Computer Graphics

October 22, 2019

Perspective Projection

- Near is supposed to be negative some value and far should also be negative some value
- The matrices given in the notes require that the near and far values be negative.

Animation in project 2

- 5 Different types of animation
  - fly around the scene, looking at center from above
  - fly down to start of maze
  - walk into maze and solve it
    - walk forward
    - turn left
    - turn right
- **Only change the lookat point and the eye points**
  - Walk forward, eye point and at point change and move forward together
  - turning, only at point changes

Animation

- Any callback function should execute as quickly as possible, so don't use a for loop
- Use idle function to redraw
  1. check the event queue
     - if there is an event in the queue, process it and go back to step 1
     - if no event, go to step 2
  2. Redraw the scene with new transformation matrices and go back to step 1
- Use a state variable:

```
typedef enum
{
    FLYING_AROUND = 0,
    FLYING_DOWN,
    WARK_FORWARD,
    TURN_LEFT,
    TURN_RIGHT,
} state;
state currentState = FLYING_AROUND;
```

```
void idle(void)
{
    if(isAnimating)
    {
        if(currentState == FLYING_AROUND) {
        :
        }
        else if(currentState == FLYING_DOWN) {
        :
        }
        else if(currentState == WALK_FORWARD) {
        :
```

```
        }
        else if(currentState == TURN_LEFT) {
        :
        }
        else if(currentState == TURN_RIGHT) {
        :
        }
    }
}

void idle(void)
{
    if(isAnimating) {
        current_step++;
        GLfloat alpha = (float) current_step / max_steps;
        if(current_step == max_steps) {
            isAnimation = 0;
            current_step = 0;
            max_step = 50;
        }
        else {
            current_something = (alpha * changing_vector) + starting;
            calculate new transformation(s)
        }
        glutPostRedisplay();
    }
}
```

Turning - eye point stays the same, you use the old AT point and the new AT' point, and you use the vector AT'-AT = V, **AT' = (alpha)V + AT**

- Keep a current_step global variable and a max_steps global variable for each of the different animation types
- alpha is current_step / max_steps