The Official ROBLOX Guide

Roblox Game Development in 24 Hours

Pearson

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Roblox Game Development

The Official ROBLOX Guide

in 24 Hours

Pearson

Roblox Game Development in 24 Hours: The Official Guide

Copyright © 2021 Roblox Corporation. "Roblox," the Roblox logo, and "Powering Imagination" are among the Roblox registered and unregistered trademarks in the U.S. and other countries. All rights reserved.

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, request forms, and the appropriate contacts within the Pearson Education Global Rights & Permissions Department, please visit www.pearson.com/permissions/.

No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-13-682973-7
ISBN-10: 0-13-682973-2

Library of Congress Control Number: 2021931227

ScoutAutomatedPrintCode

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Pearson cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the United States, please contact intlcs@pearson.com.

Executive Editor
Debra Williams Cauley

Acquisitions Editor
Kim Spenceley

Editorial Services
The Wordsmithery LLC

Managing Editor
Sandra Schroeder

Senior Project Editor
Tonya Simpson

Copy Editor
Charlotte Kughen

Indexer
Cheryl Lenser

Proofreader
Sarah Kearns

Editorial Assistant
Cindy Teeters

Cover Designer
Chuti Prasertsith

Compositor
Bronkella Publishing LLC

Contents at a Glance

Foreword

HOUR 1 What Makes Roblox Special? ... 1

2 Using Studio ... 19

3 Building with Parts ... 43

4 Building with Physics ... 59

5 Building Terrain ... 81

6 Lighting Environment ... 107

7 Atmosphere Environment ... 123

8 Effects Environment ... 141

9 Importing Assets ... 157

10 Game Structure and Collaboration ... 175

11 Lua Overview ... 193

12 Collisions, Humanoids, Score ... 215

13 Interacting with GUIs ... 235

14 Coding Animation ... 257

15 Sounds and Music ... 273

16 Using the Animation Editor ... 285

17 Combat, Teleporting, Data Stores ... 305

18 Multiplayer Code and the Client-Server Model ... 331

19 Module Scripts ... 345

20 Coding Camera Movements ... 357

21 Cross-Platform Building ... 371

22 Global Community Building ... 385

23 Monetization ... 395

24 Attracting Players ... 409

APPENDIX A Lua Scripting References ... 425

B Properties and Functions of Humanoid ... 429

Index ... 433

Table of Contents

HOUR 1: What Makes Roblox Special? **1**

Roblox Empowers Social Connectivity ... 2

Roblox Manages User Content ... 4

Roblox Enables Fast Prototyping and Iteration ... 7

Conceptualize with Ease ... 8

What's Inside Roblox's Engine ... 10

Free, Free, Free ... 13

Unlimited Possibilities ... 14

Express Your Own Aesthetic ... 14

HOUR 2: Using Studio **19**

Installing Roblox Studio ... 19

Using Studio Templates ... 22

Working with the Game Editor ... 24

Translating, Scaling, and Orienting Objects ... 30

Snapping ... 34

Collisions ... 35

Anchoring ... 35

Saving and Publishing Your Project ... 36

Playtesting ... 38

HOUR 3: Building with Parts **43**

Creating a Part ... 43

Changing a Part's Appearance ... 44

Creating Decals and Textures ... 48

HOUR 4: Building with Physics **59**

Working with Attachments and Constraints ... 60

Building a Door ... 62

Disabling CanCollide to Move a Player Through the Door 65
Adding Hinges and Springs 66
Using a Motor 74

HOUR 5: Building Terrain **81**
Using Terrain Tools to Generate Landscapes 82
Using the Edit Tab 85
Working with the Region Tab 94
Using Height Maps and Color Maps 100

HOUR 6: Lighting Environment **107**
Properties of World Lighting 108
Using Lighting Effects 113
Using SpotLight, PointLight, and SurfaceLight 116

HOUR 7: Atmosphere Environment **123**
Using Atmosphere Properties 124
Customizing Skybox 132

HOUR 8: Effects Environment **141**
Using Particles 141
Using Beams 145

HOUR 9: Importing Assets **157**
Inserting and Uploading Free Models 157
Importing with MeshParts and Asset Manager 164
Importing Textures 168
Importing Sounds 171

HOUR 10: Game Structure and Collaboration **175**
Adding Places in a Game 175
Collaborating in Roblox Studio 177
Creating and Accessing Roblox Packages in Roblox Studio 184

HOUR 11: Lua Overview 193
Using the Coding Workspace 193
Using Variables to Modify Properties 195
Adding Comments to Your Code 198
Using Functions and Events 199
Working with Conditional Statements 202
Understanding Arrays and Dictionaries 203
Using Loops 203
Working with Scope 206
Creating Custom Events 207
Debugging Code 209

HOUR 12: Collisions, Humanoids, Score 215
Introduction to Collisions 215
Detecting Collisions 220
Introduction to Humanoids 224

HOUR 13: Interacting with GUIs 235
Creating GUIs 236
Basic GUI Elements 243
Coding Interactive GUIs 244
Tweening 247
Layouts 248
Making a GUI Countdown 251

HOUR 14: Coding Animation 257
Working with Position and Rotation 257
Moving Objects Smoothly with Tween 265
Moving an Entire Model 268

HOUR 15: Sounds and Music 273
Creating a Soundtrack 273
Importing Music and Sound Assets 275
Creating Ambient Sounds 277
Triggering Sounds Using Code 278
Grouping Sounds 279

HOUR 16: Using the Animation Editor 285
Introduction to the Animation Editor 286
Creating Poses 287
Saving and Exporting Animations 291
Easing 293
Working with Inverse Kinematics 293
Animation Settings 296
Working with Animation Events 298

HOUR 17: Combat, Teleporting, Data Stores 305
Introduction to Tools 305
Teleportation 314
TeleportService 318
Using Persistent Data Stores 322
Data Store Functions 326
Protecting and Responding to Errors 328

HOUR 18: Multiplayer Code and the Client-Server Model 331
The Client-Server Model 331
What Are `RemoteFunctions` and `RemoteEvents`? 333
Server-Side Validation 337
Teams 338
Network Ownership 341

HOUR 19: Module Scripts 345
Getting to Know the Module Script 345
Understanding Client-Side Versus Server-Side Module Scripts 349
Using Module Scripts: Game Loop 351

HOUR 20: Coding Camera Movements 357
Introduction to Cameras 357
Coding a Camera Move 360
Using the Render Step 362
Offsetting the Camera 363

HOUR 21: Cross-Platform Building 371
Improving Game Performance 371
Improving Your Scripts 375
Making Your Game Mobile-Friendly 377
Console and VR 381

HOUR 22: Global Community Building 385
Introduction to Localization 385
Global Compliance 389
Privacy Policies: GDPR, CCPA, and You 390

HOUR 23: Monetization 395
Game Passes: One-Time Purchases 395
Selling Your Game Pass in Game 397
Developer Products: Consumables 399
Roblox Premium 401
Developer Exchange: Earn Real Money from Your Game 403

HOUR 24: Attracting Players 409
Game Icons, Thumbnails, and Trailers 409
Updates 413
Advertising and Notifications 414
Analytics 420

APPENDIX A: Lua Scripting References 425
Modifying Properties That Are Data Type and Enumerations 425
Conditional Structures 426
Expanding Lua Knowledge 428

APPENDIX B: Properties and Functions of Humanoid 429

Index 433

Foreword

Imagine a virtual universe built by a global community of artists, coders, storytellers, and everything in between. In this dream, people from all corners of the world come together to create and share millions of experiences with their friends and learn from one another. It would be a universe driven by imagination, where anything could be made and experienced, regardless of device, location, or time period. What if I told you this digital utopia has been a reality for over a decade?

When Erik Cassel and I co-founded Roblox in 2004, our vision was to create an immersive, 3D, multiplayer, physically simulated space where anybody could connect and have fun doing things together. In the early days of Roblox, we were fascinated by what people were making. We saw experiences where people wanted to manage their own restaurant, survive a natural disaster, or imagine what it's like to be a bird. Seventeen years later, as I gaze into the future, it's obvious this platform can become so much more.

Roblox is ushering in a new category of human co-experience, blurring the lines between gaming, social networking, toys, and media. Our team has found that the millions of daily Roblox users aren't just logging on to play games but are coming together to build communities, stories, and experiences with friends and strangers alike.

As we continue our mission to build a human co-experience platform that enables shared experiences among billions of users, there has never been a better time to join a global community of creative individuals who are contributing such amazing works to our platform. Developing 3D experiences is not only fun, but it also provides the skills and knowledge to launch a career in computer science, design, art, and so much more. Many top developers on our platform have used the money they earned from their creations on Roblox to pay for their college tuition, start their own game development studios, or put a down payment on a house for their parents.

I believe that ultimately Roblox will lead us to the creation of the Metaverse, a full-fledged digital reality that will complement our physical one. We can start to imagine a day where people aren't just coming to Roblox to play and socialize but also to hold business meetings or go to school. As the possibilities of the Metaverse increase by the day, so too does the need for innovative and creative developers who can shape the experiences we've been dreaming about in science fiction for years.

I personally invite you to join the world of Roblox not just as a player but also as a creator. Learning to develop both games and immersive 3D experiences can help connect millions of people worldwide through the power of play and create a community not defined by borders, languages, or geography. If you're at all interested in coding, game design, or the immersive 3D world of Roblox, consider peering through these pages and embracing your wildest, most creative ideas. The Metaverse depends on creators just like you.

Your imagination awaits,
David "Builderman" Baszucki
Founder + Chief Executive Officer
Roblox Corporation

About the Author

Genevieve Johnson is the senior instructional designer for Roblox, the world's largest user-generated social platform for play. In her role, she oversees creation of educational content and advises educators worldwide on how to use Roblox in STEAM-based learning programs. Her work empowers students to pursue careers as entrepreneurs, engineers, and designers. Before working at Roblox, Johnson was educational content manager for iD Tech, a nationwide tech education program that reaches more than 50,000 students ages 6 to 18 each year. While at iD Tech, she helped launch a successful all-girls STEAM program, and her team developed educational content for more than 60 technology-related courses with instruction on a variety of subjects, from coding to robotics to game design.

About the Contributors

Ashan Sarwar is a Roblox developer who has been using Roblox Studio since 2013. He is the owner of LastShot, a Roblox shooting game on Roblox.

Raymond Zeng is a Roblox developer who loves programming and teaching all levels of programmers. He has a YouTube channel under the name of MacAndSwiss where he teaches Lua, talks about Roblox news, and showcases his programming projects.

Theo Docking has been working as a gameplay programmer for four years. He likes working on exciting projects, pushing Roblox to the limit, and meeting amazing people along the way. He loves playing with Roblox's physics engine and writing back-end code for NPCs, cars, and more. When he's not writing code, he's drawing up game design plans or playing Ultimate Driving to get some fresh air.

Joshua Wood discovered Roblox in 2013 and started making his own games a year later. He is the developer of Game Dev Life, which has had more than a million play sessions. He's also the owner of DoubleJGames.

Swathi Sutrave is a self-professed tech geek. She has been a subject matter expert for several different programming languages, including Lua, for corporations, start-ups, and universities.

Henry Chang is a computer graphics artist who practices in multiple mediums, including 3D, 2D, graphics, and animation. He is a self-starter and has been involved in interactive media opportunities. For more information, visit https://www.henrytcgweb.com/.

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you email, please be sure to include this book's title and author, as well as your name, email address, and phone number. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: community@informit.com

Reader Services

Register your copy of *Roblox Game Development in 24 Hours: The Official Guide* at www.informit.com/register for convenient access to downloads, updates, and corrections as they become available. To start the registration process, go to informit.com/register and log in or create an account.* Enter the product ISBN (9780136829737) and click Submit.

*Be sure to check the box that you would like to hear from us to receive exclusive discounts on future editions of this product.

HOUR 2
Using Studio

What You'll Learn in This Hour:

- How to install and launch Roblox Studio
- How to use Studio templates
- How to navigate game editor
- How to create a part
- How to translate, scale, and orient parts
- How to save and publish your project
- Playtesting

Now that we've explored the culture and features that make Roblox special, you can start to unleash your creativity with Roblox's free game engine, Roblox Studio. Roblox Studio is a playground for developers to create, share, and play their games on the Roblox website. What's great about this platform is that you can easily build everything from volcanic islands to urban cityscapes and then drop a character into that world to immediately start playing. Imagine a huge playground filled with all the tools you need to build imaginary worlds—that's Roblox Studio.

In this hour, you'll learn how to install Studio, and then you'll learn how to use Roblox Studio with the help of templates. You'll also learn how to arrange your workspace to hold objects in the 3D world, the difference between saving and publishing your project, and finally how to test your game before publishing it to the public.

Installing Roblox Studio

We've explained how Roblox Studio is a free and immersive platform for game developers to build different terrains, cities, buildings, race games, and much more. You don't need years of coding experience or a degree to make fun games; all you need is your imagination and hands-on learning in the Roblox Studio. Roblox Studio is extremely intuitive to use. Because Roblox is cross-platform, developers can install Studio on both Windows and Mac systems.

Use the following steps to install Studio:

1. Go to https://www.roblox.com/create.
2. Click Start Creating and then click the Download Studio button in the pop-up window.
3. Navigate to the folder where you have downloaded Studio and double-click the file to install it.

NOTE

System Requirements

For Roblox Studio to run efficiently, there are some OS/hardware specifications:

- Roblox Studio cannot run on Linux, Chromebooks, or mobile devices such as smartphones.
- A Windows computer with at least Windows 7 installed, or a MacBook with version macOS10.10.
- A minimum of 1 GB of system memory.
- Internet access to download Studio and updates. It also lets you save projects (publish) to your Roblox account.

For an enhanced Studio experience, you should also have these things (not mandatory):

- A mouse with a scroll wheel, preferably a three-button mouse.
- A video card that's dedicated and not an integrated card.

Troubleshooting the Installation

If you've followed the necessary steps to install Studio but you're experiencing installation conflicts, there are a few things you can do to troubleshoot the errors:

- If you've added new hardware or drivers recently, remove and replace the hardware to determine if it's causing the problem.
- Run diagnostics software and check information on troubleshooting the operating system.
- Restart the computer.
- Uninstall and delete all the Roblox files and reinstall the latest Studio again, if required.

If you are still finding errors, you can also reference the Roblox Support forums online for additional tips.

Opening Roblox Studio

Once you are done installing the Roblox Studio, you need to open it:

1. Double-click the desktop icon if you are on Windows or click the Dock icon if you are on a Mac to open a login window (Figure 2.1).
2. Enter your Roblox username and password.
3. Click the Log In button.

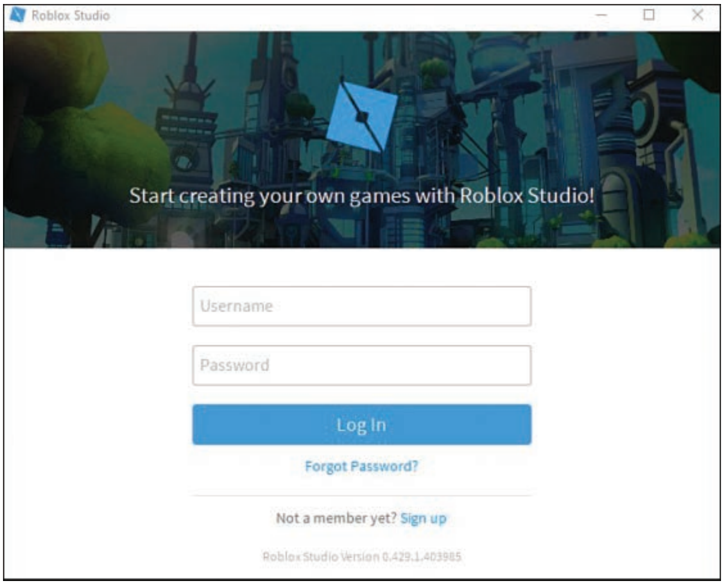

Roblox Studio
Start creating your own games with Roblox Studio!
Username
Password
Log In
Forgot Password?
Not a member yet? Sign up
Roblox Studio Version 0.429.1.403985

FIGURE 2.1
Roblox Studio login window.

Once you are logged in, you see a page with different templates and a menu sidebar with New, My Games, Recent, and Archive (Figure 2.2).

The following sections provide a quick introduction to these templates and the rest of Studio; then you can begin experimenting with the utilities of Studio.

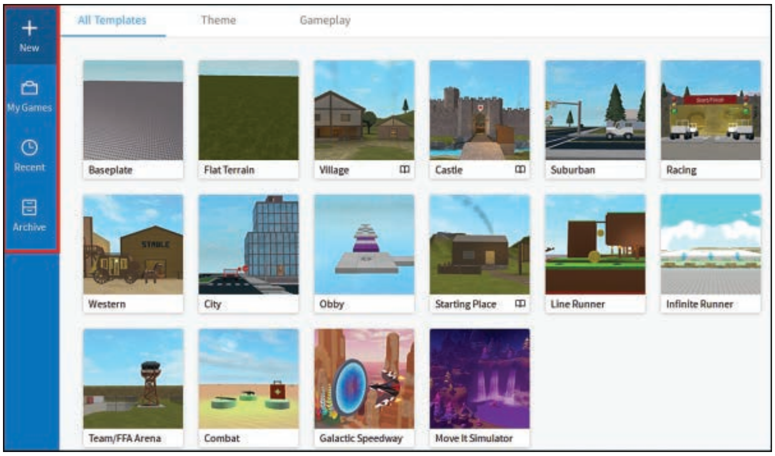


FIGURE 2.2
Roblox Studio home screen.

Using Studio Templates

When you first open Roblox Studio, under New, you see three tabs: All Templates, Theme, and Gameplay. Templates are prebuilt projects, and you can use them as a guide to build your own game world.

All Templates

The All Templates tab (Figure 2.3) is a combination of the Theme and Gameplay tabs. You can use these templates as a start for your games. For example, if you're building a medieval game, the Castle theme is equipped with feudal details, or if you want to build an interactive obby, you can build off the Obby gameplay template. Two simple templates are a good place to start:

- **Baseplate:** This is a popular choice to start with. The baseplate itself is easy to delete, leaving a blank canvas to work with.
- **Flat Terrain:** Has a flat plane of grass terrain instead of a baseplate. You can modify or clear the terrain using the terrain editor.

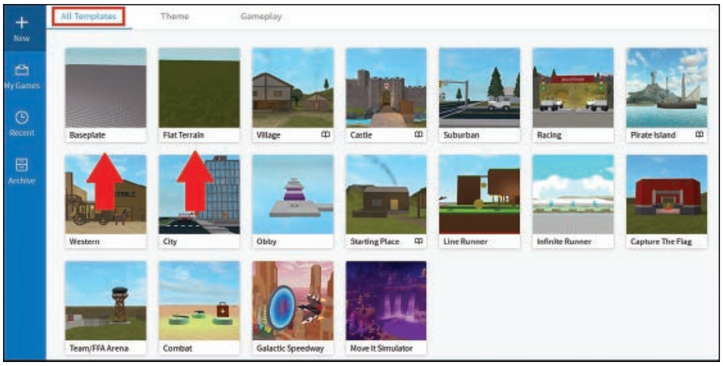


FIGURE 2.3
Roblox Studio home screen lists various templates available, such as simple templates Baseplate and Flat Terrain.

Themes

Themes are a combination of gameplays and more, and together they make a new world. It sets a mood for your game—for example, a space combat game will have asteroids and other galactic components. Roblox provides some prebuilt themes that are ready to use and modify however you would like. As you explore the game world, descriptions point out its use case or features, including tips on how the effects were created in case you want to re-create them yourself.

An example of a prebuilt theme is Village (Figure 2.4). You can explore the houses in the village and move along the pathway through the town, which leads you to a river, a bridge, and finally the dock, across which you can see small islands.

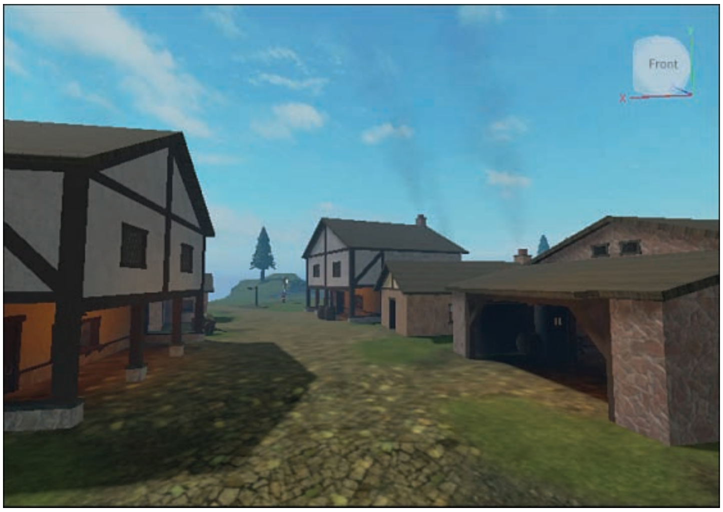


FIGURE 2.4
Example of a prebuilt *Village Theme* available in Studio.

Gameplay

Some templates include interactive gameplay. For example, this can include Team Deathmatch, Control Points, Capture the Flag (Figure 2.5), and more. A great thing about these templates is that developers can take them apart and extract any specific facet that they want—for example, using in-game radar or team spawn points. These templates help with components such as what a player can do in a game, what the goals are, and how a game can be modified.

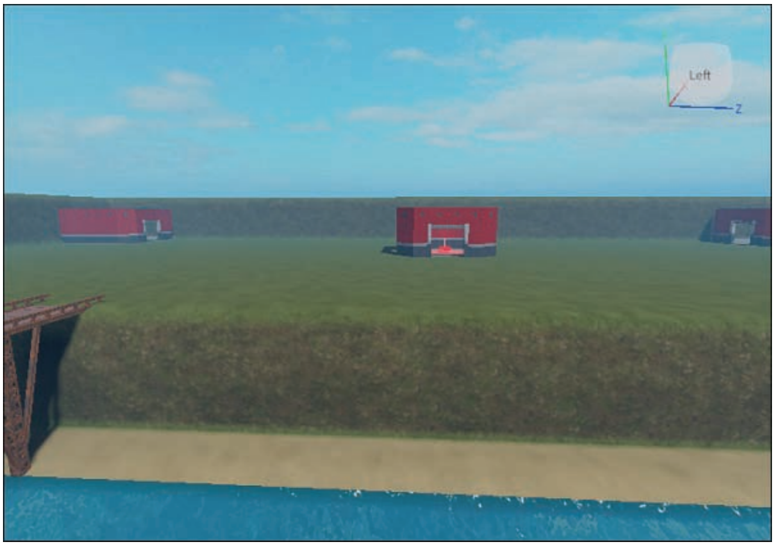


FIGURE 2.5
Example of a prebuilt *Capture the Flag* gameplay template.

Working with the Game Editor

Now that we've familiarized ourselves with Studio's homepage, let's click on the Baseplate template to get started. This opens the game editor (Figure 2.6).

The game editor is, as the name suggests, a place where you can create, modify, or test your game. At the top of the game editor, you see different tabs on the menu bar (Figure 2.7).

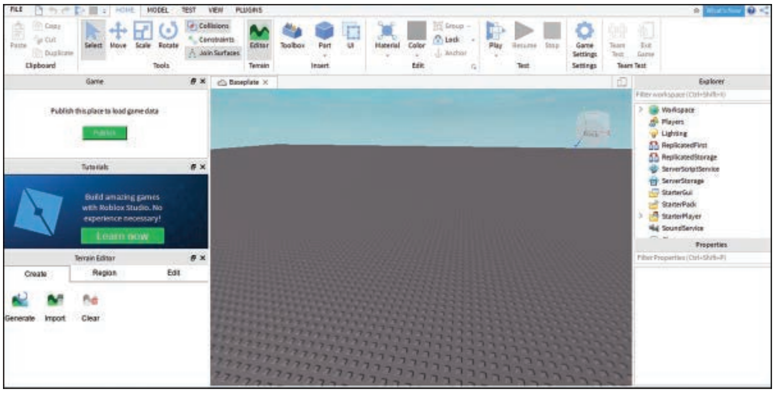


FIGURE 2.6
The game editor enables you to create, modify, or test your game.

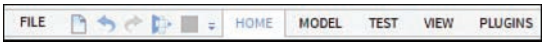


FIGURE 2.7
Roblox Studio menu bar.

- **Home tab:** A concise tab of all the features that are frequently used. These features are on the Home tab for easy access.
- **Model tab:** Has more building tools apart from move, scale, and rotate. It's also where you can create spawn locations and special effects such as fire and smoke.
- **Test tab:** Helps for testing your game. There are two options underneath: Run and Play. Run will run a simulation of what will happen to the bricks and surrounding elements, and Play will let you play your game.
- **View tab:** Lets you toggle the different windows available in the Roblox Studio. If you need to use a window that is closed, you can find them under the View tab.
  - The main windows are Explorer and Properties, which are discussed detail in later in this section.
  - The Actions section has several display features. You can take screenshots or record videos here and also toggle between full screen and windowed views.
- **Plugins tab:** An add-on to Studio. These are generally not included by default. Plugins add new custom behavior and features. You can either install plugins made by the Roblox community or create your own plugins.

Below the menu bar is a ribbon bar (Figure 2.8). The tool options change as you move between menu bar tabs.

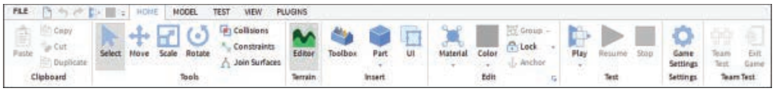


FIGURE 2.8
Roblox Studio ribbon bar.

In the following sections, we explain some of the editor's basic features and most frequently used features and discuss how to prepare your project for publishing on Roblox.

Arranging the Game Editor Workspace

Since this is the first time you are opening the game editor, extra windows that you don't require right now will automatically open on the left side. To organize the workspace in an optimal way, close the extra windows so you have more space to create.

By default, the Explorer and the Properties windows will be open (Figure 2.9), aligned one beneath the other on the right side.

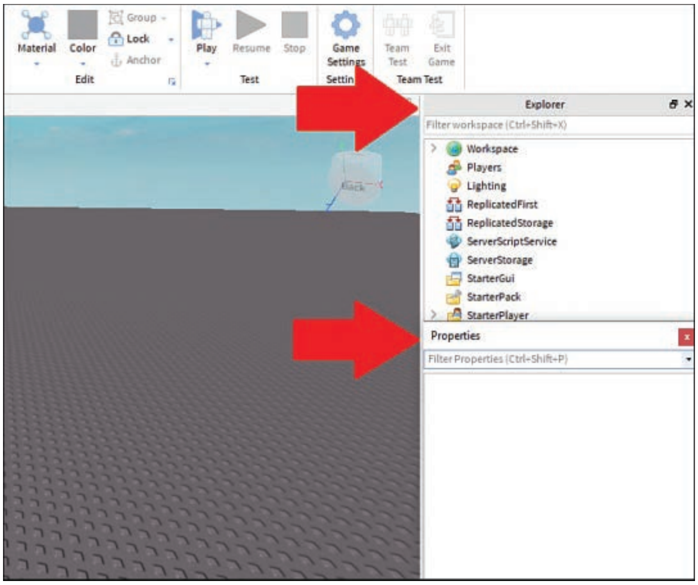


FIGURE 2.9
Workspace arrangement with the Explorer and Properties windows one below the other.

NOTE

Some Features of the Game Editor Workspace

The next time you relaunch Roblox Studio, your workspace arrangement remains intact. It is a one-time fix, unless you undo your arrangement.

When the Property window undocks, it gets difficult to dock it back below the Explorer window. It either docks itself aside or over the Explorer window. To fix this, undock both the windows and close them. Go to the View tab, open the Explorer window, dock it on the right-hand side, and then close it. Do the same with the Properties window and close it. After all this, reopen the Explorer and then the Properties window. This will align them one above the other.

Working with the Explorer Window

The Explorer window is the hierarchical representation of all the objects used in your game. It is the most crucial window because it lists all the organizing, viewing, and testing features of a Roblox game.

It uses the concept of parenting to organize all the objects. The object Game is hidden at the top of the hierarchy. For example, in Figure 2.10, you can see Workspace parent has the following children nested underneath: Camera, Terrain, and Baseplate.

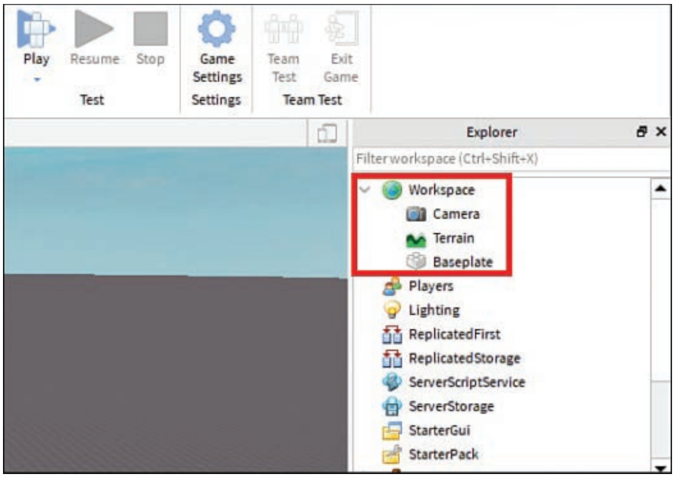


FIGURE 2.10
Objects nested under Workspace in the Explorer window.

If you want to create more child objects, you can hover over Workspace and click the plus symbol to the right (Figure 2.11). This will list all the objects that you can create. You can also drag and drop it into the desired parent object.

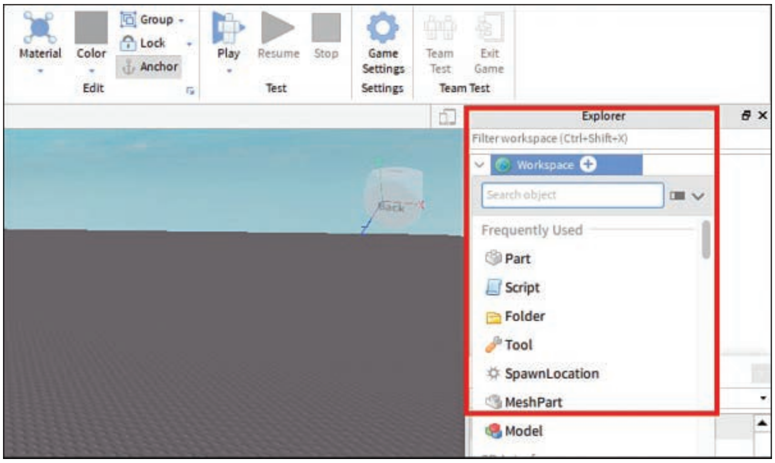


FIGURE 2.11
Add more children to your Workspace.

One of the most important children you will work with is a part, which is the foundational building block of Roblox. These physical 3D objects are also known as bricks, and when they are in the Workspace, they can interact with each other.

Creating a Part

To create a part, from the Home tab, navigate to the Insert menu in the ribbon bar and click Part (Figure 2.12).

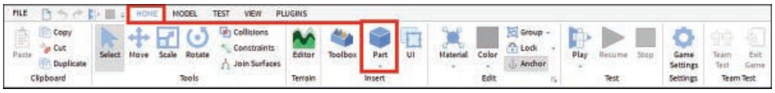
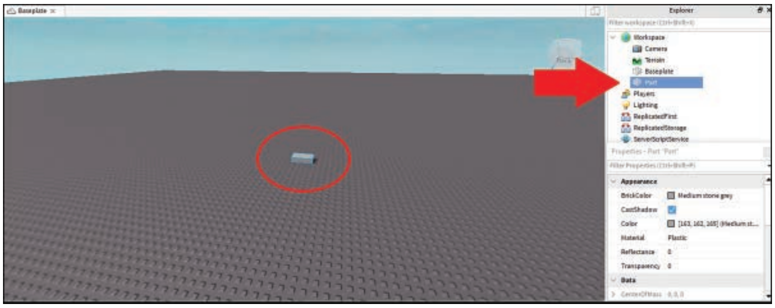


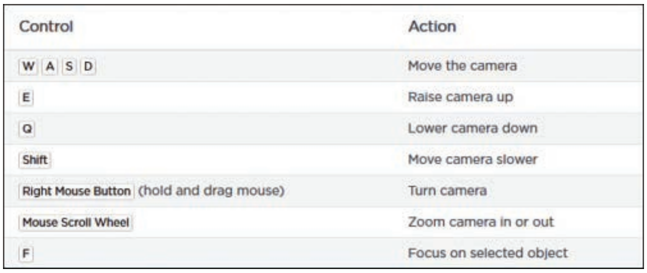
FIGURE 2.12
Create a part.

A part will appear at the exact center of your camera view (Figure 2.13). Use the **camera controls** shown in Figure 2.14 to move your camera, rotate the view, and zoom in and out.



BrickColor	▨ Medium stone grey
CastShadow	✓
Color	▨ [163, 162, 165] (Medium st...
Material	Plastic
Reflectance	0
Transparency	0
CenterOfMass	0, 0, 0

FIGURE 2.13
Part appears in your baseplate and in your Explorer.



Control	Action
W A S D	Move the camera
E	Raise camera up
Q	Lower camera down
Shift	Move camera slower
Right Mouse Button (hold and drag mouse)	Turn camera
Mouse Scroll Wheel	Zoom camera in or out
F	Focus on selected object

FIGURE 2.14
Camera controls.

To give your new part a name, do the following:

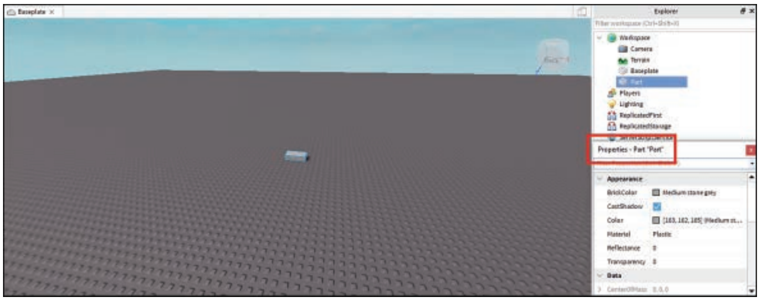
1. Double-click the part in your Explorer window.
2. Rename the part. Roblox convention is for parts to be named in PascalCase, which means the first letter is capitalized—for example, EndZone or RedBrick.

Note that your name can contain spaces, but we won't use spaces at this point in case we want to be able to access the part via code later.

You can use the Explorer to select and work with parts even if you can't see them in the game editor window.

Working with the Properties Window

When you add a part to your Workspace, you'll notice the Properties window (Figure 2.15) fills with information.



BrickColor	▨ Medium stone grey
CastShadow	✓
Color	▨ [163, 162, 165] (Medium st...
Material	Plastic
Reflectance	0
Transparency	0
CenterOfMass	0, 0, 0

FIGURE 2.15
The Properties window lists all the details about the newly added part.

Like any object, a part has properties such as size and color, and the Properties window shows all these details about how an object looks and behaves. In the next chapter, we'll go into further detail about properties of a part and how you can manipulate them.

Translating, Scaling, and Orienting Objects

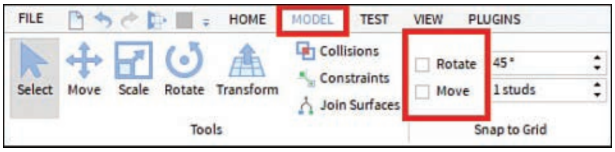
You've learned how to create a part; now you can make it move! In Roblox Studio, it is possible to move (translate) and rotate (orient) objects in the scene. There are multiple ways to get the same results, but in this section, we will solely use the Roblox Studio default tools and keyboard shortcuts.

There are two settings you can use to get greater control when moving parts: snapping and collisions.

- **Snapping** is the amount a part will move, scale, or rotate at a time. Snapping is useful when creating items that need to be exactly aligned, like how walls of buildings need to be at 90-degree angles.
- **Collisions** happen when two objects (or rigid bodies) intersect or get within a certain range of each other.

Because these two settings are most used when playing with two or more parts, turn them off for now while you freely move a single part around. Later, you'll turn them back on when we discuss how they work.

- **To turn OFF snap:** In the Model tab, uncheck the box next to Rotate or Move (Figure 2.16).



☐ Rotate 45°
☐ Move 1 studs

FIGURE 2.16
Turn off snap.

▶ **To turn OFF collisions:** In the Model tab, collisions are on if the button is highlighted gray. Click the Collisions button to toggle it off (Figure 2.17).

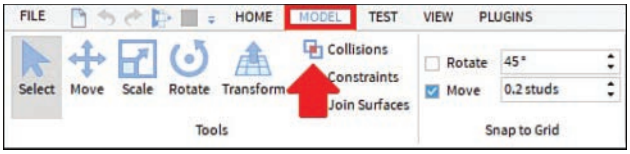


FIGURE 2.17
Turn off collisions.

Translating

Now you can freely start translating, or moving, objects. Go to the Model or Home tab and click the Move icon (Figure 2.18).

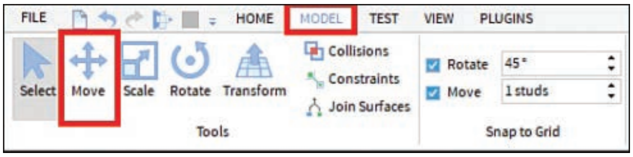


FIGURE 2.18
Move tool.

Now, a gizmo should appear on the selected objects. When you click, hold, and drag one of the arrows, the object moves along that axis (Figure 2.19).

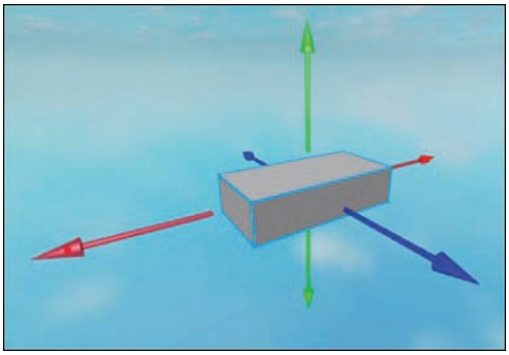


FIGURE 2.19
Moving the gizmo.

Scaling

To **scale** objects, go to the Model or Home tab and click the Scale icon (Figure 2.20).

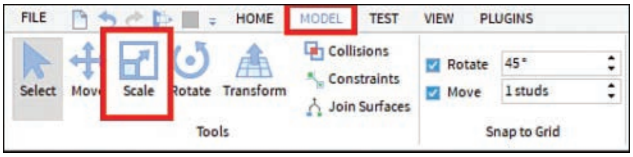


FIGURE 2.20
Scale tool.

The gizmo should appear again, this time with orbs on selected objects. When you click, hold, and drag one of the orbs, the object scales along that axis (Figure 2.21).

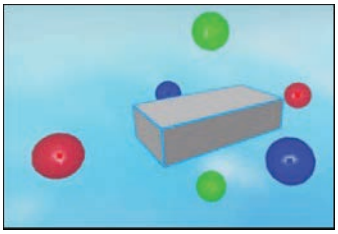


FIGURE 2.21
Scaling a gizmo.

If you want to scale on two sides simultaneously, hold Ctrl (Windows) or Command (Mac) while clicking, holding, and dragging one of the orbs.

If you want to scale while keeping the current proportions, you can do so by holding Shift while scaling.

Rotating

To rotate objects, go to the Model or Home tab, and click the Rotate icon (Figure 2.22).

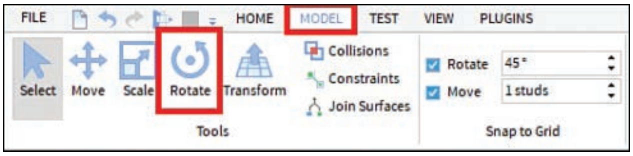


FIGURE 2.22
Rotate tool.

Another gizmo should appear, now with orbs and circular, connecting lines on selected objects (Figure 2.23). When you click, hold, and drag one of the orbs, the object will rotate along that axis.

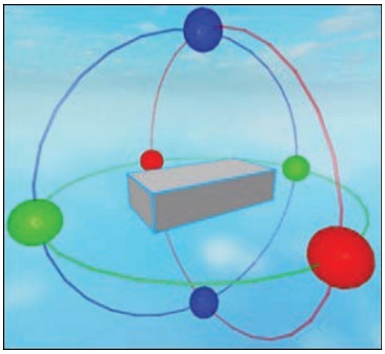
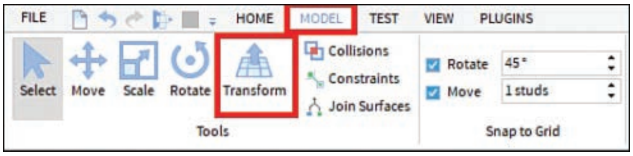


FIGURE 2.23
Rotating a gizmo.

Transforming

The transform tool (Figure 2.24) is particularly important as an all-in-one building tool. It enables multiple moves, scales, and rotations within one continuous operation. Think of it as a bundle of move, scale, and rotate. Basically, it can transform your part in any way possible. It also can lock an axis and snap to the grid.



Rotate 45°
Move 1 studs

FIGURE 2.24
Transform tool.

With your part selected, click on the transform tool and markers for manipulation appear around your part (Figure 2.25).

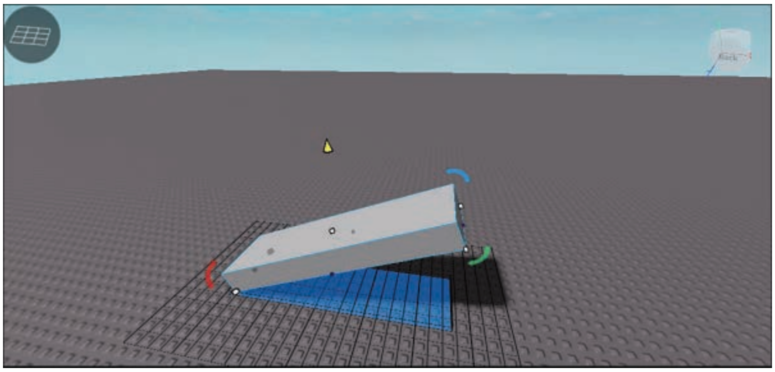


FIGURE 2.25
Using the Transform tool.

- The yellow cone is used to move the part on different planes on the Y axis. We can drag the part on its own plane once the plane is set.
- The red, green, and blue arcs are used to rotate the part by 360 degrees on the X, Y, and Z axes.
- The white boxes are used to scale the side of the part to which they are attached. The scaling happens in the measurement of studs, which is the measurement of each single square that forms the baseplate.

Snapping

Now that we understand the basics of moving a single part, let's revisit snapping and collisions. As a reminder, snapping is the amount a part will move, scale, or rotate at a time, and it allows you to align an object perfectly. There are two types of snapping: Rotation or Move.

- **Rotation** snapping enables you to turn an object by the given number of degrees. In this case, all objects will rotate 45 degrees each step.
- **Move** snapping counts for both moving and scaling. In this case, any object moves for one stud each step. Objects scale one stud each step.

Keep in mind that when you scale from the center of an object, it will scale one stud on both sides. It will then equal two studs total.

To turn snap back on, you will check the box next to Rotate or Move in the Model tab. Then, in the Rotate or Move fields, you can adjust your setting by the number of studs you want to move (Figure 2.26).

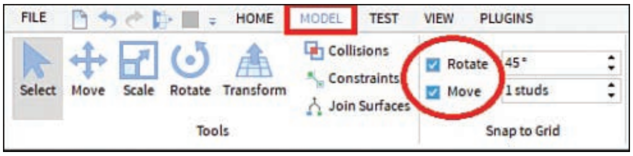


FIGURE 2.26
Snapping options.

Collisions

You can turn collisions back on and notice how they affect movement. In Roblox Studio, the collisions feature lets you control whether parts can move through each other. When collisions are on, you can't move a part into any place where it overlaps another part.

To turn collisions back on, click the Collisions button in the Model tab. This toggles it on and highlights it gray (Figure 2.27).

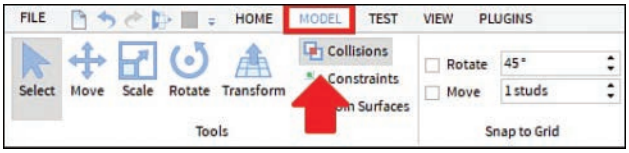


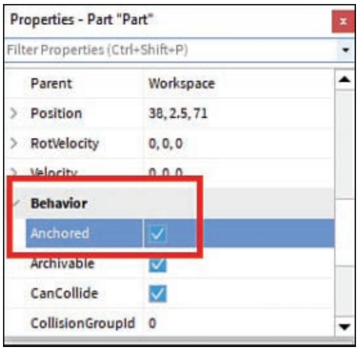
FIGURE 2.27
Collisions on.

Now as you move parts, you may notice a white outline whenever a part touches another part. This indicates that a collision is happening. We'll talk more about collisions in later hours.

Anchoring

We've talked a lot about making parts move in this chapter, but what if you don't want a part to move? If you want a part to be immobile, you need to anchor it. When you anchor a part, it remains static even when you're playing the game and other players and objects run into it. To anchor a part, do the following:

1. Go to the Properties window.
2. Scroll down to Behavior.
3. Check Anchored (Figure 2.28).


Properties - Part "Part"

Filter Properties (Ctrl+Shift+P)

Parent	Workspace
Position	38, 2.5, 71
RotVelocity	0, 0, 0
Velocity	0, 0, 0
Behavior	
Anchored	✓
Archivable	✓
CanCollide	✓
CollisionGroupId	0

FIGURE 2.28
Anchoring a part.

You can also easily Anchor and Unanchor parts with the Anchor button located in the bottom of Model tab or Home tab (Figure 2.29).

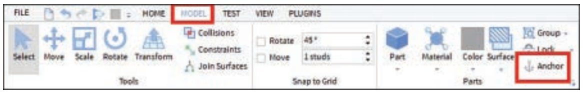


FIGURE 2.29
Anchor button.

▼ TRY IT YOURSELF

Anchoring Parts

To practice anchoring parts, do the following:

1. Create a part.
2. Move it left.
3. Rotate it 90 degrees with Snap to Grid.
4. Check the Properties window to see if it's anchored.

Saving and Publishing Your Project

Now that you are creating in the game editor, you will want to save your progress on projects from time to time because you don't want to lose any of the work you've accomplished. When you're ready for people to enjoy your creation, you may also want to publish it.

Saving Your Project

Roblox doesn't autosave your projects for you, so you need to save them. There are two places where you can save projects:

- **On your local desktop:** On the game editor menu bar, click File at the top-left corner, and then click Save to File. This retains the template name and saves the project as an .rbxl file. Instead, if you choose the Save to File As option in the same drop-down menu, you can rename the file (Figure 2.30).
- **On the Roblox server:** You can also save your project on the Roblox server by using the Save to Roblox As option in the same drop-down menu. This saves your work to a secure place in the Roblox Server but does not make it accessible to the public.

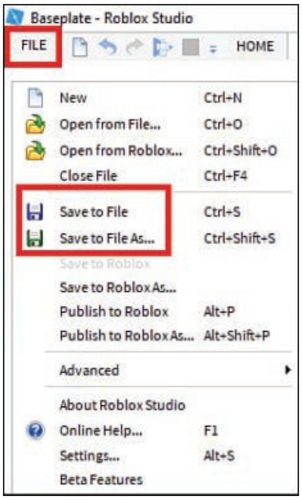


FIGURE 2.30
The Save to File commands are under the File option.

Publishing Your Project

What's the use of creating a game if no one can play it? To make it public and monetizable, we need to Publish the project by choosing the option Publish to Roblox. Publishing makes your game public and allows other players on Roblox to play it. Following are the steps to Publish to Roblox:

1. Select File, Publish to Roblox to open the publishing window.
2. Enter a name and an optional description.
3. When ready, click the Create button.

Reopening Your Project

When you want to reopen the project you were working on, you can find it on the Studio home screen (Figure 2.31) as follows:

- **File:** Select File, Open.
- **My Games:** If you have published your game to Roblox, your game will be in My Games.
- **Recent:** Look in Recent for all files that you've recently had open.

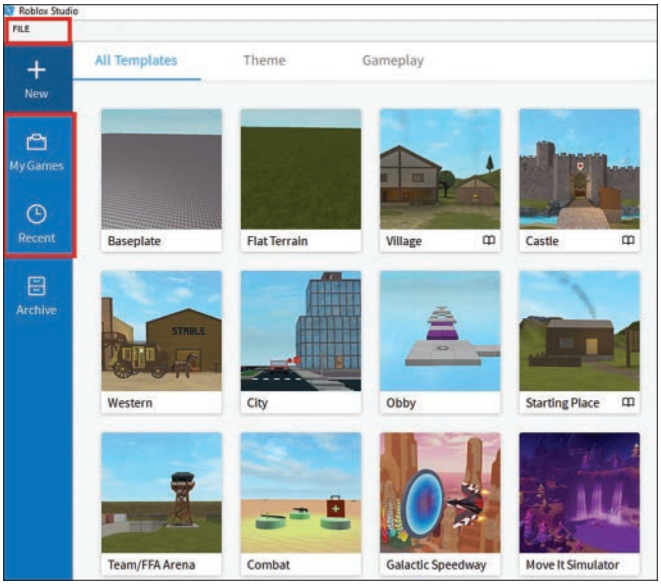


FIGURE 2.31
Reopen previous projects from the Studio home screen.

Playtesting

Playtesting is the process of playing the game to make sure everything works and figuring out how to make it even better. Don't skip this step because it's critical for a successful game. It's good practice to playtest your game whenever you make changes. You should also test your game in various modes. You can make changes in Play mode, but those changes won't be saved. You'll have to do them again when you go back to editing.

TIP

Playtesting Practices

When you playtest, do the following:

- Make sure your game works, particularly changes you just made.
- Look for areas that can be improved.
- When you are exploring or playtesting templates, make sure you thoroughly look at how the parts are named and grouped together.

Playtesting Your Game

To playtest your game, follow these steps:

1. Save your game. Don't forget to change the filename.
2. Press the Play button in the top menu bar. You can also find the Play button in the Home tab under the Test menu (Figure 2.32).

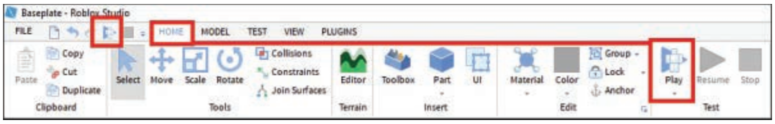


FIGURE 2.32
The Play button for playtesting your game.

Stopping Playtesting

To stop playtesting, press the red Stop button either in the top menu bar or under the Test menu (Figure 2.33). Stop the Playtest before making changes. Again, the reason for this is because you can't save changes in lay mode.

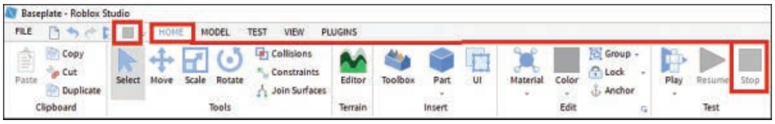


FIGURE 2.33
The Stop button for playtesting your game.

▼ TRY IT YOURSELF

Playtesting Practice

Playtest the following two templates:

- Village
- Obby

Before playtesting, you can modify the places where the parts are placed. You can drag and drop parts and watch how their properties change in the Properties window, and you can modify materials or delete them. Don't forget to save it or publish the template under a new name, and if you try to add parts or effects, make sure they are not in playtest mode.

Summary

In this hour, you've seen how easy it is to use Roblox Studio to create and share games with millions of players. You learned how to install and use the Roblox Studio, as well as how to arrange the workspace, make changes to your template, and save and publish games on Roblox to share them with the public. You also learned how to playtest your changes to ensure the success of your game.

Q&A

Q. What needs to be done if Studio isn't installing?

A. Make sure your system has the minimum system requirements. If it doesn't and Studio still ends up installing, there might be problems running Studio.

Q. Can I modify a template?

A. These templates are prebuilt projects you can use as a start for your own games.

Q. Can I save changes made during playtesting?

A. Changes made in Play mode won't be saved. You'll have to do them again when you go back to editing.

Workshop

Now that you have finished this hour, let's review what you've learned. Take a moment to answer the following questions.

Quiz

1. How do you organize your workspace?
2. Which two common starting point templates can be developed from scratch?
3. How do you move your avatar around during playtesting?
4. True or False: Publishing your project on Roblox makes it visible to everyone.
5. True or False: The Transform tool is an all-in-one building tool.

Answers

1. Closing the extra windows will give you more space to see what you're doing and keep the Explorer and Properties windows aligned below each other.
2. Baseplate and Flat Terrain are two commonly used templates on which a game developer can develop from an entire game world.
3. We use the WASD or the arrow keys to move around.
4. True. Publishing saves your work to a secure place and allows other players on Roblox to play your game. (To make it public to everyone, go to "Game Settings" after the initial publish.)
5. True. The transform tool is an all-in-one building tool. It moves, scales, and rotates in a precise way.

Exercises

Follow the exercise below to gain additional insight into the Roblox Studio.

1. Open a new Baseplate template.
2. From the Home tab, add a part block.
3. Find the new part added to the Explorer window under Workspace. Rename it as **`CenterPart`**.
4. Rename and save your baseplate; then publish it to Roblox.
5. Playtest your game.

This second exercise combines a number of things you've learned the last two hours. If you get stuck, don't forget to refer to the previous pages in this chapter! You're going to make a simple obstacle course (commonly referred to as an "obby" in Roblox).

1. Start with a couple of parts. Make sure that Anchored is enabled and place them in the sky. Feel free to color them any color that you want, or even add decals or textures!
2. Add another part at one end of the parts. This will be the start of your obstacle course game. Make sure that it is also Anchored.

3. Add your final part at the other end of the parts. This will be the end of your obstacle course game. Make sure that it too is anchored.
4. Playtest your game. Test out your game by flying over your starting point, clicking the blue arrow underneath Play in the Home tab, and choosing Play Here.
5. Bonus: Add a Spawn object from the Gameplay section of the Model tab at the top of Roblox Studio to avoid having to press Play Here and having all players start at the beginning. (It is anchored by default!)

TIP

Keep these tips in mind.

- Add at least five or six parts of differing sizes and shapes to create a jumping puzzle for players. The beginning jumps should be easier than later jumps.
- Playtest your game throughout the creation process to make sure you can make each jump and that all parts are anchored.

Index

Symbols

= (equal sign) operator, 196-197

2D axis, U and V directions, 52

3D representation in cameras, 368

A

accessibility
- of ModuleScripts, 347
- with sounds, 282

accessing
- models, 161
- packages, 186-187
- Team Create sessions, 182-183

Accessories category, hats in, 7

accounts (Roblox), creating, 17-18

actions, binding, 378

Activated event (Lua), 246

adding
- Animation Events, 298-299
- Atmosphere object to Lighting object, 124-125
- keyframes, 289
- objects to collision groups, 218
- places to games, 175-177
- Sky object to Lighting object, 124-125, 133
- teams, 338-339
- textures
  - to Beam object, 147-148
  - to particles, 143-144
- users to Team Create sessions, 180-182, 191

Add tool (Terrain Editor), 86

adjusting TextLabel properties in ScreenGui object, 238-240

Adornee property (SurfaceGui object), 242

advertising games
- purpose of, 414
- running time for ads, 421
- sponsor ads, 414-416
- user ads, 416-418

advertising outside of Roblox, 16

aesthetic of games, 14

aligning attachments, 67, 77

AllowedExternalLinkReferences compliance policy, 390

All Templates tab (templates), 22-23

Ambient property (Lighting object), 111, 119, 136-138

ambient sounds, creating, 277-278, 283

Analytics, 420-421

anchored parts

- constraints and, 77
- hanging unanchored parts from, 60-62
- network ownership, 341
- physics calculations, 374
- welds versus, 63-64

Anchored property, physics calculations, 374

anchoring objects, 35-36

AnchorPoint property (TextLabel object), 239

Angle property (Lighting object), 116

Animation Editor

- animations
  - copying ID, 292-293
  - editing uploaded, 302
  - exporting, 292
  - looping, 296-297
  - priority of, 297
  - replacing default, 301-303
  - saving, 291-292
  - sharing, 302
- events
  - adding, 298-299
  - cloning, 299
  - deleting, 299
  - enabling, 298
  - implementing in scripts, 299-302
  - moving, 299
- IK (Inverse Kinematics)
  - Body Part IK mode, 295
  - enabling, 294
  - Full Body IK mode, 295
  - modes, 294-295
  - pinning parts, 296
  - purpose of, 293-294
- keyframes
  - adding, 289
  - cloning, 290
  - creating, 288-289
  - deleting, 289
  - easing, 293
  - moving, 290
- models, requirements, 286-287
- opening, 287
- poses, creating, 287-290
- purpose of, 285-286
- toggling Move and Rotate tools, 289

AnimationPlayed() event, 431

animations. *See also* Animation Editor

- attack animations, creating, 290-291
- CFrame data type
  - components of, 270
  - creating, 259
  - moving parts relative to current position, 260-261
  - purpose of, 258
  - rotating parts, 261-264
- ClickDetectors, 259-260
- copying ID, 292-293
- editing uploaded, 302
- exporting, 292
- looping, 296-297
- of models, 268-269
- Position property, 257-258
  - moving relative to current position, 259-261
  - setting position, 259
- previewing, 289
- priority of, 297
- purpose of, 285
- replacing default, 301-303
- Rotation property, 257-258, 261-264
- saving, 291-292
- sharing, 302
- timeline units, 288
- tweens
  - between two points, 266-267
  - creating, 265
  - easing style and direction, 267-270

Annual Bloxy Awards, 3

APM Music, 9

appearance properties

- Lighting object, 110-112
- tools, 309-311

Appearance tab (Properties window), 44-48

- color, 45
- materials, 45-46
- reflectance, 47
- transparency, 47, 56

applying decals, 49-50

ArePaidRandomItemsRestricted compliance policy, 389

arguments (Lua), purpose of, 199

arranging workspace, 26-27
arrays (Lua), 203
aspect ratio constraints, 250
Asset Library, 5
Asset Manager
images, importing, 170-171
meshes, importing, 166-168
packages, accessing, 186-187
assets. *See also names of specific types of assets*
moderation, 157, 166
storage, 157
in Toolbox, 8-9
types of, 5
uploading, troubleshooting, 166
Asset Service, 10
assigning
players to teams
automatically, 340
manually, 340-341
roles in group games, 179
values to variables (Lua), 196-197
Atmosphere object. *See also* Skyboxes
adding to Lighting object, 124-125
properties
Color, 129
Decay, 131
Density, 125-126
Glare, 129-131, 138
Haze, 128-131
Offset, 126-127
setting to default, 124
purpose of, 123
attachments
aligning, 67, 77
beams, creating, 146-148
connecting, 60-62
with HingeConstraint, 66-70, 74-76
with rod constraints, 60-62
with SpringConstraint, 71-72
defined, 60
rotating, 69
types of, 11
attack animations, creating, 290-291
attracting players
advertising
purpose of, 414
running time for ads, 421
sponsor ads, 414-416
user ads, 416-418
with game updates, 413-414
with icons, thumbnails, and trailers, 409-413
notifications of updates, 419-420
Roblox Analytics, 420-421
audio
ambient sounds, creating, 277-278, 283
copyright issues, 275, 281
free music, 281
game accessibility, 282
in group games, 281
grouping sounds, 279-283
purpose of, 273
soundtracks
creating, 273-275
uploading, 275-276
triggering with code, 278-279, 282
audio assets
cost of, 171-172
listening to, 163
uploading, 171-172
audio tracks (APM Music), 9
automatically assigning players to teams, 340
avatars, 6-7. *See also* Humanoids
Avatar Shop, 6-7

B

baseplates, deleting, 82
Baseplate template, 22
Beam object
attachments, inserting, 147
properties
CurveSize, 148-149
Segments, 149-150
Width, 150-151
textures, adding, 147-148
beams
creating, 146-148
curving, 148-152
light rays, creating, 151-152
purpose of, 141, 145
resizing, 150-151
usage example, 145-146
bidding, 416, 421
BillboardGui object, purpose of, 236
billboard on highway, creating, 57

BindableEvent, creating events from, 207-208

binding actions, 378

biomes, 82

block comments (Lua), 198

Bloom effect (Lighting object), 114

blur, SunRays effect and, 119

BlurEffect effect (Lighting object), 115

Body Part IK mode, 295

bombs

exploding bombs

creating, 200-201, 212-213

exploding by touching example, 201-202

multiple bombs, creating, 198-199

translucent bombs, creating, 197

Boolean data type, 425

BorderColor3 property (TextLabel object), 237

BorderSizePixel property (TextLabel object), 237

breakpoints (Lua), creating, 209-210

bricks. *See* parts

Brightness property (Lighting object), 111-112, 116

Brush Settings (Terrain Editor), ignoring water, 92-93

building doors, 62-64

building materials. *See* materials

build optimization

meshes, 373

part count, 372

reusing meshes and textures, 373

transparency, 375

unions, 372-373

bulk importing meshes, 166-168

buttons, creating, 259-260

C

California Consumer Privacy Act (CCPA), 390

calling

functions (Lua), 200

ModuleScripts, 347-348

camera controls, 28-29

CameraOffset property, 363-365, 429

cameras

3D representation, 368

built-in behavior, removing, 368

default camera object, 359

default subject, 368

in LocalScripts, 360

mood via, 357-358

moving

changing time for, 366-367

with render step, 362, 365-366

sample exercise, 368-369

troubleshooting, 367

with tweens, 360-362

offsetting, 363-365

properties, 359

purpose of, 357

rendering, 368

camera shakes, creating, 363-365

CanBeDropped property (Tool object), 314

CanCollide property

disabling, 65, 232

physics calculations, 374

Capture the Flag gameplay, 24

capturing text for translation, 385-387

cash, converting Robux to, 403-405

CastShadow property, disabling, 375

Catalog. *See* Avatar Shop

CCPA (California Consumer Privacy Act), 390

celestial bodies

customizing, 135-136

disabling, 136

CFrame coordinates, determining from parts, 317

CFrame data type

components of, 270

creating, 259

moving parts relative to current position, 260-261

purpose of, 258

rotating parts, 261-264

CFrame property, 316, 359

ChangeState() function, 430

changing. *See also* editing

ambient light, 136-138

color

of decals, 51

of particles, 144-145, 153

of parts, 45

device support, 382

materials
of parts, 45-46
in terrain, 91-93
OffsetStudsU/OffsetStudsV of textures, 53
properties
of decals, 50-51, 56
of HingeConstraint, 70, 75-76
of parts, 44-48
of SpringConstraint, 73-74
of textures, 52-54
reflectance of parts, 47
StudsPerTileU/StudsPerTileV of textures, 54
transparency of parts, 47, 56
characters. *See* Humanoids
chatting with Roblox Studio Chat, 183
child objects, creating, 27-28
China policies, 389
ClickDetectors, 259-260
clicks (bidding), 416
Click Through Rate (CTR), 416
client-server model
improving performance, 376-377
ModuleScripts in, 349-350
network ownership, 341-344
operational overview, 331-332
RemoteEvents
creating, 335-337
order of execution, 342
purpose of, 333-334
sample exercise, 343
storage location, 334
waiting for reply, 342
RemoteFunctions
order of execution, 342
purpose of, 333-334
storage location, 334
waiting for reply, 342
replication in, 332-333
script types, 332
server-side validation, 337-338
client-side example for TeleportService, 320
client-side ModuleScripts, 349-350
ClockTime property (Lighting object), 112
cloning
Animation Events, 299
keyframes, 290
coding. *See* Lua
collaboration
group games
assigning roles, 179
configuring roles, 178-179
purpose of, 178
purpose of, 177
Roblox Studio Chat, 183
Team Create sessions
accessing, 182-183
adding users, 180-182, 191
disabling, 183
enabling, 179-180
CollisionFidelity property
physics calculations, 374
purpose of, 215-216
viewing and improving collision geometry, 216-217, 232
Collision Groups Editor
creating collision groups, 219
functions of, 218
opening, 217-218
purpose of, 217
in scripts, 219
switching collision groups, 220
workspace for, 217-218
CollisionGroupId property, 220
collision groups. *See also* Collision Groups Editor
creating, 218-219
editing interactions, 218
objects, adding to, 218
purpose of, 217
removing, 218
renaming, 218
switching, 220
collisions, 35
CanCollide property, disabling, 65
CollisionFidelity property
purpose of, 215-216
viewing and improving collision geometry, 216-217, 232
defined, 30, 215
detecting
with Debounce tool, 221-222, 232
with Touched event, 220-221, 232
trap door example, 222-223
disabling, 31, 215, 232
enabling, 35, 215
indicators for, 215
toggling, 11

color, changing
- ambient light, 136-138
- decals, 50-51
- particles, 144-145, 153
- parts, 45

Color3 property (decals), 50-51

ColorCorrection effect (Lighting object), 114-115, 119

color maps, 100-103

Color property
- Atmosphere object, 129
- Lighting object, 116
- ParticleEmitter object, 144-145, 153

ColorShift_Top property (ambient lighting), 136, 138

combat games
- error responses, 328
- persistent data stores
  - functions, 326-328
  - protecting against loss, 328
  - purpose of, 322
  - saving player data example, 324-326
  - supports and limits, 323
- teleportation
  - between places, 317-322, 329
  - game universes, 317-318
  - purpose of, 314
  - TeleportService, 318-321, 329
  - types of, 314
  - use cases, 315
  - within places, 315-317
- tools
  - creating, 306-307
  - displaying image in toolbar, 329
  - equipping in-game, 310-311
  - grip properties, 309-311
  - handles for, 307-309, 329
  - miscellaneous properties, 314
  - operational overview, 306
  - purpose of, 305-306
  - requirements, 329
  - sword creation example, 311-314

comments (Lua), 198

compliance policies, 389-390, 393

conditional operators, 427

conditional statements (Lua), 202-203. *See also* loops (Lua)

conditional structures (Lua), 426-427

Configuration page, 10

configuring roles in group games, 178-179

connecting
- attachments, 60-62
  - with HingeConstraint, 66-70, 74-76
  - with rod constraints, 60-62
  - with SpringConstraint, 71-72
- ModuleScripts to leaderboard, 348
- objects and parts, 60

connectivity, social, 2-4

console-friendly games, 381, 384

constraints
- anchored parts and, 77
- defined, 60
- for GUIs, 250-251
- HingeConstraint, 66
  - adding to doors, 66-70
  - aligning attachments, 77
  - creating motors, 74-76
- number needed, 79
- rod constraints, 60-62
- SpringConstraint
  - adding to doors, 71-72
  - changing properties, 73-74
- types of, 11
- welds, 63-64, 78-79

consumables. *See* Developer Products

content management, 4-7
- avatar customization, 6-7
- custom images in, 5-6
- moderation, 4
- organizational structure for, 4-5

content streaming, 374

ContextActionService object, 378-379

contrast, ColorCorrection effect and, 119

control settings for game loops, 351-352

controls in mobile-friendly games, 378-379

converting
- objects to packages, 184-186
- Robux to money, 403-405

copying
animation ID, 292-293
PlaceID for TeleportService, 319
terrain, 99
copyright issues, 16
for music files, 275, 281
Copy tool (Terrain Editor), 99
cost
audio assets, 171-172
Roblox, 13
uploaded audio files, 275
countdown creation example (GUIs), 251-254
Create page, 4-5
creator hub, Roblox as, 3-4
cross-platform support, 12-13
CTR (Click Through Rate), 416
CurveSize property (Beam object), 148-149
curving beams, 148-152
custom avatar rigs, 7
custom characters, walking surfaces example, 225-227
custom events (Lua), creating, 207-208
custom images, 5-6
custom Roblox data types (Lua), 426
customizing
avatars, 6-7
celestial bodies, 135-136

D

data deletion, requesting, 390
Data properties (Lighting object), 112
data store. *See* persistent data store
DataStoreService API, purpose of, 322
data types (Lua), 195, 226
custom Roblox, 426
enums, 426
primitives, 425
Debounce tool (Lua), 221-222, 232
debugging scripts (Lua), 209-210. *See also* troubleshooting
breakpoints, 209-210
log files, 210-212
string debugging, 209
decals
applying, 49-50
defined, 48
importing, 170-171
properties
changing, 50-51, 56
color, 51
textures versus, 52
uploading, 49
Decay property (Atmosphere object), 131
decomposition geometry, viewing, 216-217
default animations, replacing, 301-303
default camera object, 359
default camera subject, 368
default properties, setting, 124
Delete tool (Terrain Editor), 99
deleting
Animation Events, 299
baseplates, 82
camera's built-in behavior, 368
collision groups, 218
hidden objects, 375
keyframes, 289
landscapes, 84
models, 172
packages, 184, 191
personal information, 390-392
terrain, 99
with Erode tool, 89
with Subtract tool, 87
water, 93
deltaTime property, 366-367
Density property (Atmosphere object), 125-126
deploying translated text, 388
DepthOfField effect (Lighting object), 115-116
designing games, possibilities, 12
detecting collisions
with Debounce tool, 221-222, 232
with Touched event, 220-221, 232
trap door example, 222-223
developer console, opening, 371
Developer Exchange (DevEx), 403-405
Developer Forums, feature requests, 16
Developer Products
creating, 399-401
sample exercise, 407
when to use, 405
developers, connection opportunities for, 3-4
device emulation, 379-380

device simulation, 13

device support, changing, 382

dictionaries (Lua), 203

Died() event, 431

direction (tweens), easing, 267-270

disabling

CanCollide property, 65, 232

CastShadow property, 375

celestial bodies, 136

collisions, 31, 215, 232

physics simulation, 11

ScreenGui object, 237

snapping, 30

Team Create sessions, 183

DisplayDistanceType property, 429

displaying images on tools in toolbar, 329

DisplayName property, 429

docking windows (game editor), 27

doors

building, 62-64

CanCollide property, disabling, 65

creating, 233

HingeConstraint, adding, 66-70

SpringConstraint, adding, 71-72

troubleshooting swinging, 69

dot notation, WaitForChild() function versus, 348

Drag property (ParticleEmitter object), 145

dystopian sky, creating, 139

E

easing

keyframes, 293

tween style and direction, 267-270

easing styles for tweening, 248

EasingDirection property, 267

EasingStyle property, 267

editing. *See also* changing

collision group interactions, 218

games in playtesting, 38-40

landmarks, 104-105

landscapes, 105-106

models, 172

templates, 40

uploaded animations, 302

Edit tab (Terrain Editor), 85

Add tool, 86

Erode tool, 89

Flatten tool, 90-91

Grow tool, 87-88

Paint tool, 91-93

Sea Level tool, 93-94

Smooth tool, 89-90

Subtract tool, 87

effects (Lighting object)

Bloom, 114

BlurEffect, 115

ColorCorrection, 114-115, 119

DepthOfField, 115-116

inserting, 119

SunRays, 113-114, 119-121

troubleshooting, 113

elevating terrain, 87-88

else blocks (Lua), 202

EmissionDirection property (ParticleEmitter object), 143

emulating mobile devices, 379-380

enabling

Animation Events, 298

collisions, 35, 215

IK (Inverse Kinematics), 294

Output window, 194

perks in Game Passes, 398

ScreenGui object, 237

snapping, 34

Team Create sessions, 179-180

enclosing scope (Lua), 207

engine. *See* Roblox Studio

Enum object, 226

enums (Lua), 226, 426

EnvironmentDiffuseScale property (Lighting object), 111

EnvironmentSpecularScale property (Lighting object), 111, 119

equal sign (=) operator, 196-197

EquipTool() function, 430

Erode tool (Terrain Editor), 89

error responses, 328

errors. *See* troubleshooting

event markers, 298

events

Animation Events

adding, 298-299

cloning, 299

deleting, 299

enabling, 298

implementing in scripts, 299-302
moving, 299
creating, 207-208
exploding by touching example, 201-202
for Humanoids, 431
purpose of, 200-201, 211
RemoteEvents
creating, 335-337
order of execution, 342
purpose of, 333-334
sample exercise, 343
storage location, 334
waiting for reply, 342
time, checking for, 366-367
Touched, 220-221, 232
with Debounce tool, 221-222, 232
trap door example, 222-223

exercises
ambient sounds, creating, 283
animations, replacing default, 303
console-friendly games, creating, 384
countdown timer, adding functionality to, 254
Developer Products, creating, 407
doors, creating, 233
dystopian sky, 139
exploding bombs, creating, 212-213
fireplaces, creating, 153-154
forest scene, creating, 173-174
Game Passes, creating, 406-407
GamePass purchases, prompting, 254-255
games
adding thumbnails, 423
globalization, 394
planning updates, 422-423
highway with billboard, creating, 57
landmarks, modifying, 104-105
landscapes, importing/editing, 105-106
lasers, firing, 343-344
mobile-friendly games, creating, 384
models, spawning, 270-271
ModuleScripts, 356
moving cameras, 368-369
moving objects by clicking switches, 271
multiple places in games, planning, 191
objects, spawning via touch, 272
obstacle course, creating, 41-42
player data, saving, 330
projects, publishing, 41
RemoteEvents, 343
Roblox accounts, creating, 17-18
see-saws, creating, 78-79
sounds, triggering with code, 282
speed power up button, creating, 233
spotlights, creating, 120
sun rays, creating, 120-121
top scores, saving in data store, 330
waterfalls, creating, 155-156

experiences, types of, 14

exploding bombs
creating, 200-201, 212-213
exploding by touching example, 201-202

Explorer window (game editor), 27-28

Explosion object (exploding bombs example), 200-201

exporting animations, 292

Exposure properties (Lighting object), 112

external references policy, 390

F

Face property
decals, 50
Lighting object, 116

fans, creating, 74-76

feature requests, 16

FieldOfView property, 359

fighting games. *See* combat games

Fill tool (Terrain Editor), 99-100

filling areas of terrain, 99-100

Filtering Enabled, 10

FireAllClients() function, 333

FireClient() function, 333

fireplaces, creating, 153-154

FireServer() function, 333

firing lasers, 343-344
Fixed Plane setting (Flatten tool), 91
Flatten Mode setting (Flatten tool), 91
Flatten tool (Terrain Editor), 90-91
flattening terrain, 90-91
Flat Terrain template, 22
Focus property, 359
forest scene, creating, 173-174
for loop (Lua), 205-206
forming terrain, 86
Frame object
  in GUIs, 244-245
  Visible property, 246
frames, 362
free models
  accessing, 161
  creating, 157-158
  defined, 157
  deleting, 172
  designating base, 158
  editing, 172
  inserting, 161-163
  purpose of, 157
  scripts in, 161
  uploading, 158-160
  usage permissions, 159
  viewing details, 162
free music, 281
Full Body IK mode, 295
function data type, 425
functions
  calling, 200
  creating, 200
  exploding bomb example, 200-201
  for Humanoids, 430
  ipairs(), 206, 211
  naming, 200
  pairs(), 206, 211
  pcalls, 328
  for persistent data store, 326-328
  print(), 195, 199, 209
  purpose of, 199, 211
  RemoteFunctions
    order of execution, 342
    purpose of, 333-334
    storage location, 334
    waiting for reply, 342
  reusable for game loops, 352
  scope, 206-207
  with TeleportService, 318-319
  wait(), 204
  wrapping in pcalls, 321

G

game editor, 24-30
  Explorer window, 27-28
  menu bar, 24-25
  opening, 24
  parts
    creating, 28-29
    defined, 28
    naming, 29
  Properties window, 29-30
  Property window, docking/undocking, 27
  ribbon bar, 26
  workspace arrangement, 26-27
game loops
  components of, 351
  control settings, 351-352
  main engine in, 353-354
  purpose of, 351
  reusable functions in, 352
Game Passes
  creating, 395-397
  enabling perks, 398
  pricing for, 397
  purchases, prompting, 254-255
  sample exercise, 406-407
  selling, 397-398
  viewing ID, 397
  when to use, 405
game performance, improving, 371
  client-server model, 376-377
  loops, 377
  memory usage, 371-372
  meshes, 373
  object parents, 376
  part count, 372
  reducing physics, 374
  reusing meshes and textures, 373
  streaming content, 374
  tips for, 375
  unions, 372-373
Gameplay tab (templates), 24
games. *See also* projects
  accessibility with sounds, 282
  aesthetic of, 14
  collaboration
    group games, 178-179
    purpose of, 177

Roblox Studio Chat, 183
Team Create sessions, 179-183, 191
console-friendly, 381, 384
design possibilities, 12
device support, changing, 382
editing in playtesting, 38-40
error responses, 328
global compliance policies, 389-390, 393
localization
capturing text for translation, 385-387
deploying translated text, 388
hiring translators, 389
purpose of, 385
translating captured text, 387-388
marketing
advertising types, 414-418, 421
with icons, thumbnails, and trailers, 409-413
notifications of updates, 419-420
Roblox Analytics, 420-421
with updates, 413-414
mobile-friendly
controls, 378-379
device emulation, 379-380
percentage of players on mobile devices, 377
sample exercise, 384
UI scaling, 378
persistent data store
functions, 326-328
protecting against loss, 328
purpose of, 322
saving player data example, 324-326
supports and limits, 323
places
adding, 175-177
planning multiple, 191
playtesting, 38-40
privacy policies, 390-392
security, Filtering Enabled, 10
teleportation
between places, 317-322, 329
game universes, 317-318
purpose of, 314
TeleportService, 318-321, 329
types of, 314
use cases, 315
within places, 315-317
tools
creating, 306-307
displaying image in toolbar, 329
equipping in-game, 310-311
grip properties, 309-311
handles for, 307-309, 329
miscellaneous properties, 314
operational overview, 306
purpose of, 305-306
requirements, 329
sword creation example, 311-314
types of, 14
updating, 10
planning, 422-423
virtual reality, 381-382

game universes, 317-318
generating landscapes, 82-85
GetArrivingTeleportGui() function, 319
GetAsync() function, 326
GetDataStore() function, 323
GetGlobalDataStore() function, 323
GetKeyframeMarkerReached() function, 298
GetNetworkOwner() function, 341
GetPolicyInfoForPlayerAsync function, 389
GetService() function, 318, 340
GetState() function, 430
GetTeams() function, 340
ghosting in Atmosphere object, 127
Glare property (Atmosphere object), 129-131, 138
globalization
compliance policies, 389-390, 393
localization of games
capturing text for translation, 385-387
deploying translated text, 388
hiring translators, 389
purpose of, 385
translating captured text, 387-388
privacy policies, 390-392
global scope (Lua), 207
GlobalShadows property (Lighting object), 111
Graphical User Interfaces. *See* GUIs (Graphical User Interfaces)
graphics, levels of, 12

grid layouts, 248

GripForward property (Tool object), 309

GripPos property (Tool object), 309

grip properties for tools, 309-311

GripRight property (Tool object), 309

GripUp property (Tool object), 309

group games
- assigning roles, 179
- configuring roles, 178-179
- purpose of, 178
- sounds in, 281

grouping
- objects, 157, 185
- sounds, 279-283

Groups, 2-3

Grow tool (Terrain Editor), 87-88

GUIs (Graphical User Interfaces)
- BillboardGui object, 236
- constraints, 250-251
- countdown creation example, 251-254
- elements of, 243-245
- examples of, 235, 236
- interactive GUIs
  - coding, 244-247
  - creating, 242-243
  - tweening, 247-248
- layouts, 248-250
- memory usage, 253
- purpose of, 235
- resizing, 253
- ScreenGui object, 236-240
- SurfaceGui object, 236
  - creating, 240-243
  - as interactive GUI, 242-243
  - parenting to parts, 240-241
  - resizing TextLabel, 241-242
- types of, 236

H

handles for tools, 329
- creating, 307-309

hanging unanchored parts from anchored parts, 60-62

hardware requirements for Roblox Studio, 20

hats, uploading, 7

Haze property (Atmosphere object), 128-131

HealthChanged() event, 431

HealthDisplayDistance property, 429

HealthDisplayType property, 429

height maps, 100-101
- importing, 101
- purpose of, 103

hidden objects, deleting, 375

hiding windows, 44-45

highway with billboard, creating, 57

HingeConstraint, 66
- attachments, aligning, 77
- doors, adding to, 66-70
- motors, creating, 74-76

hiring translators, 389

Home tab (game editor), 25

hosting, 10

HTTP Service, 10

HumanoidRootPart property (characters), 316

Humanoids
- cameras for, 368
- events, 431
- functions, 430
- in object hierarchy, 224
- properties, 429-430
  - CameraOffset, 363-365
- purpose of, 224
- rig types, 224, 232
- walking surfaces example, 225-227

I

icons, attracting players with, 409-413

if blocks (Lua), 202

ignoring water in Brush Settings, 92-93

IK (Inverse Kinematics)
- Body Part IK mode, 295
- enabling, 294
- Full Body IK mode, 295
- modes, 294-295
- pinning parts, 296
- purpose of, 293-294

ImageButton object
- in GUIs, 244-245
- memory usage, 253
- properties for tweening, 247

ImageLabel object
- in GUIs, 244
- memory usage, 253

images
- custom, 5-6
- importing
  - with Asset Manager, 170-171
  - with Texture Object, 168-170

Skyboxes
creating, 132-135
uploading, 135
text in, translating, 388
on tools, displaying in toolbar, 329
uploading for Game Passes, 396

importing
audio assets, 171-172
decals, 170-171
height maps, 101
landscapes, 105-106
meshes
with Asset Manager, 166-168
with MeshPart, 164-166
textures
with Asset Manager, 170-171
with Texture Object, 168-170
troubleshooting, 101

impressions (bidding), 416

improving
collision geometry, 216-217, 232
game performance, 371
client-server model, 376-377
loops, 377
memory usage, 371-372
meshes, 373
object parents, 376
part count, 372
reducing physics, 374
reusing meshes and textures, 373
streaming content, 374
tips for, 375
unions, 372-373
scripts (Lua), 211

IncrementAsync() function, 327

inserting
attachments in Beam object, 147
free models, 161-163
lighting effects, 119
ParticleEmitter object, 142
Script object in ServerScriptService, 194
Sound objects, 277
TextLabel object into ScreenGui object, 237-238

installing Roblox Studio, 19-20
troubleshooting, 20, 40

Instance.new() function, 376

interactive GUIs
coding, 244-247
creating, 242-243
tweening, 247-248

Inverse Kinematics. *See* IK (Inverse Kinematics)

InvokeClient() function, 333-334

InvokeServer() function, 333

ipairs() function, 206, 211

IsPaidItemTradingAllowed compliance policy, 390

IsSubjectToChinaPolicies compliance policy, 389

J–K

Jump property, 429

JumpPower property, 429

keyframes
adding, 289
cloning, 290
creating, 288-289
deleting, 289
easing, 293
moving, 290
purpose of, 286

L

landmarks, modifying, 104-105

landscapes
color maps, 100-103
deleting, 84
generating, 82-85
height maps, 100-101
importing, 101
purpose of, 103
importing/editing, 105-106
ocean, 85
terrain
changing materials, 91-93
copying/pasting/deleting, 99
creating water in, 93-94
elevating, 87-88
filling areas, 99-100
flattening, 90-91
forming, 86
moving, 95-98
removing with Erode tool, 89
removing with Subtract tool, 87
scaling, 98

selecting, 94-95
smoothing, 89-90
viewing, 86
lasers, firing, 343-344
layouts for GUIs, 248-250
leaderboard
connecting ModuleScripts to, 348
score keeping example, 227-231
Lifetime property (ParticleEmitter object), 145
light rays, creating, 151-152
LightEmission property (ParticleEmitter object), 145
LightInfluence property (SurfaceGui object), 243
lighting
improving performance, 375
shaders, types of, 12
Lighting object
effects
Bloom, 114
BlurEffect, 115
ColorCorrection, 114-115, 119
DepthOfField, 115-116
inserting, 119
SunRays, 113-114, 119-121
troubleshooting, 113
properties, 108-110
Ambient, 119, 136-138
Appearance, 110-112
Data, 112
EnvironmentSpecularScale, 119
Exposure, 112
list of, 116
purpose of, 107-108
Sky and Atmosphere objects, adding, 124-125, 133
sun in, 129
types of lights, 116
PointLight, 117
SpotLight, 117, 120
SurfaceLight, 118
listening to audio assets, 163
list layouts, 248-249
LoadAnimation() function, 430
localization
purpose of, 385
text translation
capturing text, 385-387
deploying translated text, 388
hiring translators, 389
translating captured text, 387-388
local scope (Lua), 206
LocalScripts (client-server model), 332, 360
log files (Lua), debugging with, 210-212
logical operators, 427
looping animations, 296-297
loops (Lua)
for loop, 205-206
improving performance, 377
purpose of, 203
repeat-until loop, 205
while loop, 204
loot boxes, 389
losing data, protection against, 328
Lua, 193
arrays, 203
comments, 198
conditional statements, 202-203
conditional structures, 426-427
data types, 195, 226, 425-426
dictionaries, 203
ease of use, 7-10
events
creating, 207-208
exploding by touching example, 201-202
for Humanoids, 431
purpose of, 200-201, 211
Touched, 220-223, 232
functions
calling, 200
creating, 200
exploding bomb example, 200-201
for Humanoids, 430
ipairs(), 206, 211
naming, 200
pairs(), 206, 211
print(), 195, 199, 209
purpose of, 199, 211
scope, 206-207
wait(), 204
loops
for loop, 205-206
improving performance, 377

purpose of, 203
repeat-until loop, 205
while loop, 204
properties, 8
resources for information, 428
scripts
Animation Events in, 299-302
with ClickDetectors, 259-260
client-server model, 376-377
Collision Groups Editor usage in, 219
creating, 194-195
debugging, 209-212
improving, 211
for interactive GUIs, 244-247
loops, 377
multiple bombs example, 198-199
parent objects, 376
purpose of, 193, 211
renaming, 195
reusability, 199
running, 195
score keeping example, 227-231
storing, 227
sword creation example, 311-313
translucent bombs example, 197
triggering sounds with, 278-279, 282
types of, 332
tables, 203
variables
creating, 196-197
naming, 196, 230
purpose of, 195
scope, 206-207
web services, 10
workspace, initial setup, 194

M

managing content. *See* content management
ManualActivationOnly property (Tool object), 314
manually assigning players to teams, 340-341
Map Settings (Terrain Editor), 83, 93
marketing games
advertising
purpose of, 414
running time for ads, 421
sponsor ads, 414-416
user ads, 416-418
with icons, thumbnails, and trailers, 409-413
notifications of updates, 419-420
Roblox Analytics, 420-421
with updates, 413-414
marketplace fees, 401
MarketPlaceService object, 397
materials
color maps for, 102-103
of parts, changing, 45-46
in terrain, changing, 91-93
Material Settings (Terrain Editor), 83
Fill tool, 99-100
Paint tool, 92
math.rad() function, 261
MaxDistance property, 278
mechanical construction. *See* physics engine
memory usage
for GUIs, 253
improving, 371-372
Merge Empty setting (Terrain Editor)
Fill tool, 99
Move tool, 96-97
meshes
creating, 373
importing
with Asset Manager, 166-168
with MeshPart, 164-166
reusing, 373
size limitations, 166
triangle count, 373
MeshPartm importing meshes with, 164-166
mobile-friendly games
controls, 378-379
device emulation, 379-380
percentage of players on mobile devices, 377
sample exercise, 384
UI scaling, 378
models
accessing, 161
Animation Editor requirements, 286-287
creating, 157-158
defined, 157

deleting, 172
designating base, 158
editing, 172
inserting, 161-163
moving, 268-269
packages. *See* packages
purpose of, 157
rigging, 286-290
scripts in, 161
spawning, 270-271
symbol of, 184
uploading, 158-160
usage permissions, 159
viewing details, 162

Model tab (game editor), 25

moderation, 4
for assets, 157, 166
in Roblox, 101

modifying. *See* editing

ModuleScripts
calling, 347-348
in client-server model, 349-350
code accessibility in, 347
connecting to leaderboard, 348
creating, 345
game loops
components of, 351
control settings, 351-352
main engine in, 353-354
purpose of, 351
reusable functions in, 352
module table in, 346
purpose of, 345
renaming, 346
requiring another ModuleScript, 355
sample exercises, 356
storage area, 345-355

module table in ModuleScripts, 346

monetization. *See also* Robux
Developer Exchange (DevEx), 403-405
Developer Products
creating, 399-401
sample exercise, 407
when to use, 405
Game Passes
creating, 395-397
enabling perks, 398
pricing for, 397
sample exercise, 406-407
selling, 397-398
viewing ID, 397
when to use, 405
marketplace fees, 401
"Pay to Win," 403
Pending Sales, 398-399
Premium Payouts, 401-402
Roblox Premium, 401-403
tips for, 405

money, converting Robux to, 403-405

mood via camera position, 357-358

moon, 136

motors
creating, 74-76
troubleshooting, 77

MouseEnter event, 246

MouseLeave event, 246

Move snapping, 34

MoveToFinished() event, 431

MoveTo() function, 430

Move tool, 31
in Terrain Editor, 95-98
toggling with Rotate tool, 289

movie theaters, creating, 55

moving
Animation Events, 299
cameras
changing time for, 366-367
with render step, 362-366
sample exercise, 368-369
troubleshooting, 367
with tweens, 360-362
keyframes, 290
models, 268-269
objects. *See also* animations
by clicking switches, 271
by collisions, 30, 35
with Move tool, 31
Position property, 259-261
Rotation property, 261-264
by snapping, 30, 34-35
with tweens, 265-270
terrain, 95-98

multiline comments (Lua), 198

multiplayer games
client-server model
network ownership, 341-344
operational overview, 331-332
RemoteEvents, 333-337, 342-343

RemoteFunctions, 333-334, 342
replication in, 332-333
script types, 332
server-side validation, 337-338
teams
adding, 338-339
automatically assigning players, 340
manually assigning players, 340-341
as nil, 339
multiple bombs, creating, 198-199
multiple places in games, planning, 191
music
APM Music audio tracks, 9
copyright issues, 275, 281
free, 281
soundtracks
creating, 273-275
uploading, 275-276

N

NameDisplayDistance property, 429
NameOcclusion property, 429
naming
functions (Lua), 200
parts, 29
variables (Lua), 196, 230
networking, ease of use, 10
network ownership, 341-344
Network Simulator, 313
nil, teams as, 339
Nil data type, 425
notifications, sending, 419-420
number data type, 425

O

objects. *See also* parts
adding to collision groups, 218
anchoring, 35-36
connecting to parts, 60
converting to packages, 184-186
creating, 27-28, 376
grouping, 157, 185
hidden, deleting, 375
hierarchy, Humanoids in, 224
moving. *See also* animations
by clicking switches, 271
by collisions, 30, 35
with Move tool, 31
Position property, 259-261
Rotation property, 261-264
by snapping, 30, 34-35
with tweens, 265-270
network ownership, 341-344
packages. *See* packages
parent/child structure, 27
parents, setting, 376
properties, 8, 30
rotating, 32-33
scaling, 32
spawning via touch, 272
transforming, 33-34
obstacle course, creating, 41-42
ocean landscapes, 85
official avatar rigs, 7
Offset property
Atmosphere object, 126-127
GUIs, 253
TextLabel object, 239
offsets for rotation, 264
OffsetStudsU property (textures), 53
OffsetStudsV property (textures), 53
offsetting cameras, 363-365
one-time purchases. *See* Game Passes
opening
Animation Editor, 287
Collision Groups Editor, 217-218
developer console, 371
game editor, 24
projects, 38
Roblox Studio, 21-22
operators (Lua), 427
optimization. *See* game performance
organizational structure for user content, 4-5
Output window, enabling, 194

P

packages
accessing, 186-187
converting objects to, 184-186
purpose of, 184
removing, 184, 191
symbol of, 184
updating, 187-190

page layouts, 249-250

Paint tool (Terrain Editor), 91-93

pairs() function, 206, 211

parent objects, setting, 376

parent-child relationship in scripts (Lua), 199

parenting SurfaceGui object to parts, 240-241

ParticleEmitter object

inserting, 142

properties

Color, 144-145, 153

EmissionDirection, 143

list of, 145

Rate, 142-143, 152

Texture, 143-144

particles

area spread, 152

ParticleEmitter object

Color property, 144-145, 153

EmissionDirection property, 143

inserting, 142

properties, list of, 145

Rate property, 142-143, 152

Texture property, 143-144

purpose of, 141

usage example, 141-142

parts. *See also* objects; physics engine

anchored

constraints and, 77

hanging unanchored parts from, 60-62

network ownership, 341

physics calculations, 374

welds versus, 63-64

build optimization, transparency, 375

buttons, creating, 259-260

CanCollide property, disabling, 65

CFrame coordinates, 317

connecting

with beams, 146-148

to objects, 60

creating, 28-29, 43

decals

applying, 49-50

changing properties, 50-51, 56

defined, 48

textures versus, 52

uploading, 49

defined, 28

doors

adding HingeConstraint, 66-70

adding SpringConstraint, 71-72

building, 62-64

disabling CanCollide property, 65

troubleshooting swinging, 69

exploding by touching example, 201-202

fans, creating, 74-76

moving. *See also* animations

Position property, 259-261

Rotation property, 261-264

with tweens, 265-270

naming, 29

optimizing builds, 372-373

packages. *See* packages

parenting SurfaceGui object to, 240-241

pinning in Full Body ID mode, 296

primary parts, moving models, 268-269

properties, 8, 30

changing, 44-48

color, 45

materials, 45-46

reflectance, 47

transparency, 47, 56

textures

avoiding seams, 53

changing properties, 52-54

decals versus, 52

defined, 49

resizing, 54

uploading, 52, 56

unanchored, in moving models, 269

wedges, purpose of, 360

welding, 63-64, 78-79

Paste tool (Terrain Editor), 99

pasting terrain, 99

"Pay to Win," 403

pcalls

purpose of, 328

wrapping functions in, 321

Pending Sales, 398-399

performance of collisions, 216, 232. *See also* game performance

perks, enabling in Game Passes, 398

permissions for model usage, 159

persistent data stores

functions, 326-328

protecting against loss, 328

purpose of, 322

saving player data example, 324-326

supports and limits, 323

personal player information

deleting, 391-392

requesting data deletion, 390

physics engine. *See also* collisions

attachments. *See* attachments

CanCollide property, disabling, 65

constraints

aligning attachments, 77

anchored parts and, 77

defined, 60

HingeConstraint, 66-70, 74-76

number needed, 79

rod constraints, 60-62

SpringConstraint, 71-74

welds, 63-64, 78-79

motors

creating, 74-76

troubleshooting, 77

purpose of, 59

reducing physics, 374

physic simulation, 11

Physics Service API, switching collision groups, 220

pinning parts in Full Body IK mode, 296

PlaceID, copying for TeleportService, 319

places

adding to games, 175-177

in games, planning multiple, 191

teleportation between, 317-322, 329

teleportation within, 315-317

Plane Lock setting (Grow tool), 88

planning

game updates, 422-423

multiple places in games, 191

PlatformStand property, 429

player GUIs, creating, 236-240

PlayerMembershipChanged event, 403

players

assigning to teams, 340-341

attracting

advertising types, 414-418, 421

with game updates, 413-414

with icons, thumbnails, and trailers, 409-413

notifications of updates, 419-420

Roblox Analytics, 420-421

personal information

deleting, 391-392

requesting data deletion, 390

saving data, 322-326, 330

simulating, 313

teleportation

between places, 317-322, 329

game universes, 317-318

purpose of, 314

TeleportService, 318-321, 329

types of, 314

use cases, 315

within places, 315-317

playtesting games, 38-40

plugins, 9

PointLight object, 117

popping in Atmosphere object, 127

poses (Animation Editor)

creating, 287-290

keyframes, creating, 288-289

Position property, 257-258

moving relative to current position, 259-261

setting position, 259

TextLabel object, 238

PreciseConvexDecomposition property, 216, 232

Premium Payouts, 401-402

previewing animations, 289

pricing for Game Passes, 397

PrimaryPart in models, 158

primary parts, moving models, 268-269

primitive data types (Lua), 425

print() function, 195, 199, 209

priority of animations, 297

privacy policies, 390-392

process receipt callbacks, 400

processReceipt() function, 400

projects. *See also* games

publishing, 37, 41

reopening, 38

saving, 37

templates. *See* templates

PromptGamePassPurchaseFinished event, 398

PromptGamePassPurchase() function, 397

prompting GamePass purchases, 254-255

PromptPremiumPurchase() function, 402

PromptProductPurchase() function, 400

properties, 8, 30. *See also* names of specific objects, properties
- of cameras, 359
- of decals, changing, 50-51, 56
- Frame object, Visible, 246
- of HingeConstraint, changing, 70, 75-76
- for Humanoids, 429-430
- ImageButton object, for tweening, 247
- Lighting object, 108-110
  - Ambient, 119
  - Appearance, 110-112
  - Data, 112
  - EnvironmentSpecularScale, 119
  - Exposure, 112
  - list of, 116
- of parts
  - changing, 44-48
  - color, 45
  - materials, 45-46
  - reflectance, 47
  - transparency, 47, 56
- setting to default, 124
- of SpringConstraint, changing, 73-74
- TextLabel object, adjusting in ScreenGui object, 238-240
- of textures, changing, 52-54
- Tool object
  - grip properties , 309-311
  - miscellaneous properties, 314
- variables
  - creating, 196-197
  - naming, 196
  - purpose of, 195

Properties window (game editor), 29-30. *See also* properties

Property window (game editor), docking/undocking, 27

protection against data loss, 328

publishing projects, 37, 41

Q–R

radians, 261

Range property (Lighting object), 116

Rate property (ParticleEmitter object), 142-143, 152

ray effect on lighting, 151-152

reducing physics, 374

reflectance of parts, changing, 47

Region tab (Terrain Editor), 94
- Copy tool, 99
- Delete tool, 99
- Fill tool, 99-100
- Move tool, 95-98
- Paste tool, 99
- Resize tool, 98
- Select tool, 94-95

relational operators, 427

RemoteEvents
- creating, 335-337
- order of execution, 342
- purpose of, 333-334
- sample exercise, 343
- storage location, 334
- waiting for reply, 342

RemoteFunctions
- order of execution, 342
- purpose of, 333-334
- storage location, 334
- waiting for reply, 342

RemoveAsync() function, 327-329

removing. *See* deleting

renaming
- collision groups, 218
- ModuleScripts, 346
- scripts (Lua), 195

render step
- cameras, moving, 362
- changing time for, 366-367
- connecting indefinitely, 365-366
- troubleshooting, 367

RenderFidelity property, 373

rendering, 12
- cameras, 368
- settings, adjusting, 113

RenderStepped event, connecting cameras to, 365-366

reopening projects, 38

repeat purchases. *See* Developer Products

repeat-until loop (Lua), 205

replacing default animations, 301-303

ReplicatedStorage, storing RemoteFunctions and RemoteEvents in, 334

replication in client-server model, 332-333

requesting data deletion, 390

require() function, 347

RequiresHandle property (Tool object), 314

reserved servers with TeleportService, 321

ReserveServer() function, 319-321

Resize tool (Terrain Editor), 98

resizing
- beams, 150-151
- GUIs, 253
- TextLabel object, 241-242
- textures, 54

reusability of scripts (Lua), 199

reusable functions for game loops, 352

reusing meshes and textures, 373

rigging models, 286-287
- creating poses, 287-290

rigs, 7

RigType property, 429

Roblox
- accounts, creating, 17-18
- advertising outside of, 16
- aesthetic of, 14
- content management, 4-7
- copyright issues, 16
- cost of, 13
- as creator hub, 3-4
- cross-platform support, 12-13
- ease of use, 7-10
- engine. *See* Roblox Studio
- experiences, types of, 14
- feature requests, 16
- marketplace fees, 401
- models, uploading, 158-160
- moderation, 101
- overview, 1-2
- social connectivity in, 2-4
- as social website, 2-3
- Terms of Service, 4, 16

Roblox Analytics, 420-421

Roblox Blog, 3

Roblox Class API, 202

Roblox Developer Forum, 3

Roblox Developers Conference, 3

Roblox Lua. *See* Lua

Roblox Premium, 401-403

Roblox Studio
- capabilities of, 7
- collaboration
  - group games, 178-179
  - purpose of, 177
  - Roblox Studio Chat, 183
  - Team Create sessions, 179-183, 191
- device simulation, 13
- game editor, 24-30
  - docking/undocking windows, 27
  - Explorer window, 27-28
  - menu bar, 24-25
  - opening, 24
  - parts, 28-29
  - Properties window, 29-30
  - ribbon bar, 26
  - workspace arrangement, 26-27
- games, playtesting, 38-40
- installing
  - steps in, 19-20
  - troubleshooting, 20, 40
- models, symbol of, 184
- networking, ease of use, 10
- opening, 21-22
- packages
  - accessing in Asset Manager, 186-187
  - accessing in Toolbox, 186
  - converting objects to, 184-186
  - purpose of, 184
  - removing, 184, 191
  - symbol of, 184
  - updating, 187-190
- physics simulation, 11
- plugins, 9
- projects
  - publishing, 37, 41
  - reopening, 38
  - saving, 37
- rendering, 12, 113
- system requirements, 20
- templates, 22
  - All Templates tab, 22-23
  - editing, 40
  - Gameplay tab, 24
  - Themes tab, 23

Roblox Studio Chat, 183

Robux, 1, 13. *See also* monetization
- for audio assets, 171-172
- bidding, 416, 421
- converting to money, 403-405
- for uploaded audio files, 275

rod constraints, 60-62

roles in group games, 178-179
RollOffMode property, 278
Rotate tool, 32-33
  toggling with Move tool, 289
rotating
  attachments, 69
  objects, 32-33
Rotation property, 145, 257-258, 261-264
Rotation snapping, 34
RotSpeed property (ParticleEmitter object), 145
running scripts (Lua), 195

S

saving
  animations, 291-292
  player data, 322-326, 330
  projects, 37
  top scores in data store, 330
Scale property
  GUIs, 253
  TextLabel object, 238
Scale tool, 32
scaling
  objects, 32
  terrain, 98
scaling UI in mobile-friendly games, 378
scope (Lua), 206-207
score keeping example, 227-231
ScreenGui object, creating, 236-240
Script object, inserting in ServerScriptService, 194
Scripts (client-server model), 332
scripts (Lua)
  Animation Events, implementing, 299-302
  with ClickDetectors, 259-260
  Collision Groups Editor usage in, 219
  creating, 194-195
  debugging, 209-210
    breakpoints, 209-210
    log files, 210-212
    string debugging, 209
  in free models, 161
  improving, 211
    client-server model, 376-377
    loops, 377
    parent objects, 376
  for interactive GUIs, 244-247
  LocalScripts for cameras, 360
  ModuleScripts
    calling, 347-348
    in client-server model, 349-350
    code accessibility in, 347
    connecting to leaderboard, 348
    creating, 345
    game loop example, 351-354
    module table in, 346
    purpose of, 345
    renaming, 346
    requiring another ModuleScript, 355
    sample exercises, 356
    storage area, 345-355
  multiple bombs example, 198-199
  purpose of, 193, 211
  renaming, 195
  reusability, 199
  running, 195
  score keeping example, 227-231
  storing, 227
  sword creation example, 311-313
  translucent bombs example, 197
  triggering sounds with, 278-279, 282
  types of, 332
Sea Level tool (Terrain Editor), 93-94
seams in textures, avoiding, 53
seasonal game updates, 414
Seated() event, 431
security, Filtering Enabled, 10
see-saws, creating, 78-79
Segments property (Beam object), 149-150
Select tool (Terrain Editor), 94-95
selecting terrain, 94-95
selling Game Passes, 397-398
sending notifications, 419-420
server hosting, 10
ServerScriptService, 227
  inserting Script object, 194
server-side example for TeleportService, 320-321
server-side ModuleScripts, 349-350
server-side validation, 337-338
SetAsync() function, 327-328
SetNetworkOwner() function, 341
shaders, types of, 12

ShadowMap, changing to Voxel, 375
shadows, disabling, 375
Shadows property (Lighting object), 116
sharing animations, 302
short comments (Lua), 198
simulating
  mobile devices, 379-380
  players, 313
single line comments (Lua), 198
Sit property, 429
size constraints, 251
Size property
  ParticleEmitter object, 145
  TextLabel object, 238
Sky object
  adding to Lighting object, 124-125, 133
  properties, 134-135
Skyboxes. *See also* Atmosphere object
  ambient light, changing, 136-138
  celestial bodies, customizing, 135-136
  creating, 132-135
  Density property and, 125
  dystopian sky, creating, 139
  purpose of, 132
Smooth tool (Terrain Editor), 89-90
smoothing terrain, 89-90
snapping, 34-35
  defined, 30
  turning off, 30
  turning on, 34
  types of, 34
social connectivity in Roblox, 2-4
social website, Roblox as, 2-3
Sound objects, inserting, 277
SoundGroup objects, creating, 280-281
SoundId property, 278
sounds. *See also* audio assets
  ambient sounds, creating, 277-278, 283
  copyright issues, 275, 281
  free music, 281
  game accessibility, 282
  in group games, 281
  grouping, 279-283
  purpose of, 273
  soundtracks, 273-276
  triggering with code, 278-279, 282
soundscapes. *See* ambient sounds
soundtracks. *See also* audio tracks; music
  creating, 273-275
  uploading, 275-276
spawning
  models, 270-271
  objects via touch, 272
speed power up button, creating, 233
sponsor ads, 414-416, 421
SpotLight object, 117, 120
SpreadAngle property (ParticleEmitter object), 145
SpringConstraint
  doors, adding to, 71-72
  properties, changing, 73-74
stars, customizing, 136
StarterPlayerScripts, camera scripts in, 360
Start Place (in games), 317
StateChange() event, 431
stopping playtesting, 39
storage
  of assets, 157
  for ModuleScripts, 345, 355
  for RemoteFunctions and RemoteEvents, 334
  of scripts (Lua), 227
streaming content, 374
StreamingEnabled property, 374
Strength setting (Grow tool), 88
string data type, 425
string debugging (Lua), 209
Studio. *See* Roblox Studio
StudsPerTileU property (textures), 54
StudsPerTileV property (textures), 54
style (tweens), easing, 267-270
Subtract tool (Terrain Editor), 87
sun
  customizing, 135
  disabling, 136
  Glare property (Atmosphere object) and, 129, 138
SunRays effect (Lighting object), 113-114, 119-121
SurfaceGui object
  creating, 240-243
  as interactive GUI, 242-243
  parenting to parts, 240-241
  purpose of, 236
  resizing TextLabel, 241-242
SurfaceLight object, 118
swinging doors, troubleshooting, 69

switches, moving objects by, 271

switching collision groups, 220

swords, creating, 311-314

system requirements for Roblox Studio, 20

T

table data type, 425

table layouts, 249-250

tables (Lua), 203

TakeDamage() function, 430

TargetPoint property, 430

Team Create sessions

- accessing, 182-183
- adding users, 180-182, 191
- disabling, 183
- enabling, 179-180

teams

- adding, 338-339
- assigning players, 340-341
- as nil, 339

Technology property, 375

teeter-totters, creating, 78-79

teleportation

- between places, 317-322, 329
- game universes, 317-318
- purpose of, 314
- TeleportService
  - client-side example, 320
  - copying PlaceID, 319
  - functions, 318-319
  - purpose of, 318, 329
  - server-side example, 320-321
- types of, 314
- use cases, 315
- within places, 315-317

Teleport() function, 318

TeleportPartyAsync() function, 319

TeleportService

- client-side example, 320
- copying PlaceID, 319
- functions, 318-319
- purpose of, 318, 329
- server-side example, 320-321

TeleportToPrivateServer() function, 321

templates, 22

- All Templates tab, 22-23
- baseplates, deleting, 82
- editing, 40
- Gameplay tab, 24
- Themes tab, 23

Terms of Service, 4, 16

terrain

- copying/pasting/deleting, 99
- creating water in, 93-94
- elevating, 87-88
- filling areas, 99-100
- flattening, 90-91
- forming, 86
- materials, changing, 91-93
- moving, 95-98
- removing
  - with Erode tool, 89
  - with Subtract tool, 87
- scaling, 98
- selecting, 94-95
- smoothing, 89-90

Terrain Editor, 8-9

- color maps, 100-103
- Edit tab, 85
  - Add tool, 86
  - Erode tool, 89
  - Flatten tool, 90-91
  - Grow tool, 87-88
  - Paint tool, 91-93
  - Sea Level tool, 93-94
  - Smooth tool, 89-90
  - Subtract tool, 87
- height maps, 100-101
  - importing, 101
  - purpose of, 103
- landscapes
  - changing materials, 91-93
  - copying/pasting/deleting terrain, 99
  - creating water in, 93-94
  - deleting, 84
  - elevating terrain, 87-88
  - filling areas, 99-100
  - flattening terrain, 90-91
  - forming terrain, 86
  - generating, 82-85
  - moving terrain, 95-98
  - removing terrain with Erode tool, 89
  - removing terrain with Subtract tool, 87
  - scaling terrain, 98
  - selecting terrain, 94-95
  - smoothing terrain, 89-90
- purpose of, 81
- Region tab, 94
  - Copy tool, 99
  - Delete tool, 99
  - Fill tool, 99-100

Move tool, 95-98
Paste tool, 99
Resize tool, 98
Select tool, 94-95
settings, 83
View Selector, 86
Test tab (game editor), 25
text in images, translating, 388
text size constraints, 251
text translation tools
capturing text, 385-387
deploying translated text, 388
hiring translators, 389
translating captured text, 387-388
TextButton object in GUIs, 244
TextLabel object
in GUIs, 243
inserting into ScreenGui object, 237-238
properties, adjusting in ScreenGui object, 238-240
resizing, 241-242
TextTransparency property (TextLabel object), 237
Texture Object, importing images with, 168-170
Texture property
Beam object, 147-148
decals, 50
ParticleEmitter object, 143-144
TextureId property (Tool object), 314, 329
textures
adding to Beam object, 147-148
decals versus, 52
defined, 49
importing
with Asset Manager, 170-171
with Texture Object, 168-170
OffsetStudsU/OffsetStudsV, changing, 53
for particles, adding, 143-144
properties, changing, 52-54
resizing, 54
reusing, 373
seams, avoiding, 53
StudsPerTileU/StudsPerTileV, changing, 54
uploading, 52, 56
TextWrapped property (TextLabel object), 237
themes, 23
Themes tab (templates), 23
thread data type, 425
thumbnails
attracting players with, 409-413
cost of, 421
creating, 423
time, checking for events, 366-367
time of day, setting, 283
timeline units for animations, 288
TimeOfDay property (Lighting object), 112
Tipalti portal, 404-405
toggling
collisions, 11
Move and Rotate tools, 289
Tool Grip Editor plug-in, 309-310
Toolbox, 8-9
packages, accessing, 186
tools
creating, 306-307
displaying image in toolbar, 329
equipping in-game, 310-311
grip properties, 311
handles for, 307-309, 329
operational overview, 306
properties
grip properties, 309-310
miscellaneous properties, 314
purpose of, 305-306
requirements, 329
sword creation example, 311-314
ToolTip property (Tool object), 314
Touched event (Lua), 220-221, 232
with Debounce tool, 221-222, 232
exploding by touching example, 201-202
trap door example, 222-223
trading items policy, 390
trailers
attracting players with, 409-413
cost of, 421
Transform tool, 33-34
transforming objects, 33-34
translating objects. *See* moving, objects
translation tools
capturing text, 385-387
deploying translated text, 388

hiring translators, 389
translating captured text, 387-388
translucent bombs, creating, 197
transparency
of decals, 50-51
of parts, changing, 47, 56
performance and, 375
Transparency property (decals), 50-51
trap doors, creating, 222-223
TriangleCount property, 373
triggering sounds, 278-279, 282
troubleshooting. *See also* debugging
device emulation, 380
door swinging, 69
importing, 101
Lighting effects, 113
motors, 77
moving models, 269
render step priorities, 367
Roblox installation, 20, 40
uploading assets, 166
turning off. *See* disabling
turning on. *See* enabling
tweening in interactive GUIs, 247-248
TweenPosition property (ImageButton object), 247
tweens
between two points, 266-267
cameras, moving, 360-362
creating, 265
easing style and direction, 267-270
TweenSize property (ImageButton object), 247
TweenSizeAndPosition property (ImageButton object), 247

U

U direction (2D axis), 52
UI scaling in mobile-friendly games, 378
UIAspectRatioConstraint, 250
UIAspectRatioConstraint object, 378
UIGridLayout object, 248
UIListLayout object, 248-249
UIPageLayout object, 249-250
UISizeConstraint, 251
UIs. *See* GUIs (Graphical User Interfaces)
UITableLayout object, 249-250
UITextSizeConstraint, 251
unanchored parts
hanging from anchored parts, 60-62
in moving models, 269
undocking windows in game editor, 27
UnequipTools() function, 430
unions
creating, 372
triangle count, 373
UpdateAsync() function, 327
updating
animations, 302
games, 10, 413-414
notifications for, 419-420
planning updates, 422-423
packages, 187-190
uploaded animations, editing, 302
uploading
assets, troubleshooting, 166
audio assets, 171-172
decals, 49
hats, 7
images
for Game Passes, 396
for Skyboxes, 135
models to Roblox, 158-160
music files, 275-276
textures, 52, 56
user ads, 416-418, 421
user content management. *See* content management
userdata data type, 425
user interfaces (UIs). *See* GUIs (Graphical User Interfaces)
UserOwnsGamePassAsync() function, 398
users, adding to Team Create sessions, 180-182, 191

V

values, assigning to variables (Lua), 196-197
variables (Lua)
creating, 196-197
naming, 196, 230
purpose of, 195
scope, 206-207
V direction (2D axis), 52
video trailers
attracting players with, 409-413
cost of, 421

View Selector (Terrain Editor), 86

View tab

game editor, 25

menu bar, 44-45

viewing

collision geometry, 216-217, 232

Game Pass ID, 397

landscapes, 86

model details, 162

windows, 44-45

Village theme, 23

Visible property (Frame object), 246

Volume property, 278

Voxel, changing from ShadowMap, 375

VR (virtual reality) games, 381-382

W

WaitForChild() function, 353

dot notation versus, 348

wait() function, 204

walking surfaces example, 225-227

WalkToPart property, 430

WalkToPoint property, 430

water

creating in terrain, 93-94

deleting, 93

ignoring in Brush Settings, 92-93

waterfalls, creating, 155-156

web services in Lua, 10

wedges, purpose of, 360

welds, 63-64, 78-79

anchored parts versus, 63-64

in moving models, 268-269

while loop (Lua), 204

Width property (Beam object), 150-151

windows, hiding/viewing, 44-45

workspace

arranging, 26-27

initial setup, 194

world lighting. ***See*** **Lighting object**

wrapping functions in pcalls, 321

X–Y–Z

Xbox-friendly games, 381, 384

ZIndex property (TextLabel object), 239-240