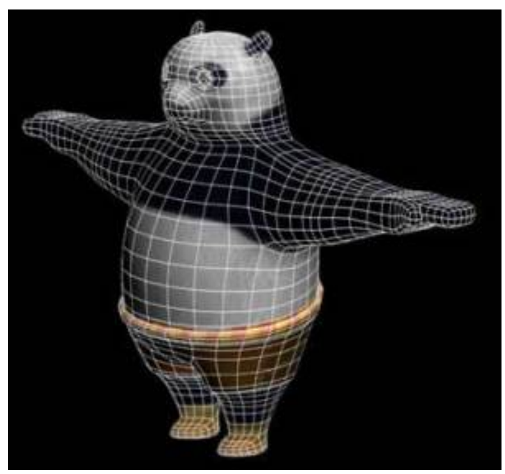
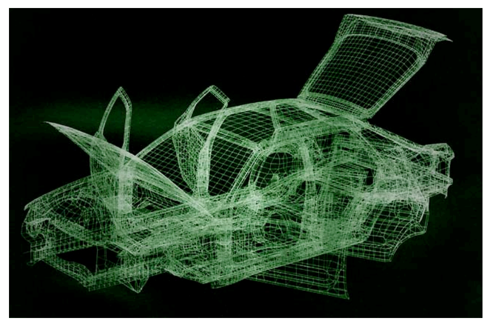
Geometric Modeling



Liz Marai

Two Basic CG Paradigms

- **Sample-based graphics (left):** discrete samples are used to describe visual information
  - pixels can be created by digitizing images, using a sample-based "painting" program, etc.
  - example programs: Adobe Photoshop™, GIMP™



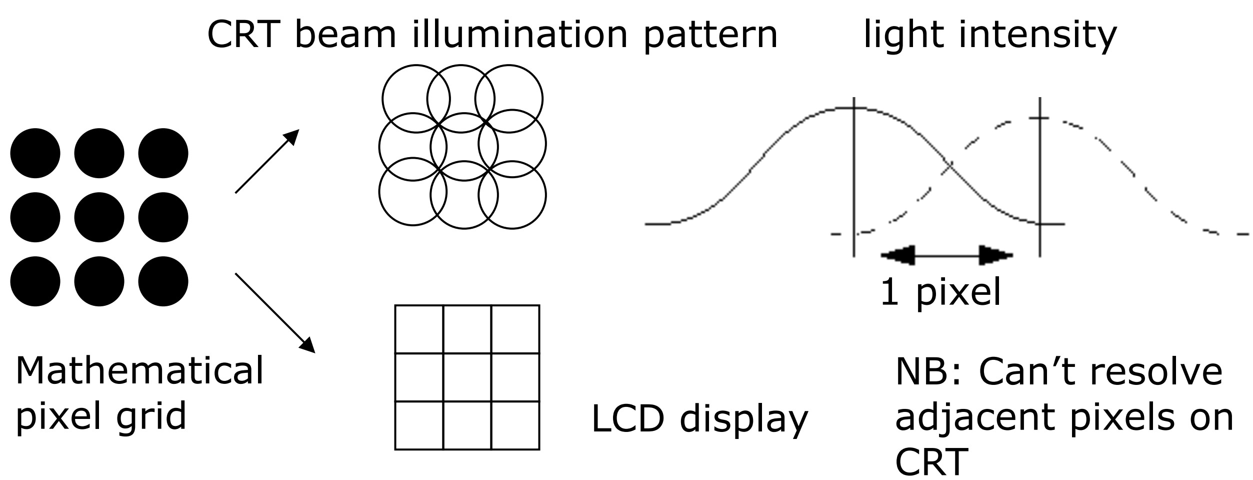
- **Geometry-based graphics (right):** geometrical model is created, along with various appearance attributes, and is then sampled for visualization (*rendering* a.k.a *image synthesis*);
  - also called scalable vector graphics or object-oriented graphics
  - example programs: Adobe Illustrator™, Autodesk's AutoCAD2008™, Autodesk's Maya™, Autodesk's 3D Studio Max™

Sampled-based Graphics

- Images are made up of grid of discrete pixels, for 2D "picture elements"

- **Pixels** are point locations with associated sample values, usually of light intensities/colors, transparency, and other control information

- When we sample an image, we sample the point location along the *continuous signal* and we *cannot* treat the pixels as little circles or squares



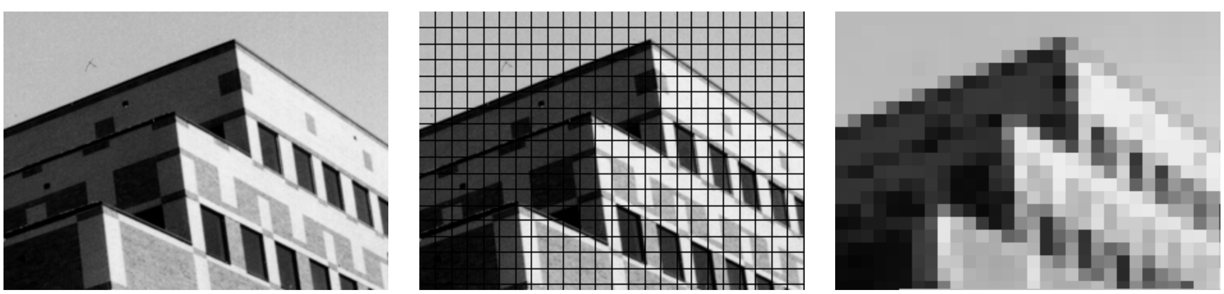
Sampling an Image

- Lets do some sampling of my alma mater's CS building

3D scene

- A color value is measured at every grid point and used to color corresponding grid square

0 = white, 5 = gray, 10 = black

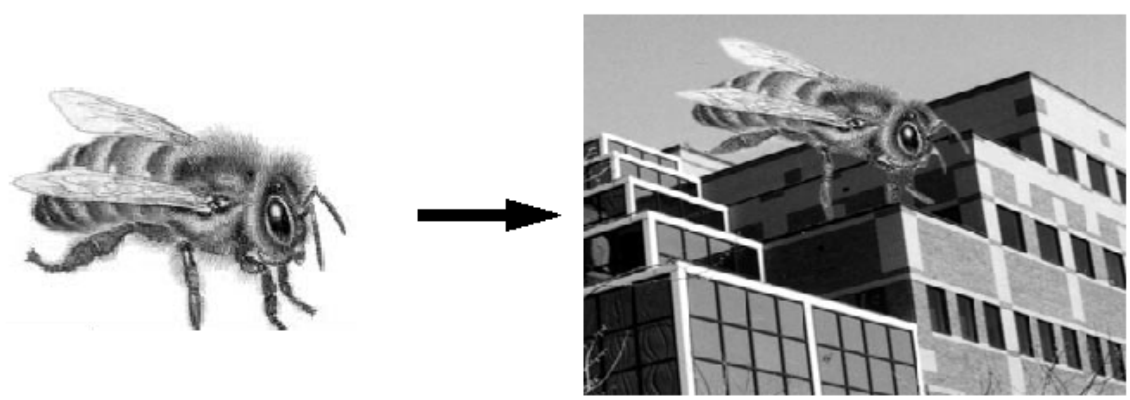


- Note: this poor sampling and image reconstruction method creates blocky image

What's the Advantage?

- Once image is defined in terms of colors at (*x*, *y*) locations on grid, can change image easily by altering location or color values
- E.g., if we reverse our mapping above and make 10 = white and 0 = black, the image would look like this:

- Pixel information from one image can be copied and pasted into another, replacing or combining with previously stored pixels



What's the Disadvantage?

- WYSIAYG (What You See Is All You Get): No additional information
  - no depth information
  - can't examine scene from different point of view
  - at most can play with the individual pixels or groups of pixels to change colors, enhance contrast, find edges, etc.

- CS1566 emphasizes geometry-based graphics

Capture of Hair Geometry from Multiple Images, Siggraph 2004

Geometry-Based Graphics

- **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, primitives for short

- **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

- 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

What is Geometric Modeling?

What is a model?

- An abstraction to capture salient features (data, behavior) of thing/phenomenon being modeled
  - data includes geometry, appearance attributes…
  - note similarity to OOP notions

- Spatial: some geometry inherent (Cartesian coords)
  - physical (e.g., actual object such as a pump)
  - non-physical (e.g., mathematical function, weather data)

- Non-Spatial: no inherent geometry, but mapped to geometry for visualization
  - organizational (e.g., company org. chart)
  - quantitative (e.g., graph of stock market)

- Nowadays, modeling is coping with complexity (hot CG research)

- Our Focus: modeling/viewing simple objects

Modeling vs. Rendering

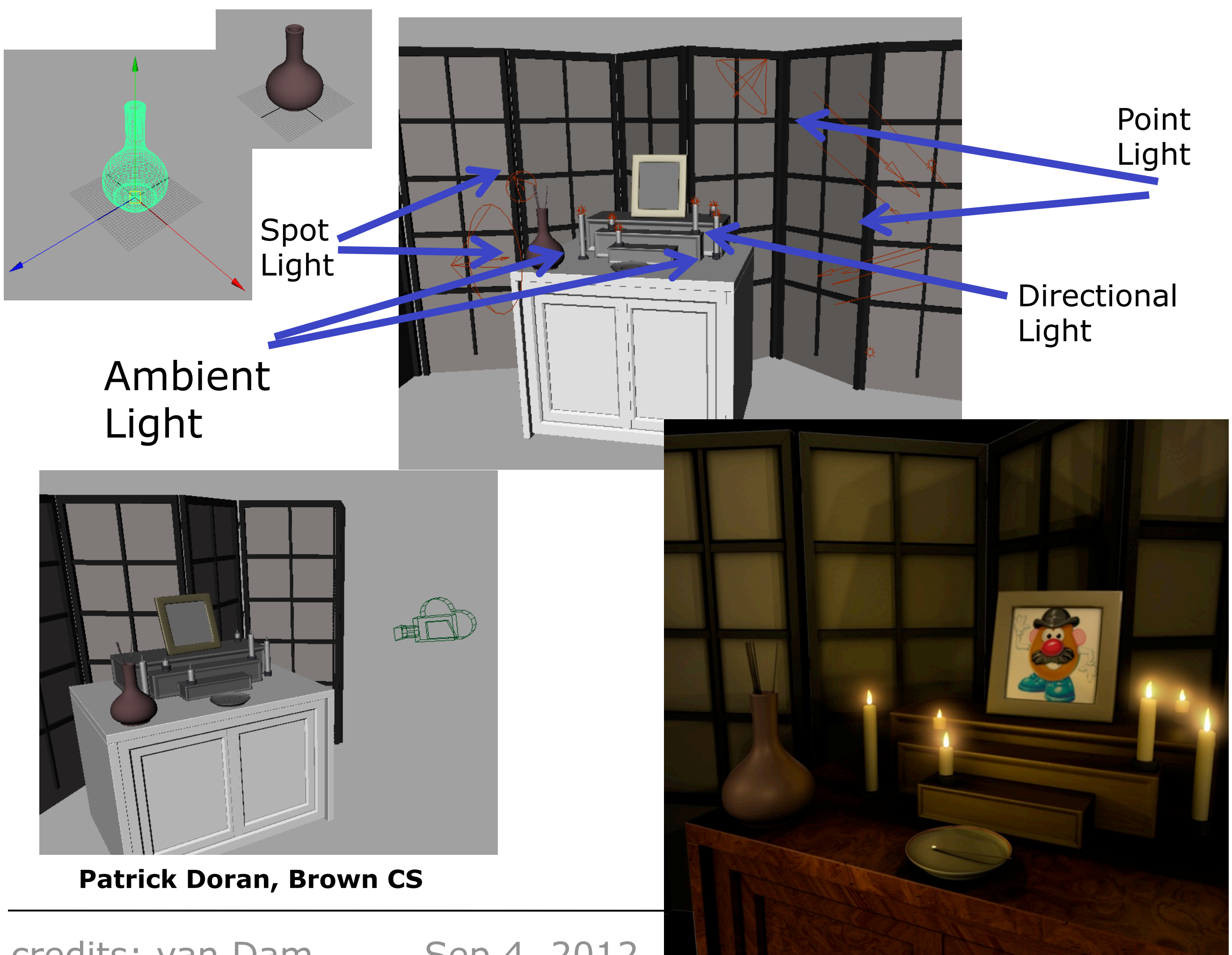
Modeling:

- Create models; Apply materials to models
- Place models around scene
- Place lights in scene
- Place the camera

Rendering: Take "picture" with camera

Both can be done with commercial software:

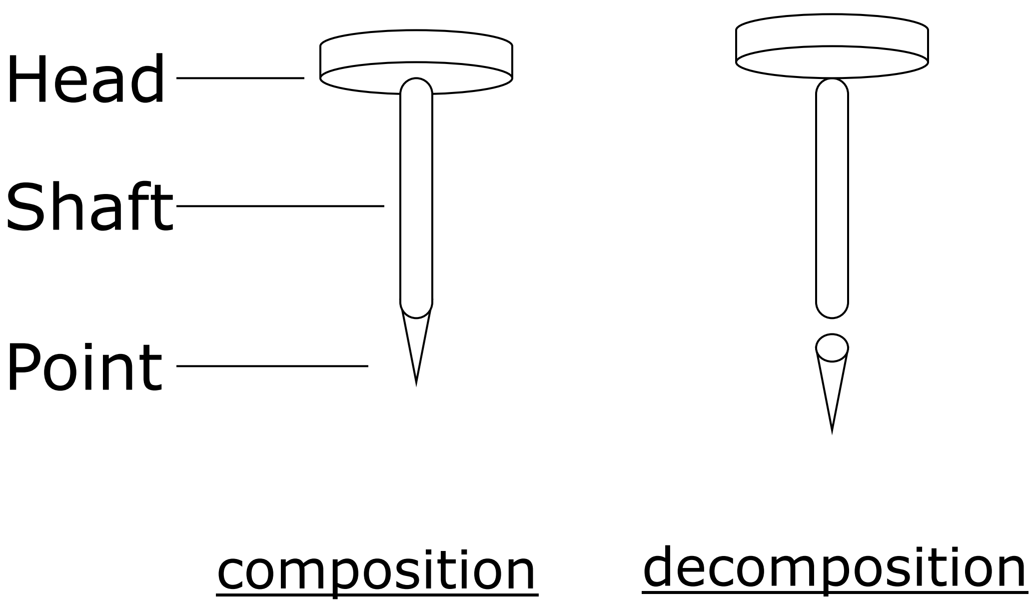
Autodesk Maya™ ,3D Studio Max™, Blender™, etc.



Patrick Doran, Brown CS

Decomposition of a Geometric Model

- Divide and Conquer
- Hierarchy of geometrical components
- Reduction to primitives (e.g., spheres, cubes, etc.)
- Simple vs. not-so-simple elements (nail vs. screw)



Hierarchical (Tree) Diagram of Nail

- Object to be modeled is (visually) analyzed, and then decomposed into collections of primitive shapes.

- Tree diagram provides visual method of expressing "composed of" relationships of model

Nail — *root node*

Head (cylinder) Body

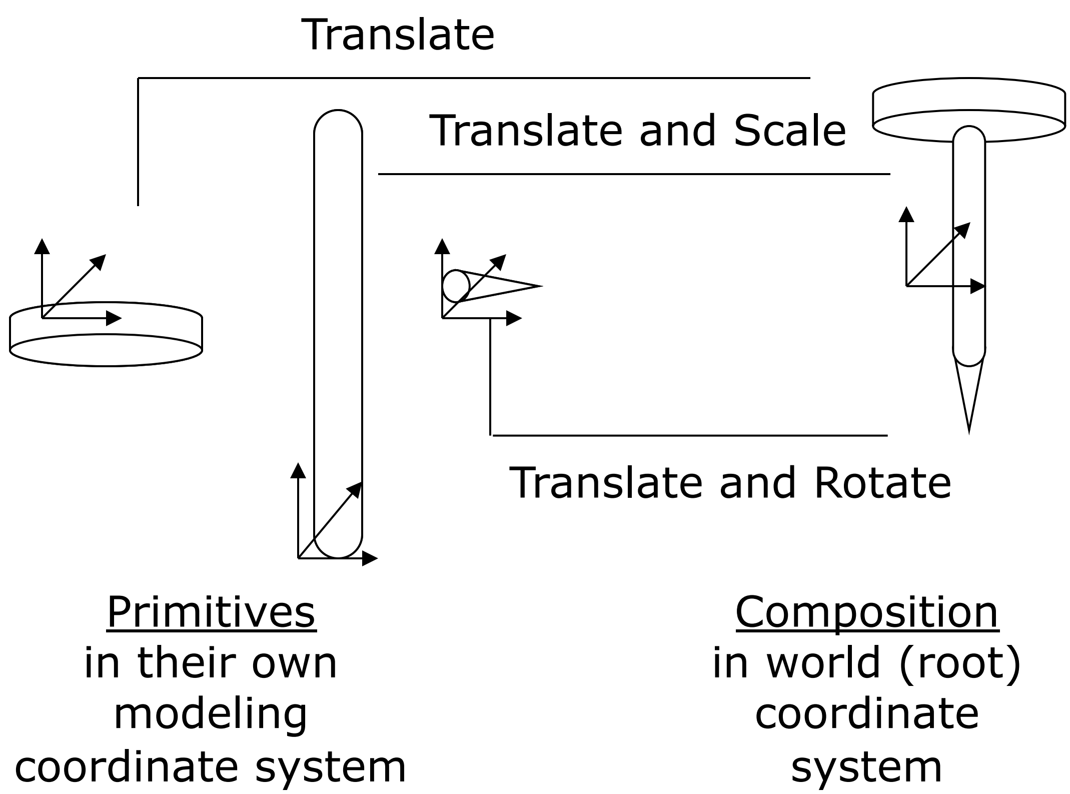
Shaft (cylinder) Point (cone) — *leaf nodes*

tree diagram

- Such diagrams are part of 3D program interfaces (e.g., 3D Studio MAX, Maya)

- As data structure to be rendered, it is called a **scenegraph**

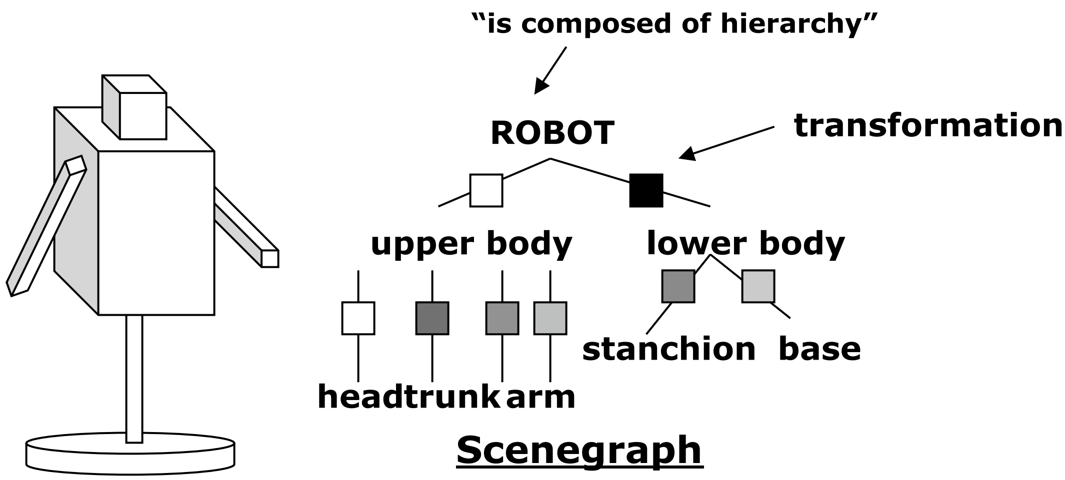
Composition of a Geometric Model



- Primitives created in decomposition process must be assembled to create final object. Done with “affine transformations,” T, R, S (as in above example).
- Other composition operators exist (e.g., Constructive Solid Geometry – CSG -- uses Boolean operators).

How Are Geometric Transformations (T,R,S) Used in Computer Graphics?

- Object construction using assemblies/hierarchy of parts à la Sketchpad's masters and instances; leaves of scenegraph contain primitives



Scenegraph

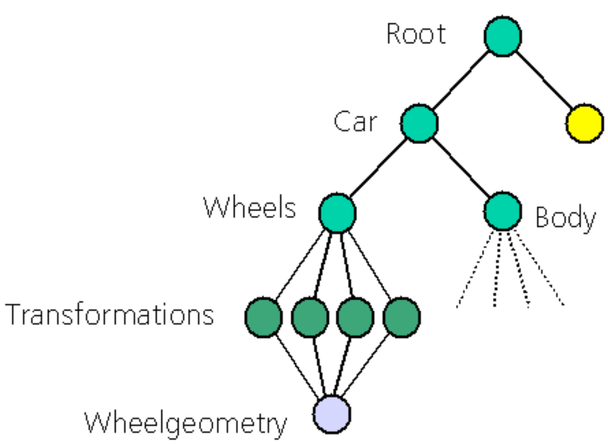
- Aid to realism
  - objects, camera use realistic motion

- Synthetic camera/viewing

- Note: Helpful applets
  - Experiment with these concepts on the cs1566 webpage: *Applets->Linear Algebra* and *Applets->Scenegraphs*

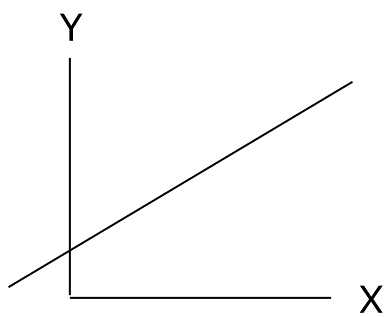
Topics of Upcoming Lectures

- We manipulate primitive shapes with **geometric transformations** (translation, rotation, scale). These transformations are essential for **model organization,** process of composing complex objects from simpler components.
- Hierarchical models and same geometric transformations are also essential for **animation**
- Once object's geometry is established, must be viewed on screen: map from 3D to 2D for **viewing** and from 2D to 3D for 2D input devices (e.g., the mouse or pen/stylus)
- While mapping from 3D to 2D, object (surface) material properties and lighting effects are used in **rendering** one's constructions. This rendering process is also called *image synthesis*

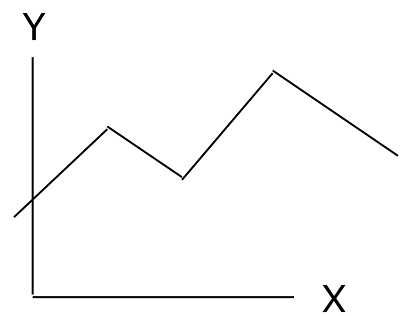


2D Primitives

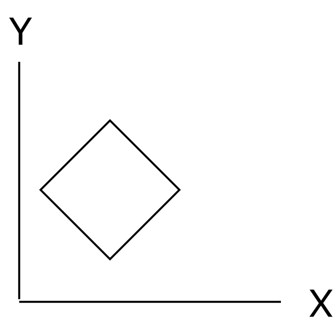
Line



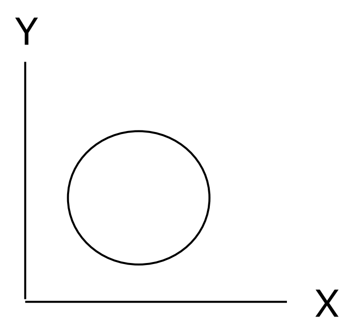
Polyline



Polygon

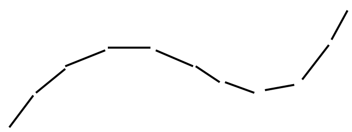


Circle



Curves

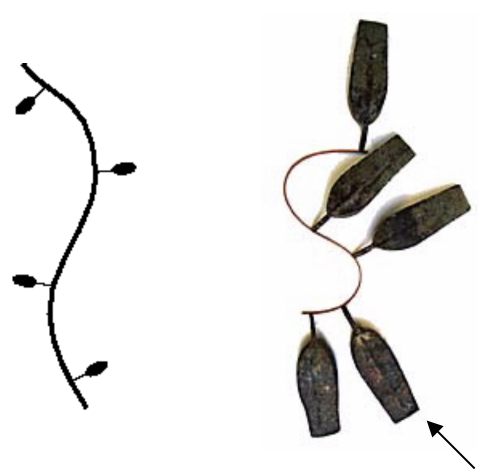
- Piecewise linear approximation



- Splines: higher-order polynomials
  - piecewise curvilinear approximation

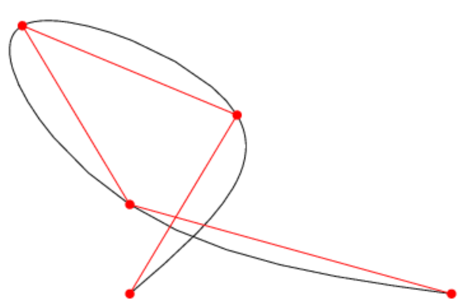
French Curves

Draftman's Spline



("duck")

Mathematical Splines



Natural Cubic Spline:

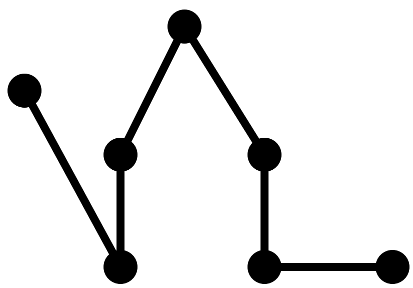
$$x(t) = a_x t^3 + b_x t^2 + c_x t + x_0$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + y_0$$

2D Object Definition (1/3)

Lines and Polylines

- *Polylines*: lines drawn between ordered points

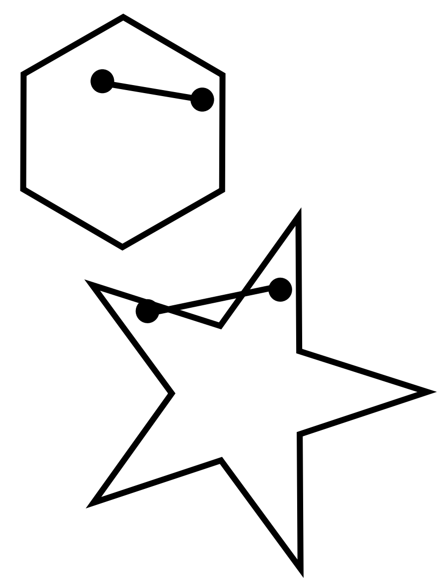


- Same first and last point make *closed polyline* or *polygon*
- If it does not intersect itself, called *simple polygon*

Convex vs. Concave Polygons

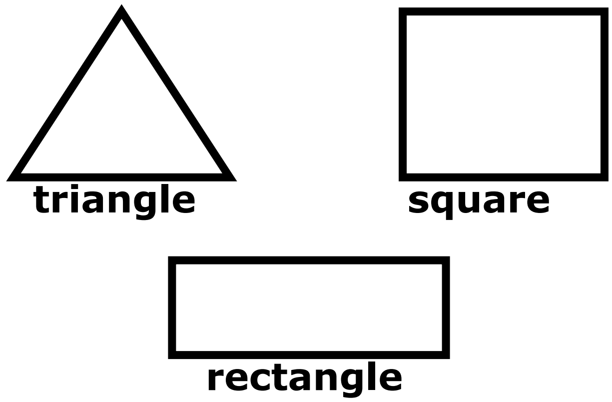
<u>Convex:</u> For every pair of points in the polygon, the line between them is fully contained in the polygon.

<u>Concave:</u> Not convex: some two points in the polygon are joined by a line not fully contained in the polygon.



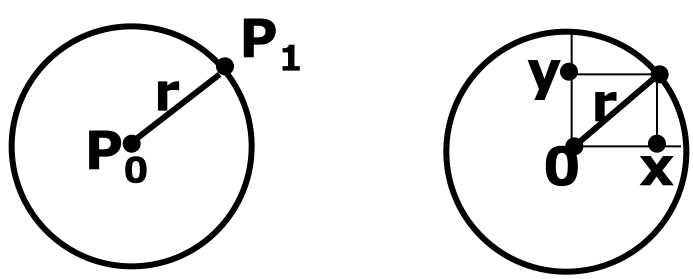
2D Object Definition (2/3)

Special polygons



Circles

- Consist of all points equidistant from one predetermined point (the center)
- (radius) $r = c$, where c is a constant

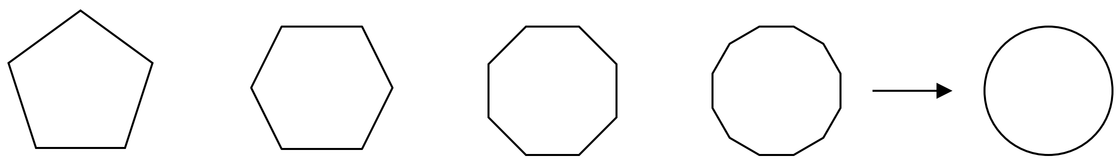


- On a Cartesian grid with center of circle at origin equation is $r^2 = x^2 + y^2$

2D Object Definition (3/3)

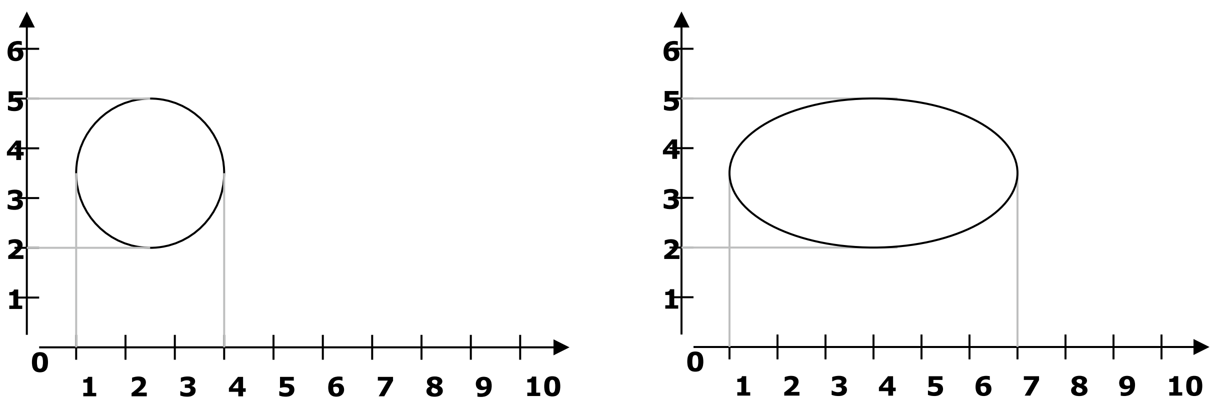
Circle as polygon

- A circle can be represented by a polygon with many sides (>15).



(Aligned) Ellipses

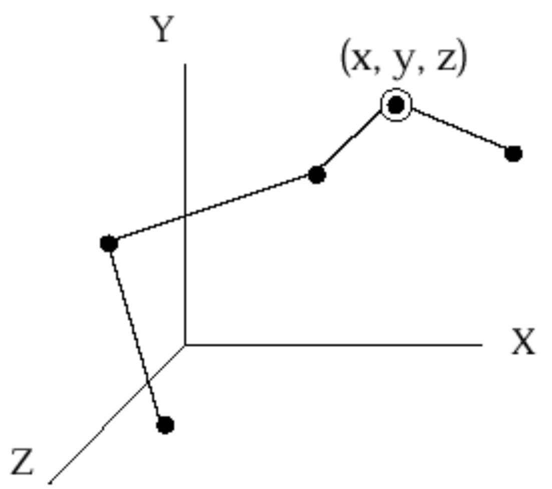
A circle scaled along the x or y axis



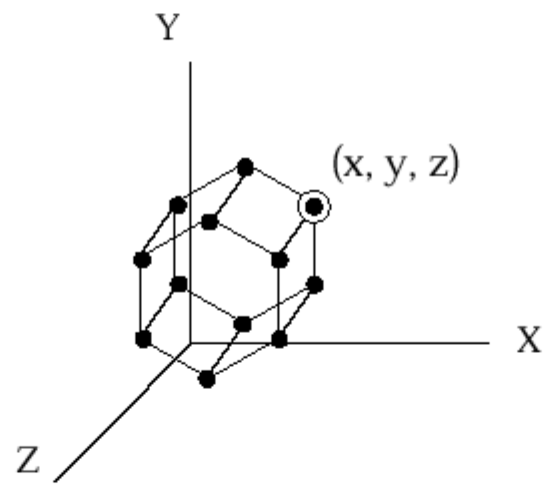
Example: height, on *y*-axis, remains 3, while length, on *x*-axis, changes from 3 to 6

Example 3D Primitives

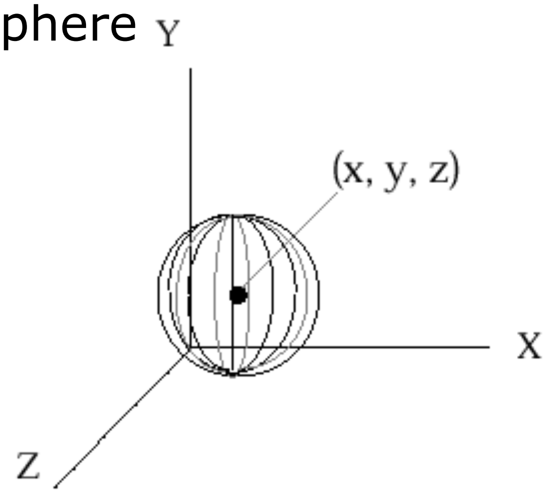
Polyline



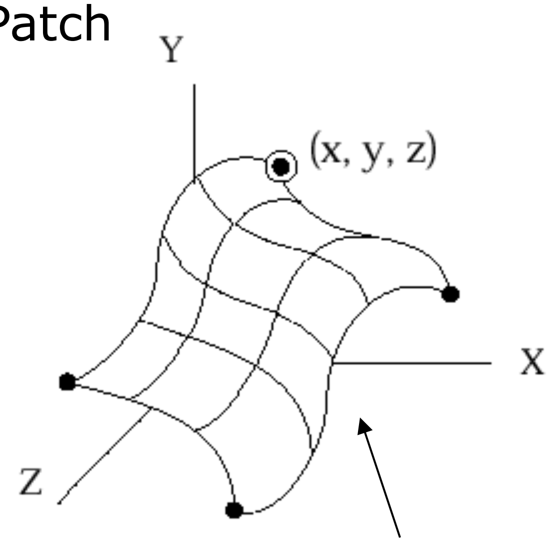
Polyhedron



Sphere



Patch

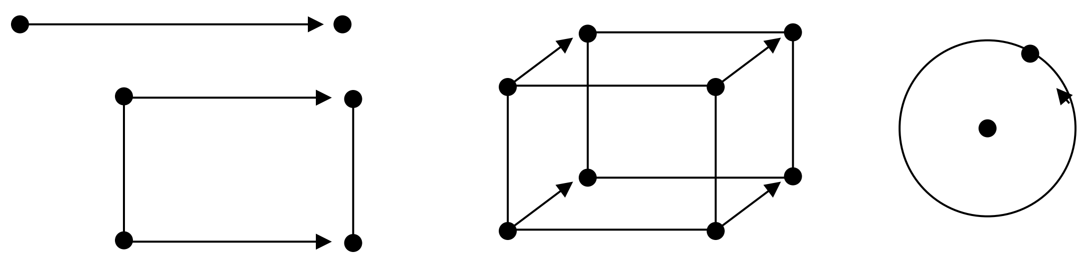


(spline boundary curve)

2D to 3D Object Definition

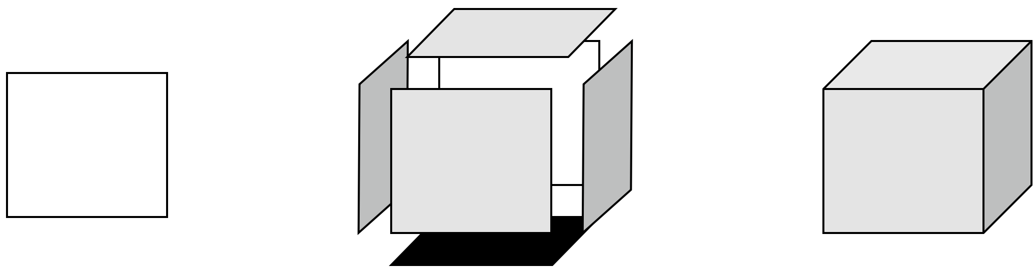
Vertices in motion *("Generative object description")*

- Line is drawn by tracing path of a point as it moves (one dimensional entity)
- Square drawn by tracing vertices of a line as it moves perpendicularly to itself (two dimensional entity)

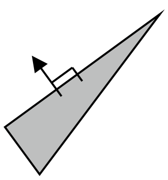


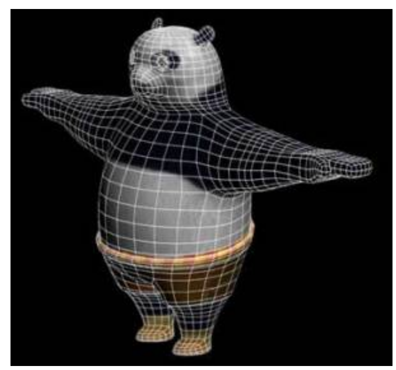
- Cube drawn by tracing paths of vertices of a square as it moves perpendicularly to itself (three-dimensional entity)
- Circle drawn by swinging a point at a fixed length around a center point

Building 3D Objects



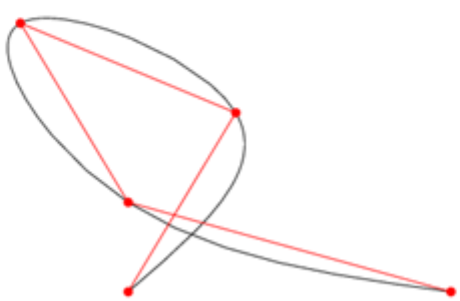
- Triangles and quads form tri-meshes and quad-meshes



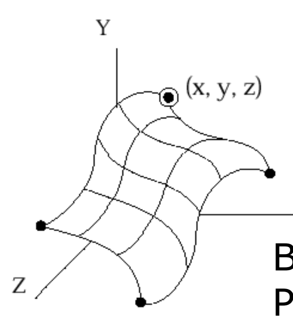



- Often made of parametric polynomials, called splines

Curves



Patches



Boundaries are
Polynomial curves
In 3D

Useful Concepts from Linear Algebra

- 3D Coordinate geometry
- Vectors in 2 space and 3 space
- Dot product and cross product – definitions and uses
- Vector and matrix notation and algebra
- Multiplicative associativity
  - E.g. A(BC) = (AB)C
- Matrix transpose and inverse – definition, use, and calculation
- *Homogeneous coordinates (x,y,z,**w**)*

We will need to understand these concepts

- See linear algebra recitation (notes)

Short Linear Algebra Digression: Vector and Matrix Notation, A non-Geometric Example (1/2)

Let's Go Shopping

- Need 6 apples, 5 cans of soup, 1 box of tissues, and 2 bags of chips

- Stores A, B, and C (Giant Eagle, Whole Foods, and 7-Eleven) have following unit prices respectively

	1 apple	**1 can of soup**	**1 box of tissues**	**1 bag of chips**
Giant Eagle	**\$0.20**	**\$0.93**	**\$0.64**	**\$1.20**
Whole Foods	**\$0.65**	**\$0.95**	**\$0.75**	**\$1.40**
7-Eleven	**\$0.95**	**\$1.10**	**\$0.90**	**\$3.50**

A Non-Geometric Example (2/2)

- Shorthand representation of the situation (assuming we can remember order of items and corresponding prices):
- Column vector for quantities, q: $\begin{bmatrix} 6 \\ 5 \\ 1 \\ 2 \end{bmatrix}$
- Row-vectors of corresponding prices at stores:

store A (Giant Eagle)	**[0.20 0.93 0.64 1.20]**
store B (Whole Foods)	**[0.65 0.95 0.75 1.40]**
store C (7-Eleven)	**[0.95 1.10 0.90 3.50]**

What do I pay?

Let's calculate for each of the three stores.

- **Store A:**

$$\text{totalCost}_A = \sum_{i=1}^{4} PA_i q_i$$

$= (0.20 \bullet 6) + (0.93. \bullet 5) + (0.64 \bullet 1) + (1.20 \bullet 2)$
$= (1.2 + 4.65 + 0.64 + 2.40)$
$= 8.89$

- **Store B:**

$$\text{totalCost}_B = \sum_{i=1}^{4} PB_i q_i = 3.9 + 4.75 + 0.75 + 2.8 = 12.2$$

- **Store C:**

$$\text{totalCost}_C = \sum_{i=1}^{4} PC_i q_i = 5.7 + 5.5 + 0.9 + 7 = 19.1$$

Using Matrix Notation

- Can express these sums more compactly:

$$P(All) = \begin{bmatrix} totalCost_A \\ totalCost_B \\ totalCost_C \end{bmatrix} = \begin{bmatrix} 0.20 & 0.93 & 0.64 & 1.20 \\ 0.65 & 0.95 & 0.75 & 1.40 \\ 0.95 & 1.10 & 0.90 & 3.50 \end{bmatrix} \begin{bmatrix} 6 \\ 5 \\ 1 \\ 2 \end{bmatrix}$$

- Determine totalCost vector by row-column multiplication
  - dot product is the sum of the pairwise multiplications
    - Apply this operation to rows of prices and column of quantities

$$\begin{bmatrix} a & b & c & d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = ax + by + cz + dw$$

Summary

- 2 Paradigms:
  - sample-based CG
  - geometric model-based CG
- Sample-based CG
  - pixels; sampling an image
  - image processing
- Geometric model-based CG
  - geometric model
  - modeling vs rendering
  - hierarchical decomposition
  - geometric transformations
  - 2D primitives
  - 2D curves
  - 2D objects
    - lines, polylines
    - convex/concave polygons
    - circles as polygons
  - 3D primitives
  - 3D objects
    - motion-based generation
    - tri-meshes, quad-meshes
    - curved surface/patch-based
- Vector and matrix notation
  - Using matrix notation