- Introduction
  - Computer stuff
  - Pixels
  - Line Drawing
- Video Game World
  - 2D 3D Puzzle
  - Characters
  - Camera
  - Time steps
- Geometry
  - Polygons
  - Linear Algebra
  - NURBS, Subdivision surfaces, etc
- Movement
  - Collisions
    - Fast Distances
    - Pre-calculated
    - Time step problems
  - AI
- Graph Algorithms
- Calculus
- Rotations (abstract algebra - Quaternions)
- Physics
- Sound
- (Pseudo) Random number generation
- Image and data compression
- Data security
- Inverse Kinematics
- Rendering
  - Texturing
  - Lighting
  - BSP
- Editor tools
  - World Creation
  - Color space reduction
  - Mathematica path tool
  - Group rotation stuff
- Summary

Introduction

2D Games – Tetris, Pacman, Pong, Solitare

3D Games

Trying to create a simulation of a world

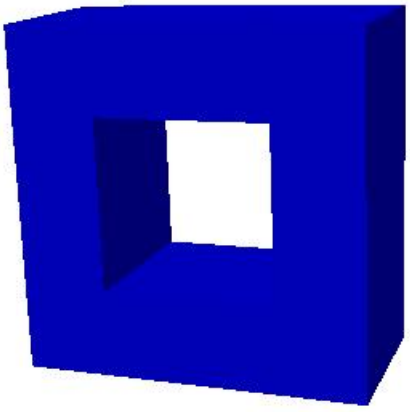
Basic game loop – how the game does its stuff:

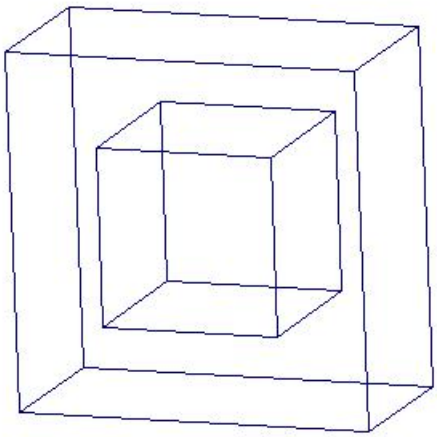
Easiest to think of time slices – game takes snapshots once per frame (30-100 frames a second, may be variable)

- Get all inputs from players
- Get all computer calculated moves (AI)
- Do physics, collisions, explosions, etc
- Send any results over network to online players
- Render screen (usually costly! 100,000 polys)
- Repeat process

Geometry

Most objects in 3D world are (currently) polygonal models





Might and Magic series

Basic Computer stuff: Bresenham's Line algorithm (1965) – now in hardware, but concepts apply. Discrete math if necessary!

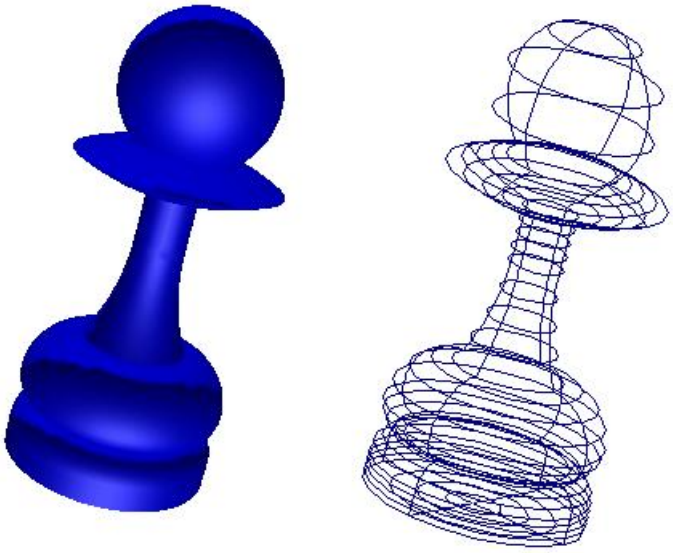
Linear algebra

- Rotations, translations, composition, stored as matrices, easy to concatenate
- The core of any 3D game is a big collection of linear algebra routines

Position = rotation*rotation*translation*projection….

Surface modeling

- Allows more freedom for geometry
- NURBS, splines, subdivision surfaces, current research areas in computer graphics



Custom modeler – Lomont

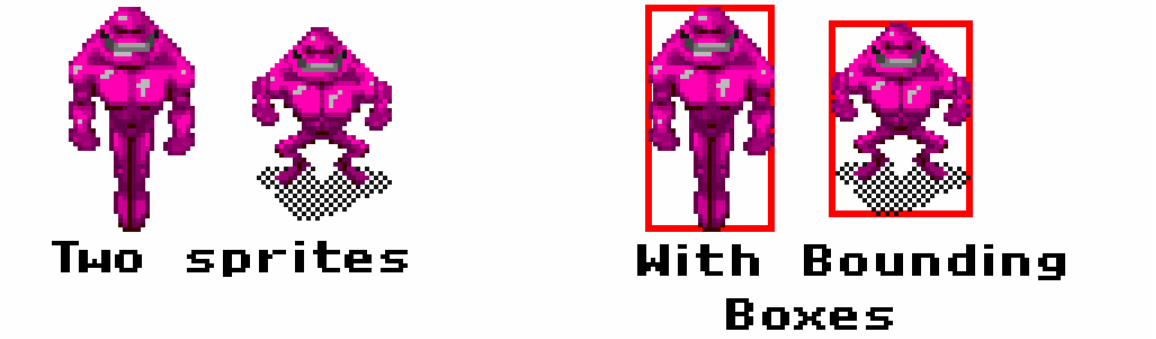
Geri's Game – Pixar

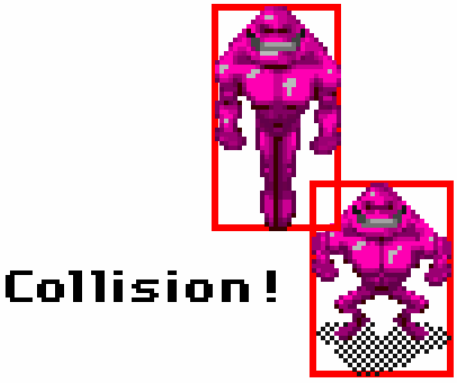
Quake 3 Screenshot – Id Software

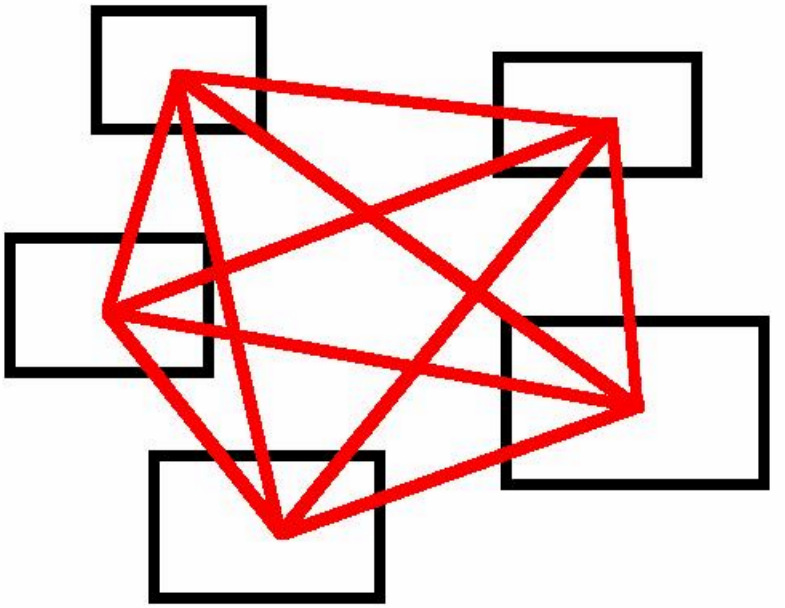
Doom 3 – Unreleased Id Software

Movement

Collisions – needs a fast system. Main idea for collision detection is to use <u>bounding boxes.</u>







Number of collision tests needed goes up as n^2

Fast square root tricks:

To find approximate distance between two 2D points, look at the axis aligned right triangle, and half the short side plus the long side will be within **11.8%** of the correct distance.

There are also very fast tricks based on IEEE floating number formats, reducing a 70 cycle sqrt on a Pentium III to about 9 cycles, with an error of about **1%.**

Similarly, in 3 space, ordering the sides in decreasing length dx_1, dx_2, dx_3

$$\mathbf{0.9398\ dx_1 + 0.3893\ dx_2 + 0.2987\ dx_3},$$

This has a max error of **6.0%**, quite accurate, and can be computed quickly with only shifts and adds (no multiplies).

Have time step problems, so need 4D versions, etc, and one can compute the optimal coefficients in each case numerically, say with Mathematica.

These problems only increase in 3D, with characters needing to navigate complex geometry and not get stuck.

Numerical analysis knowledge needed.

AI – artificial intelligence

- Must be fast (for most games)
- Opponents should be fun to play against, so must act smart.
- To read the literature, a programmer needs some math skills
- New tools usually written so level designers can easily script behavior without complex programming needs.

Graph Algorithms:

- Finding paths around the world, A* algorithm
- Finding best solutions in AI
- In a 3D world, the creatures must be able to find their way around, and this involves searching for realistic paths
- Online massively networked games need a lot of good models to make them play in realtime. 100,000 users is a lot to manage at 60fps.

Calculus:

- Numerical integration – RK4, Euler, Midpoint
- Volumes, areas, rates used.
- Many calc concepts used to derive algorithms
- Taylor series (and more complex versions like Pade and discrete methods) used for computations.
- Almost all areas of game programming need a lot of calculus understanding.

Rotations (some abstract algebra):

- Euler angle parameterization lacking
- Quaternion solution used most often for camera tracking, etc.
- Currently a new area, geometric algebra, is being investigated to simplify code.

Physics

- Dynamics – mass-spring, gravity
- Motion equations i.e., $x = x_0 + v_0 t + \frac{1}{2} a t^2$, and vector versions
- Lighting calculations
- Rotational dynamics
- Very important for flight sims, golf, baseball, sims
- Collision and impulse resolution is one of the most difficult things to get right – think of modeling a stack of boxes and balls, some hard and some soft, falling together down a flight of uneven stairs…. Very hard to do convincingly in realtime.

Sound

- 3D audio necessary for modern games.
- Acoustics for realistic sounds, mostly hardware.
- Sound incidence – where can you hear from.
- Sound decay and mixing of multiple sounds.
- A dynamic, changing soundtrack is much better than a pre-composed piece, but very hard to do.

Random number generation

- Current research area, very mathematical
- For most video games, any reasonable one will do
- Want one that is repeatable, i.e., can start at the same point – aids in debugging and logging games
- For online games where people are trying to hack into your game, you need very good security and thus very good random number generation. The C rand() function is very bad.

Image and data compression

- Wavelet, JPEG, MPEG for video
- Data compression to fit more stuff in the game
- Custom algorithms used a lot
- Gameboy Advance, for example, is memory limited, so to fit bigger games you need lots of good compression algorithms

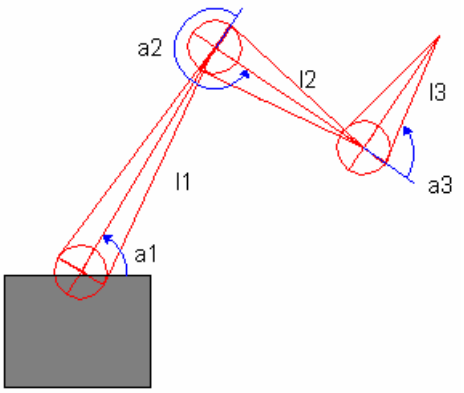
Online security – crypto and protocols

- To prevent people from cheating online
- Copy protection to make pirating harder
- To protect proprietary information
- They have a copy of your game which they disassemble, and can monitor all network traffic, so this is hard.

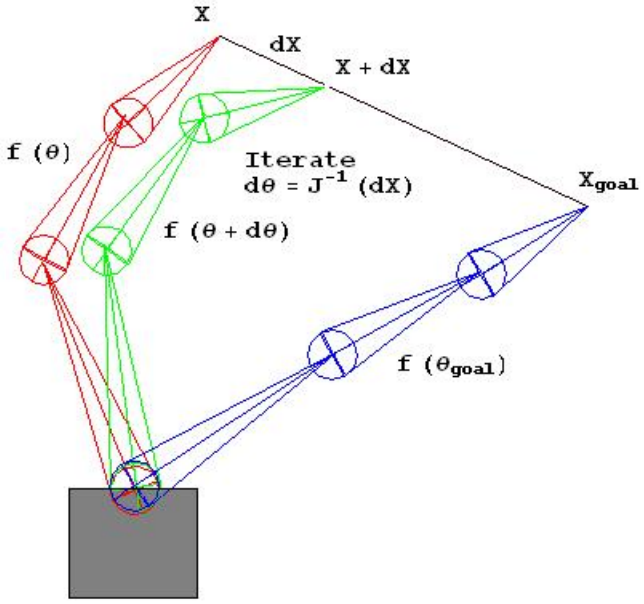
Algorithms galore

- Basics: Searching, sorting, trees.
- Advanced: almost everything used somewhere!

Inverse kinematics – for moving humanoid creatures

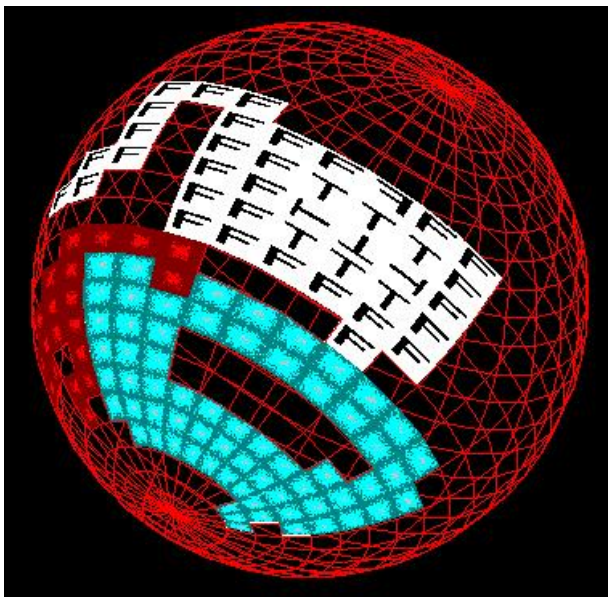


Given a skeletal structure consisting of lengths l_i and angles a_i, easy to compute the end point of the structure



Rendering – hardware based now.

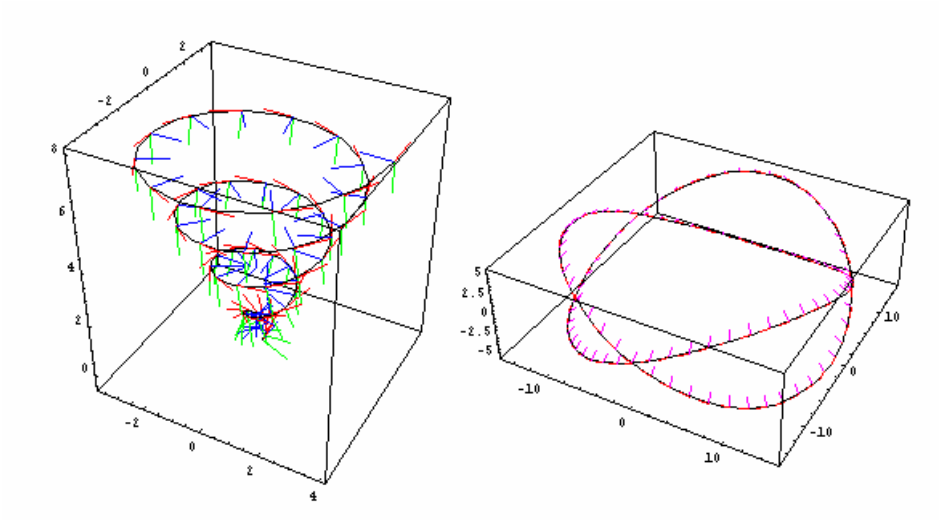
- Texturing – drawing pictures on polygons – mostly done in hardware now.
- Very time consuming to sort, clip, transform, and draw 100,000-500,000 polygons over 50 times a second
- Real-time lighting calculation affect textures
- BSP, other methods to speed up rendering
- Need to know vertex and pixel shaders – language for the graphics pipeline.



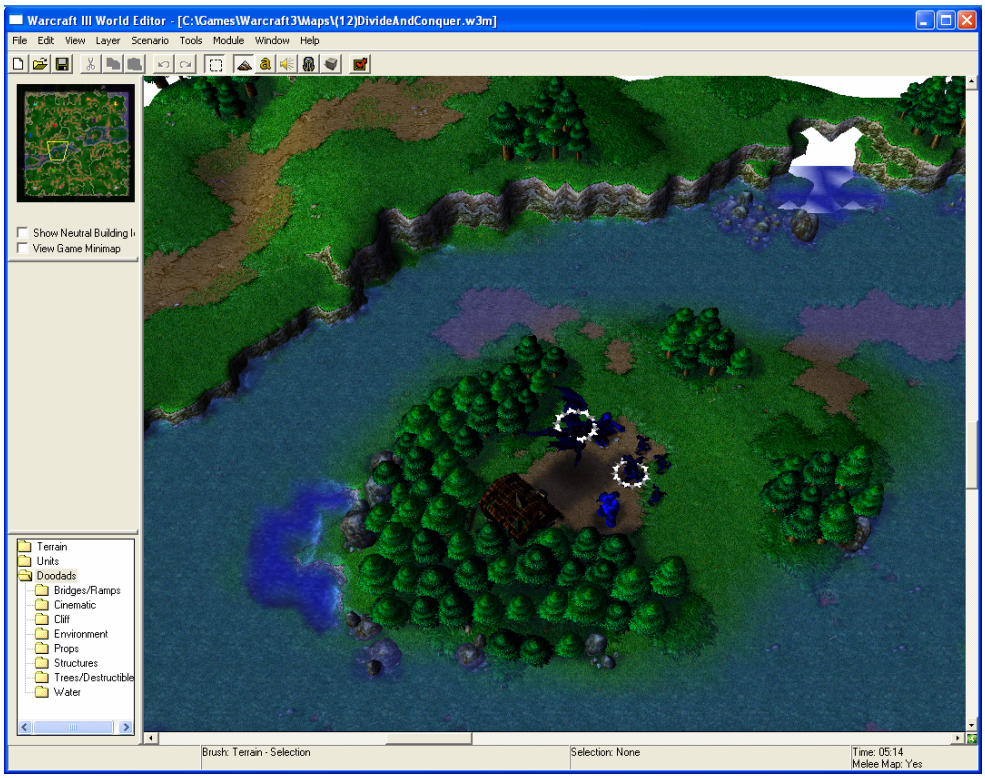
Custom tool for polygon mesh painting

Tools – Very important

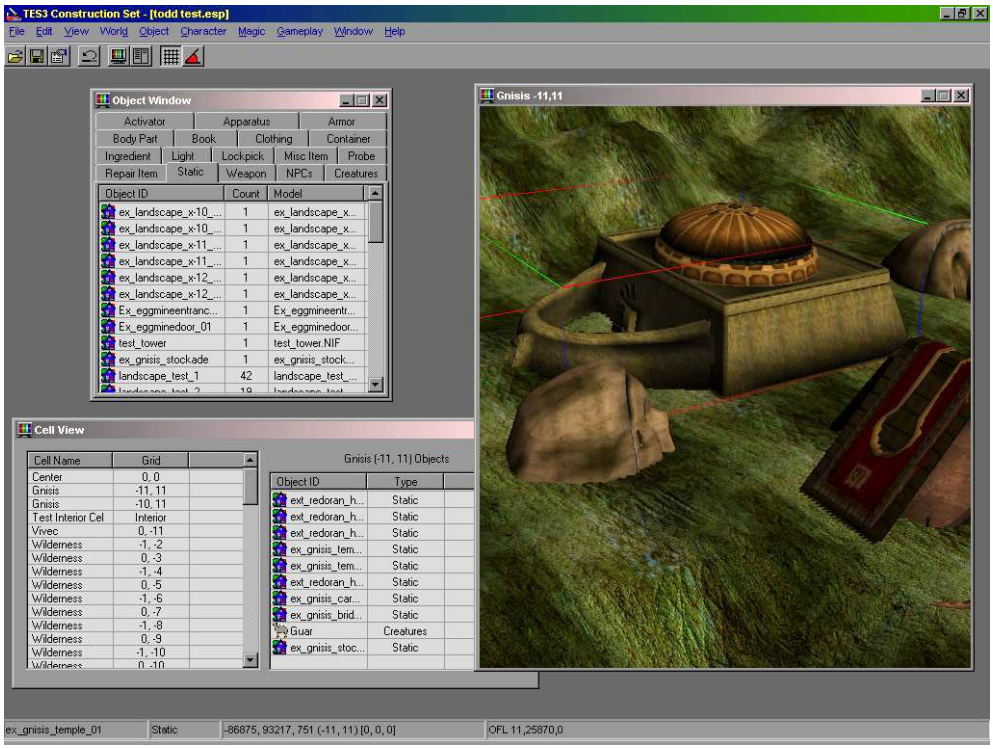
- Color space reduction (obsolete almost)
- World editor – groups
- Flight path tools
- AI scripting
- Event scripting
- Editing geometry
- Creating geometry
- Creating sounds
- Creating textures



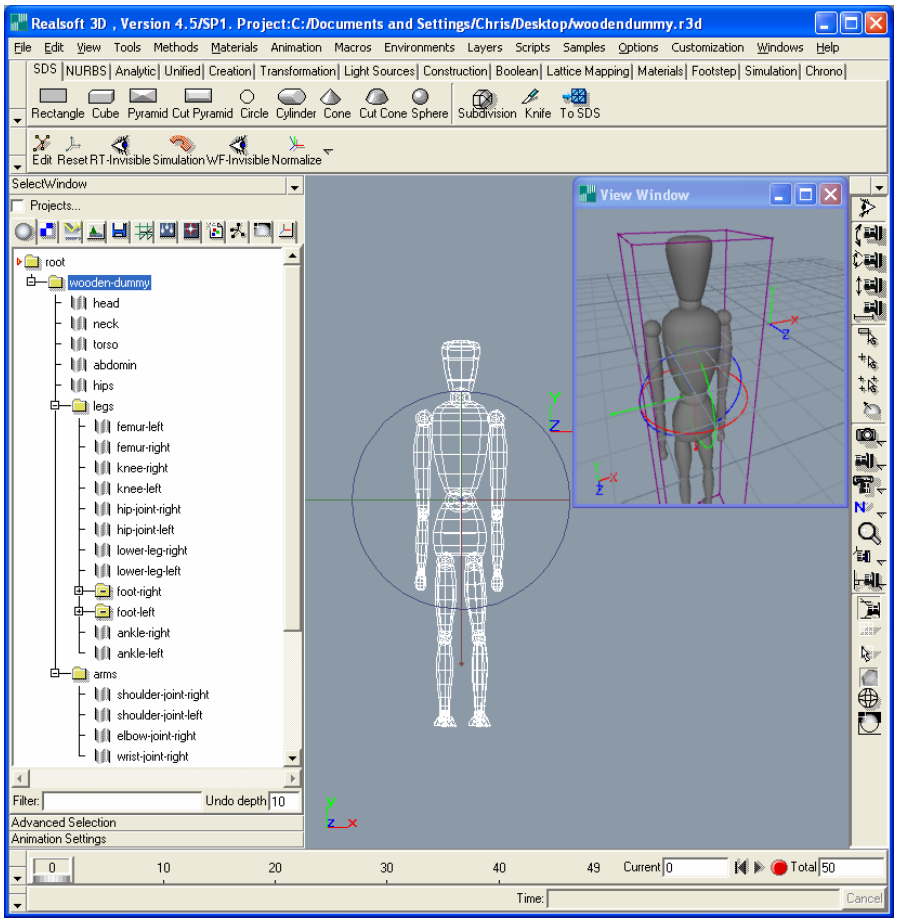
Custom path editing from Mathematica modeled flight paths

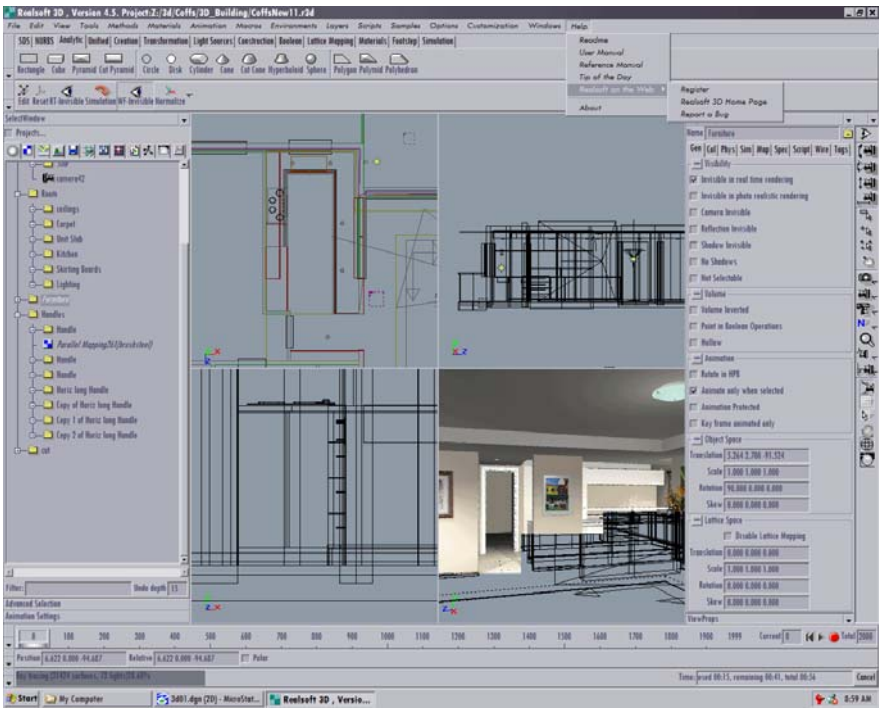


Warcraft 3 editor

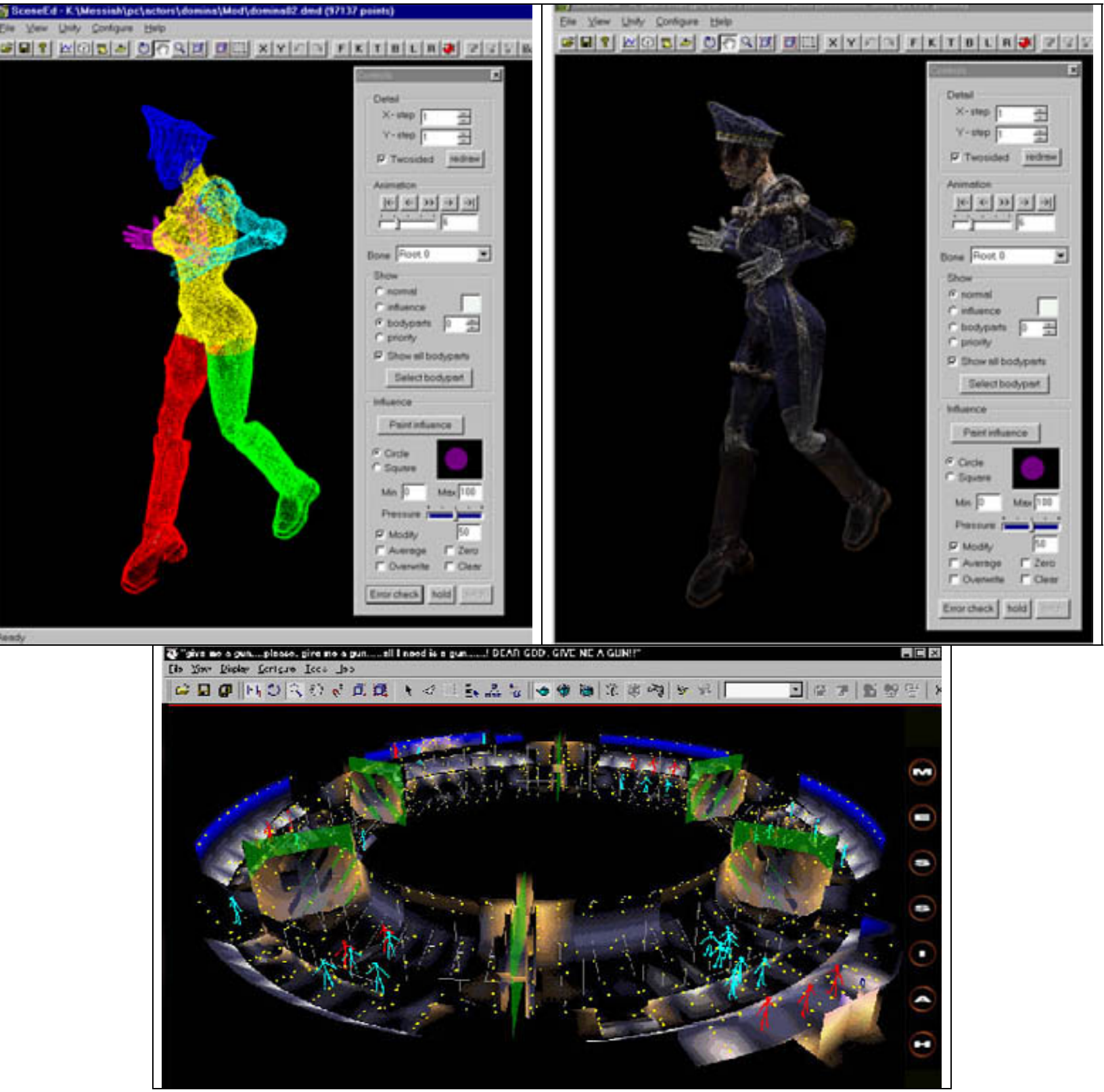


Morrowind editor





Realsoft 3D modeler and renderer

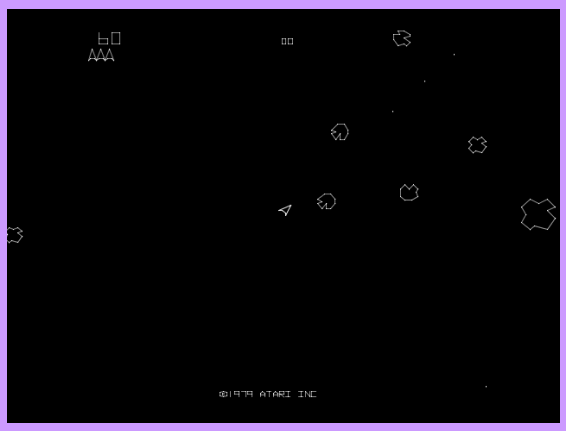


Tools from Shiny Entertainment for Messiah

A brief history of video games

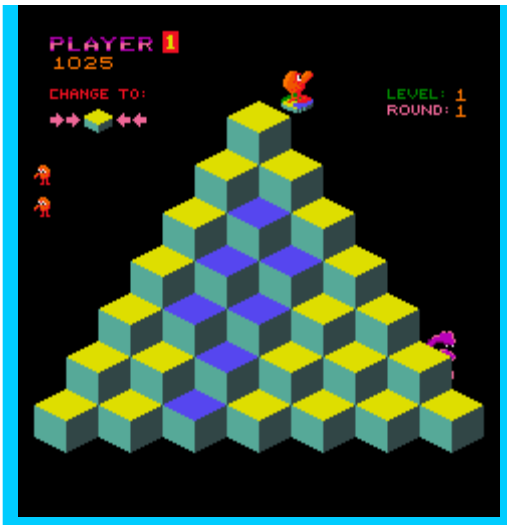
Maze - 1976 - 8080 2MHZ 256x224 2 colors

Space Invaders - 1978 - 8080 2 MHZ 224x240 2 colors



Asteroids - 1979 - M6502 1.5MHZ Vector based graphics chip

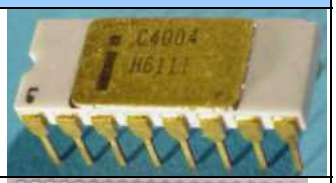
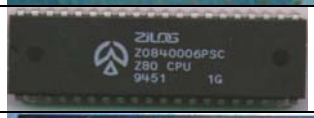
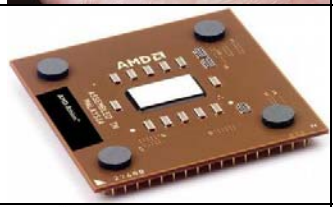
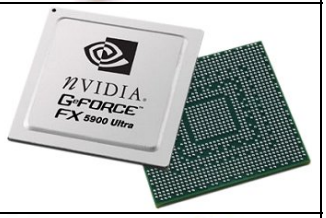
Pacman - 1980 - Z80 3 MHZ - 224x288 16 colors



Qbert - 1982 - 8086 5 MHZ, sound M6502 894 khz - 240x256 16 colors

Kings Quest 1: Quest For The Crown – 1984, commissioned by IBM as a showpiece for their PCjr, and was the first game to support EGA color cards. 160x200 – 16 color 256K RAM

Hardware then and now

1975-1980	Transistor count	Speed	Game size	Graphics	
Intel C004 4 bit CPU, 1st single chip CPU	2300	60khz	?	?	
Zilog Z80 8 bit still popular today	6000?	3.5 mhz	5-20K	300x300 2-16 color	
Motorola 6809 8 bit	6000?	1 MHZ	5-32K	300x300 2-16 color	
2004					
Intel P4 32 bit	55 million	3.4 GHZ	1-5GB	1200x1200 16.7 million color	
AMD Athlon XP 32 bit	55 million	2.2 GHZ	1-5GB	1200x1200 16.7 million color	
Nvidia 5900 Ultra 256 bit	130 million	500 Mhz	1-5GB	1200x1200 16.7 million color	
Playstation 2 CPU 128 bit	53 milliom	300 MHZ	2 GB	640x480 16.7 million color	

Realtime graphics from some modern games/demos 2004:

Morrowind

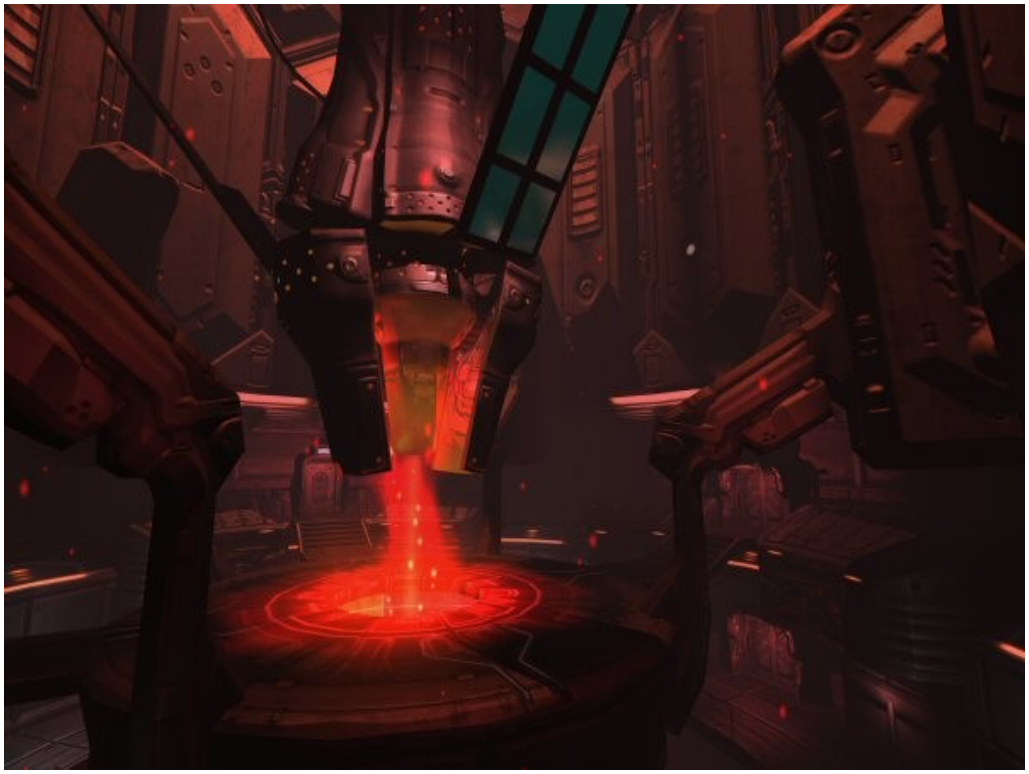
Dungeon Seige

Unreal 2003

Nvidia Vulcan demo

NVidia Dusk demo

Doom 3 – Soon to be released



Summary – some areas of math I have used in programming for the video game industry

- Linear algebra (most important area of math for a modern game programmer to know! – 3D transforms)
- Vector calculus
- Small amounts of logic – conditional reduction
- Complex analysis – rotations, used in math routines
- Calculus - areas, volumes, slopes, instantaneous velocities, center of mass
- Geometry – used in many areas of modeling, to understand relationships between components in a virtual world
- Analytic geometry – we always use coordinates!
- Graph theory – path finding, searching graphs for certain properties, does a path exist?
- Abstract algebra – quaternions, dihedral groups
- Differential equations – motion, numerical solution
- Numerical analysis – stability of solutions, analysis tricks, surface modeling
- Discrete math – pattern matching, counting arguments, random number generation, tiling problems
- TONS of game specific tricks to speed up calculations: table lookups, math tricks, reductions, hardware tricks based on platform
- No Galois theory used as far as I know

END OF FILE