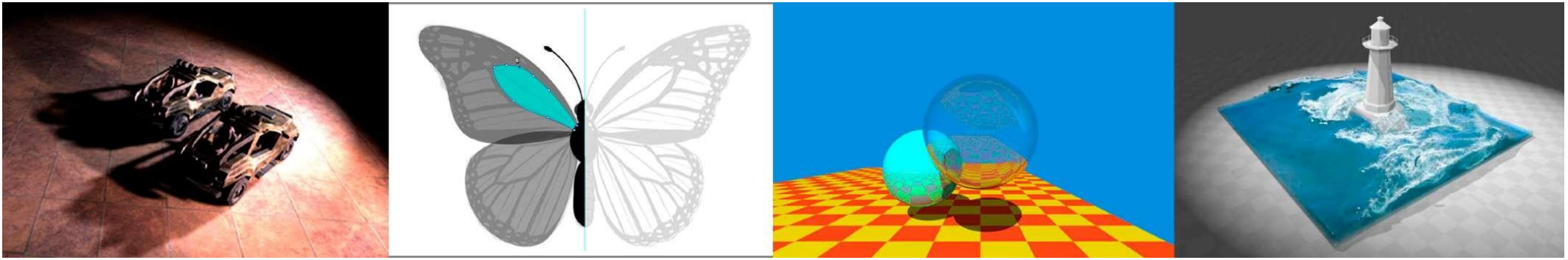
Computer Graphics

Rasterization 1
(Line Segment, Triangles)



Last Lecture

- Curves
  - Bezier curves
  - De Casteljau's algorithm （德卡斯特里奥算法）
  - Bernstein polynomial

    B-Spline (B样条)

    Bézier Surfaces
- Mesh Operations: Geometry Processing
  - Mesh subdivision (网格细分)
  - Mesh simplification （网格简化）
  - Mesh regularization（网格正则化）

Bézier Curve – General Algebraic Formula

Bernstein form of a Bézier curve of order n:

$$\mathbf{b}^n(t) = \mathbf{b}_0^n(t) = \sum_{j=0}^{n} \mathbf{b}_j B_j^n(t)$$

Bézier curve order n

(vector polynomial of degree n)

Bernstein polynomial

(scalar polynomial of degree n)

Bézier control points

(vector in R^N)

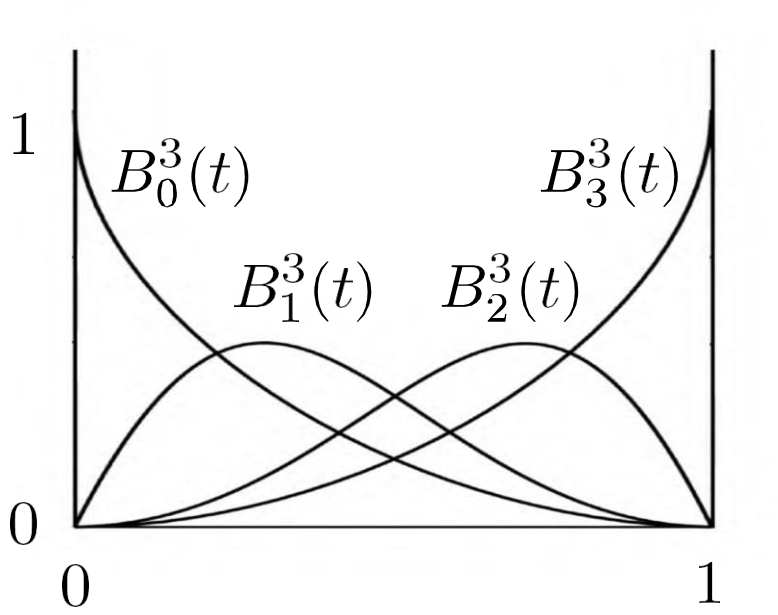
Bernstein polynomials:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

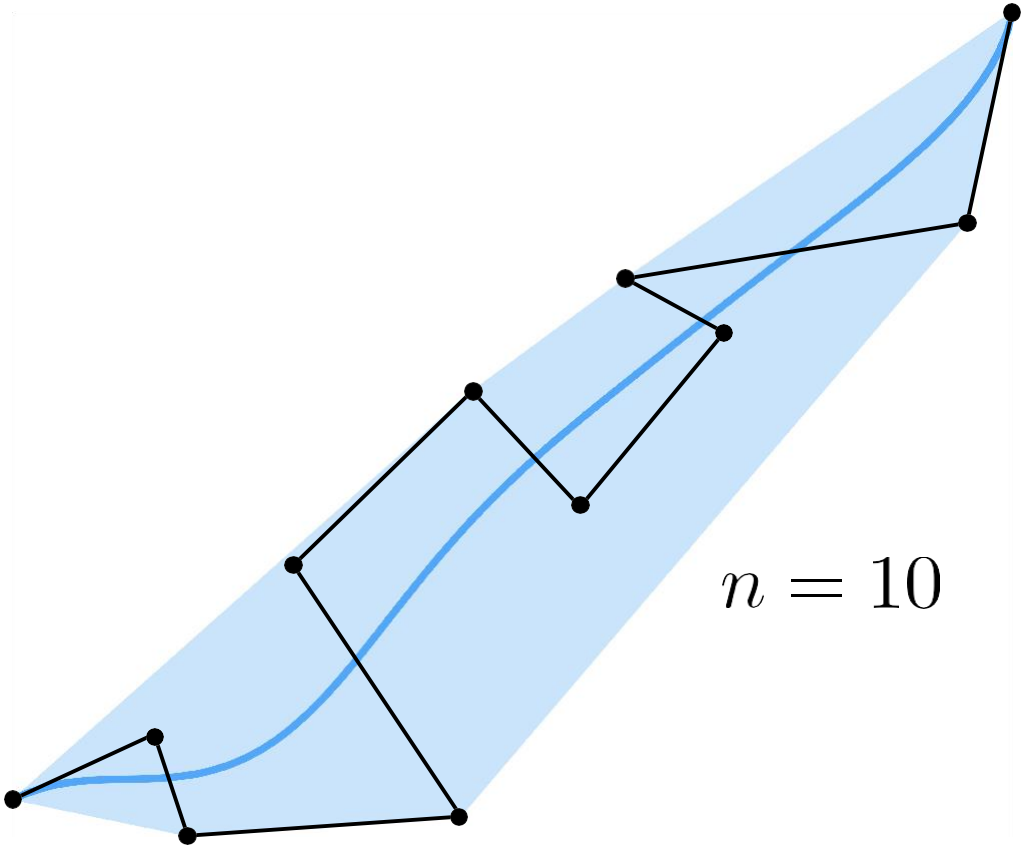
$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Bézier Curve – General Algebraic Formula

- 非负性（Non-negative）
- 端点（End point）
- 归一性（Unity）
- 对称性（Symmetry）
- 递归性（Recursive）



Higher-Order Bézier Curves?



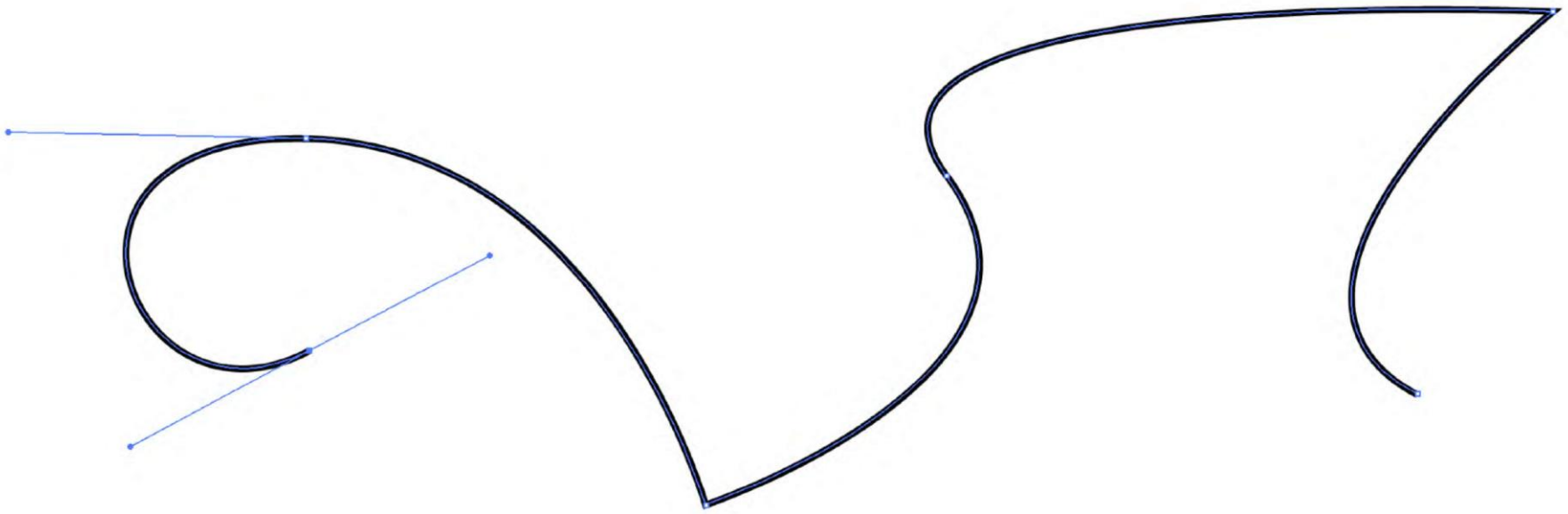
Very hard to control!

Uncommon

Piecewise Bézier Curves

Instead, chain many low-order Bézier curve

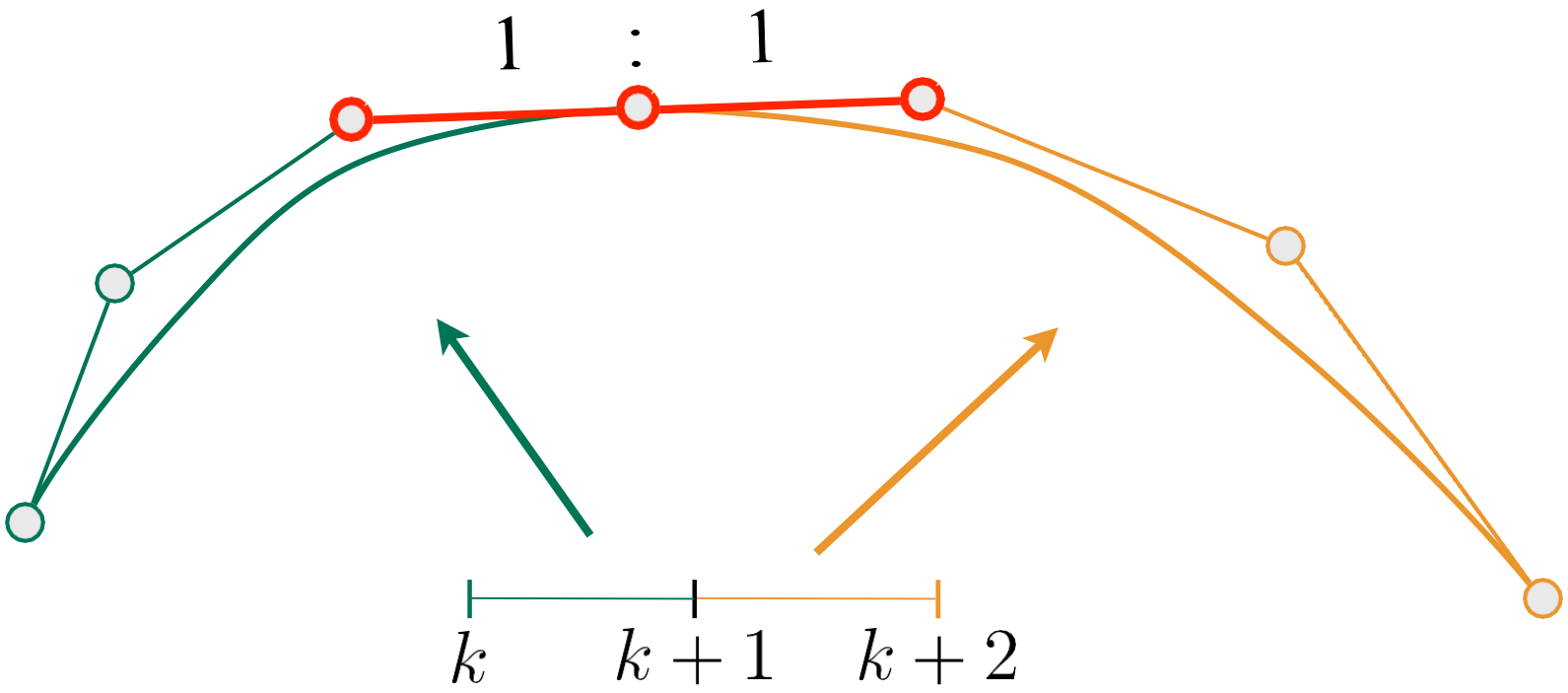
Piecewise cubic Bézier (分段三次贝塞尔) the most common



Widely used (fonts, paths, Illustrator, Keynote, …)

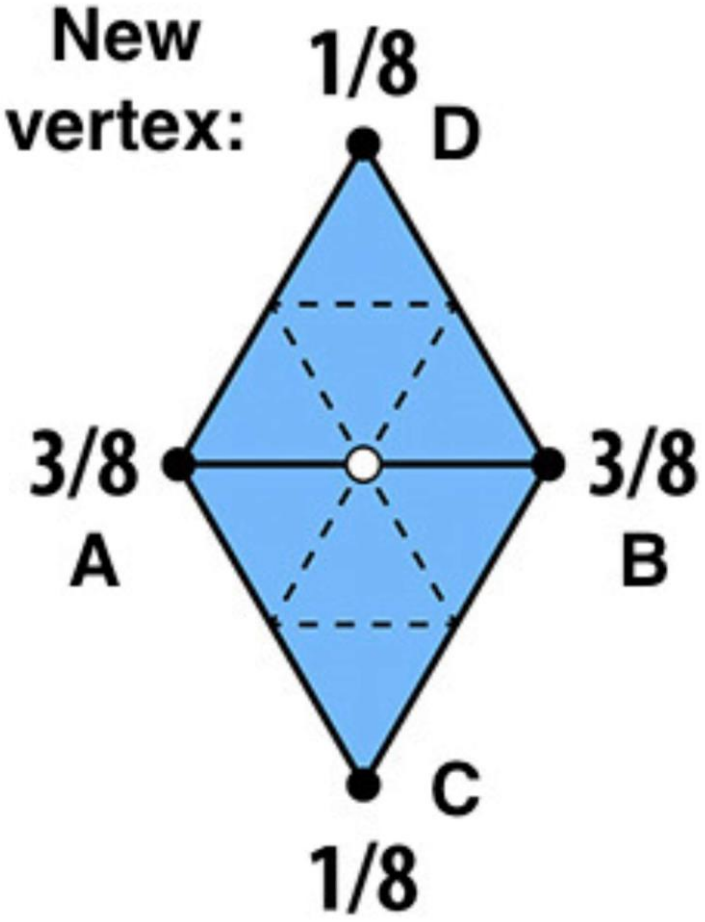
Piecewise Bézier Curve – Continuity

C^1 continuity: $$\mathbf{a}_n = \mathbf{b}_0 = \frac{1}{2}\left(\mathbf{a}_{n-1} + \mathbf{b}_1\right)$$



Loop Subdivision — Update

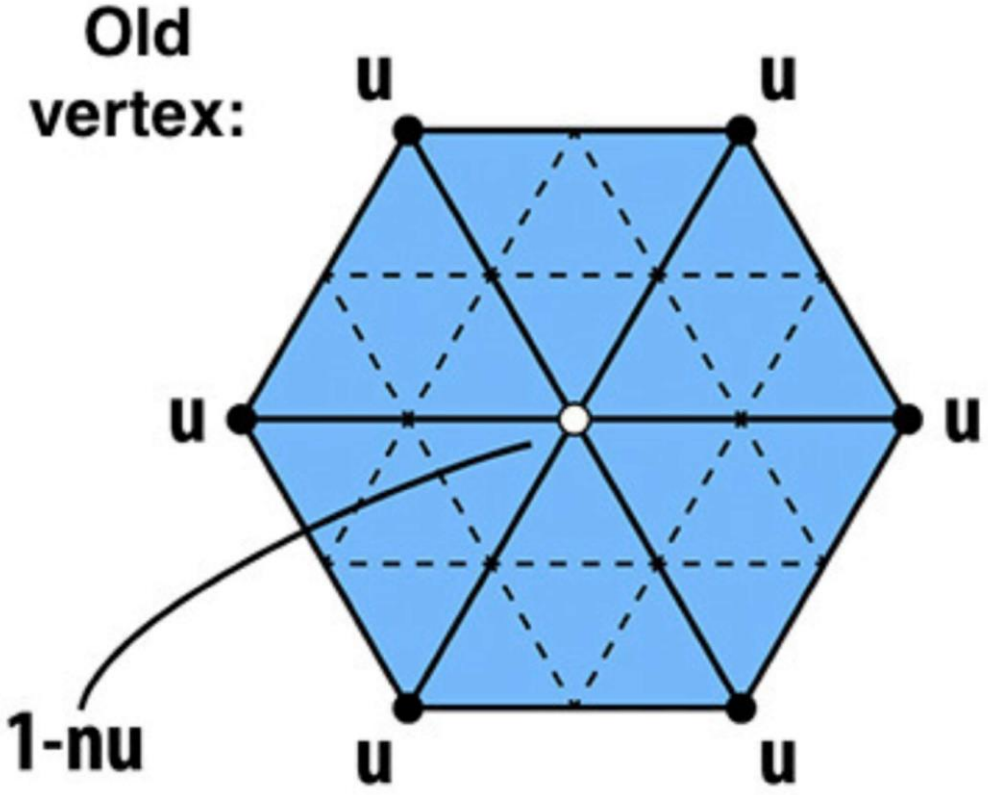
For new vertices:



Update to:
3/8 * (A + B) + 1/8 * (C + D)

Loop Subdivision — Update

For old vertices (e.g. degree 6 vertices here):



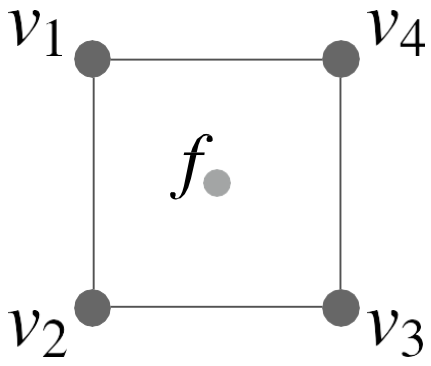
Update to:
(1 - n*u) * original_position +
u * neighbor_position_sum

n: vertex degree

u: 3/16 if n=3, 3/(8n) otherwise

FYI: Catmull-Clark Vertex Update Rules (Quad Mesh)

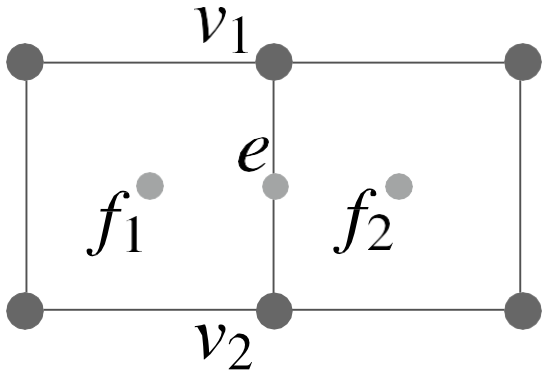
Face point

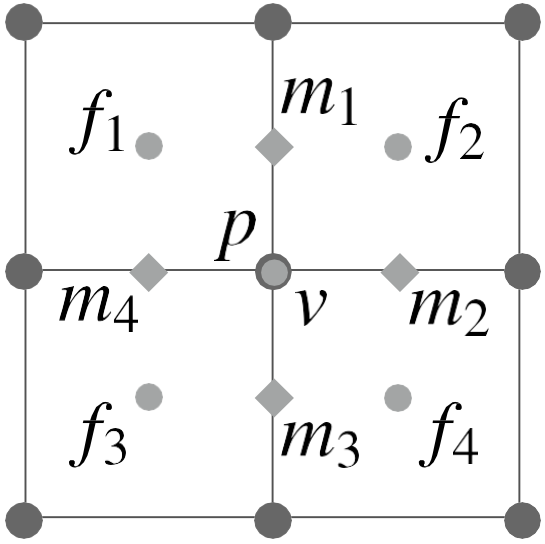


$$f = \frac{v_1 + v_2 + v_3 + v_4}{4}$$

Edge point

$$e = \frac{v_1 + v_2 + f_1 + f_2}{4}$$





Vertex point

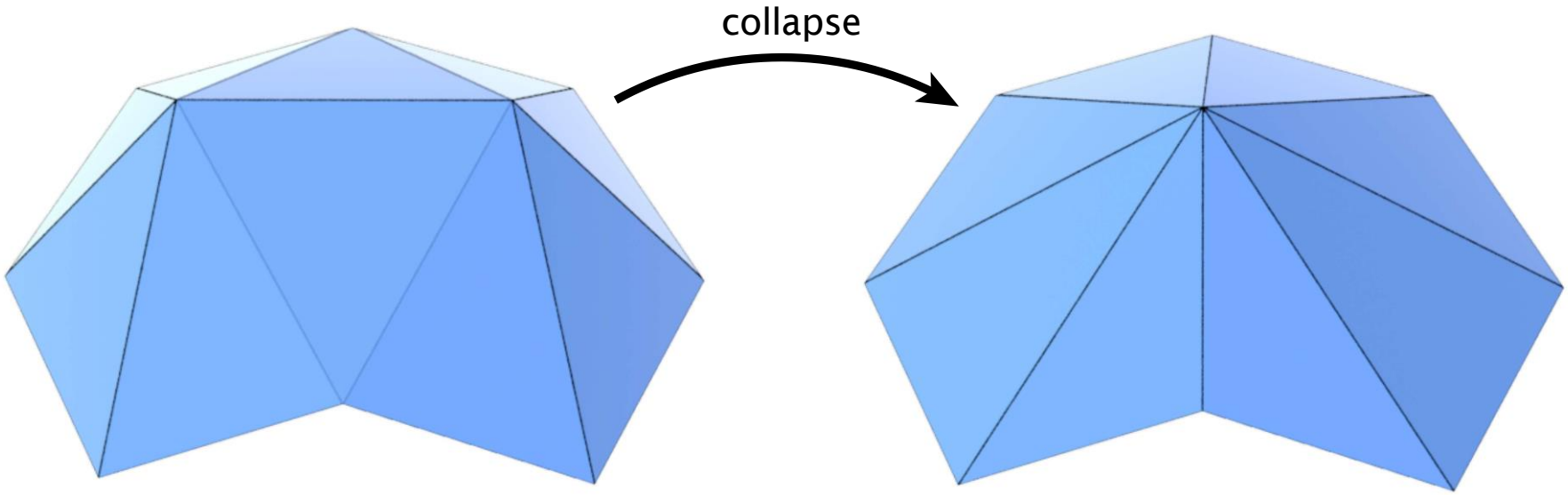
$$v = \frac{f_1 + f_2 + f_3 + f_4 + 2(m_1 + m_2 + m_3 + m_4) + 4p}{16}$$

m midpoint of edge

p old “vertex point”

Collapsing An Edge

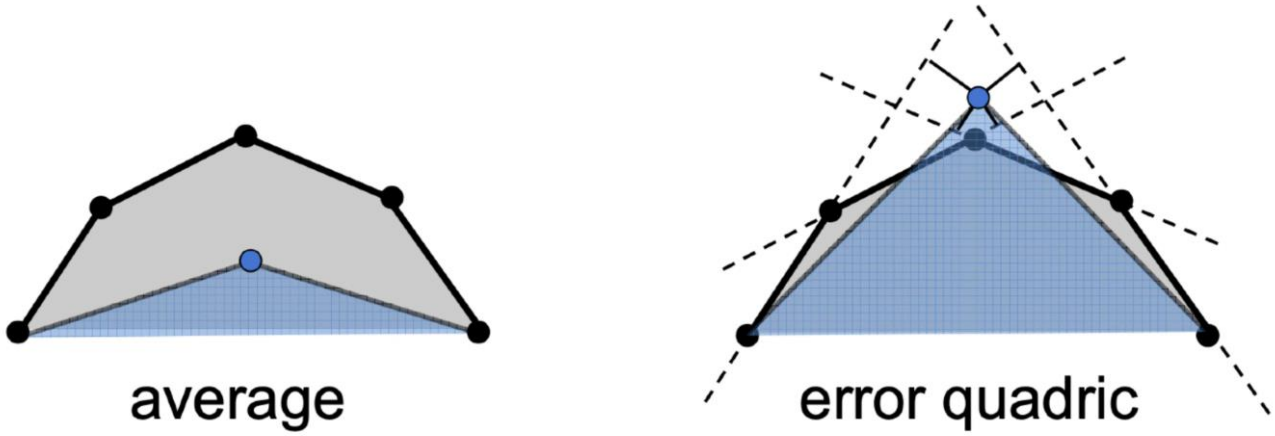
- Suppose we simplify a mesh using edge collapsing



Quadric Error Metrics

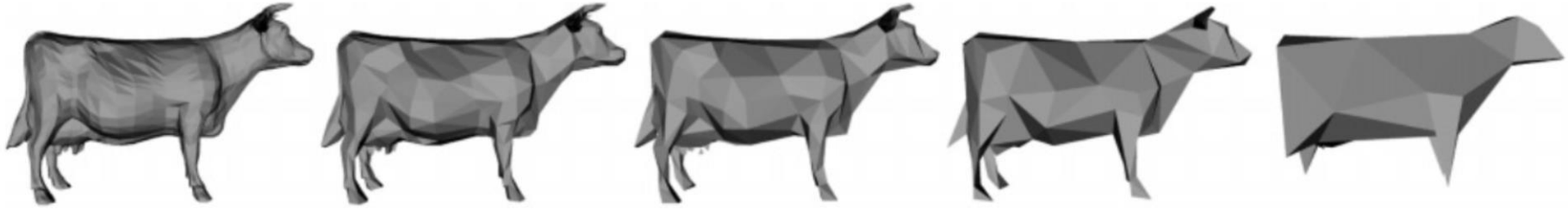
（二次误差度量）

- How much geometric error is introduced by simplification?
- Not a good idea to perform local averaging of vertices
- Quadric error: new vertex should minimize its sum of square distance (L2 distance) to previously related triangle planes!



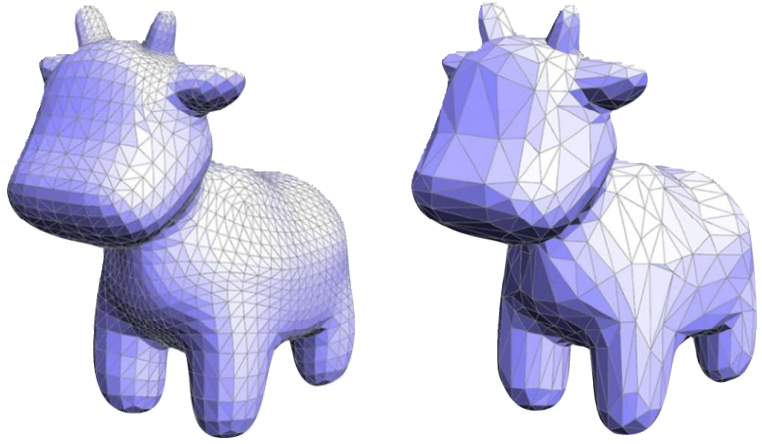
http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/08_Simplification.pdf

Quadric Error Mesh Simplification



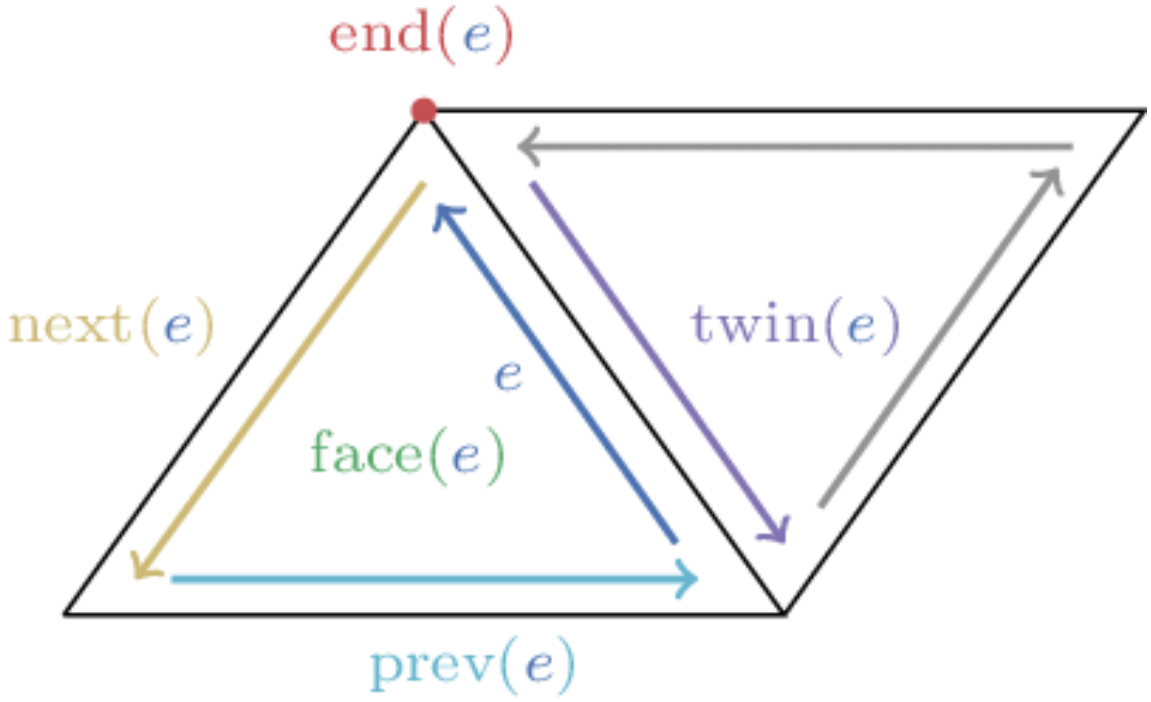
5,804 994 532 248 64

Garland and Heckbert '97



设置

请讲解一下Half-edge data structure?



https://jerryyin.info/geometry-processing-algorithms/half-edge/

作答

单选题 1分

设置

二维平面内，满足C1连续的2段三次贝塞尔曲线，控制顶点的自由度共有几个?

A 10

B 12

C 14

D 16

提交

单选题 1分

设置

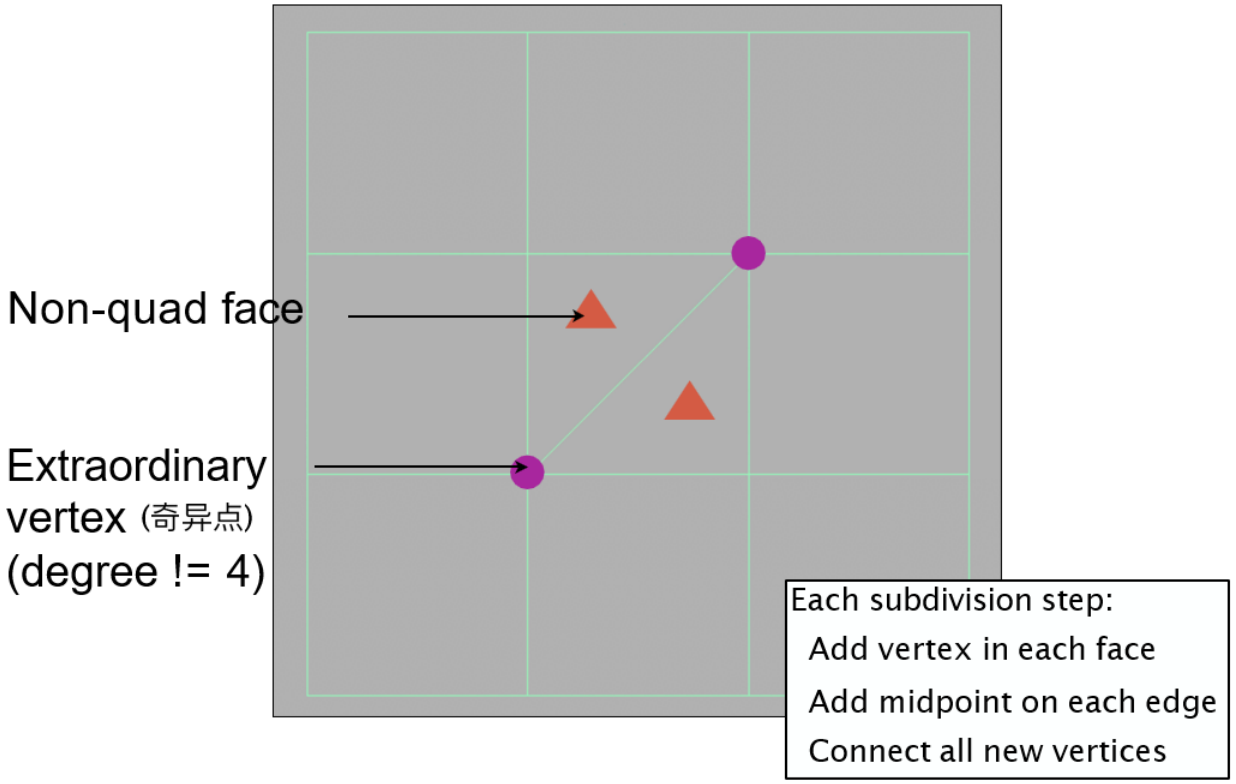
请问这个图形经过Catmull-Clark细分的一次迭代，奇异点数量是?

A 2

B 3

C 4

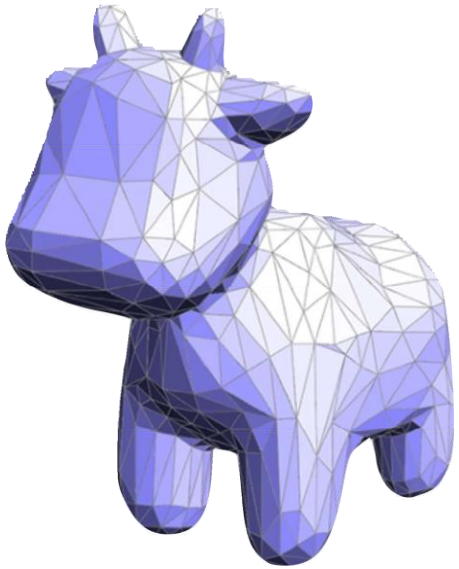
D 6



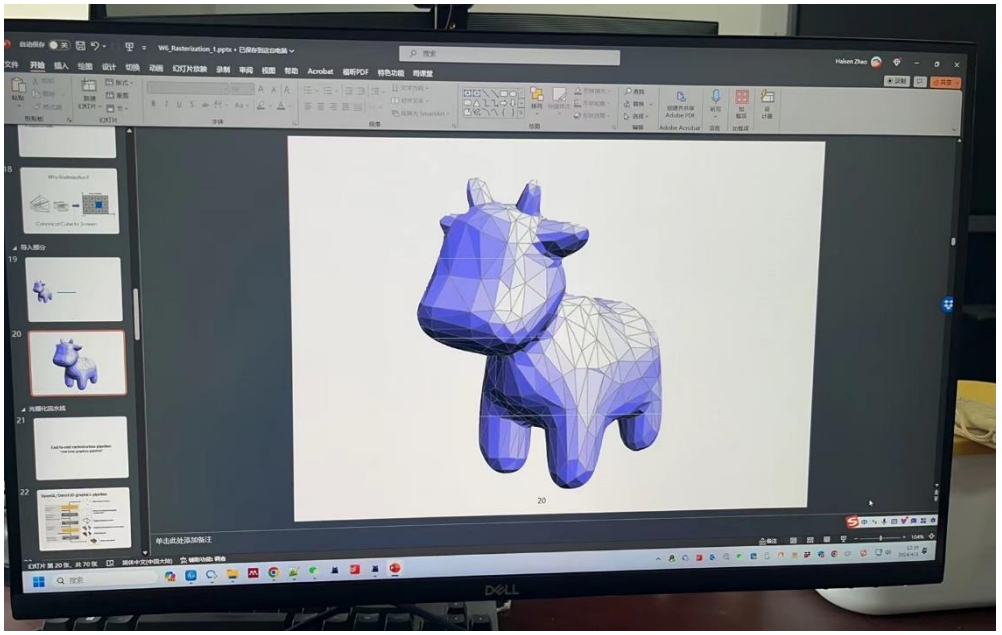
提交

Today

- Fill the gap between geometry to screen
  - Raster Displays
- Rasterization pipeline
- Rasterization (Line and Triangle)
- Antialiasing and Z-Buffering
- Next week: Clipping



?



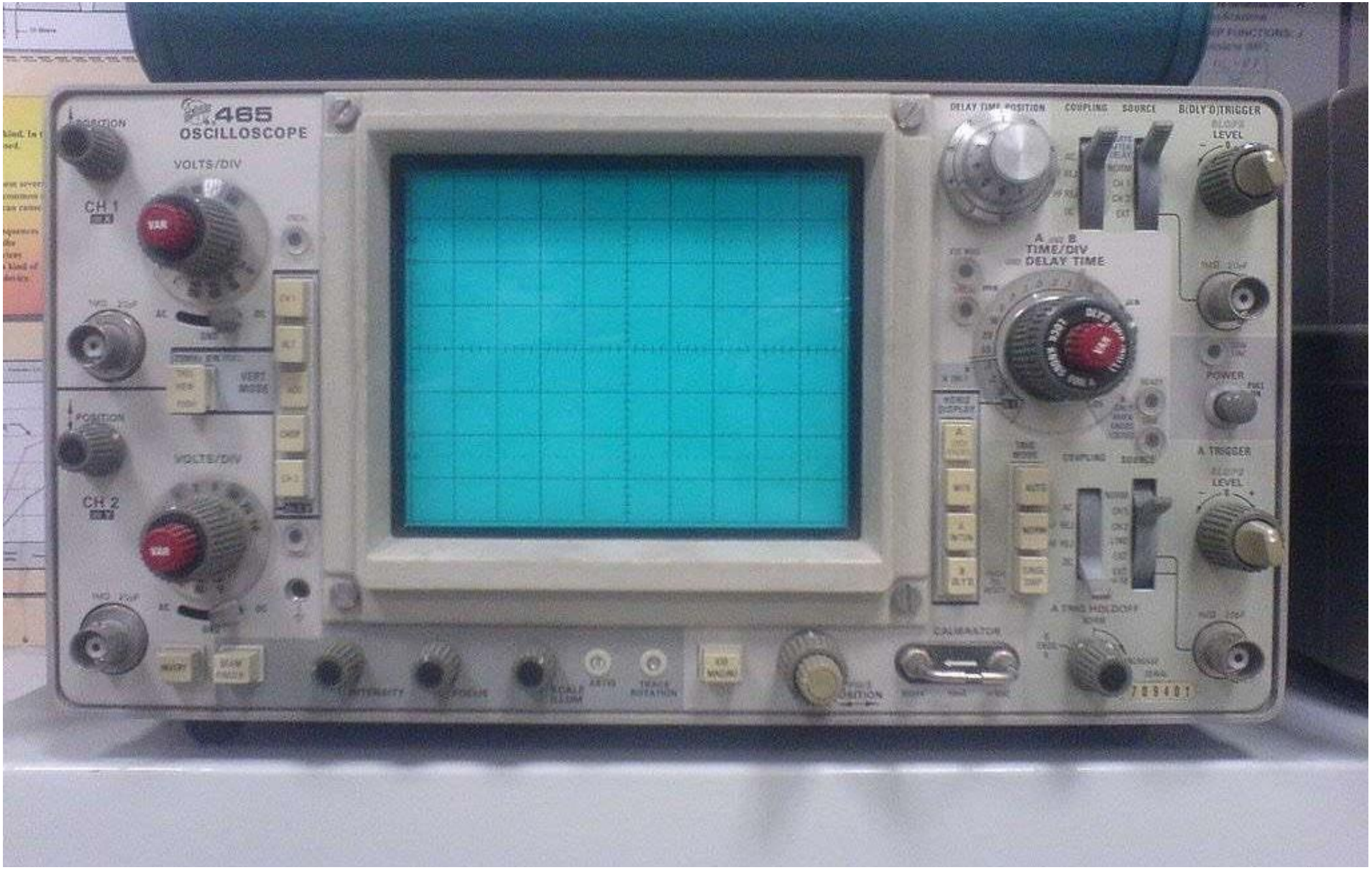
What is screen?

—Different Raster Displays

What is screen?

- What is a screen?
  - An array of pixels
  - Size of the array: resolution
  - A typical kind of raster display
- Raster == screen in German
  - Rasterize == drawing onto the screen
- Pixel (FYI, short for “picture element”)
  - For now: A pixel is a little square with uniform color
  - Color is a mixture of (red, green, blue)

Oscilloscope 示波器



https://www.bilibili.com/video/BV1K44y147LK/?spm_id_from=333.337.search-card.all.click&vd_source=c9ca290d3dbf1895a1a4e9e33b4c524a

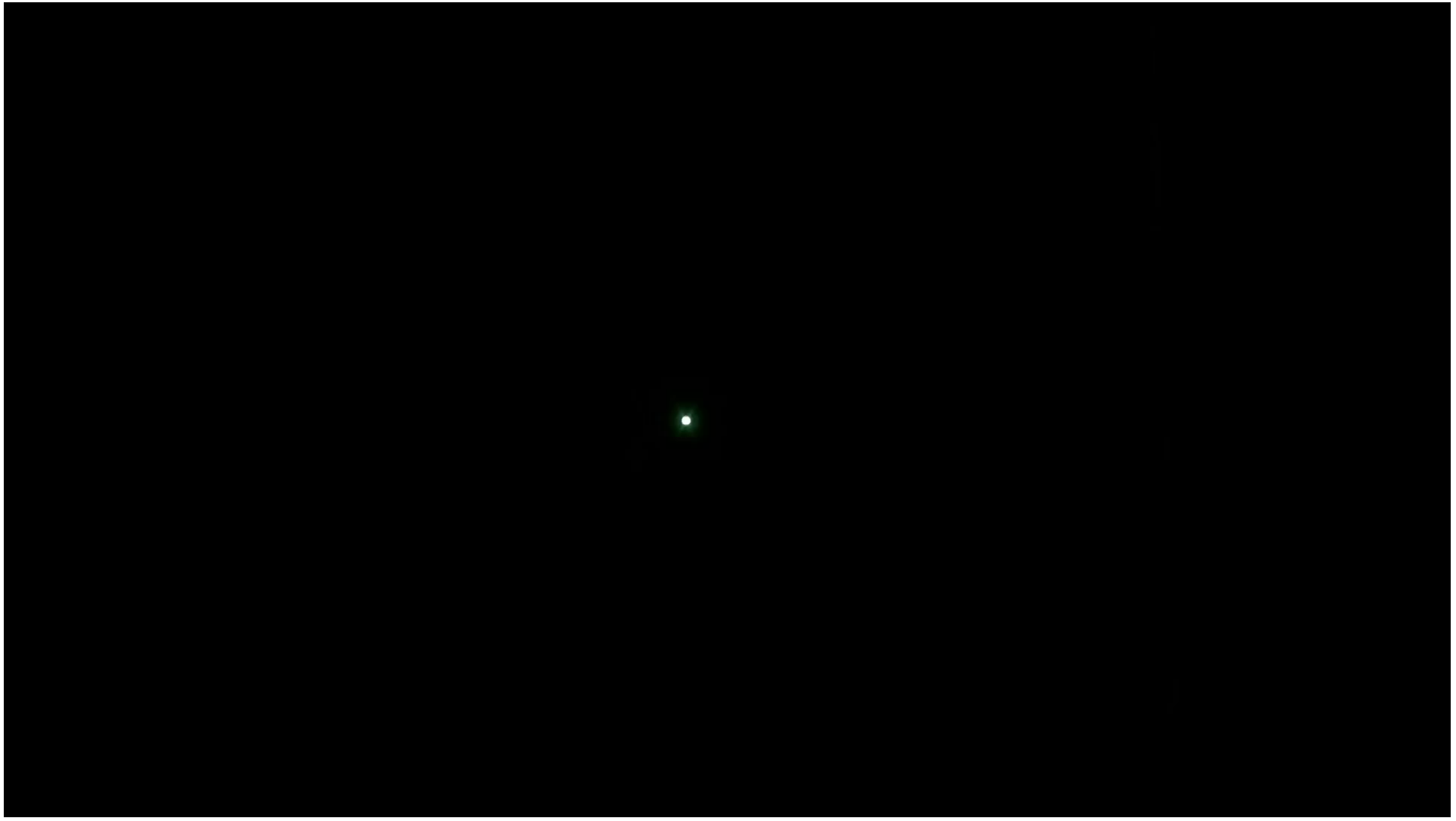

GW INSTEK GDS-2302A
示波器是甚麼
講給你聽

Oscilloscope Art

B. Laposky

“Electronic Abstractions”
1953

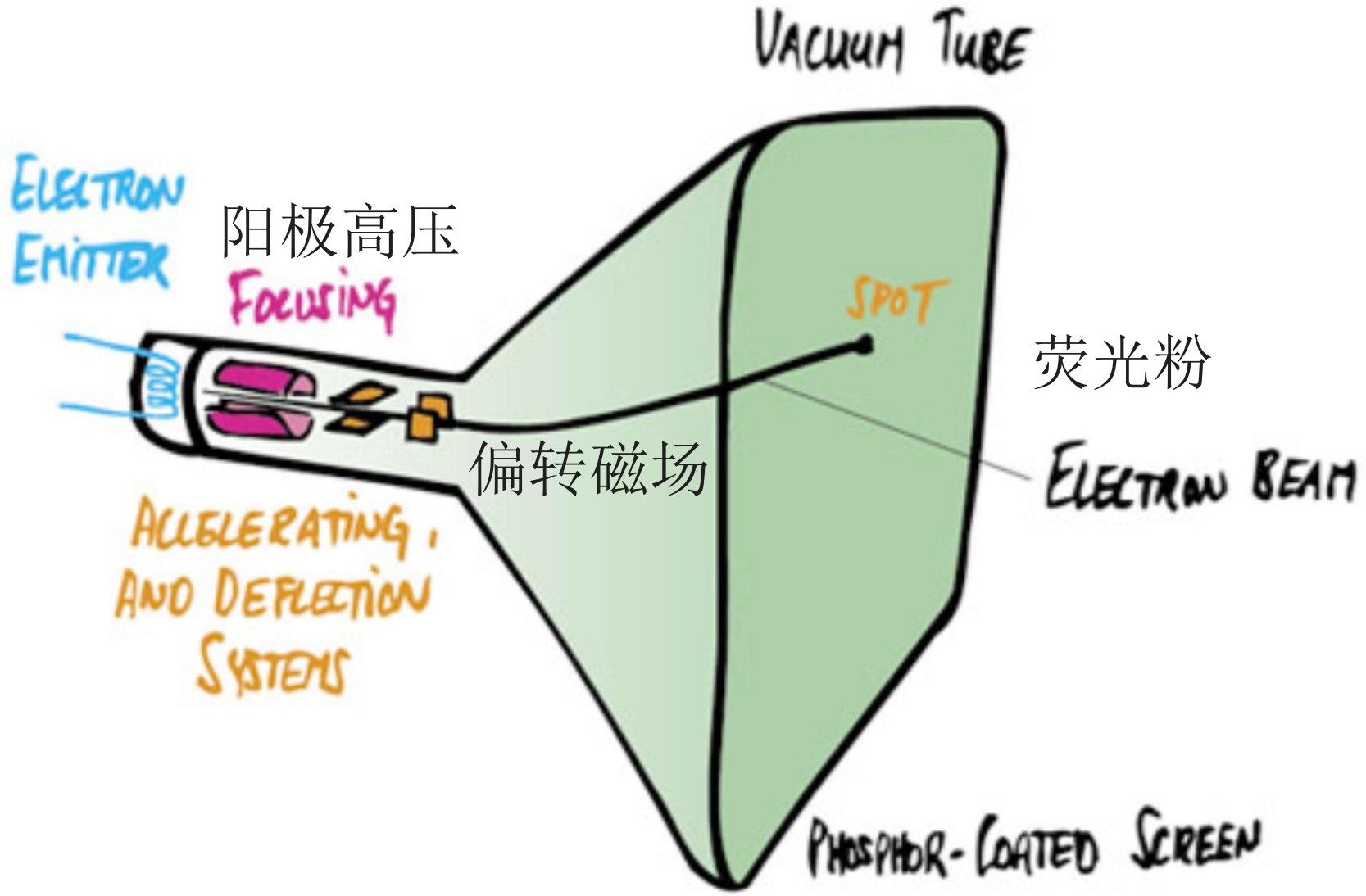
Oscilloscope Art



”How To Draw Mushrooms On An Oscilloscope With Sound”

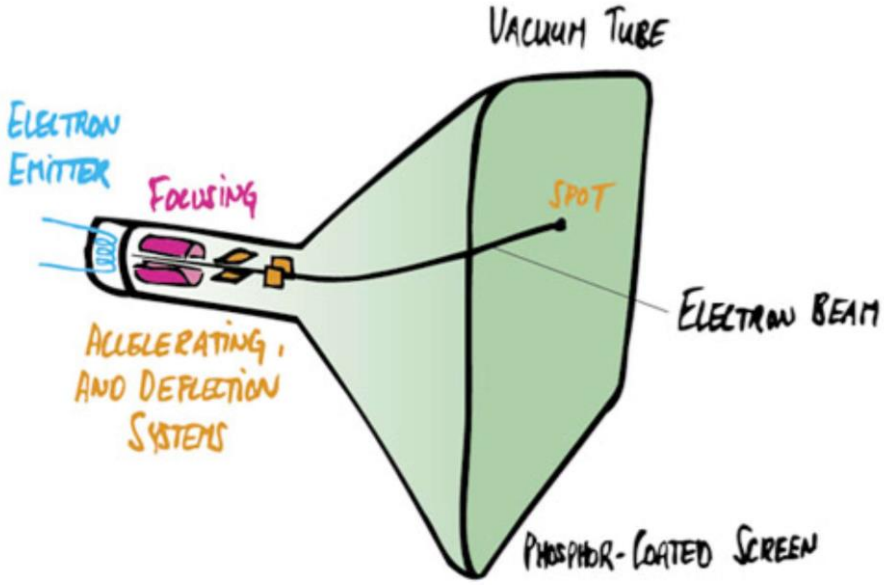
https://www.youtube.com/watch?v=rtR63-ecUNo

Cathode Ray Tube （阴极射线管）

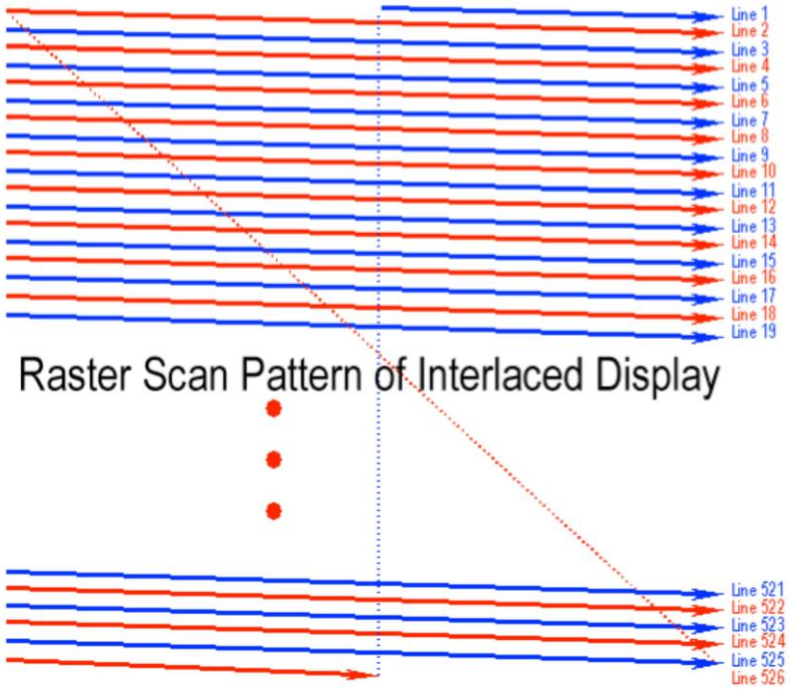


Television - Raster Display CRT

光栅显示 CRT



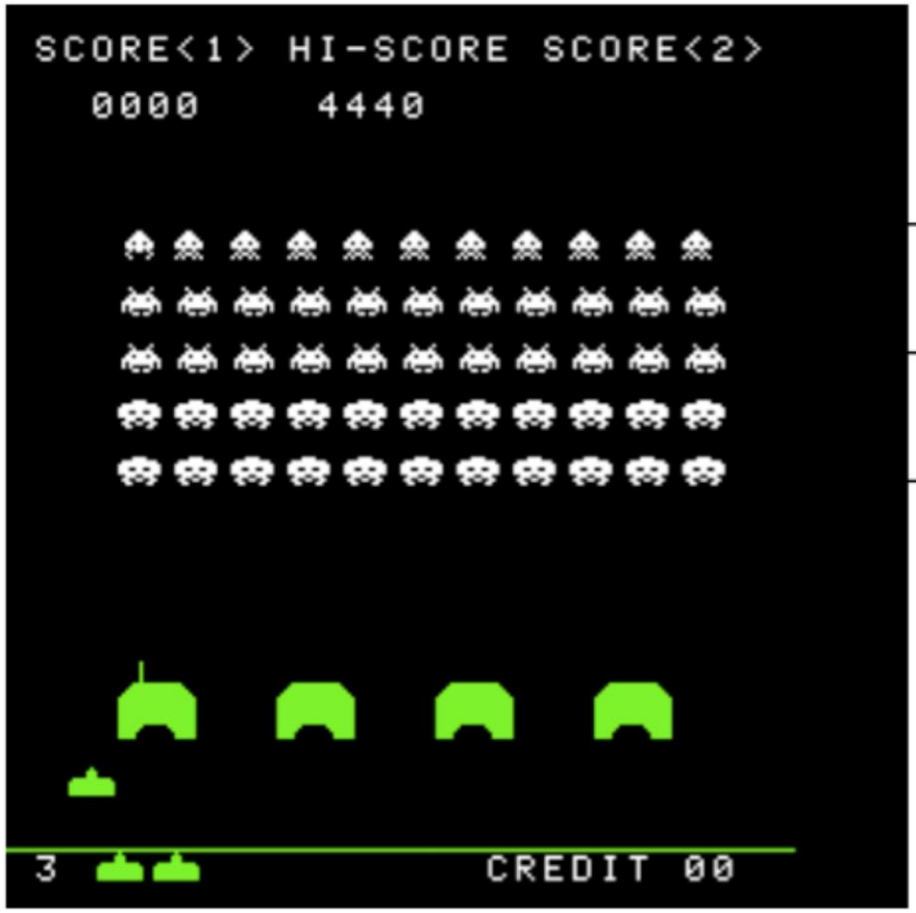
Cathode Ray Tube



Raster Scan
(modulate intensity)

Frame Buffer: Memory for a Raster Display

帧缓冲区：光栅显示的内存



DAC =
Digital to Analog Convertors

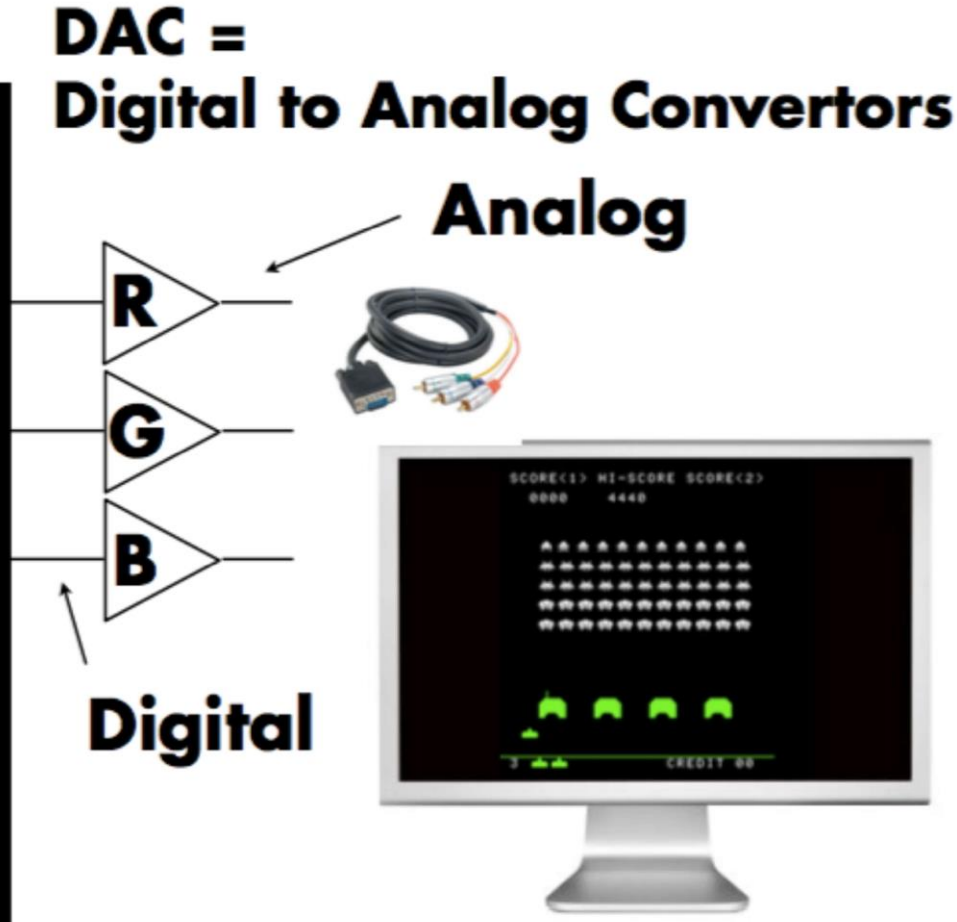


Image = 2D array of colors

Flat Panel Displays

Low-Res LCD Display

液晶显示器

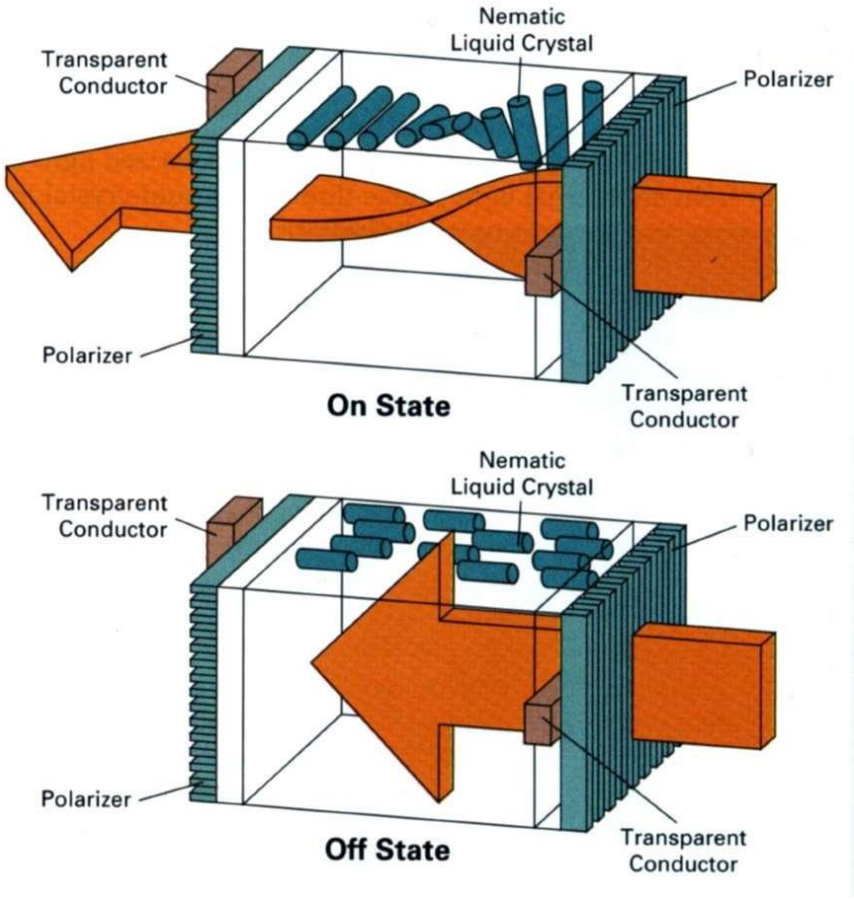
Color LCD, OLED, …

LCD (Liquid Crystal Display) Pixel

原理：通过扭曲偏振阻挡或透射光

背光照明（例如荧光灯或 LED）

通过部分扭转实现中等强度水平



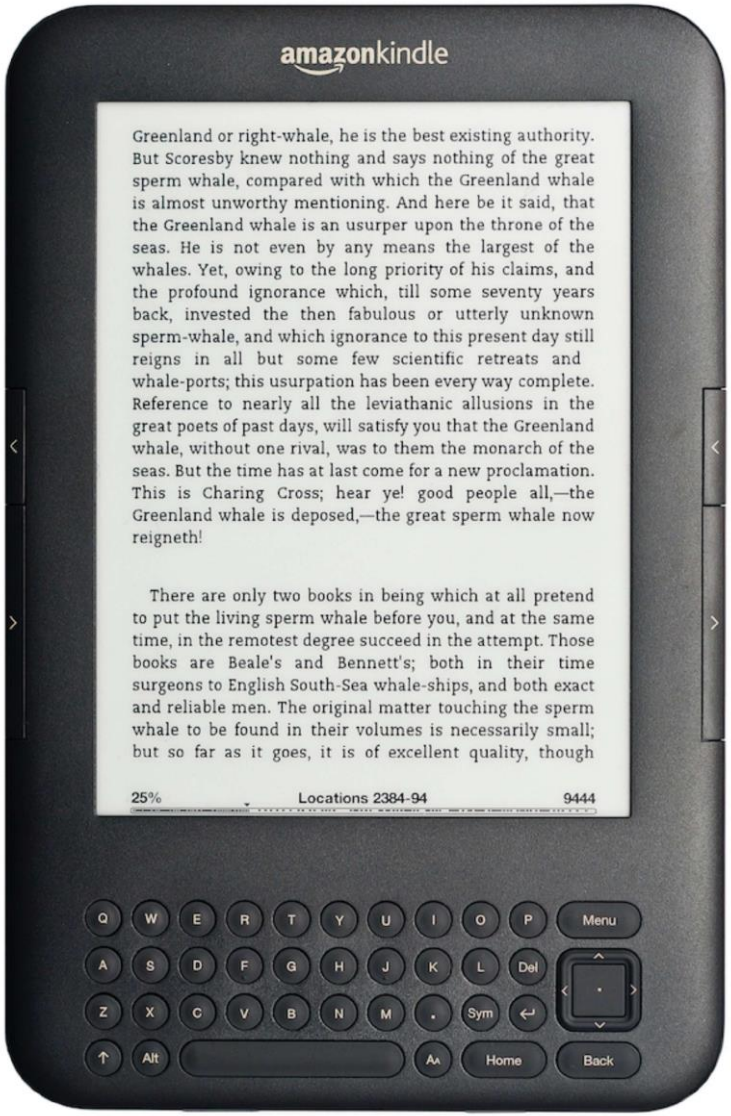
[H&B fig. 2-16]

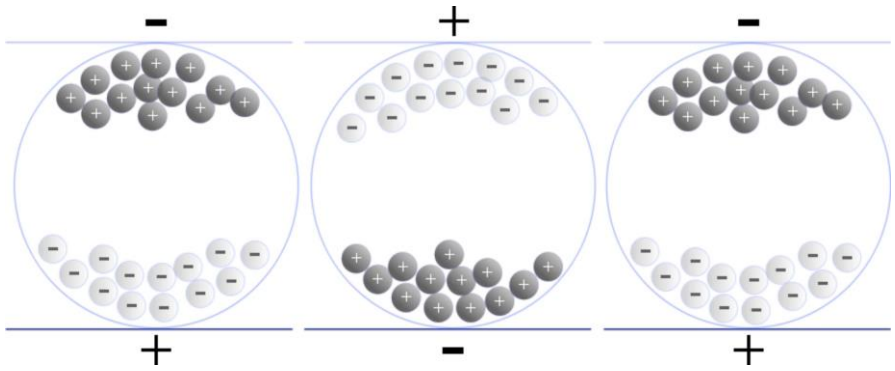
LED Array Display

Light emitting diode array

Electrophoretic (Electronic Ink) Display

电泳（电子墨水）显示器

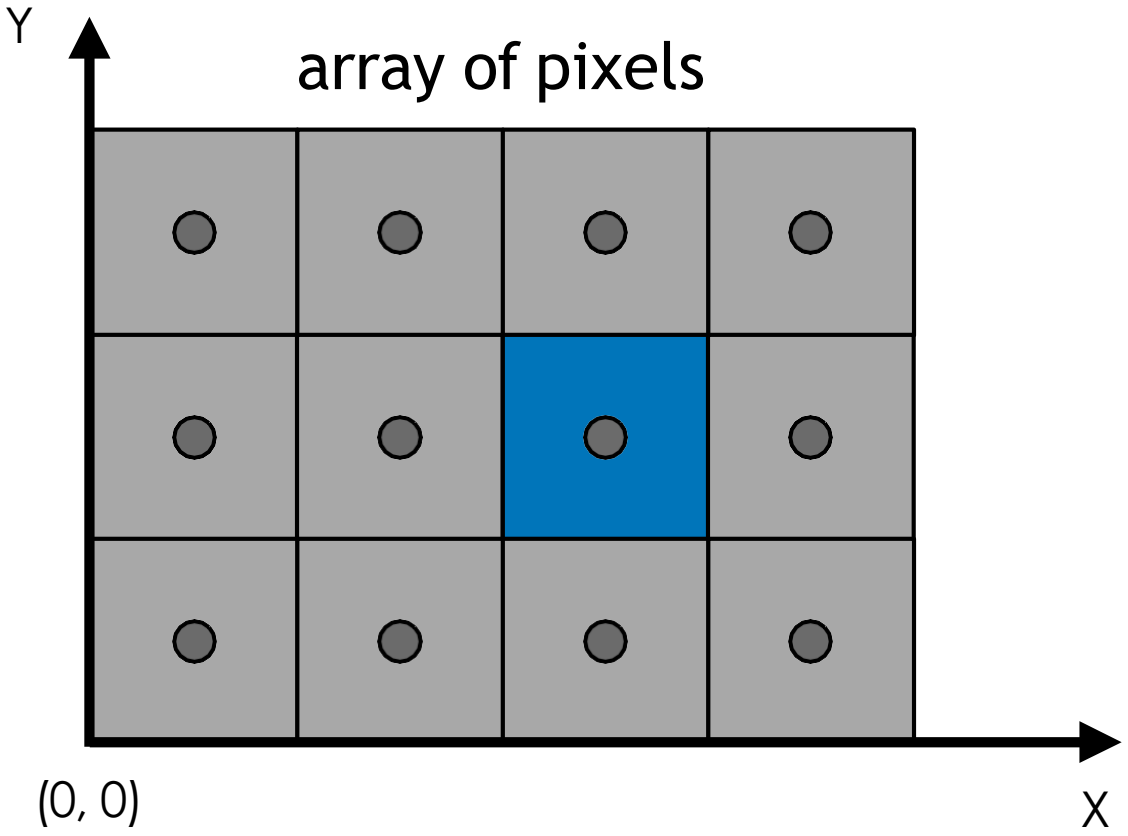




[Wikimedia Commons —Senarclens]

Screen space formulation

- Defining the screen space
  - Slightly different from the "tiger book"



Pixels' indices are in the form of (x, y), where both x and y are integers

Pixels' indices are from (0, 0) to (width - 1, height - 1)

Pixel (x, y) is centered at (x + 0.5, y + 0.5)

The screen covers range (0, 0) to (width, height)

非光栅化显示器？

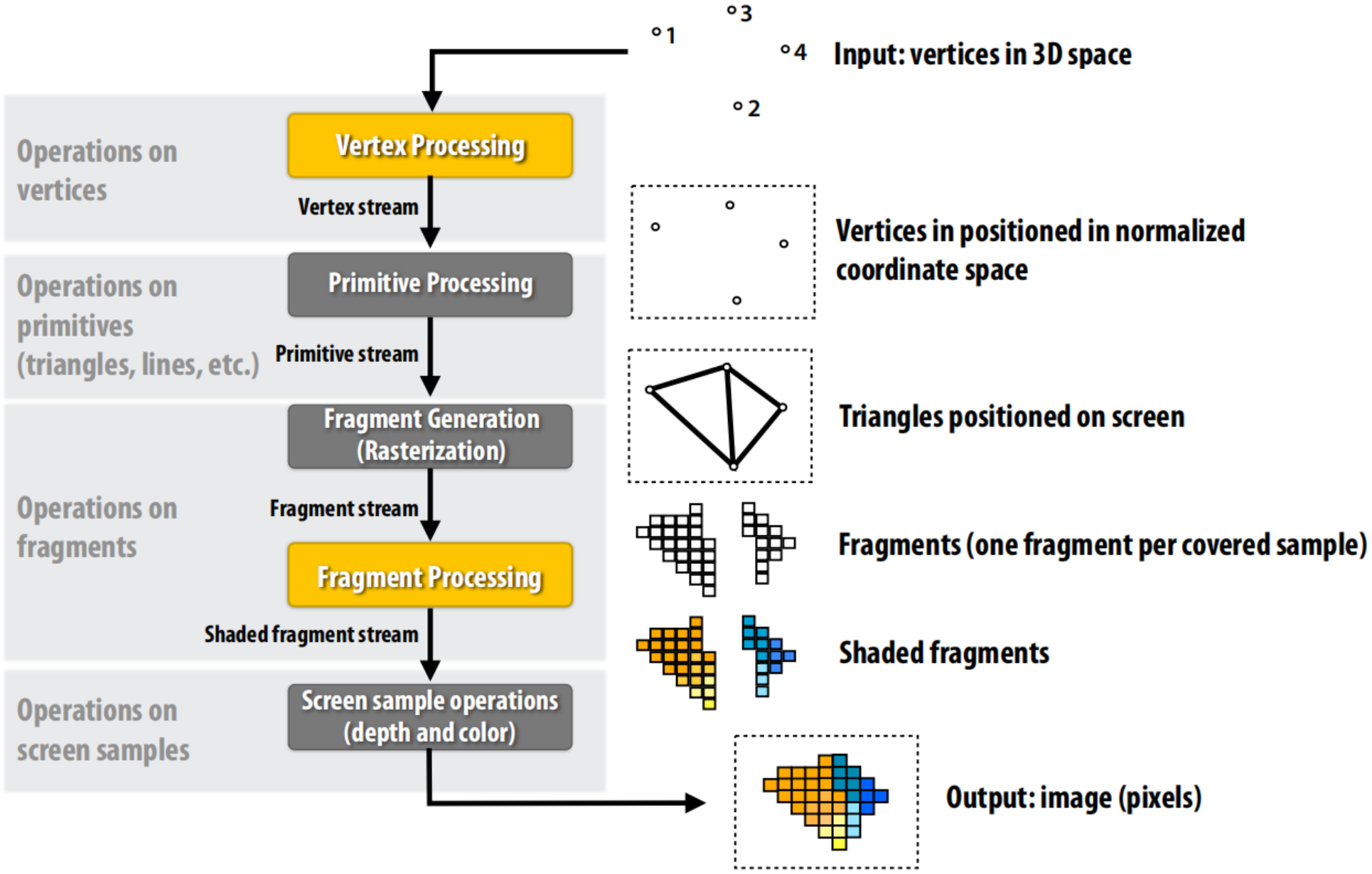
1.矢量显示器： 这种显示器使用直线、曲线和其他几何形状的描述来显示图像，而不是使用像素。矢量显示器可以显示更清晰的线条和图形，但是对于复杂图像和高分辨率显示来说，通常不如光栅化显示器效果好。矢量显示器已经相对较少使用，主要在过去的计算机和绘图设备中出现。

2.DLP（数字光处理）投影： 数字光处理技术使用微小的微镜片（Digital Micromirror Device，DMD）来控制光的方向和亮度，从而形成图像。DLP投影仪可以将图像投射到屏幕或其他平面上，而不是直接在显示器上显示。虽然它们不是直接的光栅化显示器，但在显示图像时也使用像素化的概念，只不过是通过光学方法而非电子显示来实现的。

End-to-end rasterization pipeline

“real-time graphics pipeline”

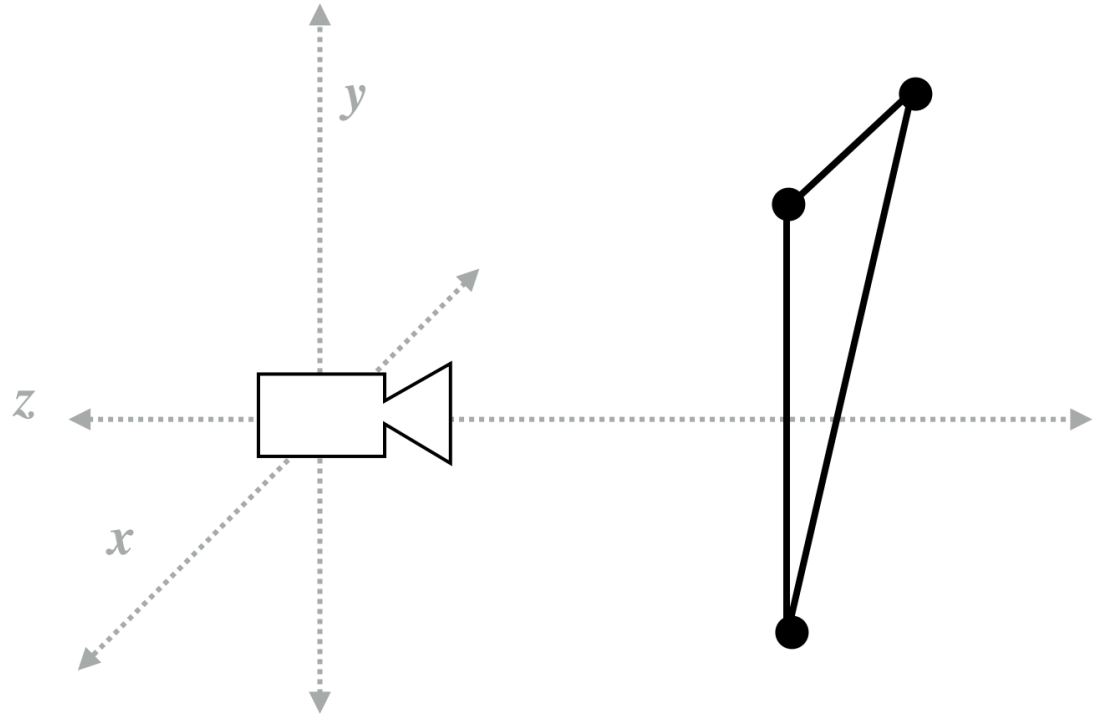
OpenGL/Direct3D graphics pipeline



Step 1 (Vertex Processing): model-view transformation and camera transformation

Input: object-to-camera-space transform T

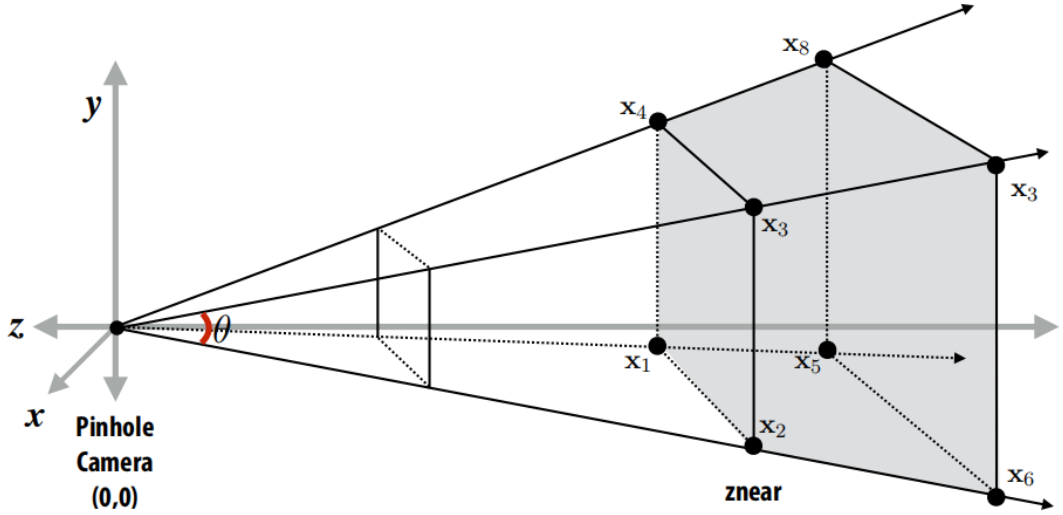
Goal: transform triangle vertices into camera space



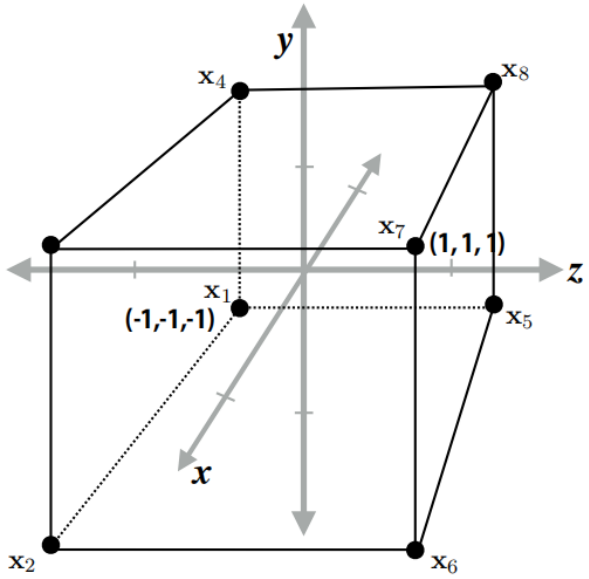
Step 1 (Vertex Processing): model-view transformation and camera transformation

Input: perspective projection transform P

Goal: transform triangle vertices into normalized coordinate space



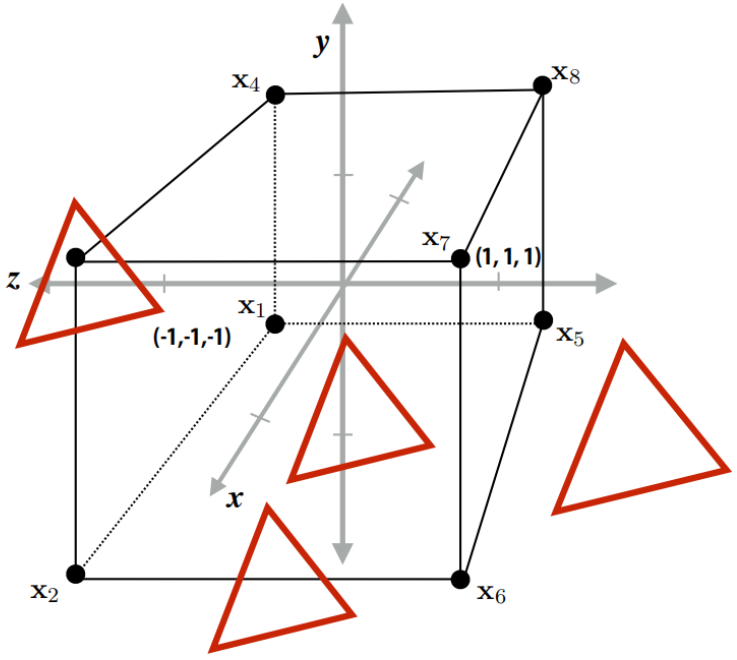
Camera-space positions: 3D



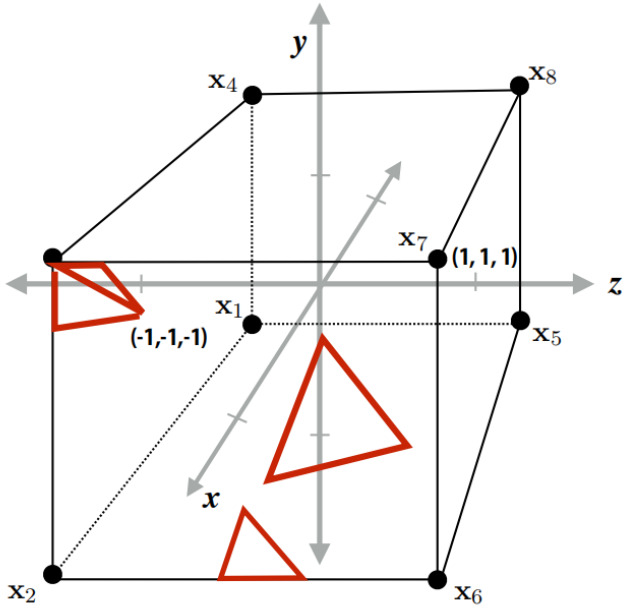
Normalized space positions

Step 3 (Primitive Processing): clipping

- Discard triangles outside the unit cube (culling)
- Clip triangles that extend beyond the unit cube to the cube



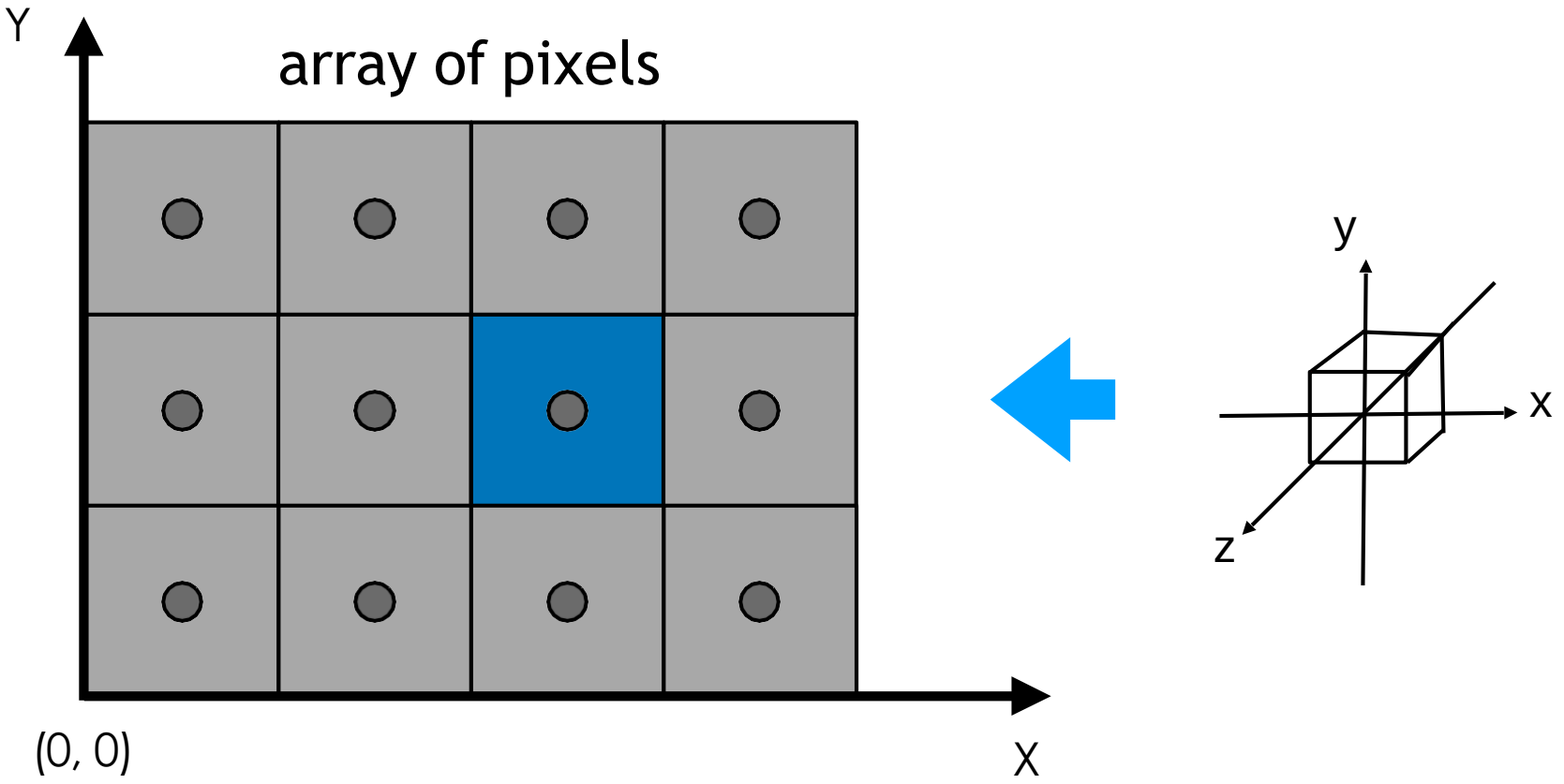
Triangles before clipping



Triangles after clipping

Step 4 (Primitive Processing): transform to screen coordinates

- Irrelevant to z
- Transform in xy plane: $[-1, 1]^2$ to [0, width] x [0, height]



Step 4 (Primitive Processing): transform to screen coordinates

- Irrelevant to z
- Transform in xy plane: $[-1, 1]^2$ to [0, width] x [0, height]
- Viewport transform matrix:

$$M_{viewport} = \begin{pmatrix} \frac{width}{2} & 0 & 0 & \frac{width}{2} \\ 0 & \frac{height}{2} & 0 & \frac{height}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

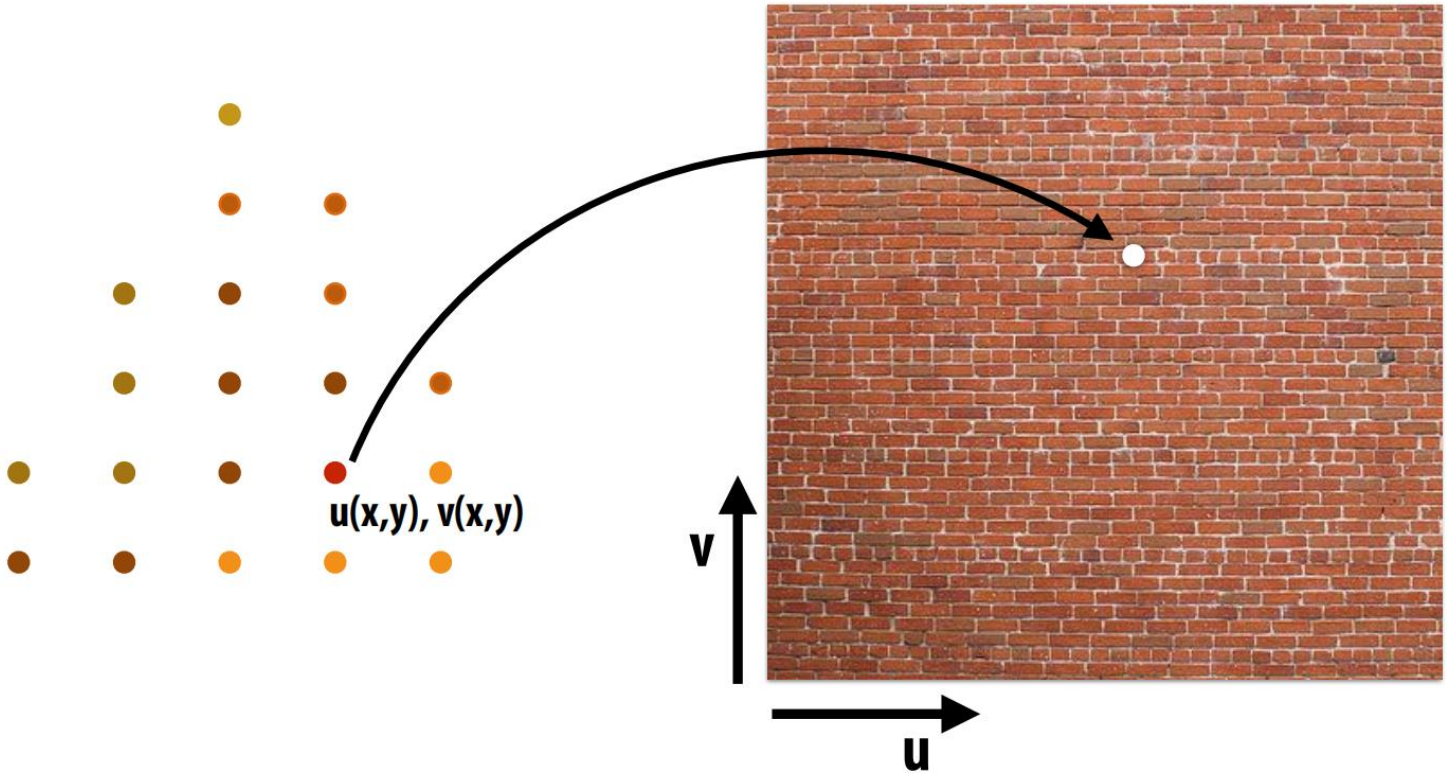
Step 5 (Primitive Processing): setup triangle

- Preprocess the triangles
- Compute triangle edge equations
- Compute triangle attribute interpolation equations

$$\mathbf{E}_{01}(x,y) \qquad \mathbf{U}(x,y)$$
$$\mathbf{E}_{12}(x,y) \qquad \mathbf{V}(x,y)$$
$$\mathbf{E}_{20}(x,y)$$
$$\frac{1}{\mathbf{w}}(x,y)$$
$$\mathbf{Z}(x,y)$$

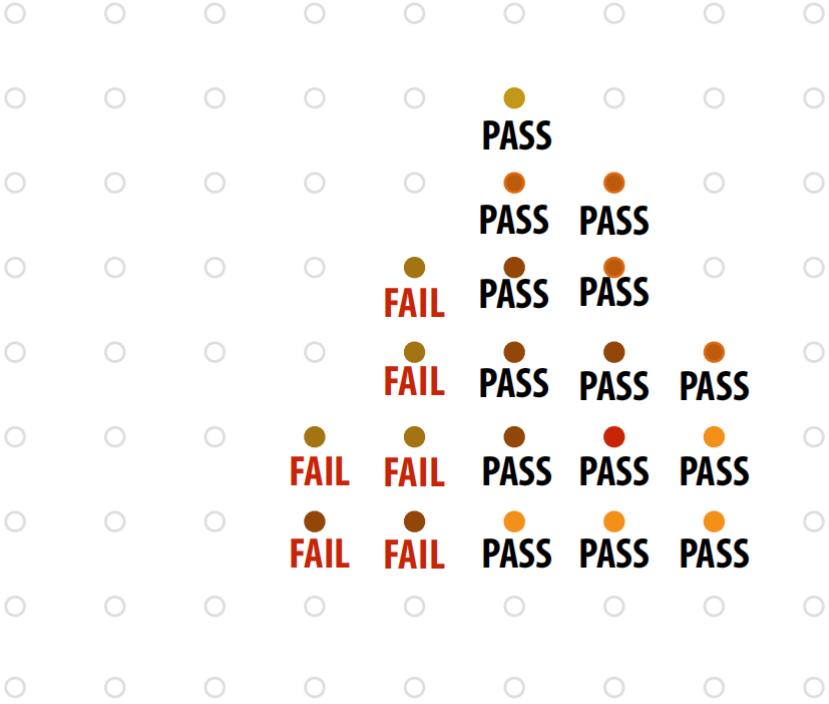
Step 6 (Fragment Processing): sample coverage and attributes

- Evaluate attributes z,u,v at all covered samples
- Sample texture map

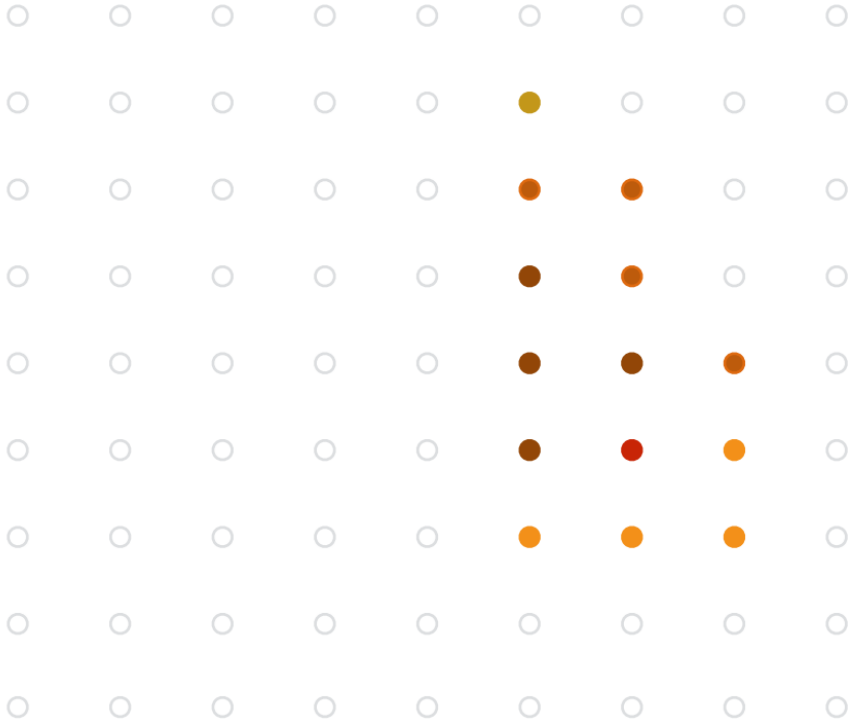


Step 7 (Screen sample operations): depth test

- Update the depth value at covered samples



Step 7 (Screen sample operations): update color buffer



High complexity 3D scenes

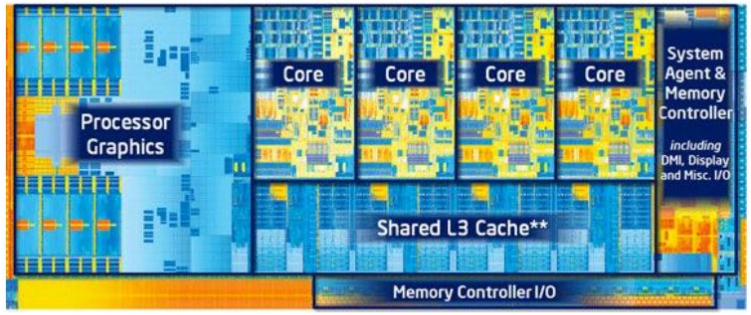
- 100's of thousands to millions of triangles in a scene
- Complex vertex and fragment shader computations
- High resolution screen outputs (2-4 Mpixel + supersampling)
- 30-60 fps

Unreal Engine Kite Demo (Epic Games 2015)

Graphics pipeline implementation: GPUs

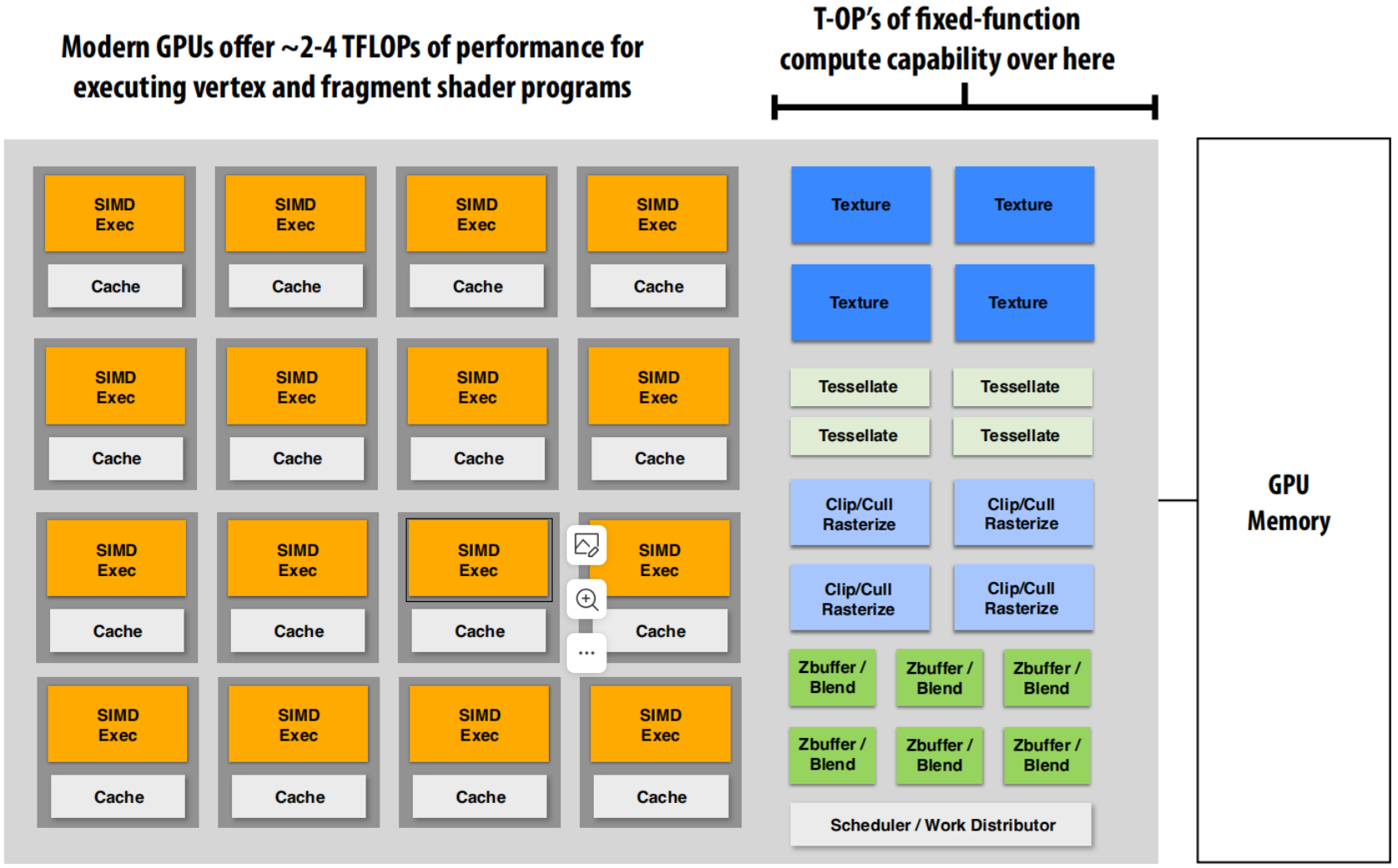
Specialized processors for executing graphics pipeline computations

Discrete GPU card
(NVIDIA GeForce Titan X)

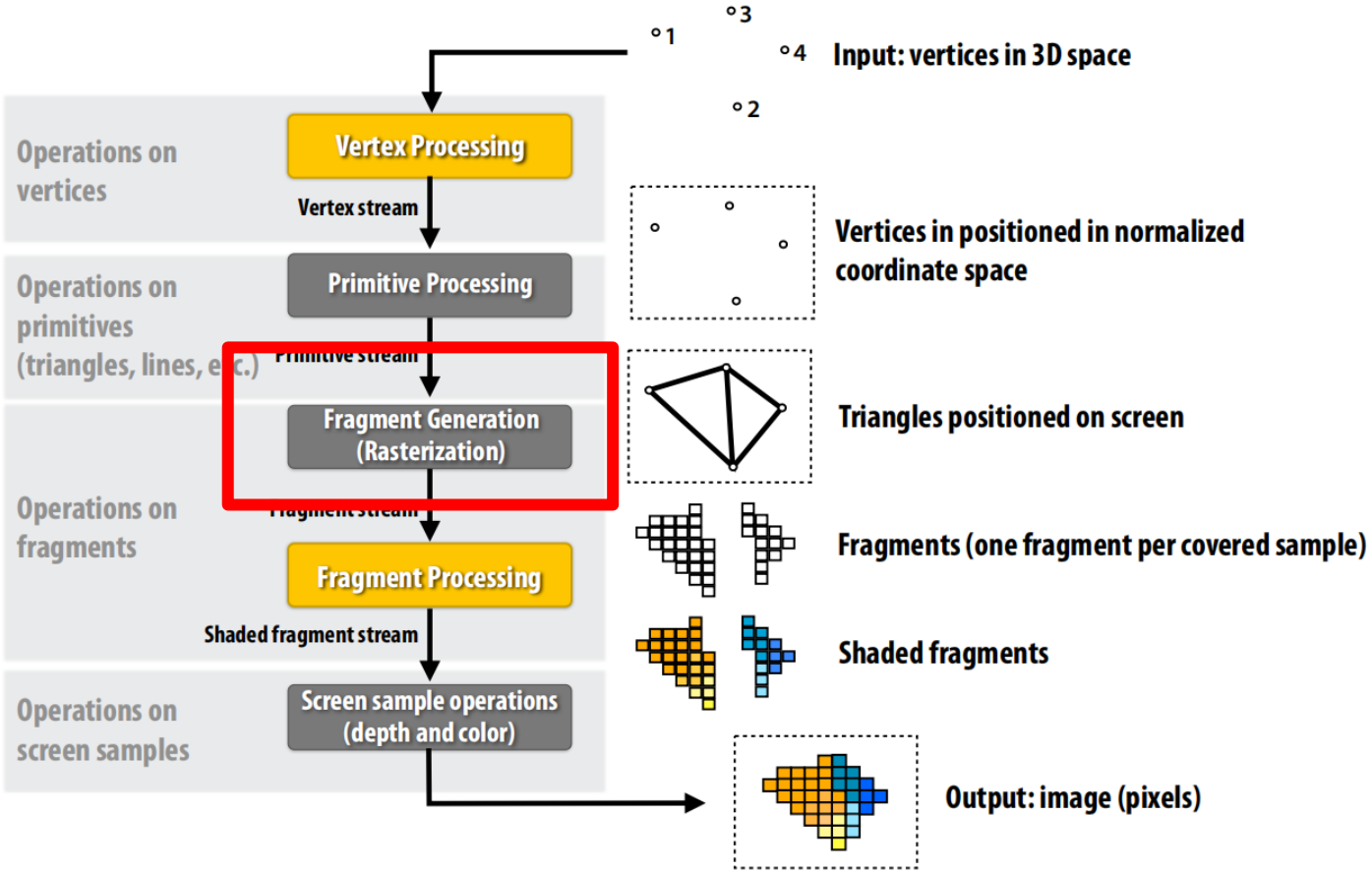


Integrated GPU: part of modern Intel CPU chip

GPU: heterogeneous, multi-core processor

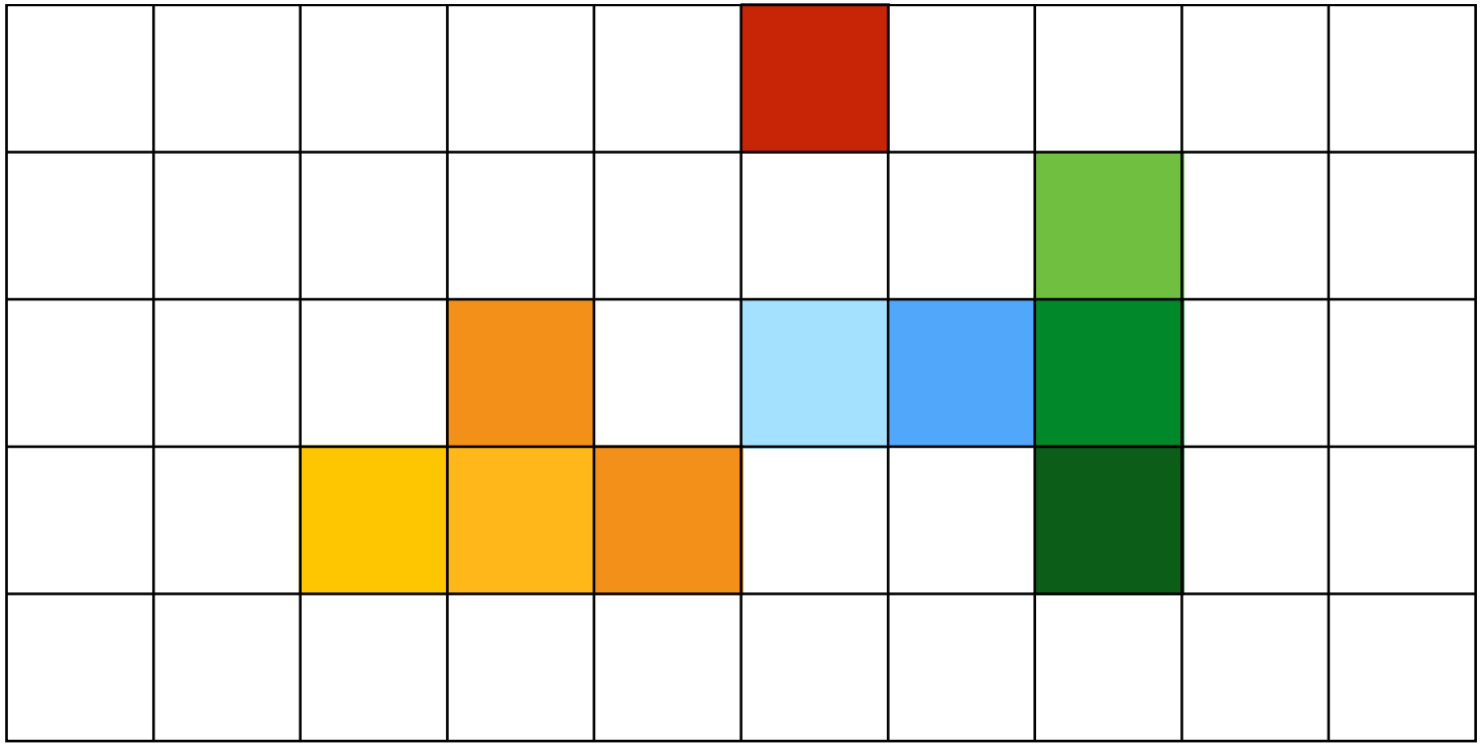


Rasterization: Drawing to Raster Displays



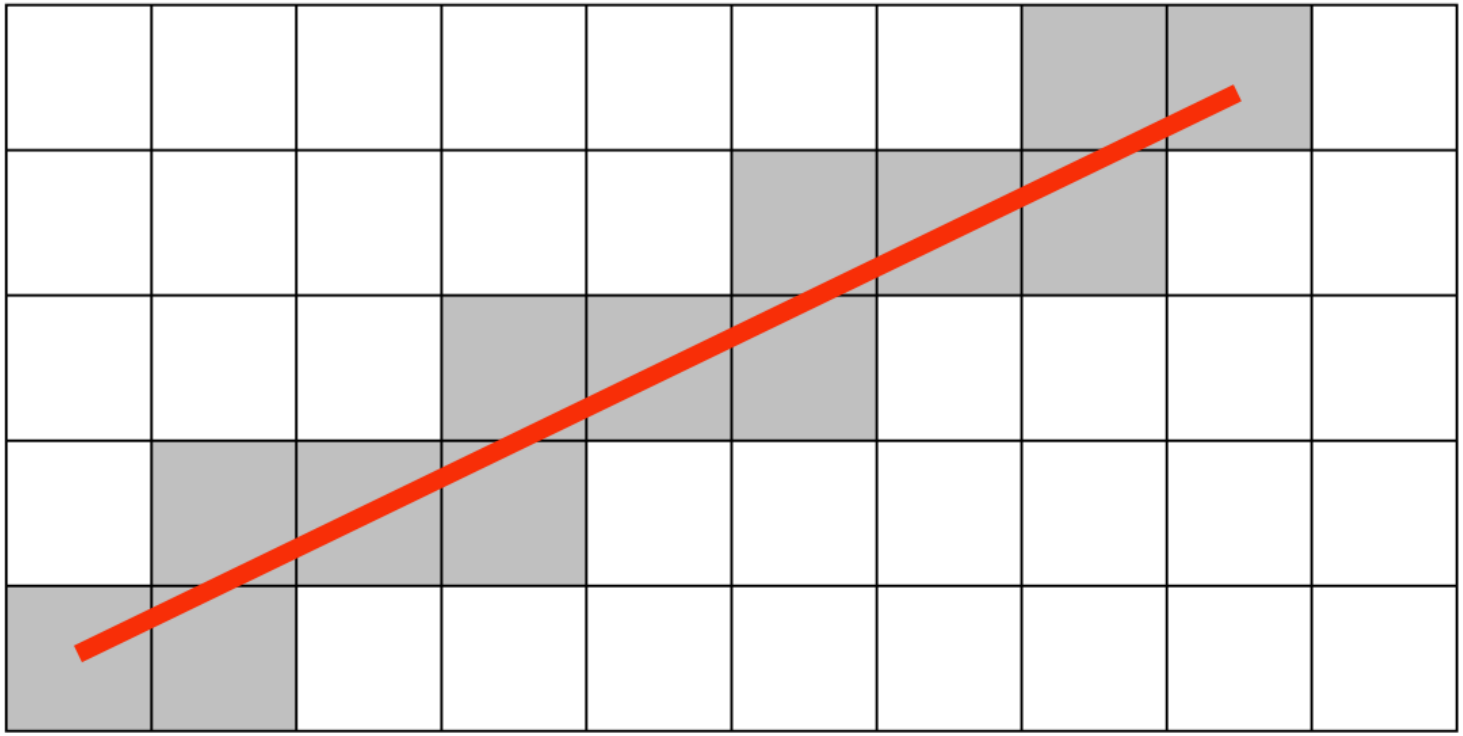
Output for a raster display

- Common abstraction of a raster display
  - Image represented as a 2D grid of “pixels”
  - Each pixel can take a unique color value



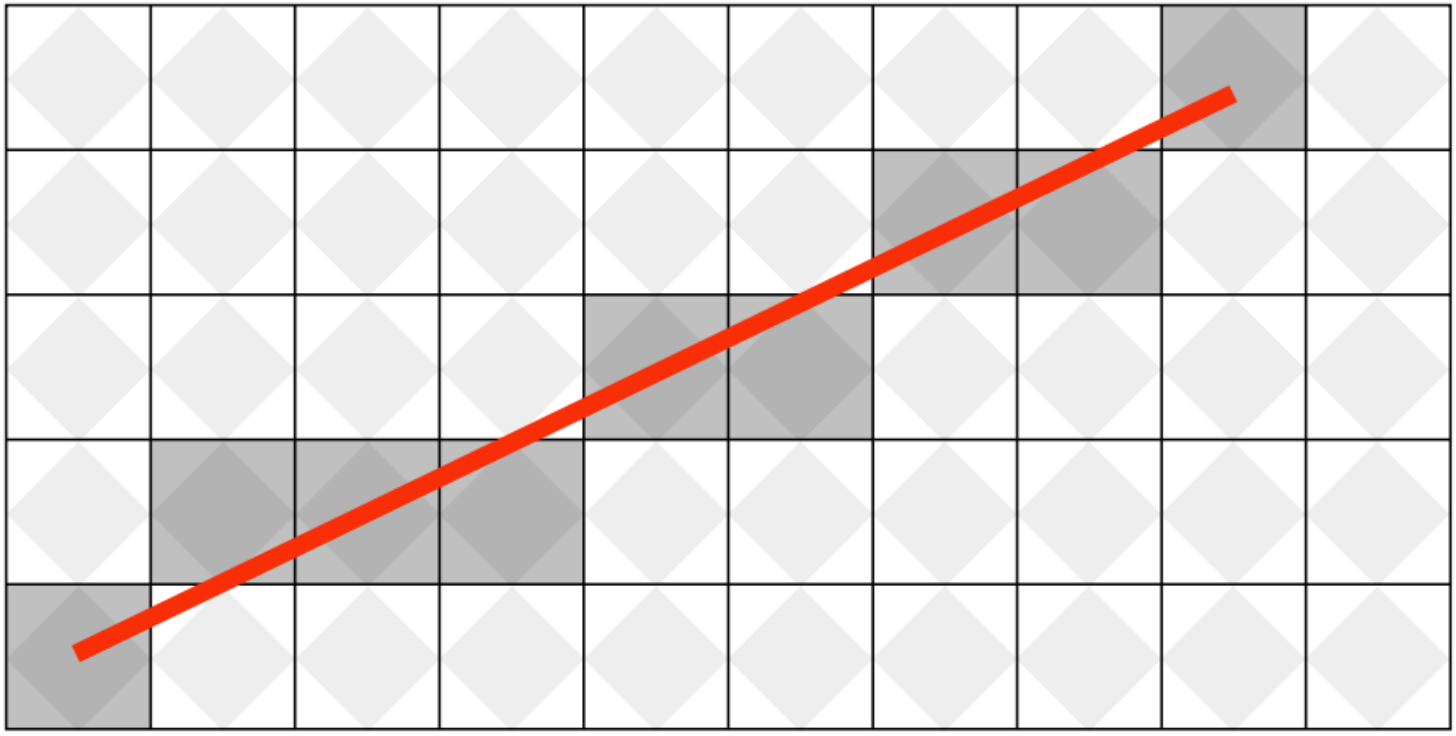
Which pixels should we color?

- Light up all pixels intersected by a line



Which pixels should we color?

- Light up all pixels intersected by a line
  - Diamond rule (used by modern GPUs)



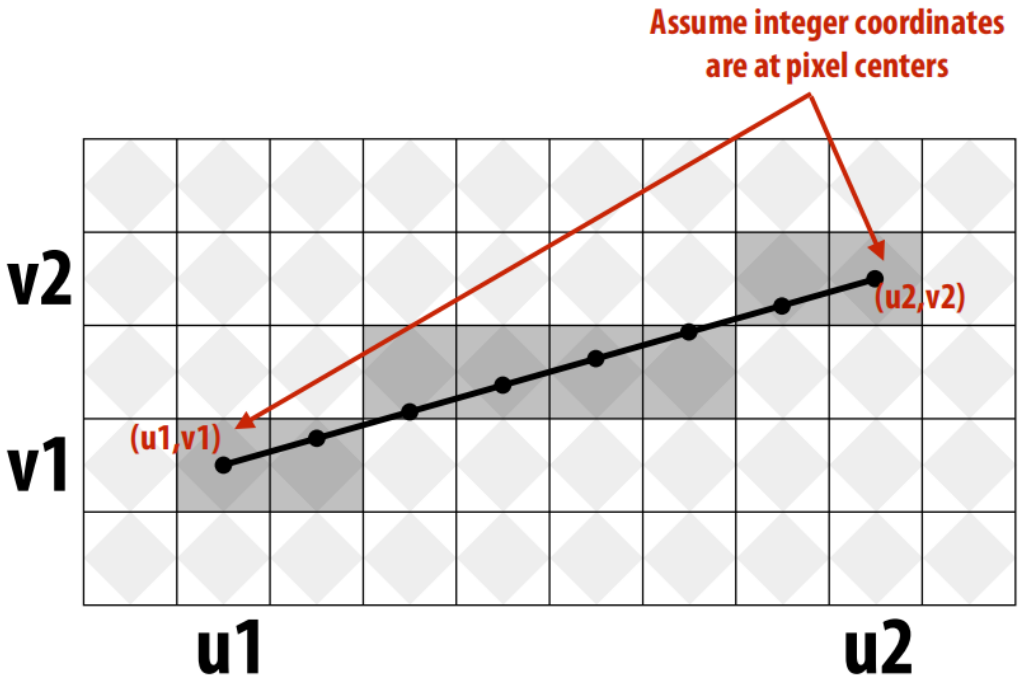
Which pixels should we color?

- Light up all pixels intersected by a line
  - Diamond rule (used by modern GPUs)
- A simple solution: check every single pixel in the image (on the screen) to see if it meeds the condition
  - $O(n^2)$ pixels in image vs. at most $O(n)$ “lit up” pixels
  - How to do better?

Incremental line rasterization

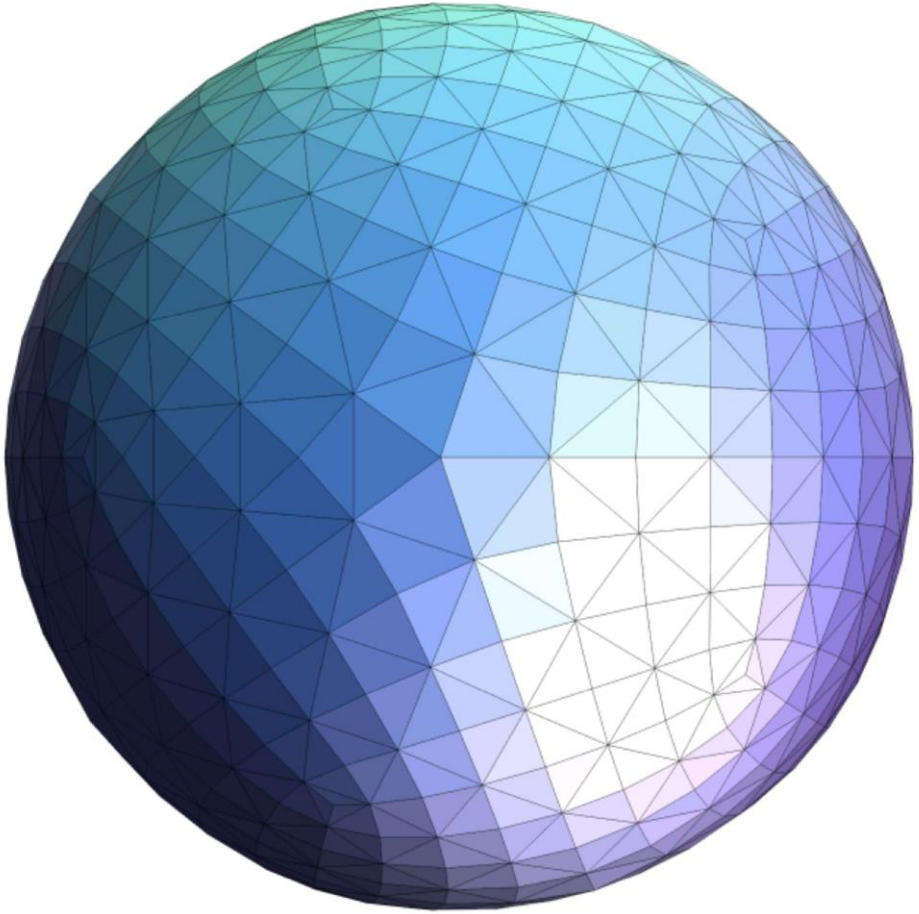
- Suppose two endpoints (u1,v1), (u2,v2)
  - Slope of line: s = (v2-v1) / (u2-u1)
  - From left to right…

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

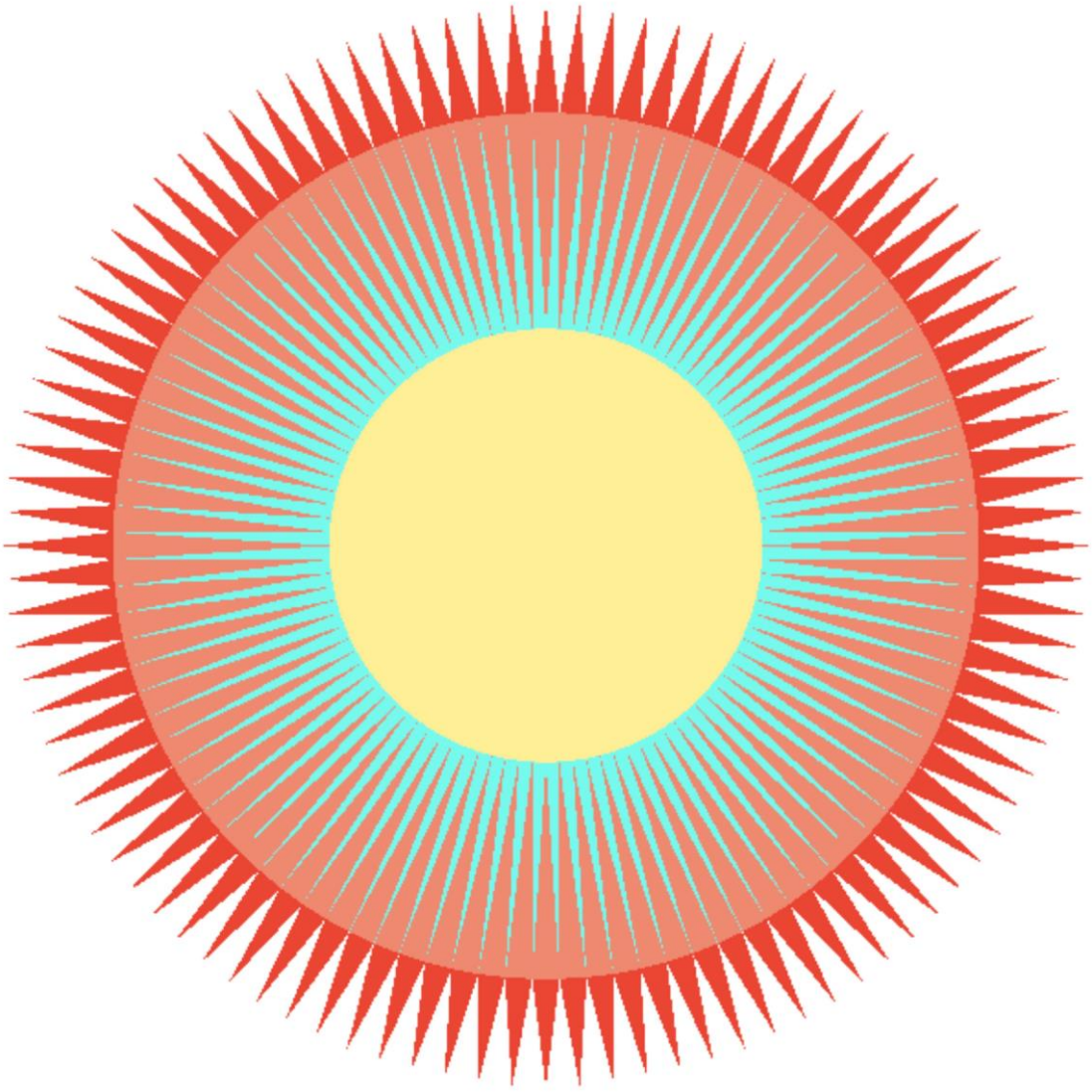


Bresenham algorithm later in this course

Triangle Meshes



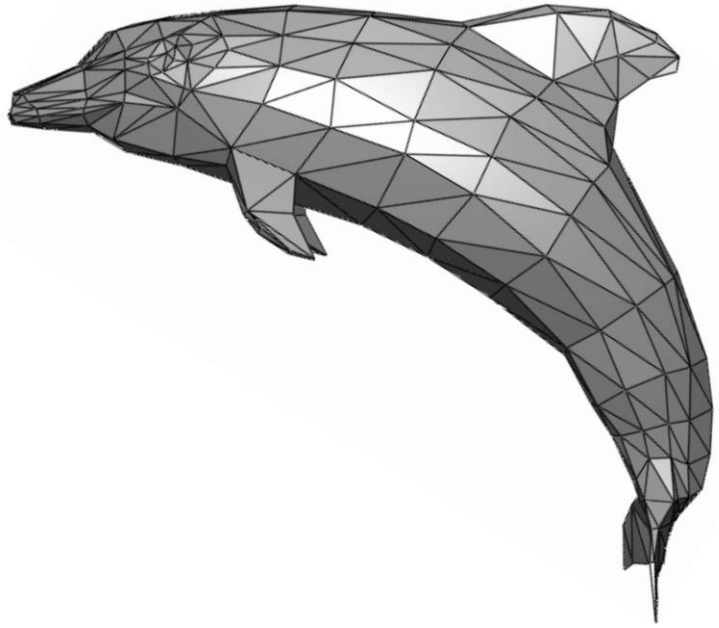
Triangle Meshes



Triangles - Fundamental Shape Primitives

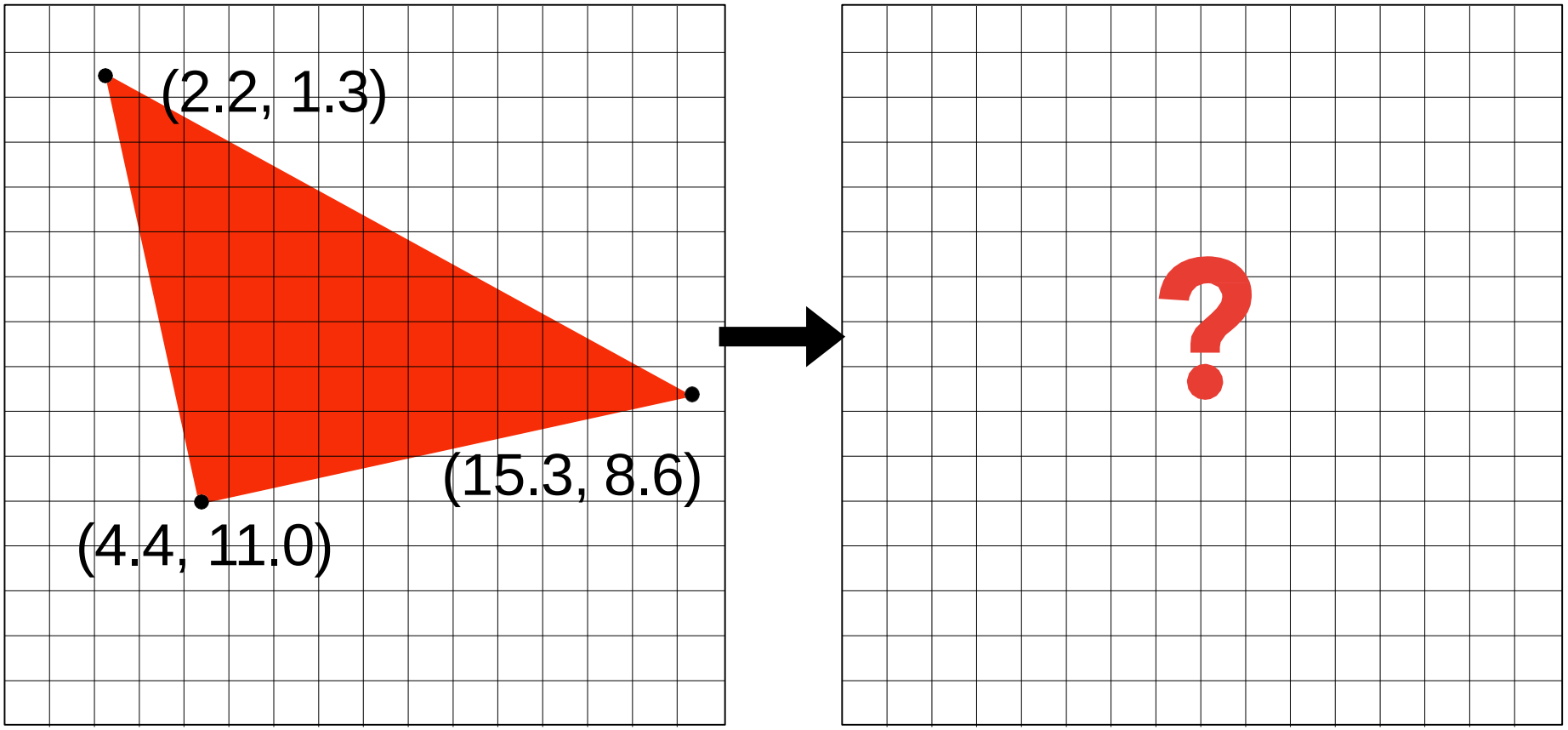
Why triangles?

- Most basic polygon
  - Break up other polygons



- Unique properties
  - Guaranteed to be planar
  - Well-defined interior
  - Well-defined method for interpolating values at vertices over triangle (barycentric interpolation)

What Pixel Values Approximate a Triangle?



Input: position of triangle vertices projected on screen

Output: set of pixel values approximating triangle

A Simple Approach: Sampling

Sampling a Function

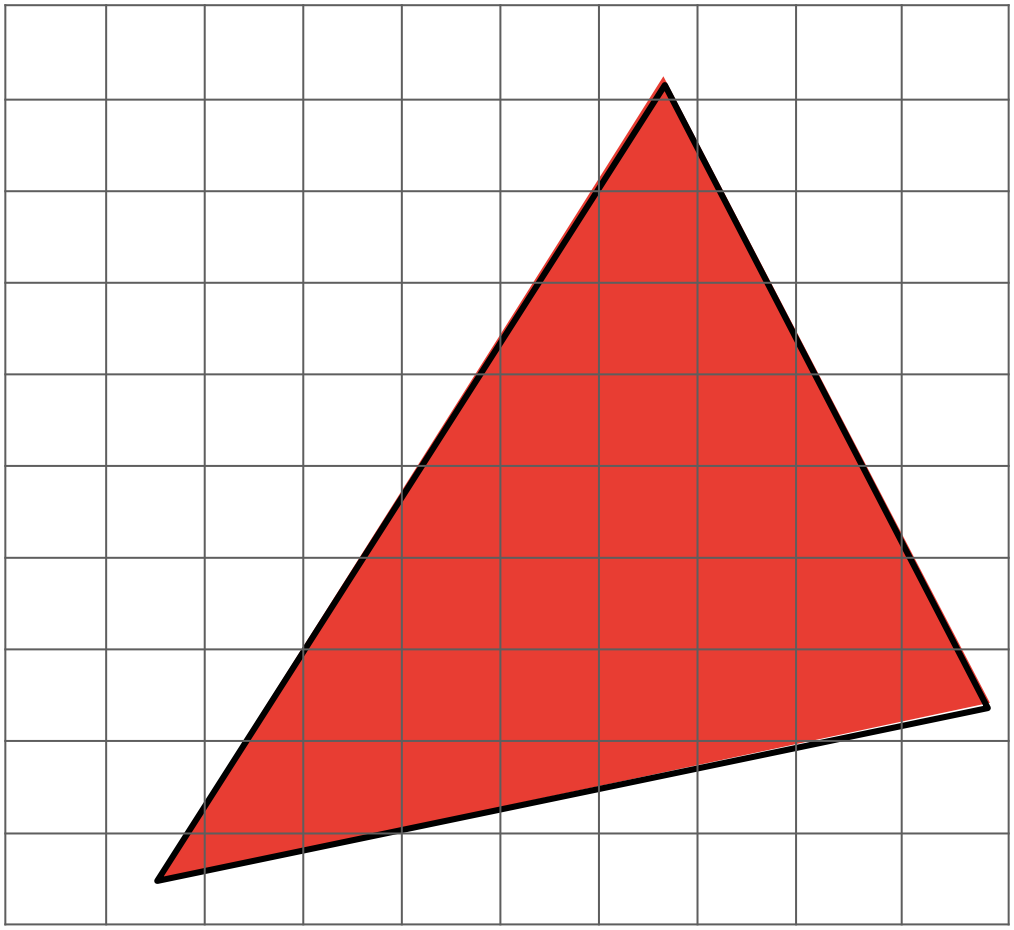
Evaluating a function at a point is sampling.

We can discretize a function by sampling.

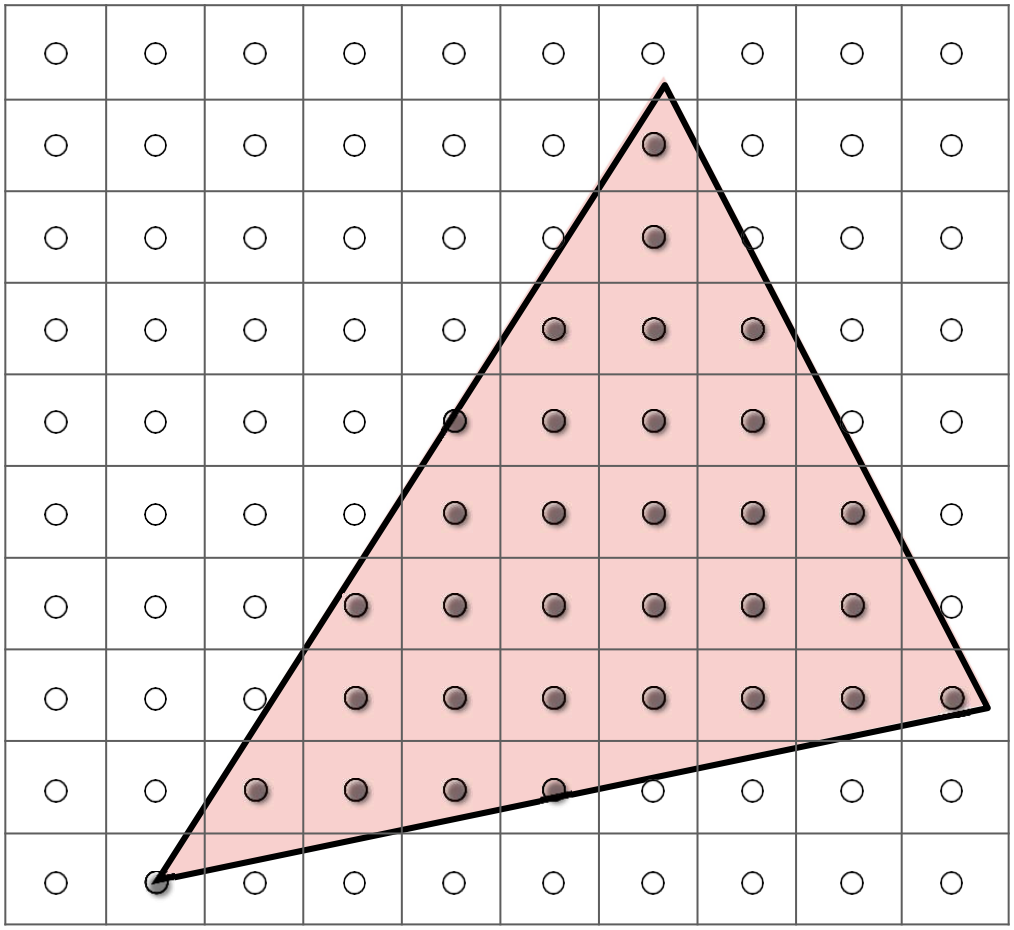
```
for (int x = 0; x < xmax; ++x)
    output[x] = f(x);
```

Sampling is a core idea in graphics.
We sample time (1D), area (2D), direction (2D), volume (3D) …

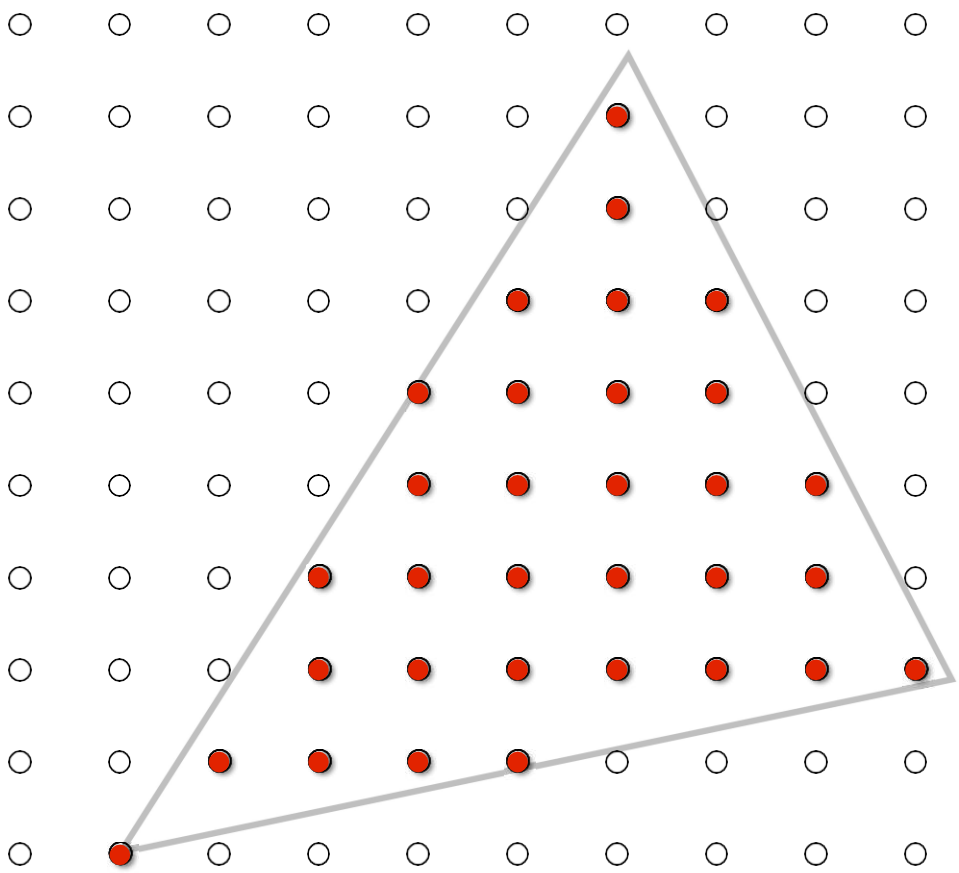
Rasterization As 2D Sampling



Sample If Each Pixel Center Is Inside Triangle

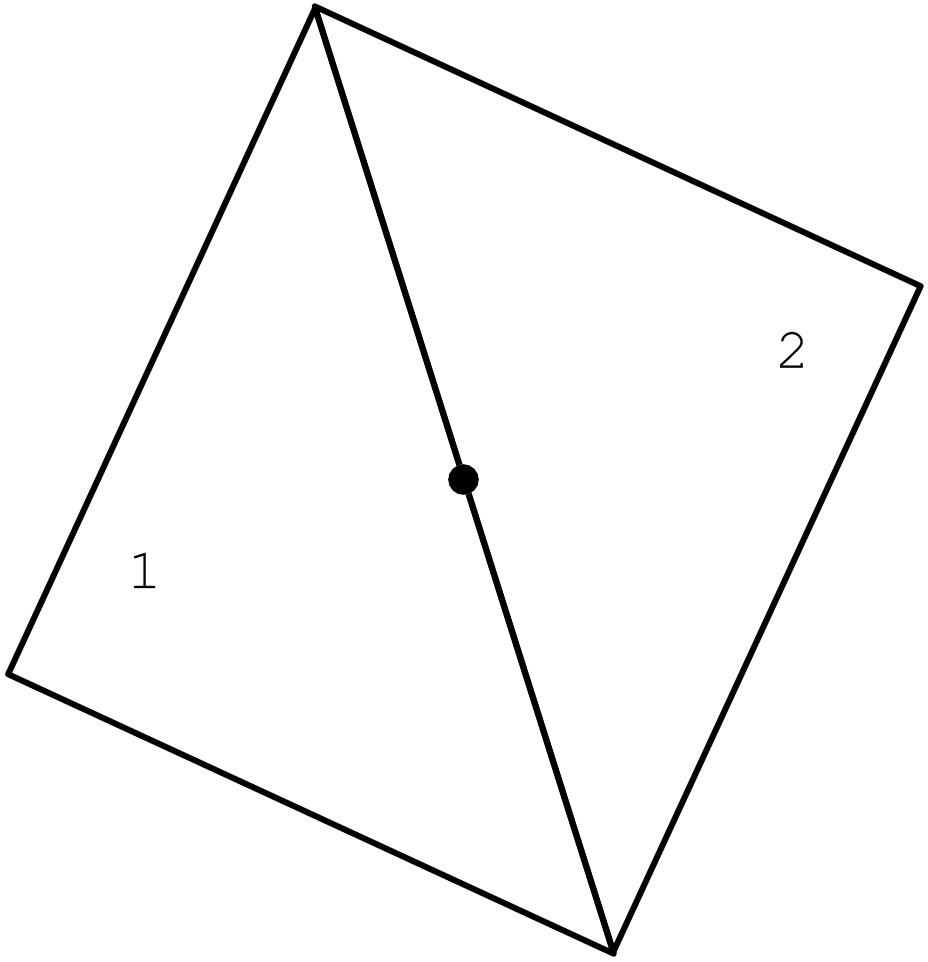


Sample If Each Pixel Center Is Inside Triangle

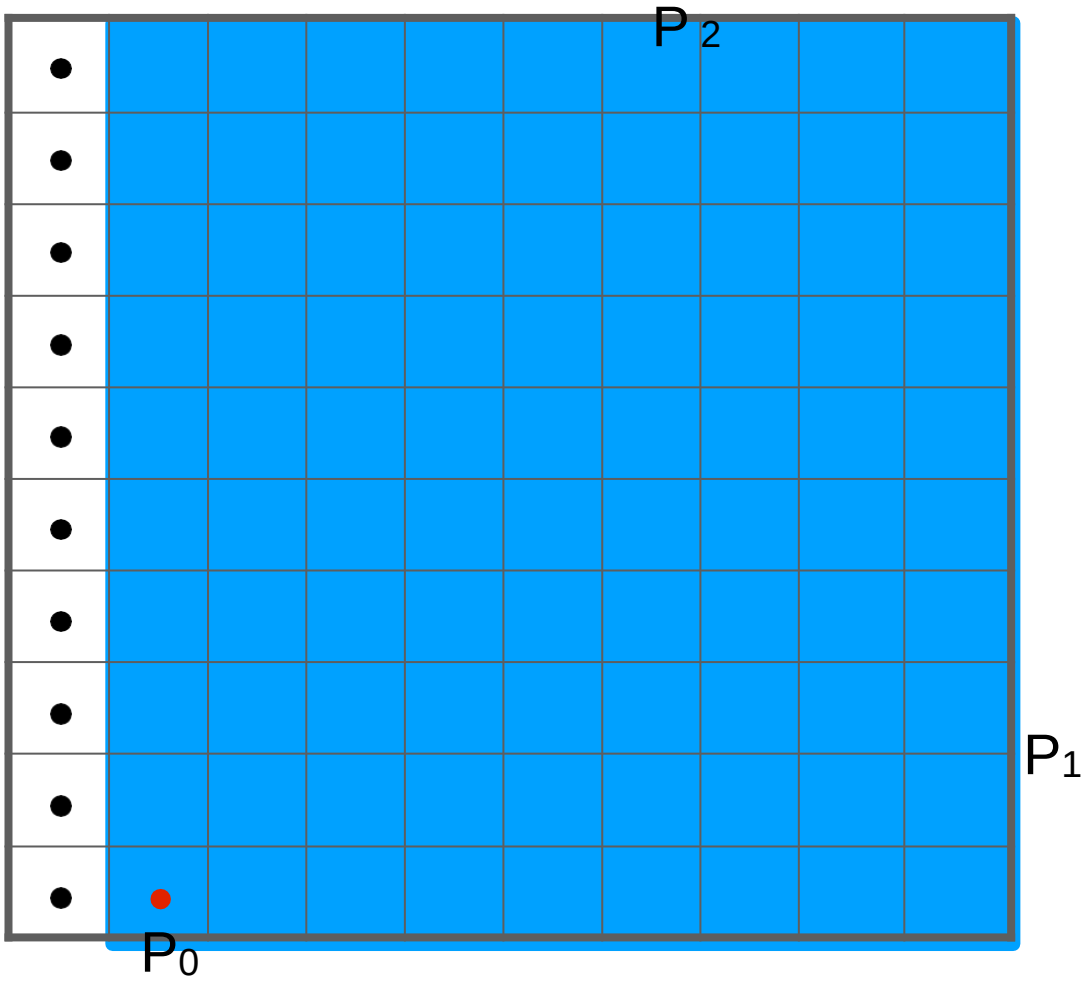


Edge Cases (Literally)

Is this sample point covered by triangle 1, triangle 2, or both?

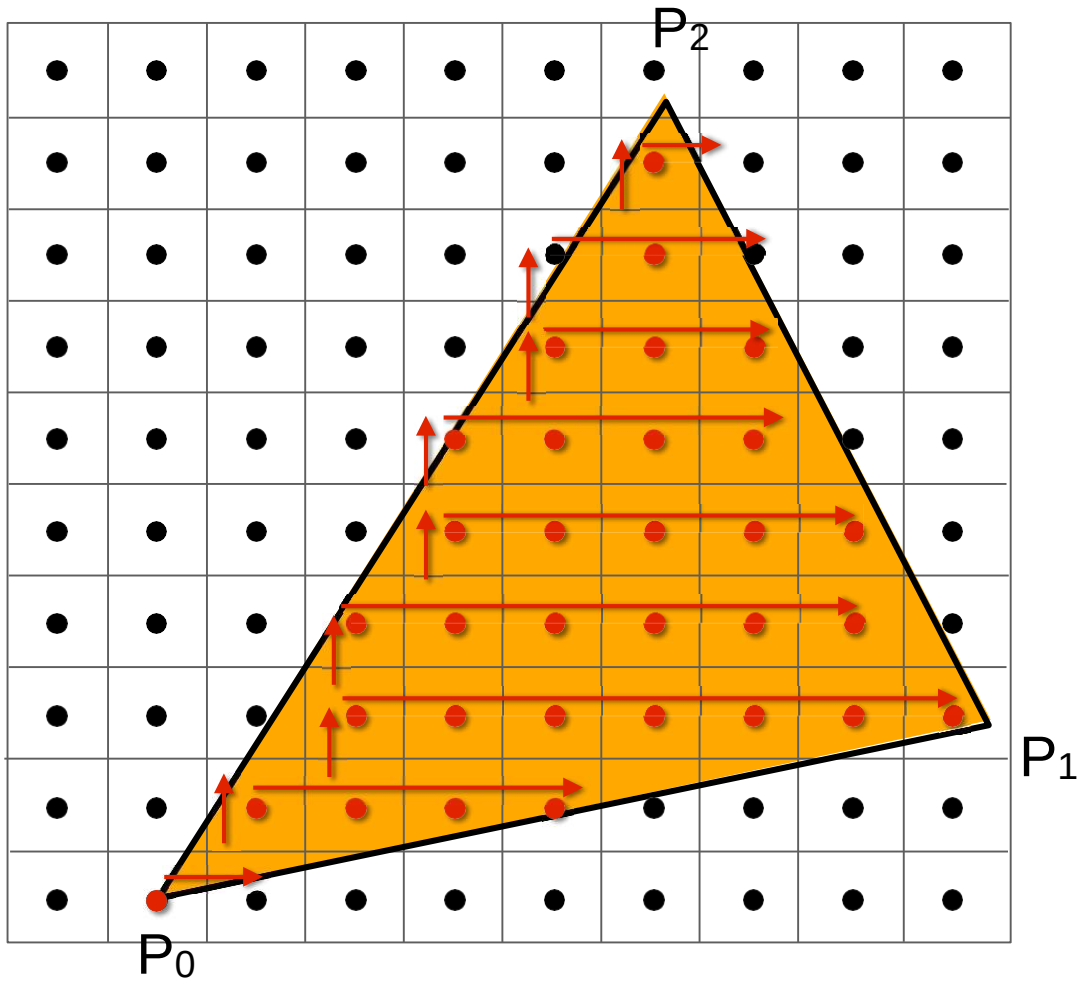


Checking All Pixels on the Screen?



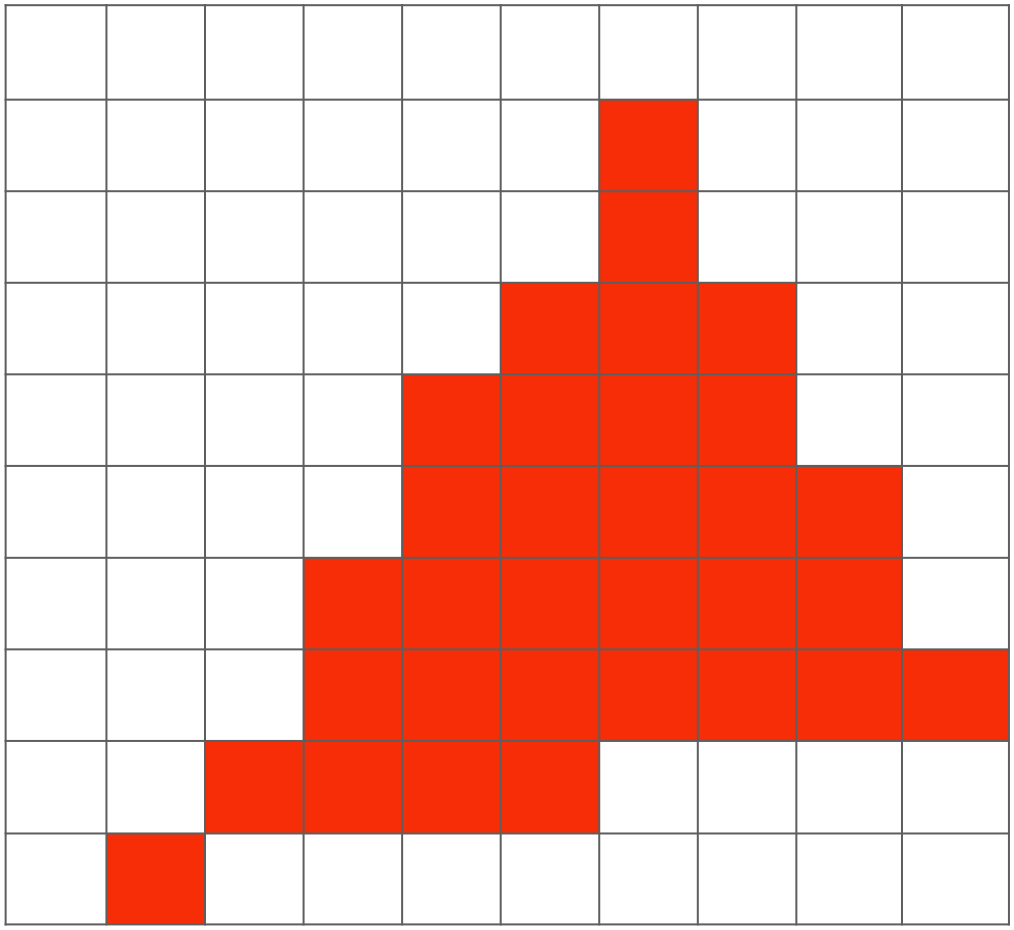
Use a Bounding Box!

Incremental Triangle Traversal (Faster?)

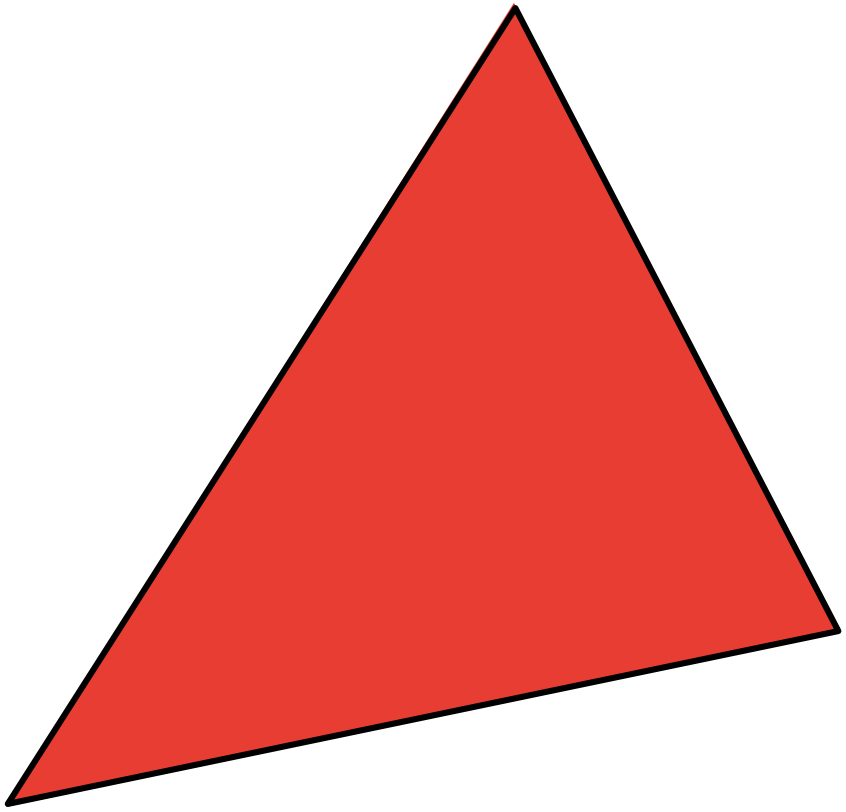


suitable for thin and rotated triangles

What's Wrong With This Picture?



Jaggies!



Thank you!