# Game Rendering

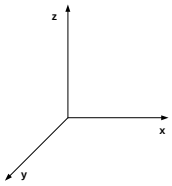
Doron Nussbaum



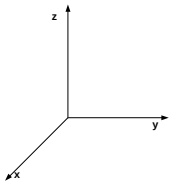
# Agenda

- Rendering Geometry
- Graphics Pipeline

# 3D Coordinate System




Left-hand Coordinate System

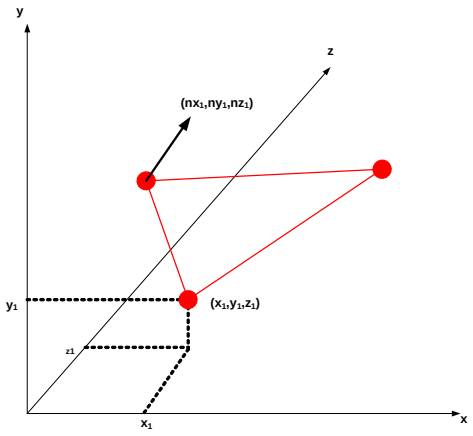



Right-hand Coordinate System

- Left-hand coordinate system is normally used



# Vertex Coordinates and Normals

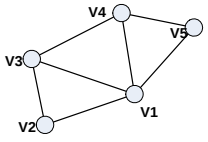


# 3D Geometry

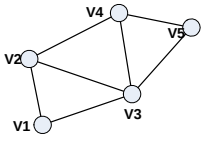
- Using triangles to represent a surface
  - Simple
  - Always in a plane
  - Convex
  - No issues with self intersecting
  - Easy to create many tessellations of the space



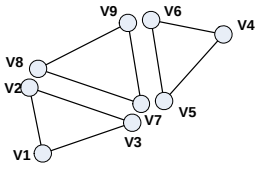
# Geometric Representation



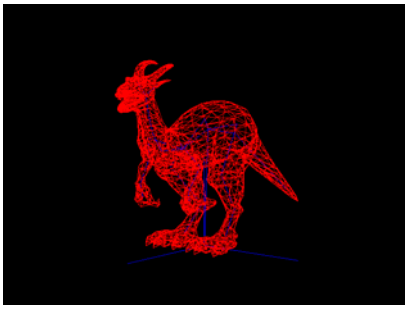

**Triangle Fan**

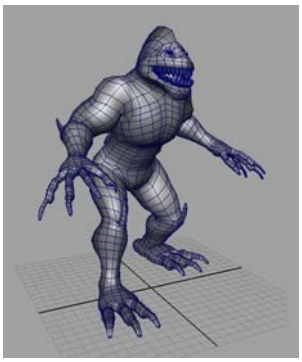



**Triangle Strip**




**Triangle List**



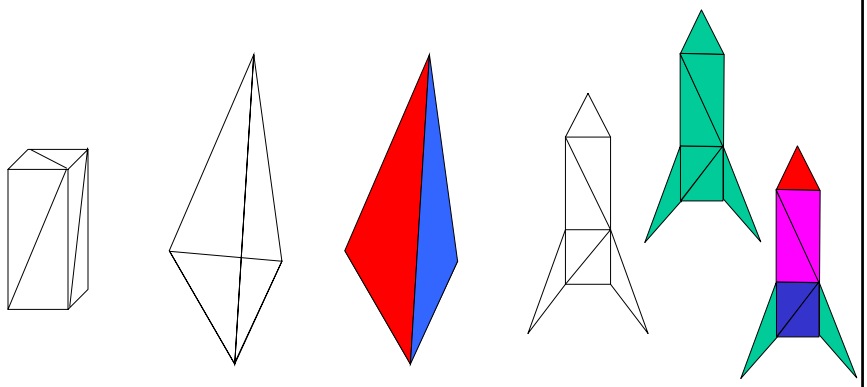


http://www.boblycat.org/~malc/okaar/outdoc/aquarex_wire.gif

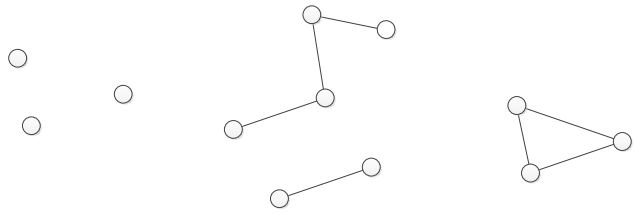
http://www.a3dam.com/images/creature_wire_01.jpg



# 3D Objects – Illusion?



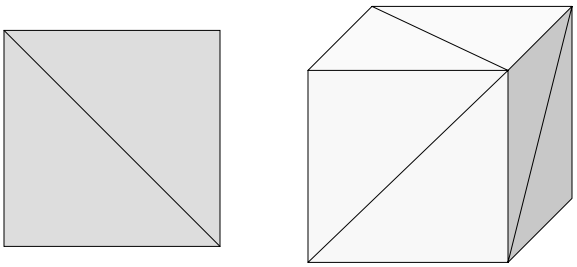
# Geometric Primitives





# Quads

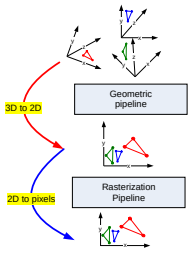
- Used to display 3D objects
  - Flexible
  - Suitable for pictures (discussed later)





**Points – 0D**

# 3D Graphics Pipeline

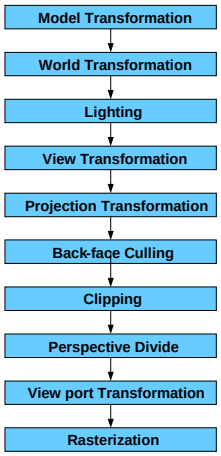



- Geometric Processing
  - Vector world
    - Processing vertices
    - Floating point arithmetic
- Rasteriztion Pipeline
  - Processing Pixels
  - Integer Arithmetic



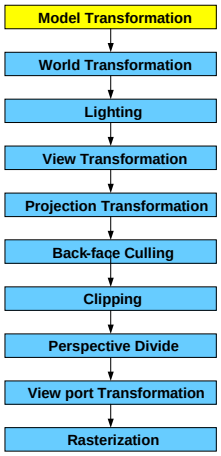
# 3D Rendering Pipeline

- Creates a 2D image from the 3D model
- Converts from vector data representation to pixel representations
- One way



# 3D Rendering Pipeline

**Transforms model coordinates locally**

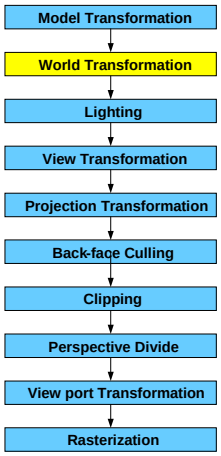





# 3D Rendering Pipeline

Transforms model coordinates locally

**Transform into 3D world coordinates**

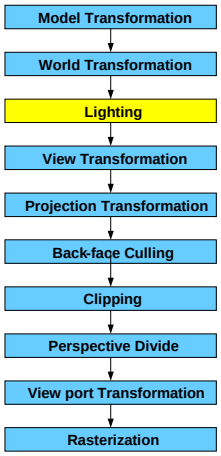


# 3D Rendering Pipeline

Transforms model coordinates locally

Transform into 3D world coordinates system

**Illuminate the objects**





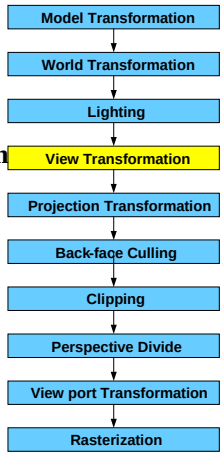

# 3D Rendering Pipeline

Transforms model coordinates locally

Transform into 3D world coordinates system

Illuminate the objects

**Transform into camera coordinates system**



# 3D Rendering Pipeline

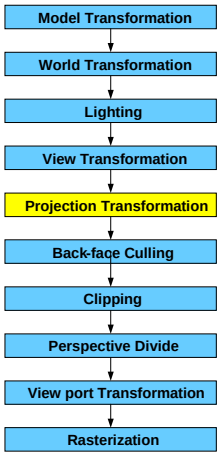
Transforms model coordinates locally

Transform into 3D world coordinates system

Illuminate the objects

Transform into camera coordinate system

**Trans. into view plane coordinate system**






# 3D Rendering Pipeline

Transforms model coordinates locally

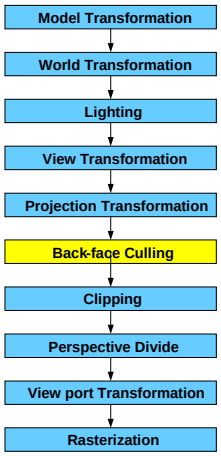
Transform into 3D world coordinates system

Illuminate the objects

Transform into camera coordinate system

Trans. into view plane coordinate system

**Remove back facing triangles**



# 3D Rendering Pipeline

Transforms model coordinates locally

Transform into 3D world coordinates system

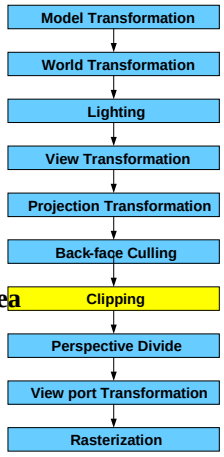
Illuminate the objects

Transform into camera coordinate system

Trans. into view plane coordinate system

Remove back facing triangles

**Clip primitives outside camera viewing area**






# 3D Rendering Pipeline

Transforms model coordinates locally

Transform into 3D world coordinates system

Illuminate the objects

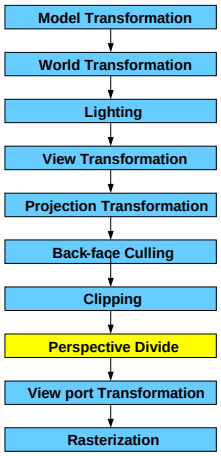
Transform into camera coordinate system

Trans. into view plane coordinate system

Remove back facing triangles

Clip primitives outside camera viewing area

**Converting to homogenous**



# 3D Rendering Pipeline

Transforms model coordinates locally

Transform into 3D world coordinates system

Illuminate the objects

Transform into camera coordinate system

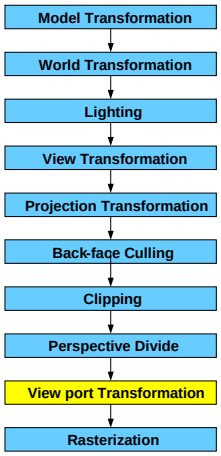
Trans. into view plane coordinate system

Remove back facing triangles

Clip primitives outside camera viewing area

Converting to homogenous

**Transform into view port (screen) coord.**






# 3D Rendering Pipeline

Transforms model coordinates locally

Transform into 3D world coordinates system

Illuminate the objects

Transform into camera coordinate system

Trans. into view plane coordinate system

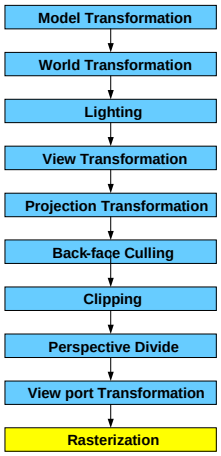
Remove back facing triangles

Clip primitives outside camera viewing area

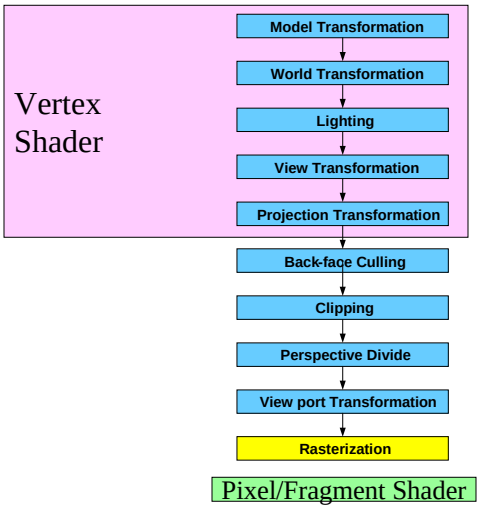
Converting to homogenous

Transform into view port (screen) coord.

**Draw pixels (textures, …)**



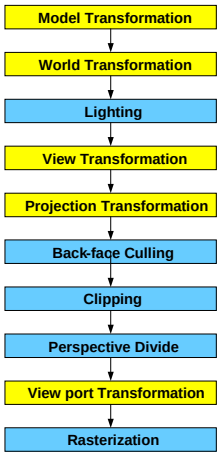
# 3D Shaders Pipeline





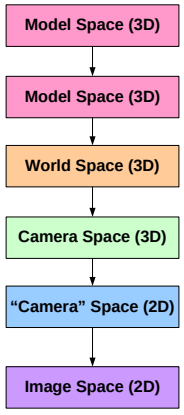

# Transformation Pipeline

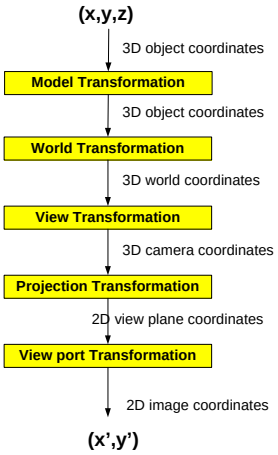
Transformations maps one coordinate system into another coordinate system



# Transformation Pipeline

Transformations maps one coordinate system into another coordinate system



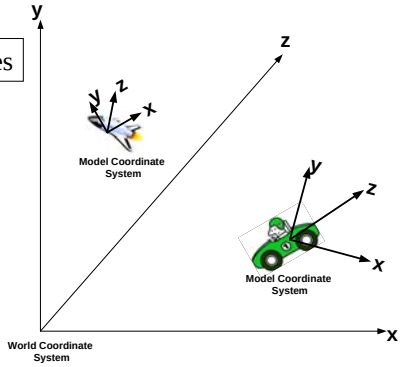






# World Transformation

Model coordinates ➔ World Coordinates

- Achieved using
  - Rotations
  - Translations
  - Scaling

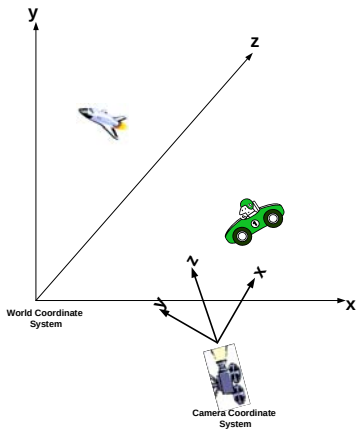


# View Transformation

World coordinates ➔ Camera Coordinates

- Achieved using
  - Rotations
  - Translations
  - Scaling

- Define by
  - Camera position
  - Look at vector
  - Up Vector





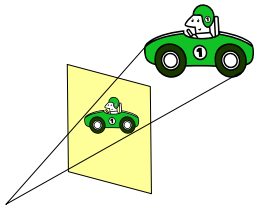

# Rotation Slides

- Different class

# Projections

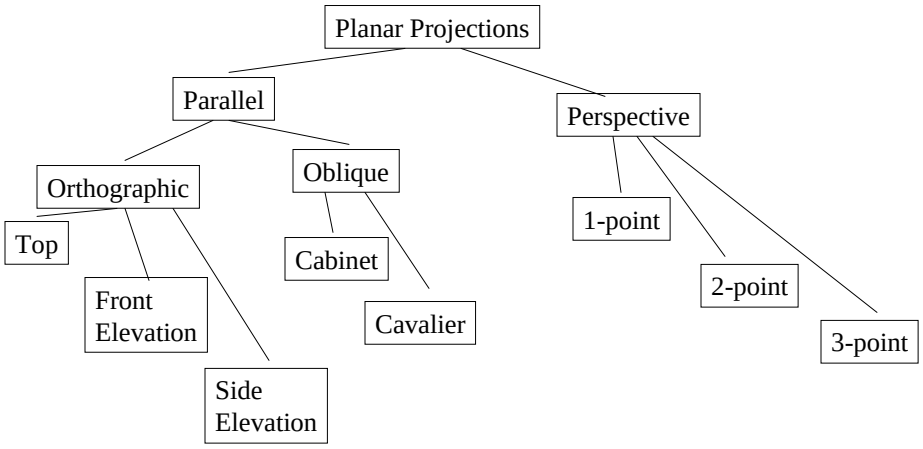
Purpose: Reduce the space dimension (3D ➔ 2D)

Parallel Projections
Perspective Projections

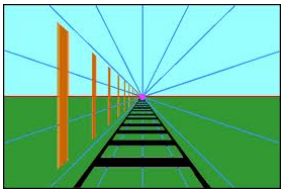




# Taxonomy of Projections



# Vanishing Points

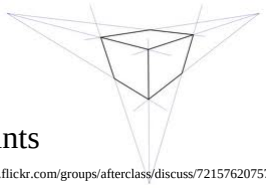


1 point

highlandtechnology.org

2 points

http://kingfishers.ednet.ns.ca/art/grade10/drawing/perspective3.html



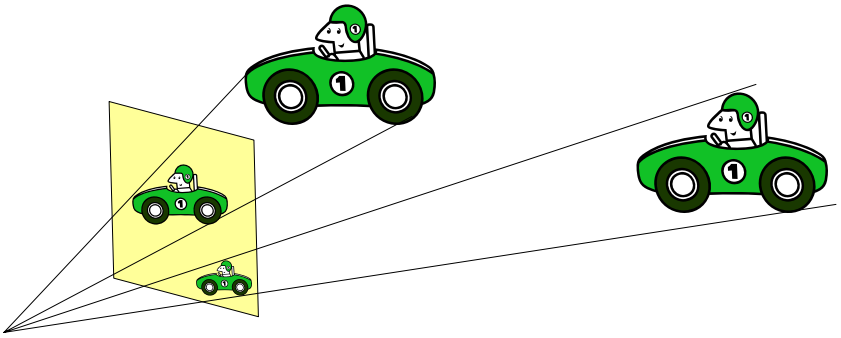
3 points

http://www.flickr.com/groups/afterclass/discuss/72157620757968823/

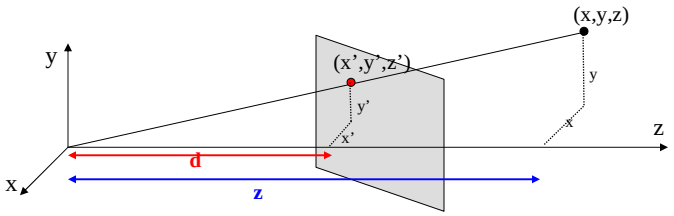


# Perspective Projection

- Maps points onto view plane via lines emanting from a single view point
- Further objects appear smaller



# Perspective Projection



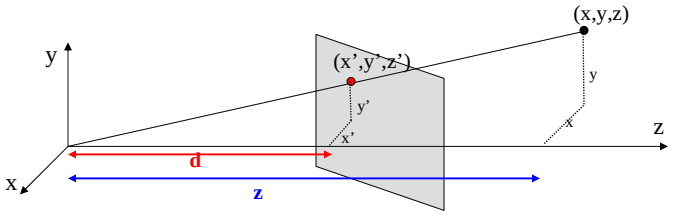

$$\frac{x'}{d} = \frac{x}{z} \Rightarrow x' = \frac{x}{\frac{z}{d}}$$

$$\frac{y'}{d} = \frac{y}{z} \Rightarrow y' = \frac{y}{\frac{z}{d}}$$

$$\frac{z'}{d} = \frac{z}{z} \Rightarrow z' = d = \frac{z}{\frac{z}{d}}$$



# Perspective Projection



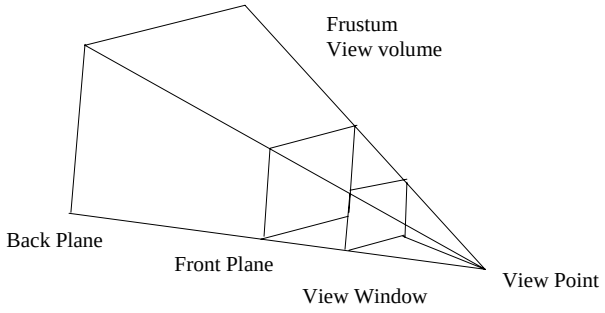

$$\frac{x'}{d} = \frac{x}{z} \Rightarrow x' = \frac{x}{\frac{z}{d}}$$

$$\frac{y'}{d} = \frac{y}{z} \Rightarrow y' = \frac{y}{\frac{z}{d}}$$

$$\frac{z'}{d} = \frac{z}{z} \Rightarrow z' = d = \frac{z}{\frac{z}{d}}$$

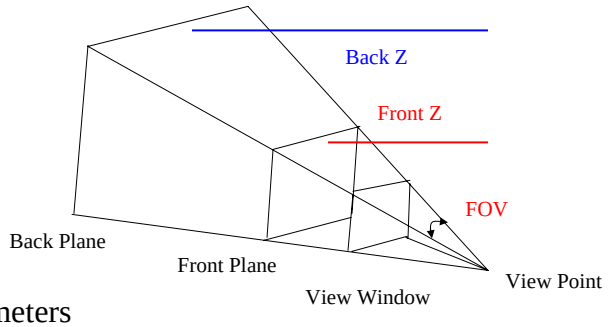
$$[x', y', z', 1] = [x, y, z, 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{d} \\ 0 & 0 & 0 & 0 \end{bmatrix} \Rightarrow \begin{array}{l} x' = \frac{x}{w} \\ y' = \frac{y}{w} \\ z' = \frac{z}{w} \end{array}$$

# Viewing Frustum





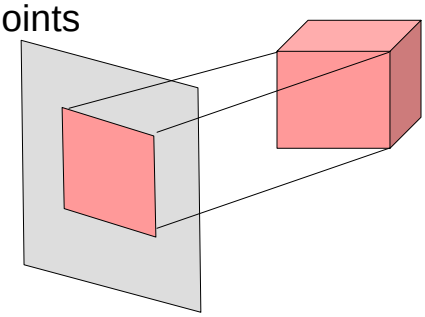

# Viewing Frustum



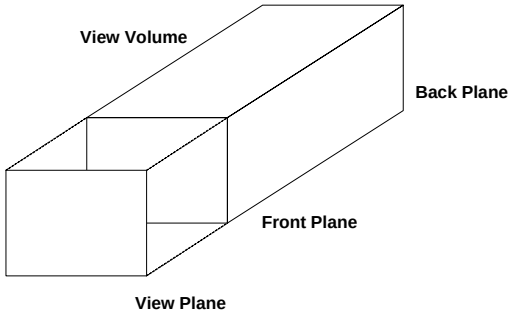

Parameters
-Field of view
-Aspect ratio (width, height)
-Front plane
-Back plane

# Parallel Projection

- View point is at infinity
  - Direction of projections is the same for all points







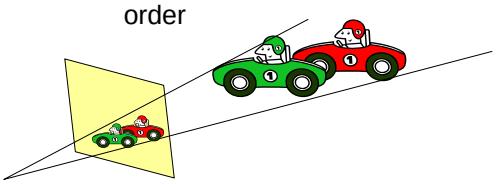
# Perspective Vs. Parallel

- Perspective
  - Size varies with distance
  - Parallel lines not preserved
  - Distances and angles are not preserved
- Parallel
  - Parallel lines remain
  - Angles are not preserved
  - Less realistic
  - Distance are preserved



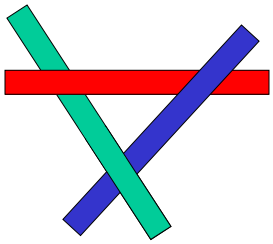
# Hidden Surface Removal

- 3D objects are bound to obscure other 3D objects
  - Closest object should always be visible
  - Often only parts of objects are visible
- Solution:
  - Painter Algorithm - sort objects and display them in reverse order



# Z - buffer

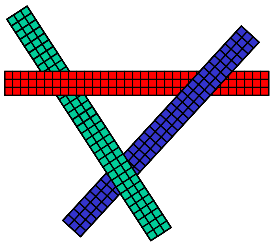
- Issues:
  - Time consuming – many objects to sort
  - May not be feasible



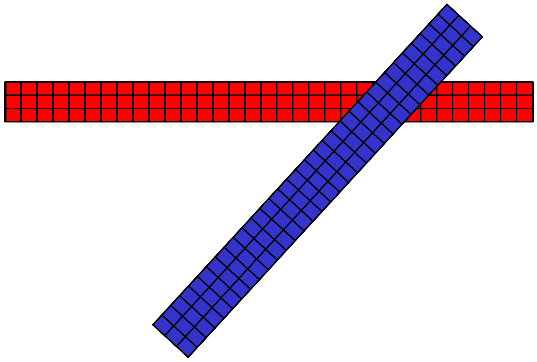


# Z - buffer

- Solution
  - “Disassembly” the object into very small fragments
  - Treat each fragment as an atomic unit
  - Each Fragment “occupies” one pixel



# Z - buffer

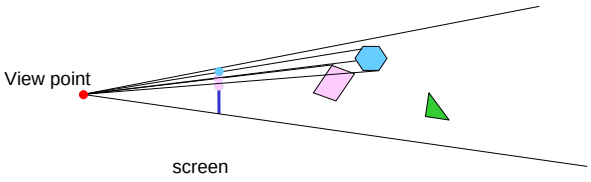




# Z - buffer

- Project objects on the screen
- Colour pixels with respect to the objects

- Solution:
  - If a pixel of an object is closer



# Z buffer

- A.K.A. as a depth buffer
  - A matrix which stores the “distance/depth” of the object (z-value)
  - Each fragment of the object is associated with a depth value
- Each object fragment is compared against the depth buffer
  - If value is smaller → fragment is visible

| Advantages | Disadvantages |
| --- | --- |
| • Simple<br>• Objects can be drawn at any order<br>• Brute force solution | • Depends on depth value accuracy<br>• Issue with transparent objects – may require sorting |