CSE 167:
Introduction to Computer Graphics
Lecture #15: Toon Shading

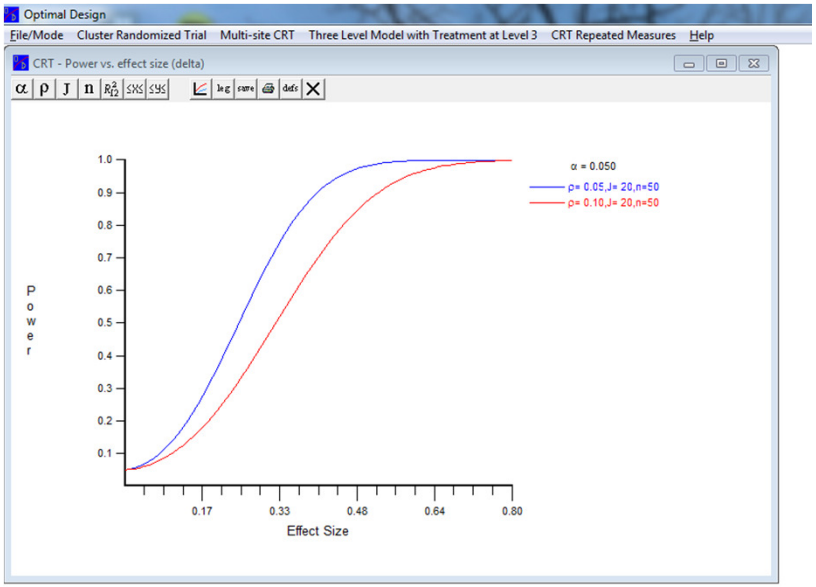
Jürgen P. Schulze, Ph.D.
University of California, San Diego
Spring Quarter 2016

Announcements

- Project 4 due this Friday
- Midterm Exam #2: Tue May 24th

Programming Opportunity

- POWER CALCULATIONS FOR RANDOMIZED SATURATION DESIGNS:
- A new paper called "Designing Experiments to Measure Spillover and Threshold Effects" provides code in Matlab that allows researchers to conduct power calculations for randomized experiments that are designed to measure spillover/contamination/threshold effects.
- What I would like to do is to hire a CS student who could help us put together a free-standing GUI similar to the Optimal Design package that would allow people to run the software and generate pictures of power curves and optimal design parameters, without having to have Matlab installed on their machine. My preference would be to have a desktop application that would be an open-source program that could be downloaded and run on a Windows or PC machine.
- Contact information: Craig McIntosh, ctmcintosh@ucsd.edu, 858 822 1125.



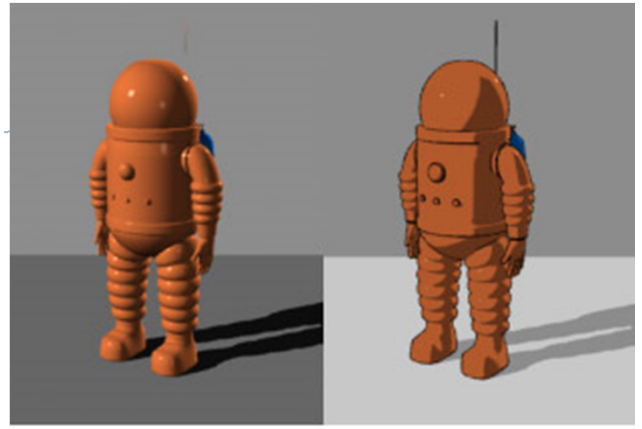
UCSD

Lecture Overview

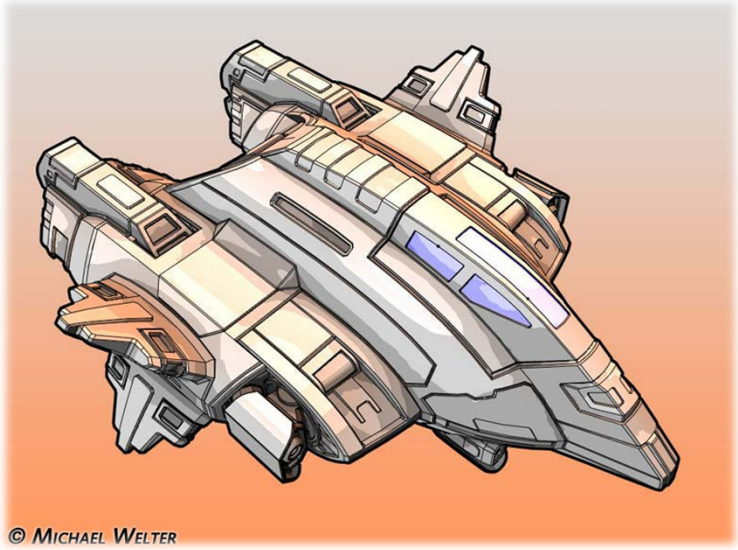
- Toon shading

Toon Shading

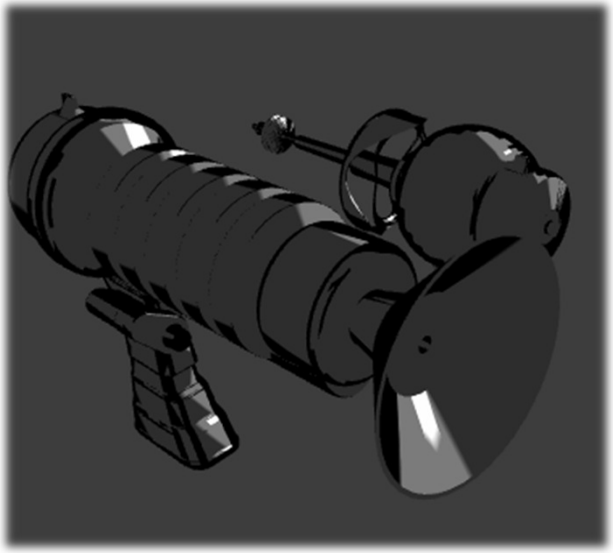
- A.k.a. Cel Shading ("Cel" is short for "celluloid" sheets, on which animation was hand-drawn)
- Gives any 3D model a cartoon-style look
- Emphasizes silhouettes
- Discrete steps for diffuse shading, highlights
- Non-photorealistic rendering method (NPR)
- Programmable shaders allow real-time performance



Source: Wikipedia



Off-line toon shader

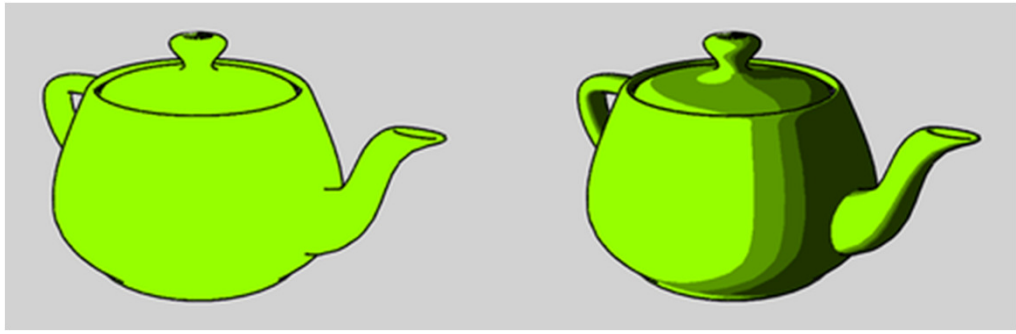


GLSL toon shader

UCSD

Approach

- Start with regular 3D model
- Apply two rendering tricks:
  - Silhouette edges
    - Emphasize pixels with normals perpendicular to viewing direction.
  - Discretized shading
    - Conventional (smooth) lighting values calculated for each pixel, then mapped to a small number of discrete shades.

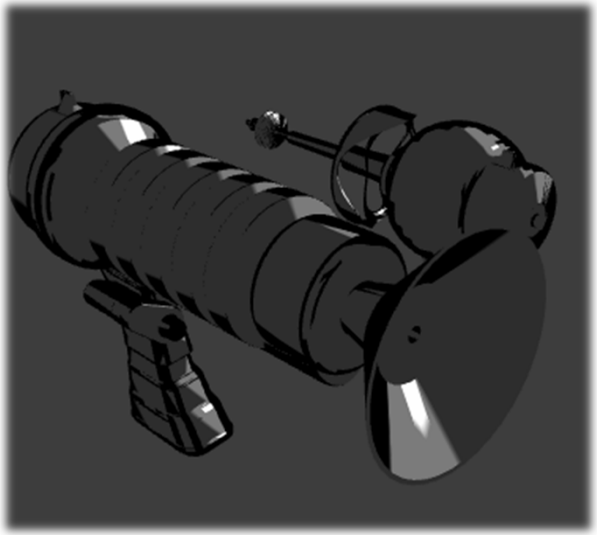


Source: Wikipedia

Silhouette Edges

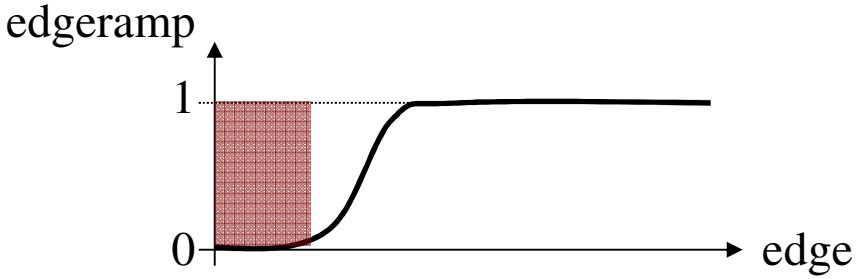
- Silhouette edge detection
  - Compute dot product of viewing direction $\mathbf{v}$ and normal $\mathbf{n}$

$$\text{edge} = \max(0, \mathbf{n} \cdot \mathbf{v})$$



  - Use cutoff value for edge:
    - if edge<0.01 draw black, else don't change pixel color
  - Use 1D texture to define edge ramp for smoother transition

```
uniform sample1D edgeramp;
e = texture1D(edgeramp,edge);
```



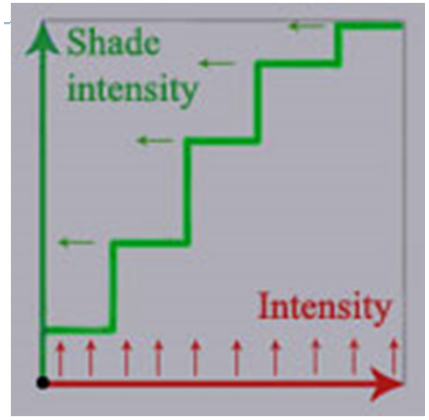
UCSD

Discretized Shading

- Compute diffuse and specular shading

$$\text{diffuse} = \mathbf{n} \cdot \mathbf{L} \quad \text{specular} = (\mathbf{n} \cdot \mathbf{h})^s$$

- Discretize shading
- Approaches:
  - If..then..else tree comparing values against thresholds
    - ```
      if (diffuse < A) diffuse = 0.0;
      else if (diffuse < B) diffuse = B;
      else if (diffuse < C) diffuse = C;
      else diffuse = D;
      ```
  - 1D textures to map diffuse and specular shading to colors
    - ```
      uniform sampler1D diffuseramp;
      uniform sampler1D specularramp;
      color = e * (texture1D(diffuse,diffuseramp) +
      texture1D(specular,specularramp));
      ```



UCSD

Toon Shading Demo

http://www.bonzaisoftware.com/npr.html