Introduction to Computer Graphics

Introduction to Graphics Systems

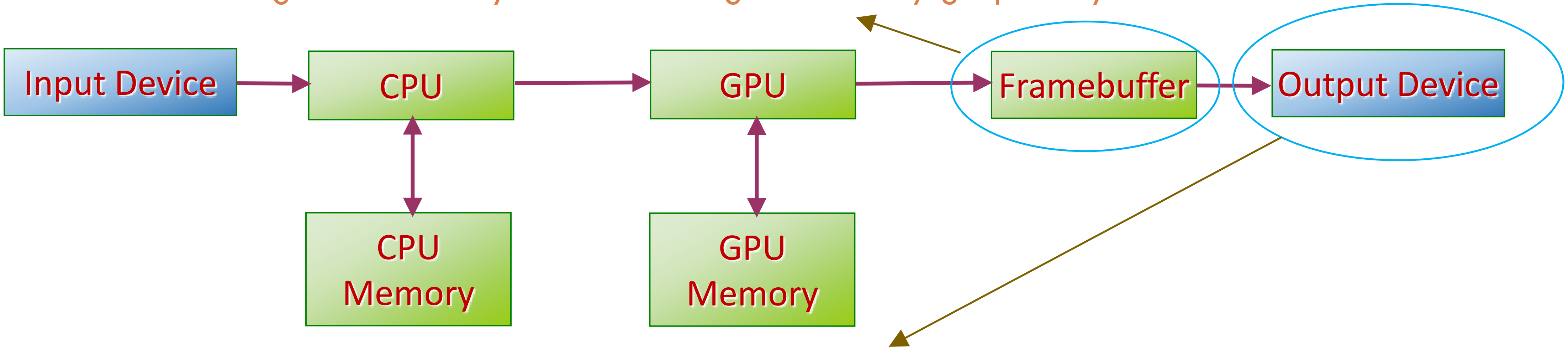
Prasanna Ghali

Plan

- Get acquainted with basic terminology

A Graphics System

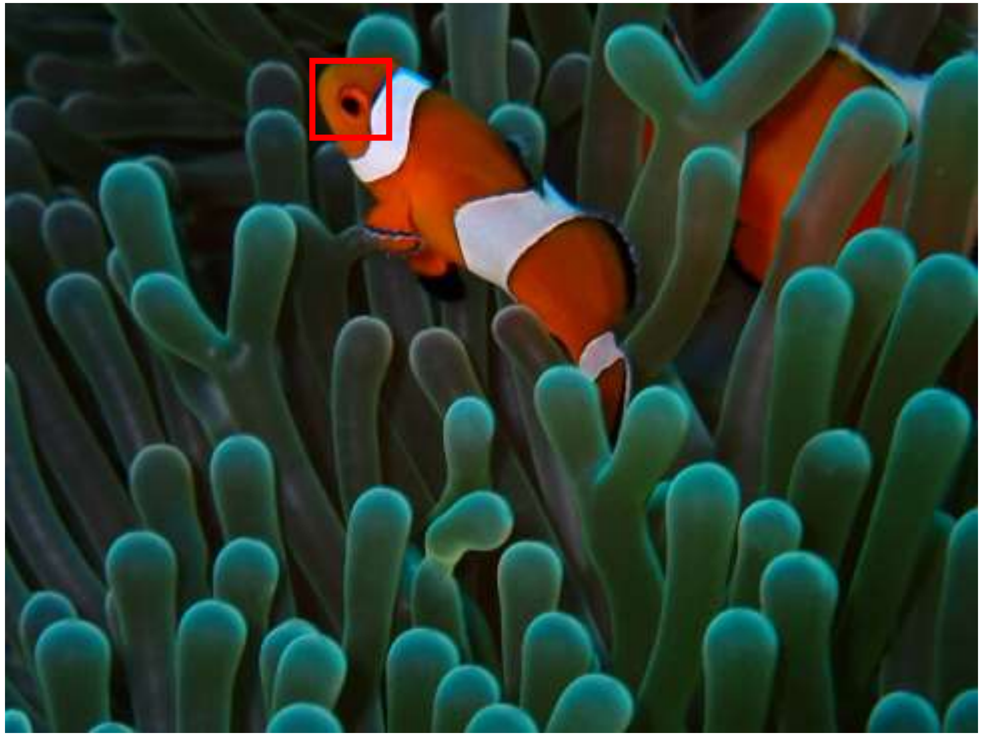
1. Graphics systems generate illusion of movement on screen by quickly displaying sequence of changing images, called *frames*
2. *Buffer* is region of memory where something of interest is stored
3. *Framebuffer* is region of memory where frame generated by graphics system is stored

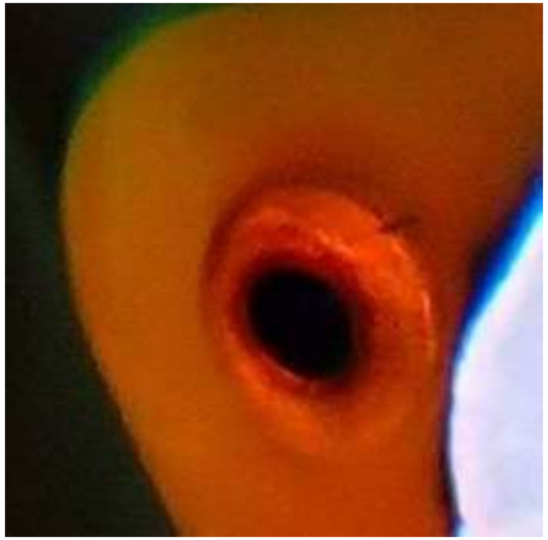


1. Virtually all modern computer graphics systems are *raster* based
2. Images are presented to user on some kind of *raster display*
3. Common example is flat-panel computer screen which has rectangular array of small light-emitting *picture elements* or *pixels* that can individually be set to different colors to create any desired image
4. Rasters are also prevalent in input devices: digital camera contains image sensor comprising grid of light-sensitive pixels, each of which records color and intensity of light falling on it

Raster Images

- Raster image is simply 2D array storing *pixel value* for each *pixel*
  - Each pixel corresponds to location or small area in image





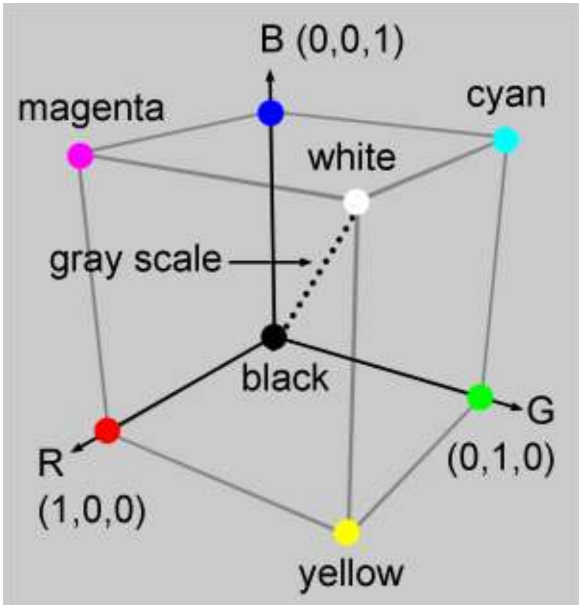
Detail of area around eye showing individual pixels

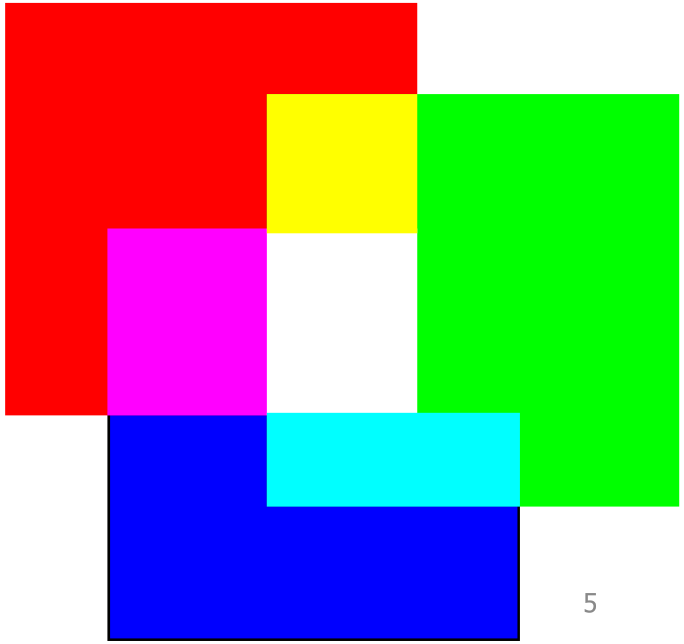
Raster Images: RGB Color Space

- Each pixel value is *usually* a color
- Color (C) of pixel is function of 3 specific components of that pixel: red (R), green (G), and blue (B) components

$$C = r_C R + g_C G + b_C B$$

r_C, g_C, b_C are scalars indicating intensities of red, green, and blue lights





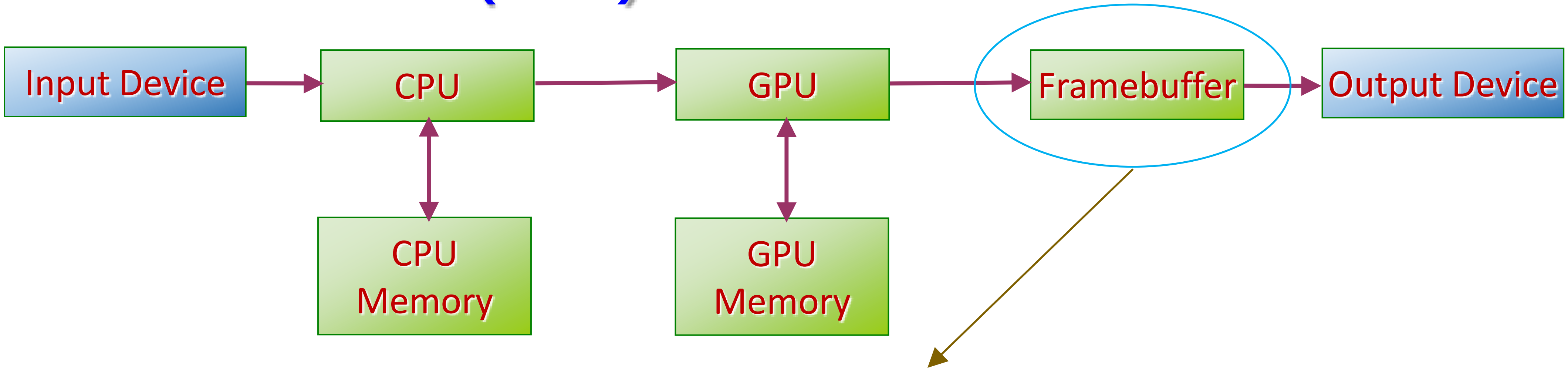
Raster Images: Resolution

- Resolution is number of pixels in raster image
  - If image has w columns and h rows, resolution is $w \times h$
  - When we use term “resolution”, we’re referring to spatial resolution
  - Larger the number of pixels in image, greater the resolution of image

Raster Image: Depth or Precision

- *Depth* or *precision* of raster image is number of bits used for each pixel and it determines how many colors can be represented in image
  - 1-bit-deep image can represent only two colors
  - 8-bit-deep image can represent $2^8 = 256$ colors
  - In true-color systems, image can represent 2^{24} colors

Framebuffer (1/3)

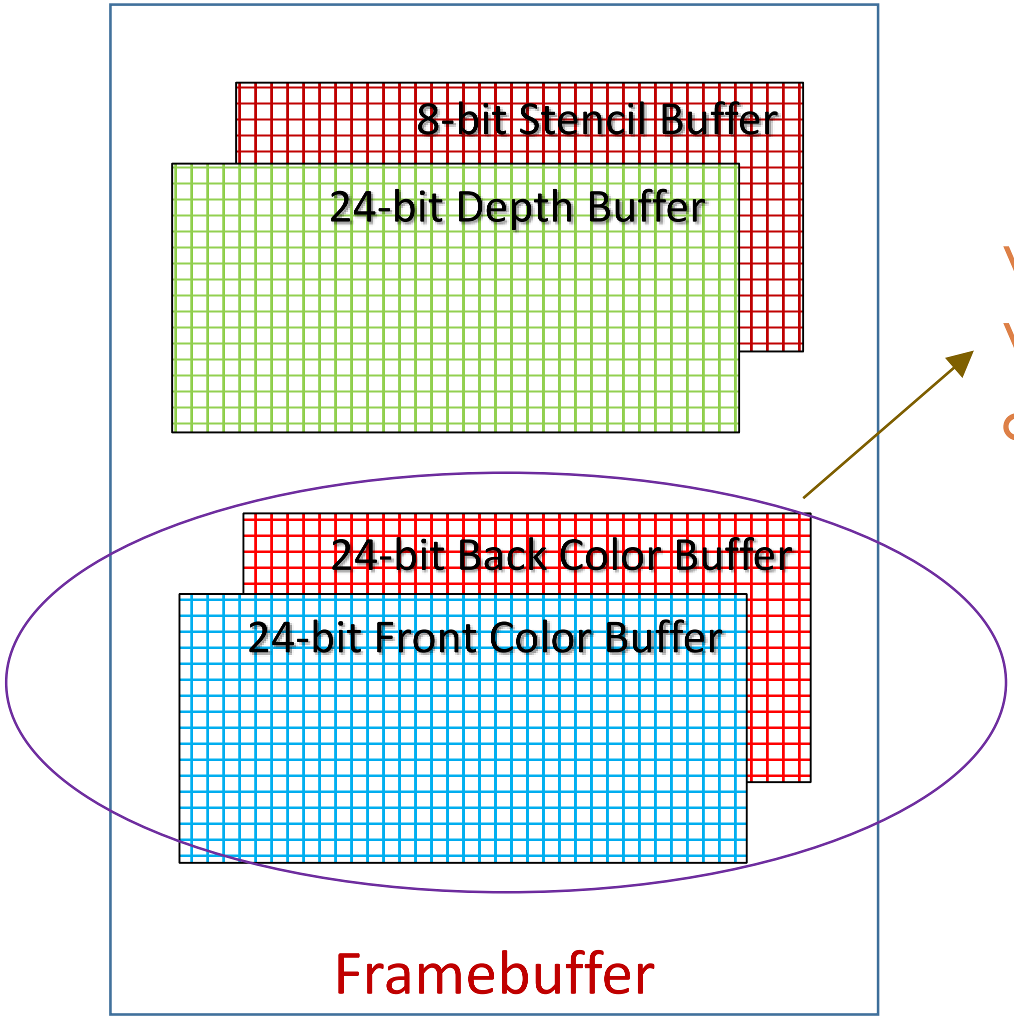


1. *Buffer* is region of memory where things of interest are stored
2. Pixels generated by GPU for display are collectively stored in part of memory called *framebuffer*
3. Framebuffer is core element of graphics system
4. You can setup framebuffer with appropriate resolution and depth
5. Although raster images usually store pixel values using integers, modern framebuffers can also store color components in floating-point

Framebuffer (2/3)

- In simple systems, framebuffer holds only color pixels displayed on output display device
- For most systems, framebuffer requires additional information
  - Depth information in *depth buffer* for hidden surface removal
  - Stencil information in *stencil buffer* to produce special effects
  - Additional buffers called *color buffers* to hold colored pixels to be displayed
- We will use *framebuffer* and *color buffer* synonymously

Framebuffer (3/3)



Why are there two color buffers? We'll explain the need for two color buffers later …

Graphics Systems: Production Pipeline (1/2)

1. Three major steps of graphics production pipeline
2. We use term *pipeline* to indicate output of one step is taken as input of next step

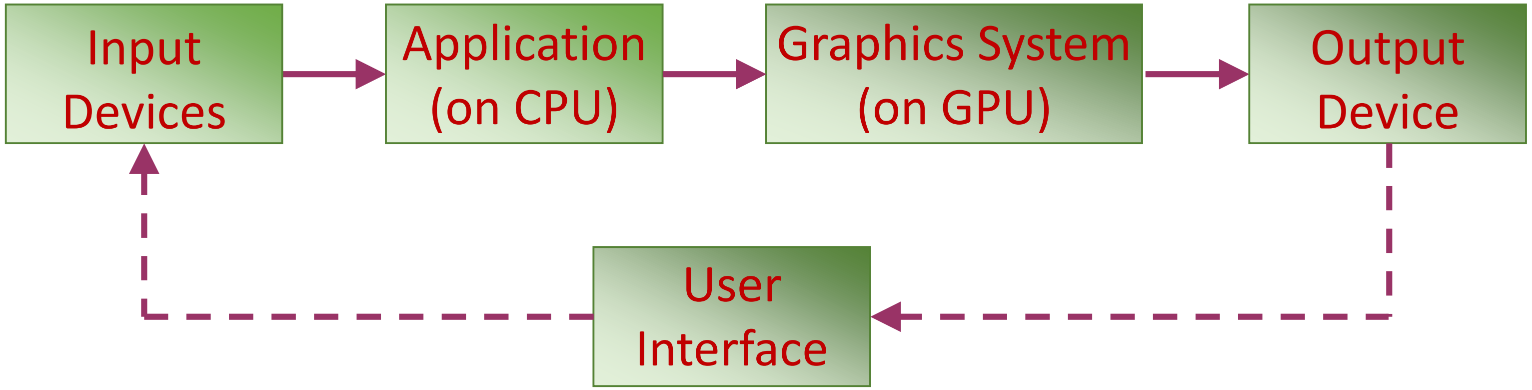
Graphics Systems: Production Pipeline (2/2)

Modeling → Animation → Rendering

1. In modeling step, artists create components of game environment off-line using software such as 3DS Max, Maya, and Photoshop
2. Graphics systems generate illusion of movement on screen by quickly displaying sequence of changing images, called *frames*
3. Artists and programmers each implement “half” of animation
    1. Artists implement off-line by rigging and generating animation frames using 3DS Max and Maya
    2. Programmers apply object’s animation created by artists, physics-based simulation, collision, and animation of lighting conditions and camera(s)
4. After conclusion of animation step, rendering step is invoked to generate a 2D image [that makes up a frame] from the 3D scene

CPU and GPU (1/4)

- What are the abstract functions of graphics program?
  1. Define 3D scene [once at start of application, called frame 0]
  2. Display scene for frame 0
  3. Update 3D scene by handling interactions [top of frame i]
  4. Display scene for frame i
  5. Go to step 3 for frame $i + 1$



CPU and GPU (2/4)

- Define 3D scene [at start of application or zero frame]
  - Construct hierarchical data structure that defines object composition in scene
  - Assign geometric data to elements of hierarchical data structure
  - Define interaction events to be handled and functions for handling them
  - Define camera parameters
  - All previous steps implemented on CPU

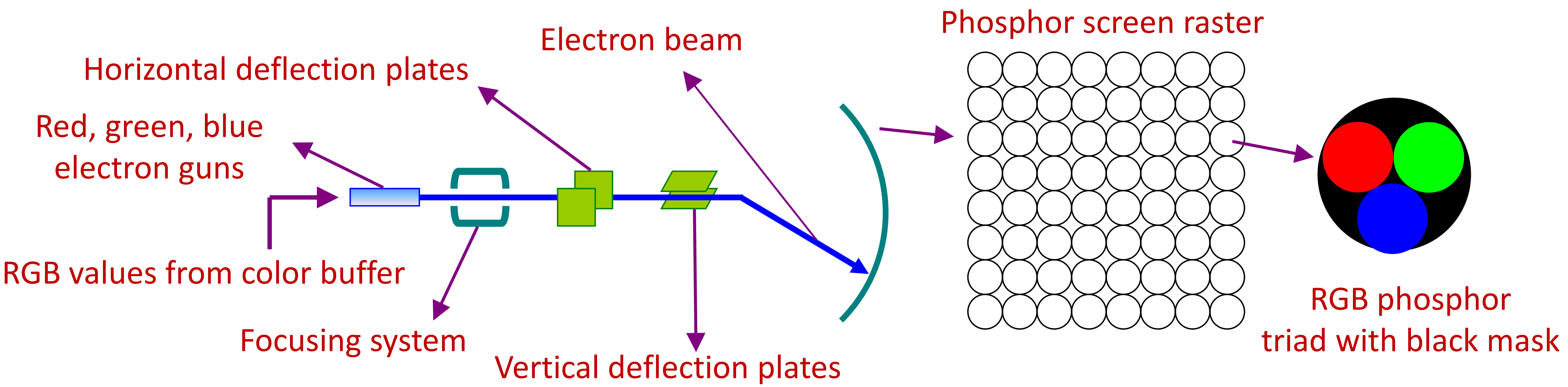
CPU and GPU (3/4)

- Display scene
  - CPU uses camera position and orientation to determine what portion of 3D scene is visible
  - For each visible object, CPU sets render state values using low-level graphics API functions [such as OpenGL or Direct3D]
  - Graphics APIs provide application programmers essential graphics functions
  - Such functions are implemented in hardware, named GPU, which is a processor specialized for graphics
  - Think of graphics API as a software interface of GPU
  - GPU driver will translate low-level graphics API functions to machine language instructions that can be executed by GPU

CPU and GPU (4/4)

- Update scene [at top of each frame]
  - Handle interactions that modify scene
  - Respond to interactions, either programmed [camera’s position and orientation] or event-driven [e.g., AI]
  - Specify new position and orientation for each object in scene

Output Devices: Refresh Video Display



1. Typical CRT will emit light for only short time after phosphor excited by electron beam
2. For humans to see steady, flicker-free image, phosphor must be refreshed at sufficiently high rate – the *refresh rate*
3. Modern TVs and monitor have refresh rate of 120Hz

Refresh Rate vs Frame Rate

- Refresh rate: Number of frames per second displayed by output device
- Frame rate: Number of frames per second generated by graphics system
- Output video is smoother if frame rate divides cleanly into refresh rate

Animation

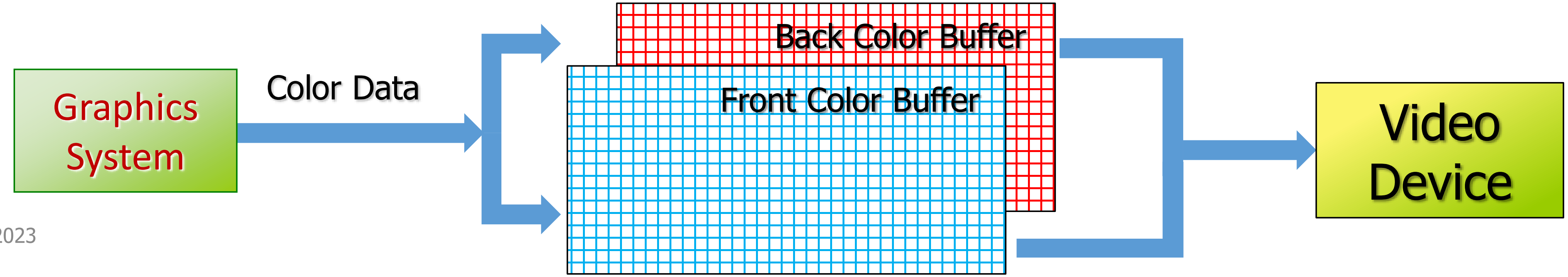
- *Animation* means drawing of moving objects
- Movement is visual illusion
  - Animation achieved by drawing succession of still scenes or frames
  - Each frame shows static snapshot at an instant of time
  - Typically 30 fps and more ideally 60 fps adequate for illusion of smooth motion on screen

Single Buffering

- Suppose frame rate is 30 fps while refresh rate is 60 fps
- No coupling between when *animated* scene is rendered into CB and when CB is redisplayed by output device
- Thus, only part of animated scene is rendered into CB when CB is redisplayed
- Single buffering causes partially rendered scenes to be displayed

Double Buffering

- *Double buffering* is graphics rendering technique that generates successive frames cleanly
- Image in [visible] *front buffer* is displayed on output device while graphics system is rendering next frame into [off-screen] *back buffer*
- When rendering is complete, two buffers are *swapped* so completed rendering is displayed as front buffer and rendering can begin anew in back buffer
  - Swapping buffers doesn't require copying from one buffer to other; instead, pointers are updated to switch identities of front and back buffers



Next …

- Introduction to 3D rendering pipeline
- Introduction to OpenGL