kd-Trees, Particles, and other fun
Overview

- kd-Trees
  - Construction
  - Intersection tests
- Point Sprites
But first ...
You implemented a lot of advanced rendering features *from scratch*!
You now have a lot of useful experience to go into the graphics industry and implement modern rendering techniques
You have a very strong understanding of multi-pass techniques and the associated issues
You have a good starting point for your projects
In the old cs248, assignments 3 and 4 could have been a whole CS 248 project.

It took 3 guys 3 weeks (including thanksgiving break) to implement:

- Normal Mapping
- Shadows
- Collision detection
- Physics
- Particles
Some Perspective
Some Perspective
Again, assignment 3 was hard, and you should be very proud of yourselves...
Assignment 4

- You be implementing collision detection using a kd-Tree
- The kd-Tree will be used to perform line-segment triangle intersection tests
- You will also implement a basic particle engine and add 3 different types of particle systems
kd-Trees - Construction

- Split the word along one axis into 2 cells
- Figure out which cell each point belongs to
- Recursively repeat this procedure in each cell
- At termination criterion:
  - Store a pointer to all the points contained in each leaf cell
kd-Trees - Construction

- Many design decisions
  - Which axis to split
    - Adaptive?
    - Cycling?
  - Where to split
    - Median
    - Midpoint
    - Others...
  - Termination Criterion
    - Number of objects
    - Cell Size
Many design decisions

- Which axis to split?
  - Adaptive
  - Cycling

- Where to split?
  - Median -> typically results in the most balanced tree
  - Midpoint
  - Others...

- Termination Criterion
  - Number of objects
  - Cell Size
In this assignment, we’re storing triangles, not points.

Treat each triangle, as a bounding box, and use the center of the BB as triangle’s location.

Have to be careful about when the bounding box crosses the splitting plane.
kd-Tree example
kd-Tree example
kd-Tree example
kd-Tree example
kd-Tree example
kd-Tree example
kd-Tree example
Issues with naïve solution

- Empty BBs
Issues with naïve solution

- BBs not tight
Possible optimizations

- Remove empty BBs at each step
- Collapse BBs so that they are tight around their triangle segment
  - Requires 2 line segment – plane intersection tests
Questions?
Collision detection

- Each particle is treated as a point
- We treat its path between time \( t \) and \( t + 1 \) as a line segment
- We collide this line segment with the triangles using tree kd-tree
  - Need to be careful about when line segment crosses splitting plane
  - Select the intersection point that is earliest
kd-Tree example
kd-Tree example
kd-Tree example
kd-Tree example
kd-Tree example
kd-Tree example
Camera Collision
Camera Collision
Camera Collision
Camera Collision
Camera Collision
OpenGL can also render points instead of triangles

- Use GL_POINTS instead of GL_TRIANGLES
- glPointSize() specifies the *rasterized* size of points
  - i.e. they don’t become smaller with distance, regardless of projection
Rendering Particles

- Particles are *almost* like points, but they have a little bit more texture to them
- **Solution:** Point Sprites!
Point Sprites

- OpenGL provides functionality for **point sprites**, which replaces points with textured quads.
// setup and bind() texture
glEnable(GL_POINT_SPRITE);

// Render Points

glDisable(GL_POINT_SPRITE);
Useful Point Sprite Functions

//shader attribute
gl_PointSize

//Modulate the color of the sprites (useful for controlling fade)
glColorPointer(size, type, stride, pointer);
//Ex. glColorPointer(4, GL_FLOAT, 4 * GL_FLOAT, my_clr_ptr)
Now all that’s left is to do all the book-keeping for a particle system.
Euler integration will be fine.

Possible structure:

```c
struct Particle {
    age;
    lifetime;
    position;
    velocity
    //etc.
    update(time);
};

class Particle_Emitter{
    init();
    spawnParticles();
    renderParticles();
    myParticles[];
    myType;
};

class Particle_Emitter{
    init();
    updateAll();
    renderAll();
    myEmitters[]
};
```
Don’t Forget! Final Project proposals are due tonight!
Good Luck!