Parallel Architectures for Computer Graphics Displays

by Marc Hannah, 1985

Philip Levis
Stanford CS349G
Winter 2021
1985

- Mac 512kB
- 6MHz, 32-bit
- US $2,795
  - $7,000 today
1985

- Mac 512kB
- 6MHz, 32-bit
- US $2,795 - $7,000 today
• Drawing graphics is expensive
  - $342 \times 512 = 175,104$ pixels/bits
  - A 6MHz processor can spend $34.2 \text{ instructions/second/pixel}$
  - Visual frame rate is $>24\text{Hz}$, at this time $10\text{Hz}$ considered OK
  - $3.4 \text{ instructions/pixel/frame}!$

• Another bit of historical context: *Racing the Beam* describes how Atari 2600 programmers raced against a CRT beam
Silicon Graphics

- Laid the foundations for graphics hardware today
- Graphics systems have long been a unique strength of Stanford EE and CS
  - OpenGL
  - CUDA
- SIGGRAPH is mostly math
  - Large gulf between what Pixar/Dreamworks do and what academics do, not just in scope but also in content
This Lecture

- Overview some concepts in graphics
- Explain some architecture background/terms
- Won't give away the ending
Linear Algebra

- 3D scenes and manipulation are all linear algebra
- A point is a vector \([x \ y \ z]\)
- All geometric transformations are matrix multiplications or additions

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(\theta) & -\sin(\theta) \\
0 & \sin(\theta) & \cos(\theta)
\end{bmatrix}
\]
Rotate around X axis

\[
\begin{bmatrix}
s & 0 & 0 \\
0 & s & 0 \\
0 & 0 & s
\end{bmatrix}
\]
Scale by \(s\)

\[
\begin{bmatrix}
1 & 0 & 0 & 10 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Translate +10 on X
(make point \([x \ y \ z \ 1]\)
Additive Transformations

• Applying multiple transformations is a matrix stack
  - E.g., scale, then rotate, then translate, then rotate
  - Each transformation is a matrix
  - Can represent entire operation as a single matrix
  - Maintain the stack so you can add/remove operations

• Often talk about different coordinate spaces
  - World coordinates: arbitrary coordinate system for objects
  - Camera coordinates: camera is at 0, 0, 0
3D to 2D
$v_x = \frac{p_x \cdot d}{z}$
Clipping

- Drawing everything takes too long: viewport is usually only a small fraction of a full 360 degree solid angle
- Only draw what's visible
- Cut out parts of objects outside viewing area
Clipping
As you draw each pixel, keep track of its z position (depth). If a pixel $z$ is greater than current pixel, don't draw (it is occluded by the current one). E.g., if $z_t > z_e$, don't draw it.
Meshes and Triangles

- Discretize a 3D geometric shape to a set of triangles
Simple lighting models consider the intensity of the light, the reflectivity of the surface, and the angle of incidence to the surface normal.
Rendering

• Process of transforming a 3D scene into a 2D image

• Ray tracing: trace rays of light as they bounce around a scene (reflection, diffusion, etc.)
Fundamental Challenge

- A lot of the field of computer graphics struggles with the challenges of quantizing the continuous
  - Line are lines, but we have to turn them into pixels
  - Quantizing a curved surface into triangles
  - Simulating fluids, materials, sound

- Related to, but different, from high-performance computing (Taylor's dissertation)
  - High performance computing is about accurately predicting the physical world (chemical reactions, structural stress/strain, heat transfer, explosions, weather, etc.)
  - Graphics is about making something that looks convincing
Dissertation

• Chapter 2: Redesign of graphics hardware pipeline to improve performance
• Chapter 3: Rasterizing geometry into frame buffer
• Chapter 5: Building the hardware
  - I won't talk about this too much, want to leave it as a surprise
Dissertation

- Chapter 2: Redesign of graphics hardware pipeline to improve performance
- Chapter 3: Rasterizing geometry into frame buffer
- Chapter 5: Building the hardware
  - I won't talk about this too much, want to leave it as a surprise
A Geometry Engine is a custom chip that contains 12 instances of a custom Geometry Engine processor, which can be dynamically configured to perform one of these (or potentially other) functions.
The Geometry Engine has 4 matrix multipliers, which constitute the matrix stack being executed. The top of the stack contains the computation of the entire stack, and is applied to every coordinate entering the pipeline.

Adding a transformation $T$ to the stack involves pushing a copy of $M_1$ and multiplying $M_1$ by $T$. There are routines to read out elements of the stack to RAM if it overflows.
The 6 clipping elements each clip points for one plane of the viewing volume (top, bottom, left, right, front, back). A point is in if it passes all 6 clippers.

Suppose you are drawing a line. There are four cases for its two points: in-in (whole line viewable), in-out (line starts in view then goes out), out-in (line starts out of view then goes in), out-out (not viewable).
Scalers

The scaler units compute the actual RGB values of the output vertex.

This computation is based on the polygon color, light source illumination, diffuse illumination, and the reflectivity of the surface.
 Commands/Instructions

• **Matrix instructions**
  - *Push* copy of M1 into stack (M1 and M2 now same)
  - *Multiply* current top of stack M1 by M
  - *Pop* off top of stack
  - *Store, Load*, etc.

• **Drawing instructions**
  - *Move* the current point to a position
  - *Draw* a line from current point to a position, update current point
  - *Point*: cause a dot to appear at a position, update current point
  - *MovePoly*: move to a position, but in Polygon Mode
  - *DrawPoly*: like Draw, but an edge of a closed polygon
  - *ClosePoly*: close the current polygon, flushing it
Example

Current Position: [0, 0, 0]
Example

Current Position: $[0, 0, 0]$

Push matrix A
Example

Current Position: [0, 0, 0]

Push matrix A
Push matrix B
Example

Current Position: [0, 0, 0]

Push matrix A
Push matrix B
Push matrix C
Example

Current Position: ABC \([x_1, y_1, z_1]\)

Push matrix A
Push matrix B
Push matrix C
Move \(x_1, y_1, z_1\)
Example

Current Position: ABC \([x_2, y_2, z_2]\)

Push matrix A
Push matrix B
Push matrix C
Move \(x_1, y_1, z_1\)
Draw \(x_2, y_2, z_2\)
Example

Current Position: ABC \([x_3, y_3, z_3]\)

- Push matrix A
- Push matrix B
- Push matrix C
- Move \(x_1, y_1, z_1\)
- Draw \(x_2, y_2, z_2\)
- Draw \(x_3, y_3, z_3\)
Rasterization

- A 3D scene is a collection of geometric objects (triangles, lines)
- Need to turn this 3D data into a 2D image: *rasterization*
Historical Aside

Harris Ryan
First Chair of Stanford EE Department
Chair 1905-1931

Responsible for introducing CRTs as a research instrument in the U.S., brought from Germany.
Speed of a CRT Beam

- $1024 \times 768 = 786,432$ pixels
- $@60Hz = 47,185,920$ pixels/second
- Pixel is $21.1\text{ns}$!
  - Doesn't include time for beam to scan back to start/top (that's when an Atari game ran its game logic)
Rasterization

- Need to have pixel ready when beam scans it
- IRIS 3000 workstation (1985) used a Motorola 68010 CPU, running at 10MHz
  - Pixels scanned at almost 50MHz!
Graphics And Cases

• Use a projection transform to convert X,Y of object into X,Y of view plane

• Need to discretize floating point coordinates into discrete point indices
Graphics And Cases

- It's often useful to walk on the long axis: make sure you cover the entire range (e.g., walk on X, compute Y, or walk on Y, compute X)
Example: Wu (1991!)

60 lines of code!

- Considers two cases: steep (dy > dx), not steep (dx > dy)
- If line has a negative dx, reverses points
- Anti-aliases lines
Discussion of Marc Hannah's Contributions