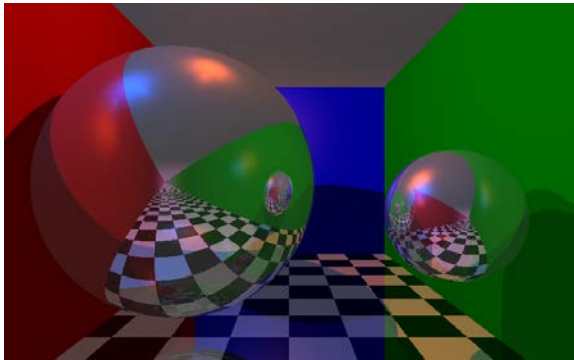
CSI 4105 Computer Graphics

Spring 2017

# Introduction to CSI 4105

Seon Joo Kim
Yonsei University



Slide credits: Michael S. Brown, Ron Fedkiw

# Lecturer

- Dr. Seon Joo Kim
  - Associate Professor, Department of Computer Science
  - Computational Intelligence and Photography Lab
- Office Location
  - Engineering Bldg #1, 311 (soon to 723 Eng. Bldg 4)
- Office Hours
  - Please arrange by email: seonjookim@yonsei.ac.kr

# Our TAs

Jae Yeon Kang
rkdjwodus@gmail.com

+

Many members of our lab.

# Module Objectives

- **This course is intended as a broad introduction to Computer Graphics**
- Focus is on real-time interactive 3D graphics
- Applications based on “OpenGL API” in C/C++
- **Outcomes**
  - **After this course, students should understand core computer graphics terminology and concepts. You should also be able to create 2D/3D graphics programs using the OpenGL API.**

# Module Objectives

- At the end of the semester, I expect the students to be able to create this

http://www.youtube.com/watch?feature=player_detailpage&v=moSFlvxnbgk#t=60

or this

http://www.youtube.com/watch?v=9the6PLLuVc#t=1

# Module Objectives

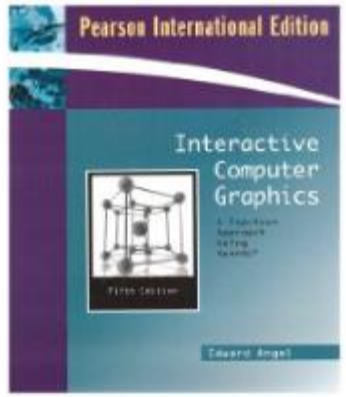
- Some sample projects from last year

# Why OpenGL?

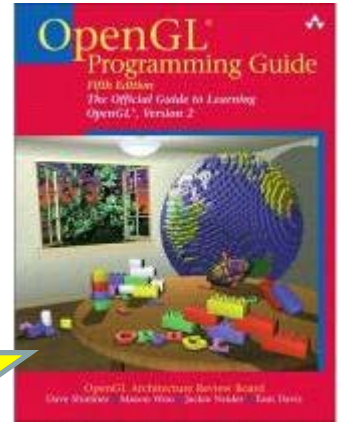

- Why OpenGL and not “DirectX”?
  1. OpenGL is still considered the industry standard for high-performance computer graphics
  2. It is adopted by more platforms (Apple, Unix, Windows, etc. . ) than DirectX
  3. It is also easier to learn than DirectX, so it more suitable for a introduction to Graphics course
  4. However, if you learn OpenGL first, much of what you learn can be applied to DirectX

# References

- Main Text (not required, but recommended)
  - Interactive Computer Graphics: A top-down approach using OpenGL, *3rd Edition*, by Edward Angel




- Other Books
  - The OpenGL Programming Guide (The Redbook), Addison-Wesley
    - 2nd Edition is freely available online at http://www.opengl.org/documentation/red_book/




- Web Resources
  - See Weblinks listed in module IVLE

Very useful book!

I'll be following Angel's book closely, but unfortunately there aren't any good CG textbooks. I'll try to make the notes as complete as possible. I'll leave it up to you whether you want to purchase the book or not.

# Module Topics

- Preliminary Math for Graphics
  - Points, Vectors, Planes, and more
  - Trigonometry, Coordinate Transforms
- Graphics Systems and Models
- Graphics Programming (OpenGL)
- Input and Interaction
  - Mouse, Keyboard, input events
- Geometric Modeling and Transformations
- Viewing
  - 3D viewing and projective transformations
- Lighting and Shading
  - Illumination Models
  - Rendering
- Vertices and Fragments (tentative)
- Discrete Methods
  - Bitmap manipulation
  - Texture maps
- Programmable Shaders (tentative)
- Hierarchical Modeling (tentative)
  - Scene Graphics
- Curve and Surfaces
- Particle Systems
- Ray Tracing
- Image Processing / Computational Photography

# Module Prerequisites

- Good programming skills in C, C++ (or Java)
- Basic Data Structures
  - E.g. arrays, linked lists, trees
- Basic Vector Operations
  - E.g. dot product, cross product, vector addition

    (I'll review this, but helps if you already know it)
- Simple Linear Algebra
  - E.g. matrix multiplication, matrix transpose
- Basic Trigonometry
  - (sine, cosine, tan, arctan, etc)

# Tentative Module Assessment

- **Midterm Exam: 25%**
- **Final Exam: 25%**
- **Four/Five Programming Assignments: 50%**

  *** 5 minute 1-to-1 grading
- **Participation: +3%**
- **One absence: -2%, (-4% for two-hour class)**

**Plagiarism of assignments** is a serious offense, please don't do it. Take this as an opportunity to learn programming + something fun. Graphics is a great module on so many levels if you take it serious.

# Project #0

- YSCEC
- Install OpenGL and learn to run it

# How good is CG these days?

# How good is CG these days?

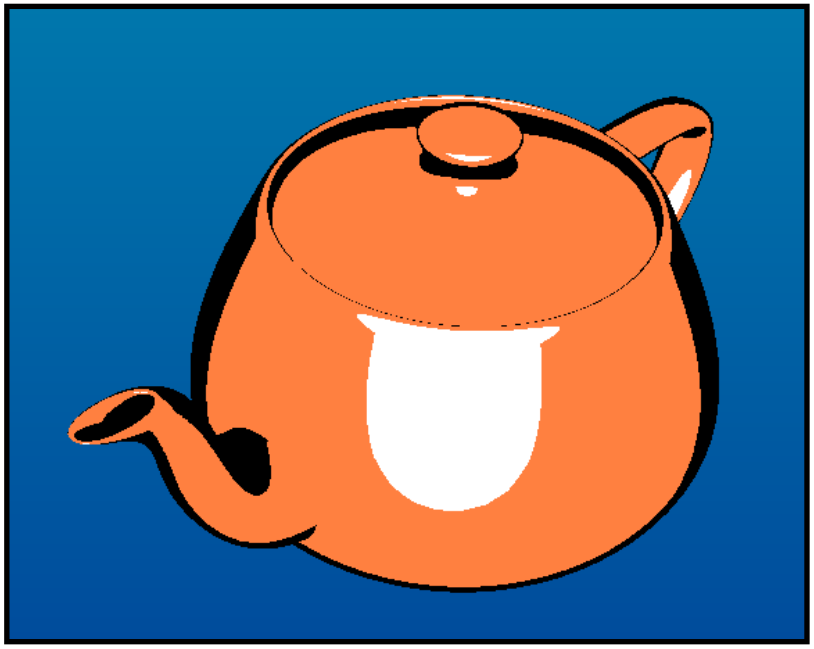
# How good is CG these days?

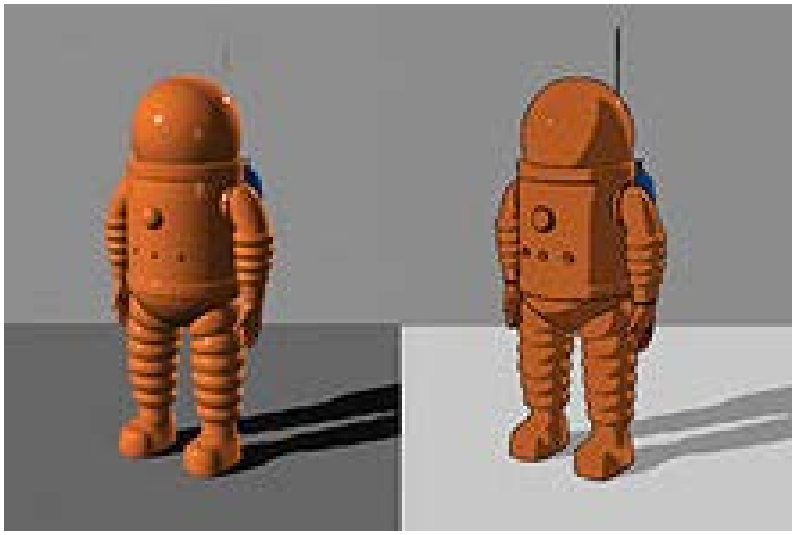
# How good is CG these days?

# That was with ray-tracing, how about real-time (interactive)?




# Sometimes we don’t want to be realistic. . .





We call this type of rendering – “Non-Photo Realistic” or NPR

# What can I do with graphics?

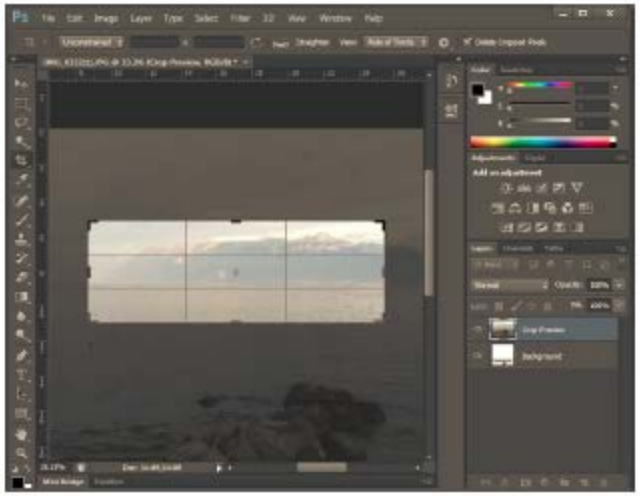

# Video Games


Spore

Crysis

Braid

# 2D Image Processing



# Motion Capture

**Motion capture of Olympic swimmer Dana Vollmer by Manhattan Mocap**

**Facial capture in Avatar**

# Visual Effects: Gases

Harry Potter and the Order of the Phoenix

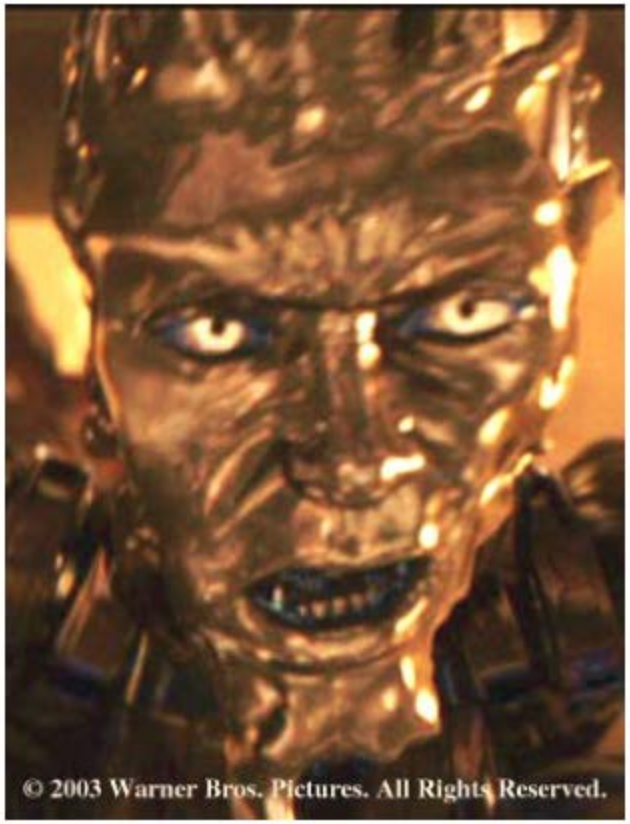
Terminator 3

Star Wars Episode III

# Visual Effects: Liquids

Battleship

The Day After Tomorrow




Terminator 2

# Visual Effects: Solids

- Destruction: fracture, explosions, etc.

**Super 8**

**2012**

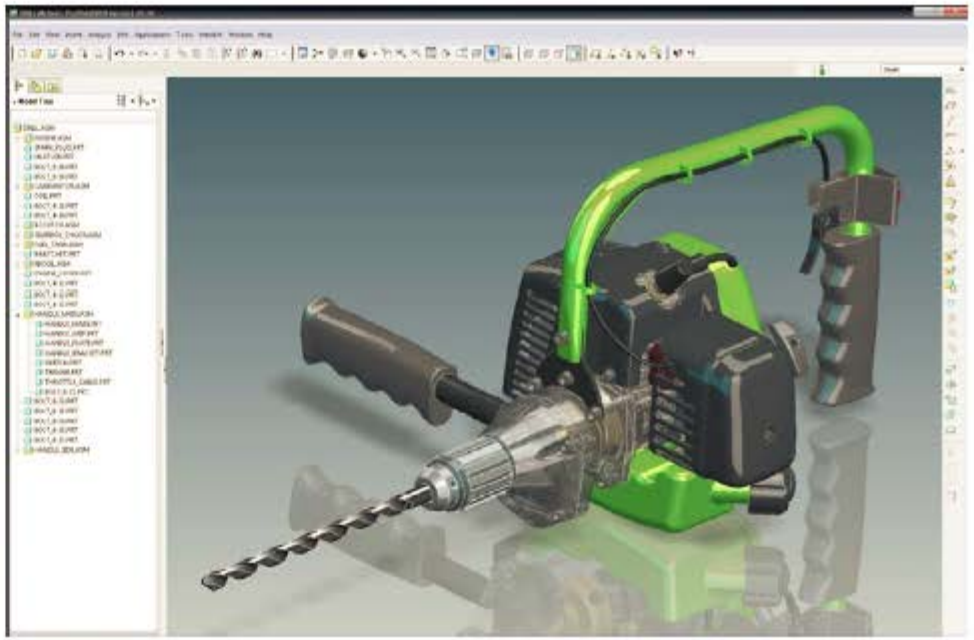
# Visual Effects: CG Creatures

Yoda, Star Wars Episode II

Sméagol/Gollum, The Lord of the Rings

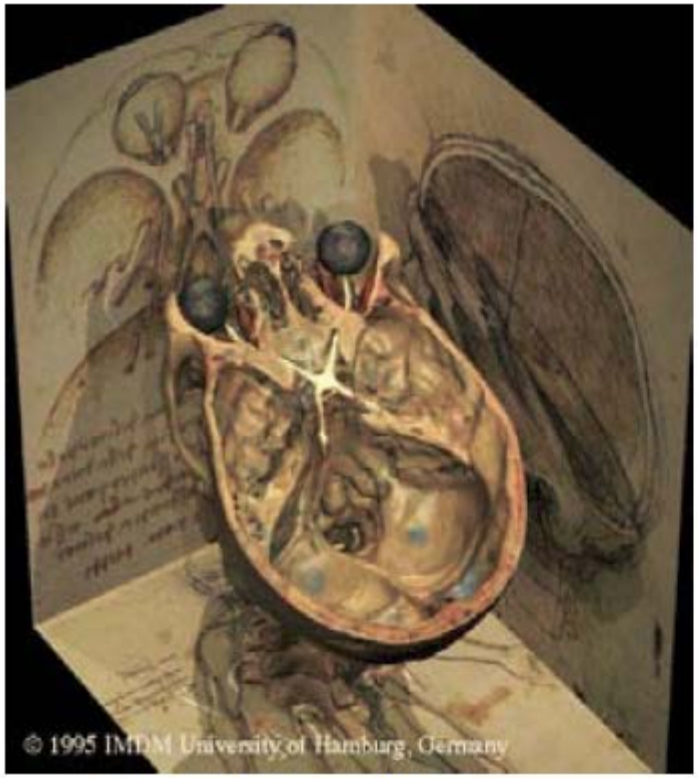
# Computer-Aided Design

Sketchup



ProEngineer

# Visualization




**The Virtual Human**
**Karl-Heinz Hoehne**

**Outside-In**
**The Geometry Center**

# Visual Simulation and Training

- Apollo spacecraft
- Flight simulators
- Driving simulators
- Surgical simulation

**davinci surgical robot**
**Intuitive Surgical**


**Driving simulator**
**Toyota Higashifuji Technical Center**

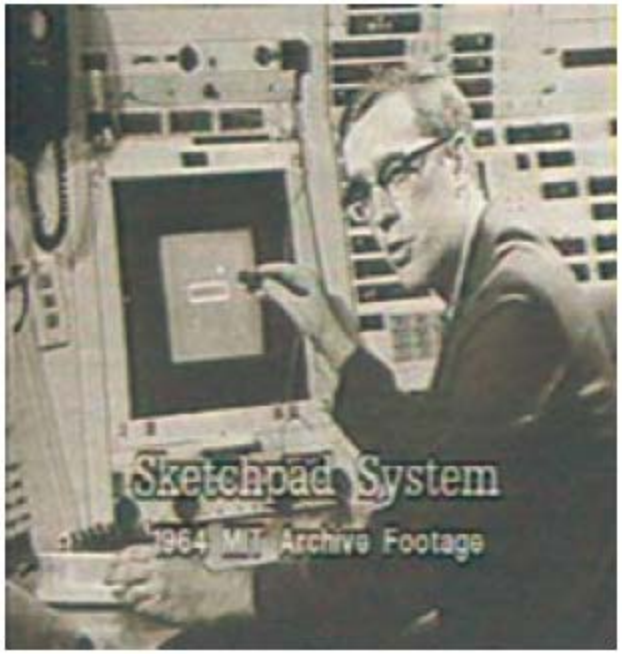
# Digital Media Technologies

- Digital photography
- Inkjet and laser printers
- Digital video and HDTV
- Electronic books
- Graphics on the web:
  - Photos (flickr)
  - Videos (youtube)

**Sony Video Camera**

**Apple Laserwriter**

# User Interfaces




**Ivan Sutherland, Sketchpad, Light-pen, vector display**

**Apple iPad**

**Console Controller**

# Virtual Reality

- Immersive interfaces
  - Input: 3D 6-DOF tracking, gloves
  - Output: Head-mounted and projection displays

**Ivan Sutherland: Head-mounted displays, with mechanical tracker**


**Oculus Rift**

# Fun Movies on Computer Graphics

**Creating a Maelstrom for Pirates 3**

http://www.youtube.com/watch?v=Tmm4BQX8TCQ

**Microsoft Hololens**

https://www.youtube.com/watch?v=pLd9WPlaMpY

**SIGGRAPH 2016**

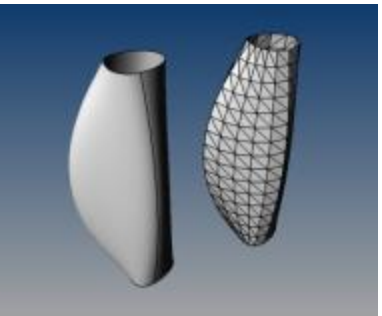
https://www.youtube.com/watch?v=dQBJ0r5Pj5s

# What is Computer Graphics? (1/2)

- Computer graphics implies the creation, storage and manipulation of **models** and images

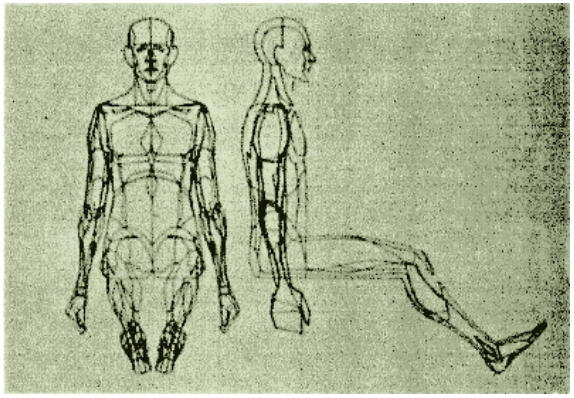
- Such **models** come from diverse and expanding set of fields including physical, mathematical, artistic, biological, and even conceptual (abstract) structures

- We often think of computer graphics as the methodology to convert these **models** into a discrete form (i.e. pixels on your display)



# What is Computer Graphics? (2/2)

- The term "*Computer Graphics*" was coined by **William Fetter** while working at the Boeing Company in 1964
- Fetter was designing a series of images (for pen plotters) to help explore cockpit designs in aircrafts. To do so, he used a 3D model of a human.

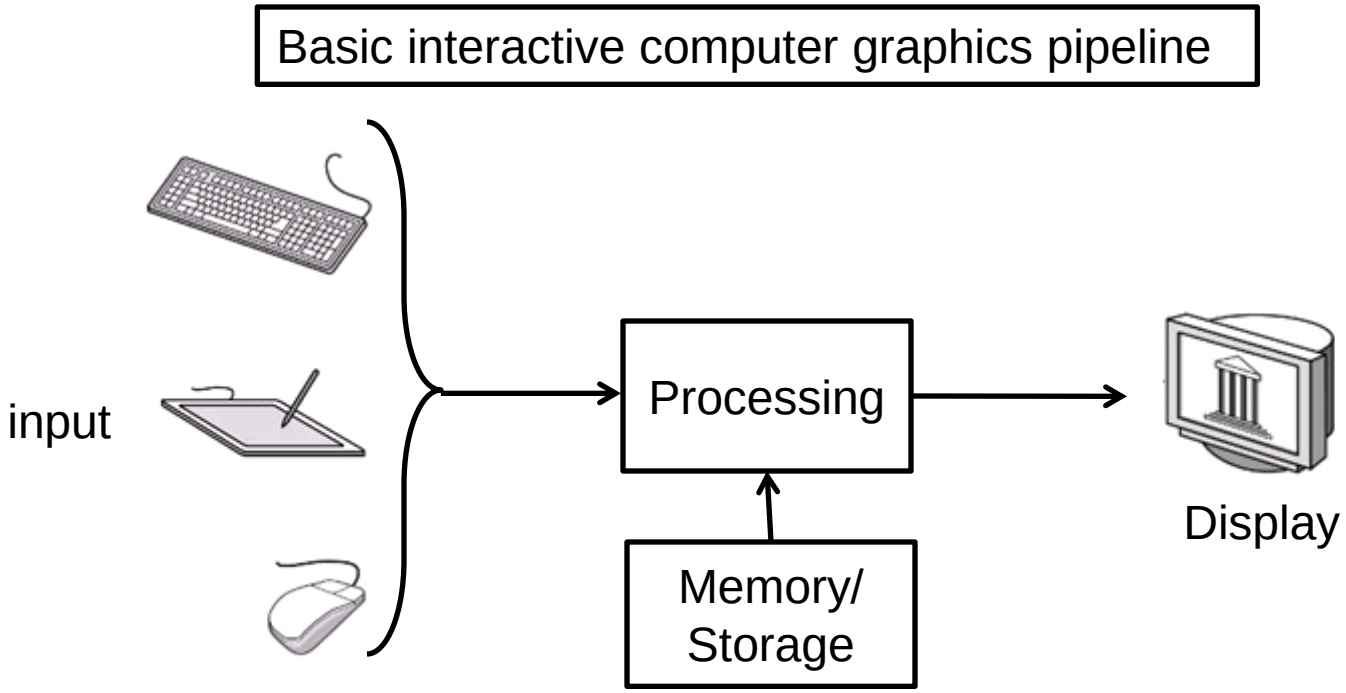


Output on pen plotter


Pen plotter
Commands:
moveto (x,y)
Pen up/pen down

# Interactive Computer Graphics (1/4)

- Interactive Computer Graphics is where the user controls/manipulates the graphics with rapid feedback

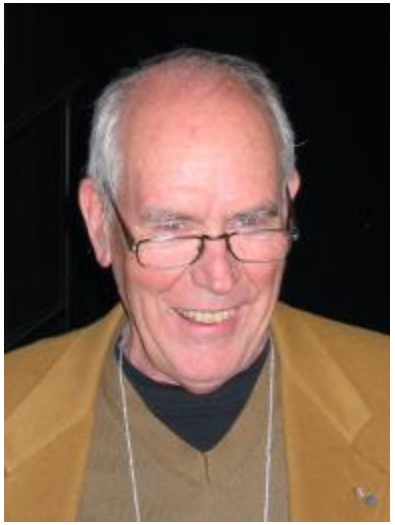



- Most video games fall into this category

# Interactive Computer Graphics (2/4)

- One of the first interactive system was demonstrated by Ivan Sutherland in 1963.
- Ivan is considered the “father” of computer graphics
  - You’ll see why through out this course . . he really is the “father”

Ivan’s 1963 MIT PhD thesis, **Sketchpad**, was on using a light-pen + CRT to draw pictures



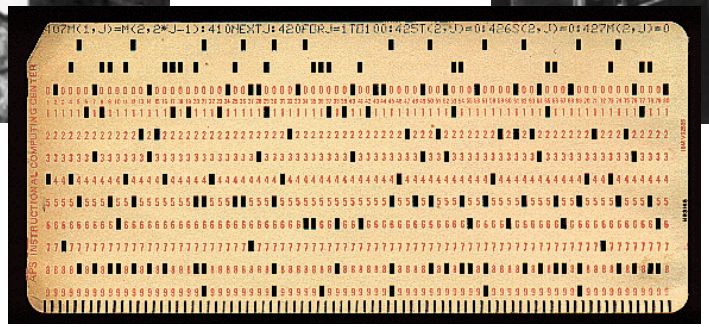
After working at Harvard, University of Utah, Evan and Sutherland’s, Cal Tech, CMU, he is now at Sun Microsystems

http://www.youtube.com/watch?v=USyoT_Ha_bA

(The man in the video is not Ivan, Ivan is pictured above.)

# Interactive Computer Graphics (3/4)

- Before, Sutherland's Sketchpad
- Input was on "punch cards", and was batch processed
- Output was on pen-plotters, or just numbers



# Interactive Computer Graphics (4/4)

- Sutherland's thesis laid the foundation for how computer graphics would operate from this point forward.

  *"The Sketchpad system uses drawing as a novel communication medium for a computer. The system contains input, output, and computation programs which enable it to interpret information drawn directly on a computer display. Sketchpad has shown the most usefulness as an aid to the understanding of processes, such as the motion of linkages, which can be described with pictures. Sketchpad also makes it easy to draw highly repetitive or highly accurate drawings and to change drawings previously drawn with it…"*

From the abstract of Sutherland's PhD thesis

http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-574.pdf
(if you are bored, you can downloaded Sutherland's thesis here.)

# Computer Graphics Revolution (1/7)

- Graphics has been key to technology growth in evolution of computing environments:
  - graphical user interfaces (GUIs)
  - visual computing, e.g., desktop publishing, scientific visualization, information visualization
- Hardware revolution drives everything
  - every 12-18 months, computer power improves by a factor of 2 in price / performance – Moore's Law
  - graphics memory and network speeds are on even faster exponentials
    - Graphics chips in particular have major improvements every six to nine months (e.g. nVidia GeForce™ series, ATI Radeon™ series)

# Computer Graphics Revolution (2/7)

- **Character Displays** (1960s-now)

  - **Display** = text + alpha + pseudo-graphics

  - **Objective** = command-line processing

  - **Application** = single task applications; terminals on large mainframe computers

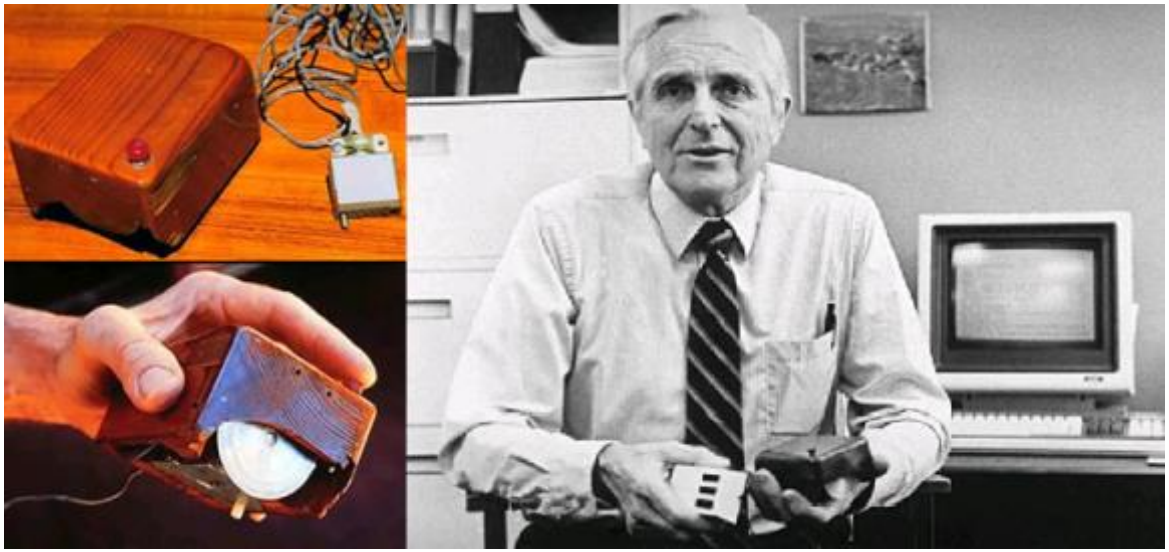

# Computer Graphics Revolution (2/7 cont')

- **Character Displays** (1960s-now)

  - First **mouse** demonstrated by Douglas Engelbart while working at Stanford Research Institute

  - It was actually demoed using a Character Display, they even introduced the notion of "cut and paste"

  - The mouse was part of what is called the "Mother of all Demos"

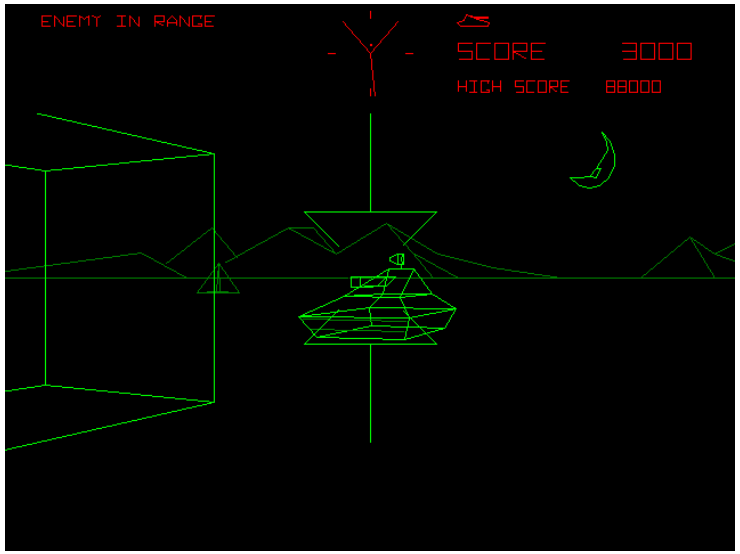


http://www.youtube.com/watch?v=JfIgzSoTMOs

# Computer Graphics Revolution (3/7)

- **Vector Graphics** (1970s-now)

  - **Display** = No framebuffer, but line drawings, stroke text; 2D/3D transformation, graphics is born
  - **Objective** = command-line typing, menus, graphics
  - **Application** = single task, multi-task on a PC was starting to emerge




http://www.youtube.com/watch?v=mtjFl8w3Em4

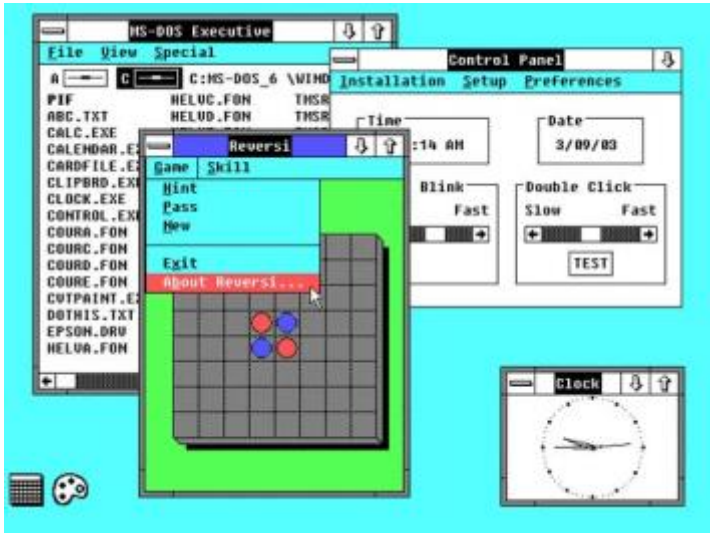
# Computer Graphics Revolution (4/7)

- **Raster Graphics** (1970/80s-now)

  - **Display** = framebuffer of pixels, windows, icons, legible text, raster graphics!
  - **Objective** = more than just typing, window's metaphor, direct manipulation of "icons" with a mouse
  - **Applications** – multi-tasking, client-server architectures

*Xerox Parc* introduced GUI interface (1972)

[Xerox Parc executives thought it had no future!]




MS Windows 2.0

Evans & Sutherland released the first commercial framebuffer in 1974, costing about US$15,000. It was capable of producing resolutions of up to 512 by 512 pixels in 8-bit grayscale color.

# Computer Graphics Revolution (5/7)

- **3D Graphics Workstation** (1980/90s-now)

  - **Display** = raster display with special hardware support

  - **Objective** = to provide high-performance Computer Graphic

  - **Applications** - visualization/animation/scientific work, all on multi-tasking machines

Silicon Graphics (SGI) was the “king” of high-end workstations in the 1990s. Machines were expensive, from $10K a workstation, to $1+million for multi-processor machines.

But something was going on in the CG industry that would be the doom of SGI. . ..

SiliconGraphics ics

# Computer Graphics Revolution (5/7)

- **3D Graphics Workstation** (1980/90s-now)

  - **Display** = raster display with special hardware support

  - **Objective** = to provide high-performance Computer Graphic

  - **Applications** - visualization/animation/scientific work, all on multi-tasking machines

Silicon Graphics (SGI) was the “king” of high-end workstations in the 1990s. Machines were expensive, from $10K a workstation, to $1+million for multi-processor machines.

But something was going on in the CG industry that would be the doom of SGI. . . . PC graphics cards were coming.

SiliconGraphics ics

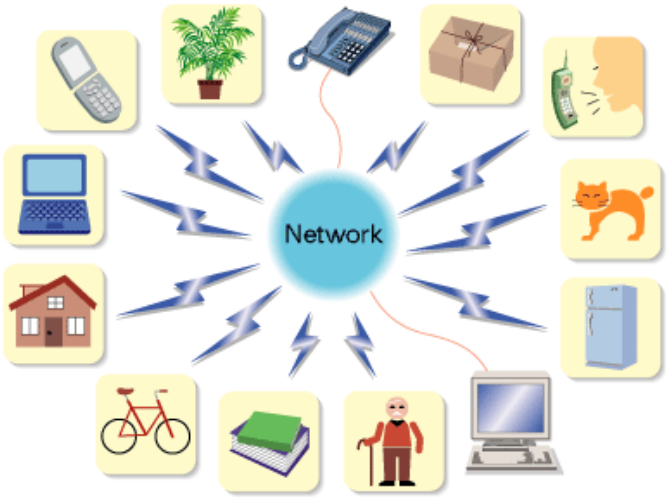
# Computer Graphics Revolution (6/7)

- High-end PC graphics cards – nVidia, ATI
  - Display = PC market + game consoles
  - Objective = low-cost, high-performance graphics
  - Application = driven by game market

Fact:The gaming industry has grown to be a powerful as Hollywood. Game releases can often make as much money as Hollywood blockbusters. . .

# Computer Graphics Revolution (7/7)

- Computing is still evolving
  - Now, we have multiple computers serving single devices (i.e. cloud computing)
  - Novel Input Devices are immerging (wii-mote)
  - Trend is small, lighter devices (netbook, iPhone)
- Graphics/GUI is still at the core
  - can we really go back to text-only displays?




# Application Distinction in CG

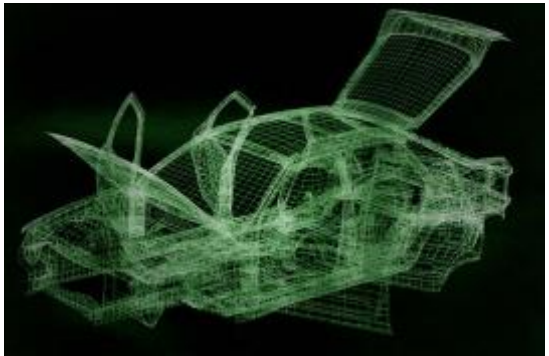
## 1. Sample-based Graphics

- discrete samples are used to describe visual information
  - pixels can be created by digitizing images, using a sample-based “painting” program, etc.
  - able to create photo-realistic images, because input are photos!
  - example programs: Adobe Photoshop™, GIMP™ , Adobe AfterEffects™

Image of the Merlion.
A photo is an example of Sample-based graphics

## 2. Geometry-based Graphics

- geometrical model is created, along with various appearance attributes, and is then sampled for visualization (result is *rendering*);
  - often some aspect of physical world is visually simulated, or “synthesized”
  - examples of 2D apps: Adobe Illustrator™, Adobe Freehand™ (formerly by Macromedia), Corel CorelDRAW™
  - examples of 3D apps: Autodesk’s AutoCAD2009™, Autodesk’s (formerly Alias|Wavefront’s) Maya™, Autodesk’s (formerly Discreet’s) 3D Studio Max™

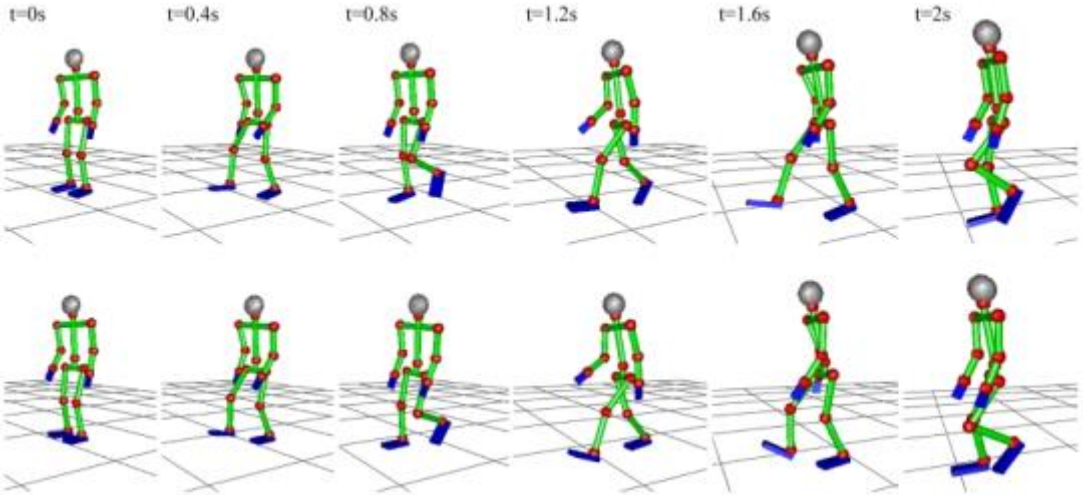


# Sample-based Graphics

- Samples created directly in paint-type program, or as sampling of continuous (analog) visual materials.
- For example, photograph can be sampled (light intensity/color measured at regular intervals) with many devices including:
  - flatbed and drum scanners
  - digital still and motion (video) cameras
  - add-on boards such as frame grabbers
- Sample values can also be input numerically (e.g., with numbers from computed dataset)
- Once an image is defined as pixel-array, it can be manipulated
  - *Image editing:* changes made by user, such as cutting and pasting sections, brush-type tools, and processing selected areas
  - *Image processing:* algorithmic operations that are performed on image (or pre-selected portion of image) without user intervention. Includes blurring, sharpening, edge-detection, color balancing, rotating, and warping. Pre-processing step in computer vision

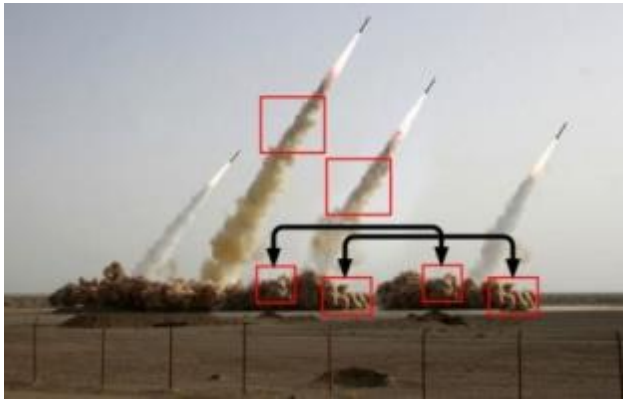
# Sample-based CG Beyond Images

- Sample-based graphics also include motion-capture
- The motion is a “sample” of real motion



# Advantages of Sample-Based CG

- Sample-based images can be easily manipulated
  - And the quality of this manipulation is getting better and better (Adobe Photoshop)
- Images can be easily combined
  - This is called “matting and compositing”



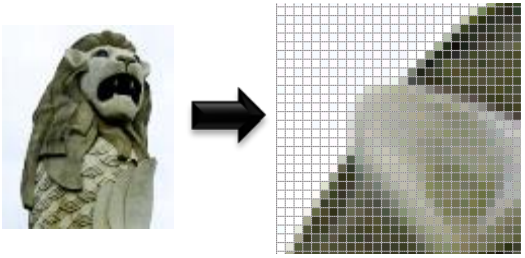
Fake Missile Image from Iran. Fake Chinese Tiger →

Matting and Compositing

Adobe Photoshop

# Disadvantages of Sample-Based CG

- What you see is all you get (WYSIAYG)
  - There is no additional information
  - Images can't be viewed from another point of view, resolution cannot be increased
  - Processing is therefore limited



- But . . . .
  - These limitations are changing
  - There is a whole area called "image-based-rendering" that deals with using images to render novel views, extract 3D, etc. . .
  - But that is another course. . (e.g. Computer Vision)

# Geometry-based CG

- **Geometry-based graphics applications** store mathematical descriptions, or “models,” of geometric elements (lines, polygons, polyhedrons…) and associated attributes (e.g., color, material properties). Elements are primitive geometric shapes.

- **Images** created as pixel arrays (via sampling of geometry) for viewing, but not stored as part of model. Images of many different views are generated from same model. We call this “rendering”.

- Users cannot usually work directly with individual pixels in geometry-based programs; as user manipulates geometric elements, program resamples and redisplays elements

- **Increasingly, rendering combines geometric and sample-based graphics, both as performance hack and to increase quality of final product**
  - **For example texture/environment maps, and motion capture**

# Geometric Modeling

**WHAT IS A MODEL?**

- Captures salient features (data, behavior) of thing/phenomenon being modeled
  - data includes geometry, appearance attributes…
  - note similarity to Object-Orient Programming Concepts
- Real: some geometry inherent
  - physical (e.g., actual object such as a car engine)
  - non-physical (e.g., mathematical function, weather data)
- Abstract: no inherent geometry, but for visualization
  - organizational (e.g., company org. chart)
  - quantitative (e.g., graph of stock market)
- Modeling needs to cope with complexity – and that is hard!
- In this class, we won't deal too much with modeling, but instead how to work with models
  - We will need to understand them, and their common representations

# Geometric Modeling

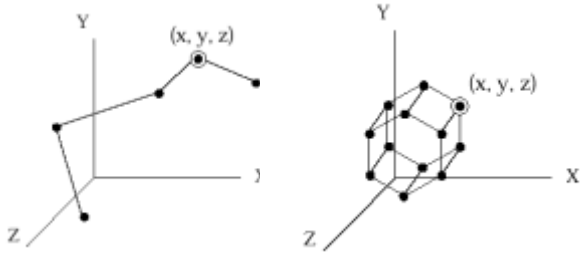
- Appreciate the modeler and the complexity!
  - Modeling is generally taught in art-school, or graphics design
  - Relies on software like 3D studio Max, Maya, Z-brush, etc
  - Takes hours/days to create models
  - ***We will not focus on model creation!***

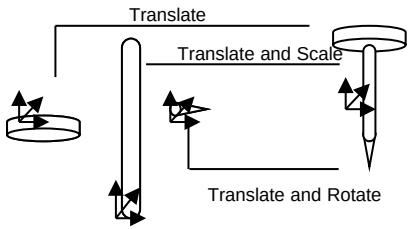
Maya

Zbrush

# What do we need to know?

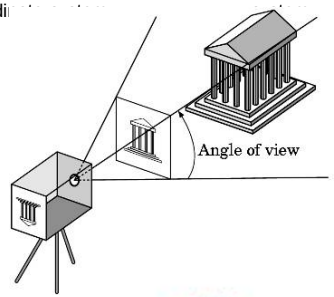
- **Basic Geometric Primitives**
  - How primitives are represented?
- **Transformations**
  - Primitives coordinates are manipulated to create models
  - Models populate scenes
- **Viewing**
  - Given a 3D world, how to we generate the 2D image on the screen
- **Lighting**
  - How can we simulate lighting to make our world more realistic
- **Texture/texture maps**
  - Use images to fake “detail” on the surface

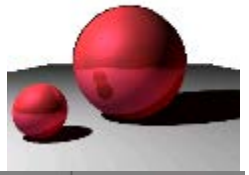



Primitives in their own modeling coordi

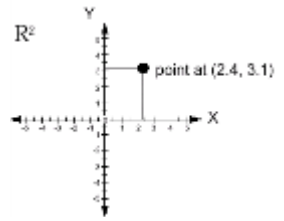
Composition in world (root) coordinate

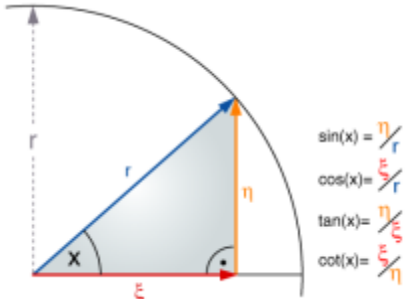


# What math do we need?

- Euclidian Geometry and Cartesian Coordinates
- Trigonometry and Polar Transforms
- Vector/Matrix Manipulation
  - Vector math
  - Matrix multiple
- Basic Calculus
  - Basic Non-linear functions and derivatives



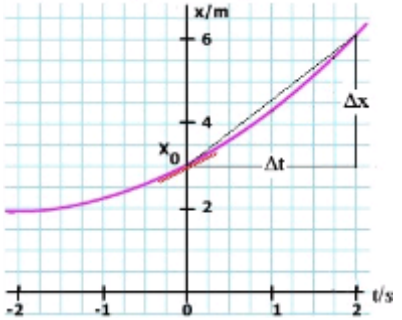

$$\mathbf{p}_1 = \mathbf{M}\mathbf{p}_0$$

$$\begin{bmatrix} x_1' \\ y_1' \\ w_1' \end{bmatrix} = \begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_1'/w_1' \\ y_1'/w_1' \end{bmatrix}$$

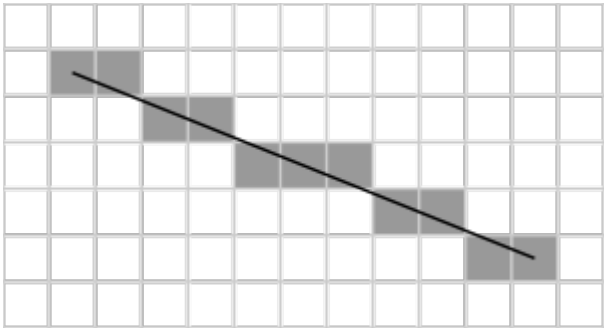
$$x_1 = \frac{m_0 x_0 + m_1 y_0 + m_2}{m_6 x_0 + m_7 y_0 + 1}$$

$$x_2 = \frac{m_3 x_0 + m_4 y_0 + m_5}{m_6 x_0 + m_7 y_0 + 1}$$



# What do we have to be thankful for?!?

- OpenGL Application Programming Interface (API)
- OpenGL does a lot of the dirty work that you want to not worry about . .
  - For example, scan convert primitives to pixels



A line is specified by two end points; this is a mathematic expression. OpenGL will draw the correct pixels.

OpenGL will convert your primitives to pixels so you don't have too.

# So, are you ready?

$$
\begin{aligned}
\mathbf{p}_1 &= \mathbf{M}\mathbf{p}_0 \\
\begin{bmatrix} x_1' \\ y_1' \\ w_1' \end{bmatrix} &= \begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \\
\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} &= \begin{bmatrix} x_1'/w_1' \\ y_1'/w_1' \end{bmatrix} \\
x_1 &= \frac{m_0 x_0 + m_1 y_0 + m_2}{m_6 x_0 + m_7 y_0 + 1} \\
x_2 &= \frac{m_3 x_0 + m_4 y_0 + m_5}{m_6 x_0 + m_7 y_0 + 1}
\end{aligned}
$$