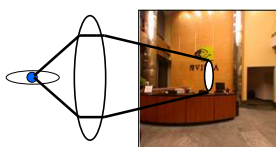
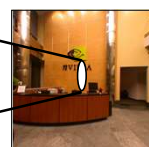







# Creating More Realistic Lens Effects

**Oregon State University**  
Mike Bailey  
mjb@cs.oregonstate.edu



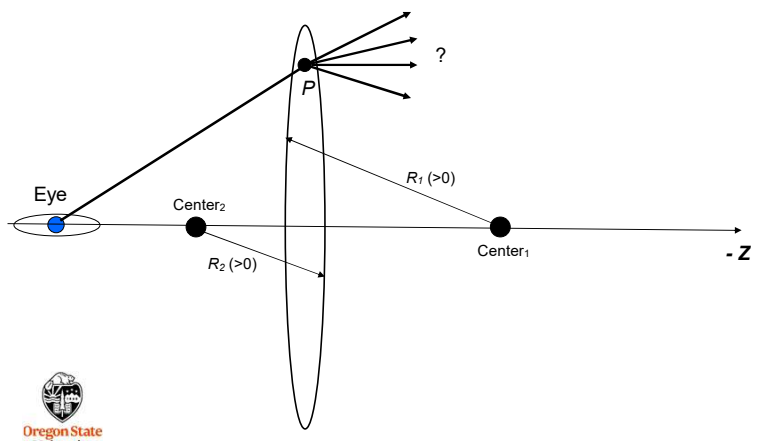
Oregon State University  
Computer Graphics

lens.pptx


mjb - December 31, 2021

1

## Convex Lens Definitions



The diagram shows an eye on the left, a convex lens in the middle, and a point P on the lens. Two centers of curvature are marked: Center<sub>2</sub> to the left of the lens and Center<sub>1</sub> to the right. Radii R<sub>1</sub> (>0) and R<sub>2</sub> (>0) are shown. A ray from the eye passes through P, and several other rays are shown originating from P, with a question mark indicating their direction.

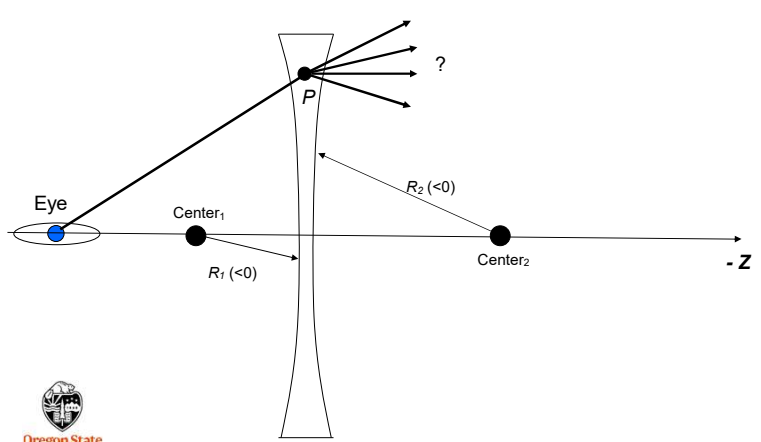


Oregon State University  
Computer Graphics


mjb - December 31, 2021

2

## Concave Lens Definitions



The diagram shows an eye on the left, a concave lens in the middle, and a point P on the lens. Two centers of curvature are marked: Center<sub>1</sub> to the left of the lens and Center<sub>2</sub> to the right. Radii R<sub>1</sub> (<0) and R<sub>2</sub> (<0) are shown. A ray from the eye passes through P, and several other rays are shown originating from P, with a question mark indicating their direction.

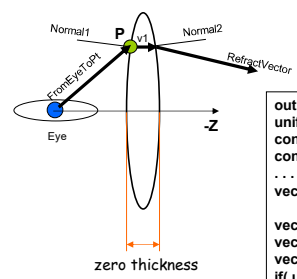


Oregon State University  
Computer Graphics

mjb - December 31, 2021

3

## lens.vert - Setup for a Cube Map Texture



The diagram shows a lens with a point P on its surface. A ray from the eye passes through P. Normal vectors Normal1 and Normal2 are shown at P. A refract vector is also shown. The lens has zero thickness.

```


out vec3    vRefractVector;
uniform float  uR1, uR2;
const float   ETA = 0.66;
const vec3    EYE = vec3( 0., 0., 0. );
...
vec3 P = vec3( gl_ModelViewMatrix * gl_Vertex );

vec3 FromEyeToPt = normalize( P - EYE ); // vector from eye to pt
vec3 Center1 = vec3( 0., 0., P.z - uR1 );
vec3 Normal1;
if( uR1 >= 0. )
    Normal1 = normalize( P - Center1 );
else
    Normal1 = normalize( Center1 - P );

vec3 v1 = refract( FromEyeToPt, Normal1, ETA ); // eta = in/out
v1 = normalize( v1 );
vec3 Center2 = vec3( 0., 0., P.z + uR2 );
vec3 Normal2;
if( uR2 >= 0. )
    Normal2 = normalize( Center2 - P );
else
    Normal2 = normalize( P - Center2 );

vRefractVector = refract( v1, Normal2, 1./ETA ); // 1./eta = out/in
gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;

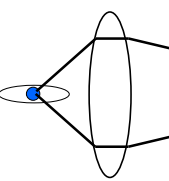

```




Oregon State University  
Computer Graphics


4

**Convex Lenses ( $R1 > 0, R2 > 0$ )**



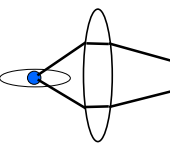
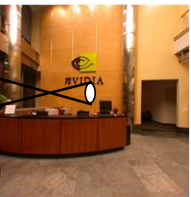
**Magnifying glasses and zoom lenses work this way**




Oregon State University  
Computer Graphics


5

**Convex Lenses ( $R1 > 0, R2 > 0$ )**



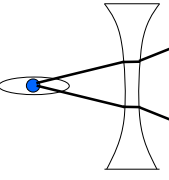
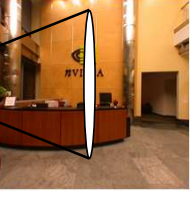
**Some telescopes work this way**




Oregon State University  
Computer Graphics


6

**Concave Lenses ( $R1 < 0, R2 < 0$ )**





**Fisheye lenses work this way**




Oregon State University  
Computer Graphics


7

**Convex**                      **Concave**



Oregon State University  
Computer Graphics



8