Google Analytics

Readability

BuySellAds.com

Please Login with:

Powered by LoginRadius

- Youtube
- Twitter
- Email
- RSS

Jump

- Home
- Blog
- Parts Store
- Our Apps
- Our Customers
- Forum
- Jobs

Physics 101 – UIKit app with Box2D for Gravity

May 03, 2010

7

Personally I was most at the edge of my seat at the “Voices that Matters Conference” in Seattle when Rod Strougo showed us how to make a physics-enabled game with Cocos2D and Box2D in under an hour. It really was as sexy as he sounds. Eros did what every good TV-cook would do, he had most things already prepared. On the flight home I wanted to see if I could just take the physics part (without Cocos2D) and make a UIKit app with it.

The goal of this experiment was to have a UIView with multiple square subviews of different sizes that would start falling as soon as the app starts. The first tricky part is how to add the latest version of the Box2D physics engine to your iPhone project. Then we need to mediate between the different units and coordinate systems of UIKit and Box2D. Finally when we got it all running, we also want to add the current gravity vector to affect the boxes.

Label

Readability

Read: NOW LATER

So, to get started we need to get the latest version of Box2D. It’s also included in Cocos2D, but for this tutorial we don’t care about getting the entire Cocos2D project. **Box2D has a site on Google Code** which tells us how to check out a read-only-copy of Box2D.

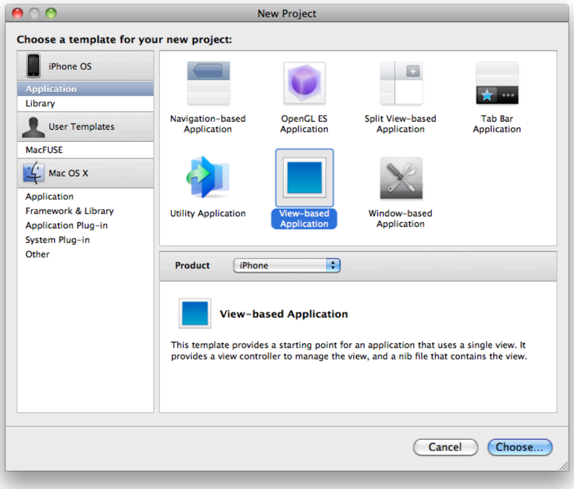
Adding Box2D

So we start a terminal, cd into a suitable directory and use this svn command to check out:

```
svn checkout http://box2d.googlecode.com/svn/trunk/ box2d-read-only
```

In the Box2D/Box2D subfolder of the folder this command creates you find all the C++ files that make up the engine.

Next we'll create a new View-based iPhone app, I'm calling mine UIKitPhysics.

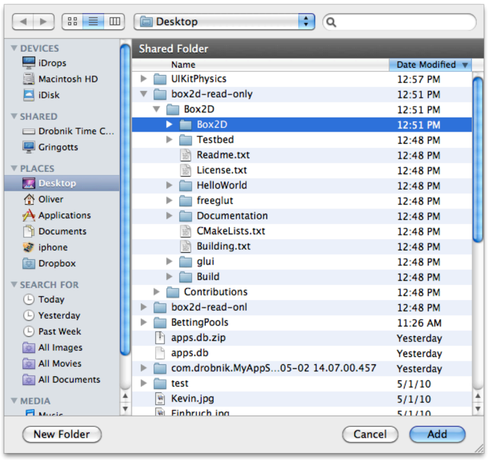


Product: iPhone

View-based Application

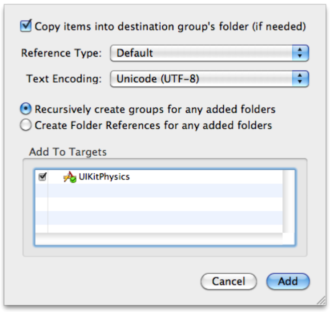
This template provides a starting point for an application that uses a single view. It provides a view controller to manage the view, and a nib file that contains the view.

Now choose Add – Existing Files. Navigate to the innermost Box2D folder of the project, it's 2 levels deep, and add it to your project.



Name	Date Modified
UIKitPhysics	12:57 PM
box2d-read-only	12:51 PM
Box2D	12:51 PM
Box2D	12:51 PM
Testbed	12:48 PM
Readme.txt	12:48 PM
License.txt	12:48 PM
HelloWorld	12:48 PM
freeglut	12:48 PM
Documentation	12:48 PM
CMakeLists.txt	12:48 PM
Building.txt	12:48 PM
glui	12:48 PM
Build	12:48 PM
Contributions	12:48 PM
box2d-read-onl	12:48 PM
BettingPools	11:26 AM
apps.db.zip	Yesterday
apps.db	Yesterday
com.drobnik.MyAppS...05-02 14.07.00.457	Yesterday
test	5/1/10
Kevin.jpg	5/1/10
Einbruch.jpg	5/1/10

If you want to be able to update it from the repository in the future, you need to not copy it to your project. If you don't care about updating then you can copy the files. I choose to copy it because it makes setting of the header search path easier later on.



☑ Copy items into destination group's folder (if needed)

Reference Type: Default

Text Encoding: Unicode (UTF-8)

● Recursively create groups for any added folders
○ Create Folder References for any added folders

Add To Targets

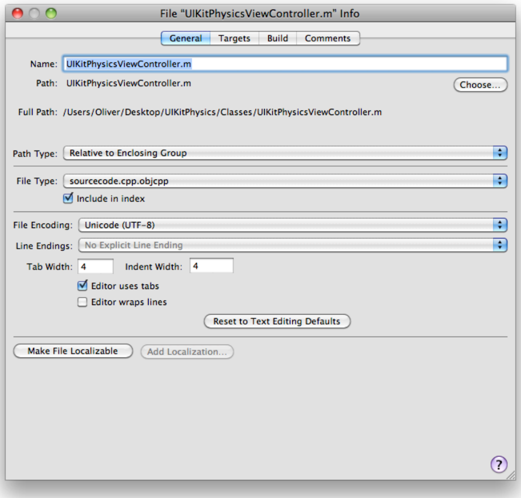
☑ UIKitPhysics

So now we have a Box2D group in our project. This is one way of adding it, the other would be to build a static library in Box2D and add this, but the above mentioned method looks easy enough for our first physics-enabled app.

There are still a couple more tweaks necessary to get it all working properly.

Box2D is written entirely in C++, but we are used to be working in Objective-C. Xcode usually uses the file extension to discern which compiler to use. Now if we are going to uses some function calls into Box2D those will be in C++ syntax. So we have two options to get the compiler to accept these as valid syntax. Either you modify the file extension to .mm or you force the compiler directly to interpret the affected files as C++. I think the latter is cleaner.

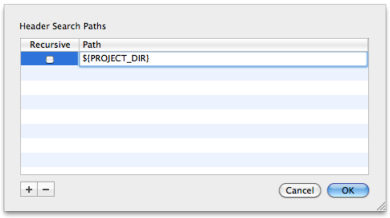
This tweak is necessary for all .m files which use C++ syntax and/or include the Box2D header. Since the default project probably has an include of the our view controller's header in the app delegate, you need to make the same adjustment for the app delegate's .m file. Get Info on the those files and change the file type to "sourcecode.cpp.objcpp". This does not hurt, your normal Objective-C code can still be compiled normally.



Name: UIKitPhysicsViewController.m
Path: UIKitPhysicsViewController.m
Full Path: /Users/Oliver/Desktop/UIKitPhysics/Classes/UIKitPhysicsViewController.m
Path Type: Relative to Enclosing Group
File Type: sourcecode.cpp.objcpp
☑ Include in index
File Encoding: Unicode (UTF-8)
Line Endings: No Explicit Line Ending
Tab Width: 4 Indent Width: 4
☑ Editor uses tabs
□ Editor wraps lines

Box2D uses angle brackets for all imports. So as the last step we need to add the location where the Box2D directory resides to the header search path. This can be done in the Project Settings or Target Settings. Since we want it to be in effect for all targets it's preferable to set it in the Project Settings. Go to Project – Edit Project Settings, look for "Header Search Paths" and add this setting.

I'm using the environment variable ${PROJECT_DIR} so that it always works regardless where our project currently resides. Should you have Box2D outside of the project then you need to add the full path at this stage.



Recursive	Path
□	${PROJECT_DIR}

Now with this setting any #import with angle brackets will also work if the Box2D folder is inside your project root directory.

UIKit, meet Mr. Box2D

Now we are ready to add physical behaviors to our UIKit app. But first a couple of tidbits to understand the differences between those two.

Any two-dimensional object in Box2D has it's own representation which is completely independent from how it is drawn. Nor does Box2D draw anything nor does it move anything. Therefore we have to create a physical representation for each view that we want to attach physics to. We have to set a timer to poll where the physical object have moved and rotated to at set intervals and update our UIKit views accordingly.

Another difference is in the coordinate system used. UIKit has the (0,0) point in the upper left, Box2D in the lower left corner. UIKit uses pixels as measurement, Box2D meters. So we have to find a good value of how many pixels should correspond to one meter in our physical model. I chose 16 pixels to make up a meter, and defined it at the top of our view controller.

```
#define PTM_RATIO 16
```

The model in which all physical calculations occur is called the "world". We also need a timer to update our views frequently, so we'll define two instance variables in our view controller.

```
#import <Box2D/Box2D.h>

@interface UIKitPhysicsViewController : UIViewController
{
        b2World* world;
        NSTimer *tickTimer;
}
```

And the appropriate dealloc in the implementation.

```
- (void)dealloc
{
        [tickTimer invalidate], tickTimer = nil;
        [super dealloc];
}
```

Next we implement a method to set up this world and define a couple of boundaries on all sides so that our objects won't fall out.

```
-(void)createPhysicsWorld
{
        CGSize screenSize = self.view.bounds.size;

        // Define the gravity vector.
        b2Vec2 gravity;
        gravity.Set(0.0f, -9.81f);

        // Do we want to let bodies sleep?
        // This will speed up the physics simulation
        bool doSleep = true;

        // Construct a world object, which will hold and simulate the rigid bodies.
        world = new b2World(gravity, doSleep);

        world->SetContinuousPhysics(true);

        // Define the ground body.
        b2BodyDef groundBodyDef;
        groundBodyDef.position.Set(0, 0); // bottom-left corner

        // Call the body factory which allocates memory for the ground body
        // from a pool and creates the ground box shape (also from a pool).
        // The body is also added to the world.
        b2Body* groundBody = world->CreateBody(&groundBodyDef);

        // Define the ground box shape.
        b2EdgeShape groundBox;

        // bottom
        groundBox.Set(b2Vec2(0,0), b2Vec2(screenSize.width/PTM_RATIO,0));
        groundBody->CreateFixture(&groundBox, 0);

        // top
        groundBox.Set(b2Vec2(0,screenSize.height/PTM_RATIO), b2Vec2(screenSize.width/PTM_RATIO,screenSize.height/PTM_RAT
        groundBody->CreateFixture(&groundBox, 0);

        // left
        groundBox.Set(b2Vec2(0,screenSize.height/PTM_RATIO), b2Vec2(0,0));
        groundBody->CreateFixture(&groundBox, 0);

        // right
        groundBox.Set(b2Vec2(screenSize.width/PTM_RATIO,screenSize.height/PTM_RATIO), b2Vec2(screenSize.width/PTM_RATIO,
        groundBody->CreateFixture(&groundBox, 0);
}
```

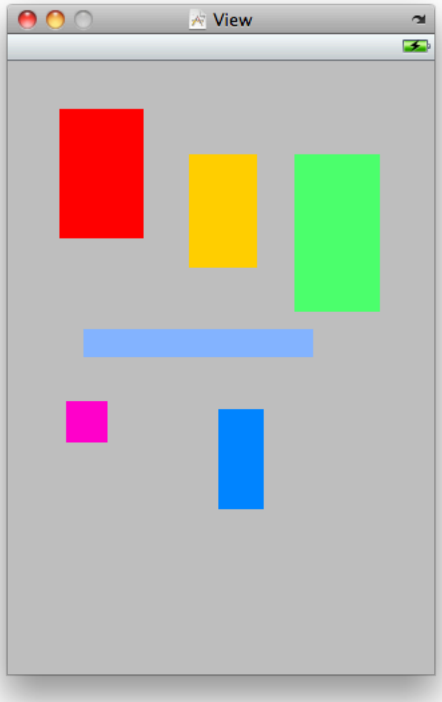
We create a helper method that finds out the necessary parameters for a view we pass to it to create a representation in the physical world. Note that we use the center of views for their position because as soon as we apply a transform (which we need for the rotation) the frame position and size become unusable. Luckily the center property can be used whatever the transform is.

```
-(void)addPhysicalBodyForView:(UIView *)physicalView
{
        // Define the dynamic body.
        b2BodyDef bodyDef;
        bodyDef.type = b2_dynamicBody;

        CGPoint p = physicalView.center;
        CGPoint boxDimensions = CGPointMake(physicalView.bounds.size.width/PTM_RATIO/2.0,physicalView.bounds.size.height

        bodyDef.position.Set(p.x/PTM_RATIO, (460.0 - p.y)/PTM_RATIO);
        bodyDef.userData = physicalView;

        // Tell the physics world to create the body
        b2Body *body = world->CreateBody(&bodyDef);

        // Define another box shape for our dynamic body.
        b2PolygonShape dynamicBox;

        dynamicBox.SetAsBox(boxDimensions.x, boxDimensions.y);

        // Define the dynamic body fixture.
        b2FixtureDef fixtureDef;
        fixtureDef.shape = &dynamicBox;
        fixtureDef.density = 3.0f;
        fixtureDef.friction = 0.3f;
        fixtureDef.restitution = 0.5f; // 0 is a lead ball, 1 is a super bouncy ball
        body->CreateFixture(&fixtureDef);

        // a dynamic body reacts to forces right away
        body->SetType(b2_dynamicBody);

        // we abuse the tag property as pointer to the physical body
        physicalView.tag = (int)body;
}
```

In the view controller's viewDidLoad method we create our world, call the addPhysicalBodyForView for each subview of our view and finally start the update timer.

```
- (void)viewDidLoad
{
        [super viewDidLoad];

        [self createPhysicsWorld];

        for (UIView *oneView in self.view.subviews)
        {
                [self addPhysicalBodyForView:oneView];
        }

        tickTimer = [NSTimer scheduledTimerWithTimeInterval:1.0/60.0 target:self selector:@selector(tick:) userInfo:nil
}
```

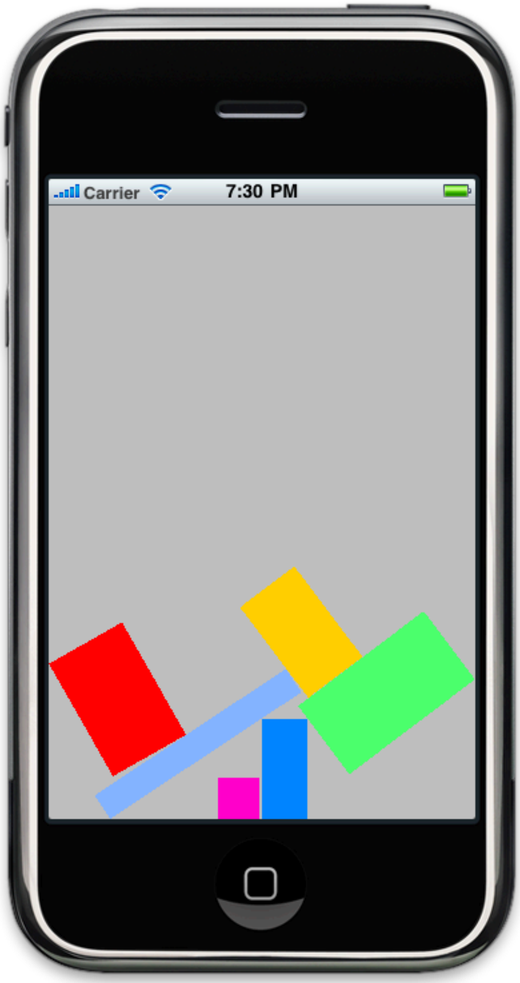
So that we have something to actually make physical let's go into Interface Builder and create a couple of subviews in UIKitPhysicsViewController.xib.



Finally, we need a timer function to update the position and rotation of our views.

```
-(void) tick:(NSTimer *)timer
{
        //It is recommended that a fixed time step is used with Box2D for stability
        //of the simulation, however, we are using a variable time step here.
        //You need to make an informed choice, the following URL is useful
        //http://gafferongames.com/game-physics/fix-your-timestep/

        int32 velocityIterations = 8;
        int32 positionIterations = 1;

        // Instruct the world to perform a single step of simulation. It is
        // generally best to keep the time step and iterations fixed.
        world->Step(1.0f/60.0f, velocityIterations, positionIterations);

        //Iterate over the bodies in the physics world
        for (b2Body* b = world->GetBodyList(); b; b = b->GetNext())
        {
                if (b->GetUserData() != NULL)
                {
                        UIView *oneView = (UIView *)b->GetUserData();

                        // y Position subtracted because of flipped coordinate system
                        CGPoint newCenter = CGPointMake(b->GetPosition().x * PTM_RATIO,
                                                        self.view.bounds.size.height - b->GetPosition().y * PTM_RATIO);
                        oneView.center = newCenter;

                        CGAffineTransform transform = CGAffineTransformMakeRotation(- b->GetAngle());

                        oneView.transform = transform;
                }
        }
}
```

Now we're set. If we start the app in simulator we see all views drop to the bottom and end up in a static heap. We remember from physics that all objects fall at the same rate in a vacuum, such is the case in this simulation. If we wanted to have air resistance to play a role we probably could add additional forces to the objects in addition to the global gravity, but for sake of simplicity we'll be satisfied with this result for the time being.



BONUS: And Now with Acceleration

As bonus chapter of this article we want to add the capability to have real gravity affect our view. For this purpose we setup and start an accelerometer and set our view controller as the delegate. This we add at the end of the viewDidLoad method. To get around the warning that your class does not the UIAccelerometerDelegate protocol we add it in angle brackets to the interface declaration:

```
//Configure and start accelerometer
[[UIAccelerometer sharedAccelerometer] setUpdateInterval:(1.0 / 60.0)];
[[UIAccelerometer sharedAccelerometer] setDelegate:self];
```

And in the delegate method we simply change the gravity vector of the physical world.

```
- (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate:(UIAcceleration *)acceleration
{
        b2Vec2 gravity;
        gravity.Set( acceleration.x * 9.81,  acceleration.y * 9.81 );

        world->SetGravity(gravity);
}
```

One more addition is necessary to get this fully working. In our construction of the physical world we stated that we would allow objects to be sleeping. This lets the Box2D send objects to sleep if they don't move anymore and don't get hit by other objects. This makes perfect sense if gravity is always coming from the same direction. But since the gravity vector can now change up to 60 times per second we need to disable sleeping. Otherwise all objects will get stuck on one side if you hold the iPhone still long enough.

```
// Do we want to let bodies sleep?
// This will speed up the physics simulation
bool doSleep = false;
```

Note that the accelerometer only sends events on an a physical device. In simulator only initially set gravity vector takes effect. Another funny effect is if you hold the iPhone face up then you can even make the objects weightless.

Conclusion

With little work it is possible to add a fully fledged albeit two-dimensional physics engine that can take control over some UIViews and have them behave as if they where real physical objects. This should allow for a couple of simple games using physics to spice up the interaction.

For anything more you'll probably have to switch to using OpenGL for the drawing part or if that's too complicated use the fully fledged Cocos2D engine.

But let me say this: "I love Physics". so I'm really anxious to see what you can make of this tutorial. Show me what you got!

0

Share |

Categories: Recipes

← Back (Almost)
Meta-Calling Multi-Param Methods →

51 Comments »

1. *rlucient*
   May 11, 2010 | 2:28 pm

   Great Toturial !!! …

   😉 You should call it, UIKit App with Box2b for gravity under 30 minutes.

   Regards,
   Raul.

   Log in to reply

2. ***chrismiles.info/***
   May 18, 2010 | 3:26 am

   Excellent howto, thanks for posting this! Got the demo working without a problem, the kids love it!

   Log in to reply

3. *umer sufyan*
   May 24, 2010 | 1:21 pm

   Thanks for the tutorial i need to move an obstacle suppose ball along path (curve) how can i achieve this…in box2d for iphone.

   Log in to reply

4. ***brinkin.myopenid.com/***
   May 29, 2010 | 6:41 am

   Amazing tutorial.

   Cannot believe how easy it was to get a physics simulation going…

   Cheers,

Ben

Log in to reply

- ***drops***
  May 29, 2010 | 9:27 am

  Thank you kind sir. What did you make?

  Log in to reply

  - *Kris*
    December 6, 2011 | 7:11 pm

    Great tutorial. Now can you please post more Box2D with UIKit tutorials. All the tutorials out there deal with Box2D with Cocos2D.

    A tutorial on how to make the bodies drag-gable will be very useful.

    We're all looking up to you.

    I know you can do it! lol

    Thank you once again for a great tutorial.

    Log in to reply

5. *markysparky*
   May 30, 2010 | 12:14 pm

   Always loved physics sims and this tutorial was the first that i felt i understood what was going on. Really like the idea of using UIViews, as it is much easier to see the setup of the Box2D world without the complexity of OpenGL and/or other drawing frameworks.

   Tried it on my new iPad and i can foresee great things! Though for the iPad i changed the constant in (void)addPhysicalBodyForView:(UIView *)physicalView from 460.0 to self.view.bounds.size.height
   Also added a couple of small UIImageViews, and as they are a sub-class of UIView the iterator picks them up and adds them to the world. So cool. Now to find something sensible to use this for…

   Many thanks for this great tutorial,
   Mark

   Log in to reply

6. *sparebrother*
   June 20, 2010 | 4:47 pm

   I just followed this tutorial to the letter (Box2D_v2.1.2, Xcode 3.2.2), but Xcode cannot find the Box2D include files.

   I added the search path ${PROJECT_DIR}, and copied/added all the Box2D files just like in your example.

   The first error of the 2241 compile errors is:

   Box2D/Collision/b2BroadPhase.h: No such file or directory

   Any ideas?

   Log in to reply

   - ***drops***
     June 22, 2010 | 3:58 pm

     You probably did not copy the Box2D folder to the correct location. All other commenting users had no problem getting it to work.

     Log in to reply

   - ***pocjoc***
     February 14, 2011 | 11:48 pm

Hi,

probably you've solved the issue, but for the other people that have this problem, one thing to control is to ensure that there are no spaces in the path, if so, then put the path between ".

For instance, if your project dir is Users/xcode projects/project, this will fail with that error and you must put:
"${PROJECT_DIR}"

Log in to reply

7. *sparebrother*
June 24, 2010 | 8:14 am

Ok, I found the problem now, I just linked in the box2d files, but seems you really need to copy them. I thought xcode would be clever enough to find files you actually add to the project without spelling their location out in the search path in addition.

Perhaps you should point out copying box2d to the project folder is not optional to make the tutorial work, for people like me who are not familiar with xcode. 😉

Log in to reply

8. ***google.com/accounts/o8…***
December 3, 2010 | 4:17 am

Thank you very much for this blog post. It was really useful.

Important notes: Box2d stopped b2PolygonShape to be used for edges, they now require b2EdgeShape to be used for that. So change the code:

```
b2PolygonShape groundBox;

// bottom
groundBox.SetAsEdge(b2Vec2(0,0), b2Vec2(screenSize.width/PTM_RATIO,0));
groundBody->CreateFixture(&groundBox, 0);

// top
groundBox.SetAsEdge(b2Vec2(0,screenSize.height/PTM_RATIO), b2Vec2(screenSize.width/PTM_RATIO,screenSize.height/PTM_RATIO));
groundBody->CreateFixture(&groundBox, 0);

// left
groundBox.SetAsEdge(b2Vec2(0,screenSize.height/PTM_RATIO), b2Vec2(0,0));
groundBody->CreateFixture(&groundBox, 0);

// right
groundBox.SetAsEdge(b2Vec2(screenSize.width/PTM_RATIO,screenSize.height/PTM_RATIO), b2Vec2(screenSize.width/PTM_RATIO,0));
groundBody->CreateFixture(&groundBox, 0);
```

TO:

```
b2EdgeShape groundBox;

// bottom
groundBox.Set(b2Vec2(0,0), b2Vec2(screenSize.width/PTM_RATIO,0));
groundBody->CreateFixture(&groundBox, 0);

// top
groundBox.Set(b2Vec2(0,screenSize.height/PTM_RATIO), b2Vec2(screenSize.width/PTM_RATIO,screenSize.height/PTM_RATIO));
groundBody->CreateFixture(&groundBox, 0);

// left
groundBox.Set(b2Vec2(0,screenSize.height/PTM_RATIO), b2Vec2(0,0));
groundBody->CreateFixture(&groundBox, 0);

// right
```

```
groundBox.Set(b2Vec2(screenSize.width/PTM_RATIO,screenSize.height/PTM_RATIO), b2Vec2(screenSize.width/PTM_RATIO,0));
groundBody->CreateFixture(&groundBox, 0);
```

Also, if someone else is having trouble knowing what file to import, the first one is the header of that same file and the second one is Box2D.h.

Finally, when setting the header search path, be sure to select recursive so it will search in the inside of the Box2d folder.

Thanks again! Keep up the good work!

Log in to reply

9. *claesjacobsson*
December 14, 2010 | 2:16 pm

I want to animate a rope/cable. Can that be done with Box2D?

Log in to reply

-
Drops
December 15, 2010 | 1:30 pm

  No idea. You'd have to ask somebody who actually knows about Box2D.

  Log in to reply

- *David Pike*
December 6, 2011 | 3:07 pm

  You will find a game called Climbers (source available) which has rope code in it which might help?

  https://github.com/haqu/climbers

  Log in to reply

10. *steamingcoaco*
December 17, 2010 | 1:40 am

Hey,

Thanks for the tutorial. I'm having trouble with some errors though. I tried copy/paste, but it gives errors, see link below.

http://img840.imageshack.us/i/errorsfr.png/

I'd appreciate your help!

-Steaming Coaco

Log in to reply

- *steamingcoaco*
December 17, 2010 | 1:44 am

  I've got more of it working, saw the other post. I've got 3 errors so far, I'll probably run into many more…

  updated photo:
  http://img689.imageshack.us/i/3errors.png/

  Log in to reply

-
Drops
December 17, 2010 | 7:37 am

  Replace all & gt; with > and all & lt; with <

  Somehow the syntax highlighter and wordpress messed that up.

  Log in to reply

steamingcoaco
December 18, 2010 | 6:25 pm

b2Body* groundBody = world->CreateBody(&groundBodyDef);

This line is still giving errors.

Error: 'amp' was not declared in this scope.
Error: Expected ';' before ')' token.

Log in to reply

Drops
December 18, 2010 | 7:25 pm

Have another look now, all amp should be & and all gt should be >

Log in to reply

steamingcoaco
December 19, 2010 | 4:52 pm

☺ I've got it working now, thanks for following this and giving help! Really cool.

11. ***karmatoad***
February 26, 2011 | 2:03 pm

This is a great blog. I hope you don't mind, I've taken your code and made an 'OpenGL' version (for IOS) – I've credited you (and linked back to here too). Hope that's OK.

Here's my Open GL version:
http://www.karmatoad.co.uk/?p=183

Log in to reply

12. *andrea*
March 3, 2011 | 10:38 pm

That's what I wanted. Pure satisfaction!!
Thank you for this nice tutorial

Log in to reply

13. *harold*
March 17, 2011 | 7:43 pm

Has anyone gotten this to work with Xcode 4? For the life of me I can't get the paths correct.

Log in to reply

14. ***Conrad Kramer***
March 29, 2011 | 12:56 am

☺ Just thought I'd let you know – this was the article gave me the idea for my tweak Graviboard

http://www.kramerapps.com/graviboard/

Log in to reply

Kris
September 25, 2011 | 3:20 am

Hi Conrad,

How did you get it to work with dragging the UIViews? I tried it, but the dragging is buggy and when released it just shows up instantly back to where it was.

Please help out if you can.

Log in to reply

15. *Mustafa*
April 5, 2011 | 1:25 pm

Hi!

I'm running into issues while trying to add Box2D into my project. I'm using Xcode4 with Box2D v2.1.2.

Issue # 1: When i try to add Box2D into my project, i (sometimes) receive the following message "The request could not be performed because it was returned by Subversion as invalid."

Issue # 2: If i somehow manage to add Box2D into the project, it doesn't compile. However, modifying Header Section Paths to ${PROJECT_DIR} with "Recursive" checkbox ON fixes the issue.

Issue # 3: Now's the part where i really get stuck! Adding #import to UIKitPhysicsViewController.h gives tons of errors, almost all of which say "No such file or directory". Somehow i feel it has something to do with the path i specified in Header Section Paths, but i don't know how to fix this? Setting the Section Header Path to "${PROJECT_DIR}/Box2D" also doesn't help. Note: Physically, the Box2D.h file is located in "/UIKitPhysics/Box2D/Box2D.h" and "UIKitPhysics.xcodeproj" is at the root, right next to "/UIKitPhysics" (New Xcode4 project and file positioning).

Log in to reply

16. ***Someone***
April 9, 2011 | 9:23 am

Any guides on how to do it in XCode 4?

Log in to reply

17. ***Basil***
April 28, 2011 | 2:12 pm

I am trying to add the Box2d , and got successful but cannot change the file type bcos iam using the sdk 4.3 and in that there is not file type …. So please help me

Log in to reply

18. ***Kazuyoshi Ueno***
July 18, 2011 | 6:00 pm

Hi !
Thanks a lot for your post !

I've just tried to make my Proj. with your post with xcode4, but i got some errors in createPhysicsWorld().

I can't 'world = new b2World(gravity, doSleep);'. The error is

/Users/myName/Documents/iOS/[S]ampleProg./Box2d/MyBox2dTest/MyBox2dTest/MyBox2dTestViewController.m:34: error: no matching function for call to 'b2World::b2World(b2Vec2&, bool&)'

I set ${PROJECT_DIR} as 'The Header Search Path' , though…..

Any advice ?

Log in to reply

- *Drops*
July 18, 2011 | 6:42 pm

Actually this tutorial might not work any more with the newer versions of Box2D. You have to check what they changed.

Log in to reply

19. ***Kazuyoshi Ueno***

July 18, 2011 | 6:45 pm

I got the answer by myself.

> I can't 'world = new b2World(gravity, doSleep);'. The error is

I just used 'tags' instead of the 'Trunk'…

Thanks.

Log in to reply

- *Kris*
  September 25, 2011 | 12:03 am

  How did you fix the "no matching function for call to 'b2World::b2World(b2Vec2&, bool&)'" error?

  I'm getting the same and I don't understand what you mean by "I just used 'tags' instead of the 'Trunk'…"

  Please explain it to me if you can. Thanks.

  Log in to reply

  - *apple_motion*
    November 30, 2011 | 3:03 pm

    1) please change that line to:-

    😃 world = new b2World(gravity); // just tested and work fine for the newly v2.21 release

    2) then, delete this line too.

    bool doSleep = true; // because you don't need this line anymore.

    😃 remark: I tested in XCode 3.25

    Log in to reply

    - *Kris*
      December 6, 2011 | 7:01 pm

      Thank you for your reply. I got it working same day, but thanks for your reply.

      Log in to reply

20. ***Kazuyoshi Ueno***
July 24, 2011 | 1:28 pm

Hi,

How can i move the UIView objects with my finger ?
Do I need to implement touch methods ?

ex)
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event

If the objects in the Box2D world can be moved by implementing the specific method of 'Box2D',
Please give the sample code. I hope that it's going to be fine with adding a few code to Your Good Tutrial.

Thanks a lot.

Log in to reply

- *Kris*
  December 6, 2011 | 7:03 pm

  Did you manage to get that working? I'm trying to do the same thing. I tried a tutorial meant for cocos2d, but could never translate it for UIKit.

If you solved it, please let me know how. Thanks.

Log in to reply

21. *Wipoo Shinsirikul*
July 28, 2011 | 12:45 pm

I found a solution for one who found 2000-3000 error

“The Project path must not contain any whitespace”
That’s all

Log in to reply

22. *Wrku*
August 2, 2011 | 6:01 am

Great help! One thing I had to change with the latest version of Box2D, as it seems that b2EdgeShape is no longer supported:

—

```
// Define the ground box shape.
b2EdgeShape groundBox;

// bottom
groundBox.Set(b2Vec2(0,0), b2Vec2(screenSize.width/PTM_RATIO,0));
groundBody->CreateFixture(&groundBox, 0);

// top
groundBox.Set(b2Vec2(0,screenSize.height/PTM_RATIO), b2Vec2(screenSize.width/PTM_RATIO,screenSize.height/PTM_RATIO));
groundBody->CreateFixture(&groundBox, 0);

// left
groundBox.Set(b2Vec2(0,screenSize.height/PTM_RATIO), b2Vec2(0,0));
groundBody->CreateFixture(&groundBox, 0);

// right
groundBox.Set(b2Vec2(screenSize.width/PTM_RATIO,screenSize.height/PTM_RATIO), b2Vec2(screenSize.width/PTM_RATIO,0));
groundBody->CreateFixture(&groundBox, 0);
```

——

Changed to….

——

```
// Define the ground box shape.
b2PolygonShape groundBox;

// bottom
groundBox.SetAsEdge(b2Vec2(0,0), b2Vec2(screenSize.width/PTM_RATIO,0));
groundBody->CreateFixture(&groundBox, 0);

// top
groundBox.SetAsEdge(b2Vec2(0,screenSize.height/PTM_RATIO), b2Vec2(screenSize.width/PTM_RATIO,screenSize.height/PTM_RATIO));
groundBody->CreateFixture(&groundBox, 0);

// left
groundBox.SetAsEdge(b2Vec2(0,screenSize.height/PTM_RATIO), b2Vec2(0,0));
groundBody->CreateFixture(&groundBox, 0);
```

```
// right
groundBox.SetAsEdge(b2Vec2(screenSize.width/PTM_RATIO,screenSize.height/PTM_RATIO), b2Vec2(screenSize.width/PTM_RATIO,0));
groundBody->CreateFixture(&groundBox, 0);
```

Log in to reply

23. *samfisher*
November 26, 2011 | 3:17 pm

Very useful tutorial… thanks a lot!!

Log in to reply

24. *Oliver*
November 28, 2011 | 10:14 am

Hi,

I am new to IPhone programming but I know Box2D from Flash-Programming and would like to use that.
I have problems to find the "Header Search Paths" to change to ${PROJECT_DIR}.
Where is that – maybe it changed in XCode 4.2? Can't find it.

I am also not quite sure about:
"Since the default project probably has an include of the our view controller's header in the app delegate, you need to make the same adjustment for the app delegate's .m file. Get Info on the those files and change the file type to "sourcecode.cpp.objcpp".
So I have to change the file type of my ViewController (.m + .h?) and and of the AppDelegate (.m+.h) Cocoa Files? Nothing to be changed in the Box2D files, right?

Cheers,

Oliver

Log in to reply

- *yo*
December 22, 2011 | 9:48 pm

For those of you having trouble with Xcode finding the Box2D.h (and related files). See the link below.

http://www.box2d.org/forum/viewtopic.php?f=7&t=7812

Log in to reply

25. *DD*
December 21, 2011 | 1:44 pm

why i can't ? xcode 4.2

"UIView *oneView = (UIView *)b->GetUserData();" -> error
"bodyDef.userData = physicalView;" -> error

Thanks

Log in to reply

Trackbacks

1. ***Box2D iPhone Physics Tutorial – No Cocos2D Required | iPhone and iPad SDK Development Tutorials and Programming Tips***
2. ***Stefan Arentz − Sound Shaker on the iPad***
3. ***Box2D – 2D Physics Engine***
4. ***clouds | Blog | Getting Box2d working in the Sparrow Framework***
5. ***Box2D UIView and Objective C C++ Wrapper | CocoaHeads***
6. ***IS – Physics on iPhone | Nine Whole Fractions***

7. ***Physic engine in standard viewcontroller - iPhone Dev SDK Forum***
8. ***Cant Integrate Box2d - iPhone Dev SDK Forum***

Leave a Comment

You must be **logged in** to post a comment.

- **Log In**
- **Register**