# Introducción a la Computación Gráfica

Ing. Gabriel Ávila, MSc.

# Librerías de graficación

GRAPHICS LIBRARIES

# Graphics APIs
## *Application Programming Interface*

***Colección de funciones que permiten dibujar imágenes y superficies tridimensionales en ventanas ubicadas en una pantalla.***

Parte fundamental en el uso de librerías de gráficos. En general, todo programa que trabaje con gráficos requieres dos APIs:

- **API Gráfico:** se encarga de la salida visual del programa.
- **API de Interfaz de Usuario:** Para capturar las señales enviadas por el usuario.

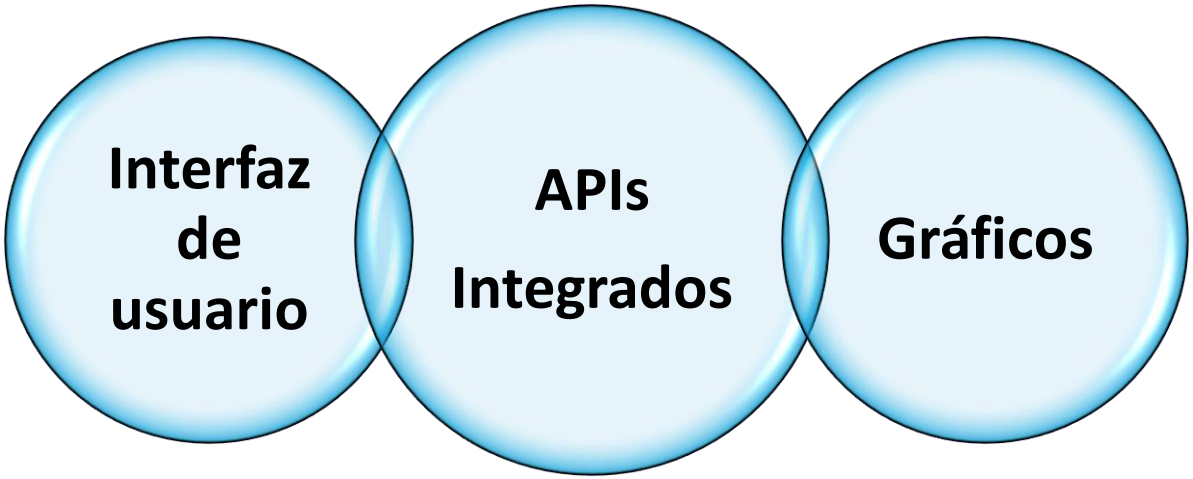
# Graphics Libraries

Apoyan el renderizado de gráficos por computador hacia el monitor, es decir, generan las imágenes en 2D como salida de una aplicación gráfica. Típicamente:

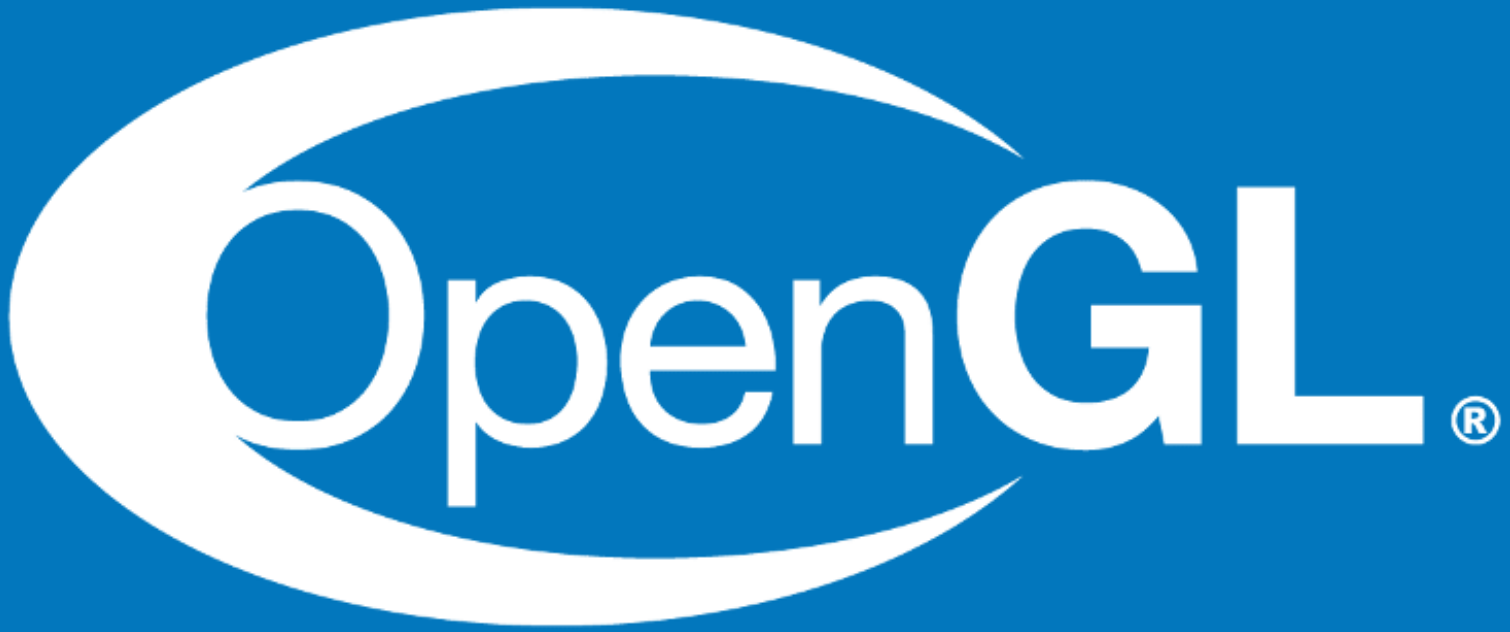
- Proveen versiones optimizadas de funciones o tareas típicas de renderizado, eliminando la necesidad de hacer esto desde la programación.
- Pueden trabajar sobre la CPU (en computadores embebidos) o aceleradas en hardware mediante la GPU (en computadores de escritorio).
- Su aplicación principal es en videojuegos y simulación.

# Graphics APIs
## *Application Programming Interface*

Los APIs pueden ser especializados en una de las áreas o integrados (caso Processing).

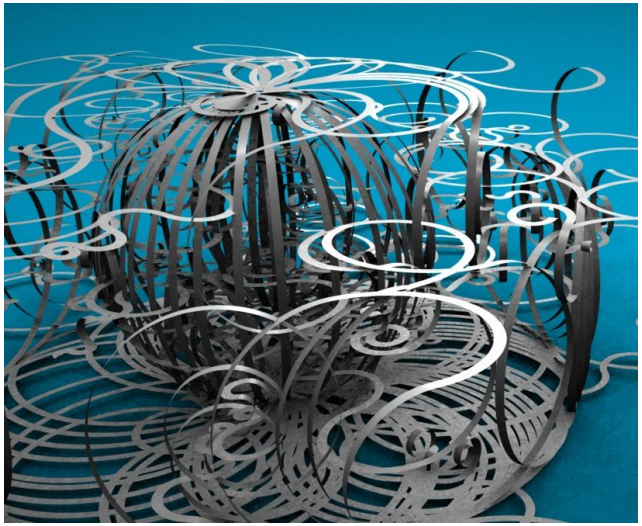


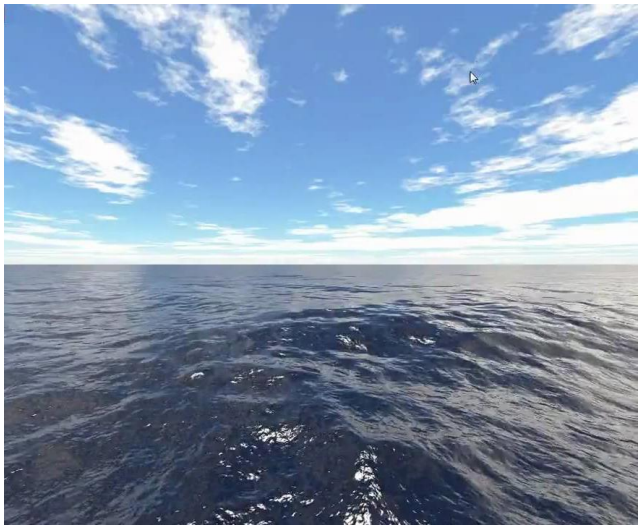

Los APIs gráficos incluyen **clases especificas**, optimizadas para generación de gráficos por computador, p.e. Vector2, Vector3, Color, Primitivas, etc.




# OpenGL

INTERFAZ ENTRE SOFTWARE Y EL HARDWARE GRÁFICO – HIGH LEVEL API





# OpenGL

***Es un API de alto nivel, que permite crear programas interactivos para producir imágenes a color de objetos tridimensionales en movimiento.***

Es posible controlar la tecnología de gráficos por computador, para la generación de **imágenes realistas**, o de **imágenes artísticas** que se alejen de la realidad.

Es independiente del hardware y del sistema operativo.

# GLUT: *OpenGL Utility Toolkit*

OpenGL es una librería encargada únicamente del tema gráfico (GL: Graphics Library).

**GLUT es un toolkit que facilita la interacción con OpenGL**, encargándose de tareas específicas del sistema, tales como: creación de ventanas y manejo de eventos de entrada (teclado, mouse).

FreeGLUT es la versión abierta y gratuita de GLUT (descarga).

# OpenGL: Estructura básica

Todas las aplicaciones en OpenGL tienen una estructura similar a la siguiente:

- Inicializar el ***estado*** asociado con la forma como los objetos deberán renderizarse.
- Especificar los objetos a renderizar.

*Renderizado: Proceso de creación de imágenes a partir de modelos.*


# Vulkan

MULTIPLATAFORMA – DISEÑADO POR KHRONOS GROUP – OPEN SOURCE

# Vulkan

*“Vulkan is a new generation graphics and compute API that provides high-efficiency, cross-platform access to modern GPUs used in a wide variety of devices from PCs and consoles to mobile phones and embedded platforms.”*

**Vulkan puede verse como el sucesor de OpenGL**, pues, aunque este último ha estado en desarrollo por más de 20 años, ha sido necesario un rediseño que permita un mejor desempeño, dadas las más recientes tecnologías.

Video comparación entre OpenGL y Vulkan

# Vulkan [2]

Se trata de un API de bajo nivel (thin-API), que se comunica de manera más directa con los drivers de la tarjeta gráfica. En este caso, se requiere aún más trabajo por el lado del desarrollador.

***The driver does as little as possible for the sake of much higher performance.***

El desarrollador debe conocer entonces las tuberías de trabajo (pipelines), así como dividir el trabajo en múltiples hilos, para lograr un uso más efectivo del hardware de la tarjeta gráfica.

***There are scenarios in which no difference in performance between OpenGL and Vulkan will be observed.***

# So….

***Should new graphics programmers be learning Vulkan instead of OpenGL?***

*I recommend starting with OpenGL, because it's easier to learn. The principles are the same, so* ***a lot of what you learn using OpenGL is transferable****.*

***OpenGL is much simpler to start with.*** *The concepts of OpenGL itself aren't too hard. Keep in mind there's a difference between learning OpenGL and learning computer graphics.*

# So….

## ***Should new graphics programmers be learning Vulkan instead of OpenGL?***

*I recommend starting with OpenGL, because it's easier to learn. The principles are the same, so* ***a lot of what you learn using OpenGL is transferable****.*

***OpenGL is much simpler to start with.*** *The concepts of OpenGL itself aren't too hard. Keep in mind there's a difference between learning OpenGL and learning computer graphics.*

*If you're getting started now, and* ***you want to do GPU work*** *(as opposed to always using a game engine such as Unity),* ***you should definitely start by learning Vulkan****. Maybe you should learn GL later too.*


# DirectX

MICROSOFT

# Direct X [3]

Desarrollado de forma exclusiva para Windows, contiene una serie de APIs que facilitan el desarrollo de aplicaciones multimedia para este sistema operativo.

La versión más reciente es DirectX 12.

*If you know DirectX, you can develop a DirectX app using native C++ and **HLSL** to take full advantage of graphics hardware.*

# HLSL

***HLSL** is the **High Level Shading Language** for DirectX.*

*Using HLSL, you can create C like programmable shaders for the Direct3D pipeline.*

*Data enters the graphics **pipeline as a stream of primitives** and is processed by the **shader stages**. The actual shader stages depend on the version of Direct3D, but certainly include the vertex, pixel and geometry stages. Other stages include the hull and domain shaders for tessellation, and the compute shader. These stages are completely programmable using the High Level Shading Language (HLSL).*

# METAL API

APPLE

# The METAL framework [4]

**Render advanced 3D graphics and perform data-parallel computations using graphics processors.**

*The Metal framework supports GPU-accelerated advanced 3D graphics rendering and data-parallel computation workloads.*

*Metal provides a modern and streamlined API for fine-grained, low-level control of the organization, processing, and submission of graphics and computation commands, as well as the management of the associated data and resources for these commands.*

*A primary goal of Metal is to minimize the CPU overhead incurred by executing GPU workloads.*


# WebGL

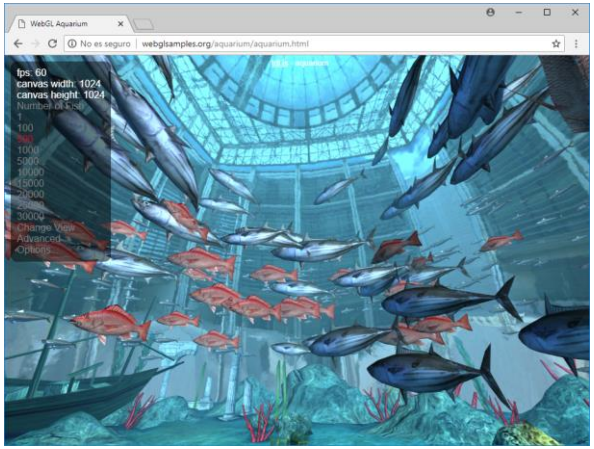
OpenGL para la web

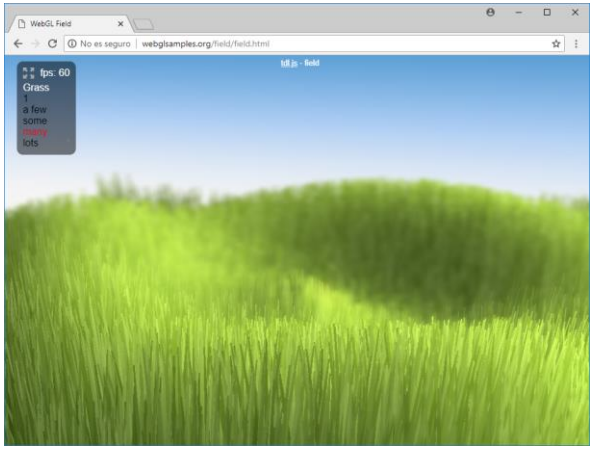
# WebGL – *Web Graphics Library*

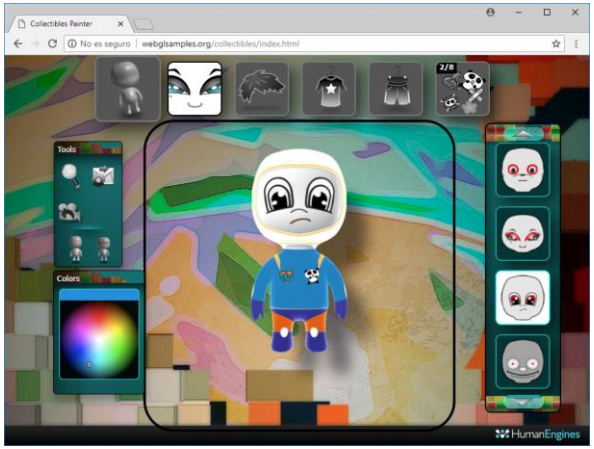
API desarrollado en Javascript que permite renderizar gráficos en 3D directamente en el navegador.

Se basa en OpenGL 3.0 y permite acceder desde el navegador a los recursos de GPU y aceleración de hardware para gráficos.

Es posible visualizar los diferentes elementos creados en WebGL en un canvas dentro de la página web.



# WebGL
# Revolución en la web

Existen muchos ejemplos de la utilización de WebGL en entornos web.

Basándose en WebGL han surgido múltiples librerías de alto nivel:

A-Frame(VR), BabylonJS, C3DL, CopperLicht, Curve3D, CubicVR, EnergizeGL, GammaJS, GLGE, GTW, JS3D, Kuda, O3D, OSG.JS, PhiloGL, PlayCanvas, Pre3d, SceneJS, SpiderGL, TDL, Three.js, X3DOM.

# Processing

HERRAMIENTA PARA ARTISTAS VISUALES

# Processing

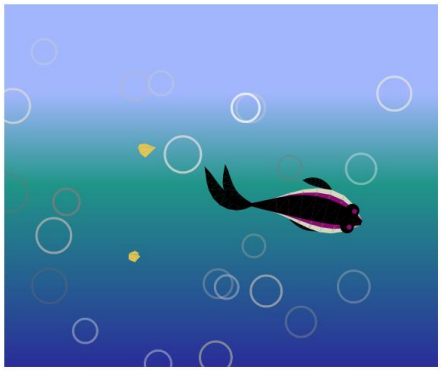
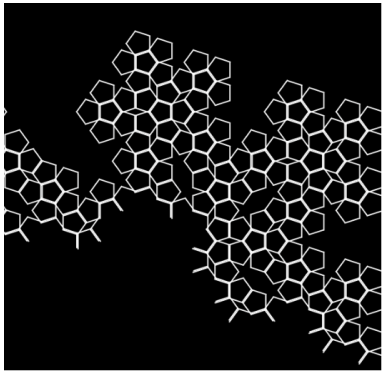
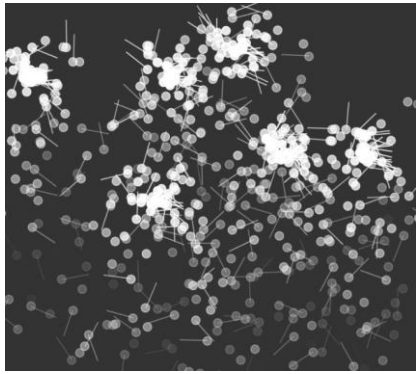
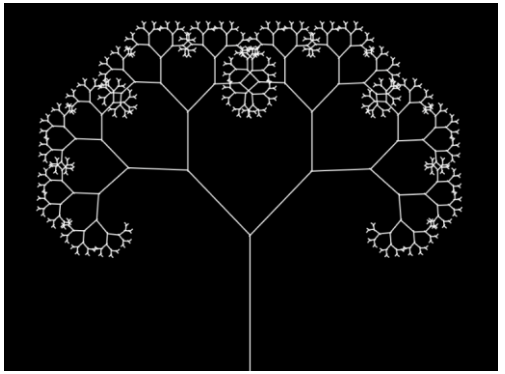
No es una librería, ni un API, sin embargo, ***integra un lenguaje de programación con un ambiente de desarrollo simplificado.***

Permite relacionar de manera práctica conceptos de programación con principios de formas visuales, movimiento e interacción.

***El lenguaje Processing está diseñado para generar y modificar imágenes***.

# ¿Qué se puede hacer?

Tareas interactivas, interfaces de usuario, en 2D y 3D, fractales, colisiones, animación, simulación, procesamiento de imágenes, sistemas de partículas, entre otros ejemplos.

Los programas hechos en Processing pueden exportarse para web fácilmente (tutorial), descargando la librería Processing.js


# Three.js

WEBGL….UN POCO MAS FACIL

# Three.js

Librería de Javascript para renderizado de gráficos interactivos en 2D y 3D, basada en WebGL.

Los elementos están dentro del <canvas> de una página HTML.

Soportado en casi todos los navegadores.

Incluye escenas, cámaras, geometría, carga de modelos 3D, luces, materiales, shaders, partículas, animación, matemáticas...

Editor 3D: https://threejs.org/editor/

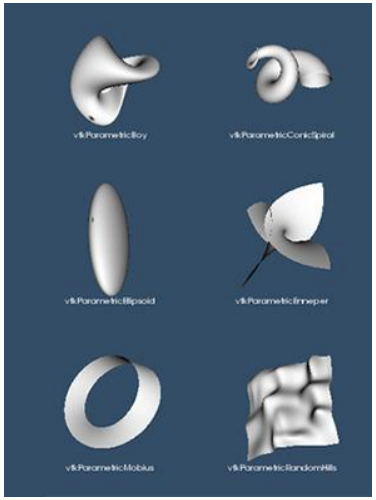
# Cómo iniciar

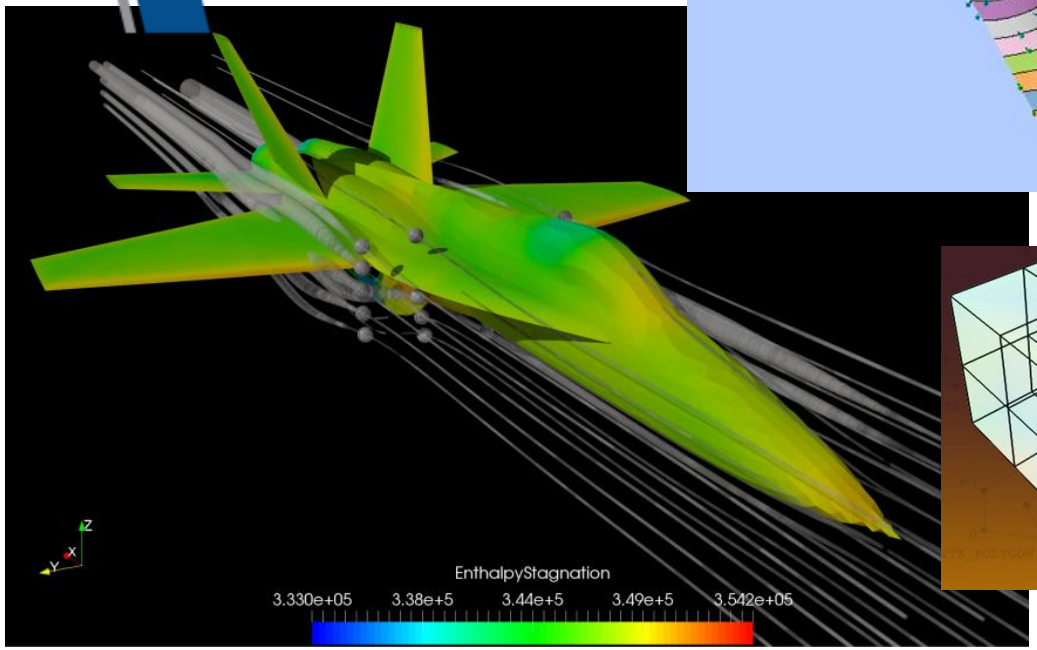
1. Descargar la librería Three.js
2. Crear un archivo llamado “index.html”
3. En la misma carpeta, crear una carpeta llamada “js” y poner dentro una copia del archivo three.min.js o three.js

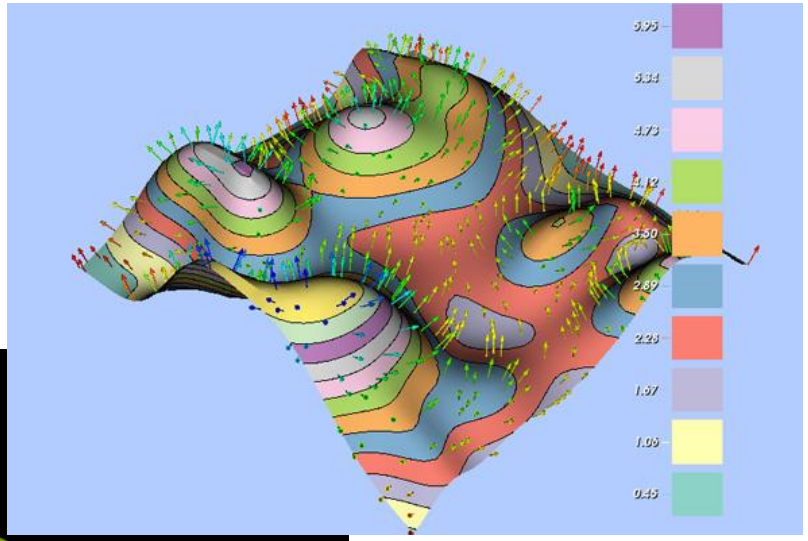
# Index.html

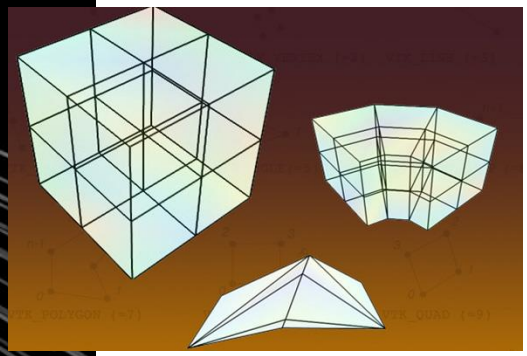
```
<!DOCTYPE html>
<html>
  <head>
    <title>Plantilla Base</title>
    <style>
      html, body { margin: 0; padding: 0; overflow: hidden; }
    </style>
  </head>
  <body>
    <script src="js/three.min.js"></script>
    <script>
        //Acá va el código de la aplicación
    </script>
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Plantilla Base</title>
    <style>
      html, body { margin: 0; padding: 0; overflow: hidden; }
    </style>
  </head>
  <body>
    <script src="js/three.min.js"></script>
    <script>
        var scene = new THREE.Scene();
        var aspect = window.innerWidth / window.innerHeight;
        var camera = new THREE.PerspectiveCamera( 75, aspect, 0.1, 1000);
        var renderer = new THREE.WebGLRenderer();
        renderer.setSize( window.innerWidth, window.innerHeight );
        document.body.appendChild( renderer.domElement );
    </script>
  </body>
</html>
```





# VTK

## VISUALIZATION TOOLKIT

# Visualization ToolKit – VTK

Sistema de software de código abierto, multiplataforma, para la creación de gráficos en 3D, procesamiento de imágenes y visualización. Su uso principal está orientado hacia la visualización de datos de tipo científico.

Consiste en varias librerías en C++ junto con algunas capas de interfaz hacia Java, Python, entre otros.

Su funcionamiento se basa en un pipeline que conecta diversos algoritmos.

Para utilizar VTK es necesario compilar el código utilizando CMAKE.

# Referencias

[1] Reas, C. y Fry, B. (2007). Processing: A Programming handbook for visual designers. Capítulo: “Structure 4: Objects I”. Pag: 395-404. TheMIT Press. 1st Ed.

[2] Vulkan API without secrets: Introduction to Vulkan. Tomado de: https://software.intel.com/en-us/articles/api-without-secrets-introduction-to-vulkan-preface

[3] Create your first Windows app using DirectX. Tomado de: https://docs.microsoft.com/en-us/windows/win32/direct3dgetstarted/building-your-first-directx-app

[4] Metal Programming Guide. Tomado de: https://developer.apple.com/library/archive/documentation/Miscellaneous/Conceptual/MetalProgrammingGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40014221