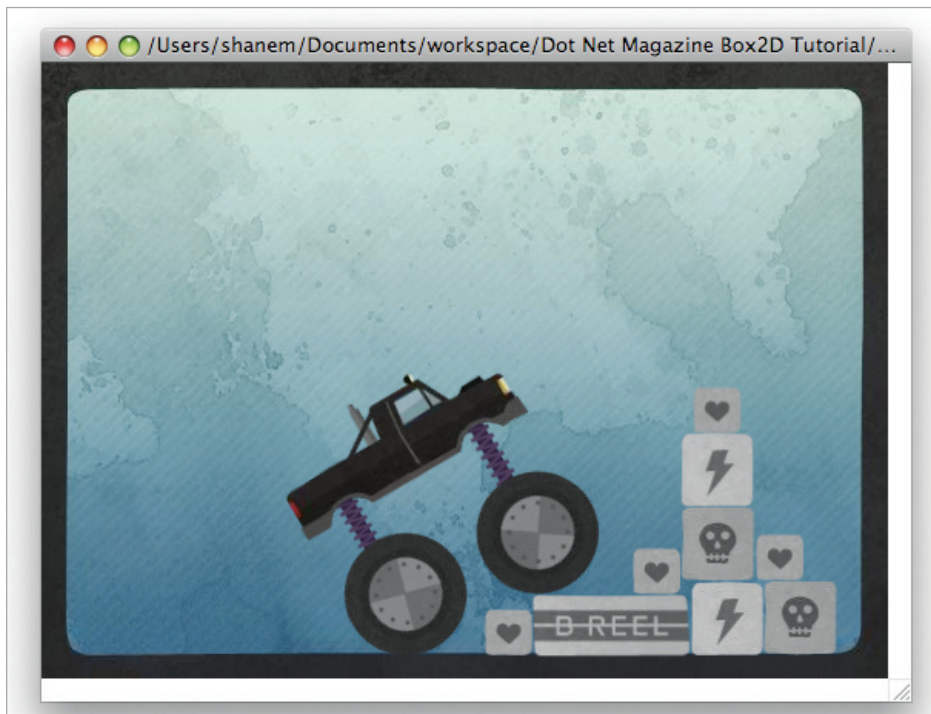


/Flash/create a physics game with Box2D



What could be more fun than creating your own monster truck game? Let Shane McCartney of top digital production company B-Reel show you how



What you'll learn We'll show you how to use QuickBox2D to create a physics-based game. Learn how to make a physics scene, basic obstacles and the 'monster truck'. We'll finish this tutorial by adding interaction and provide tips/techniques to add your own killer graphics.

- Knowledge needed** Flash, ActionScript, Photoshop
- Requires** Flash, ActionScript, Photoshop
- Project time** 8 hours

With Flash CS5 on the horizon, there's never been a better time to get involved with physics for animations and games. Although this tutorial will cover physics for real-time user interaction, physics can also be used for life-like improvisational animations.

At B-Reel we've recently been hard at work on a physics-based game for Cheestings.co.uk with FallonLondon. We'll share some inside knowledge, tips and tricks to make your own amazing physics game using the open source physics engine Box2D. We'll toe-dip you into becoming comfortable with Box2D by using another ActionScript library, QuickBox2D. This library is

not a bare-bones version of Box2D but instead interfaces with Box2D at a higher and more comfortable level. This means functionality that ordinarily requires many lines of code is neatly provided in a syntax more suited to ActionScripters.

The end product is a small game that could be developed into a fully fledged driving game. You'll also become familiar with the terminology and practices used in physics engines like Box2D and be prepared for the new features in Flash CS5. ●

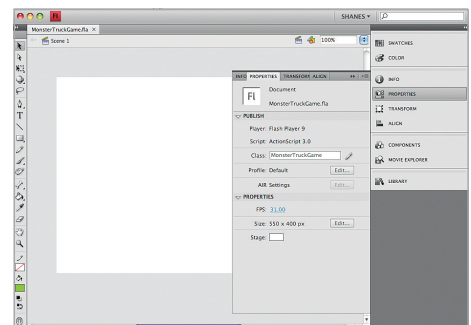


About the author

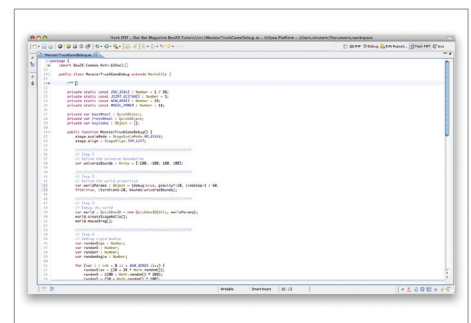
Name Shane McCartney
Site b-reel.com
Areas of expertise Flash, AS, animation and design
Clients Doritos, Cheestings
What was the last 3D movie you saw? *Avatar*

Expert tip Parameters

- A time step is a unit of time applied each time the physics engine is updated. Larger steps make the engine run faster. It's vital to keep your time step at a constant rate and not have it based to a frame rate.
- Iterations define the number of times the engine tries to solve a physics simulation. Sometimes by solving one collision, an object can be placed into collision with a different object, so iterations may eventually resolve this. Higher iterations mean a greater chance of a simulation resolving correctly but at the expense of unnecessarily achieving this.



START Boundary Create a new document class extending **MovieClip**. Start by constructing an **Array** that defines the universe boundary. The unit for these values is physical metres not screen pixels. We find it's easiest to use a ratio for the conversion of screen to physical values.



2 Define properties Next, define the properties of your world with gravity in m/s. Box2D allows for gravity in both an x and y direction but we tend to always use y. Next set the **timestep** to 1/60 and **iterations** to 20 and make sure **frame rate independent motion** is true.

```

36 // Step 2
37 // Defines the universe boundaries
38 var universeBounds : Array = [-100, -100, 100];
39
40 // Step 3
41 // Defines the world properties
42 var worldProperties : Object = {gravityY:20, timeStep:1 / 60,
43 friction: iterations(20, bounds:universeBounds)};
44
45 // Step 4
46 // Debug the world
47 var world : QuickBox2D = new QuickBox2D(this, worldProperties);
48 world.createStageWall();
49
50 // Step 5
51 // Adding rigid bodies
52 var randomSize : Number;
53 var randomX : Number;
54 var randomY : Number;
55 var randomAngle : Number;
56
57 for (var i : int = 0; i < NUM_BOXES; i++) {
58     randomSize = (20 + 30 * Math.random());
59     randomX = (200 * Math.random());
60     randomY = (200 * Math.random());
61     randomAngle = (Math.random() * 180) * Math.PI / 180; // in rads
62 }

```

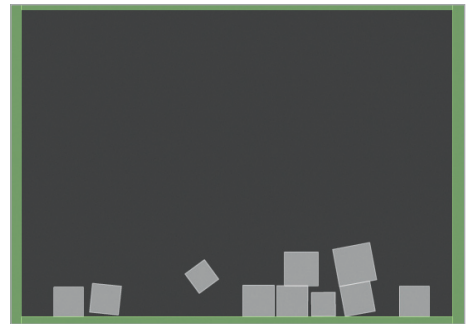
3 Create world Construct an instance of QuickBox2D with the properties of the world in its constructor. Use create StageWalls() to define temporary boundary walls fixed to the size of your Flash movie. This is different from the universe boundary (the area objects will be simulated within).

```

55 // Step 5
56 // Adding rigid bodies
57 var randomSize : Number;
58 var randomX : Number;
59 var randomY : Number;
60 var randomAngle : Number;
61
62 for (var i : int = 0; i < NUM_BOXES; i++) {
63     randomSize = (20 + 30 * Math.random());
64     randomX = (200 * Math.random());
65     randomY = (200 * Math.random());
66     randomAngle = (Math.random() * 180) * Math.PI / 180; // in rads
67 }
68
69 // Step 6
70 // Define the properties for the rigid body
71 world.addObject({x:pxToM(randomX), y:pyToM(randomY), width:pxToM(randomSize),
72 height:pyToM(randomSize), angle:randomAngle,
73 density:10, restitution:0.6, friction:0.6});
74
75 // Step 7
76 // Start the world
77 world.start();
78
79 // Step 8 & 9
80 // Build the trucks chassis & body / Create the wheels
81 var vehicleTop : QuickObject = world.addObject({density:0.5, friction:0.5, y:pxToM(75),
82 var chassis : QuickObject = world.addObject({x:3, y:3, objects:[vehicleTop,
83 frontWheel = world.addCircle({x:pxToM(150), radius:pxToM(30),
84 backWheel = world.addCircle({x:pxToM(100), y:pyToM(150), radius:pxToM(30),
85
86 // Step 10
87 // Build the trucks chassis & body
88 var vehicleTop : QuickObject = world.addObject({x:pxToM(75), y:pyToM(40),
89 width:pxToM(30), height:pxToM(30),
90 density:5, restitution:0.5, friction:0.5});
91
92 var chassis : QuickObject = world.addObject({x:pxToM(75), y:0,
93 width:pxToM(150), height:pxToM(50),
94 density:5, restitution:0.5, friction:0.5});
95
96 var vehicleBase : QuickObject = world.addObject({x:pxToM(75), y:0,
97 width:pxToM(150), height:pxToM(50),
98 density:5, restitution:0.5, friction:0.5});
99
100 var chassis : QuickObject = world.addObject({x:3, y:3,
101 objects:[vehicleTop, vehicleBase], allowSleep:false});
102
103 // Step 9
104 // Create the wheels
105 frontWheel = world.addCircle({x:pxToM(225), y:pyToM(150),
106 radius:pxToM(30),
107 density:10, restitution:0.6, friction:0.9});
108
109 backWheel = world.addCircle({x:pxToM(100), y:pyToM(150), radius:pxToM(30),
110 density:10,
111 restitution:0.6, friction:0.9});
112
113 // Step 10 & 11
114 // Adjusting your suspension I/II
115 var frontWheelCenter : b2Vec2 = frontWheel.body.getWorldCenter();
116 var backWheelCenter : b2Vec2 = backWheel.body.getWorldCenter();
117
118 frontWheelAnchor1 : b2Vec2 = frontWheelCenter.Copy();
119 frontWheelAnchor2 : b2Vec2 = frontWheelCenter.Copy();
120
121 frontWheelAnchor1.Add(new b2Vec2(-JOINT_DISTANCE, 0));
122 frontWheelAnchor2.Add(new b2Vec2(-JOINT_DISTANCE, 0));
123
124 world.addJoint({a:chassis.body, b:frontWheel.body,
125 vecA:frontWheelAnchor1, vecB:frontWheelCenter,
126 frequencyHz:10, dampingRatio:0.5});
127
128 world.addJoint({a:chassis.body, b:frontWheel.body,
129 vecA:frontWheelAnchor2, vecB:frontWheelCenter,
130 frequencyHz:10, dampingRatio:0.5});
131
132 var backWheelCenter : b2Vec2 = backWheel.body.getWorldCenter();
133 var backWheelAnchor1 : b2Vec2 = backWheelCenter.Copy();
134 backWheelAnchor1.Add(new b2Vec2(-JOINT_DISTANCE, 0));
135 var backWheelAnchor2 : b2Vec2 = backWheelCenter.Copy();
136 backWheelAnchor2.Add(new b2Vec2(-JOINT_DISTANCE, 0));
137
138 world.addJoint({a:chassis.body, b:backWheel.body,
139 vecA:backWheelAnchor1, vecB:backWheelCenter,
140 frequencyHz:10, dampingRatio:0.5});
141
142 world.addJoint({a:chassis.body, b:backWheel.body,
143 vecA:backWheelAnchor2, vecB:backWheelCenter,
144 frequencyHz:10, dampingRatio:0.5});
145
146 // Step 12
147 // Adding a motor & brakes
148 stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyUp);
149 stage.addEventListener(KeyboardEvent.KEY_UP, onKeyUp);
150 this.addEventListener(Event.ENTER_FRAME, onEnterFrame);
151
152 private function onKeyUp(event : KeyboardEvent) : void {
153     delete keyCodes[event.keyCode];
154 }
155
156 private function onEnterFrame(event : Event) : void {
157     switch(true) {
158         case Keyboard.SPACE in keyCodes:
159             backWheel.body.SetAngularVelocity(0);
160     }
161 }

```

4 Debugging Further to the world parameters we've already defined, set debug to true. This allows for wireframe rendering of a physics simulation without the need for custom graphics. Also trigger the function mouseDrag() to allow for mouse-dragging of world objects.



5 Rigid bodies All objects of mass are referred to as rigid bodies within physics simulations and do not deform on collision. Let's add some rigid body boxes with the function addBox. We can define the initial position and dimensions and even an initial angle in the unit radians.

```

63 for (var i : int = 0; i < NUM_BOXES; i++) {
64     randomSize = (20 + 30 * Math.random());
65     randomX = (200 * Math.random());
66     randomY = (200 * Math.random());
67     randomAngle = (Math.random() * 180) * Math.PI / 180; // in rads
68 }
69
70 // Step 6
71 // Define the properties for the rigid body
72 world.addObject({x:pxToM(randomX), y:pyToM(randomY), width:pxToM(randomSize),
73 height:pyToM(randomSize), angle:randomAngle,
74 density:10, restitution:0.6, friction:0.6});
75
76 // Step 7
77 // Start the world
78 world.start();
79
80 // Step 8 & 9
81 // Build the trucks chassis & body / Create the wheels
82 var vehicleTop : QuickObject = world.addObject({density:0.5, friction:0.5, y:pxToM(75),
83 var chassis : QuickObject = world.addObject({x:3, y:3, objects:[vehicleTop,
84 frontWheel = world.addCircle({x:pxToM(150), radius:pxToM(30),
85 backWheel = world.addCircle({x:pxToM(100), y:pyToM(150), radius:pxToM(30),
86
87 // Step 10
88 // Build the trucks chassis & body
89 var vehicleTop : QuickObject = world.addObject({x:pxToM(75), y:pyToM(40),
90 width:pxToM(30), height:pxToM(30),
91 density:5, restitution:0.5, friction:0.5});
92
93 var chassis : QuickObject = world.addObject({x:pxToM(75), y:0,
94 width:pxToM(150), height:pxToM(50),
95 density:5, restitution:0.5, friction:0.5});
96
97 var vehicleBase : QuickObject = world.addObject({x:pxToM(75), y:0,
98 width:pxToM(150), height:pxToM(50),
99 density:5, restitution:0.5, friction:0.5});
100
101 var chassis : QuickObject = world.addObject({x:3, y:3,
102 objects:[vehicleTop, vehicleBase], allowSleep:false});
103
104 // Step 9
105 // Create the wheels
106 frontWheel = world.addCircle({x:pxToM(225), y:pyToM(150),
107 radius:pxToM(30),
108 density:10, restitution:0.6, friction:0.9});
109
110 backWheel = world.addCircle({x:pxToM(100), y:pyToM(150), radius:pxToM(30),
111 density:10,
112 restitution:0.6, friction:0.9});
113
114 // Step 10 & 11
115 // Adjusting your suspension I/II
116 var frontWheelCenter : b2Vec2 = frontWheel.body.getWorldCenter();
117 var backWheelCenter : b2Vec2 = backWheel.body.getWorldCenter();
118
119 frontWheelAnchor1 : b2Vec2 = frontWheelCenter.Copy();
120 frontWheelAnchor2 : b2Vec2 = frontWheelCenter.Copy();
121
122 frontWheelAnchor1.Add(new b2Vec2(-JOINT_DISTANCE, 0));
123 frontWheelAnchor2.Add(new b2Vec2(-JOINT_DISTANCE, 0));
124
125 world.addJoint({a:chassis.body, b:frontWheel.body,
126 vecA:frontWheelAnchor1, vecB:frontWheelCenter,
127 frequencyHz:10, dampingRatio:0.5});
128
129 world.addJoint({a:chassis.body, b:frontWheel.body,
130 vecA:frontWheelAnchor2, vecB:frontWheelCenter,
131 frequencyHz:10, dampingRatio:0.5});
132
133 var backWheelCenter : b2Vec2 = backWheel.body.getWorldCenter();
134 var backWheelAnchor1 : b2Vec2 = backWheelCenter.Copy();
135 backWheelAnchor1.Add(new b2Vec2(-JOINT_DISTANCE, 0));
136 var backWheelAnchor2 : b2Vec2 = backWheelCenter.Copy();
137 backWheelAnchor2.Add(new b2Vec2(-JOINT_DISTANCE, 0));
138
139 world.addJoint({a:chassis.body, b:backWheel.body,
140 vecA:backWheelAnchor1, vecB:backWheelCenter,
141 frequencyHz:10, dampingRatio:0.5});
142
143 world.addJoint({a:chassis.body, b:backWheel.body,
144 vecA:backWheelAnchor2, vecB:backWheelCenter,
145 frequencyHz:10, dampingRatio:0.5});
146
147 // Step 12
148 // Adding a motor & brakes
149 stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyUp);
150 stage.addEventListener(KeyboardEvent.KEY_UP, onKeyUp);
151 this.addEventListener(Event.ENTER_FRAME, onEnterFrame);
152
153 private function onKeyUp(event : KeyboardEvent) : void {
154     delete keyCodes[event.keyCode];
155 }
156
157 private function onEnterFrame(event : Event) : void {
158     switch(true) {
159         case Keyboard.SPACE in keyCodes:
160             backWheel.body.SetAngularVelocity(0);
161     }
162 }

```

6 Defining properties All rigid bodies has a mass value calculated by density. Density determines the body's weight: 0 is weightless. Another parameter is restitution, which defines the amount of energy in a body after collision. So 0 resolves in no bounce and 1 is very bouncy.

```

55 // Step 5
56 // Adding rigid bodies
57 var randomSize : Number;
58 var randomX : Number;
59 var randomY : Number;
60 var randomAngle : Number;
61
62 for (var i : int = 0; i < NUM_BOXES; i++) {
63     randomSize = (20 + 30 * Math.random());
64     randomX = (200 * Math.random());
65     randomY = (200 * Math.random());
66     randomAngle = (Math.random() * 180) * Math.PI / 180; // in rads
67 }
68
69 // Step 6
70 // Define the properties for the rigid body
71 world.addObject({x:pxToM(randomX), y:pyToM(randomY), width:pxToM(randomSize),
72 height:pyToM(randomSize), angle:randomAngle,
73 density:10, restitution:0.6, friction:0.6});
74
75 // Step 7
76 // Start the world
77 world.start();
78
79 // Step 8 & 9
80 // Build the trucks chassis & body / Create the wheels
81 var vehicleTop : QuickObject = world.addObject({density:0.5, friction:0.5, y:pxToM(75),
82 var chassis : QuickObject = world.addObject({x:3, y:3, objects:[vehicleTop,
83 frontWheel = world.addCircle({x:pxToM(150), radius:pxToM(30),
84 backWheel = world.addCircle({x:pxToM(100), y:pyToM(150), radius:pxToM(30),
85
86 // Step 10
87 // Build the trucks chassis & body
88 var vehicleTop : QuickObject = world.addObject({x:pxToM(75), y:pyToM(40),
89 width:pxToM(30), height:pxToM(30),
90 density:5, restitution:0.5, friction:0.5});
91
92 var chassis : QuickObject = world.addObject({x:pxToM(75), y:0,
93 width:pxToM(150), height:pxToM(50),
94 density:5, restitution:0.5, friction:0.5});
95
96 var vehicleBase : QuickObject = world.addObject({x:pxToM(75), y:0,
97 width:pxToM(150), height:pxToM(50),
98 density:5, restitution:0.5, friction:0.5});
99
100 var chassis : QuickObject = world.addObject({x:3, y:3,
101 objects:[vehicleTop, vehicleBase], allowSleep:false});
102
103 // Step 9
104 // Create the wheels
105 frontWheel = world.addCircle({x:pxToM(225), y:pyToM(150),
106 radius:pxToM(30),
107 density:10, restitution:0.6, friction:0.9});
108
109 backWheel = world.addCircle({x:pxToM(100), y:pyToM(150), radius:pxToM(30),
110 density:10,
111 restitution:0.6, friction:0.9});
112
113 // Step 10 & 11
114 // Adjusting your suspension I/II
115 var frontWheelCenter : b2Vec2 = frontWheel.body.getWorldCenter();
116 var backWheelCenter : b2Vec2 = backWheel.body.getWorldCenter();
117
118 frontWheelAnchor1 : b2Vec2 = frontWheelCenter.Copy();
119 frontWheelAnchor2 : b2Vec2 = frontWheelCenter.Copy();
120
121 frontWheelAnchor1.Add(new b2Vec2(-JOINT_DISTANCE, 0));
122 frontWheelAnchor2.Add(new b2Vec2(-JOINT_DISTANCE, 0));
123
124 world.addJoint({a:chassis.body, b:frontWheel.body,
125 vecA:frontWheelAnchor1, vecB:frontWheelCenter,
126 frequencyHz:10, dampingRatio:0.5});
127
128 world.addJoint({a:chassis.body, b:frontWheel.body,
129 vecA:frontWheelAnchor2, vecB:frontWheelCenter,
130 frequencyHz:10, dampingRatio:0.5});
131
132 var backWheelCenter : b2Vec2 = backWheel.body.getWorldCenter();
133 var backWheelAnchor1 : b2Vec2 = backWheelCenter.Copy();
134 backWheelAnchor1.Add(new b2Vec2(-JOINT_DISTANCE, 0));
135 var backWheelAnchor2 : b2Vec2 = backWheelCenter.Copy();
136 backWheelAnchor2.Add(new b2Vec2(-JOINT_DISTANCE, 0));
137
138 world.addJoint({a:chassis.body, b:backWheel.body,
139 vecA:backWheelAnchor1, vecB:backWheelCenter,
140 frequencyHz:10, dampingRatio:0.5});
141
142 world.addJoint({a:chassis.body, b:backWheel.body,
143 vecA:backWheelAnchor2, vecB:backWheelCenter,
144 frequencyHz:10, dampingRatio:0.5});
145
146 // Step 12
147 // Adding a motor & brakes
148 stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyUp);
149 stage.addEventListener(KeyboardEvent.KEY_UP, onKeyUp);
150 this.addEventListener(Event.ENTER_FRAME, onEnterFrame);
151
152 private function onKeyUp(event : KeyboardEvent) : void {
153     delete keyCodes[event.keyCode];
154 }
155
156 private function onEnterFrame(event : Event) : void {
157     switch(true) {
158         case Keyboard.SPACE in keyCodes:
159             backWheel.body.SetAngularVelocity(0);
160     }
161 }

```

7 Start Start the world by triggering the start function. On publish, you can see the world in wireframe debug mode. We can see something in action so it's a good time to start tweaking parameters. This is often the case with physics simulations and it's good to be experimental.

```

83 // Step 8
84 // Build the trucks chassis & body
85 var vehicleTop : QuickObject = world.addObject({x:pxToM(75), y:pyToM(40),
86 width:pxToM(30), height:pxToM(30),
87 density:5, restitution:0.5, friction:0.5});
88
89 var chassis : QuickObject = world.addObject({x:3, y:3,
90 objects:[vehicleTop, vehicleBase], allowSleep:false});
91
92 // Step 9
93 // Create the wheels
94 frontWheel = world.addCircle({x:pxToM(225), y:pyToM(150),
95 radius:pxToM(30),
96 density:10, restitution:0.6, friction:0.9});
97
98 backWheel = world.addCircle({x:pxToM(100), y:pyToM(150), radius:pxToM(30),
99 density:10,
100 restitution:0.6, friction:0.9});
101
102 // Step 10 & 11
103 // Adjusting your suspension I/II
104 var frontWheelCenter : b2Vec2 = frontWheel.body.getWorldCenter();
105 var backWheelCenter : b2Vec2 = backWheel.body.getWorldCenter();
106
107 frontWheelAnchor1 : b2Vec2 = frontWheelCenter.Copy();
108 frontWheelAnchor2 : b2Vec2 = frontWheelCenter.Copy();
109
110 frontWheelAnchor1.Add(new b2Vec2(-JOINT_DISTANCE, 0));
111 frontWheelAnchor2.Add(new b2Vec2(-JOINT_DISTANCE, 0));
112
113 world.addJoint({a:chassis.body, b:frontWheel.body,
114 vecA:frontWheelAnchor1, vecB:frontWheelCenter,
115 frequencyHz:10, dampingRatio:0.5});
116
117 world.addJoint({a:chassis.body, b:frontWheel.body,
118 vecA:frontWheelAnchor2, vecB:frontWheelCenter,
119 frequencyHz:10, dampingRatio:0.5});
120
121 var backWheelCenter : b2Vec2 = backWheel.body.getWorldCenter();
122 var backWheelAnchor1 : b2Vec2 = backWheelCenter.Copy();
123 backWheelAnchor1.Add(new b2Vec2(-JOINT_DISTANCE, 0));
124 var backWheelAnchor2 : b2Vec2 = backWheelCenter.Copy();
125 backWheelAnchor2.Add(new b2Vec2(-JOINT_DISTANCE, 0));
126
127 world.addJoint({a:chassis.body, b:backWheel.body,
128 vecA:backWheelAnchor1, vecB:backWheelCenter,
129 frequencyHz:10, dampingRatio:0.5});
130
131 world.addJoint({a:chassis.body, b:backWheel.body,
132 vecA:backWheelAnchor2, vecB:backWheelCenter,
133 frequencyHz:10, dampingRatio:0.5});
134
135 // Step 12
136 // Adding a motor & brakes
137 stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyUp);
138 stage.addEventListener(KeyboardEvent.KEY_UP, onKeyUp);
139 this.addEventListener(Event.ENTER_FRAME, onEnterFrame);
140
141 private function onKeyUp(event : KeyboardEvent) : void {
142     delete keyCodes[event.keyCode];
143 }
144
145 private function onEnterFrame(event : Event) : void {
146     switch(true) {
147         case Keyboard.SPACE in keyCodes:
148             backWheel.body.SetAngularVelocity(0);
149     }
150 }

```

8 Body Create the truck's body with two boxes. To these we set a friction value. Friction is used to simulate objects sliding alongside each other. 0 is no friction; 1 is very strong. Next we group these boxes together using addGroup(). These boxes are now a compound shape.

```

101 radius:pxToM(30),
102 density:10, restitution:0.6, friction:0.9});
103
104 backWheel = world.addCircle({x:pxToM(100), y:pyToM(150), radius:pxToM(30),
105 density:10,
106 restitution:0.6, friction:0.9});
107
108 // Step 11
109 // Adjusting your suspension II
110 var frontWheelCenter : b2Vec2 = frontWheel.body.getWorldCenter();
111 var frontWheelAnchor1 : b2Vec2 = frontWheelCenter.Copy();
112 frontWheelAnchor1.Add(new b2Vec2(-JOINT_DISTANCE, 0));
113 var frontWheelAnchor2 : b2Vec2 = frontWheelCenter.Copy();
114 frontWheelAnchor2.Add(new b2Vec2(-JOINT_DISTANCE, 0));
115
116 world.addJoint({a:chassis.body, b:frontWheel.body,
117 vecA:frontWheelAnchor1, vecB:frontWheelCenter,
118 frequencyHz:10, dampingRatio:0.5});
119
120 world.addJoint({a:chassis.body, b:frontWheel.body,
121 vecA:frontWheelAnchor2, vecB:frontWheelCenter,
122 frequencyHz:10, dampingRatio:0.5});
123
124 var backWheelCenter : b2Vec2 = backWheel.body.getWorldCenter();
125 var backWheelAnchor1 : b2Vec2 = backWheelCenter.Copy();
126 backWheelAnchor1.Add(new b2Vec2(-JOINT_DISTANCE, 0));
127 var backWheelAnchor2 : b2Vec2 = backWheelCenter.Copy();
128 backWheelAnchor2.Add(new b2Vec2(-JOINT_DISTANCE, 0));
129
130 world.addJoint({a:chassis.body, b:backWheel.body,
131 vecA:backWheelAnchor1, vecB:backWheelCenter,
132 frequencyHz:10, dampingRatio:0.5});
133
134 world.addJoint({a:chassis.body, b:backWheel.body,
135 vecA:backWheelAnchor2, vecB:backWheelCenter,
136 frequencyHz:10, dampingRatio:0.5});
137
138 // Step 12
139 // Adding a motor & brakes
140 stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyUp);
141 stage.addEventListener(KeyboardEvent.KEY_UP, onKeyUp);
142 this.addEventListener(Event.ENTER_FRAME, onEnterFrame);
143
144 private function onKeyUp(event : KeyboardEvent) : void {
145     delete keyCodes[event.keyCode];
146 }
147
148 private function onEnterFrame(event : Event) : void {
149     switch(true) {
150         case Keyboard.SPACE in keyCodes:
151             backWheel.body.SetAngularVelocity(0);
152     }
153 }

```

9 Wheels Now add wheels using the addCircle function. Make sure you pick the correct radius and friction. Once both wheels are created, add two distance-based joints to simulate a suspension system. Each is applied left and right to a wheel's centre point to keep it in position.

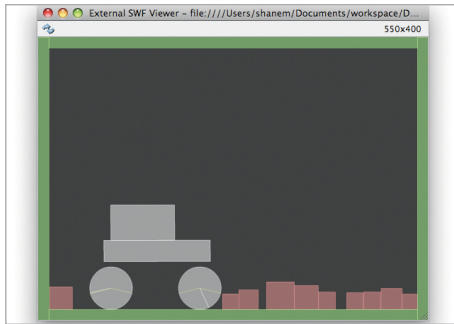
```

129 world.addJoint({a:chassis.body, b:backWheel.body,
130 vecA:backWheelAnchor1, vecB:backWheelCenter,
131 frequencyHz:10, dampingRatio:0.5});
132
133 world.addJoint({a:chassis.body, b:backWheel.body,
134 vecA:backWheelAnchor2, vecB:backWheelCenter,
135 frequencyHz:10, dampingRatio:0.5});
136
137 // Step 12
138 // Adding a motor & brakes
139 stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyUp);
140 stage.addEventListener(KeyboardEvent.KEY_UP, onKeyUp);
141 this.addEventListener(Event.ENTER_FRAME, onEnterFrame);
142
143 private function onKeyUp(event : KeyboardEvent) : void {
144     delete keyCodes[event.keyCode];
145 }
146
147 private function onEnterFrame(event : Event) : void {
148     switch(true) {
149         case Keyboard.SPACE in keyCodes:
150             backWheel.body.SetAngularVelocity(0);
151     }
152 }

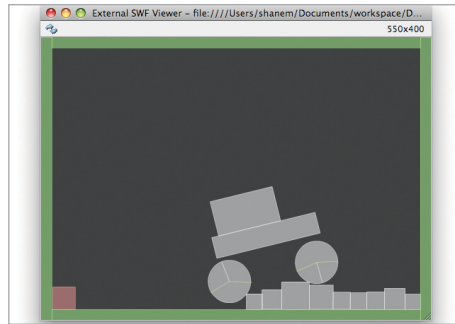
```

10 Suspension 1 When defining the joints, think about the distance between the wheels and the truck's body, as this defines suspension distance. Also important is frequencyKhZ, which defines how springy the suspension is. A higher value makes it stiffer, a lower one looser.

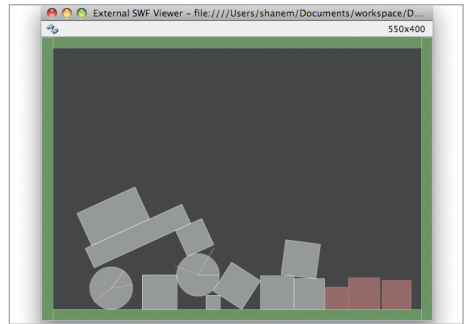
11 Suspension 2 The last important property is the damping ratio, which is like friction; however, it applies constantly. This allows for our suspension to ease into a resting position. We prefer really springy suspension systems to make the gameplay more fun, but you can try your own values to find something you prefer.



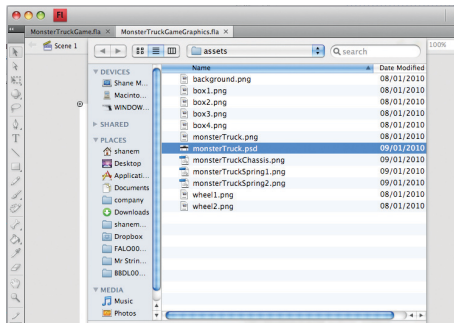
12 Motor To add keyboard interaction, create a stage event listener for the `KEY_DOWN` and `KEY_UP` events. Within each of the keyboard event listener functions, store and delete the ASCII keycode to an `Object` to later retrieve these interactions within an `ENTER_FRAME` listener.



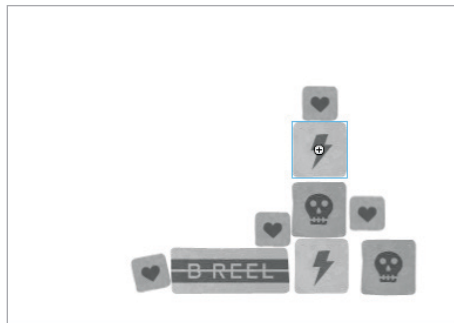
13 Brakes The key presses we're interested in are `KEY_LEFT`, `KEY_RIGHT` & `SPACE`. The `ENTER_FRAME` event listener checks which key is pressed. If it's an arrow key, we add/remove `angularVelocity` to the wheels: this turns them either clockwise or anti-clockwise. Define brakes by setting `angularVelocity` to 0.



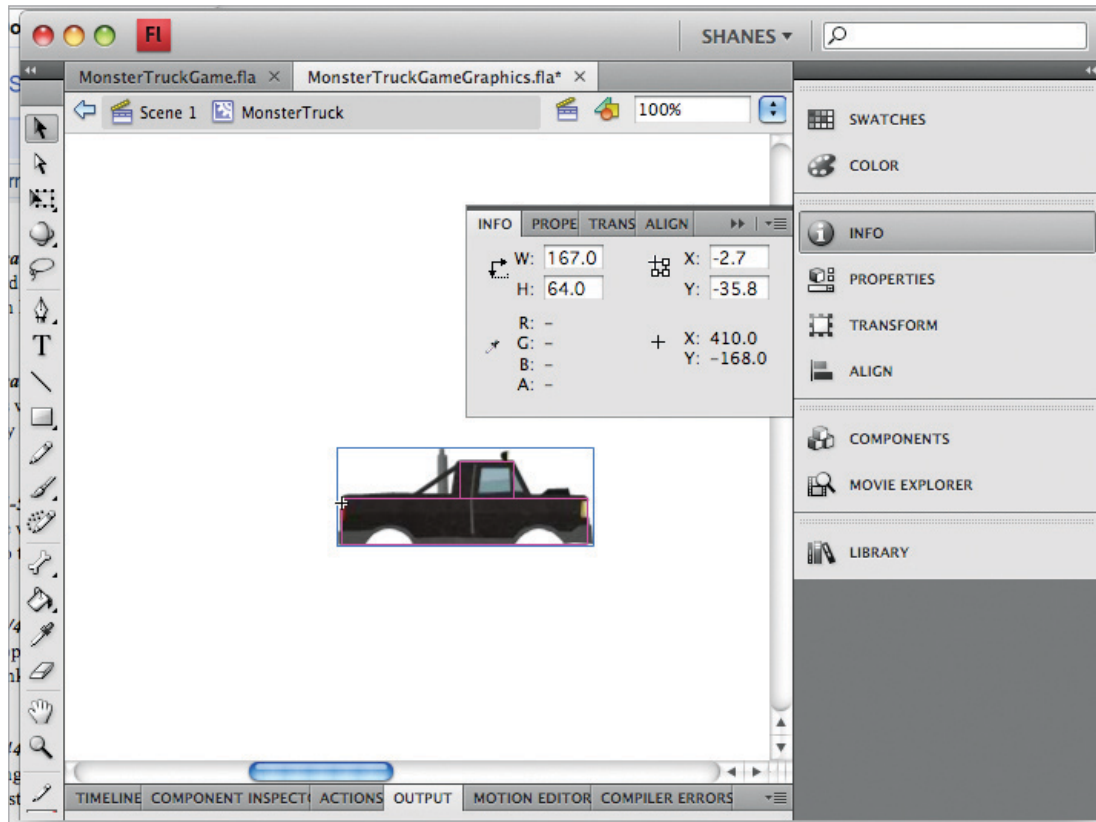
14 Ready to play We're now at a stage where you can drive around the monster truck and crash it into boxes but the artwork is totally lame. Adding custom artwork with `QuickBox2D` is easy. There are a few techniques to do this and we'll now show you the best approaches to use.



15 Graphics In `QuickBox2D` to apply your own custom artwork you use the `skin` property when adding a rigid body to the world using either `addCircle`, `addBox` or `addGroup`. The `skin` property can either be a class reference, a constructed `DisplayObject` or a linkage name string.



16 Skinning We want a big stack of boxes for our truck to crash through. Using `QuickBox2D` you can lay out as many boxes as you like, precisely within Flash's design view. `QuickBox2D` will then use the co-ordinates, dimensions and graphics automatically within the physics system.

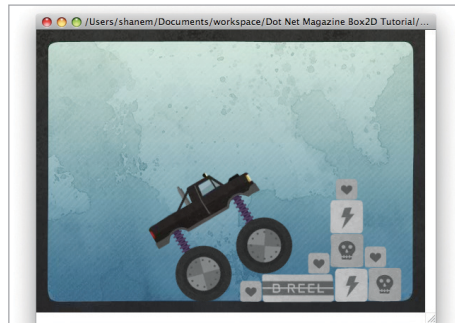


17 Skinning truck 1 If you remember, when constructing the truck's body it was made from two boxes, combined to make a compound shape. Next we need to carefully position the monster truck artwork within a containing `Sprite` in your Flash library. When defining the compound shape, we set the skin to this linkage class name.

```
switch(true) {
  case Keyboard.SPACE in keyCodes:
    backWheel.body.setAngularVelocity(0);
    frontWheel.body.setAngularVelocity(0);
    return;
  case Keyboard.LEFT in keyCodes:
    backWheel.body.setAngularVelocity(-WHEEL_POWER);
    frontWheel.body.setAngularVelocity(-WHEEL_POWER);
    break;
  case Keyboard.RIGHT in keyCodes:
    backWheel.body.setAngularVelocity(WHEEL_POWER);
    frontWheel.body.setAngularVelocity(WHEEL_POWER);
    break;
}

// Step 18
// Skinning the truck II
var frontSpringPoint : Point = frontWheelSpring.parent.localToGlobal(new Point(frontWheelSpring.x, frontWheelSpring.y));
var backSpringPoint : Point = backWheelSpring.parent.localToGlobal(new Point(backWheelSpring.x, backWheelSpring.y));
var frontWheelPoint : Point = new Point(Sprite(frontWheel.userData).x, Sprite(frontWheel.userData).y);
var backWheelPoint : Point = new Point(Sprite(backWheel.userData).x, Sprite(backWheel.userData).y);
var frontWheelDistance : Number = Point.distance(frontWheelPoint, frontSpringPoint);
var backWheelDistance : Number = Point.distance(backWheelPoint, backSpringPoint);
backWheelSpring.scaleY = backWheelDistance / 77;
frontWheelSpring.scaleY = frontWheelDistance / 77;
```

18 Skinning truck 2 Skinning wheels is easy but the springs are a bit trickier: they need to y-scale as the wheels rotate. First add the spring graphics inside the truck body. Next calculate the distance from the wheel to the spring position. Normalise this value as the spring's y-scale.



FINISH Wrap it up Set the default rendering for `QuickBox2D` to not draw lines or fills and add a background graphic to the `DisplayList`. Lastly, tweak the values for friction, density, restitution and frequencyKhz to make the physics simulation look right for the materials we've used.

Expert tip Concave or convex

The current ported AS3 version of `Box2D` only supports convex shapes. In this tutorial we show how to combine these shapes to create complicated compound shapes. In most cases this might be more than enough. However, when the shape is organic in its definition then the use of a concave shape is very handy. Thankfully `QuickBox2D` allows for this functionality built-in. The `addPoly` function within `QuickBox2D` allows for definition of the concave shape by a series of points and it will do the otherwise complex triangulation for you. Wowzas!