The Geometry Stage

3D Objects, Camera, Projections, Clipping and Viewports

Cameras and orthographic/perspective projections

Cameras

Cameras

- We view our 3D scene through the eyes of the camera, it determines how we see a scene and how much we can see at any given moment.
- To navigate through a 3D scene, we need only move the camera through the scene.
- Cameras are sometimes referred to as “eyes” in graphics literature.

- Each Camera has:
  - A position, a view vector and an up vector. These define the camera’s coordinate system.
  - A View Volume – only objects in this view volume are visible.
  - A Projection – this defines the shape of our view volume.

Cameras

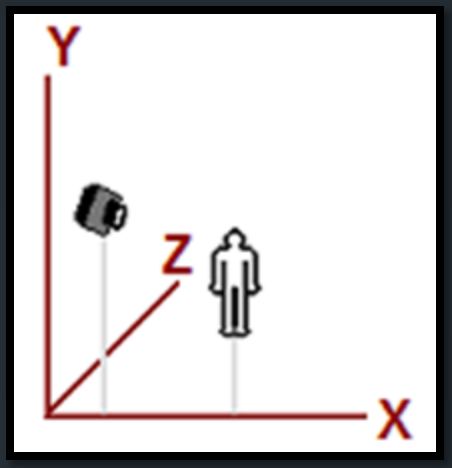

RollePollie <SCAR-L CARBINE> BFBC2Sharp
<RPG-7 AT> EA Brompb
<XM8 COMPACT> Muttley77
RU
117
US
77
Muttley77
BennyTheBuff2
RollePollie
Nok
026 G×03 / 060
First Person Camera

Cameras

Cameras


670+270
0 +0
19 +5
3/30
COMPANY OF HEROES
Engineers (3/3)
Ron_Mexico
Real Time Strategy Camera

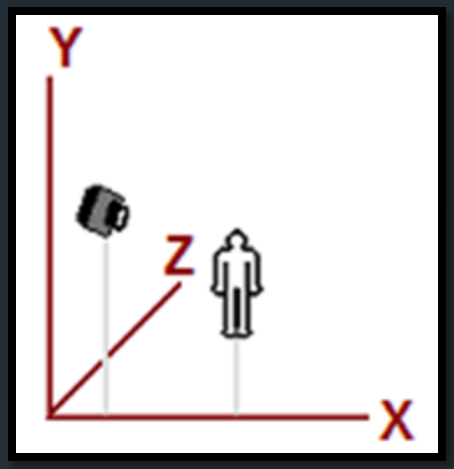
Cameras: View Space



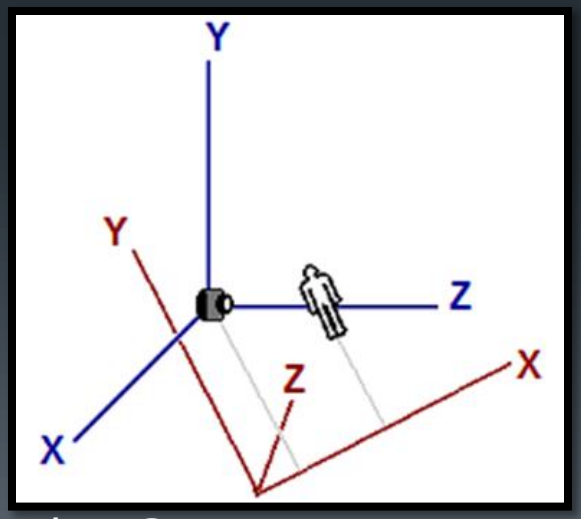
World Space

- Our 3D scene has its own coordinate system called world space and all objects are positioned relative to it.
- Our camera is also positioned and oriented in world space.
- To draw our scene it is necessary to do so from the camera's perspective. We need to transform the scene so that the camera is at the center.
- We create a new coordinate system called view space which is centered on the camera.

Cameras: View Space



World Space

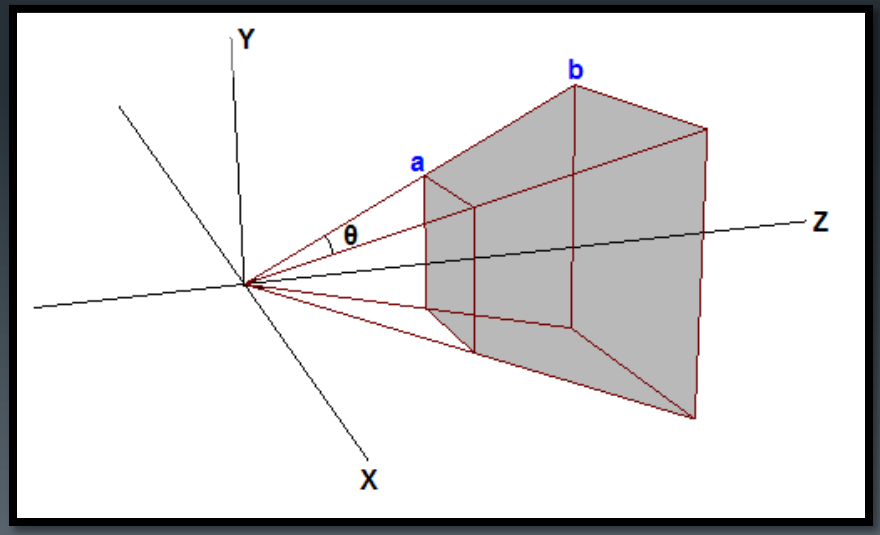


View Space

- The camera's position is the origin
- The camera's up vector is the Y axis
- The camera's view vector is in the direction of the:
  - +Z axis (left handed system)
  - -Z axis (right handed system)
- This new coordinate system is stored in a view matrix (the graphics API can usually calculate this for you).
- We transform world space into view space by multiplying all world space geometry with the view matrix.

Cameras: View Volume

- Each camera has a view volume.
- This view volume is defined by a near view plane and a far view plane and its shape is determined by the type of projection used.
- Only geometry within the camera's view volume will be visible.



Example of a View Volume
a – near view plane
b – far view plane
gray area – view volume

Cameras: Projections (The CVV)

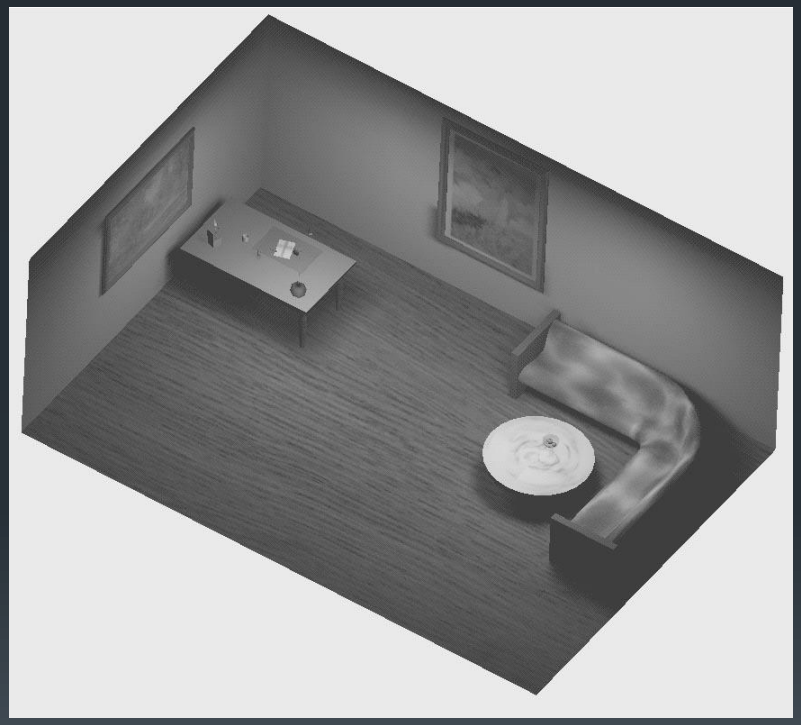
- The camera's arbitrary view volume needs to be transformed into a unit cube centered on the view space origin. This is done to improve the efficiency of the clipping and screen mapping stages.

- This standard cubic view volume is called the canonical view volume (CVV) and is used in the clipping and screen mapping stages.

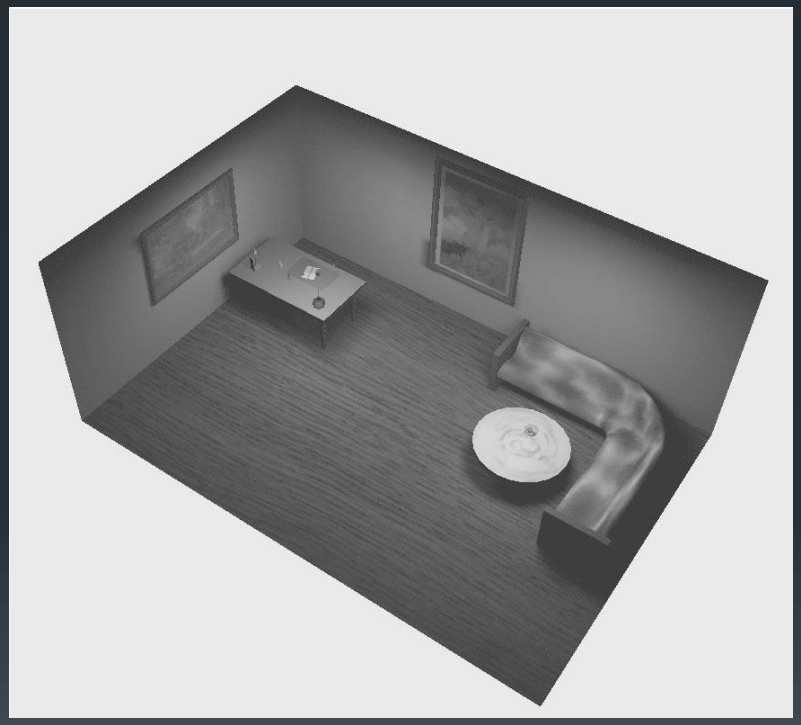
- The transformation of the camera's view volume to the canonical view volume is called a projection. Applying this projection to view space moves us into homogenous clip space.

Cameras: Projections

There are two common projections in use today:

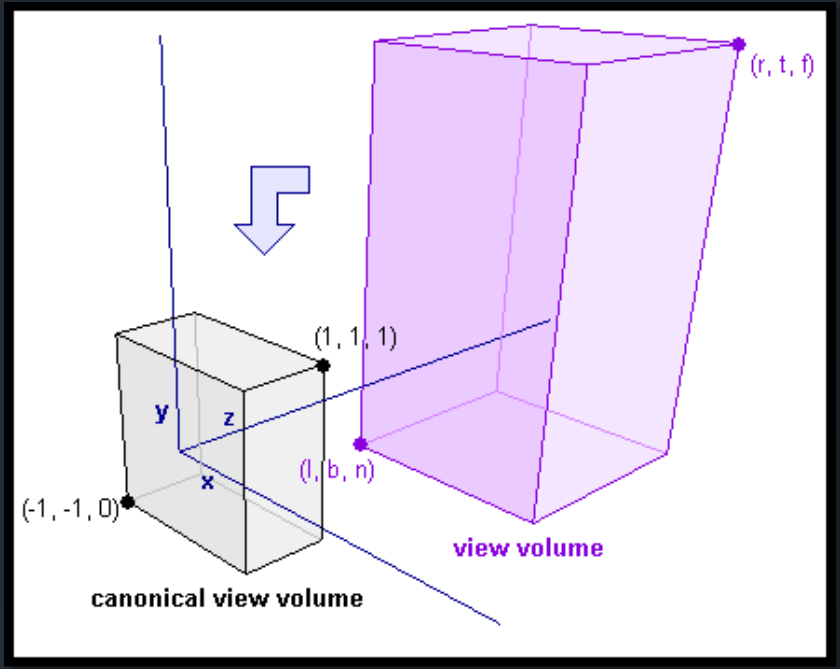


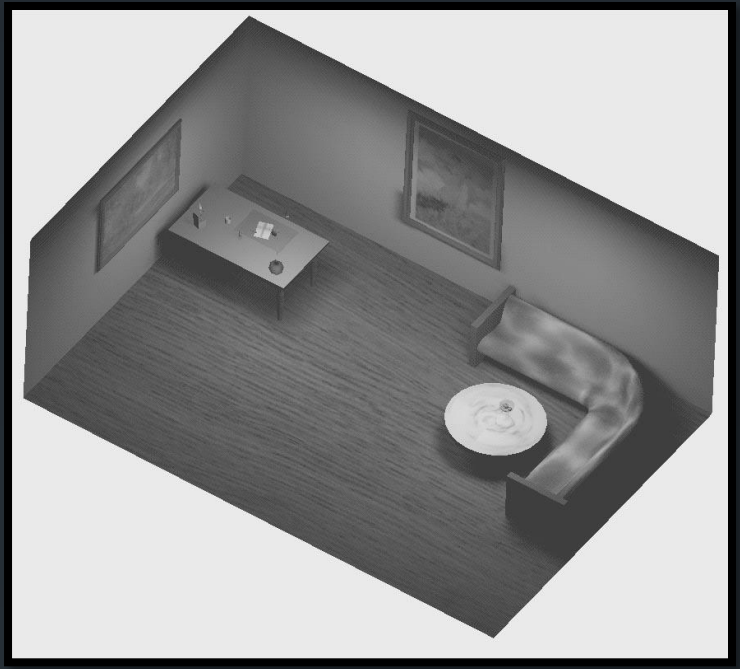
Orthographic Projection



Perspective Projection

Cameras: Orthographic Projection

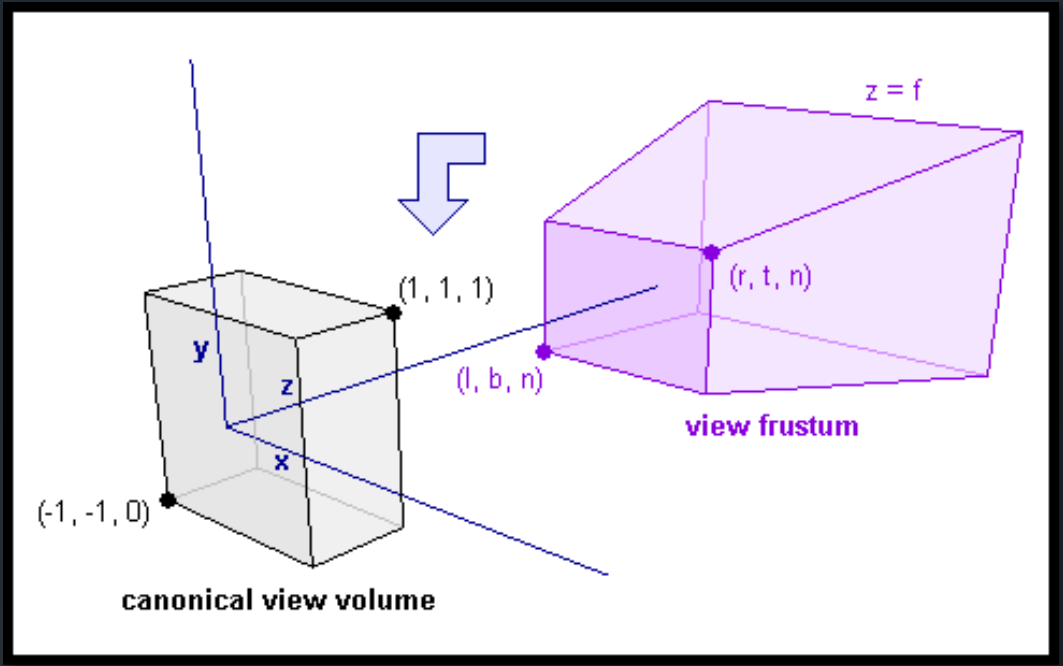


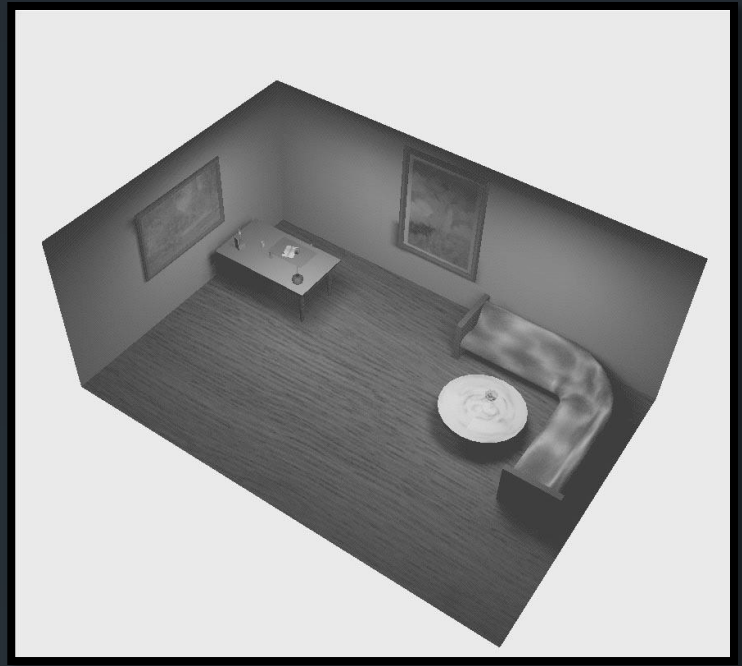


- In an orthographic projection, the view volume is a cube, and so the conversion from the orthographic view volume to the canonical one is a simple matter of scaling and translation.
- This form of projection keeps all lines parallel and doesn't scale objects according to depth and so is ideal for CAD applications.

Images from "Deriving Projection Matrices" By Joe Farrel - http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__3/

Cameras: Perspective Projection





- The perspective view volume is a cutoff pyramid (a frustum) with the apex at the camera position. It is defined by a near plane, a far plane and a field of view angle (FOV) (can be a vertical or horizontal FOV).
- This form of projection is more realistic and takes into account that objects further away from the camera are smaller than objects closer to the camera. Parallel lines are no longer guaranteed to be parallel after this projection has been applied. (why does this happen?)

Images from "Deriving Projection Matrices" By Joe Farrel - http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__3/

Cameras: Projections

- Projections are stored in Projection Matrices, most graphics API have helper functions to generate these matrices for you
  - DirectX - D3DXMatrixPerspectiveFovRH & D3DXMatrixOrthoRH
  - OpenGL - gluPerspective & gluOrtho

- Projections are applied by multiplying view space coordinates by with an appropriate projection matrix. This is done in the vertex shader program. This converts view space coordinates into homogenous clip space coordinates.

Cameras: Implementation

- Camera movement must be timer based (why?) and so camera velocities must be in “units per second”.
- When doing camera rotations and using Euler angles be careful to avoid “Gimbal Lock”.
- Gimbal Lock results in a loss of a degree of freedom due to an extreme rotation in one of the axes.
- Cameras are recommended to be entirely decoupled from rendering code and only return the view and projection matrices (or the concatenation of both matrices).

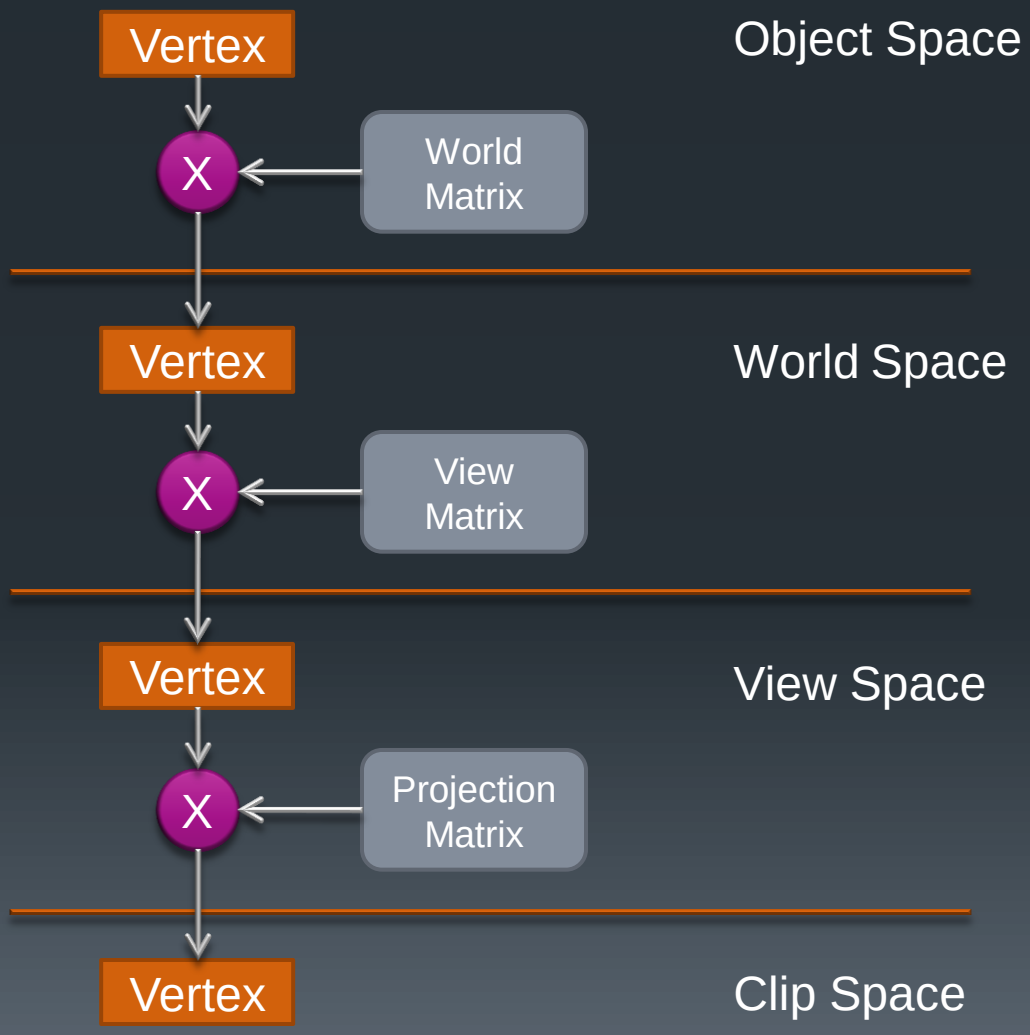
Moving Spaces: From Object Space to Clip Space

The Vertex Shader Stage

The Vertex Shader Stage

- Responsible for the transformation of vertices as well as vertex data such as texturing coordinates, colors, etc.
- The vertex shader cannot create or destroy vertices but the vertex shader can create or discard per-vertex data.
- The vertex shader outputs vertices in clip space. Can take input vertices in any space (most commonly object space).
- The vertex shader is usually responsible for transforming object space vertices into clip space vertices (how?)

Transforming Vertices



In moving from object space to clip space, the following occurs in the vertex shader:

- Object space vertices are multiplied by the world matrix to move them into world space.
- World space vertices are multiplied by the view matrix to move them into view space.
- View space vertices are finally moved into clip space by multiplying them with the projection matrix

Primitive Clipping and Screen Mapping

Clipping and Screen Mapping

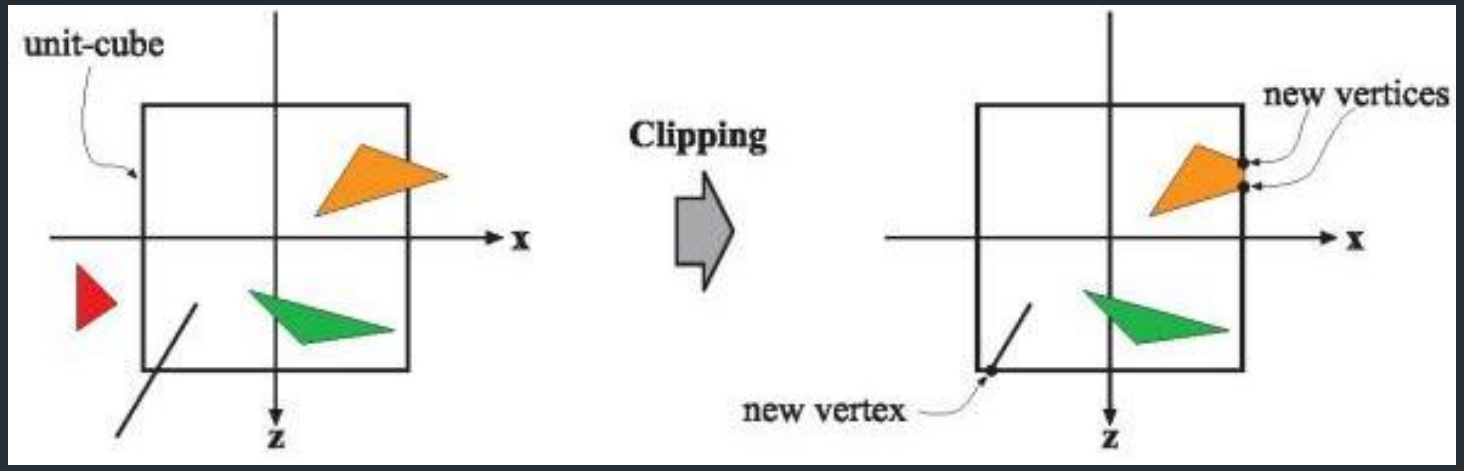
Clipping and Screen Mapping

The final stages in the geometry stage are concerned with preparing the geometric shapes for rasterization.

This is done in two steps:

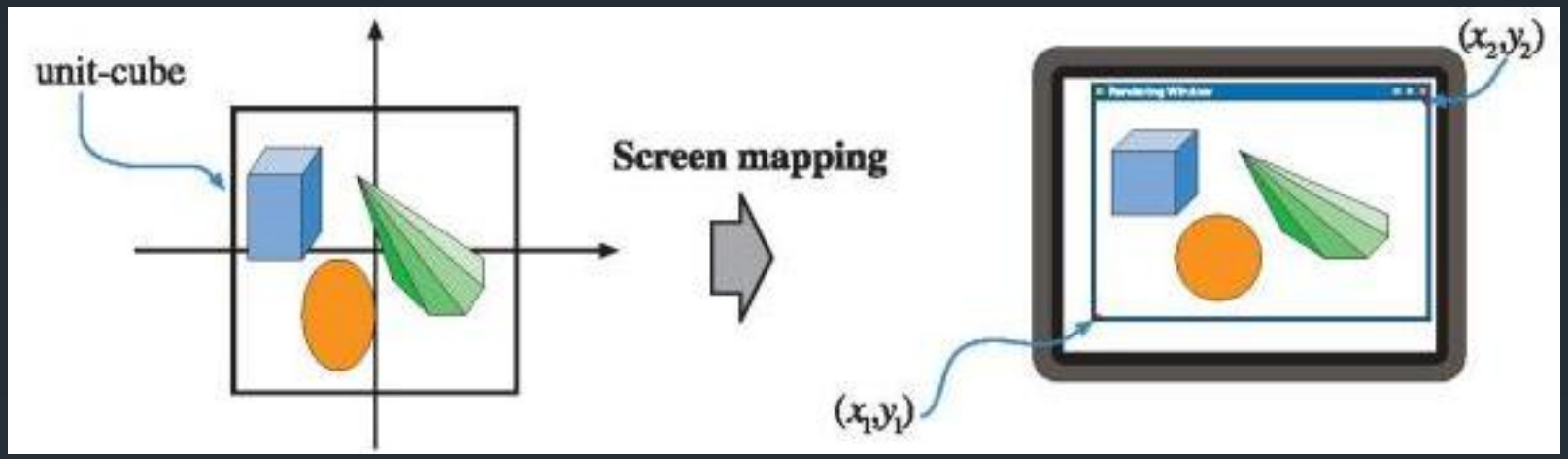
- Clipping: this process discards any geometric shapes that aren't visible to the viewer as well as modifying any shapes that are semi-visible
- Screen Mapping: simply transforms clip space coordinates to screen space coordinates.

Clipping



- Only primitives which are visible need to be passed to the rasterizer stage.
- After projection all primitives are in homogenous clip space.
- They are checked against the 6 clipping planes of canonical view volume (unit cube) before they are sent to the next stage.
  - Any primitives that lie outside the view volume are discarded
  - Any primitives that lie fully inside the view volume are passed to the next stage
  - Any primitives that lie partially in the view volume are clipped.
  - This stage is not user configurable and is processed by fixed function hardware

Screen Mapping



- After clipping has occurred, primitives are sent to the screen mapping stage, here the unit cube's X and Y ranges are mapped to the pixel dimensions (width and height) of the active viewport.
- Primitives' X and Y coordinates are transformed to be within the viewport. This makes it easy to determine which pixels in the viewport are affected by each primitive.
- Primitives' depth values (Z coordinates) remain unchanged.

Overview of the Geometry Stage

Conclusion

Conclusion

Before any 3D geometry can be rendered to the screen it needs to pass through these steps:

- Model space is converted into world space by the world transform matrix (a concatenation of several affine transformations), this has the effect of positioning the object in the world.
- World space is then converted to camera space by the view matrix. This orients the world according to the camera.
- Camera space is then projected into a canonical view volume (homogenous clip space), using the camera's projection matrix (orthographic/perspective)
- All projected geometry is then clipped against the canonical view volume and only primitives within the view volume are sent for screen mapping.
- Screen mapping, Triangle Setup and Triangle Traversal converts the 3D primitives' coordinates into screen space pixel coordinates, adds a depth value and sends them through to the rasterization stage.