# Face-tracking based lenticular effect

Weihao Qiu

# Contents

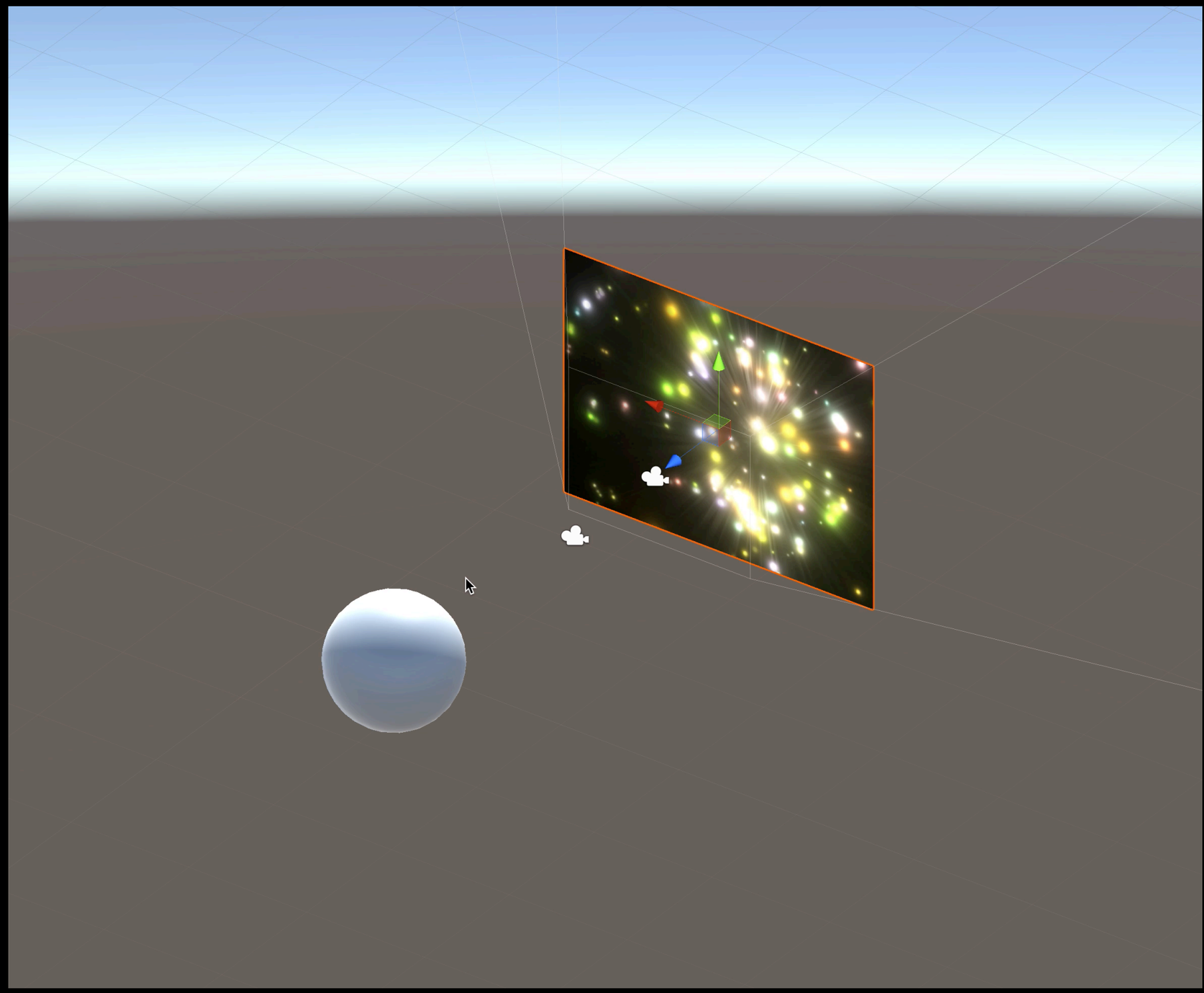
App Demo

Development highlight

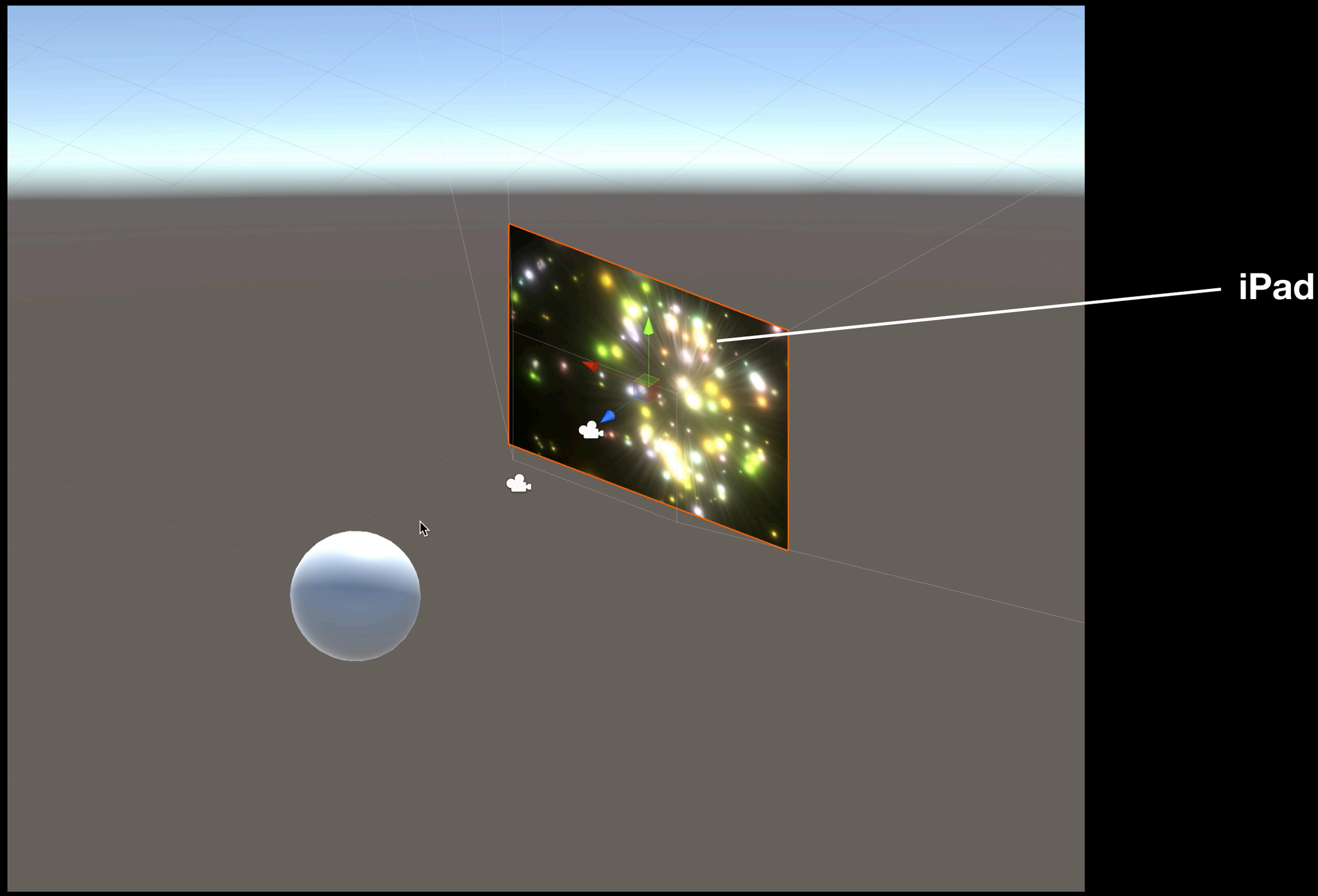
Future Work

# App Demo

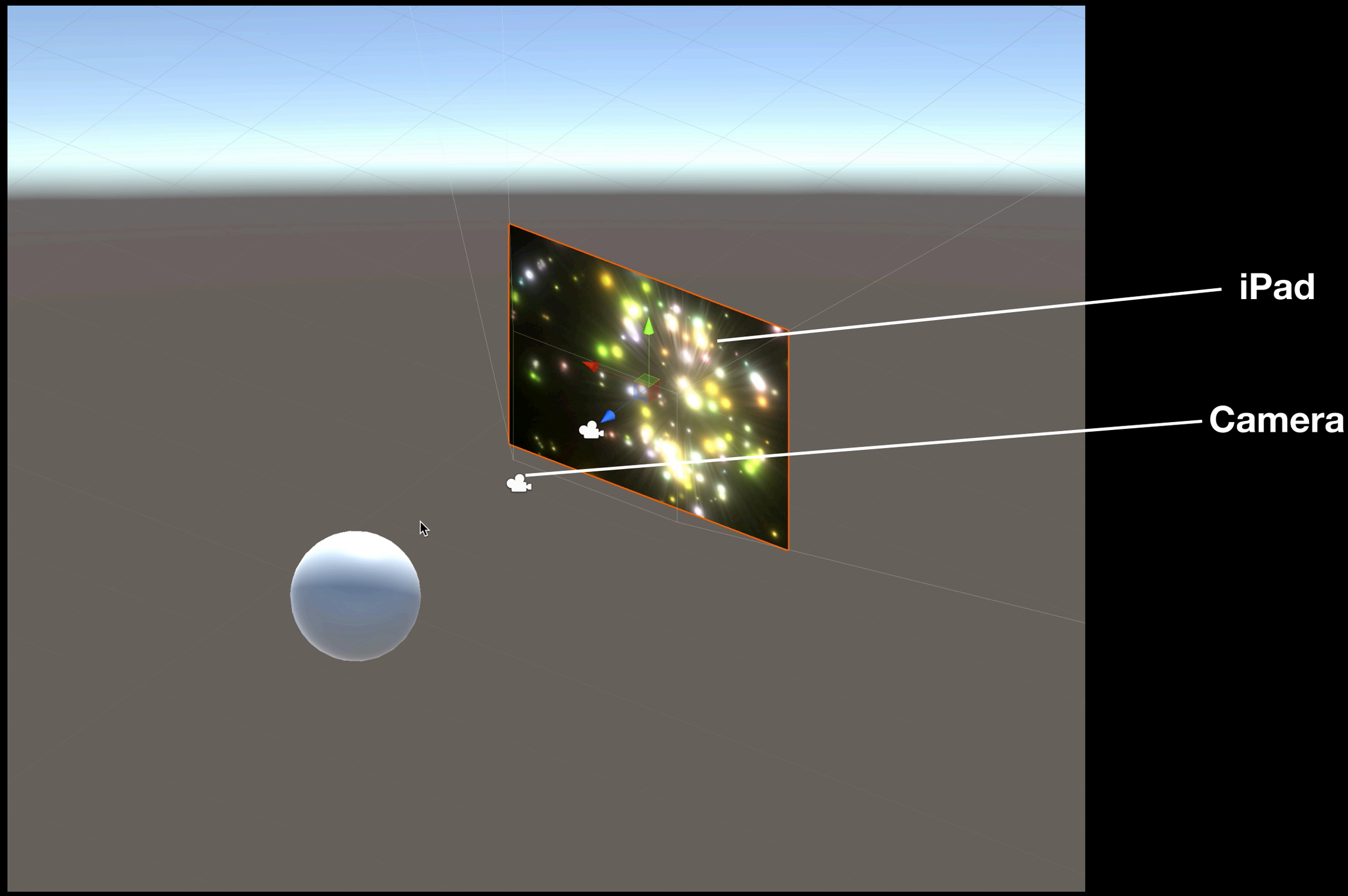
# Development Highlight



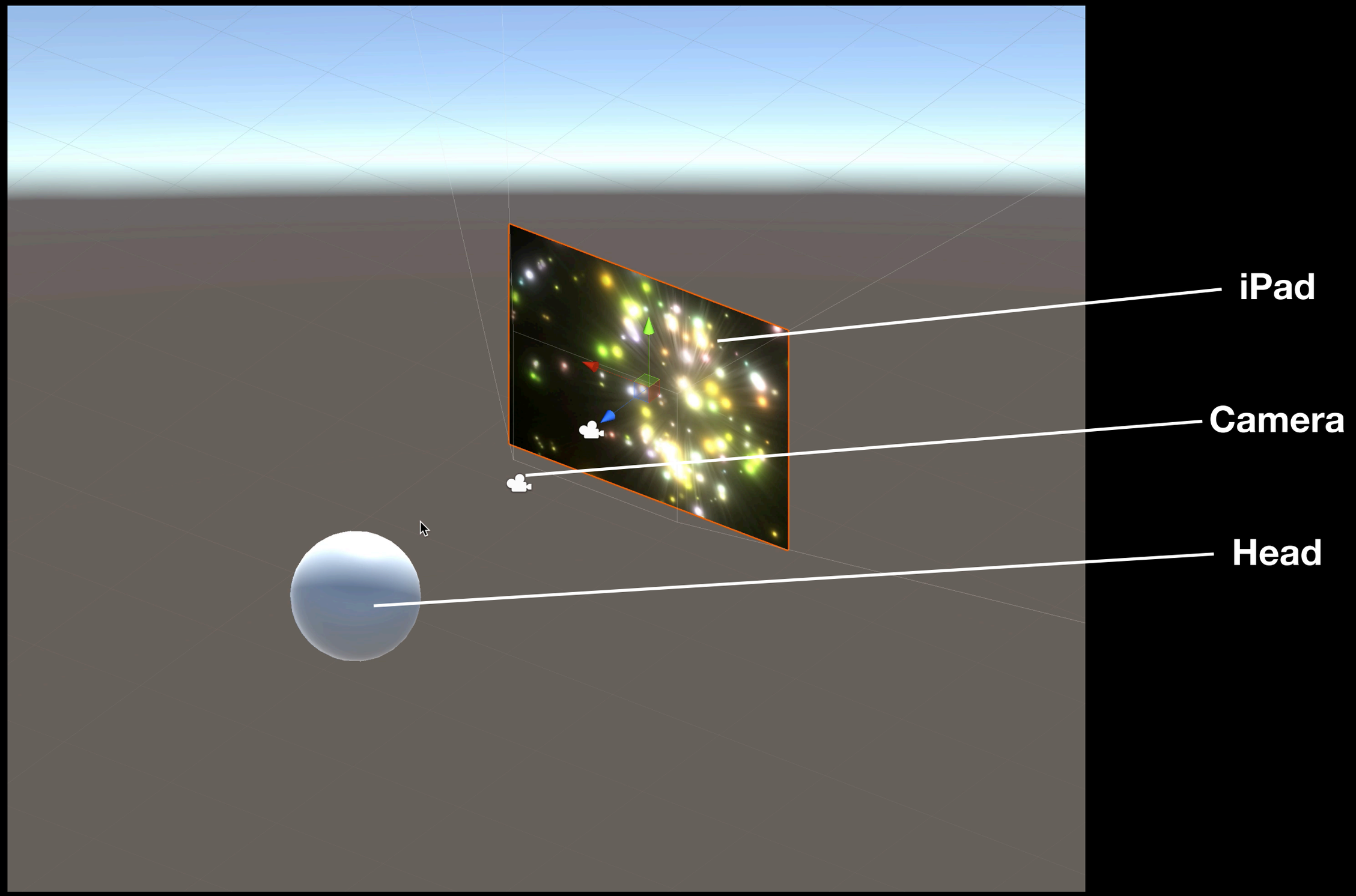
# Development Highlight



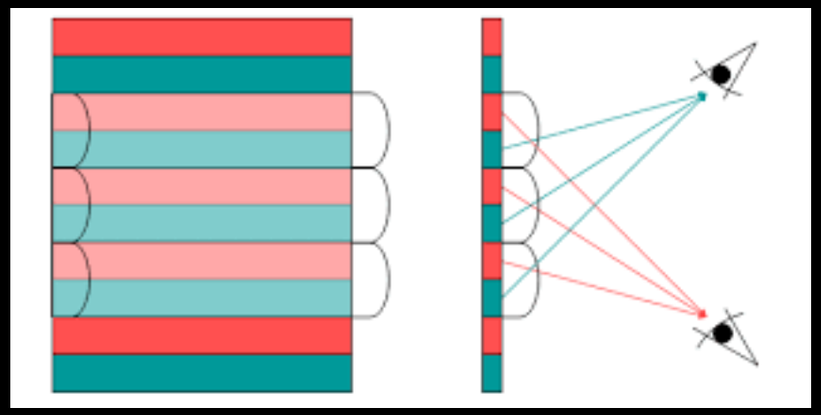
# Development Highlight



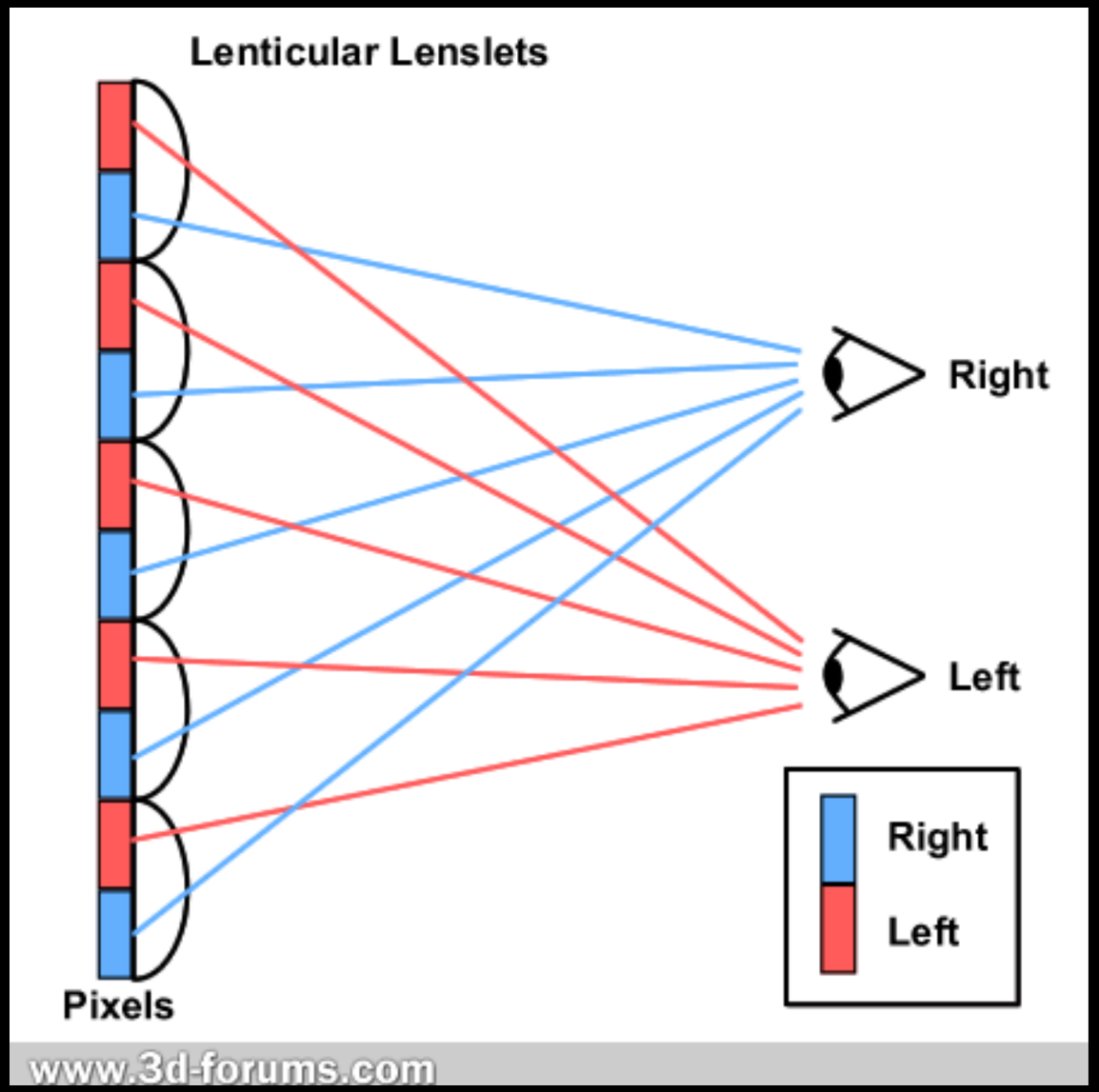
# Development Highlight



# Development Highlight



Animation




3D Effect

# Development Highlight


```
float3 viewDir = normalize(_FacePos.xyz - i.worldPos);

float angle =  acos(DotClamped(
    normalize(float3(i.normal.x, 0.0, i.normal.z)),
    normalize(float3(viewDir.x , 0.0, viewDir.z ))
));

float slotAngle = (2 * idx + 1) * radians(VisRange)/textureNum/2.0 - radians(VisRange/2.0);

float angleDiff = abs(angle - slotAngle);

visbility[idx] = 1.0 - angleDiff/radians(VisRange)*textureNum;


fixed4 col =
      tex2D(_PhotoTex1, i.uv.xy) * visbility[0]
    + tex2D(_PhotoTex2, i.uv.xy) * visbility[1]
    + tex2D(_PhotoTex3, i.uv.xy) * visbility[2]
    + tex2D(_PhotoTex4, i.uv.xy) * visbility[3]
    + tex2D(_PhotoTex5, i.uv.xy) * visbility[4]
    + tex2D(_PhotoTex6, i.uv.xy) * visbility[5]
    + tex2D(_PhotoTex7, i.uv.xy) * visbility[6]
    + tex2D(_PhotoTex8, i.uv.xy) * visbility[7]
    + tex2D(_PhotoTex9, i.uv.xy) * visbility[8]
    + tex2D(_PhotoTex10, i.uv.xy) * visbility[9]
    + tex2D(_PhotoTex11, i.uv.xy) * visbility[10]
    + tex2D(_PhotoTex12, i.uv.xy) * visbility[11]
    + tex2D(_PhotoTex13, i.uv.xy) * visbility[12]
    + tex2D(_PhotoTex14, i.uv.xy) * visbility[13]
    + tex2D(_PhotoTex15, i.uv.xy) * visbility[14]
;
```

# Development Highlight


```
float3 viewDir = normalize(_FacePos.xyz - i.worldPos);

float angle =  acos(DotClamped(
    normalize(float3(i.normal.x, 0.0, i.normal.z)),
    normalize(float3(viewDir.x , 0.0, viewDir.z ))
));

float slotAngle = (2 * idx + 1) * radians(VisRange)/textureNum/2.0 - radians(VisRange/2.0);

float angleDiff = abs(angle - slotAngle);

visbility[idx] = 1.0 - angleDiff/radians(VisRange)*textureNum;


fixed4 col =
      tex2D(_PhotoTex1, i.uv.xy) * visbility[0]
    + tex2D(_PhotoTex2, i.uv.xy) * visbility[1]
    + tex2D(_PhotoTex3, i.uv.xy) * visbility[2]
    + tex2D(_PhotoTex4, i.uv.xy) * visbility[3]
    + tex2D(_PhotoTex5, i.uv.xy) * visbility[4]
    + tex2D(_PhotoTex6, i.uv.xy) * visbility[5]
    + tex2D(_PhotoTex7, i.uv.xy) * visbility[6]
    + tex2D(_PhotoTex8, i.uv.xy) * visbility[7]
    + tex2D(_PhotoTex9, i.uv.xy) * visbility[8]
    + tex2D(_PhotoTex10, i.uv.xy) * visbility[9]
    + tex2D(_PhotoTex11, i.uv.xy) * visbility[10]
    + tex2D(_PhotoTex12, i.uv.xy) * visbility[11]
    + tex2D(_PhotoTex13, i.uv.xy) * visbility[12]
    + tex2D(_PhotoTex14, i.uv.xy) * visbility[13]
    + tex2D(_PhotoTex15, i.uv.xy) * visbility[14]
;
```

# Development Highlight


```
float3 viewDir = normalize(_FacePos.xyz - i.worldPos);

float angle =  acos(DotClamped(
    normalize(float3(i.normal.x, 0.0, i.normal.z)),
    normalize(float3(viewDir.x , 0.0, viewDir.z ))
));

float slotAngle = (2 * idx + 1) * radians(VisRange)/textureNum/2.0 - radians(VisRange/2.0);

float angleDiff = abs(angle - slotAngle);

visbility[idx] = 1.0 - angleDiff/radians(VisRange)*textureNum;


fixed4 col =
      tex2D(_PhotoTex1, i.uv.xy) * visbility[0]
    + tex2D(_PhotoTex2, i.uv.xy) * visbility[1]
    + tex2D(_PhotoTex3, i.uv.xy) * visbility[2]
    + tex2D(_PhotoTex4, i.uv.xy) * visbility[3]
    + tex2D(_PhotoTex5, i.uv.xy) * visbility[4]
    + tex2D(_PhotoTex6, i.uv.xy) * visbility[5]
    + tex2D(_PhotoTex7, i.uv.xy) * visbility[6]
    + tex2D(_PhotoTex8, i.uv.xy) * visbility[7]
    + tex2D(_PhotoTex9, i.uv.xy) * visbility[8]
    + tex2D(_PhotoTex10, i.uv.xy) * visbility[9]
    + tex2D(_PhotoTex11, i.uv.xy) * visbility[10]
    + tex2D(_PhotoTex12, i.uv.xy) * visbility[11]
    + tex2D(_PhotoTex13, i.uv.xy) * visbility[12]
    + tex2D(_PhotoTex14, i.uv.xy) * visbility[13]
    + tex2D(_PhotoTex15, i.uv.xy) * visbility[14]
;
```

# Development Highlight


```
float3 viewDir = normalize(_FacePos.xyz - i.worldPos);

float angle =  acos(DotClamped(
    normalize(float3(i.normal.x, 0.0, i.normal.z)),
    normalize(float3(viewDir.x , 0.0, viewDir.z ))
));

float slotAngle = (2 * idx + 1) * radians(VisRange)/textureNum/2.0 - radians(VisRange/2.0);

float angleDiff = abs(angle - slotAngle);

visbility[idx] = 1.0 - angleDiff/radians(VisRange)*textureNum;


fixed4 col =
      tex2D(_PhotoTex1, i.uv.xy) * visbility[0]
    + tex2D(_PhotoTex2, i.uv.xy) * visbility[1]
    + tex2D(_PhotoTex3, i.uv.xy) * visbility[2]
    + tex2D(_PhotoTex4, i.uv.xy) * visbility[3]
    + tex2D(_PhotoTex5, i.uv.xy) * visbility[4]
    + tex2D(_PhotoTex6, i.uv.xy) * visbility[5]
    + tex2D(_PhotoTex7, i.uv.xy) * visbility[6]
    + tex2D(_PhotoTex8, i.uv.xy) * visbility[7]
    + tex2D(_PhotoTex9, i.uv.xy) * visbility[8]
    + tex2D(_PhotoTex10, i.uv.xy) * visbility[9]
    + tex2D(_PhotoTex11, i.uv.xy) * visbility[10]
    + tex2D(_PhotoTex12, i.uv.xy) * visbility[11]
    + tex2D(_PhotoTex13, i.uv.xy) * visbility[12]
    + tex2D(_PhotoTex14, i.uv.xy) * visbility[13]
    + tex2D(_PhotoTex15, i.uv.xy) * visbility[14]
;
```

# Development Highlight


```
float3 viewDir = normalize(_FacePos.xyz - i.worldPos);

float angle =  acos(DotClamped(
    normalize(float3(i.normal.x, 0.0, i.normal.z)),
    normalize(float3(viewDir.x , 0.0, viewDir.z ))
));

float slotAngle = (2 * idx + 1) * radians(VisRange)/textureNum/2.0 - radians(VisRange/2.0);

float angleDiff = abs(angle - slotAngle);

visbility[idx] = 1.0 - angleDiff/radians(VisRange)*textureNum;


fixed4 col =
      tex2D(_PhotoTex1, i.uv.xy) * visbility[0]
    + tex2D(_PhotoTex2, i.uv.xy) * visbility[1]
    + tex2D(_PhotoTex3, i.uv.xy) * visbility[2]
    + tex2D(_PhotoTex4, i.uv.xy) * visbility[3]
    + tex2D(_PhotoTex5, i.uv.xy) * visbility[4]
    + tex2D(_PhotoTex6, i.uv.xy) * visbility[5]
    + tex2D(_PhotoTex7, i.uv.xy) * visbility[6]
    + tex2D(_PhotoTex8, i.uv.xy) * visbility[7]
    + tex2D(_PhotoTex9, i.uv.xy) * visbility[8]
    + tex2D(_PhotoTex10, i.uv.xy) * visbility[9]
    + tex2D(_PhotoTex11, i.uv.xy) * visbility[10]
    + tex2D(_PhotoTex12, i.uv.xy) * visbility[11]
    + tex2D(_PhotoTex13, i.uv.xy) * visbility[12]
    + tex2D(_PhotoTex14, i.uv.xy) * visbility[13]
    + tex2D(_PhotoTex15, i.uv.xy) * visbility[14]
;
```

# Development Highlight


```
float3 viewDir = normalize(_FacePos.xyz - i.worldPos);

float angle =  acos(DotClamped(
    normalize(float3(i.normal.x, 0.0, i.normal.z)),
    normalize(float3(viewDir.x , 0.0, viewDir.z ))
));

float slotAngle = (2 * idx + 1) * radians(VisRange)/textureNum/2.0 - radians(VisRange/2.0);

float angleDiff = abs(angle - slotAngle);

visbility[idx] = 1.0 - angleDiff/radians(VisRange)*textureNum;


fixed4 col =
      tex2D(_PhotoTex1, i.uv.xy) * visbility[0]
    + tex2D(_PhotoTex2, i.uv.xy) * visbility[1]
    + tex2D(_PhotoTex3, i.uv.xy) * visbility[2]
    + tex2D(_PhotoTex4, i.uv.xy) * visbility[3]
    + tex2D(_PhotoTex5, i.uv.xy) * visbility[4]
    + tex2D(_PhotoTex6, i.uv.xy) * visbility[5]
    + tex2D(_PhotoTex7, i.uv.xy) * visbility[6]
    + tex2D(_PhotoTex8, i.uv.xy) * visbility[7]
    + tex2D(_PhotoTex9, i.uv.xy) * visbility[8]
    + tex2D(_PhotoTex10, i.uv.xy) * visbility[9]
    + tex2D(_PhotoTex11, i.uv.xy) * visbility[10]
    + tex2D(_PhotoTex12, i.uv.xy) * visbility[11]
    + tex2D(_PhotoTex13, i.uv.xy) * visbility[12]
    + tex2D(_PhotoTex14, i.uv.xy) * visbility[13]
    + tex2D(_PhotoTex15, i.uv.xy) * visbility[14]
;
```

# Future Work

# Future Work

Different Textures

# Future Work

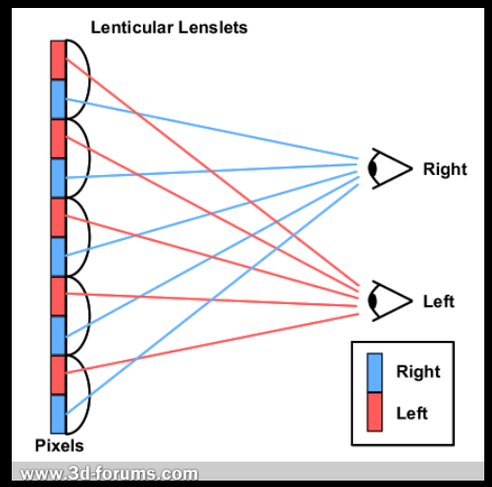
Different Textures

Glass-free Stereo display

# Future Work

Different Textures

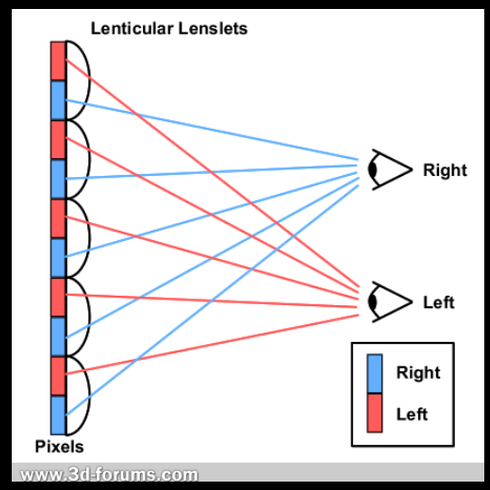
Glass-free Stereo display



# Future Work

Different Textures

Glass-free Stereo display



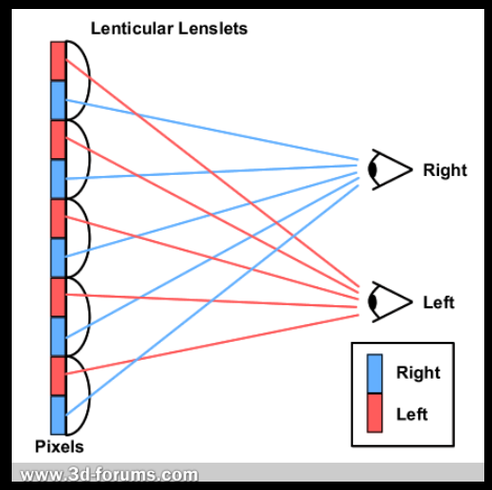


2012

# Future Work

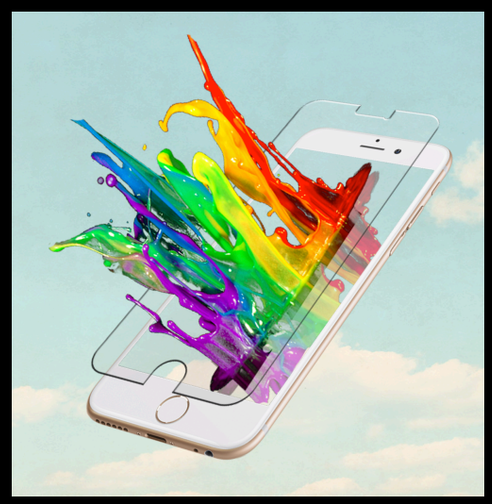
Different Textures

Glass-free Stereo display





2012



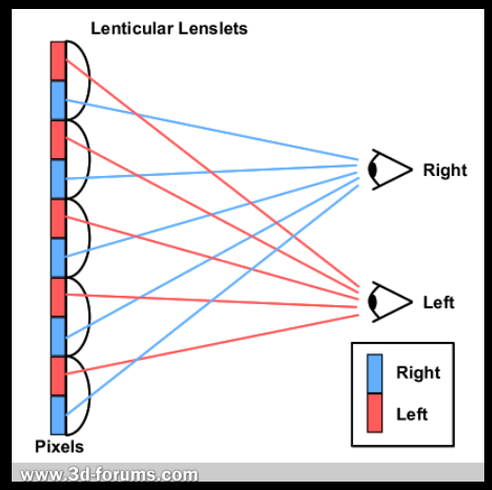
2014

# Future Work

Different Textures

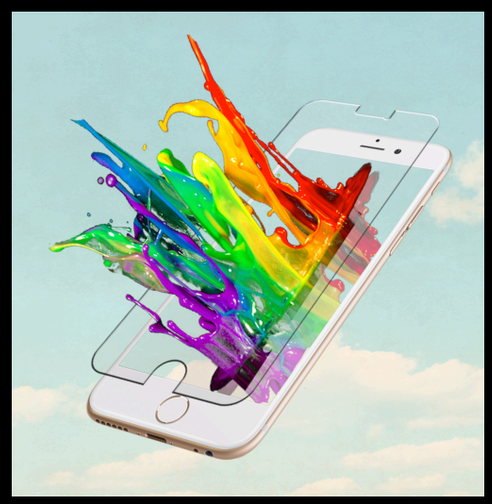
Glass-free Stereo display

Computational display





2012



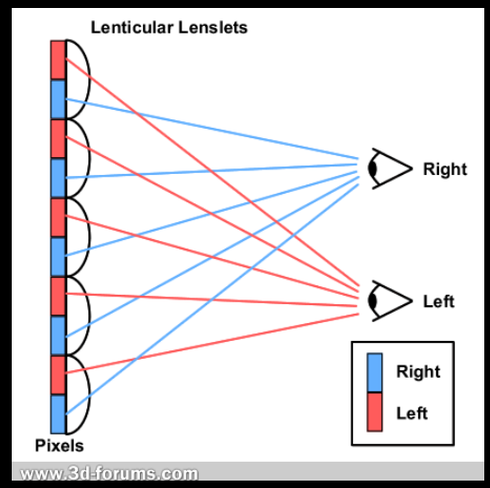
2014

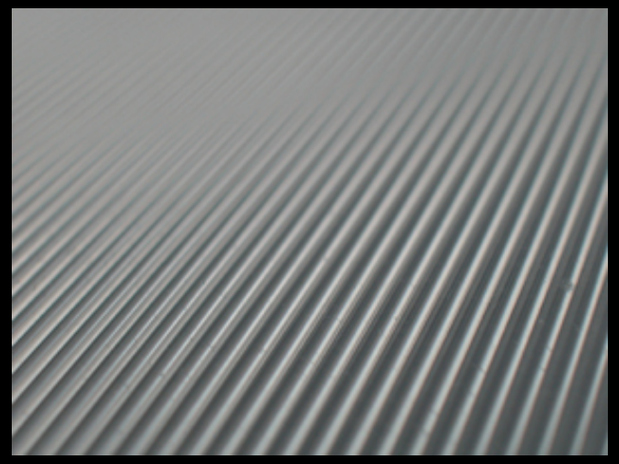
# Future Work

Different Textures

Glass-free Stereo display

Computational display



**Lenticular, 2D**


**2012**

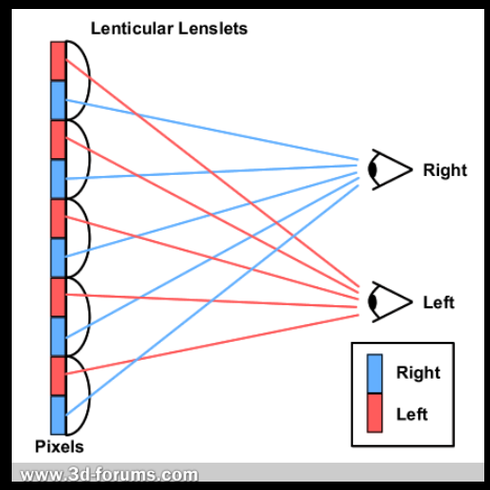
**2014**

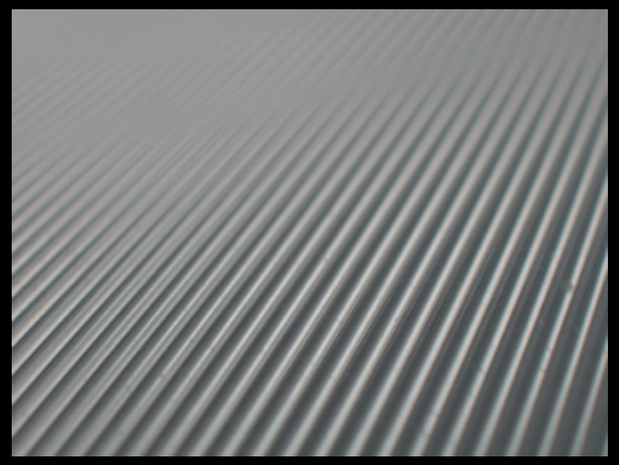
# Future Work

Different Textures

Glass-free Stereo display

Computational display



**Lenticular, 2D**


**2012**

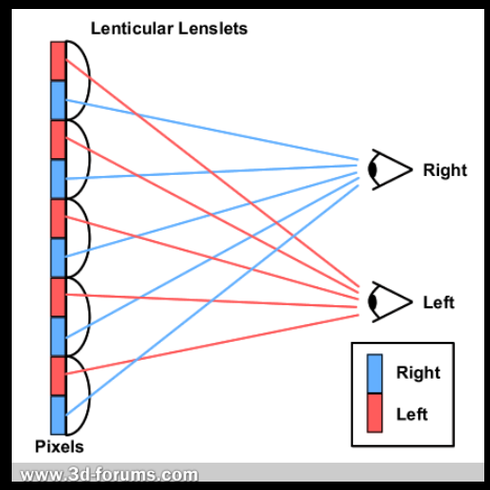
**2014**

# Future Work

Different Textures

Glass-free Stereo display

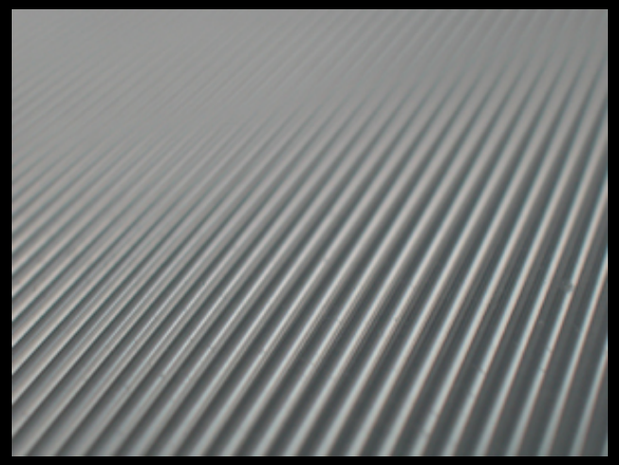
Computational display



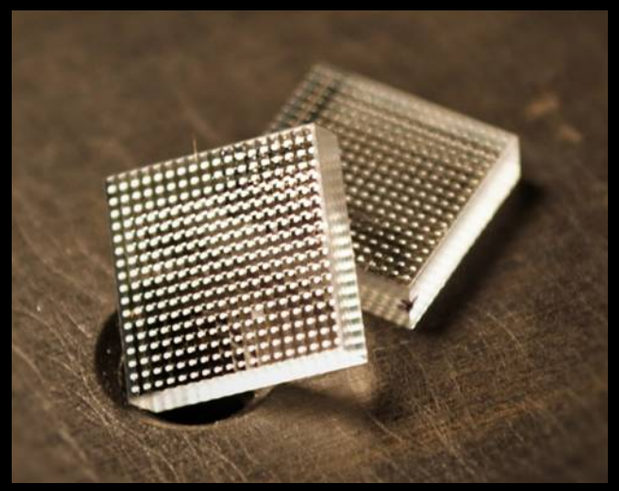


2012

2014



Lenticular, 2D



Microlens Array, 3D

# Thank you!

Q&A

## Image Sources:

https://www.apollooptical.com/microlens-arrays/
https://www.eyefly3d.com
https://www.nintendo.com/3ds/features/compare
https://en.wikipedia.org/wiki/Lenticular_lens