

Introduction to Computer Graphics 2022



Course Overview

Introduction to Computer Graphics
Yu-Ting Wu

1

Introduction to Computer Graphics 2022

Course Information

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

2

Introduction to Computer Graphics 2022

HW Late Policy HW

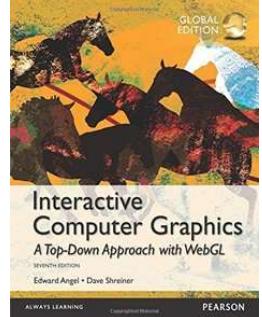
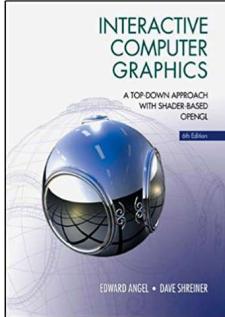
- One day 90%
- Two days 80%
- Three days 70%
- Four days 60%
- Five days+ 50%
- E.g., assume the deadline for the HW is 12/24 23:59 and you submit your HW on 12/25, you will get a 10% penalty
- You should **NOT** share codes with any **living creatures** (**if caught, you will get zero**)

3

Introduction to Computer Graphics 2022

Textbook (Optional)

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

4

1



5

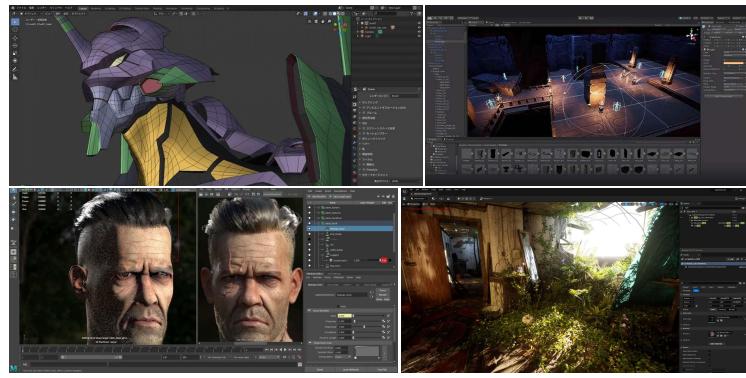
Prerequisites

- 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**

6

This course is **NOT** about using Editors

- Instead, we learn the techniques behind the software!



7

Introduction

8

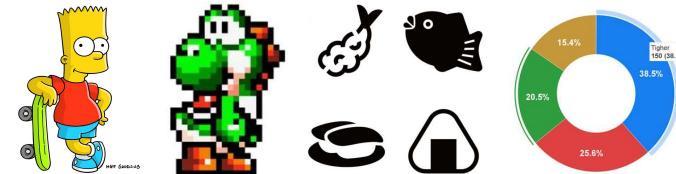
2

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*)

9

These are All Computer Graphics



What we will focus on in this course

10

10

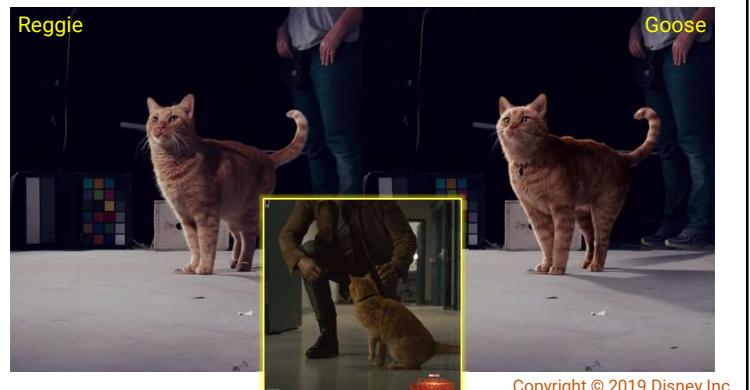
Goals of 3D Computer Graphics

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



11

Goals of 3D Computer Graphics (cont.)



Copyright © 2019 Disney Inc.

12

12

11

3

Goals of 3D Computer Graphics (cont.)



Copyright © 2018 Universal Studios

13

A Quick Overview

14

13

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

15

16

How to Synthesize an Image (cont.)

- Simulate more realistic materials and lighting phenomena

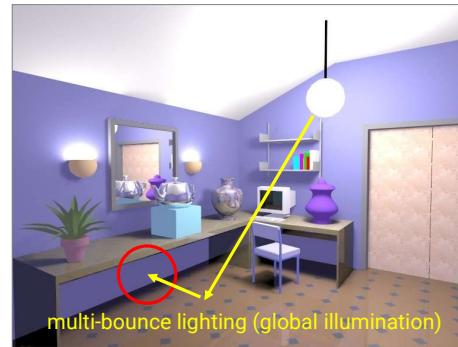


17

17

How to Synthesize an Image (cont.)

- Simulate more complex light paths

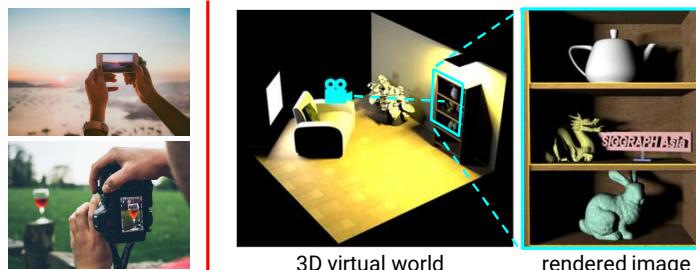


18

18

How to Synthesize an Image (cont.)

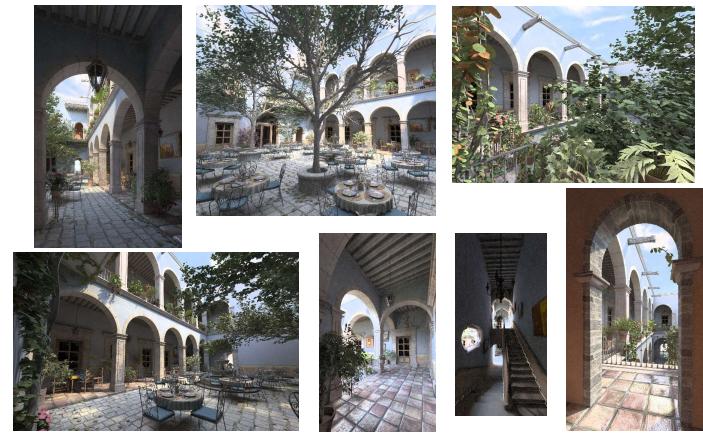
- Most displays are 2D, so we need to generate images from the 3D world
- Just like taking a picture with a camera in our daily lives
 - But with a **virtual camera** and a **virtual film**



19

19

How to Synthesize an Image (cont.)



20

20

5

Major Topics of Computer Graphics

21

21

Three Pillars of Computer Graphics



Modeling



Rendering



Animation

22

22

Modeling

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

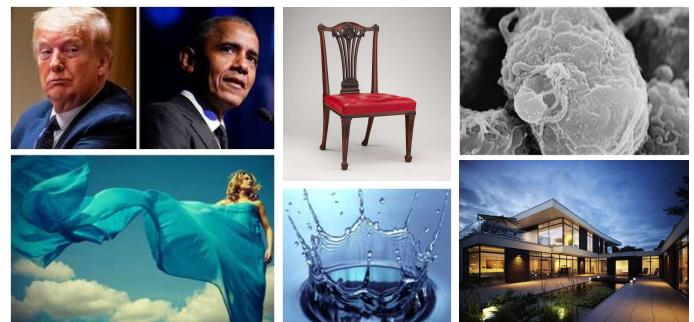


23

23

Modeling (cont.)

- World geometries are diverse!



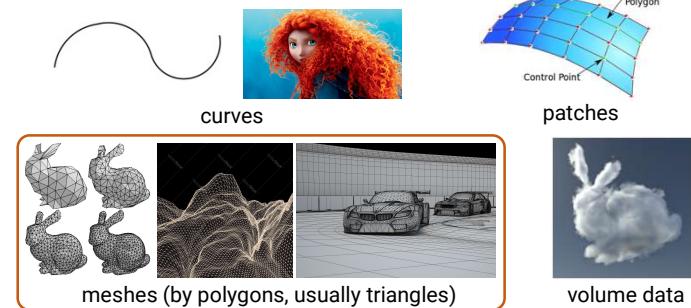
24

24

6

Modeling (cont.)

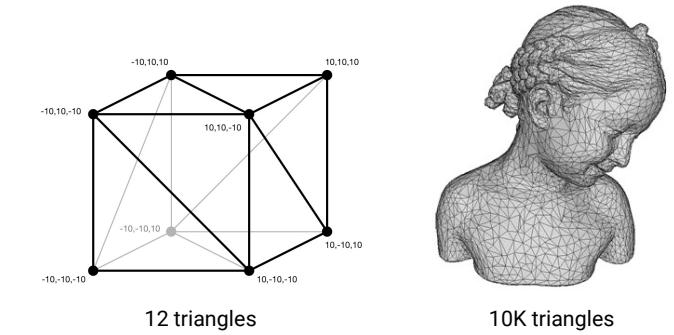
- World geometries are diverse!
- Using different representations including curves, surfaces, volumes



25

Modeling (cont.)

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



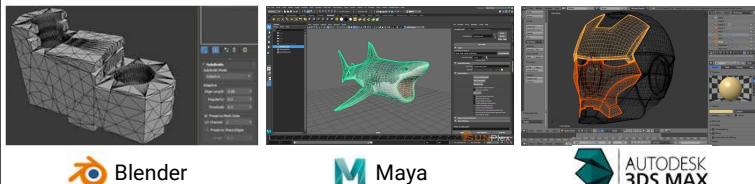
26

25

26

Modeling (cont.)

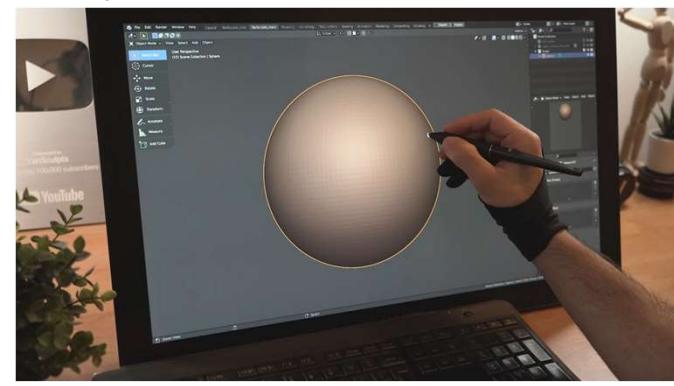
- 3D models are usually obtained by professional manipulations in 3D modeling tools



27

Modeling (cont.)

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



28

27

28

Modeling (cont.)

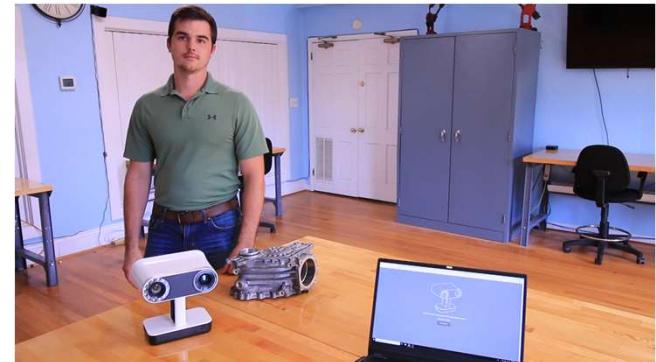
- Can also be captured and reconstructed from the real-world geometries



29

Modeling (cont.)

- Example: 3D scanner [[Link](#)]



30

29

30

Modeling (cont.)

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

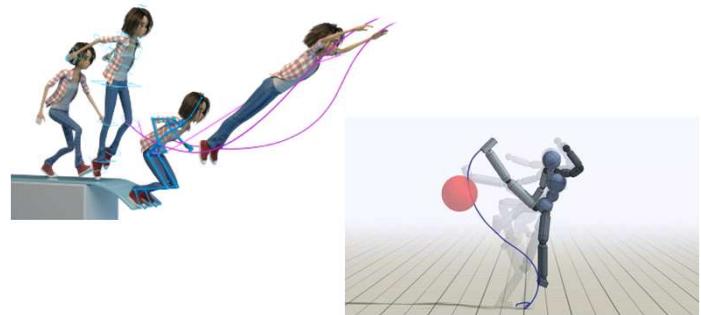


31

31

Animation

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



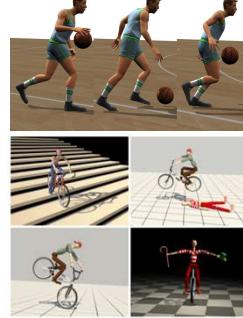
32

32

8

Animation (cont.)

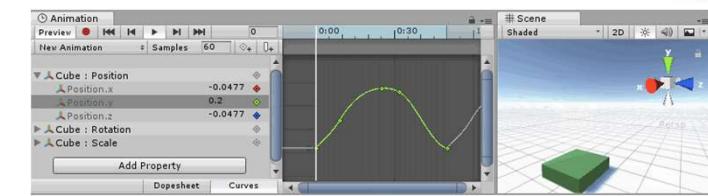
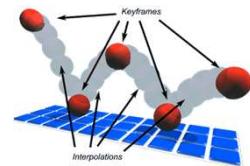
- Animations are usually expected to be physically-based



33

Animation (cont.)

- Keyframe-based animations



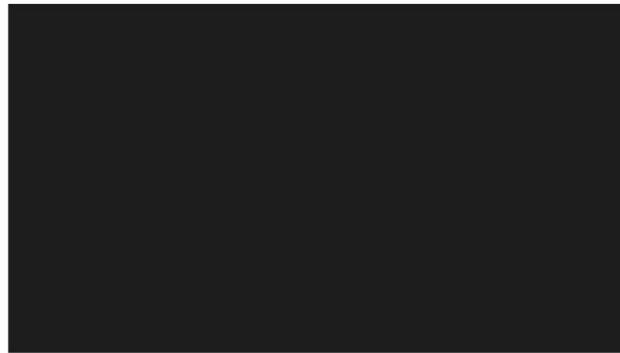
34

34

33

Animation (cont.)

- Inverse Kinematics (IK) [[Link](#)]

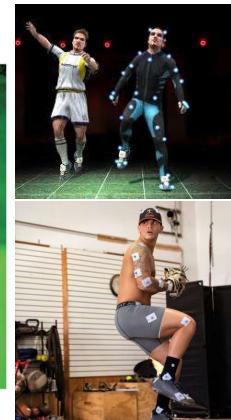


35

35

Animation (cont.)

- Motion capture



36

36

9

Animation (cont.)

- The Making of Resident Evil Village [\[Link\]](#)

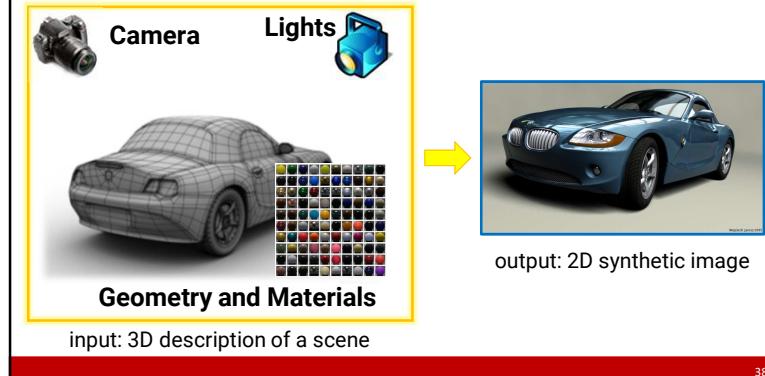


37

37

Rendering

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

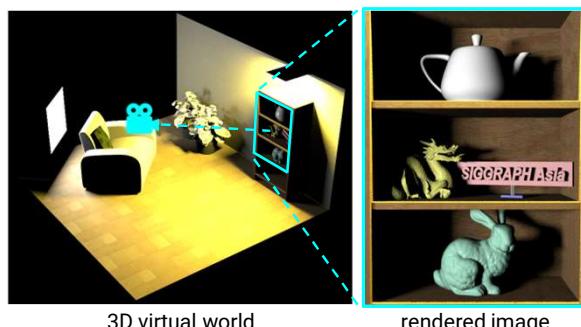


38

38

Rendering (cont.)

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

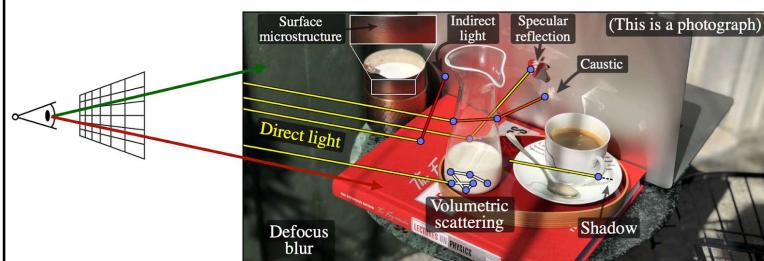


39

39

Rendering (cont.)

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



40

40

10

Rendering (cont.)

- Non-photo-realistic rendering

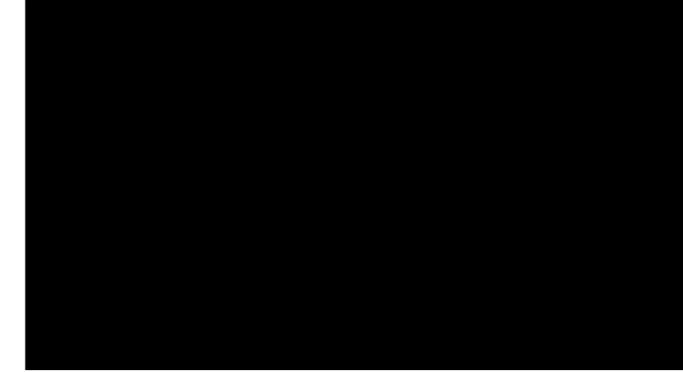
Copyright © 2020 miHoYo Inc.



41

Rendering (cont.)

- Introduction to Rendering by Pixar [[Link](#)]



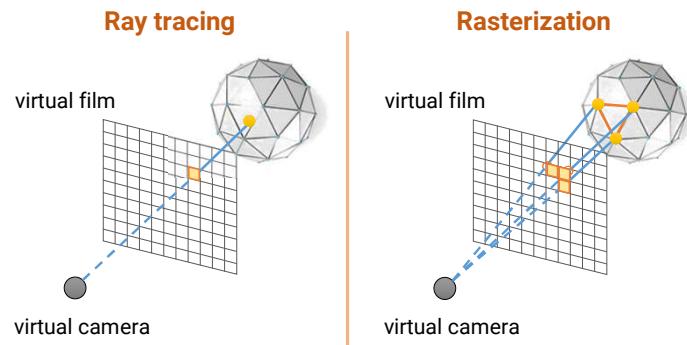
42

41

42

Rendering (cont.)

- Two ways for generating synthetic images



43

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

44

11

Case Study: Animation Production Pipeline

45

45

Animation Production Pipeline



46

46

Animation Production Pipeline (cont.)



47

47

Other Correlated Fields of Computer Graphics

48

48

12

Human-Computer Interaction (HCI)



Copyright © 2008 Disney Inc.

49

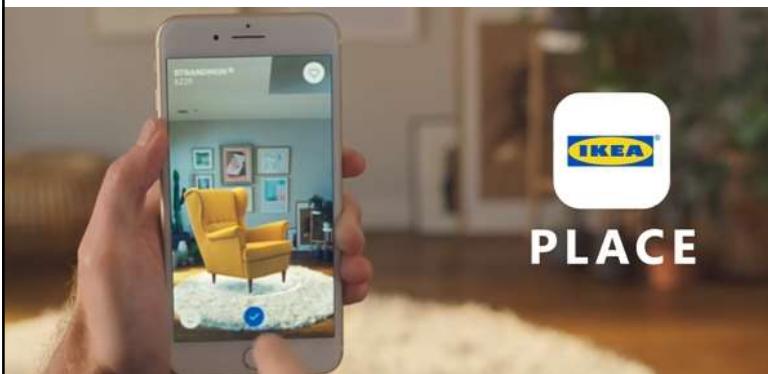
Virtual Reality (VR)



50

50

Augmented and Mixed Reality (AR, MR)

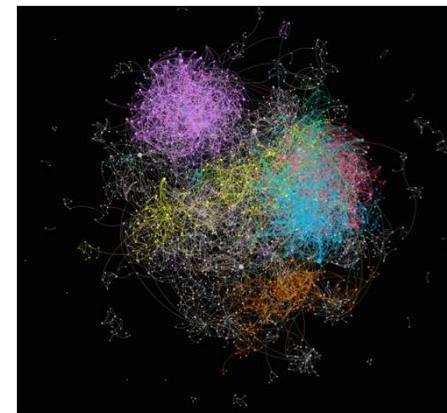


Copyright © IKEA Inc.

51

51

Visualization



52

52

13

Image Processing



53

53

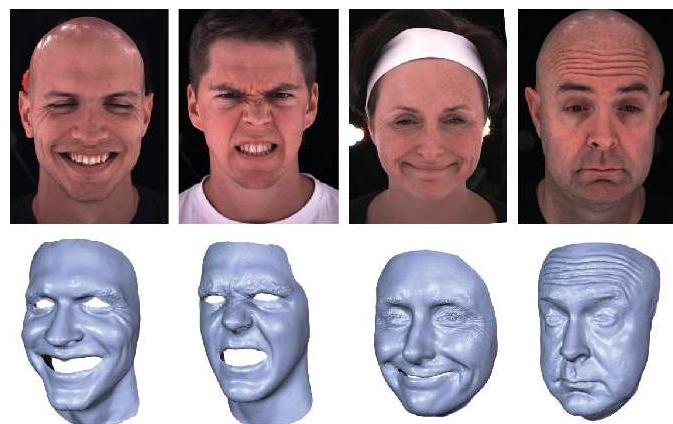
Computational Photography



54

54

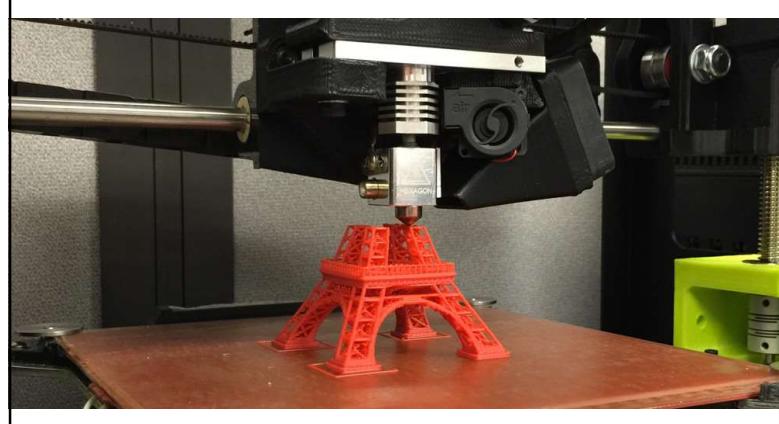
3D Reconstruction



55

55

Fabrication



56

56

14

Applications of Computer Graphics

57

57

Video Games

Copyright © 2020 SQUARE ENIX Inc.



58

58

Digital Visual Effects (VFX)

Copyright © 2012 Warner Bros. Pictures



59

59

Featured Animations

Copyright © 2022 Disney Inc.



60

60

15

Cartoons

Copyright © 20th Century Fox Television



61

Computer-Aided Design



62

61

62

Simulation



63

Medical Imaging



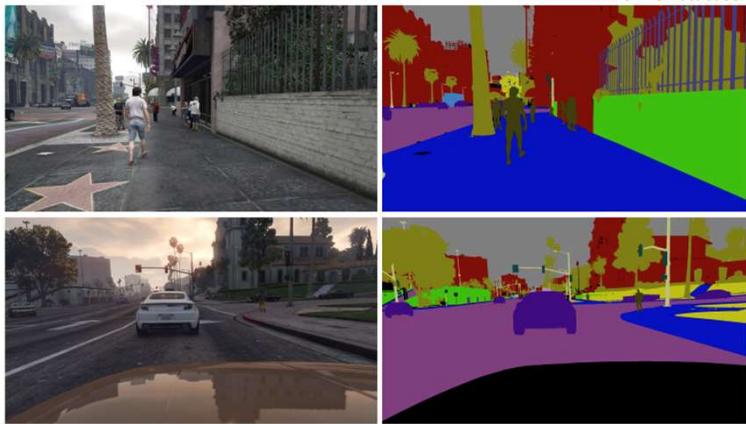
64

63

64

16

Machine (Deep) Learning



65

Graphics Programming

66

65

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

67

68

17

OpenGL + GLSL

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



```
32     static void RenderScene()
33     {
34         glClear(GL_COLOR_BUFFER_BIT);
35
36         glBindBuffer(GL_ARRAY_BUFFER, VBO);
37
38         glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);
39
40         glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
41
42         glBindArrayBuffer(GL_ARRAY_BUFFER, 0);
43
44         glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);
45
46         glUseProgram(0);
47
48         glDeleteBuffers(1, &VBO);
49
50         glDeleteProgram(shaderProgram);
51
52         CreateVertexBuffer();
53
54         Vertices[0] = Vector3f(-1.0f, -1.0f, 0.0f); // bottom left
55         Vertices[1] = Vector3f(1.0f, -1.0f, 0.0f); // bottom right
56         Vertices[2] = Vector3f(0.0f, 1.0f, 0.0f); // top
57
58         Vertices[3] = Vector3f(-1.0f, 1.0f, 0.0f); // top left
59         Vertices[4] = Vector3f(1.0f, 1.0f, 0.0f); // top right
60         Vertices[5] = Vector3f(0.0f, -1.0f, 0.0f); // bottom
61
62         // ...
63     }
64 }
```

```
#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);  
}
```

6

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;
OglDepthBuffer m_depthBuffer;
std::vector<VkImage> m_images;
VkSwapchainKHR m_swapChainKHR;
VKQueue m_queue;
std::vector<VKCommandBuffer> m_cmdBufs;
VKCommandPool m_vkCmdPool;
std::vector<VKImageView> m_views;
VKRenderPass m_renderPass;
std::vector<VKFramebuffer> m_fbfs;
VKShaderModule m_vsShaderModule;
VKShaderModule m_fsShaderModule;
VKPipeline m_pipeline;
};


```

```

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

    VkPipelineMultiSampledStateCreateInfo pipelineMSCreateInfo = {};
    pipelineMSCreateInfo.sType = VK_STRUCTURE_TYPE_PIPELINE_MULTISAMPLE_STATE_CREATE_INFO;
    pipelineMSCreateInfo.blendAttachmentState = &blendAttachmentState;
    blendAttachmentState.colorWriteMask = 0xff;

    VkPipelineColorBlendStateCreateInfo blendCreateInfo = {};
    blendCreateInfo.sType = VK_STRUCTURE_TYPE_PIPELINE_COLOR_BLEND_STATE_CREATE_INFO;
    blendCreateInfo.logicOp = VK_LOGIC_OP_NO_OP;
    blendCreateInfo.attachmentCount = 1;
    blendCreateInfo.pAttachments = &blendAttachmentState;

    ...
}

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 = &inputAssemblyInfo;
pipelineInfo.pViewportState = &viewportInfo;
pipelineInfo.pRasterizationState = &rasterCreateInfo;
pipelineInfo.pColorBlendInfo = &blendCreateInfo;

```

70

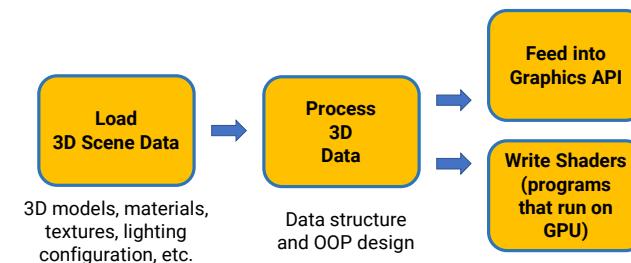
69

70

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)**

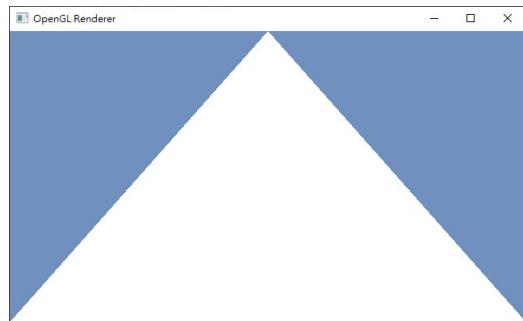


71

72

Goals (cont.)

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



73

Goals (cont.)

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



74

73

74

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

75

Any Questions?

76

75

76

19