The Graphics Pipeline

also today: particle systems

CS 4300/5310

Computer Graphics

ANNOUNCEMENTS

Upcoming Deadlines

- HW2: Raytracer
  - ~~Today!~~ February 22nd!
- HW3: Particle Systems
  - March 2nd

Late Day Policy Reminder

- 5 total across all assignments.
- There are 4 assignments.
- Cannot use late days for projects, reading responses.

Art Contest Policies

- Must be the result of your assignment or project!
- Can turn in any time before end of semester
- No more than 1 entry per contest per assignment per person
- No more than 2 entries per contest per project per group

Workshop Class Thursday

- I will be out of town

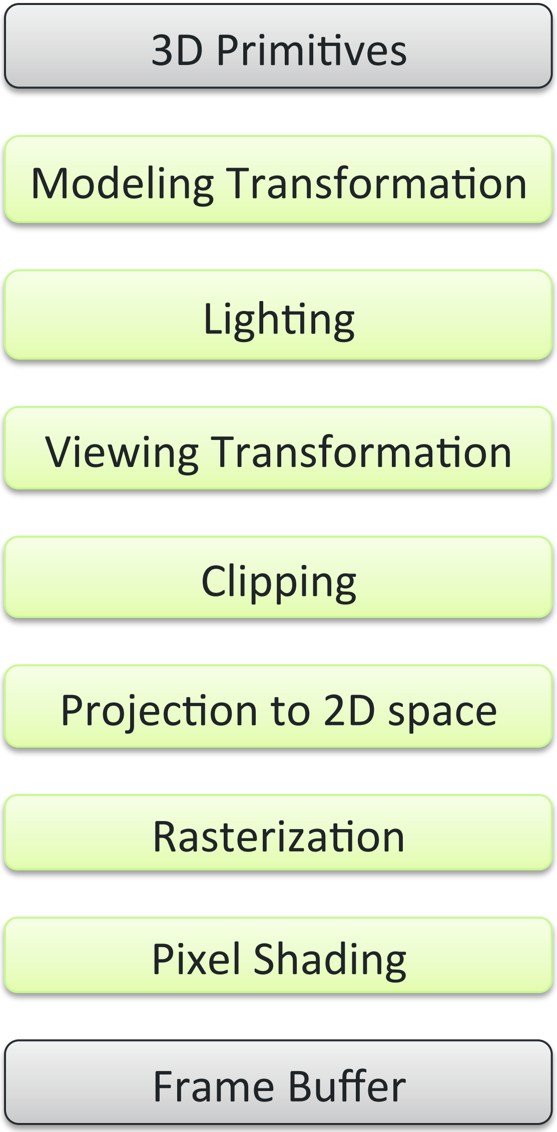
- Morteza will be running class
  - Opportunity to get started on assignment 3 with in-person support
  - Ask questions about OpenGL/DirectX setup for projects
  - Or continue working on assignment 2!

RENDERING BY RASTERIZATION

Object-Order Rendering

- Why is it called object-order rendering?

The Graphics Pipeline



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

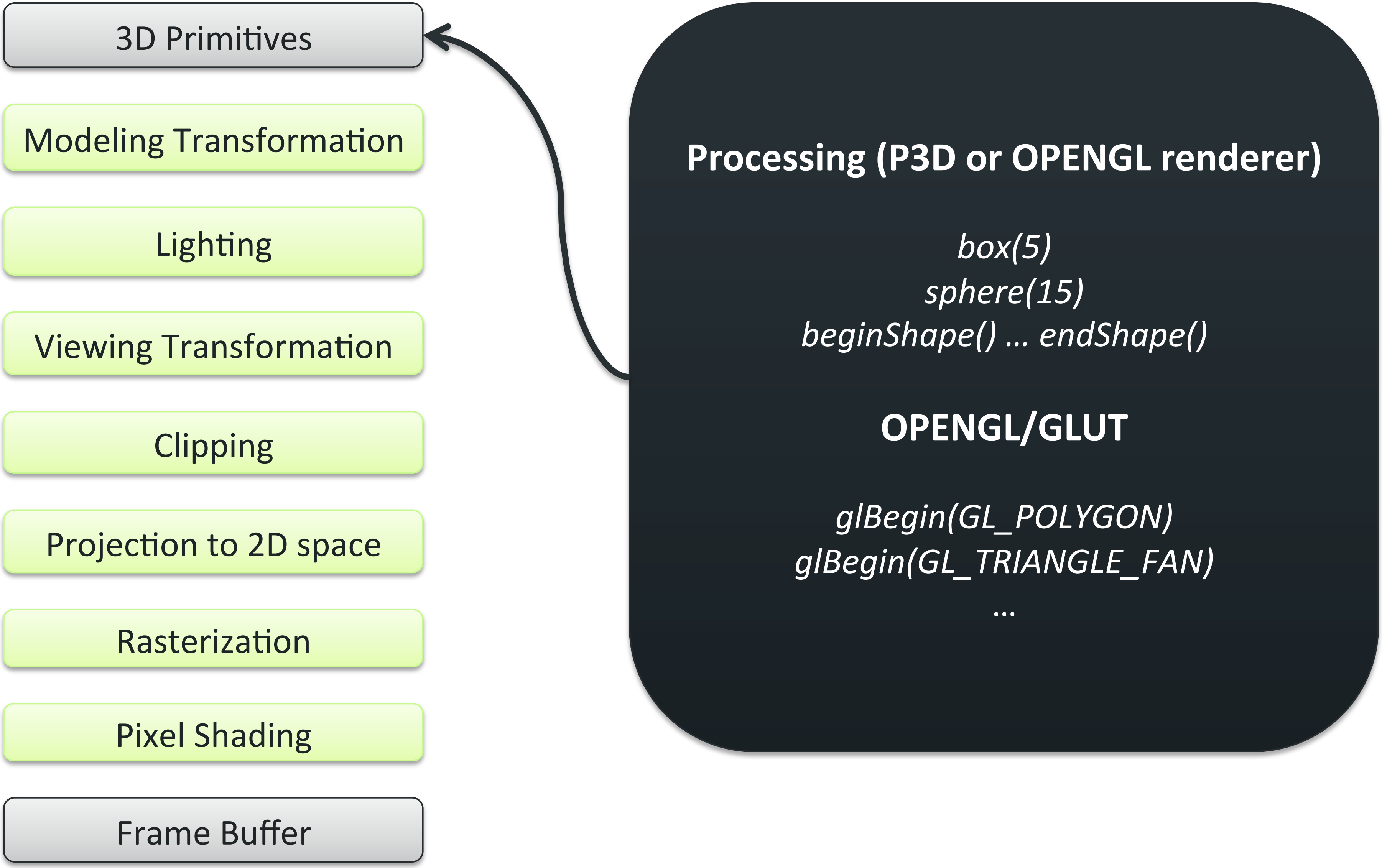
Rasterization

Pixel Shading

Frame Buffer

- A series of steps to turn a 3D environment into a 2D image
  - Output of one step = input to the next

The Graphics Pipeline



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

Rasterization

Pixel Shading

Frame Buffer

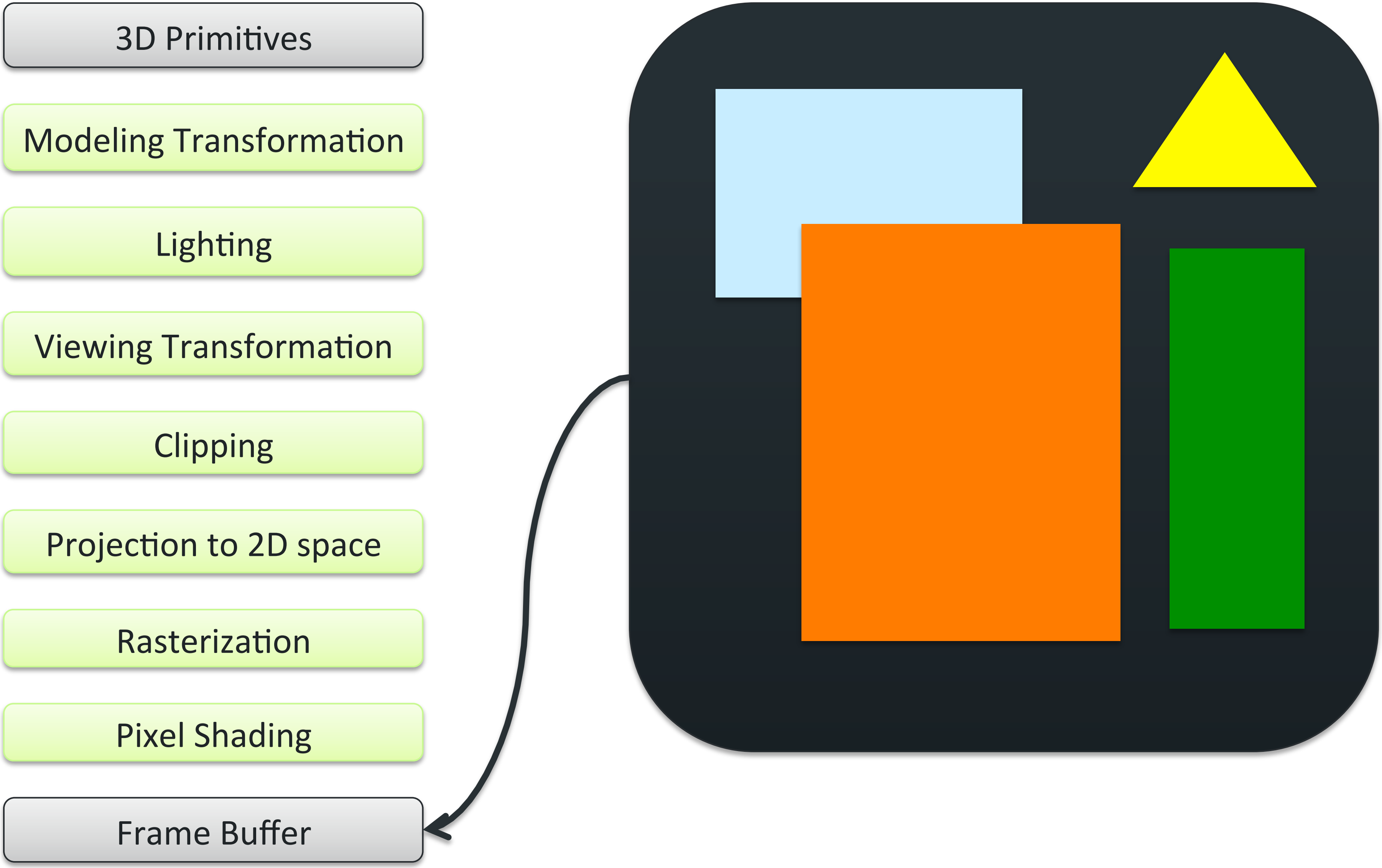
Processing (P3D or OPENGL renderer)

box(5)
sphere(15)
beginShape() … endShape()

OPENGL/GLUT

glBegin(GL_POLYGON)
glBegin(GL_TRIANGLE_FAN)
…

The Graphics Pipeline



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

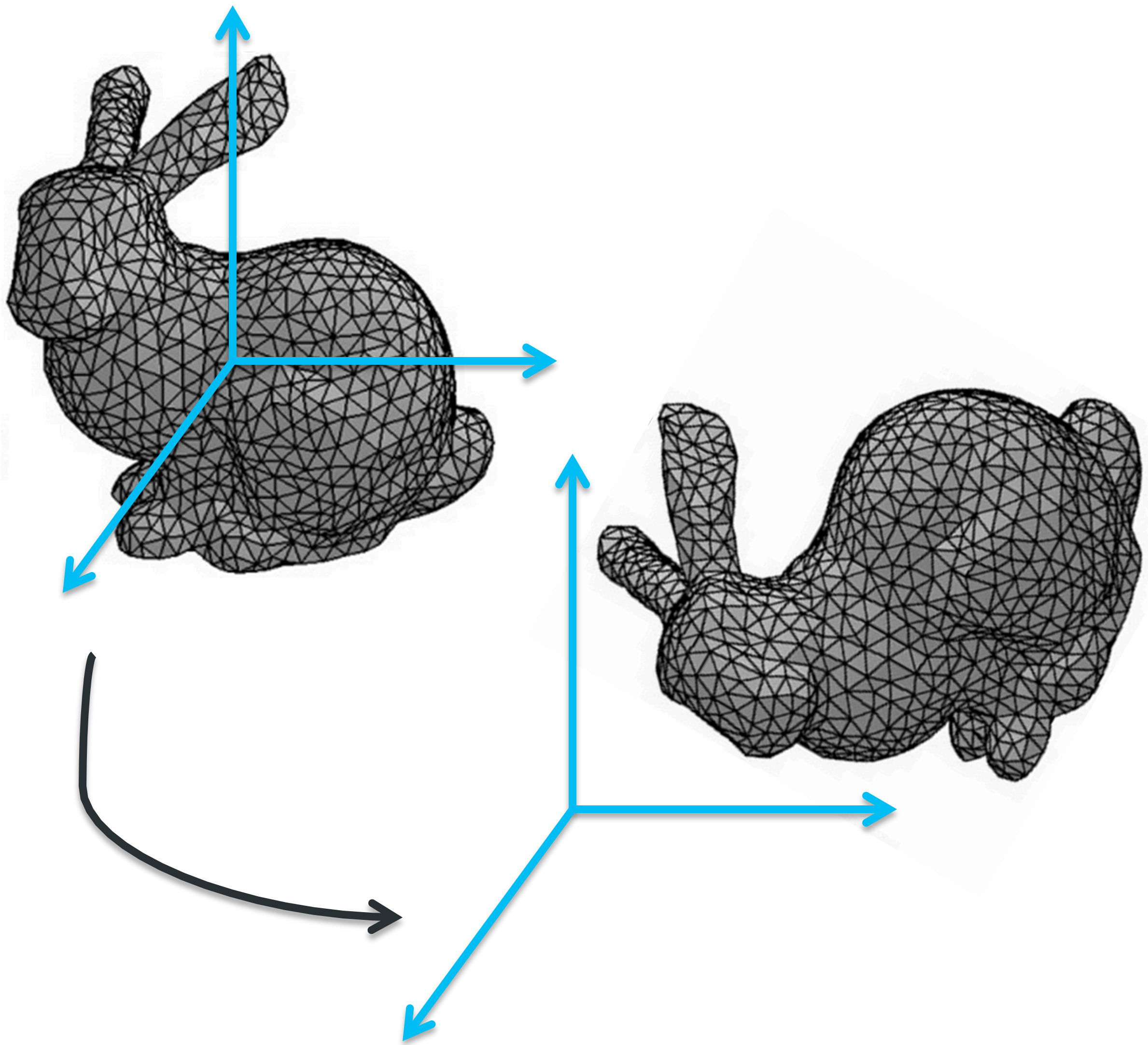
Rasterization

Pixel Shading

Frame Buffer

The Graphics Pipeline

- 3D Primitives
- Modeling Transformation
- Lighting
- Viewing Transformation
- Clipping
- Projection to 2D space
- Rasterization
- Pixel Shading
- Frame Buffer



The Graphics Pipeline

3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

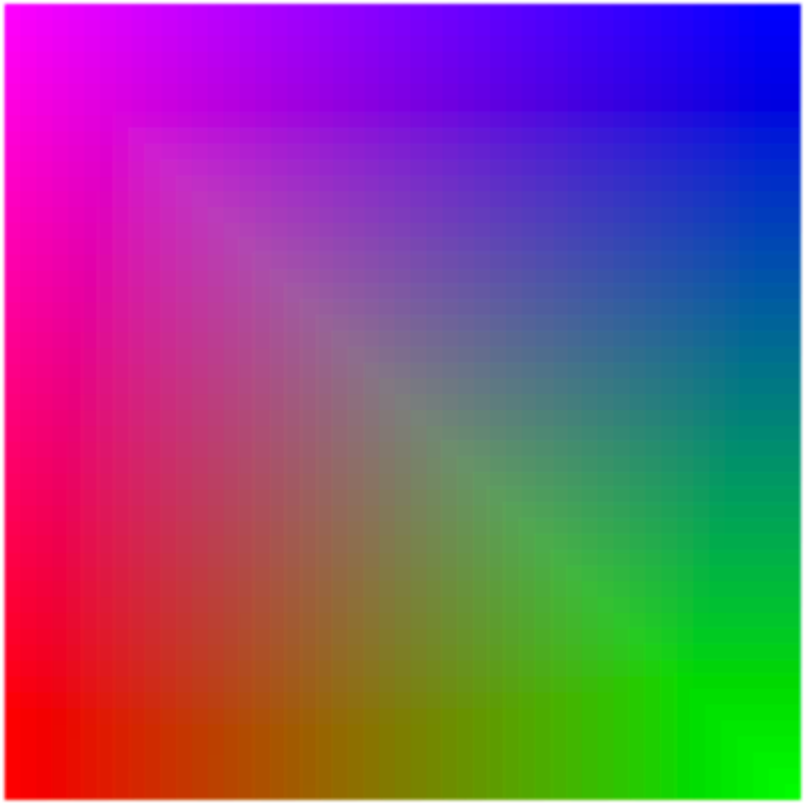
Clipping

Projection to 2D space

Rasterization

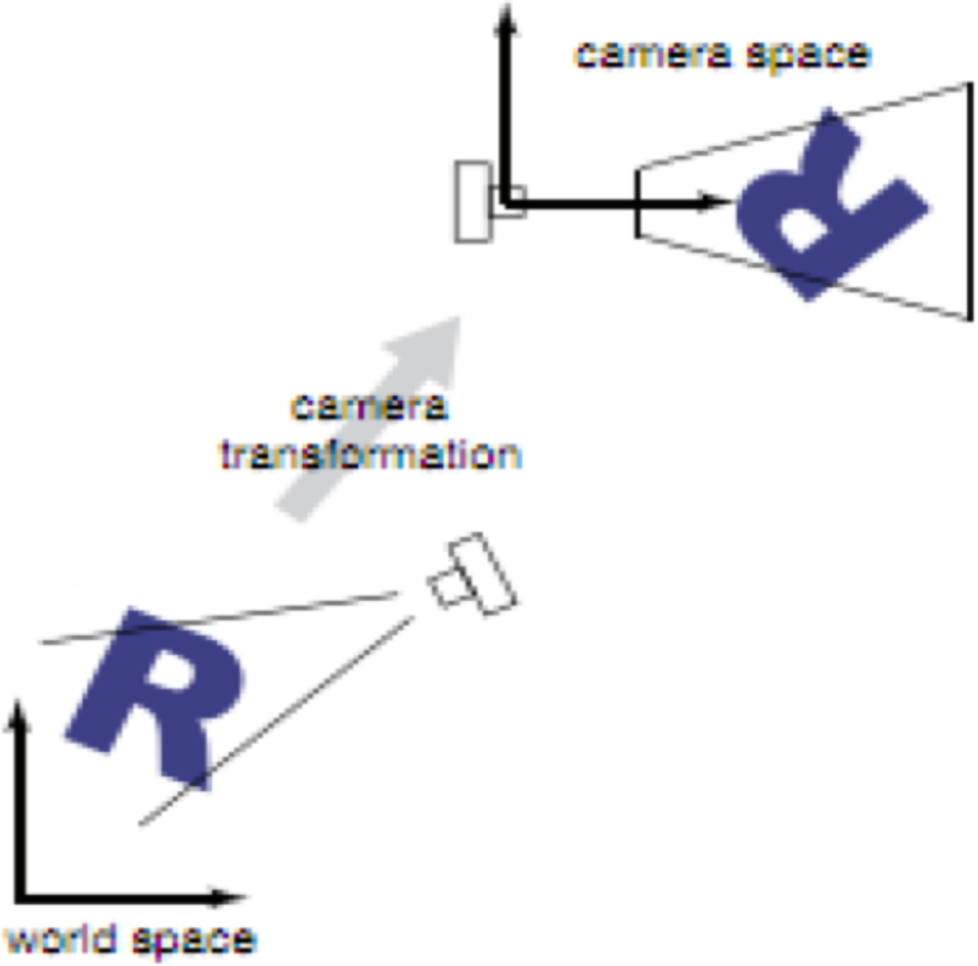
Pixel Shading

Frame Buffer



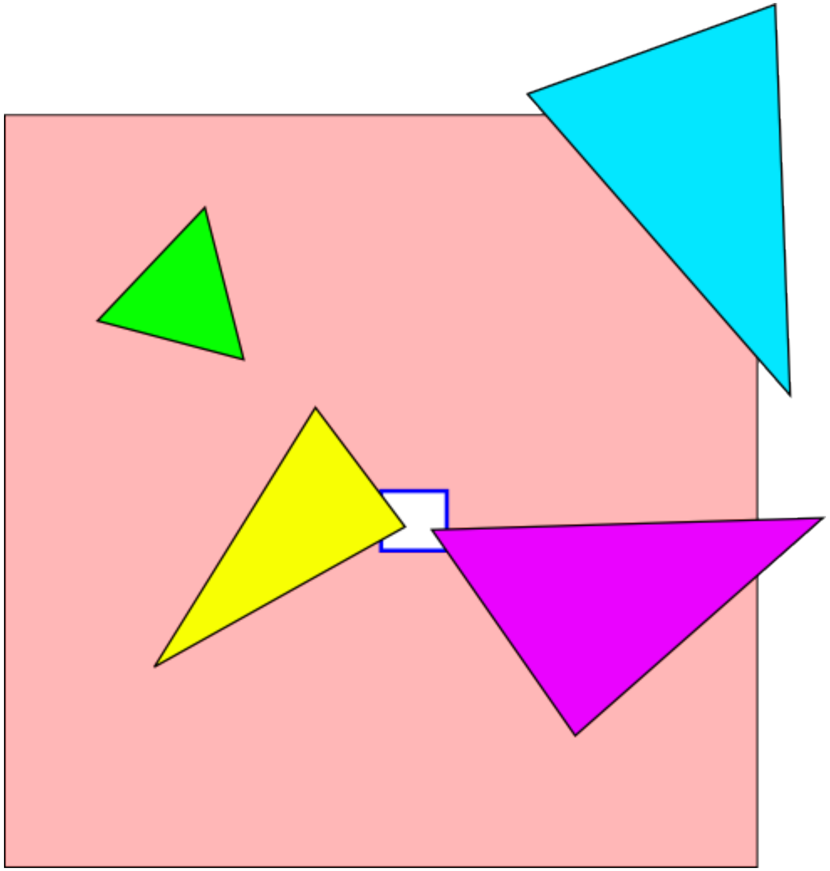
The Graphics Pipeline

- 3D Primitives
- Modeling Transformation
- Lighting
- Viewing Transformation
- Clipping
- Projection to 2D space
- Rasterization
- Pixel Shading
- Frame Buffer



The Graphics Pipeline

- 3D Primitives
- Modeling Transformation
- Lighting
- Viewing Transformation
- Clipping
- Projection to 2D space
- Rasterization
- Pixel Shading
- Frame Buffer

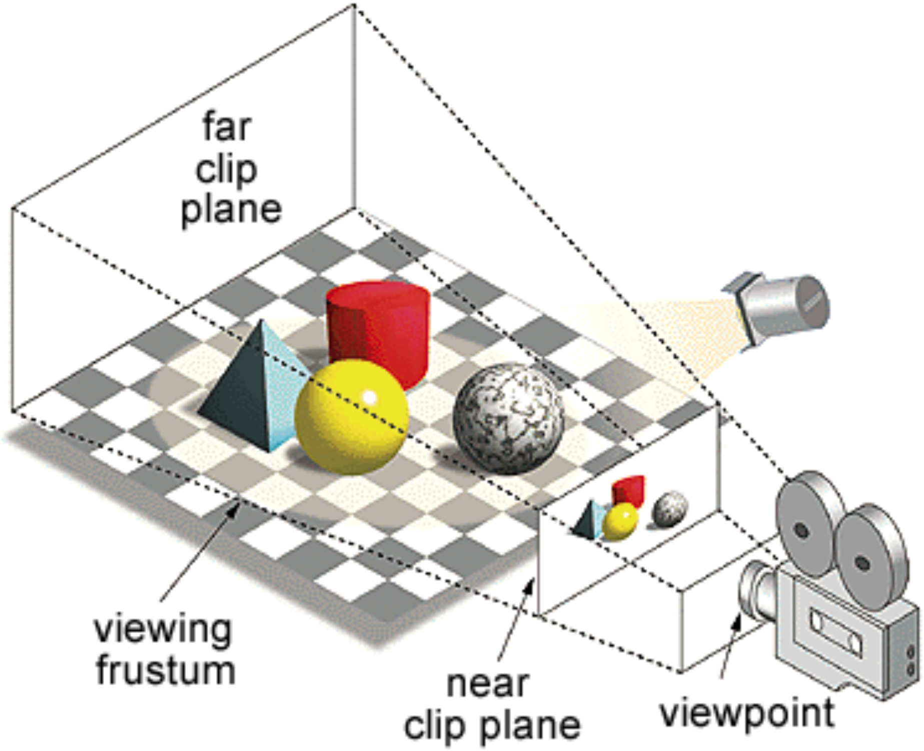


From ryg blog: http://fgiesen.wordpress.com/2011/07/05/a-trip-through-the-graphics-pipeline-2011-part-5/

The Graphics Pipeline

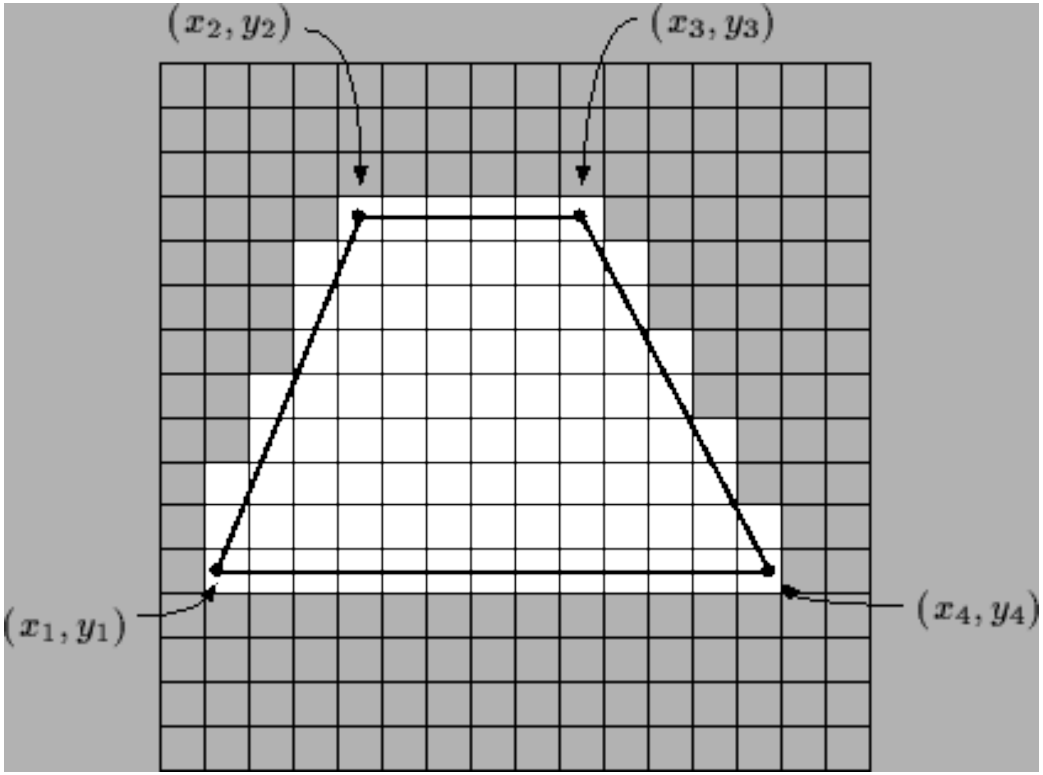
- 3D Primitives
- Modeling Transformation
- Lighting
- Viewing Transformation
- Clipping
- Projection to 2D space
- Rasterization
- Pixel Shading
- Frame Buffer

From Computer Desktop Encyclopedia
Reproduced with permission.
© 1998 Intergraph Computer Systems



The Graphics Pipeline

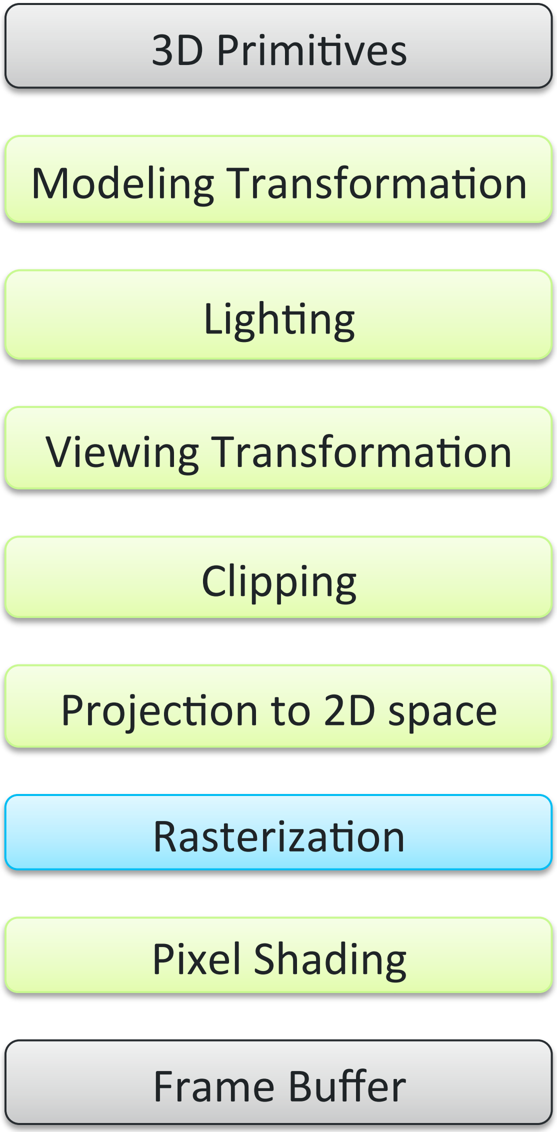
- 3D Primitives
- Modeling Transformation
- Lighting
- Viewing Transformation
- Clipping
- Projection to 2D space
- Rasterization
- Pixel Shading
- Frame Buffer



The Graphics Pipeline

- 3D Primitives
- Modeling Transformation
- Lighting
- Viewing Transformation
- Clipping
- Projection to 2D space
- Rasterization
- **Pixel Shading**
- Frame Buffer

The Graphics Pipeline



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

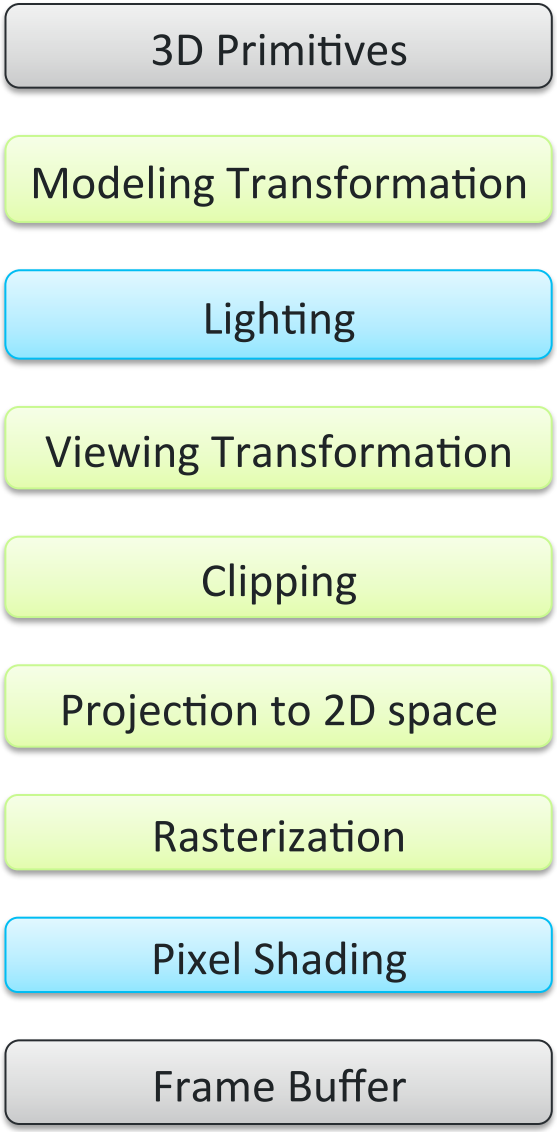
Rasterization

Pixel Shading

Frame Buffer

- We’ve seen this before…

The Graphics Pipeline



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

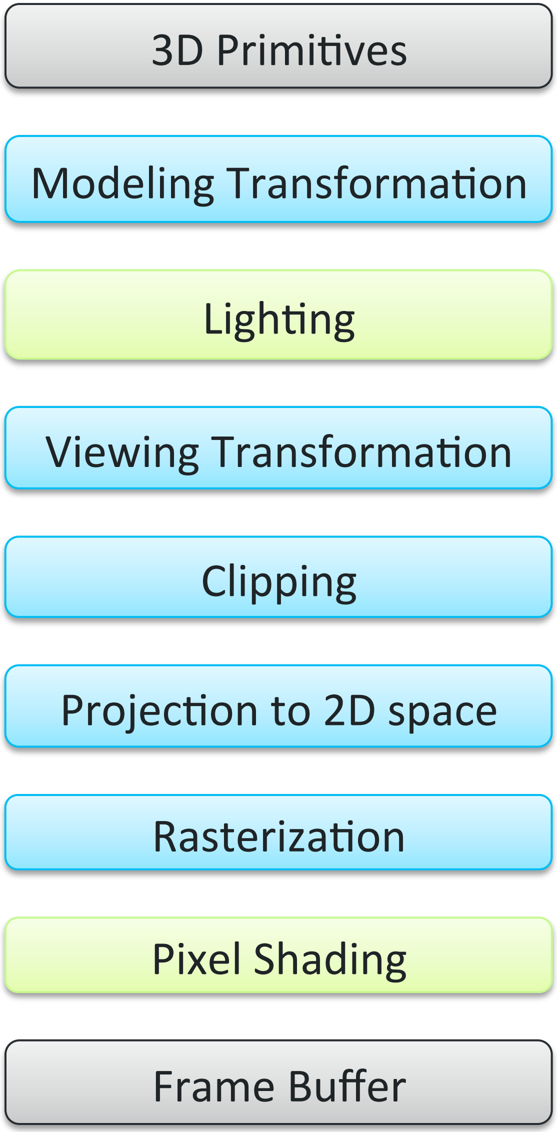
Rasterization

Pixel Shading

Frame Buffer

- Next week!

The Graphics Pipeline



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

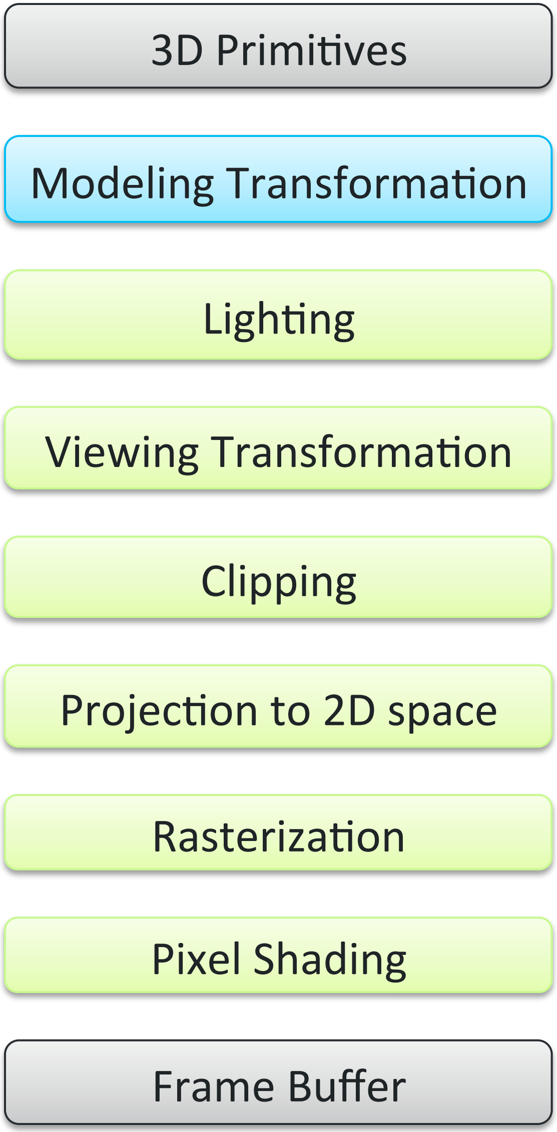
Rasterization

Pixel Shading

Frame Buffer

- The Matrix Stack
- Perspective Transformation
- Clipping and culling
- The Z-Buffer

The Matrix Stack



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

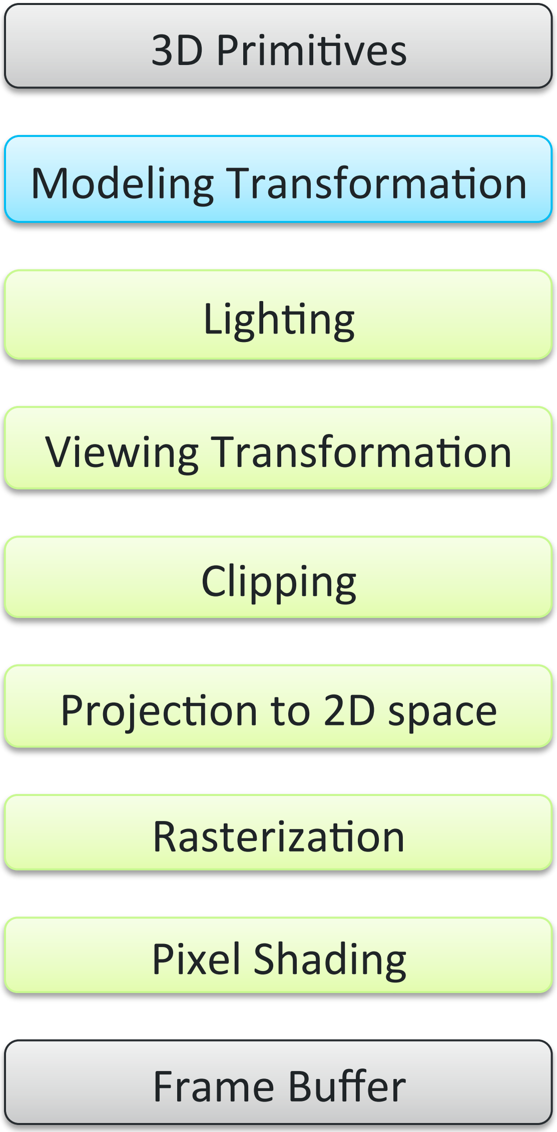
Rasterization

Pixel Shading

Frame Buffer

- What is the matrix stack?

The Matrix Stack



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

Rasterization

Pixel Shading

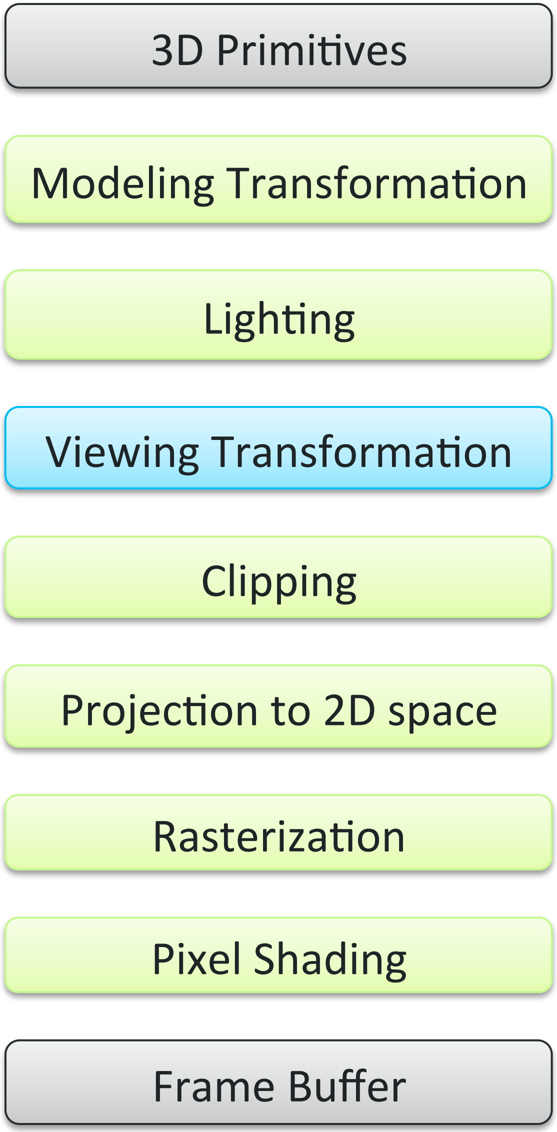
Frame Buffer

- A single transformation is continually updated

- What if I want to make a bunch of transformations and then “forget” I did them?

- pushMatrix(), popMatrix()

Viewing Transformation



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

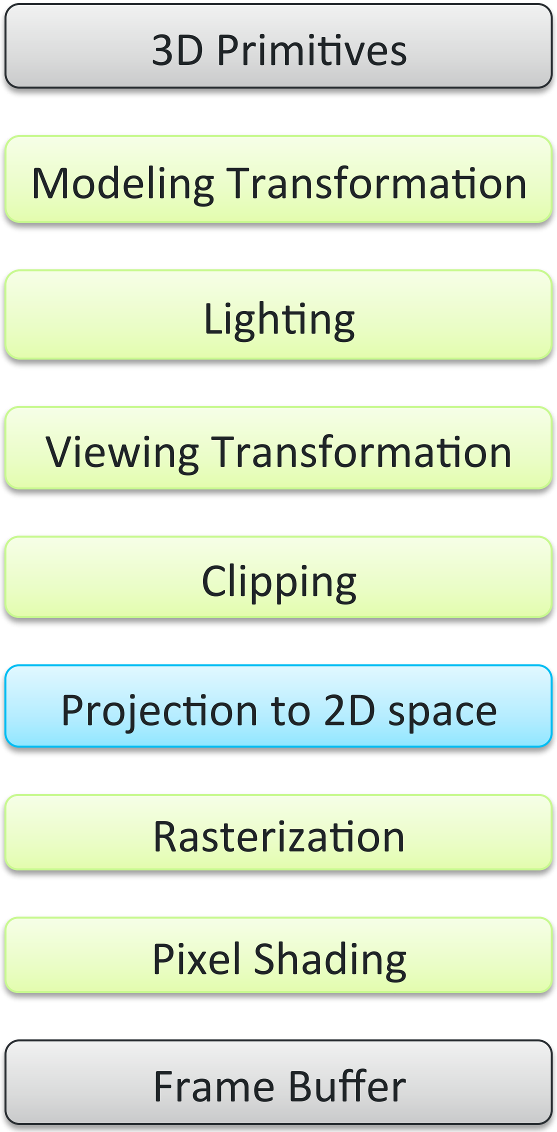
Rasterization

Pixel Shading

Frame Buffer

- Want camera to be the origin of our coordinate system

Perspective Transformation



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

Rasterization

Pixel Shading

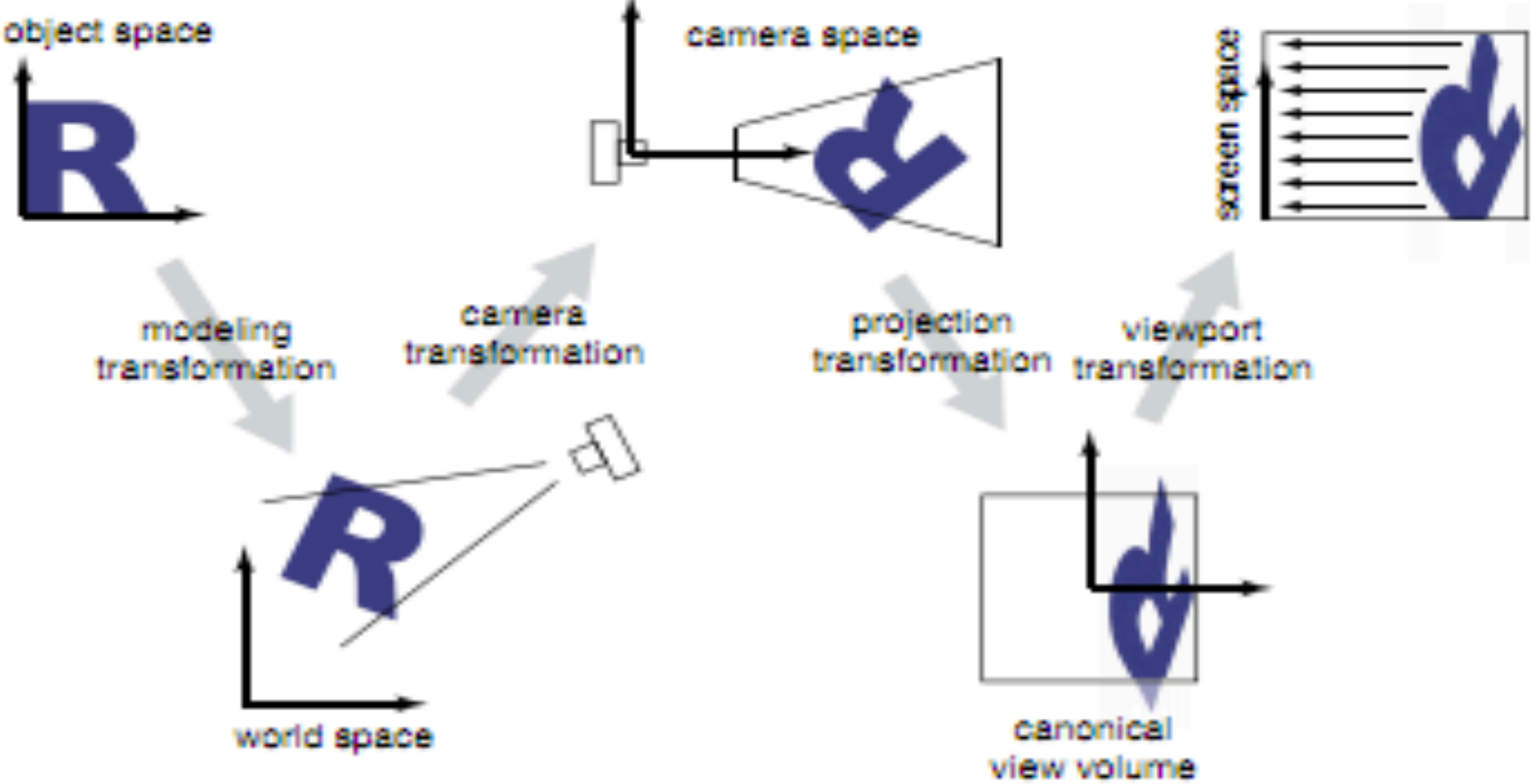
Frame Buffer

- Objects further away should appear smaller
  - User-specified near and far planes
- Viewpoint: how much of the rendered screen should I see?
  - Like a little window over the larger "screen"

Perspective Projection (Several Steps)

Move from world space to camera space coordinates

Parallel projection along z-axis
Window transformation

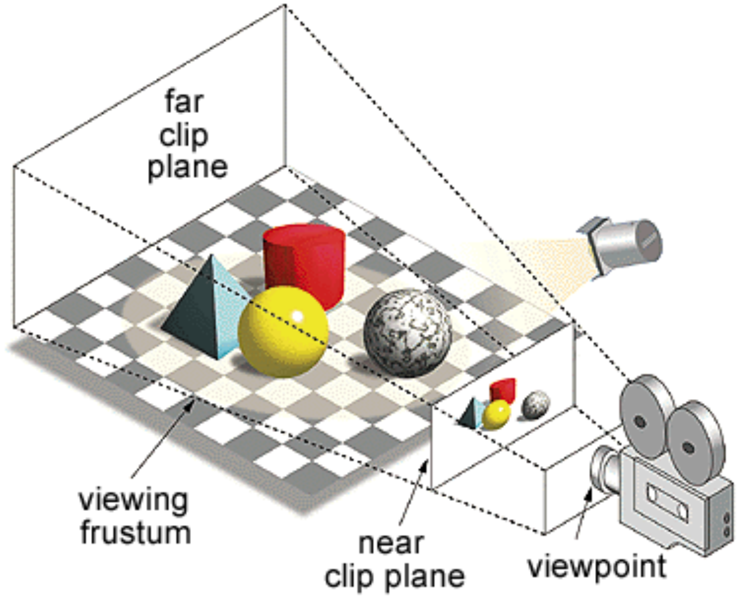


Transform the view frustum to orthographic view volume then to canonical view (-1,1) (so we can do parallel projection)

Picture from the book

Final Equation

$$\begin{pmatrix} x_{pixel} \\ y_{pixel} \\ z_{pixel} \\ 1 \end{pmatrix} = M_{vp} M_{orth} P M_{cam} M_m \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



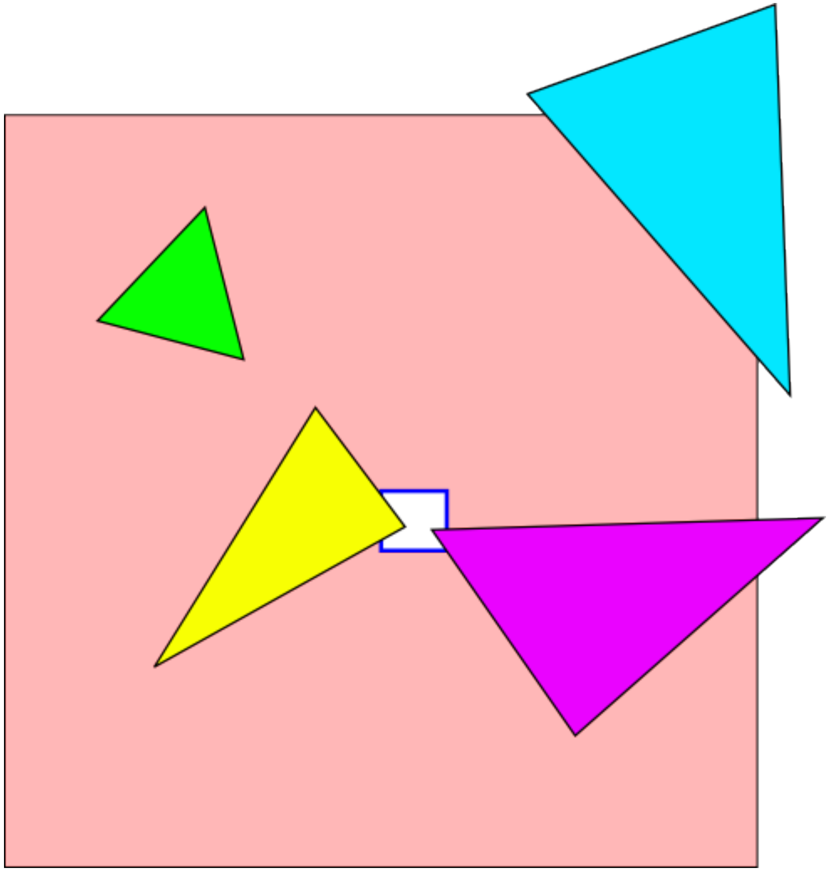
From Computer Desktop Encyclopedia
Reproduced with permission.
© 1998 Intergraph Computer Systems

$$\begin{pmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x - 1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y - 1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -x_c \\ 0 & 1 & 0 & -y_c \\ 0 & 0 & 1 & -z_c \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

M_{vp} M_{orth} P M_{cam}

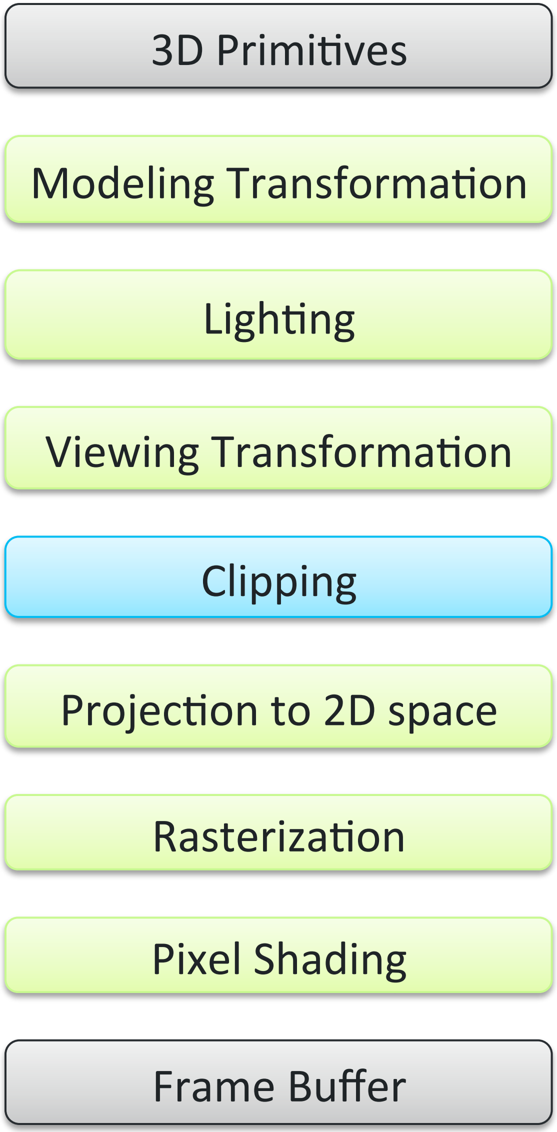
Clipping & Culling

- 3D Primitives
- Modeling Transformation
- Lighting
- Viewing Transformation
- **Clipping**
- Projection to 2D space
- Rasterization
- Pixel Shading
- Frame Buffer



From ryg blog: http://fgiesen.wordpress.com/2011/07/05/a-trip-through-the-graphics-pipeline-2011-part-5/

Clipping & Culling



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

Rasterization

Pixel Shading

Frame Buffer

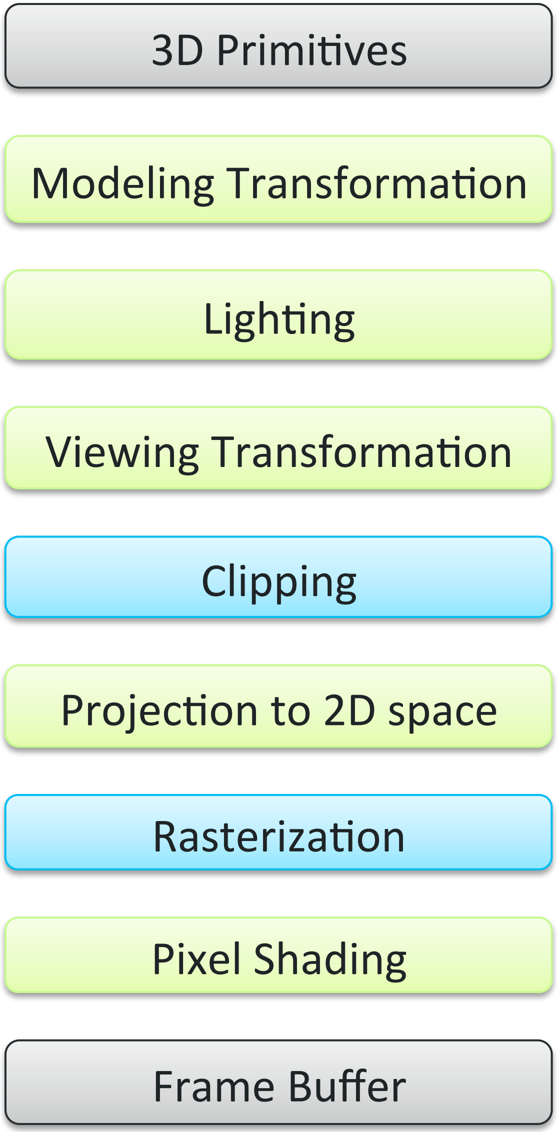
- Why would we want to not render certain objects?

Clipping & Culling

- 3D Primitives
- Modeling Transformation
- Lighting
- Viewing Transformation
- Clipping
- Projection to 2D space
- Rasterization
- Pixel Shading
- Frame Buffer

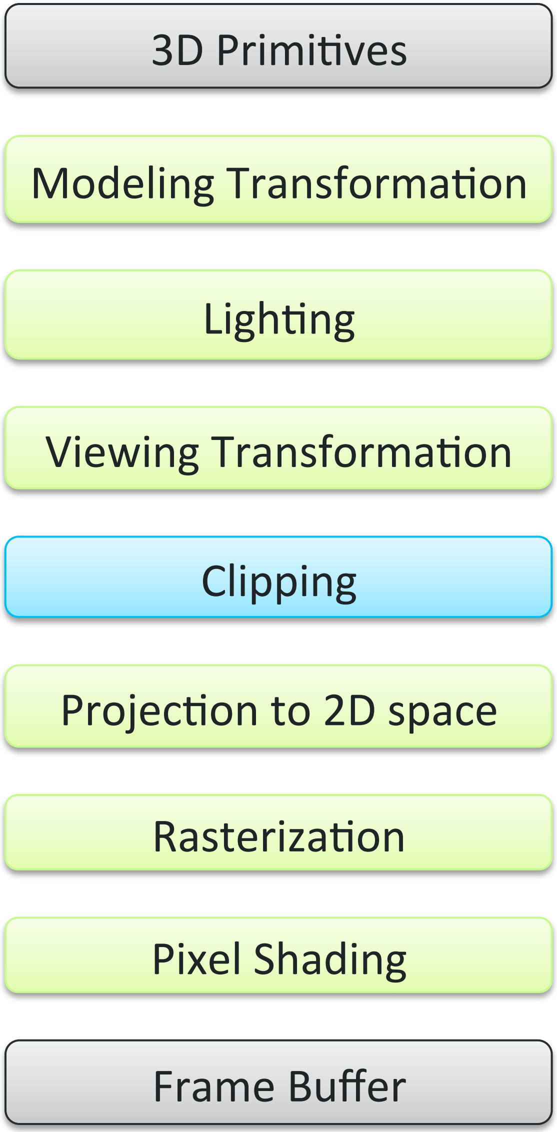
- Why would we want to not render certain objects?
  - Completely outside the viewing area
  - Partially outside the viewing area
  - Facing away from the camera

Clipping & Culling



- Why would we want to not render certain objects?
  - Completely outside the viewing area [**culling**]
  - Partially outside the viewing area [**clipping**]
  - Facing away from the camera [**backface culling**]
  - Blocked by other objects [**occlusion culling**]

Clipping & Culling



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

Rasterization

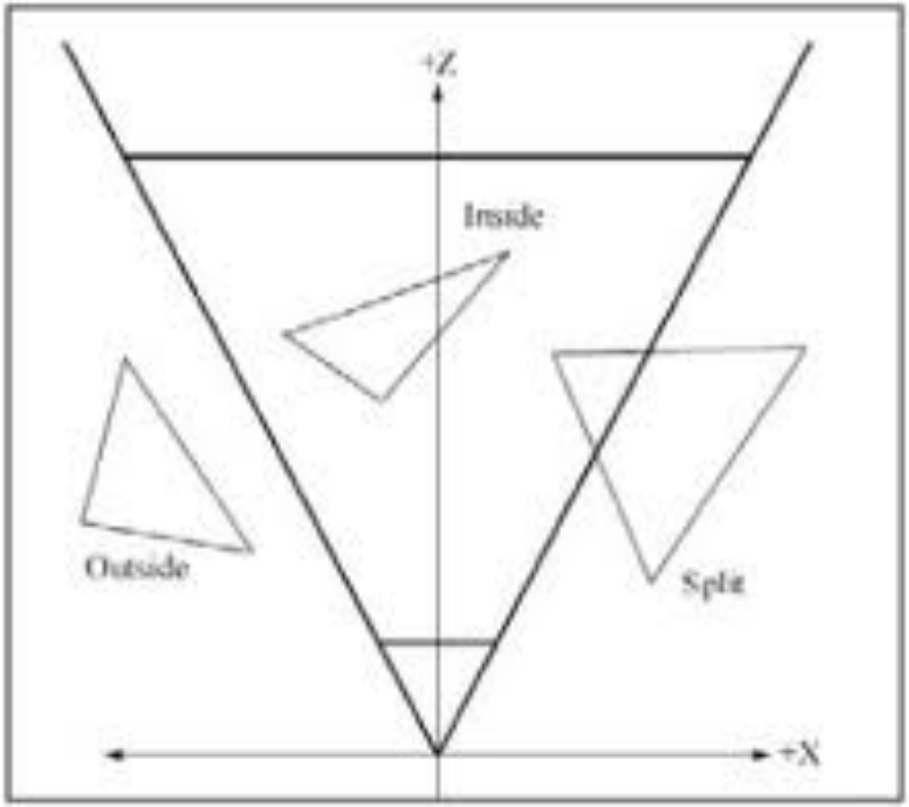
Pixel Shading

Frame Buffer

- Why would we want to not render certain objects?
  - Completely outside the viewing area [**culling**]
  - Partially outside the viewing area [**clipping**]
  - Facing away from the camera [**backface culling**]
  - Blocked by other objects [**occlusion culling**]

Clipping

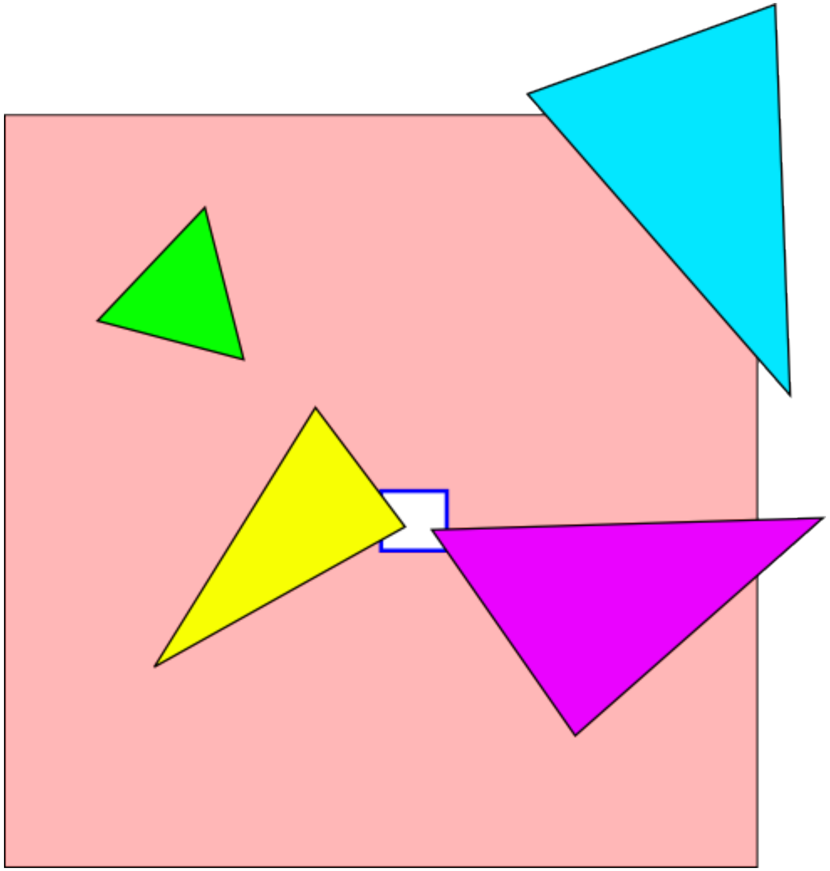
- 6 planes define the viewing frustum
  - Front
  - Back
  - Left
  - Right
  - Top
  - Bottom



From Introduction to 3D Game Programming with DirectX 9.0

Clipping: Algorithm

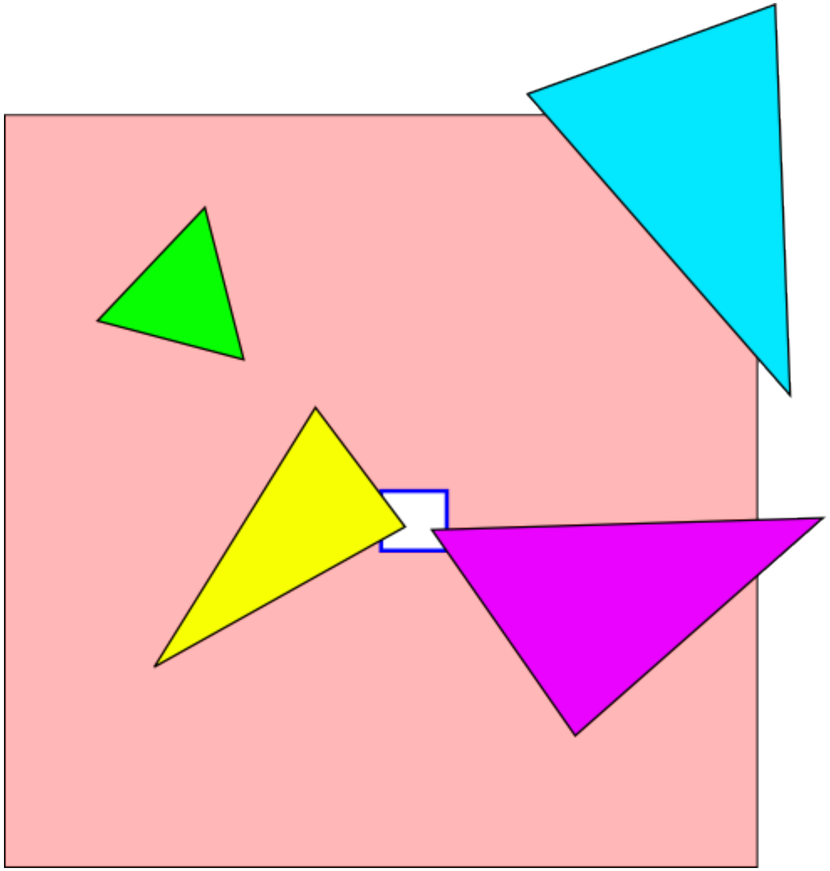
- How do you think we can do this?



From ryg blog: http://fgiesen.wordpress.com/2011/07/05/a-trip-through-the-graphics-pipeline-2011-part-5/

Clipping: Algorithm

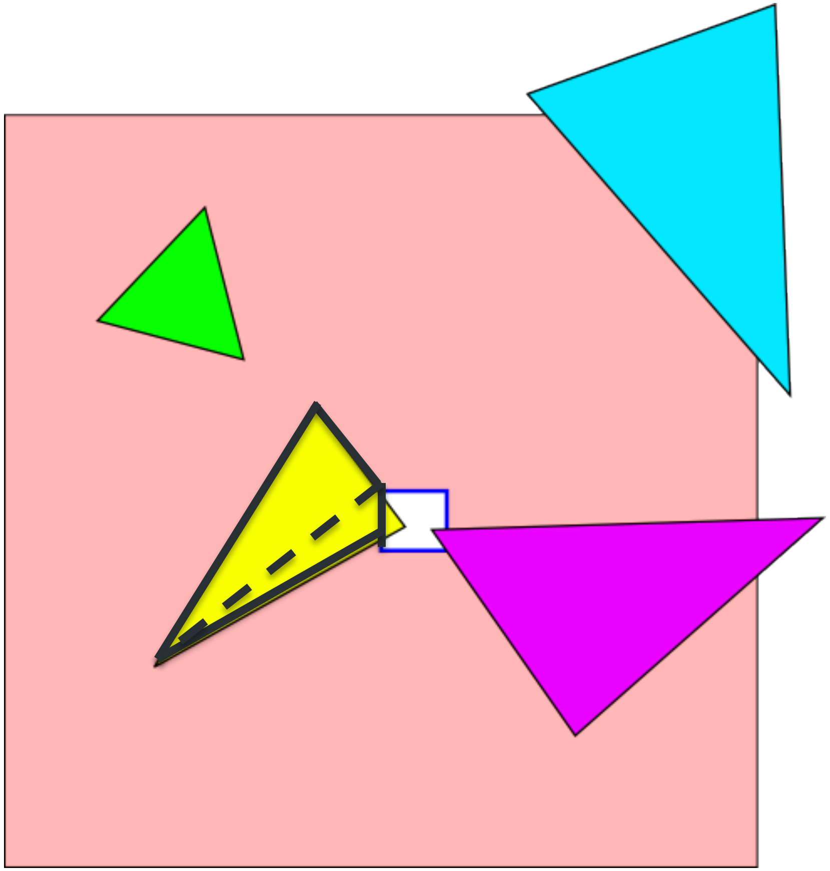
```
for each plane:
  if (t outside plane):
      break
  if (t spans plane):
      clip triangle
      if (quadrilateral):
          break into 2 triangles
```



From ryg blog: http://fgiesen.wordpress.com/2011/07/05/
a-trip-through-the-graphics-pipeline-2011-part-5/

Clipping: Algorithm

```
for each plane:
  if (t outside plane):
        break
  if (t spans plane):
        clip triangle
        if (quadrilateral):
           break into 2 triangles
```

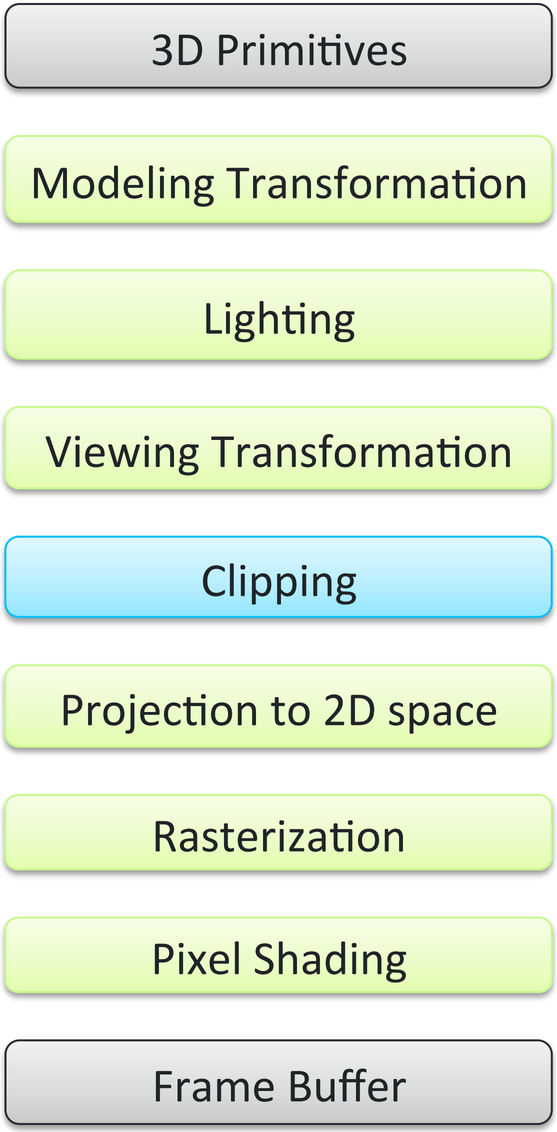


From ryg blog: http://fgiesen.wordpress.com/2011/07/05/a-trip-through-the-graphics-pipeline-2011-part-5/

Clipping: Important Things to Know

- Your graphics API **only** knows how to clip things on a triangle-by-triangle basis
- While optimized, this is still fairly time consuming
- How can we make it perform better?

Clipping & Culling



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

Rasterization

Pixel Shading

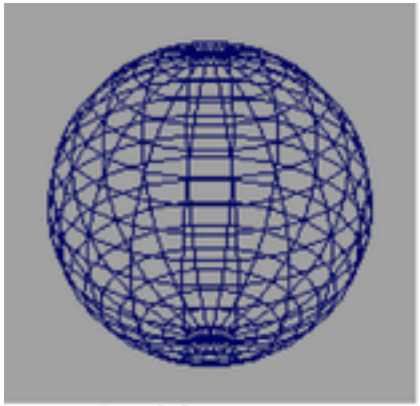
Frame Buffer

- Why would we want to not render certain objects?
  - Completely outside the viewing area [**culling**]
  - Partially outside the viewing area [**clipping**]
  - Facing away from the camera [**backface culling**]
  - Blocked by other objects [**occlusion culling**]

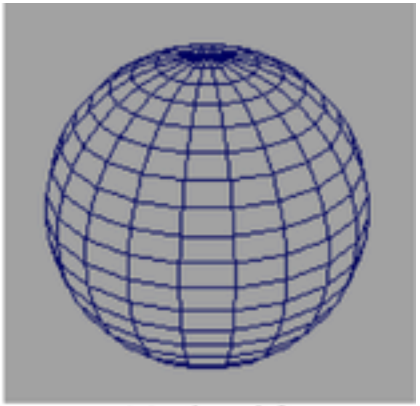
Occlusion Culling

- Idea: we only need to draw what we can see from the camera
  - Works with single-sided polygons

- How do we know if a triangle is facing away?



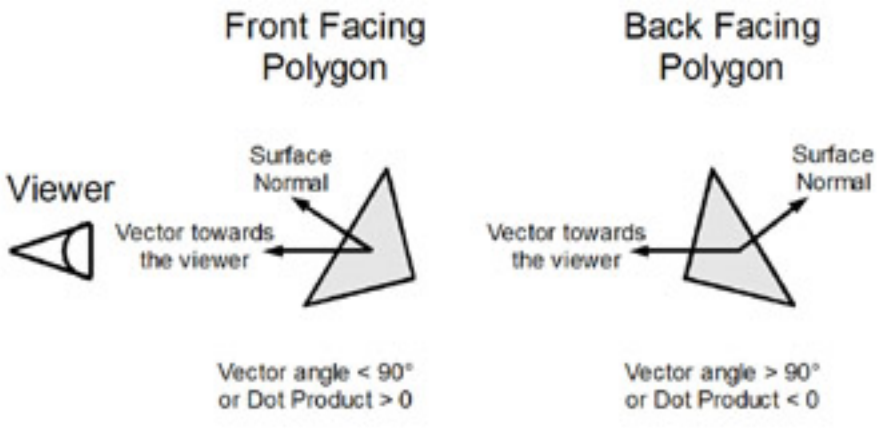
Backfaces



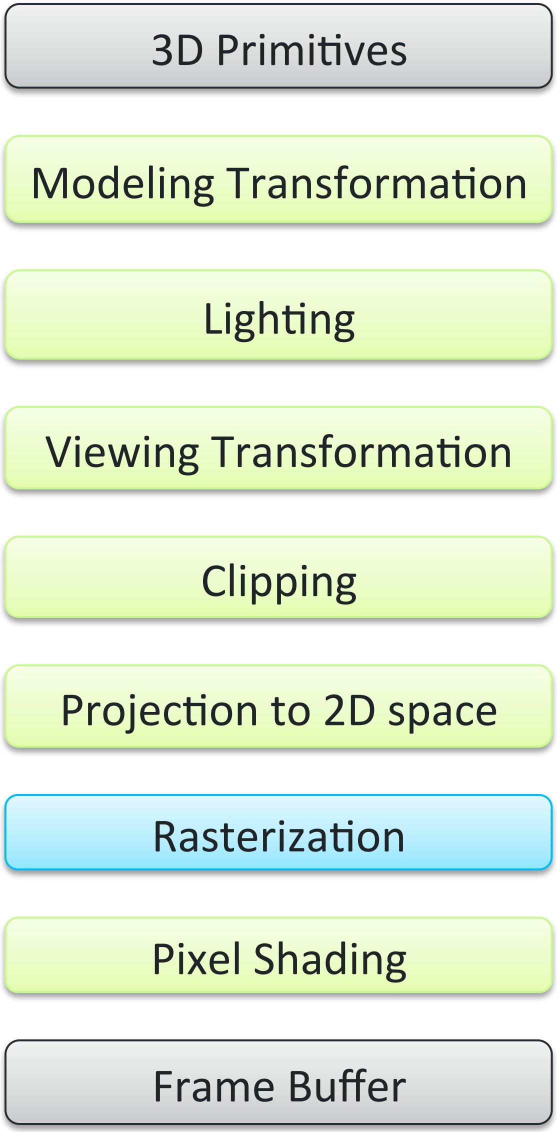
No backfaces

Occlusion Culling

- Use the surface normal!
- Dot product of surface normal with eye vector
  - if positive: front facing
  - if negative: back facing


Front Facing Polygon
Back Facing Polygon
Viewer
Surface Normal
Vector towards the viewer
Surface Normal
Vector towards the viewer
Vector angle < 90° or Dot Product > 0
Vector angle > 90° or Dot Product < 0

Clipping & Culling



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

Rasterization

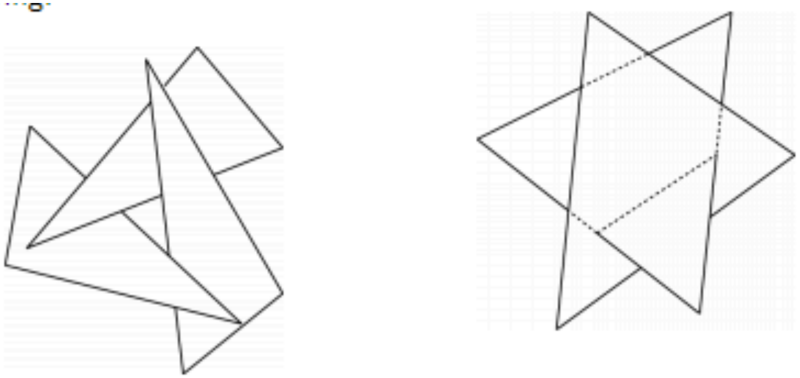
Pixel Shading

Frame Buffer

- Why would we want to not render certain objects?
  - Completely outside the viewing area [**culling**]
  - Partially outside the viewing area [**clipping**]
  - Facing away from the camera [**backface culling**]
  - Blocked by other objects [**occlusion culling**]

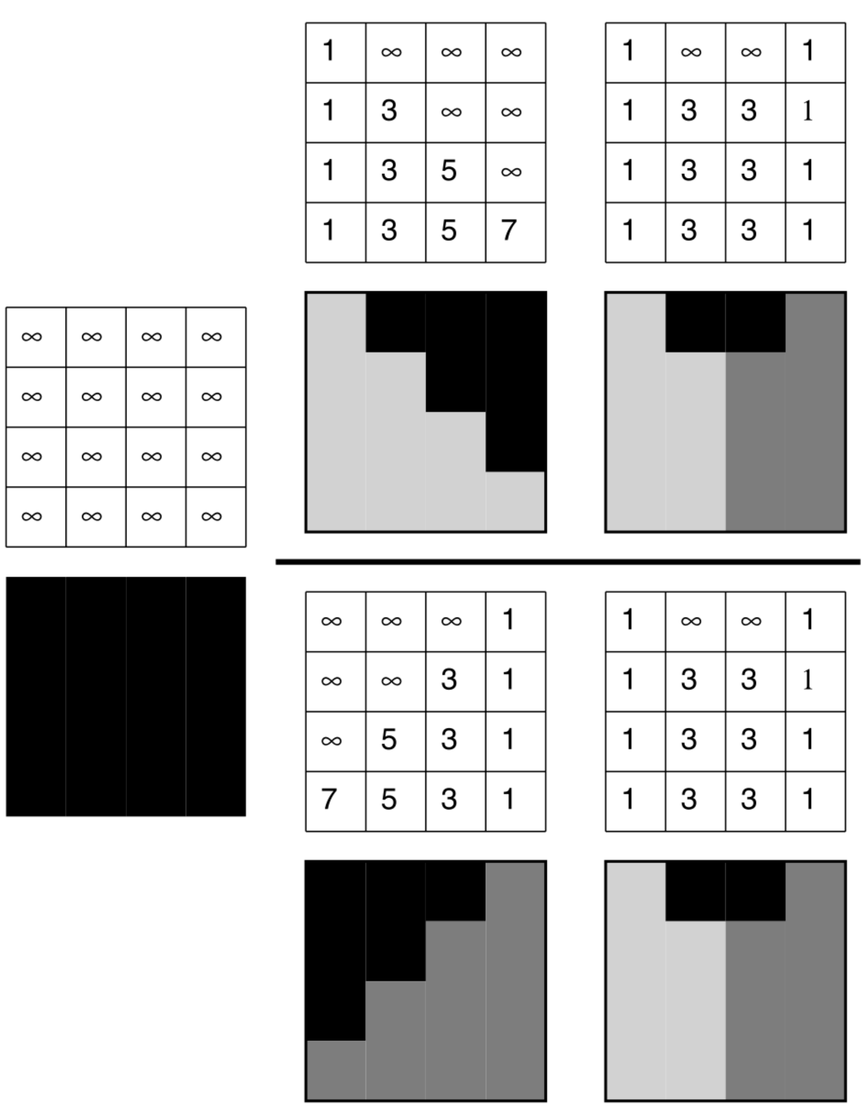
The Z Buffer

- If one object is in front of another, we only need to render the frontmost one!
- Polygons might be clipping through each other
- How do we deal with this?



The Z Buffer

- Buffer on the graphics card with *n* bit-depth
  - n: 8 – 32 bits (higher is better!)
- Pros and cons of this vs. what we do in raytracing?
- Problem: z-fighting



∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞

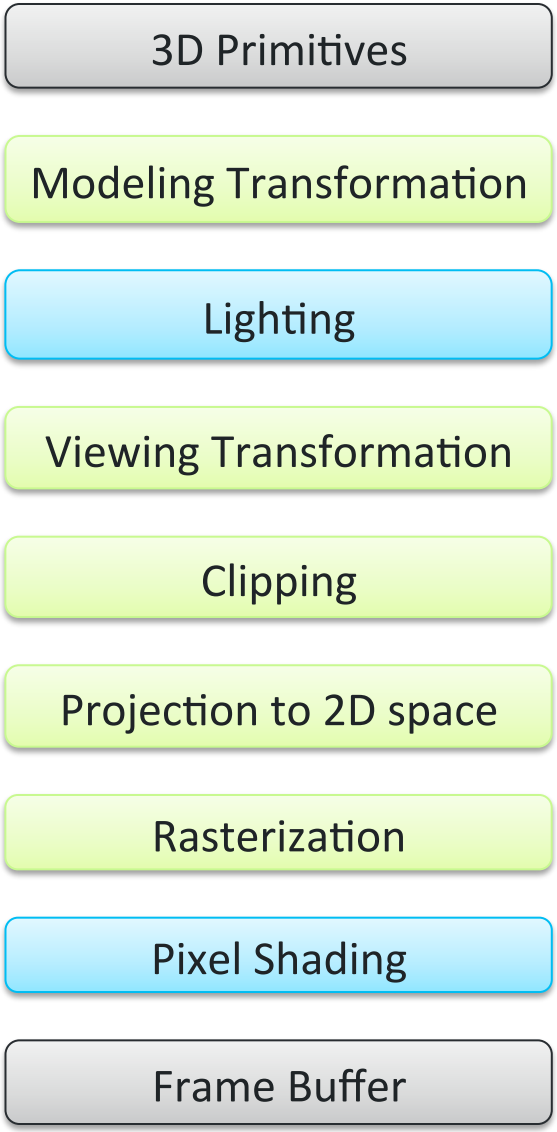
1	∞	∞	∞
1	3	∞	∞
1	3	5	∞
1	3	5	7

1	∞	∞	1
1	3	3	1
1	3	3	1
1	3	3	1

∞	∞	∞	1
∞	∞	3	1
∞	5	3	1
7	5	3	1

1	∞	∞	1
1	3	3	1
1	3	3	1
1	3	3	1

The Graphics Pipeline



3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Clipping

Projection to 2D space

Rasterization

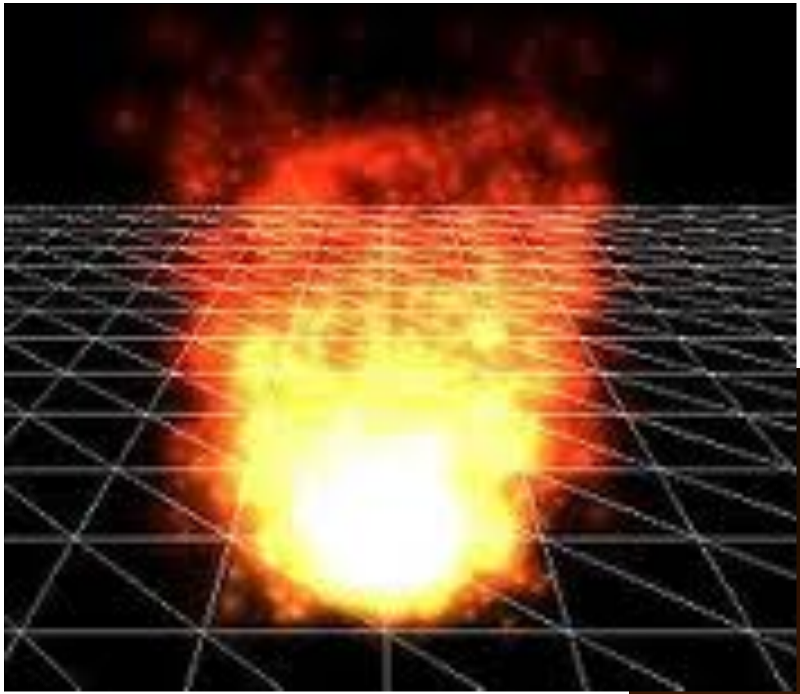
Pixel Shading

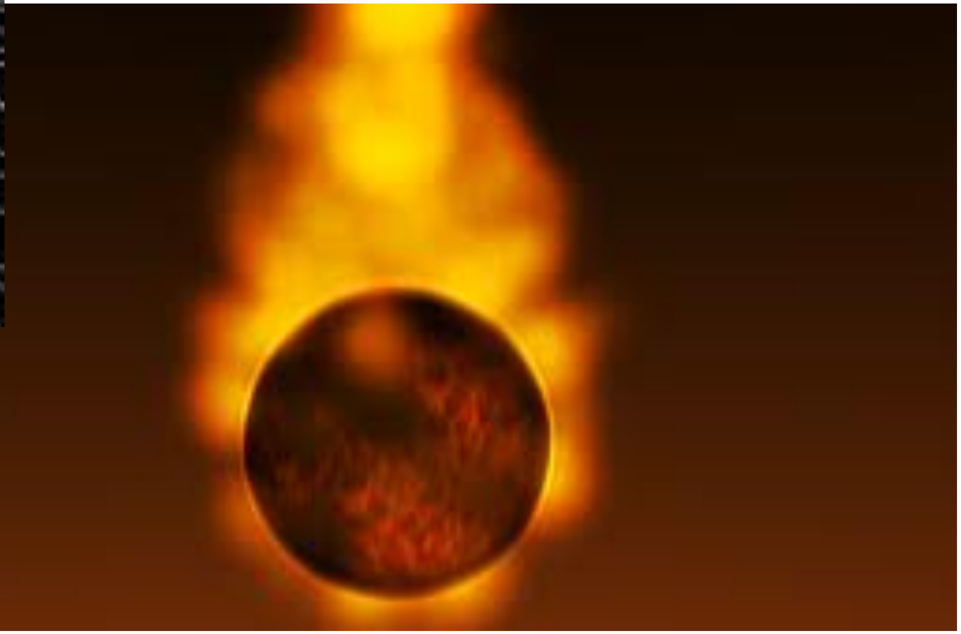
Frame Buffer

- Next week!

PARTICLE SYSTEMS

Uses: Fire

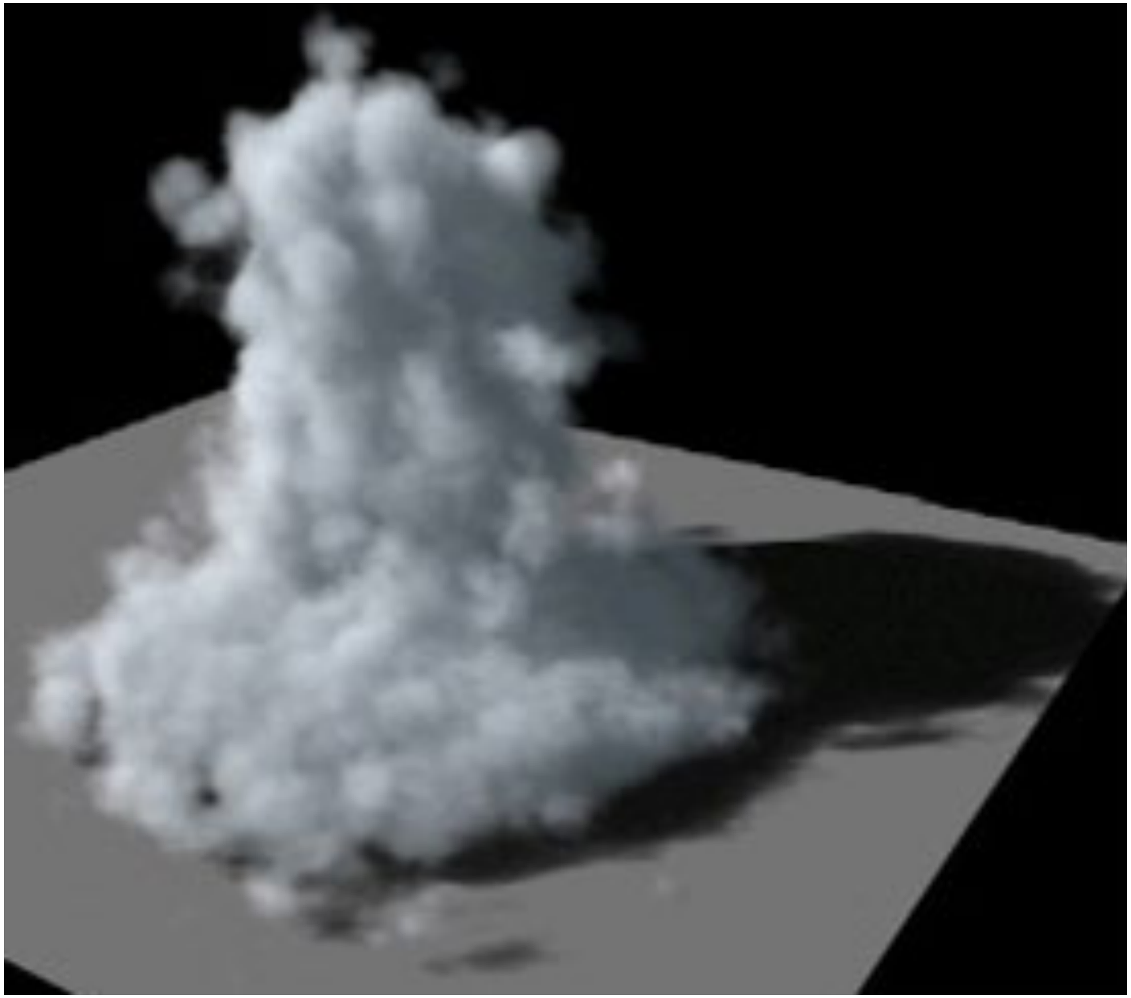




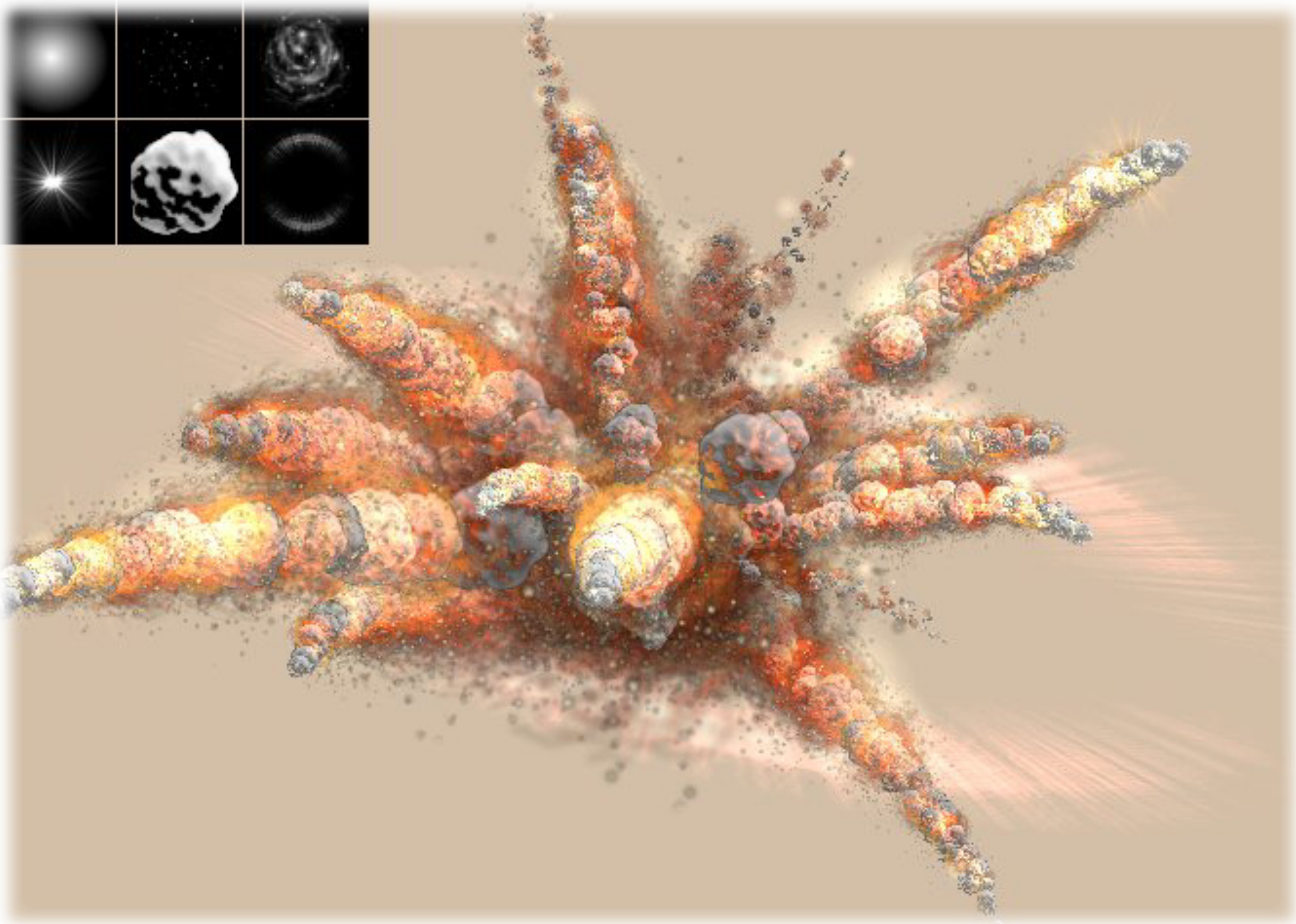
Uses: Liquids

Uses: Fireworks

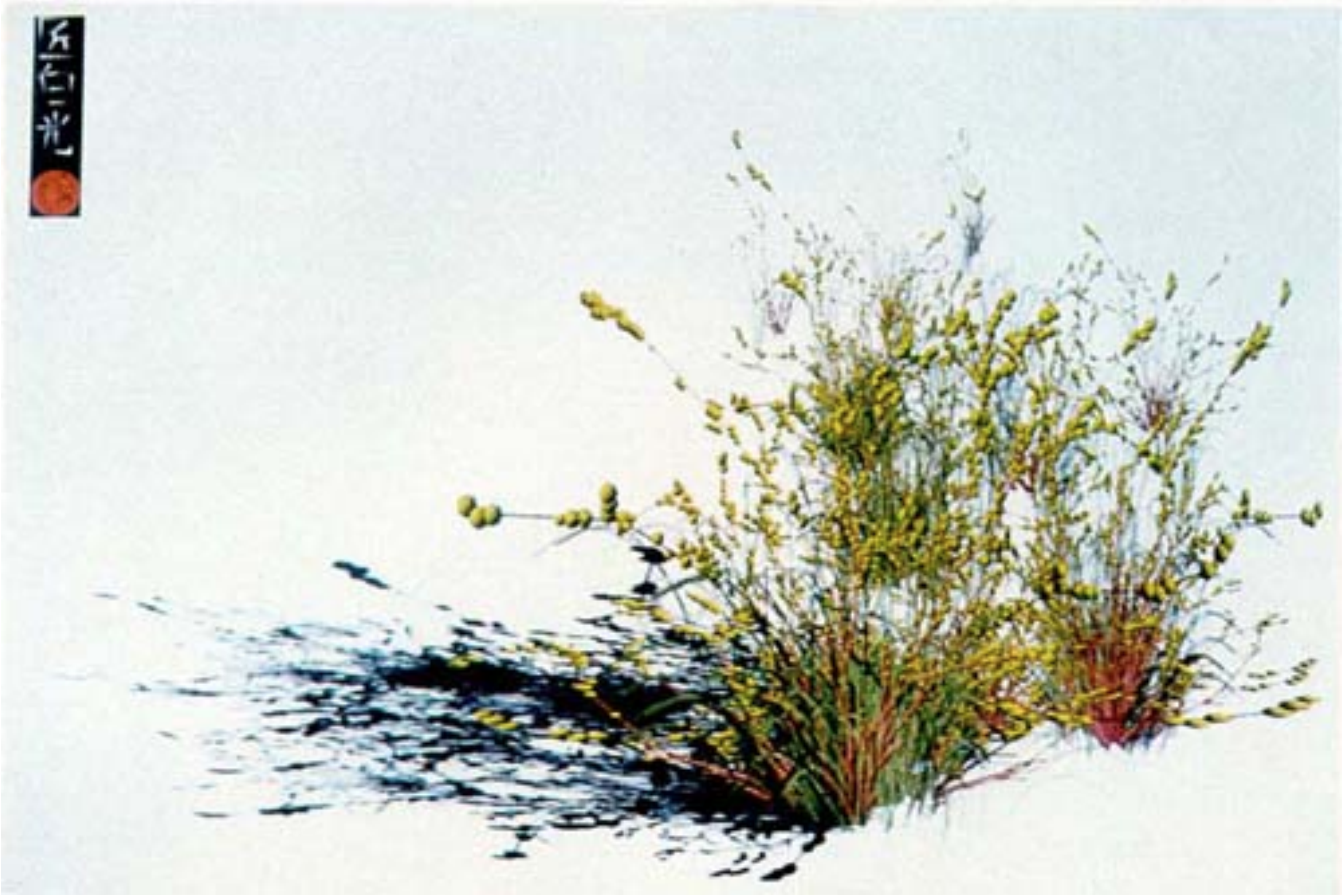
Uses: Clouds



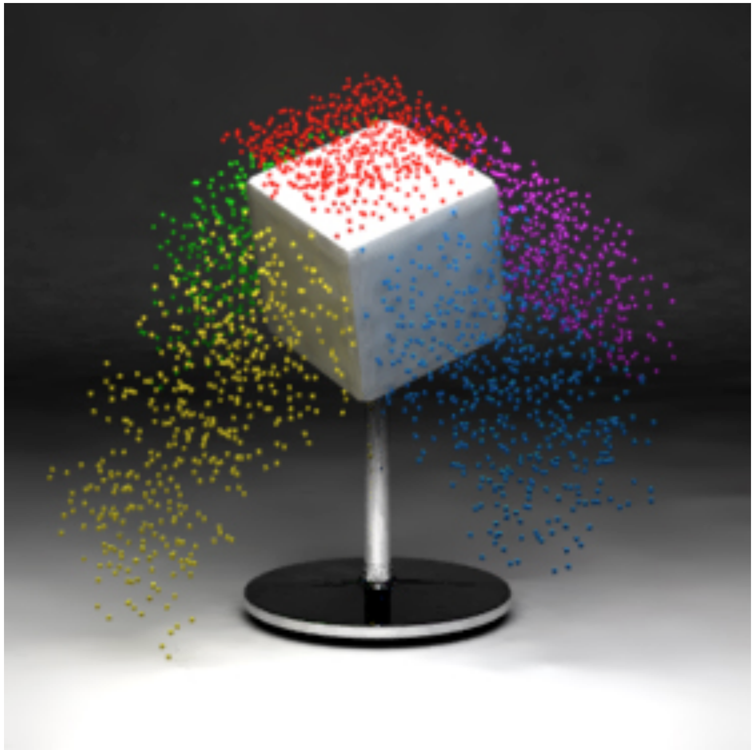
Uses: Explosions



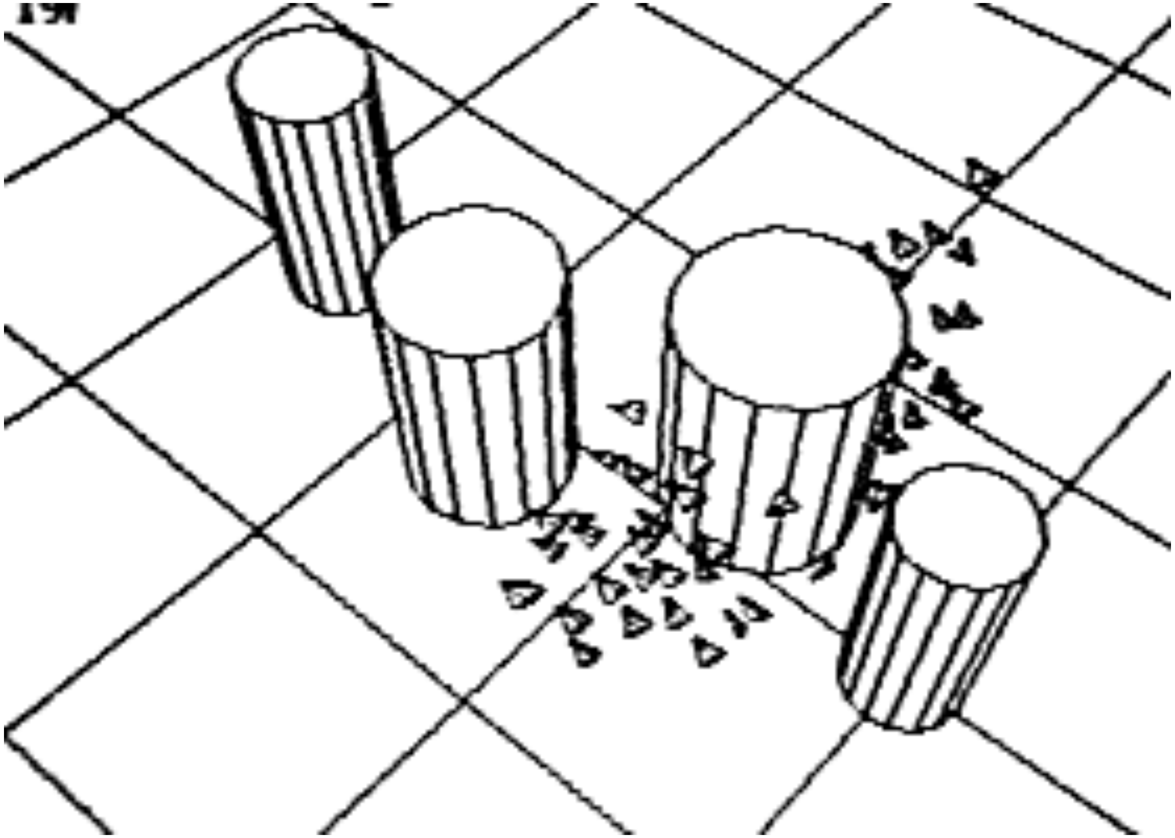
Uses: Grass



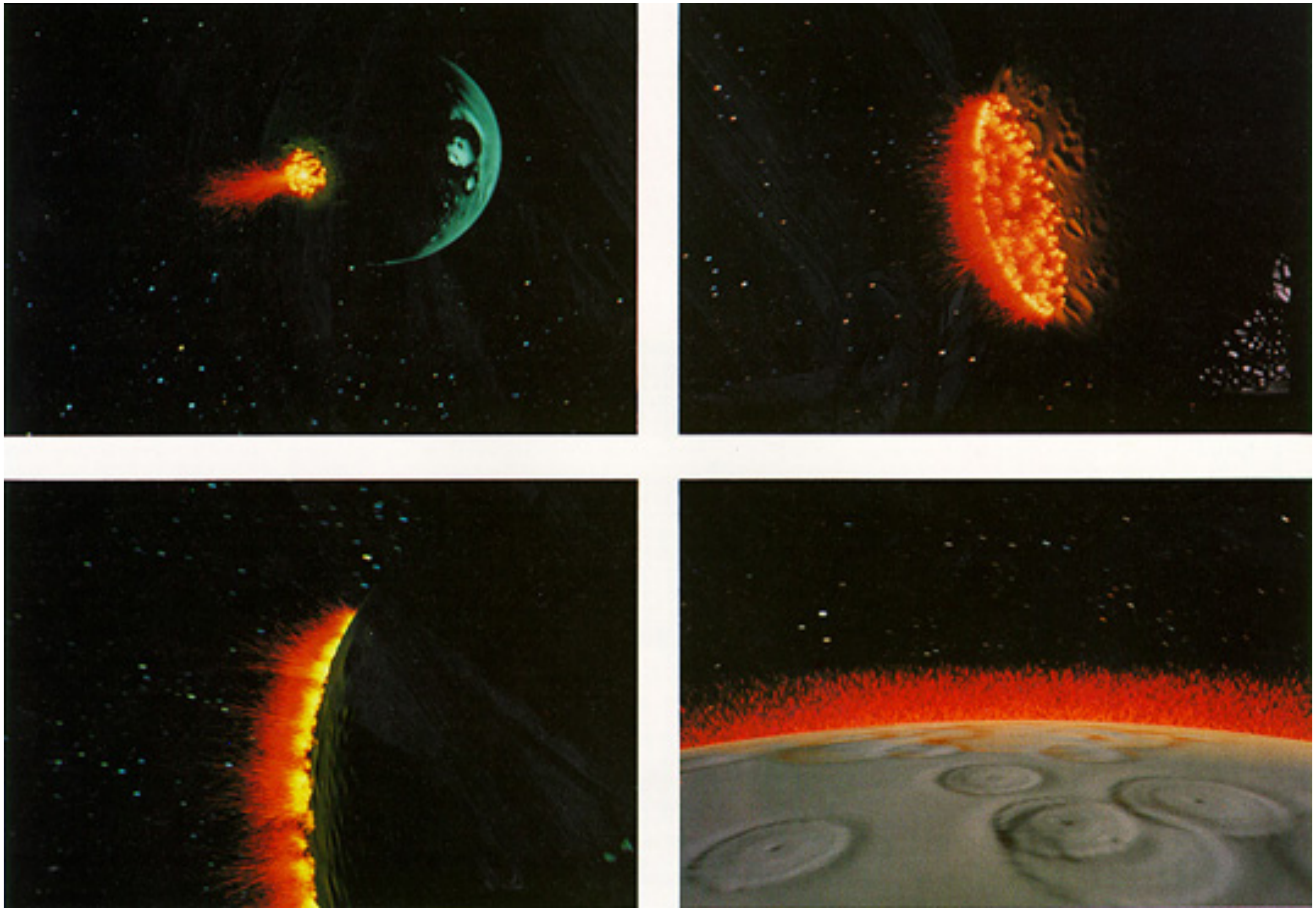
Uses: Hair



Uses: Flocking



First Use: Star Trek!



What is a Particle System?

- Emitter
  - Position
  - Surface
  - (Range of) directions

- Particles
  - Size
  - Shape
  - Position
  - Velocity (speed, direction)
  - Color/texture
  - Transparency
  - Lifetime

What is a Particle System?

- Emitter
  - Position
  - Surface
  - (Range of) directions

- Particles
  - Size
  - Shape
  - Position
  - Velocity (speed, direction)
  - Color/texture
  - Transparency
  - Lifetime

Others??

The Particle Life Cycle

- Generation
- Dynamics
- Death

The Particle Life Cycle

- Generation
  - Spawned by the emitter
  - Initial attributes (position, direction) based on emitter properties
- Dynamics
- Death

The Particle Life Cycle

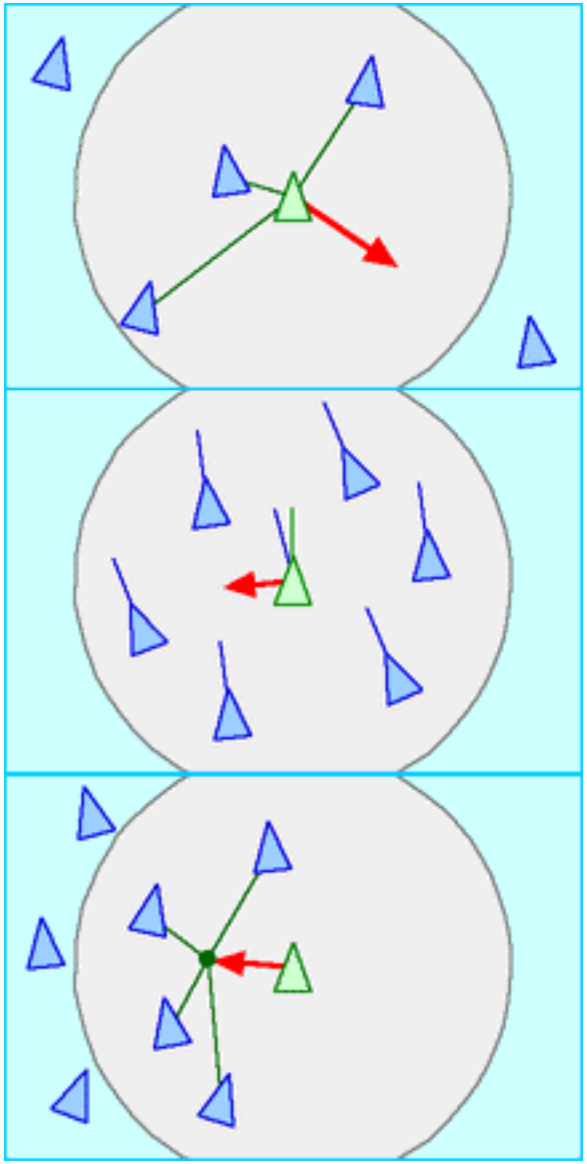
- Generation
  - Spawned by the emitter
  - Initial attributes (position, direction) based on emitter properties
- Dynamics
  - Attributes change over time or in response to events
  - Particles often independent of each other
- Death

The Particle Life Cycle

- Generation
  - Spawned by the emitter
  - Initial attributes (position, direction) based on emitter properties
- Dynamics
  - Attributes change over time or in response to events
  - Particles often independent of each other
- Death
  - When an attribute reaches a threshold
  - Due to external event

Particle Dynamics: Boids

- Three independent behaviors affecting velocity and direction
  - Separation
  - Alignment
  - Cohesion
- Only care about local flockmates
- Combined independent behaviors to simulate real life!



Your Assignment: Build a Particle System