Advanced lighting in 2D graphics

Ferdinand Majerech
Supervised by: RNDr. Ladislav Mikeš

Univerzita Pavla Jozefa Šafárika v Košiciach
UPJŠ

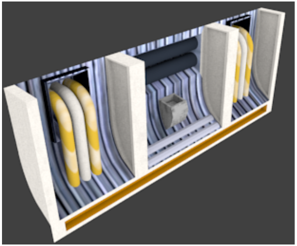
April 2, 2013

Why 2D

- Integrated, mobile GPUs
- Easy to code
- Easy to create art (even from 3D)
- Artists can "cheat", and are not HW-limited
- Don't always need 3D (RTS, isometric RPG)

Pre-rendered graphics

- 3D -> tool -> 2D -> postprocessing -> game
- 3D without game optimizations
- Can look photorealistic
- Low HW requirements



Common 2D lighting techniques

- Static lighting

Temple of Elemental Evil game by Troika Games

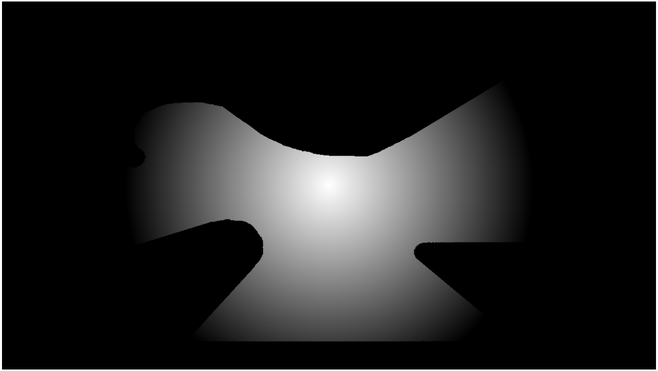
Common 2D lighting techniques

- Lighting in a 2D circle around the source

Demo by Greenblizzard

Common 2D lighting techniques

- Shadows in special cases (e.g. interior with vertical walls)



Demo by Rabid Lion Games

Dynamic "3D" lighting?

- How to *dynamically* light 2D images representing 3D objects?
- Recently, progress using normal maps (Stasis, Legend of Dungeon)

Stasis game by Christopher Bischoff

- But no public tools, documentation

Goals

- Realistic dynamic lighting in 2D
- Utilize modern hardware features
- Run on low-end hardware (integrated GPU, mobile)
- General-purpose open source tools for any project

Blinn-Phong reflection model

- Very common in (non-high-end) 3D games
- Good speed/result ratio
- Ambient, diffuse, specular
- ... But we're not using specular (right now)

$$illumintation = i_a * m_d + \sum_{l=1}^{lightCount} (a_l * m_d * i_{d,l} * directionToLight_l * normal)$$

$$a_l = 1/(1 + attenuation_l * distanceToLight_l)$$

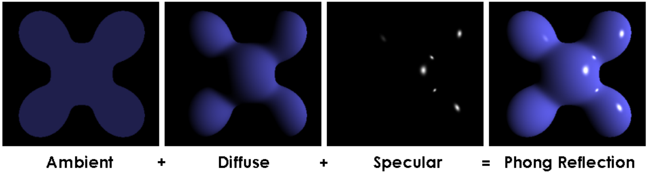


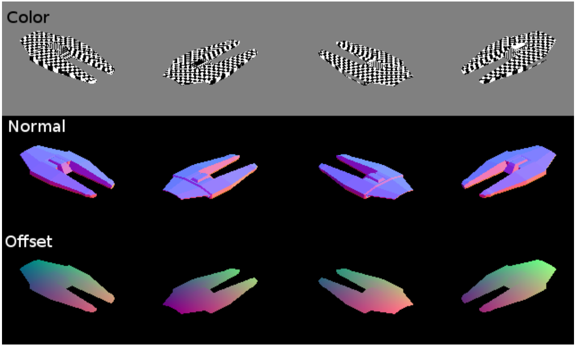
Image from en.wikipedia.org

Awesome2D

- Phong "in 2D"
- Generating 3D data on pixels of a 2D sprite.
- Calculating lighting using a 3D lighting model.
- Per-pixel data:
  - Relative 3D position
  - World-space normal
  - Color
- Data from game simulation
  - Object 3D position (even if the game is 2D)
  - Lights
- Can be extended later
  - Self-shadowed radiosity normal mapping, HDR, etc.

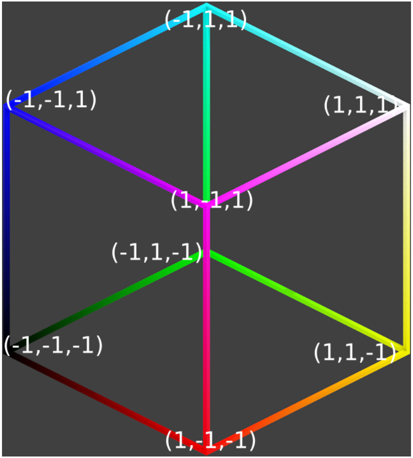
Encoding

- RGBA diffuse color
- RGB (3D vector) normal
- RGB (3D vector) position



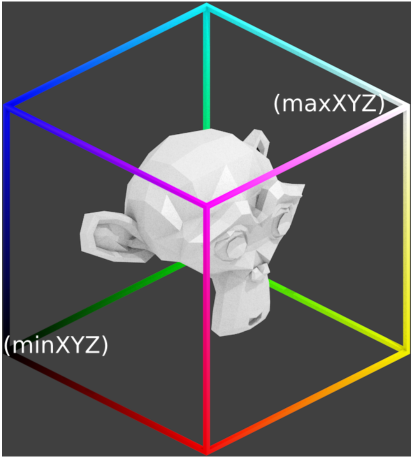
Encoding - Normals

- RGB to XYZ, 0-255 to -1 - 1



Encoding - Positions

- Bounding box (one per sprite)
- RGB to XYZ, 0-255 to bounds.min - bounds.max



Pre-renderer

- OpenGL/GLSL, using GPU shaders
- Creates 2D sprites from 3D
- Renders data for the Phong model (color, normals, offsets)
- General-purpose (not just 1 game)

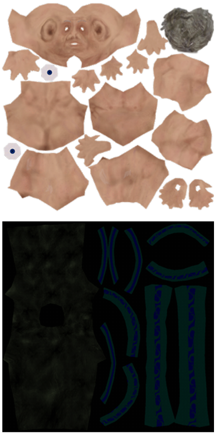
Demo & Lighting implementation

- OpenGL/GLSL again
- Lighting processed per pixel (GLSL fragment shader)
- No specular (yet?)
- Demo with a tiled dimetric map (pixel processing stress test)

Video

Performance scaling: 3D

- Time: 3D transform, vertices/triangles, screenful of pixels
- Memory: vertices, textures



Dwarf model by thecubber from http://opengameart.org

Performance scaling: 2D

- Minimum vertex overhead
- Time: pixel copy (fast)
- Memory: sprites
  - Animations take a lot of memory

Dwarf model by thecubber from http://opengameart.org

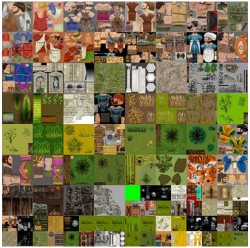
Performance scaling: 2D with lighting

- Minimum vertex overhead
- Time: screenful of pixels
- Memory: sprites
  - Animations take a lot of memory

Dwarf model by thecubber from http://opengameart.org

Feeding the GPU

- Pack pixel and vertex data
  - Vertices from multiple sprites in one buffer object
  - Power-of-two texture atlases



Texture atlas example by Christian Knudsen

- Reduce GPU/CPU communication to minimum
  - Reduce vbuffer, ibuffer, texture binds
  - Reduce uniform uploads
  - Memory alignment

Current performance

Tilemap stress test - 1024x768		
GPU	Driver	Avg FPS
Intel HD3000	Intel	60 (vsync)
GeForce8400M G	Nouveau	15
GeForce9600	NVidia	60 (vsync)
Radeon4550M	Gallium3D	46
Radeon6770	Catalyst	900
Radeon6850	Catalyst	1900

Current status

- Working demo with an isometric map & random stuff
- Basic pre-renderer
  - No AA
  - No animations
  - Single-model only

Roadmap

- Improve worst-case performance
- Improve demo, pre-renderer
- Spatial light management
- Lower bit-depth data (memory usage)?
- Future:
  - Revisit cut features, HDR, self-shadowing ...

More info

- http://kiithcoding.nfshost.com
- https://github.com/kiith-sa/awesome2D
- D
- OpenGL2/GLSL
- Assimp
- FreeType
- SDL2

Sources

- Inspiration: normal mapping in 3D games
- Bui Tuong Phong. *Illumination for computer generated pictures.* Communications of ACM 18, no. 6, 311-317 (1975)
- James F. Blinn. *Models of light reflection for computer synthesized pictures.* SIGGRAPH Comput. Graph. 11, 2, 192-198 (1977)
- Blinn, J. F. 1978. *Simulation of wrinkled surfaces.* SIGGRAPH 1978.
- P. Cignoni et. al. *A general method for preserving attribute values on simplified meshes.* (VIS '98).

Thank you for your attention!