Shadow Maps

+


AMERICA
THE SOUTH SEA
THE PERUVIANE SEA
THE PACIFICKE SEA
THE NORTH SEA
THE ATLANTICKE
THE OCEAN
THE EQUINOCTIAL CIRCLE
THE UNKNOWNE WORLD

Terms and concepts

- Light source
- Depth buffer (z buffer)
- View frustum
- Bias
- Cascading shadow maps

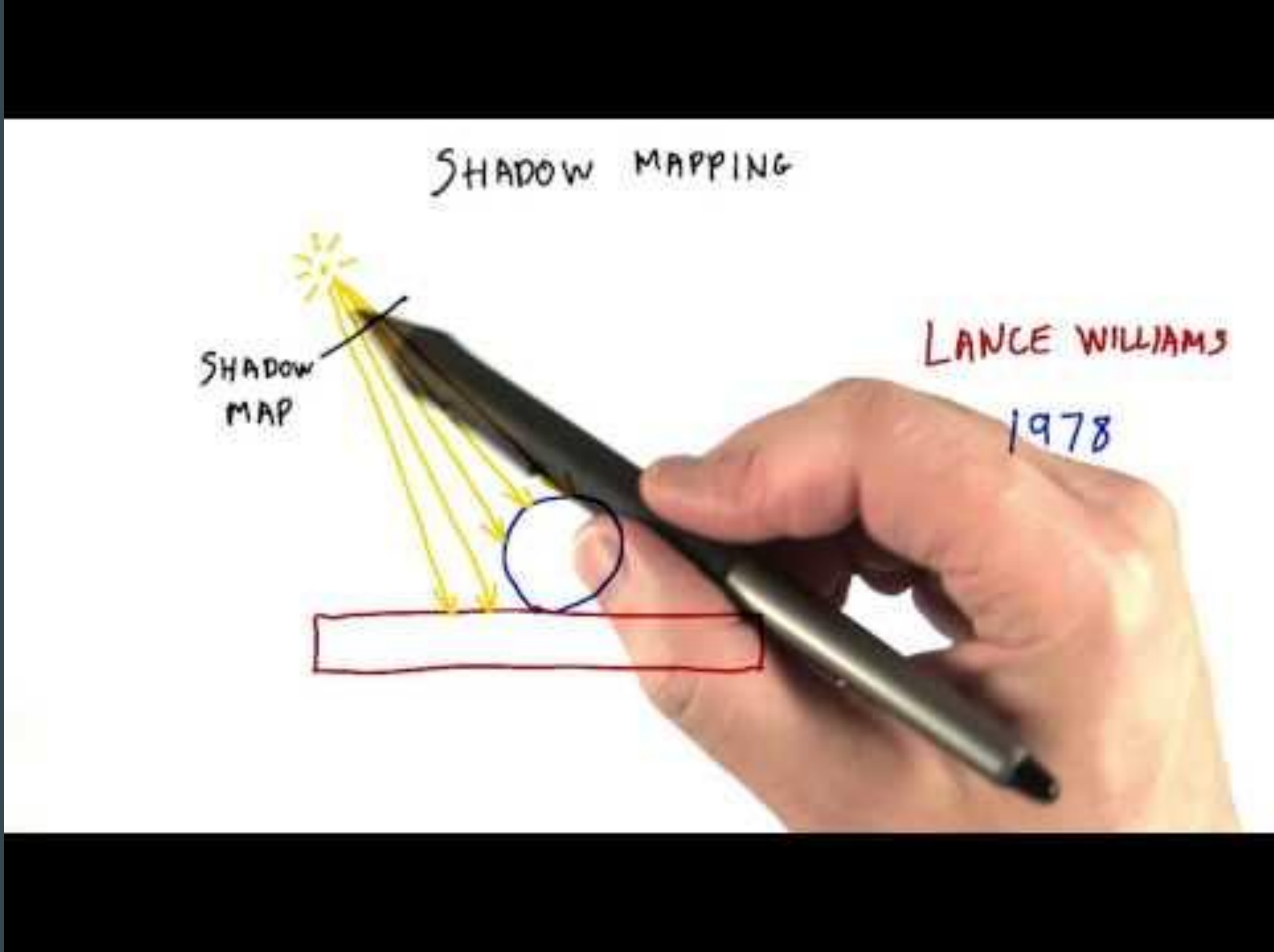
What is Shadow Mapping?

Shadow mapping or shadowing projection is a process by which shadows are added to 3D computer graphics. This concept was introduced by **Lance Williams** in 1978, in a paper entitled "Casting curved shadows on curved surfaces." Since then, it has been used both in pre-rendered and real time scenes in many console and PC games.

Also invented mip-mapping

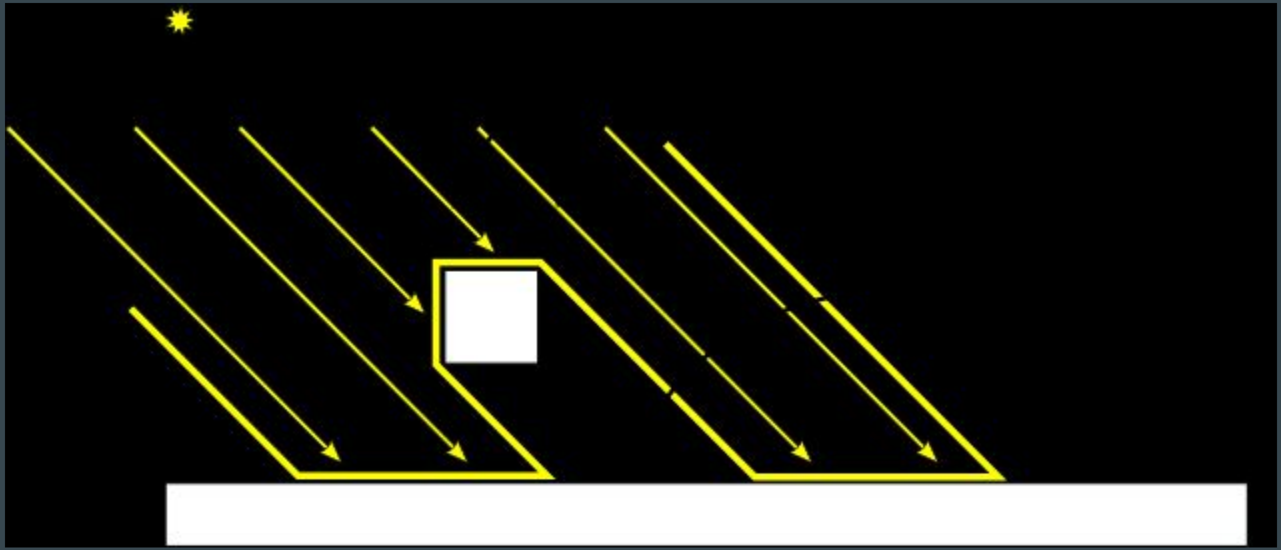
Shadow Mapping - Interactive 3D Graphics

By: Udacity



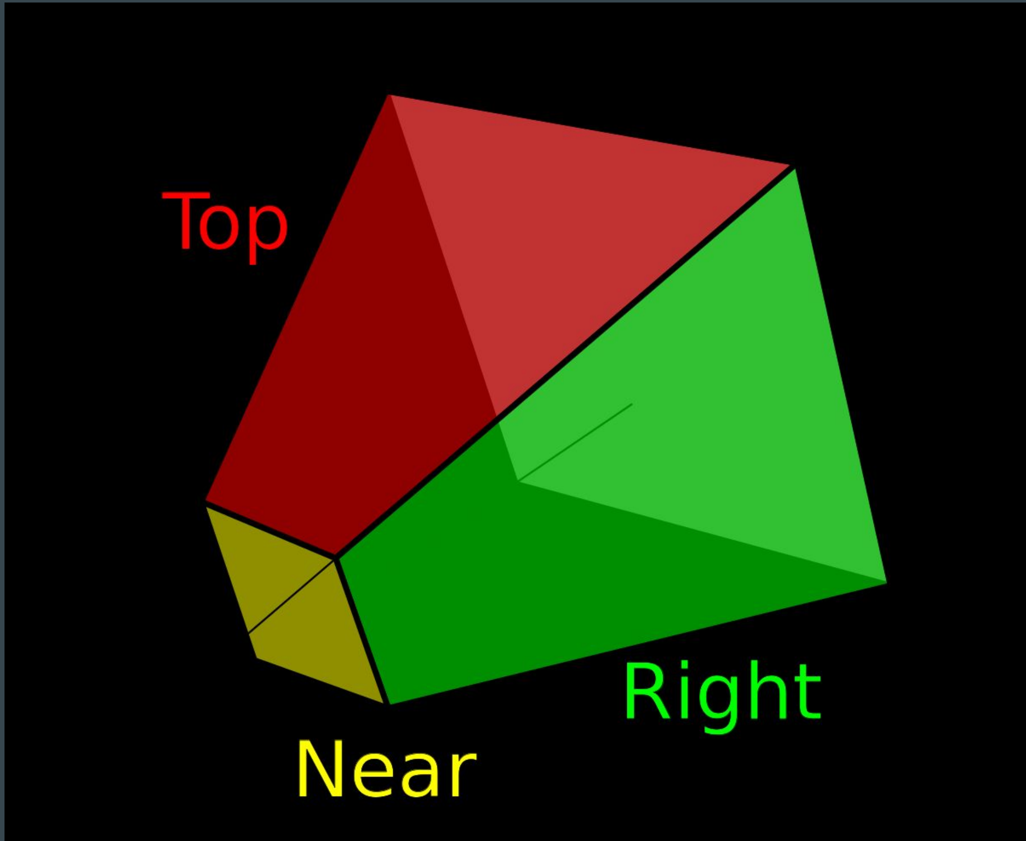
High level overview of shadow mapping

Each ray emitted by the light source can strike at most, one opaque surface. Whichever surface is closest to the light along a particular ray will receive all of the illumination carried by the ray. Any surfaces farther away in the same direction must be in shadow.



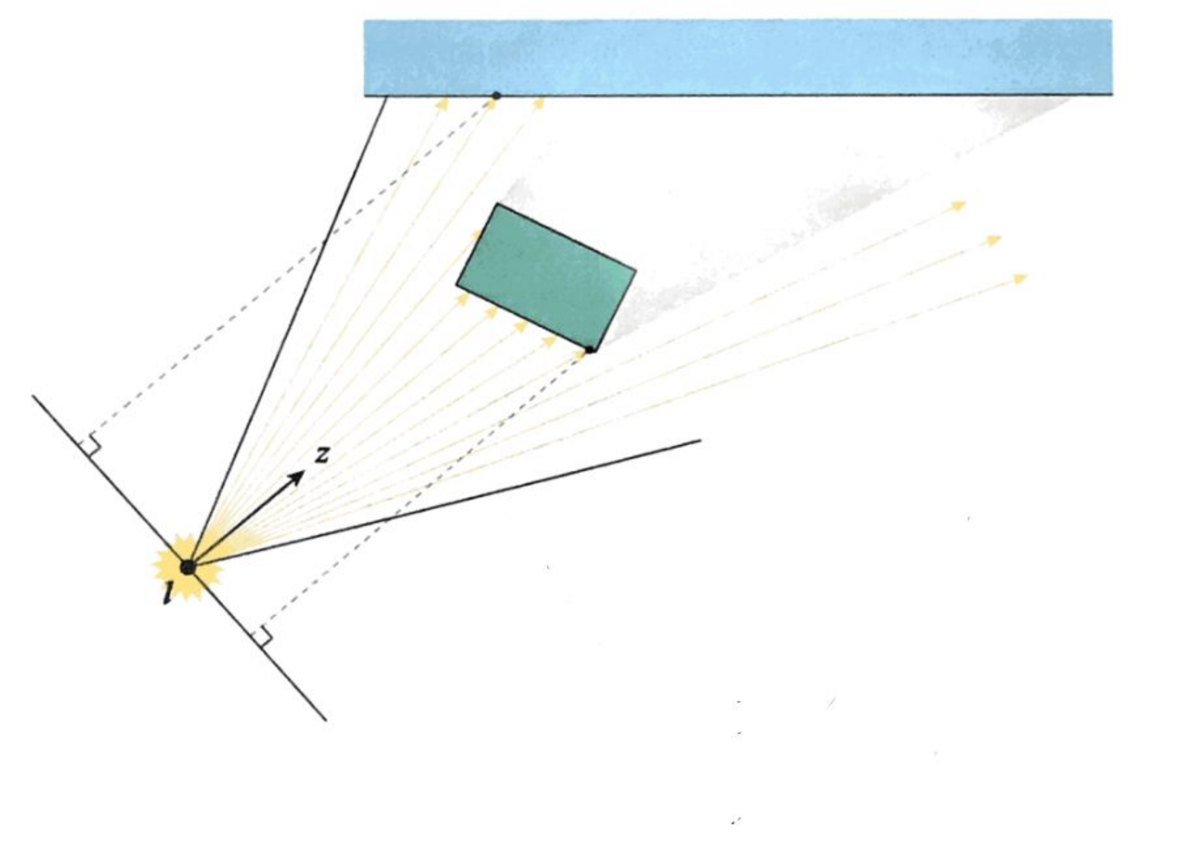
Rendering a 2D Shadow Map

To render a Shadow Map, we essentially add a second view frustum to the world. The Shadow map is simply the contents of the depth buffer that we end up with when rendering the world using the second view frustum.



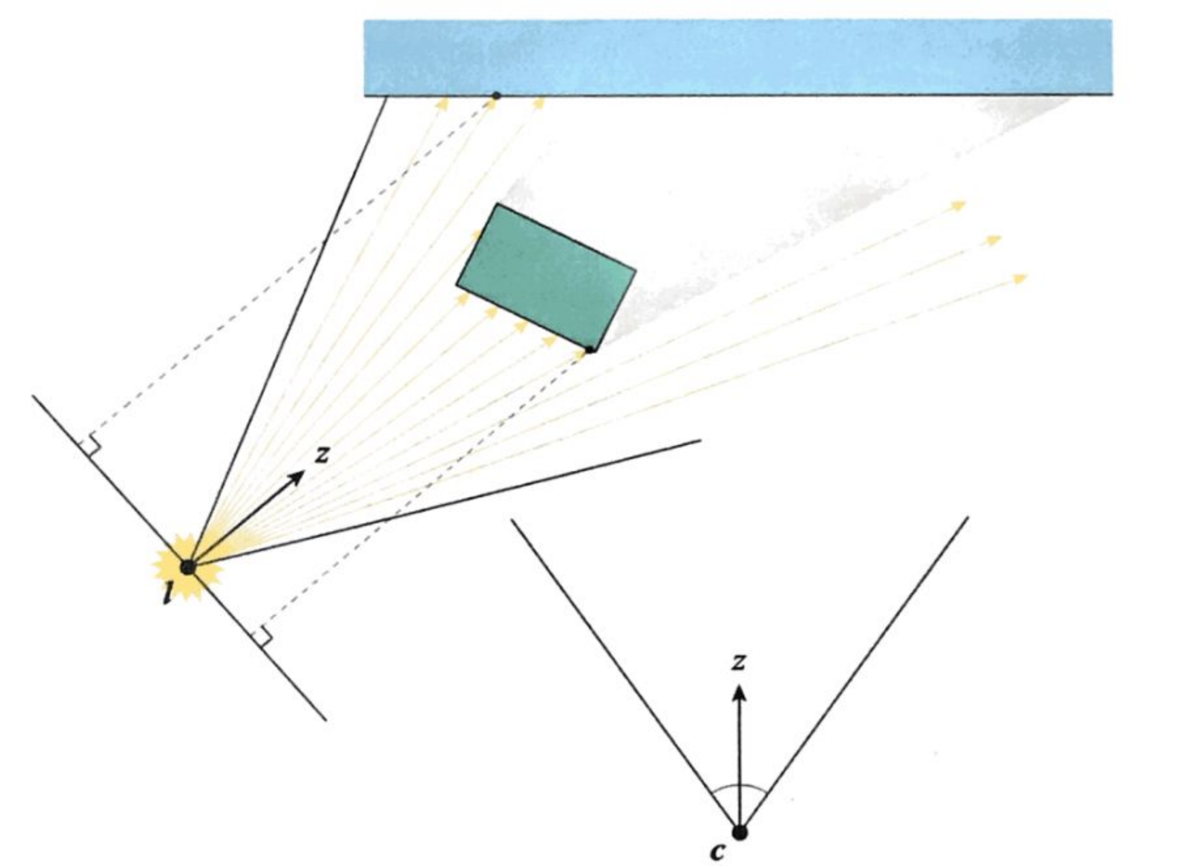
Rendering 2D Shadow Maps with a Spotlight

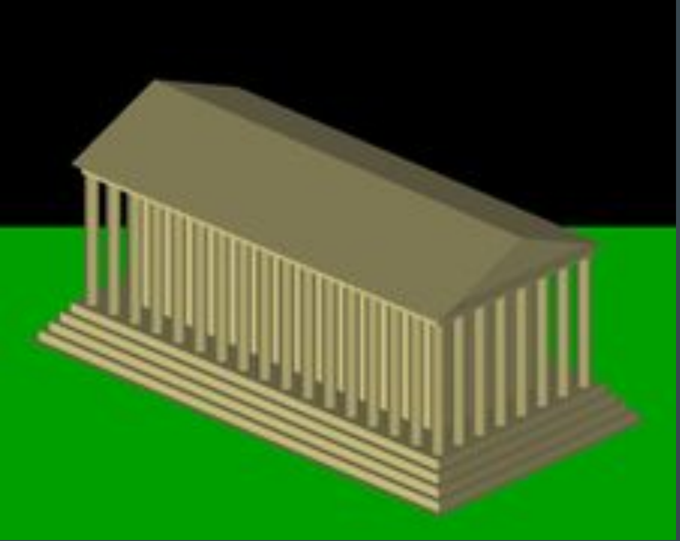
- Render a depth buffer from the perspective of the light source.
- For each ray the, starting at the lights position, the shadow map contains the projected depth, parallel to the lights z axis, corresponding to the nearest opaque surface



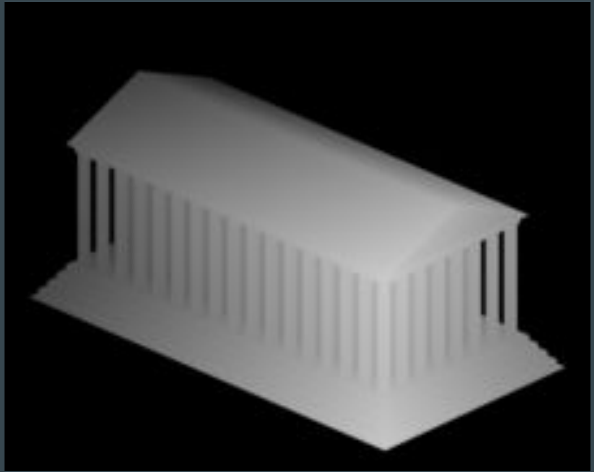
Rendering 2D Shadow Maps with a Spotlight continued

- When the scene is rendered from the camera's position, points on visible surfaces are transformed into light space.
- The Projected z coordinate of each point is compared to the value stored in the shadow map, to determine whether it is in shadow.

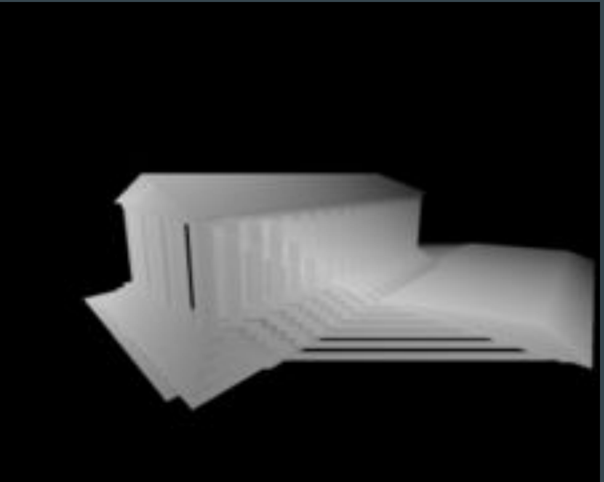




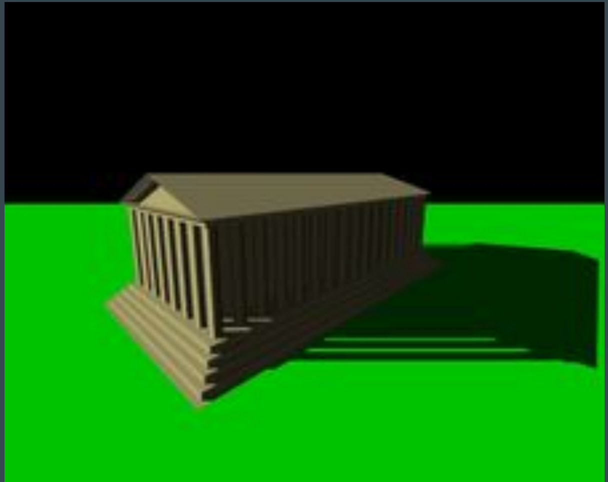
Rendered from the lights perspective



Depth map rendered from the lights POV



Depth Map projected onto the scene



Final scene, rendered and shaded

Low level Details for 2D Shadow Maps

This pixel shader function samples the 2D shadow map that has been rendered for the spotlight. The *shadow coord* parameter contains the interpolated homogeneous coordinates produced by the projection matrix below. The uniform constant *shadowOffset* specifies the space between shadow map samples, and it would typically be set to the reciprocal of the shadow map resolution in texels.

```
uniform Texture2DShadow   shadowTexture;
uniform float             shadowOffset;

float CalculateSpotShadow(float4 shadowCoord)
{
   // Project the texture coordinates and fetch the center sample.
   float3 p = shadowCoord.xyz / shadowCoord.w;
   float light = texture(shadowTexture, p);

   // Fetch four more samples at diagonal offsets.
   p.xy -= shadowOffset;
   light += texture(shadowTexture, p.xy, p.z);
   p.x += shadowOffset * 2.0;
   light += texture(shadowTexture, p.xy, p.z);
   p.y += shadowOffset * 2.0;
   light += texture(shadowTexture, p.xy, p.z);
   p.x -= shadowOffset * 2.0;
   light += texture(shadowTexture, p.xy, p.z);

   return (light * 0.2);  // Return average value.
}
```

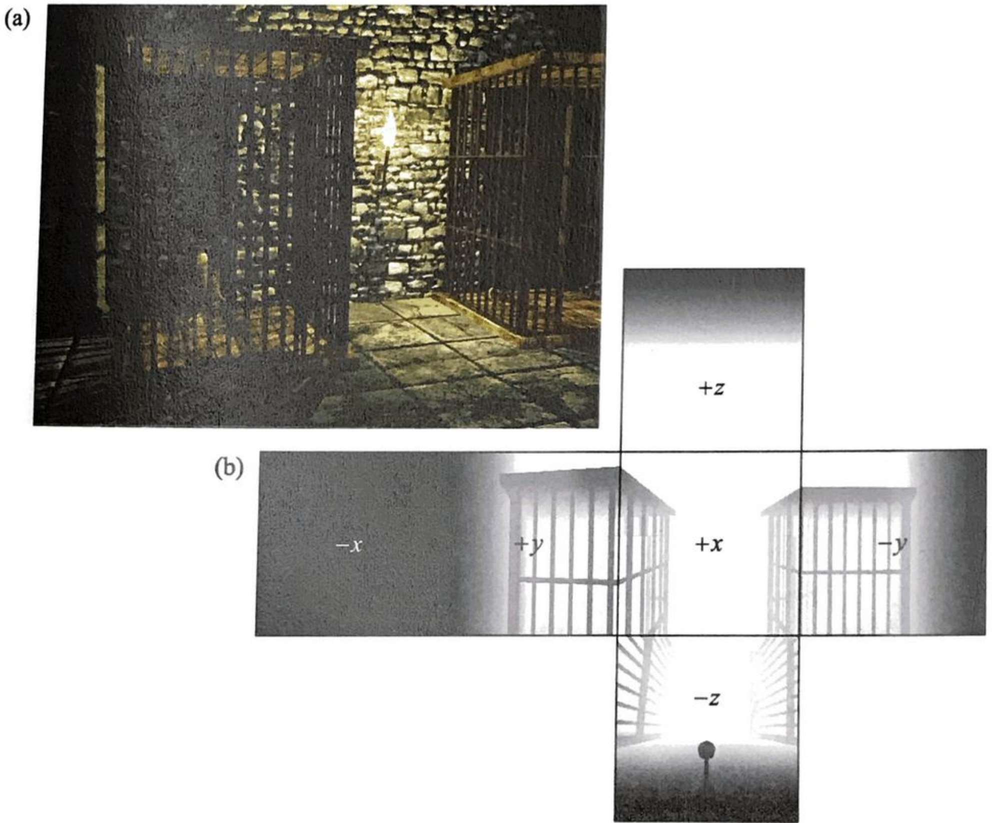
$$\mathbf{P}_{\text{shadow}} = \begin{bmatrix} g/2s & 0 & 1/2 & 0 \\ 0 & g/2 & 1/2 & 0 \\ 0 & 0 & \dfrac{r_{\max}}{r_{\max} - n} & -\dfrac{nr_{\max}}{r_{\max} - n} \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Rendering a 3D Shadow Map with point lights

To handle omnidirectional shadows, we need to surround a point light with multiple shadow maps storing the depths of shaodow casting geometries. This is most commonly stored as a cube.

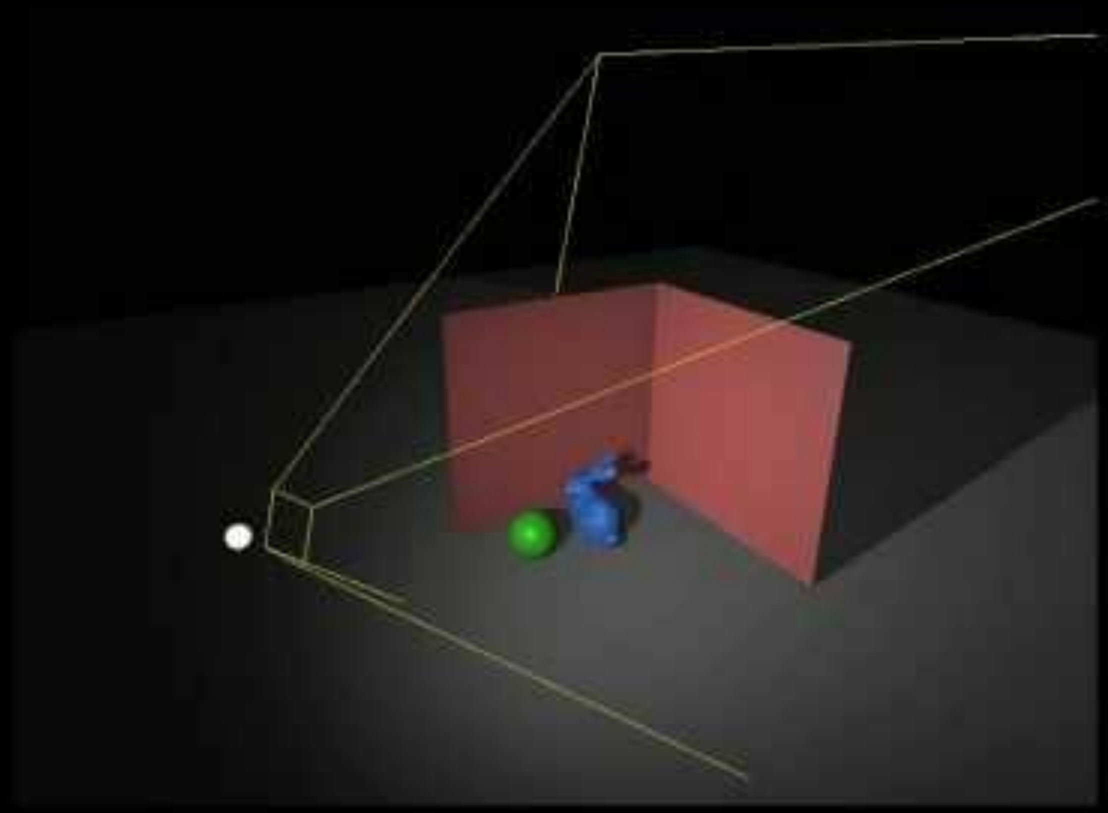
Rendering a 3D Shadow Map with point lights continued

Shadows rendered from point lights use a cube shadow map. 5 samples are stores per pixel, and the depths are stored in the shadow map as six grayscale images rendered from the perspective of the light source.



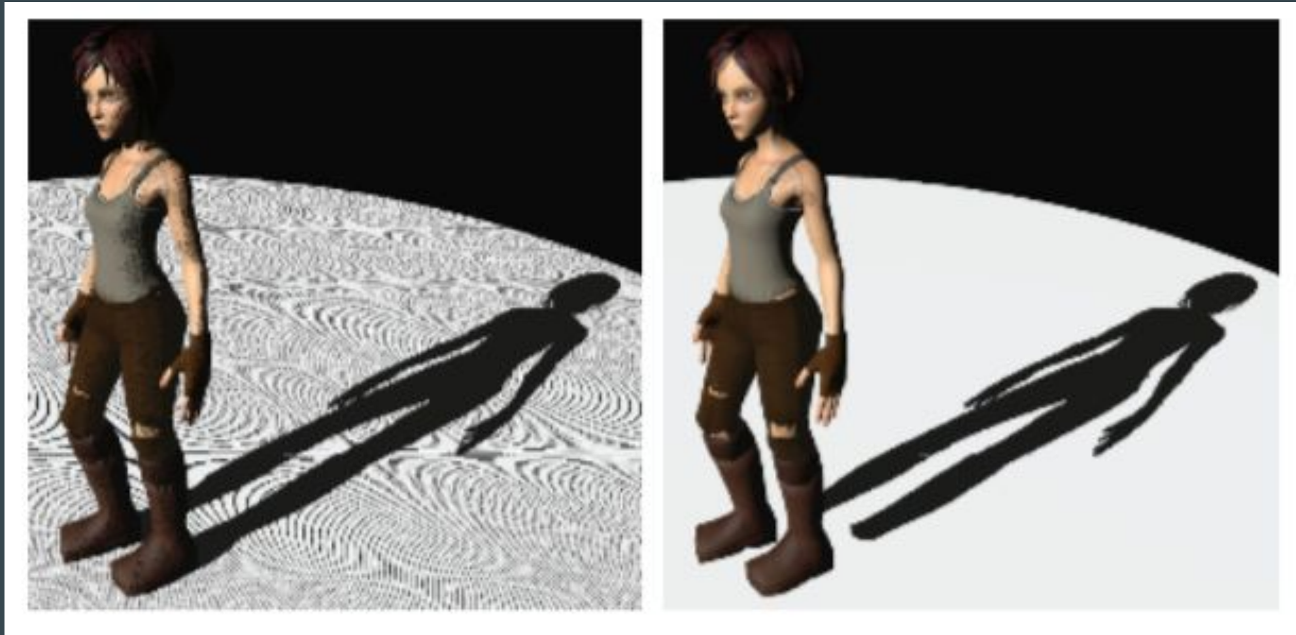
Shadow Mapping Demo

By: Dustin Biser



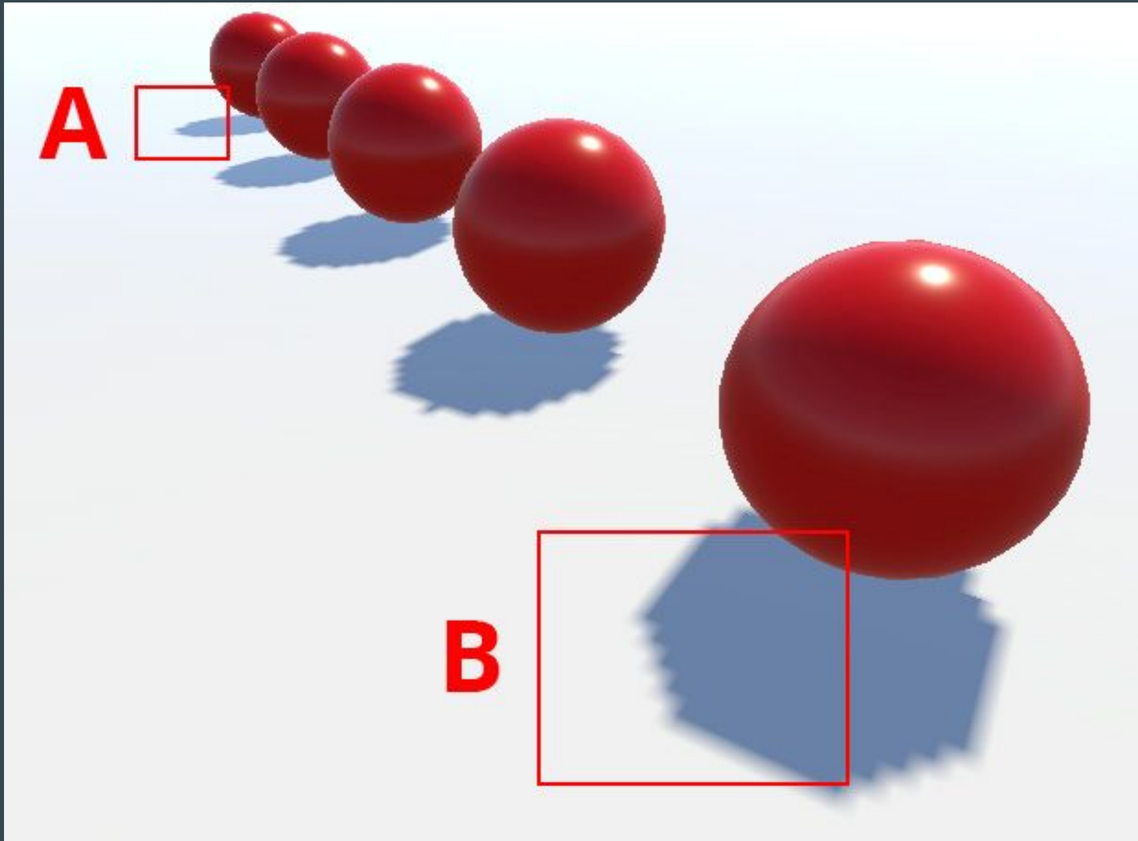
Bias

Because the shadow map is calculated at a different angle from the camera, sometimes the value of the depth in screen space can differ from the shadow map. A bias term in introduced to ensure that pixels will not shadow themselves.



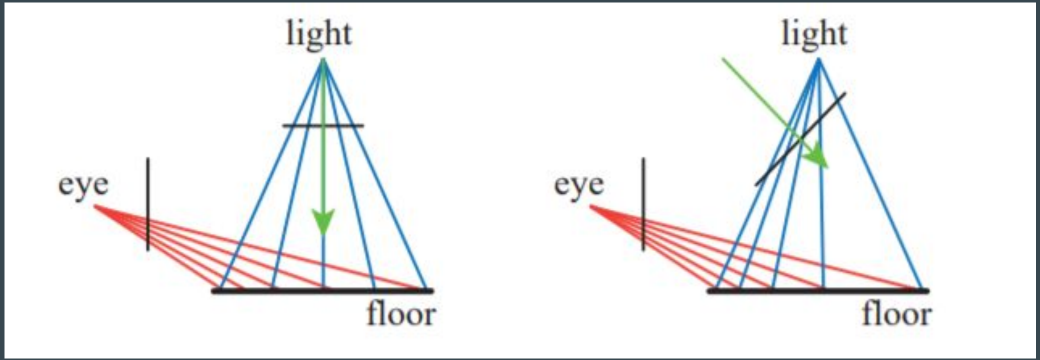
Resolution

Because shadow maps are taken from a different angle as the main camera, the shadows can become blocky.



Perspective Warping Methods

One way to deal with shadow map resolution problems is to change shadow map so that it will have a higher resolution closer to the players viewpoint. This method does not work if the light is facing the camera.

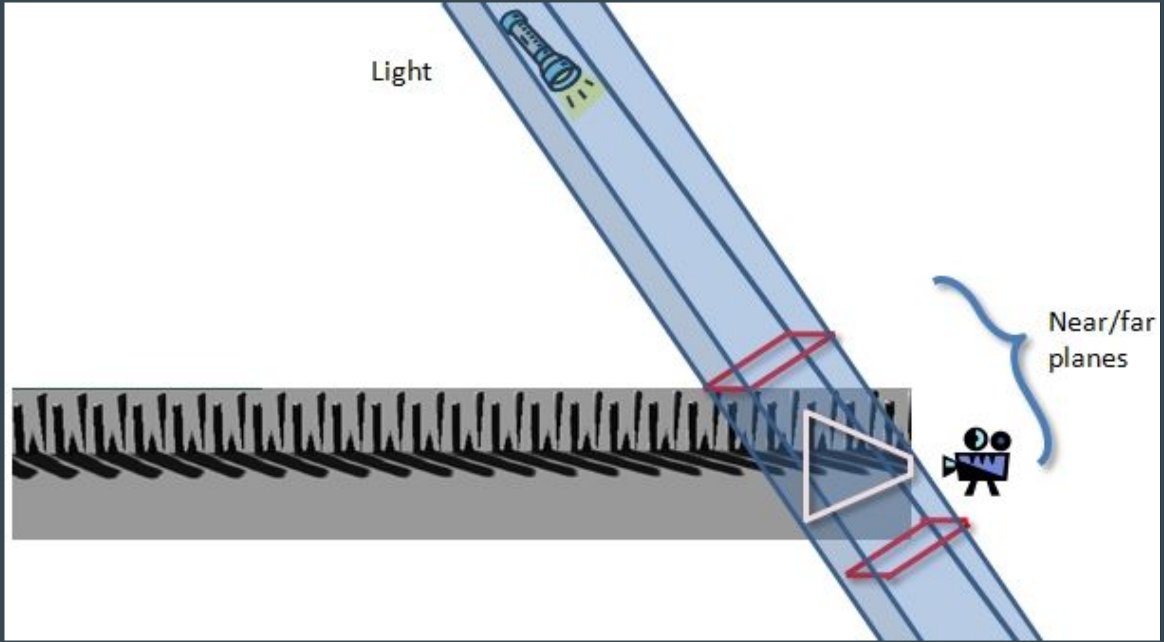


How they work in game engines: Unreal

UE4 offers 4 types of Shadow casting: Static Shadows, Dynamic Shadows, Whole Scene Shadows, and Preview Shadows. Static shadows are free as far as rendering goes in Unreal Engine 4, but dynamic shadows can be one of the biggest drains on performance.

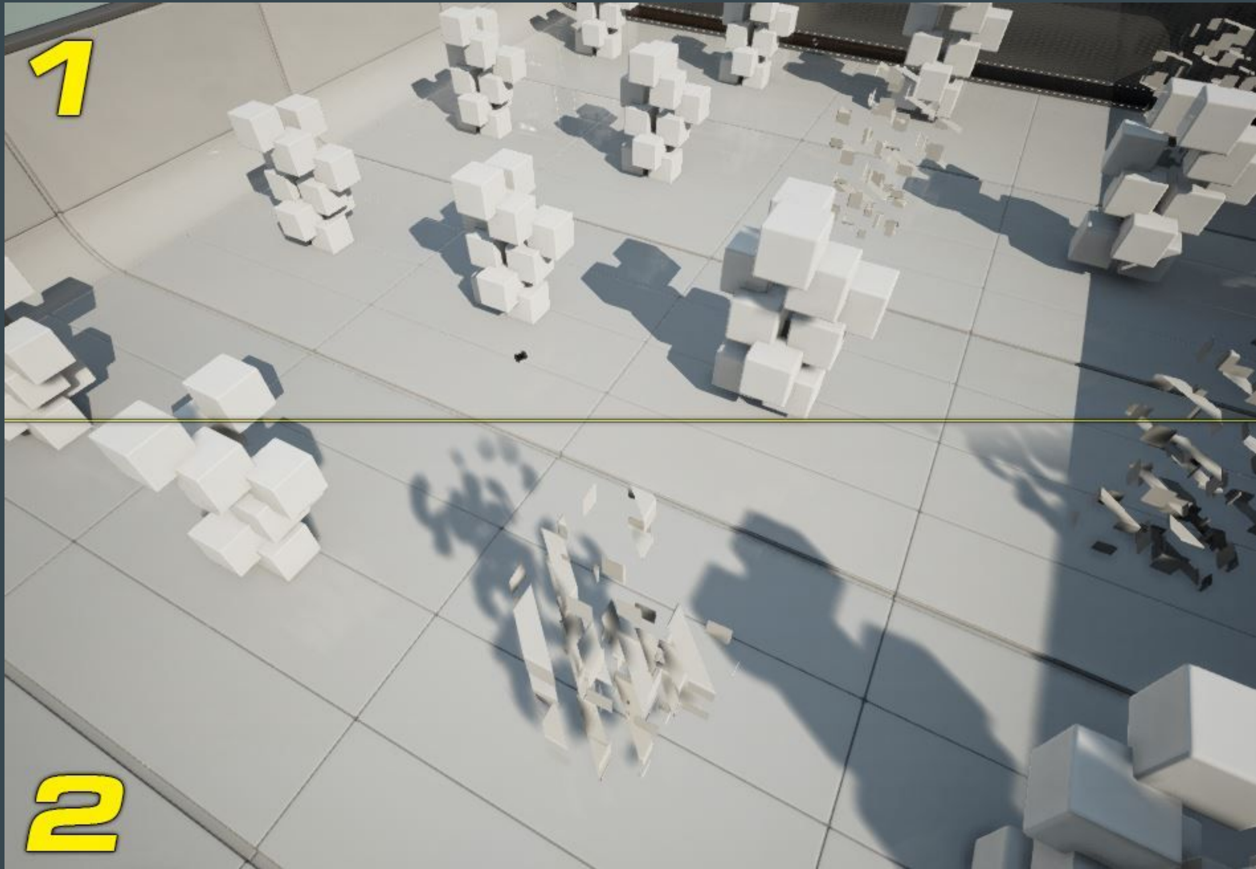
Efficiency

- Low detail models are used to cast the shadows
- Shadows for static objects are calculated only once.
- Shadows for moving objects are only calculated when they are near the camera.



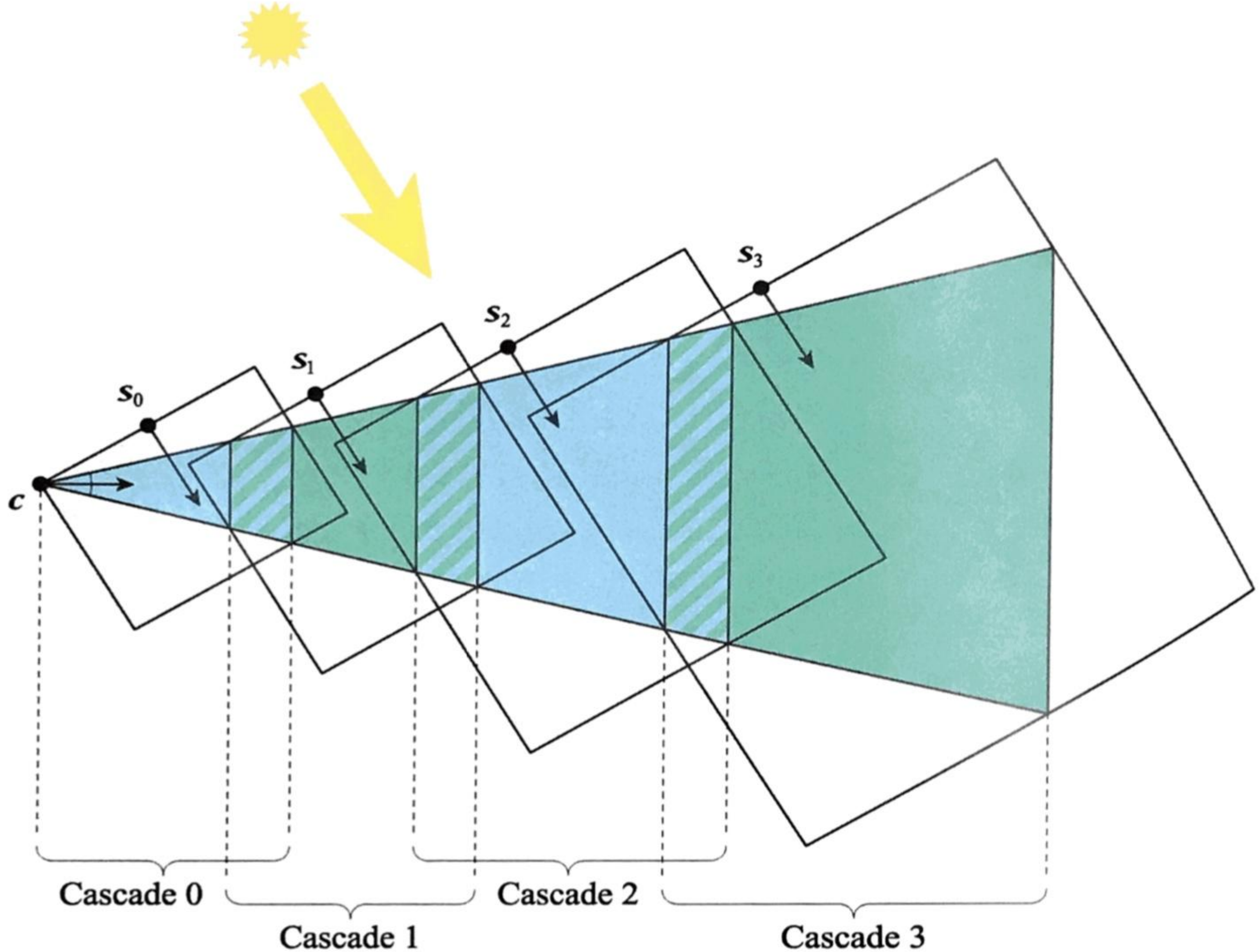
Cascading Shadow Map

The Cascading Shadow Map method divides the view frustum into a series of distance-based shadow cascades, each of which with steadily lower resolution as you move farther from the camera. This means that shadows nearest the player's view have the highest resolution while those farthest away have the lowest.



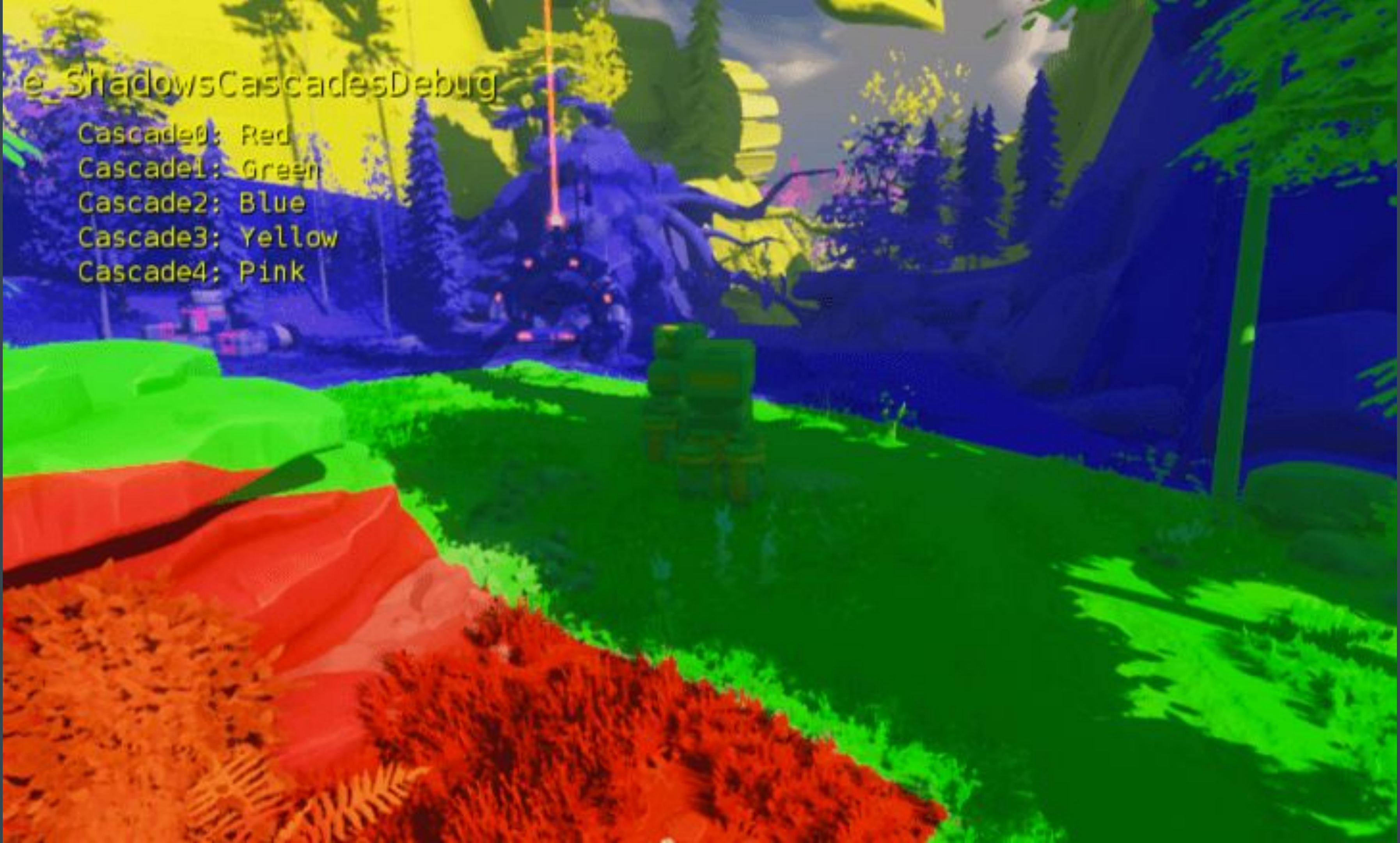
Cascade Shadow Maps continued

In order to get higher resolution closer to the camera, but still deal with the possibility of looking directly at a light source. Shadow maps of different resolutions can be nested, and different ones will be used at different depths in the scene.



s0
s1
s2
s3
c
Cascade 0
Cascade 1
Cascade 2
Cascade 3

(a)
(b)
0
1
2
3


e_ShadowsCascadesDebug
Cascade0: Red
Cascade1: Green
Cascade2: Blue
Cascade3: Yellow
Cascade4: Pink

Problems With Planes

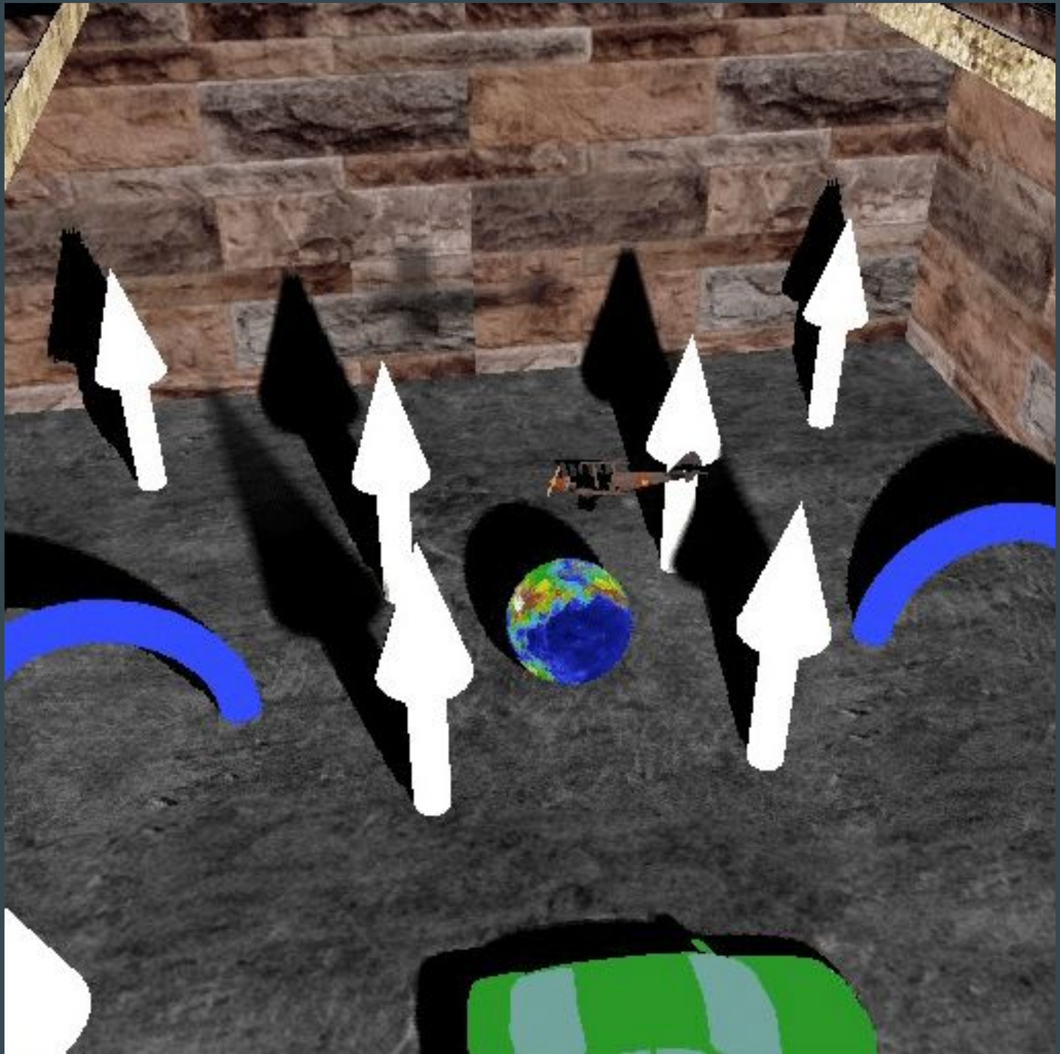
Cascade shadow maps need to divide the area between the near and far clip planes. Based on where the view frustum is placed. If it is placed too close to the camera, each shadow map needs to cover more area. If it is placed too far, bad artifacts can show up as an object is not fully rendered.


NOTICE
AUTHORIZED
PERSONNEL ONLY
100+
61
30
57
2


LightPrePass


potatochip878
Account: 13+
KOs 0
TKs 0
WOs 0
Homeless 0 0 0
potatochip878 0 0 0
Lower Class
Middle Class
Upper Class
1
Reset


1.1 - Cascading Shadow Maps



Live demo: Shader Toy

https://www.shadertoy.com/view/MlKSRm

References

- https://www.youtube.com/watch?v=0e01qWckKD4
- http://cseweb.ucsd.edu/~ravir/274/15/papers/p270-williams.pdf
- http://www.cse.chalmers.se/~uffe/ClusteredWithShadows.pdf
- https://docs.microsoft.com/en-us/windows/win32/dxtecharts/common-techniques-to-improve-shadow-depth-maps
- https://docs.unrealengine.com/en-US/Engine/Rendering/LightingAndShadows/Shadows/index.html
- https://docs.unity3d.com/Manual/ShadowOverview.html