

Worksheet 3

If we compare rendered images to photographs, flatly coloured surfaces with no visual detail is one of the first giveaways that an image is a rendering. To obtain realism in rendered images, it is important to add visual detail. One way to add visual detail is to use texturing. This set of exercises is about how to do texturing in ray tracing.

Learning Objectives

- Use texture mapping (mapping an image to a surface) to heighten the level of visual detail.
- Compute texture coordinates using inverse mapping.
- Use bilinear interpolation for texture magnification filtering.
- Use stratified jitter sampling for texture minification filtering.

Texture Mapping

Texture mapping is in four steps: (1) loading the texture image, (2) computing texture coordinates for the object to which the texture should be mapped, (3) looking up a colour in the texture image for a given set of texture coordinates, and (4) using the texture colour in a shader.

- Create a function that loads a texture image file using the HTML `img` element and copies the image data into a `Uint8Array` using the HTML `canvas` element. Once loaded, store the image data in a texture created on the GPU and create a sampler with `uv` address modes as well as minification and magnification filters. Use image plane coordinates (`uv`) to render images of the texture with `clamp-to-edge` address modes versus `repeat` address modes and with `nearest` filtering versus `linear` filtering.
- Add texture coordinates (`texcoords`) and a Boolean saying whether to use texturing or not (`use_texture`) to your hit info struct (`HitInfo`). Compute texture coordinates when finding an intersection with a plane by finding the vector from the plane origin to the intersection point and projecting it onto the tangent and binormal of the plane, respectively (this is an inverse mapping). Use a texture scaling factor of 0.2 to scale the texture coordinates. Modify the main function (`main_fs`) to have two variables: the existing variable for untextured results returned by a shader (`result`) and then one for results from a shader to be textured (`textured`). Use the Boolean (`use_texture`) to decide whether a result from a shader is added to one variable or the other. After tracing a path, multiply the sum of textured shader results (`textured`) by the colour in the texture sampled at the texture coordinates stored in the hit info struct (`hit.texcoords`) and add this to the untextured results (`result`). The ray tracing can in this way include texturing.
- Do stratified jitter sampling for anti-aliasing of your ray traced images. Implement an interface in your HTML page for incrementing and decrementing the pixel subdivision level. Implement a JavaScript function (`compute_jitters`) computing an array of vectors from the pixel centre to a jitter sampled position for each sub-pixel. Call this function whenever the pixel subdivision level is changed and store the array in a pre-allocated storage buffer. Modify the main function `main_fs` so that a ray is cast through each sub-pixel. For each sub-pixel, use the jitter vectors in the storage buffer to modify the image space coordinates used for generating the ray. Accumulate the result from each sub-pixel and divide by the number of sub-pixels.
- Render the default scene using different pixel subdivision levels. Compare result from `nearest` and `linear` texture sampler filtering and explain how scaling the texture coordinates affects the rendered texture. Divide the texture scaling factor by 10 to magnify the texture by a factor 10. Describe how texture aliasing is affected by pixel subdivision level and `nearest` versus `linear` filtering, respectively.

Reading Material

The curriculum for Worksheet 3 is (41 pages)

B Chapter 11. *Texture Mapping*.

B Section 13.4.1. *Antialiasing* (from 4th edition of the text book).