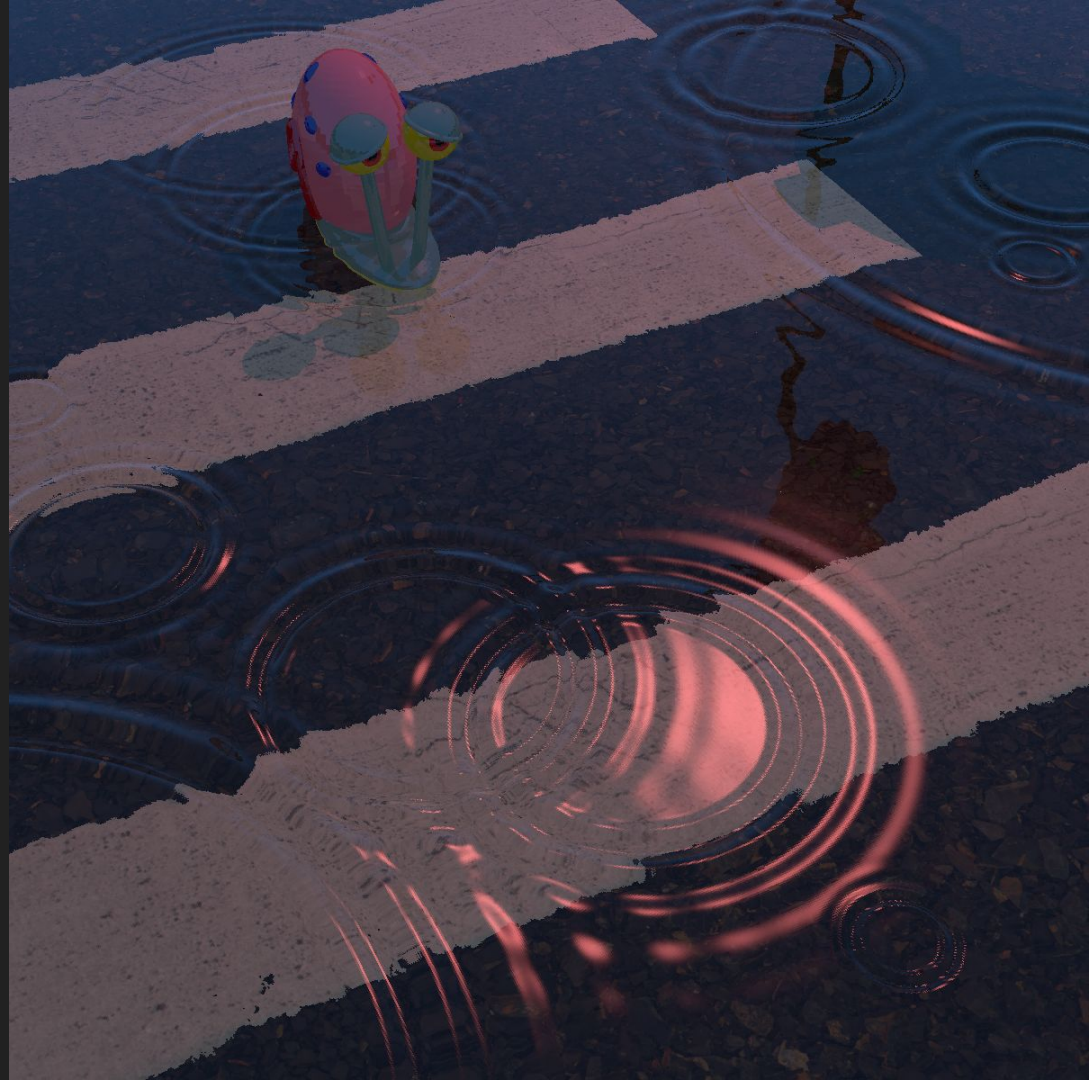
CS4620-C2

Harvey Zhu & Brian Song



Bounding Volume Hierarchy(BVH)

- Organizes **surfaces** into a **binary tree** of **bounding boxes**
- Each box contains either smaller boxes (interior node) or a leaf box (containing e.g. 8 basic geometries)
- Rays search for Hit starting from the root

- Single round render:
  - 800*600 image = 480k pixels
  - 200k geometries
  - Max_leaf_size = 8
  - → ~25k leaf nodes, ~14 levels deep
  - Without BVH:
    - 200k*480k = 96000 million tests
  - With BVH:
    - (~30 box test + ~2*8 leaf test)*480k
    - = ~ 22 million test
  - → More than 4000x speedup

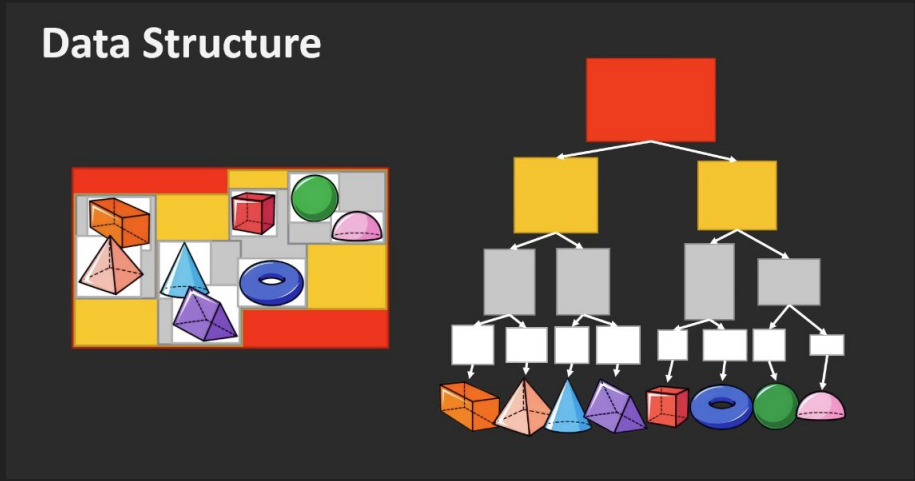


Image Source: https://www.youtube.com/watch?v=LAxHQZ8RjQ4&t=347s

Note: ~2*8 leaf test: assume on average we need to test two leafs to find a hit

Performance Comparison

(For a simplified scene 240*240, 4568 (water) + 6 (roadlines) + 1 (ground plane) primitive geometries)

A4-Intro-To-Graphics-F2025 — -zsh — 80×24

```
    self._event.wait(timeout)
  File "/opt/anaconda3/envs/4620-A4/lib/python3.10/threading.py", line 607, in w
ait
    signaled = self._cond.wait(timeout)
  File "/opt/anaconda3/envs/4620-A4/lib/python3.10/threading.py", line 320, in w
ait
    waiter.acquire()
KeyboardInterrupt

(4620-A4) zitengsong@dhcp-v12041-3293 A4-Intro-To-Graphics-F2025 % python Puddle
Scene.py
Setting up puddle scene...
Loading textures...
  Creating adaptive mesh: base 40x40, max_depth=2, threshold=0.001
  Adaptive mesh complete: 4568 triangles (would be 3200 with uniform grid)
Building BVH for 4575 primitives...
BVH construction complete.
Rendering puddle scene...
This CPU has 8 cores.
Rendering with 8 processes...
Rendering finished in 39.75 seconds.
Saving image...
Saved to puddle_scene.png
(4620-A4) zitengsong@dhcp-v12041-3293 A4-Intro-To-Graphics-F2025 %
```

With BVH: 39.75s

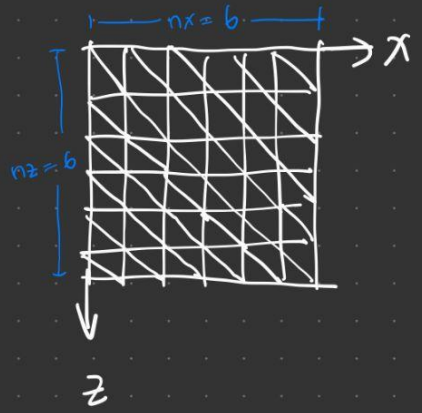
A4-Intro-To-Graphics-F2025 — -zsh — 80×24

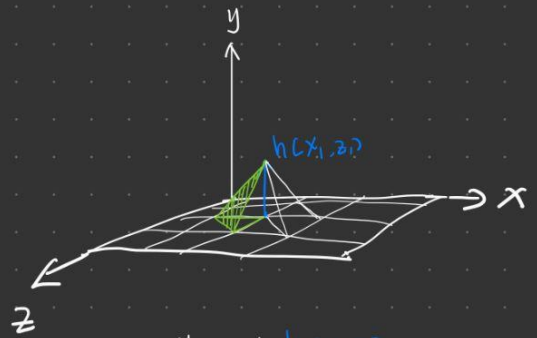
```
(4620-A4) zitengsong@dhcp-v12041-3293 A4-Intro-To-Graphics-F2025 % python Puddle
Scene.py
Setting up puddle scene...
Loading textures...
  Creating adaptive mesh: base 40x40, max_depth=2, threshold=0.001
  Adaptive mesh complete: 4568 triangles (would be 3200 with uniform grid)
Rendering puddle scene...
This CPU has 8 cores.
Rendering with 8 processes...
Rendering finished in 4059.86 seconds.
Saving image...
Saved to puddle_scene.png
(4620-A4) zitengsong@dhcp-v12041-3293 A4-Intro-To-Graphics-F2025 %
```

Without BVH: 4059.86s

Ripple Simulation - Water Mesh

```
def refine_quads(quad_list, current_depth):
    if current_depth >= max_depth:
        return
    for quad in quad_list:
        if quad.is_leaf:
            gradient = quad.get_height_gradient()
            if gradient > gradient_threshold:
                quad.subdivide(height_func)
                refine_quads(quad.children, current_depth + 1)
```





$y = y_{base} + h(x, z)$

height function

Optimization :

Define : Quad :: corners[] ← [x, y, z]

If $\max(\text{height}(\text{corners})) - \min(\text{height}(\text{corners})) \geq \text{gradient\_threshold}$.

Subdivide:

Recurse till max_depth.

```
def create_water_mesh(width, depth, base_nx, base_nz, y_base=0.0,
                      height_func=None, material=None,
                      max_depth=3, gradient_threshold=0.015):
```

Ripple Simulation - Height Function h(x,z)

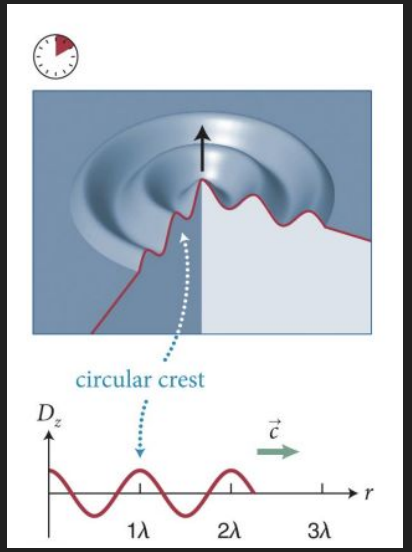
- Simplified version based on **Damped Wave Equation**
- Approximated solution: $h(r,t) = A_0 e^{-\alpha r} \sin(kr - \omega t)$
- Removed t (time) for static image, added 3 piece-wise phases for shape control

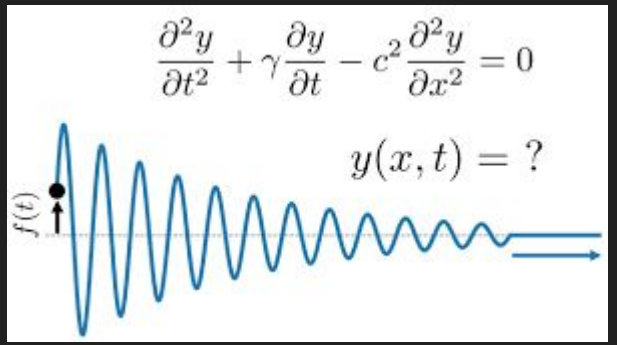
$$h(r) = \begin{cases} 0, & r < R, \\ \frac{1}{2}\left(1 - \cos\left(\pi \frac{r-R}{d}\right)\right) A\, e^{-\alpha(r-R)} \sin(k(r-R)), & R \le r < R+d, \\ A\, e^{-\alpha(r-R)} \sin(k(r-R)), & r \ge R+d, \end{cases}$$

- Redesigned parameters for better customizability:

```
{
    "center": (-0.5, 0.0, 0.5),
    "amplitude": 0.08,
    "first_ripple_radius": 0.3,
    "ring_width": 0.1,
    "fade_wavelengths": 0.5,
    "rings_to_fade": 3,
},
```

```
VISIBILITY_THRESHOLD = 0.1
```



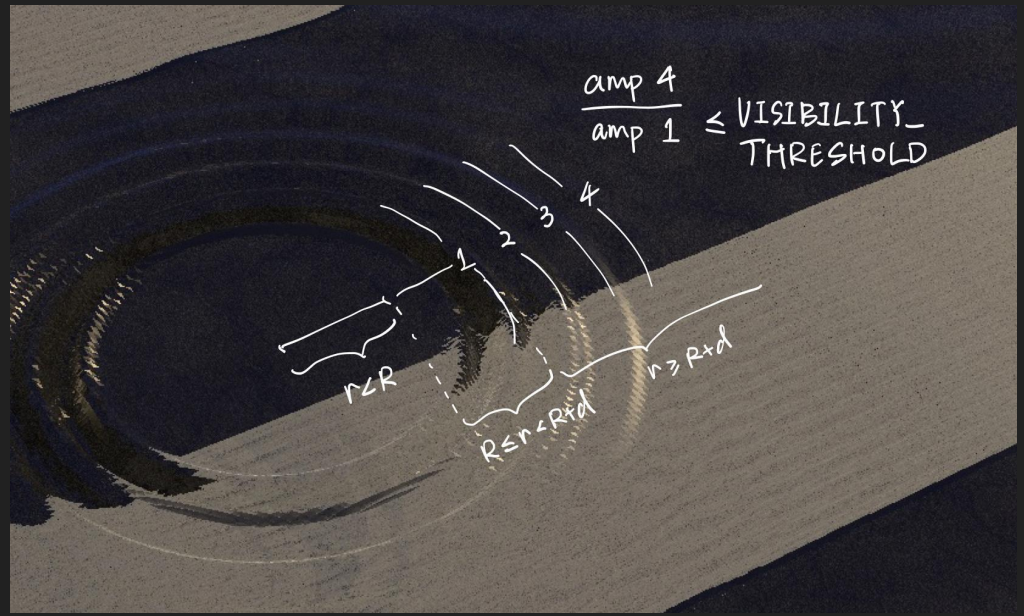


Ripple Simulation - Height Function 3 Phases

$$h(r) = \begin{cases} 0, & r < R, \\ \frac{1}{2}\left(1 - \cos\left(\pi \frac{r - R}{d}\right)\right) A\, e^{-\alpha(r-R)} \sin\big(k(r - R)\big), & R \le r < R + d, \\ A\, e^{-\alpha(r-R)} \sin\big(k(r - R)\big), & r \ge R + d, \end{cases}$$

```
{
    "center": (-0.5, 0.0, 0.5),
    "amplitude": 0.08,
    "first_ripple_radius": 0.3,
    "ring_width": 0.1,
    "fade_wavelengths": 0.5,
    "rings_to_fade": 3,
},
```

```
VISIBILITY_THRESHOLD = 0.1
```

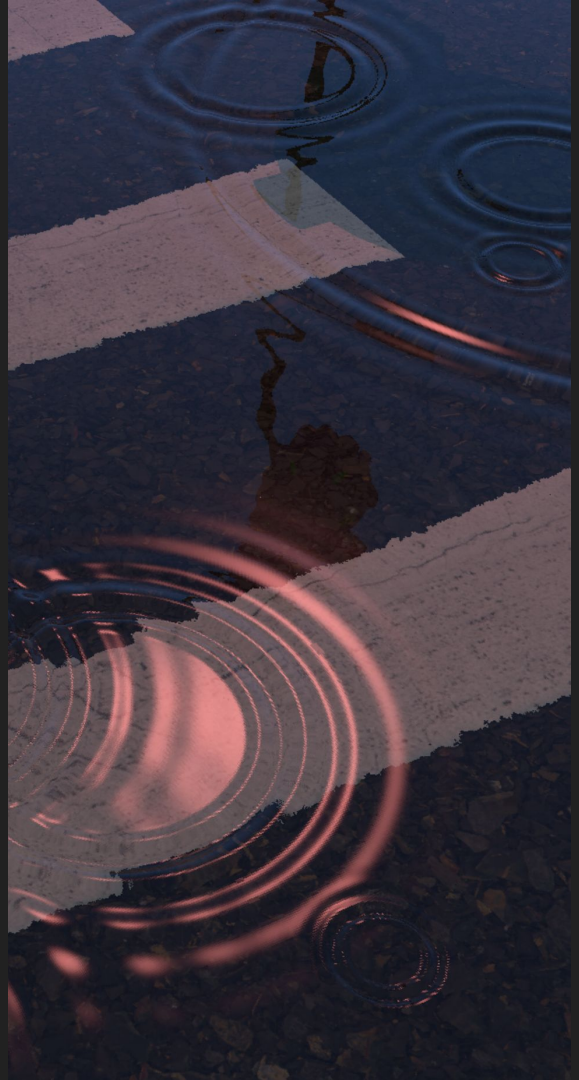


Shading Improvements for Water

- **Refraction**
  with Snell's Law

- **Schlick Fresnel**

- **Transparent Shadows**
  Shadow rays accumulate transmittance (how much light remains) through multiple transparent surfaces along the way to the light source



Texture Mapping

- **Diffuse Texture Mapping**
- **Normal Mapping**

Texture source: https://ambientcg.com/

Blender Models