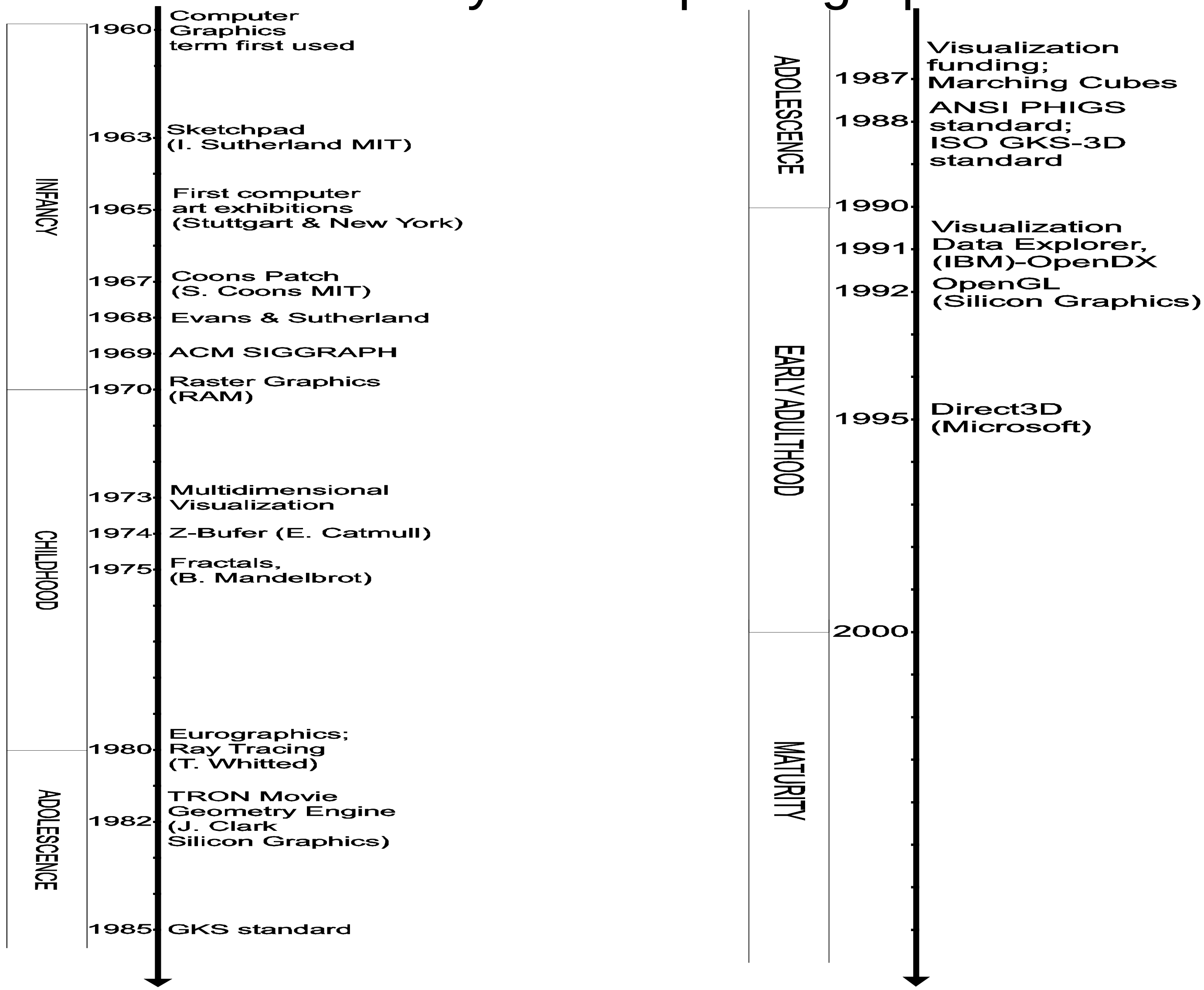
Graphics & Visualization

Chapter 1

Introduction

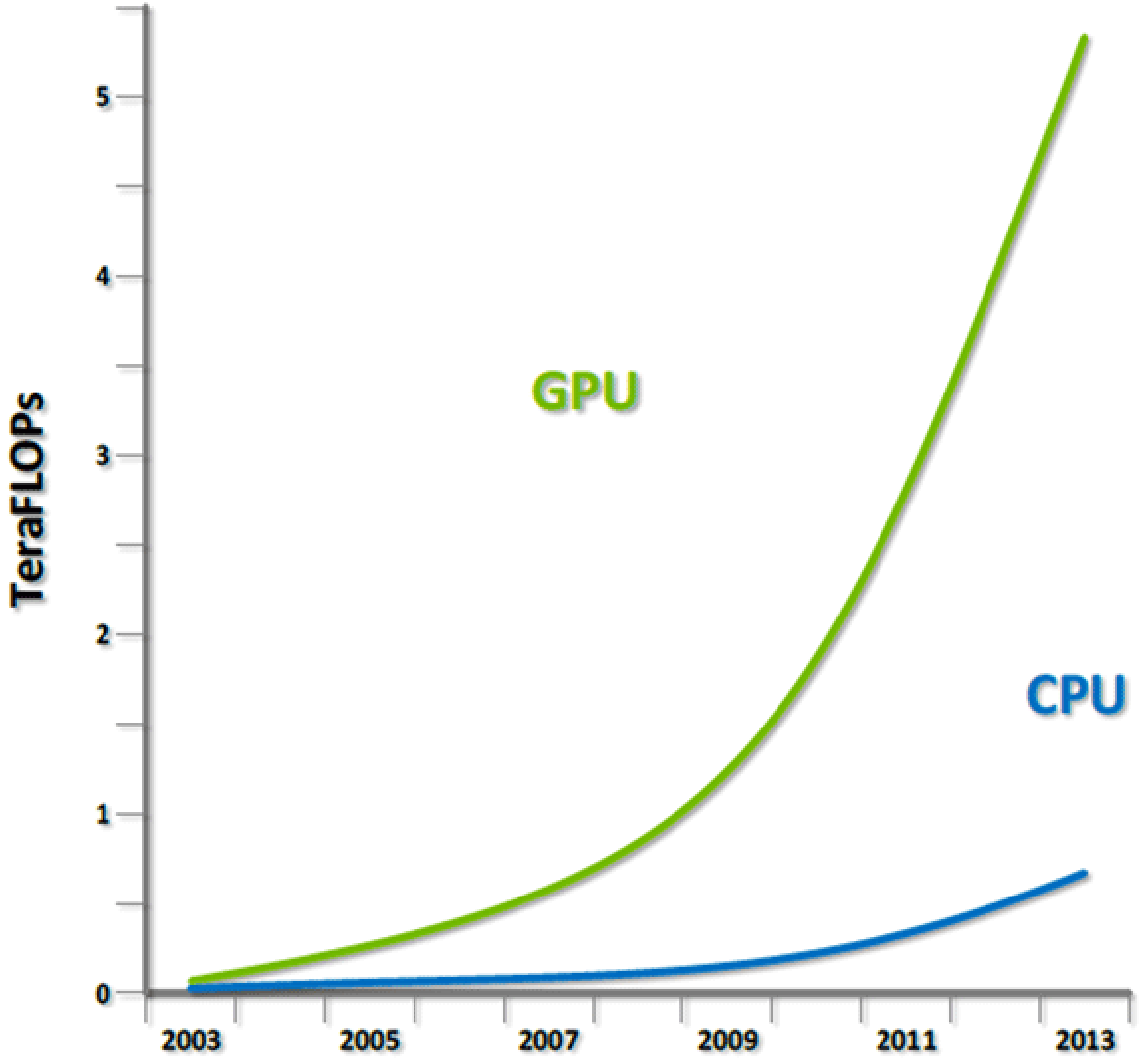
Brief History

- Milestones in the history of computer graphics:



Brief History (2)

- CPU Vs GPU

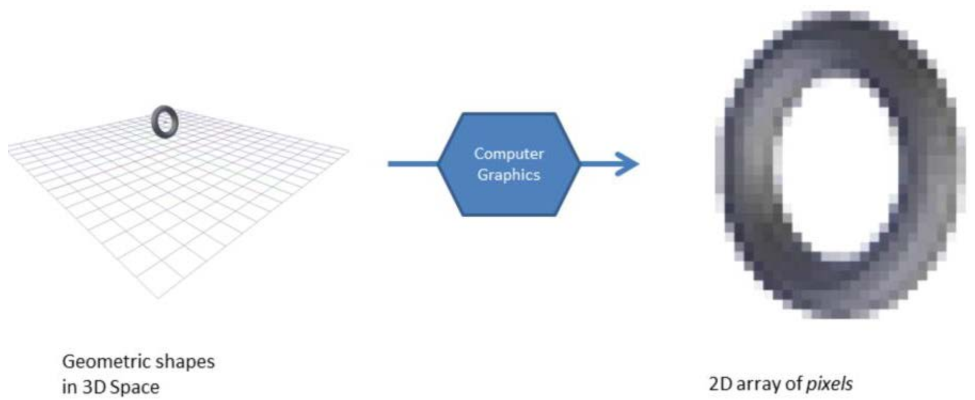


Applications

- Computer and mobile games – the driving force
- Special effects – films and advertisements
- Graphical user interfaces (GUIs)
- Data visualization – enabling scientific exploration
- Interactive simulation – Virtual reality, flight simulation etc
- Computer-aided design – design and test before building
- Computer art

Concepts

- 3D or 2D scenes are composed of primitives (e.g. points, lines, curves, polygons, mathematical solids or functions)
- A **raster image** is a 2D array of pixels
- **Computer Graphics** use principles and algorithms to generate from a scene, a raster image that can be depicted on a display device
- **Scene ⇨ Computer Graphics ⇨ Raster Image**



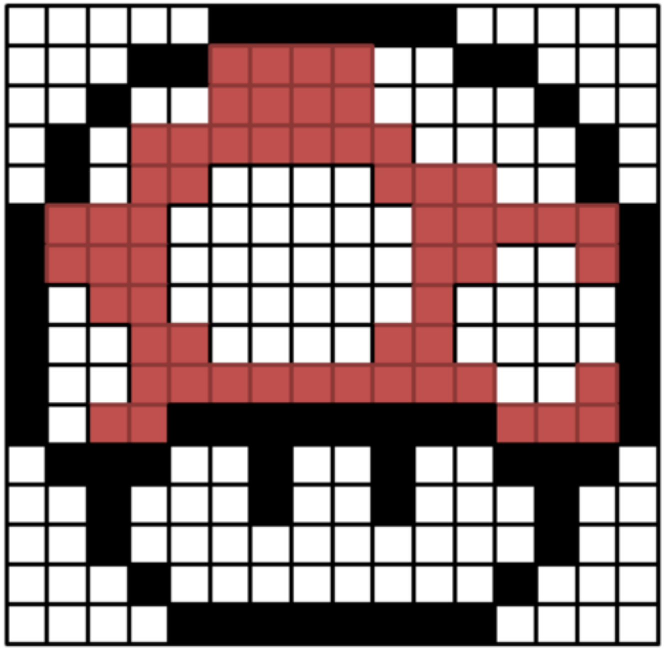
Concepts (2)

- **Visualization** exploits visual presentation of large data sets to increase understanding
- The result of visualization is a **visualization object**
- **Modeling** encompasses techniques for the representation of graphical objects
- **Data Set** ⇨ **Visualization** ⇨ **Model**
- **Graphics Pipeline** is a sequence of stages that create a digital image out of a model
- **Model** ⇨ **Graphics Pipeline** ⇨ **Image**

Concepts (3)

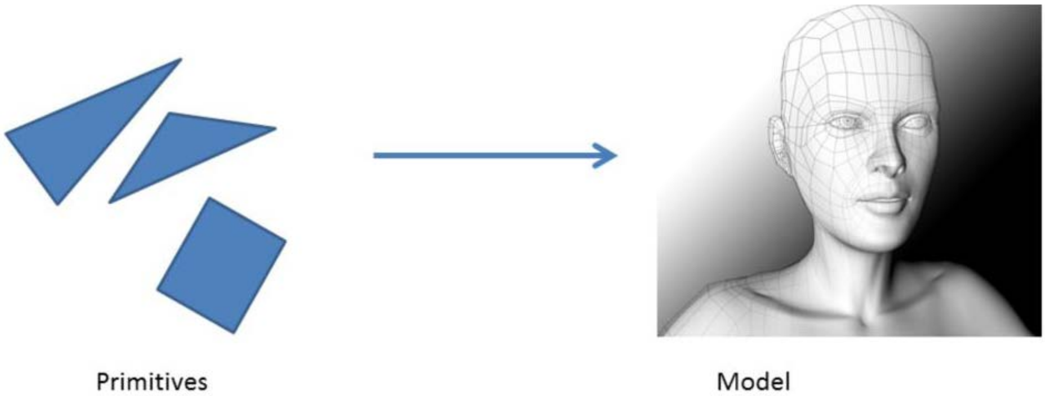
```
FFFFFFFFFF000000000000FFFFFFFFFF
FFFFFF000777777777FFFF0000FFFFFF
FFFF00FFFF77777777FFFFFFFF00FFFF
FF00FF77777777777777FFFFFFFF00FF
FF00FF7777FFFFFFFF777777FFFF00FF
00777777FFFFFFFFFFFF777777777700
00777777FFFFFFFFFFFF7777FFFF7700
00FF7777FFFFFFFFFFFF77FFFFFFFF00
00FFFF7777FFFFFFFF7777FFFFFFFF00
00FFFF777777777777777777FFFF7700
00FF7777000000000000000077777700
FF000000FFFF00FFFF00FFFF000000FF
FFFF00FFFFFF00FFFF00FFFFFF00FFFF
FFFF00FFFFFFFFFFFFFFFFFFFF00FFFF
FFFFFF00FFFFFFFFFFFFFFFF00FFFFFF
FFFFFFFF0000000000000000FFFFFFFF
```

Visualization

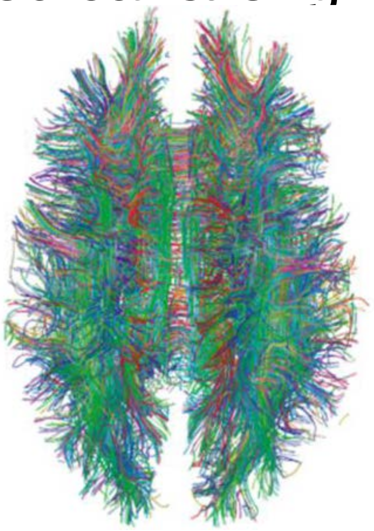


Concepts (4)

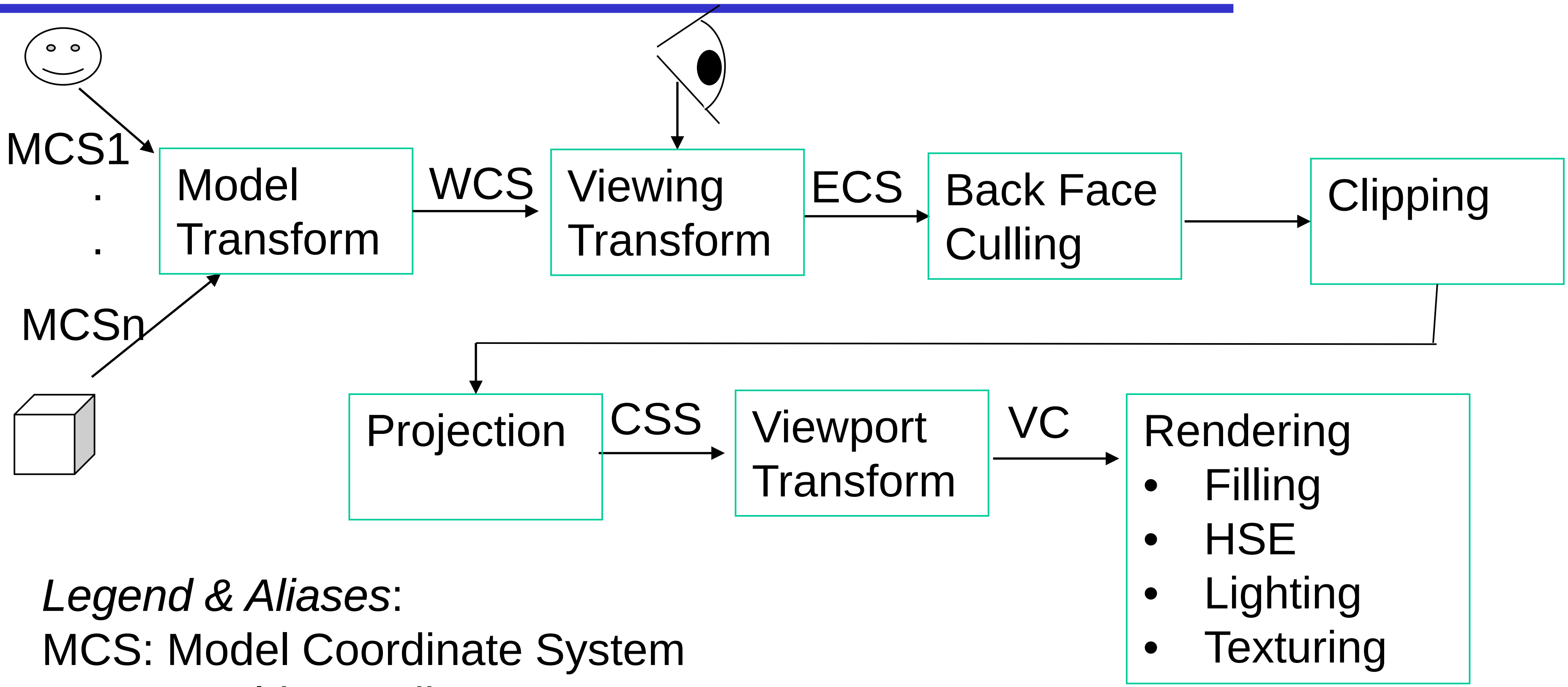
- Boundary models: games, special effects…



- Solid models: medical & engineering simulations, CAD, 3D printers



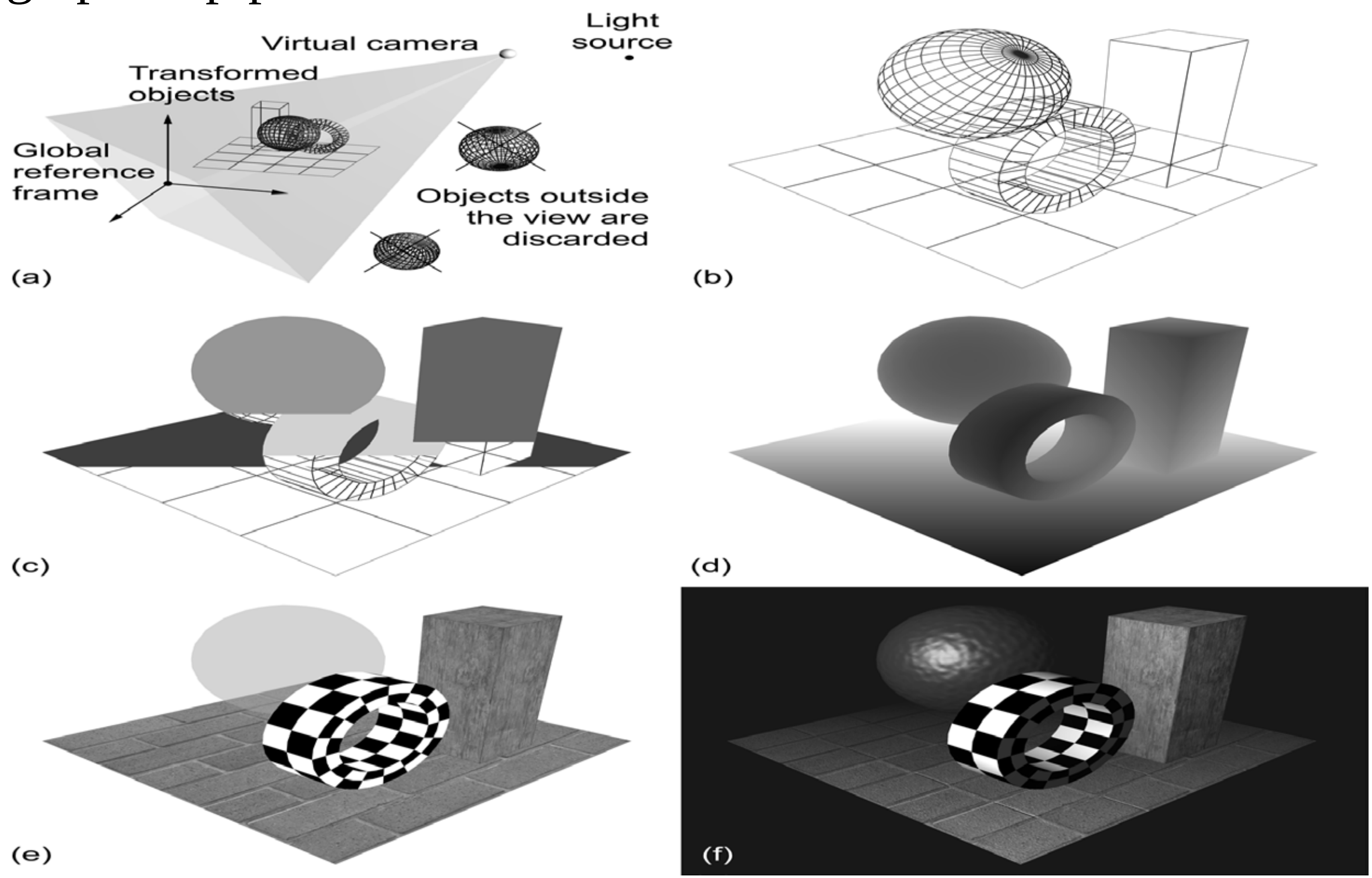
Graphics Pipeline: Conceptual



Legend & Aliases:
MCS: Model Coordinate System
WCS: World Coordinate System
ECS: Eye Coordinate System = VCS: View Coordinate System
CSS: Canonical Screen Space=NDC: Normalised Device Coordinates
VC: Viewport Coordinates = DC: Device Coordinates = SC: Screen C.
Projection=Perspective Division
Rendering=Rasterization
Lighting= Shading

Graphics Pipeline (2)

- Operations on primitives in the standard direct rendering graphics pipeline :



Graphics Pipeline (3)

- For the above example:

(a) Geometry transformation to a common reference frame and view frustum clipping.

(b) Primitives after viewing transformation, projection, and backface culling.

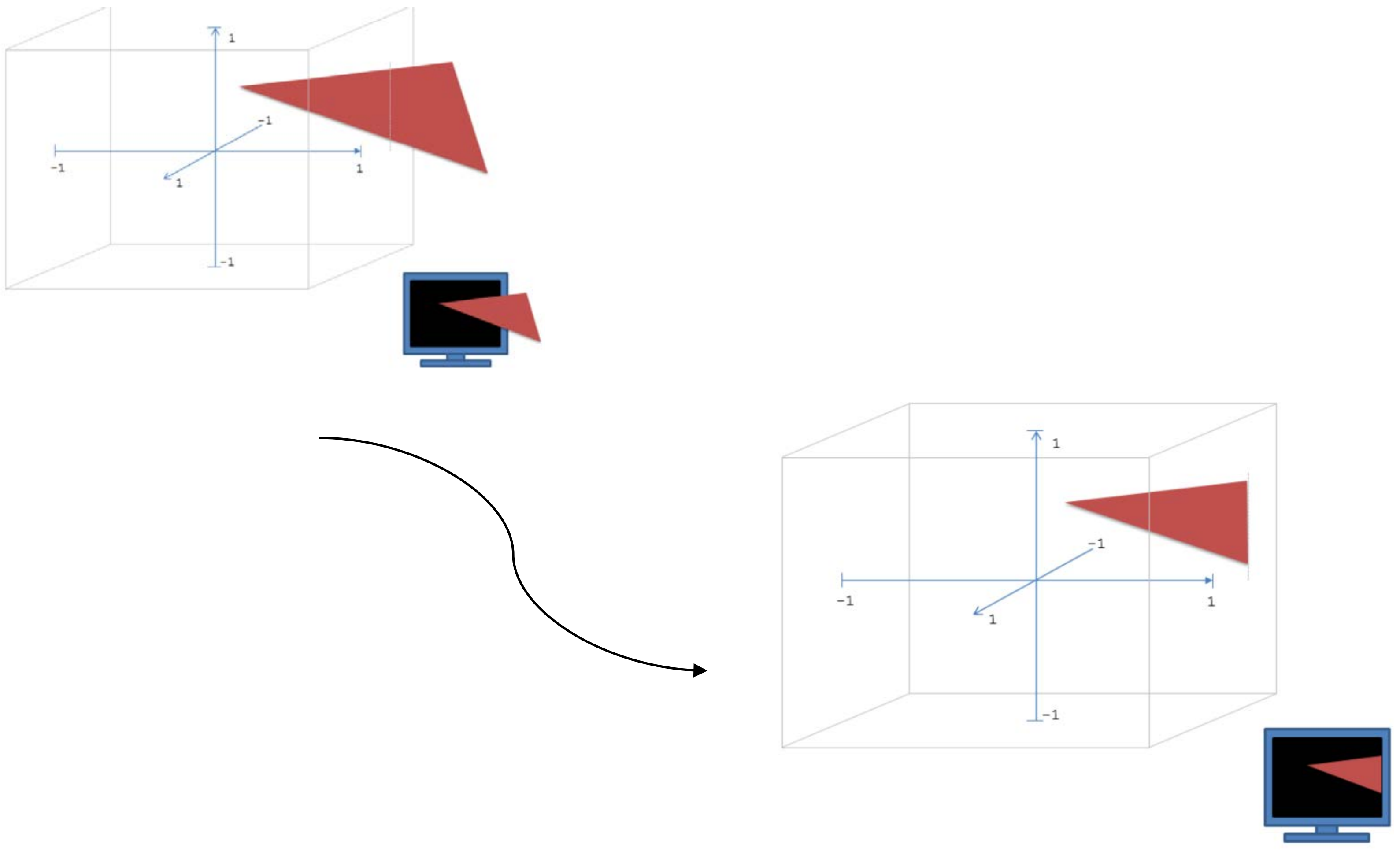
(c) Rasterization

(d) fragment depth sorting: the darker a shade, the nearer the corresponding point is to the virtual camera.

(e) Texturing.

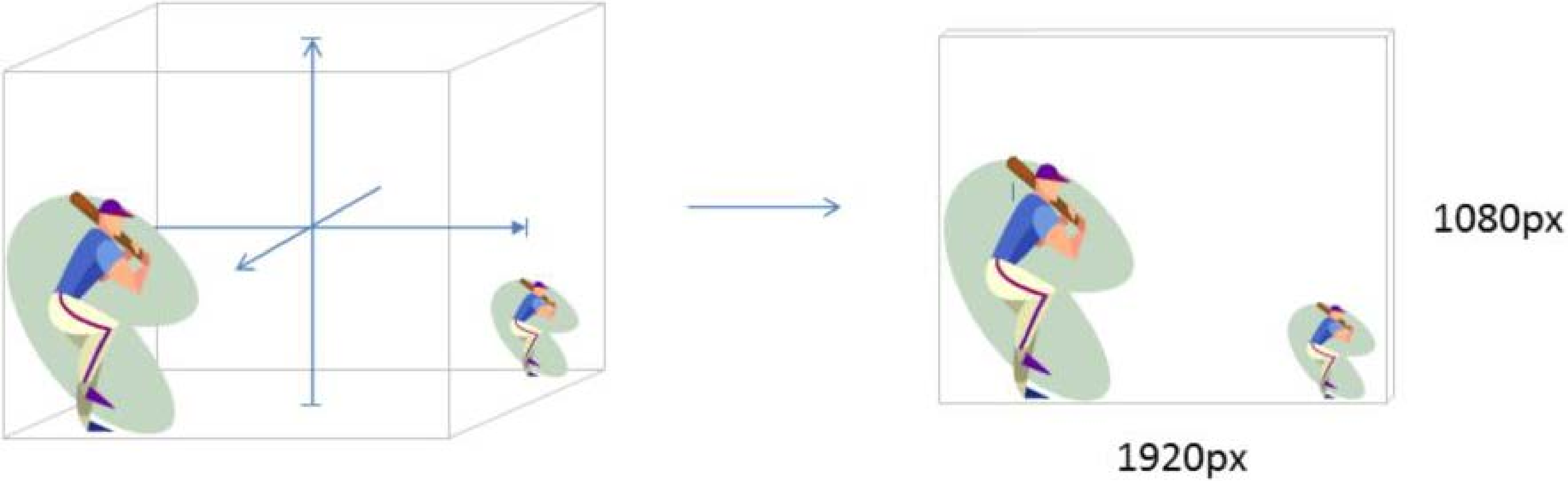
(f) Lighting and other fragment operations (such as fog).

Clipping



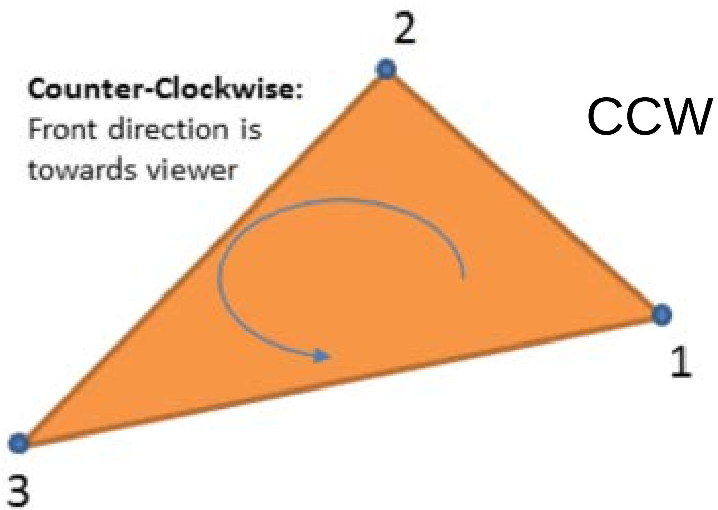
Viewport Transform

(Translate & Scale)



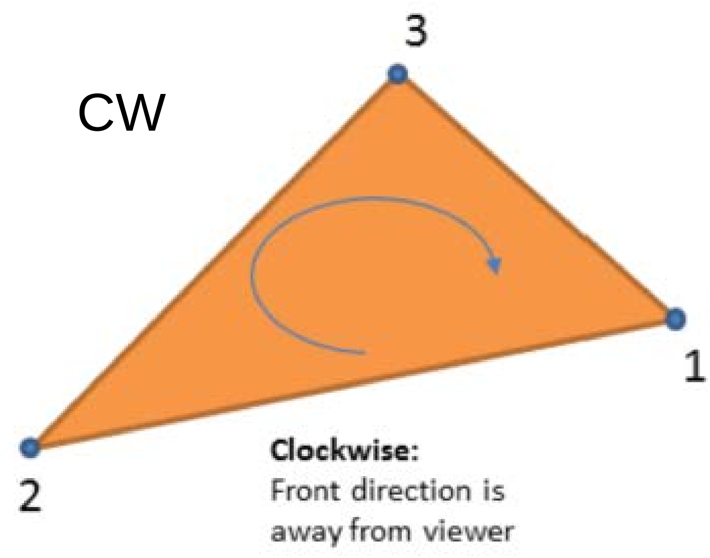
Triangle Sides & Back Face Culling

Triangle vertices must be taken in a *unique order*



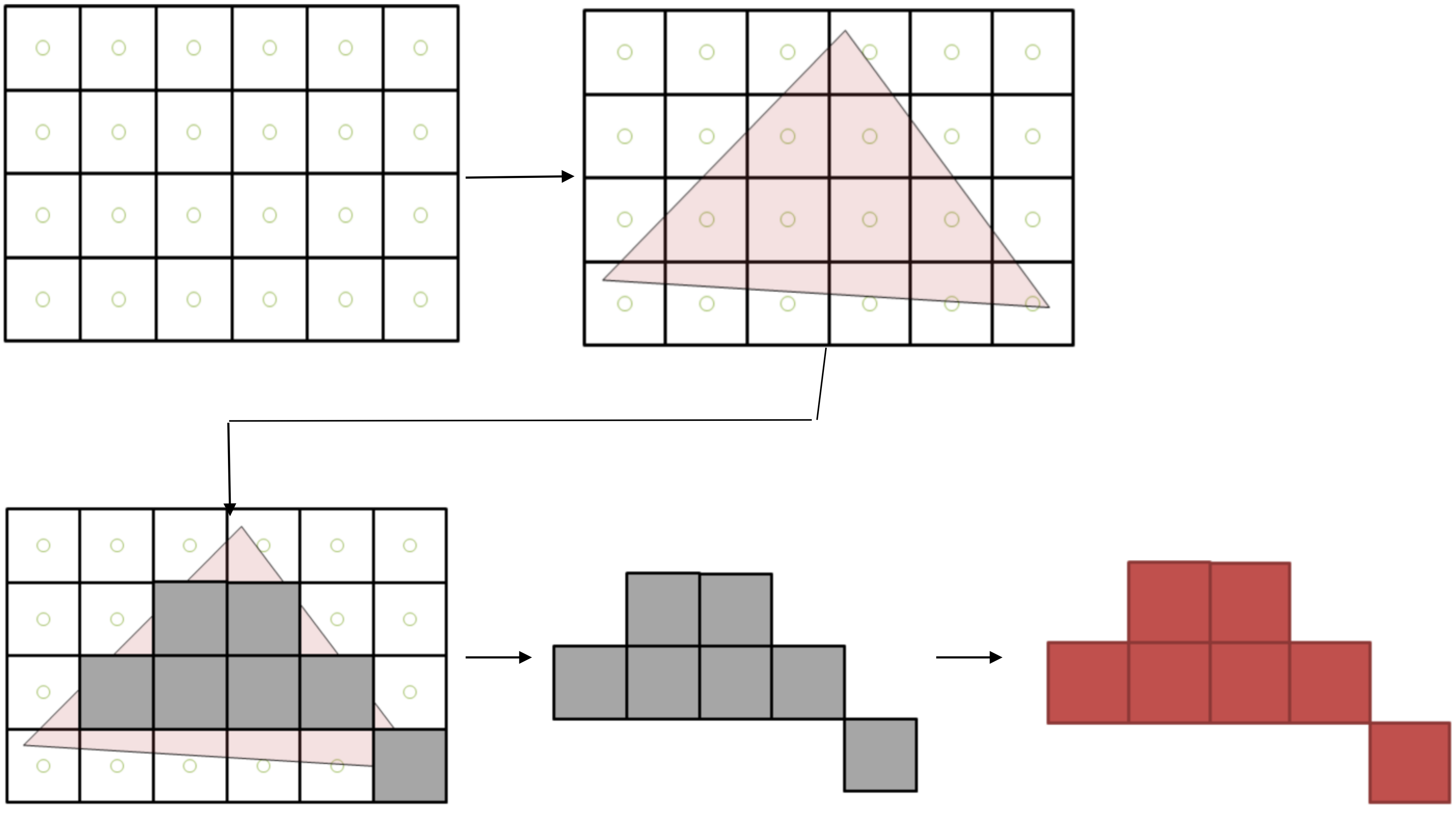
CCW

CW

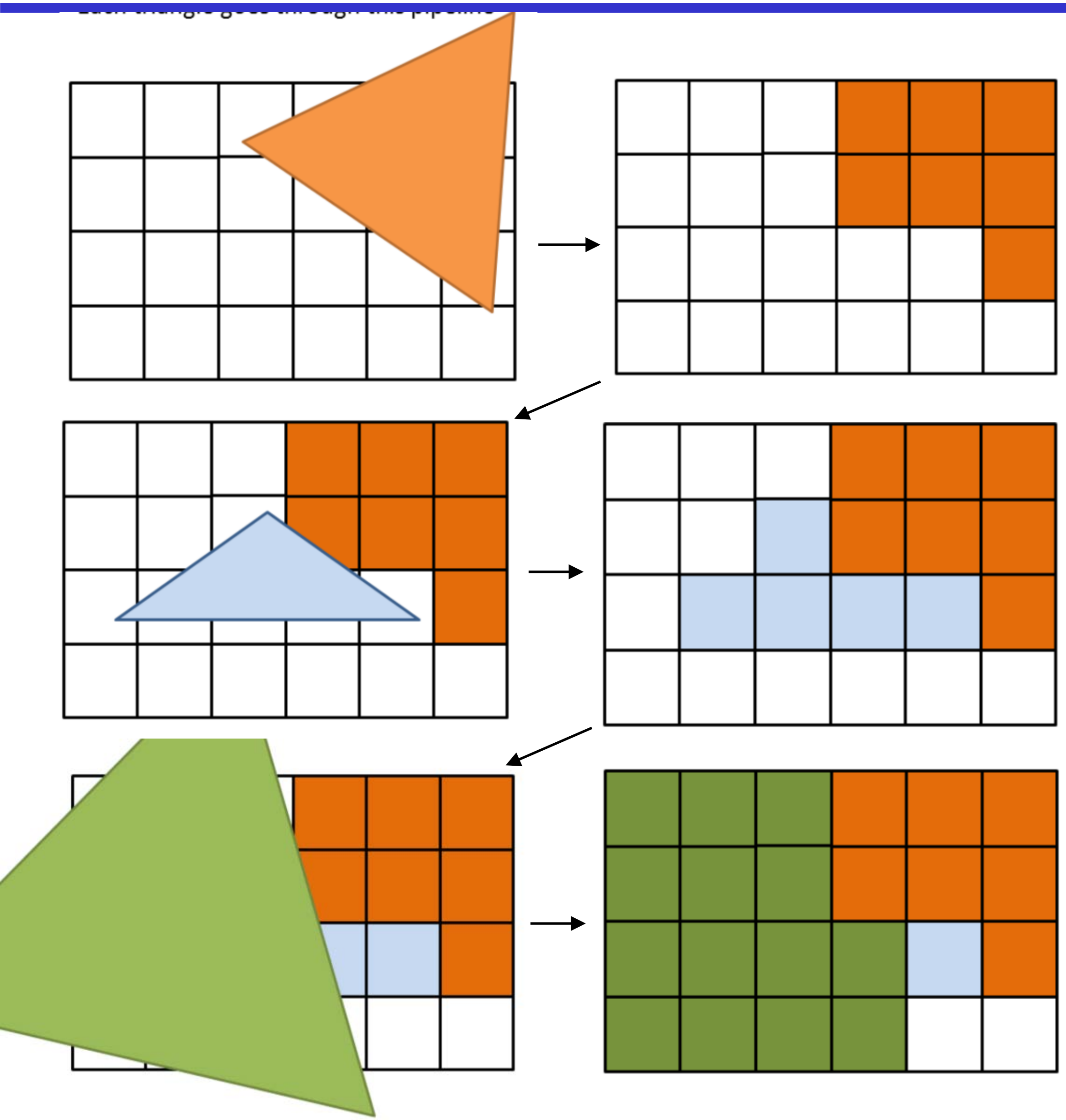


Rasterization

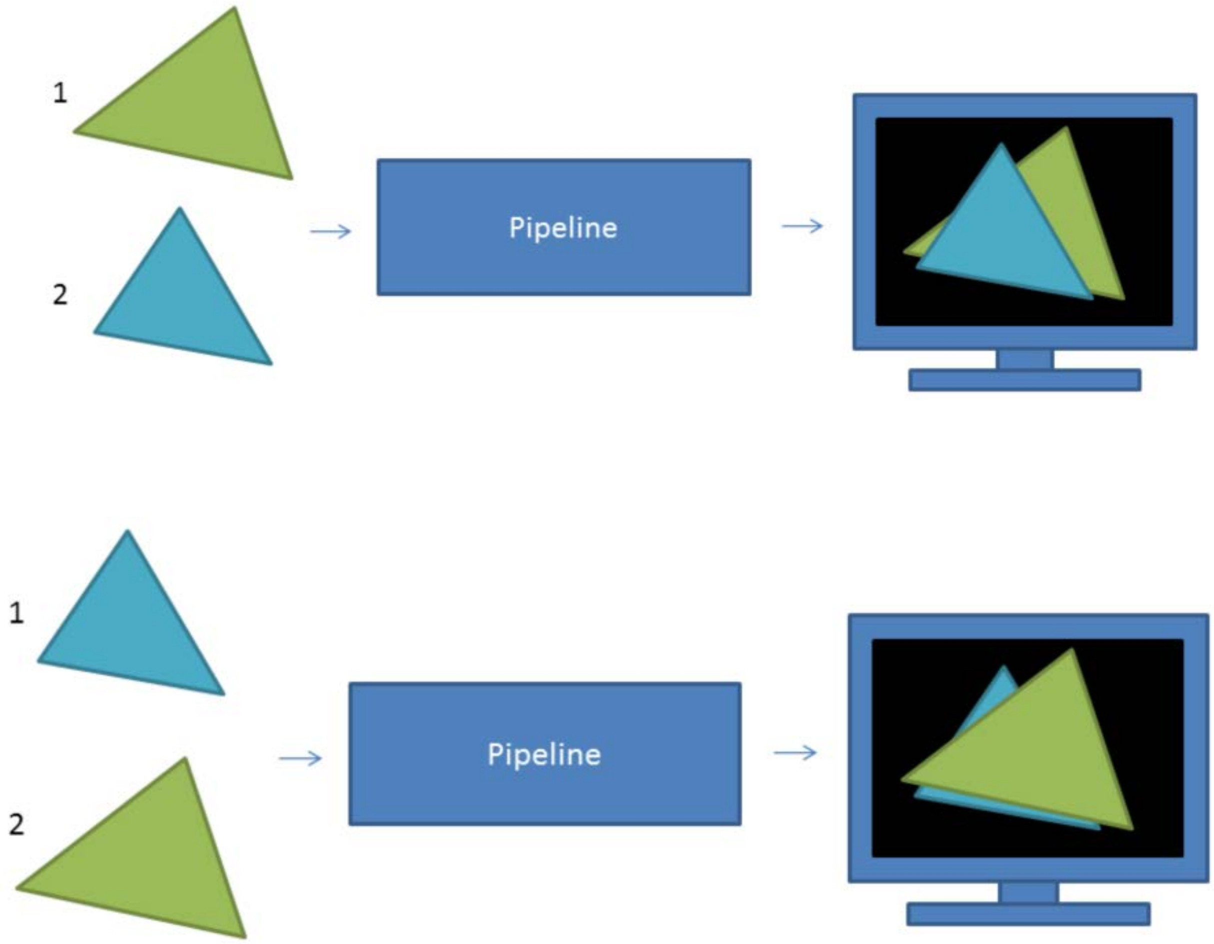
Turn Shapes into Fragments (elements within a pixel)



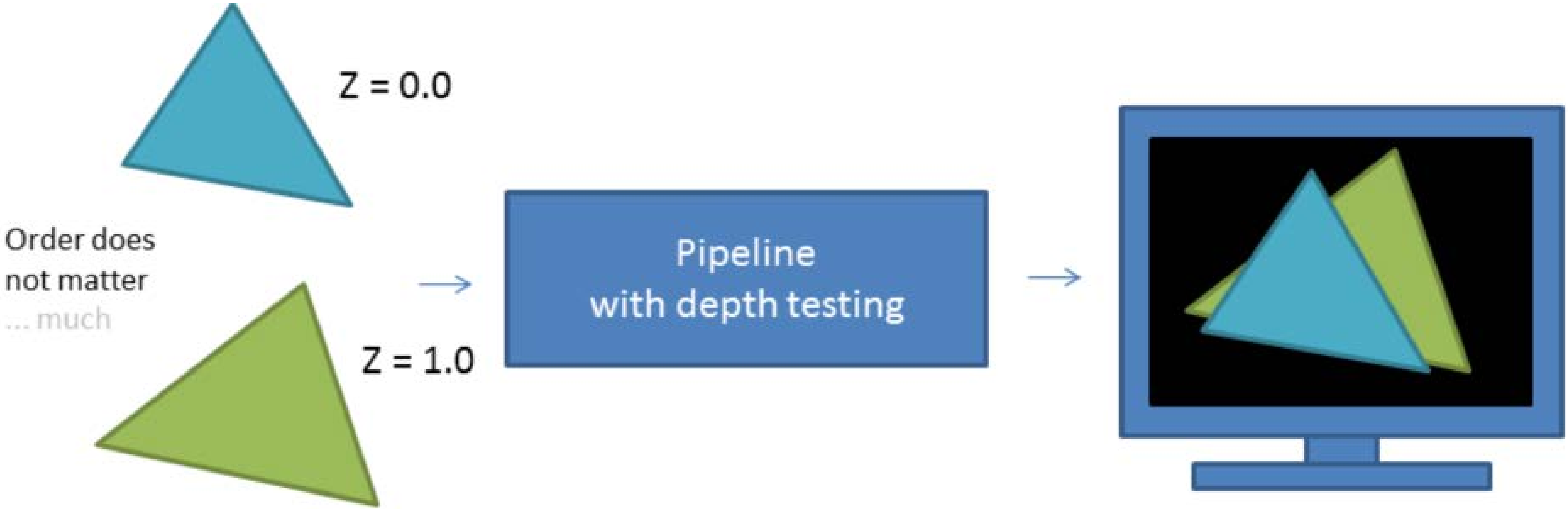
Each Triangle goes through Pipeline



Drawing Order Matters

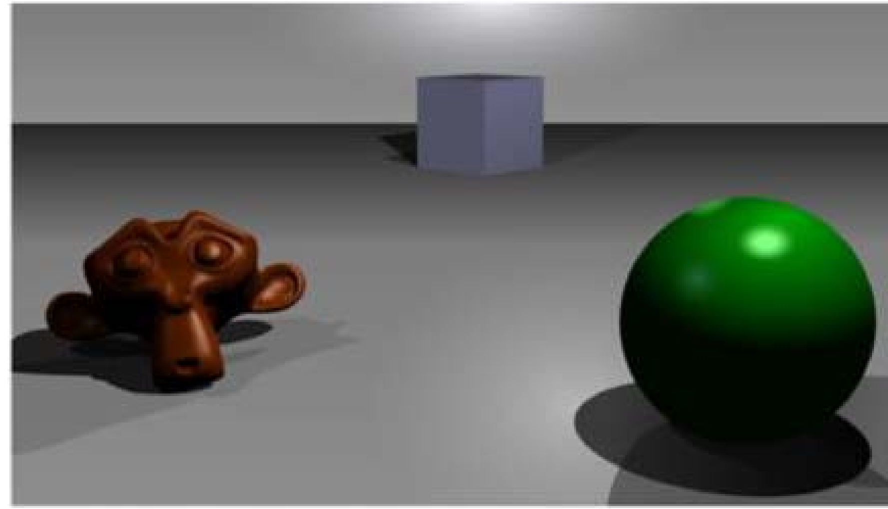


Z-Testing Makes Order NOT Matter



Depth (Z) Buffer

Typically 24bpp,
Same resolution as frame buffer
- 0.0 -> at near clip plane
- 1.0 -> at far clip plane



A simple three-dimensional scene

Z-buffer representation

Z fighting

Sometimes appears during depth testing

- Due to numerical accuracy
- e.g. two overlapping planes

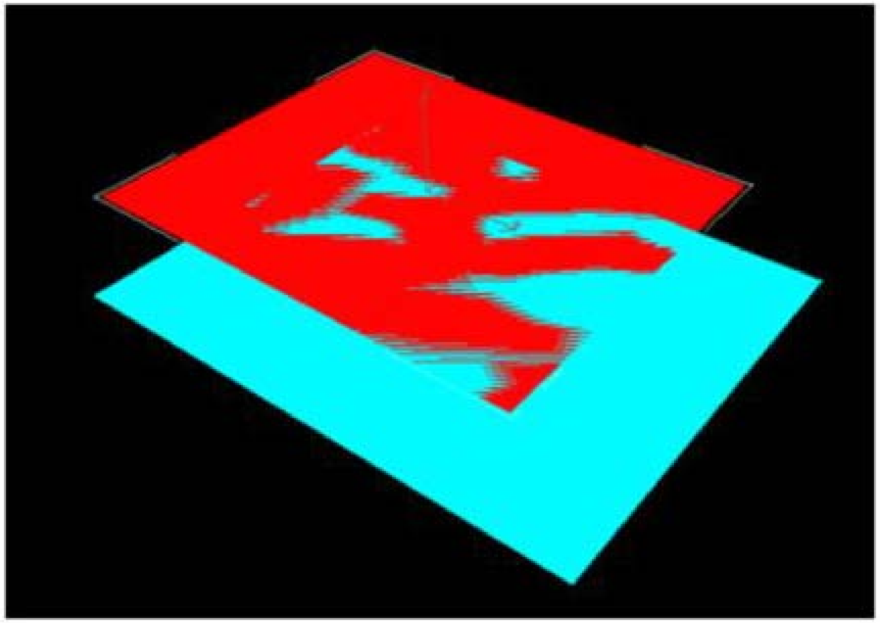


Image Buffers

Storage and Encoding of Digital Images :

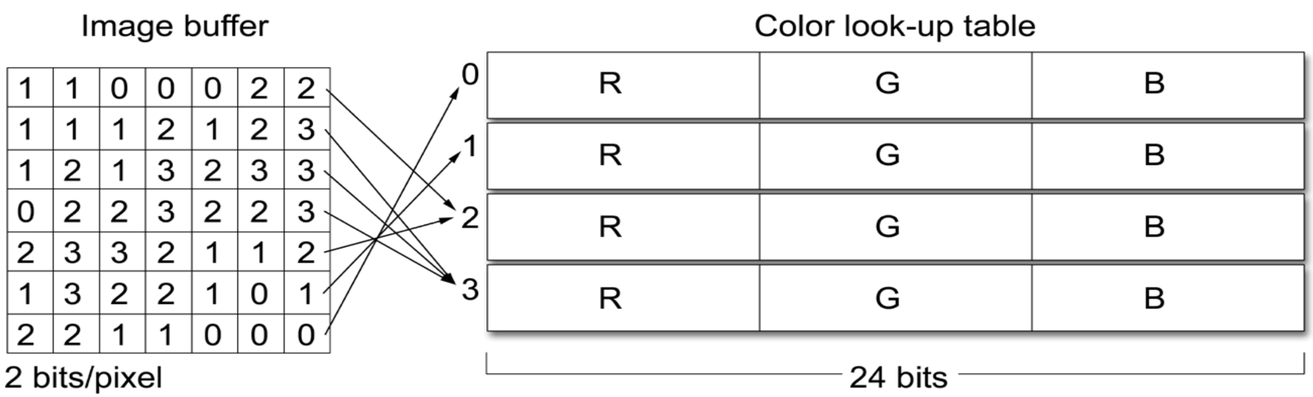
- **Image buffer** is a 2D array of dimensions w x h
- Size of the image buffer is at least (*w x h x bpp*) / 8 bytes
- **Color depth (bpp)** : # bits used to store the color of each pixel
- Color representations:
  - Monochromatic (grayscale)
  - Multi-channel (red/green/blue)
  - Palleted (CLUT)

True-color

- **True-color**: image buffer stores full color intensity information of each pixel
- **Color look – up table** (CLUT):
  - bits per pixel do not affect the accuracy of the displayed color

Image Buffers (2)

- Image buffer with CLUT:



- Image buffers occupy contiguous space of memory

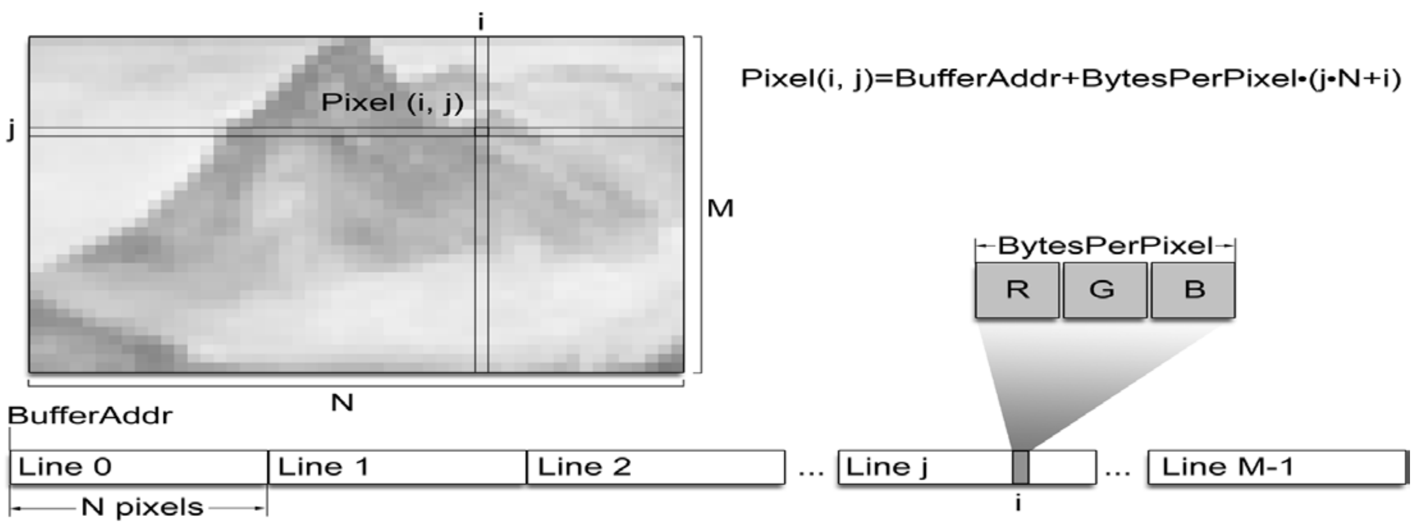


Image Buffers (3)

Frame Buffer:

- memory where all pixel color information from rasterization is accumulated before being driven to the graphics output
- **double buffering**

Depth Buffer or Z-buffer:

- stores distance values
- used for hidden surface elimination

Other Buffers:

- Stencil Buffer
- Accumulation Buffer

Framerates

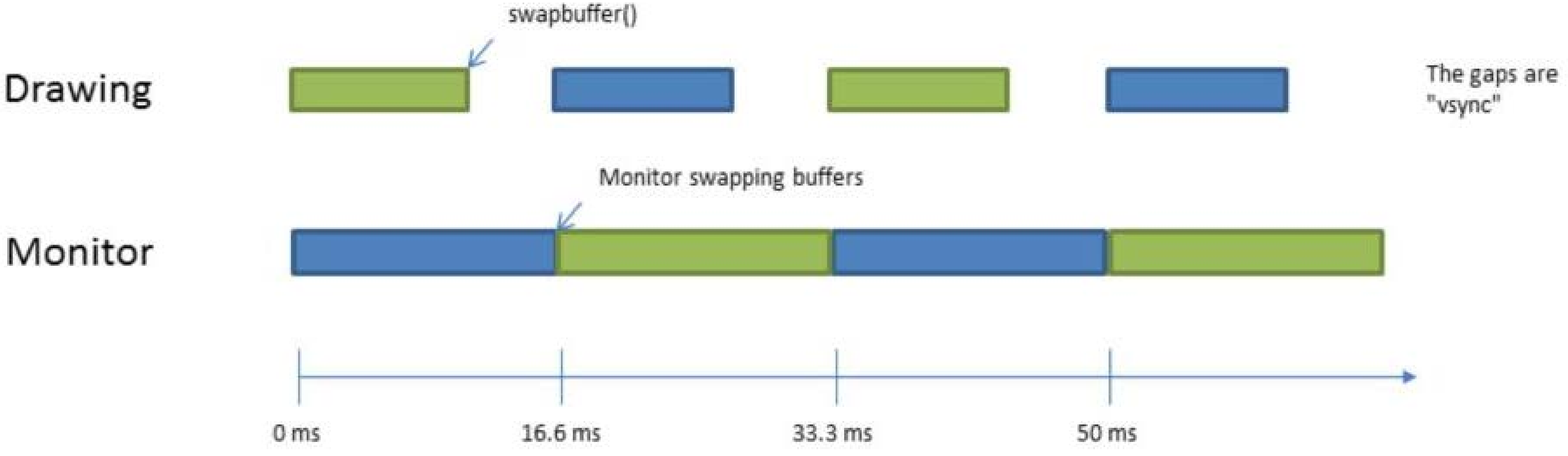
- **Eye** perceives smooth motion at >= 30 fps
- **Monitor** needs refreshing at constant rate (typically 60 fps)
- **Application** produces frames as fast as it can
  - No point going beyond monitor refresh rate
- Display controller (typically on GPU)
  - Reads frame buffer and feeds monitor at 60 fps

Tearing

- **Tearing:** when application writes to frame buffer at the same time the display controller is reading

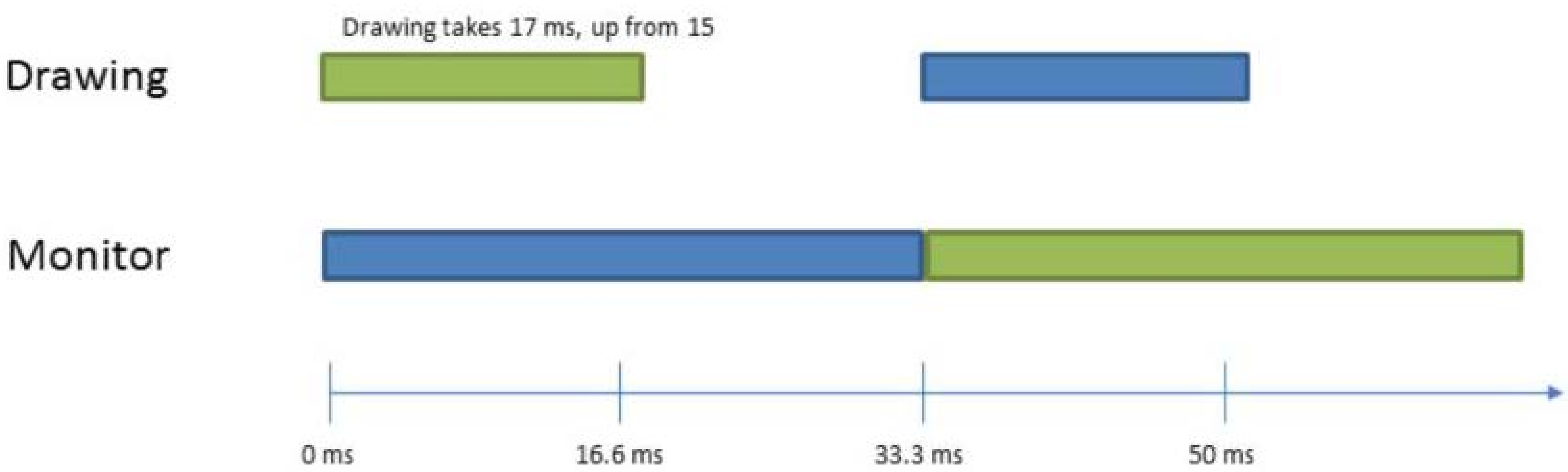
Double Buffering

- Avoids tearing
  - Frame buffer 1 is read by display controller (every 16.6ms for 60fps)
  - Frame buffer 2 is written by application



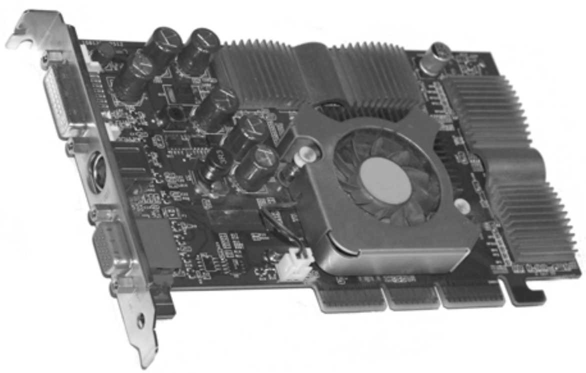
Double Buffering (2)

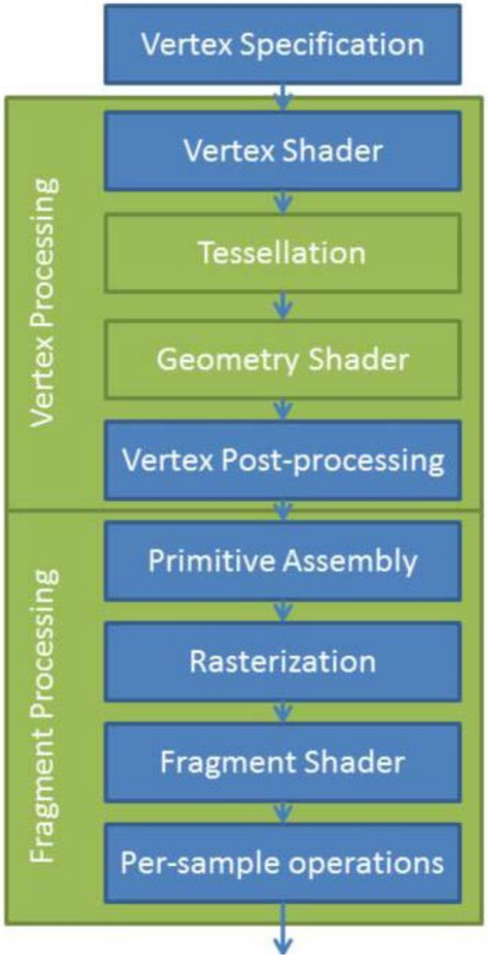
- Buffer swap rate is an integer multiple of monitor refresh rate
  - If Drawing takes 17ms, buffer swap every 33.3ms



Graphics Hardware

- After a ~20 year development, we have
  - Programmable graphics hardware
  - 2 types of shaders-programs (vertex, fragment)





Conventions

- Scalars : x, y, z
- Vector quantities :
  - Points: **a**, **b**
  - Vectors: $\vec{\mathbf{a}}, \vec{\mathbf{b}}, \overrightarrow{\mathbf{Oa}}$
  - Unit vectors: $\hat{\mathbf{e}}_1, \hat{\mathbf{n}}$
- Matrices: **M**, $\mathbf{R}_x$
  - Column vectors: $\vec{\mathbf{v}}^{\mathrm{T}} = [0,1,2]$
- Functions:
  - Standard mathematical functions and custom functions: sin(θ)
  - Functions follow the above conventions for scalar and vector quantities
- Norms: $\left|\vec{v}\right|$
- Standard sets : $\mathbb{R}, \mathbb{C}$
- Algorithm descriptions are given in pseudocode based on standard C and C++
- Advanced sections are marked with an asterisk $\circledast$