + \frac{top+bottom}{top-bottom}z \\ az + b \\ -z \end{pmatrix} \tag{2.16}$$

可以求得矩阵 M：

$$\begin{pmatrix} \frac{2N}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2N}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & -\frac{F+N}{F-N} & \frac{-2FN}{F-N} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{2.17}$$

在实际应用中，更常使用视角 $Fovy$ 和投影面的宽高比 $Aspect$ 来进行投影变换矩阵的描述，$Fovy$ 和 N F $left$ $right$ top $bottom$ 的关系如图所示：

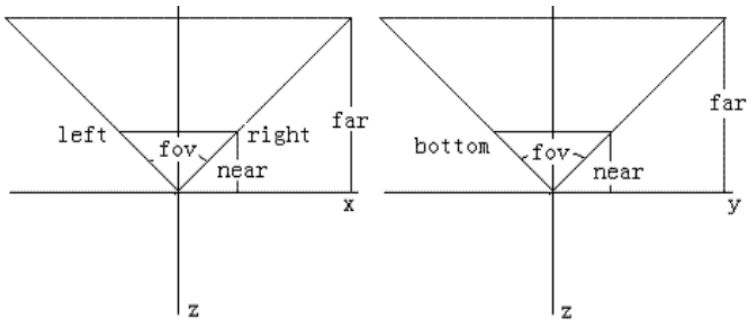

Figure 2.4: 视角 Fovy 示意图

$Fovy$ 即视野，是视锥体在 xz 平面或者 yz 平面的开角角度。在 OpenGL 中使用 yz 平面。由图2.4可以得到如下关系：

$$\begin{aligned} right &= N \times tan(fovy/2) \\ left &= -right \\ top &= N \times tan(fovy/2) \\ bottom &= -top \end{aligned} \tag{2.18}$$

而

$$Aspect = \frac{right - (-right)}{top - (-top)} = \frac{right}{top} \tag{2.19}$$

将式子 Eq.2.18 和式子 Eq.2.19 代入式子 Eq.2.17 可得最终的投影矩阵 M:

$$\begin{pmatrix} \frac{cot\frac{Fovy}{2}}{Aspect} & 0 & 0 & 0 \\ 0 & \frac{Fovy}{2} & 0 & 0 \\ 0 & 0 & -\frac{F+N}{F-N} & \frac{-2FN}{F-N} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{2.20}$$

2.1.10 透视除法

透视除法将裁剪坐标空间变换为 NDC 空间，方法如下：

$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} /w = \begin{pmatrix} x/w \\ y/w \\ z/w \\ 1 \end{pmatrix} \tag{2.21}$$

2.1.11 视口变换

最后一个变换为视口变换，它将 NDC 空间变换到最终屏幕坐标系，视口变换矩阵的推导过程如下：视口变换是将下面立方体中的物体变换到视口即屏幕中：

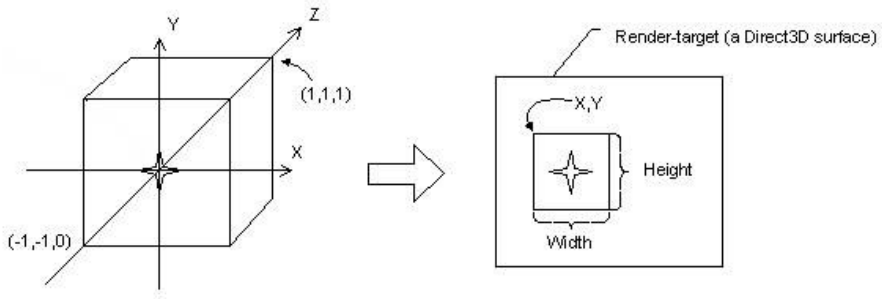

Figure 2.5: 视口变换示意图

其中，立方体的坐标范围为：

$$\begin{cases} -1 \leq x \leq 1 \\ -1 \leq y \leq 1 \\ 0 \leq z \leq 1 \end{cases} \tag{2.22}$$

而视口的坐标范围是：

$$\begin{cases} X \leq x \leq X + Width \\ Y \leq y \leq Y + Height \\ MinZ \leq z \leq MaxZ \end{cases} \tag{2.23}$$

在图2.5中，视口的起点为 (X, Y)，宽高分别为 $Width$ 和 $Height$，x 轴向右为正，y 轴向下为正，y 轴的方向与三维坐标正好相反。视口是一个 2D 平面，在视口变换中，Z 坐标也是跟着变换的，只是在图中没有体现。假设变换矩阵的第一列为 $[x^{'}, y^{'}, z^{'}, 1]^T$，根据上图，立方体在 (-1,1,0,1) 映射到视口中的起点 (X,Y,MinZ,1)，立方体中的右上角点 (1,1,0,1) 映射到视口中的点 (X+Width,Y,MinZ,1)，于是可以列式：

$$\begin{cases} [-1, 1, 0, 1] * [x^{'}, y^{'}, z^{'}, 1]^T = X \\ [1, 1, 0, 1] * [x^{'}, y^{'}, z^{'}, 1]^T = X + Width \end{cases} \tag{2.24}$$

可以求得变换矩阵的第一列：

$$[\frac{Width}{2}, 0, 0, x + \frac{Width}{2}]$$

同理可以求得变换矩阵的第二列、第三列、第四列，从而可以求得视口变换矩阵：

$$\begin{pmatrix} \frac{Width}{2} & 0 & 0 & 0 \\ 0 & -\frac{Height}{2} & 0 & 0 \\ 0 & 0 & MaxZ - MinZ & 0 \\ X + \frac{Width}{2} & Y + \frac{Height}{2} & MinZ & 1 \end{pmatrix} \tag{2.25}$$

通常的视口坐标系的起点为 $[X, Y] = [0, 0]$，我们也不需要关注 Z 方向上的变换。

2.2 裁剪

裁剪是一个非常基础且重要的功能，为了渲染整个物体，需要将物体切分为一个个的三角形进行统一操作，其中切分为三角形后的第一步，就是需要裁剪，目的有两个：

1. 裁剪掉在视野范围内不应该被看到的部分

2. 通过裁剪可以减少计算的工作量，在视野外的部分不需要渲染

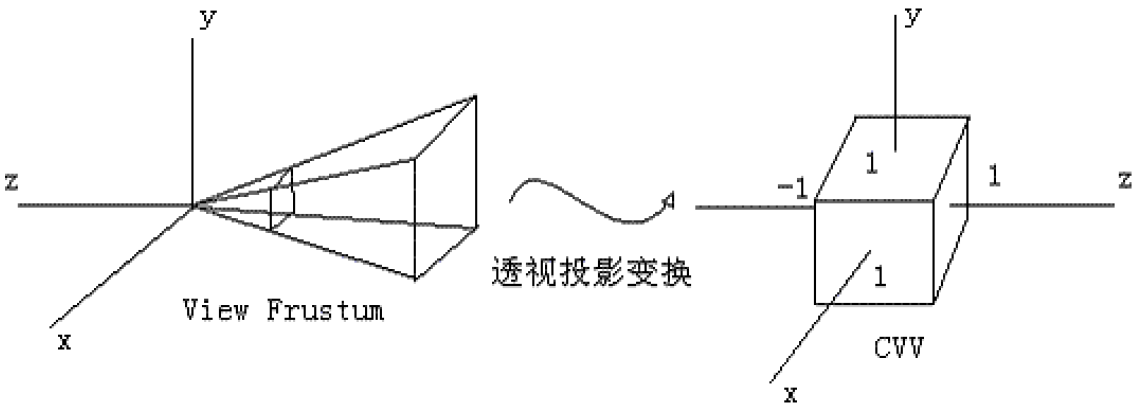

Figure 2.6: 透视投影变换

裁剪针对的是已经分割好为若干三角形的几何体，分割的步骤比较简单就不详述，在处理这些三角形的时候，需要先从空间坐标变换到 cvv 裁剪坐标进行裁剪，针对上下左右前后六个面进行裁剪操作，对于每一个面的裁剪，坐标都有变化，所以使用六个函数来进行裁剪操作：

```
1 static void doClippingInCvv(vector<Triangle> &triList){
2       _doClppingInCvvAgainstNearPlane(triList);
3       _doClppingInCvvAgainstFarPlane(triList);
4       _doClppingInCvvAgainstLeftPlane(triList);
5       _doClppingInCvvAgainstRightPlane(triList);
6       _doClppingInCvvAgainstTopPlane(triList);
7       _doClppingInCvvAgainstBottomPlane(triList);
8    }
```

裁剪是在 cvv 空间进行的操作，整个裁剪流程如下图所示：

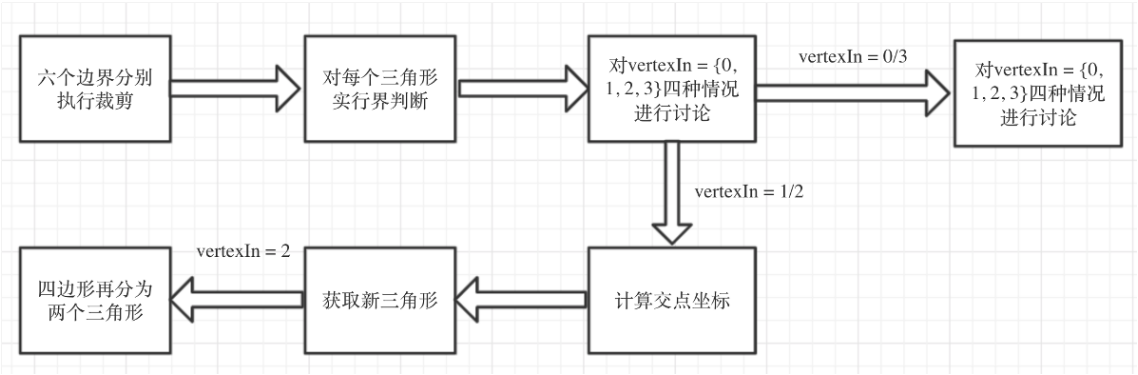

Figure 2.7: 裁剪流程

首先将图像分割出的一组三角形向量通过六个面分别进行裁剪操作，对于每一个边，需要判断其内外关系，其中按照边界内的点数，分为 0、1、2、3 这四种情况，这四种情况如下图所示：

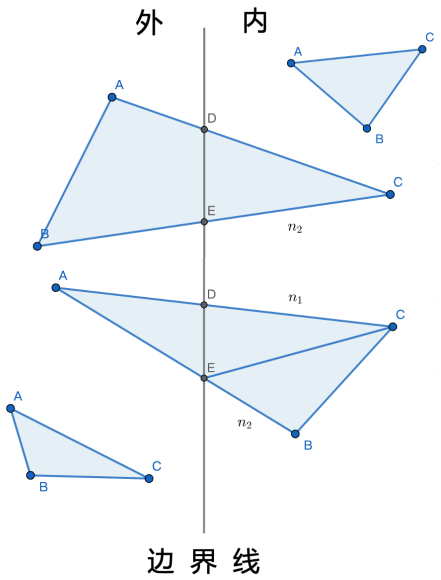

Figure 2.8: 三角形相对于边界的四种情况

第一种情况，内部点有三个，即三角形全在内部，此时不进行裁剪

第二种情况，内部点有一个，有两条边和边界有交点，此时需要用 A、D、C 三点的计算出 CD/AC 的占比 n_1，再利用 n_1 在 AC 边上做插值求得 D 点的坐标，E 点的坐标用相同的方法可以求得，此时生成的三角形 CDE 即为在内部的三角形，加入输出向量

```
vector<int> indiceOut;
int inIdx = 0;
for (int i = 0; i < bList.size(); ++i)
{
    if (bList.at(i)){
        indiceOut.push_back(i);
    else {
        inIdx = i;
    }
    VertexOut &vertIn = vertice.at(inIdx);
    for (int i = 0; i < indiceOut.size(); ++i)
    {
        int index = indiceOut.at(i);
        VertexOut &vertOut = vertice.at(index);
        double factor = _getFactorForLeftPlane(vertIn, vertOut);
        VertexOut vertNew = vertIn.interpolate(vertOut, factor);
        vertice[index] = vertNew;
    }
    tri.v1 = vertice[0];
    tri.v2 = vertice[1];
    tri.v3 = vertice[2];
```

第三种情况，内部点有两个，和第二种情况求法一致，但不同的是最后得到的是一个四边形而非三角形，所以要多一步分割的过程，将四边形 BCDE 分割为三角形 BCE 和三角形 CDE

第四种情况，内部点有零个，这代表三角形完全不在分界线以内，也就是说三角形应当被完全剔除，输出为空，这样也为后面的计算减少了工作量

2.3 线框画线

光栅中常用的算法仍是 Bresenham 算法，我们的项目也一样。关于 Bresenham 算法的原理课堂上已经有详细的描述，此处不再赘述。书本上对 Bresenham 算法的描述限制在 $x > 0, y > 0$ 且直线斜率在 $[0, 1]$ 之间，即下图中的 1a 象限：

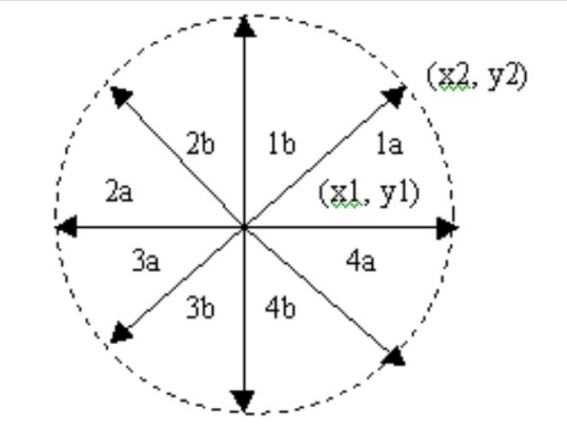

Figure 2.9: Bresenham 画线象限示意图

为了实现光栅渲染，必须将 Bresenham 扩展到全部 8 个象限。其实其他象限的基本处理思路与 1a 是一样的，只不过在 1a 中 y 随 x 递增而递增，在其他象限可能是 x 随 y 递增而递增，y 随 x 递增而递减，具体情况如下表所示，只需要判定当前属于哪一象限，然后相应使用 Bresenham 即可。

象限	取值
1a、2a	$y_{i+1} = y_i$ 或 $y_i = y_i + 1$
3a、4a	$y_{i+1} = y_i$ 或 $y_i = y_i - 1$
1b、2b	$x_{i+1} = x_i$ 或 $x_i = x_i + 1$
3b、4b	$x_{i+1} = x_i$ 或 $x_i = x_i - 1$

Table 2.1: 象限取值表

3

附加功能

3.1 键鼠交互的相机旋转移动

3.1.1 平移旋转变换

其实，所谓相机的旋转移动可以看成相机固定不动，而物体在进行逆变换，例如，相机左移一个单位其实就是物体向右移动一个单位。因此要相机的旋转移动，只要实现物体对应的旋转移动即可。所用到的变换矩阵如下：

1. 平移变换矩阵

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{pmatrix} \tag{3.1}$$

2. 绕 X 轴旋转变换矩阵

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & sin\theta & 0 \\ 0 & -sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

3. 绕 Y 轴旋转变换矩阵

$$\begin{pmatrix} cos\theta & 0 & -sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.3}$$

4. 绕 Z 轴旋转变换矩阵

$$\begin{pmatrix} cos\theta & sin\theta & 0 & 0 \\ -sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.4}$$

5. 缩放变换矩阵

$$\begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.5}$$

3.1.2 键鼠交互

为了增加交互性，我们自己加入了键鼠交互方面的功能，即：键盘通过上下左右或 WASD 键可以进行视角的移动、通过 TAB 可以切换场景，鼠标通过拖动也可以实现视角的移动，这些移动本质上是通过物体坐标的改变来实现的

首先需要一个全局事件监听函数，响应 event 并判断对应的类型，最后执行特定功能函数：

```
void App::OnEvent(SDL_Event* event , double dt)
{
    switch (event->type)
    {
        case SDL_QUIT:
            running = false;
            break;
```

```
8
9           case SDL_MOUSEBUTTONDOWN:
10              mouseIsDown = 1;
11              break;
12
13          case SDL_MOUSEBUTTONUP:
14              mouseIsDown = 0;
15              break;
16
17          case SDL_MOUSEMOTION:
18              if(mouseIsDown){
19                  onMouseDrag(*event);
20                  break;
21              }
22              break;
23
24          case SDL_KEYDOWN:
25              onKeyPress(event->key.keysym.sym , dt);
26              break;
27
28          default:
29              break;
30      }
31  }
```

键盘的监听可以使用 SDL2 库中的键盘事件接口来实现：

```
1   void App::onKeyPress(SDL_Keycode keyCode , double dt) {
2       auto camera = Camera::getInstance();
3       auto v = 45;
4       auto velo = camera->getMoveVelo();
5       switch (keyCode) {
6           case SDLK_ESCAPE:
7               running = false;
8               break;
9           case SDLK_w:
10              camera->offsetPosition(camera->forward() * velo * dt);
```

```
            break;
        case SDLK_s:
            camera->offsetPosition(-camera->forward() * velo * dt);
            break;
        case SDLK_a:
            camera->offsetPosition(-camera->right() * velo * dt);
            break;
        case SDLK_d:
            camera->offsetPosition(camera->right() * velo * dt);
            break;
        case SDLK_z:
            camera->offsetPosition(camera->up() * velo * dt);
            break;
        case SDLK_x:
            camera->offsetPosition(-camera->up() * velo * dt);
            break;
        case SDLK_UP:
            camera->offsetDirection(v * dt, 0);
            break;
        case SDLK_DOWN:
            camera->offsetDirection(-v * dt, 0);
            break;
        case SDLK_LEFT:
            camera->offsetDirection(0, -v*dt);
            break;
        case SDLK_RIGHT:
            camera->offsetDirection(0, v *dt);
            break;
        case SDLK_TAB:
            Camera::getInstance()->resetCamera();
            Canvas::getInstance()->addScene();
            Canvas::getInstance()->resetNode();
            break;
        default:
            break;
    }
}
```

鼠标的监听则要复杂一些，因为要实现拖动的交互效果，SDL2 中并没有直接的接口，需要先通过 MouseIsDown 进行判断，如果鼠标被按下，则调用 MouseMotion 监听鼠标移动轨迹，通过方向、速率等参数传达到移动函数中，最后调整一些参数，使得拖动效果看起来更真实

```
void App::onMouseDrag(SDL_Event& event) {
    auto v = 45;
    auto camera = Camera::getInstance();
    auto velo = camera->getMoveVelo();
    camera->offsetPosition(camera->right() * velo * event.motion.xrel *
        0.001);
    camera->offsetPosition(-camera->up() * velo * event.motion.yrel *
        0.001);
    camera->offsetDirection(0 , -v * event.motion.xrel * 0.0028);
    camera->offsetDirection( v * event.motion.yrel * 0.0028, 0);
}
```

3.2 区域填充

本渲染器用到的区域填充算法是课堂中学过的扫描线填充算法，不同的是这里分了更多情况进行优化分析，在通常情况下，扫描线填充算法运用于一般多边形是比较复杂的，所以项目中我们都采取先转化为三角形单元的形式，但是三角形也分为一般三角形、平底三角形和平顶三角形，后面两种三角形对于扫描线算法而言非常友好，因为其一边可以恰好与扫描线重合。对于一般三角形，可以采取从中间分成一个平底三角形和平顶三角形的策略来进行扫描，最后扫描的情况只

需要分别针对这两种特殊的三角形即可：

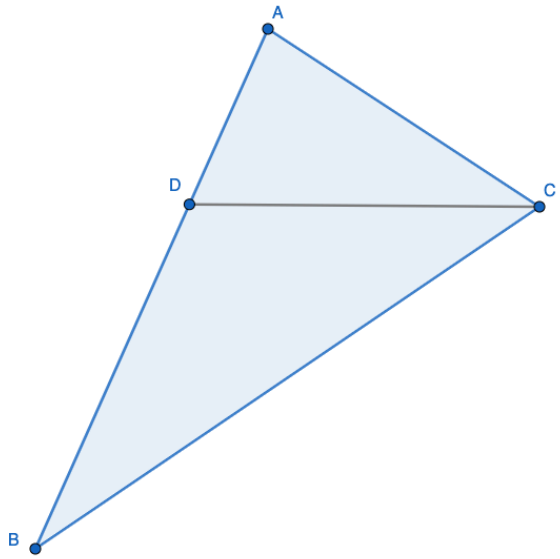

Figure 3.1: 普通三角形的切分

比如上图的普通三角形 ABC，就可以分为一个平底三角形 ACD 和一个平顶三角形 BCD

```
if (MathUtil::equal(pVert1->pos.y, pVert2->pos.y))
{
    triangleBottomRasterize(*pVert1, *pVert2, *pVert3);
}
else if (MathUtil::equal(pVert2->pos.y, pVert3->pos.y))
{
    triangleTopRasterize(*pVert1, *pVert2, *pVert3);
}
else
{
    double ty = pVert2->pos.y;
    double factor = (ty - pVert1->pos.y) / (pVert3->pos.y - pVert1->
        pos.y);
```

```
13          VertexOut tVert = pVert1->interpolate(*pVert3, factor);
14          triangleTopRasterize(*pVert1, tVert, *pVert2);
15          triangleBottomRasterize(*pVert2, tVert, *pVert3);
16      }
```

对于平顶三角形而言，扫描线于顶部重合，自顶向下循环遍历 y 值，通过比例计算插值从而计算出中间情况的需要填充的像素，scanLineFill 直接调用底层的 drawPixel 对像素进行着色

```
1       for (int py = startPY; py * sign <= sign * endPY; py = py + sign)
2       {
3           double ld = 1.0f;
4           double factor = (py - startPY) * ld / (endPY - startPY);
5           VertexOut vertStart = pVert1->interpolate(*pVert2, factor);
6           VertexOut vertEnd = pVert1->interpolate(*pVert3, factor);
7           scanLineFill(vertStart, vertEnd, py);
8       }
```

3.3 纹理贴图

3.4 光照

3.4.1 冯氏光照模型

现实世界的光照是很复杂的，物体的材质和光照的特性都会影响最终的渲染效果，在有限的计算能力下，我们很难模拟完全真实的光照条件。因此，图形学的渲染更倾向于使用一种简化的模型对现实情况进行近似。

本次我们参考了 OpenGL 的处理方式，采用冯氏光照模型 (Phong Lighting Model) 进行光照模拟。冯氏光照模型的主要结构由 3 个分量组成：环境 (Ambient)、漫反射 (Diffuse) 和镜面 (Specular) 光照。下图展示了这些光照分量的影响：

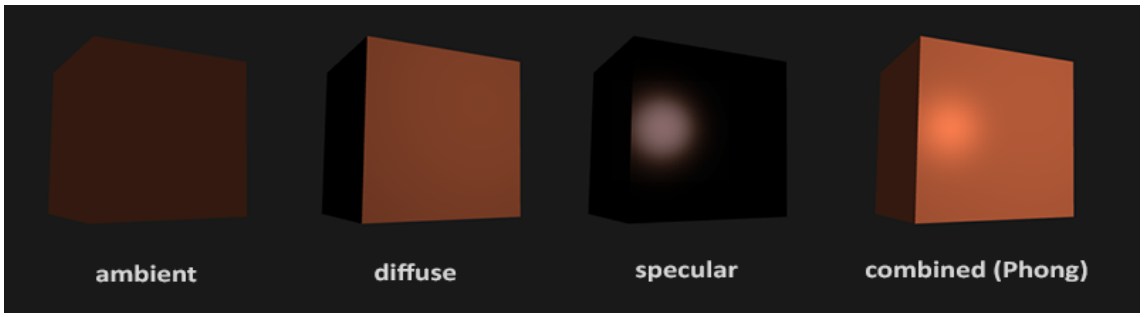

Figure 3.2: 冯氏光照模型分量

- 环境光照 (Ambient Lighting)：即使在黑暗的情况下，世界上通常也仍然有一些光亮（月亮、远处的光），所以物体几乎永远不会是完全黑暗的。为了模拟这个，我们会使用一个环境光照常量，它永远会给物体一些颜色。

- 漫反射光照 (Diffuse Lighting)：模拟光源对物体的方向性影响 (Directional Impact)。它是冯氏光照模型中视觉上最显著的分量。物体的某一部分越是正对着光源，它就会越亮。

- 镜面光照 (Specular Lighting)：模拟有光泽物体上面出现的亮点。镜面光照的颜色相比于物体的颜色会更倾向于光的颜色。

将上述三个分量全部叠加后即可得到最终的渲染颜色。

3.4.2 环境光照

在现实环境中，光源可能有很多个，并可能通过多个物体进行反射从而简介对其他物体产生影响。考虑到这种情况的算法叫全局光照 (Global Illumination) 算法，但是这种算法开销极高，且运算较为复杂。

在冯氏光照中采用了最简单的方法模拟环境光照，即用常量因子乘以环境颜色获得一个近似结果，它在代码中的体现如下：

```
1  Color PhongShader::getAmbient(const VertexOut &frag)
       const {
2  Color ambient = _ambient.color * _ambient.factor;
3  return ambient;
4  }
```

3.4.3 漫反射光照

比起环境光照，漫反射光照分量最能产生显著的视觉影响。它的思想是，同光线角度最接近的片段可以获得最高的亮度：

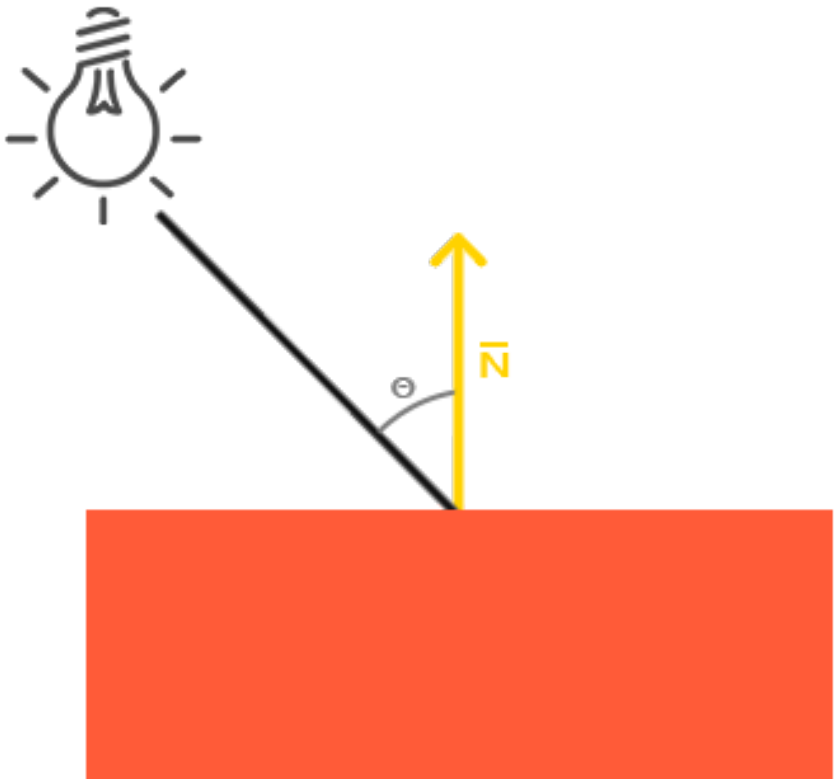

Figure 3.3: 漫反射分量示意图

如图，其中 $\overline{N}$ 代表其中一个片元的法向量，它与光照方向产生了角度 θ；对光线入射向量和法向量分别标准化，之后取两者的点积可以得到 $cos\theta$，将该值与光线颜色相乘，取一个比例系数之后得到结果，该值的大小反映了夹角大小的同时

也指代了光线的明暗。

值得注意的是，有时候这个点积会得到负值，但这意味着光线与某个法向量成钝角，也就是在片元的“另一边”，此时在实现时只需要将它取为 0 即可，下面是漫反射分量在代码中的体现：

```
1    double cosTheta = ray.dot(normal);
2    double diff = max(cosTheta , (double)0.0f);
3    Color diffuse = _light.color* _light.factor * diff *
         _material.diffuseFactor;
4    return diffuse;
```

3.4.4 镜面光照

镜面光照将观察者的观察向量也融入到了渲染之中。假设物体是一面镜子，那么入射的光将被完全反射，此时就必须考虑该反射光和观察者视角对于物体亮度的影响。

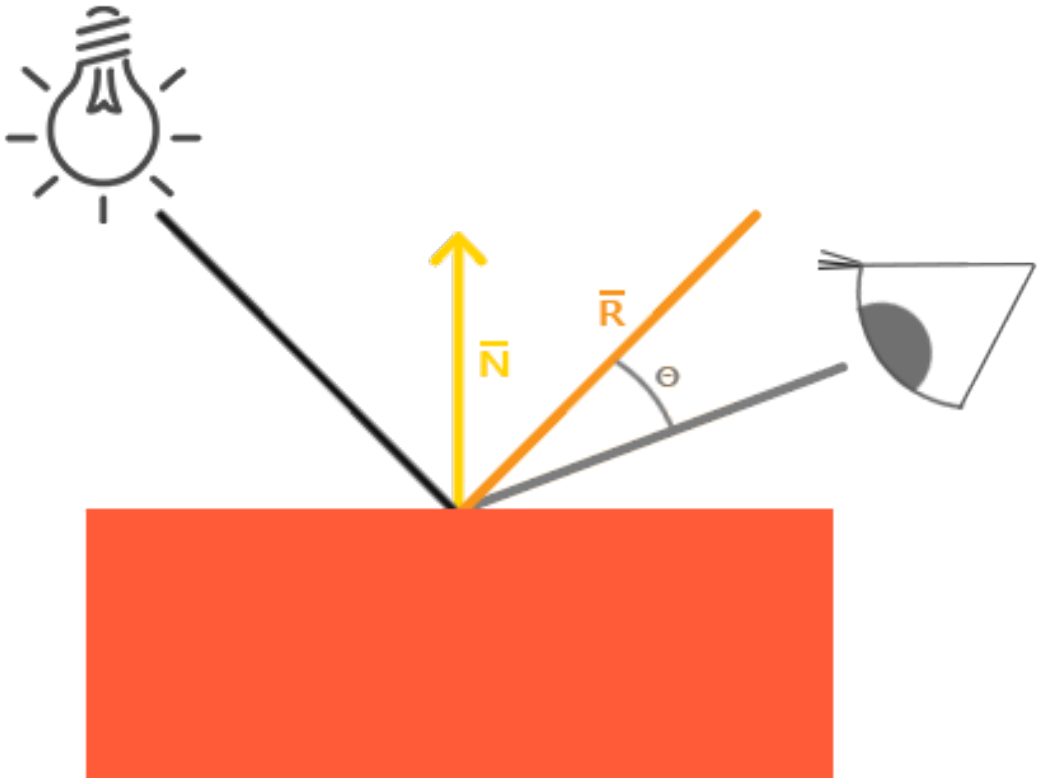

Figure 3.4: 镜面分量示意图

如图，我们通过反射法向量周围光的方向来计算反射向量。然后我们计算反射向量和视线方向的角度差，如果夹角越小，那么镜面光的影响就会越大。它的作用效果就是，当我们去看光被物体所反射的那个方向的时候，我们会看到一个高光，它的代码实现如下：

```
1  Vec3 center = (ray + viewDir).getNormalize();
2    auto spec = pow(max(center.dot(normal), 0.0), _material.
```

```
        shininess);
3   Color specular = _light.color * _light.factor * spec *
        _material.specularFactor;
4   return specular;
```

在实现时，计算反射光的法向量比较麻烦，为此，先将入射光单位向量和观察方向单位向量相加求出两者的角平分线，之后再计算角平分线与法向量的夹角，这个计算的合理性如下：

$$\frac{2\alpha+\theta}{2}-\alpha=\frac{\theta}{2} \tag{3.6}$$

其中，α 是入射角，θ 是待求夹角。

在下一行，将求得的值取了一个次幂，所取的指数叫做反光度 (Shininess)，一个物体的反光度越高，反射光的能力越强，散射得越少，高光点就会越小。下面的图片反映了不同反光度对渲染效果的影响：

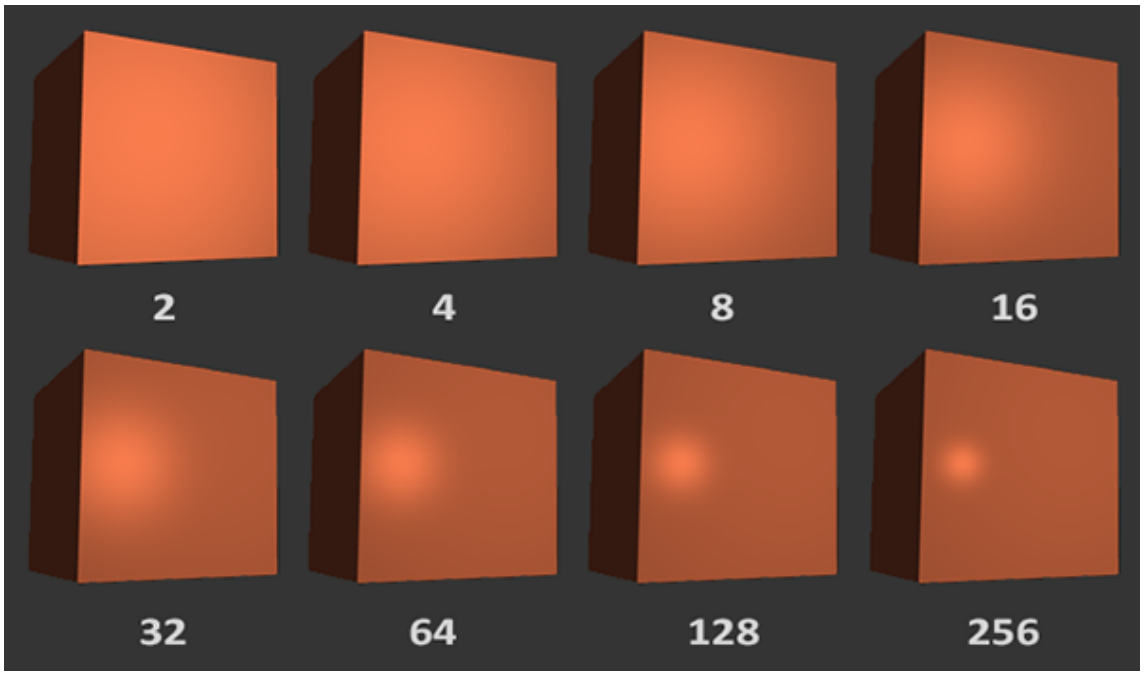

Figure 3.5: 不同反光度的视觉效果

最后，将上述三个分量参数简单相加后乘以物体本身的颜色，即输出冯氏光照的结果。

3.5 三维模型

3.6 天空盒子

4

功能效果演示

Your main contributions go here

4.1 顶点变换

4.2 裁剪

4.3 线框画线

4.4 键鼠交互的相机移动

4.5 扫描线填充

4.6 纹理贴图

4.7 光照

4.8 三维模型

4.9 天空盒子

References