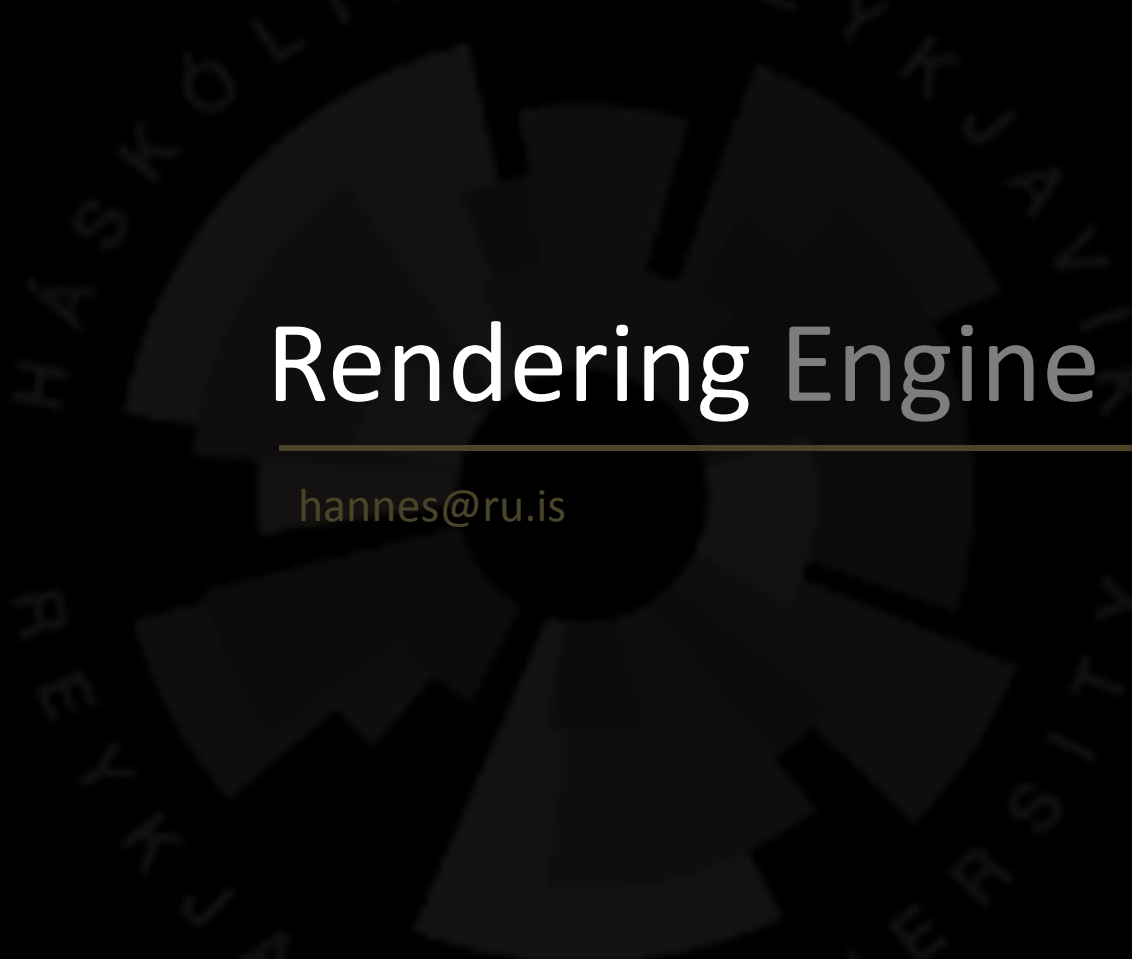


Rendering Engine Part B

hannes@ru.is

Virtual Camera

- An ideal focal point with an imaging rectangle floating in front of it
- Imaging rectangle contains virtual light sensors corresponding to screen pixels
- Rendering is the process of determining what color and intensity of light would be recorded by each such sensor

View Space

- The focal point is the origin of view space (camera space)
- Camera usually looks down positive or negative z-axis
- When rendering a triangle, its vertices are transformed first from model space to world space, and then from world space to view space (requires a world-to-view matrix)

View Space

- World-to-view matrix, or view matrix, is simply the inverse of the view-to-world matrix
- The view matrix is often concatinated with the model-to-world matrix for efficiency, this combined matrix is then called the model-view matrix

Projections

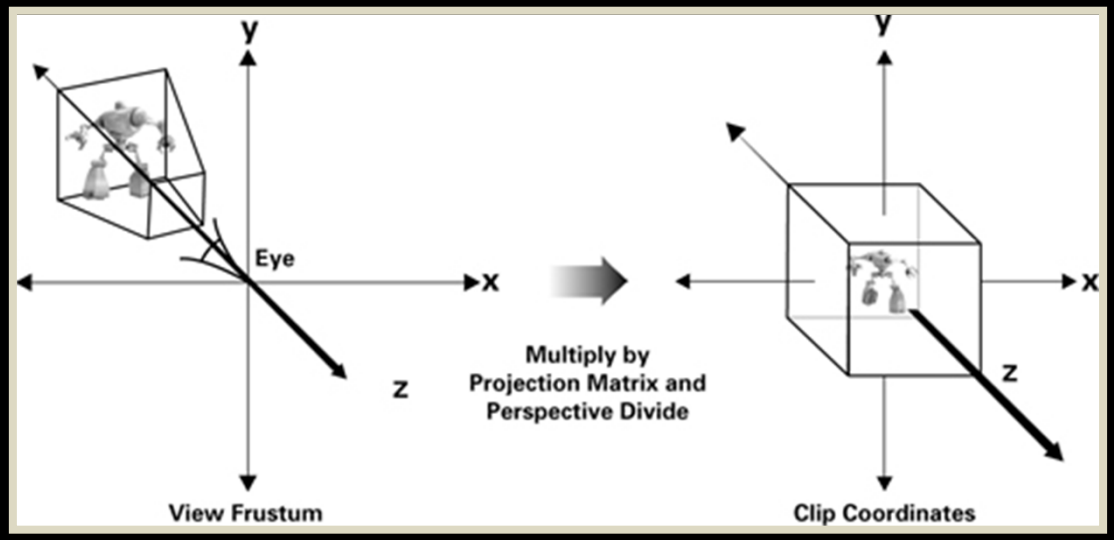
- We require a projection from a 3D scene onto a 2D image plane
- Typical projections include
  - Perspective projection (provides perspective foreshortening)
  - Orthographic projection

View Volume and Frustum

- View Volume: Region of space seen by camera
- The region is bounded by six planes: Four sides, a near plane and a far plane
- Perspective projection shapes the view volume into a frustum
- Orthographic projection shapes it into rectangular prism

Homogeneous Clip Space

- Purpose of clip space is to convert camera-space view volume into a canonical view, independent of projection, resolution and aspect ratio – easy to clip (axis aligned)

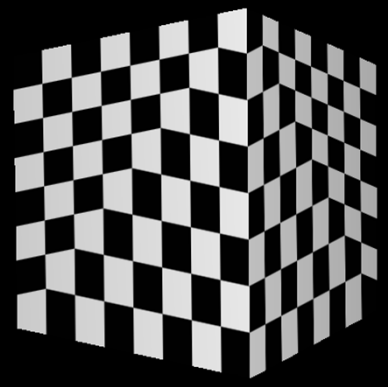


Perspective Projection

- Transforms vertices from view space into homogeneous clip space
- Perspective projection results in each vertex‘s x- and y-coordinates being divided by its z-coordinate
- This produces the effect of perspective foreshortening

Perspective-Correct Interpolation

- Vertex attribute interpolation is performed in screen space
- We iterate over each pixel of the screen and attempt to determine the value of each attribute at the corresponding location on the surface of the triangle
- We must account for perspective foreshortening by performing perspective-correct attribute interpolation



Screen Space

- Screen space is a 2D coordinate system in pixels
- X-axis to right, with origin a top-left and y pointing down (direction of CRT gun movement)
- Aspect Ratio: Width / Height
- Triangles in homogeneous clip space can be rendered here by dropping z-coordinate
- Also need to scale and shift from clip-space into screen space (screen mapping)

The Frame Buffer

- Final rendered image is stored in a bitmapped color buffer called frame buffer
- Frame buffer is periodically read by display hardware
- Double buffering used to avoid tearing, i.e. having hardware read buffer while new image has not been fully stored there
  - One buffer displayed while the other is updated by rendering engine, then swap buffers

Render Target

- Frame buffer is only one kind of buffer you can render into
- Any buffer that rendering engine draws into is called a render target
- There are many kinds of off-screen render targets, e.g. depth buffer and various intermediate rendering results for special effects

Rasterization and Fragments

- Creating an on-screen image of a triangle involves filling in the pixels it overlaps. This is called rasterization
- Each region of a triangle that corresponds to a single screen pixel is called a fragment
- A fragment must pass a number of tests before being written into the frame buffer
- Passing the test results in its color written to the buffer or blended with the existing color

Antialiasing

- Rasterized triangles can have jagged edges because they are represented by a discrete set of pixels
- Antialiasing is the process of blending the edge colors with surrounding colors
- Full-screen Antialiasing (FSAA) renders full image into a buffer twice as big as screen, then down-sampled
- Multisample Antialiasing (MSAA) breaks triangles into more fragments than pixels

Occlusion and Depth Buffer

- Painter‘s algorithm of rendering triangles from back-to-front order does not always work
- Instead of doing this per triangle, we can work with each fragment
- Depth buffer contains depth information for each pixel already in frame buffer
- Every fragment has a z-coordinate (interpolated from vertices) stored in depth buffer when its pixel is colored

Occlusion and Depth Buffer (cont)

- When another fragment is drawn to same pixel, it overwrites only if its Z value is closer than the Z value already stored in the depth buffer for that pixel

Z-Fighting

- Depth buffer typically uses the Z value from the clip-space, i.e. after perspective projection and normalization into a fixed range (e.g. 0-1)
- Precision of this value is not evenly distributed across the near-to-far range because of the division by view space z
- Distant triangles will „bunch up“ towards the far end, possibly overlapping in Depth Buffer

W-Buffer

- We can solve this by storing the view space z-coordinate in the depth buffer
- View-space z-coordinates vary linearly with distance from camera
- This is called w-buffering (this old z-coordinate appears as the w component of the homogeneous clip-space coordinate)