Course Overview

Introduction to Computer Graphics

Yu-Ting Wu

1

Course Information

- **Meeting time:** 09:10 - 12:00, Monday
- **Classroom:** 電1F-03
- **Instructor:** 吳昱霆 (Yu-Ting Wu)
- **Teaching assistants:** 劉勇佑
- **Course webpage:**
  - *https://kevincosner.github.io/courses/ICG2022/*
- **Grading:**
  - Assignments: 60% (4 programming homework)
  - Final exam: 35%
  - Participation: 5%

2

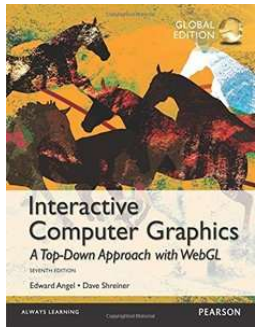
HW Late Policy HW

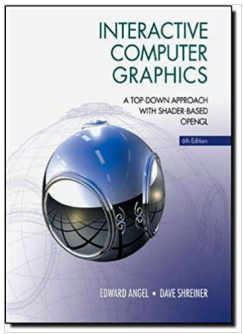
- One day 90%
- Two days 80%
- Three days 70%
- Four days 60%
- Five days+ 50%
- E.g., assume the deadline for the HW is 12/24 23:59 and you submit your HW on 12/25, you will get a 10% penalty
- You should **NOT** share codes with any **living creatures (if caught, you will get zero)**

3

Textbook (Optional)

- **Interactive Computer Graphics: A Top-Down Approach with WebGL (7th) / Shader-based OpenGL (6th)**





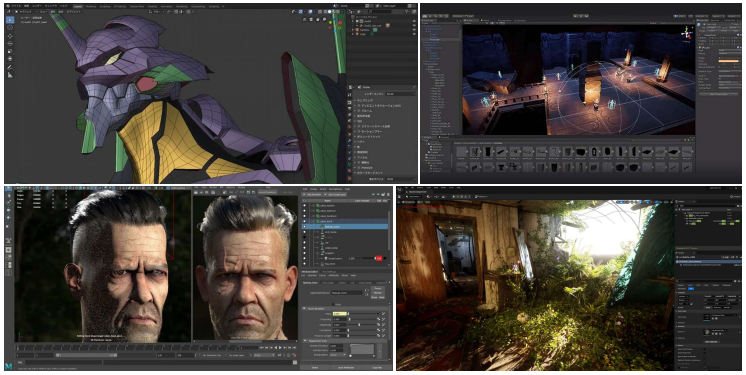
4

Prerequisites

- **C++ programming experience is required**
- Basic knowledge of **data structure** and **objected-oriented programming** is essential
- A not-too-bad computer for running your programs
  - Run the test program to validate your computer
- It is a **plus** if you
  - Are familiar with **linear algebra**
  - Have taken my course, **multimedia technology and applications**
  - Have experience in **image processing**

This course is NOT about using Editors

- **Instead, we learn the techniques behind the software!**

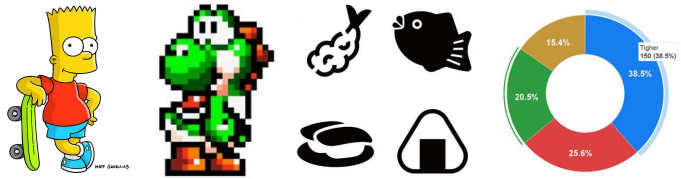


Introduction

What is Computer Graphics

- A sub-field of computer science that studies methods for **digitally synthesizing** and **manipulating** visual content (from *wiki*)
- Is concerned with all aspects of **producing pictures or images using a computer** (from our *textbook*)

These are All Computer Graphics

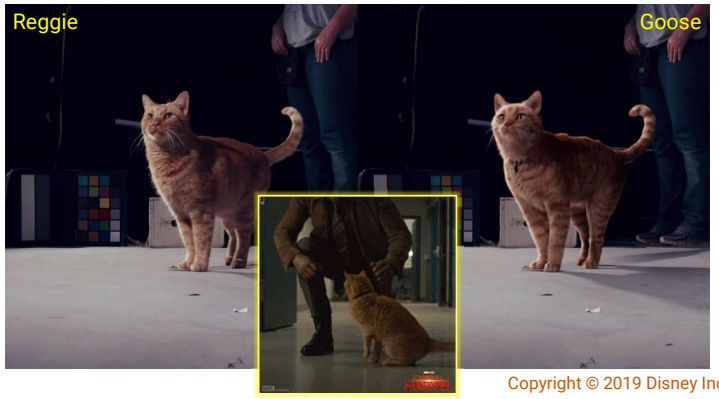


What we will focus on in this course

Goals of 3D Computer Graphics

- **Digitally synthesize** and **manipulate** a virtual world

Goals of 3D Computer Graphics (cont.)



Goals of 3D Computer Graphics (cont.)

13

A Quick Overview

14

How to Synthesize an Image

- Model geometry of the 3D objects (scene)

15

How to Synthesize an Image (cont.)

- Model materials of the 3D objects and simulate lighting

16

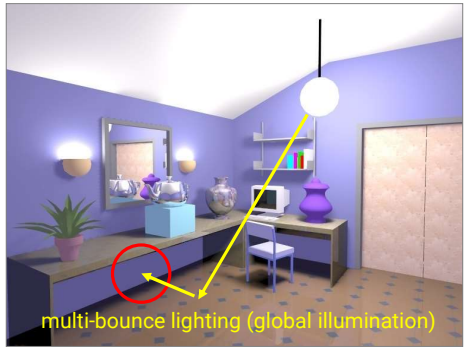
How to Synthesize an Image (cont.)

- Simulate more realistic materials and lighting phenomena

17

How to Synthesize an Image (cont.)

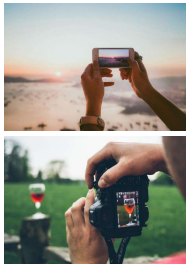
- Simulate more complex light paths

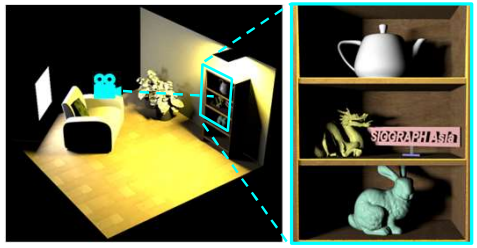


18

How to Synthesize an Image (cont.)

- Most displays are 2D, so we need to generate images from the 3D world
- Just like taking a picture with a camera in our daily lives
  - But with a **virtual camera** and a **virtual film**

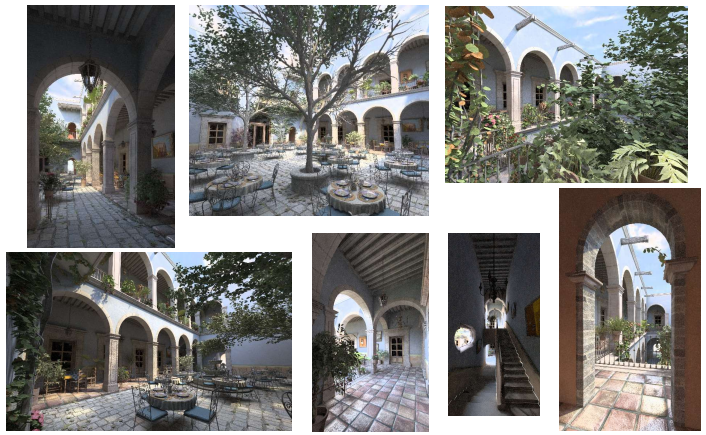




3D virtual world rendered image

19

How to Synthesize an Image (cont.)



20

Major Topics of Computer Graphics

Three Pillars of Computer Graphics

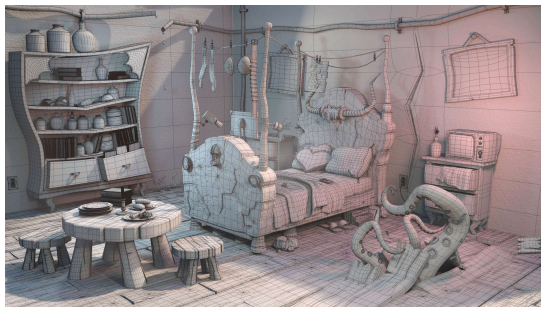
Modeling

Rendering

Animation

Modeling

- Build 3D representation of the virtual world
- The process of generating “data” in computer graphics



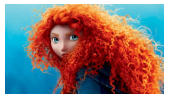
Modeling (cont.)

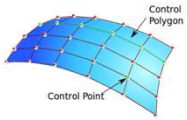
- World geometries are diverse!

Modeling (cont.)

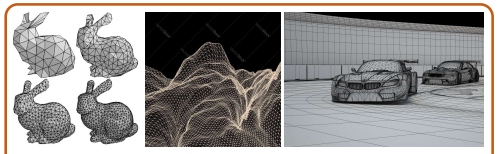
- World geometries are diverse!
- Using different representations including curves, surfaces, volumes

curves





patches

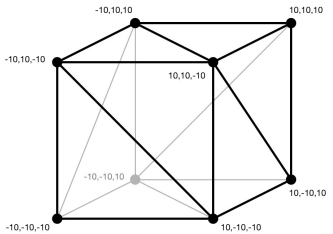


meshes (by polygons, usually triangles)

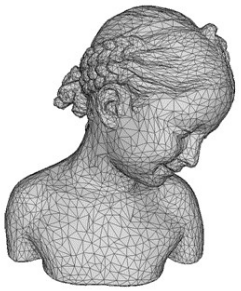
volume data

Modeling (cont.)

- **Triangle mesh** is the most popular representation
- Define the **positions** and **adjacencies** of **vertices**



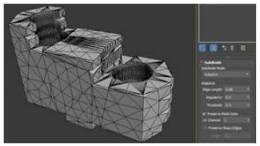
12 triangles



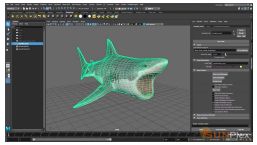
10K triangles

Modeling (cont.)

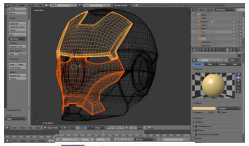
- 3D models are usually obtained by professional manipulations in 3D modeling tools



Blender



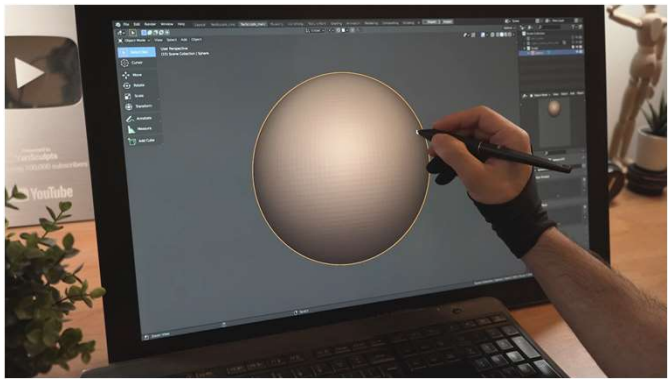
Maya



AUTODESK 3DS MAX

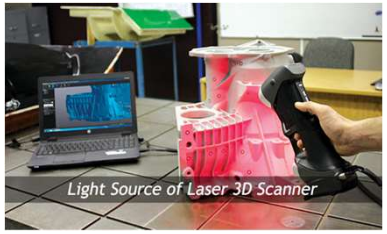
Modeling (cont.)

- Example: create a 3D character model in Blender [Link]



Modeling (cont.)

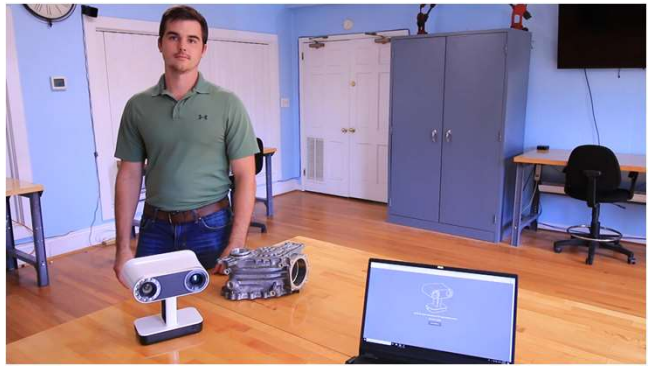
- Can also be captured and reconstructed from the real-world geometries



29

Modeling (cont.)

- Example: 3D scanner [Link]



30

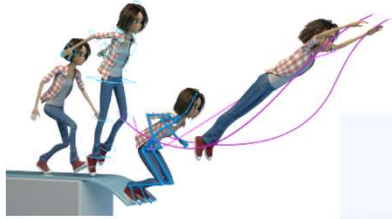
Modeling (cont.)

- Example: create geometry from a set of photos [Link]

31

Animation

- Describe (or simulate) how the geometry changes / moves over time



32

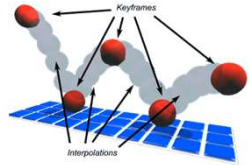
Animation (cont.)

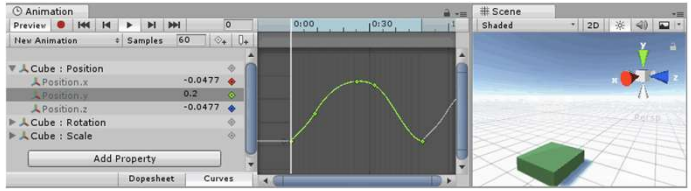
- Animations are usually expected to be physically-based

33

Animation (cont.)

- Keyframe-based animations





34

Animation (cont.)

- Inverse Kinematics (IK) [Link]

35

Animation (cont.)

- Motion capture

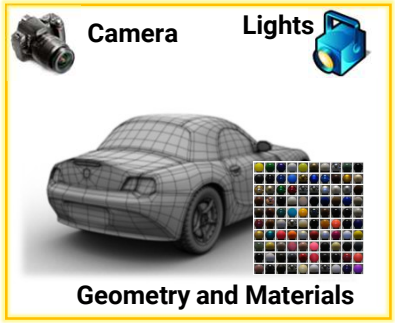
36

Animation (cont.)

- The Making of Resident Evil Village [Link]

Rendering

- Simulate the appearance of virtual objects and synthesize the final image

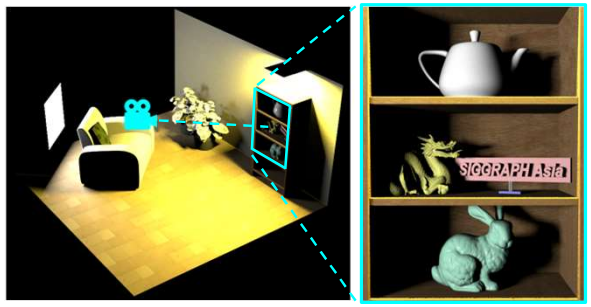


input: 3D description of a scene

output: 2D synthetic image

Rendering (cont.)

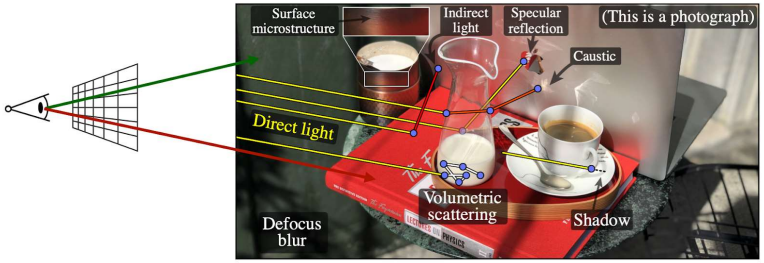
- Simulate the appearance of virtual objects and synthesize the final image



3D virtual world rendered image

Rendering (cont.)

- **Physically-based rendering**
  - Uses **physics** and **math** to simulate the interaction between matter and lights, **realism** is the primary goal



Rendering (cont.)

- Non-photo-realistic rendering

41

Rendering (cont.)

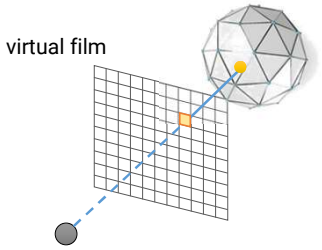
- Introduction to Rendering by Pixar [Link]

42

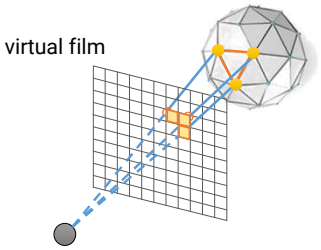
Rendering (cont.)

- Two ways for generating synthetic images

Ray tracing



Rasterization



43

Rendering (cont.)

- We will focus on the **rasterization-based** rendering because
  - It is widely used in **interactive computer graphics** and has more applications in our daily lives
  - It is more commonly used in Taiwan's industry
    - Thus, can be a great help to your future jobs
  - It takes less time to generate an image
- However, the knowledge is the same and we will also give an overview of ray tracing at the end of this course

44

Case Study: Animation Production Pipeline

Animation Production Pipeline


story

text treatment


storyboard


voice


storyreel


look and feel

Animation Production Pipeline (cont.)


modeling / articulation


layout


animation

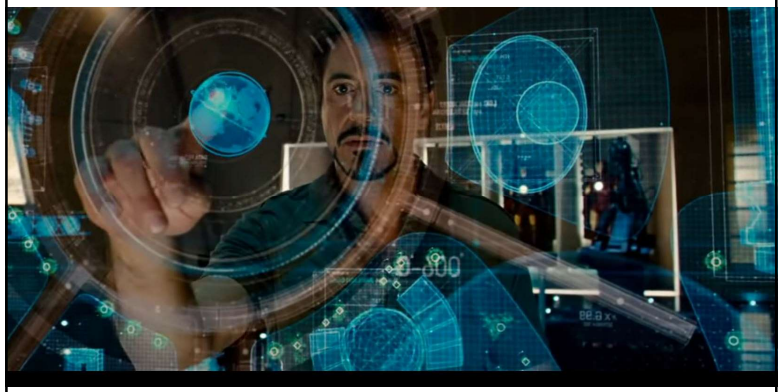

shading / lighting


rendering


final touch

Other Correlated Fields of Computer Graphics

Human-Computer Interaction (HCI)



49

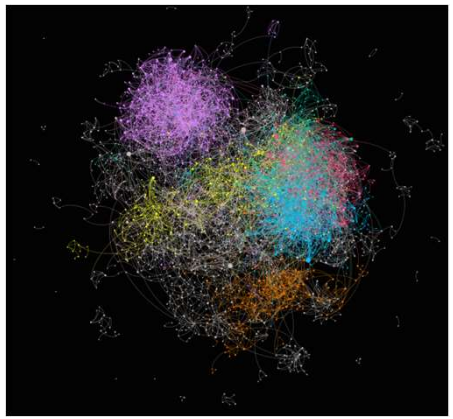
Virtual Reality (VR)

50

Augmented and Mixed Reality (AR, MR)

51

Visualization



52

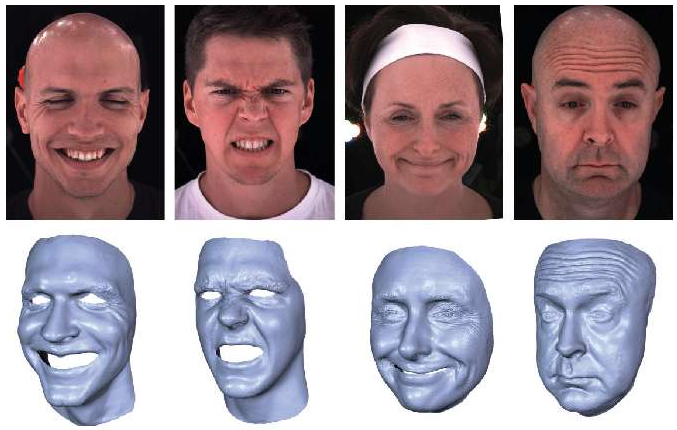
Image Processing

53

Computational Photography

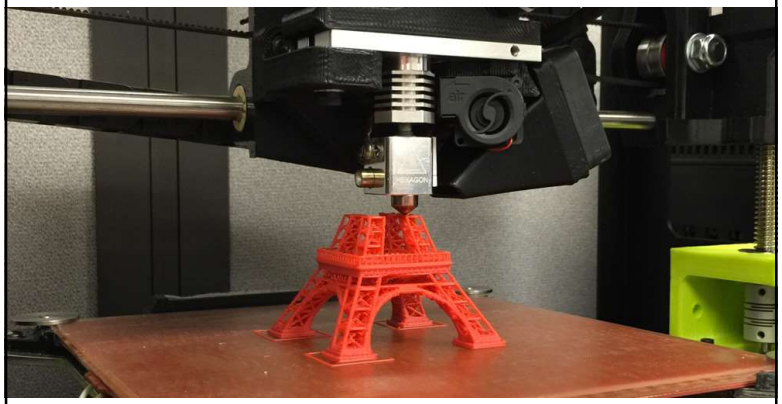
54

3D Reconstruction



55

Fabrication



56

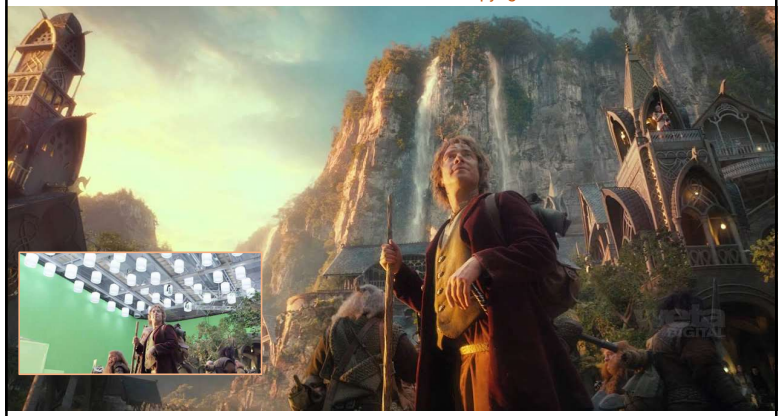
Applications of Computer Graphics

57

Video Games

58

Digital Visual Effects (VFX)



59

Featured Animations

60

Cartoons

61

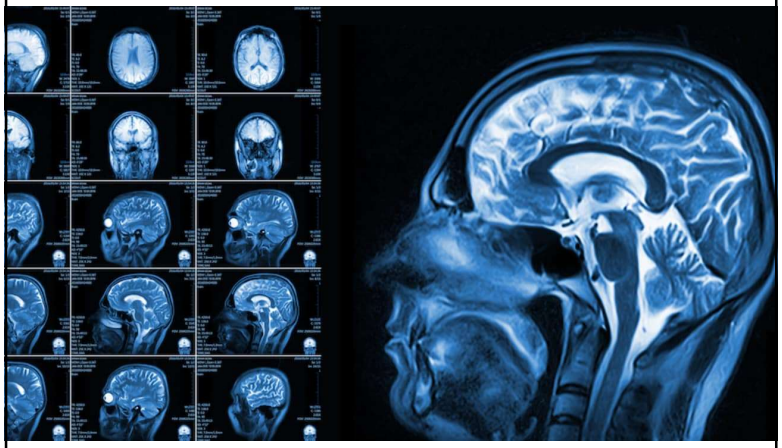
Computer-Aided Design

62

Simulation

63

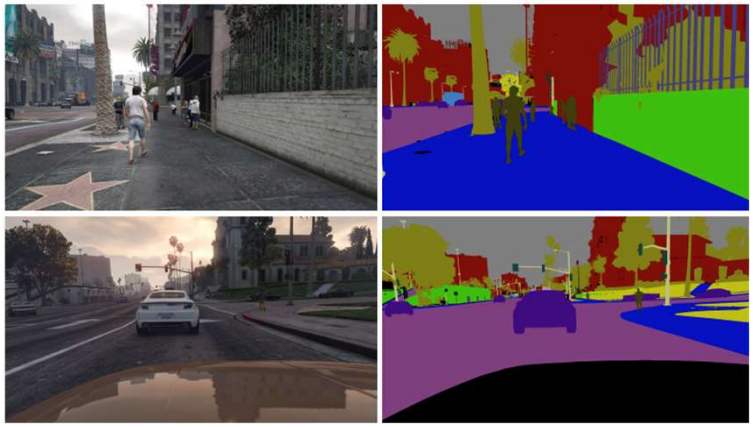
Medical Imaging



64

Machine (Deep) Learning

GTA5 Database



65

Graphics Programming

66

Graphics Programming

- For rasterization-based graphics, programs are usually implemented with graphics **application programming interface (API)** and **shader programs**
- Common choices are
  - OpenGL + GLSL (OpenGL shading language)
    - OpenGL ES
    - WebGL
  - DirectX + HLSL (High-level shading language)
  - Vulkan + GLSL/HLSL

67

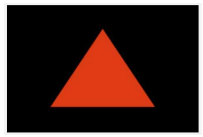
OpenGL

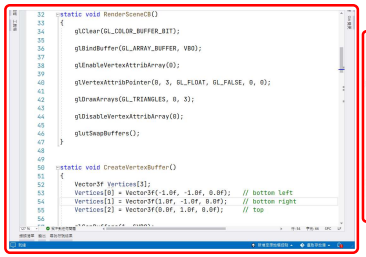
- A **cross-platform** API for rendering 2D and 3D vector graphics, typically used to interact with a graphics processing unit (GPU)
- Developed by Silicon Graphics Inc. (SGI) in 1991
- Managed by a non-profit technology consortium **Khronos Group** after 2006

68

OpenGL + GLSL

- A simple program to draw a triangle on the screen
  - 176 lines of C++ code and 16 lines of shader code





```
#version 330 core

layout (location = 0) in vec3 Position;

void main()
{
    gl_Position = vec4(0.5 * Position.x, 0.5 * Position.y, Position.z, 1.0);
}


#version 330 core

out vec4 FragColor;

void main()
{
    FragColor = vec4(1.0, 0.0, 0.0, 0.0);
}
```


Why not Teaching Vulkan in this Course?

- A simple program to draw a triangle on the screen
  - **457** lines of C++ code

```
    void CreateSwapChain();
    void CreateCommandBuffer();
    void CreateRenderPass();
    void CreateFramebuffer();
    void CreateShaders();
    void CreatePipeline();
    void RecordCommandBuffers();
    void RenderScene();

    std::string m_appName;
    VulkanWindowControl* m_pWindowControl;
    OgldevVulkanCore m_core;
    std::vector<VkImage> m_images;
    VkSwapchainKHR m_swapChainKHR;
    VkQueue m_queue;
    std::vector<VkCommandBuffer> m_cmdBufs;
    VkCommandPool m_cmdBufPool;
    std::vector<VkImageView> m_views;
    VkRenderPass m_renderPass;
    std::vector<VkFramebuffer> m_fbs;
    VkShaderModule m_vsModule;
    VkShaderModule m_fsModule;
    VkPipeline m_pipeline;
};
```

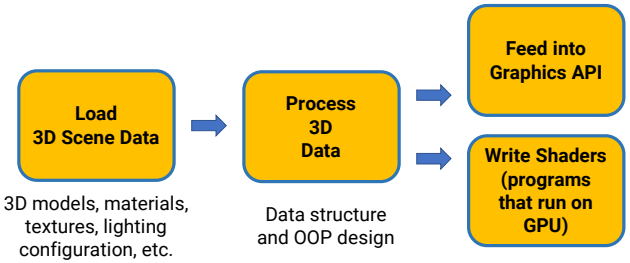
...

```
    rastCreateInfo.polygonMode = VK_POLYGON_MODE_FILL;
    rastCreateInfo.cullMode = VK_CULL_MODE_BACK_BIT;
    rastCreateInfo.frontFace = VK_FRONT_FACE_COUNTER_CLOCKWISE;
    rastCreateInfo.lineWidth = 1.0f;

    VkPipelineMultisampleStateCreateInfo pipelineMSCreateInfo = {};
    pipelineMSCreateInfo.sType = VK_STRUCTURE_TYPE_PIPELINE_MULTISAMPLE_STATE_

    VkPipelineColorBlendAttachmentState blendAttachState = {};
    blendAttachState.colorWriteMask = 0xf;

    VkPipelineColorBlendStateCreateInfo blendCreateInfo = {};
    blendCreateInfo.sType = VK_STRUCTURE_TYPE_PIPELINE_COLOR_BLEND_STATE_CREAT
    blendCreateInfo.logicOp = VK_LOGIC_OP_COPY;
    blendCreateInfo.attachmentCount = 1;
    blendCreateInfo.pAttachments = &blendAttachState;

    VkGraphicsPipelineCreateInfo pipelineInfo = {};
    pipelineInfo.sType = VK_STRUCTURE_TYPE_GRAPHICS_PIPELINE_CREATE_INFO;
    pipelineInfo.stageCount = ARRAY_SIZE_IN_ELEMENTS(shaderStageCreateInfo);
    pipelineInfo.pStages = &shaderStageCreateInfo[0];
    pipelineInfo.pVertexInputState = &vertexInputInfo;
    pipelineInfo.pInputAssemblyState = &pipelineIACreateInfo;
    pipelineInfo.pViewportState = &vpCreateInfo;
    pipelineInfo.pRasterizationState = &rastCreateInfo;
```


Scope of This Course

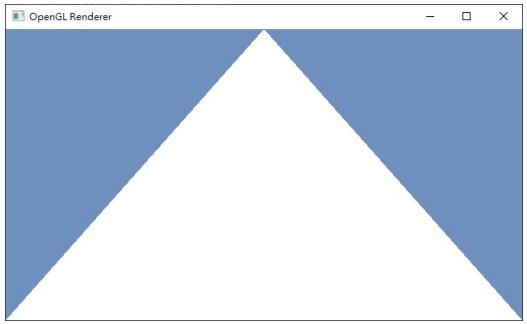
Goals

- Introduce the basic concepts of **3D** computer graphics, especially in **modeling** and **rendering**
- Introduce how to program with **graphics API (OpenGL)**



Goals (cont.)

- We will start by teaching how to render a single triangle



Goals (cont.)

- And at the end of this course, your program can render this scene

Topics We Plan to Cover

Basic

- Raster images
- Colors
- Geometry representation
- Transformations
- Camera
- GPU graphics pipeline
- Shading
- Textures
- Transparency

Advanced

- Terrain
- Shadows
- Deferred shading
- Ray tracing
- Advanced rendering techniques

Any Questions?