3: Geometric Detail, Symbols

Schematic development of the massing model, adding and trimming openings, defining and inserting blocks, etc.
Building Details: openings

Layers
• Define “trim” layers to simplify control.

Use booleans to
• “subtract” punched window openings
• “remove” front & back door-way notches
• “add” decks and flooring where needed
Building Details: trim

Building door & window trim #1
- Trace edges, snapping to opening vertexes
- Offset inward.
- Extrude (&cap) inner and outer edges.

Building door & window trim #2
- Trace edges to provide “rail” for sweep
- Draw cross-section at rail-head
- Sweep cross-section on rail
Building Details: glazing

Fixed glazing #1

• Define 1 plane of glass within wall for all windows in that wall.
Symbols: Copies & Instances

Copies (copy & paste, “copy”, etc.)
- Repeated geometry = repeated data
- Manual change propagation

Instances (blocks, symbols, etc.)
- Name subdrawings
- Repeated geometry = repeated name references
- Automatic change propagation
Symbols: Attributes

Original geometry attributes
• Location relative to global origin
• Layer, color, linestyle ("object attributes")

Instance attributes
• Each instance *is an object* in it’s own right

Predicting appearance ... can be a challenge
• Interaction of original attributes & instance attributes
Symbols: Instance Attributes

- **Original objects**: by layer, by parent, custom (green)
  - **Default**: "brown"

- **Instance color**: layer = red
  - by layer
  - by parent
  - custom (teal)
Blocks vs. Files

- *Existing geometry* may be made into a block and inserted (insert).
- An *existing model* (.3dm) file may be inserted, creating a block and instance (insertFile).
- Internally defined blocks may be *redefined* (block), and are saved with the current file.
- External files may optionally be “linked” and automatically “re-inserted” at times, facilitating change propagation.
Some cautions

• File inserts $\rightarrow$ “origin” becomes insertion pt.
• “block” command requires insertion pt.
  – Pick carefully (some are more useful than others)
  – Pay attention if re-defining or things will move!
• Redefining the block does not change instance references to the block.
• Blocks defined in the inserted file become blocks in the receiving file (libraries?).
Hidden Surface Strategies

Thinking in MODEL SPACE

• We see the *surfaces that face* us
  – Discard surfaces facing away (*culling*)
  – Fails with one-sided sheets
  – Fails in a concave world

• We see only the *surfaces that are closest* to us.
  – Relies on the “destructive write” computer displays
  – Sort surfaces by depth in scene (*depth sorting*)
  – There is a time-cost when sorting complex scenes
  – Fails when surfaces intersect or overlap

• Other: scan-line, etc.
Hidden Surface Strategies

Thinking in IMAGE SPACE – “Z-buffering”

• Answer the “closest surface” question on a per pixel basis?
• The graphics pipeline “paints” polygons to the screen one at a time as raster data.
• In eye space the “z-coordinate” is depth into the scene.
• Knowing the depth at the corners allows us to interpolate depth at every pixel.
• Storing the depth at each pixel allows us to decide what is visible.
Hidden Surfaces: Z-buffering

- eye space
- display screen
- 3D Model
- screen image
- row of pixels
- (screen.x, screen.y, eye.depth) at each vertex
What about curved surfaces?

• Not all “surfaces” are \textit{planar} surfaces!
• \textit{Approximate} curved forms with many polys?
  – Slows all operations
  – Turns out to be difficult due to human vision
• \textit{Render} tessellated surfaces to \textit{look} smooth?
  – Called “smoothing”
    • Many versions, including “Gouraud” and “Phong”
    • Implemented in OpenGL
Smoothing Tessellated Models
128 polygons vs. 11,680 polygons
3: Geometric Detail, Symbols

Questions?