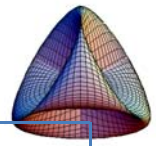


Comp5514

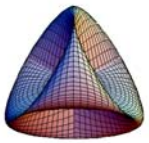
Computer Graphics

Lecture 02

www.comp.polyu.edu.hk/~csgeorge/comp5514/lec

George Baciu PQ825

csgeorge@comp.polyu.edu.hk x7272



2

Raster

Image Construction

1

Graphics

Pipeline Systems

3

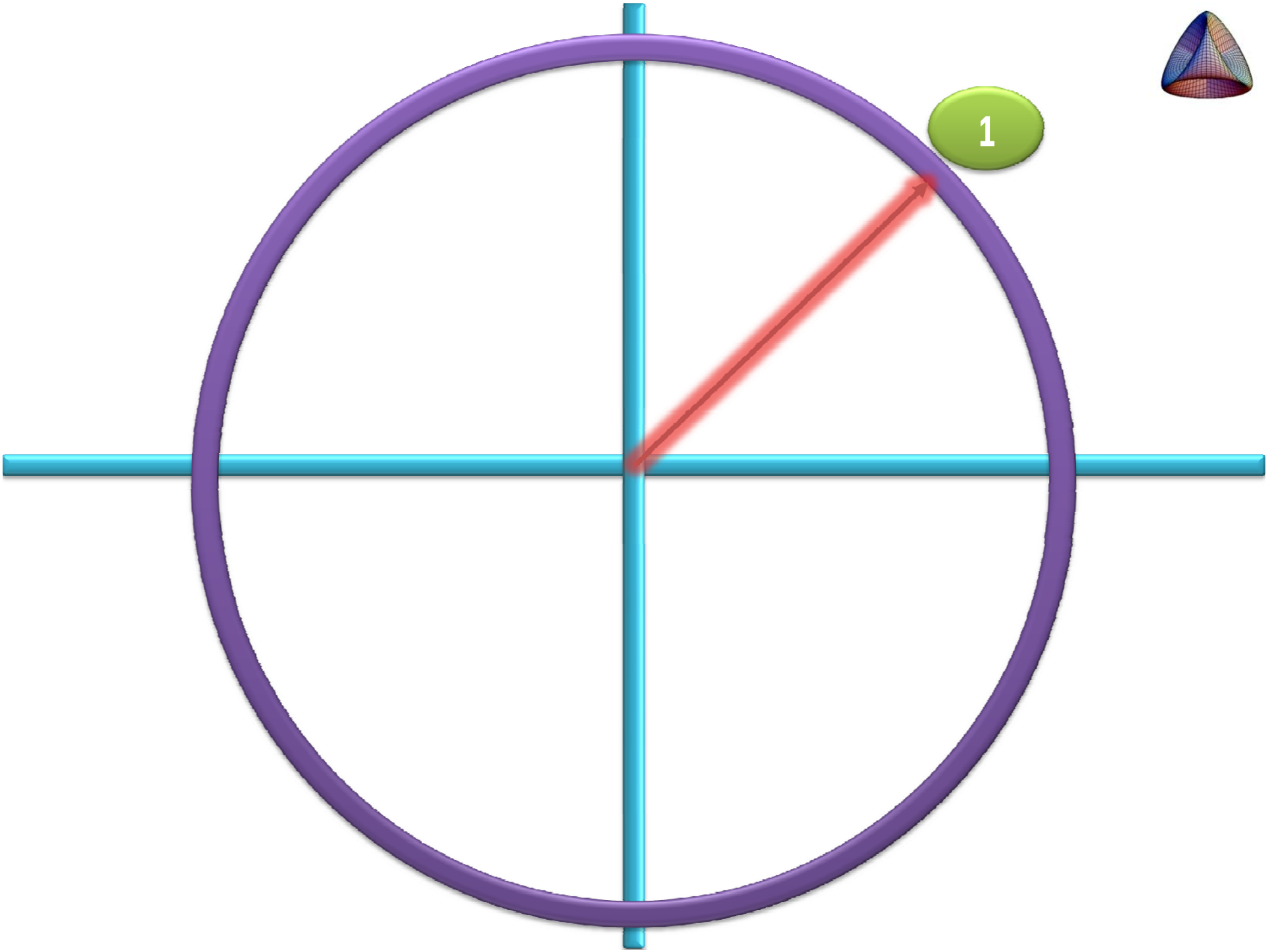
Frame

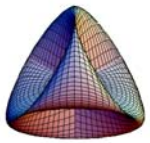
Buffer Imaging

4

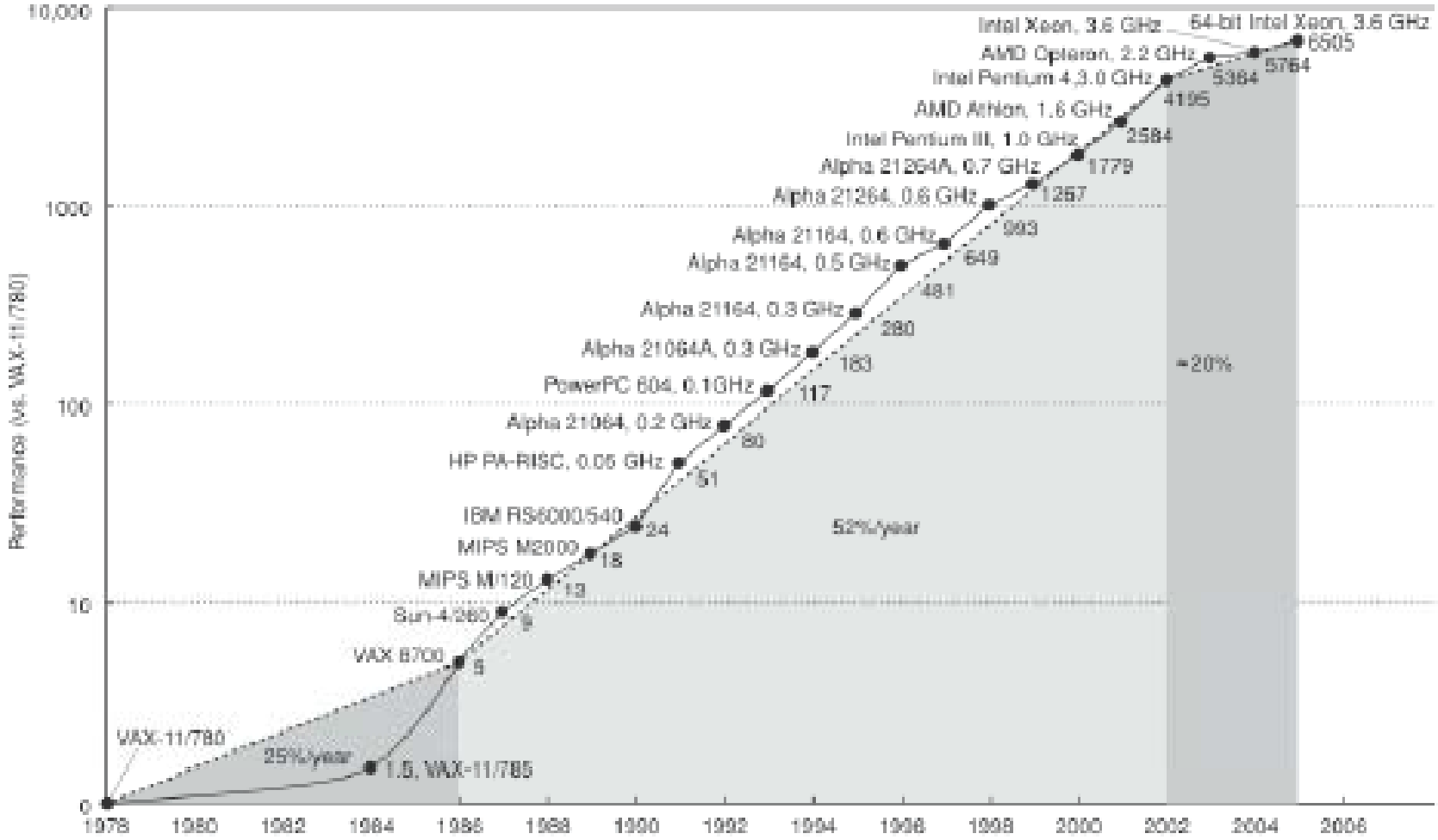
Interactive

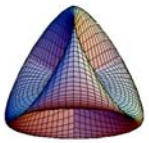
API Programs


1



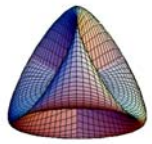
Single Processors



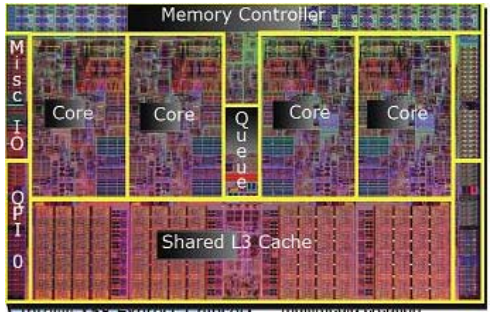


GP vs GPUs

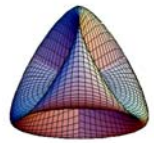
Embedded Parallelism



- Major CPU vendors support multi-core.
- Interest in General Purpose (GP) programmability on GPUs.
- Industry moving:
  - From: "instructions per second"
  - To: "instructions per watt"



GPs M-cores



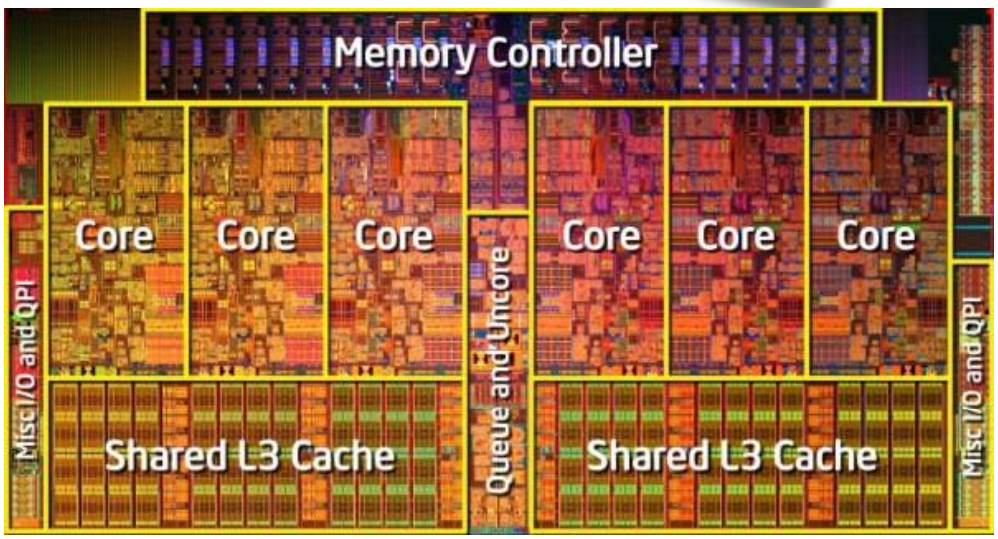
GP:

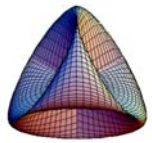
General Processor

Single Core: Atom N475

Multi-core: Core i7

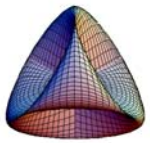
- 6 cores (980-X Extreme)
- 8 processing threads
- SSE4.2 instruction set
- Integrated memory
- 3 Channel DDR3 1066 MHz





SSE4 Instructions

- Streaming SIMD Extensions (ver. 4.2)
- Intel and AMD
  - Floating Point Instructions/Integer Instructions
  - Memory-Register and Cache data movement
  - Arithmetic
  - Compare
  - Data shuffle and unpacking
  - Data type conversion
  - Bit-wise logical ops



GPU

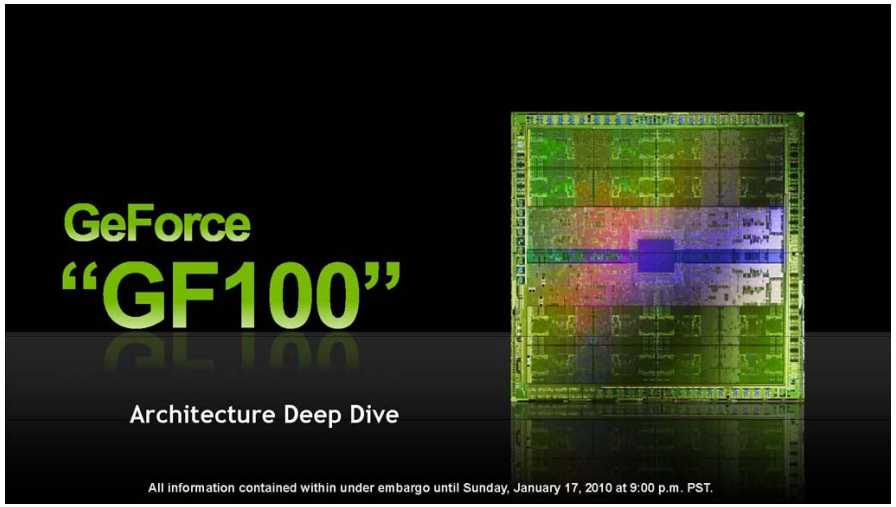
GPU:

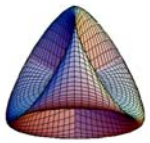
Graphics Processor Unit

Massive parallelism

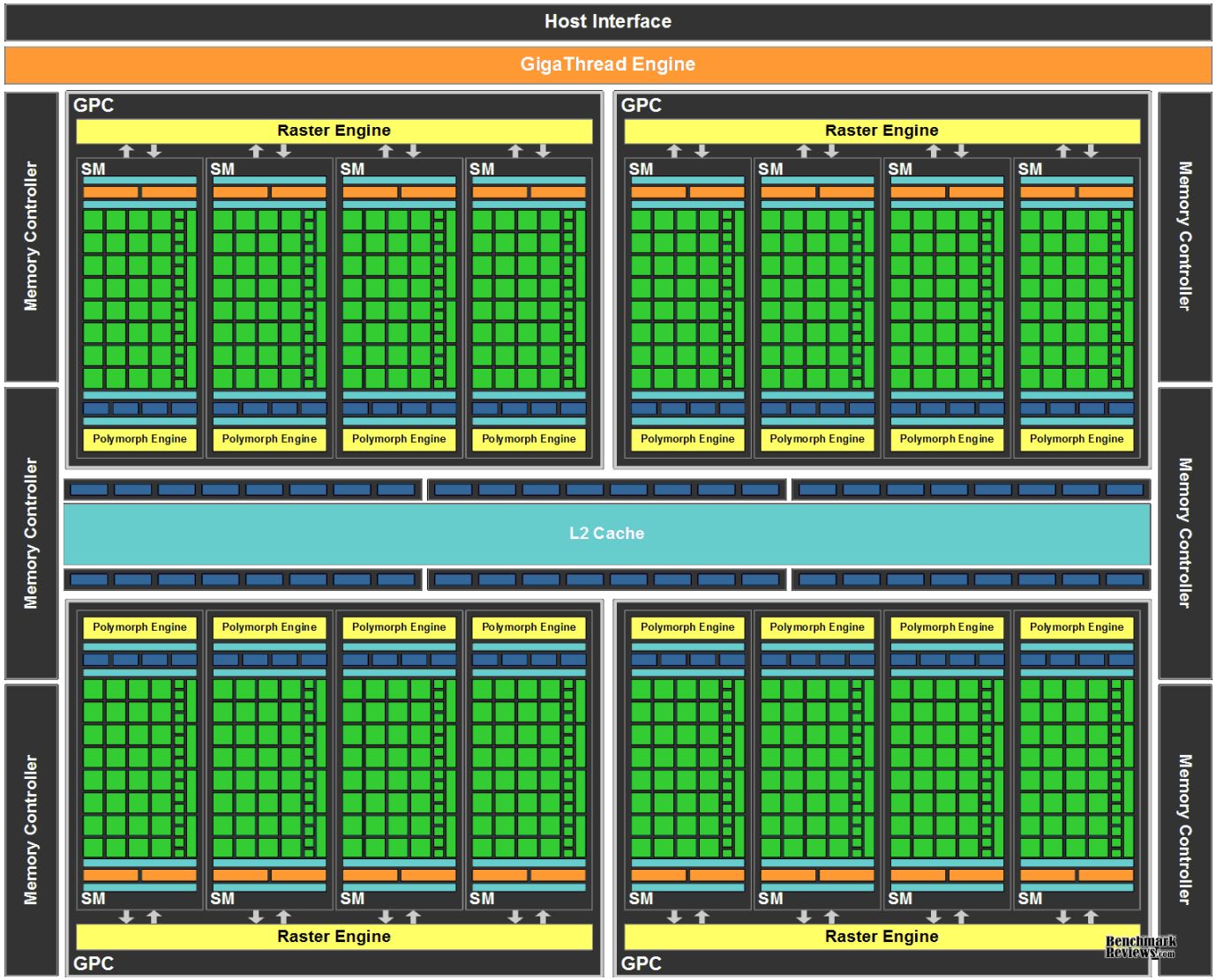
Up to 512 Processing Units

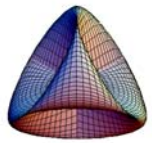
- 8 Polymorph Engines
- 8 Raster Engines
- 3.2B transistors
- 384 bit memory bus
- 1GB Texture memory





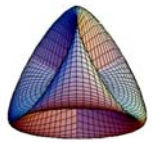
Nvidia Fermi GF100





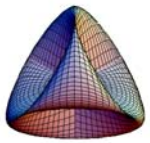
GP vs GPUs?

- The future of desktop is parallel:
  - We are just beginning to see massive embedded parallelism.
- GPUs and Multi-cores are different:
  - Multicore: coarse, heavyweight threads, better performance per thread
  - GPUs: Fine, lightweight threads, single-thread performance is poor.



Why GPUs?

- Interaction with the world is visual.
- GPUs have well-established programming models.
- GPU market is > 500M / year.



Rendering Pipeline

Application

GP

Modeling: Compute 3D geometry
API: calls to graphics API

GPU

Geometry

Transform Geometry from 3D to 2D (in parallel)

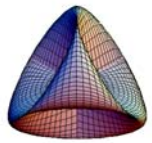
Rasterization

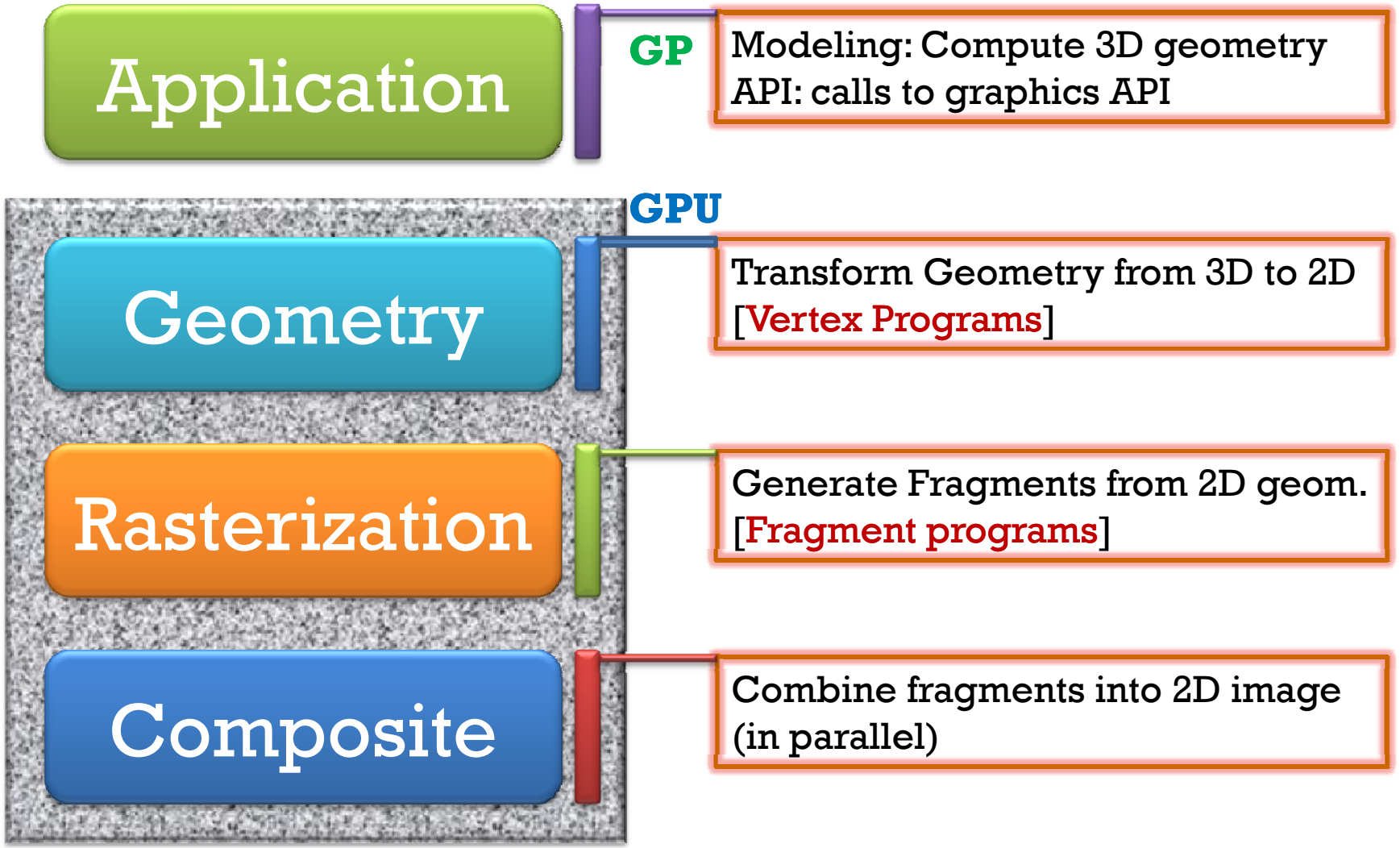
Generate Fragments from 2D geom. (in parallel)

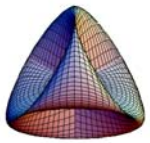
Composite

Combine fragments into 2D image (in parallel)

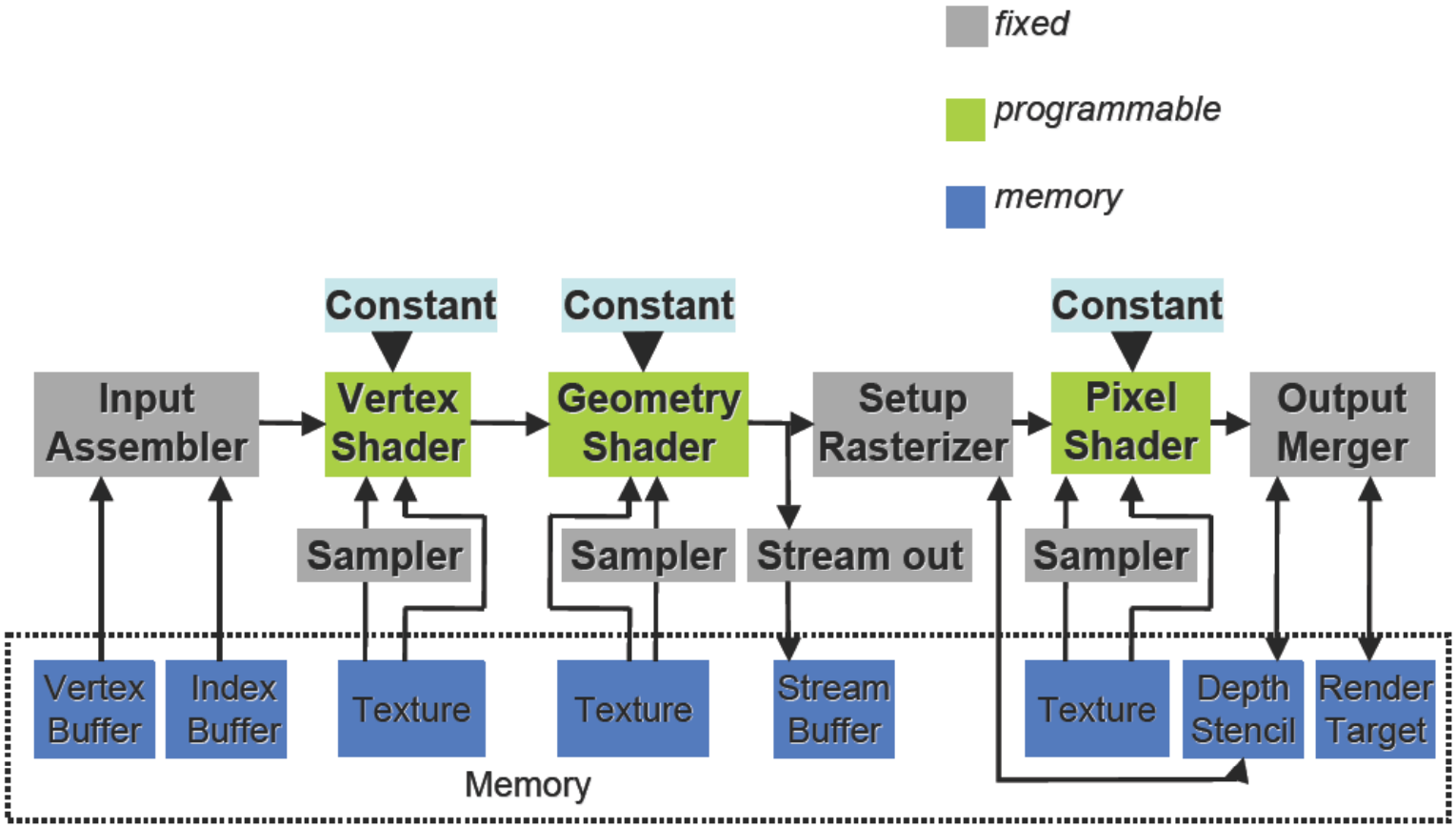
Programmable Pipeline

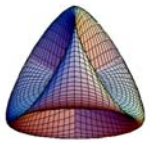




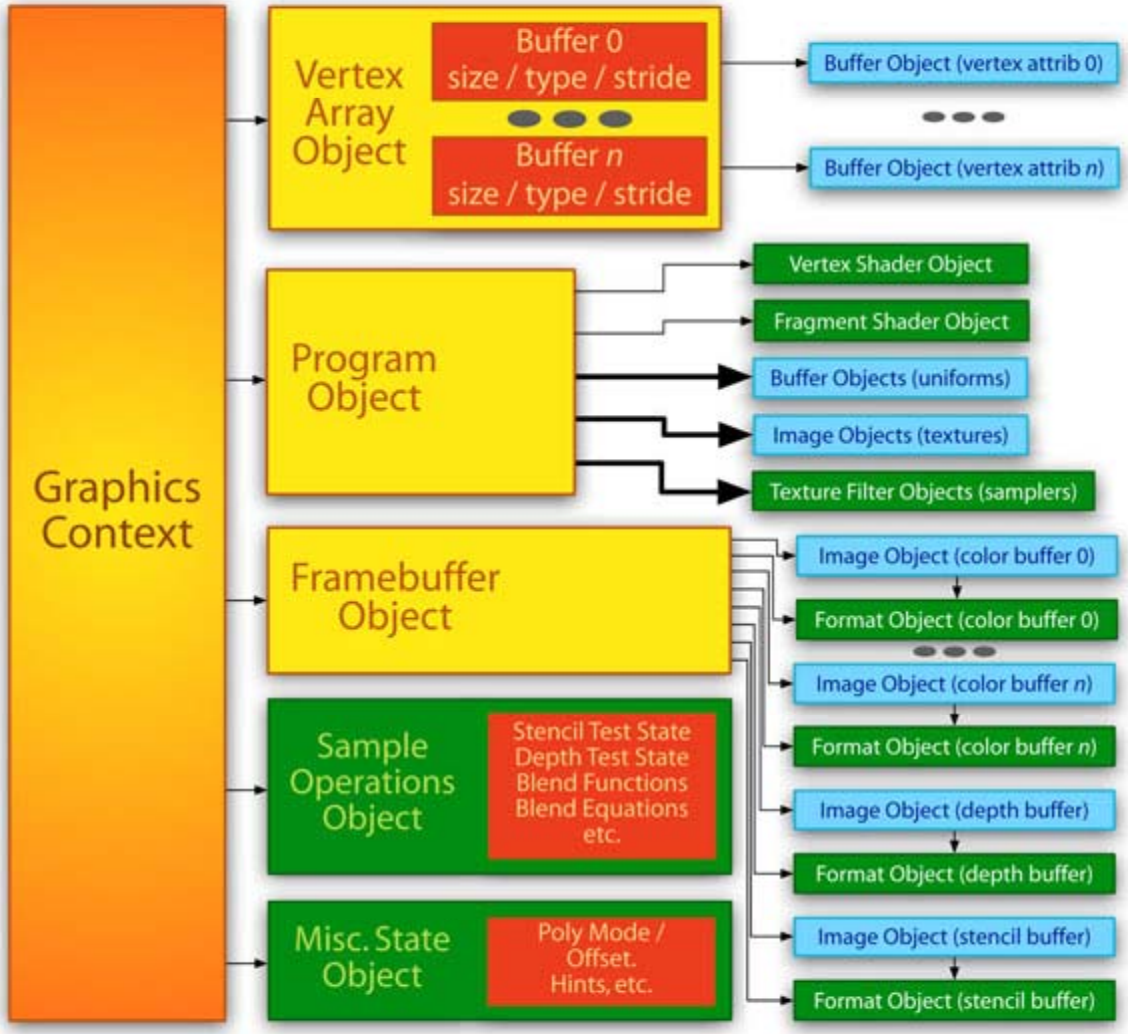


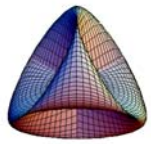
DirectX10 Pipeline



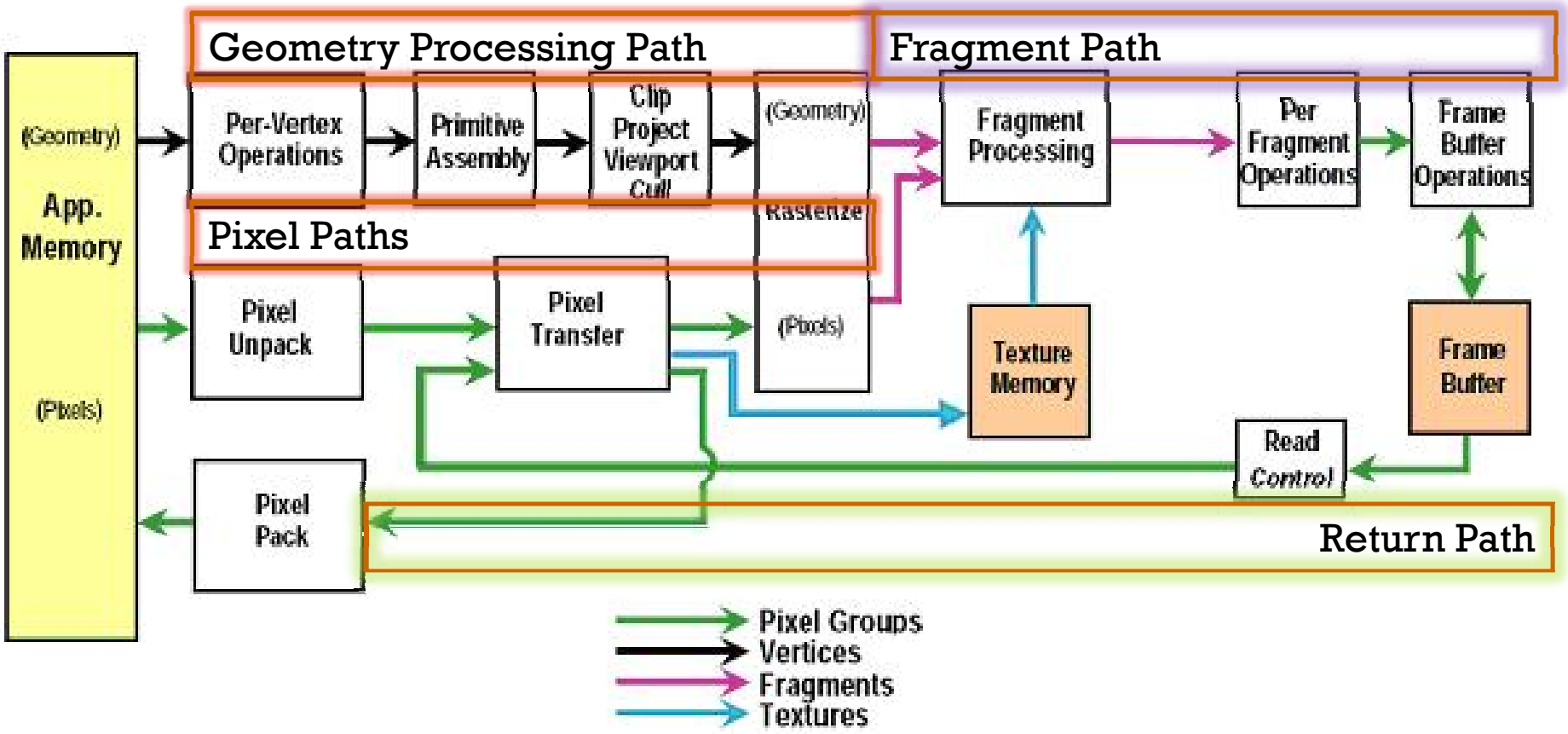


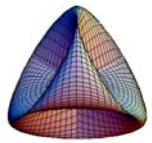
OpenGL Pipeline



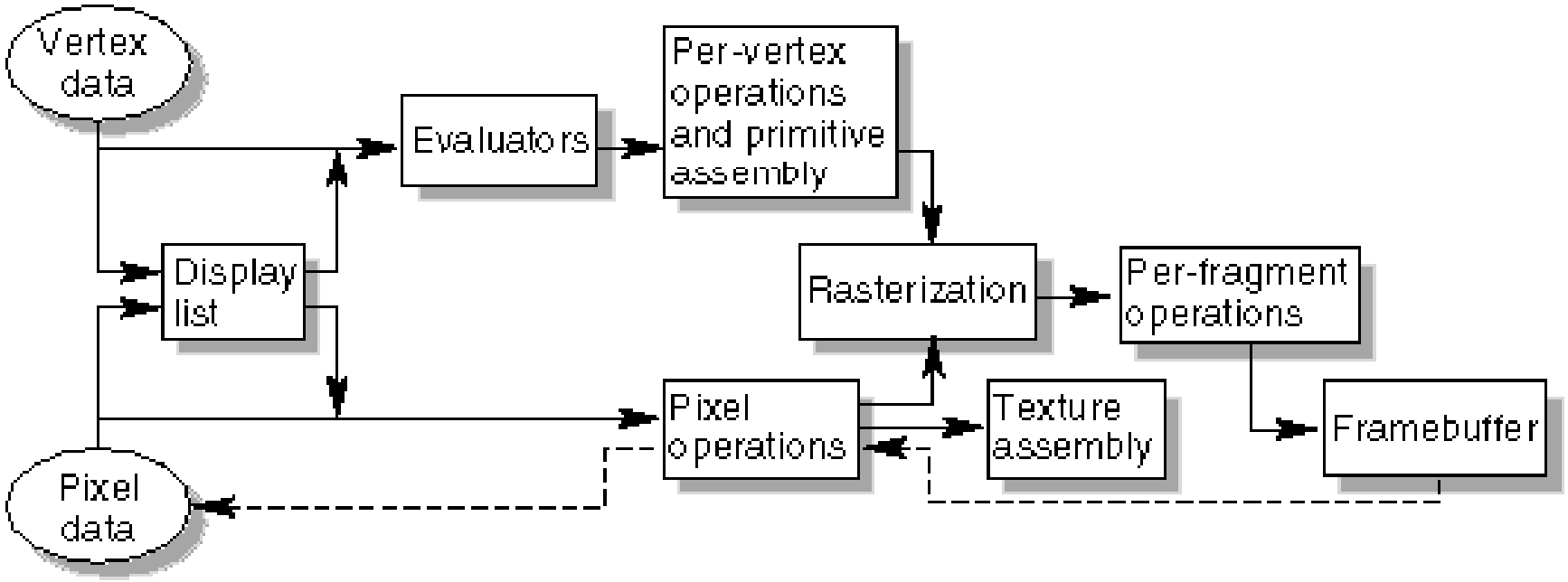


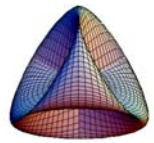
Another Pipeline View



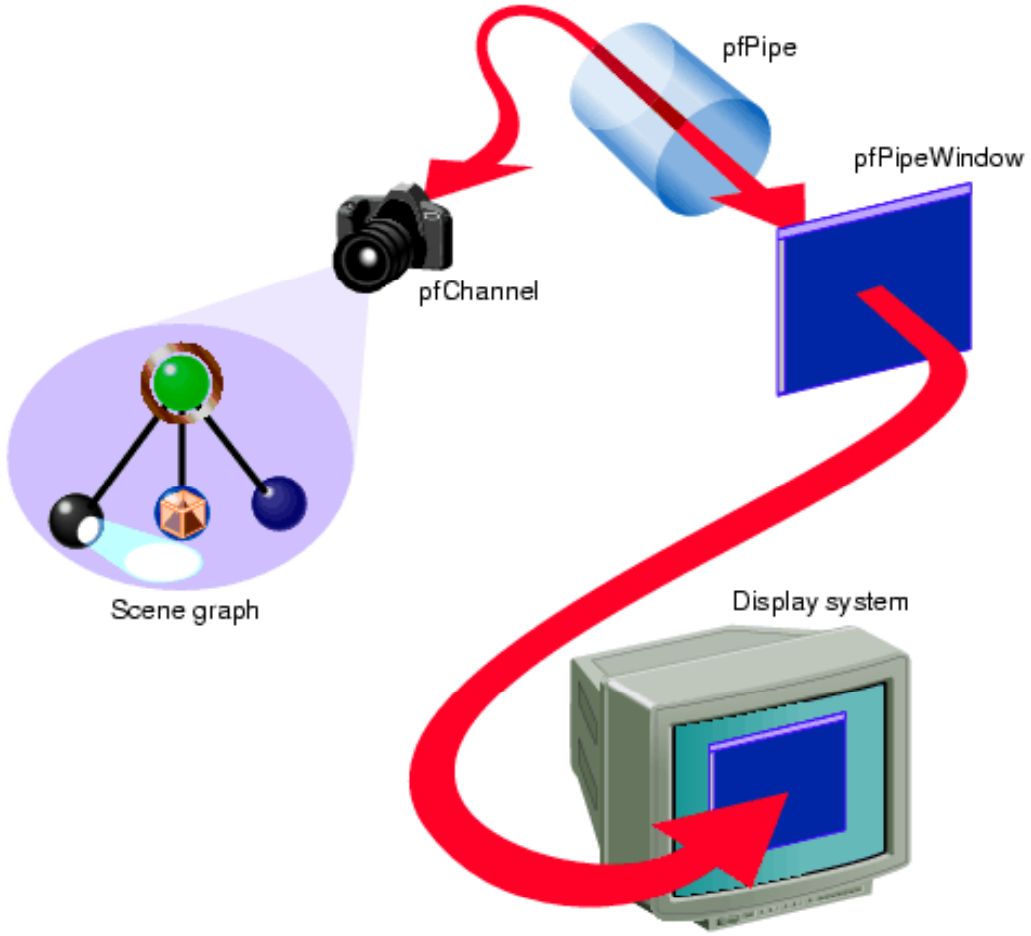


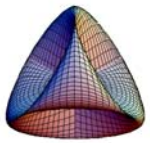
The Simplest View



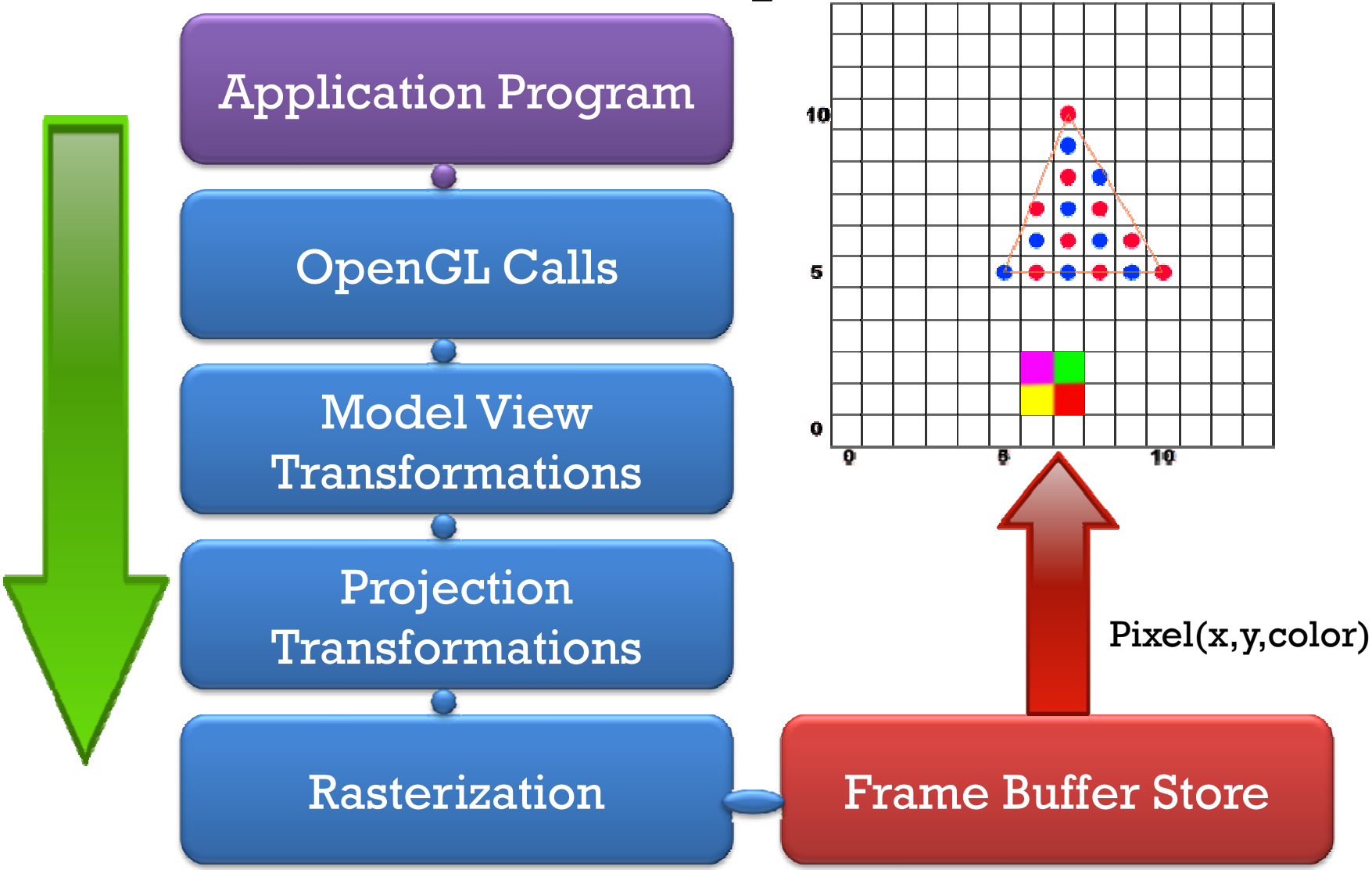


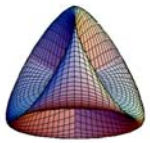
It looks a bit like this…





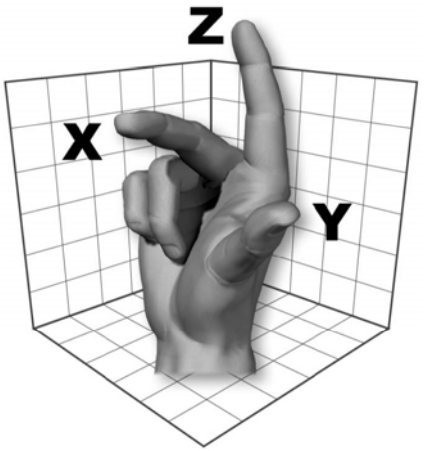
Geometry Phases





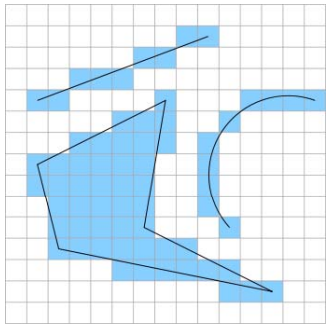
3D-2D Mapping

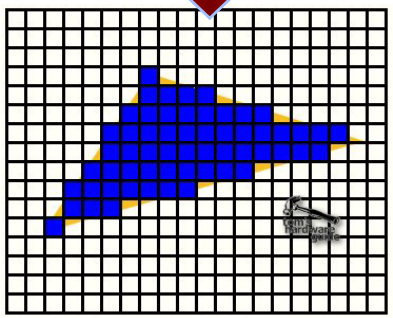
- 3D Modeling

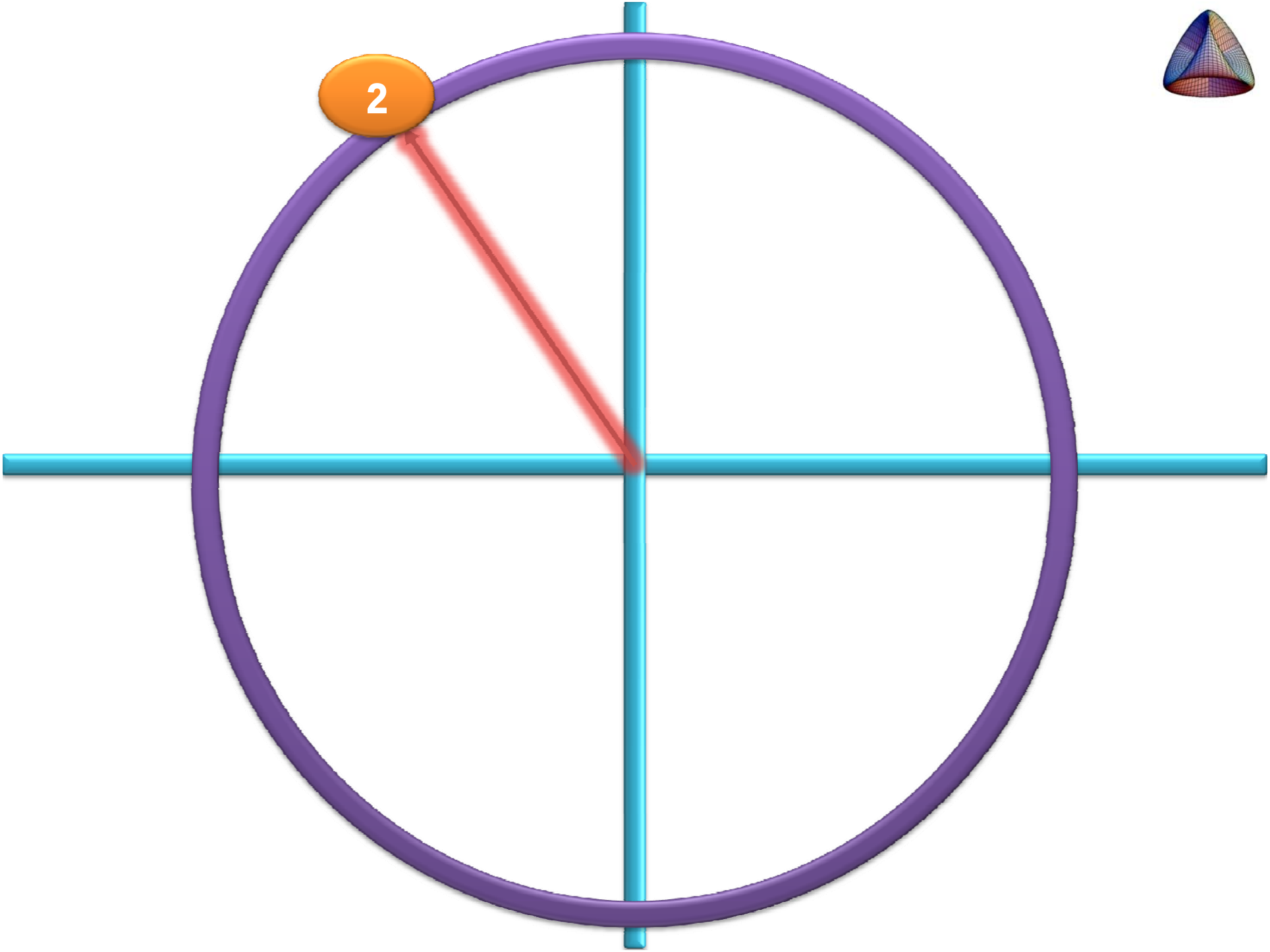


$T_1 \ldots T_n$ *Transformations*

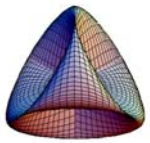
- Rasterization





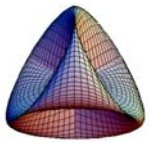

2

Raster Resolution



- PIXEL = PICture Element
- Resolution:

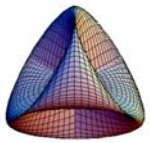
$$R = x_{pixels} \times y_{pixels}$$



Raster Scan

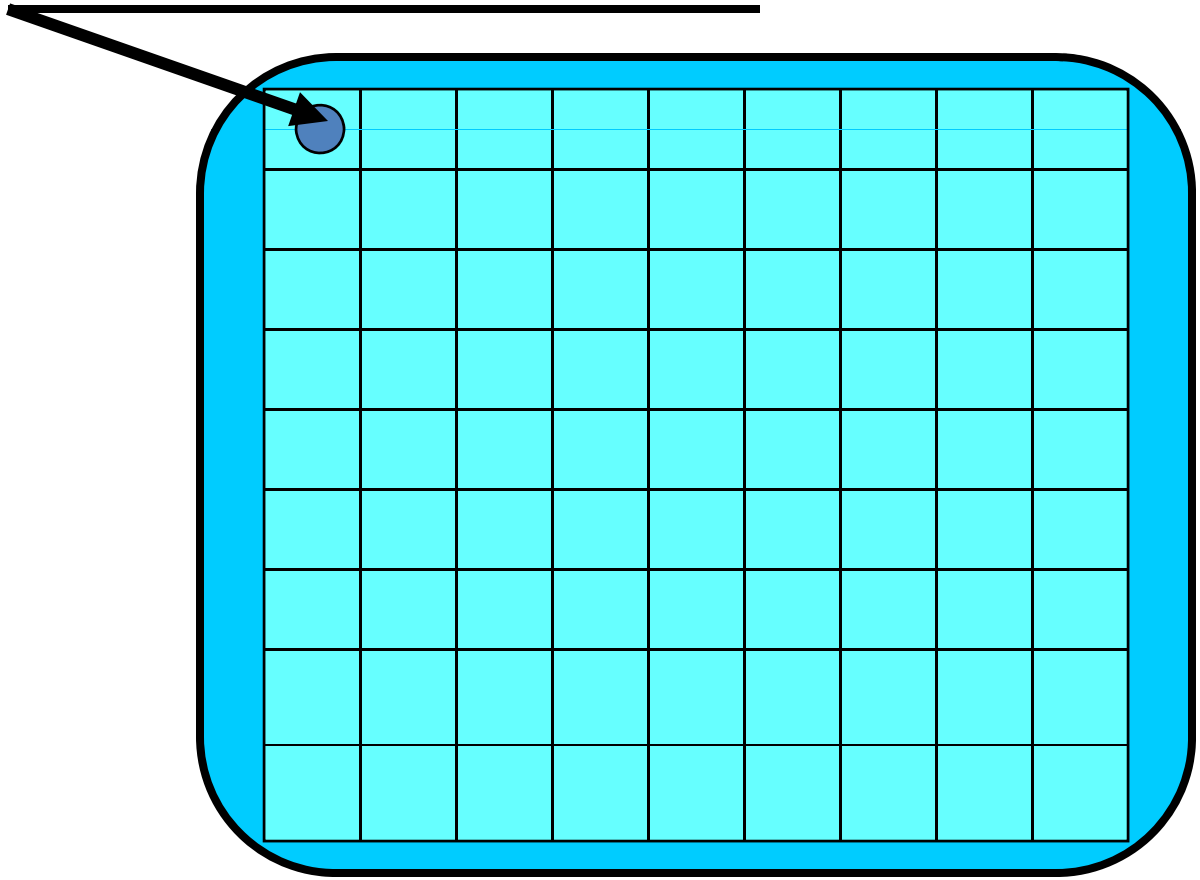
- Raster = Scanning Grid
- Electron beam scan pattern:

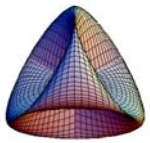
1. Left-to-Right: *a line scan*
2. Right-to-Left: *horizontal retrace*
3. Top-to-Bottom: *a frame*
4. Bottom-to-Top: *vertical retrace*



Raster Display

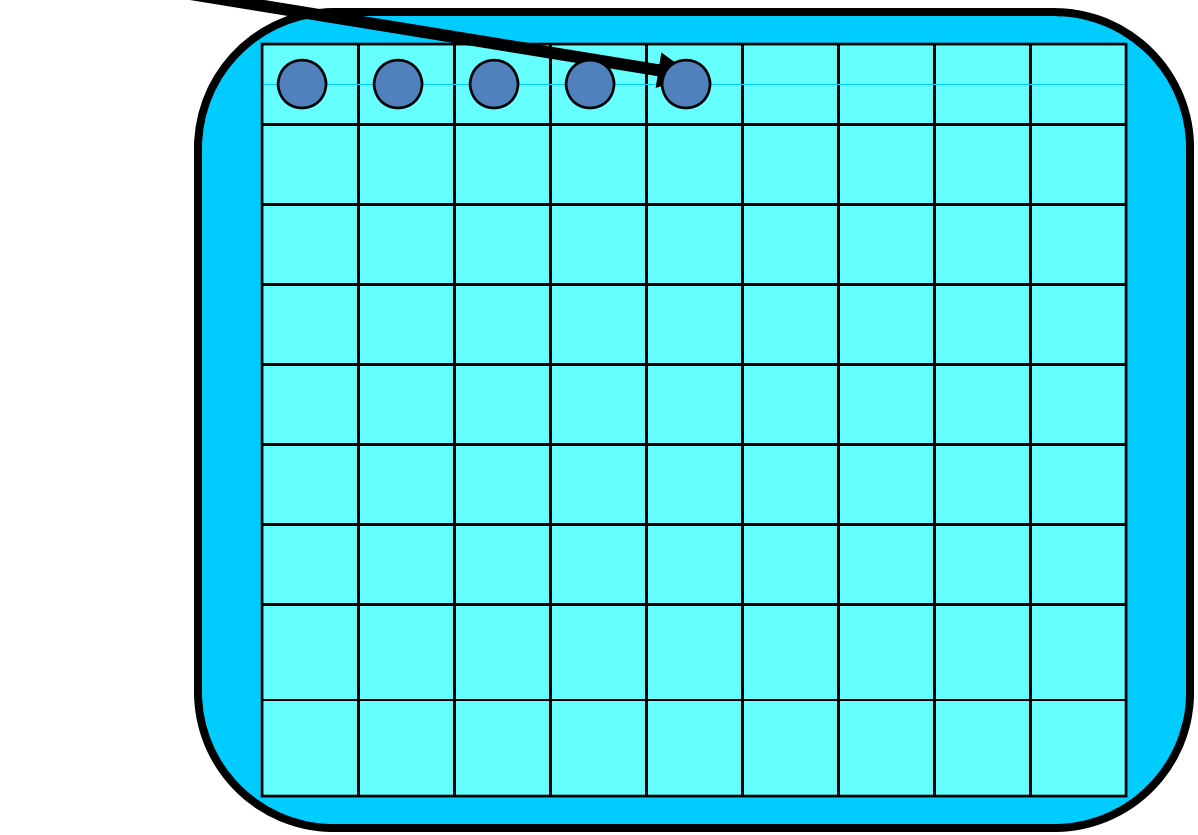
Beam starts in the top-left corner

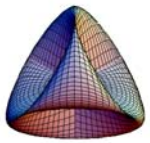




Raster Display

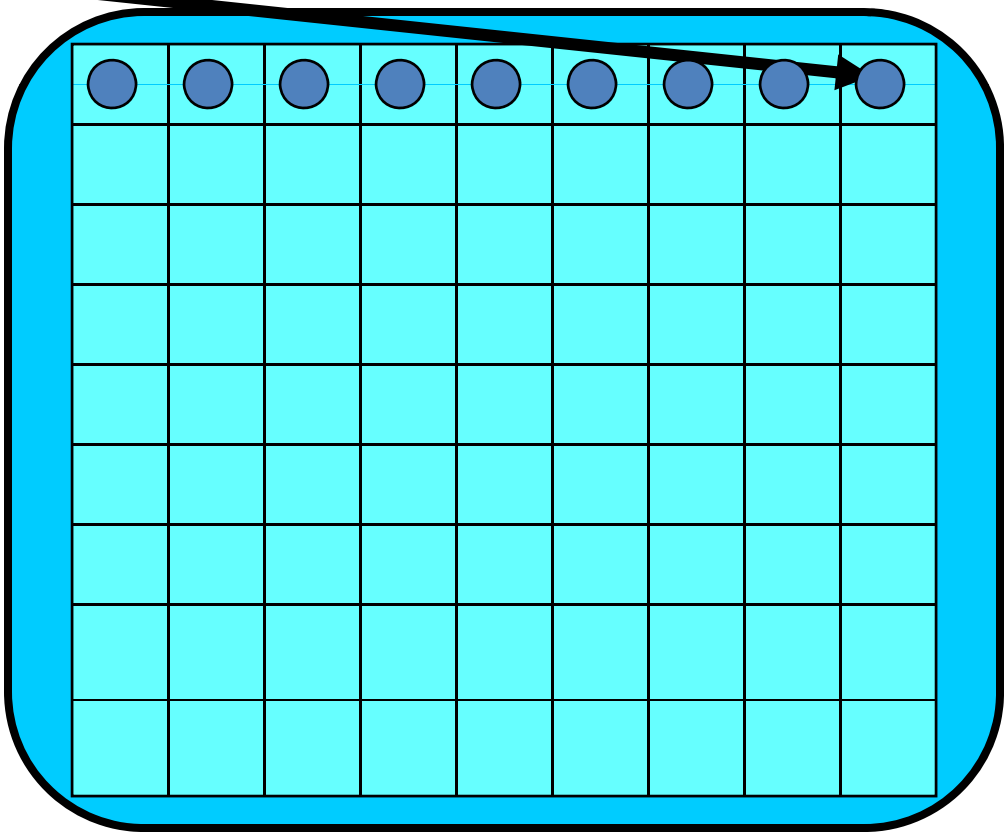
Horizontal scan from left-to-right

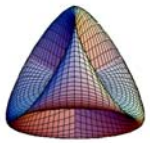




Raster Display

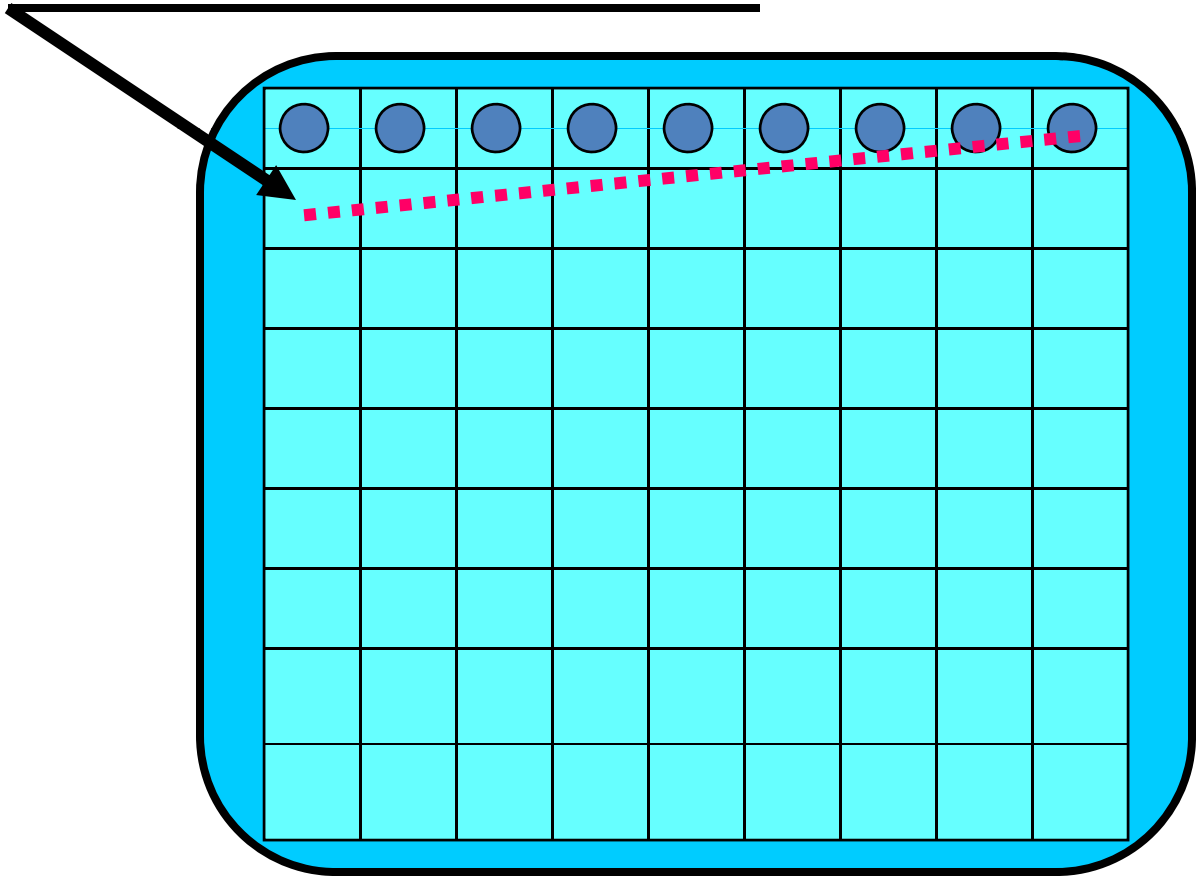
Horizontal scan from left-to-right

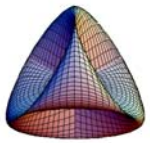




Raster Display

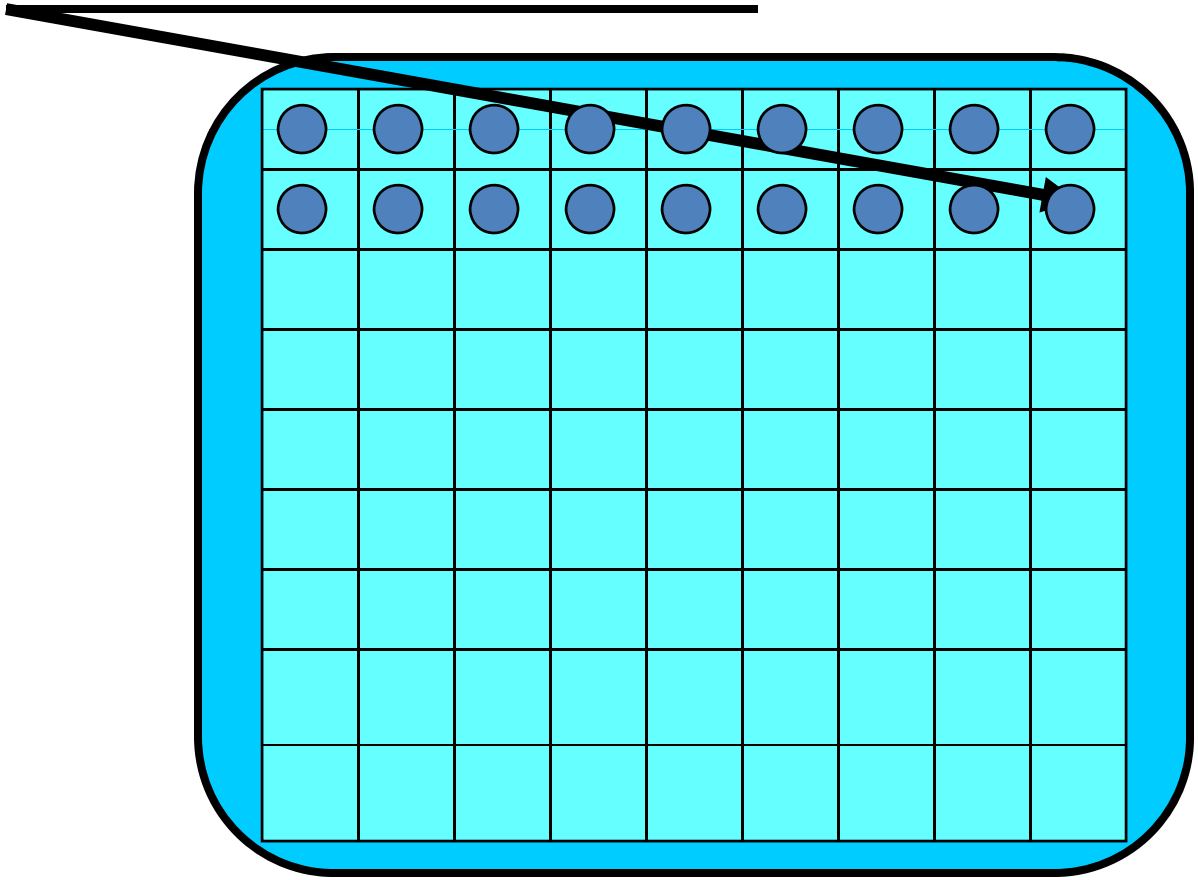
Horizontal retrace from right-to-left

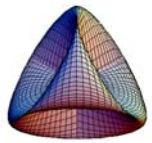




Raster Display

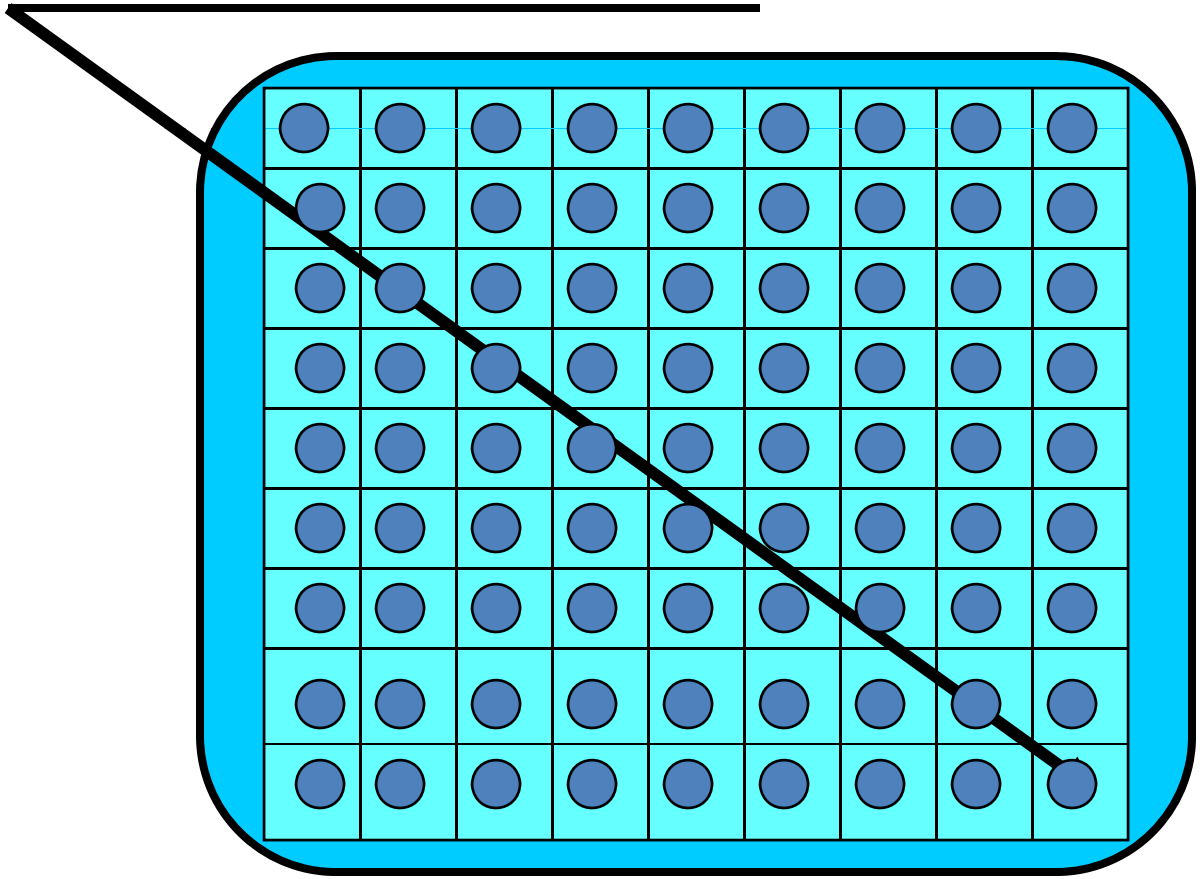
Horizontal scan from left-to-right

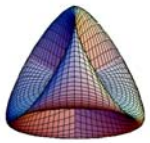




Raster Display

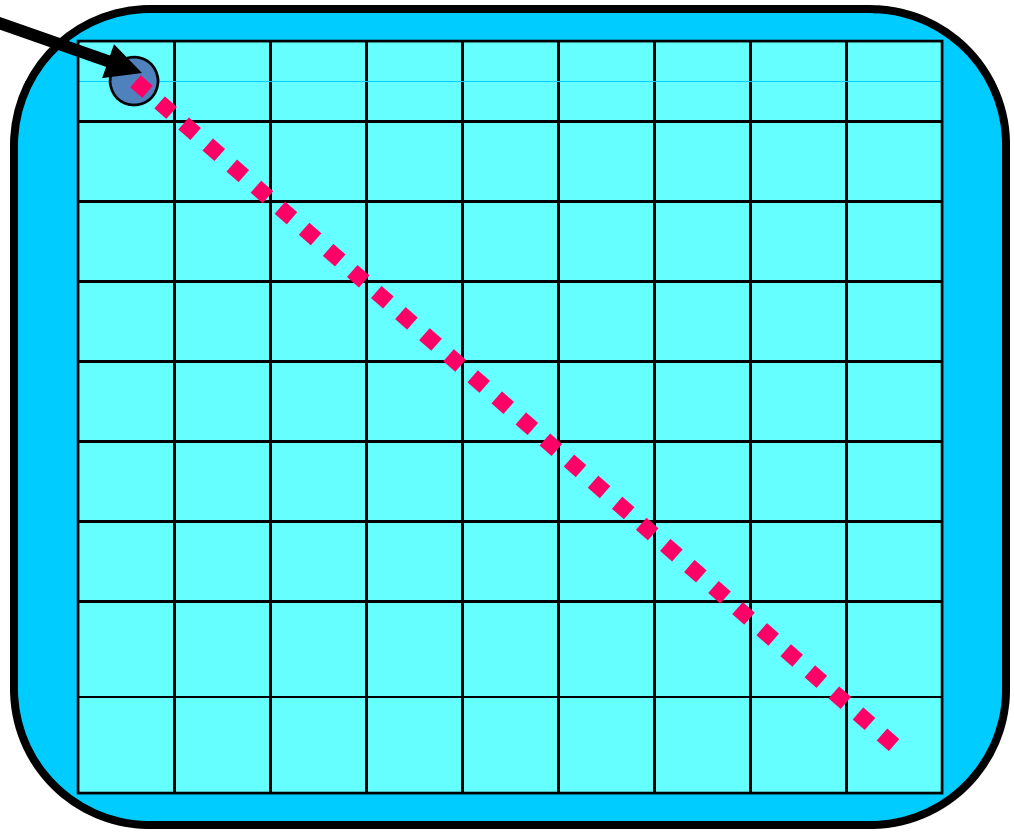
A Frame



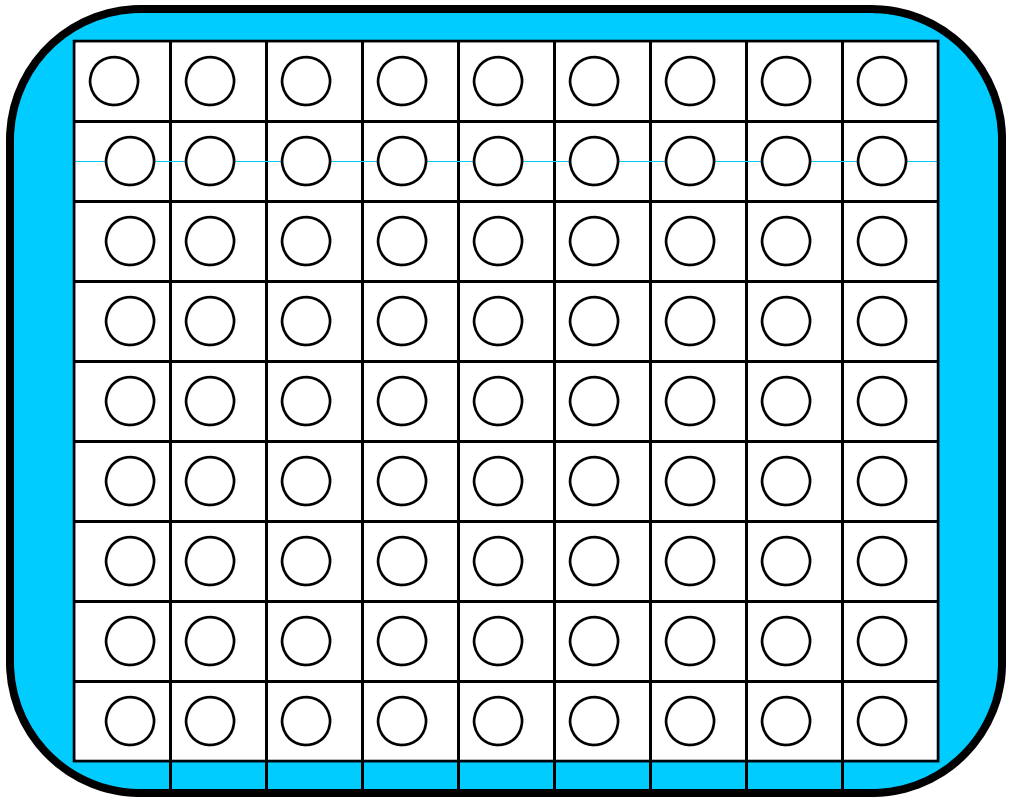


Raster Display

Vertical retrace from bottom-right



Noninterlaced Scan



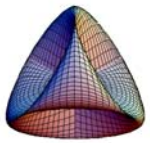
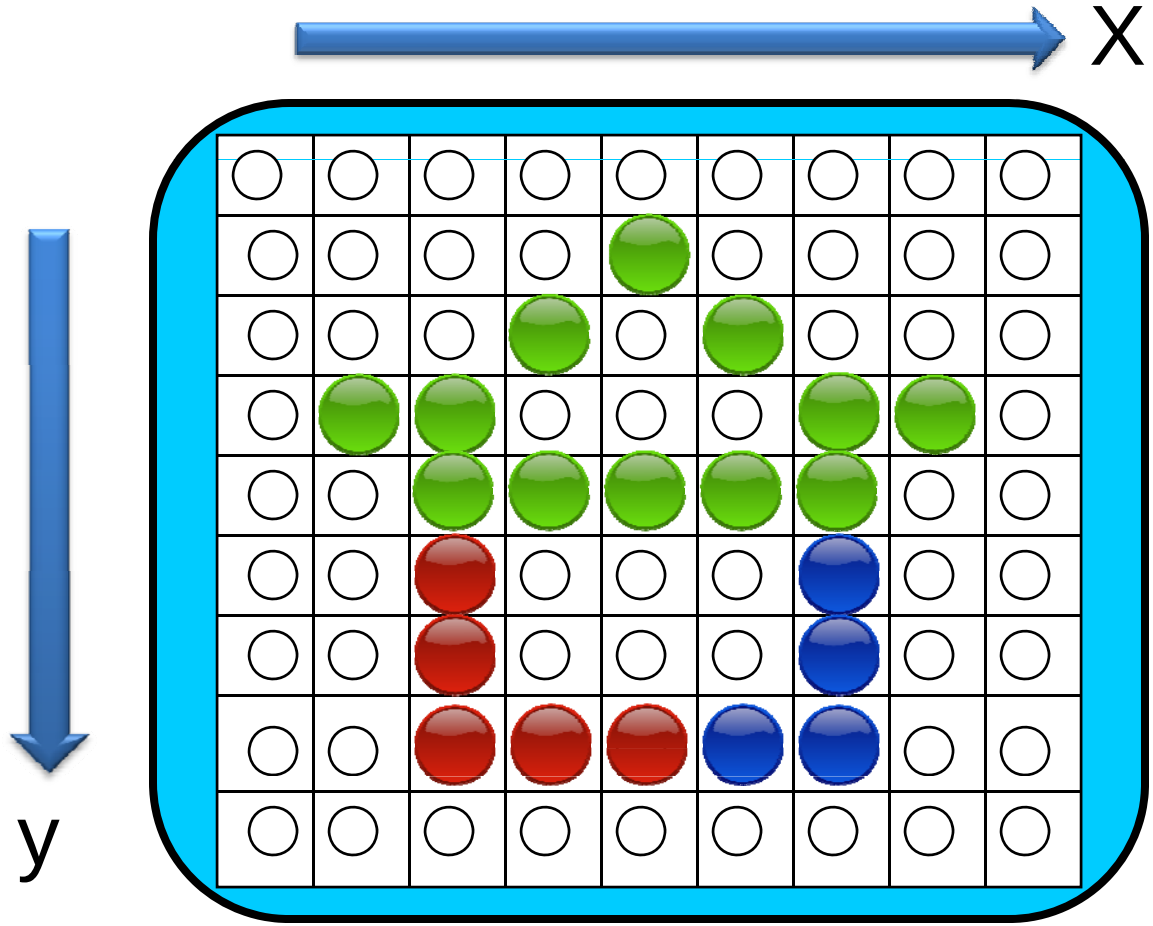
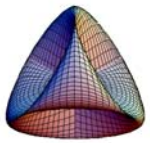


Image Formation

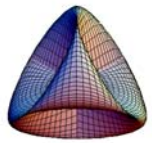
Some pixels are turned on
and some are turned off





Raster Size Parameters

- **Dimensions:** **X** by **Y**
  - Example: 343 by 274 mm
  - or: 13.5 by 10.79 in
  - or Diagonal: 17.282 in.
- **Aspect Ratio:** **Y : X** $\Rightarrow \frac{y}{x}$
  - Example: 274 / 343 = 0.799



Aspect Ratio

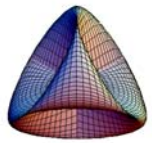
- Two specifications

1. Vertical to Horizontal (***)

$$y : x \Rightarrow \frac{y}{x} \quad \text{e.g.} \quad \frac{274}{343} = 0.799$$

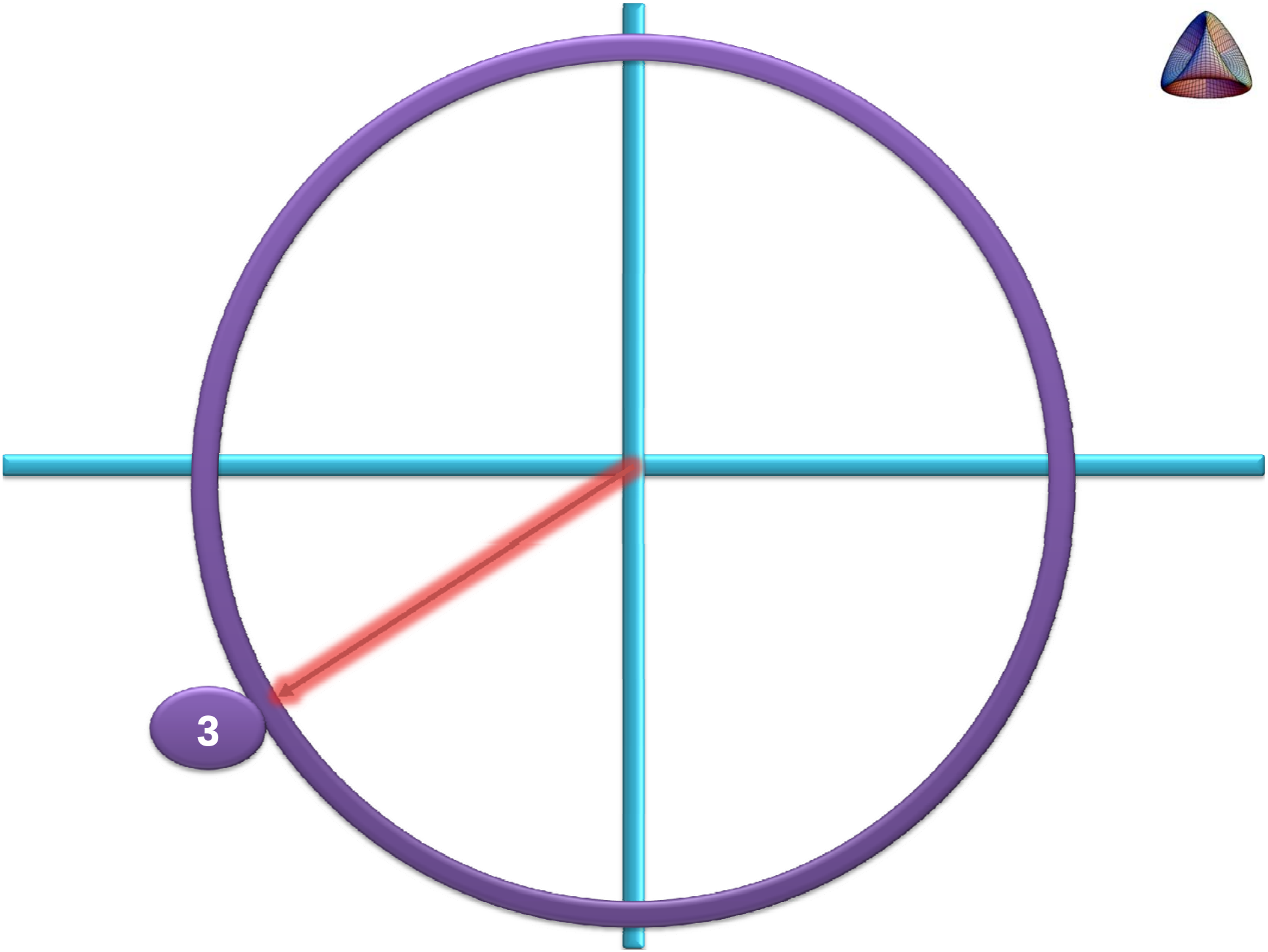
2. Horizontal to Vertical

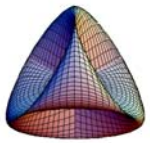
$$x : y \Rightarrow \frac{x}{y} \quad \text{e.g.} \quad \frac{343}{274} = 1.252$$



Picture Resolution

Resolution	X	Y
low	640	480
low+	720	512
medium	800	600
medium+	1024	768
standard	1280	1024
high	2560	2048

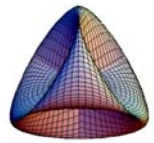

3



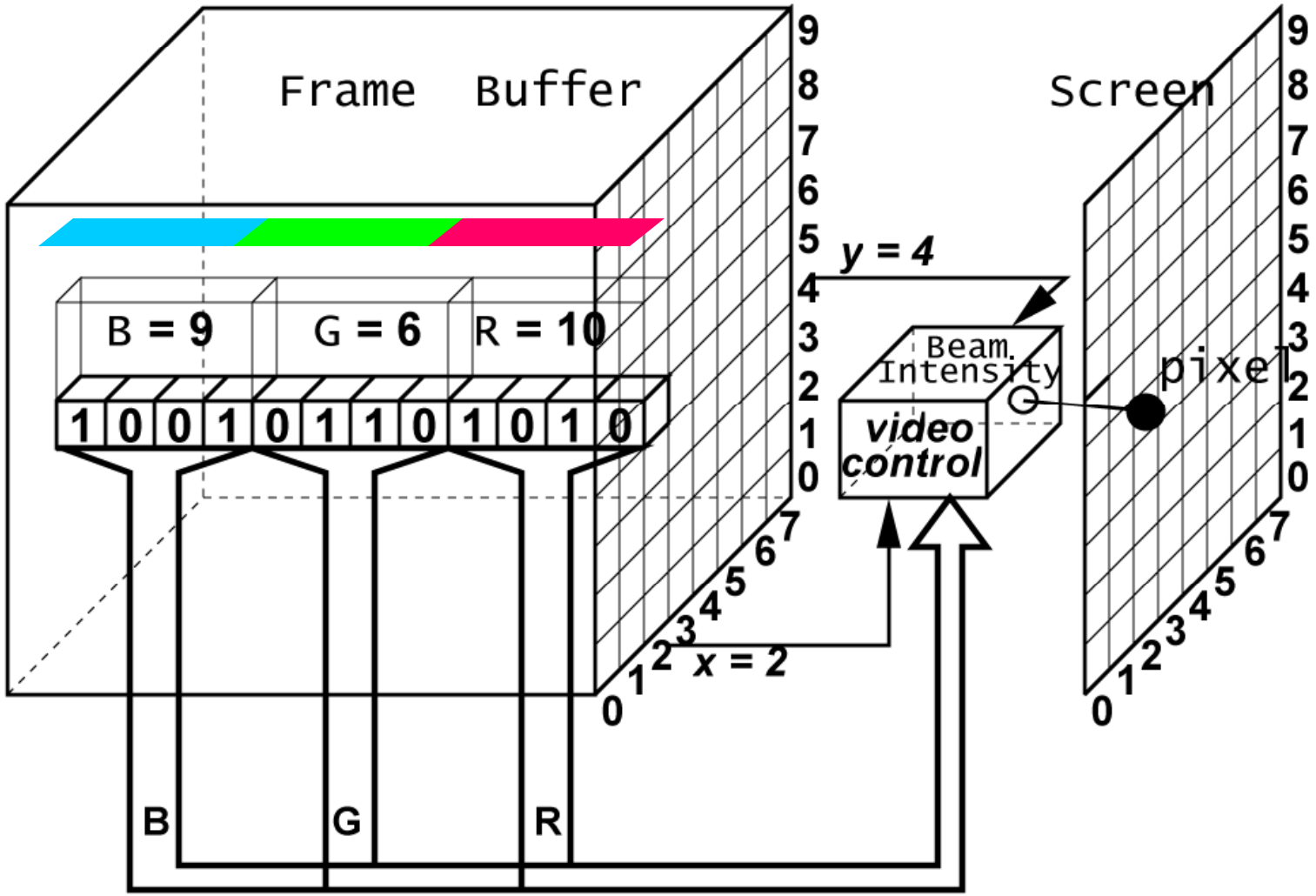
The Frame Buffer

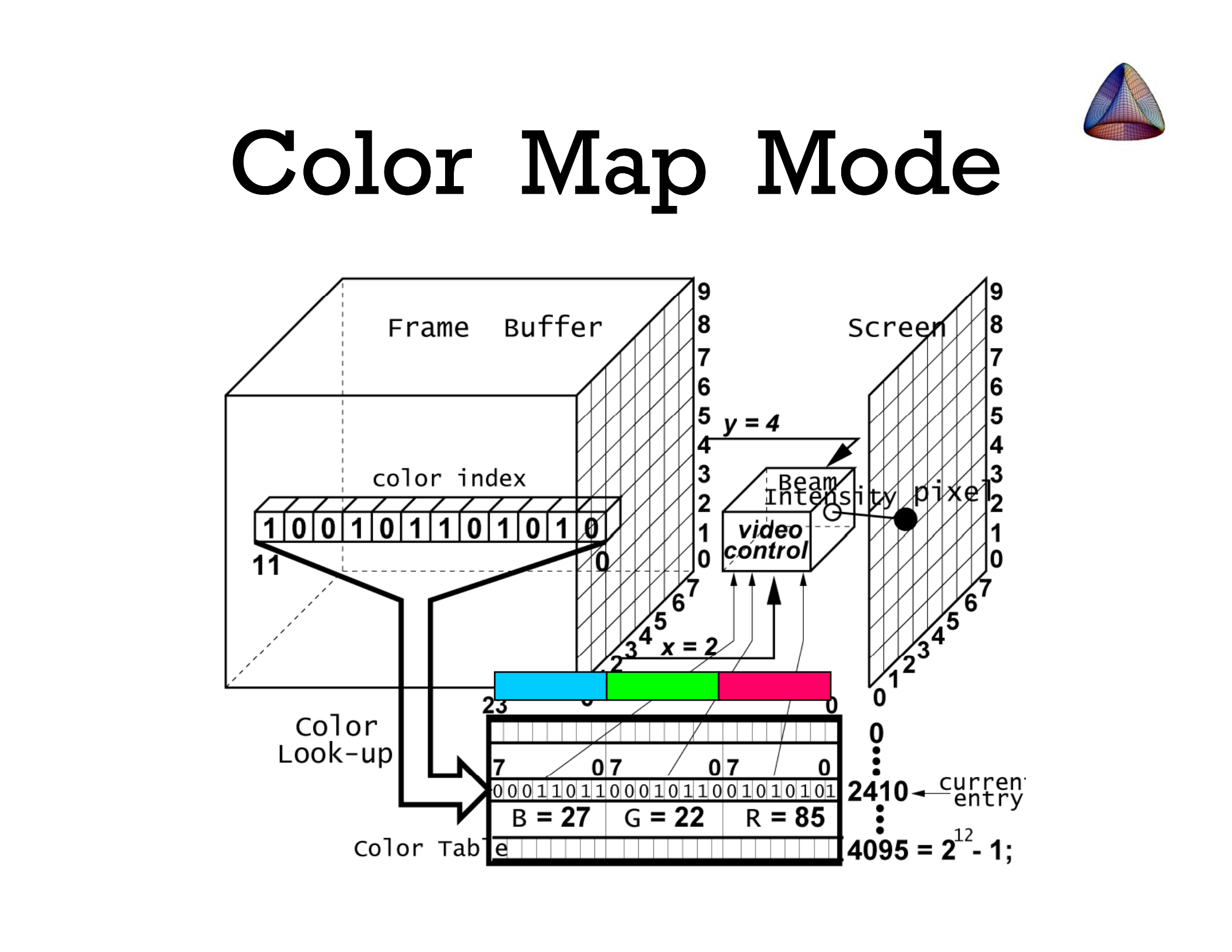
- One memory cell for each pixel
- Contains intensity or color data

- Models:
  1. RGB
  2. Color Map

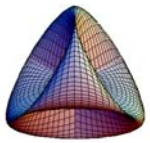


RGB Mode


Frame Buffer
B = 9
G = 6
R = 10
1 0 0 1 0 1 1 0 1 0 1 0
B
G
R
y = 4
x = 2
Beam Intensity
video control
Screen
pixel
0 1 2 3 4 5 6 7 8 9

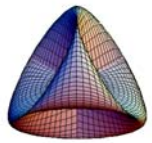


Color Map Mode



Color Map Mode

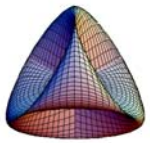
- A color Look-Up Table is used
- Pixel memory contains an index
- Index points to an entry in LUT
- LUT entries contain RGB values



Frame Rate

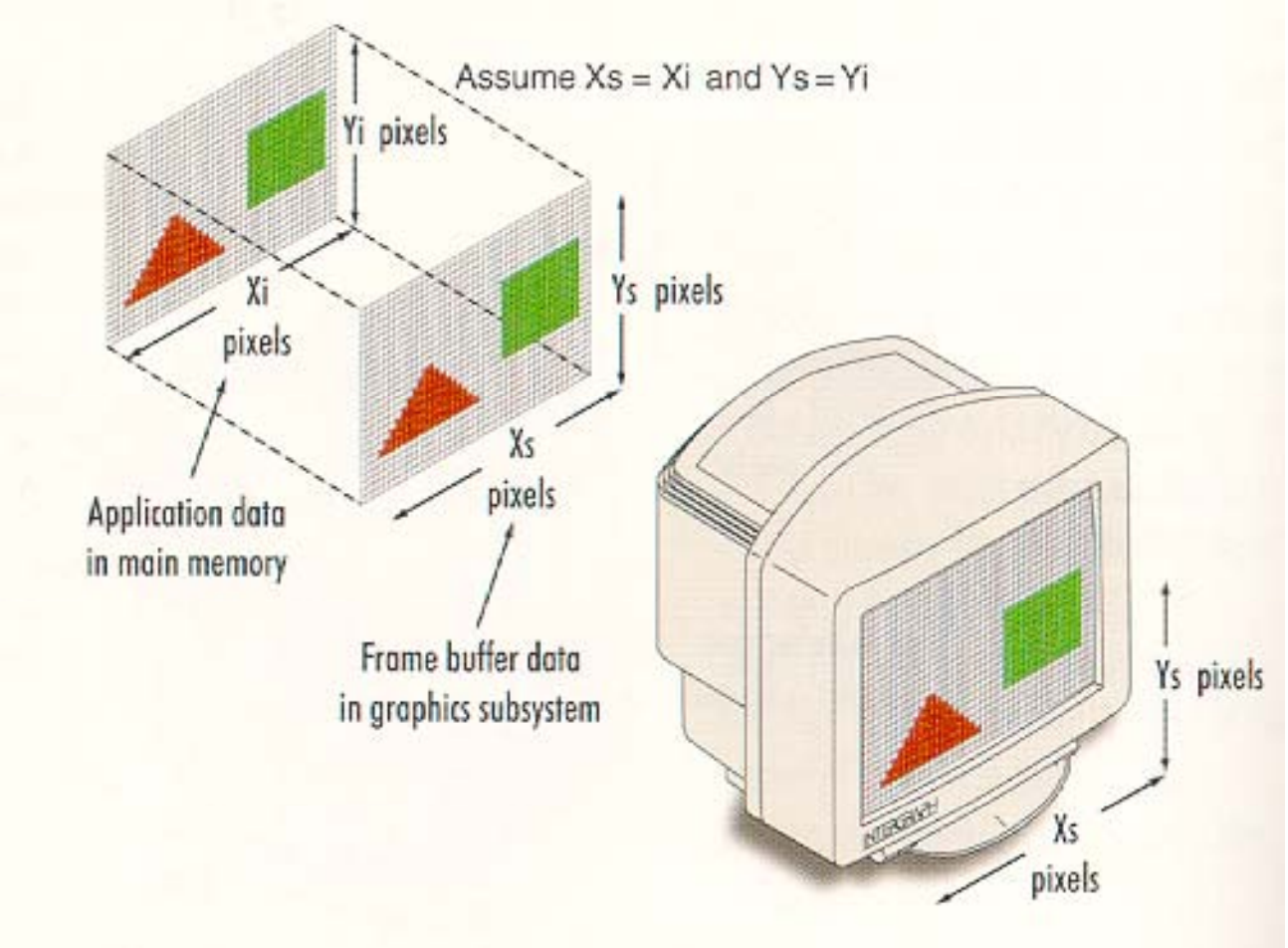
- Frame Rate = N frames / second
- Examples: 30Hz, 60Hz, 72Hz

Frame Rate

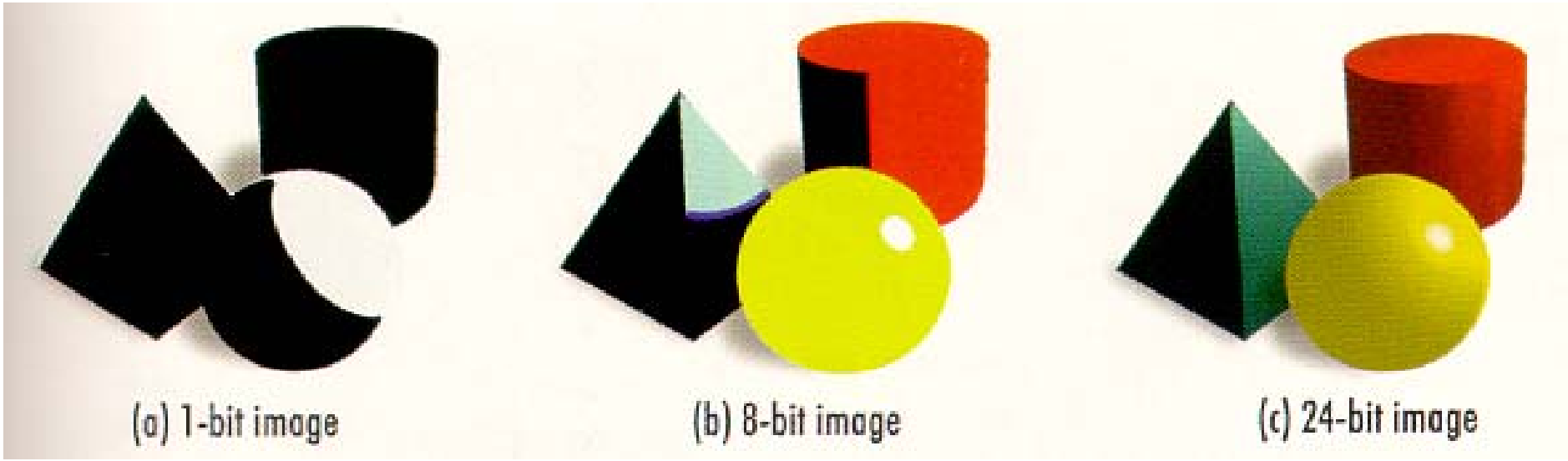


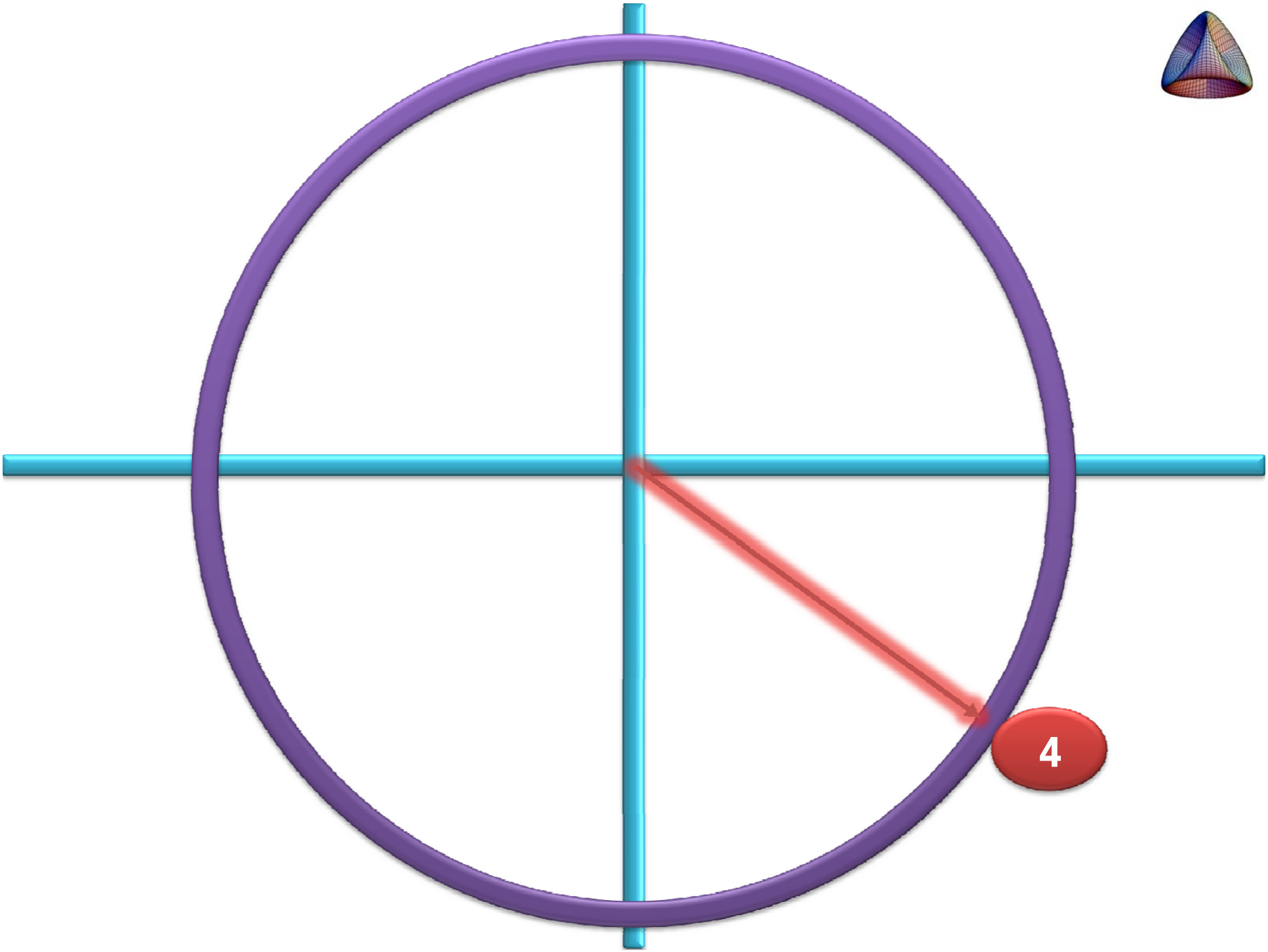
- Frame Rate = N frames / second
- Examples: 30Hz, 60Hz, 72Hz
- Noninterlaced: scan all lines in one pass

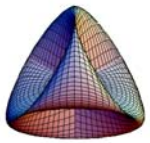
Drawing Objects



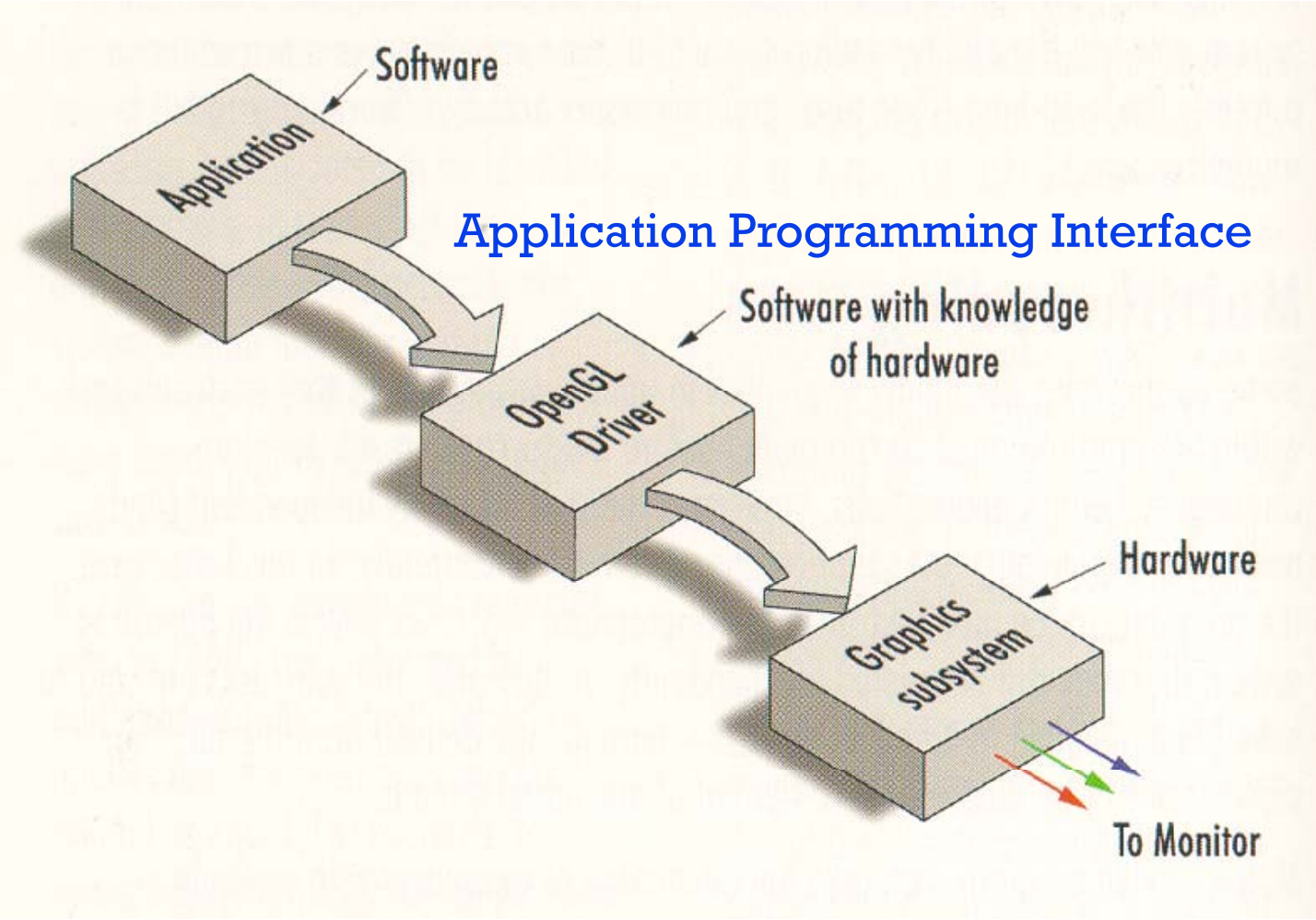
How many bits per pixel?

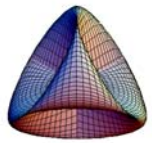



4



The API


Software
Application
Application Programming Interface
Software with knowledge
of hardware
OpenGL
Driver
Hardware
Graphics
subsystem
To Monitor



The Graphics Pipeline

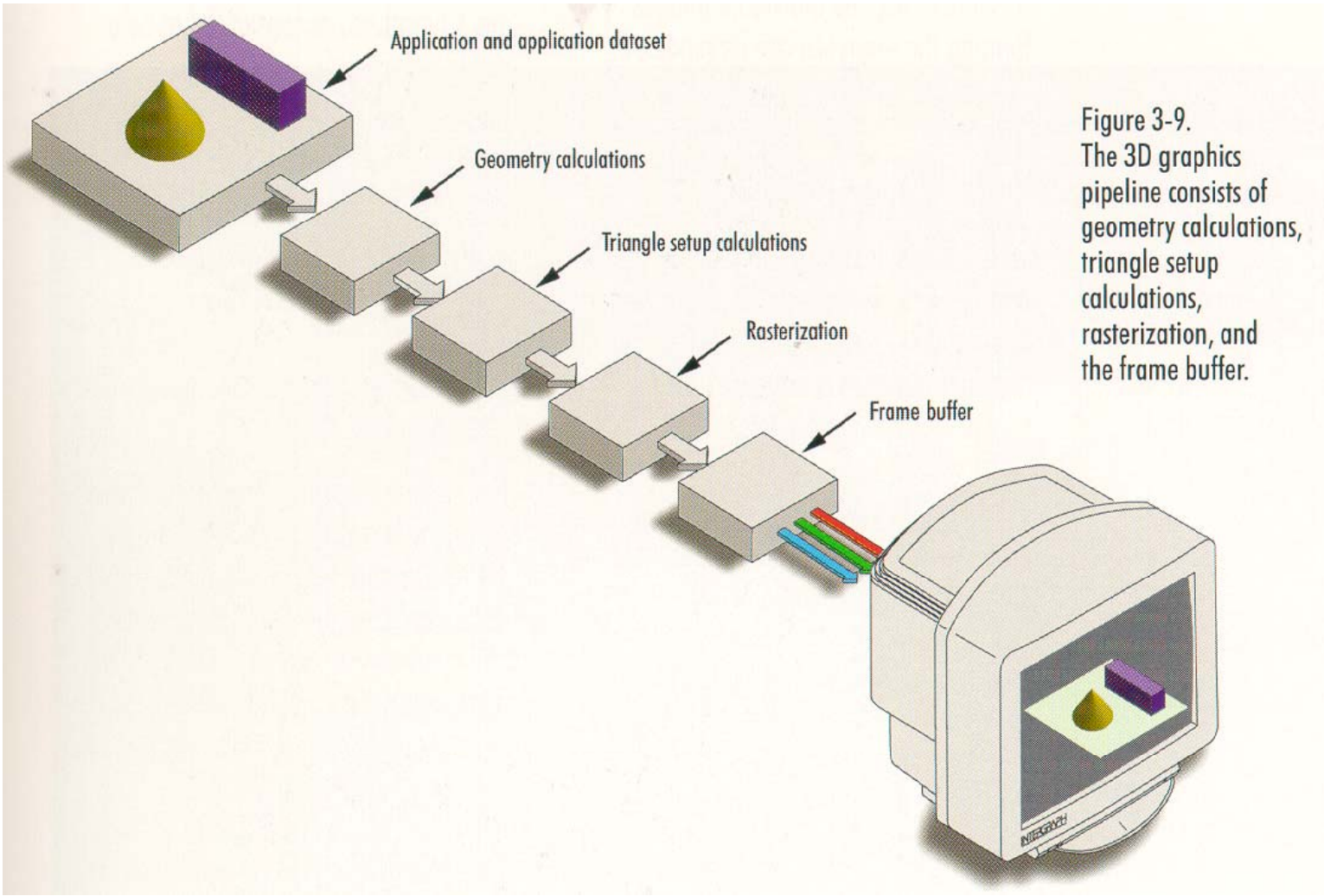
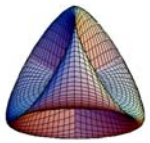


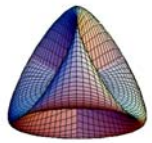
Figure 3-9.
The 3D graphics pipeline consists of geometry calculations, triangle setup calculations, rasterization, and the frame buffer.



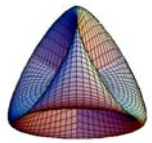
Input Devices

- Keyboard
- Mouse
- Joy Stick
- Trackball
- Dials and button boxes
- Spaceball
- Body gloves
- Motion capture systems

Input Device Interface

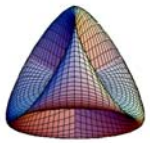


- Devices generate information:
  1. EVENTS
  2. DATA



Interactions

- Interactions via an ***event-handler***
- Input devices generate events
- Applications must handle them
- A response is generated:
  - a visual change
  - another action



Event Handling

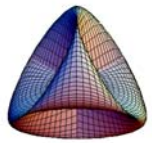
1. Polling: synchronous

   application is busy waiting until it receives an event.

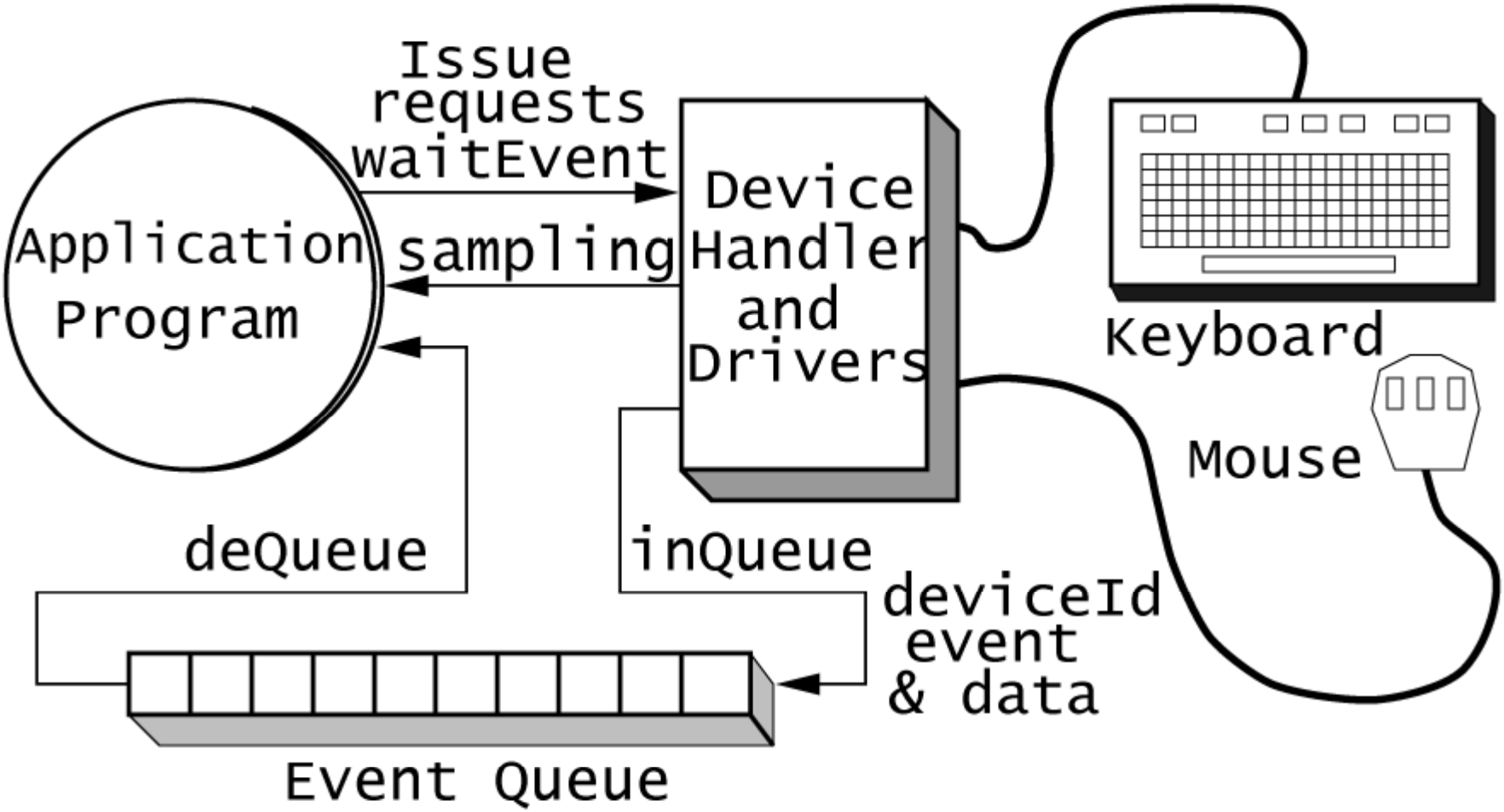
2. Event-Driven: asynchronous

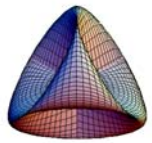
   events generate interrupts

   events are stored in a queue



Event Handler





The END

The END