COMPUTER GRAPHICS (CCG3013)

LESSON 2

GRAPHICS HARDWARE AND SOFTWARE

UOW
MALAYSIA
KDU PENANG
UNIVERSITY COLLEGE

PART OF THE UNIVERSITY
OF WOLLONGONG AUSTRALIA
GLOBAL NETWORK

UNIVERSITY OF
LINCOLN
UNITED KINGDOM

COURSE OUTLINE

Lesson	Topic
1	Introduction to computer graphics
2	Graphics hardware and software
3	Geometry in 2D graphics
4 & 5	Geometry in 3D graphics
6 & 7	User interfaces and interactions
8	Colour
9 & 10	Motion and animation
11	Lighting and rendering
12	Surface shadings

TOPIC LEARNING OUTCOMES

1. Explain the graphics processing unit (GPU) and graphics development tools.

2. Setup a graphics development tool.

3. Implement a simple graphics program.

ASSESSMENTS

Structure	Marks (%)	Hand-out	Hand-in
Assignment 1 (Individual)	30	Week 1(Unofficial) Week 3(Official)	Week 6
Assignment 2 (Group up to four only)	30	Week 1(Unofficial) Week 3(Official)	Week 12
Final examination	40	Exam week	

REPLACEMENT CLASS

Affected date: -

Date: -

Time: -

CONTENT

No.	Topics	Duration (Minutes)
1	Mini lecture 1: Graphics processing unit (GPU)	15
2	Exercise 1	10
3	Mini lecture 2: Graphics accelerators	15
4	Exercise 2	10
5	Break	10
6	Mini lecture 3: Graphics libraries	15
7	Exercise 3	10
8	Mini lecture 4: Display settings	15
9	Exercise 4	10

REVIEW I: IMAGE

1. Computer graphics is a visualisation of geometry objects that display or render on the screen.

2. Abstract image contains either text, sketch, drawing, diagram or the combination of those.

3. A real-life image which usually captured from the real world using a camera.

4. A dot in a digital image represents a picture element (pixels).

5. A pixel consists of the position (x, y) and its light properties.

REVIEW II: VIDEO

1. A video is a series of images which is played over certain period of time.

2. The content represents a motion or action for multiple scenes.

3. An image in a video is called a frame.

4. The rate of flipping through the images is measured in frame per second (fps).

REVIEW III: USER INTERFACES

1. A user interface (UI) is a container which contains the graphics elements.

2. A UI can be visualized in either in 2D, 2.5D, 3D or 4D.

3. A standard layout of an UI consisted of a header, a body, and a footer.

MINI LECTURE 1
GRAPHICS PROCESSING UNIT (GPU)

…

GRAPHICS PROCESSING UNIT (GPU)

There are three main modules in GPU;

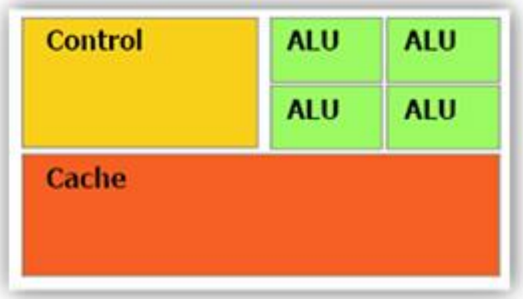
1. **Driver**, firmware for CPU to recognize and use the GPU.
2. **Graphics accelerator**, graphics libraries that render 2D/3D graphics.
3. **Projection**, transform 3D graphics to 2D display.

EXAMPLE GPU: NVIDIA GEFORCE GTX TITAN X

1. **Memory**: 12GB
2. **Resolution**: 5,120 × 3,200 pixels
3. **Texture fill rate**: 192 per second
4. **Connectivity**: Digital video interface (DVI), High-definition multimedia interface (HDMI), DisplayPort.
5. **Polygon counts**

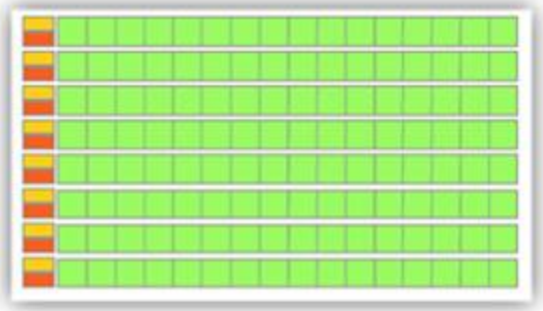
CPU VERSUS GPU

CPU



- Low compute density
- Complex control logic
- Large caches (L1$/L2$, etc.)
- Optimized for serial operations
  - Fewer execution units (ALUs)
  - Higher clock speeds
- Shallow pipelines (<30 stages)
- Low Latency Tolerance
- Newer CPUs have more parallelism

GPU



- High compute density
- High Computations per Memory Access
- Built for parallel operations
  - Many parallel execution units (ALUs)
  - Graphics is the best known case of parallelism
- Deep pipelines (hundreds of stages)
- High Throughput
- High Latency Tolerance
- Newer GPUs:
  - Better flow control logic (becoming more CPU-like)
  - Scatter/Gather Memory Access
  - Don't have one-way pipelines anymore

EXERCISE 1

This activity takes about ten minutes.

1. Go to Nvidia website.

2. Find a model for a GPU.

3. Get the specifications; Memory, resolution, texture fill rate or framerate, connectivity, and polygon counts.

MINI LECTURE 2
GRAPHICS ACCELERATORS

• • •

GRAPHICS BASIC PIPELINE

Graphics components consist of

(a) Display,

(b) Renderer,

(c) Matrices transformation,

(d) Inputs and callbacks,

(e) Motion and animation,

(f) Maps

GRAPHICS PROGRAMMING TOOLS

Sometimes it refers to graphics accelerator included

1. **OpenGL**
2. **DirectX**, now it is part of Microsoft Windows SDK
3. **Processing**
4. **CUDA**

OPENGL

1. Developers: Originally Silicon Graphics (SGI), presently Khronos Group.
2. Cross-language: C++, Java, etc.
3. Cross-platform application programming interface (API): Linux, Windows, Mac OSX, etc.
4. 2D/3D graphics rendering.

DIRECTX

1. Developed by Microsoft.
2. It is mainly built for games and multimedia applications.
3. It originally builds on top of the Windows API, later in .NET framework, and presently in Windows SDK.

PROCESSING

1. Developers: Casey Reas and Benjamin Fry.
2. They are from Aesthetics and Computation Group, at MIT Media Lab.
3. It is built for electronic arts, media arts, and visual design.
4. It is currently in GNU’s GPL, LGPL licenses.
5. Filename extension: .pde.
6. It supports for Arduino and Raspberry Pi.

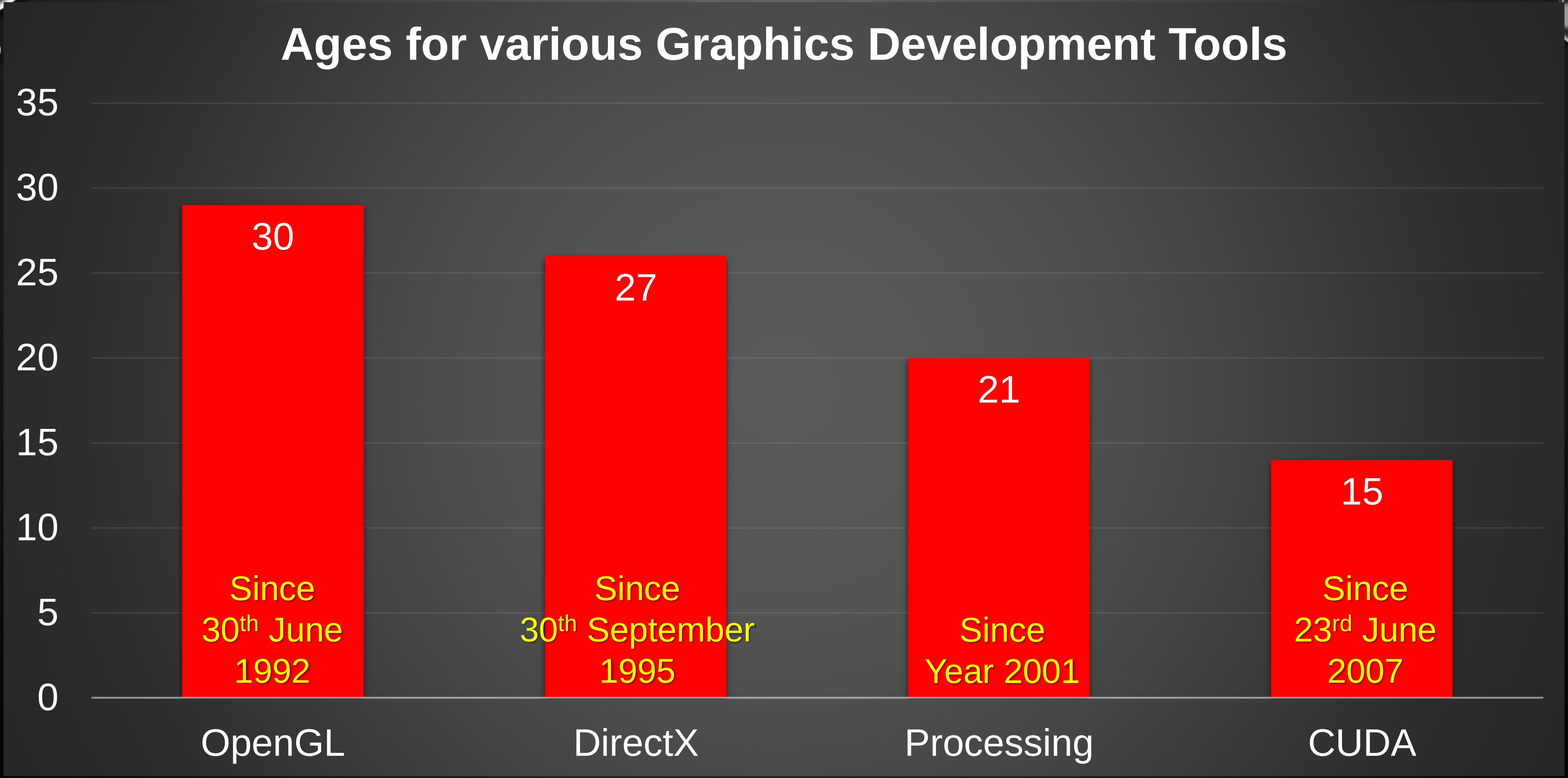
CUDA

1. CUDA stands for compute unified device architecture.

2. Developed by NVidia Corporation.

3. It is mainly built for parallel computing platform and application programming interface (API).

4. Operating system supports included Windows, Mac OSX, and Linux.


Ages for various Graphics Development Tools
35
30
25
20
15
10
5
0
30
Since 30th June 1992
27
Since 30th September 1995
21
Since Year 2001
15
Since 23rd June 2007
OpenGL
DirectX
Processing
CUDA

EXERCISE 2

This activity will takes about ten minutes.

1. State four available graphics accelerators.

2. What are the six main components for a basic of graphics?

3. Which graphics accelerators is the oldest to present day?

4. OpenGL is originally developed by ___.

5. Which graphics accelerators are free to download?

BREAK

…

MINI LECTURE 3
GRAPHICS LIBRARIES

…

GLUT

1. GLUT stands for OpenGL utility toolkit.
2. It is written by Mark J. Kilgard.
3. It is a library of utilities for OpenGL program.
4. It performs input and output (I/O) controls for a host operating system.

GLU

1. GLU stands for OpenGL utility library.

2. It is a library for computer graphics.

3. It provides the primitive functions for OpenGL.

4. Functions included coordinates system, view settings, 2D/3D primitive modelling, and mappings.

GLUI

1. It stands for OpenGL user interface library.
2. It built on top of GLUT.
3. It provides the basic controls to interact with the OpenGL window.
4. Controls included buttons, checkboxes, radio buttons, spinners, etc.

EXERCISE 3

This activity will takes about five minutes.

Find and explain three other OpenGL libraries from The Khronos Group, Inc.

MINI LECTURE 4
DISPLAY SETTINGS

...

DISPLAY MODE

Function name	glutInitDisplayMode()
Purpose	Initializes display mode for OpenGL window.
Arguments or parameters	GLUT_SINGLE GLUT_DOUBLE GLUT_RGB GLUT_RGBA GLUT_DEPTH
Return value	None

ARGUMENTS FOR DISPLAY MODE

GLUT_SINGLE	Set single frame buffer.
GLUT_DOUBLE	Set double frame buffer.
GLUT_RGB	Set RGB mode frame buffer.
GLUT_RGBA	Set RGBA mode frame buffer.
GLUT_DEPTH	Set 32-bit depth buffer.

DISPLAY CALLBACK

Function name	glutDisplayFunc()
Purpose	It is a callback function for the current OpenGL window.
Argument or parameter	renderFunction(). It is the function where you can customize objects that you want to render. Note that the function name can be changed.
Return value	None

DISPLAY LOOP

Function name	glutMainLoop()
Purpose	It repeats the rendering until the termination of the OpenGL application.
Argument or parameter	None
Return value	None

WINDOW CREATION

Function name	glutCreateWindow()
Purpose	It creates an OpenGL window.
Argument or parameter	char* name, it set the name for the window. Value for instance, “Hello OpenGL!”.
Return value	Integer number as an identifier for the window.

WINDOW SIZE

Function name	glutInitWindowSize()
Purpose	It set the window size. Default size is 300 × 300 pixels.
Arguments or parameters	WIDTH, measured in pixels. HEIGHT, measured in pixels.
Return value	None

WINDOW POSITION

Function name	glutInitWindowPosition()
Purpose	It set the position of OpenGL window.
Arguments or parameters	offsetFromLeft, offset from the left of the screen. offsetFromTop, offset from the top of the screen.
Return value	None

BACKGROUND

Function name	glClearColor()
Purpose	It set the background colour for an OpenGL window.

Arguments or parameters	Red;
	Green;
	Blue;
	Alpha;

Return value	None

BACKGROUND RESET

Function name	glClear()
Purpose	It clear the window with the colour selected by glClearColor().
Argument or parameter	GL_COLOR_BUFFER_BIT
Return value	None

CLEAR MEMORY FOR A RENDER

Function name	glFlush()
Purpose	It flushes all OpenGL commands in the queue and frame buffer for each render call.
Argument or parameter	None
Return value	None

FINISH THE RENDERING

Function name	glFinish()
Purpose	It blocks until all GL executions are completed, then only it exits from render function.
Argument or parameter	None
Return value	None

EXERCISE 4

This activity will takes about five minutes.

Write a complete program in C++ OpenGL, to display a window with the following parameters.

Properties	**Value**
Frame buffer	Single frame
Width × Height	1280 × 720
Offset X, offset Y	50, 100
Window title	Viewport01
Background colour	Green

REFERENCES

Main reference:

Hajek, D. (2019). Introduction to Computer Graphics 2019 Edition. Independently Published.

Additional reference:

Marschner, S. and Shirley, P. (2021). Fundamentals of Computer Graphics, 5th Edn. CRC Press: Taylor's & Francis.