Texture and Environment Maps
Fall 2018
Texture Mapping

- Problem: colors, normals, etc. are only specified at vertices
- How do we add detail between vertices without incurring penalty for higher poly count?
- Solution: Specify details in an image (the texture) and specify how to apply the image to the geometry (the map)
- Works for shading parameters other than color
Basic Mapping

- Textures live in a 2D space
  - Parameterize points in the texture with 2 coordinates: \((s, t)\)
- Define the mapping from \((x, y, z)\) in world space to \((s, t)\) in texture space
- For polygons:
  - Specify \((s, t)\) coordinates at vertices
  - Interpolate \((s, t)\) for other points
Basic Texturing Concepts and Terms

• Vertices are coordinates that define geometry
• Texture coordinates specified at vertices and interpolated across triangles
• Texture values for points mapping outside the texture image can be generated in various ways:
  • REPEAT, CLAMP, etc
• Width and height of texture images is often constrained
  • Powers of two
  • Sometimes required to be a square
• Fragments are single pixels (or sub-pixels) that must be “lit” and rasterized
Aliasing Artifacts

- Texture sampling (aliasing) creates visual artifacts

- Mipmaps and other filtering techniques are the solution
Multitexturing

- Multitexturing hardware provides pipeline of texture units to work on fragments
  - Apply multiple textures to a primitive in the same pass
- Example
  1. Apply color map
  2. Modify illumination to simulate bumps
  3. Modify opacity

Note that this is not the same as a multi-pass rendering — done in a single pass!
Pixel Shaders

- Pixel shaders operate on fragments in parallel
  - (Vertex shaders operate on vertices in parallel)
- Can compute texture coordinates, do general texture look-ups, modify color/depth/opacity, and some other functions
- More general than multi-texturing and very powerful
- Shader languages include GLSL, HLSL and Cg
- We won’t go into shading for this course, but here is a basic tutorial:
  
  http://www.lighthouse3d.com/tutorials/glsll-tutorial/
Textures in Games

- Modern game engines provide a lot of texture support
  - Automatic texture atlasing
  - High level of control for importing and editing
  - Varying resolutions to support hardware performance
- Artist tools often included for texture management
  - Design texture images
  - Specify how to apply to object
  - Profiling to maintain good memory and performance
Texture Atlasing

- A packed set of textures (or sprites)
- 2D Example: a sprite sheet

![Sprite Sheet Image]
3D Example: Texture Tool
Packing Textures

- Problem: Limits on texture width/height make it inefficient to store many, high-quality textures
- Solution: Artists pack the textures for many objects into one image
  - The texture coordinates for a given object may only index into a small part of the image
  - Care must be taken at sub-image boundary to achieve correct blending
  - Mipmapping is restricted
  - Best for objects that are at a known resolution
Combining Textures
A Larger Example
Animating Texture

- The texture matrix can “transform” the texture in memory
  - Less expensive than loading multiple textures onto the graphics card
  - Allows texture to slide, rotate, and stretch/shrink over surface
- If the texture matrix is changed from frame to frame, the texture will appear to move on the object
- Useful for things like flame, swirling vortices, or pulsing entrances...
Projective Texturing

• The texture can be projected onto the scene as if from a slide projector
  • Equate texture coordinates with world coordinates
  • Wherever a world point appears in the projector’s view, it picks up the texture at that point
• Useful for other texture projections such as shadow mapping

• Original paper:
  • [http://www.nvidia.com/object/Projective_Texture_Mapping.html]
What Do Textures Represent?

• The graphics hardware doesn’t know what is in a texture
• It applies a set of operations using values it finds in the texture, the existing value of the fragment (pixel), and maybe another color
• The programmer decides what these operations are
• Examples:
  • Scalar luminance data (multiplies the fragment color)
  • Alpha data (multiplies the fragment’s alpha channel)
  • Vector data (modifies the surface normals)
  • Depth data (determines distance from light source for shadow mapping)
Textures In Digital Art

• Assets designed for modern graphics pipeline
  • Low-poly, high-texture
  • Multiple maps for multiple effects
• Example:
  • https://www.artstation.com/artwork/2yVKB

(Arnab Roy, Maya)
Question

• How can we capture reflection in a local shading model that doesn’t consider neighboring objects?
Environment Mapping

- Environment mapping produces reflections on shiny objects.
- Texture is transferred in the direction of the reflected ray from the environment map onto the object.
Environment Mapping cont’d

• Reflected ray: \( R = I - 2(N \cdot I)N \)
Example
Mapping Approximations

- We can’t store a separate map for each point, so one map is used with the eye at the center of the object
  - Introduces distortions in the reflection
  - Distortions minimized for a small object in a large room
- Object will not reflect itself
- Mapping can be computed at each pixel or only at the vertices
Types of Environment Maps

- Environment map may take one of several forms:
  - Cubic mapping
  - Spherical mapping (two variants)
  - Parabolic mapping
- Describes the shape of the surface on which the map “resides”
- Determines how the map is generated and how it is indexed
- Issues in choosing a map:
  - Memory
  - Computation
  - Accuracy
Cube Mapping

- The map resides on the surfaces of a cube around the object
  - Typically align the faces of the cube with the coordinate axes
- Can make map rendering arbitrarily complex as its possible to do off-line
- For each face of the cube either:
  - Render the world from the center of the object with the cube face as the image plane
  - Or take 6 photos of a real environment with a camera in the object’s position
Cube Map Example