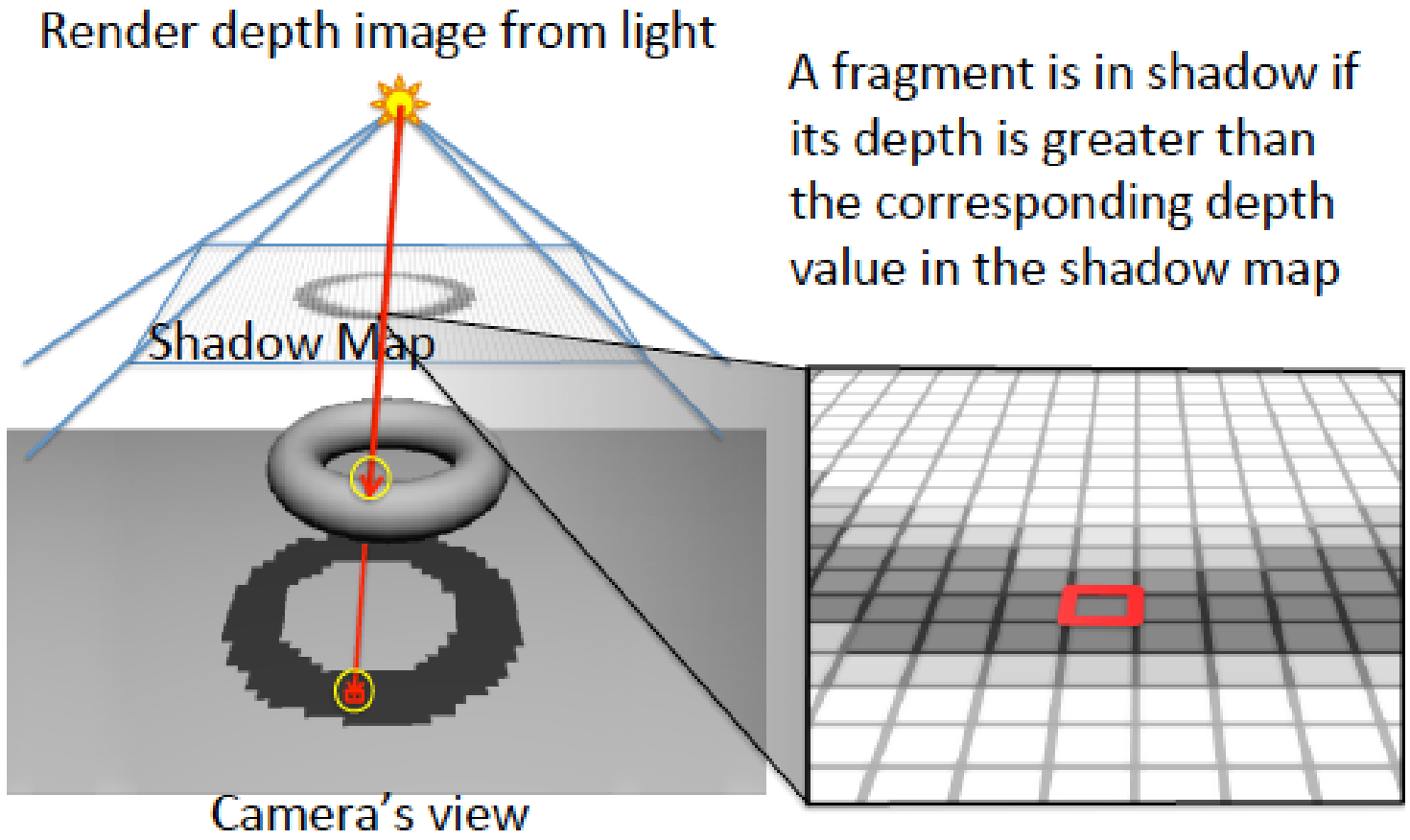
DigiPen
INSTITUTE OF TECHNOLOGY

Cascaded Shadow Maps

Cascaded Shadow Maps

- Improvement for the shadow mapping algorithm
- Long range views:
  - Open world landscapes
  - Large scale terrain

Shadow Map



Generating Shadow Map

- Place a “virtual camera” at the light position and orient it with the light orientation
- Render the scene’s depth to a texture map
- Since we need only the depth value, minimize the amount of work in the shader

Using Shadow Map

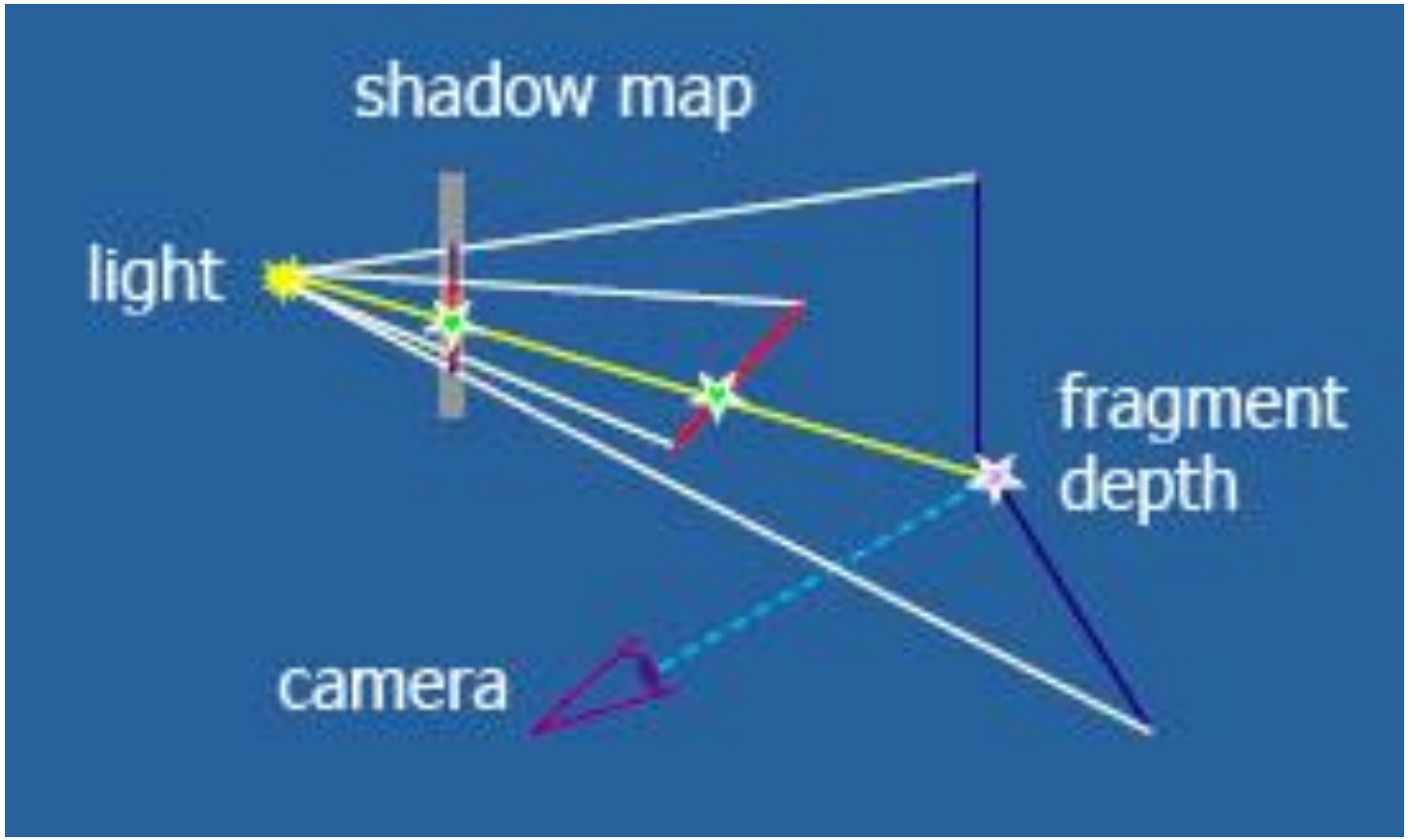
- Set the texture generated in the 1st pass as one of the input texture for the shadow receiver object(s)

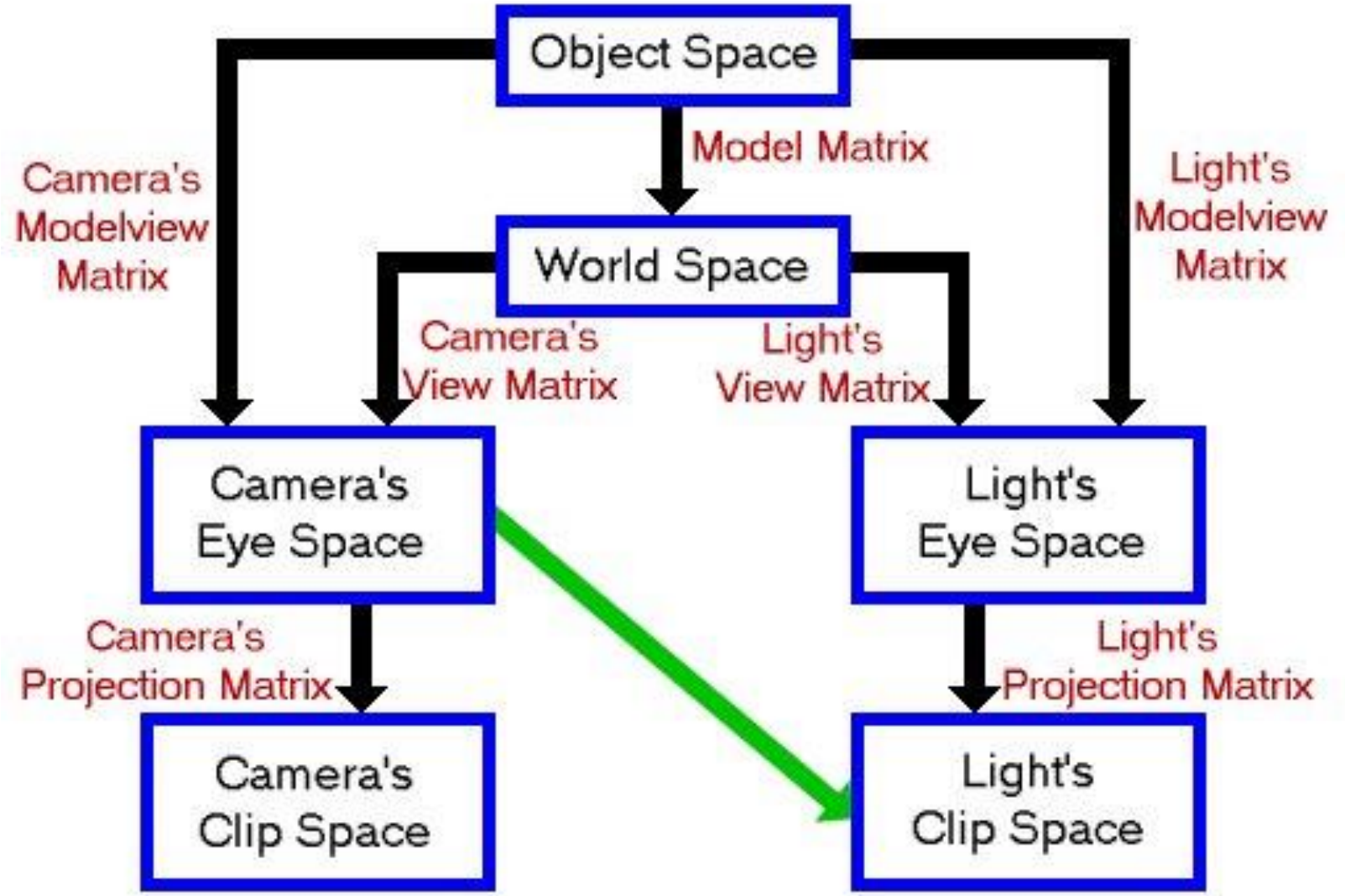
- While rendering each pixel of the shadow receiver object(s):
  - Transform the pixel position into the **light space**
  - Convert the position into texture coordinate
  - Access the depth value from the depth texture map
  - If the pixel depth value is greater than the depth value from the depth texture, it means there is an object that is closer to than the current pixel, hence the pixel is in shadow

Using Shadow Map

- Once it is determined that a pixel is in shadow, adjust the diffuse and specular component of the pixel color accordingly
- Typically, multiply the light components by some value between 0 to 1 to make it darker
- A better approach is to adjust the diffuse and specular component in the lighting equation based on the shadow status

Shadow Mapping in Action





Converting Pixel to Light Space

- Matrices needed to transform the pixel position from **view space**:
  - The camera-to-world transformation matrix V_c^{-1}
    - Inverse of the world-to-camera transform
  - The world-to-light transformation matrix L_V
    - A viewing matrix with the light position and orientation as the camera position and orientation
  - The light projection matrix L_P
    - The projection matrix used to project objects in light space to the texture map
  - The texture coordinate mapping matrix L_S.
    - Matrix used to map values from [-1, 1] to [0, 1]

Used when generating the shadow map

Converting Pixel to Light Space

- Once L_V, LS, LP and W are calculated:

$$TC = L_S * L_P * L_V * V_c^{-1}$$

- Where:
  - V_c^{-1} is the inverse of the world to camera matrix.
  - L_V is the light view matrix.
  - L_P is the light perspective matrix.
  - L_S is the matrix that transforms from range [-1,1] to [0,1].
  - TC is the homogenous texture coordinate

Types of light

- Each type of light will have differences in how the shadow map is generated:

- Spotlight
  - Perspective Matrix
    - FOV is based on the outer angle of the cone
  - Map type
    - 2D square texture
- Point light
  - Perspective Matrix
    - FOV is 90 degrees per face
  - Map type
    - Cube map texture
- Directional light
  - Orthogonal Matrix
  - Map type
    - 2D square texture

Aliasing

- Each pixels stores a depth value with the corresponding precision issues
  - Z-fight on depth comparisons → Shadow acne
- Solutions:
  - Compare using a bias value

Aliasing

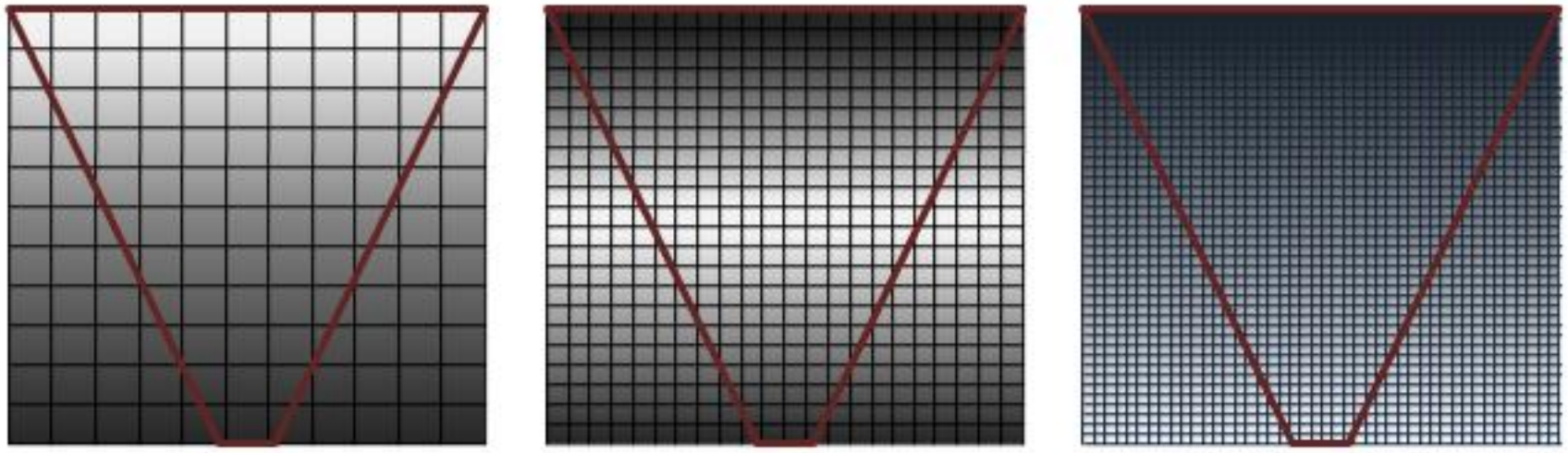
- Shadow maps have a **discrete** amount of information
  - Jaggies on the shadow edges
- Solutions:
  - Increase resolution
    - Increase sampling rate
  - Percentage Closer Filtering
    - Multi sample when reading shadow map

Aliasing

- Increase bias
- Increase resolution
  - From 512x512 to 2048x2048
- PCF with 9x9 neighborhood

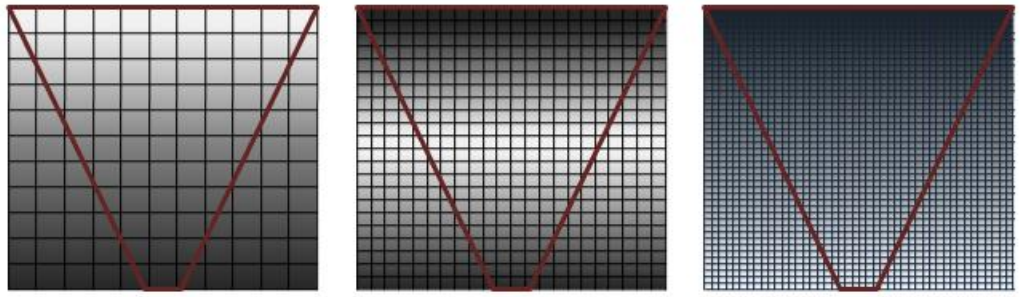
Aliasing

- Increasing the shadow map resolution will not always be possible:
  - Too many big shadow maps impact performance
  - Memory limitations
  - …



Cascaded Shadow Maps (CSM)

- Open world landscapes



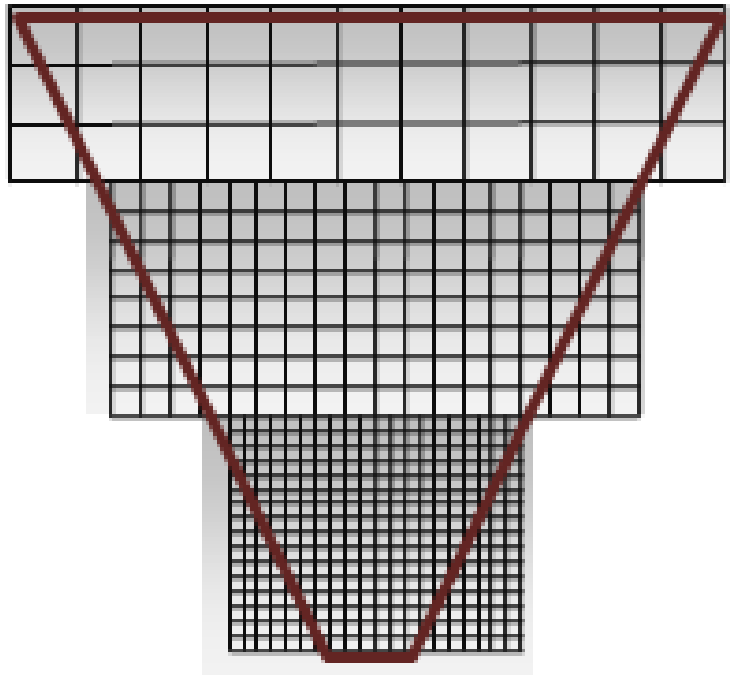
Undersampling close to near plane

Oversampling on far plane and undersampling on near

Oversampling on far plane

Cascaded Shadow Maps (CSM)

- Different areas of the camera frustum require shadow maps with different resolutions
  - Objects nearest the eye require a higher resolution than do more distant objects
- Partition the camera frustum into multiple frusta
  - Each subfrustum will have its shadow map
- Mainly useful for lights that have a long area of effect
  - Directional light



Cascaded Shadow Maps (CSM)

1. Partition the frustum into subfrusta
2. Compute a projection for each subfrustum
3. Render a shadow map for each subfrustum
4. Render the scene
   - The pixel shader does the following:
     - Determines the proper shadow map
     - Transforms the texture coordinates if necessary
     - Samples the cascade
     - Lights the pixel

Partitioning the Frustum

- Uniform/Linear partition:
  - Each subfrustum will have the same
  - Usually does not provide enough extra sampling near the eye

$$z_i = n + (\frac{i}{N})(f - n)$$

Where:

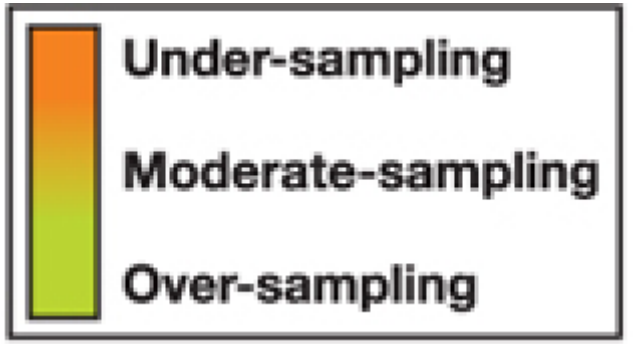
N: Number of cascade subdivisions

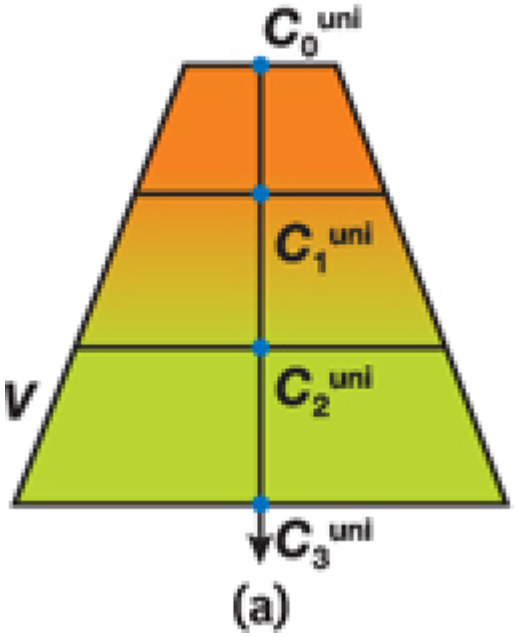
n: near distance

f: far distance

i: index of the split (from 0 to N)

z_i: depth for the ith split





Partitioning the Frustum

- Logarithmic partition:
  - Wider splits the further from the eye
  - The sampling rate is not desirably spread throughout the frustum

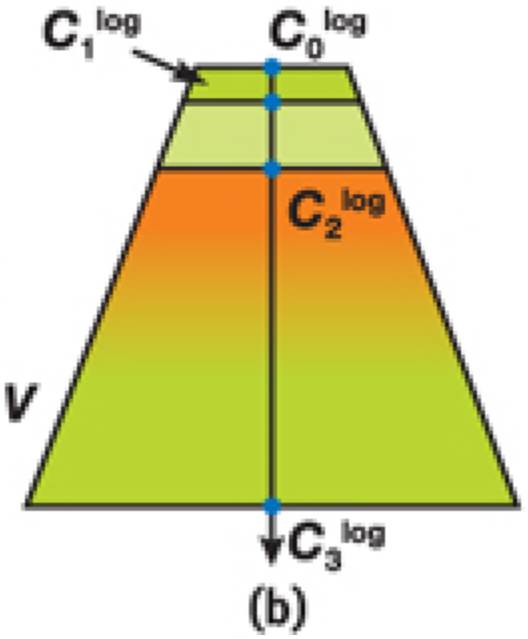
$$z_i = n\left(\frac{f}{n}\right)^{\frac{i}{N}}$$

Where:
N: Number of cascade subdivisions
n: near distance
f: far distance
i: index of the split (from 0 to N)
z_i: depth for the ith split

- Logarithmic partition:
  - Wider splits the further from the eye
  - The sampling rate is not desirably spread throughout the frustum

$$z_i = n\left(\frac{f}{n}\right)^{\frac{i}{N}}$$

Where:
N: Number of cascade subdivisions
n: near distance
f: far distance
i: index of the split (from 0 to N)
z_i: depth for the ith split

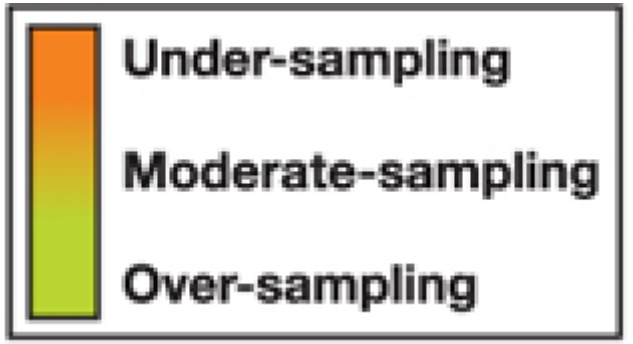


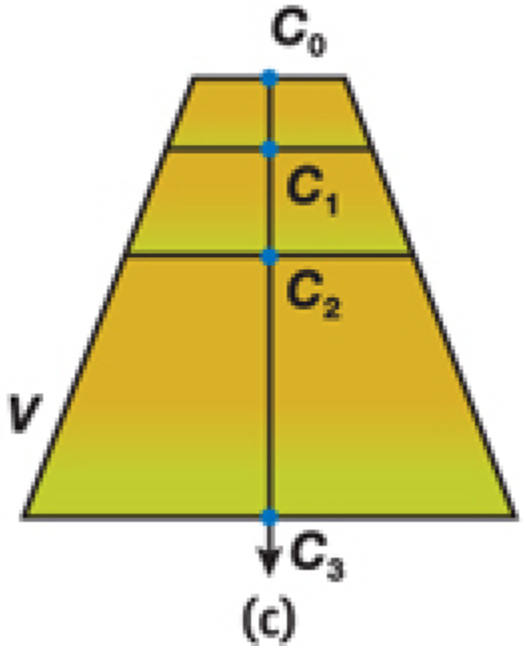
Partitioning the Frustum

- Mixed partition:
  - Interpolation between the uniform and logarithmic approach
  - Adjustable through a λ parameter

$$z_i = \lambda\left(n\left(\frac{f}{n}\right)^{\frac{i}{N}}\right) + (1-\lambda)\left(n + \left(\frac{i}{N}\right)(f-n)\right)$$

Where:
N: Number of cascade subdivisions
n: near distance
f: far distance
i: index of the split (from 0 to N)
z_i: depth for the ith split





Projection for each Frusta

- Directional lights use a orthogonal projection matrix and each subfrustum will need to compute its corresponding one
- Orthogonal Projection from CS250:

$$\begin{bmatrix} \frac{1}{S_w} & 0 & 0 & 0 \\ 0 & \frac{1}{S_h} & 0 & 0 \\ 0 & 0 & 0 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where:
d: focal length
S_w: half width of the projection plane
S_h: half height of the projection plane

Orthogonal Projection

- $Z - axis$ needs to be mapped to the $[-1,1]$ range for input values in range $[near, far]$

$$\begin{bmatrix} \frac{1}{S_w} & 0 & 0 & 0 \\ 0 & \frac{1}{S_h} & 0 & 0 \\ 0 & 0 & A & B \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{cases} T(0,0,-n,1) = (0,0,-1,1) \\ T(0,0,-f,1) = (0,0,1,1) \end{cases}$$

Orthogonal Projection

- $Z - axis$ needs to be mapped to the $[-1,1]$ range for input values in range $[near, far]$

$$\begin{cases} \begin{bmatrix} \frac{d}{S_w} & 0 & 0 & 0 \\ 0 & \frac{d}{S_h} & 0 & 0 \\ 0 & 0 & A & B \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -n \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -An + B \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \end{bmatrix} \\ \begin{bmatrix} \frac{d}{S_w} & 0 & 0 & 0 \\ 0 & \frac{d}{S_h} & 0 & 0 \\ 0 & 0 & A & B \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -f \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -Af + B \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \end{cases} \Rightarrow \begin{cases} -An + B = -1 \\ -Af + B = 1 \end{cases}$$

Orthogonal Projection

- $Z - axis$ needs to be mapped to the $[-1,1]$ range for input values in range $[near, far]$

$$\begin{cases} -An + B = -1 \\ -Af + B = 1 \end{cases}$$

$$-Af + (An - 1) = 1 \Longrightarrow (n - f)A = 2 \Longrightarrow A = \frac{2}{n - f}$$

$$B = \left(\frac{2}{n - f}\right)n - 1 = \frac{2n}{n - f} - \frac{(n - f)}{n - f} = \frac{2n - n + f}{n - f} = \frac{n + f}{n - f}$$

Orthogonal Projection

- $Z - axis$ needs to be mapped to the $[-1,1]$ range for input values in range $[near, far]$

$$\begin{bmatrix} \frac{1}{S_w} & 0 & 0 & 0 \\ 0 & \frac{1}{S_h} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & \frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthogonal Projection

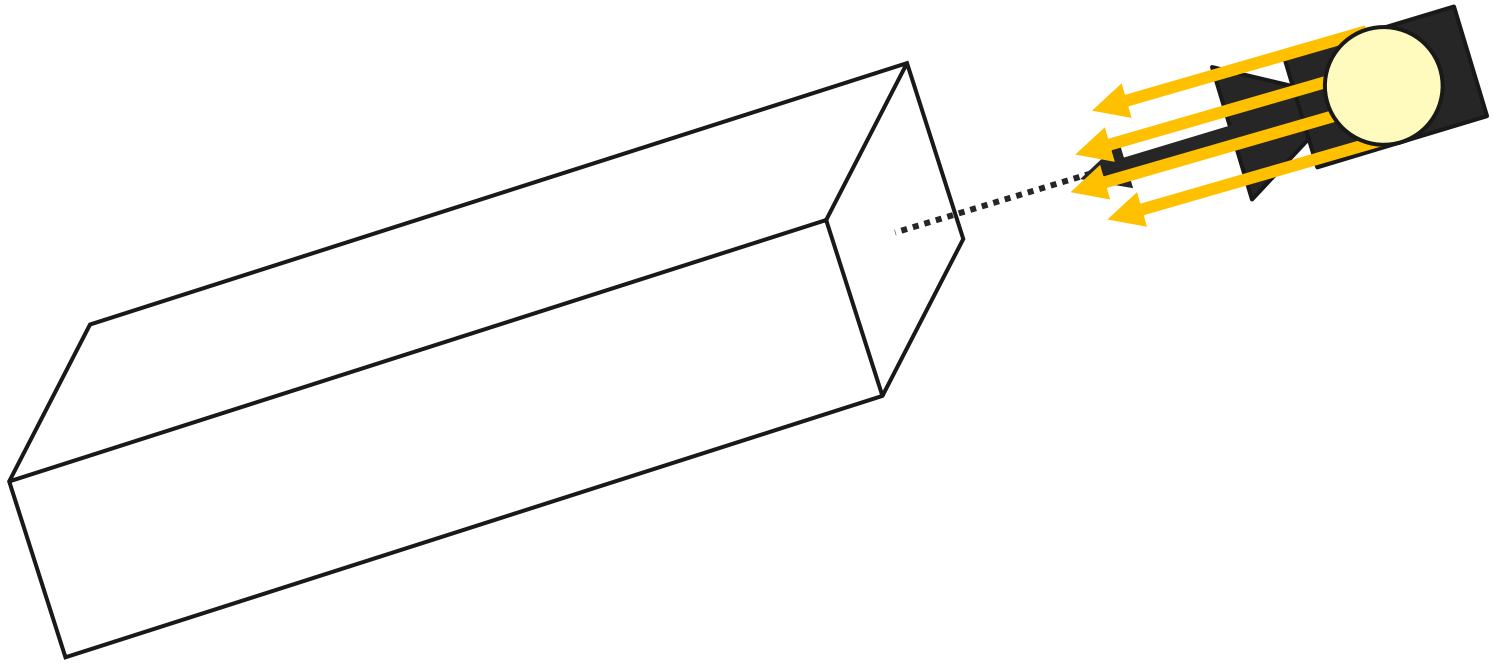
$$\begin{bmatrix} \frac{1}{S_w} & 0 & 0 & 0 \\ 0 & \frac{1}{S_h} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & \frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation

```
template<typename T>
GLM_FUNC_QUALIFIER mat<4, 4, T, defaultp> ortho(T left,
                                                T right,
                                                T bottom,
                                                T top,
                                                T zNear,
                                                T zFar)
{
    mat<4, 4, T, defaultp> Result(1);
    Result[0][0] = static_cast<T>(2) / (right - left);
    Result[1][1] = static_cast<T>(2) / (top - bottom);
    Result[2][2] = - static_cast<T>(2) / (zFar - zNear);
    Result[3][0] = - (right + left) / (right - left);
    Result[3][1] = - (top + bottom) / (top - bottom);
    Result[3][2] = - (zFar + zNear) / (zFar - zNear);
    return Result;
}
```

Orthogonal Projection

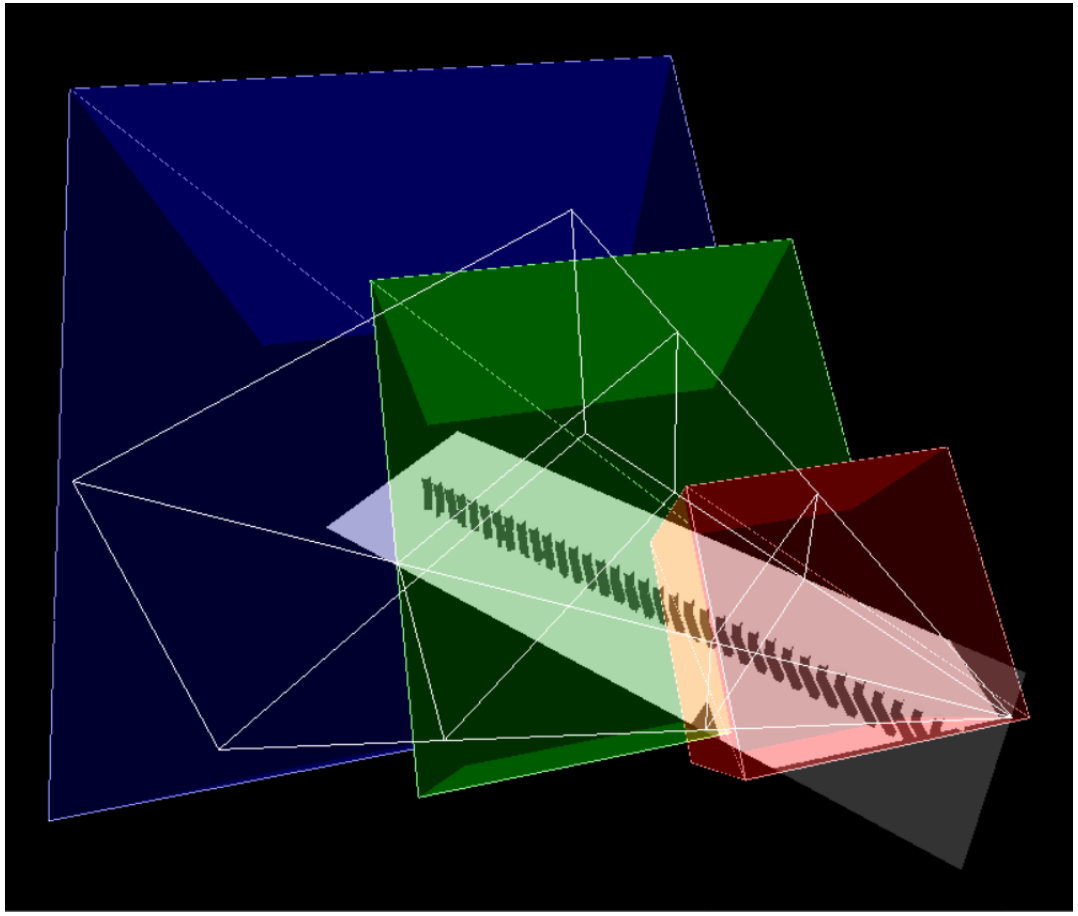
- Translation allows to moved the projected box (not a frustum)
  - Right, left, top, bottom don't have to be 0 centered
  - Allows to offset the rectangle away independently from camera (light) position



Projection for each Frusta

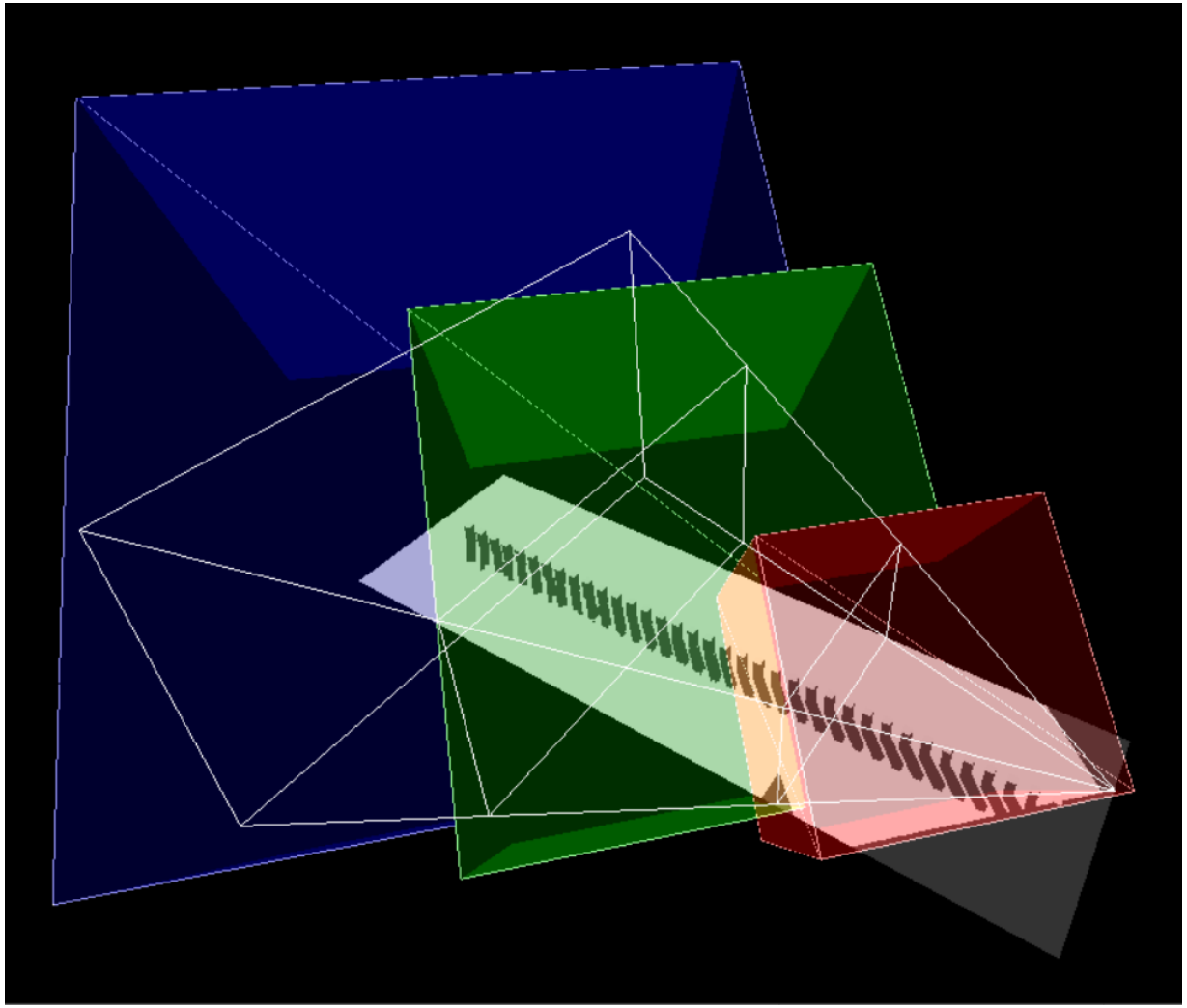
- Compute the points of the subfrustum in light view space
  - Transform the vertices in world using the light´s view matrix
- Compute the AABB in light view space
- Create an orthogonal projection matrix using the AABB data
  - X and Y axes need to have the same dimension
    - Tight for the maximum of those axes
    - Coarse for the other one
  - Z axis
    - Tight to the subfrustum

Projection for each Frusta

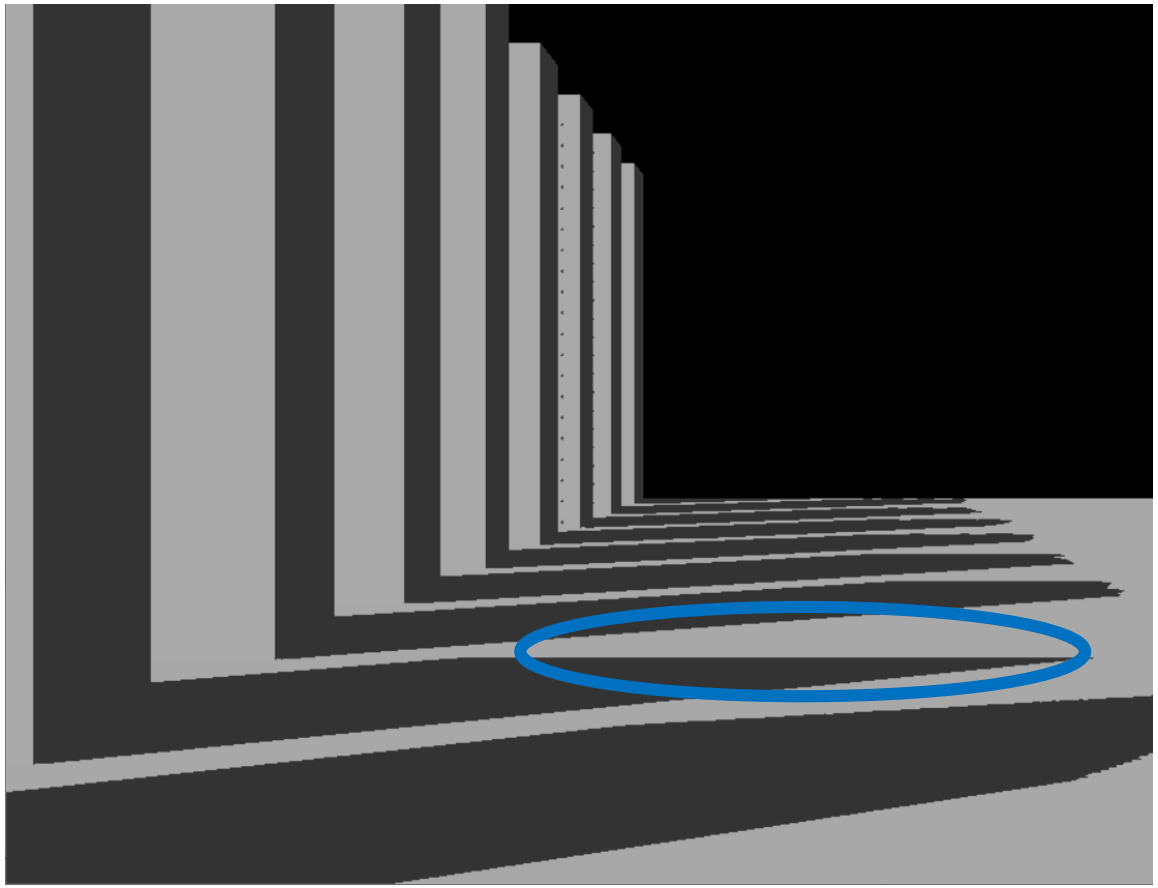


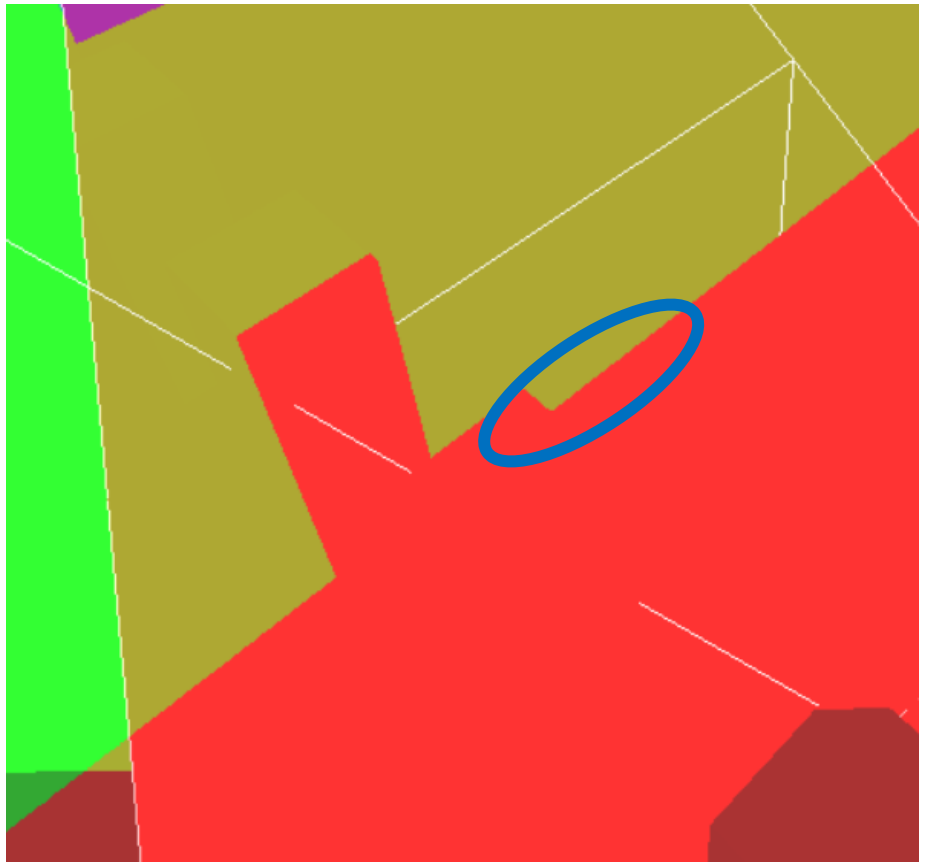
Occluder Distance

- Shadows inside each of the frustum boxes will only be casted by geometry that lays inside the boxes
  - Problem…
- The projection boxes do not start at infinity and directional lights do not have a position



Occluder Distance

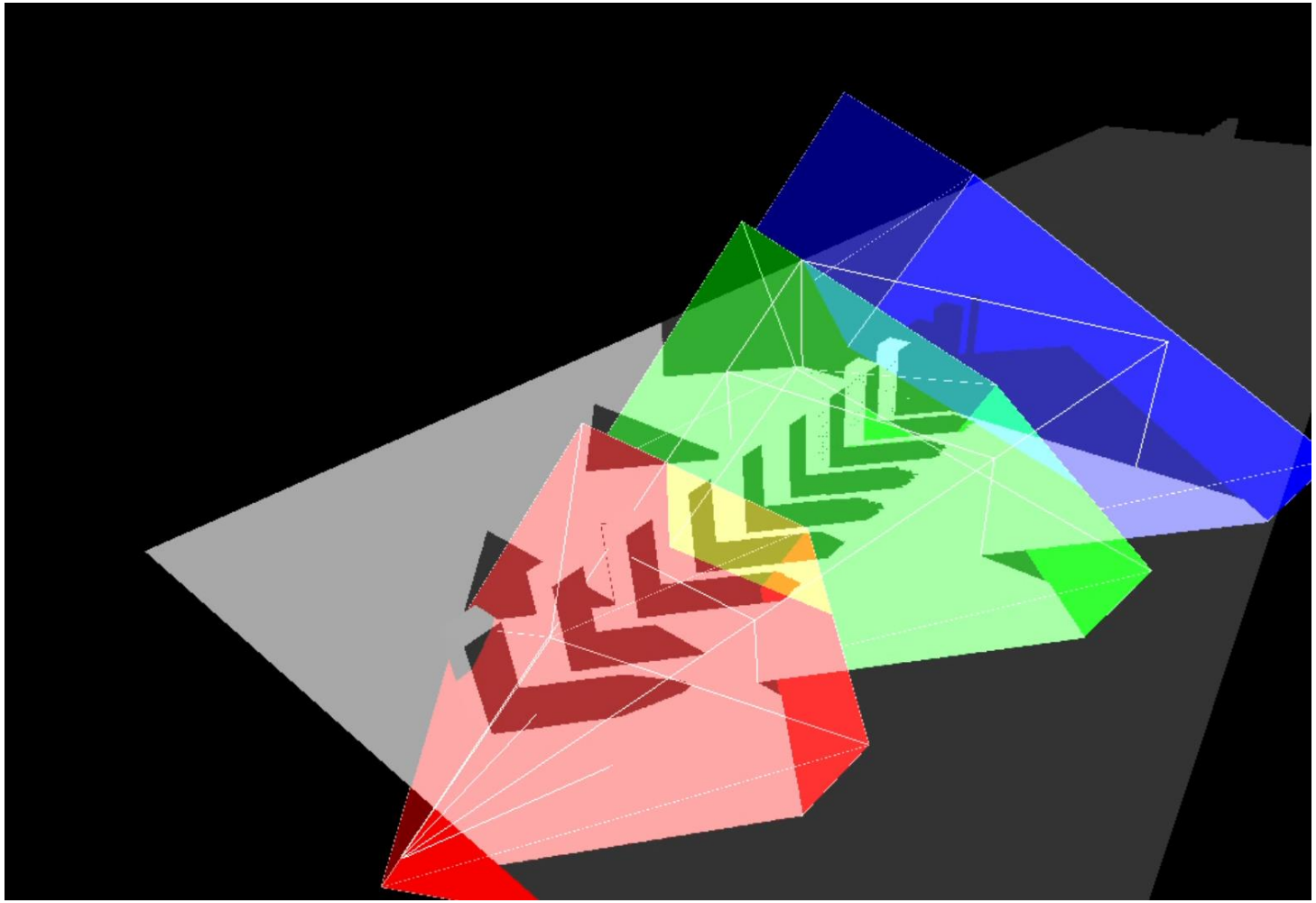




Occluder Distance

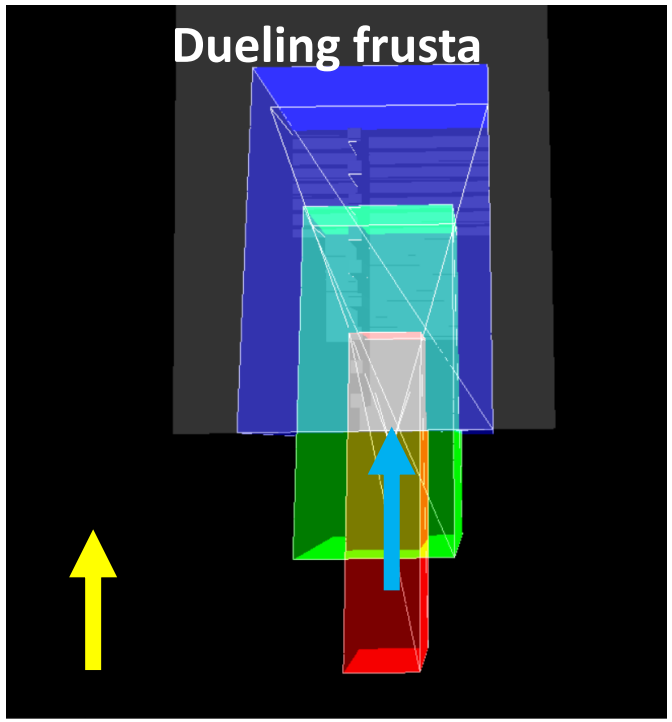
- Solution:
  - Do not tightly adjust the shadow map box to the subfrustum
  - Expand it by moving the near distance of the box by a specified amount
    - Maximum occluder casting distance
      - If an mesh is at a greater distance we cannot guarantee that it will always produce a shadow

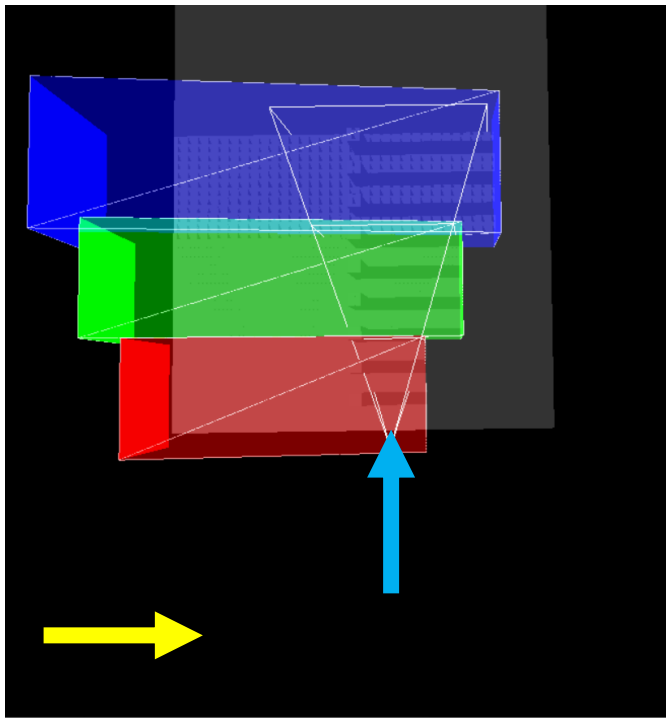
Occluder Distance

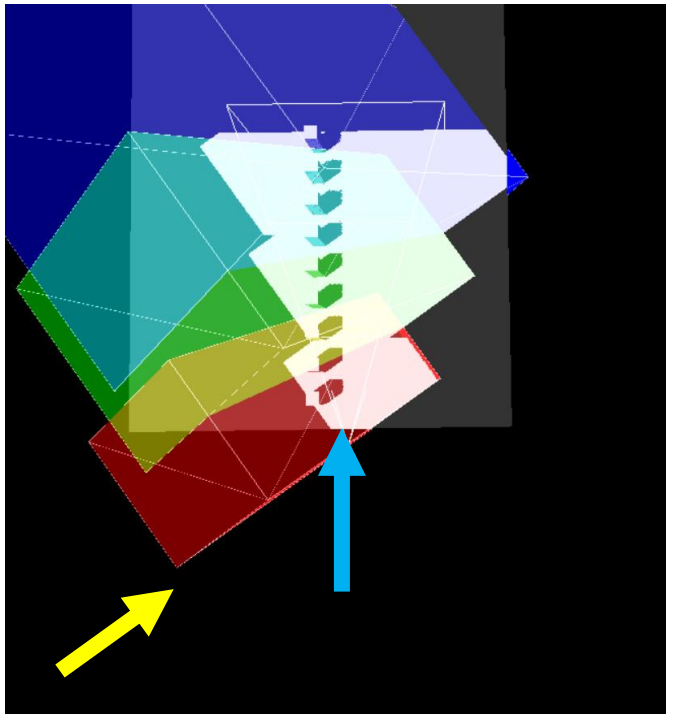


Orientation of the Scene Geometry

- The relative orientation of the **directional light** and the cameras **view vector** will affect the shadow quality





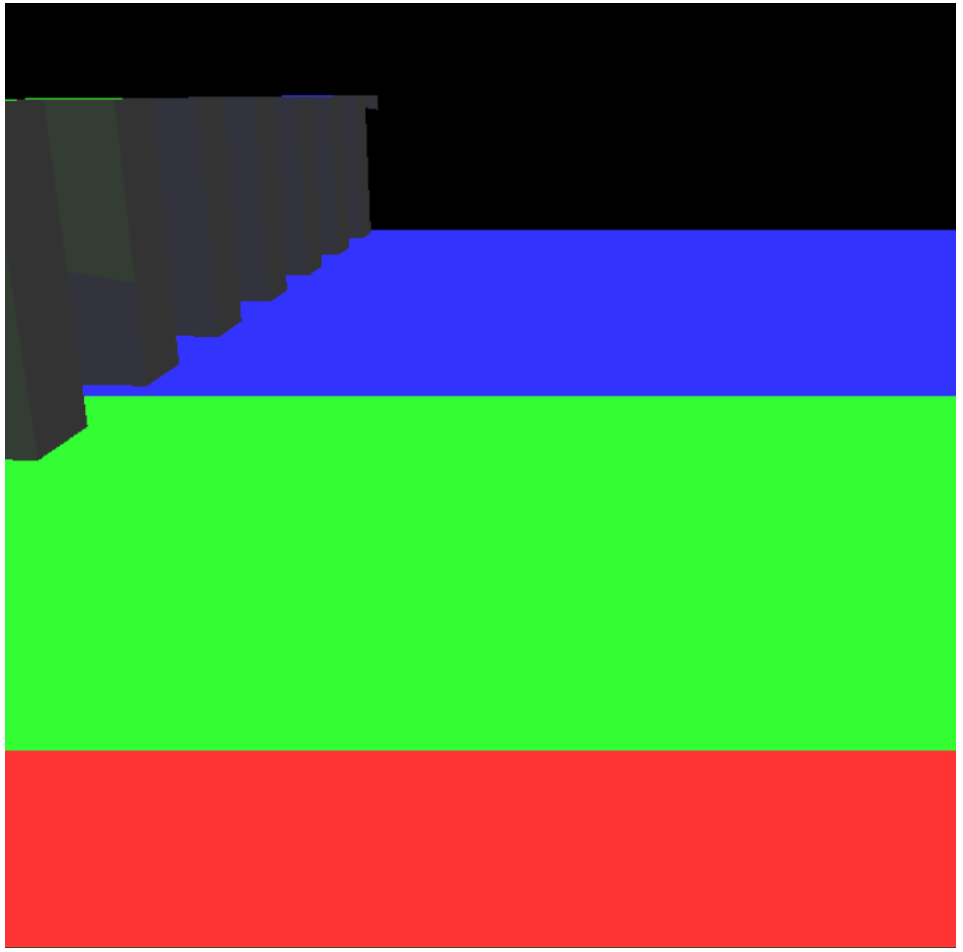


Shadow Map Selection

- When applying the shadow map on the lighting equation, one of the maps needs to be selected
  - Interval-based cascade selection
  - Map-Based Cascade Selection

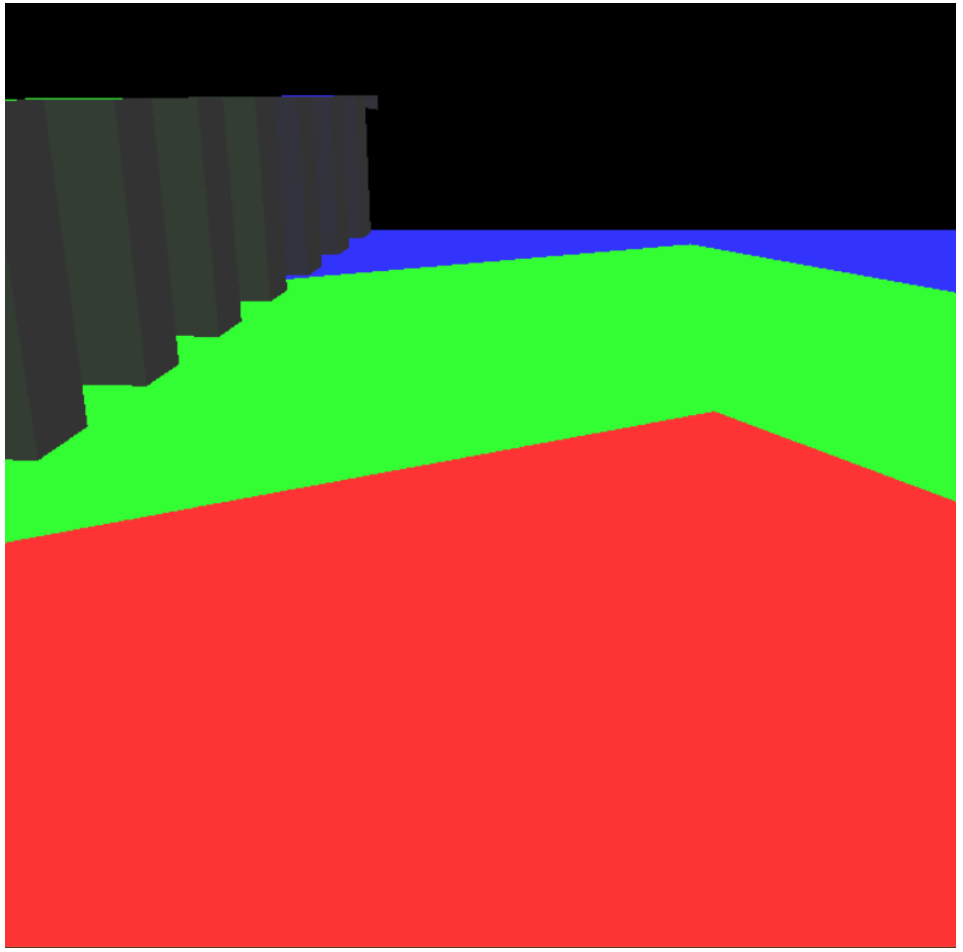
Interval-based Cascade Selection

- Given the split distances, select the cascade based on the depth from the camera
  - Easy to compute and more efficient
- Produces consistent perpendicular cuts



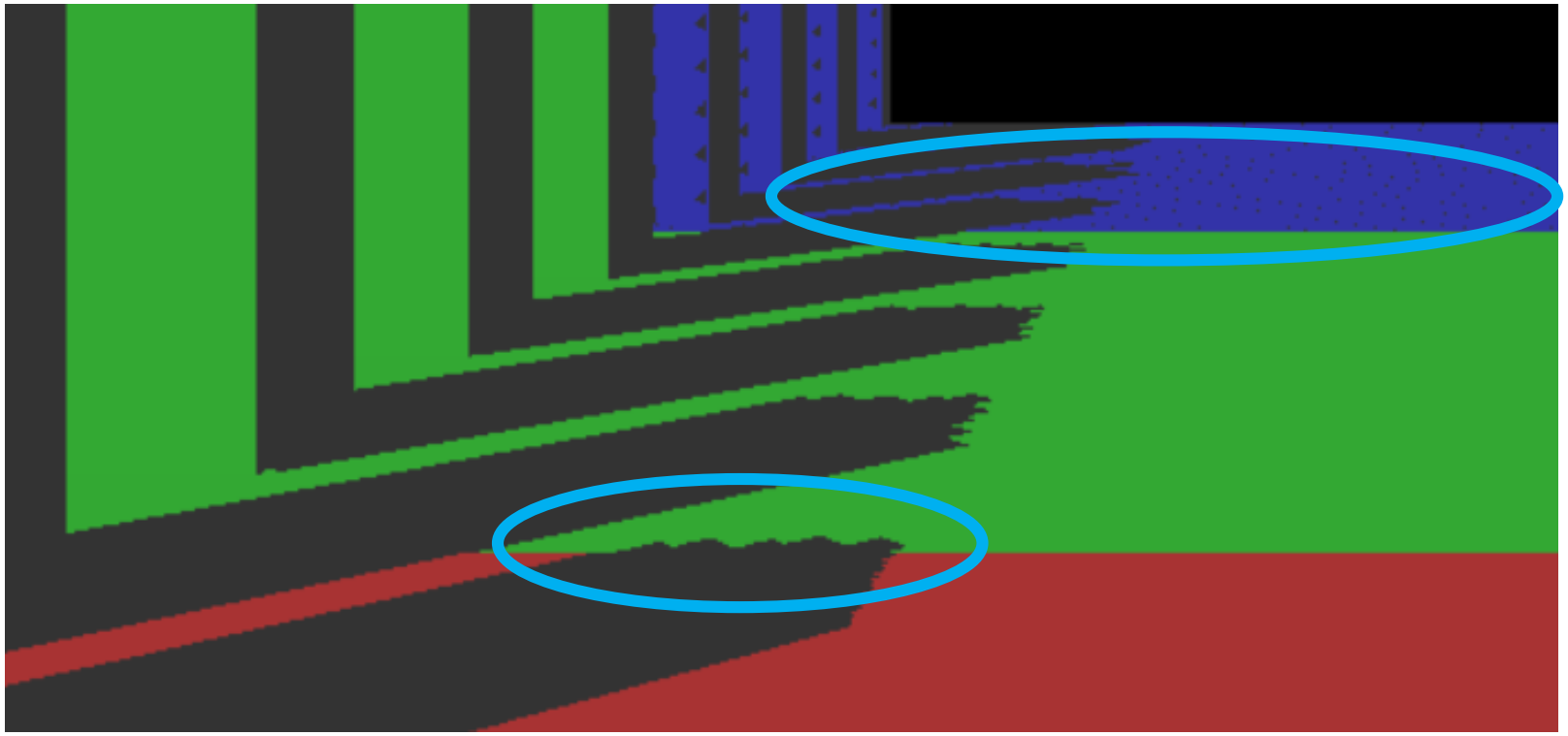
Map-based Cascade Selection

- Select best shadow map that contains shadow information for that pixel
  - Best usage of the shadow maps
- Compute the shadow map texture coordinates and select if inside the bounds (between 0 and 1)



Shadow Map Selection

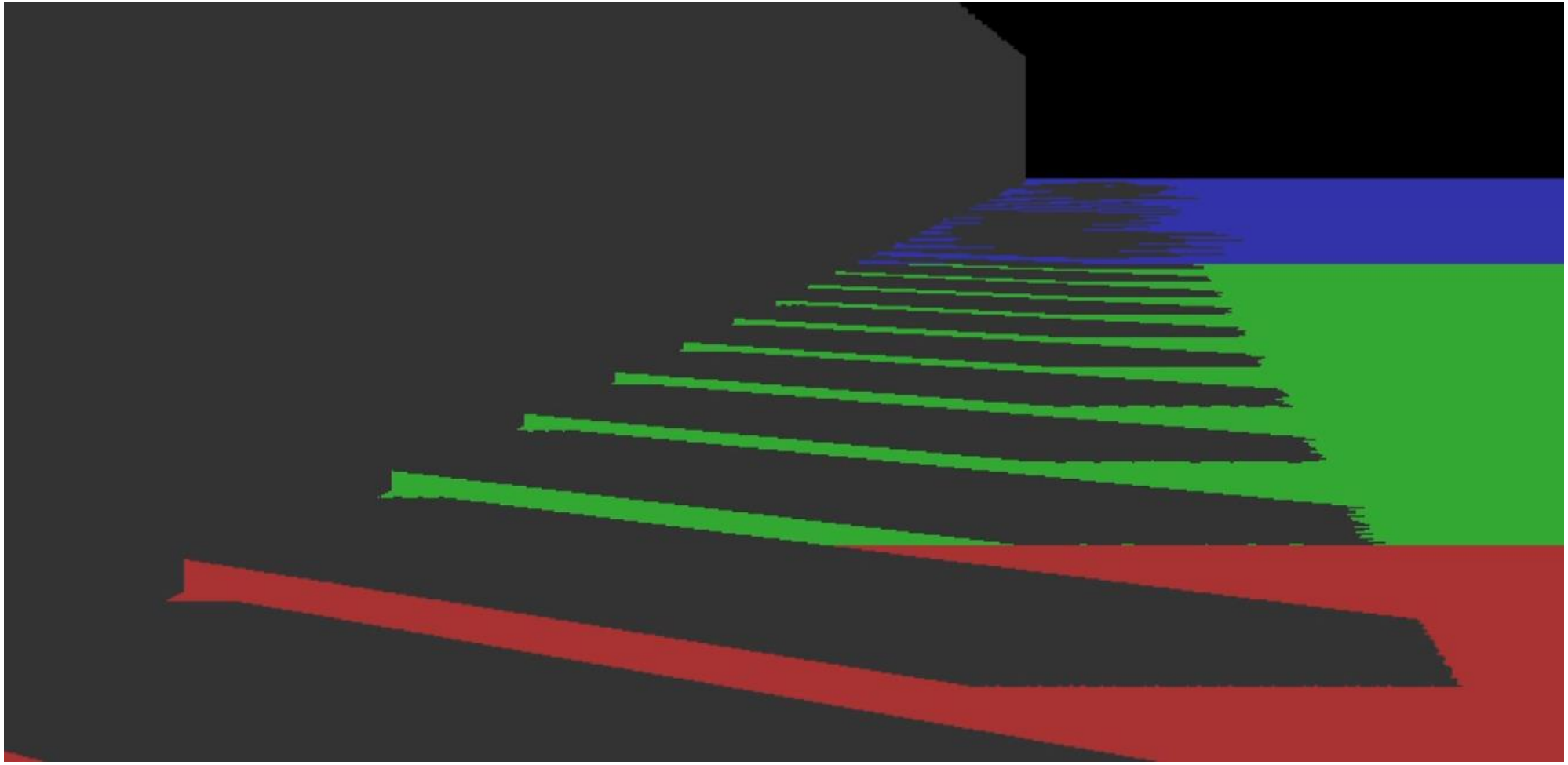
- These edges between different cascades will produce artifact due to the abrupt shadow quality change



Cascade Blending

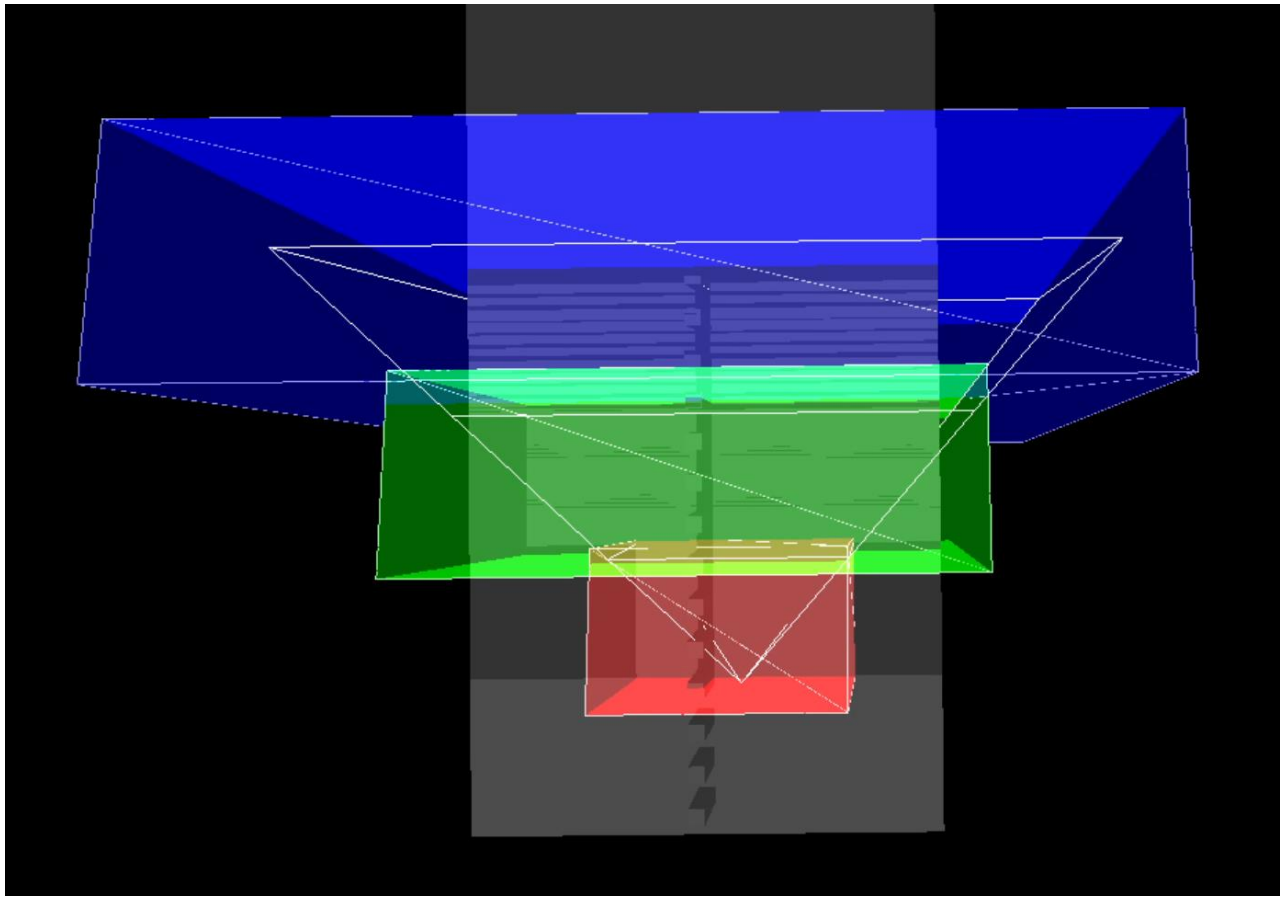
- Sampling from individual cascades produces clear seams
- Solution:
  - Create an area of smooth transition between different shadow maps
  - Linearly interpolate inside the area
  - We will need to define the length of this interpolation band

Cascade Blending



Cascade Blending

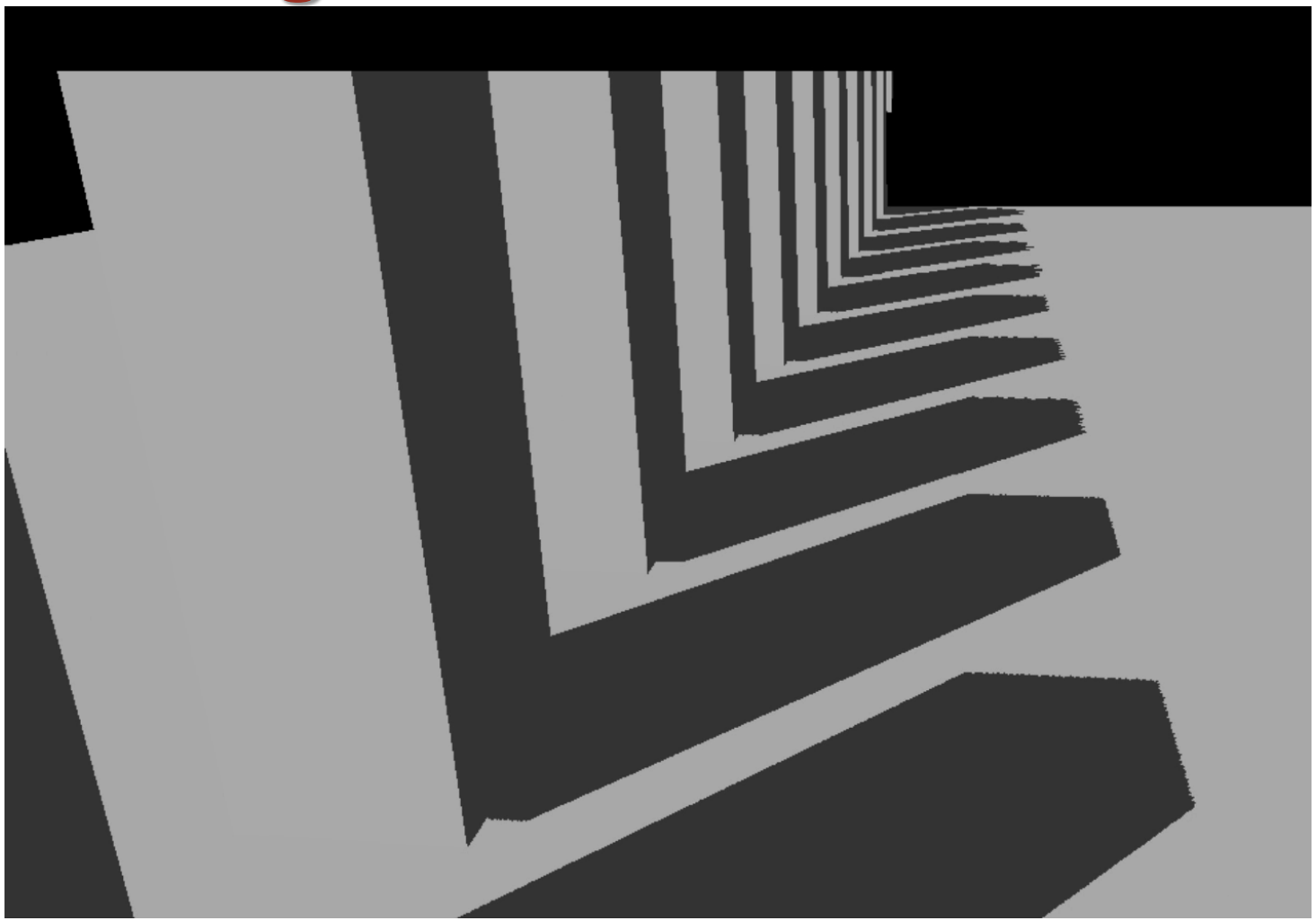
- In order to extend the blending area, the subfrusta cannot be mutually exclusive
  - Blending area requires to be included on both shadow maps



Aliasing

- Cascaded shadow mapping will not solve aliasing
  - Minimizes artifacts
  - Improves shadow map usage
- It can be combined with other enhancement techniques like Percentage Closer Filtering (PCF)

Cascade Blending with PCF



References

- https://developer.download.nvidia.com/SDK/10.5/opengl/src/cascaded_shadow_maps/doc/cascaded_shadow_maps.pdf
- https://docs.microsoft.com/en-us/windows/win32/dxtecharts/cascaded-shadow-maps
- https://dev.theomader.com/cascaded-shadow-mapping-1/
- https://docs.microsoft.com/en-us/windows/win32/dxtecharts/common-techniques-to-improve-shadow-depth-maps
- https://developer.nvidia.com/gpugems/gpugems3/part-ii-light-and-shadows/chapter-10-parallel-split-shadow-maps-programmable-gpus