

Extracted from:

# 3D Game Programming for Kids, Second Edition

Create Interactive Worlds with JavaScript

This PDF file contains pages extracted from *3D Game Programming for Kids, Second Edition*, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit <http://www.pragprog.com>.

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printed versions; the content is otherwise identical.

Copyright © 2018 The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

The Pragmatic Bookshelf

Raleigh, North Carolina

The  
Pragmatic  
Programmers

# 3D Game Programming for Kids

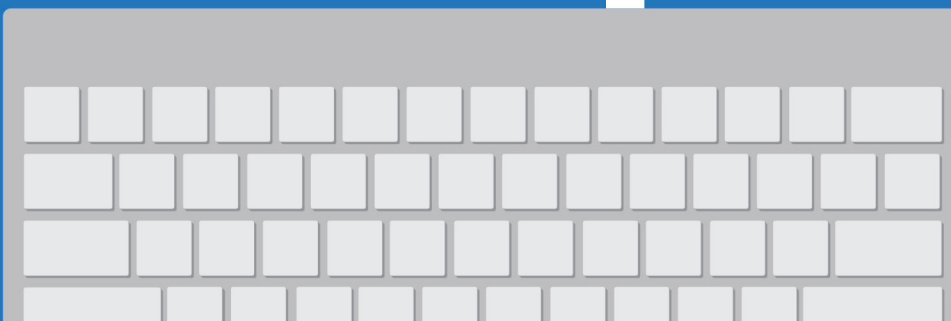
Second Edition



Create Interactive  
Worlds with JavaScript

Chris Strom

*edited by Adaobi Obi Tulton*



# 3D Game Programming for Kids, Second Edition

Create Interactive Worlds with JavaScript

Chris Strom

The Pragmatic Bookshelf

Raleigh, North Carolina



Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf, PragProg and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at <https://pragprog.com>.

The team that produced this book includes:

Publisher: Andy Hunt

VP of Operations: Janet Furlow

Managing Editor: Brian MacDonald

Supervising Editor: Jacquelyn Carter

Development Editor: Adaobi Obi Tulton

Copy Editor: Paula Robertson

Indexing: Potomac Indexing, LLC

Layout: Gilson Graphics

For sales, volume licensing, and support, please contact [support@pragprog.com](mailto:support@pragprog.com).

For international rights, please contact [rights@pragprog.com](mailto:rights@pragprog.com).

Copyright © 2018 The Pragmatic Programmers, LLC.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

ISBN-13: 978-1-68050-270-1

Book version: P1.0—July 2018

## Getting Started

Instead of drawing and moving shapes in this chapter, we're going to explore the JavaScript programming language. We can do this in the JavaScript console, so let's start by opening that. Refer to [Opening and Closing the JavaScript Console, on page ?](#), if you don't remember how.

The JavaScript console is a web programmer's best friend. Modern browsers pack the JavaScript console with all kinds of features that help programmers improve network performance, security, memory usage, and lots more. In this book, we use it to debug our code and to experiment with JavaScript. We talked about debugging in [Chapter 2, Debugging: Fixing Code When Things Go Wrong, on page ?](#). In this chapter, we experiment. And when programmers experiment, we play!

First, a quick tip...

### You Can Change Code in the Console (sort of)

If you make a mistake in your code and then press Enter, you can't just click on the mistake to change it.

```
> var name = "Alice";
✖ Uncaught SyntaxError: Invalid or unexpected token
> |
```

But, you can press the Up arrow key on your keyboard to bring up the last thing that you typed.

```
> var name = "Alice";
✖ Uncaught SyntaxError: Invalid or unexpected token
> var name = "Alice";
```

Then you can fix the problems and press Enter to try again.

---

#### Don't Type the Same Thing Over and Over in the JavaScript Console.

---



We programmers have to type code. But we programmers like tools that make typing easier. Use the up and down arrow keys to access and navigate the *history* of the code you type in the JavaScript console. Making small changes to code you already typed saves lots of time.

---

All right, let's have some fun with JavaScript in the console!

## Describing Things in JavaScript

Have you noticed how we introduce new things in JavaScript?

```
var speed = 10;
var title = '3D Game Programming for Kids';
var isCool = true;
```

The `var` keyword declares new things in JavaScript. It tells both the computer and the humans reading the code, “Get ready—something new is coming!”

### The `var` Keyword

The `var` keyword is short for *variable*. A variable is a thing that can change.

```
> var i
< undefined
> i = 0;
< 0
> i = 1;
< 1
> i = 2;
< 2
```

First, the variable `i` is set to the value of 0. Then, it is set to 1. Last, the variable is set to the value of two. The value that the variable points to changes—it *varies*—as the code does stuff (or as we type changes in the console).

We always use `var` when we first introduce a variable in our code. Without it, JavaScript can get confused about which value gets set to a variable and when. In other words, not using `var` can make buggy code that is hard to fix. So *always* use `var` when you introduce a variable...

...Except in the JavaScript console. Code in the JavaScript console is for quick experiments, so there’s little chance to create bugs. When writing these quick experiments, programmers save time by not typing `var` or ending lines with semicolons.

The other reason to skip `var` in the JavaScript console is that the value being set is not reported. When you set a variable in the JavaScript console without `var`, the value is reported after pressing `Enter`.

```
> title = "3D Game Programming for Kids"
< "3D Game Programming for Kids"
```

Weirdly, if you set a variable with `var` in the JavaScript console, the console reports the result as `undefined`.

```
> var title = "3D Game Programming for Kids"
< undefined
```

It's *not* undefined though. Typing the variable name alone and pressing Enter will report its value.

```
> title
< "3D Game Programming for Kids"
```

JavaScript is a great language. It can still behave oddly at times.

---

#### Always Introduce Variables with `var` (but Not in the JavaScript Console).

---



Code is better if you always use `var` when first introducing a variable in a project or inside a function. It's easier to read and less likely to lead to weird bugs.

But don't use `var` in the JavaScript console, because JavaScript is weird.

---

## Different Kinds of Things in JavaScript

JavaScript can set variables to lots of different things. It can set numbers, words, dates, and true-and-false values. JavaScript has two different ways to set “nothing” (JavaScript is weird). There are a couple of ways to list things. There are even ways to set a variable to 3D shapes!

We'll take a close look each of these—and ways to code with them.

## Code Is for Computers and Humans, Comments Are Only for Humans

We write code for both computers *and* people. Code is written for computers so computers can do stuff. Code is written for people so we can understand code and add features to it. Sometimes code can be a little hard for people to understand. When that happens, programmers add *comments* to their code.

Computers know that they're supposed to ignore comments. Programmers know that we should read them to better understand the following code.

In JavaScript, double slashes indicate comments. Type the following in the JavaScript console.

```
// Set today's date
date = new Date()

// Set January 1, 2050
date = new Date(2050, 0, 1)
```

Unless you've seen JavaScript dates before, you probably wouldn't know what that code does (especially that 0 is the first month in JavaScript). But, thanks to comments, it's pretty easy to figure it out.

---

**You Don't Need to Type the Comments (but You Should).**

---

The comments you see in this book are meant to give you helpful hints. You don't *have* to type them in. But you should. When you open your code later to make a change, comments will help you remember why you did things.



Really, instead of copying comments, you should add your own. If you have a hard time understanding something when you code it, chances are you will have a hard time understanding it when you come back to add new stuff. Comments are little hints for your future self!

---

Next, let's take a close look at the different kinds of JavaScript things, starting with numbers.