Introduction to 2D Game Physics Engine Development

Physics engines play an important part in many types of games. A believable physics interaction between game objects has become a key element of most modern PC and console games as well as, more recently, browser and smartphone games. The range of topics within physics for games is broad and includes but is not limited to areas such as rigid body, fluid dynamics, soft-body, vehicle physics and particle physics. This book will cover the fundamental topics needed for you to get started in understanding and building a general purpose rigid body physics engine in two dimensions. The book also aims to provide you with a reusable game physics engine, which can be used for your own games, by guiding you through the process of building a physics engine step-by-step from scratch. This way you will gain a foundational understanding of the concepts and components required for a standard 2D rigid body system.

While you can just download a physics engine library and continue on with your game or engine development, building your own game engine from scratch has its own advantages. Besides giving you an underlying understanding of how the physics engine operates it gives you more control over the flexibility, performance, functionality, and usability of the engine itself.

As stated, this book will cover the foundation of 2D rigid body physics. The topics will include properties and behavior of rigid bodies, collision detection, collision information encoding, and collision response. The goal is to obtain a fundamental understanding of these concepts which are required to build a usable physics engine.

The book approaches physics engine development from three important avenues: practicality, approachability, and reusability. While reading the book, we want you to get involved and experience the process of building the game engine. The step by step guide should facilitate the practicality of this book. The theories and practices introduced in this book are based on research and investigation from many sources which cover the topics in varying detail. The information is presented in a way that is approachable by allowing you to follow along as each code snippet is explained in parallel to the overall concepts behind each component of the engine. After following along and creating your own engine you will be able to extend and reuse the finished product by adding your own features.

This chapter describes the implementation technology and organization of the book. The discussion then leads you through the steps of downloading, installing, and setting up the development environment; guides you through building your first HTML5 application; and extend this first application with the JavaScript programming language to run your first simulation.

# Setting Up Your Development Environment

The physics engine you are going to build will be accessible through web browsers that could be running on any operating system (OS). The development environment you are about to set up is also OS agnostic. For simplicity, the following instructions are based on a Windows 7/8/10 OS. You should be able to reproduce a similar environment with minor modifications in a Unix-based environment like the Apple macOS or Ubuntu.

Your development process includes an integrated development environment (IDE) and a runtime web browser that is capable of hosting the running game engine. The most convenient systems we have found are the NetBeans IDE with the Google Chrome web browser as the runtime environment. Here are the details:

* *IDE*: All projects in this book are based on the NetBeans IDE. You can download and install the bundle for HTML5 and PHP from <https://netbeans.org/downloads>.
* *Runtime environment*: You will execute your projects in the Google Chrome web browser. You can download and install this browser from <https://www.google.com/chrome/browser/>.
* *Connector Google Chrome plug-in*: This is a Google Chrome extension that connects the web browser to the NetBeans IDE to support HTML5 development. You can download and install this extension from <https://chrome.google.com/webstore/detail/netbeans-connector/hafdlehgocfcodbgjnpecfajgkeejnaa>. The download will automatically install the plug-in to Google Chrome. You may have to restart your computer to complete the installation.

Notice that there are no specific system requirements to support the JavaScript programming language or HTML5. All these technologies are embedded in the web browser runtime environment.

**Note** As mentioned, we chose a NetBeans-based development environment because we found it to be the most convenient. There are many other alternatives that are also free, including and not limited to IntelliJ IDEA, Eclipse, Sublime, Microsoft’s Visual Studio Code, and Adobe Brackets.

# Downloading and Installing JavaScript Syntax Checker

We have found JSLint to be an effective tool in detecting potential JavaScript source code errors. You can download and install JSLint as a plug-in to the NetBeans IDE with the following steps:

* Download it from <http://plugins.netbeans.org/plugin/40893/jslint>. Make sure to take note of the location of the downloaded file.
* Start NetBeans, select Tools ➤ Plugins, and go to the Downloaded tab.
* Click the Add Plugins button and search for the downloaded file from step 1. Double-click this file to install the plug-in.

The following are some useful references for working with JSLint:

* For instructions on how to work with JSLint, see <http://www.jslint.com/help.html>.
* For details on how JSLint works, see <http://plugins.netbeans.org/plugin/40893/jslint>.

# Working in the NetBeans Development Environment

The NetBeans IDE is easy to work with, and the projects in this book require only the editor and debugger. To open a project, select File ➤ Open Projects. Once a project is open, you need to become familiarized with three basic windows, as illustrated in Figure 1-1.

* *Projects window*: This window displays the source code files of the project.
* *Editor window*: This window displays and allows you to edit the source code of your project. You can select the source code file to work with by double-clicking the corresponding file name in the Projects window.
* *Action Items window*: This window displays the error message output from the JSLint checker.

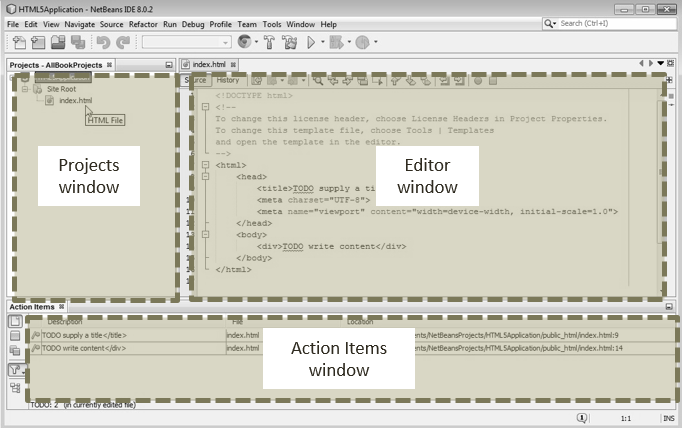


Figure 1-1. The NetBeans IDE

**Note** If you cannot see a window in the IDE, you can click the Window menu and select the name of the missing window to cause it to appear. For example, if you cannot see the Projects window in the IDE, you can select Window ➤ Projects to open it.

## Creating an HTML5 Project in NetBeans

You are now ready to create your first HTML5 project.

1. Start NetBeans. Select File ➤ New Project (or press Ctrl+Shift+N), as shown in Figure 1-2. A New Project window will appear.

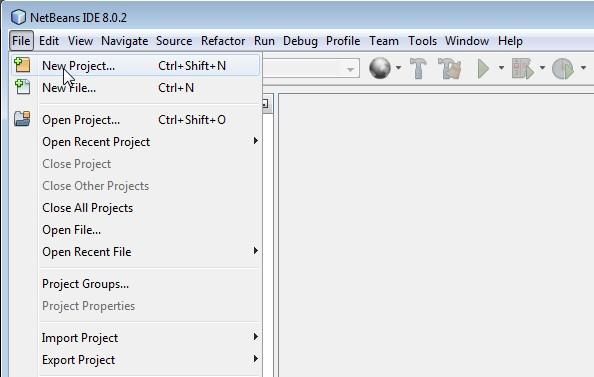


Figure 1-2. Creating a new project

1. In the New Project window, select HTML5 in the Categories section, and select HTML5 Application from the Projects section, as shown in Figure 1-3. Click the Next button to bring up the project configuration window.

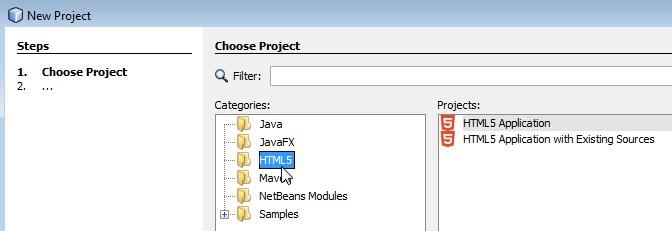


Figure 1-3. Selecting the HTML5 project

1. As shown in Figure 1-4, enter the name and location of the project, and click the Finish button to create your first HTML5 project.

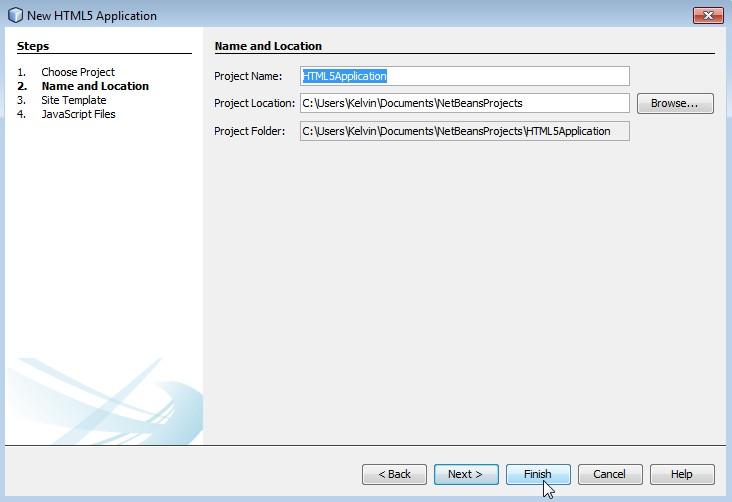


Figure 1-4. Naming the project

NetBeans will generate the template of a simple and complete HTML5 application project for you. Your IDE should look similar to Figure 1-5.

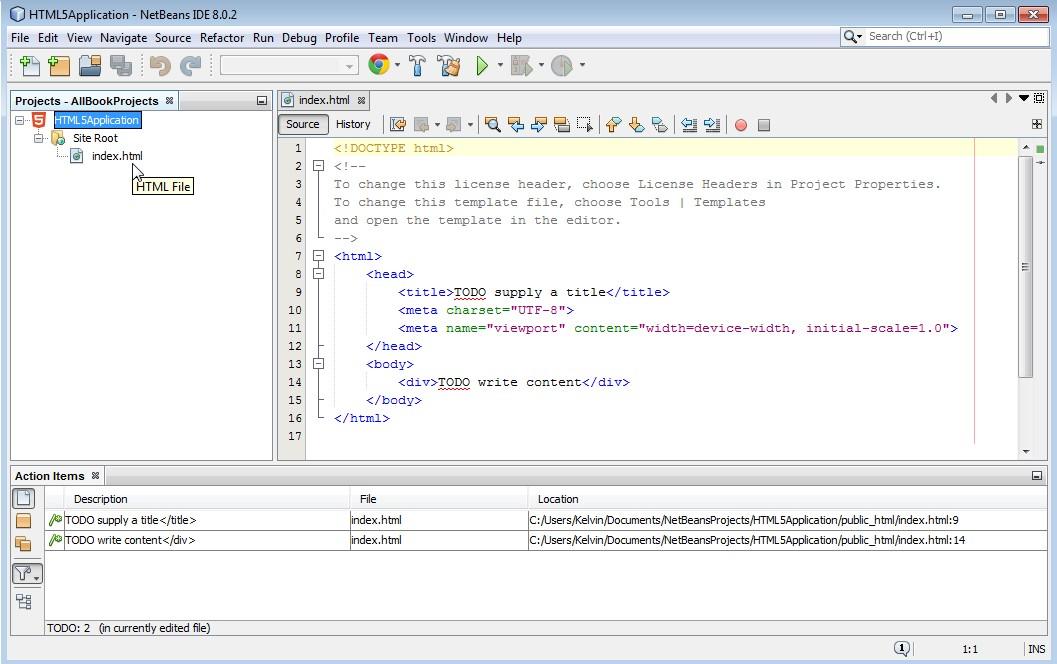


Figure 1-5. The HTML5 application project

By selecting and double-clicking the index.html file in the Projects window, you can open it in the Editor window and observe the content of the file. The contents are as follows:

<!DOCTYPE html>

<!--

To change this license header, choose License Headers in Project Properties.

To change this template file, choose Tools | Templates

and open the template in the editor.

-->

<html>

<head>

<title>TODO supply a title</title>

</head>

<body>

<div>TODO write content</div>

</body>

</html>

The first line declares the file to be an HTML file. The block that follows within the <!-- and --> tags is a comment block. The complementary <html></html> tags contain all the HTML code. In this case, the template defines the head and body sections. The head sets the title of the web page, and the body is where all the content for the web page will be located.

You can run this project by selecting Run ➤ Run Project or by pressing the F6 key. Figure 1-6 shows an example of what the default project looks like when you run it.

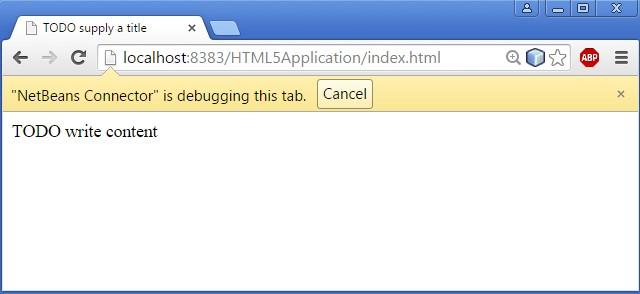


Figure 1-6. Running the simple HTML5 project

To stop the program, either close the web page or click the Cancel button in the browser to stop NetBeans from tracking the web page. You have successfully run your first HTML5 project. You can use this project to understand the IDE environment.

## The Relationship Between the Project Files and the File System

Navigate to the HTML5Application project location on your file system, for example with the Explorer OS utility in Windows 7. You can observe that in the project folder NetBeans has generated the nbProject, public\_html, and test folders. Table 1-1 summarizes the purpose of these folders and the index.html file.

Table 1-1. Folders and Files in a NetBeans HTML5 Project

|  |  |
| --- | --- |
| **NetBeans HTML5 project: folder/file** | **Purpose** |
| nbProject/ | This folder contains the IDE configuration files. You will not modify any of the files in this folder. |
| public\_html/ | This is the root folder of your project. Source code and assets from your project will be created in this folder. |
| public\_html/index.html | This is the default entry point for your web site. This file will be modified to load JavaScript source code files. |
| test/ | This is the default folder for unit testing source code files. This folder is not used in this book and has been removed from all the projects. |

# HTML5 Canvas Project

This project demonstrates how to set up the core drawing functionality for the engine as well as define a user control script. Figure 1-7 shows an example of running this project, which is defined in the project folder.

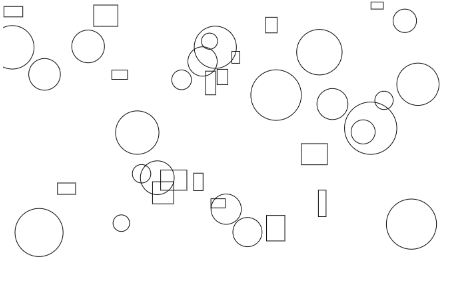


Figure 1-7. Running the HTML5 project with drawing core and user control.

The goals of the project are as follows:

* To learn how to set up the HTML canvas element
* To learn how to retrieve the canvas element from an HTML document for use in JavaScript
* To learn how to create a reference context to a HTML canvas, and use it to manipulate the canvas
* To get familiar with basic user control scripting.

# Drawing Core

This engine will use simple drawing code for the sole purpose of simulating the physics engine code. After all, the only thing the simulation needs to show is how simple objects interact after the physics engine code is implemented. Thus, advanced graphical functionalities such as, illumination, texturing, or shadow rendering will only serve to further complicate the code base. For that reason, a simple HTML5 canvas with primitive drawing support will serve the purpose of rendering the physics simulation during creation and debugging.

## Creating the HTML Canvas

In this step, you will create an empty HTML5 canvas for the drawing of all the objects.

1. Open the index.html file in the editor by double-clicking the project name in the project view, then open the Site Root folder, and double-clicking the index.html file.
2. To create your HTML canvas for drawing, add the following line in the index.html file within the body element

<table style="padding: 2px">  
 <tr>  
 <td>   
 <div>  
 <canvas id="canvas"></canvas>   
 </div>  
 </td>  
 </tr>  
 </table>

The code defines a canvas element with an id of canvas. An id is the name of the element and can be used to retrieve the corresponding element when the web-page is loaded. Notice that no width and height is specified in the code, this is because you will specify those attributes in the next step. You will use the canvas id to retrieve the reference to the actual canvas drawing area where you will draw into.

## Creating the Core Script

This section details the steps needed to create your first script, the drawing canvas initialization. This script will evolve to contain more core functionalities for the physics engine. For this step, you will only write the initialization code for the drawing canvas.

1. Create new folder named EngineCore inside the Site Root folder by right-clicking and creating a new folder.
2. Create a new JavaScript file within the EngineCore folder by right-clicking the EngineCore folder. Name the file Core.js.
3. Open the new Core.js file for editing
4. Create a static reference to gEngine by adding the following code.

var gEngine = gEngine || {};

gEngine.Core = (function () {  
 }());

gEngine.Core is where all the physics engine core functionality will reside.

**Note** All global variable names begin with a g and are followed by a capital letter, as in gEngine.

1. Inside the gEngine.Core you want to access and define the width and height of the canvas element. To do this you will create a variable mCanvas, and reference it to the canvas element of index.html such that you could set and modify the canvas attributes. You also need the variable mContext, which will keep a reference to all the methods and properties needed to draw into the canvas. Add the following code to accomplish these.

var mCanvas, mContext, mWidth = 800, mHeight = 450;  
 mCanvas = document.getElementById('canvas');  
 mContext = mCanvas.getContext('2d');  
 mCanvas.height = mHeight;  
 mCanvas.width = mWidth;

**Note** All instance variable names begin with a m and are followed by a capital letter, as in mCanvas.

1. Create an object variable mPublic, because you need to make some of the engine core variables and functions accessible by other scripts later in the development of the engine. For now, mPublic will only need to keep three variables accessible, that is the width and height of canvas, and the mContext to draw into the canvas.

var mPublic = {  
 mWidth: mWidth,  
 mHeight: mHeight,  
 mContext: mContext  
 };  
 return mPublic;

1. Finally, for the Core.js to be included in the simulation, you need to add it into the index.html file. To do this simply add the following code inside the body element.

<script type="text/javascript" src="EngineCore/Core.js"></script>

# User Control

In this section, you will be introduced to basic user control event handlers using JavaScript. This is to enable you to test your implementation in every step of the physics engine’s incremental development. For this chapter, the user control script will be used to test if you have correctly initialized the canvas and implemented drawing functionality properly.

## Create User Control Script

Let’s get started:

1. Create a new JavaScript File within the Site Root folder by right-clicking the Site Root folder. Name the file UserControl.js.
2. Open the new UserControl.js file for editing
3. Here you want to create a function that will handle all the keyboard input, let’s name the function userControl. This function will have a variable called keycode that will keep track of the user keyboard input. To do this add the following code inside the UserControl.js.

function userControl(event) {  
 var keycode;  
 }

1. Since some browsers handle input events differently, you want to know in which type of browser the simulation will run. Add the following code within the control function to distinguish between an IE key event handler and other browser key event handler.

if (window.event) { // IE  
 keycode = event.keyCode;  
 }  
 else if (event.which) { // Netscape/Firefox/Opera  
 keycode = event.which;  
 }

This script will enable you to handle keyboard input events from the browser as well as process the input and response accordingly. In this case, you want to test the canvas you just created in the last section. This testing can be achieved by drawing rectangles and circles when keyboard inputs are received, as detailed in the next section.

## Using the User Control Script

In this section, you will complete the UserControl.js file for this chapter by adding some user input responses to draw rectangle or circle in random positions on the canvas when F or G keys are pressed.

The control script will be triggered by the HTML onkeydown event. It is important to recognize that in the browser, each keyboard key is associated with a unique key code. For example, “a” is associated with a keycode of 65, “b” is 66, and so on.

**Note** The UserControl.js will evolve over the development to handle more keyboard inputs and more complex responses.

1. Open the UserControl.js file for edit.
2. You need to access the width and height of canvas, and the context to draw into the canvas. Add the following lines of code inside the control function.

var width = gEngine.Core.mWidth;  
 var height = gEngine.Core.mHeight;  
 var context = gEngine.Core.mContext;

1. Create a rectangle at a random position if the ‘F’ key (with key code value of 70) is pressed, and a circle if the ‘G’ key (with key code value of 71) is pressed. Add the following lines to accomplish this task.

if (keycode === 70) { //f  
 //create new Rectangle at random position  
 context.strokeRect(Math.random() \* width \* 0.8, // x position of center

Math.random() \* height \* 0.8, // y position of center

Math.random() \* 30 + 10, Math.random() \* 30 + 10); // width and height location  
 }  
 if (keycode === 71) { //g  
 //create new Circle at random position  
 context.beginPath();  
 //draw a circle  
 context.arc(Math.random() \* width \* 0.8, // x position of center

Math.random() \* height \* 0.8, // y position of center

Math.random() \* 30 + 10, 0, Math.PI \* 2, true); // radius  
 context.closePath();  
 context.stroke();  
 }

1. Next, for the UserControl.js to be included in the simulation, you need to add it into the index.html file. To do this simply add the following code inside the body element.

<script type="text/javascript" src="EngineCore/Control.js"></script>

1. Finally, you want HTML to handle the key pressing event. Open the index.html file to edit and add the onkeydown attribute to the body tag to call your JavaScript function control. Modify your index.html file so the body tag will look like the following.

<body onkeydown="return userControl(event);" >

Now if you run the project and press the key F or G, the simulation will draw either a circle or rectangle at a random position with random sizes as shown in Figure 1-7 above.

# Summary

By this point the physics engine’s basic drawing function has been initialized, and should be able to draw a rectangle and a circle onto the canvas with basic input required from the user. In this chapter, you have structured the source code that supports future increase in complexity with a simple way to draw rigid bodies. You are now ready to extend the functionalities and features of your project into a physics engine. The next chapter will focus on the core functionalities needed for any game or physics engine (engine loops, vector calculation), as well as evolving rectangle and circle into rigid body object oriented objects to encapsulate their drawing and behaviors.