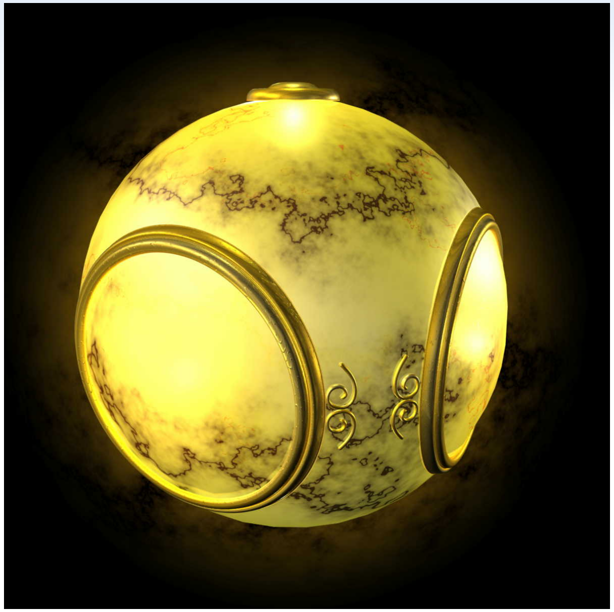
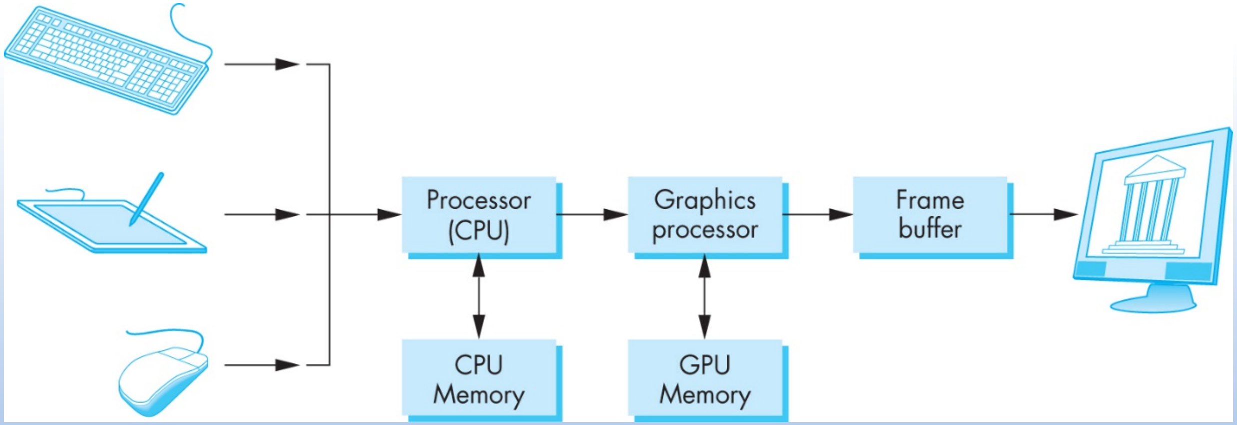
# Introduction

# What is Computer Graphics?

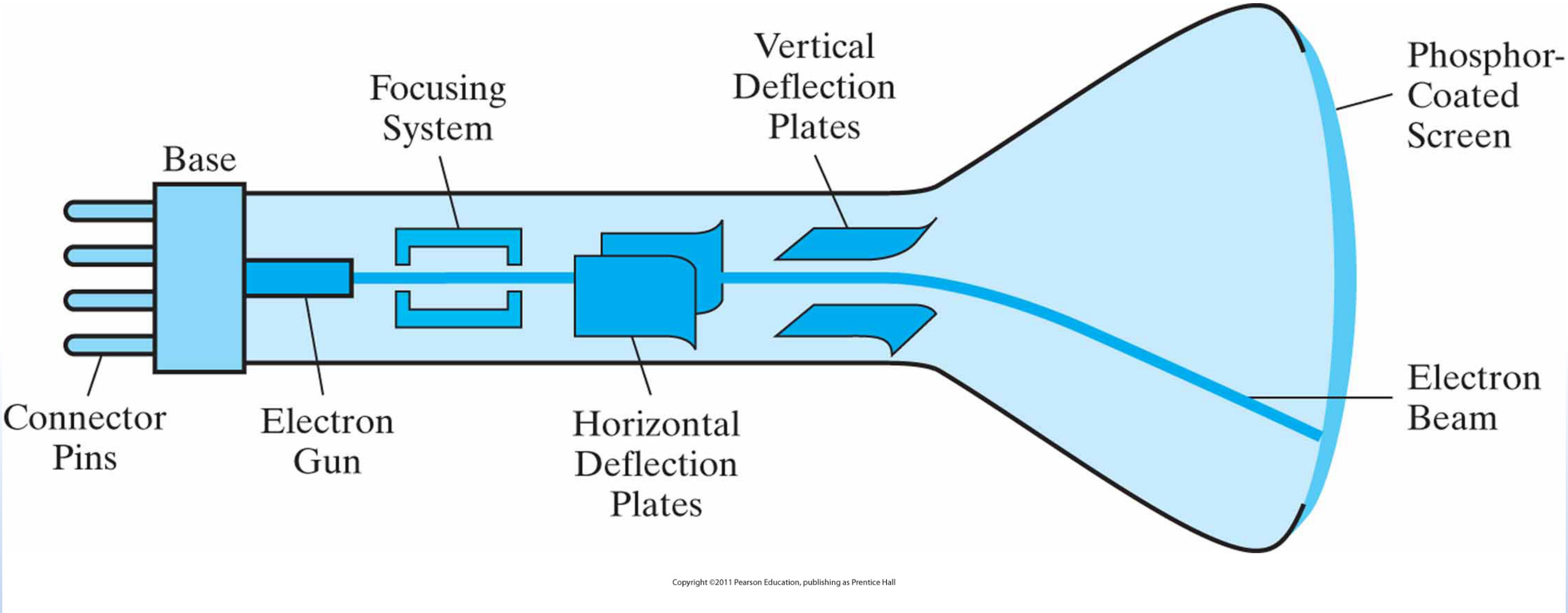


- Computer graphics
  - Deals with all aspects of creating images with a computer
    - Hardware
    - Software
    - Applications
- Basic graphics system



# Display Devices

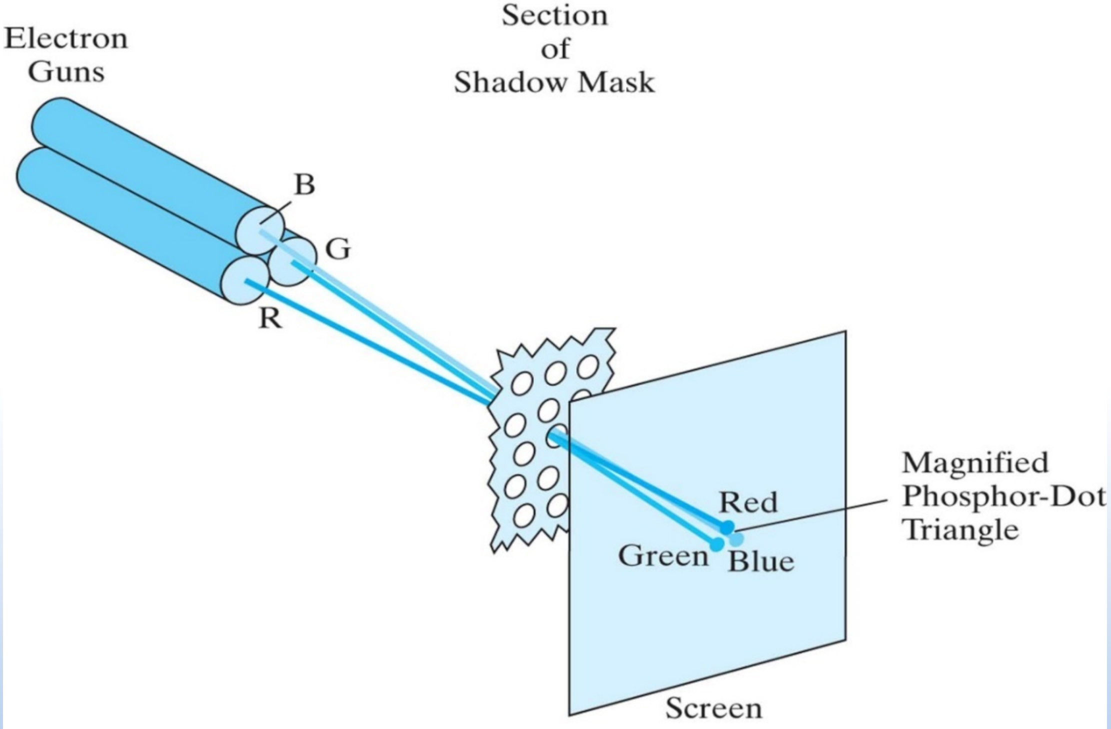
- Cathode-Ray Tubes (CRT)
  - Electrostatic deflection of the electron beam in a CRT



Copyright ©2011 Pearson Education, publishing as Prentice Hall

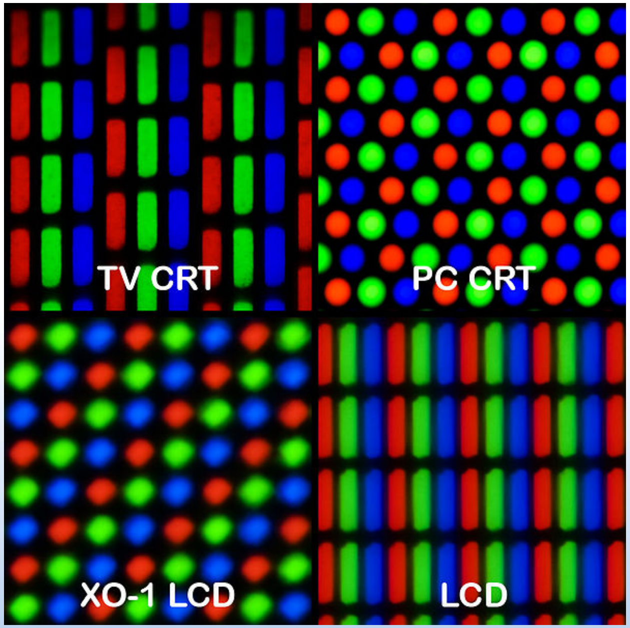
# Display Devices

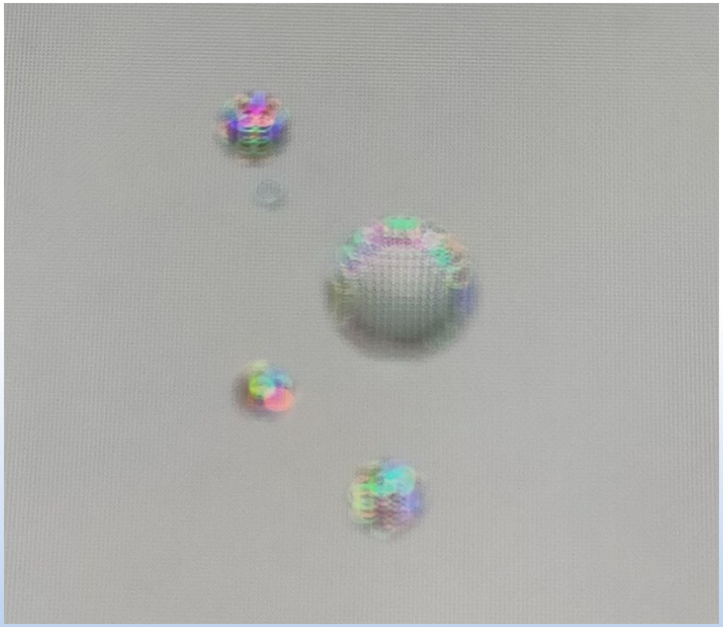
- Three electron guns
  - ➢ Aligned with the triangular colour-dot patterns on the screen, are directed to each dot triangle by a shadow mask



# Display Devices

- Sub-pixels
  - Pixel grid is divided into single-colour regions that contribute to the displayed or sensed colour when viewed at a distance

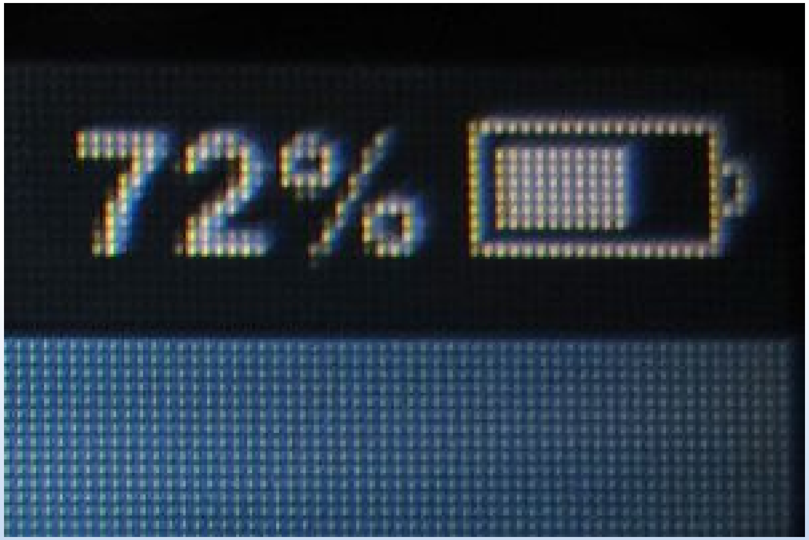


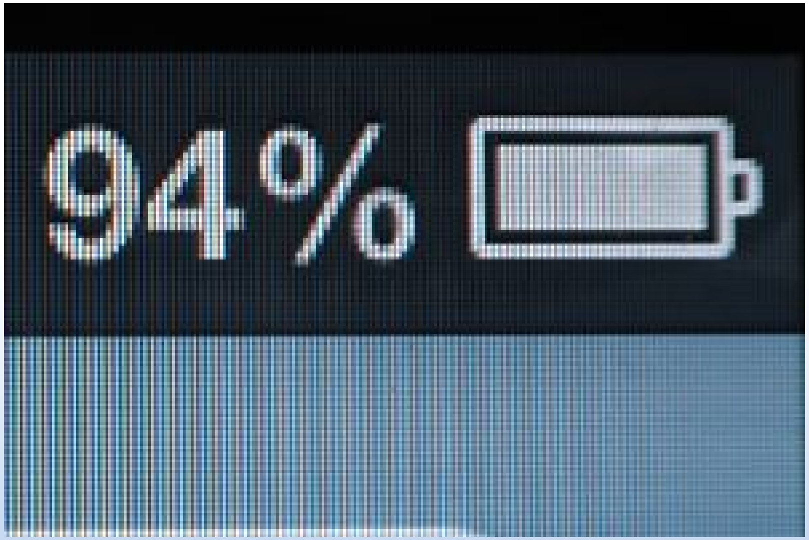


https://commons.wikimedia.org/wiki/File:Pixel_geometry_01_Pengo.jpg

# Display Devices

- “Retina display”
  - ➢ Apple’s brand name for screens that have a higher pixel density than their previous models
  - ➢ High pixel density, pixels per inch (PPI)

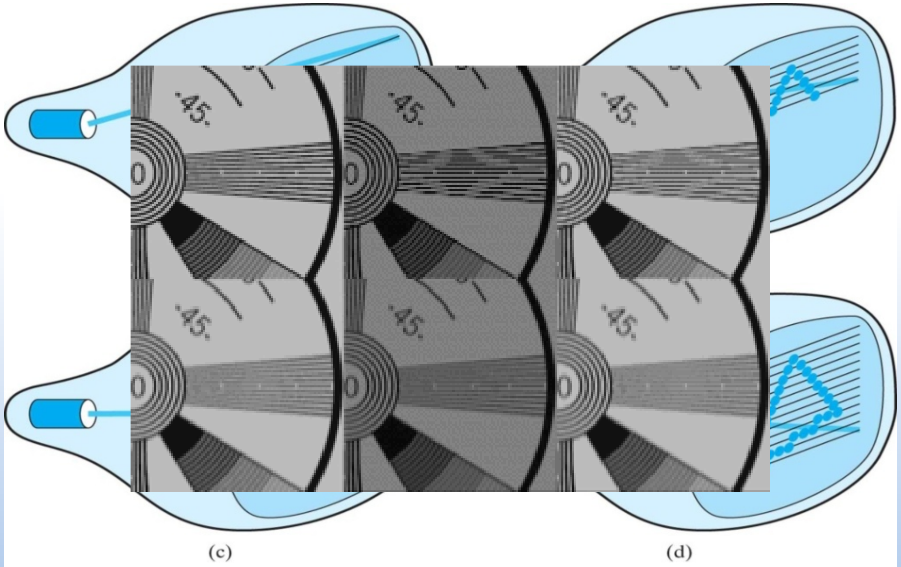




https://en.wikipedia.org/wiki/Retina_display

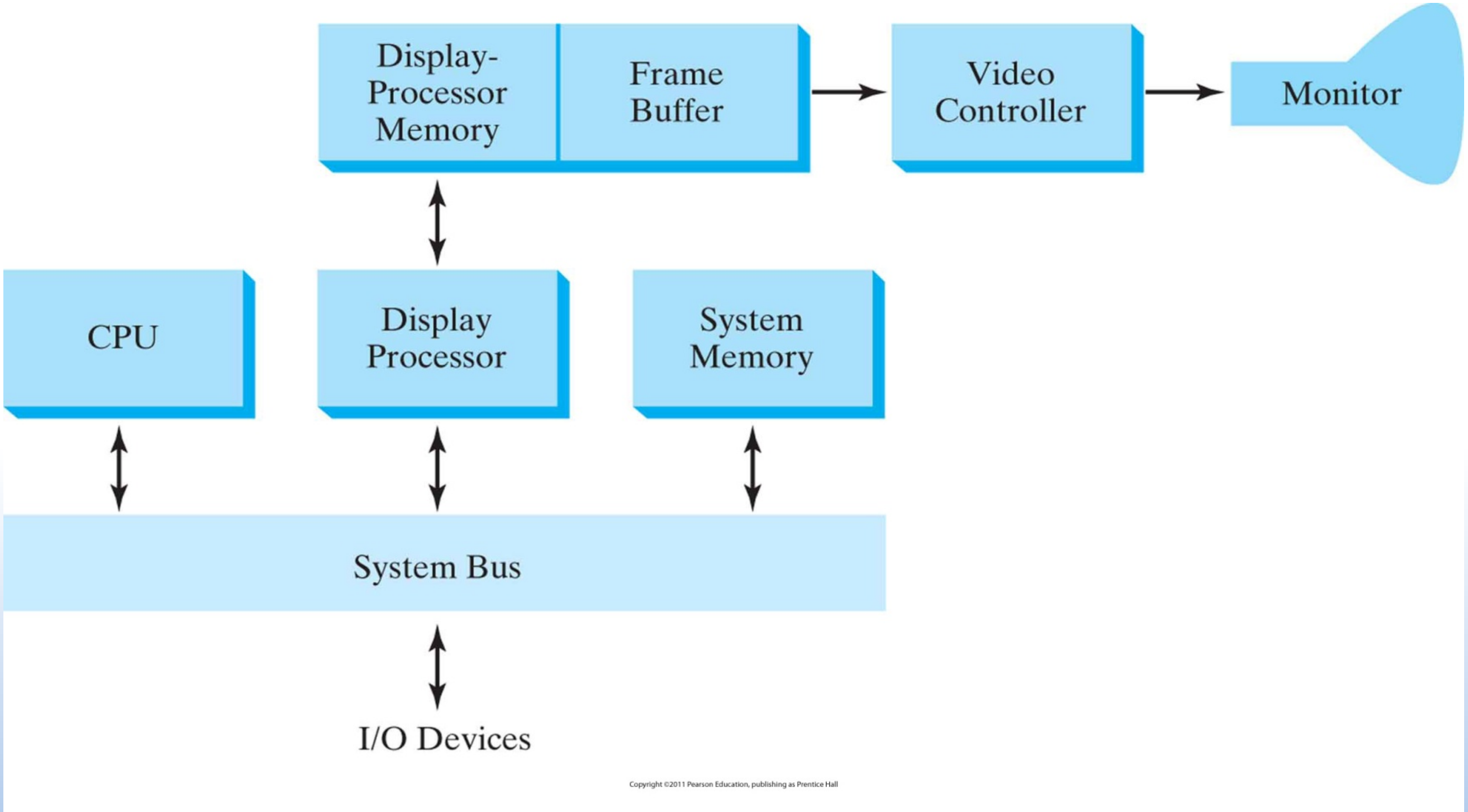
# Raster-Scan Displays

- A raster-scan system displays an object as a set of discrete points (pixels)
  - ➢ Each row is referred to as a **scan-line**
  - ➢ Pixel information (not necessarily only colour) stored in buffer locations, collectively referred to as the **frame buffer**
- Refresh rate
  - ➢ Frequency at which a picture is redisplayed on the screen
  - ➢ Vertical synchronisation
    - **vsync**



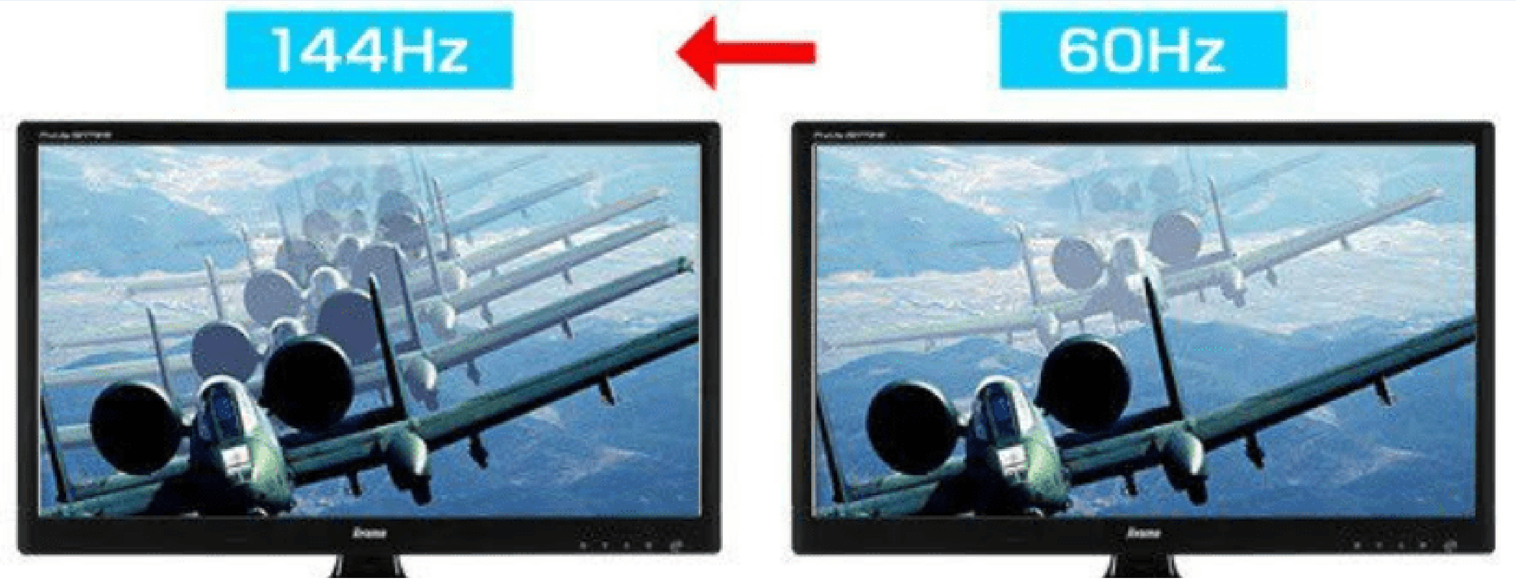
# Raster-Scan Systems

- Architecture of a raster-graphics system with a display processor



Copyright ©2011 Pearson Education, publishing as Prentice Hall

# Refresh Rates



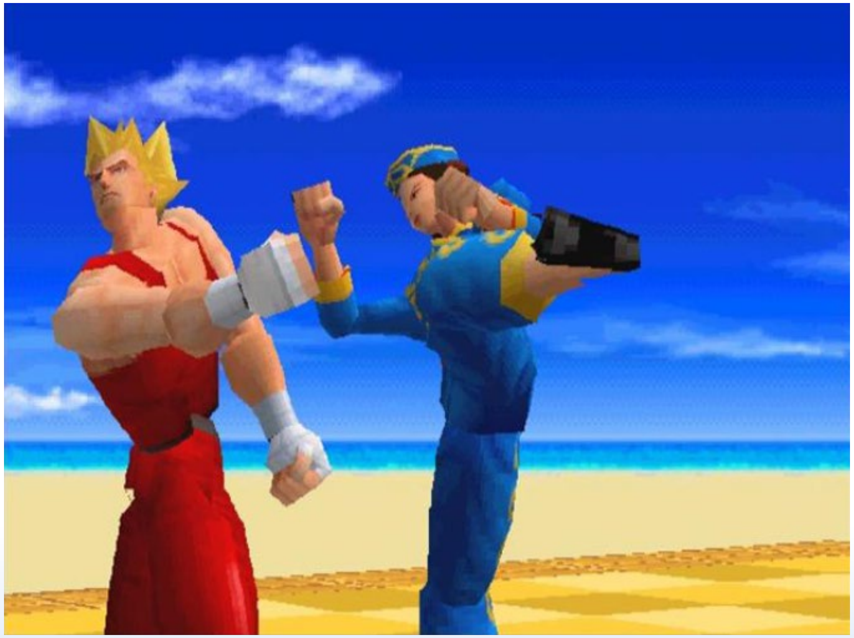
https://computercoachaustralia.com.au/screen-refresh-rate-what-does-it-do/
http://www.dual-monitors-guide.com/do-you-really-need-monitors-with-faster-refresh-rates.html

# Real-Time Rendering

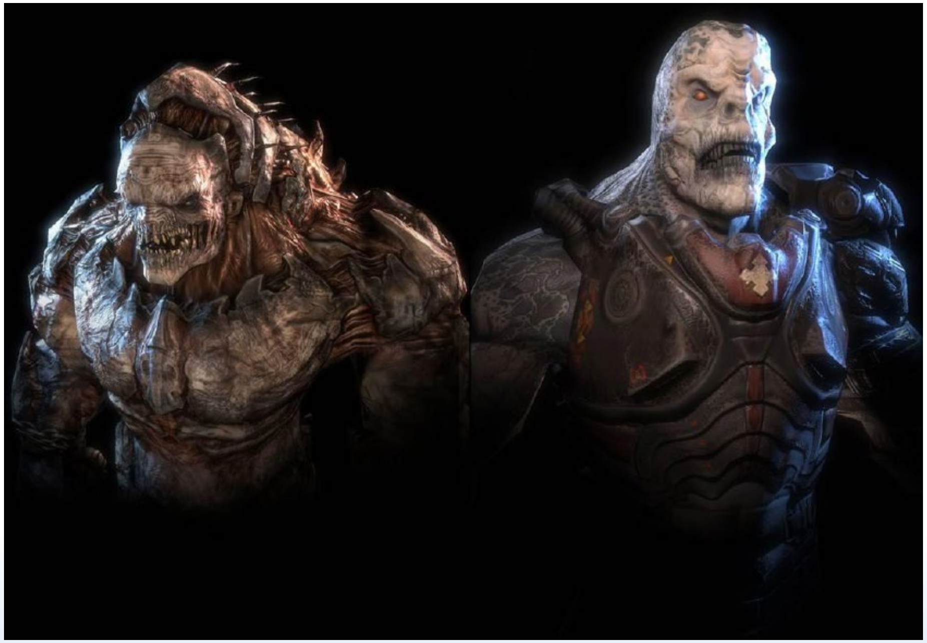
- What is rendering?
  - ➢ The process of converting data into visually perceivable form
- In general, real-time rendering
  - ➢ Rendering at interactive rates
  - ➢ Display rate >10 images per second
  - ➢ Video games aim for 60 frames per second (fps)
    - Some argue that 30 fps is enough
    - In general, the faster the better
  - ➢ Nowadays done with the help of graphics processing units (GPUs)

# Real-Time Rendering

- Real-time rendering on graphics hardware



1997



2004

Chris Seitz, “Evolution of GPUs”, NVIDIA Corporation

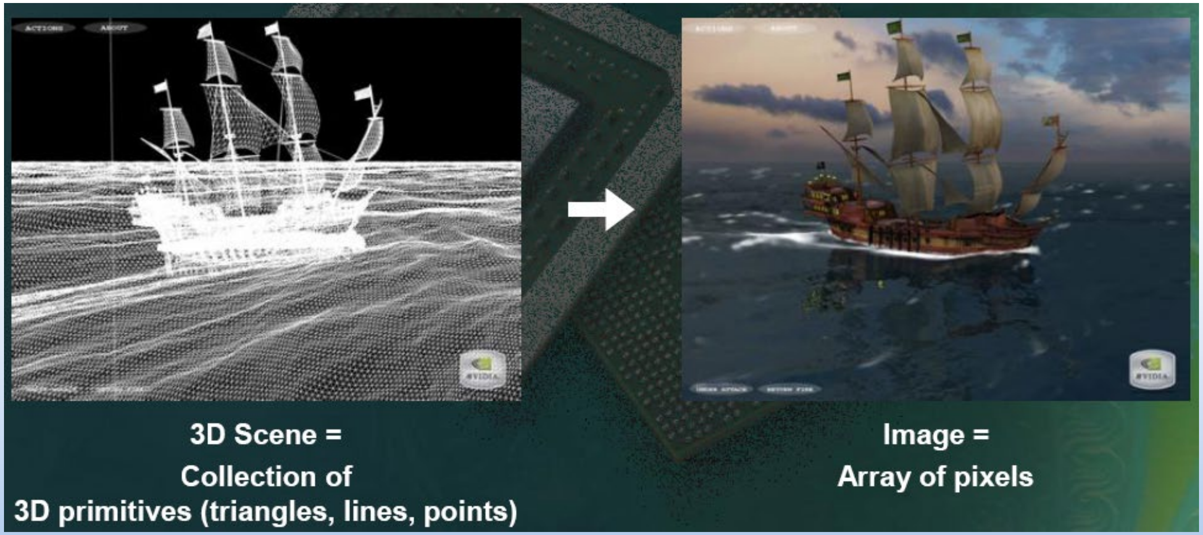
# Graphics Hardware

- Graphics processing unit (GPU)
  - ➢ Dedicated to providing high-performance, visually rich, interactive 3D graphics
  - ➢ All of today's commodity GPUs structure their graphics computations in a similar organisation called the **computer graphics pipeline**
  - ➢ Why is it so fast?
    - Parallelism, specialisation, little data-dependency, etc.

https://www.nvidia.com/en-us/geforce/20-series/
https://www.amd.com/en/products/graphics/amd-radeon-rx-5700-xt

# The Computer Graphics Pipeline

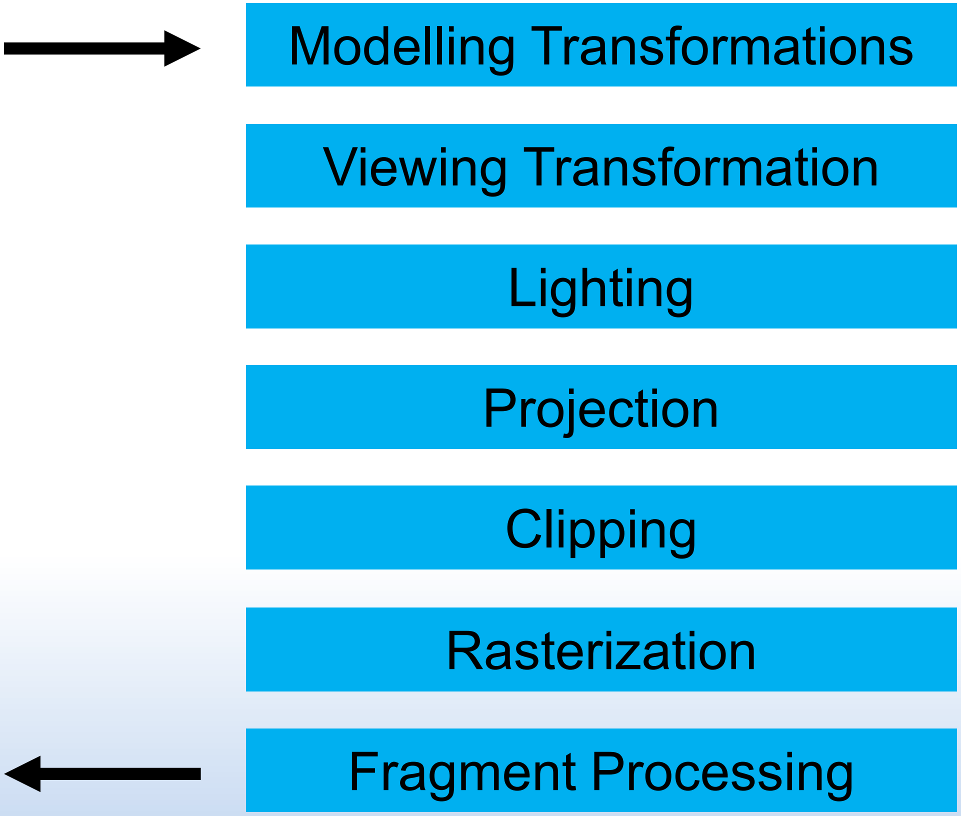
- Graphics system
  - Task is to synthesise an image from a description of a scene
  - Scene contains geometric primitives, descriptions of lights, the way each object reflects light, the viewer's position and orientation



Chris Seitz, “Evolution of GPUs”, NVIDIA Corporation

# The Computer Graphics Pipeline

- Overview of stages in the graphics pipeline
  - ➢ Input
    - Vertex data
      - – Geometric models



Modelling Transformations

Viewing Transformation

Lighting

Projection

Clipping

Rasterization

Fragment Processing

  - ➢ Output
    - Pixels for display

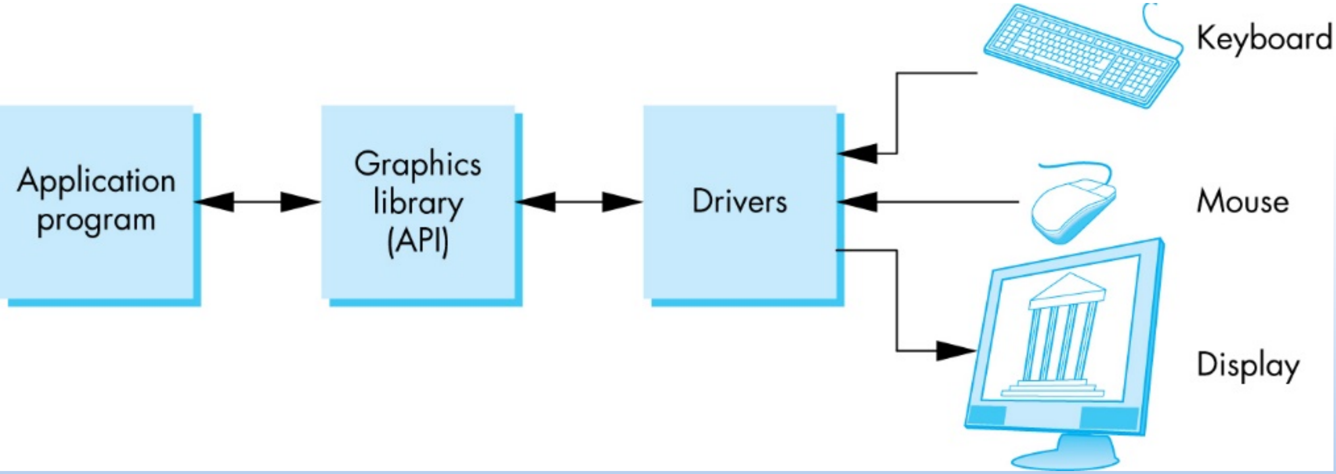
# The Programmer’s Interface

- Programmer sees the graphics system through a software interface
  - ➢ Graphics application programmer interface (API)
    - Library of graphics functions that can be used in a programming language
    - DirectX, OpenGL, and recently Vulkan and Metal

Microsoft® DirectX®

OpenGL®

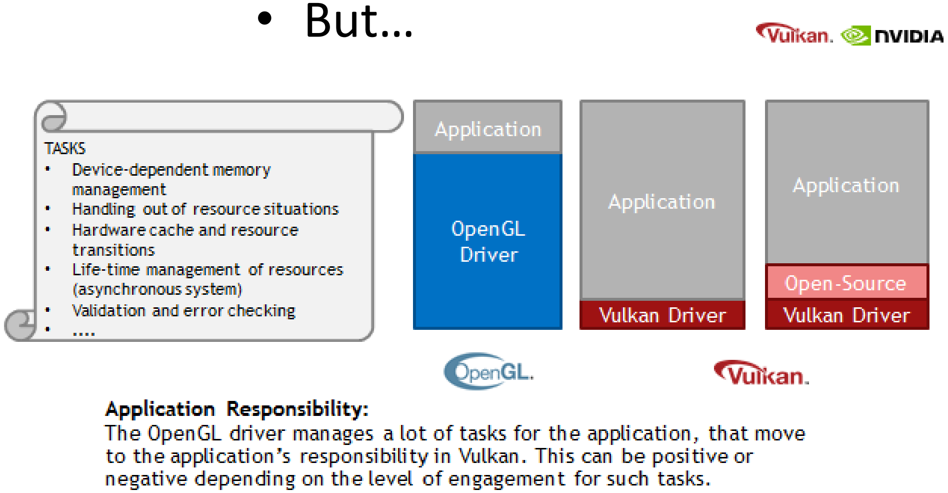
Vulkan™



Metal

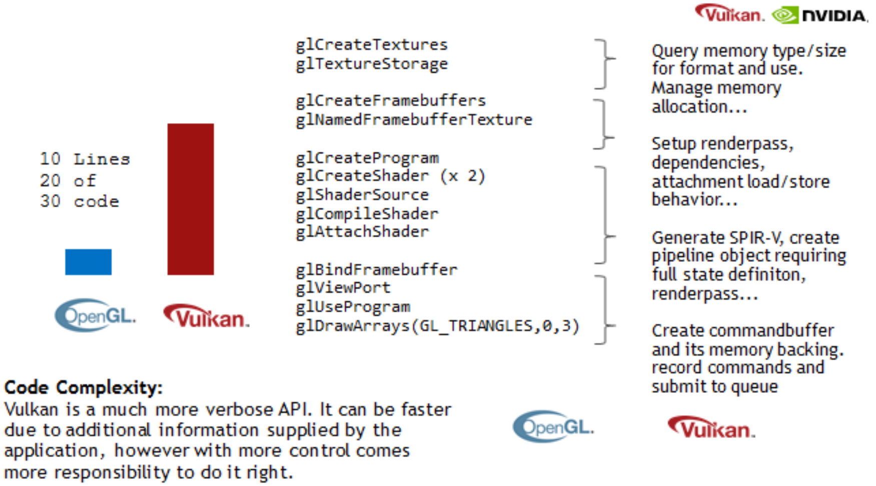
# The Programmer’s Interface

- What is Vulkan?
  - 1.0 launched 2016
  - Cross-platform API
  - Low overhead
    - Gives programmers even more control over the hardware
    - But…



TASKS
- Device-dependent memory management
- Handling out of resource situations
- Hardware cache and resource transitions
- Life-time management of resources (asynchronous system)
- Validation and error checking
- ....

**Application Responsibility:**
The OpenGL driver manages a lot of tasks for the application, that move to the application’s responsibility in Vulkan. This can be positive or negative depending on the level of engagement for such tasks.



```
glCreateTextures
glTextureStorage

glCreateFramebuffers
glNamedFramebufferTexture

glCreateProgram
glCreateShader (x 2)
glShaderSource
glCompileShader
glAttachShader

glBindFramebuffer
glViewPort
glUseProgram
glDrawArrays(GL_TRIANGLES,0,3)
```

Query memory type/size for format and use. Manage memory allocation…

Setup renderpass, dependencies, attachment load/store behavior…

Generate SPIR-V, create pipeline object requiring full state definiton, renderpass…

Create commandbuffer and its memory backing. record commands and submit to queue

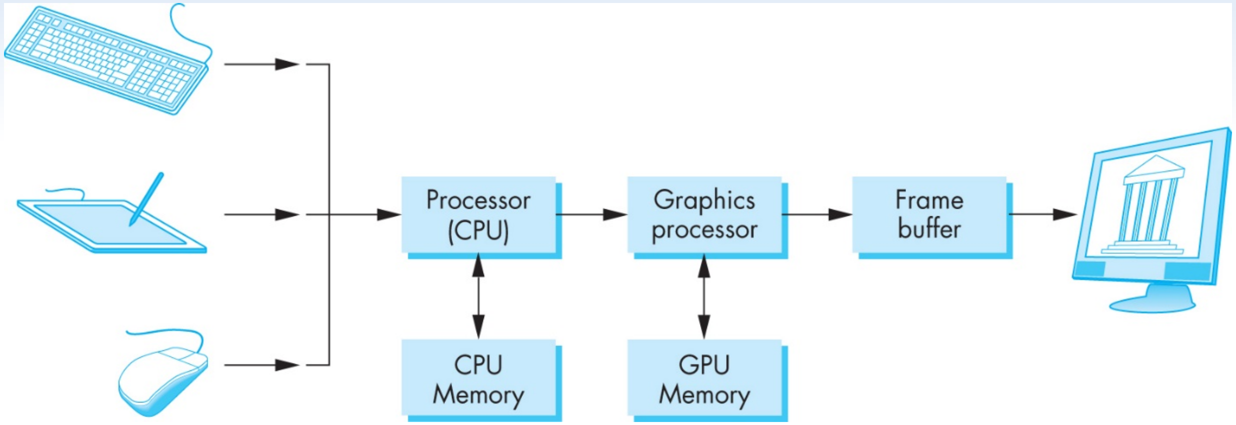
**Code Complexity:**
Vulkan is a much more verbose API. It can be faster due to additional information supplied by the application, however with more control comes more responsibility to do it right.

http://www.startrek.com/article/pluto-moon-named-vulcan-logically
https://developer.nvidia.com/transitioning-opengl-vulkan

# Creating a Window

- Create a window and context (and handle input)
  - OpenGL context represents many things
    - Stores all OpenGL states, represents buffers, etc.
    - Think of a context as an object that holds all of OpenGL
  - **GLFW**
    - There are others like freeGLUT
- Load OpenGL extensions
  - Otherwise some platforms default to OpenGL 1.2
    - This subject is about modern OpenGL, i.e. version 3.3+
  - **GLEW**
    - There are others like GLAD

# Creating a Window



- Frame buffer
  - ➢ Collection of buffers used for rendering
    - Colour buffer, depth buffer, stencil buffer
  - ➢ The term **frame** refers to the total display area
  - ➢ Often when people refer to the frame buffer, they are really talking about the **colour buffer**

# Creating a Window

- Colour buffer
  - ➢ Where RGBA colour values are stored
    - Colour channels
      - – RGBA: red, green, blue, alpha
  - ➢ Setting the clear colour

```
glClearColor(0.2f, 0.2f, 0.2f, 1.0f);
```

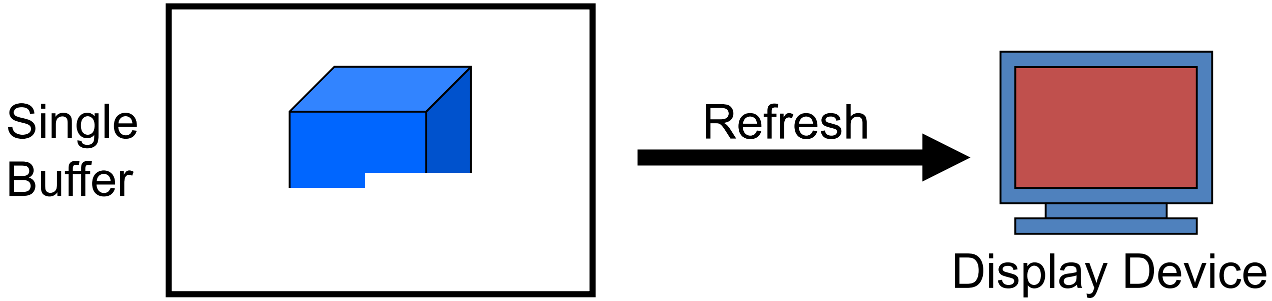
    - If not set, OpenGL will use the default clear colour (e.g., black)
    - In general, not recommended to set as black
      - – Errors in rendering may be displayed in black
  - ➢ Clearing the colour buffer

```
glClear(GL_COLOR_BUFFER_BIT);
```

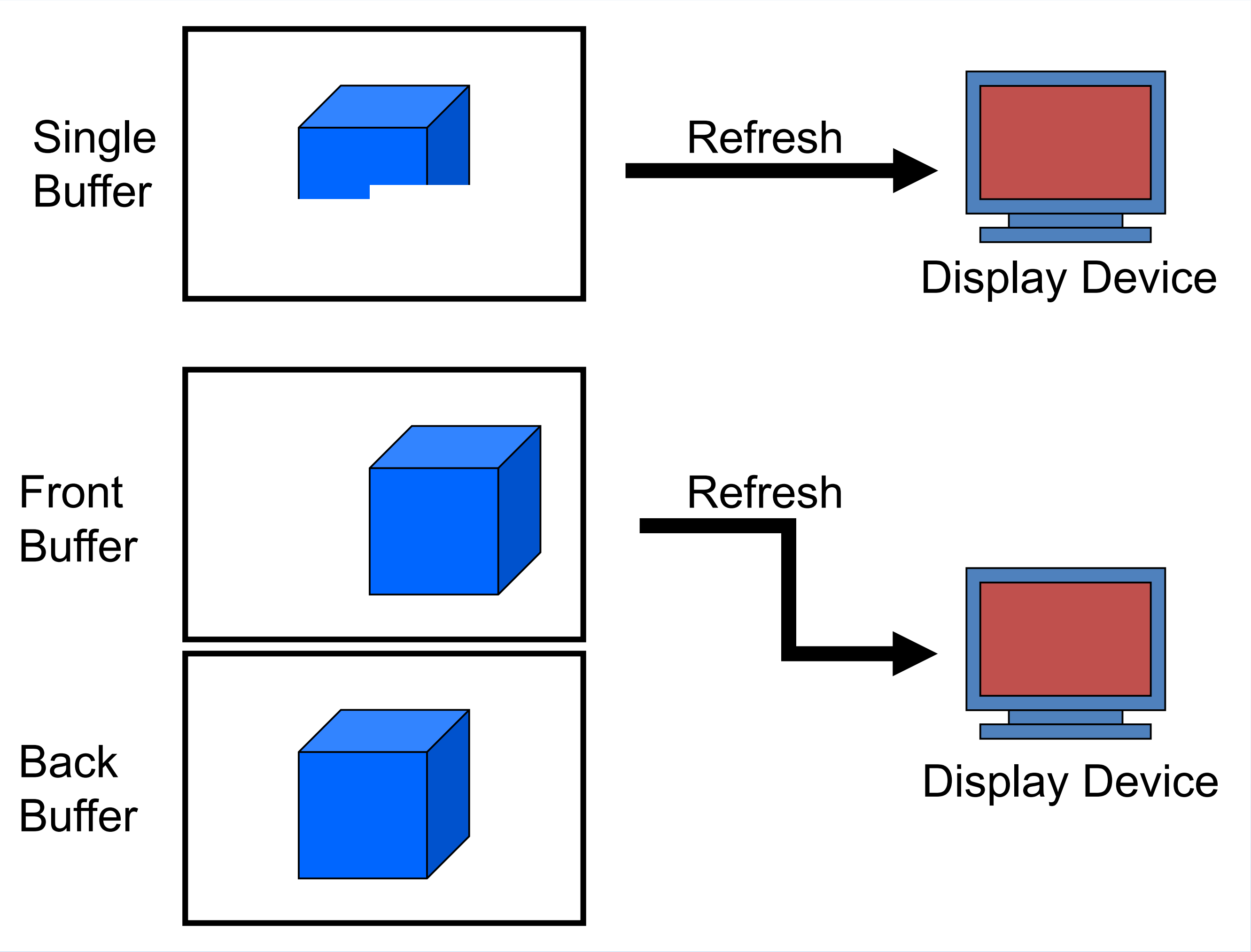
# Creating a Window

- Colour buffer
  - ➢ Double buffering
    - **Two** separate colour buffers
      - One for displaying, the other for rendering
      - Referred to as **front** and **back** buffers
      - Display content from the front buffer, render into the back buffer
    - Why?
      - To prevent flickering and other undesirable artifacts that will appear if an image that is currently being displayed is updated
      - Only want to display the frame when the rendering of that frame has finished

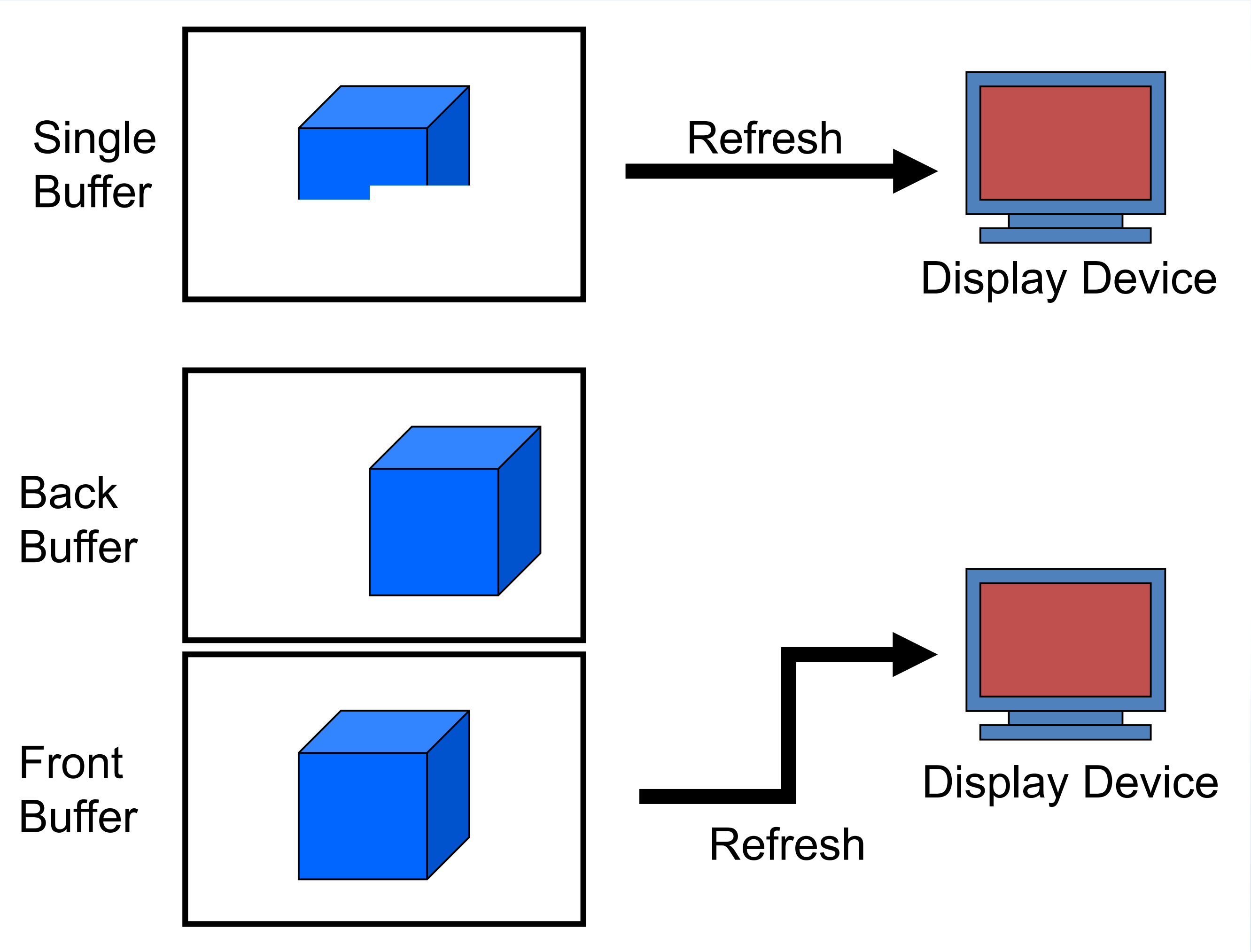
# Single Buffer



# Double Buffering



# Double Buffering



# Creating a Window

- Rendering loop
  - ➢ Loops until the window is closed
  - ➢ Each loop pass renders a single frame
  - ➢ Swap buffers once complete

    ```
    glfwSwapBuffers();
    ```

  - ➢ Swap buffers with display device’s **vsync**

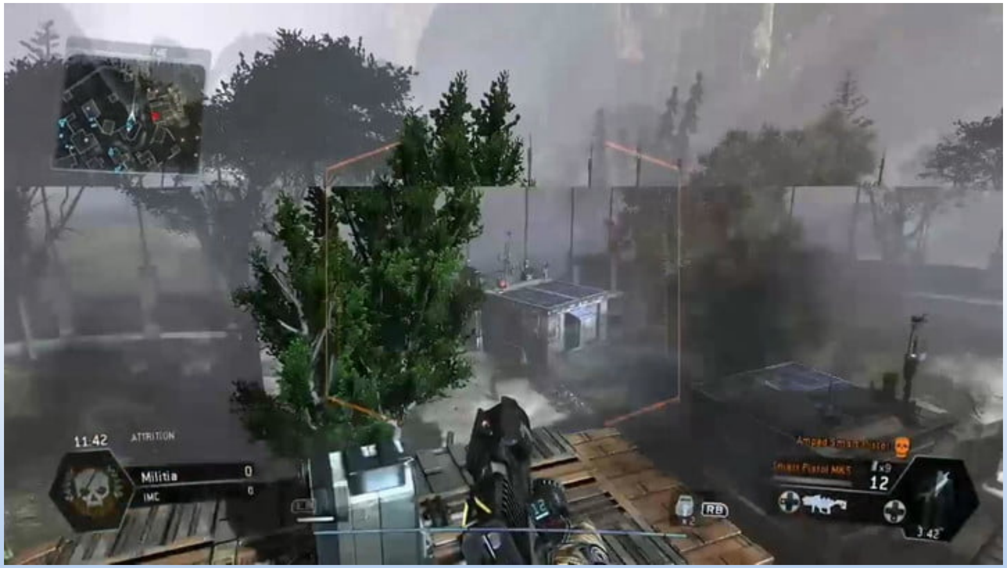
    ```
    glfwSwapInterval(1);
    ```

    - Note that some GPU drivers do not honor this request
  - ➢ Check and process events

    ```
    glfwPollEvents();
    ```

# Vertical Synchronisation

- Vertical synchronisation
  - ➢ **Vsync**
    - Synchronises graphics **frame rate** and display device’s **refresh rate**
    - A way of dealing with screen tearing
      - – When display devices shows portions of multiple frames in a single refresh



https://www.digitaltrends.com/computing/what-is-vsync/

# References

- Among others, material sourced from
  - Hearn, Baker & Carithers, “Computer Graphics with OpenGL”, Prentice-Hall
  - Angel & Shreiner, “Interactive Computer Graphics: A Top-Down Approach with OpenGL”, Addison Wesley
  - Chris Seitz, “Evolution of GPUs”, NVIDIA Corporation
  - https://www.khronos.org/opengl/wiki/
  - https://www.glfw.org/docs/
  - http://developer.nvidia.com
  - http://www.nvidia.com
  - http://www.amd.com
  - http://en.wikipedia.org/wiki/