Introducción

Prof. Dr. Hans H. Ccacyahuillca Bejar

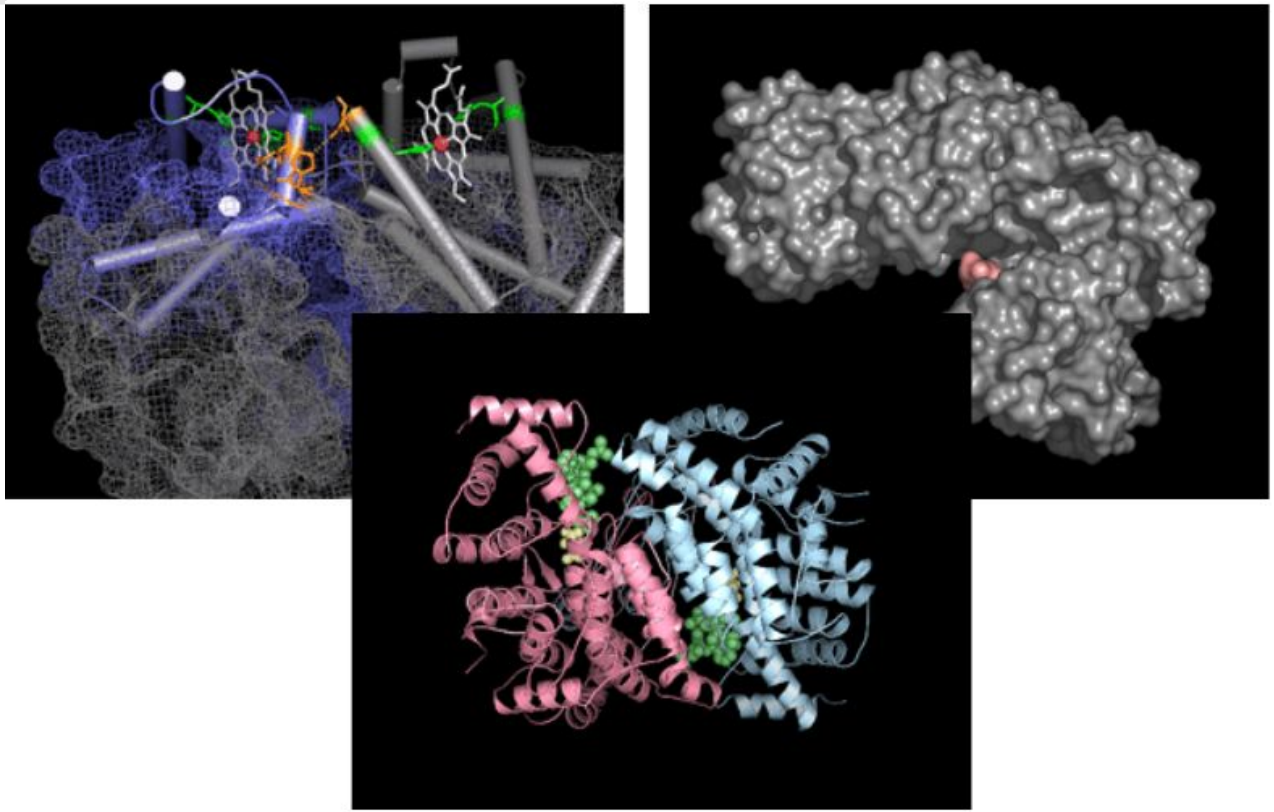
Aplicaciones

- Entretenimiento
  - Películas y animaciones
  - Efectos especiales
  - Games
- Ciencia e ingeniería
  - Computer-aided design
  - Visualización (e.g. científica, informacional)
- Esquematización virtual
- Entrenamiento y simulaciones

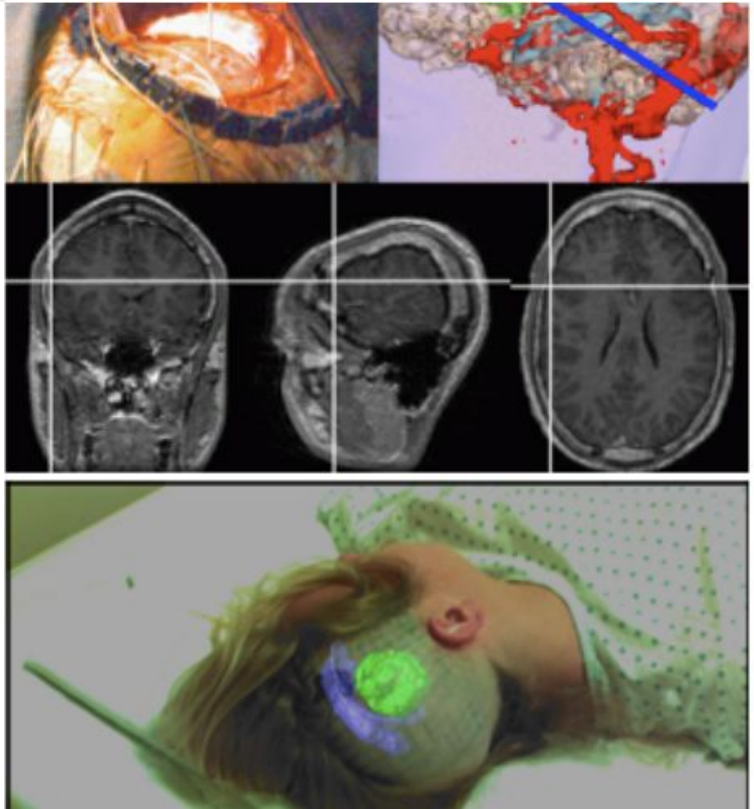
Películas

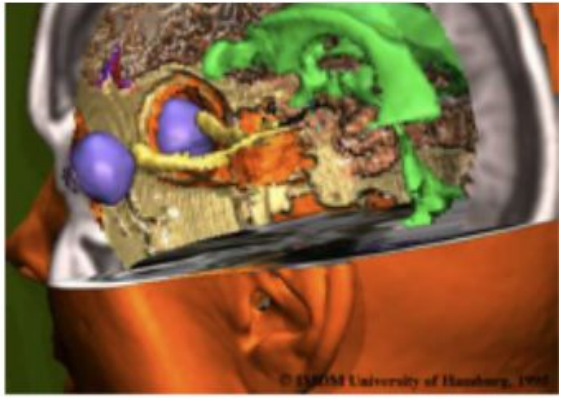
Games

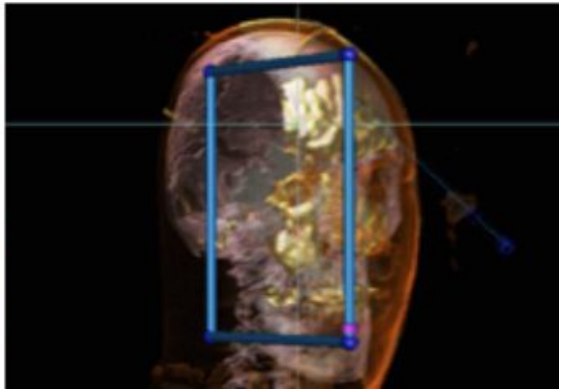
Visualización científica



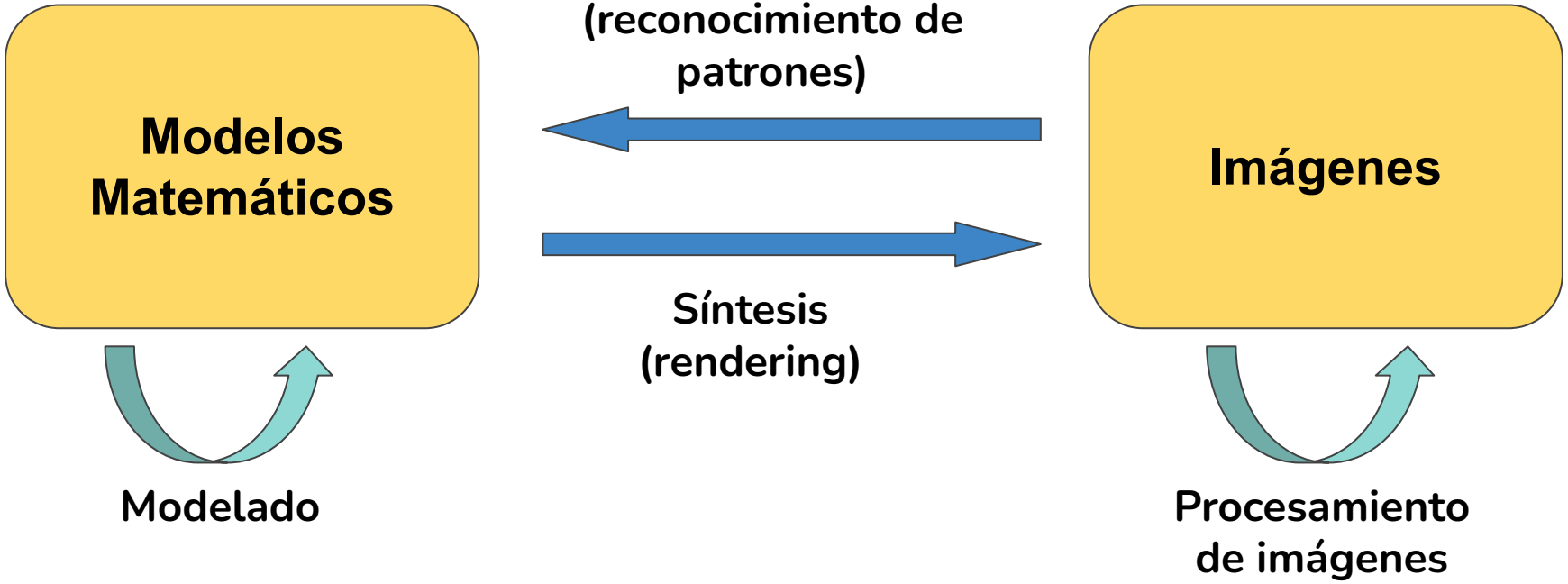
Visualización científica







Computación gráfica

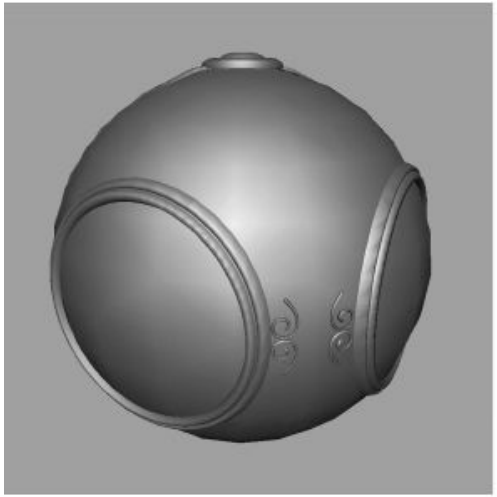


Computación Gráfica

En computación gráfica se ven aspectos relacionados a la creación o sintetización de una imagen por computador:

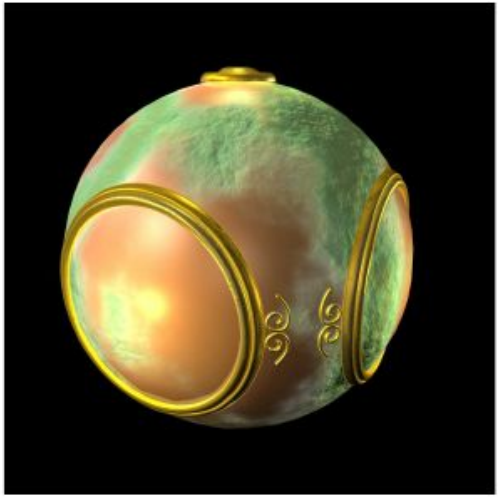
- Hardware
- Software
- Aplicación

Aumento de Realismo (1980-1990)



smooth shading

environment mapping



bump mapping

1990-2000

- OpenGL API
- Películas interactivamente generadas por computador (e.g. ToyStory, 1995)
- Nueva capacidades en hardware
- Mapeamento de textura

2000-2010

- Placas gráficas para PC que dominan el mercado:
- NVidia, AMD (ex ATI), Intel
- Pipelines programables
  - GPU

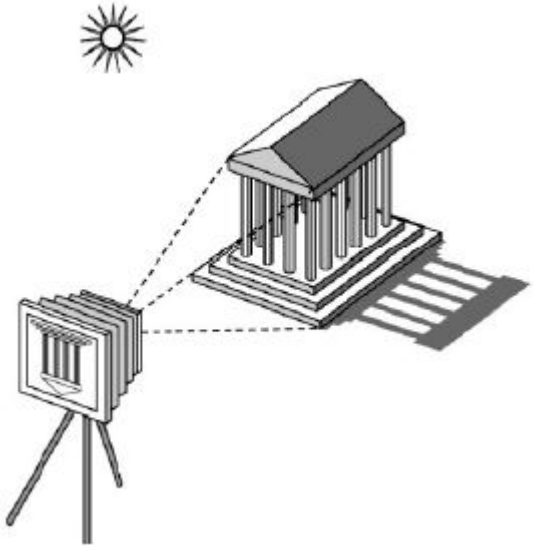
2011-actual

- CG es ubicua
- Telefones celulares
- Dispositivos embebidos (autónomos)
- OpenGL ES y WebGL
- Realidade alternativa e aumentada
- TVs e filmes em 3D

Realidad aumentada

Formación de una imagen

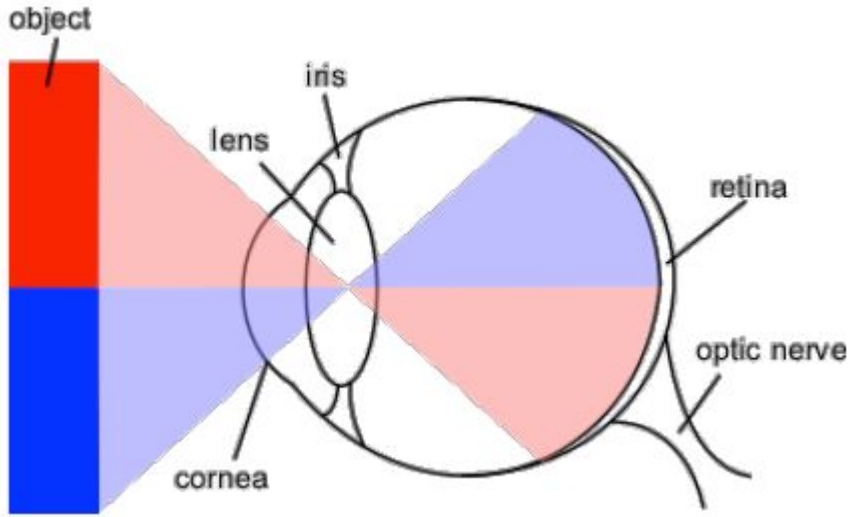
- Usa un proceso análogo a los formados por sistemas de imagen físicos (e.g. cámaras, microscopios, sistema visual humano)
- Elementos da formação da imagem
  - Objetos
  - Observador (Viewer)
  - Fuente(s) de iluminación



Sistema visual humano

Tiene dos tipos de sensores

- Bastoncillos;
  - Monocromáticos, visión nocturna
- Conos:
  - Sensibles a colores
- Procesamiento inicial de la luz en humanos sigue los mismos principios de los sistemas ópticos.

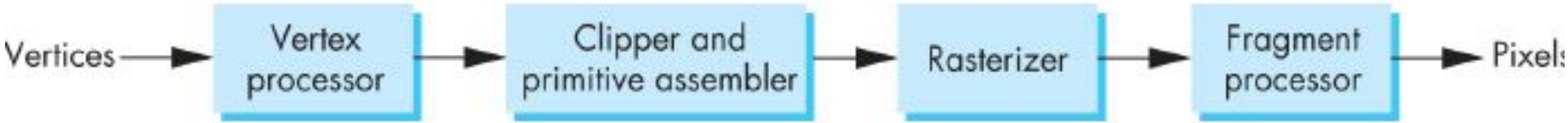


Definición de una aplicación gráfica

- Cómo podemos utilizar un modelo de una cámara sintética para crear aplicaciones en Computación Gráfica?
  - Application Programmer Interface (API)
- Solo es necesario especificar:
  - Objetos
  - Materiales
  - Observador
  - Luces
- Cómo lo implementamos?

Enfoque práctico

- Procesa un objeto a la vez, en el orden que son generados por la aplicación.
- Considera solo la iluminación local.
- Arquitectura del pipeline (flujo de procesamiento):



- Todos estos pasos pueden ser implementados en hardware (ej.: placa gráfica, GPU)

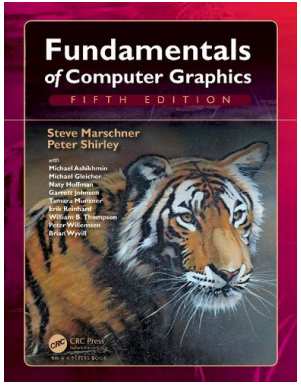
Especificación de objetos

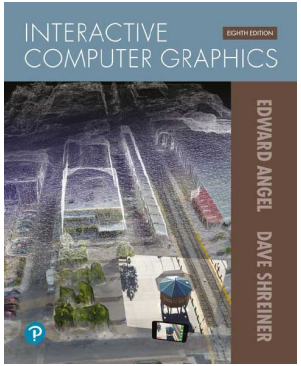
- La mayoría de APIs soportan un número limitado de primitivas
  - Puntos (objeto 0D)
  - Segmentos de recta (objeto 1D)
  - Polígonos (objetos 2D)
- Algunas curvas y superficies:
  - Superficies cuadráticas
  - Polígonos
- Son todos definidos a través de coordenadas en el espacio (vértices)

Referencias

- Shirley & Marschner
  - Fundamentals of Computer Graphics, 3rd Ed., CRC Press, 2022.

- Edward Stanley Angel, Dave Shreiner
  - Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL,8th Ed., Pearson, 2020





WebGL

- https://www.khronos.org/webgl/wiki/Demo_Repository
- http://webglsamples.org/
- http://davidwalsh.name/webgl-demo

Ejemplo OpenGL

- Almacenamiento de vértices en un arreglo:

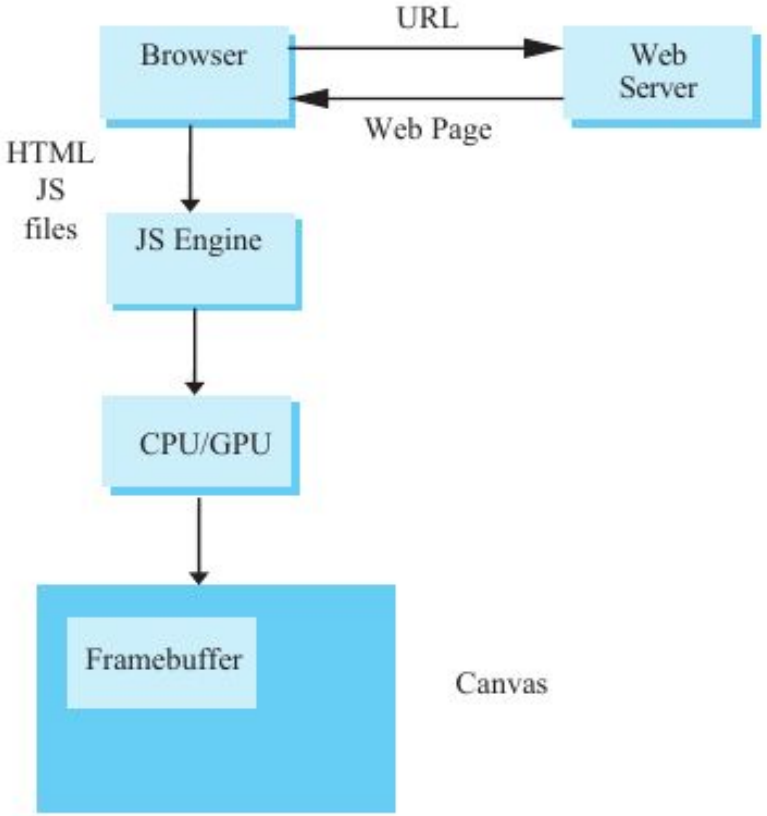
```
var points = [
 vec3(0.0, 0.0, 0.0),
 vec3(0.0, 1.0, 0.0),
 vec3(0.0, 0.0, 1.0),
];
```

```
var pts = new Float32Array(9);
pts[0]=0.0; pts[1]=0.0; pts[2]=0.0;
pts[3]=0.0; pts[4]=1.0; pts[5]=0.0;
pts[6]=0.0; pts[7]=0.0; pts[8]=1.0;
```

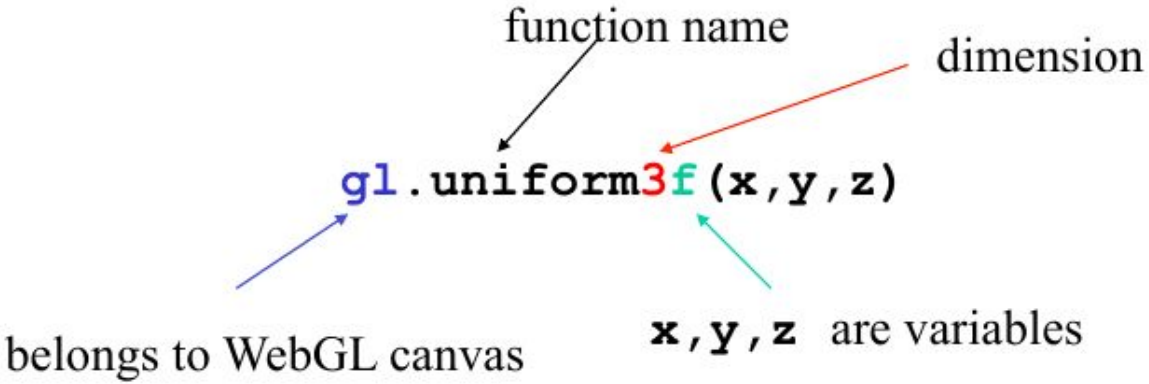
- Envía el arreglo al GPU
- Instruye al GPU para que lo renderiza como un triángulo.

WebGL

- Implementación de ES 2.0 (Embedded system) en JavaScript a través del elemento Canvas de HTML5
- Soportado por los browsers actuales



Formato WebGL




```
gl.uniform3fv(p)
```

p is an array

Constantes

- La mayoría de constantes están definidas en el objeto canvas
- En OpenGL de escritorio, son definidas en el archivo #include (e.g. gl.h)
- Ejemplos
  - OpenGL
    - glEnable(GL_DEPTH_TEST);
  - WebGL
    - gl.enable(gl.DEPTH_TEST);
    - gl.clear(gl.COLOR_BUFFER_BIT);

GLSL

- OpenGl Shading Language
  - Parecido con el lenguaje C
  - vectores y matrices (dimensiones 2, 3 y 4)
  - constructores estilo C++

WebGL (5 pasos)

1. Definición de la página web (archivo HTML)
    - Solicitar un archivo WebGL Canvas
    - Cargar cualquier archivo necesario
2. Definir los shaders (archivo HTML)
    - Puede ser en archivos separados (depende del browser)
3. Especificación y generación de datos (archivo JS)
4. Enviar los datos al GPU
5. Renderizar los datos (archivo JS)

Square.html

```
<!DOCTYPE html>
<html>
<head>
<script id="vertex-shader" type="x-shader/x-vertex">
attribute vec4 vPosition;
void main()
{
    gl_Position = vPosition;
}
</script>

<script id="fragment-shader" type="x-shader/x-fragment">
precision mediump float;
void main()
{
    gl_FragColor = vec4( 1.0, 1.0, 1.0, 1.0 );
}
</script>
```

Square.html (cont.)

```
<script type="text/javascript" src="../Common/webgl-
utils.js"></script>
<script type="text/javascript" src="../Common/
initShaders.js"></script>
<script type="text/javascript" src="../Common/MV.js"></
script>
<script type="text/javascript" src="square.js"></script>
</head>

<body>
<canvas id="gl-canvas" width="512" height="512">
Oops ... your browser doesn't support the HTML5 canvas
element
</canvas>
</body>
</html>
```

Square.js

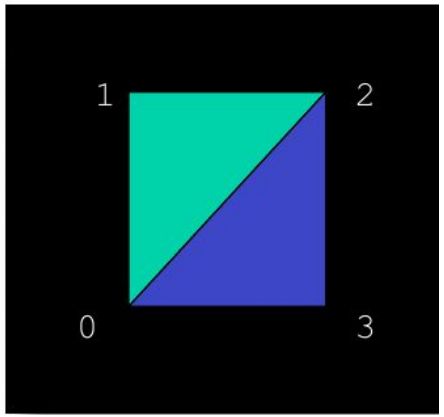
```
var gl;
var points;

window.onload = function init() {
var canvas = document.getElementById( "gl-canvas" );

    gl = WebGLUtils.setupWebGL( canvas );
    if ( !gl ) { alert( "WebGL isn't available" );
    }

    // Four Vertices
    var vertices = [
        vec2( -0.5, -0.5 ),
        vec2( -0.5,  0.5 ),
        vec2(  0.5,  0.5 ),
        vec2(  0.5, -0.5)
    ];
```

Square.js (cont.)

```
//  Configure WebGL
gl.viewport( 0, 0, canvas.width, canvas.height );
gl.clearColor( 0.0, 0.0, 0.0, 1.0 );

//  Load shaders and initialize attribute buffers
var program = initShaders( gl, "vertex-shader", "fragment-shader" );
gl.useProgram( program );

// Load the data into the GPU
var bufferId = gl.createBuffer();
gl.bindBuffer( gl.ARRAY_BUFFER, bufferId );
gl.bufferData( gl.ARRAY_BUFFER, flatten(vertices), gl.STATIC_DRAW );

// Associate out shader variables with our data buffer
var vPosition = gl.getAttribLocation( program, "vPosition" );
gl.vertexAttribPointer( vPosition, 2, gl.FLOAT, false, 0, 0 );
gl.enableVertexAttribArray( vPosition );
```

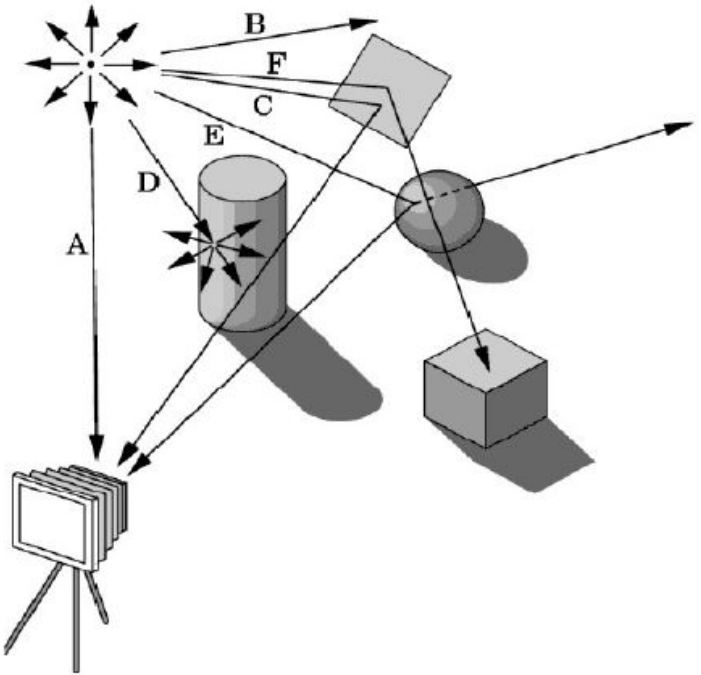
Square.js

```
    render();
};

function render() {
    gl.clear( gl.COLOR_BUFFER_BIT );
    gl.drawArrays( gl.TRIANGLE_FAN, 0, 4 );
}
```



Ray tracing

- Una manera de formar una imagen es seguir los rayos de luz para encontrar cuales de esos rayos entran en la lente de la cámara.



Raytracing