MAKE YOUR OWN GAME
WITH
BUILDBOX
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>I.</th>
<th>Introduction</th>
<th>Page 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>II.</td>
<td>Creating the Basics</td>
<td>Page 3</td>
</tr>
<tr>
<td>III.</td>
<td>Collision Shapes and Object Settings</td>
<td>Page 18</td>
</tr>
<tr>
<td>IV.</td>
<td>Build Your First Levels</td>
<td>Page 32</td>
</tr>
<tr>
<td>V.</td>
<td>Making It Unique</td>
<td>Page 42</td>
</tr>
<tr>
<td>VI.</td>
<td>Actions Effects and Logic</td>
<td>Page 49</td>
</tr>
<tr>
<td>VII.</td>
<td>Animated Menus and Game Sounds</td>
<td>Page 65</td>
</tr>
<tr>
<td>VIII.</td>
<td>Making Multiple Game Modes</td>
<td>Page 74</td>
</tr>
<tr>
<td>IX.</td>
<td>Coin Shop and Multiple Characters</td>
<td>Page 81</td>
</tr>
<tr>
<td>X.</td>
<td>Exporting</td>
<td>Page 86</td>
</tr>
<tr>
<td>XI.</td>
<td>Tips, Tricks, and Hacks</td>
<td>Page 91</td>
</tr>
<tr>
<td>BONUS</td>
<td>Polishing Your Game</td>
<td>Page 103</td>
</tr>
</tbody>
</table>
Chapter I : INTRODUCTION

With this manual, you’ll learn how to make your first game from scratch with Buildbox. We’re supplying you with all the art, sound and tutorials you need... and there’s no programming required.

You’ll learn about Buildbox and how to use it to make a wall jumper game called ‘GLTCH’ from start to finish. A lot of information will be covered to provide you with a solid foundation. We’ll explain exactly what you need to do each step of the way and we will do it all without using a single line of code.

Ready to get started? The first thing you’ll need is Buildbox. If you don’t already have Buildbox, go to the URL below to get Buildbox and tons of game art, sound, music and more:

www.buildbox.com/make-your-own-game
What is **BUILDBOX**?

Buildbox is a game creation software that allows anyone to make a video game. Buildbox is the engine used for the creation of hit mobile games, like Color Switch, The Line Zen, Sky, Phases, The Pit, and more.

Buildbox is a powerful drag-and-drop game creation software that is so easy to use, anyone can make games with it. It also exports to multiple platforms such as iOS, Android, Amazon OS 10, TV OS, Windows Store, Windows EXE, and Steam.

Not only is Buildbox easy to use, game development is also incredibly fast. Great games like Color Switch and the Line Zen were made in less than a week!

With its fun drag-and-drop architecture, you’ll find that Buildbox is the easiest and fastest game creation software today. So, get excited, because you’re about to start an awesome adventure!
Chapter II: CREATING THE BASICS

In this chapter, you’ll get a detailed tour of the software’s options and settings, which will help you become more familiar with the features and editing tools that you’ll be using.

We’ll cover the creator tool and how to use it to make a playable skeleton of your game ‘GLTCH.’ We’ll also cover how to use the Scene Editor to customize your game with different backgrounds, characters, menu screen graphics, and more.

LET’S GET STARTED.

First, we start up Buildbox. We’ll be greeted by the welcome screen.
In the welcome screen you’ll find:

- a list of previous projects we can open with one click. The last item brings up a dialog to browse to any project on your computer.
- tutorial videos on Buildbox.com
- demos to help you get a feel for Buildbox
- news related to Buildbox, its teams, and its developers

But, what we want to click now is the “Create New” button.
Clicking the “Create New” button will bring out the Creator Window. Write the title of your game on the “Name” field. In this case, write ‘GLTCH’.

Pay close attention to the Menu and World Settings. These sections setup a lot of elements automatically.

In the creator window, under Basic Settings, you can set the Orientation of the game (which way is up), to either Portrait or Landscape. You can also set up the Score Type to Distance, Coins, or Points Collected. Under World Settings, you can set the Game Type to include a Single World or Multiple Worlds. You can also decide whether you want your World End Action to be Endless or have an End Scene.

All of these can be changed later – however, this is the fastest way to get your game settings set up before starting.
The most exciting part about the Creator Window is the Gameplay Settings. Check out the awesome list of Gameplay Types!

You can try them all out another time. For now, let’s have these settings:

- Name: GLTCH
- Orientation: Portrait
- Score Type: Distance
- Game Type: Single World
- World End Action: Endless
- Gameplay Type: Wall Jump
- Menu Settings: Has Pause Menu and Game Over Menu

Press Create once you’re settings are set up like this.
Clicking “Create” will take you to the SCENE EDITOR.
The Scene Editor is the screen you will be spending most of your time while using Buildbox. It is where you edit the scenes that make up a World.

Click on the preview button in the upper-right hand corner to see the game the Wall Jump preset has generated for us.

You’ll see that the game is already playable! Buildbox has already generated the basic gameplay mechanics, as well as a few scenes to help you get started.
Before we make edits on the game, let’s talk about

THE ANATOMY OF THE SCENE EDITOR (P1)

On the upper left-side of the Scene Editor is the Menu Editor button.

Underneath that is the Assets Panel. The Assets Panel is where all assets available for our game are listed, including any new ones we add. Assets are listed in expandable menus. We have menus for:

a. Characters
b. Game Objects
c. Game Actions
d. Power-ups
e. Effects

Next to the Assets Panel is the Scene Tree. It shows a tree of all items in the scene. You can select any item in the scene by clicking on it in the tree. By default, all items are listed in the order they were created. In the scene, items on top of the tree appear in front of other items below it. You can rearrange them by clicking and dragging an item. All non-background items are under Game Play Layer at the top. Press the trash icon down at the bottom to delete the currently selected item.

There are two columns to the right, one with an eye symbol, and the other with a lock symbol. Next to each item there are two dots that line up with those symbols. Clicking those dots will toggle their setting. You can use the dots located under the eye icon to prevent the display of an item. The dots under the lock symbol will allow you to lock an item down so you can’t accidentally move it.
Before we make edits on the game, let’s talk about

THE ANATOMY OF THE SCENE EDITOR (P2)

On the far right you’ll find the Options Panel. Clicking on an item will reveal the options for that item.

If an item is selected from the Assets Panel, options to modify that item’s global properties will appear in the Options Panel.

The play button at the top is used to preview the game. We’ll tackle the buttons next to it later on in another chapter.

In the left column of the Options Panel, you will see four buttons:

- **Show Game Frame**: overlay the game frame over the current Scene in yellow.
- **Lock Backgrounds**: Lock all background elements. Note: you can lock and unlock individual background elements through the Scene Tree.
- **Snap Movement**: snap to a grid on object placement, to make things easier to line-up.
- **Activate Connection Mode**: used for connecting items in the game.
- **Debug Mode**: This allows you to see the collision shapes of the objects on the screen.
Before we make edits on the game, let’s talk about THE ANATOMY OF THE SCENE EDITOR (P3)

Located at the bottom of the Scene Editor are “tabs” for each scene in the game.

The scene showing in the in the scene editor will be highlighted yellow.

Pressing the “D” key will duplicate the highlighted scene.

And of course, there’s the very thing that Buildbox got famous for: the active scene and its drag and drop features.

In the active scene, we design by moving objects around. We can scroll out if we like using the mouse scroll button and we can move the editor around by holding down the spacebar, left-clicking and dragging. We use ctrl z or cmd z to undo. We can keep things in a line if we hold shift while we're dragging.

We can also use the arrow keys for fine-tuning, the arrow keys will move things around by 1px or by 5px if the shift key + arrow key is used. We can duplicate objects by pressing w, a, s, or d on the keyboard.

Feel free to play around with an item’s handles a little bit. You can enlarge, rotate, move, and do just about anything with an item on the scene.
I’m sure by this time you’re excited to get into editing your levels. First, let’s find out how to change the graphics of an object in the game.

Let’s start with the platform. Click on the platform item in the Assets Panel. Its global options will appear in the Options Panel in the far right side of the Scene Editor. From the “MakeYourOwnGameGraphics” folder you downloaded from the site, select the Platform.png file and drag it into the default animation slot in the Options Panel.

Let’s do the same thing for the enemies. Click on the enemy item on the Assets Panel. From the “MakeYourOwnGameGraphics” folder, select the Enemy Spikes.png file and drag it into the default animation slot.

It’s that simple. It’s that quick. Now, let’s edit the backgrounds. First, unlock the backgrounds so we can select them and delete them.
The easiest way to add characters, objects, actions, and backgrounds is to drag and drop a PNG file onto the Scene Editor. To add a background to our GLTCH game, we simply drag the Background1.png onto the Scene Editor.

As soon as the mouse is over the Scene Editor you will see the overlay shown above. Drop the PNG file into the relevant segment, which is in this case “Background”. You should see the item appear in the Scene Tree, and in the current scene you are editing.

Now our background is in the game, but we want it behind everything else. In order to do that, either right-click on the background and select send to back, or drag it to the bottom of the Scene Tree. Elements at the top of the list appear in front of other elements and elements at the bottom of the list appear behind other elements.

Repeat the steps for Background2.png.
We can modify the speed of the background’s scrolling. Let’s try and make background1 scroll faster than background2. We simply select a background, modify and assign its speed number in the Options Panel. The higher the number, the faster a background will scroll.

Press the preview button to see how we’ve progressed so far.

It already looks amazing! Next, you’ll change the graphics for the character.
So far, we’ve been adding single image graphics into Buildbox. But, Buildbox can also compose complete animations from sequential PNG images.

When we select our character, you’ll see in the Options Panel that Buildbox offers slots for different instances of our character. We can have animations for when our character is shooting, jumping, moving, and defeated. When the character isn’t doing any of those, the default animation will play.

For GLTCH, you will only need the default animation for the character. You can delete the other animations by hovering onto the animation slot and clicking on the “X” on the corner of the slot.

Select all the images in the character folder and drag the entire sequence into the default animation slot.
Next, we need to change our menus to match our new graphics.

Clicking on the Menu Editor button (the button in the upper-left) takes us into the Menu Editor.

The Menu Editor is where we will find all the user interfaces in the game and the game worlds themselves. The blue nodes are world nodes and the green nodes are user interface nodes.

Double-click on the main menu so that we can replace the graphics inside just like we did before for the platform, enemies, and character.

Once inside the main menu, select the background and drag in the new image in a similar fashion.

We can scroll out if we like using the mouse scroll button and we can move the editor around by holding down the spacebar, left-clicking, and dragging. It works the same way in the Scene Editor. As we design our levels and menus, we can move things around, get a better look at things, and make a few adjustments using the image handles.
Next, let’s replace the start button. Note that if there are any issues selecting an asset, we can hide or lock things in the Scene Tree.

Drag the start.png file from the graphics folder into the image slot of the start button.

You can do the same thing for other items in the main menu.

To enhance the look of our menu, let’s add a few images. Drag and drop the character image into Buildbox. When the drag-and-drop-wheel appears, drop the image in the image option.
Using the lessons you’ve learned and the graphics you’ve downloaded, make your Main Menu, Game Over Menu, and Pause Menu look like the ones below:

![Main Menu](image1)
![Game Over Menu](image2)
![Pause Menu](image3)

Remember that you can edit the graphics of the items and buttons already in the menus.

You can transform, rotate, resize, move, and transform items in the menu using the handles that appear when you click an item. It’s not too different from the process you use when making levels in the Scene Editor.

You can drag in images and drop them into the image segment of the drag and drop wheel. Hiding and locking other items may prove useful in this exercise.

You should also replace the pause button in the World UI with the one from the graphics folder so that the overall imagery of the game is consistent.
Chapter III:
COLLISION SHAPES & OBJECT SETTINGS

So far, we used the Creator to build a playable skeleton of a wall jump game and we dropped in a ton of graphics to customize it, including our menus.

Here in Chapter Three, we’ll work on the foundation of level design. We will discuss editing the collision shapes for our new graphics. We’ll also bring a new enemy into our game and bring it to life by changing the object settings and the object components.

First, let’s talk about Collision Shapes.

Collision Shapes are what Buildbox uses to determine where objects collide. For example, a character looks like it hits the wall and has stopped because the collision shape of my character has collided with the collision shape of the wall, and so without collision shapes, the image for my character would just fly off the screen, and it would be no fun.

A collision shape is the shape of an object as "seen" by the collision/physics engine. Just because you have a circle for the image of an object doesn’t mean that Buildbox automatically realizes that the object is circular. We have to determine the collision shape for Buildbox to recognize it. Same goes for other shapes.
First, let’s start with the character. In the Assets Panel to the far left, select the character. You will see the global options appear in the Options Panel to the far right directly above the default animation. Directly above the default animation of any object and character is the button for editing the collision shape. Click on that button to open the Collision Shape Editor.

First, let’s get our object nice and big in the editor so we can be as precise as possible. Simply resize the window to increase its size. Similar to any editor in Buildbox the mouse scroll button will zoom in and out. Holding the spacebar down while we click and drag will move the editor around.

You’ll notice almost immediately that our collision shape is off. Our collision shape is the orange shape. It is what Buildbox uses to determine where it hits something. What we want is for the collision shape to match the graphic.
We have the right shape already because our character is circular and we replaced it with a circular graphic. Simply drag the collision shape from the middle and move it so that it is centered. You can move the side dot to resize the collision shape of the circle.

If the collision shape isn’t the same as that of our character, for example, if the collision shape is square, we can edit it using the options in the upper left corner of the editor.

We can choose between a polygon shape and the circular shape we have now. We can move the points around to get a more precise fit. We can add points. We can subtract points, and we can reset the image if we mess things up. When you’re done editing collision points for an object, simply close the window.
Now let’s edit the collision points for the enemies: the spikes.

Drag the points so it reaches the end points of the shape.

Remember to add a little leeway. The enemies do not need to have an exactly precise collision shape unlike that of the character. Whenever we create collision shapes for enemies, we want to make sure they are just a bit forgiving. This way we don’t run the risk of the user feeling cheated when they get killed, then deciding the game is unfair. Also, when you give the user leeway with collision shapes they experience excitement when they have close calls. That moment when they think – “I can’t believe I just lived through that”, they experience a lift in their excitement level making your game more fun, and adding a sense of achievement. So, leave a little leeway when designing your collision shapes and you will increase the fun factor of your game.
Since the enemy isn’t a perfect triangle, click the plus sign in the upper right part of the Collision Shape Editor window to add a collision point.

Note that our collision shapes must be convex. We can’t create concave shapes because it greatly increases the processing power required to run our game. Trust Buildbox with this one. Let’s work around that and create the shape below instead. It will have the same effect as it would have if the collision shape were the exact shape of the spike image. Remember to give a little leeway:
Now let’s open the One Way Block object. First, replace the graphics with the same one we used for the walls:

Next, make it so we can move through this object, but not go back through the other way. Let’s open the Collision Shape Editor.

Notice how the collision shape is off? To immediately get a precise collision shape for a square object, simply click the reset button.
The reset button gives you precise collision shapes for any object that is a perfect square.

Notice the big green arrow? This denotes pass-through properties for an object. It can be turned on or off with the button on the upper left side of the window.

We can move the arrow in any direction. Wherever that arrow is pointed, that is the direction in which the character can pass through. Any other direction, and the character will hit the block.

We can be creative with this and come up with clever game mechanics later on. Right now, just leave it as is. It’s only important that you’re aware of this function.
With all this talk about collision shapes, we should now talk about Debug Mode. Debug Mode allows you to see the collision shapes of all the objects in the game.

Simply click on the debug button:

Once it’s highlighted, you’ll be able to see red shapes that represent the collision shapes of the objects on the screen.

Seeing collision shapes is an important aspect of level design. You’ll be able to turn on Debug Mode while previewing the game as well.

Simply click the Debug Mode button to activate it.
Now let’s make this square enemy for our game.

To do that, search for the folder containing the images of this enemy. Select all the enemy images and drag them into Buildbox. Once the wheel shows up, hover on to the “Object” side of the wheel and then release.

You’ll see that it is now an object with an animation sequence using the images we dragged in. Let’s move it to the center of Scene 1.
Next, we need to edit its collision shape. Because of the way its spikes are set up, the circle collision shape works best for this enemy.

If you preview the game, you’ll notice that despite the enemy looking like a spinning spike machine of death, it doesn’t harm the character in any way.

To make sure this spinning death trap harms our character as it should, simply change the preset from platform (its default mode), to enemy.

With my Preset setting set to enemy, any time that pink circle touches me, he loses! Bwahahaha
Now let’s talk about adding linear movement to the enemy square. While it was just standing there, it seems hard enough to avoid, but we may want to add some movement to make it seem more “sentient” and give the player a little bit more of a challenge.

Still familiar with the X and Y axis you were taught in school? It will prove useful now. At last. Remember that the x-axis really runs left and right, and the y-axis runs up and down.

When you click on an enemy, or any object for that matter, that is in a scene, you’ll see the movement section in the Options Panel.

The first row represents the X axis. The second row represents the Y axis. There are two boxes per row, but we’ll discuss the first column of boxes first.

Simply put, if you put in a positive number in the X axis row, the object will move to the right. If you put in a negative number, it will go to the left.

Same goes for Y. Positive numbers make it go up. Negative numbers make it go down. Play around with these numbers and preview the game. Experiment on this and you’ll get an understanding of the linear velocity quick.
What about the angular velocity? The angular velocity puts the object into orbit. Try putting 100 in the angular velocity. See how the object orbits?

Now, try this again with the number 4. You’ll realize that the higher the number, the tighter the orbit. A smaller number makes its orbit so huge it looks like it is moving in an arch in the scene. Go ahead and experiment with this feature to learn more about it.

What about the second column? Let’s call it the randomizer.

Let’s say you put 0 in the box in the first column for the x variable and the number 10 in the box in the second column.

It will look like this: 0 ~ 10.

With this, the second column will add a random number between negative 10 to positive 10 to the number in the first column. It will be a random number every time the number appears.

Even if you duplicate the enemies, the randomizer will make them move at different speeds in the actual game.

If you put numbers in the randomizer for the Y variable and angular variables, you’ll see that the enemies are now moving unpredictably. Play with these numbers for a while.
Let’s talk about object components.
First, the Wake Up component.

The wake up component allows the object to lay dormant until we either collide with them or get close to them.

For our enemy, we want to have a wake up component that is distance based.

Try these settings for our enemy:

With these settings, the spiky square enemy doesn’t move until the character is 300 pixels away from the enemy.

If we want the enemy to stop moving when we’re 200 pixels away, we can simply enable the sleep option, choose distance based, and write in 200. Preview the game to see how the wake up component works.

How about the spawner component?

The spawner spawns additional copies of an object. The number we put in the first box determines how many seconds until we spawn another copy of the object.

In our example, which is also the default option, the number is 1. This means a copy of the enemy will spawn every 1 second. Note that the original copy of the enemy will not appear if it has a spawner component. You can also use numbers less than 1.
The “spawn action” option determines the conditions required to enable the spawning component of the object. Should the character be jumping or shooting for it to start spawning? If it is set to permanent, the only condition needed for the spawning to start is that the object should be awake. That works best for this game so set the spawn action to permanent.

We can combine the Wake Up component with the Spawner component. Try these settings:

With this, the spawning will begin when the enemy wakes up, which is when the enemy is at a distance of 300 pixels or nearer to the enemy. The spawning, and any movement we may have put in the linear velocity and angular velocity boxes will stop once the character is 300 pixels away from the enemy.
In this chapter, we’re going to take a closer look at world settings and how they affect your game. We’ll show you how to customize and edit your world settings to make your levels interesting. You’ll learn how to play around with various properties to adjust the gravity, time warp, bounce settings, and more within your game.

We’ll also further explain the different character gameplay settings, which more specifically defines the behavior of your character in that world. And then we are going to get into building more levels for GLTCH. We’ll learn the art of layering and how to test out your levels in the debug mode.

First, let's talk about world settings and character settings. The great thing about using the creator in chapter two in this manual is that the creator set us up with all the world and character settings ready to go. We could literally release this game and never touch a single world or character setting. It’s that thorough, but if we wanted to customize these settings, let’s talk about how we would do that.
First, let’s click the Menu Editor button. Clicking this sends us to the Menu Editor where you’ll see all the user interfaces and worlds that live in our game’s Mind Map. The worlds are blue, while the user interfaces are green. Clicking on a world reveals the settings for that world in the options to the right. You can set different settings for different worlds. However, for GLTCH, you only have one world so let’s click on that and see how the settings work.

First, you’ll see the setting for the name of the world. It’s not exactly important for GLTCH, but imagine a game with multiple worlds. It’s a good habit to name your worlds and user interfaces accordingly to avoid confusion when you start working on a more complicated game. Naming your worlds as you create them would make it much easier to find what you are looking for in the mind map.
Now let’s take a look at the gravity setting. The first box is the x-axis gravity and the second is the y-axis gravity. That setting structure should now be familiar to you after going into detail on the object settings in the previous chapter.

Gravity is the continuous force acting on every physics object as gravity does in the real world. Insert the platform object and set its object type to “physics”. Then, scale it to 0.5 by 0.5.

Play with the gravity setting and see how this object reacts when you preview the game. Remember to delete this object after playing with the gravity settings. Set the gravity back to 0 as we won’t be needing gravity for a game like GLTCH.
Time warp is the pace of the game. The higher the number, the faster everything in the game will go. Right now, its default setting of 50 is a good pace for beginner. A setting of 100 will be too fast and might scare off the new user. However, you can play with this in the future. Perhaps a turbo mode for GLTCH?

Friction simply refers to the friction between all objects. As you can imagine, a high setting will make it hard for a character trying to push a physics object on a platform.

Velocity Drag places a force to slow anything that is moving. It directly opposes all movement unlike air-drag for character, though very similar. Imagine moving through mud.

Bounce places a bounce force between all objects in the game.

Score Multiplier is for distance-based scoring. Scoring is 1 point per pixel moved. A value here will increase that. For example, a value of 10 will mean 10 points per pixel.

Game Direction is for the movement of the game. In clock-wise degrees where 0 has the character moving right, -90 will be moving up, 90 will be moving down, and 180 will mean moving left.
Camera Smooth is an advance setting for how jittery the camera gets for certain game mechanics. Best to leave it alone for now.

The Object Deletion Threshold refers to how far off the screen do objects need to be before being deleted. This is for optimizing your game for mobile devices.

Forced Movement is a setting that makes our game scroll or stand still like a platformer. You can play around with all the world settings to learn more, just make sure you reset them back to their default settings afterwards.

Now, click on the blue world and then click on “characters”. This will open up a list of all your characters. Click on your character. We’re going to talk about character settings. They’re similar to the world settings. The difference is that while world settings apply to the whole game, character settings apply only to the character. We can have multiple characters with different character settings.

You’re already familiar with the animation settings from the previous chapter. We have sound settings here which are drag and drop. Buildbox will accept mp3s so just drag an mp3 into those settings. There are settings for lighting and tilting your character when it moves. We have game over effects which we’re going to cover in another chapter. There are also options for how this character becomes available to the user.
Let’s turn our focus towards the Character Gameplay Settings. There are two boxes for some of these settings, these boxes are for the X and Y movements.

Max Speed is just that. The maximum speed allowed for a character’s movement.

Bounce Force is just like the one for the World Settings, but it applies only to the character. If you ever want to make a game like Trey Smith’s Phases, this is the setting you want to play with.

Jump Force would be how powerful we want the character’s jump to be. Jump Timeout allows us to vary the size of the jump. A higher value allows the player to hold the tap longer to allow for a bigger jump. Jump Counter is how many jumps can be executed before hitting the ground. If you want to have a double jump feature for the character, we put a 2 in here.

Ground Threshold is a setting to detect when the player is on the ground. Unchecking “Jump From Ground” allows the player to jump in midair.
Left lean force and right lean force will lean and rotate the character in that direction.

Platform Friction is the force that brings the character to a stop on a platform. A lower number gives the character an icy slide effect on a platform.

The Rotation Drag is just like the velocity drag, but it applies to rotation instead. Meanwhile, Air Drag is similar, but it slows down the character’s movement in the air.

Direct Movement allows the character to move around the screen using directional controls.

Forced Movement is a setting that moves the character with the game.

Fixed Rotation means the character will never rotate.

Image Direction means the character will point to the direction of movement.
You may have noticed the Advanced Move section of the Character Settings. We’re going to talk about that and the components in another chapter. Right now, you have a pretty good idea of the settings we need to modify gameplay.

Speaking of gameplay, we need to add more scenes to our game. Delete unnecessary objects from the scene by clicking on them and hitting delete. You can also do this by deleting them from the Scene Tree.

Drag objects from the Assets Panel into the scene. Once they’re in the scene, don’t forget to change their settings.

You can also copy objects from one scene to another with [ctrl] + C for PC or ⌘ + C for Mac. You can paste them on the scene with [ctrl] + V for PC or ⌘ + V for Mac.
I think it’s time we create a new enemy to add variety to the game.

Another enemy? Are you kidding me?!

Same as we did before, search for the folder containing the images of this enemy. This is what it looks like:

Select all images in the “Ball Enemy Animation” Folder and drag them into Buildbox. Once the wheel shows up, hover over the “Object” side of the wheel and then release.

Don’t forget to change its collision shape to make it circular and adjust accordingly. Using the lessons we learned in the previous chapter, change its preset to “Enemy” and give it movement in the scene. Now’s a good idea to reread and review the previous pages.

Your mission, should you choose to accept it, is to add ten more scenes to your game. Remember, selecting a scene and pressing the “D” key will duplicate the scene.

You can select a scene and press “S” to solo it, which means when you preview the game only that scene plays. You can also press “M” to mute the scene so that it doesn’t appear when you play the game. These are great for testing.
Keep in mind that you should try and make all your levels unique. This way, players won’t feel like they’re just repeating the same levels again and again.

Because the scenes are randomized, remember to test the levels out so that any scene works when followed by any other scene. Speaking of randomizing, sometimes enemy movements have to be more specific. Too much randomization can make your game unplayable. Always ask yourself while making a level if your game is too hard or too easy. The most popular games have the right balance between the two.

Now, build those levels. Have fun!
Our goal is to help you create a game that could potentially be featured by Apple or reach the top of the charts. To accomplish this, you have to make your game unique. There are hundreds of games released on a regular basis across every single platform. This makes originality crucial for not only visibility in the App Store, but overall player engagement.

In this chapter, you’ll learn how to make a game that’s truly unique. We’ll go beyond graphics and explore game characters and the way they interact. Watch and learn as we add a new gameplay element to ‘GLTCH’ to set our game apart enough to make it memorable and fun.

You’ll learn how to add a barrier obstacle and a new way to add a fun twist. We’ll also show you how to easily create a ‘taking damage’ animation in Buildbox.

We already made the game unique with the amazing graphics, but it doesn't end there. We need to offer a gameplay element that sets our game apart, and makes it memorable and fun. In the previous chapters, we used the creator to get us started on this journey, but we want to offer more than just a good starting point.
Let’s add a new gameplay element. I think it would be neat if we came upon a barrier that we can place anywhere on the screen. A barrier would narrow the space our player will have to maneuver around obstacles, and we could offer some variety on that blockade by leaving openings to travel through, and force the player to make decisions on which side of the screen to travel on.

But the barrier itself isn’t the fun and memorable part. We are indeed going to add a twist to this new element later in this chapter.

We start by importing a new graphic which is very similar to our walls. Let’s duplicate the last scene by clicking on it and hitting “D” to duplicate it, and let’s delete our swirly enemies from this scene, so we have more room to work this all out. Now, to import our new element, just drag the Barrier.png graphic into the object option on the drag and drop wheel, select the object in the Assets Panel, and check the collision shape.
Okay, so let’s hold shift down while we drag this to keep it in a nice line. Put it near the bottom of the scene, and we’ll duplicate it upwards. Hold shift while dragging the duplicate to the top of the scene.

Now let’s solo this scene by selecting it and hitting “S”. Before we test this, this would be a good time to mention that Buildbox has the ability to make our game generate all our scenes in the order they appear at the bottom of the Scene Editor, or generate them randomly for endless gameplay.

To do that, we have to open the Timeline Editor, which is done by pressing the arrow where the cursor is in the above image. This will easily show you all the scenes and how they are arranged. To go back to the simplified view, press the same button again.
The Start scene is a special scene that will always be first. All other levels can be placed anywhere in the timeline. If there are no overlaps, then scenes will just appear in order.

Where two or more scenes overlap, Buildbox will randomly select from each of them at run-time. Each level will have an even chance of appearing.

A scene in green means it is soloed, and scenes in red are disabled from the Preview. Now, we shall test our game to make sure placing the barriers does not ruin the gameplay experience.

Unfortunately, this screenshot might be the end result. Remember, if you are going to randomize your scenes, one scene has to work with all the other scenes in the game. What we see here with this small and unplayable gap resulting from our last scene ending and beginning with itself is an example of why, since it serves no function. It just looks sloppy, and if this mistake kills our player, that will cause issues with player retention.
Let’s make this more compatible with similar setups from other scenes. Let’s drag these barriers more into the middle, remember hold shift down while dragging to keep things in a nice line, and let’s adjust the size of our mountain enemies so it’s a little easier to get by on this scene. You can hold shift while resizing these to keep their aspect ratio.

But, what if when we hit this barrier, it threw us up against the wall as a punishment mechanic, essentially making it a bumper. The way we are going to achieve this is by learning about a character component called advanced move.

Click on the character on the Assets Panel to bring up its global settings on the far right so we can tinker with these advanced moves.
This whole time, you have been seeing advanced move components in action every time we play GLTCH, combined with a switch button on our games user interface, which we will see in a later chapter. The advanced move has been changing our character’s linear velocity every time we tap the switch button that’s covering the screen. When you look at the character settings, you’ll see that this component gives us our initial velocity at the game start. The velocity is changed every time the switch button is activated.

Now, click on “Add Component” and click on “Advanced Move.”

For the new “Advanced Move” we have to make it so that the character changes the direction when it collides with our barrier no matter which direction the character is going. To do this, let’s set it to collide, and select our barrier as the affected asset. Give it a very tiny timeout, so the calculation stops immediately. The best way to change an object’s direction to go in the opposite direction is to change its positive or negative value to the opposite value. Let’s set our advanced move to multiply our x-axis velocity by -2 and have no change on the y value. Now when we hit the barrier, we should see an instant change in our direction along the x-axis.

When you test it, you should see that it looks perfect. What would be really neat, is if the bumper changed in some way to provide even more feedback when we hit it. I think we can achieve this by making this an enemy that does zero damage. This way, when we hit it, we can do damage to it, and trigger an animation called a Taking Damage Animation. This will look really neat when we get it all set up, and we only need a few steps.
Select the barrier from the Assets Panel to reveal its global options on the far right. Add two components: the first being “Health”; and the second being “Damage.”

Let’s give the barrier a Damage of “0” so it won’t harm our player in any way. We can give it a health of “999” so it can survive the damage our player will give when it hits the barrier.

Next, we drag in the PNG sequence from the “Barrier” folder into the Taking Damage Animation slot. We then make the barriers enemies in the scene. Don’t worry, since it has 0 damage. It won’t harm our players in any way. What this merely does is our character can now harm the barrier and make it play its “Taking Damage Animation” to give the barrier the bumper effect we’re looking for.

To trigger the “Taking Damage Animation” for the barrier, we need to give our character a damage component. Click on the character on the Assets Panel to bring up the global settings on the far right of the screen. Click on “Add Component” and click on “Damage.” Since our character now has a damage component, we need to increase the health of all our enemies in the game, which is at a default of 1. This way, enemies don’t disappear when our character crashes into them.

Take the scene off of solo and preview our game. Test if the barriers are playing their animations nicely every time our character bumps into them. You’ll have to make sure all bumpers are set to “enemies” for this to work. You may have to make a few adjustments to the other objects as well.
Chapter VI: ACTIONS EFFECTS AND LOGIC

In this chapter, we’ll cover actions, effects, and logic pieces in-depth. You’ll learn how to add actions like coins, invincibility, power-up magnets, and action animations.

Creating an in-game economy with coins and other actions such as power-ups is an excellent way to give your players goals and reward them. When you make your own game with this in mind your overall gameplay experience and player retention will be much better. The option to collect coins also empowers users to unlock characters and other special items you decide to include, which can lead up to an in-app purchase, generating real money.

We’ll cover how to import coins for players to collect, a coin tracker, invincibility power-ups, and path logic pieces.

You’ll also learn how to add beautiful light effects and particle effects in creative ways to make your game look really professional.
First, look for the PNG animation for the coin in the graphics folder. Select the entire PNG sequence for our coin, and drag it into Buildbox, dropping it on the action option of the drag and drop wheel.

Let’s select the action in the Assets Panel and make sure we name it something easy to find in the Scene Tree. Also, actions can be tied to buttons, so, make sure you name your actions so you can reference them in the button’s settings.

Next, let’s check out the collision shape and make the appropriate changes, but this time we want to go in the opposite direction than we did with enemies because this is something the player will want to hit, so let’s make large enough for them to do so.
There are many different types of actions that can be created in Buildbox:

- a coin that we can use as the currency of our in-game economy;

- a kill all enemies action which will kill all the active enemies in the scene, meaning enemies with wake components that are not yet active will live through the action, but otherwise, using this will take some tweaking if we don’t want to sterilize our scene of all the aesthetics;

- Invincibility, which will make our players survive anything that would normally kill it. We are going to add an invincibility power up later in this chapter.

- a power-up magnet, which is an action that, when collected, will attract all the other actions on the screen towards your character;

- the strike is an action that will kill all enemies in contact with the player;

- set checkpoint will save the users progress, and next checkpoint will move the player to the next set checkpoint;

- restart checkpoint will restart the character at the last set checkpoint touched;

- and finally, an action that changes gameplay options in something that is a mix of world and character settings
The action animation is the animation that happens when the action is collected by the player. The animation can take place where the action was when it was touched or displayed full screen. The animation can overlay or replace those options. We have sound options here for the action. Remember, Buildbox will accept mp3s. Just drag them into the settings here. In the properties section, we can set actions to reward coins or points, and show either upon collection. The duration is how many seconds the action will remain in effect, and the appearance chance is what percent chance the action has of showing up in the game. We have options to trigger a camera shake or camera flash when the action is collected. This is measured by seconds. So, a five in each would shake the camera for five seconds, and the camera flash would dissipate at the end of five seconds.

Now, let’s set our action to coin, with its animation placement at the character as an overlay, and reward two coins per collection. Let’s duplicate the coins along the wall, and take a look at our action by previewing the game.
Upon previewing the game, you’ll see it’s working correctly. However, we don’t have a way to keep track of how many coins we have collected. Let’s fix that. Click on the Menu Editor button and go to the World UI. Add two labels into the UI. One in the Upper left, with the function “User Text” and have the text say “Coins.”

Duplicate this label to the right, so it stays perfectly in line. This time, change its function to “Score” keeping track of all the worlds even though we currently only have one, and change it to count coins. We want to keep track of all the coins that are collected, so the user knows when they finally have enough for something they’ve been wanting to buy in the coin shop, which we will create later in this chapter.
If you preview the game you’ll see that our changes in the World UI now allows us to see the number of coins we collect. But, we can hardly see it. We’ll have to fix the font colors. Click on the Font Editor button next to the preview button:

Here, you’ll end up in the Font Editor. This screen is where you set the font used in the labels throughout your game. The fonts available are all fonts installed on your computer, so if you want more fonts for Buildbox, just install them as per normal on your computer.

Add a new font, name it something like “Coin Tracker” and use the eyedropper tool to make its color the same as the Pause Button.

Let’s also turn off the gradient, and let’s close the font editor and make sure our font selection for our new label is set to our new font. Let’s have a look at our label now by previewing the game. That looks much better. You can make the necessary adjustments we need to get it looking good.
At this point our game is looking good and nearing completion. Let’s add the invincibility power-up mentioned a few pages ago.

![Invincibility power-up](image)

I’m invincible!

Look for the PNG sequence for the shield. It’s in the Power Up folder in the GLTCH_Extra folder. Select all and drag them into the “Action” option of the drag and drop wheel. Using the lessons learned in this chapter, name it, change the collision shape, and choose its action type to Invincibility. You can put the Invincibility Power up in scenes where you want the character to pick it up.
Now, let’s drag in an animation for the invincibility that communicates its lifespan. We always want to have a way to tell our player when invincibility is going to end.

Let’s go into the invincibility actions animation editor and adjust our speed to play our animation for the duration we want. Buildbox will always play animations at 30 frames per second. So, let’s adjust the speed accordingly. The proper speed here, using our Math prowess, is 0.033.

Now, we need to set the animation placement and behavior, and also make sure the duration follows our frame timing we set up in the animation editor. In this case, duration should be 4 seconds.
This time, let’s add some light into our scenes. Lighting is a great way to set an atmosphere and to use coloring and shadows in creative ways. To the far left in the Assets Panel, let’s open the effects menu, and pull a light into our scene.

With the light selected, there are some options we have in the Options Panel to the far right. We can name the light so if we have several lights in one scene we can easily find which one we are looking for in the Scene Tree. Clicking on the color bar will bring up the color chooser. You can use the eyedropper to select a color or cook up whatever you like and our lights have the ability to cast shadows. You can also play with opacity. Play around with this settings until you achieve the desired effect.
Now, let’s try and play with the Particle Effects. Drag one into a scene from the same panel you dragged the light.

Click on the Edit button in the Options Panel so we can play around with emitters.

This will bring up the Particles Editor. Let’s use the mouse scroll button to zoom in and hold the space bar while we click and drag to move the editor around, and let’s take a look at the options in this editor. This window can also be resized, so make it as nice and big as you like. We can move the position of the emitter around, or we can randomize the x and y value of its position. For instance, if we put a 100 in the x-axis randomizer, the particle will emit at random positions 100 units on either side of 0 along the x axis. That setting structure is likely familiar to you by now. Let’s change that back to 0. We can rotate the emitter and change the scale. This is where the emitter gets really fun. Let’s drag in a new texture for the particle it is emitting.
Total particles denotes how many particles to have on the screen at any one time. So, a new particle will not be spawned unless existing ones dissipate. If we change this to three, there will never be more than three on the screen at one time.

The emission rate is how fast to emit the particles. So, if we change this to one, it will emit one particle per second.

Checking Blend Additive means that overlapping particles will add to each other making the overlapped areas brighter. This is great for spell animations and especially flames.

Start size is the size of the particle when it is spawned, and we can randomize that as well. So, if we put 30 as a starting point and 25 in the randomizer. We will get a bunch of different sizes coming out between five and 55. End size is the size of the particle when it dissipates. If we put a 1 in the end size, every particle, no matter the start size, will shrink into nothingness.

Start spin is a rotation of the particle when it is spawned. So, if we put a 45 in the randomizer, you can see them coming out at different rotations. End spin is the rotation when it dissipates. So, if we change the start spin to 0 and put a 90 in the randomizer on the end spin, you can see all the particles rotate as they dissipate.
Gravity is the wind force on the particles. They are also the settings for x and y, so if we put a 25 in the x-axis setting, the particles will dissipate along the positive direction of the x axis. If we put -45 in the y setting, you see that they get blown down along the negative direction of the y axis.

Radial acceleration is how much acceleration, away from the point of emission, is given to the particle.

Speed is the initial speed of the particles when they spawn and that can also be randomized. The angle setting is the initial direction of travel.

And of course, we can randomize the angle. Life is the life span of the particle before it begins to dissipate. If we make that 1, and remove the randomizer, you can see that the particles last one second before they end.

The color is the tint of the color when it is spawned, and when it dissipates. Color variation is a setting to allow the variation in the color setting, so, if we put yellow in our starting variation, some will be tinted yellow. And finally, the position type of the particles is how the particles will act once emitted from a moving emitter.
We can also add particles to the animations of the character. Open the character’s default animation editor and drag in a particle and send it to back. Open the editor for the emitters.

Grouped means the particles will move and dissipate in a manner that looks attached directly to the emitter.

Free means that when the particles emit, they are free to dissipate based on where they spawned.

Relative means the particles dissipate in a manner that looks attached to the game world and not the emitter. This setting most closely resembles real world emission, like the exhaust on a motor vehicle.
Now, let’s talk about Logic. This is where objects that dynamically control game objects appear. It appears under Effects and above Labels on the main bar to the left. There is currently a single Logic item available. That is not the only thing that can be used to alter gameplay. Actions can also be used to alter gameplay settings.

Let’s go ahead and delete the particle effect from our character’s default animation, and also delete the particle emitter in our scene. Drag in a new one before we talk about logic pieces. Logic pieces are items that dynamically control game objects. A path logic piece can force objects to follow its path or force our character to follow its path as well. Let’s drag a path into our scene right above our particle emitter. After dragging onto the design screen, you will see the Path represented as a green line.

Note the following elements:

- Click this to select the Path

- Click the larger version of this to add a segment to the end of the line. Click any of the smaller versions in the middle of a segment to split a segment in two (if they are not currently visible, the split action will still occur).

- Click and drag a corner like this to move it

Having the path selected, we see the options in the Options Panel. We can name our path or adjust its position. The search radius is how close an object or character needs to be to the starting point of the path in order for it to be forced to follow it. You can see the change easily if we put 500 in there. Put 200 because if the search radius is larger than the path, it could force us to never leave it.
The speed is how fast affected assets will travel along the path. Again, let’s put 200 in there. Attraction force is how much magnetic force is in each joint in the path which maxes out at 100. So, low numbers will make for some very interesting gravitational effects. As a path is simply a magnetic force which can be escaped from with certain gameplay styles, keep velocity allows the object or character to return to its original velocity once it exits the path.

Image direction is used to keep the graphic of the affected asset pointing in the direction of travel. Play mode allows for some variations on how the path is followed. And finally, let’s select our character in the affected asset option.

Let’s test this path out:
The path is invisible in the game but if we turn on debug mode, we can peek behind the curtain and see the character follow the path.

Now let’s copy and paste some elements from other scenes, and drag some coins in along the path, and see it all come together.
Chapter VII: ANIMATED MENUS AND GAME SOUNDS

No game is complete without music or sound effects. Music and sound are principal elements in gaming. Good background music sets the atmosphere and tone. It can help users connect emotionally with the game as well. Sound effects at crucial moments or during specific actions can inform players and give them direction on the next move to make.

In this lesson, we’ll cover how to not only animate the menu sections like the game title or start button, but how to load music into your game and trigger specific sound and animation effects as well. You’ll learn new ways to integrate sound and music throughout your game.

When you’re a game developer, it’s important to get your music and sound effects in your game just right. So, included in this lesson is also some bonus material that focuses on how to add defeated sounds and animations properly to your “Game Over” menu. The secret to making a good game great is in the details, like sounds and effects.
Let’s start off with animating the main menu. Let’s click on the Menu Editor button in the upper left and Buildbox will take us up into the Menu Editor. And now let’s double-click on our main menu user interface. At the bottom of the window is where we will find the options to animate menu elements. Let’s say whenever we open this menu, for instance, when the game starts, we want the title to come down from the top and the start button to come up from the bottom. Let’s start with animating the title. Let’s select the title and, at the bottom where the movement editor is, let’s select “Open”, as this is the animation we want to happen any time we open this menu. Let’s push the record button, so we are in record mode, and now let’s move our title off-screen towards the top.

As long as the record button is enabled like it is, once we get our title into position and let go of the mouse button, Buildbox will record the position data at that frame. We are at frame 0 and we can see a green bar added to the movement editor. These green bars signify a position point for Buildbox to use, and we can add quite a few things we want Buildbox to record.
Let’s expand the movement editor at the bottom. If we want to rotate the title, Buildbox will remember the rotation at that frame, and as we feed Buildbox more position points along the timeline, Buildbox will extrapolate everything in between for us.

Now let’s move our orange frame bar to the next point in time that we want to record a position point. Let’s say 20 frames in, and let’s drag our title to our desired position and let go of the mouse button. When we do this, we see another green position point added to our timeline at the bottom. Take a look at what happens now when this menu opens. If done right, the title starts off the screen and enters as we intended.

Let’s rewind our orange timeline marker back to frame 0, and shift drag our start button off-screen. Now let’s drag our orange timeline marker to frame 20 and drag our start button to where we want it, and when we let go of the mouse button, we see another position point recorded for our start button.
We can also play with what we have inside the editor, so we don’t have to preview our game every time. Just take the editor off record and bring our orange frame marker back to zero and play it. Let’s animate our character to fade in off the wall as it comes to its resting point. Let’s hide everything except the walls and the character. Lock the walls and let’s zoom in where it’s nice and easy to see. Now, we have our title and start button coming to rest at frame 20, and I want our character to be exactly where it is now at frame 20, but I want it to fade in with some movement.

Select the character, and at frame 20, let’s click in the position boxes in the far right of the screen and just hit enter. This will force Buildbox to record a position point. Then at frame 20, let’s set the opacity to 1. Then we can bring our orange timeline marker to frame 0 and move our character more against the wall and set the opacity to 0. Let’s take off the record button and play the movement editor by pressing the play button next to the record button.
Now we’ll get into the idle animation sequence by animating the start button by giving it a little shake. Remember that Buildbox animates at 30 frames per second, so 60 frames will give us a two-second loop. Select the start button, and get our timeline marker to frame 0, and enable record. Select the rotation timeline, and let’s make sure we force Buildbox to record a position point by clicking in the rotation option and just hitting enter.

![Image of Buildbox animation](image)

We see the position marker appear in the timeline. Let’s do the same thing at frame 10, but for frame 12, 14, 16, and 18, let’s shake things up by changing the rotation between negative 2 and 2 degrees until again recording 0 at frame 20. Now, let’s preview our game. The start button should do a little shake every two seconds.

Now, using these lessons, you have to animate the Game Over Menu and its buttons. You can also add a “Close” animations for the menus. Be creative with your menus and have fun.
Let’s talk about sounds. No game is complete without music and sound effects. Although it is in style to not include music in some games today, we will just give our player the option to mute it if they so desire. So, let’s start by bringing some music into the game. To hear music in the game, we actually need to load music into the user interfaces that we use. Having just animated our “Game Over” menu, let’s go back up into the Menu Editor, and put music in the game over the menu that will get triggered every time it loads. You can see the option for music here. Just drag an mp3 into that slot and you’re done. It’s that simple.

Now when the “Game Over” menu is loaded, Buildbox will play that music until it encounters another UI with music on it. So, if we wanted a different song to play when the game goes back to the main menu, we would select our game’s main menu, and add music there. But, keep in mind that each time our game loads, it will load that music. So, if our player is dying a lot and restarting different music over and over and over, that might get really annoying. So, one way to avoid this is to put music on just the start menu, and then no other menu would trigger or retrigger music. And if we already added the music we want to another menu, we can just add it here without dragging it in again.
This time, let’s talk about sound effects, and the perfect place to start is right in our main menu. We have options to add sounds to a lot of different things in Buildbox, including buttons in our menus. So, let’s drag an mp3 into the click slot of our start button. If we wanted to add the same sound to the other buttons in the other menus just like the music, we don’t have to drag it in again. We can just select it in the slot.

Let’s put the same sound in our pause menu buttons and our buttons we just animated in the Game Over UI as well. Now that we have animated our Game Over Menu and added sounds to the buttons, there are some options we need to look at when the player dies that determines how soon the game over menu gets loaded.

First, go back to the Scene Editor and select our character from the Assets Panel. Drag in an mp3 for the Defeated Sound slot.

Now when we die, we should hear the defeated sound play. You will also notice that a few things happen before the game over menu loads. We see our player turn into a particle effect, we see a camera shake and a camera flash.
We may want to change the default defeated animation Buildbox has generated for us. The defeated animation can be changed. We can just drag in any image or PNG sequence into the option. We just adjust the speed at which the frames play. We can loop it if we want, but, if not, and we would rather give the death animation time to play out, we just have to set our game over effects properly.

Game over delay is measured in seconds. So, a 2 in here would give us a two-second delay from the time the player dies to the time the game over menu loads. The camera shake and camera flash is the same as it is in the actions that we covered in part 5, and is also measured in seconds. Let’s make our camera shake 1.5 and our flash 0.25. Fall attribute is a setting that if set to a non-zero value, it will cause the character to fall with the gravity or direction of the game and the number represents how many times the character will bounce before falling. So, this is great for platformers, but it also works with our game because if we set this to zero, we see that our player pretty much stops dead while the game over mechanics plays out. So, let’s make it 0.5.
Let’s add some other sounds into the game. Let’s select our swirly enemies here and give them some sounds to play when they wake up, and also let’s give our bumper a taking damage sound. So that every time we trigger that cool taking damage animation that throws us up against the wall, it also plays a nice sound with it. Let’s add a sound to our invincibility power-up, and also to our coins. Preview the game and hear our game really come together for the first time.
Chapter VIII:
MAKING MULTIPLE GAME MODES

Congratulations! If you’ve followed the steps right, you’re already finished with the basic gameplay for GTLCH. It’s time to add in a “Turbo Mode” and a “Hard Mode” to increase user retention for the game.

Balance is crucial when you make a game. You don’t want your game to be too easy, but at the same time, you don’t want to make it so hard right off the bat either. It’s important to find an equilibrium. This can be easily managed when you implement multiple game modes.

In this chapter, you’ll learn how to make a game with multiple game modes for your players to choose from. Watch and learn how to make an exciting turbo mode and a hard mode all within Buildbox.

You’ll also learn how to make multi-level menu navigation for ‘GLTCH’ that can be used to navigate players to the new game modes. We’ll show you how to add lock buttons and share a few tips to make the entire process of adding multiple modes into your game super easy and quick.
Let’s make that turbo mode right now. Making a turbo mode may be easier than you think with the power of Buildbox. In fact, we could easily make one right now with the world we have and only a few short steps. Let’s start by duplicating our world. Yep, you heard me right. We can duplicate entire game worlds. Let’s select our game world and press W, A, S, or D to duplicate it. Now we have enough for a turbo mode, a hard mode, and whatever other mode you can think of.

Before we go any further, we are going to name these worlds, so we know what’s what.

Let’s name our original world mode – ‘World 1 Original’, then name another one ‘World 1 Turbo’ and another one ‘World 1 Hard’. We’ll leave these other two for whatever inspires us, maybe, even more, worlds, and just use what’s already there as a sandbox to copy and paste from. So, let’s just name one of these worlds ‘World ?’, and the other ‘Sandbox.’
That only took a few seconds, and we laid the foundation for our different modes by duplicating entire worlds. If that world had 200 scenes in it, we just saved ourselves weeks of work with that feature. Let’s give our players a way to get to these different modes.

Let’s copy our main menu so we can use a copy of it, and tweak it for our different modes. Let’s name it ‘Turbo UI’ and let’s double click on our Main Menu UI and start.

In our Main Menu UI, let’s add a navigation button, we can just drag in our graphic, the one that looks like the image below, and drop it on the navigation button option on the drag and drop wheel. Place it below the start button and make sure to name it “Turbo”.

**GLITCH**

**TURBO**
By now, the Main Menu UI should look like this:

Let’s click on the Menu Editor button in the upper left and go back into the Menu Editor. Let’s connect our turbo button to the new Turbo UI. This can be done by clicking on the circle next to the word “Turbo” and dragging the line into the circle labeled “Load” in the Turbo UI.

That’s how easy it is to connect Navigation Buttons to other UIs and other worlds.
Double Click on the Turbo UI. Let’s update some graphics in there to distinguish it from the normal menu and give it some personality. Drag in “GLTCH_0001_Background_Dark-Version.png” and change its opacity. Arrange the image in the Scene Tree so it’s above the background.

Make sure that record is not enabled. Let’s add a label in our coin tracker font to read turbo. Let’s scale it to 2, rotate it a bit over the title, and let’s give the title a 0.75 opacity so that turbo is a little easier to see. Now, enable record and animate the label to fade in at frame 20. Disable record and test it out. The Turbo UI should look like this:

Drag in a new navigation button so that the players will be able to return to the Main Menu UI. You can connect it to the Main Menu UI via the method we used in the previous page, or just give it a “Back Jump” function in its settings. Decrease its opacity to 0.75 and record its animation.
Use the same methods to create a “Hard UI”. This means there should be a navigation button to go there from the Main Menu UI, a button in Hard UI to return to the main menu, and the Hard UI must have distinct graphics and animation as well. Instead of “GLTCH Turbo” for the title, it should read “GLTCH Hard”.

Now, we can just connect the start button of our Turbo UI to the World 1 Turbo, and the start button of our Hard UI to World 1 Hard. Let’s select our World 1 Turbo in the Menu Editor and change the time warp to 75. Change World 1 Hard’s time warp to 57.
To make these modes unlockable, let’s change the start buttons on our turbo UI and hard UI to be lock buttons. Let’s double click on our turbo UI and select our start button and delete it.

Now, let’s drag a lock button from the Assets Panel into the menu. Lock buttons can be set up to be unlocked using in-game currency like coins or as an in-app purchase for real money. Let’s start by naming the button ‘start’ and dragging in our start button graphic into the image option. And then we need an image to display while it is still locked. Try to use something that communicates its cost.

Now let’s scroll down and set its price. For now, make it ten coins, so it’s easier to test, but you would, of course, make it match the communicated cost.

Let’s repeat the process for the hard UI, let’s delete the old start button, drag in the lock button, get the appropriate images in order, set the price, go back out, and make sure the new lock buttons are all connected to their appropriate worlds, and play our game.
Adding coin shops with unlockable characters and items can significantly improve your game’s replay value. Players are more likely to continue playing to earn enough coins to purchase or unlock a new cool wacky character that may have caught their eye. It’s also a great way to ease monetization options like in-app purchases into the mix.

In this chapter, we’re going to show you exactly how to set up and customize a coin shop. We’ll also explore what you can put in your game shop and different ways for players to buy characters.

You’ll learn how to add character purchasing features to your shop, and we’ll explain the various character settings available further as well. The lesson will also teach you how to add unlock buttons to each item or character for advanced monetization later down the line.

There’s also a really effective trick included in this lesson on creating multiple characters via duplication that will make it easy to fill up your shop with unique unlockable characters.
Let’s start by double-clicking on our main menu and bringing in a new graphic for a navigation button to head for a coin shop we’re going to make.

Let’s name the button, and go back into the Menu Editor, and duplicate our turbo UI. Let’s rename this UI coin shop, and double click on it, and go in and remove buttons and other things that we don’t want to show. Okay, now that we have our coin shop primed and ready for a character store. It should look like this:

This is where we’ll be putting the character selection options for our player. But first, let’s make some new characters for our game.

Let’s go into our original world and select our character in the Assets Panel. To create a character store, we need more than one character, and there are a couple of ways to do this. 1. We could drag in a new PNG sequence for our new character and go through and get it all set up as we did with our first character, or we could just duplicate our current character and tweak it.
Let’s select our character in the Assets Panel and hit D to duplicate it. Let’s be sure to name our original character. Let’s replace our default animation for this new character, and rename it as well. Let’s double check the collision shape.

Now let’s scroll down to the monetization options. You’ll see a few options in here, but we want in-game currency. We’re going to set this real low, so we can test it out. So, let’s just make it 10 coins.

Now, at the bottom, let’s add a character component. Let’s add a character icon component. Now, remember anytime there isn’t another graphic to display, Buildbox will display the default animation. So, we could have the default animation play in the character store, or we could add an image or sequence for what we would want to see in the shop when it’s locked or unlocked. Let’s repeat the process for another character, we could make as many as we want, but a few will get us started.
Let’s go back into the coin shop now and create a way to buy these characters. So, now in the character section, we see two buttons we could use as a way to purchase characters. The single unlock button, and the multiple unlock button. The single will let you list each character separately, but let’s drag a multiple unlock button into our menu. Let’s resize this and make it bigger, so there is more area for our user to swipe.

By now, the Coin Shop should look like this:

For your Unlock Button image slot, drag this graphic in:
This is the graphic that you have to touch to unlock your character.
There are options in Multiple Unlock for the label. There are options here for sounds when you unlock a character, even ones you don’t have enough currency for and a graphic for a random unlock button which you could also set up with a character button. Just drag it in, and select unlock random character. Once you have them all, the random unlock character button will disappear from the menu.

So, there are a lot of neat things you can put in a coin shop. It doesn’t stop at just characters. Once you monetize your game with Ads by putting your account keys in the project settings and adjust your UIs to serve Ads, you can place a remove Ads button in the shop, but we will cover that whole process in another manual.

An option to buy characters gives players more motivation to collect coins, or better yet, pay with real money!
After you make your own game, it’s vital that you take the time to review and prepare it for exporting. In this lesson, we’ll further explain the settings options and explore the general tab section.

You’ll learn how to get your game ready for export and all the necessary information you need to fill out to get your app up and running in your chosen platform’s marketplace.

With Buildbox you can easily export your game to any platform including Android, Windows, iOS, Mac, Steam, Apple TV and more. Our game engine also automatically adjusts button and interface positions as needed for different devices so your game will look great on any device you choose to export it to.

In this chapter, you’ll learn how easy it is to fill out the necessary information, choose export target, and let Buildbox automatically export all files in code for compiling in the platform of your choice.
We are going to learn about exporting our game to both iOS and Android. Buildbox can export to multiple platforms, but before we can export, we need to add a few things in the project settings.

Let’s click on this button now.

We see three tabs in this window. General, where we would add our final specifics like an icon and our bundle id for the app store. The general tab has the finishing touches we need to pack this thing up. This image slot here is for your game’s icon. You can name your game here. We can see a controllers tab and the advertisements tab as well. Let’s go over these in more detail.
The advertisement tab is where you would add your different keys from any ad networks you have an account with. Here’s where you would select your target export option and paste your keys into the appropriate slots and set the priorities for how you want those ads served to your users, which we will cover in another manual.

The controller’s tab allows you to specify controller commands, pretty straightforward.
We set up an iOS export by naming the game and putting in our bundle id. Getting a bundle id takes an Apple developer account, and some setup, but we will cover that in another manual.

Once we have the necessary information to put in, we hit ok, and go up to file and choose our export target which is iOS. Buildbox will then export all the necessary files into a folder for us all ready for compiling in Xcode which we will cover in another manual.
And then if we wanted to export an Android version, we just bring up the settings window again, select the Google Play store as our target, and then we put in the necessary information.

After that, we go up to file and select export, choose Android, and Buildbox will prepare a folder with all the necessary files in it for compiling in the appropriate development tool, but we will cover that in another manual.
Chapter XI:
TIPS TRICKS AND HACKS

Now that you know how to make a game app without coding, we’re going to backtrack and reveal some of the best tips and hacks to get the most out of our software. We’ll revisit some of our previous lessons and show you a few new tricks to help take your game to that next-level.

In this last chapter, you’ll learn a clever tip to get your collision shapes precise. We’ll show you how to give your character an exploding death, so it looks like he shatters into tiny little versions of himself.

We’ll explore some of the other notable features in Buildbox like the Menu Jump and event observer and show you how you can integrate timers within your gameplay. You’ll also learn a secret tip about making secret worlds unrepeatable.

After you’ve completed the eleventh and final chapter of our manual, we encourage you to join our official forum to learn more about how to make your own game using Buildbox.
Let’s talk about precise collision shapes for our mountain enemies. Way back in chapter 3, when we went over collision shapes, we knew we weren’t exact with our shapes, and we mentioned making a composite object was our solution. Let’s get a blank scene ready, and let’s drag in different parts of our mountains.

The graphics supplied us with our mountain enemies, but it also supplied us with the enemies broken down into its components. So, let’s drag those in one at a time, and get their collision shapes exact.

Remember to keep in mind our forgiving collision shape philosophy. You can see the advantage here because now we can start to rotate them in new ways, even scaling them differently. And this gives us control over not only better collision shapes, but now we have full control for replacement, scaling, and rotation of each spike.
We can also bring in our enemy mountain as it is and let’s get our new shapes to mimic the original rendition.

Now, let’s give our character an exploding death, so he looks like he shatters into tiny little versions of himself. Just go into any world, and drag in an image of our character onto the object option of the drag and drop wheel. Name it “shatter” and delete it from the scene. We will only be using this for the Defeated Animation of the character.
Select our character in the Assets Panel, and delete the current defeated animation. Go to the character’s options and edit the defeated animation.

Pull our shatter object into the very center of the defeated animation which is position 0,0 and scale it down to 0.15 or any number for a size you think would be best. Set it to be a physics object and give it some movement with 500 in the velocity randomizers. Give it a spawner component with a very large spawner rate like 999. Refer to the settings at the right.

Duplicate the shatter objects and arrange them to form the shape and size of the character. That way, when the character dies, the shatter objects that spawn from the explosion give the illusion that they came from parts of the character.
You could go crazy with the exploding death animation, but keep in mind it has to be able to run on a device. So, make sure to always test your games on a device. Spawning a billion physics objects may give your iPhone or Android phone a hernia so, you might want to balance your animations and effects with playability.

Now, remember those extra worlds we have in the games mind map? Why don’t we put those to good use by creating a secret world for our player to discover?

Let’s create a world that has a lot of bumpers for fun and coins to grab for the shop. In fact, one that has so many coins to grab, that we don’t want the player to be able to play through it more than once, and exploit grabbing those coins. First, create a secret world with a ton of coins and bumpers.

Next, we’ll find a way so that once the world is discovered and played through, it can never be played through again.
The best world to put the portal leading into the secret world would be the Hard World. This way, it gives the players incentive to play on hard mode. To make this portal work, we will be using the Menu Jump feature.

With the Menu Jump feature, you can add a bonus or secret levels in your game simply by placing a Menu Jump logic piece into the scene. You can use it to end your game, take players to another world, or to an animated cut scene.

Go to the Assets Panel and grab a Menu Jump from the logic pieces. Place that Menu Jump with our marker and label it ‘coin grab’. Leave the settings as is. You can put in graphics on top of the Menu Jump to let players know of its existence. Perhaps a particle emitter? A portal graphic? A door? It’s your call.
Go back up into the Menu Editor and see our connection in our World 1 Hard node and let’s go ahead and grab and connect the World 1 Hard to the Coin Grab World node.

Now, here’s the trick. We want to be able to play that coin grab, but we need it to have its own world UI so we can tweak it just a bit. So, let’s duplicate the world UI using S and connect the Coin Grab world to that UI instead of the original.

We gave the Coin Grab World its own UI because we want to change its Event Observer. The Event Observer feature automatically monitors your game for specific events. With it, you can schedule a countdown or timer to be activated within your game. You can make players race against the clock to complete the level in time or kill all enemies.
Click on the Event Observer in the new UI. Change the game over observer to read 50 coins, making sure it is set to session. Session means the current play session, so you could set this to 5000 or whatever number fits your game, or you could make this world a certain length, and let them play the whole thing. It’s all up to you, but for demo purposes, 50 coins will work well.

Drag a new Event Observer into the UI and set the Event Type to “Second Play”. Set the name to “Played”.

This “second play” event type was originally designed for tutorial menus and tutorial worlds. The first time you play a game, the tutorials will appear and teach the player how to play your game, but when they come back for a second time the tutorials would get skipped. We are going to use it to make sure the world can never be played again using those same mechanics.
Go back into the Menu Editor and connect the event observers back to the World 1 Hard node.

If you test the game, you’ll see that we now go to the Coin Grab World once and then we can not return to the Coin Grab World twice. However, you can see it shows the world for just a moment while the observer is routing us back.

The best way to remove this unappealing feature is to insert a black image to cover both World UIs and set its autohid to 0.1 seconds. This gives that fade in look on every node change which makes the transition more polished.

Okay, let’s look at our final tip, trick, and hack – the timer trick. Let’s say when players went into that secret coin grab, that we only wanted to give them 10 seconds to grab as many coins as they can. This next part, our 100th page, is going to be a little complicated.
Go up into the Menu Editor and get into our original world node. Create a new action, just grab any old graphic, it’s never going to be seen in the game, and drag anything into the action option on the drag and drop wheel.

Rename it to “timer action”. Set the coin to 0 and points to 1, not showing either.

Then, delete the default animation of the timer action. Next, create a new object, just grab any old graphic, it’s never going to be seen in the game, and drag anything into the action option on the drag and drop wheel. Delete its default animation too then rename it “timer object”.

Select the timer action in the scene, and cut it to the clipboard with either ctrl X for PC users or command X for Mac users. Select the timer object in the Assets Panel and edit the default animation we deleted. Clicking in the animation editor, so it has focus, paste our invisible action in with ctrl V or command V, making sure we set its position to 0, 0.
Next, select our character from the Assets Panel, and edit its default animation. Drag in our timer object in at position 0, 0 and make sure to set it to decoration. Give it a high-speed negative y-axis take off, so it leaves the scene and gets deleted in the delete threshold once it spawns. Give it a spawn rate of 1 with spawn action set as permanent.

Go into our coin grab UI and duplicate our coin’s label and make it read timer. Duplicate the coin counter, and adjust it to keep score of the current world with our score type being coins, and the amount set to current.
Then, let’s change our Game Over event observer to keep track of the session points with a total of ten. So, now when we play hard mode and go into our coin grab after 10 seconds, we get kicked back to hard mode because we reached 10 seconds and times up.

Speaking of timers, our time together is up as well.

This is the final chapter, but we have a bonus chapter where we’ll give you tips to better polish your game.

I’ll miss you!
Bonus Chapter:
POLISHING YOUR GAME

There’s one more aspect of creating games that we haven’t covered yet, which is polishing your game. Virtually every successful game is polished in one way or another. In this special bonus video we’ll teach you exactly what polishing is and how to properly add it to your game.

We’ll reveal some of the late stage design decisions we made with GLTCH and provide some core tips that you can use to build your own game. We’ll cover several effective strategies to use when you’re adding the final touches to your app to make it appear more polished. You’ll also learn the steps necessary to make your game publisher ready.

Polishing your game is all about finding that perfect balance between minimal and simple, but sleek. Throughout this bonus video from the Make Your Own Game series, you’ll learn a faster way to make design decisions. Watch and follow along as we clean up the UI of GLTCH.

We’ll share some essential game design principles like elegance and tips that you can use to make your own game appear more minimal yet sleek. You’ll also learn the most important rule you need to follow to make your game publisher ready and much more.
Polish often means toning it down. Always go too much first, then dial it back. Find too much first because that’s usually obvious, then dial it back, and you’ll eventually come across ‘just right’. This is a much faster way to make design decisions. To polish a game and make it publisher worthy, we want minimal, and we want simple, but we want sleek. It needs to look simple, but attractive. Too simple can sometimes look unfinished, but sleek looks like you do simple on purpose and there’s a difference. You’ll see changes we made to GLTCH when we published it on the App Store.

We cleaned up the UI making sure we went simple but sleek. To enhance the glitch like theme, we added glitchy animation elements to the main menu components.

We also completely bypassed the hard mode UI and turbo mode UI and take people straight to those modes from the same main menu. This is a quicker less complicated experience. It allows a single main menu to access the entire game. A coin tracker was added to show available in-game currency for those unlock buttons.
The coin shop UI was completely removed, and the character shop was moved to the game over UI.

With the removal of the coin shop, it was directly incorporated into the game over UI. This brings an elegance to the game over interface.

Elegance in game design is when an element or group of elements can serve more than one purpose. We discussed similar elegant changes made to the main menu, having the mode shop be in one centralized interface.

The game over interface now serves as the character store as well. So, a coin counter was brought in to show how much in-game currency is available. Also, you can now see the distance being tracked on the game over UI as well.
On the gameplay UI, there were also changes made to the pause button using a backdrop and a white pause button. The button seems easier to see.

Also in the UI itself, the button is simply an overlay on those images, rather than using images within a button. This makes for an easier layout and certainly easier to press for the player.
We also replaced the graphics. The first thing we did was change our coins. Realizing that our graphics utilized shadows to give dimension, but not really 3D, we felt the use of a three-dimensional coin might be an aesthetic mismatch.

After further study, the coins were changed to be round like the character, and not have any sharp edges like the enemies of the game.

And even though that’s a late stage design decision, it’s one that can be made easily and quickly inside of Buildbox.
We also thought that when the character picked up a coin, showing the coin amount in the traditional way didn’t quite look right at the speed this game takes place. So, the decision was made to incorporate visual feedback of the amount to be made part of the actions animation.

There was also an info node added to the game, with a button in the main menu, showing credits for the game.
Let’s talk about the coin grab. Recent changes in the newer versions of Buildbox have presented an opportunity to show you another option when designing special entrances. The second play observer has recently been renamed – redirect and has been given some flexibility in its options.

Redirect now has options to customize how many plays before the UI redirects the player. The coin grab was modified to use a Menu Jump instead of redirect. So, we can show you that play cooldown can also be used in creative ways. Now, the scene with the entrance to the coin grab can only be accessed after 50 player deaths once played.
Overall, the changes bring a simple, sleek elegance to the flow of the experience. The Buildbox team has a rule that a game isn’t publisher ready unless it has at least 100 scenes available in a game where scenes are randomized. We have supplied you with the final version at

www.buildbox.com/make-your-own-game

Feel free to fill this game out with as many scenes as you can. Building out a minimum of 100 scenes will give you two very important things:

1. Important experience designing levels
2. You can make this your own game

You are now ready!

Our sincere hope is that this left you with enough information to find success making games with Buildbox.

Thank you so much!

I hope by now you have joined us in the Buildbox community forums.

Stop by the official announcements forum and say hi… I’ll see you there!