# Introduction

## What is Computer Graphics?

* Computer Graphics - turning digital information into sensory stimuli
  + The use of computer to synthesize and manipulate sensory information
  + The use of computation to turn digital information into sensory stimuli
  + We can see this in:
    - Movies, games, cartoons, art and design, industrial design, computer aided engineering, architecture, scientific/mathematical visualization, Communication / mapping, Medical visualization, Navigation

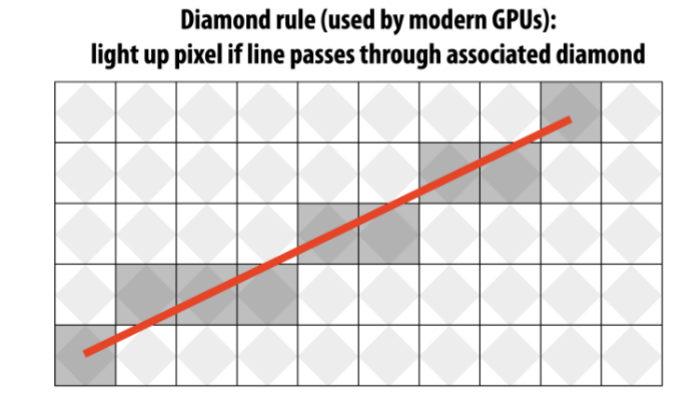
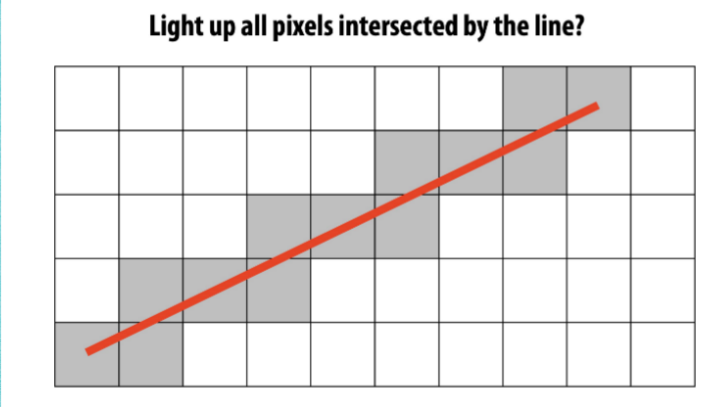
## Foundations in Computer Graphics

* All these applications demand sophisticated theory and systems
  + Theory:
    - Basic representations
    - Sampling and aliasing
    - Numerical methods
    - Radiometry and light transport
    - Perception
  + Systems:
    - Parallel, heterogeneous processing
    - Graphics-specific programming languages
* Perspective means objects look smaller as they get further away

## How do computers draw lines?

### Raster Display

* Displays that use the raster scan technique for assembling an electronic image on a screen by drawing a raster of horizontal lines. They are the most popular kind of graphics display.
* Rasterization - process of converting a continuous object to a discrete representation on a raster grid (pixel grid)



* For more complex pictures we need richer more complex models using:
  + Geometry, light, materials, cameras, motion and more.

### Class major focus

* Major areas of focus:
  + Imaging - how do computers store/generate images?
  + Geometry - how do we represent shape?
  + Rendering - How do we simulate light?
  + Animation - how do we synthesize motion?

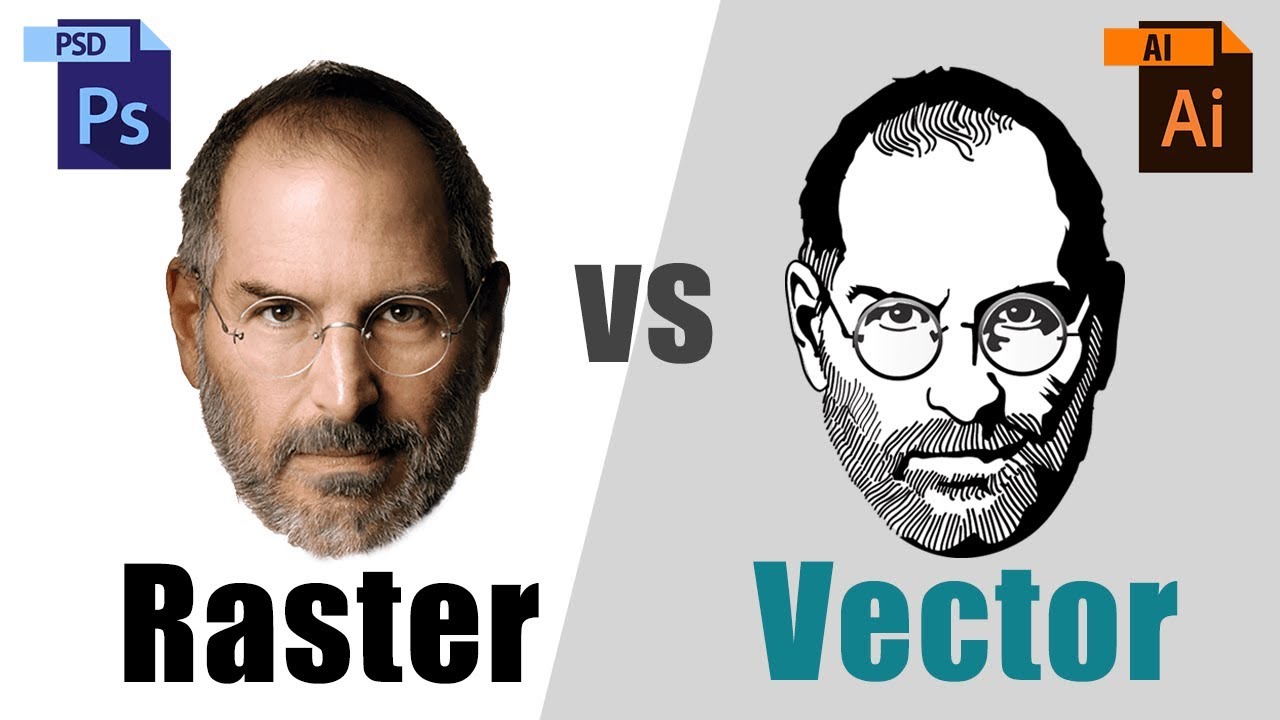
## Graphics APIs

* What is an API?
  + a collection of documented libraries and commands that communicate with the computer hardware in creating 2D and 3D applications
* What would be a graphics API?
  + Direct3D (a subset of DirectX)
  + Glide.
  + Mantle developed by AMD.
  + Metal developed by Apple.
  + OptiX 7.0 and Latest developed by NVIDIA.
  + OpenGL and the OpenGL Shading Language.
  + OpenGL ES 3D API for embedded devices.
* Two dominant paradigms
* Integrated approach – Java
* Graphics and user-interface toolkit – portable packages
* Second – Direct3D and OpenGL
* Drawing commands are part of a software library tied to a language

## Graphics Pipeline

* What is a graphic pipeline?
  + a conceptual model that describes what steps a graphics system needs to perform to render a 3D scene to a 2D screen
    - Every modern desktop has one
    - Specialized software / hardware subsystems
    - Efficiently draw 3D primitives in perspective
    - Shade the triangles
    - Look realistic and appear in proper back to front order
    - Geometric manipulation used in pipeline can be accomplished in 4D coordinate space
    - Manipulated using 4x4 matrices and 4-vectors
    - Therefore graphics pipeline contains much machinery for efficient processing and composing those matrices and vectors

## Raster vs Vector

* Raster files are also commonly used for editing images, photos, and graphics. Vector files work better for digital illustrations, complex graphics, and logos.
* Raster
  + Also differ in field of file formats
    - GIF PNG JPEG – raster formats
  + Data size larger
    - GIF PNG – lossless data compression
    - JPEG – lossy data compression
* Vector
  + SVG – vector format (although can include raster images)
  + A XML-based language – describes 2D vector graphics
  + Scalable Vector Graphics
  + Scalable is an advantage of vector graphicsM

## 3D Graphics Basics

* Most common – vector graphics
* Geometric modeling
* A collection of geometric shapes – arranged in 3D space
* Objects have attributes

## Geometry

* Start with empty 3D world
* Space exists conceptually – good to visualize 3D though
* Coordinate system – each point in space – 3 numbers x, y and z
* World coordinates
* Build a scene – with geometric objects
  + Smallest building blocks - geometric primitives
  + May have different primitives based on system
  + A complex scene contains large number of primitives
  + Programmer should chunk primitives into reusable components
* Reusable objects – in complex models – called hierarchical modeling
* Think of a wheel – what primitives can we use to create?
  + Once wheel is created to use it in other scene we may need to adjust rotation, scale or translation
  + Called geometric transformation – size, orientation and position
  + Scaling – size of object
  + Rotation – orientation of object
  + Translation – position of object

## Appearance

* Attributes – color – graphics – called material
* Properties that determine the intrinsic visual appearance of a surface
* Also – think how does that material interact with light or how do you want it to interact with light
* Material properties include:
  + color, shininess, roughness and transparency
* Useful material property – texture
  + Varying material properties from point-to-point on a surface
  + Surface may be painted or change values like transparency or bumpiness
  + Allow detail in a scene without extra primitives
* Material is intrinsic property – actual appearance depends on the environment – where object is viewed
  + Need light to see anything
  + Each light can have own properties (color, intensity and direction)

## Image

* Ultimate goal of 3D is to produce 2D images of 3D world
  + Transformation from 3D to 2D – viewing and projection
* World looks different from different points of view
  + Need to setup a viewer's – point of view
  + We will specify direction and position of our viewer - cameras
* Assign colors to individual pixels in 2D image
* Rasterization – and whole process of producing an image – called rendering the scene
* Ultimate goal to create single image – but also animation (sequential movement of images)