

RENDERIZACION DE IMAGENES

Alan Hernan Tintaya Montecinos



¿QUÉ ES “IMAGE-BASED RENDERING” (IBR)?

- **Definición:** IBR es un conjunto de técnicas que generan vistas de escenas usando directamente imágenes capturadas, en vez de reconstruir toda la geometría y simular la luz desde cero.
- **Idea clave:** “Aprovechar datos de fotografías para renderizar entornos o reflejos con alta fidelidad visual.”



TIPOS PRINCIPALES DE IBR

ENVIRONMENT MAPPING (CUBEMAPS, SPHEREMAP):

Usa seis imágenes o una única mapa esférico para simular reflexiones o skyboxes.

DEPTH-IMAGE BASED RENDERING (DIBR):

Combina imágenes 2D con un mapa de profundidad para generar vistas intermedias.

LIGHT FIELDS / LUMIGRAPH:

Captura grandes conjuntos de imágenes desde múltiples posiciones; permite interpolar cualquier punto del volumen visual.



¿POR QUÉ USAR CUBEMAPS?

Simplicidad

sólo seis imágenes, API WebGL built-in.

Versatilidad

texturas cúbicas optimizadas, mipmaps automáticos.

Rendimiento

skybox, reflexiones especulares, iluminación ambient.

Calidad

captura realista del entorno, sin costosos cálculos de iluminación.

VERTEX SHADER DE LA ESFERA



```
1 <script id="vs" type="vertex">
2     #version 300 es
3     precision mediump float;
4     uniform mat4 uMatrizProyeccion;
5     uniform mat4 uMatrizVista;
6     uniform mat4 uMatrizModelo;
7     layout(location = 0) in vec3 aVertices;
8     layout(location = 2) in vec3 aNormal;
9     out vec3 vNormal;
10    out vec3 vViewDir;
11    layout(location = 1) in vec2 aCoordenadasDeTextura;
12    out vec2 vCoordenadasDeTextura;
13    void main() {
14        vec4 posView = uMatrizVista * uMatrizModelo * vec4(aVertices, 1.0);
15        vViewDir = normalize(-posView.xyz);
16        vNormal = normalize(mat3(uMatrizVista * uMatrizModelo) * aNormal);
17        gl_Position = uMatrizProyeccion * posView;
18    }
19 </script>
```

VERTEX SHADER DE SKY-BOX

```
● ● ●  
1 <script id="vsSky" type="vertex">  
2 #version 300 es  
3 precision mediump float;  
4 layout(location=0) in vec3 aPosition;  
5 uniform mat4 uView;  
6 uniform mat4 uProjection;  
7 out vec3 vTexCoord;  
8 void main() {  
9     vTexCoord = aPosition;  
10    vec4 pos = uProjection * uView * vec4(aPosition, 1.0);  
11    gl_Position = pos.xyww;  
12 }  
13 </script>
```

¿COMO CARGA EL CUBEMAP?



```
1 
2 
3 
4 
5 
6 
```



```
1 function leeCubemap(gl, codigoCubemap) {
2     gl.bindTexture(gl.TEXTURE_CUBE_MAP, codigoCubemap);
3     const caras = [
4         { target: gl.TEXTURE_CUBE_MAP_POSITIVE_X, id: "posx" },
5         { target: gl.TEXTURE_CUBE_MAP_NEGATIVE_X, id: "negx" },
6         { target: gl.TEXTURE_CUBE_MAP_POSITIVE_Y, id: "posy" },
7         { target: gl.TEXTURE_CUBE_MAP_NEGATIVE_Y, id: "negy" },
8         { target: gl.TEXTURE_CUBE_MAP_POSITIVE_Z, id: "posz" },
9         { target: gl.TEXTURE_CUBE_MAP_NEGATIVE_Z, id: "negz" },
10    ];
11    caras.forEach((c) => {
12        const img = document.getElementById(c.id);
13        gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, false);
14        gl.texImage2D(c.target, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, img);
15    });
16    gl.generateMipmap(gl.TEXTURE_CUBE_MAP);
17    gl.texParameteri(
18        gl.TEXTURE_CUBE_MAP,
19        gl.TEXTURE_MIN_FILTER,
20        gl.LINEAR_MIPMAP_LINEAR
21    );
22    gl.texParameteri(gl.TEXTURE_CUBE_MAP, gl.TEXTURE_MAG_FILTER, gl.LINEAR);
23    gl.texParameteri(
24        gl.TEXTURE_CUBE_MAP,
25        gl.TEXTURE_WRAP_S,
26        gl.CLAMP_TO_EDGE
27    );
28    gl.texParameteri(
29        gl.TEXTURE_CUBE_MAP,
30        gl.TEXTURE_WRAP_T,
31        gl.CLAMP_TO_EDGE
32    );
33    gl.texParameteri(
34        gl.TEXTURE_CUBE_MAP,
35        gl.TEXTURE_WRAP_R,
36        gl.CLAMP_TO_EDGE
37    );
38    gl.bindTexture(gl.TEXTURE_CUBE_MAP, null);
39 }
```

FUNCTION DIBUJA()



```
1 function dibuja() {
2     let viewNoTrans = MatrizVista.slice();
3     viewNoTrans[12] = viewNoTrans[13] = viewNoTrans[14] = 0;
4     perspective(
5         MatrizProyeccion,
6         fov,
7         gl.canvas.width / gl.canvas.height,
8         0.1,
9         200.0
10    );
11    gl.uniformMatrix4fv(uMatrizProyeccion, false, MatrizProyeccion);
12    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
13
14    // skybox
15    gl.depthFunc(gl.LEQUAL);
16    gl.useProgram(skyProg);
17    gl.uniformMatrix4fv(uViewSky, false, viewNoTrans);
18    gl.uniformMatrix4fv(uProjSky, false, MatrizProyeccion);
19    gl.bindVertexArray(skyboxVAO);
20    gl.drawArrays(gl.TRIANGLES, 0, 36);
```



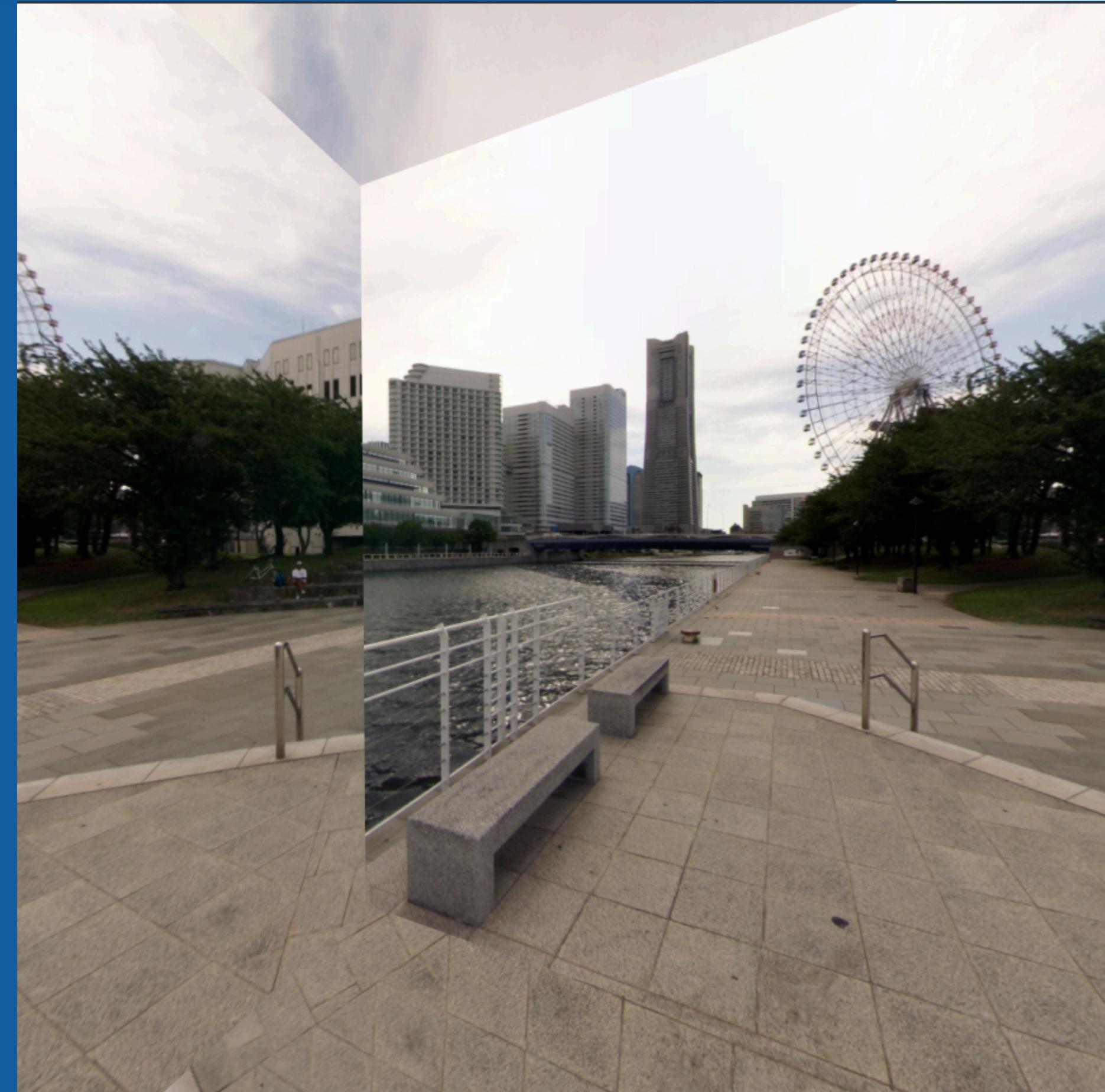
```
1
2    // esfera
3    gl.depthFunc(gl.LESS);
4    gl.useProgram(programaID);
5    identidad(MatrizVista);
6    rotacionX(MatrizVista, -rotX);
7    rotacionY(MatrizVista, -rotY);
8    gl.uniformMatrix4fv(uMatrizVista, false, MatrizVista);
9
10   gl.bindVertexArray(textura.esferaVAO);
11   gl.drawElements(
12       gl.TRIANGLES,
13       textura.cantidadDeIndices,
14       gl.UNSIGNED_SHORT,
15       0
16   );
17
18   requestAnimationFrame(dibuja);
19 }
```

¿COMO FUNCIONA?

Prácticamente, las imágenes las renderiza como si estuviéramos dentro de un cubo

En el ejemplo se repitió la misma imagen para lograr ver las separaciones

La esfera nos ayuda a poder navegar de esta forma (esfera), y así tener la interactividad



¿COMO FUNCIONA?



GRACIAS!