Advanced lighting in 2D graphics

Ferdinand Majerech
Supervised by: RNDr. Ladislav Mikeš

Univerzita Pavla Jozefa Šafárika v Košiciach
UPJŠ

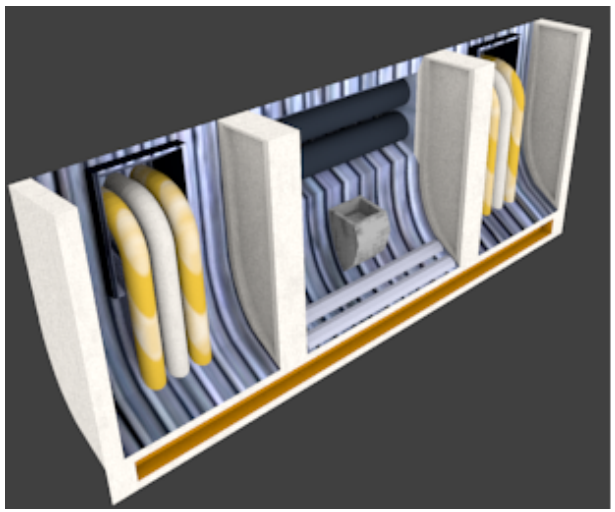
April 24, 2013

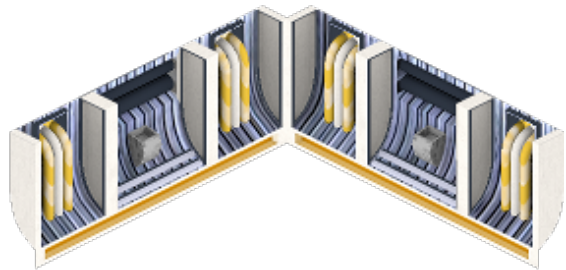
Why 2D

- Low HW requirements
- Good graphics on integrated, mobile, old GPUs
- Easy to code
- Easy to create art (even from 3D)
- Artists can “cheat”, 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





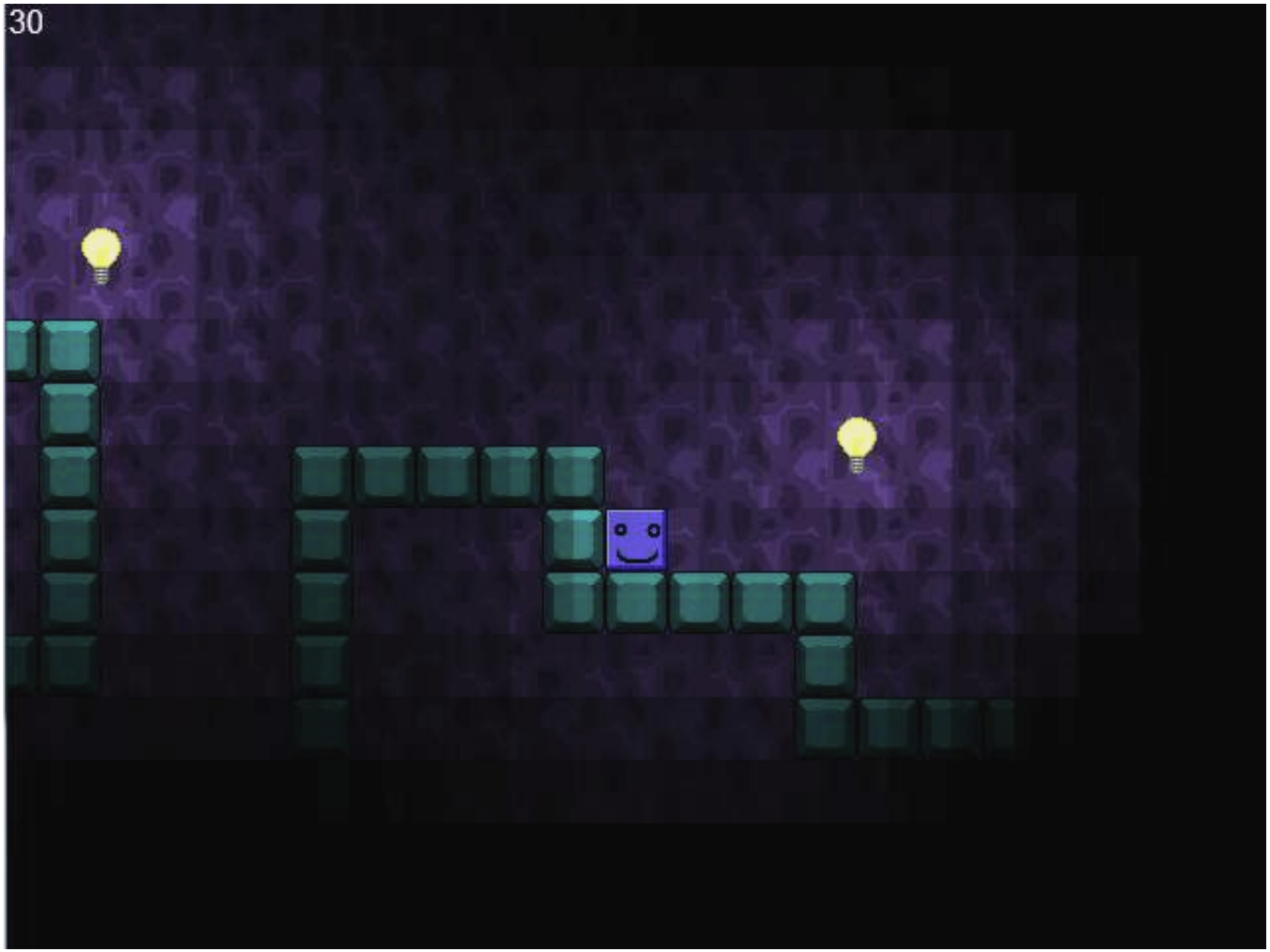
Common 2D lighting techniques

- Static lighting

Temple of Elemental Evil game by Troika Games

Common 2D lighting techniques

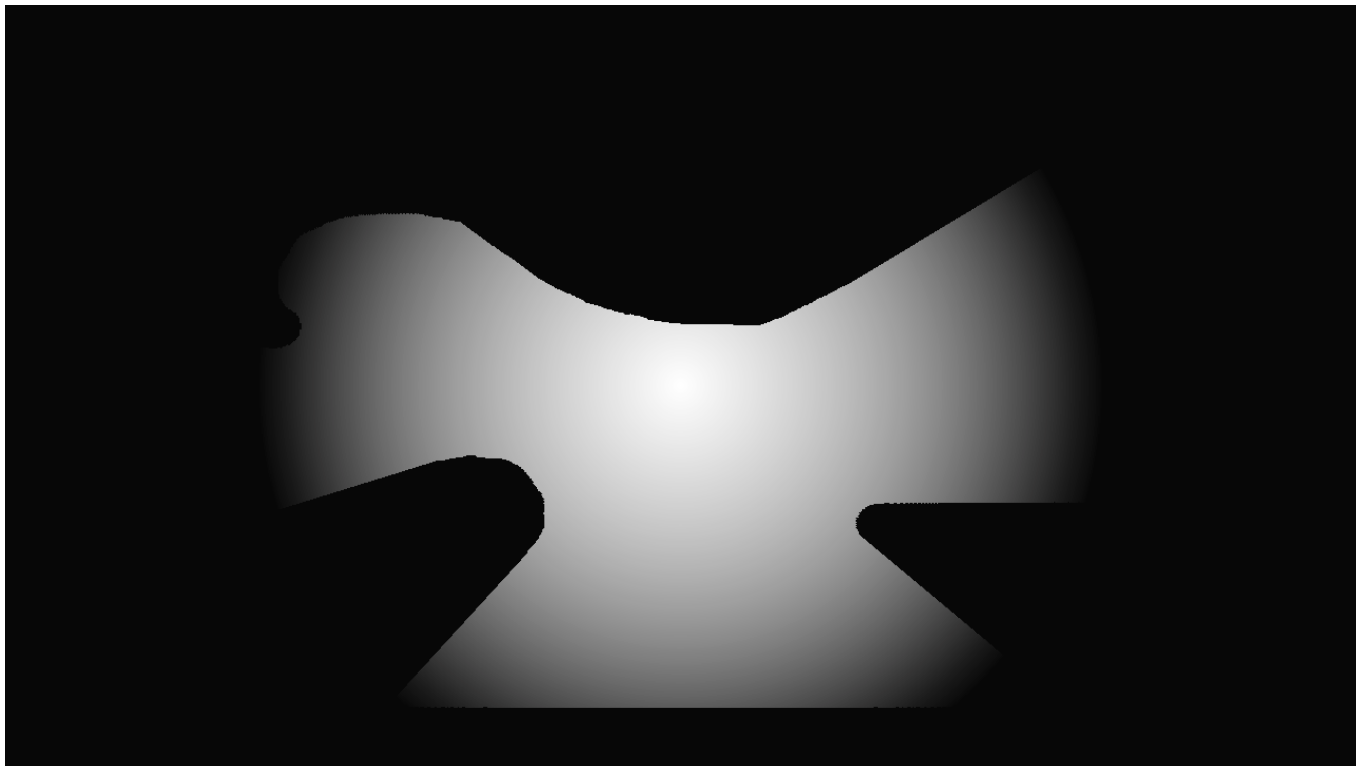
- Lighting based on 2D distance



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, Project Eternity)

Project Eternity, game by Obsidian Entertainment

- No public tools, documentation

Goals

- Believable dynamic lighting in 2D
- Utilize hardware features not available back in 2000
- Run on low-end hardware
- General-purpose open source tools for any project

Blinn-Phong reflection model

- Common in (non-high-end) 3D games
- Good time/quality ratio
- Ambient, diffuse, specular
- ... But we're not using specular (right now)

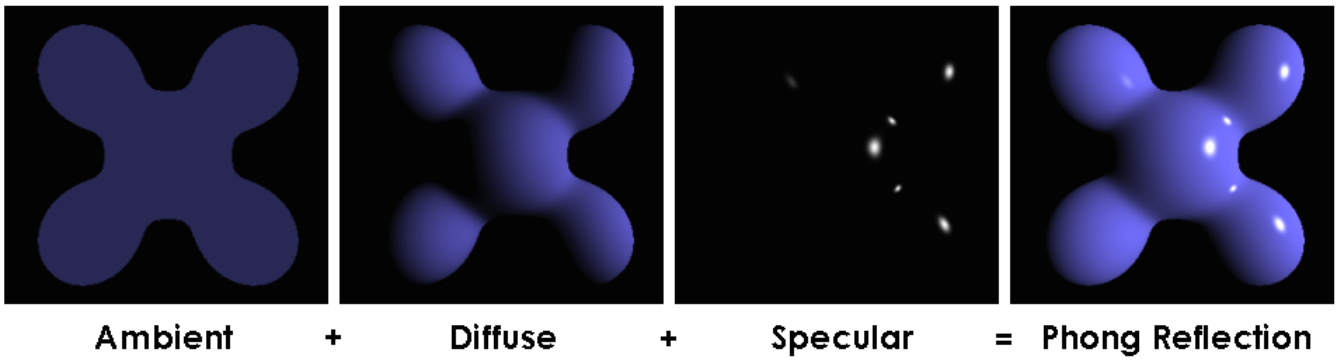


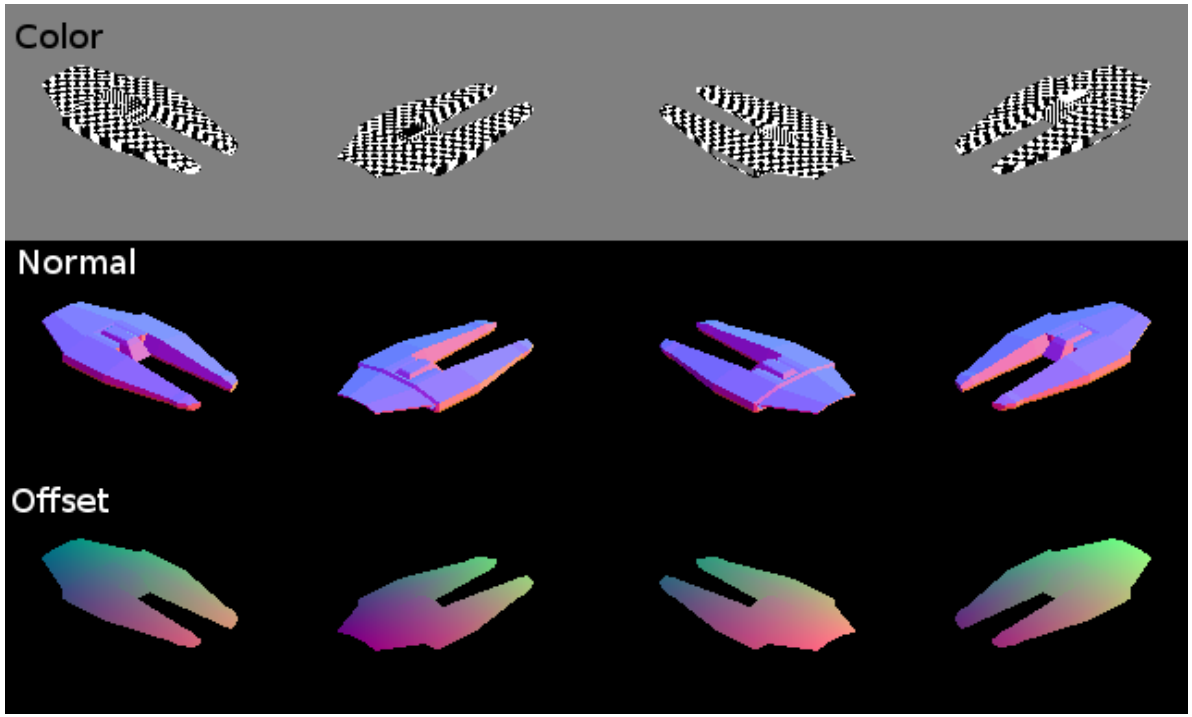
Image from en.wikipedia.org

Awesome2D

- Blinn-Phong on pixels of a 2D image.
- Generating 3D data from pixels on the GPU.
- 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 using existing techniques from 3D games.
  - Self-shadowed radiosity normal mapping, HDR, etc.

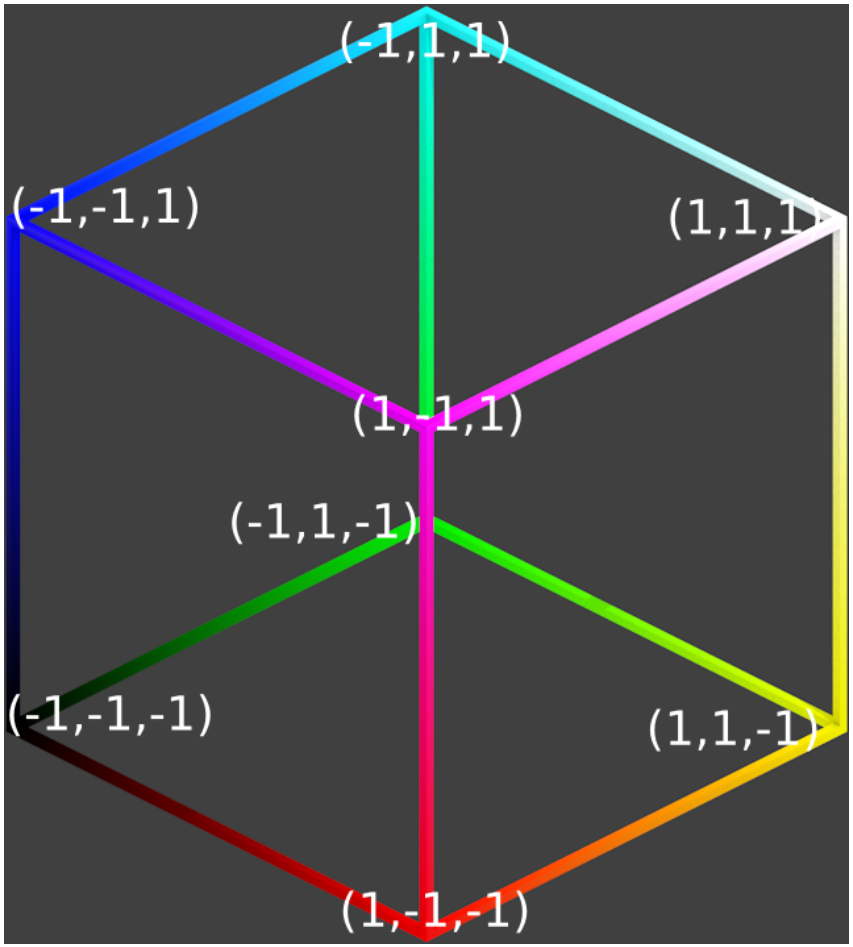
Encoding

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



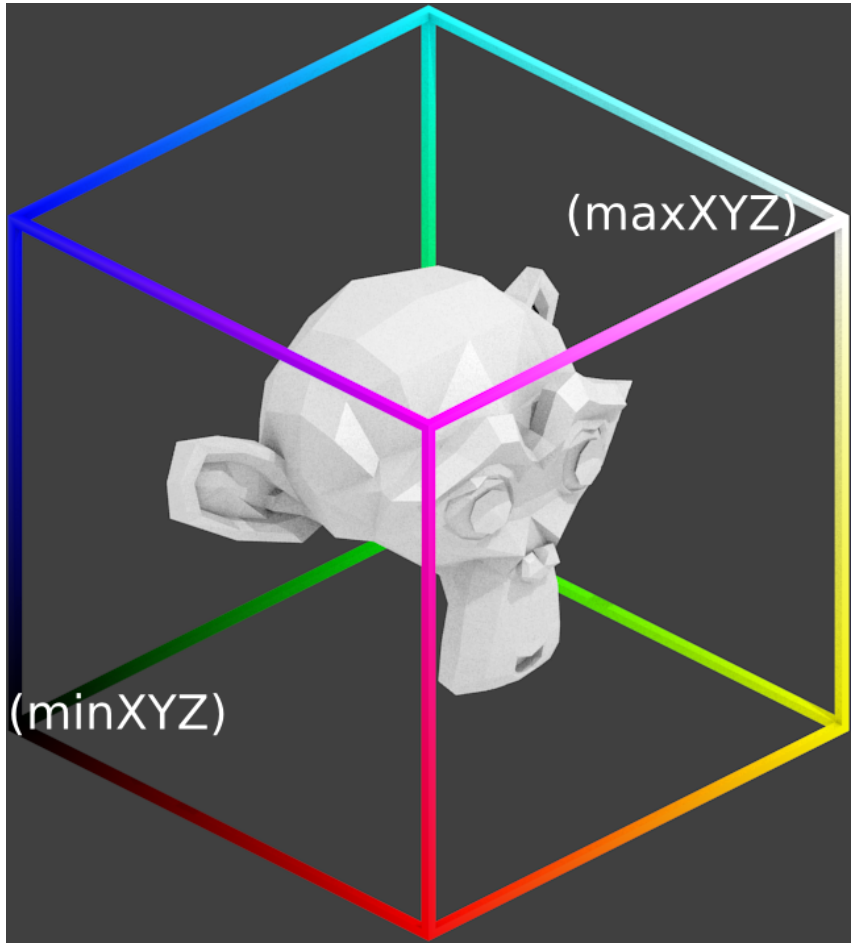
Encoding - Normals

- 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, using shaders in GLSL
- Creates 2D images from 3D
- Generates data for lighting (colors, normals, offsets)
- Usable by any 2D project (even different lighting models)
- CLI for scriptability (GUI in development by Tomáš Nguyen)
- Extensible (might render more data in future)

Demo & Lighting implementation

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

Video

Performance scaling: 3D

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

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

Performance scaling: 2D

- Negligible vertex overhead
- Time: pixel copy (fast)
- Memory: images (32bpp)
  - Animations can take a lot of memory (Need a separate image for every frame)

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

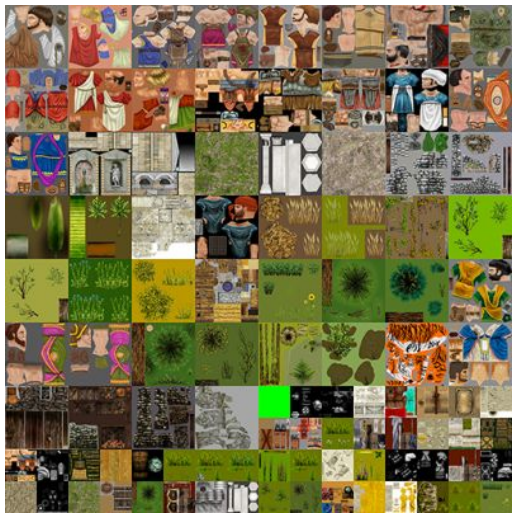
Performance scaling: 2D with lighting

- Negligible vertex overhead
- Time: screenful of pixels
- Memory: images (80bpp)
  - Animations can take a lot of memory (Need a separate image for every frame)

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

Feeding the GPU

- Many 4-vertex buffers are slow
- Many small non-power-of-two textures are slow
- Pack pixel and vertex data
  - Pack vertices for multiple images in one buffer
  - Pack images into power-of-two texture atlases



Texture atlas example by Christian Knudsen

- Reduce GPU/CPU communication to minimum
  - Upload buffers/textures to the GPU once, don't touch them later
  - Reduce vbuffer, ibuffer, texture binds

Current performance

- Acceptable: 24FPS

Tilemap benchmark - 1024x768		
GPU	Min FPS	Avg FPS
Intel HD3000	60	60 (vsync)
GeForce 8400M G	15	19
GeForce 9600	60	60 (vsync)
Radeon 4550M	40	46
Radeon 6770	852	870
Radeon 6850	1800	1900

Tilemap benchmark - 1920x1080		
GPU	Min FPS	Avg FPS
Intel HD3000	30	35
Radeon 6770	180	200

Current status

- Working demo with an isometric map & random stuff
- Basic pre-renderer
  - No animations
  - Single-model scenes only (an image can only contain one object)

Future

- Improve worst-case performance
- Improve demo, pre-renderer
- Spatial light management
- Heightmap for positions
- Z-buffer
- Revisit cut features, HDR, self-shadowing ...

More info

- http://kiithcoding.nfshost.com
- https://github.com/kiith-sa/awesome2D
- D
- OpenGL2/GLSL
- Assimp
- FreeType
- SDL2
- gl3n

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!