CMU 15-462

前言

参考资料:

- 15-462/662 Fall 2022
- 001-【CMU15-462/662】【Computer Graphics 】【计算机图形学】【Course Intro】_哔哩哔哩_bilibili

lec1

什么是计算机图形学?

- 希望使用**视觉**传递信息，图形化层次人机交互，而非一串数字
  - 视觉信息的传递的**带宽**最大，大脑的30%
  初步的定义：**使用计算机合成视觉信息**，进一步的扩展定义，**使用计算机合成感官刺激**
- 除了CG，也有其他领域，例如合成声音、触觉等等
- 计算机视觉，CG逆向的一个学科，将刺激物转化为数字信息

一些应用例如数字模型转换为物理实物，如3D打印...以及非常多的其他应用

本门课概述

- **All these applications demand *sophisticated* theory & systems**
- **Theory**
  - **basic representations** *(how do you digitally encode shape, motion?)*
  - **sampling & aliasing** *(how do you acquire & reproduce a signal?)*
  - **numerical methods** *(how do you manipulate signals numerically?)*
  - **radiometry & light transport** *(how does light behave?)*
  - **perception** *(how does this all relate to humans?)*
  - **…**
- **Systems**
  - **parallel, heterogeneous processing**
  - **graphics-specific programming languages**
  - **…**

建模与画出一个立方体

两个问题：

- **如何建模**：如何用数字信息描述图形
- **如何渲染**：数字信息转化为图形

建模

ACTIVITY: modeling the cube

- Suppose our cube is...
  - centered at the origin (0,0,0)
  - has dimensions 2x2x2
  - edges are aligned with x/y/z axes
- QUESTION: What are the coordinates of the cube vertices?

```
A: ( 1, 1, 1 )   E: ( 1, 1,-1 )
B: (-1, 1, 1 )   F: (-1, 1,-1 )
C: ( 1,-1, 1 )   G: ( 1,-1,-1 )
D: (-1,-1, 1 )   H: (-1,-1,-1 )
```

- QUESTION: What about the edges?

```
AB, CD, EF, GH,
AC, BD, EG, FH,
AE, CG, BF, DH
```

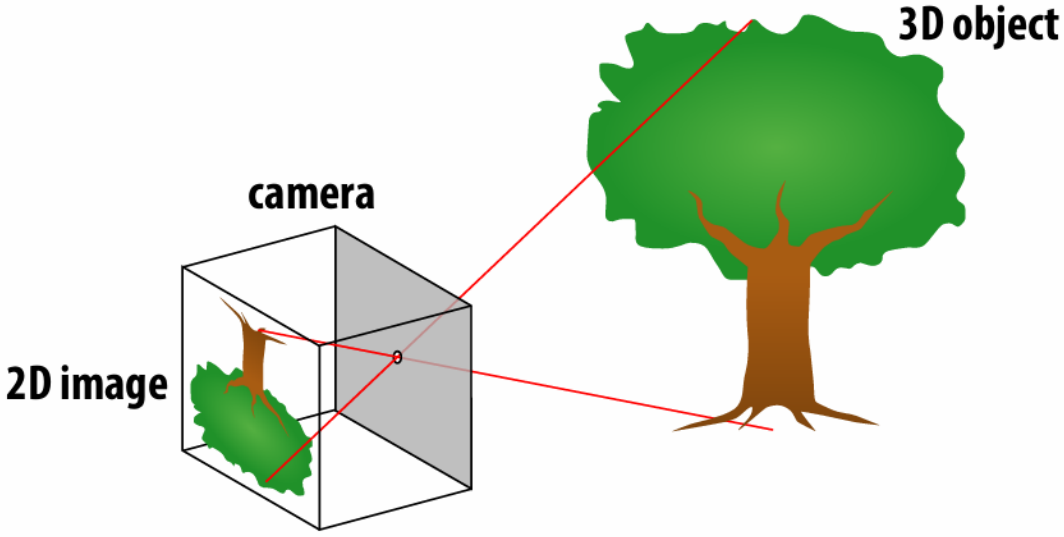
- 第一个假设不能泛化，不如找出顶点坐标与边
  - 同边的两个顶点只有一个坐标在变化

画：Perspective projection

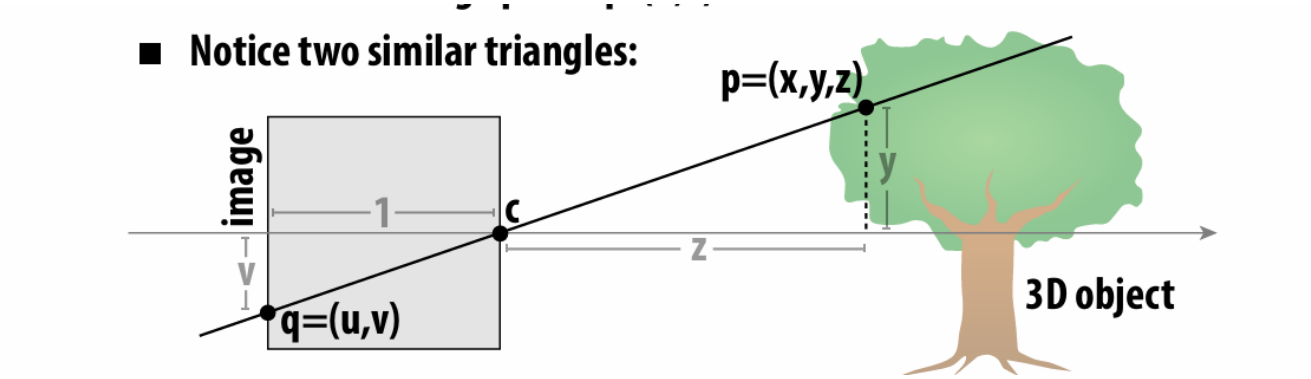
如何把三维物体画到二维上？

- 丢掉一个坐标轴不能获得立体感
  基本策略是将3D点**映射**到2D点上，然后把2D点用**直线**连起来。

我们可以参考相机的工作方式：



假设相机位于远点，从侧面来看：



- v = y/z，再从顶部来看即可得到 u = x/z
- 当z越来越大，2D图形将会越来越小
- 如果相机不位于远点，则需移动整个坐标系

画

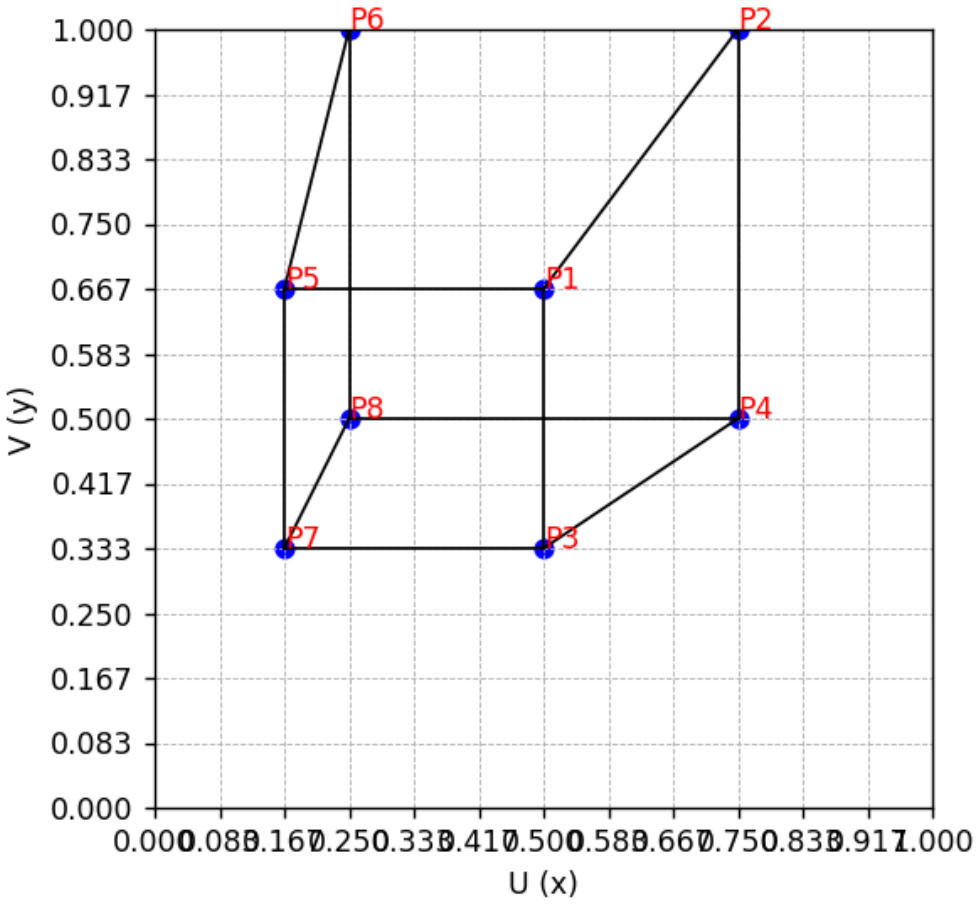
一段简单的代码输出 3D 与 2D 点的映射关系：

- lec1.py

```python
# point: xyz, 从xyz，映射到一个垂直于z轴的平面
def convert(point):
    # x轴方向x`=x/z，y轴方向y`=y/z
    u, v = point[0] / point[2], point[1] / point[2]
    print(point, " -> (", u, v, ")")
    return u, v


camera = (2, 3, 5)
vertices = list(itertools.product([-1, 1], repeat=3))
# 平移
vertices = [(x - camera[0], y - camera[1], z - camera[2]) for x, y, z in vertices]
# 投影
projected_vertices = [convert(vertice) for vertice in vertices]

u_vals, v_vals = zip(*projected_vertices)
draw(u_vals, v_vals)
```



> 画图部分的代码用到了GPT

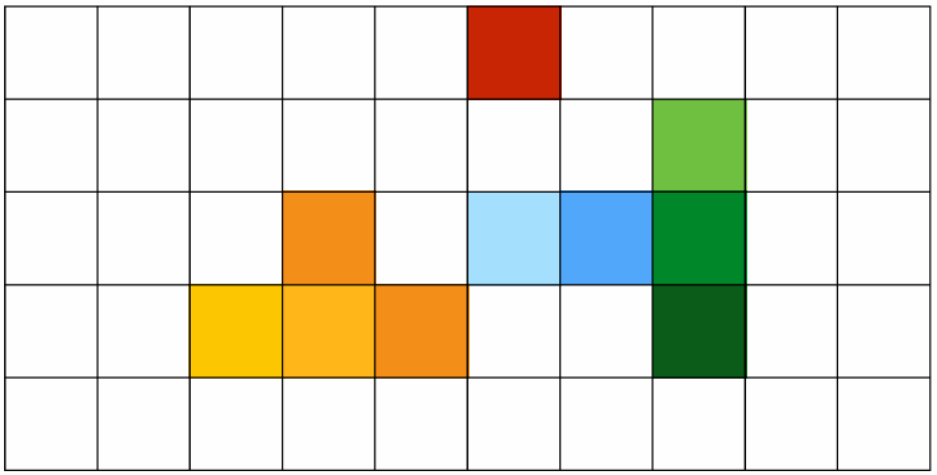
Pixels & Raster Display

离散的像素

如何在电脑上画线？**非连续**的像素 **pixels**（picture elements）

基于像素，我们可以把图形光栅化 **raster** 到一个网格 **grid** 中，每一个网格的值是该像素的 **RGB** 值

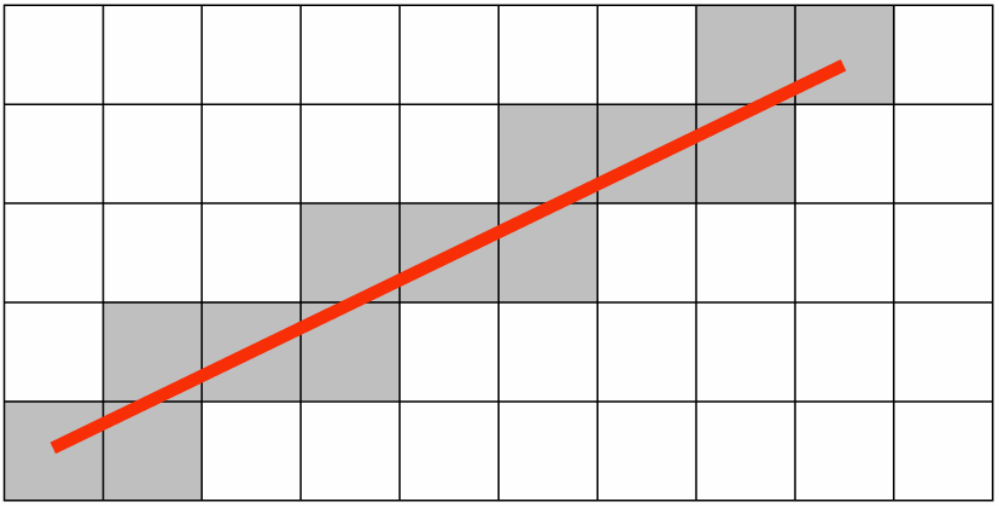
- 光栅化即离散化，我们的网格是**不连续**的，而实际图形是连续的



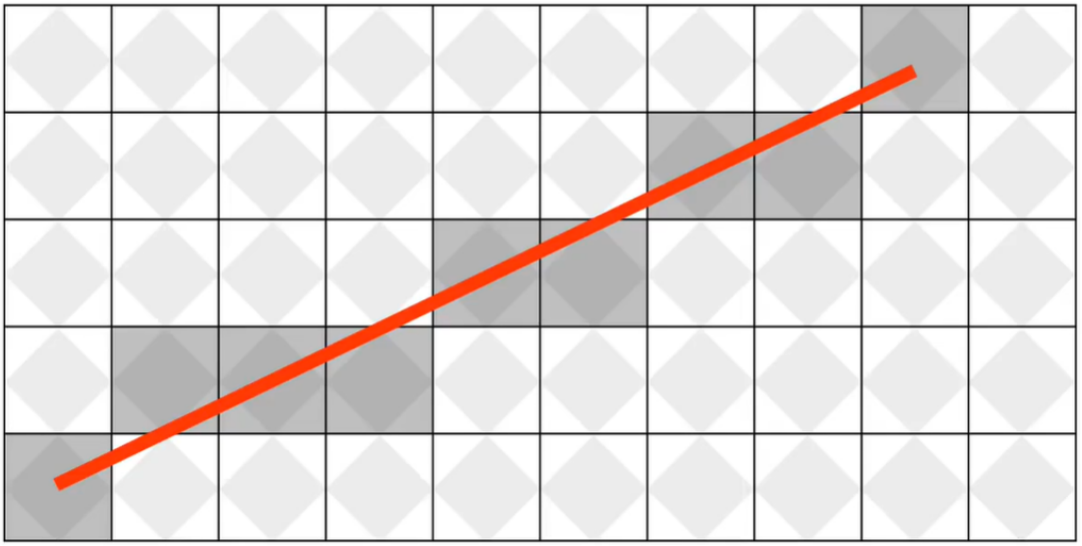
钻石规则

在光栅化一条线段时，我们可以：

- 把任何触及触及线段的网格绘出



- **钻石规则**：只绘出通过菱形的方格



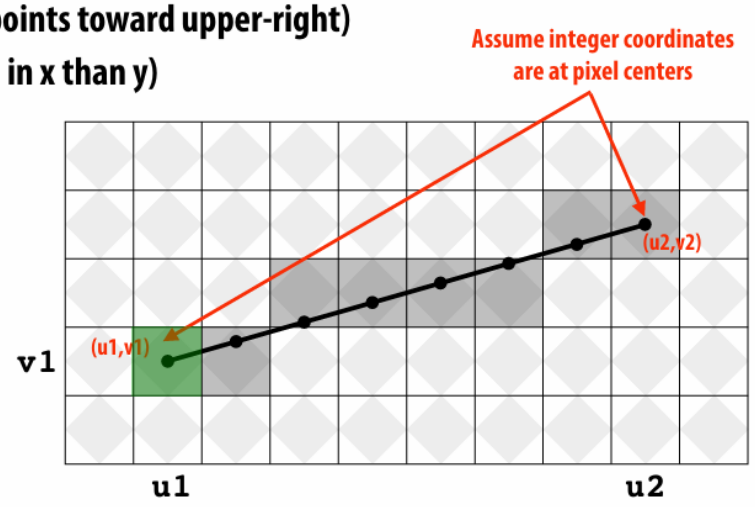
- 需要**综合考虑**各种条件给出一个好的标准，例如你需要考虑线的厚度。做事情的方法不止一种

增量光栅化

- 考虑每个像素，假设显示器的像素个数为nxn，则复杂度为 $O(n^2)$
- 然而预计中，一条线段最多覆盖 $O(n)$ 个元素

- Let's say a line is represented with integer endpoints: (u1,v1), (u2,v2)
- Slope of line: s = (v2-v1) / (u2-u1)
- Consider an easy special case:
  - u1 < u2, v1 < v2 (line points toward upper-right)
  - 0 < s < 1 (more change in x than y)

```
v = v1;
for(u=u1; u<=u2; u++)
{
    v += s;
    draw(u, round(v))
}
```



Easy to implement… not how lines are drawn in modern software/hardware!