

# Introduction to Bodies

## Recommended Reading

[Rendering Lights](#)

[Introduction to Meshes](#)

## Video

[http://www.leadwerks.com/files/Tutorials/CPP/Introduction\\_To\\_Bodies.wmv](http://www.leadwerks.com/files/Tutorials/CPP/Introduction_To_Bodies.wmv)

## Getting Started

We start with a simple template program:

```
#include "engine.h"

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

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

    //Load SDK directory
    RegisterAbstractPath("C:\\Leadwerks Engine SDK");

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

    //Create a camera
    TCamera cam=CreateCamera();
    CameraClearColor(cam, Vec4(0,0,1,1));
    MoveEntity(cam, Vec3(0,0,-5));

    //Create a light
    TLight light=CreateDirectionalLight();
    RotateEntity(light, Vec3(65,45,0));

    //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:
    return Terminate();
}

```

Before the main loop, add this code:

```

TBody body=CreateBodyBox();
SetBodyMass(body,1);

```

Now anywhere in the program add this line. This will enable wireframe physics visualization:

```

DebugPhysics(1);

```

Run the program and you will see a wireframe box that falls off the screen.

## Adding Collision

Let's create another body to act as the ground. Add this code before the main loop:

```

TBody ground=CreateBodyBox(10,0.1,10);
PositionEntity(ground,Vec3(0,-1,0));

```

If we run the program as it is right now, the box will fall right through the ground. We have to enable collisions. Add this code right before the main loop, after the two bodies have been created. This tells the engine that both the body and ground are collision types one, and when two bodies of collision type one intersect, a collision should occur:

```
Collisions(1,1,1);
EntityType(body,1);
EntityType(ground,1);
```

Our complete program now looks like this:

```
#include "engine.h"

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

    //Create a graphics context
    Graphics(640,480);

    //Load SDK directory
    RegisterAbstractPath("C:\\Leadwerks Engine SDK");

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

    //Create a camera
    TCamera cam=CreateCamera();
    CameraClearColor(cam, Vec4(0,0,1,1));
    MoveEntity(cam, Vec3(0,0,-5));

    //Create a light
    TLight light=CreateDirectionalLight();
    RotateEntity(light, Vec3(65,45,0));

    //Create a render buffer
    TBuffer buffer=CreateBuffer(640,480, BUFFER_COLOR|BUFFER_DEPTH|BUFFER_NORMAL);

    //Create a physics body
    TBody body=CreateBodyBox();
    SetBodyMass(body,1);

    //Enable physics visualization
    DebugPhysics(1);

    //Create the ground
    TBody ground=CreateBodyBox(10,0.1,10);
    PositionEntity(ground, Vec3(0,-1,0));

    //Enable collisions
    Collisions(1,1,1);
    EntityType(body,1);
    EntityType(ground,1);

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

        //Update the world
        UpdateWorld();
    }
}
```

```

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

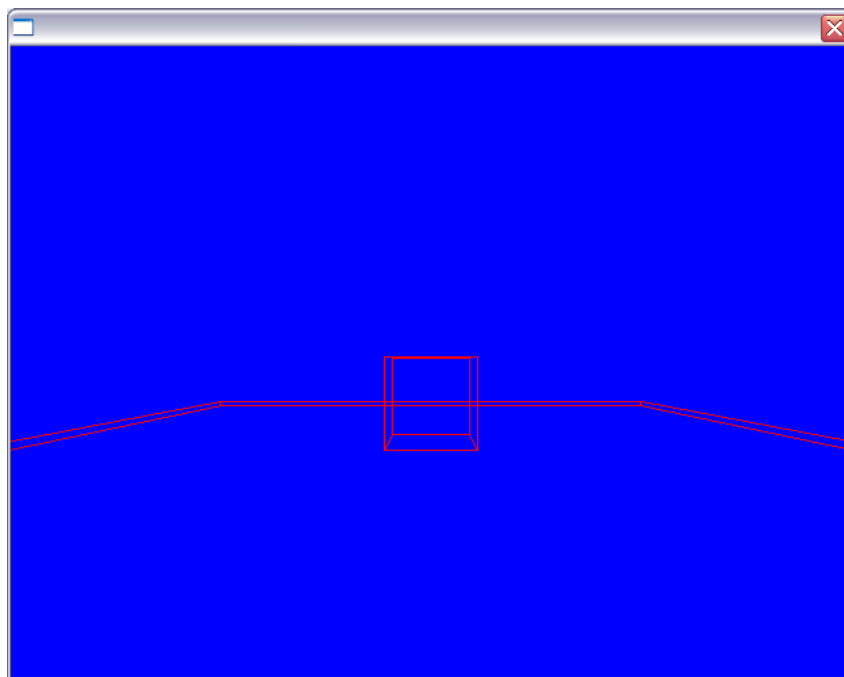
        //Render lighting
        SetBuffer(BackBuffer());
        RenderLights(buffer);

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

    exitapp:
    return Terminate();
}

```

When we run the program, the box will fall to the ground and stop.



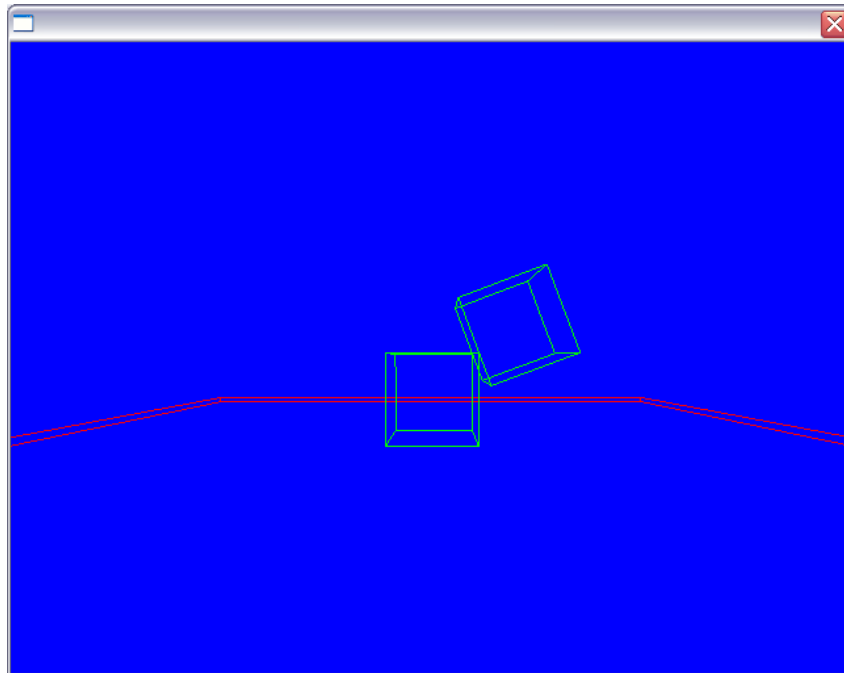
If we add a second box, we can see the two interact. Add this code before the main loop:

```

TBody body2=CreateBodyBox();
SetBodyMass(body2,1);
PositionEntity(body2,Vec3(0.5,2,0));
EntityType(body2,1);

```

One box will fall onto the other and topple over.



### Adding Visible Meshes

Of course we don't want to play a game in wireframe, so we are going to add a visual mesh to our physics bodies. Here is our code we used to make the first body:

```
TBody body=CreateBodyBox();  
SetBodyMass(body,1);
```

Replace it with this code, which will create a cube mesh and parent it to the body. When an entity is parented to another entity, the child entity will move and rotate wherever the parent entity goes.

```
TBody body=CreateBodyBox();  
SetBodyMass(body,1);  
TMesh mesh=CreateCube();  
EntityParent(mesh,body);
```

Here is our code we used to create the second body:

```
TBody body2=CreateBodyBox ();
SetBodyMass (body2,1);
PositionEntity (body2,Vec3 (0.5,2,0));
EntityType (body2,1);
```

Replace it with this. Note that EntityParent is called before the box is positioned, while it is still at position 0,0,0. If we parented the mesh to the body after the body was positioned, the mesh's position relative to the body would be offset, and the mesh would not appear to be colliding correctly.

```
TBody body2=CreateBodyBox ();
SetBodyMass (body2,1);
TMesh mesh2=CreateCube ();
EntityParent (mesh2,body2);
PositionEntity (body2,Vec3 (0.5,2,0));
EntityType (body2,1);
```

To make the ground, we create a cube mesh and scale it. Again, we have to parent the mesh to the body before the body is positioned:

```
TBody ground=CreateBodyBox (10,0.1,10);
TMesh groundmesh=CreateCube ();
ScaleEntity (groundmesh,Vec3 (10,0.1,10));
EntityParent (groundmesh,ground);
PositionEntity (ground,Vec3 (0,-1,0));
```

Here is our final code:

```
#include "engine.h"

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

    //Create a graphics context
    Graphics (640,480);

    //Load SDK directory
    RegisterAbstractPath ("C:\Leadwerks Engine SDK");

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

```

//Create a camera
TCamera cam=CreateCamera();
CameraClearColor(cam,Vec4(0,0,1,1));
MoveEntity(cam,Vec3(0,0,-5));

//Create a light
TLight light=CreateDirectionalLight();
RotateEntity(light,Vec3(65,45,0));

//Create a render buffer
TBuffer buffer=CreateBuffer(640,480,BUFFER_COLOR|BUFFER_DEPTH|BUFFER_NORMAL);

//Create a physics body
TBody body=CreateBodyBox();
SetBodyMass(body,1);
TMesh mesh=CreateCube();
EntityParent(mesh,body);

//DebugPhysics(1);

//Create the ground
TBody ground=CreateBodyBox(10,0.1,10);
TMesh groundmesh=CreateCube();
ScaleEntity(groundmesh,Vec3(10,0.1,10));
EntityParent(groundmesh,ground);
PositionEntity(ground,Vec3(0,-1,0));

//Enable collision
Collisions(1,1,1);
EntityType(body,1);
EntityType(ground,1);

//Create another body
TBody body2=CreateBodyBox();
SetBodyMass(body2,1);
TMesh mesh2=CreateCube();
EntityParent(mesh2,body2);
PositionEntity(body2,Vec3(0.5,2,0));
EntityType(body2,1);

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

    //Update the world
    UpdateWorld();

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

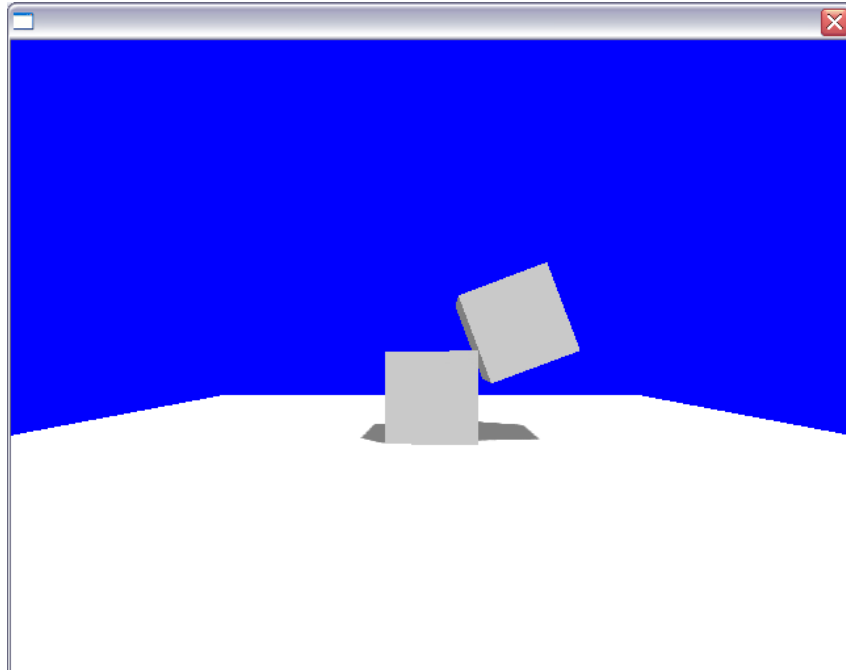
    //Render lighting
    SetBuffer(BackBuffer());
    RenderLights(buffer);

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

exitapp:
return Terminate();
}

```

Here we can see that meshes are visible, and they conform to the shape of the physics bodies:



### Expanding our Knowledge

Now let's have some fun. Add this code right before the main loop. What happens when we run the program?:

```
PositionEntity(cam, Vec3(0, 2, -10));  
  
float x;  
float z;  
for ( int n=1 ; n<=100; n++ )  
{  
    body = CopyEntity( body ) ;  
    x = -5+((float)rand()/RAND_MAX)*10;  
    z = -5+((float)rand()/RAND_MAX)*10;  
    PositionEntity( body, Vec3(x, n, z) );  
}
```

Copyright Leadwerks Software 2008

All Rights Reserved.