1 PhysBAM Ray Tracer

1.1 Render Examples

1.2 Spring Example

- Show how to run it (run without arguments to do this)
- Walk through scene file contents. Point out that this only uses a subset of features.
- Mention stuff at top
  - View parameters, will get to later.
  - Mention size, output file, quality.
  - Also option for rendering only patch to improve turn around.
- Lights
  - Makes renders not black
  - Light type. These are point lights, also have area lights
  - Briefly mention what the options are for
- Shaders
  - How objects look
  - Colors - for defining new colors
  - Shaders can have sub-shaders
  - Shaders - Phong, Lambertian
  - Special - reflection, transparency, blending
- Objects
  - The scene being rendered
  - Boxes
  - Triangulated surface
  - Spheres
  - Cylinders
- Show high quality
- Show full movie

1.3 Torus Example

- Show how to get the camera script file from viewer
- Mention stuff at top
  - View parameters
- Lights
  - Note extra options to point light.

- Shaders
  - Go through "New" shader tree for deformable object torus
- Objects
  - Deformable object - sim data
  - Similar features for other sim data (rigid bodies, fluids)
- Show high quality
- Show full movie

2 Ray Tracing

2.1 Forward Tracing Vs. Backward Tracing

- Forward
  - Follow the photons
  - Start at lights
  - Bounce around, end at camera
- Backward
  - Only trace rays that will be observed
  - Start at camera
  - When hit something, shot rays to lights

2.2 Camera Setup

- Draw a diagram with a camera, a piece of film, two objects, and two lights.
- Make sure one of the objects casts a shadow on the other.
- Draw pixels on the film, shoot rays through them.
- Shoot a few rays, some intersect objects, some go into space

2.3 Intersections

- Must hand ray-object intersections
- Lots of them, should be fast
- Go through a few examples
  - Ray-plane (Derive)
  - Ray-sphere (Derive)
  - Ray-box (intersect each side)
    - Find t-min and t-max for both x and y faces
    - Make sure these intervals overlap
    - Intersect the time intervals
    - Compute t-min and t-max for both z faces
    - Make sure these intervals overlap
    - Intersect the time intervals

2.4 Surface Shading

- At each pixel, we know which object light ray came from
- What color is it? How bright?
- If the light came from an object, how did light get there?
- What if the ray intersected nothing? (environment map - sky)
- To determine color in case of an object, need a shading model.

2.4.1 Phong Shading

- Intensity: $I = k_a i_a + \sum_{\ell \in lights}(k_d I_{d,\ell} + k_s I_{s,\ell}) i_\ell$
- Ambient approximates indirect lighting that is too expensive to compute
- Diffuse
  - $I_{d,\ell} = \cos\theta = L \cdot N$
  - θ is angle between light ray and surface normal
  - L is direction towards light
  - N is surface normal
  - Diffuse surface emit evenly in all directions - does not depend on view direction
  - Emit energy proportional to energy received
- Specular
  - $I_{s,\ell} = \cos^p\phi = (R \cdot V)^p$
  - ϕ is angle between viewing ray and reflected light ray
  - V is direction towards camera
  - $R = 2(L \cdot N)N - L$ is reflected light ray
  - Exponent p determines the sharpness of the specular highlight
- The coefficients k_a, k_d, and k_s determine the brightness of the surface.
- The computation is performed per RGB channel, so colored surfaces are modeled by having coefficients that differ in each channel.

2.4.2 Shadowing

- Shadows complicate things a bit.
- A light should not contribute to the color emitted from a surface if it is not visible from that surface.
- From the surface, shoot a ray to each light.
- Normally only need to test visibility. Do not need to evaluate shaders.
- Does not take into account that light is emitted by other surfaces

2.4.3 Reflection

- Ray tracing naturally handles reflective surfaces.
- Shoot off a new ray in reflected direction.
- Mix with another model to get semi-reflective surfaces

2.4.4 Refraction

- Surfaces like glass are transparent
- Fresnel equations
  - Part of the light reflects from the surface
  - Part is refracted (transmitted but bent)
  - Polarization dependent
- Snell's law: $n_1 \sin\theta_1 = n_2 \sin\theta_2$
- Index of refraction: n
- Water $n \approx 1.333$, Air $n \approx 1$
- Critical angle
- What about shadowing?

2.5 Acceleration

- Take advantage of spatial coherence (spatial partitioning)
- Many ways to do this..
- Demonstrate these two
  - Bounding box hierarchy
    - Divide and conquer
    - ray-box intersection tests to search
  - Regular grid
    - Rasterize to grid
    - Bresenham's line algorithm to search
    - Adaptivity