

Camera Controls

Recommended Reading

[Rendering Lights](#)

Video

http://www.leadwerks.com/files/Tutorials/CPP/Camera_Controls.wmv

Required Files

http://www.leadwerks.com/files/Tutorials/CPP/Camera_Controls_Files.zip

Getting Started

We start with this template program which creates a world, a camera, a render buffer, and a single directional light, then renders the world with lighting:

```
#include "engine.h"

int main(int argc, char** argv)
{
    Initialize();

    //Create a graphics context
    Graphics(800,600);

    //Create a world
    if (!CreateWorld()) {
        MessageBoxA(0,"Error","Failed to create world.",0);
        goto exitapp;
    }

    //Create a camera
    TEntity cam=CreateCamera();

    //Create a light
    TLight light=CreateDirectionalLight();
    RotateEntity(light,Vec3(45,45,0));

    //Create a render buffer
    TBuffer buffer=CreateBuffer(800,600,BUFFER_COLOR|BUFFER_DEPTH|BUFFER_NORMAL);

    //Main loop
    while(!KeyHit(KEY_ESCAPE)) {
```

```

        //Update the world
        UpdateWorld();

        //Render the scene
        SetBuffer(buffer);
        RenderWorld();

        //Render lighting
        SetBuffer(BackBuffer());
        RenderLights(buffer);

        //Swap the front and back buffer
        Flip();
    }

    exitapp:
    Terminate();
    return 0;
}

```

Loading a Mesh

We need to load a mesh to give us a point of reference before we add camera controls. Add this line to load a mesh before the main loop:

```
TMesh mesh=LoadMesh("scene.gmf");
```

We can make it so the program will display an error if the mesh is not found:

```

TMesh mesh=LoadMesh("scene.gmf");
if (mesh==0) {
    MessageBoxA(0,"Error","Failed to load mesh.",0);
    goto exitapp;
}

```

Let's reposition the camera so we can see the scene better. Add this code after the camera is created, before the main loop:

```
PositionEntity(cam,Vec3(0,2,-10));
```

Here is the complete code with the changes we have made:

```
#include "engine.h"

int main(int argc, char** argv)
{
    Initialize();

    //Create a graphics context
    Graphics(800,600);

    //Create a world
    if (!CreateWorld()) {
        MessageBoxA(0,"Error","Failed to create world.",0);
        goto exitapp;
    }

    //Create a camera
    TEntity cam=CreateCamera();
    PositionEntity(cam,Vec3(0,2,-10));

    //Load a mesh
    TMesh mesh=LoadMesh("scene.gmf");
    if (mesh==0) {
        MessageBoxA(0,"Error","Failed to load mesh.",0);
        goto exitapp;
    }

    //Create a light
    TLight light=CreateDirectionalLight();
    RotateEntity(light,Vec3(45,45,0));

    //Create a render buffer
    TBuffer buffer=CreateBuffer(800,600,BUFFER_COLOR|BUFFER_DEPTH|BUFFER_NORMAL);

    //Main loop
    while(!KeyHit(KEY_ESCAPE)) {

        //Update the world
        UpdateWorld();

        //Render the scene
        SetBuffer(buffer);
        RenderWorld();

        //Render lighting
        SetBuffer(BackBuffer());
        RenderLights(buffer);

        //Swap the front and back buffer
        Flip();
    }

    exitapp:
    Terminate();
    return 0;
}
```

If we run the program now we will see our loaded mesh. You must have the *scene.gmf* file in the same directory as your source code. You should also have all textures and materials in the same directory.

Adding Camera Controls

Now that we have a visible mesh loaded, let's add controls to navigate the scene. First we define the following variables before the main loop. These will be used to control camera rotation:

```
TVec3 camrotation=Vec3(0);  
float mx=0;  
float my=0;
```

Right before the main loop, move the mouse to the center of the screen with this code:

```
MoveMouse(GraphicsWidth()/2,GraphicsHeight()/2);
```

Now comes the main part. We are going to get the difference between the mouse position and the center of the screen each frame, then reset the mouse position to the center. Add this code inside the main loop:

```
mx=MouseX()-GraphicsWidth()/2;  
my=MouseY()-GraphicsHeight()/2;  
MoveMouse(GraphicsWidth()/2,GraphicsHeight()/2);
```

The variable *mx* indicates how far the user has moved the mouse horizontally, while the variable *my* tracks vertical mouse movement. Now we want to add the horizontal mouse movement to the camera yaw, and add the vertical mouse movement to the camera pitch. Add this code inside the main loop, after the previous block of code:

```
camrotation.X=camrotation.X+my;  
camrotation.Y=camrotation.Y+mx;  
RotateEntity(cam,camrotation);
```

Here is our final program:

```
#include "engine.h"

int main(int argc, char** argv)
{
    Initialize();

    //Create a graphics context
    Graphics(800,600);

    //Create a world
    if (!CreateWorld()) {
        MessageBoxA(0,"Error","Failed to create world.",0);
        goto exitapp;
    }

    //Create a camera
    TEntity cam=CreateCamera();
    PositionEntity(cam,Vec3(0,2,-10));

    //Load a mesh
    TMesh mesh=LoadMesh("scene.gmf");
    if (mesh==0) {
        MessageBoxA(0,"Error","Failed to load mesh.",0);
        goto exitapp;
    }

    //Create a light
    TLight light=CreateDirectionalLight();
    RotateEntity(light,Vec3(45,45,0));

    //Create a render buffer
    TBuffer buffer=CreateBuffer(800,600,BUFFER_COLOR|BUFFER_DEPTH|BUFFER_NORMAL);

    TVec3 camrotation=Vec3(0);
    float mx=0;
    float my=0;

    MoveMouse(GraphicsWidth()/2,GraphicsHeight()/2);

    //Main loop
    while(!KeyHit(KEY_ESCAPE)) {

        //Camera look
        mx=MouseX()-GraphicsWidth()/2;
        my=MouseY()-GraphicsHeight()/2;
        MoveMouse(GraphicsWidth()/2,GraphicsHeight()/2);

        camrotation.X=camrotation.X+my;
        camrotation.Y=camrotation.Y+mx;
        RotateEntity(cam,camrotation);

        //Update the world
        UpdateWorld();

        //Render the scene
        SetBuffer(buffer);
        RenderWorld();

        //Render lighting
```

```

        SetBuffer(BackBuffer());
        RenderLights(buffer);

        //Swap the front and back buffer
        Flip();
    }

    exitapp:
    Terminate();
    return 0;
}

```

Run the program and move the mouse to look around. You will notice that moving the mouse up looks up, moving it down looks down, moving it right looks left, and moving it left looks right. To fix this, we will subtract *mx* from the camera yaw instead of adding it:

```
camrotation.Y=camrotation.Y-mx;
```

Fine Tuning

The camera looking is fast and jerky. We want it to be slower and smoother. We can achieve this by dividing the mouse movement variables with a damping value as follows:

```
camrotation.X=camrotation.X+my/10.0;
camrotation.Y=camrotation.Y-mx/10.0;
```

We can smooth the motion by using the Curve function. This is not a part of the Leadwerks Engine library, but it is declared in the C header, and can be easily reproduced in any programming language. The source code for the Curve function is as follows:

```

flt Curve(flt newvalue, flt oldvalue, flt increments)
{
    int sign;
    flt slip;
    sign=sgn(oldvalue-newvalue);
    slip=(oldvalue-newvalue)/increments;
    oldvalue-=slip;
    if(sign!=sgn(oldvalue-newvalue)) return(newvalue);
    return(oldvalue);
}

```

The Curve function accepts a destination value, a current value, and divides the difference by a smoothing value. Here, we use the mouse movement for the destination value. Our current movement value is the second parameter. A smoothing value of 6 is used:

```
mx=Curve(MouseX()-GraphicsWidth()/2,mx,6);
my=Curve(MouseY()-GraphicsHeight()/2,my,6);
```

A higher smoothing value will result in even smoother motion, while a lower smoothing value will make the camera more responsive.

Here is our final code:

```
#include "engine.h"

int main(int argc, char** argv)
{
    Initialize();

    //Create a graphics context
    Graphics(800,600);

    //Create a world
    if (!CreateWorld()) {
        MessageBoxA(0,"Error","Failed to create world.",0);
        goto exitapp;
    }

    //Create a camera
    TEntity cam=CreateCamera();
    PositionEntity(cam,Vec3(0,2,-10));

    //Load a mesh
    TMesh mesh=LoadMesh("scene.gmf");
    if (mesh==0) {
        MessageBoxA(0,"Error","Failed to load mesh.",0);
        goto exitapp;
    }

    //Create a light
    TLight light=CreateDirectionalLight();
    RotateEntity(light,Vec3(45,45,0));

    //Create a render buffer
    TBuffer buffer=CreateBuffer(800,600,BUFFER_COLOR|BUFFER_DEPTH|BUFFER_NORMAL);

    TVec3 camrotation=Vec3(0);
    float mx=0;
    float my=0;

    MoveMouse(GraphicsWidth()/2,GraphicsHeight()/2);

    //Main loop
    while(!KeyHit(KEY_ESCAPE)) {

        //Camera look
```

```

mx=Curve(MouseX()-GraphicsWidth()/2,mx,6);
my=Curve(MouseY()-GraphicsHeight()/2,my,6);
MoveMouse(GraphicsWidth()/2,GraphicsHeight()/2);

camrotation.X=camrotation.X+my/10.0;
camrotation.Y=camrotation.Y-mx/10.0;
RotateEntity(cam,camrotation);

//Update the world
UpdateWorld();

//Render the scene
SetBuffer(buffer);
RenderWorld();

//Render lighting
SetBuffer(BackBuffer());
RenderLights(buffer);

//Swap the front and back buffer
Flip();
}

exitapp:
Terminate();
return 0;
}

```

You can see the camera looking is much smoother and easier to control now.

Adding Motion

We can add camera movement by using the keyboard input to control the camera. We'll call the variable for forwards and backwards motion *move* and the variable for left and right motion *strafe*. Add these variables below the other camera variables we declared, before the main loop:

```

float move=0;
float strafe=0;

```

Now let's add some code in the main loop to control motion. Here we see if the W key is pressed, the move value will be 1. If the S key is pressed the move value will be 0 minus 1, or -1. If both the W and S key are pressed, the move value will be 0, as it should be:

```

move=KeyDown(KEY_W)-KeyDown(KEY_S);
strafe=KeyDown(KEY_D)-KeyDown(KEY_A);

```

Then we simply move the camera with a vector using the move and strafe values:

```
MoveEntity(cam, Vec3(strafe, 0, move));
```

We can smooth motion the same way we smoothed the rotation. A smoothing value of 20 will give very smooth motion:

```
move=Curve(KeyDown(KEY_W)-KeyDown(KEY_S), move, 20);
strafe=Curve(KeyDown(KEY_D)-KeyDown(KEY_A), strafe, 20);
MoveEntity(cam, Vec3(strafe/10.0, 0, move/10.0));
```

Finally, we are going to hide the mouse cursor by adding a call to HideMouse before the main loop:

```
HideMouse();
```

Here is our finished code with smooth camera movement:

```
#include "engine.h"

int main(int argc, char** argv)
{
    Initialize();

    //Create a graphics context
    Graphics(800, 600);

    //Create a world
    if (!CreateWorld()) {
        MessageBoxA(0, "Error", "Failed to create world.", 0);
        goto exitapp;
    }

    //Create a camera
    TEntity cam=CreateCamera();
    PositionEntity(cam, Vec3(0, 2, -10));

    //Load a mesh
    TMesh mesh=LoadMesh("scene.gmf");
    if (mesh==0) {
        MessageBoxA(0, "Error", "Failed to load mesh.", 0);
        goto exitapp;
    }

    //Create a light
    TLight light=CreateDirectionalLight();
    RotateEntity(light, Vec3(45, 45, 0));

    //Create a render buffer
```

```

TBuffer buffer=CreateBuffer(800,600,BUFFER_COLOR|BUFFER_DEPTH|BUFFER_NORMAL);

TVec3 camrotation=Vec3(0);
float mx=0;
float my=0;
float move=0;
float strafe=0;

HideMouse();
MoveMouse(GraphicsWidth()/2,GraphicsHeight()/2);

//Main loop
while(!KeyHit(KEY_ESCAPE)) {

    //Camera look
    mx=Curve(MouseX()-GraphicsWidth()/2,mx,6);
    my=Curve(MouseY()-GraphicsHeight()/2,my,6);
    MoveMouse(GraphicsWidth()/2,GraphicsHeight()/2);

    camrotation.X=camrotation.X+my/10.0;
    camrotation.Y=camrotation.Y-mx/10.0;
    RotateEntity(cam,camrotation);

    //Camera movement
    move=Curve(KeyDown(KEY_W)-KeyDown(KEY_S),move,20);
    strafe=Curve(KeyDown(KEY_D)-KeyDown(KEY_A),strafe,20);
    MoveEntity(cam,Vec3(strafe/10.0,0,move/10.0));

    //Update the world
    UpdateWorld();

    //Render the scene
    SetBuffer(buffer);
    RenderWorld();

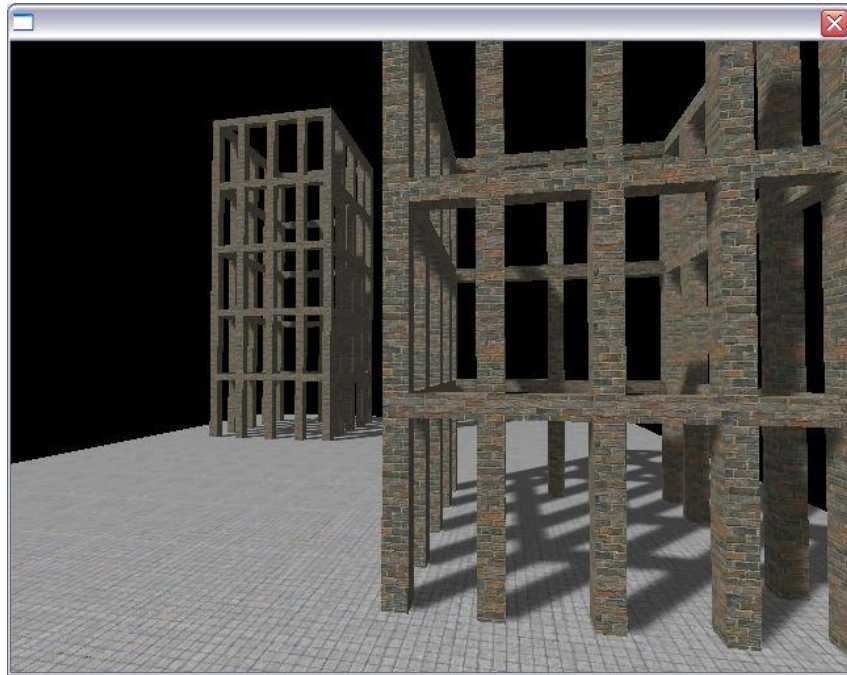
    //Render lighting
    SetBuffer(BackBuffer());
    RenderLights(buffer);

    //Swap the front and back buffer
    Flip();
}

exitapp:
Terminate();
return 0;
}

```

Now you can navigate the scene properly.



Copyright Leadwerks Software 2008

All Rights Reserved.