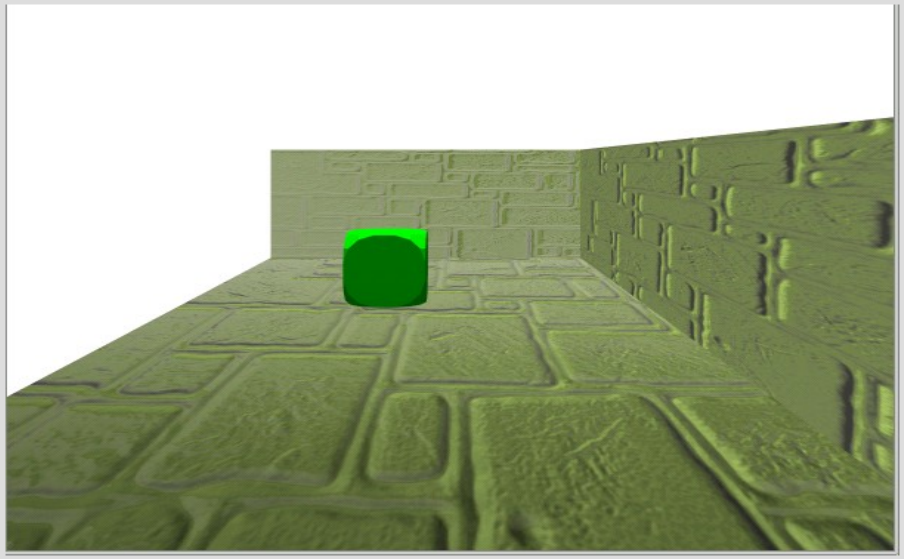
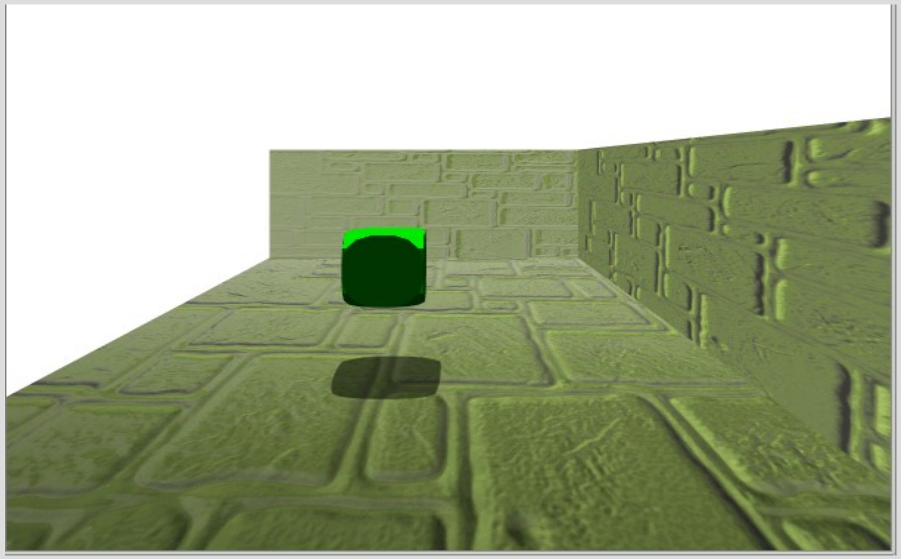
Shadows in Computer Graphics

by Björn Kühl
im/ve
University of Hamburg, Germany

Importance of Shadows

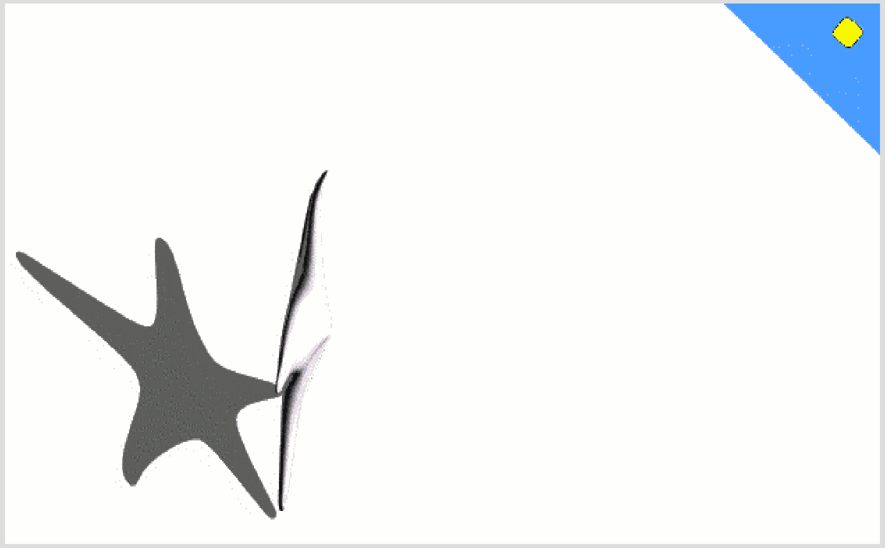
- Shadows provide cues
  - to the position of objects casting and receiving shadows
  - to the position of the light sources

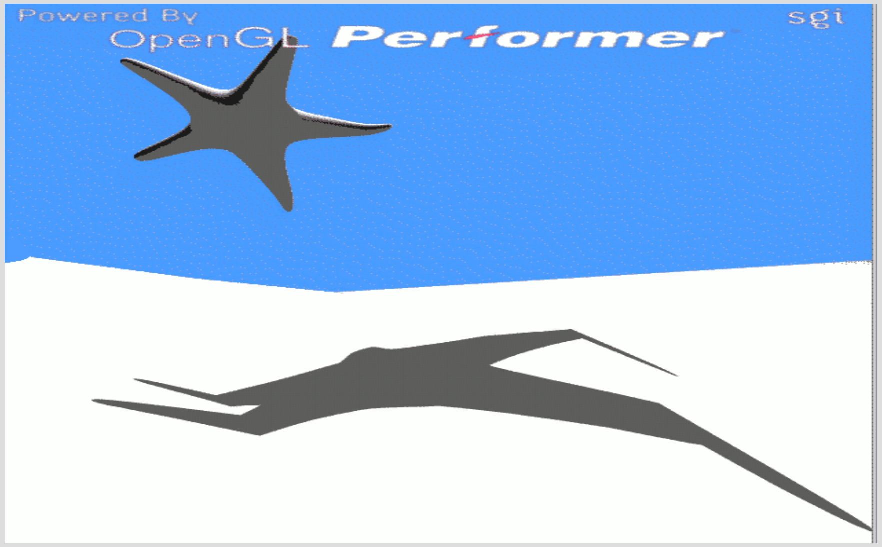




Importance of Shadows

- Shadows provide information
  - about the shape of casting objects
  - about the shape of shadow receiving objects





Shadow Tutorial

- Popular Shadow Techniques
- Implementation Details
  - Stencil Shadow Volumes (object based)
  - Shadow Mapping (image based)
- Comparison of both techniques

Popular Shadow Methods

- Shadows can be found in most new games
- Two methods for shadow generation are predominately used:
  - Shadow Mapping
    - Need for Speed
    - Prince of Persia
  - Stencil Shadow Volumes
    - Doom3
    - F.E.A.R
    - Prey

Stencil Shadow Volumes

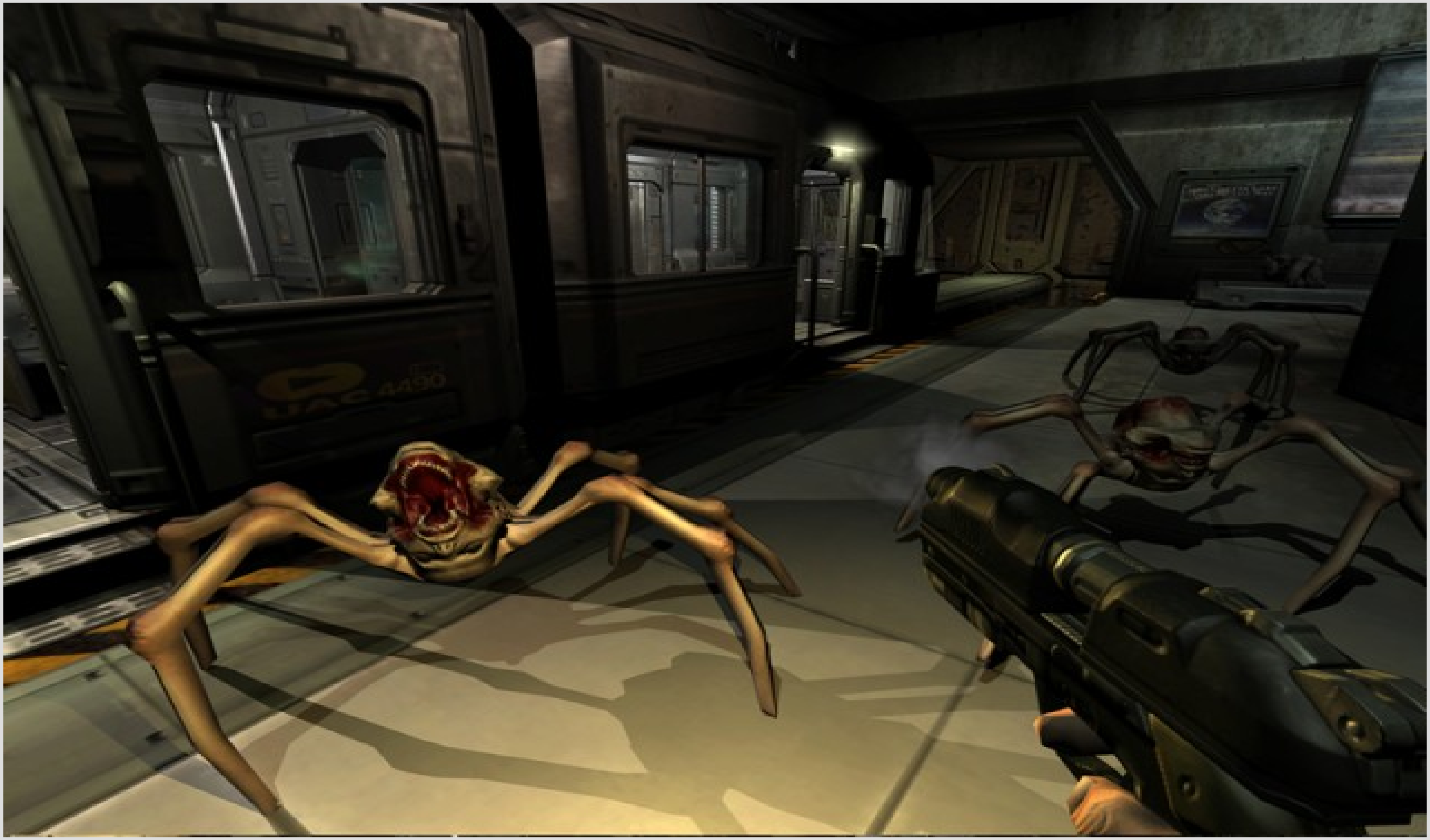
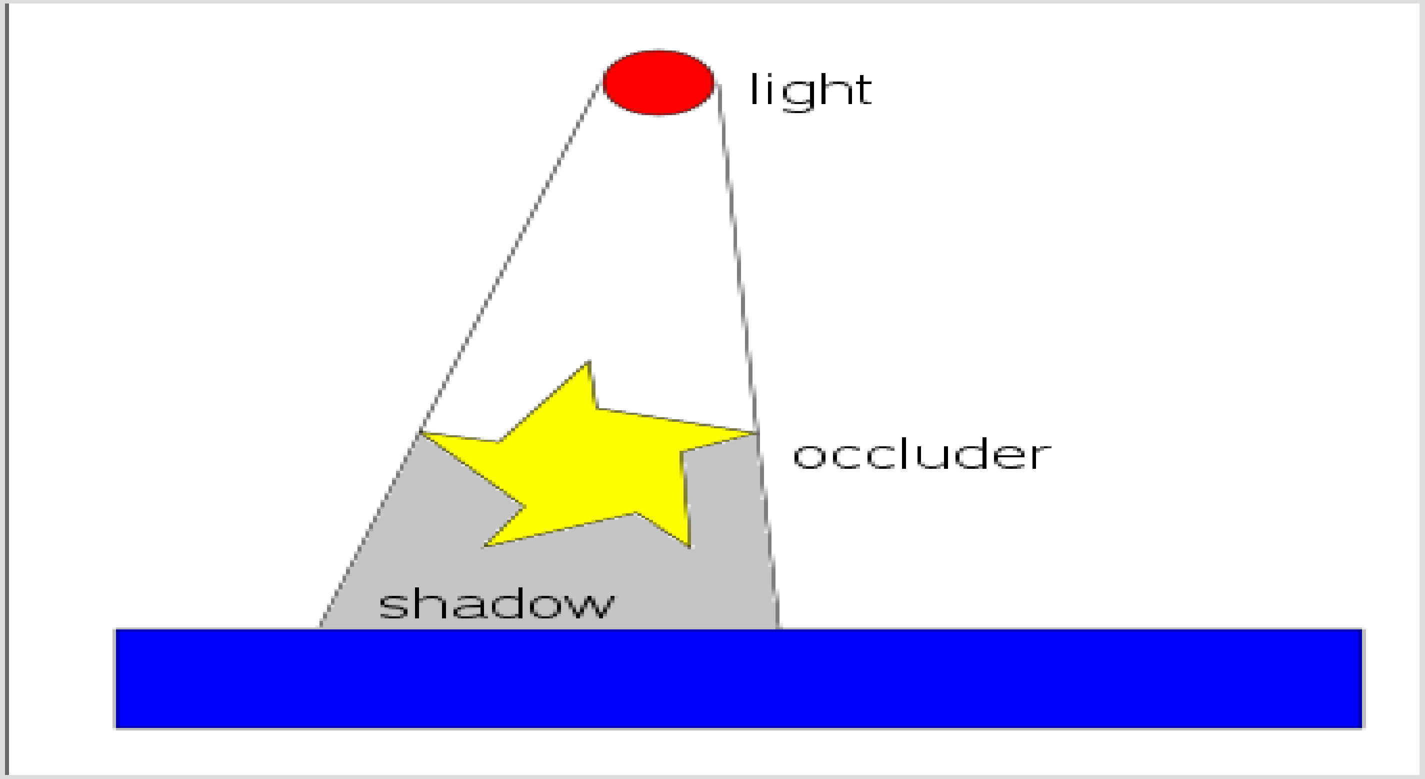


Figure: Screenshot from id's Doom3

Stencil Shadow Volumes introduction

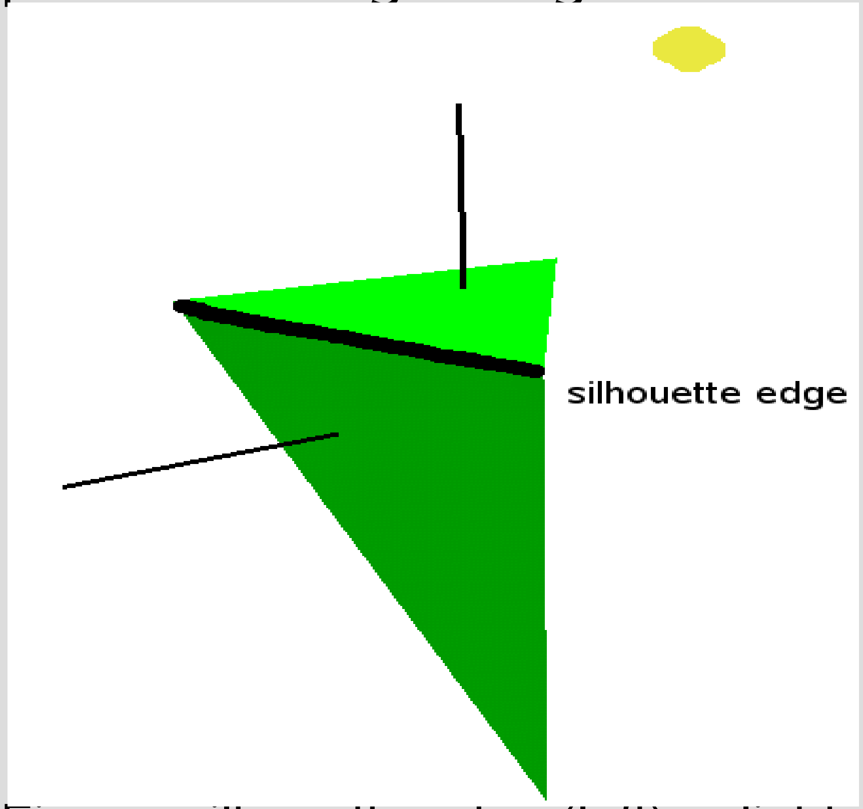
- Frank Crow introduced his approach using shadow volumes in 1977.
- Tim Heidmann of Silicon Graphics was the first one who implemented Crow´s idea using the stencil buffer.

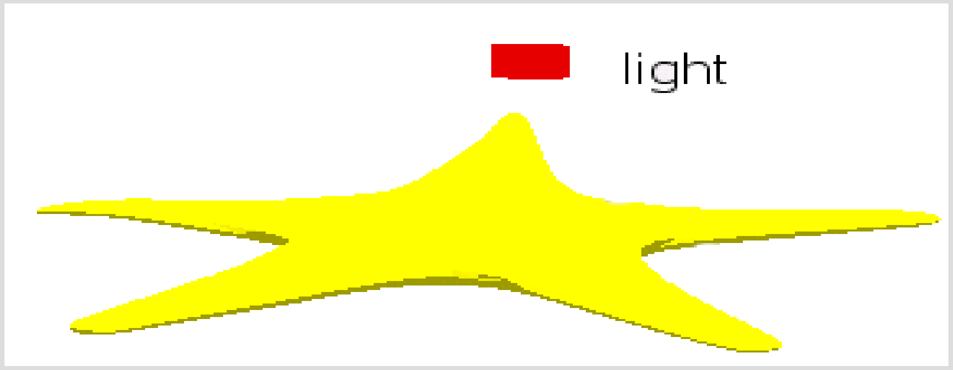


Stencil Shadow Volumes

Calculating Shadow Volumes With The CPU

- Calculate the silhouette:
- An edge between two planes is a member of the silhouette, if one plane is facing the light and the other is turned away.





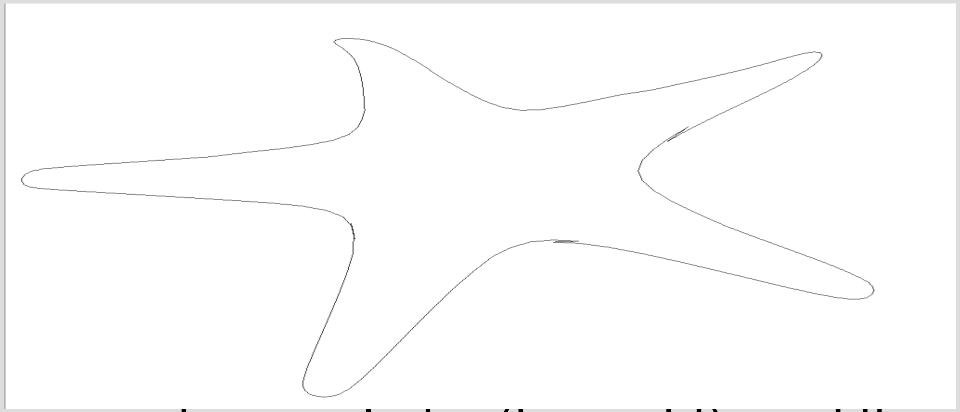
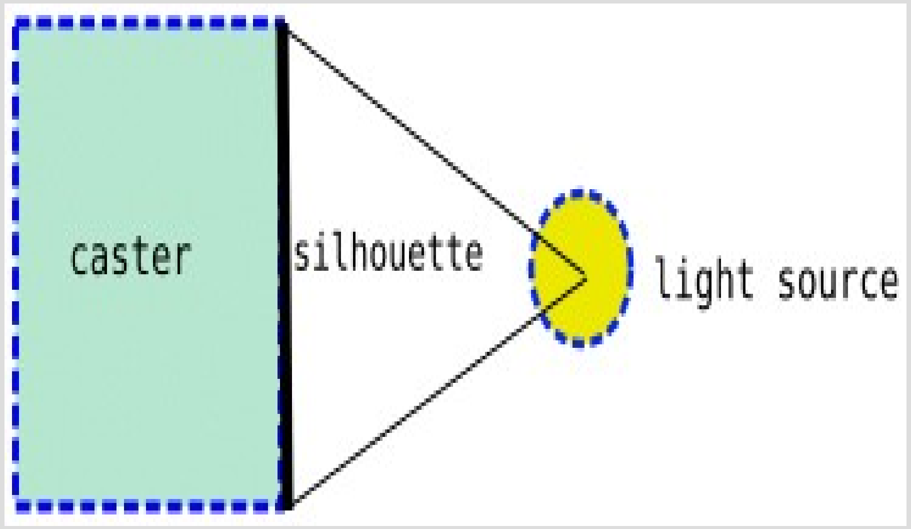


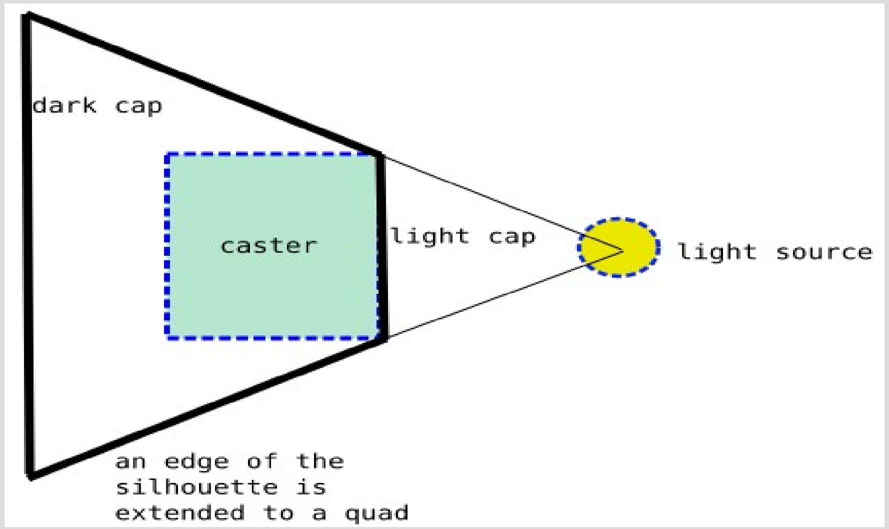
Figure: silhouette edge (left), a light source and an occluder (top right), and the silhouette of the occluder (down right)

Stencil Shadow Volumes

Calculating Shadow Volumes With The CPU

- The shadow volume should be closed
- By extruding the silhouette, the side surfaces of the shadow volume are generated
- The light cap is formed by the planes facing the light
- The dark cap is formed by the planes not facing the light
  - which are transferred into infinity.





Stencil Shadow Volumes

Calculating Shadow Volumes With The GPU

- Vertex shaders are specialized for transforming vertices
- ... however, they can not create new vertices
- That is why the silhouette can not be extruded
- Idea:
  - One-time enhancement of the object by additional vertices
  - displacement of vertices which are turned away from the light source

Stencil Shadow Volumes
Calculating Shadow Volumes With The GPU

- Replace all edges by degenerate rectangles.

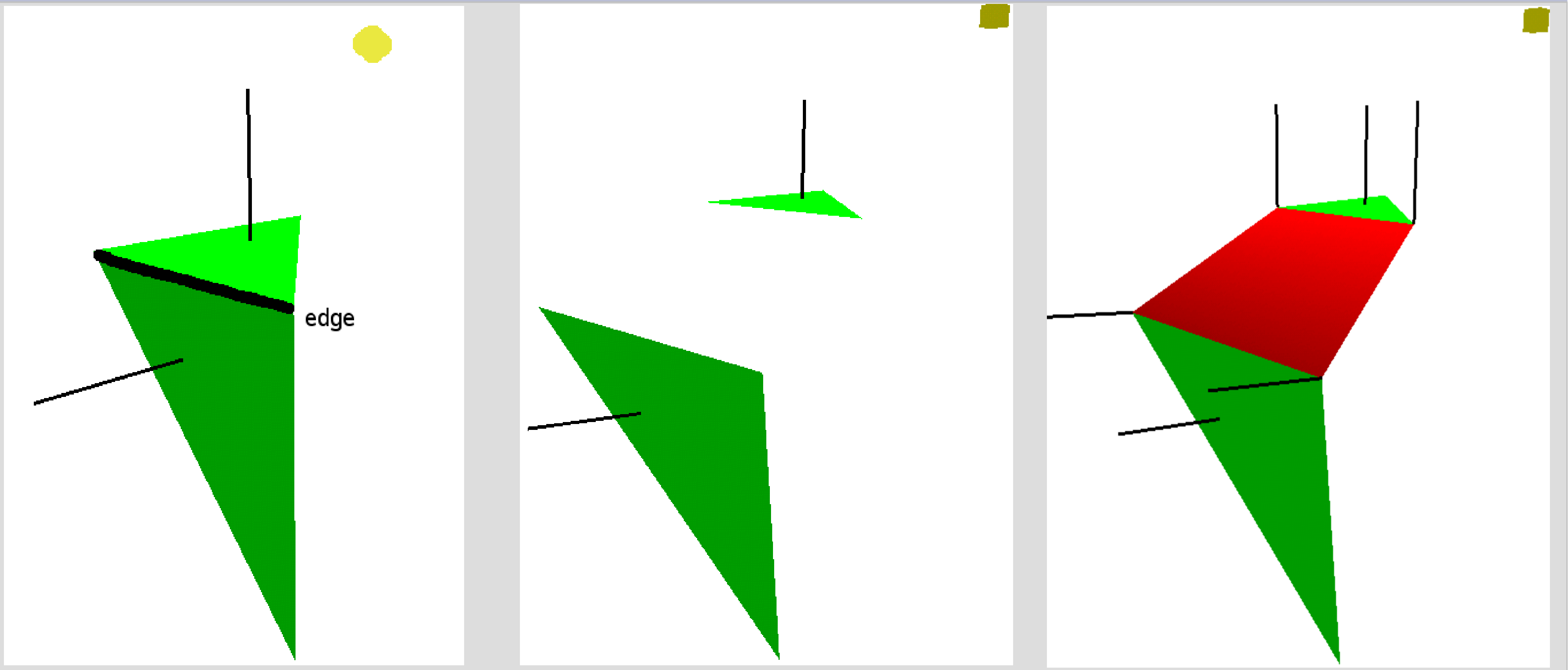
- They are degenerate because they have no surface area
- The new rectangles consist of new vertices each with
  - a position
  - a normal, the same as the vertex of the adjacent plane

- The shadow volume is build by testing each vertex if it is facing the light.

- If it isn't facing the light, then it has to be extruded into infinity in the direction of the vector from the light to the vertex.

Stencil Shadow Volumes

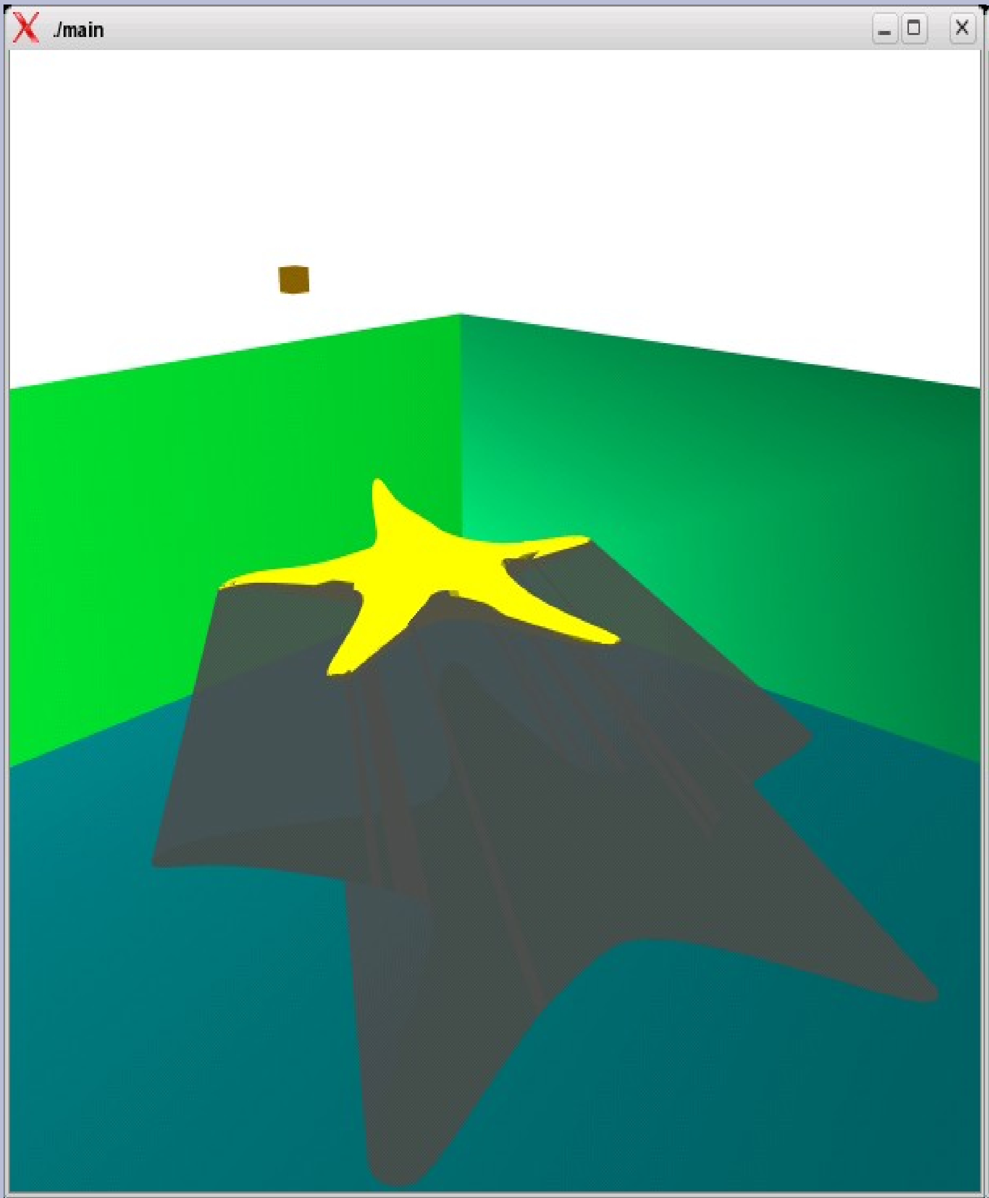
Calculating Shadow Volumes With The GPU

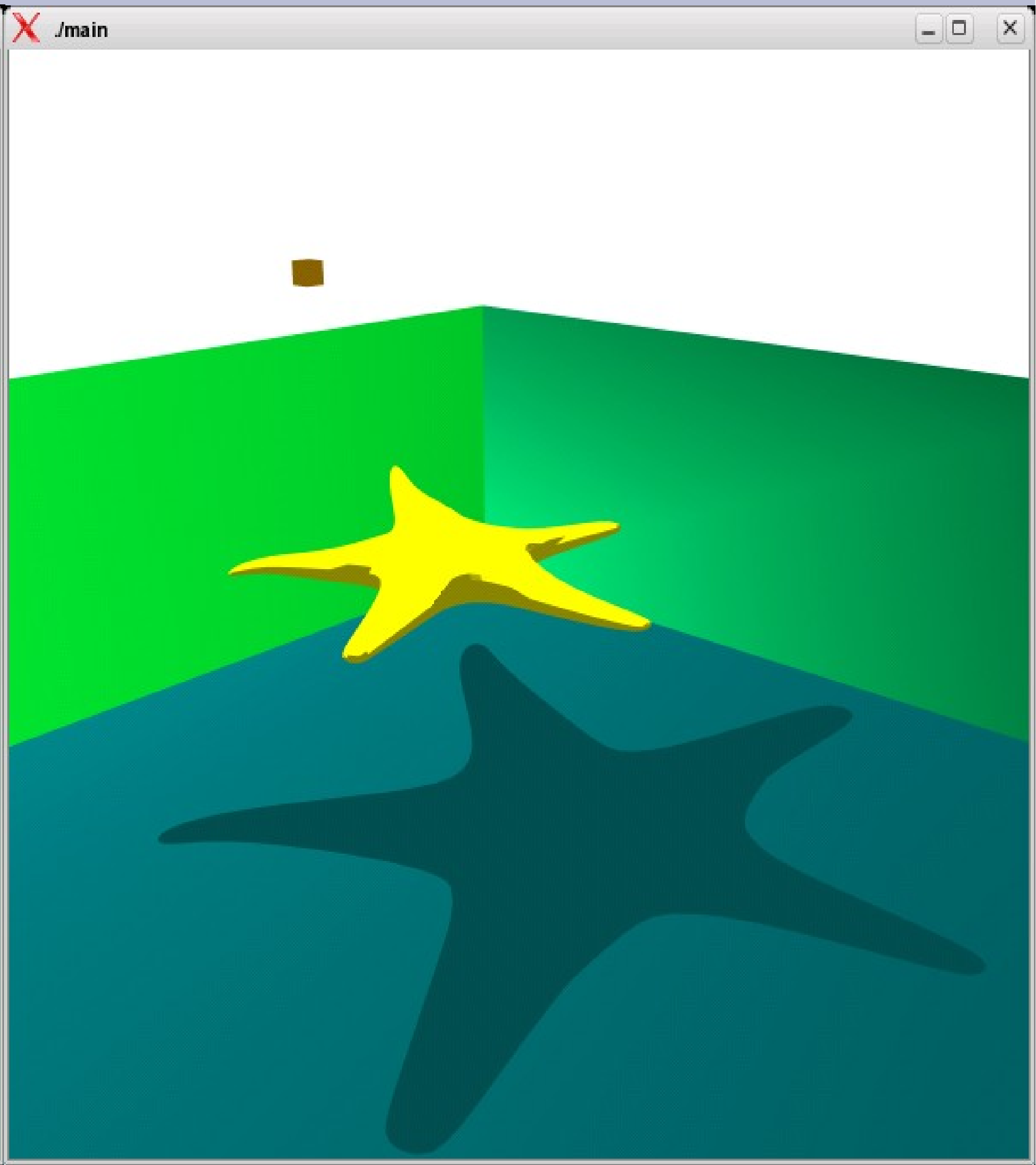


- All edges are replaced by degenerated rectangles
- By moving all vertices which don't face the light, the degenerated rectangles become “real” rectangles forming the side surfaces of the shadow volume.

Stencil Shadow Volumes

Using the shadow volumes to calculate the real shadows





Stencil Shadow Volumes

The Stencil Buffer

- Used like a real stencil to mask a scene.
- Examples:
  - Cockpit (mask only the windows to draw the landscape)
  - Reflections (mask the area where to draw the same scene mirrored)
  - Shadows ...

Stencil Shadow Volumes

The Stencil Buffer

- Can draw geometry in it.
  - Doing this will update the values in the stencil buffer.
- Can look up the value in the stencil buffer.
- Can use a comparison function (among others <, >, =, !=) to then
  - update the value in the stencil buffer
  - draw the fragment
  - discard the fragment

Stencil Shadow Volumes
The Stencil buffer

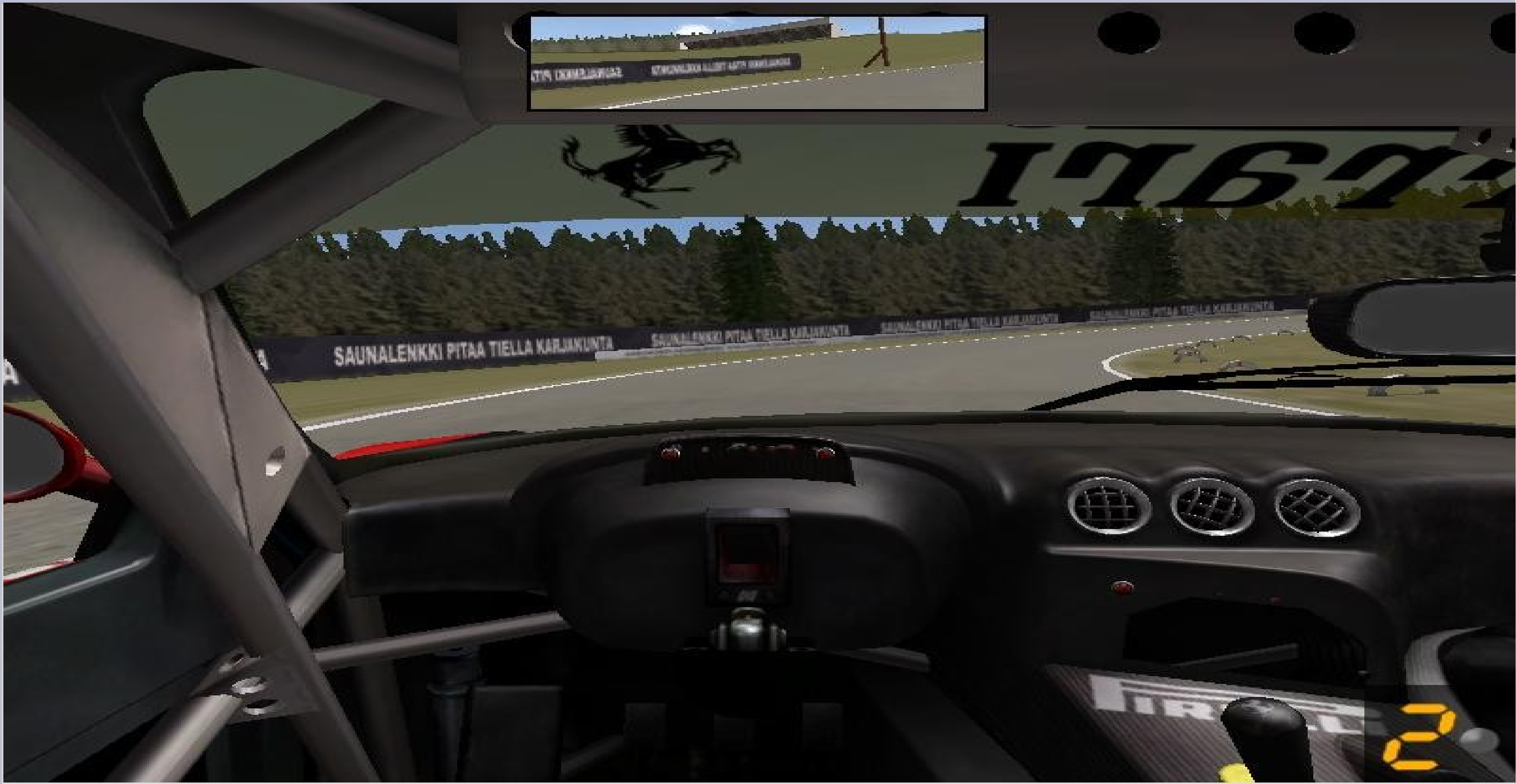

SAUNALENKKI PITAA TIELLA KARJAKUNTA
SAUNALENKKI PITAA TIELLA KARJAKUNTA
SAUNALENKKI PITAA TIELLA KARJAKUNTA
SAUNALENKKI PITAA TIELLA KARJAKUNTA

Figure: Screenshot from SimBins GTR2. The mirrors and windows are masked areas.

Stencil Shadow Volumes
Depth-Pass vs. Depth-Fail

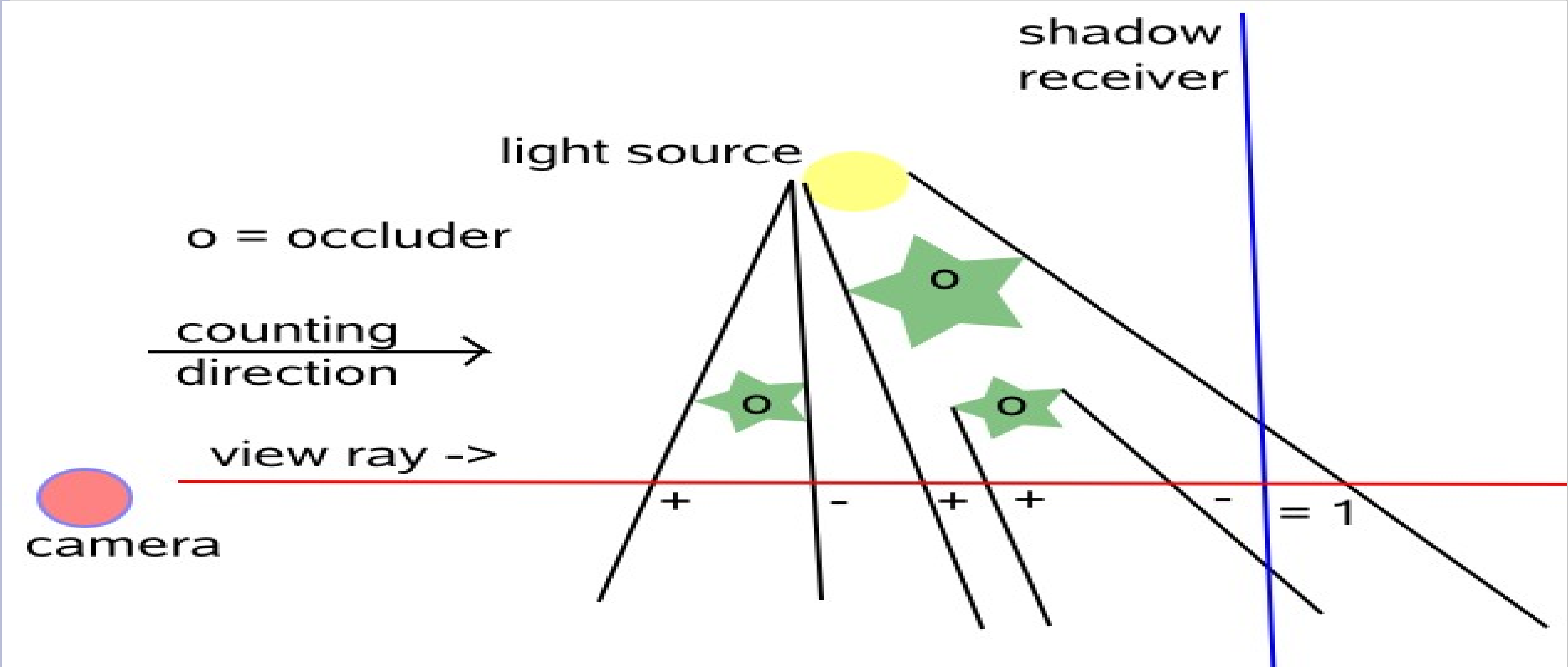
- To decide if a fragment lies in a shadow, two methods exist:

  - 1. Depth-pass = z-pass
  - 2. Depth-fail = Carmack´s Reverse = z-fail

    Both methods calculate a value for each fragment in the stencil buffer

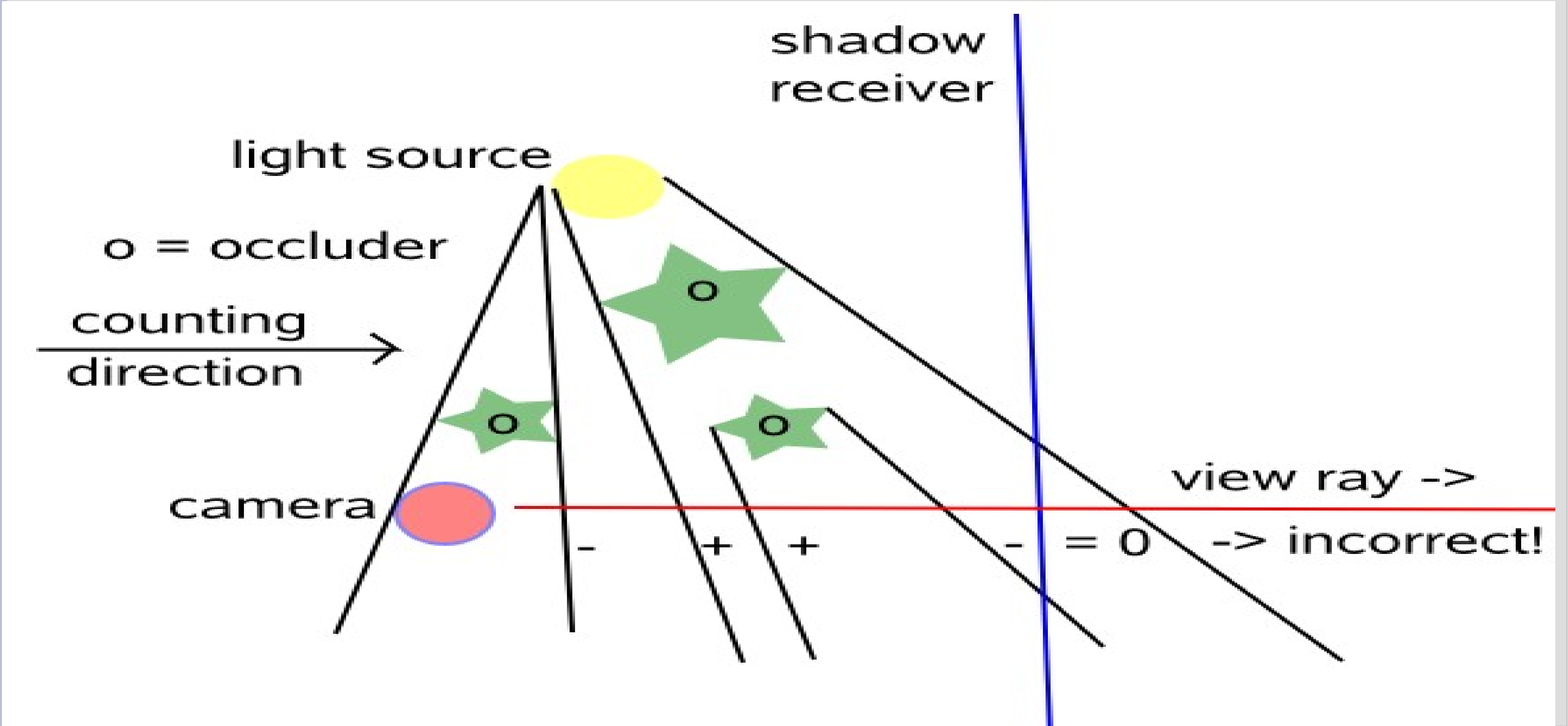
Stencil Shadow Volumes
Depth-Pass

1. Disable writing to depth and color buffers
2. Render front face of shadow volume. If the depth test passes, increment the stencil value.
3. Render back face of shadow volume. If the depth test passes, decrement the stencil value.



Stencil Shadow Volumes
Depth-Pass

If the camera is located inside of a shadow volume, the entry in this volume is missed and the stencil value isn't incremented.
This causes incorrect shadows.



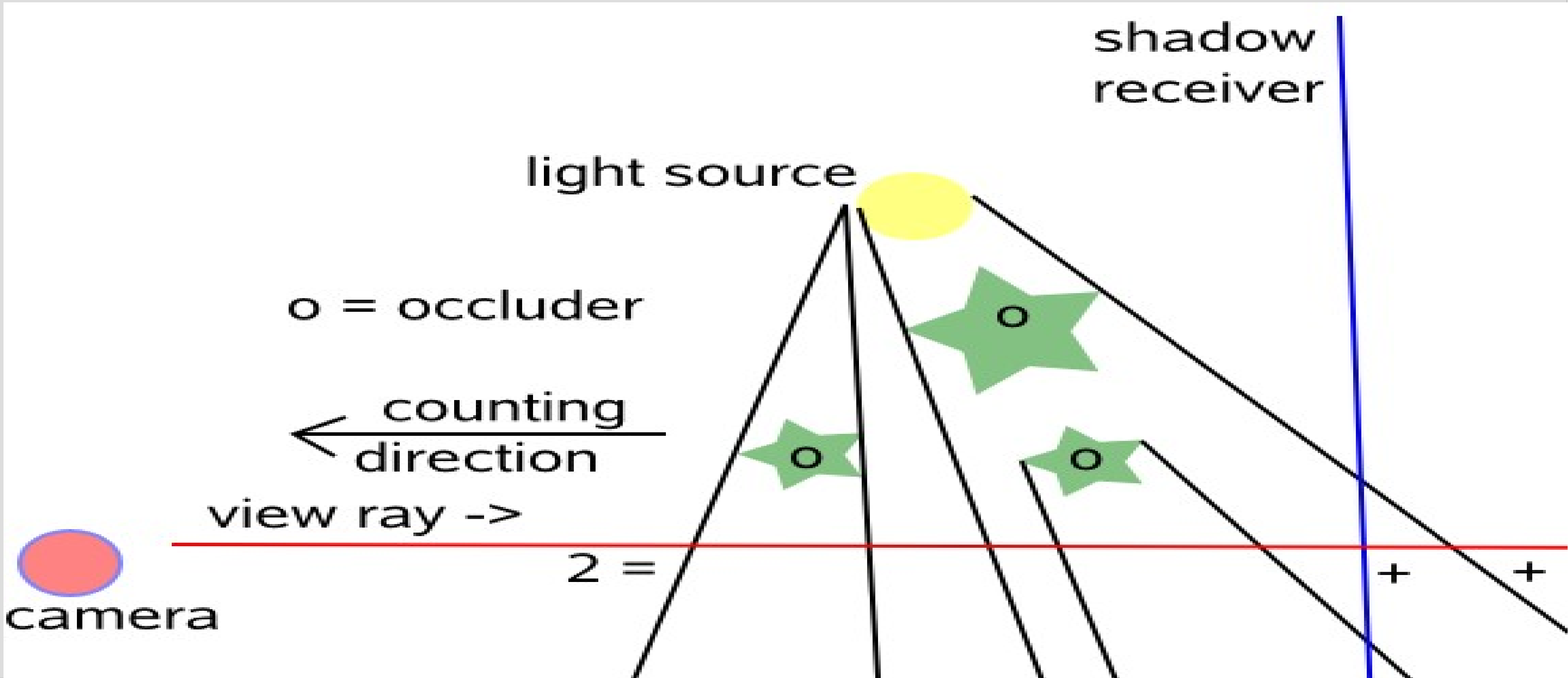
Stencil Shadow Volumes

Depth-Pass - OpenGL Stencil Configuration

```
// disable writing to the color- and depth buffer:
glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE);
glDepthMask(GL_FALSE);


// Configuration of the stencil function and the stencil operator:
//1) first stencil test – increment if drawing of the front faces is successful
glEnable(GL_STENCIL_TEST);
glStencilFunc(  GL_ALWAYS, 0, 0xffffffff);
glStencilOp( GL_KEEP, GL_KEEP, GL_INCR);
glEnable(GL_CULL_FACE);
glFrontFace(GL_CCW);
drawShadowVolume(); // at first only front faces


// 3) second test – decrement if drawing of the back faces is successful
glStencilOp(GL_KEEP, GL_KEEP, GL_DECR);
glFrontFace(GL_CW);
drawShadowVolume(); // now the back faces
```

Stencil Shadow Volumes
Depth-Fail / Carmack's Reverse

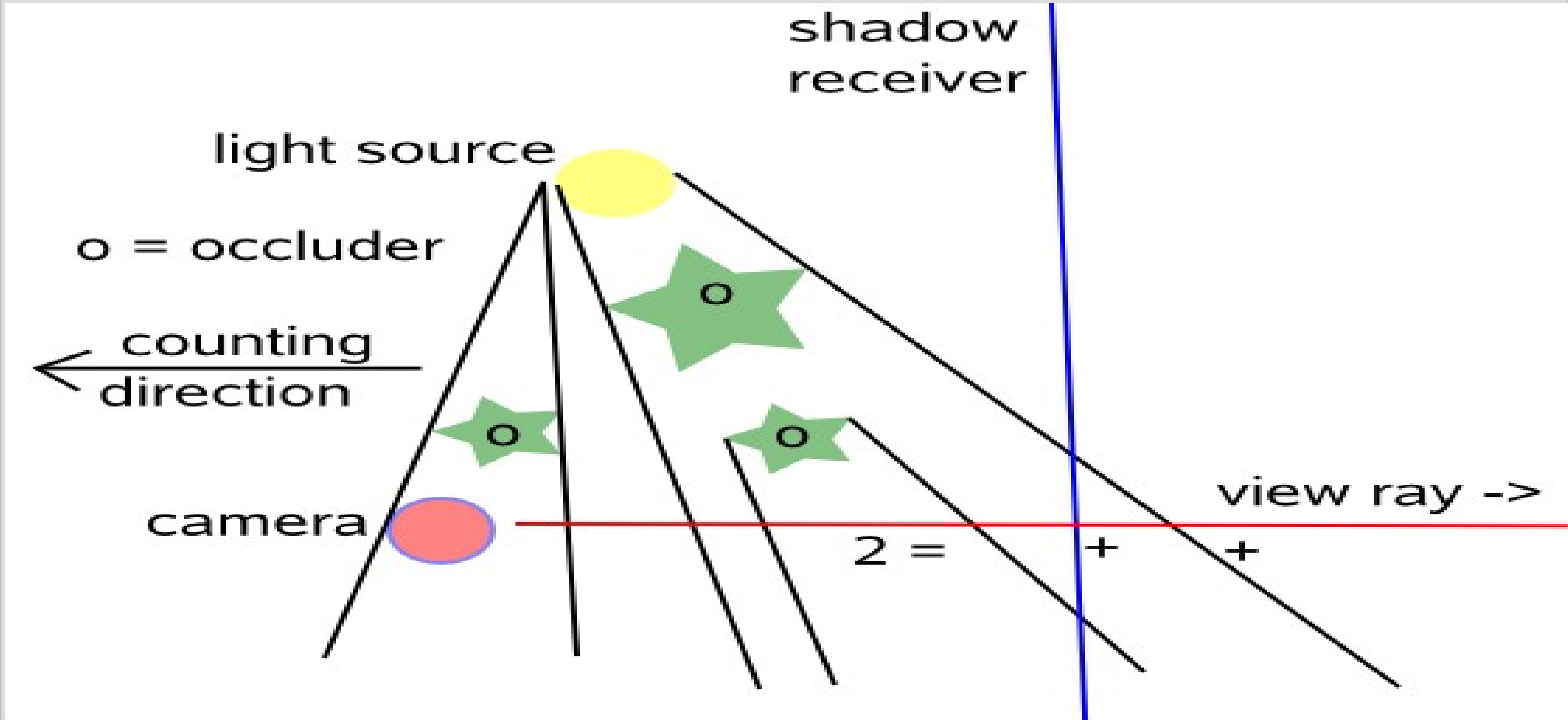
1. Disable writing to depth and color buffers
2. Render back face of shadow volume. If the depth test fails, increment the stencil value.
3. Render front face of shadow volume. If the depth test fails, decrement the stencil value.



Stencil Shadow Volumes

Depth-Fail

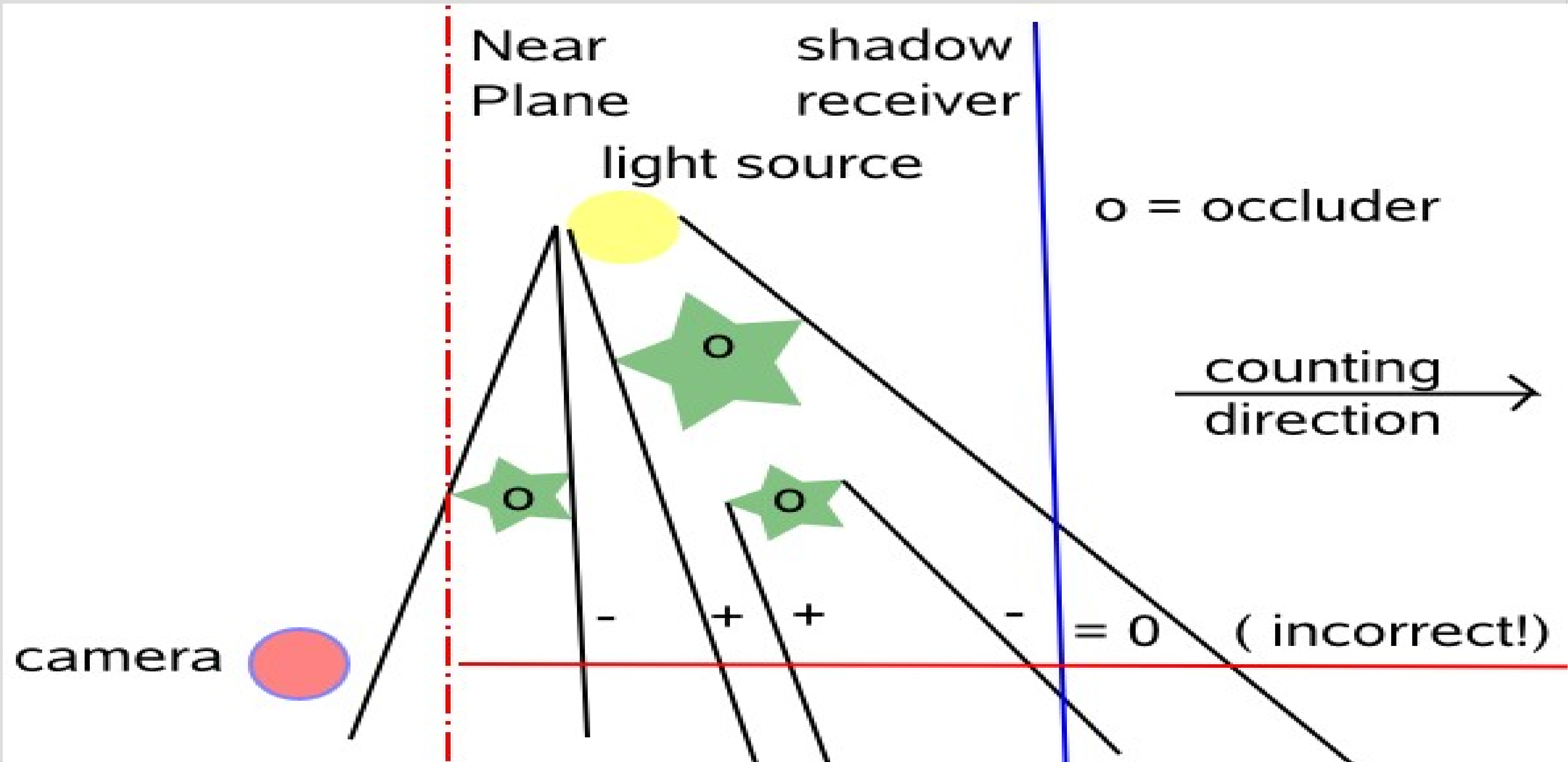
- If the camera is located inside of a shadow volume, the result is still correct!
- The more robust method! (capping is needed!)



Stencil Shadow Volumes

Near Clipping Plane Issues

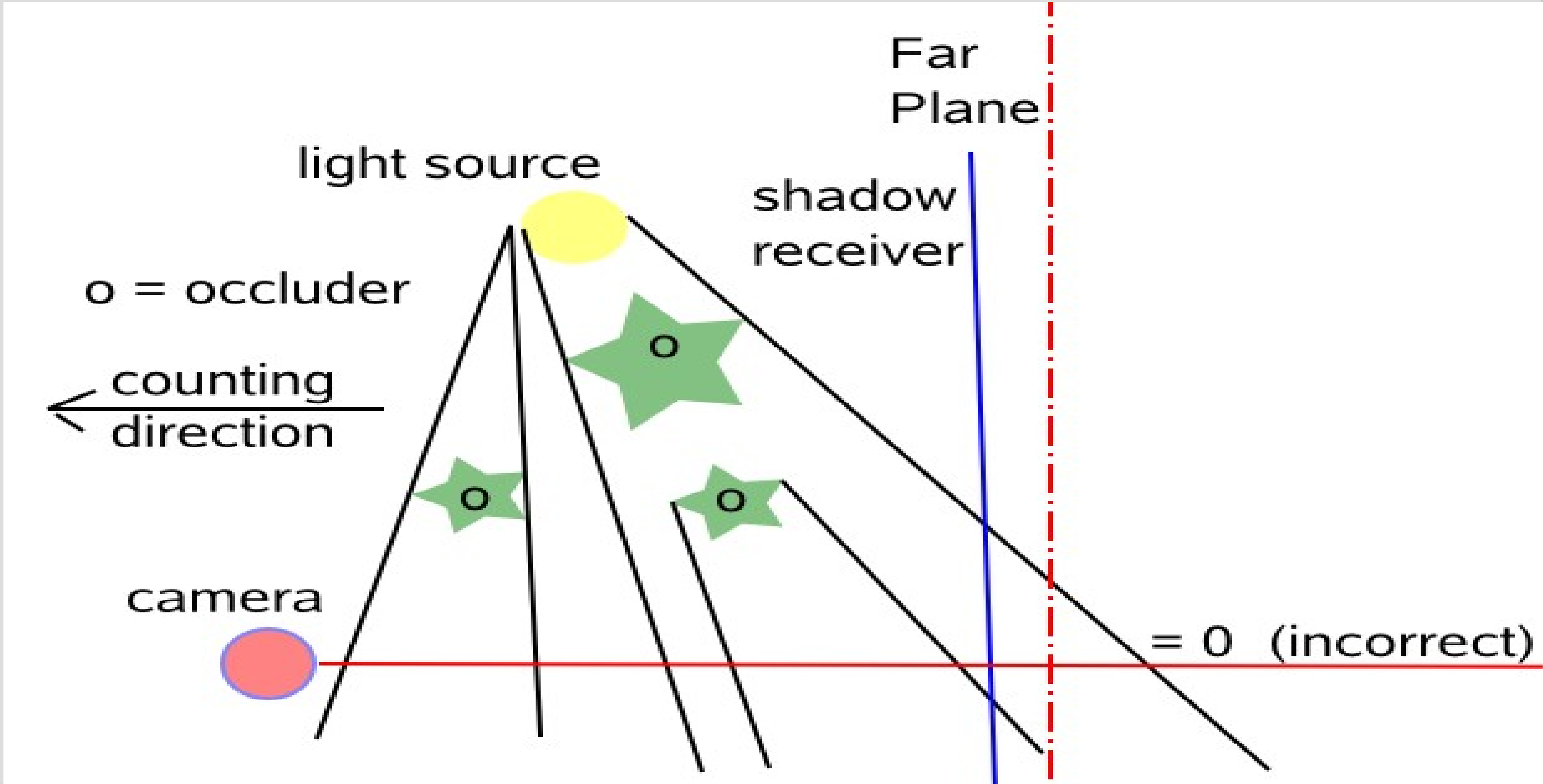
Both methods have problems if the shadow volume intersects a clipping plane.
Depth-pass doesn't function, if the volume is intersected by the near plane.



Stencil Shadow Volumes

Far Clipping Plane Issues

Depth-fail doesn't function, if the volume is intersected by the far plane.
Moving the far plane to infinity is the solution to the problem.



Stencil Shadow Volumes

Depth-Fail

- Three approaches for moving the far clipping plane to infinity:
  - 1. Direct manipulation of the projection matrix
  - 2. Defining the kind of projection using a negative value for the distance to the far clipping plane
  - 3. Using the extension NV_depth_clamp by nVidia

Stencil Shadow Volumes

Depth-Pass vs. Depth-Fail

- Advantages:
- front & back caps are unnecessary
- less geometry
- speedier than depth-fail
- easier to implement
- Disadvantages:
- not applicable if the camera is located inside of a shadow volume
  - near plane clipping issues
  - not really robust

- Advantages:
- also applicable if the camera is located inside of a shadow volume
- More robust because the „far plane clipping issue“ can be eliminated:
  - by translating the near-clipping-plane to infinity
  - the depth-clamping-extension (nVidia) assigns all geometry behind the far clipping plane with the maximal z-value
- Disadvantages:
- front & back caps necessary
- more geometry to render
- slower
- harder to implement

Shadow Mapping

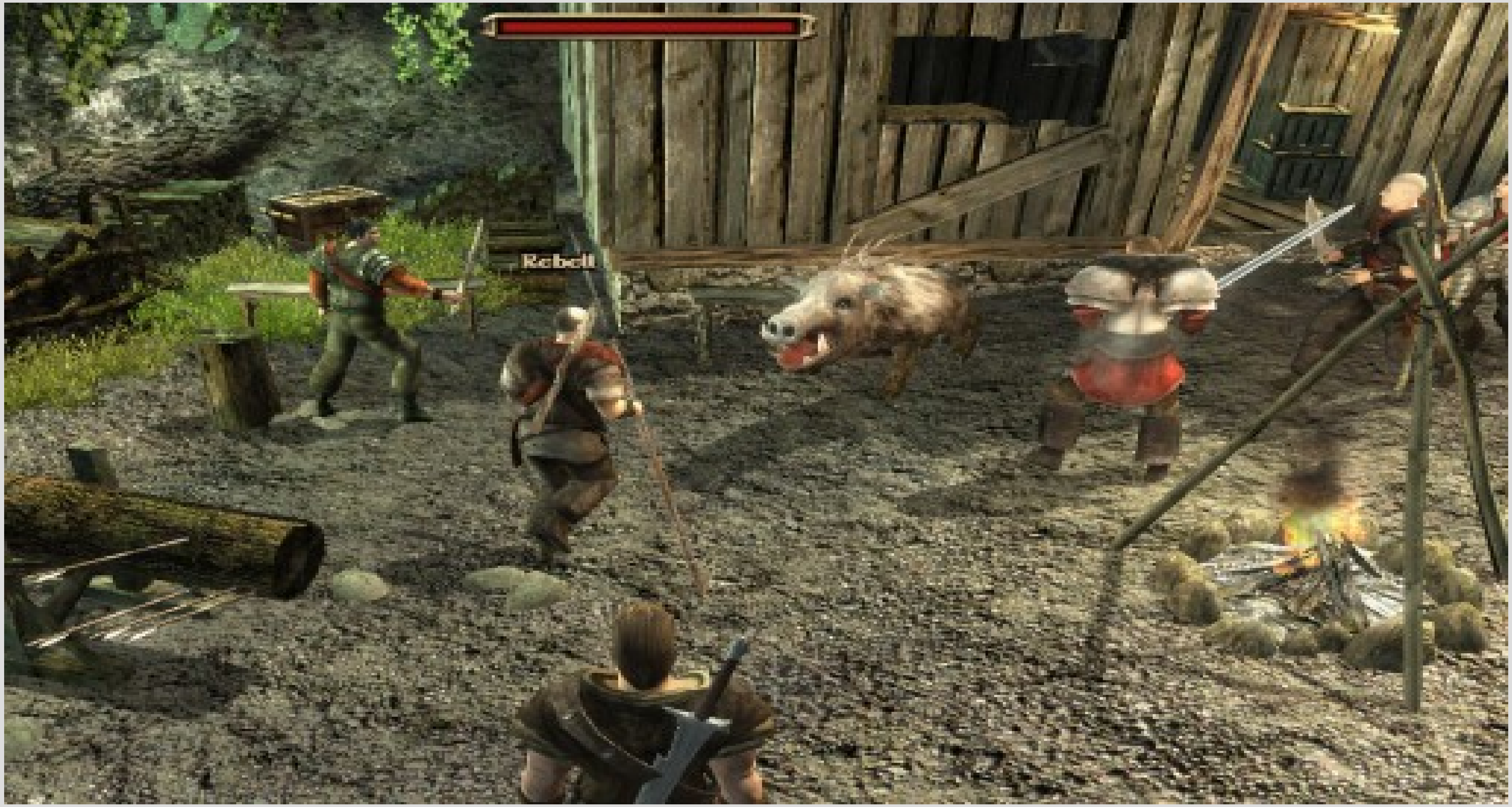
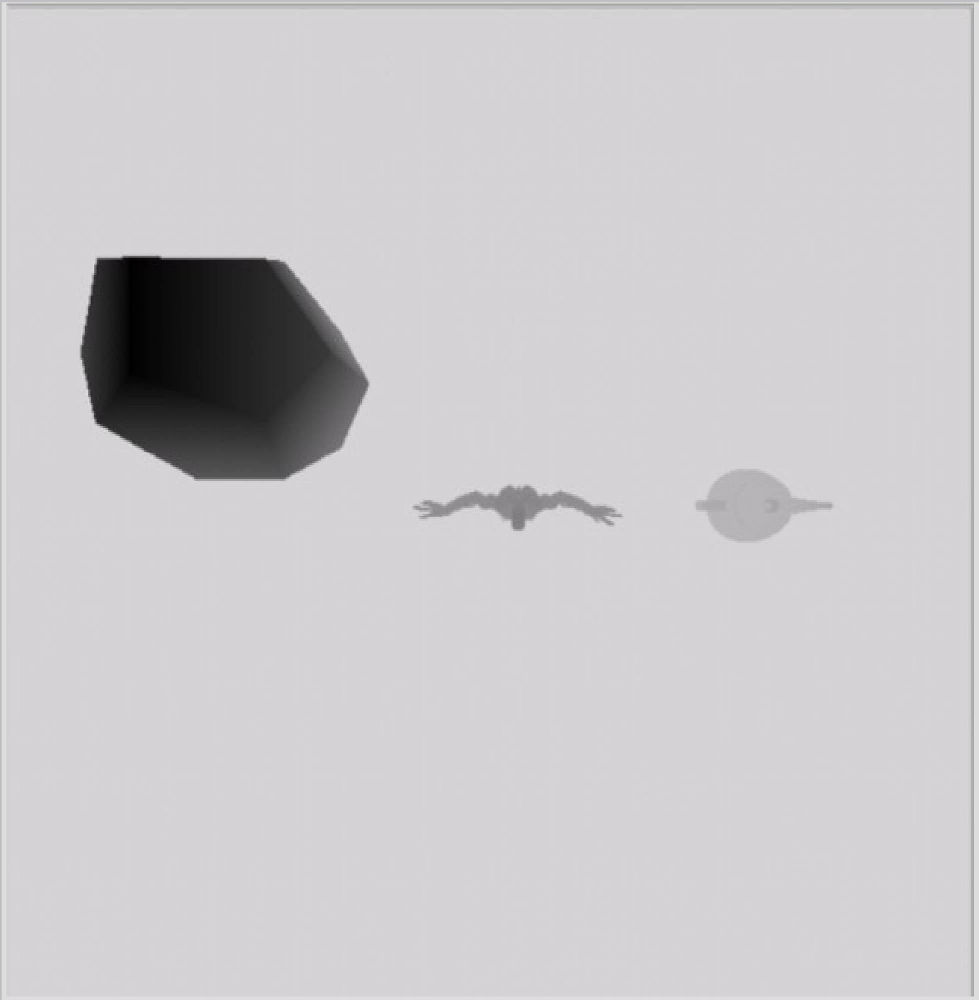


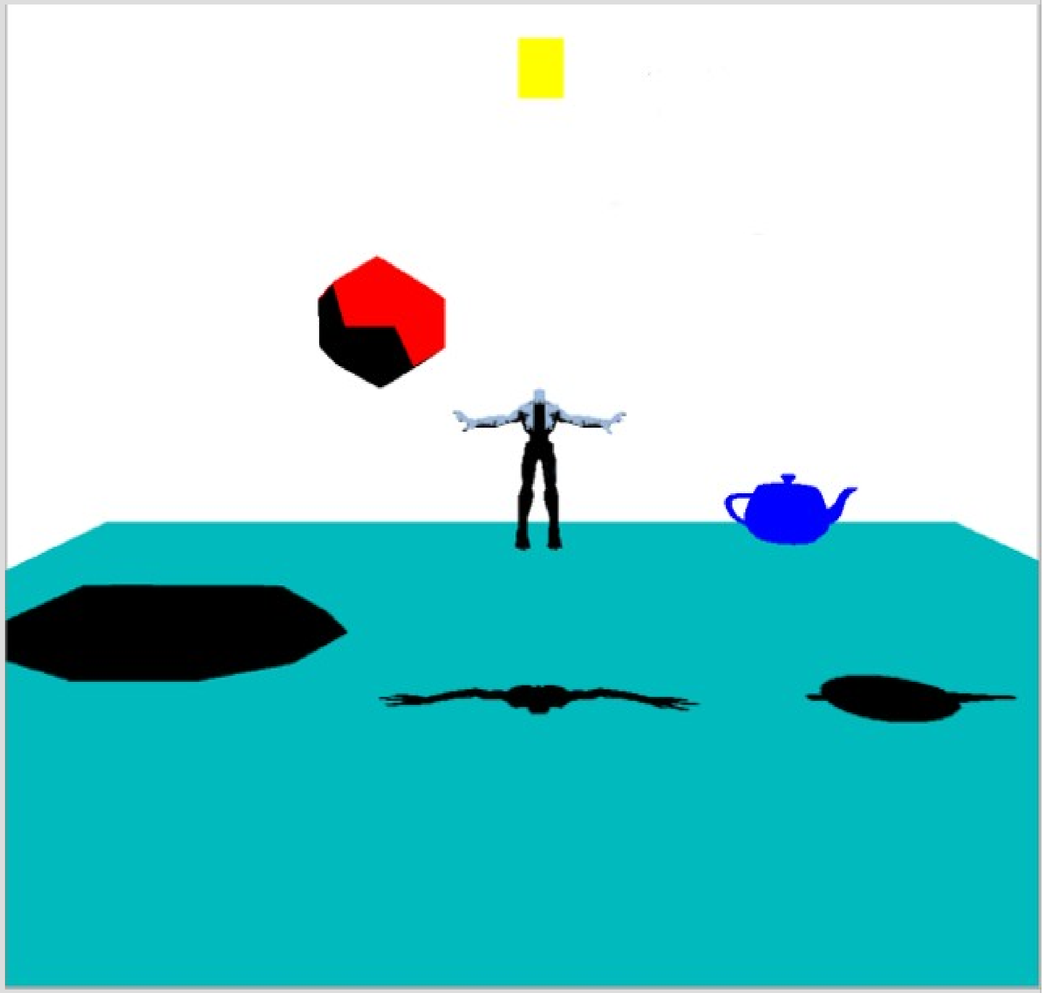
Figure: Screenshot of Gothic III

Shadow Mapping
Concept

- Concept by Lance Williams 1978
  - 1. Draw scene from light's viewpoint
    - Store the depth values in a shadow texture
  - 2. Draw scene from camera's viewpoint
    - Determine each fragment's position relative to the light
    - Compare the transformed depth value to the value in the depth map at position XY.
    - If the value is greater than the value of the depth map
      - then the fragment is covered from the light´s perspective
      - and therefore must be shadowed.

Shadow Mapping
Render-to-Texture





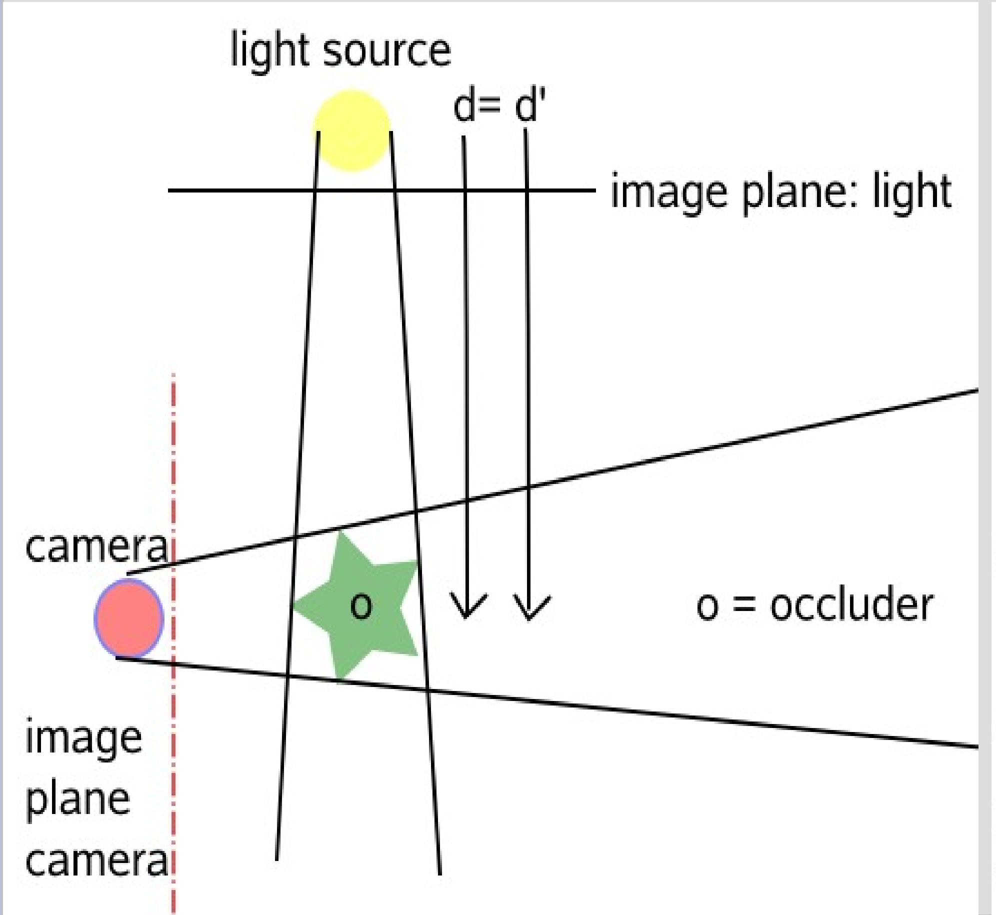
The depth values from light's viewpoint interpreted as b/w colors and the same scene from camera's viewpoint with shadows

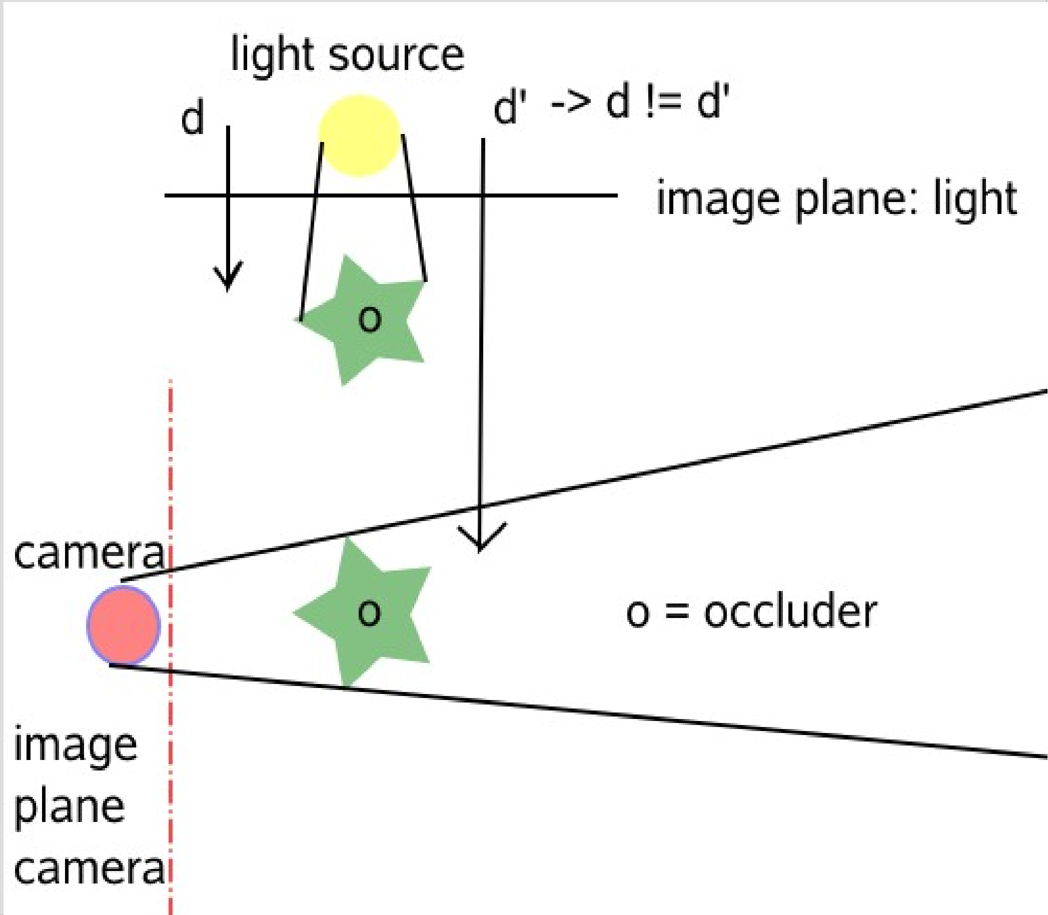
Shadow Mapping Render-to-Texture: OpenGL Texture Configuration:

```
// Create and bind a 2D texture:
glGenTextures(1, &shadowMapTexture);
glBindTexture(GL_TEXTURE_2D, shadowMapTexture);

// configure the texture: ( dimension, and type of content)
 glTexImage2D( GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT32, 500, 500, 0,
   GL_DEPTH_COMPONENT32, GL_FLOAT, NULL);

// copy values from the depth buffer ( GL_DEPTH_COMPONENT) to the texture:
glCopyTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT, 0, 0,
   frameSizeWidth, frameSizeHeight, 0);

// alternatively copy rgb-values from the color buffer ( GL_RGBA) to the texture:
glCopyTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 0, 0, frameSizeWidth,
   frameSizeHeight, 0);
```

Shadow Mapping
Depth Comparison

- Draw scene from camera´s viewpoint
  - Determine each fragments position relative to the light (d')
  - Compare the transformed depth value to the value in the depth map at position XY (d).





Shadow Mapping

Configuration of the depth comparison in OpenGL using the CPU

```
// Bind the shadow map as the current texture and activate texturing:
glBindTexture(GL_TEXTURE_2D, shadowMapTexture);
glEnable(GL_TEXTURE_2D);

//Enable shadow comparison
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_COMPARE_MODE_
   A     RB, GL_COMPARE_R_TO_TEXTURE);

//Shadow comparison should be true if r<=texture
glTexParameteri(GL_TEXTURE_2D,
   GL_TEXTURE_COMPARE_FUNC_ARB, GL_LEQUAL);

//Shadow comparison should generate an INTENSITY result
glTexParameteri(GL_TEXTURE_2D,
   GL_DEPTH_TEXTURE_MODE_ARB, GL_INTENSITY);
```

Shadow Mapping

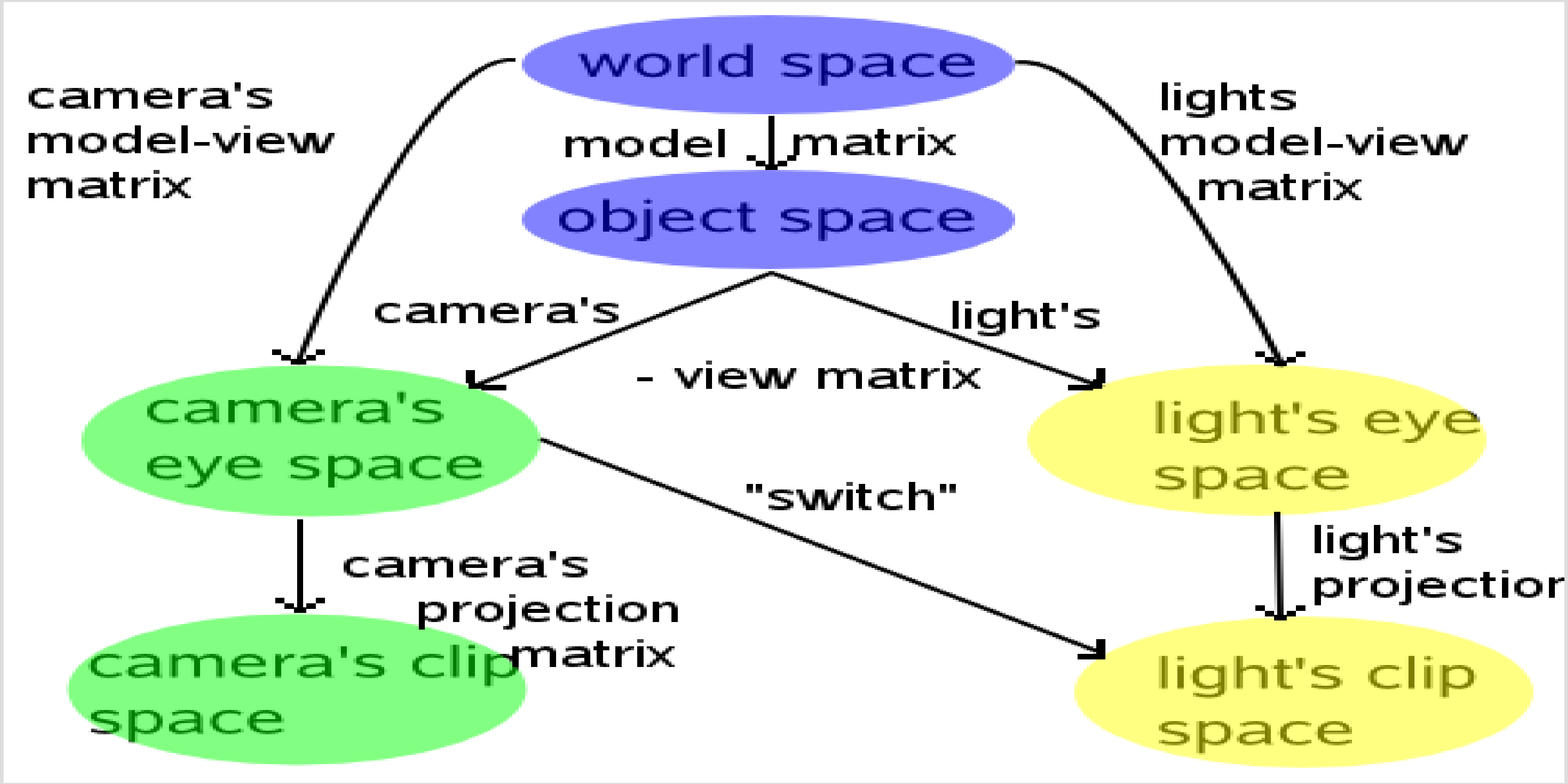
Configuration of the depth comparison in OpenGL using the GPU

Performing the depth comparison in the fragment shader:

```
uniform sampler2DShadow shadowMap;
uniform float threshold;
varying vec4 projCoord;

void main ()
{
   // performing the depth comparison:
   float compResult = shadow2DProj(shadowMap, projCoord).z;

   // if both values are equal: compResult = 1.0
   // otherwise compResult = 0.0;

   // we can use the compResult to darken the color of the fragment
   gl_FragColor = gl_Color * compResult;
}
```

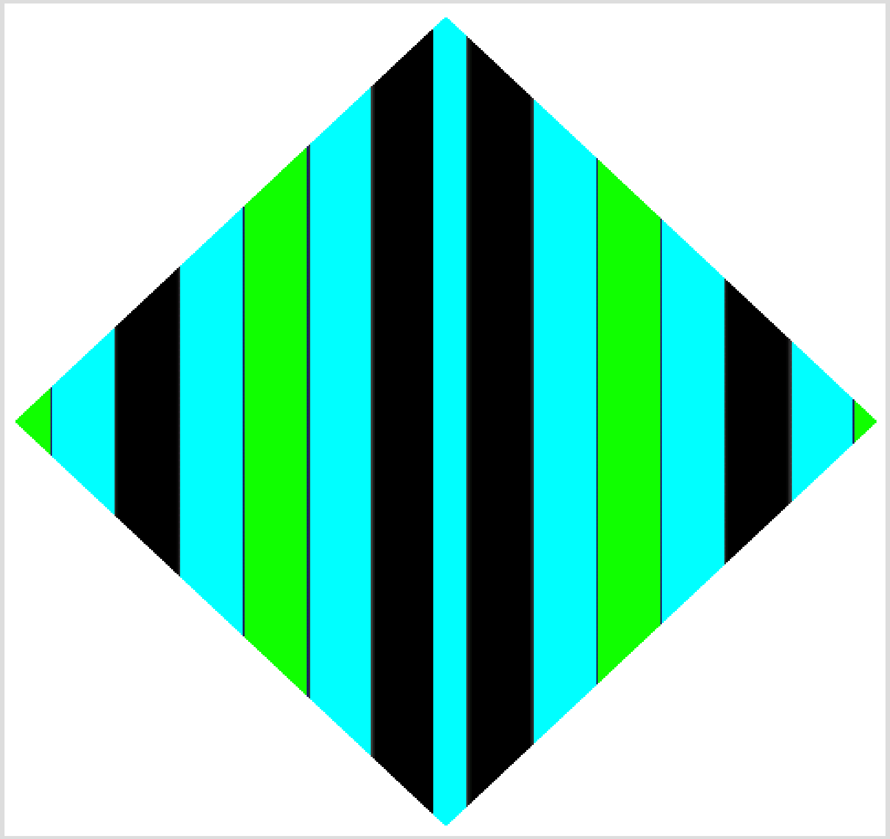
Shadow Mapping

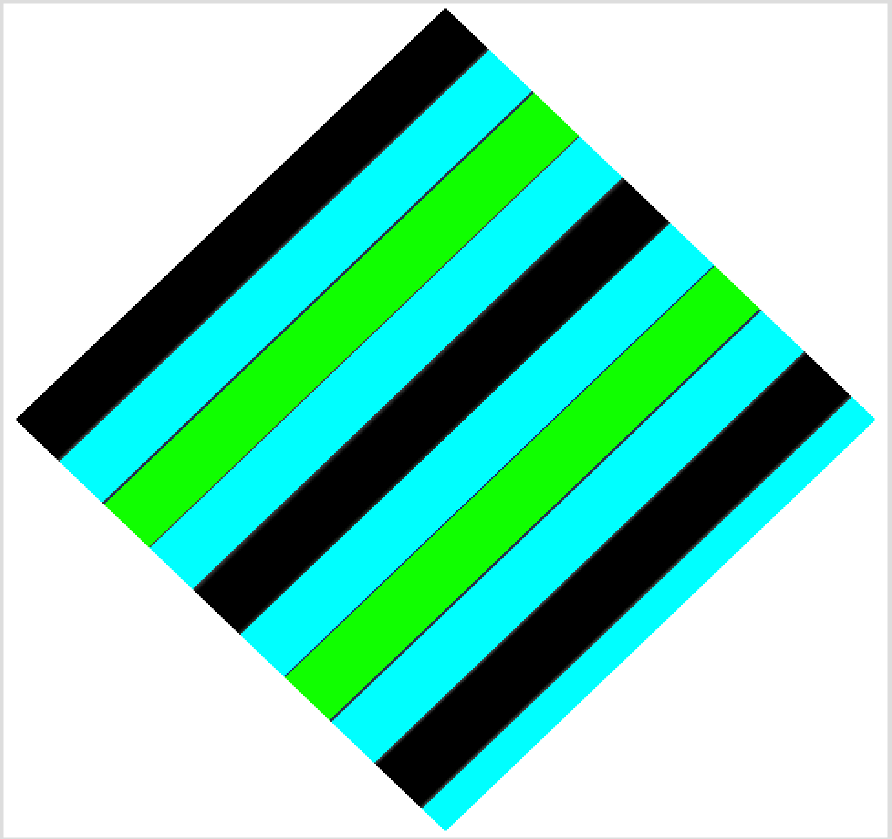
Switching Between The Coordinate Systems



Determine the position of the fragment relative to the light source means transforming the eye-coordinates into clip-coordinates of the light source (“switch”).

Shadow Mapping
Generation Of Texture-Coordinates


relative to the display borders


relative to the object borders

Shadow Mapping

Generation of texture coordinates in OpenGL

```
//Calculate texture matrix for projection
//This matrix takes us from eye space to the light's clip space
//It is postmultiplied by the inverse of the current view matrix when specifying texgen

static MATRIX4X4 biasMatrix(0.5f, 0.0f, 0.0f, 0.0f, 0.0f, 0.5f, 0.0f, 0.0f, 0.0f, 0.0f, 0.5f, 0.0f,
0.5f, 0.5f, 0.5f, 1.0f); //bias from [-1, 1] to [0, 1]
MATRIX4X4 textureMatrix = biasMatrix * lightProjectionMatrix * lightViewMatrix;

//Set up texture coordinate generation.
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR);
glTexGenfv(GL_S, GL_EYE_PLANE, textureMatrix.GetRow(0));
glEnable(GL_TEXTURE_GEN_S);

glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR);
glTexGenfv(GL_T, GL_EYE_PLANE, textureMatrix.GetRow(1));
glEnable(GL_TEXTURE_GEN_T);

glTexGeni(GL_R, GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR);
glTexGenfv(GL_R, GL_EYE_PLANE, textureMatrix.GetRow(2));
glEnable(GL_TEXTURE_GEN_R);

glTexGeni(GL_Q, GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR);
glTexGenfv(GL_Q, GL_EYE_PLANE, textureMatrix.GetRow(3));
glEnable(GL_TEXTURE_GEN_Q);
```

Shadow Mapping
Polygon Offset

- With infinite resolution and unlimited precision pixels
  - transformed into light's coordinate system
  - & passing the visible test
  - have the same depth value as the one in the depth texture.
- But in real life, resolution and precision are limited
  - --> producing artefacts.
- We can avoid this by using a polygon offset before drawing to the shadow map.
- An appropriate polygon offset must be found for every individual scene.

Shadow Mapping

Polygon Offset

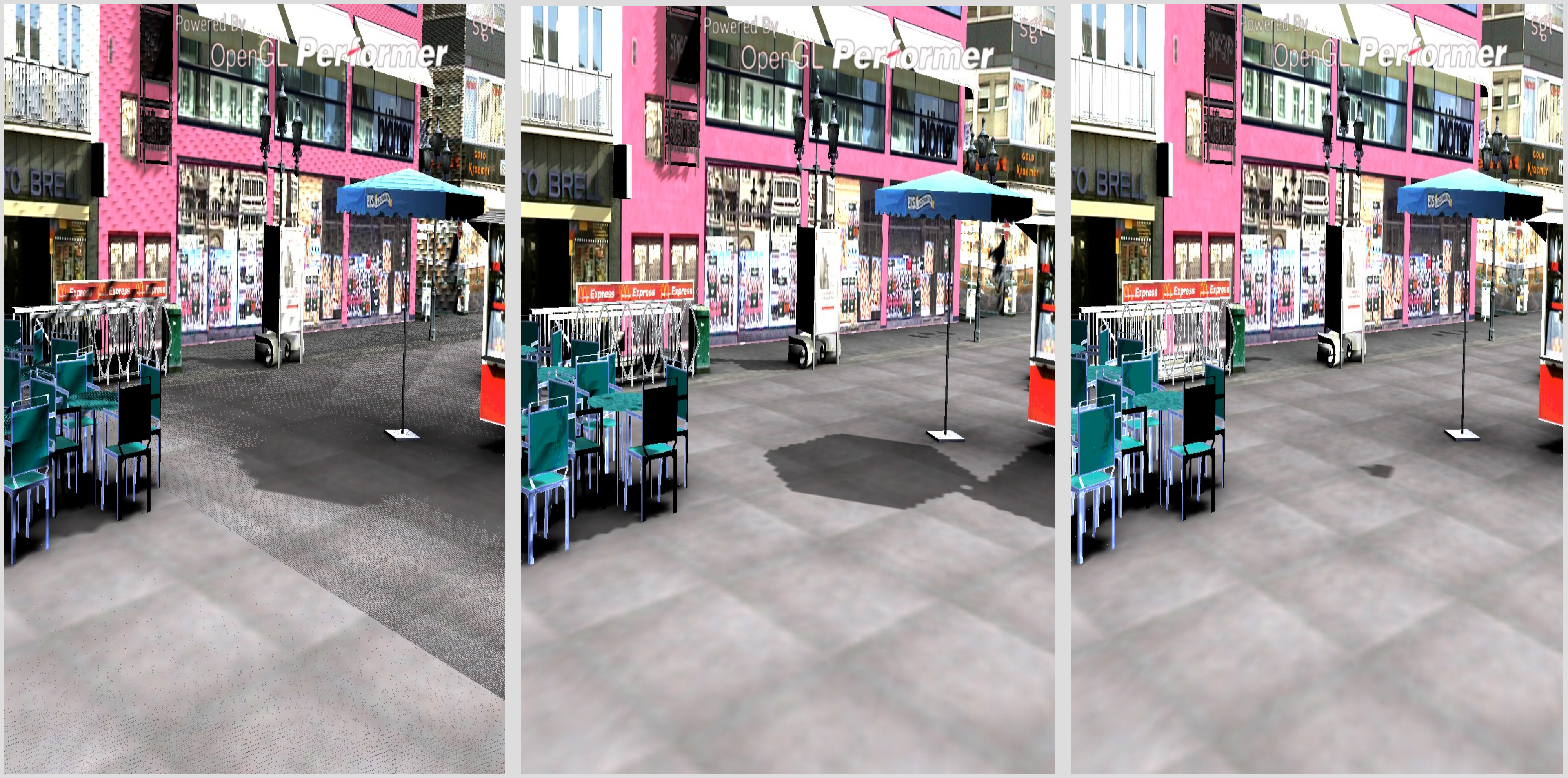


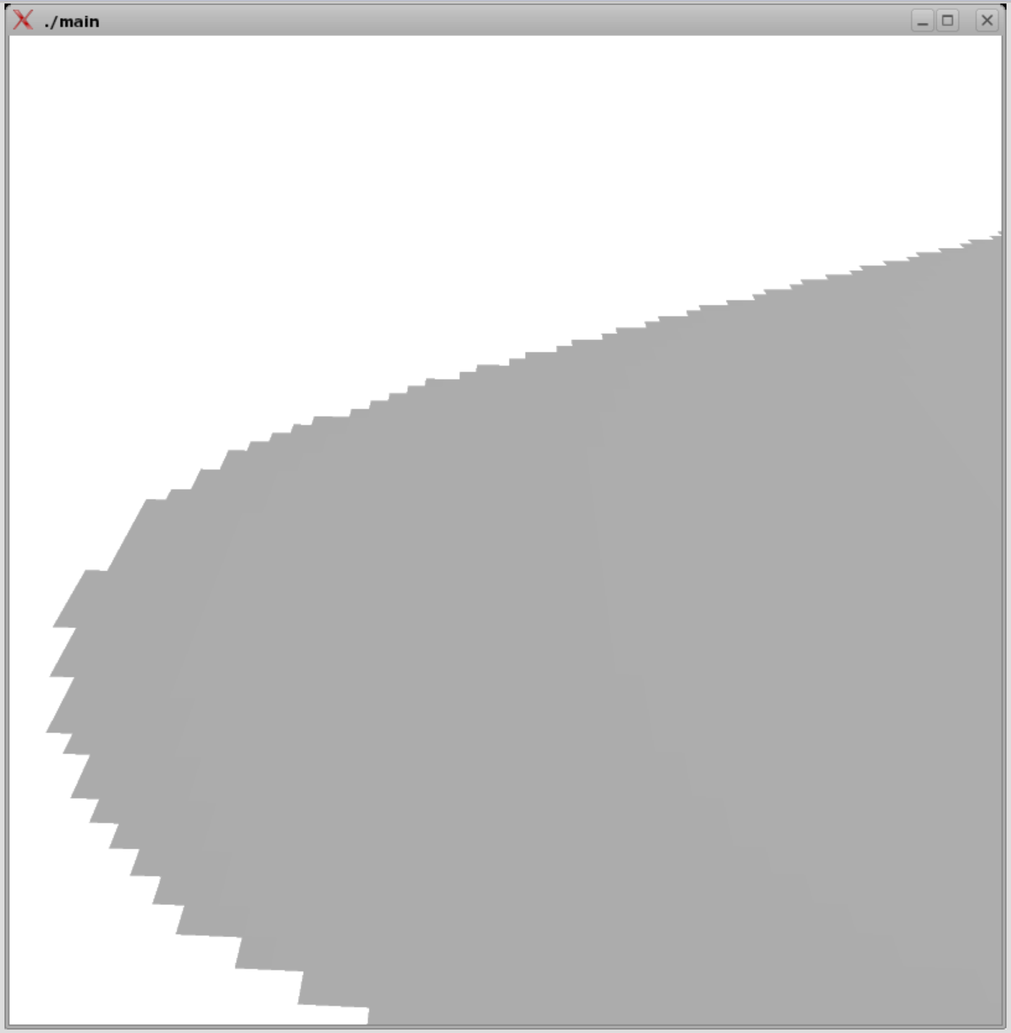
Figure: The same scene using three different polygon offsets. Polygon offset is too low (left), just right (middle), and too high(right).

Shadow Mapping
Polygon Offset

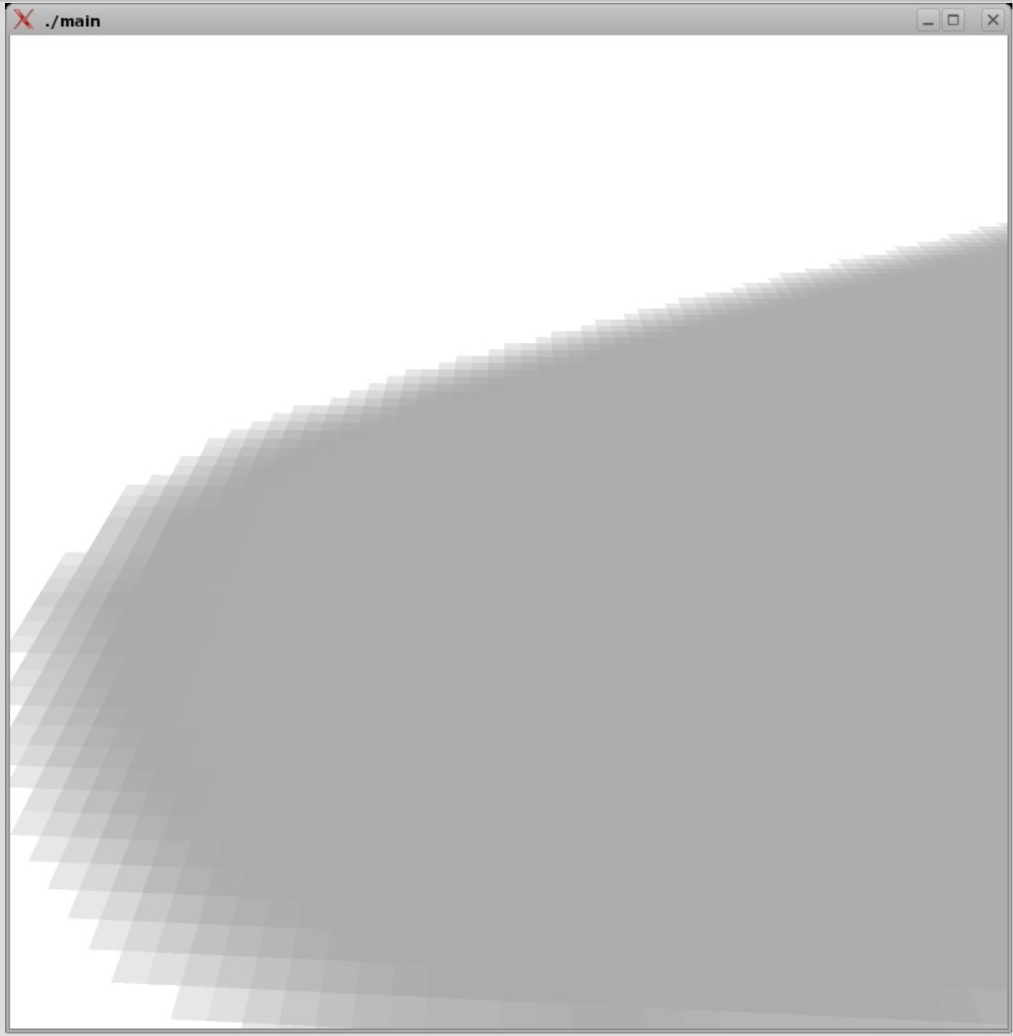
- Polygon Offset is controlled through two parameters: *factor* and *units*.
- Factor scales the maximum depth slope of the polygon
- Units scales the smallest value guaranteed a resolvable difference in window coordinate depth values
- Polygon offset = m * factor + r * units
- This offset is added to every given z-Value of each vertex.

Shadow Mapping

percentage closest filtering

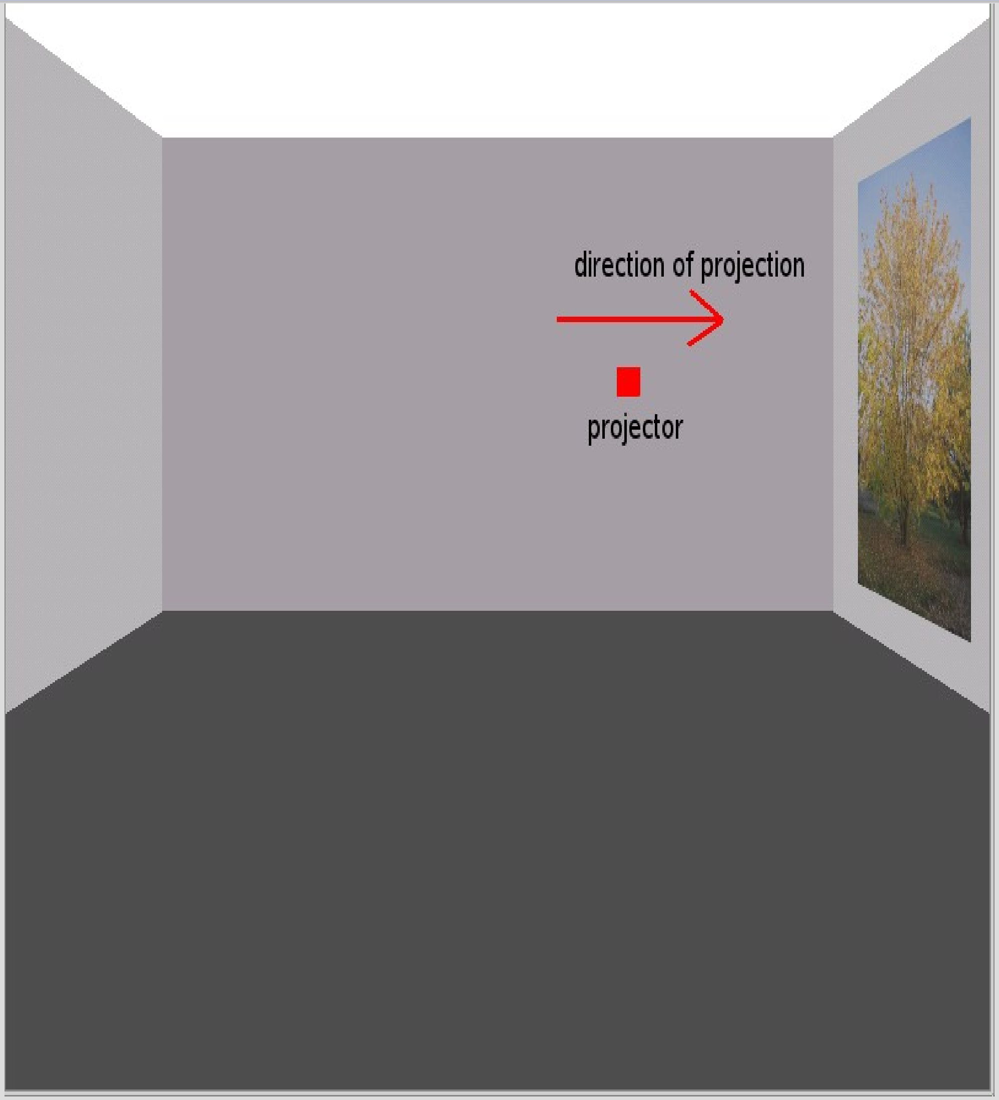


without pcf

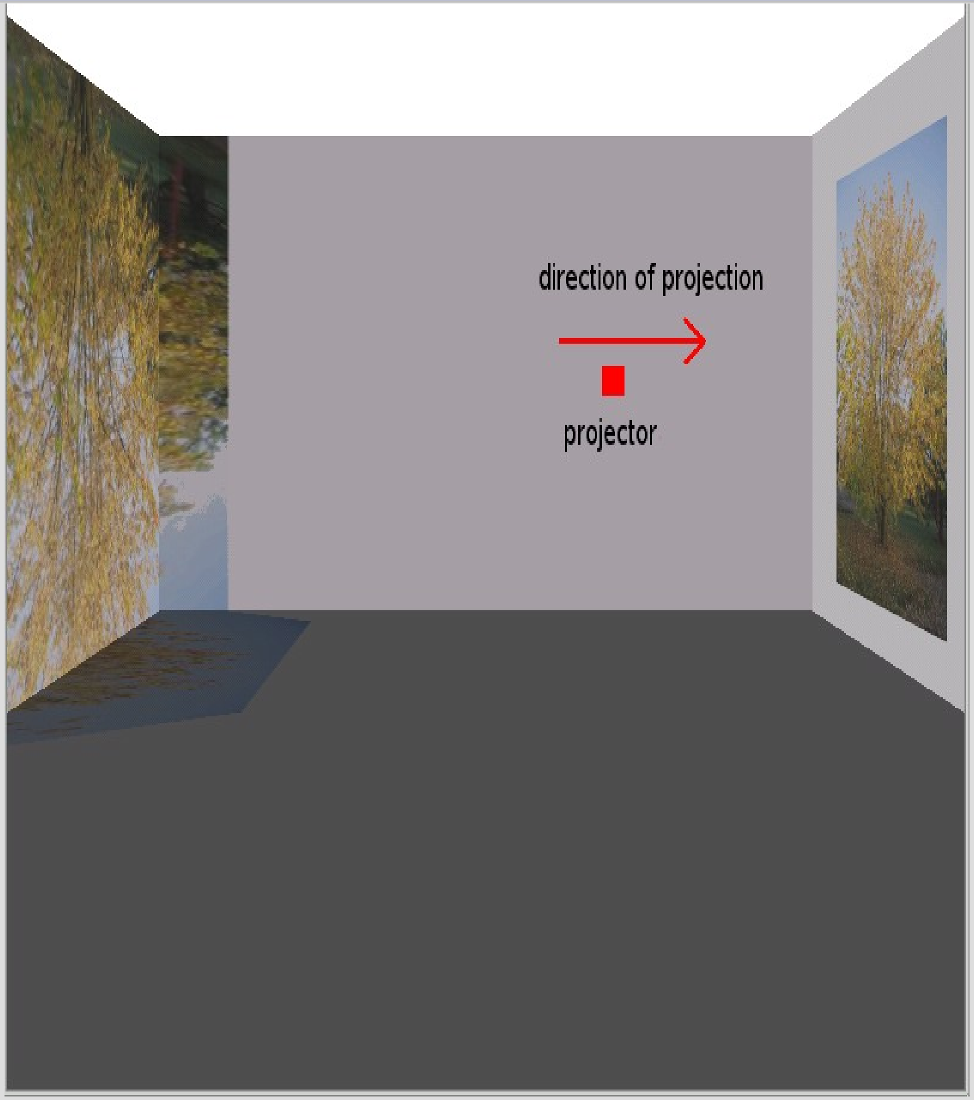


with pcf

Shadow Mapping Back Projections



Expected behavior of a projector



Troublesome back projections

Comparison Of Techniques

- Stencil Shadow Volume
  - + automatic self-shadows
  - + omni-directional lights
  - + no aliasing effects
  - - dependent on scene's complexity, because additional geometry must be rendered.
  - - soft shadows are not possible
  - - consumes a lot of fillrate
  - - clipping plane issues

- Shadow Mapping
  - + automatic self-shadows
  - + independent of scene complexity
  - + no additional geometry to be rendered
  - + soft shadows possible
  - - aliasing effects caused by sampling errors
  - - only directional lights
  - - incorrect self-shadowing
  - - incorrect back-projections

Literature

- Stencil Shadow Volumes
  - Morgan McGuire – GPU-Gems - Addison-Wesley Longman
  - Ashu Rege: Shadow Considerations (paper) - developer.nvidia.com
  - http://www.gamedev.net/columns/hardcore/shadowvolume/page1.asp
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/ShadowVolume_Sample.asp
- Shadow Mapping
  - Mark J. Kilgard - Shadow Mapping with Today's OpenGL Hardware (Paper) – developer.nvidia.com
  - http://www.paulsprojects.net/tutorials/smt/smt.html
- OpenGL
  - Shreiner,Woo,Neider - OpenGL Programming Guide The Official Guide to Learning OpenGL, Version 2, Addison-Wesley Longman
  - Rost, Kessenich, Lichtenbelt – OpenGL Shading Language, Addison-Wesley Longman