**Using Graphics Hardware**

1. What does the hardware do?
2. How does it do that?
3. What can't hardware do?
4. What do we want to do?
5. How do we map what we want to do to what we can do?

---

**The abstractions**

- Primitives
- Vertex operations (TCL) (local) (phong)
- Rasterization / Fragment / Interpolation
- Per-Fragment Computation
- Texture Mapping
- Z-buffer (order "doesn't matter")
- Other per-fragment ops
- Frame Buffer
- Blend into frame buffer
- Re-use FB (as texture, or otherwise)
- FB reads / bus traffic issues

**Implementation**

- Queues and caches
- Fragment generation
- Impedance of bivalencies
- Parallelism

---

**Some tweaks:**

- Ordering of polygons
  - Back to front (transparency)
  - Front to back (culling)
- Early Z / hierarchical Z
- Dependent reads
What kinds of things to do?

What kinds of things can't you do directly with this?

- complex lighting
- very complex geometry / shape
- atmospheric effects

Kinds of Tricks

- Texture maps to do other things
  - toon shading

- Fancy kinds of mapping
  - Bump mapping, relief mapping, ....

- Deferred Shading / Frame buffer tricks

- Pre-Computed Lighting

- Fancy Computation of texture coordinates
  - Decal Projection / Slide Projection
  - Environment Mapping
Visual Effects

- Water
- Glass
- Hazy from heat
- Semi-invisibility
- Fire/explosion
- Flashlight

Performance

- Multi-Threading
- What methods do get used

Content

- Procedural Textures
- Destructible/Dynamic
- Recursive Views
- Interactive Properties
- Secondary Motion

AI

- Path Finding
- Reactiveness
- Believable Behavior

Animation

- Fluid/responsive characters
- Ragdoll Physics
- Authoring/Behavior
- Climable points

Sound (spatialized)