Mathematics in Computer Graphics and Games

Prof Emmanuel Agu

Computer Science Dept.
Worcester Polytechnic Institute (WPI)
About Me

- Professor in WPI Computer Science Dept
- Grad school at Umass Amherst (MS, PhD)
  - Research in Computer graphics for 20 years
  - Teaching computer graphics for 14 years
What is Computer Graphics (CG)?

- Computer graphics: algorithms, mathematics, programs ..... that computer uses to generate PRETTY PICTURES
- E.g Techniques to draw a line, polygon, cube

Computer-Generated!
Not a picture!
Uses of Computer Graphics

- **Entertainment**: games

*Courtesy: Final Fantasy XIV*  
*Courtesy: Super Mario Galaxy 2*
Uses of Computer Graphics

- movies, TV (special effects, animated characters)

Note: Games and Movie industries Are two biggest hirers of computer Graphics professionals!!
Uses of Computer Graphics

- Displaying Mathematical Functions
  - E.g., Mathematica®
2 Main Career Paths in Computer Graphics

1. **Artist:** Designs characters
   - No math skills required!!

2. **Programmer:** Writes programs to Make characters move, talk, etc
   - Lots of math, programming skills required!!

Your students probably Follow programmer path
Some High School Math Used in CG

- Geometry
- Linear algebra: Matrices, vectors
- Trigonometry
- Complex numbers
- Boolean logic
- Probability
Fractals

- Mathematical expressions to generate pretty pictures
- Evaluate math functions to create drawings
  - Evaluated function approached infinity -> converge to image
  - i.e. \( f(1), f(2), f(3) \ldots \) \( F(\infty) \)
- Fractal image exhibits self-similarity: See similar sub-images within image as we zoom in
Sierpinski Gasket: Popular Fractal

Start with initial triangle with corners

1. Pick initial point \( p = (x, y) \) at random inside triangle
2. Randomly select 1 of 3 vertices
3. Find \( q \), halfway between \( p \) and randomly selected vertex
4. Draw dot at \( q \)
5. Replace \( p \) with \( q \)
6. Return to step 2
Example: Fractal Terrain

Terrain designed with only fractals
Example: Fractal Art

Courtesy: Internet Fractal Art Contest
Example: Mandelbrot Set
Mandelbrot Set

- Function of interest:
  \[ f(z) = (s)^2 + c \]

- Pick constants \( s \) and \( c \)

- **Orbit:** sequence of values (i.e. \( d_1, d_2, d_3, d_4, \text{ etc} \)):
  
  \[
  \begin{align*}
  d_1 &= (s)^2 + c \\
  d_2 &= ((s)^2 + c)^2 + c \\
  d_3 &= (((s)^2 + c)^2 + c)^2 + c \\
  d_4 &= ((((s)^2 + c)^2 + c)^2 + c)^2 + c
  \end{align*}
  \]

- Question: does the orbit converge to a value?
Mandelbrot Set

- Examples orbits:
  - \( s = 0, \ c = -1, \) orbit = 0,-1,0,-1,0,-1,0,-1,….. \textit{finite}
  - \( s = 0, \ c = 1, \) orbit = 0,1,2,5,26,677…… \textit{explodes}
- Orbit depends on \( s \) and \( c \)
- Basic question:
  - For given \( s \) and \( c \),
    - does function stay finite? (within Mandelbrot set)
    - explode to infinity? (outside Mandelbrot set)
- Definition: if \(|d| < 2\), orbit is finite else infinite
Mandelbrot Set

- Mandelbrot set: use complex numbers for $c$ and $s$
- Set $s = 0$, $c$ as a complex number
- E.g: $s = 0$, $c = 0.2 + 0.5i$
- Definition: Mandelbrot set includes all finite orbit $c$
- Mandelbrot set program:
  - Choose $s$ and $c$,
  - program calculates $d_1$, $d_2$, $d_3$, $d_4$ and tests if they are finite
  - Choose colors

Values of $c$ in mandelbrot set
Other Fractal Examples

Gingerbread Man

The Fern
Geometric Representations: 3D Shapes

- Generated using closed form geometric equations
- Example: Sphere

\[ x^2 + y^2 + z^2 = R^2. \]

- **Problem:** A bit restrictive to design real world scenes made of spheres, cones, etc.
Geometric Representations: Meshes

- Collection of polygons, or faces, that form “skin” of object
- More flexible, represents complex surfaces better
- Mesh? List of \((x,y,z)\) points + connectivity
- Digitize real objects: very fine mesh

Each face of mesh is a polygon

Digitized mesh of statue of Lucy: 28 million faces
Affine Transformations

- Translation
- Scaling
- Rotation
- Shear
Affine Transforms: General Approach

- We can transform (translation, scaling, rotation, shearing, etc) object by applying matrix multiplications to object vertices

\[
\begin{pmatrix}
    P'_x \\
    P'_y \\
    P'_z \\
    1
\end{pmatrix} =
\begin{pmatrix}
    m_{11} & m_{12} & m_{13} & m_{14} \\
    m_{21} & m_{22} & m_{23} & m_{24} \\
    m_{31} & m_{32} & m_{33} & m_{34} \\
    0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
    P_x \\
    P_y \\
    P_z \\
    1
\end{pmatrix}
\]

- Note: point \((x,y,z)\) needs to be represented as \((x,y,z,1)\), also called **Homogeneous coordinates**

Transformed Vertex → Transform Matrix → Original Vertex
3D Translation using Matrices

- Move each object vertex by same distance \( \mathbf{d} = (d_x, d_y, d_z) \)
- **Example**: If we translate a point (2,2,2) by displacement (2,4,6), new location of point is (4,6,8)

Translate object

\[
\begin{pmatrix}
4 \\
6 \\
8 \\
1
\end{pmatrix}
= 
\begin{pmatrix}
1 & 0 & 0 & 2 \\
0 & 1 & 0 & 4 \\
0 & 0 & 1 & 6 \\
0 & 0 & 0 & 1
\end{pmatrix}
\times
\begin{pmatrix}
2 \\
2 \\
2 \\
1
\end{pmatrix}
\]

- Translate x: \( 2 + 2 = 4 \)
- Translate y: \( 2 + 4 = 6 \)
- Translate z: \( 2 + 6 = 4 \)

**General form**

\[
\begin{pmatrix}
x' \\
y' \\
z'
\end{pmatrix}
= 
\begin{pmatrix}
1 & 0 & 0 & d_x \\
0 & 1 & 0 & d_y \\
0 & 0 & 1 & d_z \\
0 & 0 & 0 & 1
\end{pmatrix}
\cdot
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
\]
Scaling Transform

- Expand or contract along each axis (fixed point of origin)

- **Example:** If we scale a point (2, 4, 6) by scaling factor (0.5, 0.5, 0.5)
  Scaled point position = (1, 2, 3)

  - Scaled x: 2 x 0.5 = 1
  - Scaled y: 4 x 0.5 = 2
  - Scaled z: 6 x 0.5 = 3

\[
\begin{bmatrix}
1 \\ 2 \\ 3 \\ 1
\end{bmatrix} = \begin{bmatrix}
0.5 & 0 & 0 & 0 \\
0 & 0.5 & 0 & 0 \\
0 & 0 & 0.5 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \times \begin{bmatrix}
2 \\ 4 \\ 6 \\ 1
\end{bmatrix}
\]

Scale Matrix for Scale(0.5, 0.5, 0.5)

**General Form**

\[
\begin{bmatrix}
x' \\ y' \\ z' \\ 1
\end{bmatrix} = \begin{bmatrix}
S_x & 0 & 0 & 0 \\
0 & S_y & 0 & 0 \\
0 & 0 & S_z & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \times \begin{bmatrix}
x \\ y \\ z \\ 1
\end{bmatrix}
\]
Why Matrices?

- Sequence of transform matrices can be pre-multiplied
- One final resulting matrix applied (efficient!)
- E.g. transform 1 \( \times \) transform 2 ....

\[
\begin{pmatrix}
Q_x \\
Q_y \\
Q_z \\
1
\end{pmatrix} =
\begin{pmatrix}
m_{11} & m_{12} & m_{13} & m_{14} \\
m_{21} & m_{22} & m_{23} & m_{24} \\
m_{31} & m_{32} & m_{33} & m_{34} \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
m_{11} & m_{12} & m_{13} & m_{14} \\
m_{21} & m_{22} & m_{23} & m_{24} \\
m_{31} & m_{32} & m_{33} & m_{34} \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
P_x \\
P_y \\
P_z \\
1
\end{pmatrix}
\]

- Computer graphics card has fast 4x4 matrix multiplier!!!
Why do we need Shading?

- Sphere without lighting & shading:

- Sphere with shading:
  - Has **visual cues** for humans (shape, light position, viewer position, surface orientation, material properties, etc)
What Causes Shading?

- Shading caused by different angles with light, camera at different points
Calculating Shade

- Based on Lambert’s Law: $D = I \times k_D \cos(\theta)$
  - Calculate shade based on angle $\theta$

- Represent light direction, surface orientation as vectors
- Calculate $\theta$? Angle between 2 vectors
Shading: Diffuse Light Example

Different parts of each object receive different amounts of light.
References

- Angel and Shreiner, Interactive Computer Graphics (6th edition), Chapter 1