

CSE 167 DISCUSSION 5

OCTOBER 31, 2018



TA: Jimmy Ye ft. Tutor: Weichen Liu
University of California, San Diego
Fall Quarter 2018

Halloween Theme is from CSE167 F16 Discussion 6 slide

Announcements

- Happy Halloween!
- Project 3 due this Friday 11/02
- Feel free to get some candies before you leave

Halloween Theme is from CSE167 F16 Discussion 6 slide

Overview

- View frustum culling
- A little Scene Graph
- Some most asked questions (if we still have time)

Halloween Theme is from CSE167 F16 Discussion 6 slide

Culling

- It’s possible to do frustum culling in different spaces:
  - View space (Camera space)
  - World space
  - NDC space (Canonical volume space)

Halloween Theme is from CSE167 F16 Discussion 6 slide

Culling

- It’s possible to do frustum culling in different spaces:
  - View space (Camera space)
    - Do culling after applying view matrix
    - We only need to calculate the plane representation once

Halloween Theme is from CSE167 F16 Discussion 6 slide

Culling

- It’s possible to do frustum culling in different spaces:
  - World space
    - Do culling after applying model matrix (when we have world space coordinates)
    - We need to re-calculate the plane representation each time we move the camera.

Halloween Theme is from CSE167 F16 Discussion 6 slide

Culling

- It’s possible to do frustum culling in different spaces:
  - NDC space (Canonical volume space)
    - Plane representation is trivial
      - Note: In NDC space, view frustum is simply a [2x2x2] cube.
    - But bounding sphere is no longer a “sphere” (skewed), we would need to re-calculate the radius of it.

Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- Why this one?
- How it works?

Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- Why this one?
  - Easiest
  - Less expensive (than in NDC space)

(I may go over ~~culling in other space~~ EC briefly if we have enough time)

Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- How it works?
  - Recap: To do culling, we actually want to calculate point-to-plane distance.
    - $dist(x) = (x - p) \cdot n$, s.t. x is the point, p is a point on the plane, n is plane normal
    - When dist(x) > 0, the point x is on the side of the plane the normal points to.

Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- How it works?

$$dist(x) = (x - p) \cdot n$$

  - Recap: To do culling, we actually want to calculate point-to-plane distance.
  - Therefore, we need following information in camera space:
    - Point coordinate
    - A point on each plane
    - Plane normal of each plane

Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- How it works?
  - Recap: Camera space coordinate
  - Transform a point p to camera space: $p' = C^{-1}Mp$
    - glm::vec3 new_p = Window::V * toWorld * old_p;

Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- How it works?
  - Recap: To do culling, we actually want to calculate point-to-plane distance.
  - Therefore, we need following information in camera space:
    - ~~Point coordinate~~
    - A point on each plane
    - Plane normal of each plane

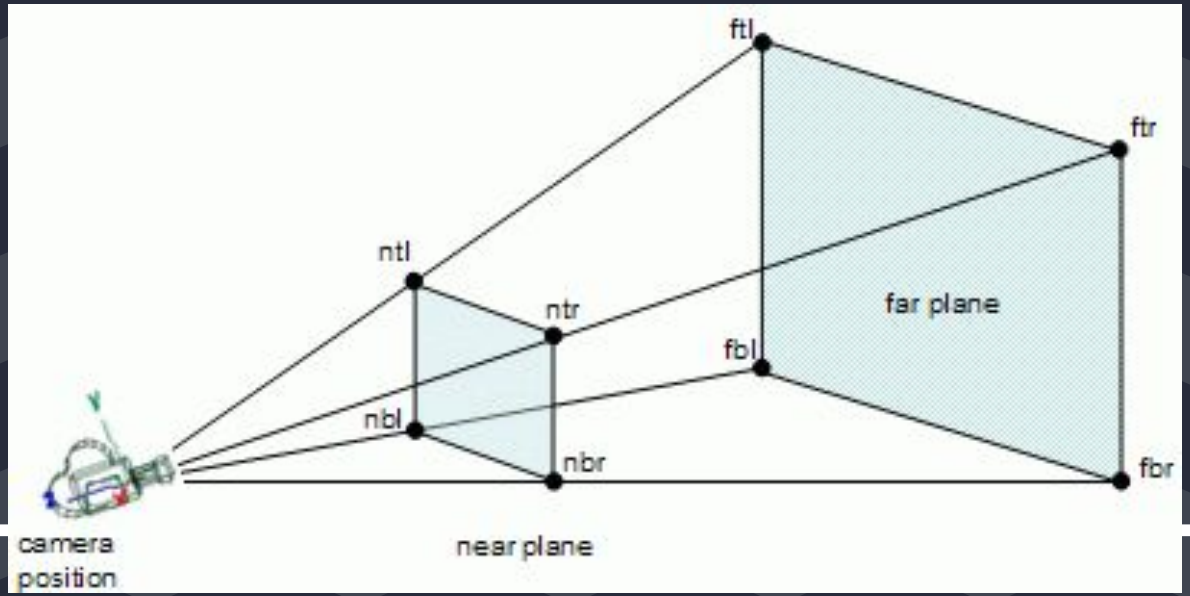
Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- How it works?
  - A point on each plane:
    - We want the 8 corners of view frustum
    - We only need to calculate two points (Chalkboard time).
    - The rest can be done by symmetry.

Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- How it works?
  - 

Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- How it works?
  - Recap: Camera space coordinate
    - +y axis: normalize(cam_up)
    - +z axis: -cam_forward = -normalize(cam_lookat - cam_pos)
    - +x axis: normalize(cross(cam_up,+z axis))
  - Recall: Window::P = glm::perspective(fovy, aspect, near, far);

Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- How it works?
  - A point on each plane:
    - We want the 8 corners of view frustum
    - We only need to calculate two points.
      - Ex. $$\begin{aligned} ntl &= (-near \cdot tan(\frac{fovx}{2}), near \cdot tan(\frac{fovy}{2}), -near) \\ &= (-aspect \cdot height, near \cdot tan(\frac{fovy}{2}), -near) \end{aligned}$$
    - The rest can be done by symmetry.

Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- How it works?
  - Recap: To do culling, we actually want to calculate point-to-plane distance.
  - Therefore, we need following information in camera space:
    - ~~Point coordinate~~
    - ~~A point on each plane~~
    - Plane normal of each plane

Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- How it works?
  - Plane normal of each plane:
    - We just calculated the points on each plane.
    - Assume P_lt, P_rt, P_lb, P_rb are four corners of a plane (Chalkboard time)

Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- How it works?
  - Back to point-to-plane distance formula, $dist(x) = (x - p) \cdot n$
  - Pseudocode:

Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- How it works?

$$dist(x) = (x - p) \cdot n$$

```
vector<pair<vec3, vec3>> planes;          // planes saves p and n for each plane.
void Init(){
    Calculate two corners;                // this two lines only need to be done once,
    Get the rest 6 corners by symmetry;   // unless you change projection matrix later
    for(int i = 0; i < 6; i++){
        vec3 p_lt, p_rt, p_lb;            // pick three corner on that plane

        vec3 v_1 = p_rt - p_lt;           // two linear independent vectors on the plane
        vec3 v_2 = p_lb - p_lt;

        vec3 normal = normalize(cross(v_2, v_1));    // get normal vector pointing inward.

        planes.push_back(make_pair(p_rt, normal));
    }
}
```

Halloween Theme is from CSE167 F16 Discussion 6 slide

View space culling

- How it works?

$$dist(x) = (x - p) \cdot n$$

```
bool IsVisible(vec3 x, float r){
    for(auto plane : planes){
        vec3 p = plane->first;
        vec3 n = plane->second;          // Assume all n are pointing inward.

        float dist = dot((x - p), n);
        if(dist < -r){                   // -r because we still want to display it
            return false;                // if it's partially visible
        }
    }
    return true;
}
```

Halloween Theme is from CSE167 F16 Discussion 6 slide

Scene Graph

- Need to maintain a tree of nodes for the android army
  - Can be iterated through by calling draw() on the root
    - It recursively calls draw() on the rest of the tree
- Transform is only responsible for transformation, while Geometry is only responsible for render.

Halloween Theme is from CSE167 F16 Discussion 6 slide

Scene Graph

- Simply add transformations to animate your androids
  - All modifications to Transform should be in idle_callback()
    - (You don’t have to do this, but it makes your code cleaner)
- You should not have multiple copies of vertices in memory for the same geometry.
  - Don’t do this: having 3 * 6909 * 50000 glm::vec3 in memory for body, just because you want to render 50000 robots

Halloween Theme is from CSE167 F16 Discussion 6 slide

Texture/Change in OBJ files

- Make sure to parse as:
  - f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3
- And realign so that we can use a single std::vector of indices
- Make sure to use std::vector<glm::vec3> for positions/normals and std::vector<glm::vec2> for texture coordinates
  - For HW2 we could use std::vector<float> for positions/normals, since the memory layout is the same as std::vector<glm::vec3>

Halloween Theme is from CSE167 F16 Discussion 6 slide

Extra Credit

- Debug Mode
- Hierarchical Culling

Halloween Theme is from CSE167 F16 Discussion 6 slide

Announcements

- Happy Halloween!
- Project 3 due this Friday 11/02
- Feel free to get some candies before you leave

Halloween Theme is from CSE167 F16 Discussion 6 slide