Doomish

Nick Polach

Labs

- Lab 1
  - Used as base code for game
  - Camera controls
- Lab 3
  - Steady physics rate
  - States using bit-fields
    - keep track of pressed keys
- Lab 5
  - Collision
    - Bullet/Enemy & Enemy/Player collision
- Lab 7
  - Audio
    - Gun & fireball sounds

Labs Cont.

- Lab 8
  - Billboarding
    - All sprites are billboarded
  - Blending
    - Blending used for damage tint
- Lab 12
  - FPS display
  - Enable/disable VSync
  - Full screen

Camera

- Problems:
  - Couldn't combine moves (couldn't move forward and left at same time)
  - Did not move correctly relative to the looking direction
- To fix it I had to totally rewrite the camera code

```
void moveBackward()
{
    float viewX = view[0] - pos[0];
    float viewZ = view[2] - pos[2];

    Flt newX = pos[0] - viewX * moveSpeed;
    Flt newZ = pos[2] - viewZ * moveSpeed;

    if (!wallCollide(newX, newZ)) {
        pos[0] = newX;
        pos[2] = newZ;

        view[0] -= viewX * moveSpeed;
        view[2] -= viewZ * moveSpeed;
    }
}

void moveLeft()
{

    float viewX = view[0] - pos[0];
    float viewY = view[1] - pos[1];
    float viewZ = view[2] - pos[2];

    float x = ((viewY * upv[2]) - (viewZ * upv[1]));
    float y = ((viewZ * upv[0]) - (viewX * upv[2]));
    float z = ((viewX * upv[1]) - (viewY * upv[0]));

    float magnitude = sqrt( (x * x) + (y * y) + (z * z) );

    x /= magnitude;
    y /= magnitude;
    z /= magnitude;

    Flt newX = pos[0] - x*strafeSpeed;
    Flt newZ = pos[2] - z*strafeSpeed;

    if (!wallCollide(newX, newZ)) {
        pos[0] = newX;
        pos[2] = newZ;

        view[0] -= x*strafeSpeed;
        view[2] -= z*strafeSpeed;
    }
}
```

```
void lookDown(Flt scale)
{
    angleV += lookSpeed*scale;
    if (angleV > 1.0)
        angleV = 1.0;

    view[1] = sin(angleV);
}

void lookLeft(Flt scale)
{
    angleH += lookSpeed*scale;
    view[0] = pos[0] + sin(angleH);
    view[2] = pos[2] + -cos(angleH);
}
```

Key Presses - Bit Field

```
// Masks for byte
// #7 = 0x80 = 1000 0000
// #6 = 0x40 = 0100 0000
// #5 = 0x20 = 0010 0000
// #4 = 0x10 = 0001 0000
//
// d = 0x08 = 0000 1000
// s = 0x04 = 0000 0100
// a = 0x02 = 0000 0010
// w = 0x01 = 0000 0001
// Using w, a, s, d, shift
key_states = 0x00;
w_mask = 0x01;
a_mask = 0x02;
s_mask = 0x04;
d_mask = 0x08;
```

Declare masks

```
// Player movement
if (g.key_states & g.w_mask) {
    if (g.player.pos[2] > -37.4f) {
        g.player.moveForward();
    }
}
if (g.key_states & g.a_mask) {
    if (g.player.pos[0] > -22.4f) {
        g.player.moveLeft();
    }
}
if (g.key_states & g.s_mask) {
    if (g.player.pos[2] < 7.4f) {
        g.player.moveBackward();
    }
}
if (g.key_states & g.d_mask) {
    if (g.player.pos[0] < 22.4f) {
        g.player.moveRight();
    }
}
```

Make move if bit enabled

```
if (e->type == KeyRelease) {
    switch(key) {
        case XK_w:
            g.key_states = g.key_states&~(g.w_mask);
            break;
        case XK_a:
            g.key_states = g.key_states&~(g.a_mask);
            break;
        case XK_s:
            g.key_states = g.key_states&~(g.s_mask);
            break;
        case XK_d:
            g.key_states = g.key_states&~(g.d_mask);
            break;
    }
}

if (e->type == KeyPress) {
    //Was there input from the keyboard?
    switch(key) {
        // Camera angle
        case XK_1:
            break;
        case XK_w:
            g.key_states = g.key_states|g.w_mask;
            break;
        case XK_a:
            g.key_states = g.key_states|g.a_mask;
            break;
        case XK_s:
            g.key_states = g.key_states|g.s_mask;
            break;
        case XK_d:
            g.key_states = g.key_states|g.d_mask;
            break;
```

Toggle bits

Billboarding

- Similar code from lab 8
- Sprites rotate left and right to face the player

- Problem:
  - Sprites would rotate up and down when player moved closer to them
  - Fix: All Y coordinates in the view matrix were commented out

```
///// Billboarding
//Setup camera rotation matrix
//
Vec v;
VecSub(g.brutes[i].pos, g.player.pos, v);
Vec z = {0.0f, 0.0f, 0.0f};
make_view_matrix(z, v, g.cameraMatrix);
//
//Billboard_to_camera();
//
float mat[16];
mat[ 0] = g.cameraMatrix[0][0];
mat[ 1] = g.cameraMatrix[0][1];
mat[ 2] = g.cameraMatrix[0][2];
mat[ 4] = g.cameraMatrix[1][0];
mat[ 5] = g.cameraMatrix[1][1];
mat[ 6] = g.cameraMatrix[1][2];
mat[ 8] = g.cameraMatrix[2][0];
mat[ 9] = g.cameraMatrix[2][1];
mat[10] = g.cameraMatrix[2][2];
mat[ 3] = mat[ 7] = mat[11] = mat[12] = mat[13] = mat[14] = 0.0f;
mat[15] = 1.0f;
glMultMatrixf(mat);
//
///// End Billboarding
```

Sprite Animation

```
// Sprite Animation
g.fireballs[i].timer.recordTime(&g.fireballs[i].timer.timeCurrent);
double timeSpan = g.fireballs[i].timer.timeDiff(&g.fireballs[i].timer.animTime, &g.fireballs[i].timer.timeCurrent);
if (timeSpan > g.fireballs[i].delay) {
    ++g.fireballs[i].spriteFrame;
    if(g.fireballs[i].spriteFrame >= g.fireballs[i].FRAMECOUNT)
        g.fireballs[i].spriteFrame = 0;
    g.fireballs[i].timer.recordTime(&g.fireballs[i].timer.animTime);
}
```


- Keep track of animation times
- Move forward a frame after a specified delay

```
float tx = g.fireballs[i].spriteFrame * .20;

glRotatef(90, 1, 0, 0);
glBegin(GL_QUADS);

glTexCoord2f(tx, 0.0f);
glVertex3f( w, h,-d);

glTexCoord2f(tx+.20, 0.0f);
glVertex3f(-w, h,-d);

glTexCoord2f(tx+.20, 1.0f);
glVertex3f(-w, h, d);

glTexCoord2f(tx, 1.0f);
glVertex3f( w, h, d);
```

- Use current frame number to calculate coord of current animation frame

Fireball/Bullet Movement (+Enemy Movement)

```
Flt speed = .25;

// Camera center - flier center
Vec v;
v[0] = g.player.pos[0] - x;
v[1] = g.player.pos[1] - y;
v[2] = g.player.pos[2] - z;

// Normalize vector
Vec norm = {v[0], v[1], v[2]};
vecNormalize(norm);

MakeVector(x, y, z, g.fireballs[g.nfireballs].pos);
MakeVector(norm[0]*speed, norm[1]*speed, norm[2]*speed, g.fireballs[g.nfireballs].dir);
```

- The direction towards the player is calculated by subtracting player position from the enemy
- Similar code is used to calculate bullet direction
- Direction is only calculated at fireball/bullet creation time

- Similar code is also used to calculate enemy movement
  - Enemy movement direction is calculated every frame

Collision

- Checks if fireball's (or brute's) position is within a box around the player
- If fireball (or enemy) is within that box, the player is damaged

```
// collision with player
if (g.fireballs[i].pos[0] < g.player.pos[0] + 1 && // right of player
        g.fireballs[i].pos[0] > g.player.pos[0] - 1 && // left of player
        g.fireballs[i].pos[1] < g.player.pos[1] + 3 && // top of player
        g.fireballs[i].pos[1] > g.player.pos[1] - 3 && // bottom of player
        g.fireballs[i].pos[2] < g.player.pos[2] + 1 && // front of player
        g.fireballs[i].pos[2] > g.player.pos[2] - 1)   // back of player
{
    g.player.health -= 10;
```