



LEVEL UP

THE EDUCATOR'S EDITION

MAKE GAMES. TEACH CODING

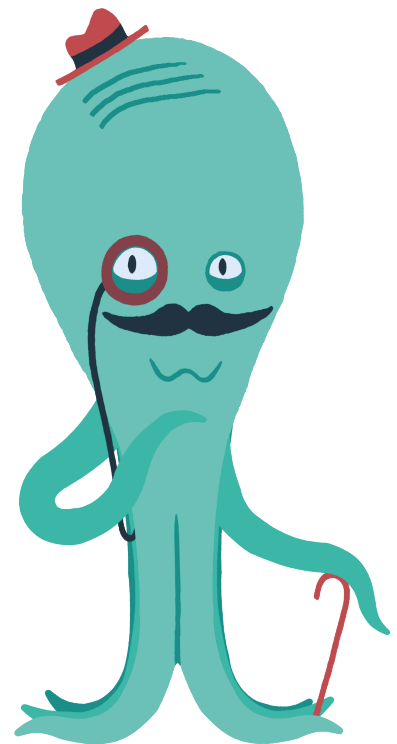


TABLE OF CONTENTS

WELCOME TO LEVEL UP - EDUCATION EDITION	7
INTRODUCTION	8
<hr/>	
OPENING AND USING C3	10
<hr/>	
SYSTEM REQUIREMENTS.....	11
Supported Browsers	11
Supported Operating Systems	11
WebGL Support	11
<hr/>	
USING AN ACCOUNT	12
Access Codes	12
Setting up a Classroom.....	16
<hr/>	
CONSTRUCT'S EVENT SYSTEM.....	17
What Is an Event?	17
How do Events Work?	18
And / Or Blocks	18
Event Ordering	19
Variables	19
Global Variables	20
Local Variables	20
Instance Variables	21
Functions	21
Parameters	23
Return Values.....	24
<hr/>	
COMMON CONVENTIONS IN CONSTRUCT 3.....	25

Common units.....	25
Zero-based indexing	25
Ranges.....	25
<hr/>	
OVERVIEW	28
Changing the Theme	29
Simplified User Interface	29
<hr/>	
START PAGE	31
Starting a New Project	31
Opening Existing Projects	31
Resource Links.....	32
The Example Browser	32
<hr/>	
MAIN MENU	34
<hr/>	
MAIN TOOLBAR	37
<hr/>	
VIEWS.....	38
Layout View.....	38
Adding, Modifying and Deleting Ob- jects.....	39
Scrolling and Zooming	40
Selection Wrapping	41
Containers	42
Other Considerations.....	42
Event Sheet	43
Creating Events.....	43
Modifying Events	43
Comments	44

Scrolling and Scale.....	44	The Debugger.....	60
Finding In Events	45	File Editors.....	61
<hr/>		The Array Editor	61
BARS	46	Dictionary Editor	62
Project.....	46	Scripting.....	63
Organising Projects	46	<hr/>	
Managing Items in the Project.....	47	CLOUD SAVE	66
Importing Files	47	<hr/>	
Properties	47	USING LOCAL FILES.....	67
Special Features for Number Value		<hr/>	
Properties.....	48	USING PROJECT FOLDERS	67
Layers.....	48	<hr/>	
The Layers List.....	49	SAVE TO LOCAL BROWSER.....	67
Tilemap	50	<hr/>	
Toolbar tools	51	SHARING YOUR PROJECTS	68
Editing tile collision polygons	52	<hr/>	
Bulk editing	52	SHARING WITH SUBSCRIPTION ADMIN	68
<hr/>		<hr/>	
ANIMATIONS EDITOR	53	PART 1 - GETTING STARTED	71
Colour Palette	53	Create a New Project	71
Image Tools.....	53	Changing Layout Properties.....	71
Drawing Tools	54	Adding Your First Object.....	72
The Frames Pane	55	<hr/>	
The Animations and Properties Panes..	56	PART 2 - CREATING PLATFORMS	75
The Image Points Pane	56	Using a Tilemap for Platforms.....	75
Editing Image Points.....	57	<hr/>	
Drag-and-Drop Importing of Files.....	57	PART 3 - ADDING THE PLAYER.....	78
<hr/>		Adding the Player Sprite	78
C3 ON MOBILE.....	58	Setting Up Origin and Image Points.....	78
Accessing Bars.....	58	Animations and their Properties	79
Changing UI Mode.....	59	<hr/>	
<hr/>		PART 4 - ADDING BEHAVIORS	80
ADVANCED FEATURES	60	About Behaviors.....	80
		Creating a Helper Sprite.....	80

The Behaviors Dialog	81	PREVIEW SETTINGS.....	105
PART 5 - YOUR FIRST EVENTS	83	EXPORTING FOR WEB	107
Keeping our Sprites Together.....	83	Publishing to the Construct Arcade	108
PART 6 - IMPROVING PLAYER ANIMATIONS	86	EXPORTING FOR MOBILE	109
Mirroring Sprites.....	86	EXPORTING FOR DESKTOP	110
Jumping and Idling Animations	87	SELECTION AND CONDITIONAL LOGIC	113
Setting up Events for Multiple Anima- tions	87	ITERATIVE TESTING AND DEBUGGING..	113
PART 7 - CREATING AN ENEMY.....	90	VARIABLES AND DATA TYPES.....	113
Adding the Enemy.....	90	LICENSING AND COPYRIGHT	113
Creating Enemy Movement.....	92	LEVEL DESIGN.....	114
Interactions with the Player.....	94	ORGANISATIONAL SKILLS.....	114
PART 8 - ADDING COLLECTIBLES	96	WHAT NEXT?	115
Create the Collectible Item	96	USEFUL RESOURCES.....	115
Collecting and Scoring.....	97	THANK YOU!.....	117
MY BARS HAVE DISAPPEARED!	100		
MY CHARACTER ISN'T MOVING PROPERLY!.....	100		
I CAN'T ADD THIS OBJECT!.....	101		
I REFRESHED/CLOSED THE BROWSER!	101		
PREVIEW TYPES	103		
Debug Layout.....	103		
Preview Project.....	103		
Remote Preview	103		

WELCOME TO LEVEL UP - EDUCATION EDITION

So, you've decided to use Construct 3 in your classroom, or perhaps you've been told that this shiny new tool is yours to use. Either way, let me take this opportunity to thank you for using Construct and welcome you to this eBook that will accompany you on your first steps with the program.

The first chapters will go through information about Construct 3, the program's interface and explain what various things do. Following that, we'll show you how to make a simple game which is a great starting point for you and your students, and then we'll look at Construct's various export options.

This book is based on existing Construct 3 documentation including the **manual** and **tutorials** which are available on our website - www.construct.net.

I hope you enjoy your journey with Construct, and I hope that your students love the program too.

Laura Donaghy
Editor



INTRODUCTION

Once upon a time, the world of game development was only open to those skilled in the ways of programming. People looked on as these heroes created the early classics that we came to adore.

Fast forward to today and the landscape has changed dramatically. Development tools are now more readily available than ever and range from full blown code-writing tools to simpler (but still powerful) block-based tools. Anyone can turn on their computer (or even their phone or tablet these days) and start working away on the game of their dreams.

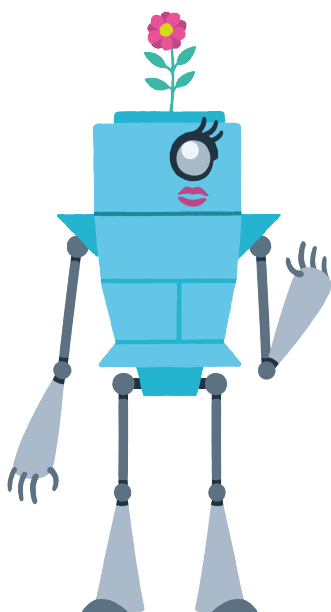
The video games industry is growing at an incredible rate, and it's brilliant how anyone can become a part of it. For example, according to figures from UK Interactive Entertainment (UKIE), in 2022, the consumer games market in the UK alone was valued at a whopping £7.16 billion, with over 2500 games businesses – that's a lot of jobs to fill. Computer science, coding and game design courses are likely to become increasingly important as we move further into a highly technological world, and we think Construct 3 is well placed to help students learn vital skills.

Here at Scirra, we're proud to be part of that growing industry, and to have been a part of it since 2011. The Construct engine started as an open-source project called Construct Classic. Classic evolved into a commercial product – Construct 2. And in 2017, after being downloaded more than 4 million times worldwide, the product evolved again into Construct 3.

But what is Construct 3?

Construct 3 is a browser-based, 2D game engine that allows you to create games without having to use a single line of traditional code. The core toolkit starts with Construct's unique event block system. Combine that with built-in behaviors and you can have a game up and running in minutes. It's a great place to start learning about programming and game design, and it can help your students really unleash their creativity. It's cliché, but the only limit is their imagination!

If you've got higher level students, or some that are really enjoying getting stuck in, they can supplement the event block system with JavaScript code or ditch the events completely and write purely in JavaScript. This enables students to stretch their knowledge ever further with a professional standard coding language, all in the same program.





GETTING TO KNOW CONSTRUCT 3

OPENING AND USING C3	10
SYSTEM REQUIREMENTS	11
Supported Browsers	11
Supported Operating Systems	11
WebGL Support	11
USING AN ACCOUNT	12
Access Codes	12
Setting up a Classroom.....	16
CONSTRUCT'S EVENT SYSTEM	17
What Is an Event?	17
How do Events Work?	18
And / Or Blocks	18
Event Ordering	19
Variables	19
Global Variables	20
Local Variables	20
Instance Variables	21
Functions	21
Parameters	23
Return Values.....	24
COMMON CONVENTIONS IN CONSTRUCT 3	25
Common units.....	25
Zero-based indexing	25
Ranges.....	25



This segment will take you through Construct 3 itself, starting with how to open and use the program, and then moving on to explain what some of the core features do. After all, it's helpful to know what something does before you begin using it in earnest!

OPENING AND USING C3

Construct 3 runs in your browser, so there's nothing to download or install!

To open the program, visit editor.construct.net in your browser on a supported system and Construct 3 will start. The browser-based nature of the program makes it easy to switch between devices, use public computer terminals (even with strictly limited access), or painlessly deploy Construct 3 across a computer lab or classroom.

Despite this, Construct 3 works offline - you don't have to always have an active Internet connection, you only need to be online the first time you load Construct 3. When you first open Construct 3, it will download in the background meaning it can be used offline later. After a while, you should see a notification in the corner indicating that this download is complete, and the program is ready to work offline.



Note that if you purchase a subscription and work offline, you must start Construct 3 while connected to the Internet at least once every 7 days to re-validate your subscription. However, if you're using the free edition, you can use Construct 3 offline permanently.

Sometimes, **Install as app** will appear in Construct's main menu if the option is available.

Clicking this menu option will install Construct 3 as an app on your device. Alternatively, if you are using Chrome, you can do this manually from Chrome's menu with the following steps:

On Windows and MacOS choose **Install Construct 3** from the menu

On Linux, choose **More tools → Create shortcut**

On Chrome OS, choose **More tools → Add to shelf**

On Android, choose **Add to Home screen**

This is a great way to reach Construct more easily and saves space on your screen since it hides the browser address bar and tabs. It's also a much more convenient way of using Construct offline.

The program automatically stays up to date, so you don't need to worry about a thing. In Construct's settings, you can opt in for notifications to update to beta releases, though this is not recommended for classroom use as typically, beta builds are less thoroughly tested than stable builds.

Construct will notify you when there's a new version available and in the case of beta releases it will ask if you want to update. The Construct website also provides a list of all releases with detailed information about changes in each update and provides links to run older versions in case there's a problem with an update.

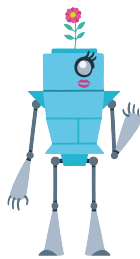
You can check which version of Construct you're currently using by opening the main menu and selecting About.

SYSTEM REQUIREMENTS

SUPPORTED BROWSERS

Construct 3 can run in the following browsers:

- Google Chrome 57+
- Other browsers that use the Chrome browser engine (Chromium), such as Opera and Yandex, providing they are updated to Chromium 57+
- Firefox 55+
- Safari 11+
- Microsoft Edge 16+

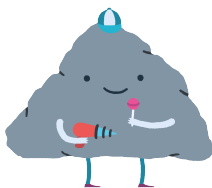


Please note that Construct 3 does not support Internet Explorer. However, in Windows 10 Microsoft replaced Internet Explorer with the Edge browser, which is supported from version 16+.

SUPPORTED OPERATING SYSTEMS

Construct 3 can work on the following operating systems:

- Windows: Windows 7, Windows 8, Windows 8.1, Windows 10 or newer
- Mac: OS X / macOS 10.9 or newer
- Linux: 64-bit Ubuntu 14.04+, Debian 8+, openSUSE 13.3+, or Fedora Linux 24+
- Chrome OS: Any Chrome OS device updated to v57+
- Android: Any Android 5.0+ device with at least 1GB RAM
- iOS: Any iOS 11+ device



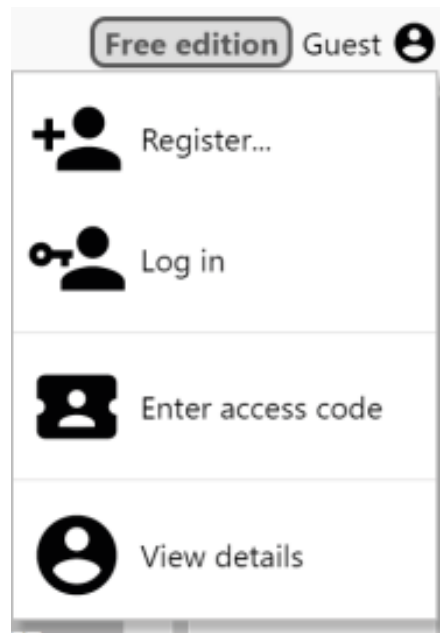
Note that Safari users on Mac must use Safari 11+, which is only supported on macOS El Capitan (10.11) or newer. Users on older versions of macOS/OS X can use Chrome or Firefox.

WEBGL SUPPORT

Construct 3 requires the browser to support WebGL, which is a modern high-performance graphics technology for browsers. Almost all modern devices support WebGL. However, if you see a message about WebGL not being supported, try installing any available system updates, and check your graphics drivers are up to date.

USING AN ACCOUNT

When you first start Construct 3, you'll use it as a **Guest**. This means you are not logged to an account. Construct shows your account status near the top-right corner. You can click this "badge" to show a menu with some account options.



In this menu, you can register a new account, log in to an existing account or enter an access code (more on that shortly.) You can also select View Details to open a dialog displaying more information about your account.

Until you log into an account with an active subscription, or input an active access code, Construct 3 works in a limited **Free edition** mode, as indicated by the "Free Edition" label on the account badge. The complete list of limitations of the Free edition are listed on the Construct 3 plans comparison page and cover several aspects of the software including number of events, number of layers or effects and the number of lines of JavaScript a project can use.

There are also some differences in the free edition limits depending on whether you're using Construct as a guest, or with an account:

- **Guests** have an event limit of 25
- **Creating an account** increases the event limit to 40
- **Verifying your email** increases the event limit further to 50

If you have a Construct 3 subscription, you must be **logged in** with the same account you purchased with or be using an Access Code to make use of the full features of Construct 3. The **Free edition** label next to your account will disappear to indicate you have an active subscription and no longer have the Free edition limits imposed.

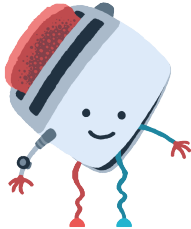
ACCESS CODES

Access Codes have already been mentioned within this chapter, but what are they?

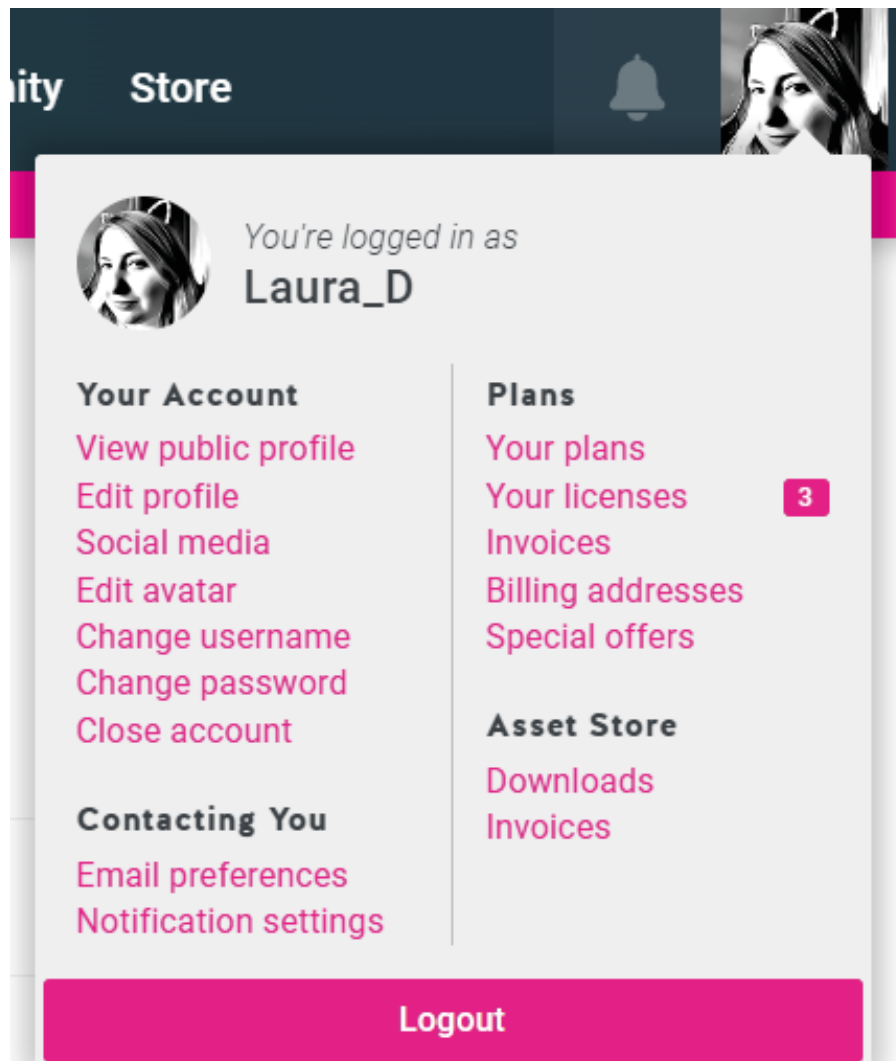
An access code is a time-limited code generated by an education plan holder which allows students to use the full version of Construct 3 without having to create an

account. This means that Scirra Ltd holds no data from the students using the access code, only the account data of the admin controlling the education plan.

To generate an access code, you'll need to visit the Subscription Management section of your account, which is found on our main website. When you go to www.construct.net, you should see either your profile picture in the top right corner of the page, or an option to Register or Log In. If you haven't logged in, make sure you do so that you can access the drop-down menu that appears when you hover over your profile image.



Please bear in mind that Construct 3 and [construct.net](http://www.construct.net) use the same accounts but different login systems. Being logged into one won't automatically log you into the other.



From this dropdown menu, select Your Plans which will bring you to a dashboard of your currently active Construct plans. If you select Manage Plan on your education plan, you'll be taken to the dashboard for that specific plan.

You can do multiple things from this dashboard including edit your profile information, manage licenses directly assigned to students and check any projects that have been shared with you via the 'Share with Admin' feature in C3. However, for this section of the book we're only focusing on **Access Codes**. Click on the Access Codes link under the Overview menu on the left-hand side of the page. This takes you to the Access Code Management screen.

Here, you can see options for creating codes, your current Active and Expired codes and the ability to restrict IP Addresses if you find your access codes are being misused. The default view is to create a new Access Code.

The drop-down menu allows you to select the number of seats from your available license pool to use for Access Codes.

As an example, let's say we need a code for a class of twenty students – we'd need to pick twenty as the number of seats required. And we're going to give the class a

new code each week, so we can choose to specify the end date as seven days from when the code was created.

You don't have to have an end date for the code, they can be valid indefinitely, or until you manually set them to expire.

Once you've set the parameters for your Access Codes, click create and you'll be transferred to the Active Codes page where you can see all of your current active codes.

Access code has been created!

Status	Code	Ends	Seats	Shared Projects
Running End	8 W N T E C 2 2 Created on 27 Sep, 2023	Ends in 30 days 27 Oct, 2023 at 15:47	5	0

0/5 live sessions [Edit Code](#)

Now you can share the code with the students you want to give access to. It'll also show you when the code is due to expire and how many of your seats are currently in use (live sessions.) If you want to end your code early, you can use the End button beneath the Running Status on the left-hand side.

The Edit Code button takes you to a new screen where you can adjust various parameters for the selected code. You can add or remove seats, change the end date, or set the code to run indefinitely. You can also end codes from here.

Your Construct 3 Plan

EDIT ACCESS CODE

THIS ACCESS CODE IS CURRENTLY LIVE!
This code is due to end in 4 hours, 57 mins on 3 Oct, 2023 at 13:07

5 Seats
E772W9A2
Echo Seven Seven Two Whiskey Nine Alpha Two

Extend Code
Add days and hours time to this access code.
If you do not extend the duration of this access code it will expire in 4 hours, 57 mins [EXTEND](#)

Set End Date
End this access code in days and hours
This access code is due to end in 4 hours, 57 mins [SET](#)

End Now

OVERVIEW
Overview
Your Organisation
Your Seats 20
Access Codes 1
Create Code
Active 1
Expired 4
IP Restrictions
Sub Users 0
Activity Logs
Access Logs

SHARED PROJECTS
Shared Projects 0

PAYMENTS
Invoices

OTHER
Extend Plan
Get Help

[LAUNCH CONSTRUCT 3](#)

Once you have created an access code, you can share it with your class. Students should then start Construct 3, choose the Enter access code option in the Account menu type in the access code, and click OK. This will fill one of the available seats within the access code and unlock the full version Construct 3. Once the access code expires, users will be notified, and Construct will revert to the Free edition.

SETTING UP A CLASSROOM

When setting up a classroom, there are multiple ways to get your students using Construct 3. One of the advantages of Construct is the flexibility it offers in how it can be used in the classroom – it's easy to tailor to your needs!

If you're only using the program for a short period of time, say for a summer camp or short computer science module, the free edition is probably enough for what you need. In which case, simply visit editor.construct.net and you can get started. If you want a bit more use out of the free edition, you can use accounts, but this will require each student to set up an account with us.

If you have an education plan, then your students can access the full version of Construct either using their own accounts (or accounts you've set up for them), by creating access codes, or a combination of the two.

An education plan is made up of a pool of licenses or seats and when you generate an access code, or assign a license to a student account, it is taken from that pool.

For example, if a teacher had a plan with 30 seats, one would be taken for their admin account leaving 29 in the pool. If this teacher had a particularly keen student, they could assign a license directly to an account for that student – this would leave 28 in the pool. The teacher could then create 28 access codes from the remaining seats. However, should an extra student come along and try to use one of those 28 codes, an existing student would be logged out to make room for the new student.

CONSTRUCT'S EVENT SYSTEM

One of Construct's biggest assets is its unique Event System, familiar enough to those who've used block-based coding before but structured more like a traditional programming language which helps to teach those core principles and computational thinking. The conditional logic also means it's quite easy to pick up. We briefly mentioned Events when discussing the difference between the Free and Paid editions of Construct 3, but what are Events and what do they do?

WHAT IS AN EVENT?

Events are designed to be easily readable and to intuitively "just work". However, they have specific, well-defined ways of operating.

An event is made up of two parts, **Conditions** and **Actions**. Conditions test if certain criteria are met, e.g. "Is spacebar down?". If all conditions are met, the event's **actions** are run, e.g. "Create a bullet object". After the actions have run, any sub-events are also run - these can then test more conditions, then run more actions, then more sub-events, and so on. Using this system, we can build sophisticated logic for our games and apps.

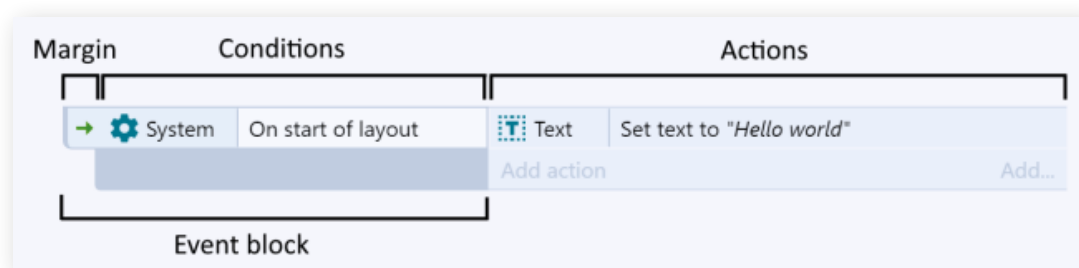
In short, an event basically runs like this:

Are all conditions met?

- **Yes:** run all the event's actions.
- **No:** go to next event (not including any sub-events).

That's a bit of an oversimplification. Construct provides a lot of event features to cover lots of different things you might need to do. However, for now, that's a good way to think about it.

The following image illustrates the key parts of an event as you'll see it in the Event Sheet.



These sections are:

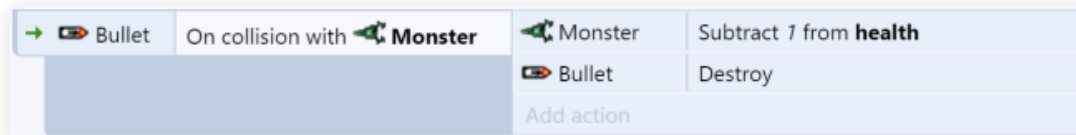
1. The event block, which contains the conditions. Notice the margin to the left of the condition which allows you to select the entire event.
2. The conditions, which are each listed inside the event block.
3. The actions, which are listed to the right of the event block.

Conditions and actions can be selected by clicking on them. The entire event can be selected (which also selects all its conditions and actions) by clicking the event margin, or the bottom part of the event block. The event margin can also be right-clicked to access a menu allowing things like adding conditions or sub-events.

HOW DO EVENTS WORK?

Now you know what makes up an event, let's go into a bit more detail about how they work - beyond the "are conditions met? Then do this" approach.

Events work by filtering specific instances that meet some conditions. The actions then run for *those instances only*. For example, consider the following event:



In this example, when a **Bullet** collides with a **Monster** the event condition is met. The specific instances of **Bullet** and **Monster** that collided in the game are "picked" by the event. Actions only run on the "picked" instances. If there are other instances of **Bullet** and **Monster** in the layout, they won't be affected by the **Subtract 1 from health** and **Destroy** actions. It would be very difficult to make good games if every bullet hurt every monster!

After an event ends, the next event begins from scratch. Its conditions will start picking from all instances again.

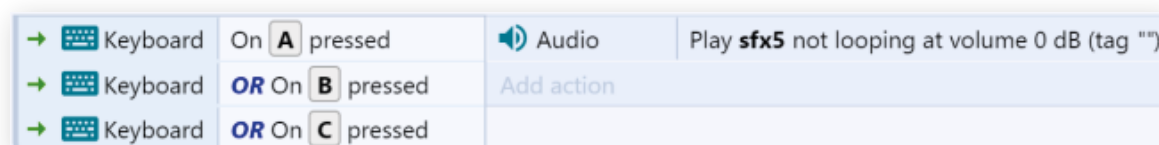
On the other hand, sub-events (which appear indented) carry on from where its parent event left off. A sub-event will further filter the instances left over by the event that came before it.

If an event has two sub-events, they both pick from the same set of instances the parent left behind - the second sub-event is not affected by the first. In other words, events at the same indentation level always pick from the same set of instances, and events at a lower indentation level are always working with the instances handed down from above.

AND / OR BLOCKS

As mentioned before, all conditions must be met for an event to run. This is called a '**Logical AND**', because "condition 1 **AND** condition 2 **AND** condition 3..." must be true. However, you can change an event to run when any condition is true. This is called a '**Logical OR**', because the event will run if "condition 1 **OR** condition 2 **OR** condition 3..." are true.

Normally, event blocks work as **AND** blocks. To make an **OR** block, right-click the block and select Make **OR** block. It will then display with an **OR** between each condition, as shown below.



Be aware that because **OR** blocks run if any condition is true, it's possible the event will still run if some conditions were false and did not pick any instances. In

this case, the actions will still run, but possibly with zero instances picked for any objects where no instances met the condition. If any actions are run for objects with no instances picked, nothing happens.

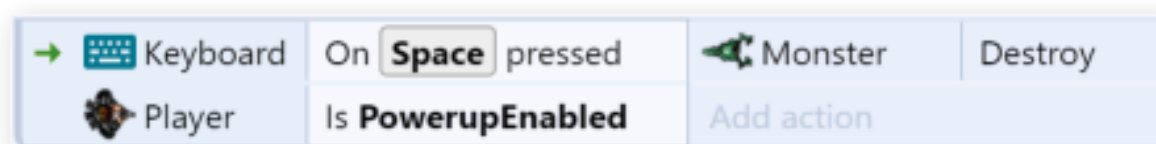
You can combine the block types by using sub-events. This allows you to build up more advanced logic, such as an **OR** block followed by an **AND** block.

EVENT ORDERING

The order of events is important. Every event is checked once per tick (about 60 times a second on most computers), and they are run from top to bottom in the event sheet. The screen is drawn once every event has been run, then the process starts again. This means if one event does something and the next event undoes it, you'll never see that anything happened.

The same applies within events: conditions are checked from top to bottom, and the actions run from top to bottom.

However, triggers are an exception. Notice that there is a green arrow to the left of **Keyboard: On Space pressed** in the following example:



This indicates the event is triggered. Rather than running once per tick, this event simply runs (or "fires") upon something happening. In this case, the event runs when the user hits the Spacebar key on the keyboard. It is never checked any other time.

Since triggers run upon an event happening, they aren't checked in top-to-bottom order like other events. This means the ordering of triggers relative to other events is not important (except relative to other triggers of the same type since triggers still fire top-to-bottom).

There can only be one trigger in an event because two triggers cannot fire simultaneously. However, multiple triggers can be placed in **OR** blocks.

VARIABLES

Event variables are number or text values which are either global to the whole project or local to a range of events. They are modified using the Event Variable dialog. To add an event variable, right-click on an event, another variable, or an empty space in the event sheet, and select **Add global variable** or **Add local variable**, or press the V keyboard shortcut. Variables at the root level of the event sheet (not indented beneath anything else) become global variables, whereas variables in groups or sub-events become local variables.

Event variables are modified with the system actions in the Global & Local Variables category. They can be retrieved by simply using their name in expressions.

Global Variables

Global variables show a globe icon. They are always at the top level of an event sheet - they are not sub-events or inside any groups.

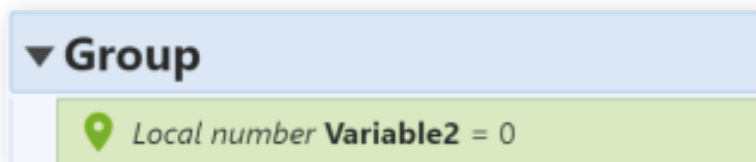


Global variables store their values between layouts. Events in any layout can access any global variable, even if it was created in a different event sheet that is not included.

Global variables can be moved to another event sheet by cutting and pasting them. After being cut, references to the global variable will disappear because it has been removed; this is normal and nothing to worry about. When you paste the global variable, the references that disappeared will reappear again. Alternatively, you can right-click the global variable and select Move to event sheet.

Local Variables

Local variables are variables placed nested under other events, or inside a group. They also show a different icon to global variables.



The main difference between global and local variables is that local variables can only be accessed in their scope - the scope being its level of sub-events. All other events at the same level of indentation, or lower levels, can access the local variable. Events above it (less indented) cannot access the local variable.

For example, if an event variable is in a group of events, it becomes a local variable. Then, it will only appear as an option for a variable in events inside that group. In other groups or in other event sheets it does not appear at all and cannot be accessed. This makes the variable local to the scope in which it is placed.

Local variables are convenient for temporarily holding variables over a short range of events, such as to calculate an average value (where a temporary sum variable may be necessary). It also helps keep the project simple, since it prevents the need to create more global variables, which appear everywhere in the project even if they are not needed everywhere.

The scope of local variables is designed to mimic how the scope of variables works in real programming languages.

By default, local variables reset to their initial value whenever entering their scope (usually every tick), like local variables in programming languages. If the variable is marked static in the Event Variable dialog it will persist its value permanently, like a global variable.

Both global and local variables can be marked constant. This makes them read-only: they can be retrieved and compared, but not changed.

Instance Variables

While not strictly part of the event system as they are part of objects, it's worth mentioning Instance Variables in this section.

Instance Variables are added to object types but store numbers, text or booleans (on/off flags) individually for each instance. This makes them ideal for things like health counters in a game since each instance tracks its own value. Instance variables are added to object types with the Object Instance Variables dialog, and the initial values for each instance can be set from the Properties Bar.

Instance variables can also be used to help control instances independently of each other. For example, a Boolean instance variable could be used to determine if an enemy is hunting down the player (true) or running away (false). If instances all have different values, the condition Is boolean instance variable set can be used to apply actions to enemies hunting down the player. Inverting the condition (picking instances with the value being false) can then be used to apply actions to enemies running away. The result is multiple instances of the same object type acting independently: some chasing and others running away. This is a simple example - much more complex methods can be made using multiple instance variables. In other words, an instance's state can be controlled using instance variables.






FUNCTIONS

Functions are special kinds of events that can be called from actions. They are designed to be analogous to functions in real programming languages. Using functions can help you organise events and avoid having to duplicate groups of actions or events.

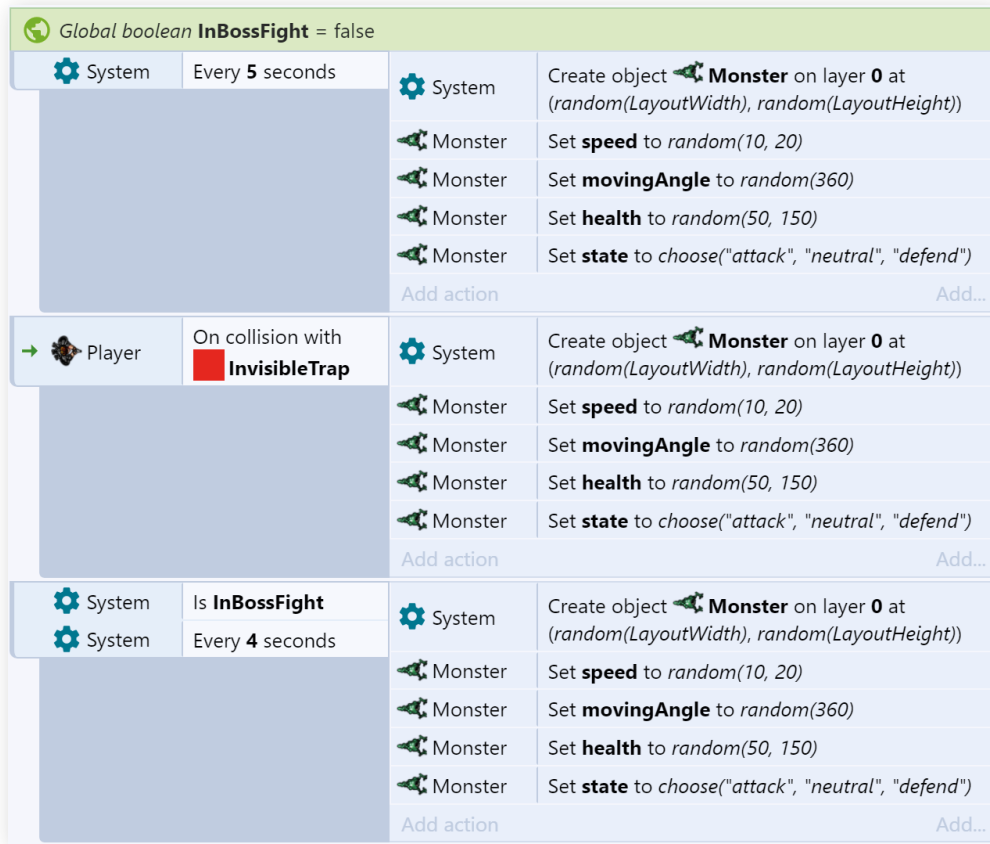
You can add conditions, actions, and sub-events to functions, just like you can with normal events. However, functions do not run unless you call them in an action.

A good example of using functions is to eliminate repeated sets of actions or events.

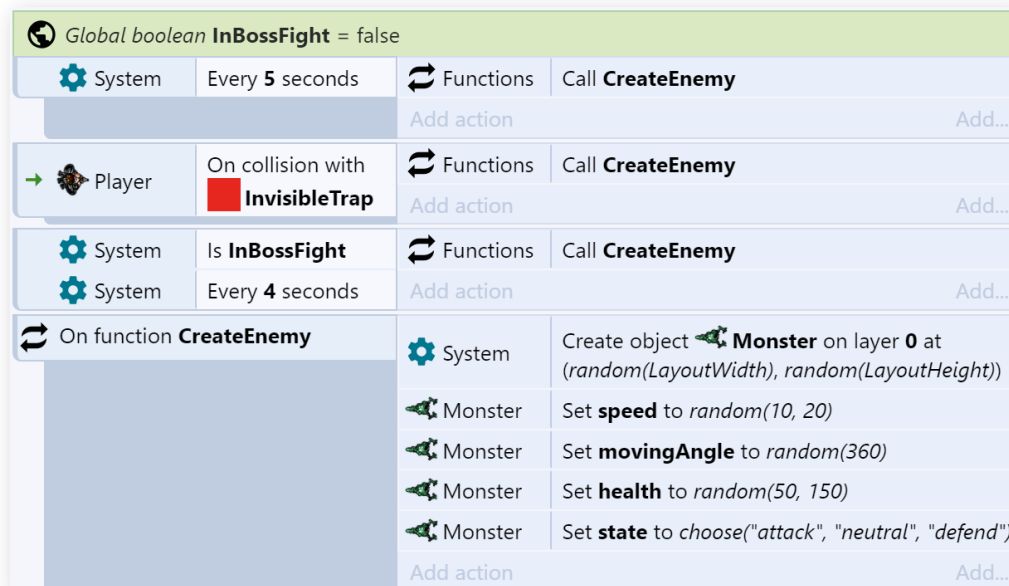
For example, suppose you create an enemy with random properties every 5 seconds using this event:

System	Every 5 seconds
System	Create object  Monster on layer 0 at $(\text{random}(\text{LayoutWidth}), \text{random}(\text{LayoutHeight}))$
	Set speed to $\text{random}(10, 20)$
	Set movingAngle to $\text{random}(360)$
	Set health to $\text{random}(50, 150)$
	Set state to $\text{choose}(\text{"attack"}, \text{"neutral"}, \text{"defend"})$
<div style="display: flex; justify-content: space-between;"> Add action Add... </div>	

Suppose there are two other events where you want to create an enemy the exact same way: one when a player walks into a trap, and another one every 4 seconds when in a boss fight. Without functions, you may have to copy-and-paste the same actions multiple times, like this:



Notice this is becoming inconvenient. There may be times you need to repeat the actions in even more places. If you want to make a change, you then have to find every place you repeated the actions and repeat the change. We can remove the repetition using functions. By creating a CreateEnemy function which has the repeated actions, we can replace all the repeated actions with function calls, like this:



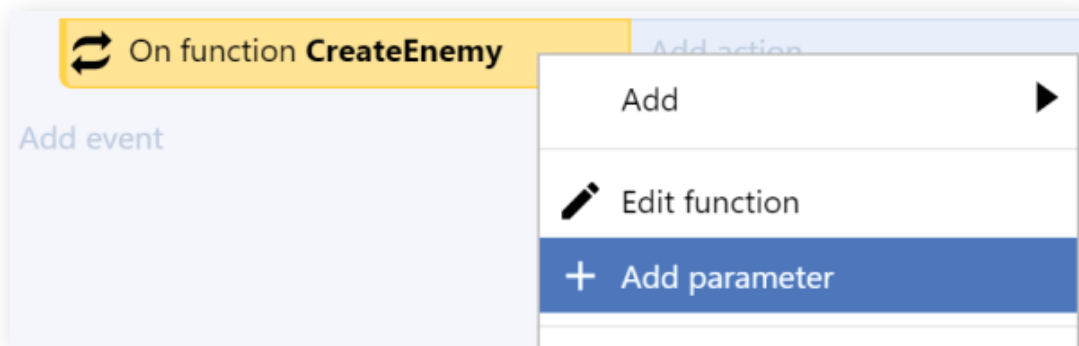
This works identically to the previous events but is much shorter and more convenient. We can call the CreateEnemy function anywhere in our events we want to create an enemy, and it uses the same set of actions in the corresponding On function event.

It is often useful to split many parts of your events in to functions like this, so they can be conveniently re-used across event sheets.

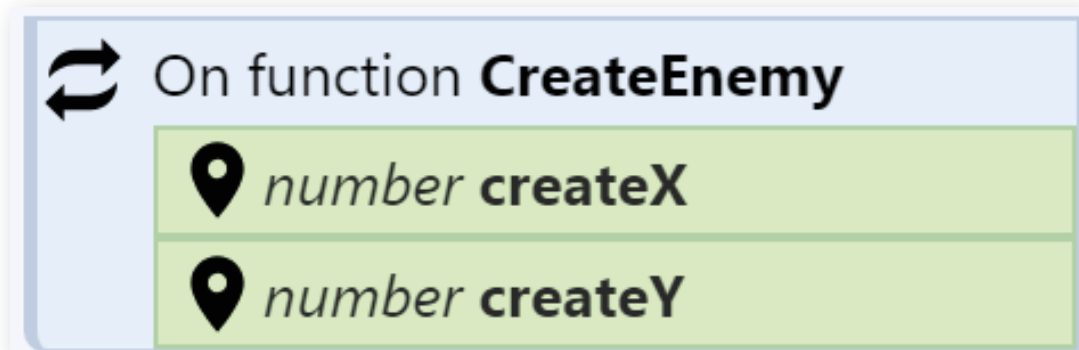
Parameters

When calling a function, you can also pass parameters. These are simply numbers or strings that are made available to the function. For example, the CreateEnemy function from the previous example could be modified to take two parameters: the X and the Y co-ordinates at which to create the enemy. This helps functions to be made more general purpose by using extra information from the action calling the function.

To add a parameter to a function, use the Add parameter menu option when right-clicking the function. (Note you need to right-click on the header or margin, since if you right-click a condition, it will show a menu for the condition instead.)



When you select this the Add function parameter dialog appears for you to fill in details about the parameter, including its name, description, and type. Parameters appear similar to local variables, but inside the function block.



Parameters work very similarly to local variables - you can use them in expressions, compare them, and set them just like any other kind of local variable. Like local variables they are limited in scope to just the function event and its sub-events.

Now when you call the function, you can also provide the parameters. Notice the name and description you set for the parameters are used. These appear like parameters for any other action, but they will set the values of the parameters when calling the function.

Return Values

Functions can also return a result. For example, a factorial function could calculate the mathematical result and return it.

By default, functions have a return type of None, meaning they don't return any value. This also means they are used as actions. However, if you set a return type of Number, String or Any, the function returns a value. This also means it is used as an expression instead, so it won't appear as an action.

A function can set its return value using the Set return value action in the built-in Functions object. It can then be called using it as an expression, such as:

Functions.MyFunction

Parameters can also be added in parentheses, e.g.:

Functions.MyFunction(1, 2, 3)

The expression returns the value set by the Set return value action in the function call.

Functions which return a value will also appear in the Expressions dictionary, also show up in autocomplete, and show call tips when entering parameters, just like other expressions. In summary, while functions with no return type are essentially custom actions, functions with a return type are essentially custom expressions.

COMMON CONVENTIONS IN CONSTRUCT 3

There are some things about Construct 3 that are worth knowing before you dive in, but they don't fit into the category of UI or how to use it. These common conventions will be outlined here.

COMMON UNITS

In Construct, sometimes you need to enter values such as angles, speeds, or sizes. For consistency these always use the same units, except where noted by descriptions or tips shown in the editor.

- **Positions** are in pixels. The origin (0,0) is at the top-left of the layout, and the Y axis increments downwards (as is often the case with game engines).
- **Sizes** are in pixels.
- **Angles** are in degrees. 0 degrees faces right and increments clockwise.
- **Times** are in seconds.
- **Speeds** are in pixels per second.
- **Accelerations** are in pixels per second per second.

ZERO-BASED INDEXING

To be consistent with programming languages, all features of Construct using a number of an item in a list (indices) start from 0 instead of 1. So, where an everyday list would be numbered 1, 2, 3... any lists in Construct will be numbered 0, 1, 2... to remain in keeping with programming languages.

RANGES

Sometimes the documentation will refer to ranges of valid values. These are in square brackets for an inclusive range, such as $[0, 1]$ meaning any value between 0 and 1, including both 0 and 1. For example valid values in this range are 0, 0.5, and 1. A round bracket indicates a non-inclusive boundary of the range, such as $[0, 1)$ meaning any value between 0 and 1, including 0 but not including 1. For example, valid values in this range are 0, 0.5 and 0.999, but not 1.

THE USER INTERFACE

OVERVIEW	28
Changing the Theme	29
Simplified User Interface	29

START PAGE	31
Starting a New Project	31
Opening Existing Projects	31
Resource Links	32
The Example Browser	32

MAIN MENU	34
------------------------	-----------

MAIN TOOLBAR	37
---------------------------	-----------

VIEWS	38
Layout View	38
Adding, Modifying and Deleting Objects	39
Scrolling and Zooming	40
Selection Wrapping	41
Containers	42
Other Considerations	42
Event Sheet	43
Creating Events	43
Modifying Events	43
Comments	44
Scrolling and Scale	44
Finding In Events	45

BARS	46
Project	46
Organising Projects	46



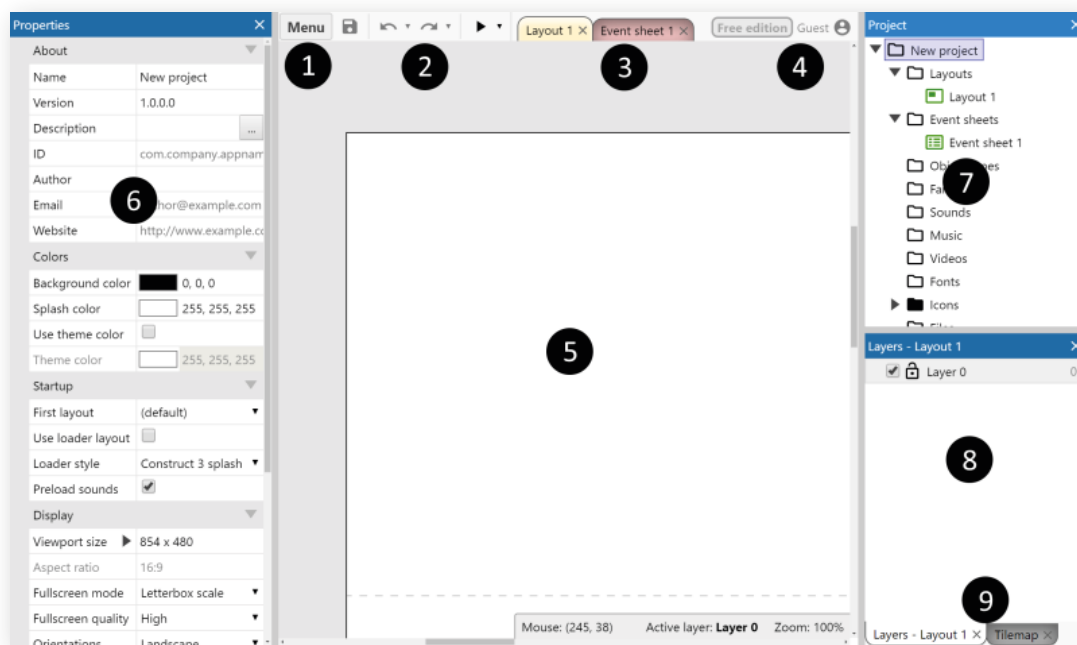


Managing Items in the Project.....	47
Importing Files	47
Properties	47
Special Features for Number Value Properties	48
Layers.....	48
The Layers List.....	49
Tilemap	50
Toolbar tools	51
Editing tile collision polygons	52
Bulk editing	52
<hr/>	
ANIMATIONS EDITOR	53
Colour Palette	53
Image Tools.....	53
Drawing Tools	54
The Frames Pane	55
The Animations and Properties Panes.....	56
The Image Points Pane	56
Editing Image Points.....	57
Drag-and-Drop Importing of Files.....	57
<hr/>	
C3 ON MOBILE.....	58
Accessing Bars.....	58
Changing UI Mode.....	59
<hr/>	
ADVANCED FEATURES	60
The Debugger.....	60
File Editors.....	61
The Array Editor	61
Dictionary Editor	62
Scripting.....	63

OVERVIEW

This chapter will introduce various aspects of the Construct 3 User Interface, starting with the most important features.

The following image highlights the important parts of the Construct 3 user interface (UI) with numbers. An overview of each part is provided below, and later in this chapter, each section will be covered in more detail. Note that initially, only the Start Page is visible. Much of the interface will not appear until you create or open a project. Also, please note the exact appearance of Construct 3 can depend on which theme you have selected.



1. Main menu button

Click this to open the main menu. This provides options for basic tasks like opening and closing projects, exporting, changing settings and so on.

2. Main toolbar

This provides shortcuts to the most used features: save, undo, redo and preview. The arrows next to some buttons provide a dropdown menu with more options.

3. View tabs

These tabs let you switch the main view between different layouts (where you place objects) and event sheets (where you define logic using the event system).

4. Account badge

This shows your account status. Click it to show the Account menu.

5. Main view

This is where the currently selected Layout View or Event Sheet View appears. The view tabs select which is visible. In this picture, it's showing an empty Layout View.

6. Properties bar

This lists all the properties for the selected item, allowing you to change settings for it.

7. Project bar

This lists everything in your project. It gives you an overview of what you've added and lets you navigate around the project as well, such as by opening layouts or event sheets to view them.

8. Layers bar

When a Layout View is open, this shows the layers on the layout.

9. Tabs

By default, the Layers bar and Tilemap bar are docked together. You can use these tabs to switch between the bars. You can drag and drop bars around to rearrange them. You can dock or tab together any combination of bars you want to customise the interface.

CHANGING THE THEME

You can choose a different theme to alter the appearance of Construct's interface, for example, choosing a dark theme. The current theme can be changed from Settings. All the screenshots in this book are with Construct 3 using its Default theme - it also has two other themes, Dark and Light.

SIMPLIFIED USER INTERFACE

Construct 3 has many features, which is great for people who want to make complex and advanced games. However sometimes the large amount of available tools can be a bit overwhelming. For example if you are teaching a class of young students, you don't need to have so many plugins and properties. It can be easier to work with a reduced set of options, helping make it clearer what to do and minimising the chance of a mistake.

Construct's Simplified user interface mode is designed to solve this problem. It hides all advanced features, leaving behind a simple, straightforward set of features that are ideal for beginners.

To enable this mode, simply open the Settings dialog (Menu ► Settings). In the User interface section, check the Use simplified user interface option. Then close the Settings dialog.

You'll see a message reminding you the simplified user interface is enabled. You can ignore it and click OK. That's just there to help stop people getting confused in case they're wondering why some options have disappeared.

You don't need to restart Construct, nor reload your project - the setting takes effect right away.

Here is a list of some of the things hidden in simplified mode. This isn't an exhaustive list, and it may be tweaked over time, but it serves to give you an idea of what is hidden.

- All features related to JavaScript coding
- All third-party addons
- Advanced plugins, like **WebSocket**, **XML** and **AJAX**, as well as all platform-specific plugins (e.g. **Facebook**, **NW.js**) other than the **Scirra Arcade** plugin
- Advanced behaviors, like the Custom movement, No Save and Shadow Caster
- Advanced or less commonly used properties in the Properties Bar, like effect properties, container settings, UID & Z Index information, advanced project/layout/layer settings, and advanced plugin/behavior settings like form control IDs, origin location, and Sine behavior randomizers
- Advanced or less commonly used event features, such as positioned audio and effects in the Audio object, advanced physics modes, key-code based input events, effects settings and JSON saving/loading

In many cases, particular sections are radically simplified. For example, Project Properties reduce from a long and complex list into a few options for name, description, and viewport size. The Audio plugin's actions also reduce from a wide set of positioning and effects features to simple playback control. Overall, this can make it much easier to guide students to the right parts of the software and discourages them getting side-tracked with irrelevant features.

The setting has no impact on project compatibility. You can save and load any projects as normal and open them in either mode. The simplified user interface mode only hides options in the user interface, so the change is purely cosmetic.

It could be confusing to open an advanced project in simplified mode, because it will use many features which are hidden. Those features will continue to work, but it will be difficult to edit or recreate them, since the relevant options aren't visible. For this reason, we recommend starting a new project in simplified mode, or only using basic existing projects that have been prepared for this mode.

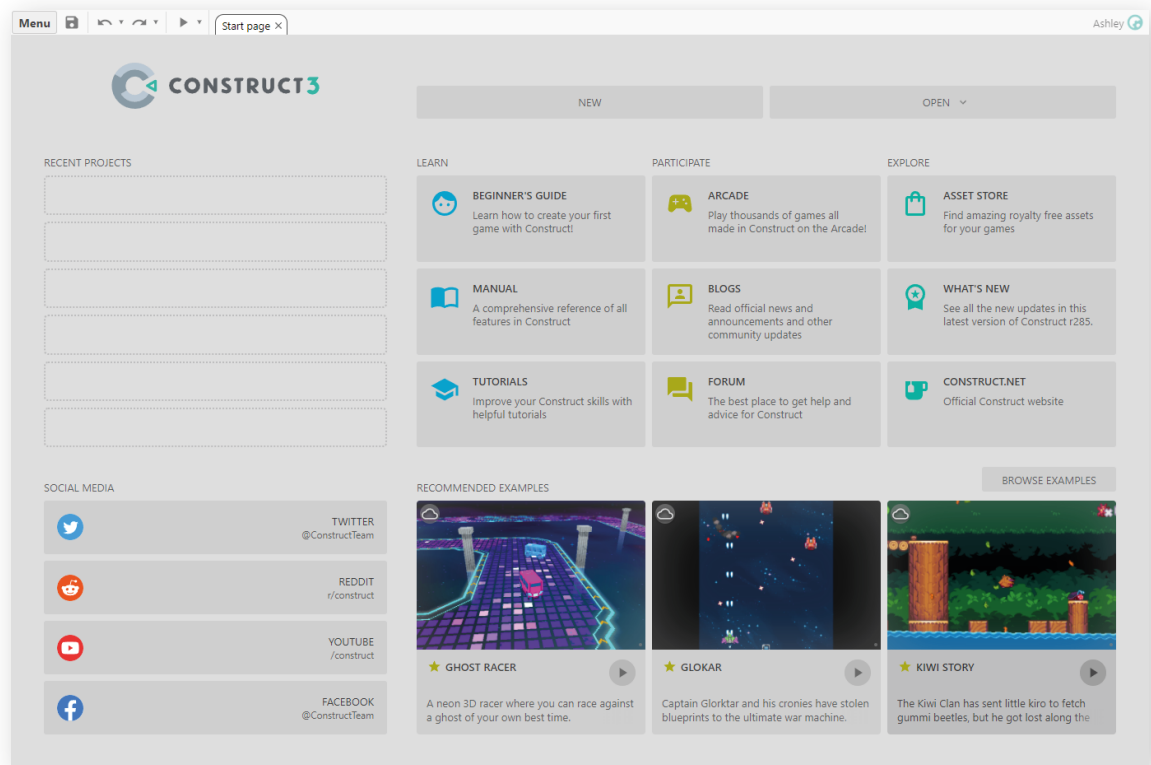
You can turn off simplified mode, restoring the full features of Construct in the interface, simply by opening Settings again and unchecking the Use simplified user interface option. As before you don't need to restart Construct or reopen the project, it takes effect immediately.

It might be a good idea to start complete beginners in simplified mode, and as they gain more experience with Construct, later turn off simplified mode and move on to more advanced features.

START PAGE

When you first start Construct 3, it shows the Start Page which gives you a useful starting point whenever you launch Construct 3. It provides shortcuts for tasks like creating a new project, opening an existing project, a set of useful links, and a library of game demos, templates and examples that you can browse through.

The appearance of the Start Page changes depending on the size of the window or screen. It will look something like this on a desktop display.



The Start Page initially fills the whole window. When you create or open a project, the rest of Construct 3's interface will appear.

STARTING A NEW PROJECT

Click New project to create a new empty project. You'll be prompted for some basic details about the project to create. You don't need to enter anything though, just click Create and you'll get a new empty project with default settings.

OPENING EXISTING PROJECTS

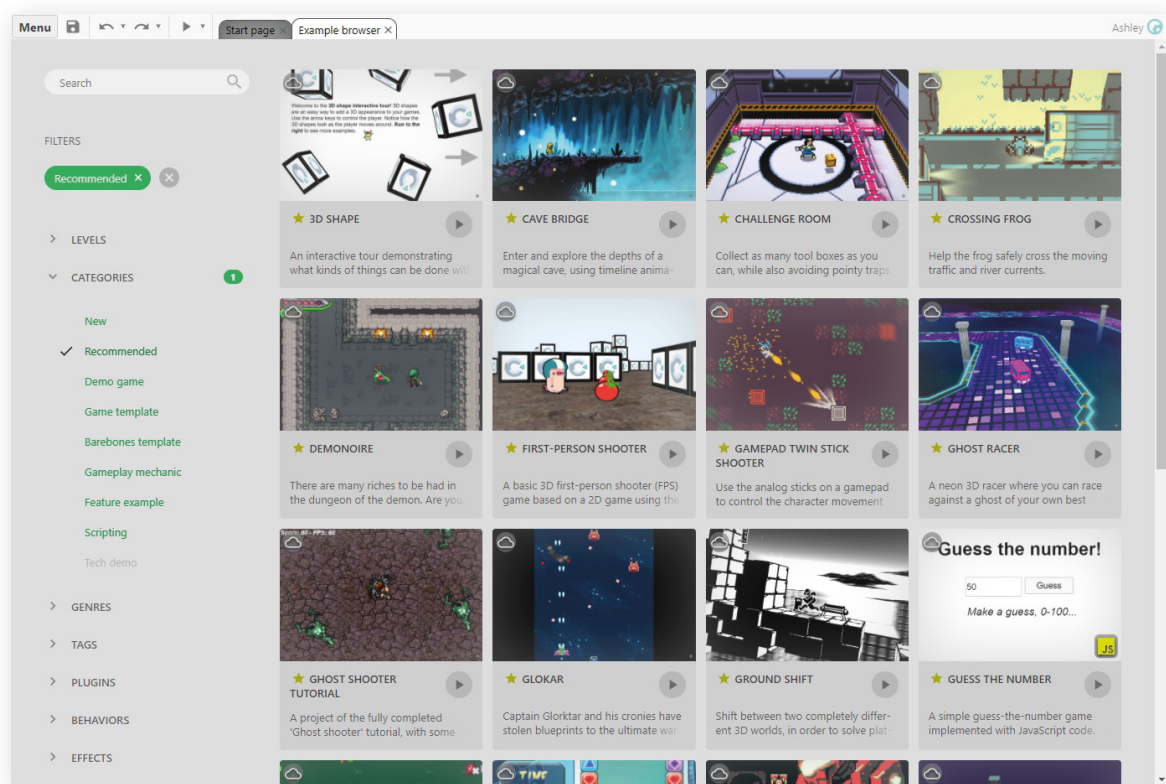
You can open projects from several sources: Cloud (projects saved to a cloud storage service like Google Drive) or local files and folders (depending on browser support). You can also find previously saved projects in the Recent projects section. We'll talk more about Construct's various saving methods in a later chapter.

RESOURCE LINKS

There are lots of links on the Start Page to other resources to help you get started and find out more about Construct. You can find links to community resources like the forums, social media accounts for Construct where you can follow news and updates, and other learning resources like tutorials.

THE EXAMPLE BROWSER

A great potential tool for teachers is the Example Browser. Construct comes with hundreds of example projects, and these are all housed in the Example Browser. All our examples are available to view in the free edition of Construct so you can still learn from them, however, if they exceed the free edition limits, you won't be able to edit them.



There are several ways to open the Example Browser. The main ways are:

- Click [Browse examples](#) on the Start Page
- Click one of the three recommended examples on the Start Page to open it in the Example Browser
- Select [Menu](#) ► [View](#) ► [Example browser](#)

When no project is open, the Example Browser fills the whole window, like the Start Page and will appear in its own tab along the top of the window. This makes it easier to browse the content.

To help navigate the volume of content, the Example Browser has lots of tags which are broadly organised into the following categories:

- **Levels:** tags that indicate the approximate difficulty level to understand a project, covering [Beginner](#), [Intermediate](#) and [Advanced](#).

- **Categories:** tags describing broad categories of example projects. These include:
 - **New:** example projects added since the last stable release
 - **Recommended:** a hand-picked selection of the best or most interesting example projects
 - **Demo game:** complete games, covering title screens, multiple levels, and an ending
 - **Guided tour:** step-by-step interactive guides that show you how to get started with using various features of Construct. These are great for beginners or quick introductions to other features of Construct you might not have used before.
 - **Game template:** projects with a single level demonstrating a game concept
 - **Barebones template:** minimal projects with placeholder graphics demonstrating a game concept
 - **Gameplay mechanic:** projects demonstrating a specific mechanic of a game, such as a type of movement or special effect
 - **Feature example:** projects demonstrating some of Construct's features, showing how they work and what they can do
 - **Scripting:** projects making use of JavaScript coding
 - **Tech demo:** performance benchmarks and other demonstrations of the capabilities of Construct's engine
- **Genres:** tags indicating the game genre of the example (if applicable)
- **Tags:** some other miscellaneous tags. These include:
 - **3D:** projects making use of Construct's various 3D features
 - **Mesh distortion:** projects making use of Construct's mesh distortion feature
 - **Mobile:** projects designed to work well on a mobile device with touch input
 - **Performance:** benchmarks or demonstrations of the performance of Construct's engine
 - **Scene graph:** projects making use of Construct's scene graph (aka hierarchies) feature
 - **Timeline:** projects making use of Construct's Timelines animation feature
- **Plugins:** projects sorted by which plugins they use
- **Behaviors:** projects sorted by which behaviors they use
- **Effects:** projects sorted by which effects they use

Click a tag to toggle whether the list is filtering with that tag. A list of all filter tags appears in the Filters section. Only projects matching all tags will be listed. By default, the Recommended tag is selected to show only the recommended example projects, but you can click the tag to remove it and filter the list another way.

You can also enter search terms in the search box. The list will further be filtered down to those matching both all tags and all the entered search terms.

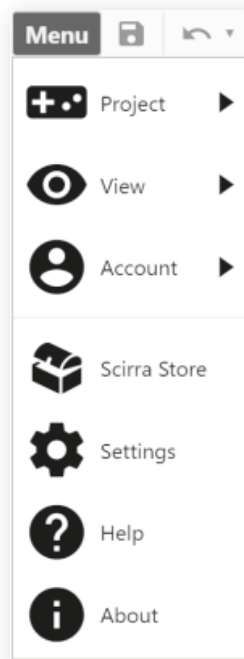
Click a project's card to expand it and see more details about it.

You can quickly preview a project by clicking its Preview button. To open the project and see how it works, click its card to expand it, then click the thumbnail image or the Open button. Once open you can then also try it out by pressing the Preview button in the main toolbar, or by pressing F5. Once you're done, close the project by selecting Menu ► Project ► Close project. When the project closes, you'll see the Example Browser again where you can choose another example project to preview or open.

You can get a direct link to an example by right-clicking a card in the Example Project and selecting Copy direct link. This will copy a link with the example embedded, such as <https://editor.construct.net/#open=kiwi-story>. When visiting this link, Construct will load and automatically open that project. This is a useful way to share examples.

MAIN MENU

Click the Menu button next to the main toolbar to access the main menu. The exact options which appear in the menu depend on whether you have a project open, and whether you are logged in.



If you're using Construct 3 with an education plan and you have a project open, you'll see the following menu structure:

Project

Save

Saves the current project to its last saved location. If it has not been saved before, this will use the *Default Save Location* in Settings.

Save As

Cloud save: save to a cloud storage service.

Save as single file: This option will be shown where it is supported by the browser. It will save to a local .c3p file representing the entire project.

Save as project folder: This option will be shown where it is supported by the browser. It will save as multiple files in a local folder, following the structure in the Project Bar. This is more suitable for large projects.

Save to local browser: This option will be only shown where file/folder saves are not supported. This option will save to the browser's storage on the device.

Download a copy: Save a copy of the current project by downloading a file.

Share with admin: Upload a copy of your project so your subscription administrator can access it.



More details regarding saving projects, including how Share with Admin works and which services are supported with Cloud Save can be found in the Saving and Sharing section of this book.

Preview

This runs a preview of the current layout.

Remote Preview

Starts a Remote Preview of the current project – this allows you to test or share projects on other devices by visiting a URL or scanning a QR code. This will be covered in more details in the Saving and Sharing chapter.

Debug

Runs a preview of the current layout with the debugger enabled.

Export

Opens the Export dialog to begin exporting the current project for publishing.

Close project

Closes the current project, prompting you to save if there are any changes.

New

Creates a new empty project – this opens the same dialog as if you created a new project from the Start Page.

Open

Brings up a further menu to select where to open a file from including Cloud services, local files, or a local project folder.

Open recent

Shows a list of recently opened projects for easy access.

Guided Tours

Beginner's Guide

A step-by-step walkthrough of our Beginner's Guide tutorial

Make a Platform Game

Similar to the Beginner's Guide, only with a focus on building a platform game

Get Started with Timeline Animations

A step-by-step introduction to using Construct's Timeline feature for animations

Get Started with JavaScript

An introduction to using JavaScript within Construct

View

Bars

In this submenu, you can hide and show any of the bars visible in the interface. If a bar is missing, or you accidentally close one of them, use this menu to bring it back. Anything with a tick next to it should be visible!

Start page

Ticking or unticking this option allows you to show or hide the Start Page.

Addon manager

Opens the Addon Manager to view and manage addons. You can find a lot of third-party addons to extend Construct 3's capabilities in the [Addon Exchange](#) on [construct.net](#)

Export manager

Opens the Export Manager dialog, which lists the last few exports and allows you to download them again.

Account

Register

Takes you to [construct.net](#) in order to register a new account to use with Construct.

Log in

Opens the log in dialog to allow you to use Construct with an existing account.

Enter access code

Allows you to enter an access code to allow temporary use of the full version of Construct. As previously mentioned, this does not apply for personal licenses. If you're already using an access code, the option to remove it will appear instead.

View details

Opens a dialog displaying more information about the Construct account currently in use.

Get Addons

This takes you to the Addon Exchange – a listing space for developers to share their free plugins, effects and other addons for Construct 3.

Asset Store

Takes you to the Asset Store where you can purchase game assets, plugins and more.

Settings

Opens the Settings dialog where you can customise Construct 3's settings to work the way you prefer. You can set options in here for things like C3's theme, your UI preferences and whether C3 prompts you about new beta releases.

Help

Opens the Construct 3 manual.

About

Opens a dialog displaying information about the version of Construct 3 you're using, as well as credits, storage information and diagnostic details.

MAIN TOOLBAR

The main toolbar provides quick access to a few of the most used options in Construct. It appears next to the main menu button.



The main toolbar has the following buttons:

Save

Saves the current project to its last saved location. If the project has not been saved yet, this defaults to Cloud Save. You can change the default location under Construct 3's settings.

Undo and Redo

Undoes the last performed action in the editor. After pressing undo, you can then redo the action again. Click the dropdown arrow next to the button to see a list of the undo or redo actions. Selecting an item from the list will undo or redo all the actions up to the chosen item.

Preview

Runs a preview of the current layout. By default, this opens a pop-up window; you may be prompted to allow pop-ups. Click the dropdown arrow next to the button to see a list of other kinds of preview. (Further information about preview types can be found in the Testing and Publishing chapter of this book.)

Debug layout

This runs the current layout in a special debug mode.

Preview project

This starts a preview from the first layout in the project. This is either the first layout that appears in the Project Bar, or whichever layout is set in the First layout project property.

Remote preview

This allows you to preview your project on a different device. It is also useful for testing different browsers on the same device.

VIEWS

LAYOUT VIEW

The Layout View is a visual designer for your objects and your game objects are organised into layouts.

A **layout** is a pre-arranged set of objects and can represent various aspects of a game including levels, menus, or title screens. They can be added, renamed, and deleted from the Project Bar and they are edited from within the Layout View. Every layout has an associated event sheet which defines how the layout works.

Layouts contain a stack of layers, and a layout must have at least one layer – try not to get **LAYOUTS** and **LAYERS** mixed up! Objects that appear on the screen do not belong directly to a layout - they belong to one of the layers in the layout.

Layouts do not have a background colour. To set a background colour, make the bottom layer opaque and set its background colour. This can be done in the Layers Bar but won't be necessary in the game we'll make later.

Layouts can also have effects applied to them, which affects all content appearing in the display.

Now you know what a layout is and what it's for, let's move on to the Layout View:

The dashed rectangle in the top left of the layout area indicates the viewport size in the layout – so this is the size of the area visible to the player at any given time. The viewport is set to this position by default, but you can change its size and orientation in the project properties.



In the bottom right-hand corner of the layout view, you'll see a small status bar with information about the current mouse position in the layout, the current zoom level, and the current active layer. The active layer is important since it is the layer new object instances are added to. The active layer can be changed by selecting a different layer in the Layers Bar.

ADDING, MODIFYING AND DELETING OBJECTS

In Construct, objects perform most of the useful work in a project. A lot of the things you see in a Construct project are represented by objects, and there are also hidden objects for other purposes (e.g., audio playback).

When inserting a new object, typically you first choose the plugin in the dialog (e.g., *Sprite*). This then creates an object type (e.g. *TrollEnemy*). When the mouse turns to a crosshair this allows you to place the first *instance*, and you can duplicate the instance to create more of them.

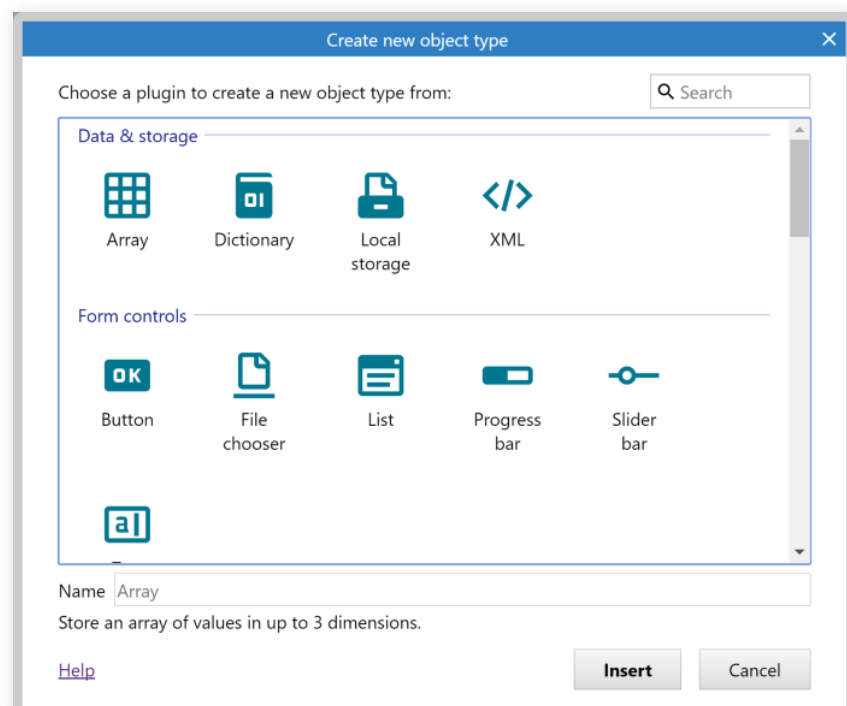
Understanding the differences between them is essential to use Construct effectively, especially *object types* and *instances*.

Object types are a central part of game design in Construct. Object types define a 'class' of an object. For example, *TrollEnemy* and *OgreEnemy* could be different object types of the *Sprite* plugin. They have different animations, and events can be applied separately to make them look and act differently, despite the fact they are both *Sprite* objects.

There can be multiple instances of an object type in a project. For example, you may wish for there to be four *TrollEnemy* objects awaiting the player in a game. These four instances share the same animations, images, behaviors, instance variables and events. (In the case of instance variables, each instance stores its own unique value, e.g. for health, and behaviors work independently for each instance too.)

So how do we go about adding objects into a project?

Double-click a space in the layout or right-click and select Insert new object to add a new object type. This will bring up the Create New Object Type dialog.



To create new instances of an existing object type, you can either copy and paste, click and drag the existing object while holding the Ctrl key, or drag and drop the object in from the Project Bar.

A shortcut for importing image files as *Sprite* objects is to drag and drop image files straight from a local folder into the Layout View. This automatically creates a new

Sprite object type with the dragged image. This will also name Sprites as Sprite1, Sprite2, Sprite3 and so on. It is recommended that you re-name your sprites to make your life easier down the line!

If multiple image files are dragged into the editor at once, the Sprite is assigned an animation with the dragged images as animation frames.

Where supported, animated image file formats like GIF and APNG can also be dragged and dropped in and will be used as a Sprite animation. (Animated image file formats can also be imported to the Animations Editor where they will be split out in to separate frames.)

Instances can be moved by dragging and dropping them with the mouse. Hold Shift to axis-lock the drag to diagonals. Alternatively, they can be nudged 1 pixel at a time with the arrow keys (hold shift to nudge 10 pixels), or coordinates can be typed in directly to the Properties Bar.

The Delete key or right-click Delete option deletes instances. Deleting all instances of an object does not remove the object type from the project. To entirely remove an object from the project it should be deleted via the Project Bar.

Click objects to select them. Objects cannot be selected if their layer is locked. Hold Control while clicking to select multiple objects or click and drag a selection rectangle to select all objects in an area. The Properties Bar displays properties for *all* currently selected objects, so changing a property sets it for every selected object.

When a single object is selected it appears with resize handles around it.



Click and drag the resize handles to stretch the object and if you hold down the Shift key, you will proportionally resize the object. Hold control to resize relative to the object origin, which appears as a small dot on the selected object.

Rotatable objects like Sprite can be rotated by moving the mouse just outside the resize handles, away from the object. When you do this the mouse cursor will change to a rotation arrow. When you see this, click and drag to rotate the object.

Sometimes the resize handles, or rotate cursor, can get in the way of other objects. If this happens, hold Alt to temporarily hide the resize handles and disable rotation. This allows you to select another object instead of modifying the selected object.

Scrolling and Zooming

There are a few ways to scroll in the Layout View:

- You can use the vertical and horizontal scrollbars at the edges of the view.
- Scrolling the mouse wheel will scroll the view vertically. To scroll horizontally, hold the shift key while scrolling the mouse wheel.
- Hold the middle mouse button or the Spacebar and drag the mouse.

On desktop systems, middle-mouse dragging is probably the most convenient way to move around the layout, while the Spacebar option makes this method more viable for laptops with trackpads.

Zooming is useful to focus on a small area or see an overview of the entire layout.

There are several ways to zoom:

- The Zoom options in the View menu when right-clicking in the Layout View
- Hold Control and scroll the mouse wheel. Hold both Control + Shift to double or halve the zoom (e.g., 100%, 200%, 400%...)
- Ctrl and + or - on the keyboard. Hold Shift to double or halve the zoom.

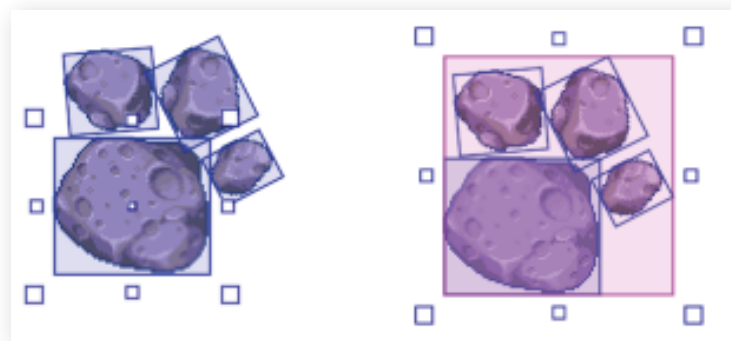
Press Control + 0 to return to 100% zoom.

If you're using Construct on mobile, you can scroll across the Layout View by swiping with two fingers, and you can zoom in or out by using pinching gestures.

Selection Wrapping

If you select two or more objects, you can wrap the selection by pressing Enter or right-clicking and selecting Wrap selection. This is a little more complicated to do on mobile, but you can hold one object, then tap others to add them to the selection. Then simply tap and hold on one object in the selection to open the menu where you'll find the option to wrap the selection.

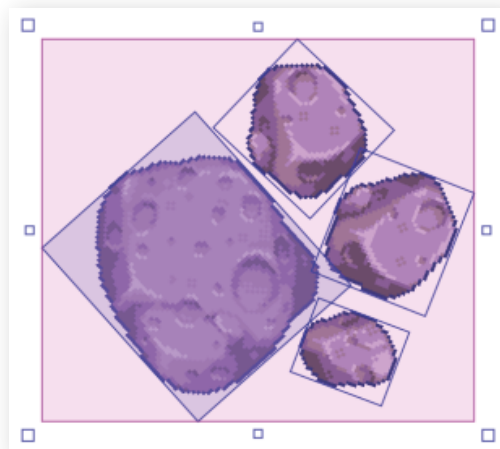
These wrapped selections will appear with a different coloured selection box, like so:



Wrapped selections can be resized and rotated as if they are one large object. For example, the selection can be enlarged and rotated, and all objects maintain their position relative to each other.

While a selection is wrapped, click any of the objects in the wrapped selection to make that object the rotation origin.

Click or tap outside of a selection to clear it.



Containers

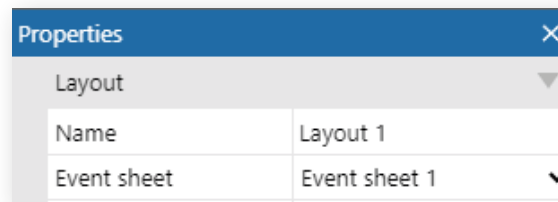


Containers are an advanced feature only available in the full edition of Construct 3. They are a little beyond the scope of this book, however we will briefly mention them here to differentiate between types of wrapped selections. You can read more about them in the Construct 3 Manual.

Objects that are grouped into Containers are highlighted in yellow in the Layout View. Containers can also be set to automatically wrap their selection. If you still need to select an individual object in an automatically wrapped selection, hold Alt and click one of the objects.

Other Considerations

As previously mentioned, each layout has an associated event sheet, which is easily accessible by either pressing Ctrl + E or right-clicking the layout and selecting Edit event sheet. It is possible to create layouts without a new event sheet – if for example you want to associate an existing event sheet with a new layout. So whenever something works on one layout, but not another, the first thing to check is whether the layout has an event sheet associated with it.



The Z Order of objects (that's the order which determines what objects sit in front of others) within a layer can be adjusted by right-clicking and selecting **Z Order** → **Send to top of layer** or **Z Order** → **Send to bottom of layer**. The full version of Construct 3 has a dedicated bar to allow advanced control over the Z Order.

Objects can be snapped to a grid for tile placement, and the collision polygons of the displayed objects can also be outlined. These features can be enabled in the layout's properties, which can be found in the Properties Bar when the layout in question is selected.

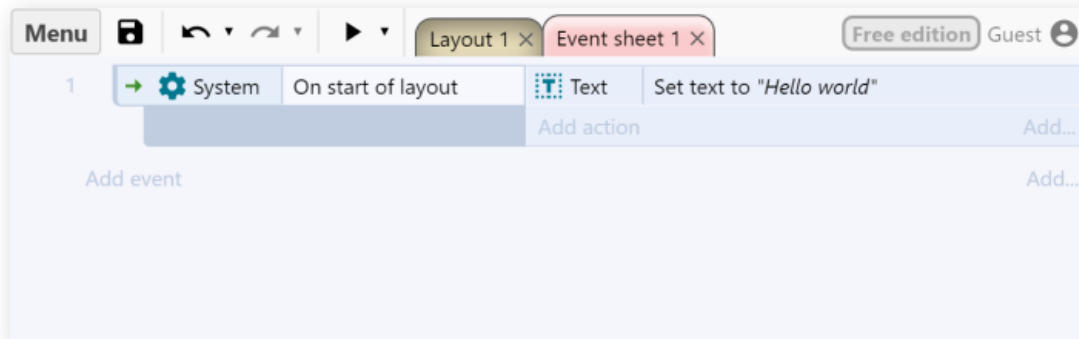
The right-click menu in the layout view also provides some alignment tools under the Align sub-menu. These allow you to quickly space objects equally or align objects along their edges. When aligning, the objects are aligned to the particular object you right-clicked.

The Animations editor can be brought up by double-clicking objects with images or animations like Tiled Background and Sprite. You can also double-click Text objects to edit their initial text in a dialog.

If you want to see any effects you've applied to objects, there's a setting under Project Properties which will display them in the layout view.

EVENT SHEET

The Event Sheet View is where events can be added, viewed, and edited in an event sheet using the event system.



Creating Events

There are several ways to add a new event:

- Double-click a space in the event sheet, or right-click in a space to see a menu of things to add.
- Click the Add event link which comes after the last event in a sheet or group or click the Add... link on the right.
- Right-click an event's margin and choose an item from the Add menu.

When you add a new event, the dialog that appears is for adding the first condition. To add more conditions to an event, right-click the margin or an existing condition and select **Add another condition** or use the Add... link on the right of the **Add action** link.

Actions can be added by clicking the Add action link (if it has not been hidden in the ribbon), or right-clicking the margin or an existing action and selecting Add another action.

Modifying Events

Double-click or select and press Enter on condition or action to edit it.

Events, conditions, and actions can be dragged and dropped around the event sheet. Holding Control and dragging will duplicate the dragged event, condition, or action. Event items can also be cut, copied, and pasted.

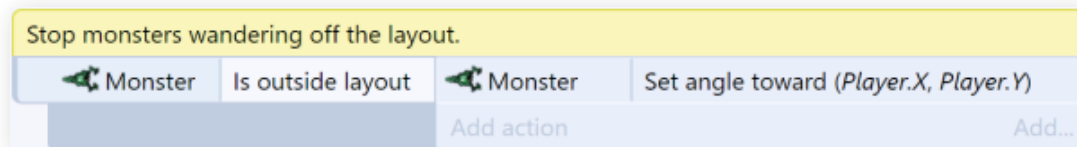
As with the Layout View, multiple selections can be made by holding Control and clicking different items. However, you can only have either events, conditions or actions selected at a time (e.g., you can't have both a condition and action selected at once). You can also hold Shift and click an event, condition, or action to select all the items in a line between the selection and clicked item.

Press R or right-click and use the Replace object option to quickly swap objects referenced in the selection. Note that objects with references to instance variables or behaviors in the selection can only be swapped with other objects with the same instance variables and behaviors which have the same names and types.

You may find it convenient to organise events into groups, which can also be activated and deactivated as a whole. Groups can also have their text and body colours changed to help with organisation.

Comments

Comments are a great way to help keep event sheets organised and to explain what various event blocks do. Having a well-commented project is a great habit to get into and makes it easier for others to understand how the project works.



To add a comment, right-click on an event or an empty space in the event sheet and select Add comment or press the Q keyboard shortcut. Comments can be edited by double clicking on them. You can make a comment with a line break by holding shift and pressing Enter.

Comments can also be added in between actions. The Q keyboard shortcut will add an action comment if an action is selected. Alternatively, you can right-click an existing action and select **Add comment** or use the Add... menu next to the Add action link.

If you use Construct a lot, you will find comments essential to help yourself organise and understand large projects. Coming back to a project after a few months with no comments at all can be very difficult, so don't underestimate the importance of comments.

Comments do not affect how anything works at all. They are solely for your information. Nothing typed into comments is exported to the game whatsoever.

By default, comments have a yellow background and appear above the event they are describing. However, you can move them around the event sheet and change their colours to help with organisation. When writing comments, you can also use BB Code for formatting.

Scrolling and Scale

There are several ways to scroll in the Event Sheet View:

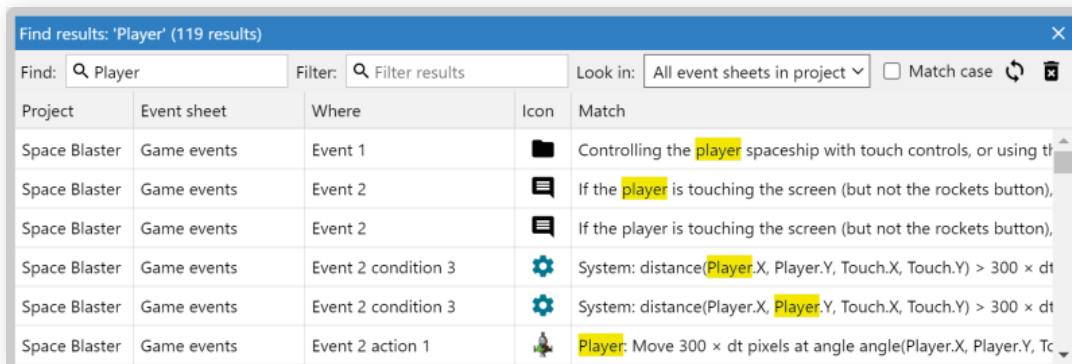
- The vertical scrollbar to the right of the view
- Scrolling the mouse wheel
- Middle-clicking to pan the view
- Pressing Space, up/down arrows or page up/down

There are some options to adjust the text size in this view as well:

- Hold Control and scroll the mouse wheel
- Press Control and + or -
- Right-click and use the Event sheet → Font size menu
- Use the browser's zoom feature, but note this will scale the whole of Construct, not just the text scale in the Event Sheet View.

Finding In Events

In the full version of Construct 3, you can search for some text in an event sheet by pressing Ctrl + F or right-clicking and selecting **Event sheet** → **Find....** This opens a dialog that allows you to enter text to search for, with options to look in the current sheet or the entire project, and whether to make it a case-sensitive search. (Case sensitive searches count uppercase and lowercase characters as different, e.g. "SPRITE" and "sprite" would be considered different in a case-sensitive search.) When you click **Find**, the results are displayed in the **Find Results Bar**.

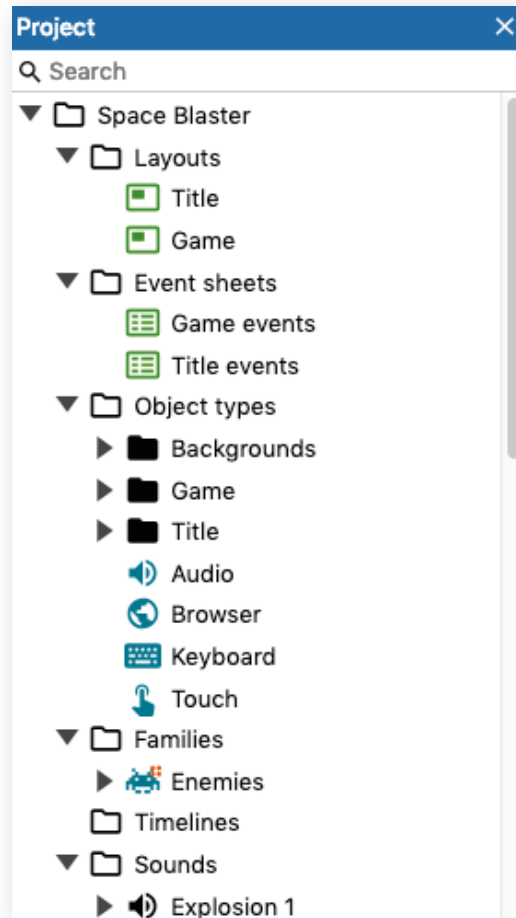


However, a text search is not always appropriate for finding in events. For example, if you want to find all events referring to an object named Sprite, searching for the text *Sprite* will also return results for other names like Sprite2, since they also include the search term. To solve this, you can use the **Find all references** feature. This is available in many places in Construct for various kinds of things like behaviors and instance variables as well. For objects, you can right-click an object in the Project Bar and select **Find all references**. This will open the Find Results Bar with a comprehensive and precise list of all references to that object, excluding any other references that happen to include the object name. This is a great way to easily review your project with confidence the results are what you want.

BARS

PROJECT

The Project Bar shows an overview of everything in your project. Everything is pre-arranged into folders according to object type, like Layouts, Sounds, Files etc.



If something in the project has changed, it is displayed in *italics*. When you save the project, everything reverts to normal text, indicating nothing has changed since the last save.

To view the properties of your current project, select the project item - the item at the top of the Project Bar with the name of the project. When you select it, the Properties Bar displays properties affecting the whole project. You can also right-click the project item to show a list of options and submenus.

Organising Projects

You can hold Control or Shift to select multiple items and drag them into a folder at the same time. However, you can only organise items into folders of the same type, e.g., you can't drag an event sheet into a layout folder.

You can also further organise your project by using Construct 3's subfolder system - available in the full version. Subfolders can be added by right-clicking a folder and selecting Add subfolder. Then, you can drag and drop folders and items to organise them into the right folders.

Managing Items in the Project

Right-click any item in the Project Bar to show a list of options. Most items can be renamed and deleted. This is the only way to fully remove an object type from a project – if you delete something from the Layout View, it will simply remove that instance, not the whole object type.

Right-clicking a folder also has the option to add a new item to that folder, such as a new layout or event sheet. Objects are more commonly added in the Layout View, but you can still add them from the Project Bar too.

There is also a search box located at the top of the Project Bar for easy project searching.

Importing Files

You can import additional external files to the project, including web fonts. These can be categorised into Sounds, Music, Videos, Fonts, Icons or the general-purpose Files folder.

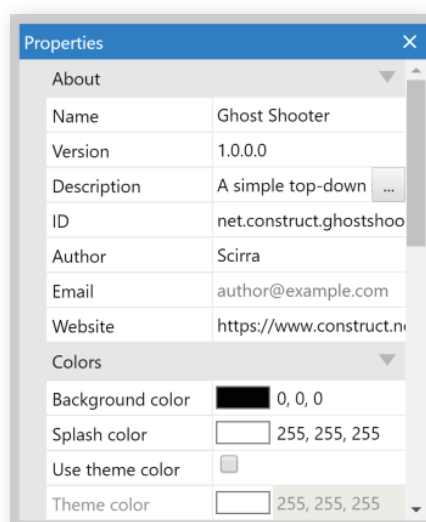
To import audio files, for example, right-click either the Sounds or Music folders and select Import sounds or Import music to open the Import Audio dialog. This allows you to pick audio files from your computer and import them to the project. Construct will convert them to the necessary format to support all browsers. To play back audio in your game, make sure you add the Audio object to the project. Most other files can be imported in a similar way.

If you have the same audio file in multiple formats, it appears as a single item in the Project Bar to save space. However, it can still be expanded to show the different versions of the file. For example, if you have *sfx.m4a* and *sfx.webm* in your project, the Project Bar will show a single item named *sfx*, with two child items for *sfx.m4a* and *sfx.webm*.

From the Project Bar, you can preview several kinds of files added to the project. Audio and video files can be played back. Web fonts can be previewed with a dialog showing some text using the font. SVG files can also be previewed. Other kinds of file can be viewed and edited using the full version's file editors.

PROPERTIES

The Properties Bar is an essential part of the interface. It displays a list of all the settings you can change on whatever is selected. The picture below shows the Properties Bar displaying a project's properties.



Properties are organised into categories which can be expanded and collapsed. There are many kinds of properties, including number fields, text fields, dropdown lists and clickable links. The property name appears in the left column, and the editable value appears in the right column.

Whenever something in the project is clicked or selected, its properties display in the Properties Bar. For example, selecting objects in the Layout View or clicking items in the Project Bar shows the relevant properties in the Properties Bar.

A lot of Construct's elements have properties, including projects, layouts, layers and object instances. Many plugins, behaviors and effects have their own properties as well. There is also a Help link displayed at the end of every property list. Click that to open the relevant manual section for those properties.

All properties also have a description which provides additional information about what the property is used for. This is displayed in a panel at the bottom of the Properties Bar – it's worth keeping an eye on this since it can contain useful hints and tips. An example is shown below.

Event sheet: Choose the event sheet to run for this layout. You can include other event sheets from this sheet.

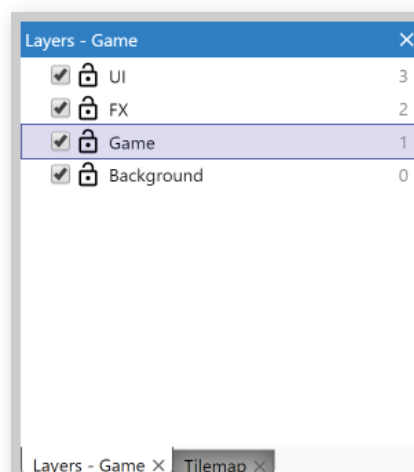
Special Features for Number Value Properties

In number values, you can type calculations like **1920 / 2** and press enter to set the value to the result of the calculation (960). The syntax used is the same as expressions used in events. You can also use some basic system expressions like **sqrt(64)**.

Number values can also be smoothly dragged with instant feedback in the Layout View. This is useful to try out a range of values and easily see which is best. To do this, click and drag vertically inside the number value cell. You can also hold Control or Shift while dragging to increase or decrease the rate of change. If you have trouble getting this to work, try first clicking inside the cell (which should select the text in the cell), and then click on the selected text and drag vertically.

LAYERS

The Layers Bar is used to add, edit and remove layers in a layout. A layer is like a sheet of glass objects are painted on to. This allows easy arrangement of which objects display in front of other objects, for example showing foreground objects in front of the background sprites. It also allows for interesting depth effects like parallax, and layers can be individually scaled and rotated as well.



A common arrangement for layers might be:

- HUD (top layer - health bar, UI info etc.)
- Foreground (objects appearing on top, e.g. explosions and effects)
- Middleground (main game objects such as the player and enemies)
- Background (bottom layer - the background)

Note that the Free edition is limited to using two layers only.

Layers can be dragged and dropped in the Layers Bar to change their order. Layers at the bottom of the list are displayed at the back (e.g. background objects), and layers at the top of the list are displayed at the front (e.g. HUD objects).

Selecting a layer displays its properties in the Properties Bar and sets it as the active layer which new objects are inserted to.

The Layers List

Each layer in the list has the following:

- A checkbox to toggle whether the layer is visible in the editor (this does not affect the game when previewing or exporting)
- A padlock icon which toggles the layer's locked status. If a layer is locked, objects on that layer cannot be selected in the editor. This is useful to prevent accidental selections on rarely used layers like backgrounds.
- A number to the right which is the zero-based index of the layer (the first layer is number 0, not 1). If you need to enter a layer number in the event system, this is the corresponding number. You can also enter layer names in the event system, which is often more convenient since unlike the numbers, the names don't change if you reorder layers.

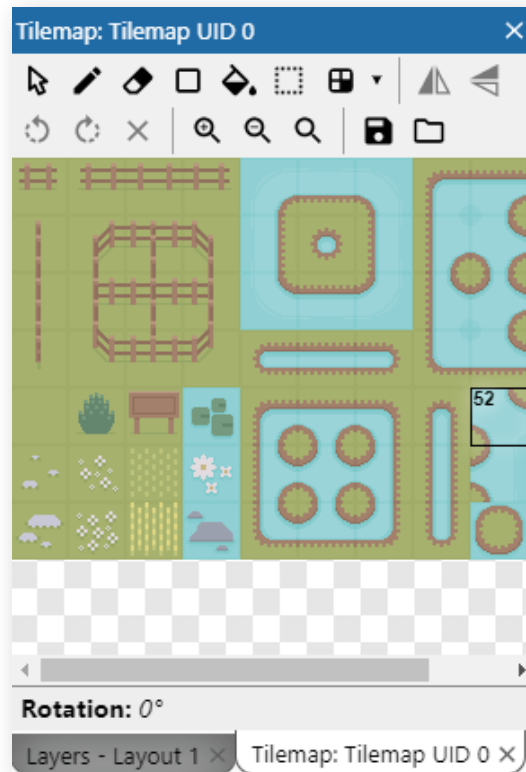
You can right-click a layer to see a menu of additional options, such as to add a new layer at the top or bottom, rename or delete layers, or shortcuts to show, hide, lock, or unlock all layers (or all but the selected one).

Layers can also be added as sub-layers of another layer. They can also be moved to be sub-layers via drag-and-drop. Sub-layers appear indented in the Layers Bar to show they come under another layer, and the layer they belong to can be expanded or collapsed to show or hide all its sub-layers.

Sub-layers can be used solely to help organize long layer lists, acting like layer folders. However, applying an effect to a layer will also affect all its sub-layers. This allows for more efficiently processing effects (instead of having to add the same effect repeatedly to several layers), as well as more advanced effect composition across different groups of layers.

TILEMAP

The Tilemap Bar allows editing tilemaps in the Tilemap object from the Layout View. It provides a toolbar with various tools and options, and a view of the current tileset image.



To add a tilemap and start editing it, follow these steps:

1. Add a Tilemap object to the layout and make sure it is selected
2. Choose the Pencil or Rectangle tool from the Tilemap bar's toolbar
3. Select a tile in the tileset showing in the Tilemap bar
4. Click inside the Tilemap object to start drawing the selected tile

You can hold shift and right-click a tile in the Layout View to pick that tile to draw with. You can also hold shift and drag the right mouse button over a range of tiles to select that range of tiles as a patch you can stamp out.

To stop editing the tilemap's tiles and return to normal layout editing, click the mouse cursor on the Tilemap bar's toolbar to restore normal layout view selection. This also allows you to move and resize the entire tilemap object.

If you have multiple tilemap objects, only the selected tilemap is edited. It is often useful to layer tilemap objects directly on top of each other, in which case the tilemap to edit can be most easily selected using the Z Order Bar if you're using the full version of Construct or hiding/locking layers with the Layers Bar.

If you are dealing with small tiles, you can also zoom the tileset image using the toolbar buttons. You can also access some of these options via a menu when right-clicking inside the Tilemap Bar.

Toolbar tools

The Tilemap Bar's toolbar has the following options:

Normal layout view selection

Stop editing tiles and select the Tilemap object like any other object.

Pencil tile tool

Draw tiles with the mouse. You can also select an area of tiles by dragging across several tiles in the displayed tileset, and then use this tool to stamp that region of tiles in to the tilemap. You can also hold shift and right-click to drag an area over the Tilemap object to select a region of tiles to copy or use the selection tool to do the same.

Erase tile tool

Erase tiles from the tilemap so they appear as transparent space. Larger areas can be erased by selecting a larger area of tiles in the tileset. A shortcut for erasing single tiles is to right-click while another tool is selected.

Rectangle tile tool

Draw a rectangular area of tiles by clicking and dragging in the Tilemap object. You can also select a 3x3 area of tiles in the displayed tileset, and the tool will automatically nine-patch the tiles. This also works for drawing single rows or columns with smaller selections such as 1x3 or 3x1, where the first and last tile are the first and last in the selection, and the rest are the middle tile repeated.

Fill tool

Much like using a fill tool in an image editor, this allows filling a continuous area with a new kind of tile. If multiple tiles are selected in the tileset, they are repeated over the fill area.

Select tool

Click and drag to select a range of tiles to use in the Tilemap object. Then switch to another tool to use that selection. For example, switching to the Pencil tool allows you to stamp out copies of the selected range. A shortcut for this is to hold shift and right click and drag an area while the Pencil tool is selected. The individual tiles in the selection will be highlighted in the Tilemap bar

Auto tile tool

This tool uses predefined brushes to automatically place the correct tile as you draw. Just pick the brush from the dropdown menu next to the tool's button and start using it. The brushes that are created by default are configured to work properly with the default tileset image. To create or modify brushes for use with different tilesets use the Tilemap Brush Editor.

Mirror

When using the Pencil tool, tiles will be placed flipped horizontally. This can also apply to an entire patch of tiles.

Flip

When using the Pencil tool, tiles will be placed flipped vertically. This can also apply to an entire patch of tiles.

Rotate anti-clockwise

When using the Pencil tool, tiles will be rotated 90° anti-clockwise. This can also apply to an entire patch of tiles. Click repeatedly to keep rotating tiles another 90°.

Rotate clockwise

When using the Pencil tool, tiles will be rotated 90° clockwise. This can also apply to an entire patch of tiles. Click repeatedly to keep rotating tiles another 90°.

Reset transformation

Restores tiles to no mirror, no flip and no rotation.

Zoom in, Zoom out, Reset zoom

Adjust the zoom of the source tileset image displayed in the Tilemap Bar. This is useful if you are dealing with particularly small tiles.

Save to TMX

Export a zip with the current tileset image and the current tiles as a .tmx file (as used by the Tiled editor). Note that Construct does not support all of Tiled's features, so importing then exporting a TMX may lose some data, such as terrain definitions. Also since in Construct a Tilemap object represents a single layer of tiles, the exported TMX file will also only ever have one layer.

Load TMX

Import a .tmx tilemap as used by Tiled. All the tiles in the object are replaced with tile data from the TMX file. In Construct a Tilemap object represents a single layer of tiles, so if the TMX file has multiple layers you will be asked which layer to import. To import all layers, create a different tilemap object for each layer and import them separately. The tileset image can also be replaced by choosing a new image file. Note you can also drag-and-drop individual .tmx files, image files, and .zip files of both, in to the Tilemap Bar. This opens the load TMX dialog with all relevant fields already filled in, so you only need to press OK.

Editing tile collision polygons

Each tile can have an individual collision polygon which is used when testing for collisions with the tilemap object. To edit a tile's collision polygon, double-click the tile in the Tilemap Bar. The Animations Editor will open to edit that tile. You can use the collision polygon tool to edit the tile's collision polygon. While the tool is active, you can also right-click and choose Toggle collision polygon to disable collisions for that tile entirely, such as if it is for decorative purposes only.

You can also use the image editing features of the Animations Editor to alter the image of the tile.

When hovering the mouse over a tile in the Tilemap Bar, its collision polygon is shown as an outline, if it has one. This helps you to quickly review the collision polygon set for each tile.

Bulk editing

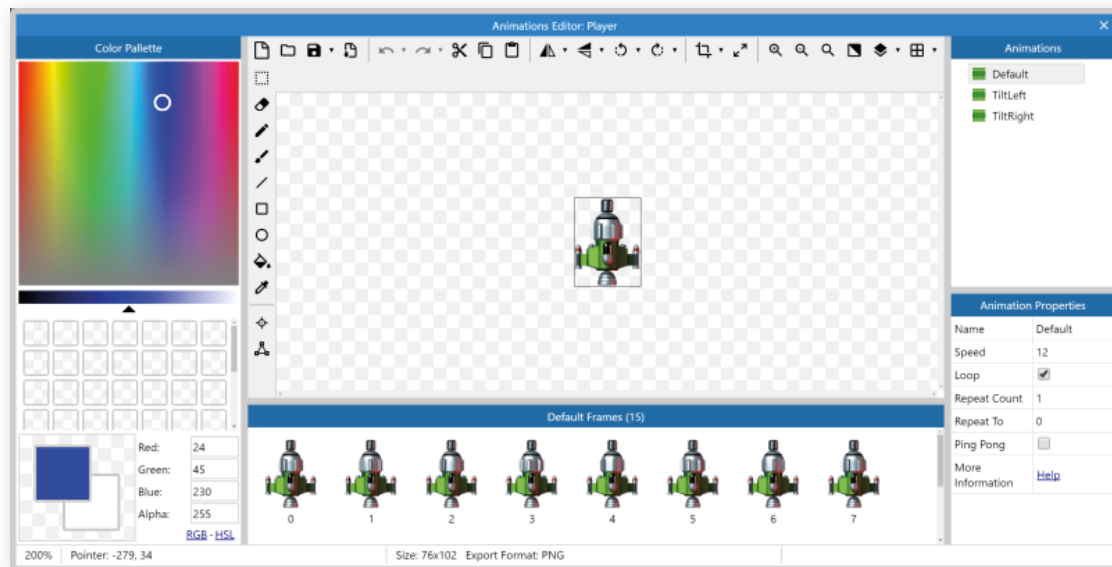
There are four context menu options to toggle the state of multiple collision polygons at the same time, they are the following:

- **Enable selected tile collisions:** enable the collision polygon of all the tiles highlighted in the Tilemap bar.
- **Disable selected tile collisions:** disable the collision polygon of all the tiles highlighted in the Tilemap bar.
- **Enable all tile collisions:** enable all the collision polygons of the tilemap.
- **Disable all tile collisions:** disable all the collision polygons of the tilemap.

ANIMATIONS EDITOR

Construct 3 has a fully featured, built-in image and animations editor, used to create animations for the Sprite object. When opening for an object without animations, such as Tiled Background, the animation editing features are hidden and it acts as a normal image editor. For brevity, it is consistently referred to as the Animations Editor, even in cases where it is only editing a single image.

To open this editor, double-click an object with an image or animations in the Layout View or Project Bar. It will also open when you add a new object that requires an image or animations.



Each pane in the Animation Editor can be resized by dragging the borders, similar to how you can with the main Construct 3 interface. This lets you customise the layout of the Animation Editor.

COLOUR PALETTE

The colour palette appears on the left and allows a colour to be picked for the drawing tools. You can choose both a primary and secondary colour with left and right-click. The pane also has cells that can be used to remember a set of colours. Right-click a cell to save or use the primary or secondary colour. By default, left-clicking the cell will set the primary colour.

You can use colour codes to specify a colour too – these can in any of the following formats:

- r, g, b or r, g, b, a
- rgb(r, g, b) or rgba(r, g, b, a)
- hsl(h, s, l) or hsla(h, s, l, a)
- Hex as either #ffffff or ffffffff

IMAGE TOOLS

The top toolbar in the image pane provides tools that affect the entire image, such as mirroring and flipping. If you hover over an icon on the toolbar, it will show you the name of the tool and an associated shortcut if it has one.

In order, from left to right, the tools available are:

Clear Image: this resets the image to a transparent box.

Open: imports an image from a local file. You can also choose SVG files, but these will be rastered into a bitmap at a given size.

Save: exports a copy of the current image. In the browser, this downloads the current image as a PNG file. You can use the dropdown next to the button to save the current animation, or all animations, bundled in a zip file.

Set export format: this opens the *Image Format dialog*, allowing you to choose whether the image is saved as PNG or JPEG when the project is exported. This can also be applied to the current animation or all animations. Bear in mind that Construct stores all images in the project in a lossless PNG format; images are only converted on export.

Undo and Redo: allows you to undo or redo actions – use the drop-down arrows next to each button to select multiple actions to undo/redo.

Cut, Copy, Paste: performs clipboard operations with the image.

Mirror and Flip: inverts the image on one of its axes. Mirror inverts horizontally, while flip inverts vertically. Use the dropdown next to the button to affect the entire animation.

Rotate: rotates the image in 90° either clockwise or anti-clockwise depending on which tool you use. Use the dropdown next to the button to affect the entire animation.

Crop: automatically removes transparent space around the edges of the image. This is a good idea to save memory. This action leaves a 1px transparent border to improve the image quality at the edges. Use the dropdown next to the button to affect the entire animation.

Resize: resizes the current image. A dialog will open with options for the resize, including a checkbox to apply the resize to the whole animation.

Zoom options: the three magnifying glass type icons are your zoom controls – zoom in, out and reset zoom. These adjust the zoom level in the image editor, or you can use Control + mouse wheel.

Toggle background brightness: switch between a light and dark background for the image editor. Changing to a dark background can be useful when editing very light images.

Toggle onion skin: this feature is only available in the full version of Construct 3 and allows you to display adjacent frames translucently over the current image when editing an animation. This can help when drawing animations. The options are to display the previous frame, next and previous frames, or next and previous two frames over the current image.

Grid: toggle the display of a grid over the current image. Use the dropdown next to the button to adjust the grid settings, such as the grid size, colour and whether to snap to the grid.

DRAWING TOOLS

The side toolbar provides some tools for drawing in the image, as well as some extra Construct-specific tools for setting image points and adjusting the collision polygon. Some tools have extra settings, such as the size for the brush tool, which appear underneath the top toolbar. The following tools are available, in order from top to bottom:

Rectangle select: allows you to select rectangular sections of the image. You can then move, delete, cut, copy and paste them.

Pencil: draws with a solid square, useful at 1px size for pixel art.

Brush: draws with a soft round brush.

Line: draws straight lines. Hold shift to lock the angle to 5° increments.

Rectangle: draws a rectangle in the image, using the secondary colour as the border colour. Hold shift to draw a square.

Ellipse: draws an ellipse in the image. As with the rectangle tool, the secondary colour is used as a border. Hold shift to draw a circle.

Fill: fills a continuous area of the image with a colour. You can also choose to enable or disable flood fill to alter how the editor manages filling.

Eye dropper: selects a colour from the image. Alternatively, hold Control and click with another tool selected.

Image points: this button displays the origin and image points for your image and allows you to edit them.

Collision polygon: This allows you to adjust the collision polygon – that is, the area that counts as colliding for this image. By default, Construct guesses a collision shape, but it is not always accurate, so you can click and drag the points of the collision polygon to alter its shape as you need to. Right-click on a point or in a space to display a menu of additional options for the collision polygon, such as adding and deleting points. Some objects, like Tiled Background, do not use collision polygons.

THE FRAMES PANE

The bottom pane displays a list of all frames in the current animation. You can add or delete frames from here or drag and drop them to alter their sequence. If you right-click a frame you have the options to duplicate or delete that frame. You can also adjust the size of the frame icons appearing in the pane by adjusting the Thumbnail size.

If you want to edit the image of a different frame, simply select it in the Frames pane. This also shows a single property in the properties pane: *Duration*, which is a multiplier for the amount of time to spend on the frame. For example, a frame duration of 2 will spend twice as long on that animation frame, 0.5 half as long, etc. relative to the current animation speed.

Right-click in an empty space in the pane to see additional options for managing the animation, which include:

Add frame: add a new empty frame at the end of the animation

Duplicate last: add a new frame which is a copy of the last frame in the animation

Reverse frames: invert the sequence of frames in the animation

Import frames

From files: add multiple animation frames by selecting a set of local image files to import

From strip: add multiple animation frames by selecting a local image file with multiple images placed on it (often called a *sprite strip*) and cutting it up into individual images. You must specify the number of cells horizontally and vertically, and the direction to read cells in.

THE ANIMATIONS AND PROPERTIES PANES

The right side of the Animations Editor shows the Animations pane, where animations can be added, edited, and deleted. If you right-click in a space, you can add a new animation or add a subfolder to organise animations. Whereas if you right-click on an animation itself, you'll see options like Preview, which shows how the animation will look in the game. If you're using the full version of Construct 3, you can also use **Find all references** for an animation to locate all its references in events.

Selecting an animation also switches which frames are showing in the frames pane and displays settings for the animation in the Properties pane. The following properties are available for animations:

Name: the name of the animation. This can also be directly edited in the Animations pane.

Speed: the rate at which to play the animation, in animation frames per second. For example, if set to 5, each frame will last 1/5th of a second. Remember that this cannot be faster than the game framerate, which is typically 60. Set the speed to 0 if you do not want the animation to play, which is useful if you want to control which frame is showing by events. You can also use negative speeds, which causes the animation to play backwards. If you do choose to do this then repeating animations should set the Repeat to frame at the end of the animation, otherwise, by default, it repeats to frame 0 (the start of the animation), causing the animation to stop after playing in reverse.

Loop: enable to infinitely repeat the animation.

Repeat count: if the animation is not looping, the number of times to repeat the animation.

Repeat to: the zero-based index of the animation frame to go back to when the animation loops or repeats.

Ping pong: play the animation alternately forwards and backwards when looping or repeating.

THE IMAGE POINTS PANE

When you select the Image points tool in the image editor, the left pane switches to a list of image points for the current animation frame.

Image Points		
	Name	Number
⊕	Origin	0
▣	Imagepoint 1	1

The Origin is a special kind of image point defining the centre of the object, or its point of rotation. It has a different icon to denote it.

The term *image point* usually means "image points including the origin". Image points have a zero-based index, and the first image point (number 0) is always the origin. The origin cannot be renamed.

You can also add additional image points. These are useful to create spawn points for other objects. For example, you may place an image point at the end of a gun, so spawned bullets can appear at the end of the gun barrel instead of at the origin. Image points can also be given a name and referred to in events by this name.

Editing Image Points

Select an image point in the list and a corresponding point will appear on the image. Left click to place the point under the mouse. The arrow keys can also nudge it 1 pixel in each direction.

An image point can be quickly placed using the number pad, e.g. 1 for the bottom-left corner or 5 for centred. Alternatively, the image point can be right-clicked in the Image Points pane and an option chosen from the quick assign menu.

Right-clicking an image point in the Image Points pane also provides **Apply to whole animation** and **Apply to all animations** options. These set the image point in the same relative place in all frames in the current animation, or all frames in all animations, respectively. If an image point does not exist in all frames, this option also creates it. Holding shift while placing the image point is a shortcut for this.

DRAG-AND-DROP IMPORTING OF FILES

There are various ways you can import images by drag-and-drop into the Animations Editor window.

An image file can be dropped into the main image editing pane to replace the content of the current frame with the dropped image file. This works the same way as using the Open button on the top toolbar.

Dropping a single image file into the Frames pane will prompt you asking how the image should be treated. The image can be treated as a plain image file, in which case the image is added as a new frame in the current animation. This works the same way as the context menu option **Import** → **From files**. The image can also be treated as a sprite sheet, which works the same way as the context menu option **Import** → **From strip**.

Dropping multiple image files into the Frames pane will add a new frame for each dropped image file. This works the same way as the context menu option **Import** → **From files**.

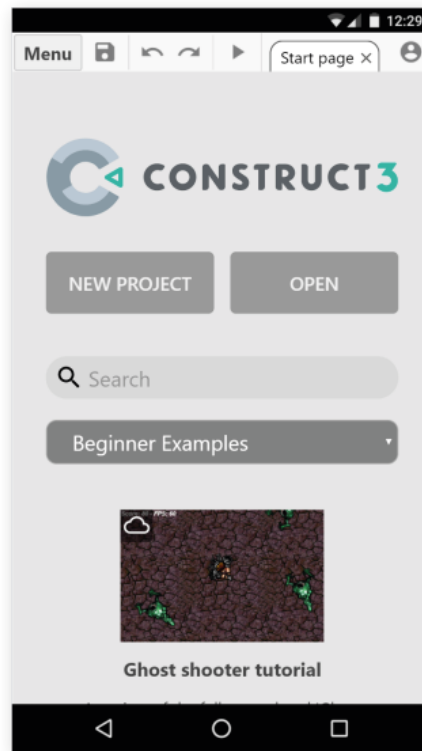
You can drag-and-drop either a single image file or multiple image files into the Animations pane, and it is handled in the same way as with the Frames pane, except that the frames are added to a new animation.

In supported browsers, animated image file formats like GIF and APNG can be imported as a sequence of frames. This covers any method of importing an image file, including via drag-and-drop or by the toolbar Load frames option.

C3 ON MOBILE

Construct 3 works on mobile devices like phones and tablets. While Construct 3 can be used on a phone, it is much more comfortable to use a tablet device with a larger screen if you have one available.

Construct 3 adapts its appearance to better suit these devices. On mobile it will look something like this:



When using Construct 3 on a mobile device, you can use touch equivalents to mouse clicks, for example:

- Where you would click on an item, tap on it.
- Where you would double-click, double tap instead.
- If you need to right-click something, or open a context menu, tap and hold on the item. After a moment a menu will appear.

ACCESSING BARS

Many mobile devices have small screens, so, to make the most out of the space available, Construct 3 hides bars by default. To access these bars, swipe in from the side and the bar will slide in. Since there are a number of bars and only two sides, you can access the other bars by repeatedly swiping in from the side again. As you do this the previous bar will slide out and the next bar will slide in.

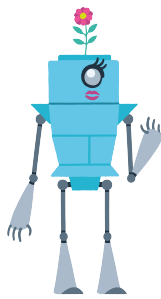
On the left side the sequence of bars is:

1. Properties Bar
2. Bookmarks Bar
3. Find Results Bars
4. Z Order Bar

On the right side the sequence of bars is:

1. Project Bar
2. Layers Bar
3. Tilemap Bar

If you reach the end, the sequence will start again, cycling through the set of bars for that side of the screen. Bars can be closed by swiping them back the other way. The next time you swipe in from the side of the screen, you'll always get back the last bar you used that side. That helps you keep using the same bar for a while, and you can keep swiping to switch between bars at any time.



Note that only some of Construct's bars have their functions covered in this book. For more details about all of Construct 3's functionality, please refer to the manual.

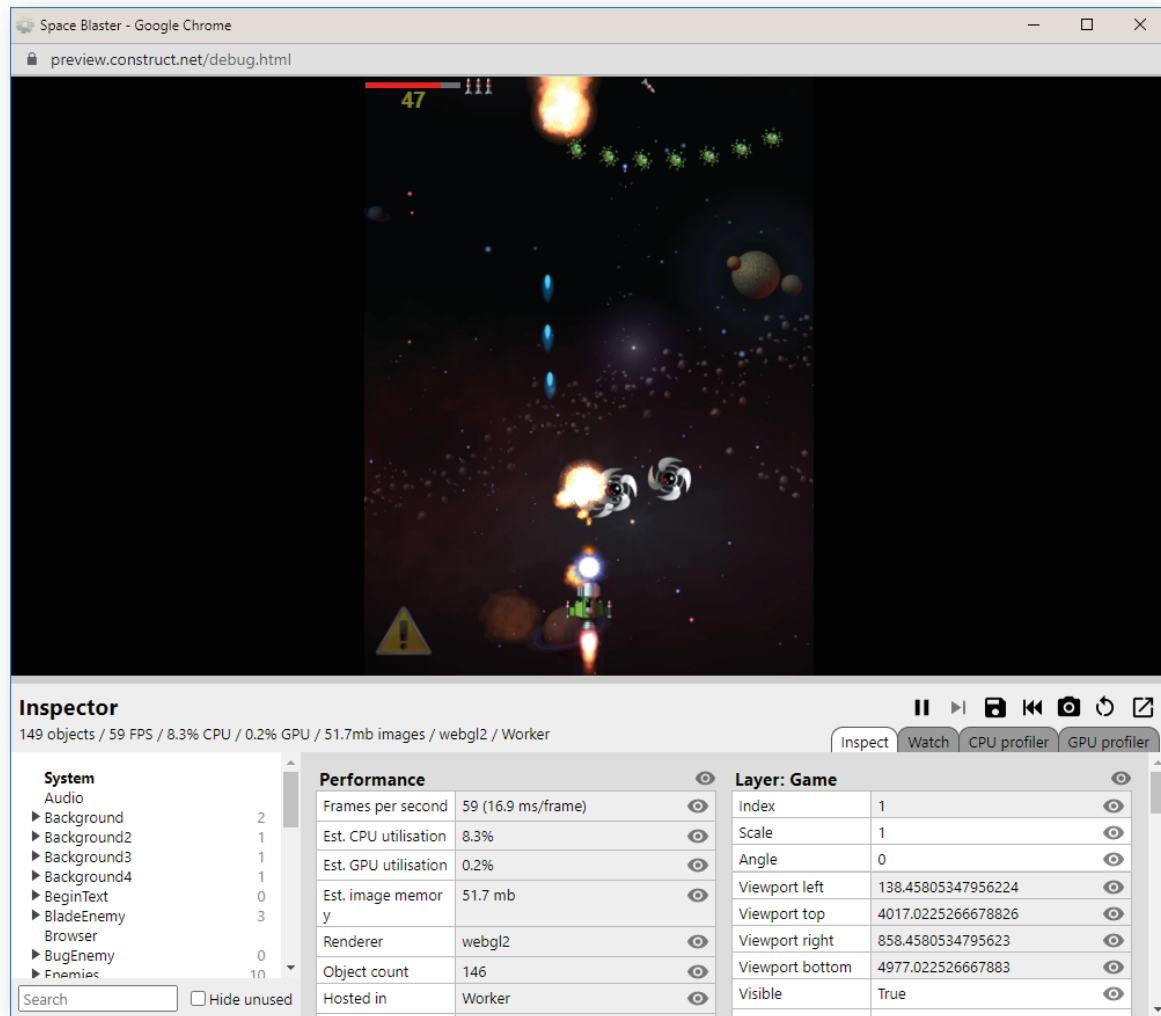
CHANGING UI MODE

Some devices have sufficiently large enough screens to use Construct 3 in desktop mode. However, depending on the device, C3 may still use the mobile UI intended for touchscreens. You can make Construct 3 switch to desktop mode by changing the UI mode to Desktop under Construct's settings. This will always load Construct 3 using the full desktop UI. You can change this at any time, but you'll need to reload C3 for it to take effect.

ADVANCED FEATURES

THE DEBUGGER

Construct 3 contains a full debugging tool to ensure indie devs have the tools necessary to test their games thoroughly. However, it's also a great tool to encourage students to get into the habit of regular testing and iterative development.



The debugger has four tabs: Inspect, Watch, CPU profiler and GPU profiler. The free edition of Construct 3 has access to Inspect, but if you want to use the Watch or profiler tabs with more advanced students, you'll need the full edition.

You can open the debugger from the Preview menu or by pressing Ctrl/Cmd + F5 and it will default to the Inspect tab. The Inspect tab is divided into two sections. On the left appears a list of all the object types in the project, including the System object. On the right is a list of tables of values relating to the selected object, like the Properties Bar.

The Watch tab shows you select value tables that you can select in the Inspect tab. Next to each value under Inspect is an eye icon, clicking this will add that value to the Watch tab so you can create your own views to keep an eye on specific parts of a project.

The CPU profiler tab provides a more detailed breakdown of the estimated CPU usage. The project must be running continuously for the profiler to be able to collect and display information. It then displays a breakdown of the estimated CPU time spent in each part of the project logic. It updates once a second and the values shown are for the previous second only.

The GPU profiler tab provides a more detailed breakdown of the estimated GPU usage. This covers work done to render the project's graphics, which is typically done on separate hardware (the Graphics Processing Unit, or GPU). The project must be running continuously for the profiler to be able to collect and display information. It then displays a breakdown of the estimated GPU time spent on each layer. It updates once a second and the values shown are for the previous second only.

Remember that both profiler tabs provide estimates only! For more information on the debugger and its profilers, [please check the Construct 3 manual](#).

FILE EDITORS

If you're using the array or dictionary data structures, the file editors for each data structure that are available in the paid version of Construct make creating and editing these structures substantially easier.

The Array Editor

The Array object stores lists of values (numbers or text), and it is analogous to arrays in traditional programming languages. You can build arrays in up to three dimensions. For example, a simple list of ten values would be a 10 x 1 x 1 array. Note that you should not set a size of 0 on any of the dimensions else the entire array becomes empty; it is correct to have a size of 1 on unused dimensions.

The Array editor allows editing an array data file for the Array object. The data you enter can be loaded at runtime by loading the project file into the Array object. It provides a visual way to set the initial data for an Array. The Array Editor appears when editing or adding an array data file (in JSON format) in the Project Bar.

Width	10	Height	10	Depth	1	Sheet	0	
	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4								
5								
6								
7								
8								
9								

Initially the array is sized to 1 x 1, which means there is just one cell available. Use the Width and Height settings to change how many cells are available. This determines how many rows and columns appear in the editor, allowing you to enter more data.

You can add values to cells simply by typing in them. Note that the type of a cell is determined automatically: if you enter a number (e.g. 4.2), the value is set to a number type, otherwise it is saved as a text string. You can also navigate between cells using Tab or Ctrl + Arrow keys on the keyboard.

The Array Editor only displays a 2D grid of numbers, like a spreadsheet. However, you can create a 3D array by setting the Depth greater than 1. For example, if the width, height, and depth are all 10, then there are 1000 elements in a 10 x 10 x 10 array.

To allow editing 3D arrays conveniently, you simply set which Z index you are editing, and edit values in that 2D plane of the array. Use the Sheet setting to move between each 2D "sheet" of the 3D array and edit them separately. This allows you to set all the values in a 3D array, while only editing a 2D section at a time.

Right-click a cell to open a context menu with options to clear, insert or delete rows and columns.

Dictionary Editor

The Dictionary object associates keys with values. Keys are strings, and their associated value can be a number or a string. It is a data storage object - it does not do any spell checking or language-specific features. Key names in the Dictionary object are always case sensitive. This means the key "SCORE" is considered different to the key "score".

The Dictionary editor allows editing a dictionary data file for the Dictionary object. The data you enter can be loaded at runtime by loading the project file into the Dictionary object. It provides a visual way to set the initial data for a Dictionary. The Dictionary Editor appears when editing or adding a dictionary data file (in JSON format) in the Project Bar.

Initially the dictionary has a single item, which means there is just one row available. Use the Size setting to change how many items are available. This determines how many rows appear in the editor, allowing you to enter more data. Each row represents a key in the dictionary (in the left column) and its associated value (in the right column).

Key	Value
GateKeeper	Warrior
ShopKeeper	Merchant
CastleGuard	Warrior
Castle Soldier	Warrior
Town Doctor	Healer
KeepGuard	Archer
Royal Adviser	Wizard

Like the Array Editor, you can type directly into the cells to add new values, navigate using Tab or Ctrl + Arrow keys on the keyboard and cell types are determined automatically.

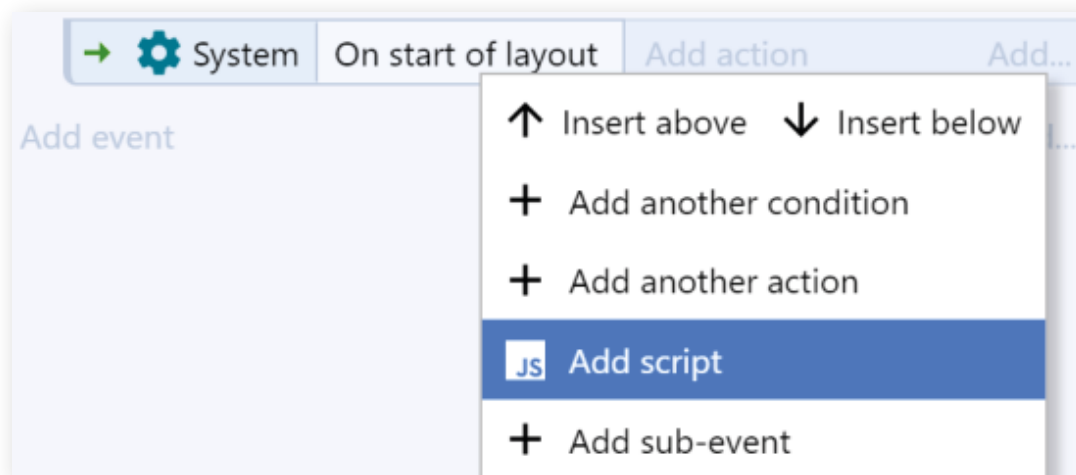
SCRIPTING

Construct supports writing JavaScript code in the place of actions and events, as well as in separate JavaScript files. This is a great way to learn to code, as well as taking advantage of the full power of the JavaScript programming language in your projects.

A good starting point for learning JavaScript is writing snippets of JavaScript in the event sheet.



JavaScript code can be added as an action. The code runs whenever the action is run, i.e., after all the event's conditions are met, and all previous actions have run. To add some code as an action, use the Add... menu to the right of the Add action link, and select Add script. Alternatively, you can use the J keyboard shortcut when an action is selected. A code editor will appear for you to enter the JavaScript code to run.



Alternatively, JavaScript code can be added as a block, in the same place as other event blocks appear. The code runs whenever an event block in that place would be run. You can add a script block using any of the following methods:

- Right-click in the margin of an existing event block and select Add Add script
- Click the Add... menu at the end of a group, or the end of the event sheet, and select Add script
- Use the J keyboard shortcut when an event block is selected (note if an action is selected, a script action will be inserted instead)

Remember that the event sheet runs all events in top-to-bottom order every tick, so a script in a block at the top level of an event sheet is run every tick (as if it was an action in an Every tick event). Often it is more useful to use script blocks as sub-events, which only run if their parent event block was true.

Of course, you can always ditch event blocks entirely and use JavaScript sheets to code your games. Script files can be added in the Scripts folder of the Project Bar.

Existing JavaScript files (.js) can also be imported using the Import scripts option instead. Once you add a script a code editor appears. The first script added will have some default code added to help you get started.

There is a lot to talk about when it comes to Scripting, but that is beyond the scope of this book. Be sure to check out the Construct website for the [full Scripting section of the manual](#), our thirteen-part [Learn JavaScript Course](#) and a [QuickStart Guide](#) for those comfortable with JavaScript already.

Plus, the Example Browser also has a section for Scripting-based examples for you to use!

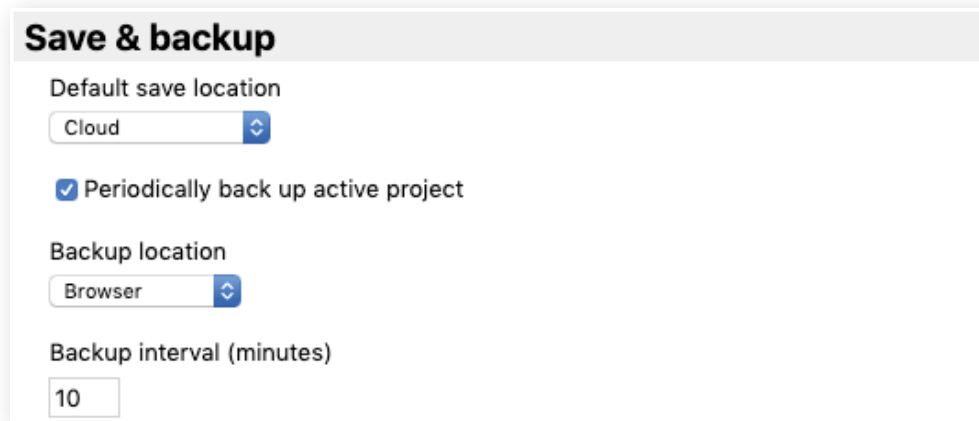


SAVING AND SHARING

CLOUD SAVE	66
USING LOCAL FILES	67
USING PROJECT FOLDERS	67
SAVE TO LOCAL BROWSER	67
SHARING YOUR PROJECTS	68
SHARING WITH SUBSCRIPTION ADMIN	68

In Construct 3, there are three main places you can save your work, each of which we will cover in more detail shortly. By default, pressing Save on a new project will save with the Cloud Save option. You can select a different option, using the Save As menu.

In Construct 3's settings, there are a few options for Saving and Backups. Here you can change Construct's default saving option if you don't want to always save to the cloud.



You can also configure C3's built-in backup system from this part of the settings menu. You can choose whether you want Construct to make backups for you, as well as how frequently you want them made, and where you would like them saved.

We would recommend that you back up your projects regularly to avoid losing too much work when things do go wrong. We would also recommend having multiple backups in various places to protect against things like hard drive failure or having your cloud accounts hacked.

CLOUD SAVE

You can save your work to a cloud storage service, allowing you to access your work wherever you go. Since Construct 3 runs in the browser and can be used on any device, this is a great way to ensure you can carry on from where you left off no matter which device you end up using. Many cloud storage services also provide built-in backups and file histories, helping ensure your work is safe even in the face of disaster.

Construct 3 currently supports Google Drive, Dropbox and Microsoft OneDrive. The first time you select **Menu** → **Project** → **Save as** → **Cloud save**, a dialog will appear asking you to choose one of the supported services. When you choose one, you'll be prompted to log to your cloud storage account, so Construct 3 has permission to save and open files from your account. Once you've entered your details they will be remembered, so you can keep using Cloud Save without having to keep entering your details.

When you press Save with a Cloud Save project, Construct 3 will save your project and upload it to your cloud storage account. The upload will continue in the background showing the upload status in the corner of the window, allowing you to continue working on your project. Bear in mind you cannot save again until the upload completes.

Next time you use Construct 3, you can choose **Cloud open** under the Open menu to find your project again. It'll also appear in the Recent Projects section of the Start Page.

USING LOCAL FILES

For security reasons, browsers can't write files directly to your device. However, you can select **Menu** → **Project** → **Save as** → **Download a copy** to download your project as a local file. Construct 3 will ask if you want to change the downloaded filename; you can leave it empty to use the default. Construct 3 projects use the `.c3p` file extension.

Normally the file will go to your Downloads folder, but you may also be prompted to save to a different location depending on the browser – you can alter this setting in most major browsers. You can also choose a new default download location in most browsers too. Alternatively, you can usually drag-and-drop the resulting file directly out of the browser, such as from Google Chrome's downloads footer section.

USING PROJECT FOLDERS

Where browsers support local files, they also allow the option to use project folders. These work similarly to saving local files, but instead of choosing a file, you select a folder to save to. Construct then saves the entire project as separate files within this folder. Be sure to choose an empty folder to avoid ending up with a confusing mix of files.

This option is good for very large projects, since saves are faster, as it only has to update the changed files in the folder, rather than generate an entire new `.c3p` file. It is also a good option to use with source control tools like GitHub, since you can track changes to individual text-based files – for a guide on that see the tutorial [How to collaborate on Construct projects with GitHub](#).

This option can also be useful if you work with lots of JavaScript files in a Construct project and want to use an external editor with them. When saving as a folder project, new options will appear in the menu when right-clicking the script folder in the Project Bar. These options allow you to reload all script files from the project folder again either as a one off (also by pressing F9), or automatically every time the project is previewed. Note this reloading cannot be undone, so make sure you always make edits in the same place, as alterations within Construct will be overwritten when reloading.



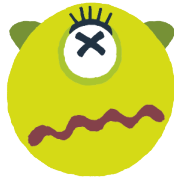
Construct's `.c3p` files are actually just a zipped folder project, with the extension `.zip` replaced with `.c3p`. You can convert a `.c3p` file to a folder project by renaming `.c3p` to `.zip` and extracting it. Similarly, you can convert a folder project to a `.c3p` by zipping it and renaming `.zip` to `.c3p`.

SAVE TO LOCAL BROWSER

If saving local files is not supported, Construct provides an option to save projects to the local browser's storage instead. This storage is unique to both the specific device and browser. For example, if you save a project to browser storage on a specific laptop with Chrome, you can only find it again by using the same browser (Chrome) on the same device (that specific laptop).

Construct will ask for permission to use persistent storage the first time you use this option, to ensure the browser won't automatically delete your data. Be aware that browsers sometimes also have storage limits. You can also check the status of the

persistent storage permission, as well as how much space the browser is allowed to use and how much it is using, in the About dialog.



If you use this option, be very careful about clearing your browser data. If you choose the wrong option while clearing browser data, you could still erase all your projects saved to browser storage. For this reason, using a different save option where possible is recommended.

SHARING YOUR PROJECTS

Sharing your project or game is a great way for your students to gain feedback on their work, either from their peers or from you as their teacher! It doesn't hurt to share projects more often than at the final deadline for extra opinions.

The easiest and quickest way to share your project, especially during development, is using Remote Preview. We'll cover this feature in more detail in the Testing Your Game chapter, but a quick explainer for Remote Preview – it allows you to preview your project on a different device by either scanning a QR Code or visiting a specific URL.

But Remote Preview isn't always the best option (it's also not available in the free edition.) Let's say you've hit a wall with your development, and you need a bit of help solving a problem. You might need to send someone your project for them to have a look at. In this case, you can either:

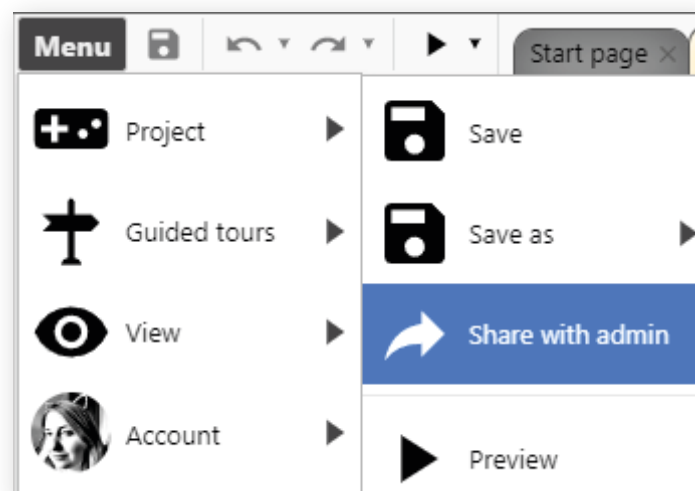
- Use Cloud Save, and use the storage service's sharing features
- Use the *Download a copy* option to save a local file and share that.

If you want people to play your finished project, you should export it, which produces a playable game ready for publishing on a given platform. Exporting projects will be covered in more depth in a later chapter.

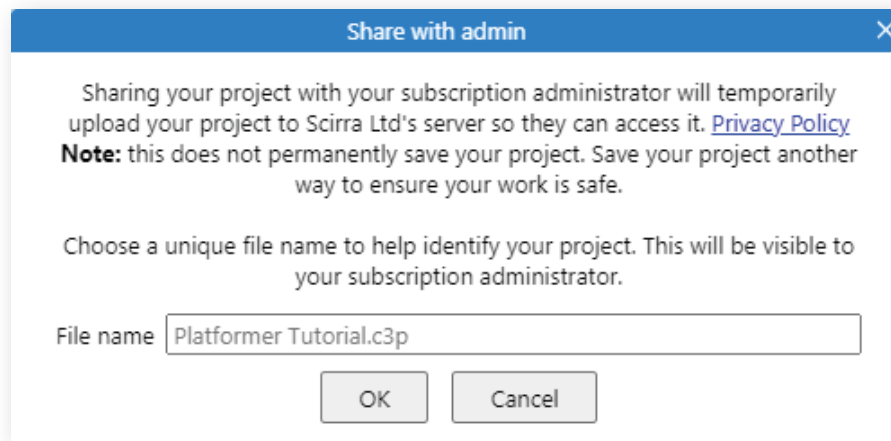
SHARING WITH SUBSCRIPTION ADMIN

This only appears for two kinds of accounts:

- When using access codes
- When using an account created for a seat as part of an education subscription



In both cases, choosing Share with admin will upload the project to a server operated by Scirra. It will then be made available to the subscription administrator. By visiting the same Plan Dashboard, we mentioned in the chapter on Access Codes, you can find all the projects currently shared with that admin.



You'll need to make sure your students name their files in a manner that means you'll know whose work is whose when you come to check it later. The system only shows either the Access Code or the Construct username for the project submitted.

From your Plan Management dashboard, you have a tab that shows any projects that have been uploaded to you via the Share with Admin feature. You can also see when the project was uploaded and how long before it expires and is removed from the server – projects are not permanently stored on our system with this feature. Click the Download button on the right-hand side of the screen to download it to your own system.

Your Construct 3 Plan

SHARED PROJECTS

Filename	Date Shared	Shared By	Expiry	Download
3DCastleMaze_StudentX.c3p	3 Oct, 2023 at 08:38	Access Code E772W9A2 IP address 5.39.180.50	3 days 6 Oct, 2023 at 08:38	DOWNLOAD NOW 144.81 KB 0 total downloads

OVERVIEW

- Overview
- Your Organisation
- Your Seats 20
- Access Codes 1
- Sub Users 0
- Activity Logs
- Access Logs

SHARED PROJECTS

- Shared Projects 1

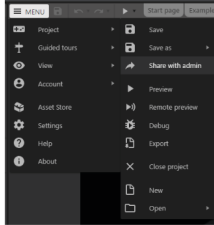
PAYMENTS

- Invoices

OTHER

- Extend Plan
- Get Help

LAUNCH CONSTRUCT 3



Sharing Projects

Anyone who is using an access code for this plan or who you've assigned a seat to can select **Menu > Projects > Share with Admin**. This uploads their project *temporarily* and will make the project available to download from this page.

The shared projects will automatically be deleted after around 72 hours. This feature should only be used for convenience and **not** as a way to store, save or backup data. All downloads are at your own risk.



MAKING YOUR FIRST GAME

PART 1 - GETTING STARTED	71
Create a New Project	71
Changing Layout Properties.....	71
Adding Your First Object.....	72
PART 2 - CREATING PLATFORMS	75
Using a Tilemap for Platforms.....	75
PART 3 - ADDING THE PLAYER	78
Adding the Player Sprite	78
Setting Up Origin and Image Points.....	78
Animations and their Properties	79
PART 4 - ADDING BEHAVIORS	80
About Behaviors.....	80
Creating a Helper Sprite.....	80
The Behaviors Dialog	81
PART 5 - YOUR FIRST EVENTS	83
Keeping our Sprites Together.....	83
PART 6 - IMPROVING PLAYER ANIMATIONS	86
Mirroring Sprites.....	86
Jumping and Idling Animations	87
Setting up Events for Multiple Animations.....	87
PART 7 - CREATING AN ENEMY	90
Adding the Enemy.....	90
Creating Enemy Movement.....	92
Interactions with the Player.....	94
PART 8 - ADDING COLLECTIBLES	96
Create the Collectible Item	96
Collecting and Scoring.....	97

PART 1 - GETTING STARTED

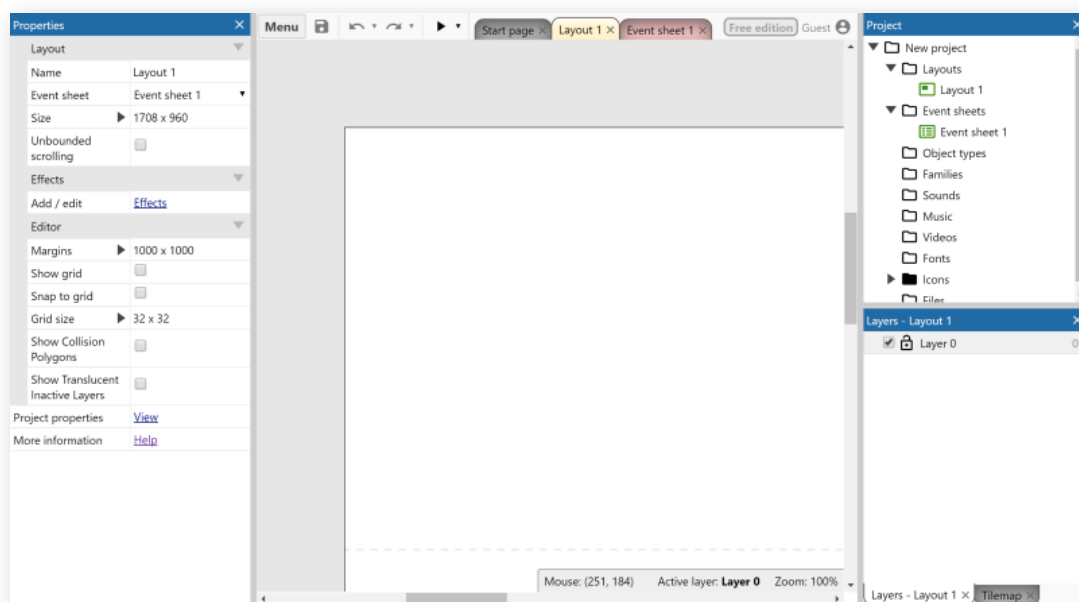
Let's get started creating your first game. We'll be making a platformer game following one of our beginner's tutorials.

You will also need some assets for this game. You can make your own using the animations editor or download something from an asset website - [OpenGameArt](#) and [Kenney.nl](#) are good starting points! For this project, we'll be using a pack from [Kenney.nl](#) - the **Platformer Pack Redux**.

We're going to create this game from scratch, so the first thing to do is load up Construct 3 - simply head to [editor.construct.net](#) and you should be greeted by the Start Page. Now, we're ready to get started.

CREATE A NEW PROJECT

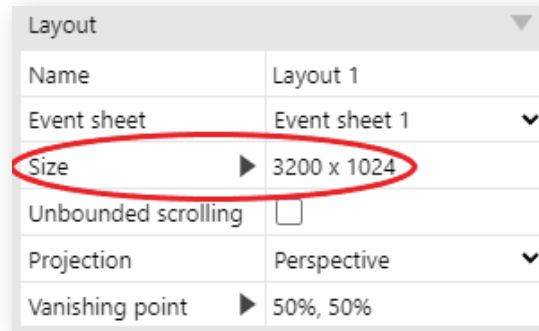
Click the New project button. A dialog will appear asking for some details. You don't have to change anything, but you can type in a name for your project if you like (how about *My super awesome game?*) Click Create, and you should see a new empty project, something like this.



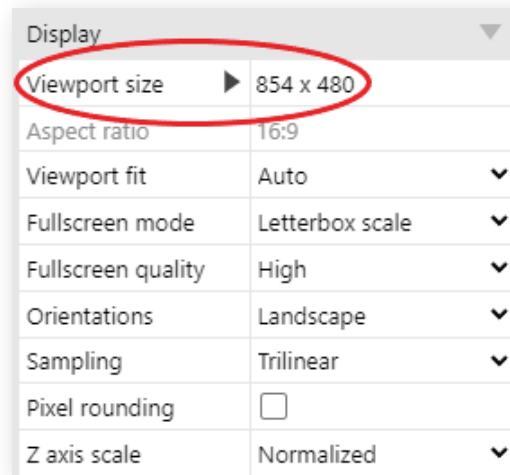
CHANGING LAYOUT PROPERTIES

As explained earlier in this book, properties for various aspects of your project can be adjusted in the Properties Bar - usually situated on the left-hand side of the Layout View. We're going to make use of that now as we need to make our layout bigger than the standard size. You can do this when you create the project, but it's helpful to outline that you can also do this at any point via the Properties Bar.

Make sure you have Layout 1 selected in the Project Bar and its properties should appear in the Properties Bar. We can now change the size of the layout to 3200x1024. This should leave plenty of space to create a platforming level.



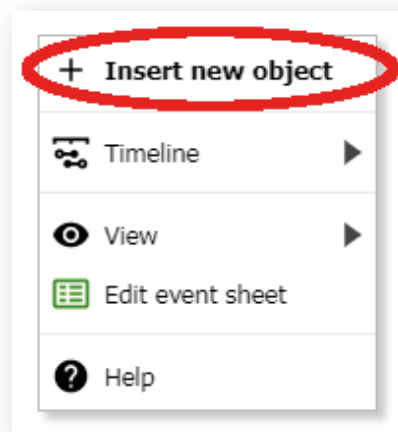
Since we've changed the size of the layout, it would also make sense to change the size of the viewport so we can see a bit more of the level. The Viewport is a project property, so make sure you've got your project selected in the Project Bar and then change the Viewport size in the Properties Bar to 854x480.



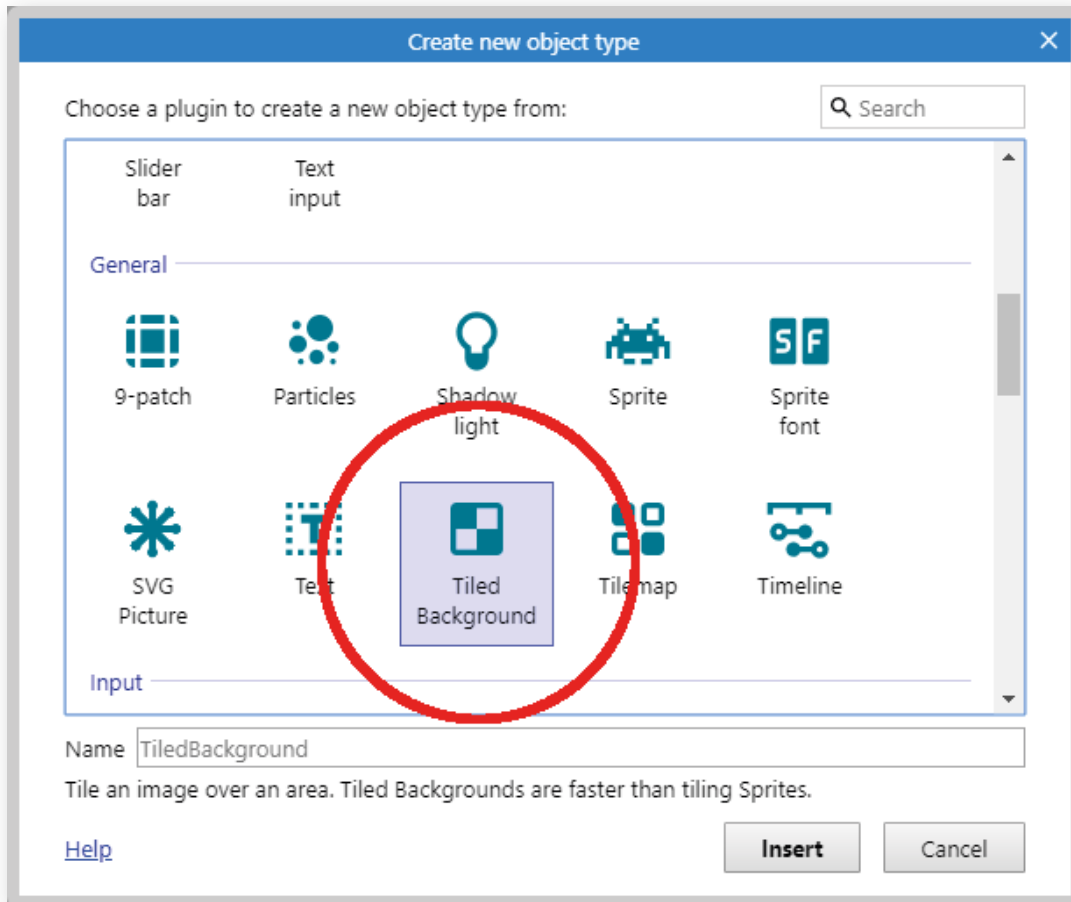
ADDING YOUR FIRST OBJECT

Your first object will be a Tiled Background - these are great for quickly creating repeating images across a space.

There are a couple of ways to add objects in Construct, the simplest is to double click a space in the layout. If your layout is full, you can also right-click and select Insert new object.



Once the Create new object dialog appears, double click the Tiled Background object.



The mouse will turn in to a crosshair for you to indicate where to place the object. Click somewhere near the middle of the layout. The image editor now opens for you to draw or import the image to tile. Now, we can import one of the backgrounds from the asset pack we mentioned earlier. In the **PNG** folder of the pack, you should find a folder called Backgrounds. In here are several options, we chose to use blue_grass. Click the image you want to use, and it'll be opened in the image editor.



Close the image editor by clicking the X in the top-right. Now you should see your tiled background object in the layout. Let's resize it to cover the entire layout. Make sure it's selected, then the Properties Bar on the left should show all the settings for the object, including its size and position. Set its position to 0, 0 (the top left of the layout), and its size to 3200 x 1024. This should give a nice extended version of the chosen background image.

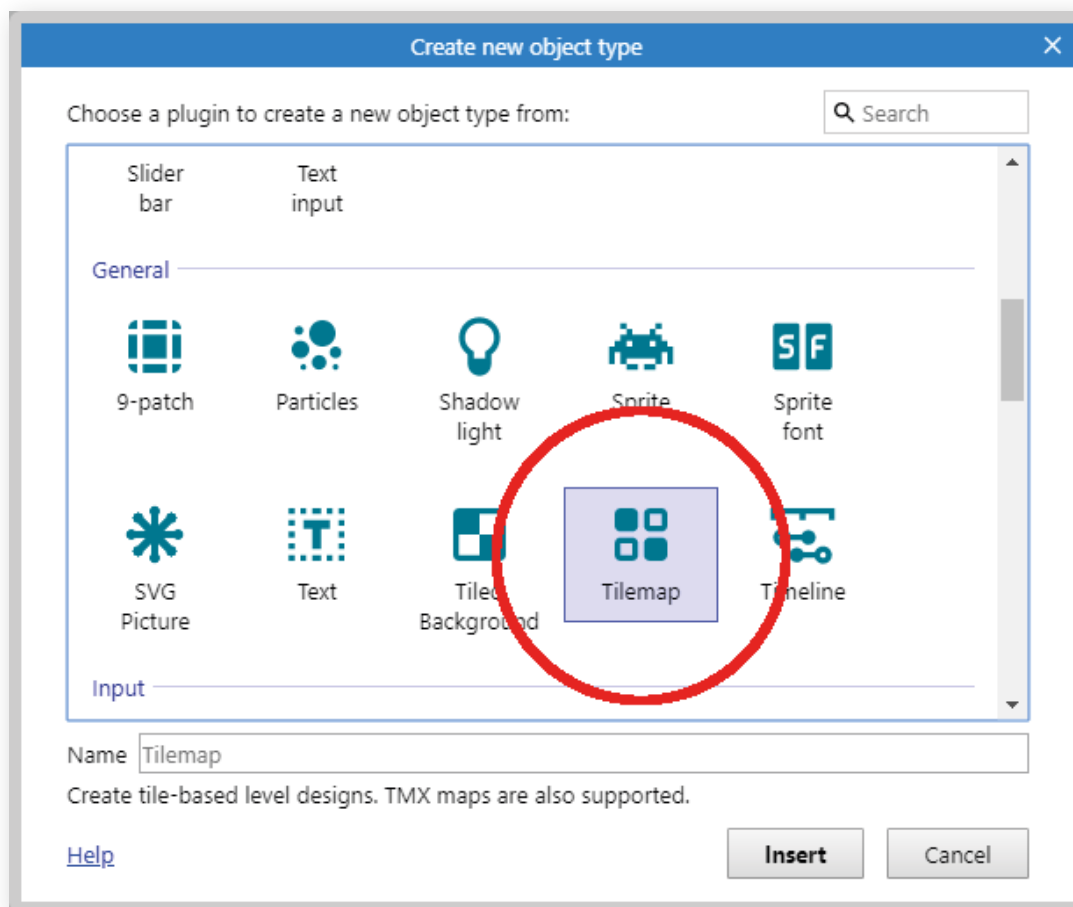
Before we continue, the tiled background should be locked. As we create and move around objects on top of it, it's easy to accidentally select, change or move the background. Since we don't need to change the background anymore, locking it makes it unselectable, so it won't get in the way. To lock it, right-click on the tiled background and select Lock ► Lock selection. (If you do want to change it later, simply right-click and select Lock ► Unlock all.)

PART 2 – CREATING PLATFORMS

USING A TILEMAP FOR PLATFORMS

The next stage for our project is to start building a level and we're going to create the platforms using a Tilemap. The tilemap object is ideal for this project as we can quickly draw in a bunch of platforms using a tileset – the image that the tilemap object draws from.

Double-click a space in the layout. This will bring up the Create New Object Type dialog again. This time, double-click the Tilemap object to insert it.



As with the Tiled Background, the mouse cursor turns into a crosshair, and it doesn't matter where you are clicking in your layout.

The Tilemap Editor opens. It contains a default tileset, but we will want to use our own tileset. This is the same process as adding the image we used for the Tiled Background. Navigate to where your assets are saved, and then in the Spritesheets folder, look for the file `spritesheet_ground_64px`. You don't need to do any editing of this image, so the editor can be closed.



By default, the tilemap object will cover the whole of the layout which is perfect for this project. It's also automatically selected once you close the image editor which is handy as we'll need to change some of the properties of the object.

If you have deselected the tilemap object, click on it in the Project Bar and its properties will then appear in the Properties Bar. When you use a tileset image with the tilemap object, you may need to adjust properties like Tile Width/Height or Tile Spacing to ensure you can use those tiles correctly.

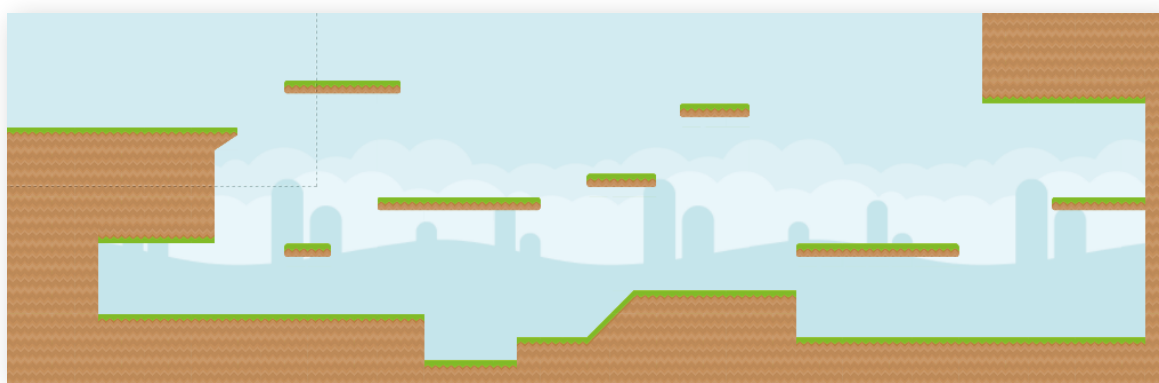
The default tile size for the inbuilt tileset is 32x32px and as you might have guessed from the filename, our tiles are 64x64px, so we need to adjust the tile properties to suit the new tileset. Change Tile Width and Tile Height to 64 in the properties bar and our new tileset is ready to use!

Properties	
Image	Edit
Initially visible	<input checked="" type="checkbox"/>
Tile width	64
Tile height	64
Tile X offset	0
Tile Y offset	0
Tile X spacing	0
Tile Y spacing	0

This is also a good opportunity to talk about renaming objects. Construct will name things by default as you add objects, but when you need multiple sprites this can start to get very confusing - fifty objects name Sprite1, Sprite2 etc. gets messy very quickly! You can rename objects either using the Properties Bar, or by right-clicking them in the Project Bar and selecting Rename. For now, let's rename our Tilemap object to Platforms. That way, if later on you decide you want an extra tilemap then you'll know which is which.

Next, we need to actually build a level!

The tools for tiling are kept in the Tilemap Bar which, by default, is tabbed in the same section as the Layers Bar in the bottom right-hand corner of the editor. You can read more about what each of the tools does in the Manual, but to draw out our level, we need the Pencil tile tool - this allows you to draw tiles using the mouse. Click the tile you want to draw with in the Tilemap Bar and then you can draw with it in the layout. Try playing around with the various tools and building yourself some platforms!



When you're done drawing your level, make sure to select the Select Tool (the one that looks like a mouse cursor) to stop drawing tiles. Once you're done editing the tilemap, be sure to lock it in the same way you did the Tiled Background. Now we can move into the next stage of building the game!

PART 3 - ADDING THE PLAYER

ADDING THE PLAYER SPRITE

The next thing to add to our game is the Player. We'll use a Sprite Object for this - you've added a couple of objects now, so the process should be familiar.

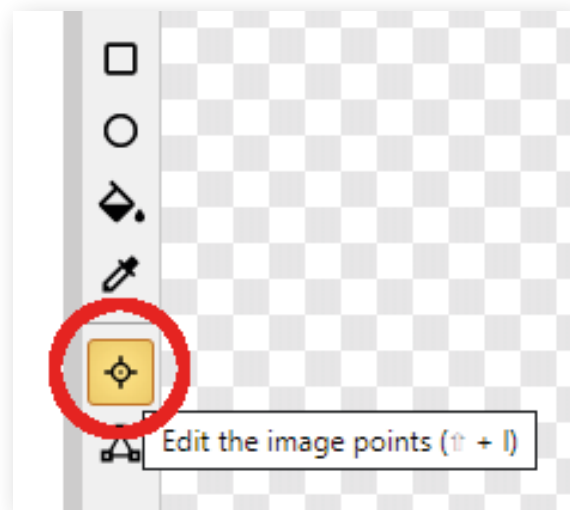
When the Animations Editor opens, we're going to import several animations rather than the single images we used for the Tiled Background and Tilemap.

The asset pack has several options for the Player Sprite, each in their own folders - in the PNG folder, you'll find a folder called Players and we're using the 128x256 sized sprites. Pick your favourite character and Open the two 'walk' frames - the process is the same, but if you select multiple images in your file browser, they'll be added to the animation as individual animation frames.

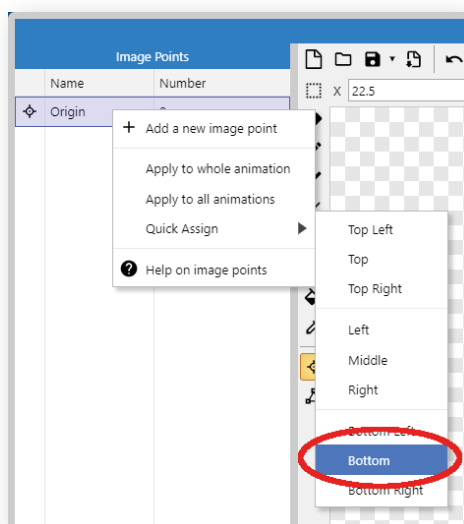
SETTING UP ORIGIN AND IMAGE POINTS

When you import a new animation frame, Construct will guess a collision polygon for it and will apply the origin point to the centre of the image. The best location for your origin point will vary depending on how your animations are set up. Often, in platformers, it's best to have the origin by the player's feet. This means if the animation changes height as it plays, they grow upwards, rather than into the floor.

To set the origin, click the Edit the image points button in the toolbar.



You should notice a crosshair-style icon appear on the player - this is the origin. You can click anywhere in the animation frame to change it, or you can set it to pre-defined positions. We want it bottom-middle aligned and we can either do this by right-clicking the Origin in the Image Points pane and selecting the Quick Assign > Bottom option, or by hitting 2 on the num pad (if num lock is on).



It can be a hassle to do this for every frame, but luckily there's another short-cut: in the Image points pane to the left, right click the line Origin and click Apply to whole animation.

Bingo! The origin should be set on every animation frame.

ANIMATIONS AND THEIR PROPERTIES

We're going to do a bit of admin and tweaking for our animation next. In the same way that you renamed sprites, you can rename animations. Right-click **Animation 1** and rename it **Walk**.

You may have noticed that when you selected the animation in the animations list, its properties appeared underneath. This acts the same as the Properties Bar in the main editor and you can also rename your animations from here.

You can also change the general properties of your animation from here. You can also preview your animations at any time through the same right-click menu you used to rename your animation. Try changing some of the properties for the **Walk** animation and see what happens.

Once you're done tinkering, the final properties we're going to use for this project are a Speed of 5 and Loop enabled.

Animation Properties	
Name	Walk
Speed	5
Loop	<input checked="" type="checkbox"/>
Repeat Count	1
Repeat To	0
Ping Pong	<input type="checkbox"/>
More Information	Help

PART 4 - ADDING BEHAVIORS

ABOUT BEHAVIORS

Construct comes with lots of behaviors. These make your objects work in pre-defined ways, which often saves loads of time. It's possible to do everything the behaviors can using the event system, but it is often difficult and time consuming to do. That's why behaviors are really handy to get your game up and running quickly!

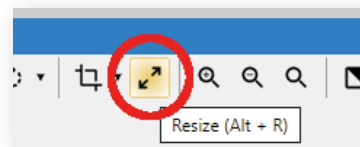
The Platform behavior can take care of the complexities of platform movement for us. However, there's one important tip for using it: the behavior should be applied to an invisible rectangle object, and the player positioned on top. The Platform behavior works much better if the object with the behavior doesn't animate, since changing animation frame can leave the object partly sticking into a wall which can confuse the Platform behavior. Also, it avoids silly collision situations like your player hanging off a ledge by their nose or something they're holding.

CREATING A HELPER SPRITE

To ensure that the animations don't get in the way of the Platform behavior, we're going to create an invisible helper sprite and give it the correct behavior.

The first thing we need to do is create a new Sprite object. Within the Animations Editor, we can resize animations which will affect how they appear in the layout.

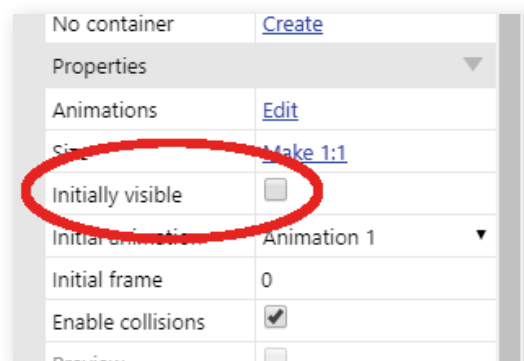
The top toolbar in the Animations Editor contains the resize tool, and if you click it then the resize dialogue will appear. Our Player sprite's dimensions in the layout are 64x256 and its Helper sprite should be set to 64x128 - it doesn't need the same height as the sprite itself.



The appearance of the Helper sprite doesn't matter as we'll be hiding it later on, so for now, select the Fill tool and pick a colour for the Helper sprite. When you're done, close the Animations Editor to finish adding the object.

As this sprite will be used alongside the player object, let's rename it PlayerBox - a box to handle the collisions and movements of the player. However, we don't want the helper sprite to be visible during the game so in the Properties Bar (ensuring PlayerBox is selected) we can uncheck **Initially Visible** and it'll be kept invisible unless we specify otherwise in the game's events.

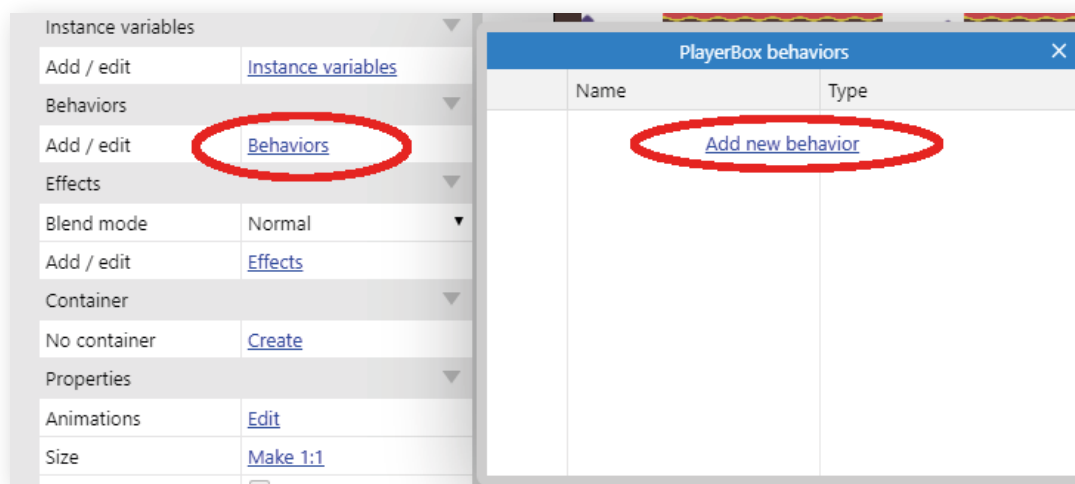
Next, we'll need to ensure our PlayerBox can move.



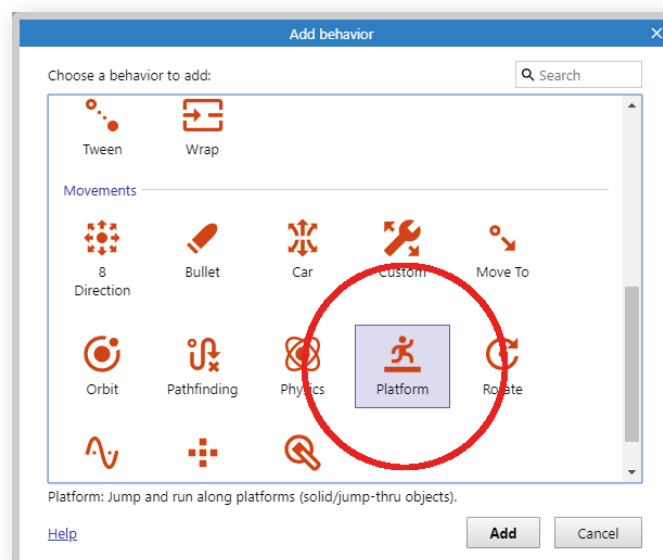
THE BEHAVIORS DIALOG

There are a couple of ways to add a behavior to an object. You can right-click the object you want to add the behavior to and select Behavior under the Add sub-menu, or you can select the object and use the Behaviors section of the Properties Bar.

This section of the Properties Bar will show any behaviors an object has associated with it as well as the properties for those behaviors that can be adjusted. For now, as our object has no behaviors, it's blank, save for the behaviors link. If we click this, the behavior dialog will open.



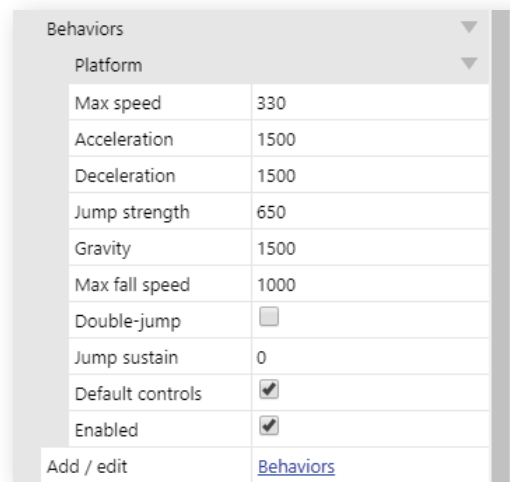
The first behavior that needs to be added is the Platform behavior, which as previously mentioned, will handle all the movement for our Player character. You can double click any behavior in the Add behavior dialog to add it to an object.



The PlayerBox object needs another behavior adding to it – Scroll To. This behavior will make sure the camera follows the player and is the quickest way to implement a basic camera system. Repeat the above steps to add the Scroll To behavior to the PlayerBox object.

Once you've added both behaviors, you should see their properties now appear in the Properties Bar. Now you can tweak various values for the Platformer movement to make it work better for your game – make sure your player moves at the right speed and has enough jump strength to make sure you can reach the platforms you've made! You can also enable Double Jump to help reach trickier platforms!

While we're talking about behaviors, the Tilemap object needs one too. As it's going to be used for creating the level, it would make sense to ensure that the player can't fall through the platforms. A quick way to do this is to add the Solid behavior to the Tilemap object. Solid does exactly what it says on the tin and makes the object it's applied to act like a Solid – in this case, our PlayerBox won't be able to pass through it.



Remember that if you've locked the Tilemap object as recommended earlier in this tutorial, you'll need to unlock it, or its layer to be able to edit the object's behaviors. Once you've selected the Tilemap object, go ahead and add the Solid behavior to it. It's worth locking your Platforms layer or the Tilemap object again once you're done adding behaviors.

At this point, you can preview the project, but not a lot will be happening. The Player object will remain stationary, and while the PlayerBox object might be moving, you can't see it as it's been set to Initially Invisible. Ideally, what we need to do is make sure the Player object moves along with the PlayerBox so that it looks like our character is moving rather than just walking on the spot. To do that, we're going to need to start adding some Events.

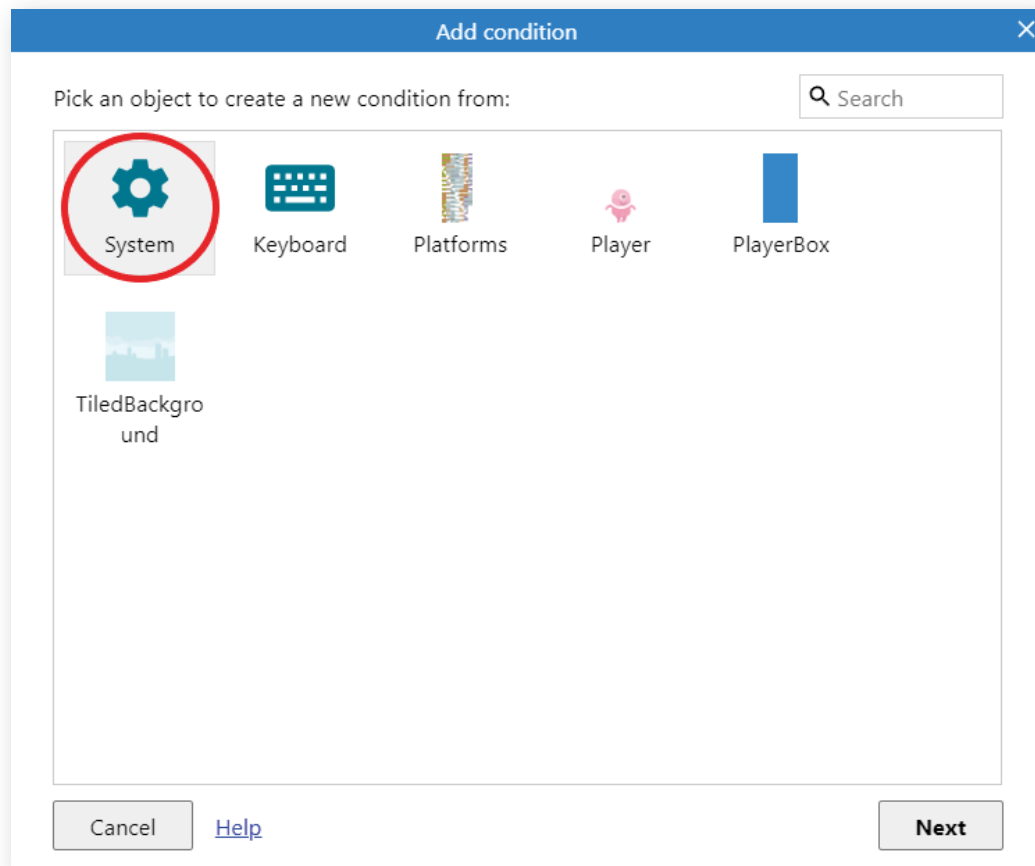
PART 5 – YOUR FIRST EVENTS

KEEPING OUR SPRITES TOGETHER

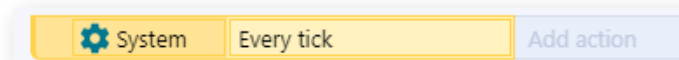
Head over to the Event Sheet by selecting the Event Sheet 1 tab on the main toolbar. If you need a reminder of how events work, or what things are in the Event Sheet view, take a look at the chapters on Construct's Event System or the User Interface Chapter on C3's Event Sheet View.

There are several ways to make sure your Player and PlayerBox objects stay together, but we're going to achieve this by manually setting the Player's position every tick. Every Tick is essentially a way to say 'do this all the time' but it'll fire every tick which is roughly 60 times per second.

Right-click in the empty event sheet to add a new event (the Event Sheet view chapter covers more methods of adding events) – this will open the Add Condition dialog. Every Tick is a System condition, so you'll need to double-click the System icon in the dialog.

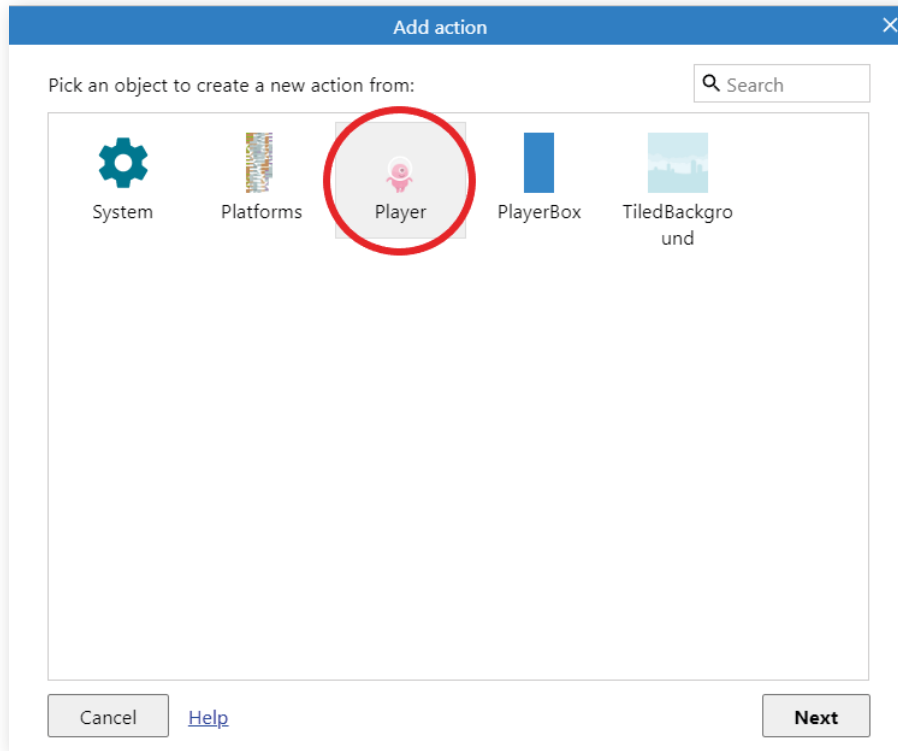


This brings up a list of all the available System conditions and there are quite a few! (Take a look at the Construct 3 manual if you want to learn more about System Conditions, Actions and Expressions!) You can either scroll through the list or use the search bar at the top to find 'Every tick'. Once you've found it, double-click it and that will create an event block containing only the *Every Tick* condition.

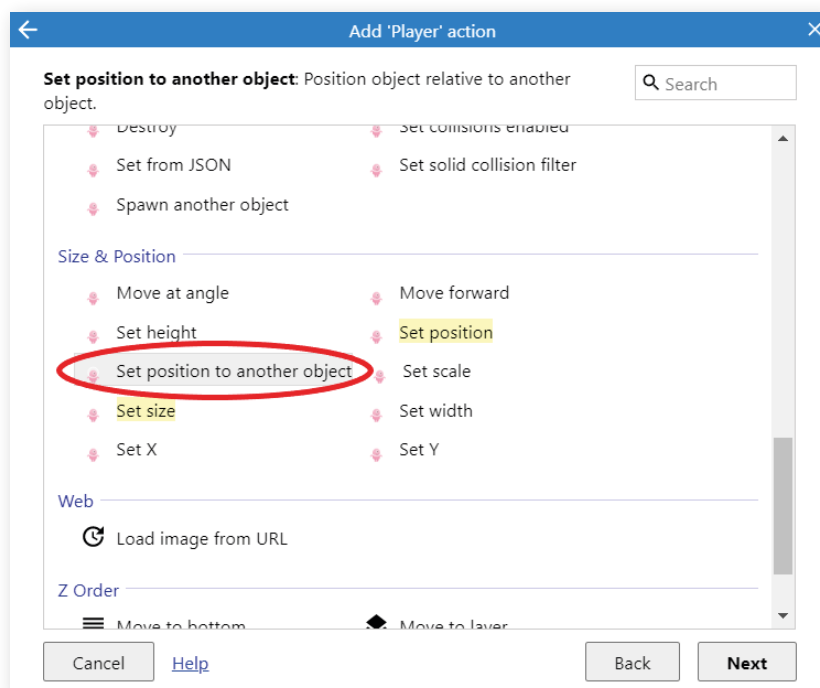


Now we need to build the rest of the Event Block – start by clicking the Add Action button in the block. This opens the Add Action dialog.

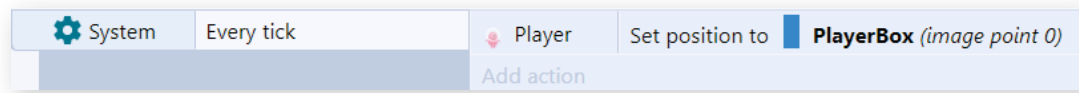
As the point of this event block is to set the Player object’s position, the action needs to be Player specific – so double-click the Player icon in the dialog. As with the previous step, this will bring up a list of all the available actions for the Player object.



Scroll or search to find Set Position to Another Object, select it and in the following dialog, make sure to pick PlayerBox and leave the Image Point set to 0 (this means that the Player object will have its position set to the Origin of the PlayerBox object.)



Press Done when you're finished, and you've built your first event! It should look like this:



Now is a good time to preview the project – you should see the Player character running and jumping around as you control it with the arrow keys. We could make it look better though – making sure the player faces left when moving in that direction and giving it movement animations are a good starting point!

This is also a good time to find out if your platforms are in the right places so your player can navigate the level. Try playing your level and making any adjustments you need to ensure that the whole level can be reached by the player, either by editing your tilemap, the parameters of the Platform behavior, or both! Depending on which tiles in the tileset you've used, you may need to change the collision polygons of certain tiles – you can do this by double-clicking a tile in the Tilemap Bar. Don't forget, if you want to edit your tilemap, you'll need to unlock it, make the changes and then lock it again.

PART 6 - IMPROVING PLAYER ANIMATIONS

MIRRORING SPRITES

Up until this point, the default controls from the Platform behavior have been sufficient for this project – however, in order to get the player to face in their direction of travel we’re going to need to add events that occur when a key is pressed. To do this, the Keyboard object needs to be added to the project.

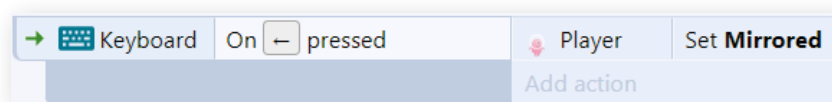
The Keyboard object is added in the same way as the Sprite and Tilemap objects however, once added you’ll no longer find it in the Create New Object dialog. Input objects along with several other objects in Construct can only be added once but will then be available to the entire project.

To get the sprite facing the correct way when they move in a direction, you could have separate animations for either direction. However, creating all the extras can be time consuming and for the purposes of this example, not particularly useful. Instead, we can Mirror our existing sprites.

Head back to the event sheet and we can start to build the event blocks needed for this functionality.

Following the same process as earlier, we need to add a new event, this time the Condition will be On key pressed, which is a Keyboard condition. Select the Keyboard object in the New Condition dialog to find On key pressed in its list of conditions and select it. Construct will need to know which key you want to use for this event, so clicking the <click to choose> button will activate a key detector – hitting the left arrow key will pick it as the trigger key for this event. Once you’re finished, you can press done to create the new block.

The Action needed for this is Set Mirrored. As we want to mirror the Player sprite, we need to pick that object in the New Action dialog. Scroll or search to find Set Mirrored and select it. Another dialog will open with some parameters for the action – for this event, you can leave the State and Mirrored. Click done, and you should have a new complete event block.

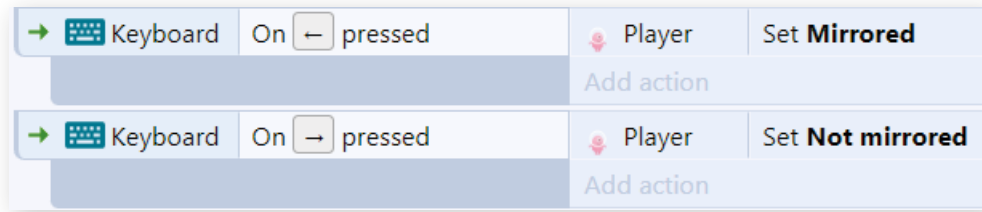


Once you’ve created an event block, you can copy and paste them as you need to and that can save you some time. To finish the mirroring logic, we need a second event very similar to the one just created. So, either right-click your event block to open the menu, or select the event and hit Ctrl/Cmd + C to copy the event. Then, either right-click in a blank space on the event sheet and select Paste, or use the Ctrl/Cmd + V shortcut to paste the event and you should now have a duplicate event block.

We can now edit this duplicate block to make some adjustments. Double-clicking the On key pressed Condition will reopen its key detector dialog and by clicking the Left Arrow box, you can pick a new key – this time you’ll need the Right Arrow key. Press done once you’ve made the change and the condition will be updated.

You can do the same with the Set Mirrored action. Double-click it and this time you’ll want to change the State to Not Mirrored.

Once you've made the changes, you should have these two event blocks:



If you preview the game, the Player sprite should now face the correct direction when it moves either left or right.

JUMPING AND IDLING ANIMATIONS

So, we've got the player facing the right way, but it could still look better. To improve it further, we're going to add two more Animations.

The first extra animation to add will be a Jump animation - this is done through the Animations Editor. Double-clicking the Player object either on the layout or in the Project Bar will open the Animations Editor so we can edit the artwork.

You should have a Walk animation already and if you right-click in the Animations Pane, you can add a new animation - this one should be called Jump.

Head back to the folder where you've saved the Player assets and as you did with the Walk animation, add the Jump frame for your chosen player to the Animations Editor. Again, assign the Origin Point to the bottom of the sprite.

The second new animation will be an Idle animation. As you just did, add a new animation, name it Idle and add the Idle frame for your chosen player. Set the origin point to the bottom of the sprite and you're done.

Because these animations only contain one frame, we don't need to make any changes to the animation properties.

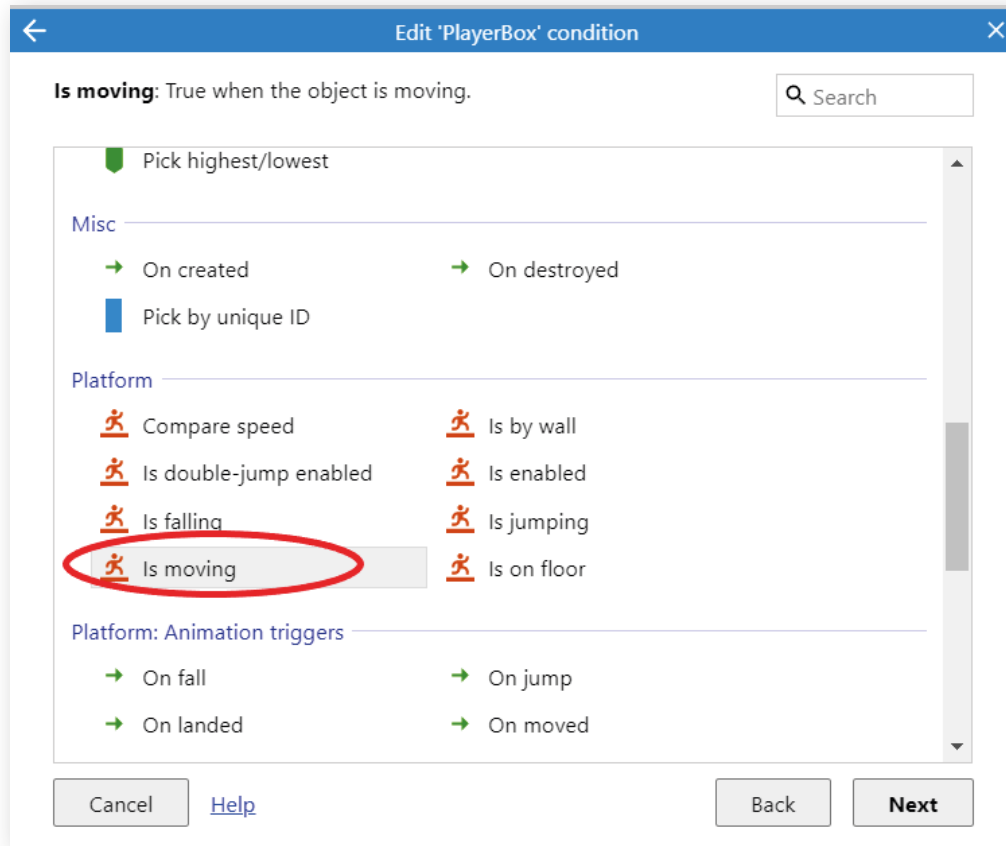
Now the animations are in place, we need some events to make sure they're triggered in the right places.

SETTING UP EVENTS FOR MULTIPLE ANIMATIONS

As the first animation for the Player is Walk, whenever you load the game, the Player sprite will be playing the walking animation, regardless of its actual state. The mirroring provides the directional control, but we need a few more events to use the new Jump and Idle animations and to make sure the Walk animation is used correctly. These event blocks will make use of the inbuilt States of the Platform behavior.

The first event block will ensure that when the Player isn't moving, it'll default to its Idle animation.

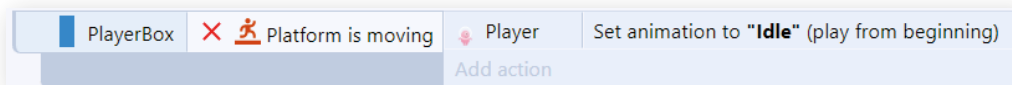
The Condition will rely on the Platform behavior, so make sure you choose the PlayerBox object to set the condition. You'll then want the condition Is Moving which is found under the Platformer section.



Select this and then add the event block. This is a good start, but when something is idling, typically it's *not* moving but handily it's only a quick change to make this event work properly. Right-click the condition to open its menu and select Invert. This will put a red X by the condition, signifying that it has been inverted, or in this case, the condition now means Is NOT moving.

The Action for this event block will be Set Animation – all of the animations are stored on the Player object, so select that for the New Action. Scroll or search to find Set Animation in the Add Action dialog and then set the Parameters to "Idle" for the Animation name, and From Beginning. Click done, and the event block will be complete.

Bear in mind, when you're putting strings into parameters, they are case sensitive, so make sure what you're entering here matches your animation name exactly!



Next, let's make sure the Walk animation works correctly.

Note: Now that you're more familiar with adding events, we'll be writing future events in a more short-form style. The first section represents the object you need to pick in the condition/action dialog, followed by the condition or action you want and then any parameters to fill in.

Following a similar process as above, you'll need the following event:

Condition: PlayerBox ► Is moving

Action: Player ► Set Animation ► "Walk", From Beginning

This is a good start for this event block, but to ensure it works correctly with other animations, we need to add a second condition. Right-click the event block and select Add ► Add another Condition. Now, you can add a second (or third, or however many you need) condition to the original event block.

For this, you'll need the following:

Condition: PlayerBox ► Is on floor

Once added, this event block will default to an AND block – all the conditions must be true for the actions to run. This is perfect for this block, but we'll need an OR block for our final animation controls.

The Jump animation is the final one to finish implementing and it needs to play in two situations, when the player jumps or, as we don't have another animation for this, when the player falls from a platform. Hence the need for an OR block.

To begin with, you'll need to add the following two conditions and one action to an event block:

Condition: PlayerBox ► Is jumping

PlayerBox ► Is on floor

Action: Player ► Set Animation ► "Jump", From Beginning



If you don't see this option in the menu, make sure you're selecting the whole condition block and not a single condition.

Now to get the block working as intended, a few tweaks are needed. Firstly, right-click on the condition section of the block to open their menu and select Make OR block.

The last tweak to make to the Jump animation events is to Invert the Is on floor condition to turn it to Is NOT on floor.

Now if you preview your project, you should find that all three animations work as they should – when the player is still, the Idle animation plays, when walking, we get the Walk animation and whenever the player is jumping or falling, we get the Jump animation.

Next up, let's implement an enemy character!

PlayerBox	Platform is moving	Player	Set animation to "Idle" (play from beginning)
Add action			
PlayerBox	Platform is moving	Player	Set animation to "Walk" (play from beginning)
Add action			
PlayerBox	Platform is on floor	Player	Set animation to "Jump" (play from beginning)
Add action			
PlayerBox	Platform is jumping	Player	Set animation to "Jump" (play from beginning)
Add action			
PlayerBox	OR Platform is on floor	Player	Set animation to "Jump" (play from beginning)
Add action			

PART 7 - CREATING AN ENEMY

ADDING THE ENEMY

A good way to add a challenge to a platformer game is to add an enemy character for the player to avoid or destroy. This enemy will be relatively simple – it will wander back and forth, cause damage to the player if they walk into it and be destroyed if the player jumps on it. A bit like those famous Mario Goombas.

First of all, we need a new object to act as our enemy. You should be familiar with adding objects, behaviors and animations now, so the only the basic steps will be listed here. If you need reminding, please refer to earlier chapters.

1. Add a new Sprite object in the layout.
2. Add frames from the Enemies folder in the asset pack. We used the blue slime frames `slimeBlue` and `slimeBlue_move`.
3. Set the animation to looping with a speed of 5.
4. Set the origin to the base of the frame and apply it to the whole animation.

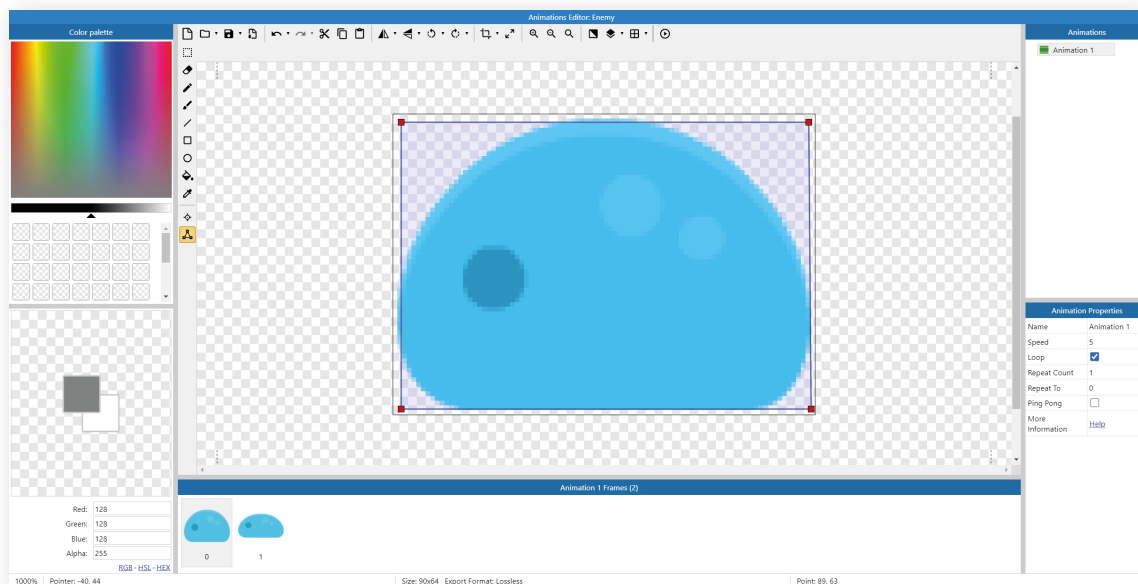
We're not going to give the enemy a helper sprite, so to ensure that it plays well with the rest of the level, we're going to adjust its Collision Polygon. If you click the button below the Image Points button on the left-hand toolbar, you'll find your animation frame now has some red points and blue lines on it – this is the collision polygon!

This was mentioned briefly in the chapter covering the Animations Editor but it doesn't hurt to reiterate here. The collision polygon is the area that counts as colliding for this image. By default, Construct guesses a collision shape, but it is not always accurate, so you can click and drag the points of the collision polygon to alter its shape as you need to. Right-click on a point or in a space to display a menu of additional options for the collision polygon, such as adding and deleting points.

The polygon shape we've ended up with as a default isn't going to sit very well on a flat platform, so we'll adjust it to be a bit more useful. First, in the right-click menu, select Set to bounding box – this will create a collision polygon that fills the whole frame. This is a good start, but it could be a bit unfair on the player as there is some blank space that counts as a collision. Making it a bit smaller should solve this problem.

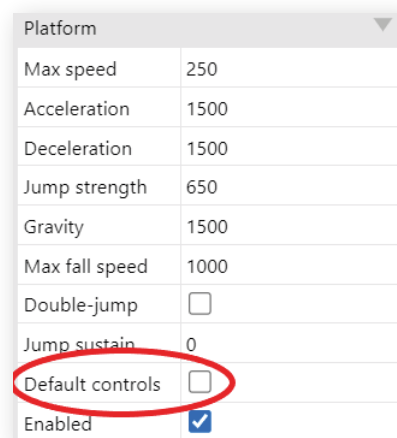
Clicking one of the red collision points brings up an X and Y value underneath the top toolbar. If we change these, we'll adjust the collision polygon, or you can do the same by dragging the red points around.

Feel free to play around with your collision polygon, but you'll want to make sure the box fits snugly around the whole slime.



This won't be quite right for the second animation frame but will be close enough. So, you can either apply this polygon to the whole animation via the right-click menu or you can repeat the above process for the second animation frame.

Now that we've sorted the enemy's collisions, close the animations editor and place on in your level. Remember to rename the object to 'Enemy' and you'll also need to give it the Platform behavior. We want this enemy to move on its own, so it will be controlled by events - this means we need to disable the Default Controls on the platform behavior, so make sure that box is unchecked in the Properties Bar.



The Enemy object will need an Instance Variable alongside the Platform behavior to help us create its movement. A simple technique to control enemies or NPCs in games is to use states. This can be controlled via an instance variable, and that's what we'll do here.

Adding Instance Variables to an object is a similar process to adding behaviors. Select the Enemy object, then in the Properties Bar, click Instance variables on the line Add / Edit under 'Instance variables'.

This opens the Instance Variables dialog – a list of all the instance variables on the current object. The list for our Enemy object is (unsurprisingly) empty, so click Add new instance variable to add a new one. Set the Name to Direction, the Type to String, and the Initial Value to Left (for moving left). Again, remember that strings are case sensitive, so when using your instance variables in the event sheet, remember exactly how you typed them!

Enemy instance variables				
Name	Type	Initial value	Show	Description
Direction	String	Left		
Add new instance variable				

CREATING ENEMY MOVEMENT

If we're implementing Goomba-like movement, that roughly boils down to walking in a specific direction until it finds an obstacle and changing direction when it does.

We already have one potential obstacle built in – our Platforms tilemap. As this is a solid object, it will be detected by the Platform behavior's Has wall to left/right condition which we'll be using for this movement type. However, sometimes it's not feasible to use the tilemap itself, for instance on a floating platform that has no walls. So, we'll need another helper sprite – this time called EdgeMarker.

Add yourself a new square sprite, make it 64x64 in size, give it the solid behavior and pop a couple at either end of a floating platform in the level. Don't forget to name it EdgeMarker, and like PlayerBox, this should be marked as Initially invisible in the Properties Bar.

Next, the logic for enemy movement can be implemented.

The movement will depend on the current string in the Direction variable so the event blocks will check the variable and then run actions accordingly:

Condition: Enemy ► Compare instance variable ► Direction, Equal to, "Left"

Action: Enemy ► Platform, Simulate control ► Left

Enemy ► Set Not Mirrored

To save time, you can copy and paste your new event block and replace the Left parameters with Right. For the Right directional event block, the enemy should be set to Mirrored.

Enemy	Direction = "Left"	Enemy	Simulate Platform pressing Left
		Enemy	Set Not mirrored
Add action			
Enemy	Direction = "Right"	Enemy	Simulate Platform pressing Right
		Enemy	Set Mirrored
Add action			

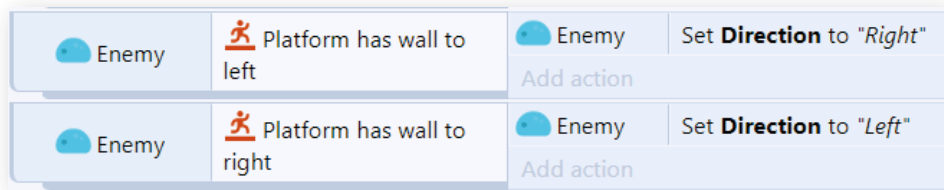
So, the enemy objects should now move depending on how their variables are initially set - remember when you add an enemy to your layout, you'll need to manually set the variable string. If you've not done that for enemies in your layout, do so now - a blank variable means your enemies won't move!

To get the complete behavior we're after, the enemies should change direction when they come up against a wall - this can be done quite easily thanks to the Platform behavior. Set up the first event block as follows:

Condition: Enemy ► Is by wall ► Left

Action: Enemy ► Set instance variable ► Direction, "Right"

You can copy and paste the block and swap the lefts and rights to get your second wall-detection block.



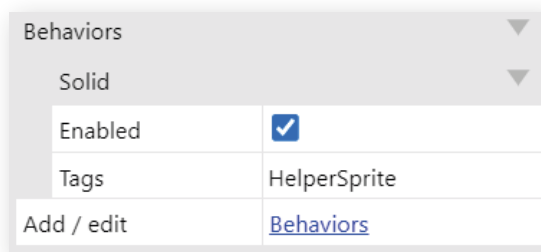
And with that, the enemies should potter about the level and not get stuck against any walls. Unfortunately, the player can still get stuck on the invisible EdgeMarkers that assist the enemy objects. We can use Tags to filter what solids the player collides into.

The tags need to kick in as soon as the game starts and handily there's an event to trigger that:

Condition: System ► On start of layout

Action: PlayerBox ► Set solid collision filter ► Exclusive, "HelperSprite"

Then, all you need to do is add the tag HelperSprite to the Solid behavior on the EdgeMarker object and the PlayerBox will be able to pass through them.



INTERACTIONS WITH THE PLAYER

It's all well and good having an enemy, but if it doesn't interact with the player at all, it's essentially just decoration. For this project, we're going to implement the following:

- If the player runs in to the enemy from the side, they take damage and flash to give the player feedback.
- If the player jumps on top of the enemy, the enemy is destroyed.

Starting with the logic for damaging the player, we first need an extra behavior and an instance variable. Add the Flash behavior to the Player object and add a Number instance variable called Health to PlayerBox.

Now we can start building the logic itself. Start with the following condition:

Condition: PlayerBox ► On collision with another object ► Enemy

With this condition, any actions will run whenever the player hits the enemy object, regardless of the angle. We need to be a bit more specific, so we're going to use sub-events. Right-click your new event block to open its menu and select Add ► Add sub-event.

You can alternatively press S or B when the top-level is selected. S will open the Add Condition dialog and allow you to directly select the condition of your choice to add. B will create a new blank sub-event you can add conditions/actions to.

You'll need two conditions for this sub-event:

PlayerBox ► Is falling

PlayerBox ► Compare Y ► Less than, Enemy.Y

This will check if the player is not only above the enemy, but they've also initiated a jump or fall to squash the enemy beneath them. As the Y axis increases downwards, if the player's Y co-ordinate is lower than the enemy's, they are above it.

The actions required for the event block are:

Enemy ► Destroy

PlayerBox ► Set vector Y ► -550

This will destroy the enemy and give the player a bit of feedback by making them bounce. 'Set vector Y' basically just sets the vertical speed of the platform movement; setting it to a negative value sets it upwards (again, the Y axis increases downwards), and 550 is a little less than the default jump strength of 650.

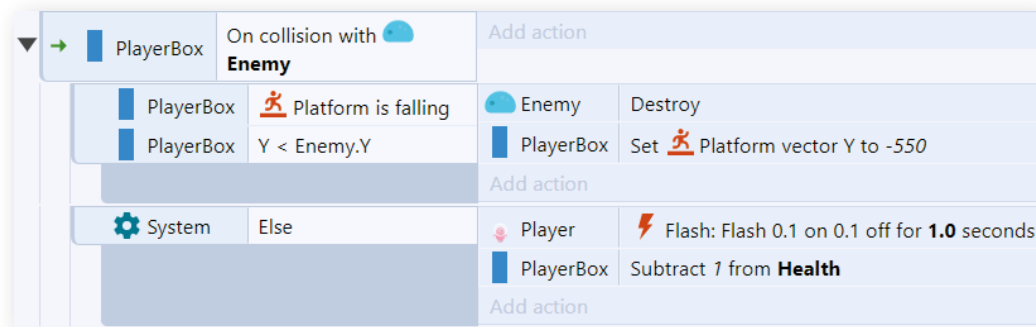
We're not quite done though. Right-click the margin of the 'Is falling' event (the space just to the left of the PlayerBox icon) and select Add ► Else. 'Else' is a special condition that runs if the previous event did not run.

So, this event will run if we collided with the enemy but we weren't jumping on top of it - we ran in to it from the side, say. In this event we want the player to be hurt. Add the following actions to the Else block:

Player ► Flash ► (leave default values and click Done)

PlayerBox ► Subtract From ► Health (Number), 1

This will give the player a bit of feedback that they've taken damage and subtract one from their health value.



If you want to make things a little more challenging for the player, why not add more enemies? You can either drag a new copy of the Enemy object onto the layout from the Project Bar or you can copy and paste the one you've already got.

PART 8 - ADDING COLLECTIBLES

Collectibles are another way to add some quick engagement to a platformer and can be implemented pretty quickly in Construct. Let's add a collectible Star to the level.

CREATE THE COLLECTIBLE ITEM

The collectible star will be a new Sprite object, so create a new sprite and import the star image from the PNG/Items folder in the asset pack. The frame has quite a bit of empty space around it that's not needed, so use the Crop tool on the main toolbar to remove it.

The collision polygon doesn't really matter for the Star object, so while you could change it to a bounding box, it's not necessary. Once you've finished in the Animations Editor, rename your new sprite to Star and place a few around the layout.

To make them look a little more interesting, you can use the Sine behavior. The Sine behavior can adjust an object's properties (like its position, size or angle) back and forth according to an oscillating sine wave. In this case, we're going to adjust the Star's position to make it look like it's floating and not just plonked onto the level.

Add the Sine behavior to the Star object and you can mess around with its properties. The movement doesn't need to be huge, just a subtle motion to give the Star a bit more life. The properties we settled on were:

Sine	
Movement	Vertical
Wave	Sine
Period	2
Period random	0
Period offset	0
Period offset random	0
Magnitude	5
Magnitude random	0
Enabled	<input checked="" type="checkbox"/>
Preview	<input type="checkbox"/>

This is a quick way to add some visual feedback to an object without having to create more animations.

COLLECTING AND SCORING

The basic mechanic for collecting stars should be:

- Player collides with Star
- Star is destroyed (to avoid being repeatedly collected)
- Player's Star count increases

In order to track the Player's current star count, we'll need a Global Variable. If you right-click in an empty space in the event sheet, or on an existing event you'll find options to add a Global Variable. If you have no event selected, you can also use the V key as a shortcut.

In the *Add global variable* dialog, set the Name of the new variable to **StarTotal** and the rest of the parameters can stay as they are - we need the variable to be a Number and it makes sense for it to start at 0! Close the dialog when you're finished.

Next, you'll need the events to create the collection mechanic.

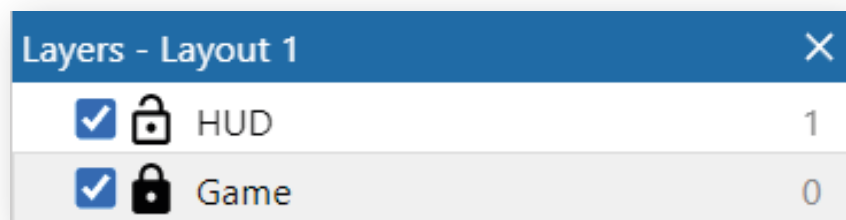
- Condition:** Player ▶ On collision with another object ▶ Star
- Action:** System ▶ Add to Global Variable ▶ StarTotal, Value of 1
Star ▶ Destroy

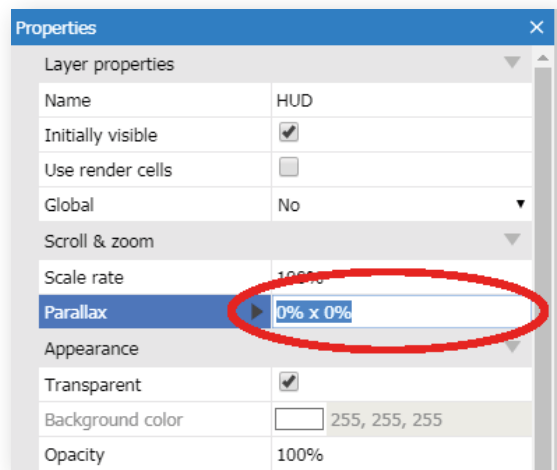
If you preview the game using the Debugger, you'll be able to see that when the player collides with a Star that the object is indeed destroyed and the global variable StarTotal increases by 1, which is great, the mechanic is working but it's not very good for the player. The player needs some visual feedback to tell them that their count is going up - we can add this using a text object and a HUD layer.

HUD stands for head-up display and is great for showing information to the player while they're playing. The information is kept out of the main view, but the player can glance to look at it or see it in their peripheral vision while they play the game.

To create our HUD, you'll need a new layer. Make sure you've got the Layers Bar open (the one that's usually in the bottom right with the Tilemap Bar) and right-click in space, then select *Add layer at top*. At this point, it's probably worth renaming both layers. You can access the rename option with the right-click menu on each layer - name the new one HUD and the one containing your game objects and level, Game.

As you're going to be working exclusively with the HUD layer for the next bit, it's worth locking the Game layer to avoid accidentally adding things to or changing the wrong layer. Click the padlock icon on the right-hand side of the Game layer line in the Layers Bar to lock it (and click it again to unlock it.)





The HUD layer needs a quick property change to ensure it functions correctly and the player can always see their current Star count. This is done through Parallax. A layer with its parallax set to 0, 0 will not scroll at all, so any objects placed on this layer will always stay in the same place on-screen. Note that in this case, objects should be placed within the dashed rectangle that appears in the top-left of the Layout View (the Viewport).

With your HUD layer selected, change its Parallax property in the Properties Bar to 0, 0.

To show the player their current count, you're going to need a Text object. Add one in the same way you added your sprites and tiled background and rename it to StarText. You can change properties like font and colour in the Properties Bar, so once you've got your text looking just how you'd like, make sure the object is up in the top left corner of the Viewport.

Next, you'll need an event to actually show the count value. Handily, this will just involve editing an existing block.

Find your Every Tick event block and add a new action:

StarText ► Set text ► "Stars Collected: " & StarTotal

Breaking down the string you've just used to set your text you've combined the string "**Stars Collected:** " which will be a static piece of text, with your Global Variable **StarTotal**. This is a good example of concatenation and it's something used a lot as projects get more complex.

Now if you preview the game and collect some stars, you should see the count being increased as you play!

That's all for the tutorial as far as building the game is concerned, congratulations for getting this far! The next chapter covers options for testing the game and publishing it.



TROUBLESHOOTING

MY BARS HAVE DISAPPEARED! 100

MY CHARACTER ISN'T MOVING PROPERLY! 100

I CAN'T ADD THIS OBJECT! 101

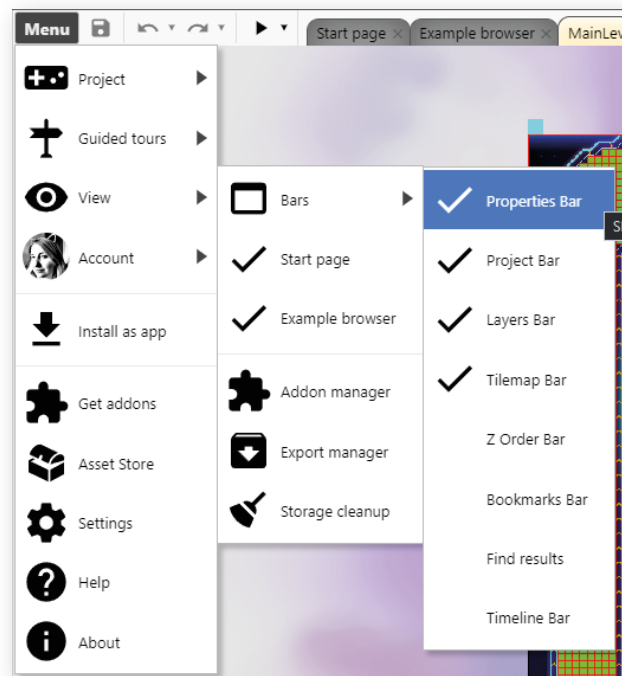
I REFRESHED/CLOSED THE BROWSER! 101

Inevitably, as with any project, something will probably go wrong at some point. So, this chapter looks at some of the common issues that students and teachers run into during a lesson.

MY BARS HAVE DISAPPEARED!

Construct has a lot of flexibility in how you want its UI to look and as a result, it's quite easy to move or close the various bars. However, it can be quite a shock if you accidentally close a bar and don't know how to get it back.

As mentioned in the chapter discussing Construct 3's main menu, you can use the View menu under the main menu to customise and re-open any bars you've closed.



If you've accidentally moved a bar, you can drag it back to where you'd like it to be. Construct has a set of snap points for bars so you can customise where you want to put everything.

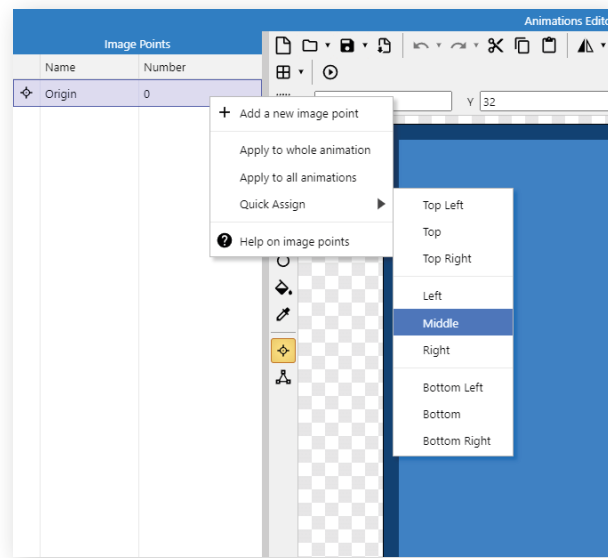
MY CHARACTER ISN'T MOVING PROPERLY!

This is actually quite a broad issue, but in the scope of the tutorial in this book, there are a couple of things to check if a player has some movement issues.

In Construct, you can add multiple copies of the same behavior to an object which can have its uses, but if it's not set up correctly, conflicting behaviors can cause strange movement. So, if your character isn't moving as it should, make sure there is only one copy of the movement behavior applied.

Movement behaviors also rely on the origin point of the object they're applied to. As mentioned in the section about the Image Points Pane, **the Origin is a special kind of image point defining the center of the object, or its point of rotation.**

So, if you move the origin point in the Animations Editor, it will affect how the movement plays out in your game.



You can reset the origin point in the Animations Editor's Image Points pane. By right-clicking on the origin, you can find the quick assign menu which allows you to quickly put the point back to where it needs to be.

I CAN'T ADD THIS OBJECT!

It can be frustrating when you go to add an object and it's nowhere to be found in the Create New Object dialog - there are a couple of reasons as to why this might happen.

Some objects, like the Keyboard are global objects that can only be added to a project once. When they're added, they're available to the whole project, so there's no need to add it again, so Construct removes it from the object list.

Another reason things might not appear in the New Object dialog is that the Simplified User Interface has been enabled. The full details about what the Simplified User Interface hides is outline in the User Interface chapter of this book, but it will hide some of the more advanced plugins and objects. You can check if it has been enabled in Construct's settings.

I REFRESHED/CLOSED THE BROWSER!

Unfortunately, if you've not saved your work recently, if you close the browser or refresh Construct, any work will be lost. It's very important that you make regular backups, either manually or using Construct's built-in backup system, and don't close your browser unless you're finished with your work. You can always see if you have any unsaved work in the Project Bar - if anything in the Project Bar is displayed in italics, then you have unsaved changes.



PREVIEWING YOUR GAME

PREVIEW TYPES	103
Debug Layout.....	103
Preview Project.....	103
Remote Preview	103

PREVIEW SETTINGS.....	105
------------------------------	------------

To test your game during development, you can preview it by clicking the "play" icon in the main toolbar, by selecting Menu → Project → Preview, by right-clicking a layout in the Project Bar and selecting **Preview**, or by pressing F5. This will start your game from the current layout. We recommend testing your projects frequently, as well as having regular backups, so if you accidentally break your project, it's easy to go back without having to remember everything you did!

PREVIEW TYPES

Previewing your project is a great way to keep track of how it's working as you're building it. Pressing the preview button on the toolbar will start a preview of the current layout, but as mentioned earlier in this book, you can also use Project, Debug and Remote previews.

In the main toolbar, there is a dropdown arrow next to the Preview button that shows a menu with more preview options. These can also be found in the Menu → Project submenu, or by right-clicking the project name in the Project Bar.

DEBUG LAYOUT

This runs the current layout in a special debug mode. The debugger is a special development tool which helps you inspect the state of the game (such as the value of expressions and variables). It also provides diagnostic tools such as advancing the game frame-by-frame, changing values, destroying objects, setting breakpoints in events, and more. This can bring invaluable insight to how your game is working, particularly if you run to a problem.

More detail on the debugger can be found in its dedicated section in the UI chapter.

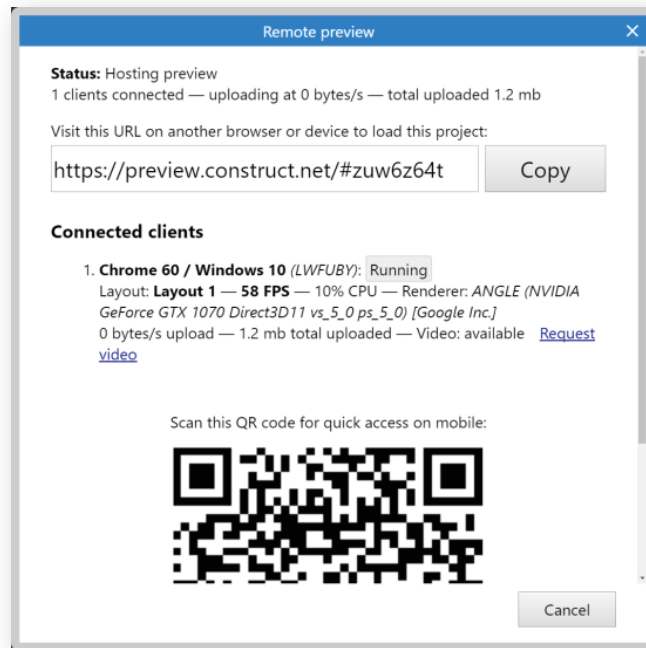
PREVIEW PROJECT

This simply starts a preview from the first layout in the project. This is either the first layout that appears in the Project Bar, or whichever layout is set in the *First layout* project property.

REMOTE PREVIEW

This allows you to preview your project on a different device. It is also useful for testing different browsers on the same device. Starting a Remote Preview does not actually directly run your game. It will open a dialog that provides a URL you can use to load the game, or a QR code to scan. All you need to do is open the URL on another browser or device, or share the URL with someone else, or scan the QR code, and the game will start to load and run in the browser.

The game is loaded directly from your device using a peer-to-peer connection; it is not uploaded anywhere else but is still accessible from anywhere on the Internet. As soon as you close the Remote Preview dialog, the game is no longer available from the provided URL. You can open the Remote Preview dialog to its own window to help keep it out of the way, by right-clicking on its caption and selecting Open to pop-up window.



Once the game starts running, they appear in the Remote Preview dialog as a connected client. You can have multiple copies of the game running simultaneously.

You can view some basic system details and real-time performance information for connected clients, including their browser & OS, which layout they are on, the framerate and approximate CPU usage (and approximate GPU usage if available), and their graphics hardware. You can also click Request video to see a video stream of their game.

Like with a standard preview, you can update a remote preview by selecting the **Remote preview** option again. This updates the version of the project available at the same URL. Any existing clients will have to reload their browser to see the new version of the project.

Clients who are viewing your project via Remote Preview will see notifications in the following situations:

- When the host updates the project, clients will see a notification indicating an update is available. They must reload their browser to load the new version.
- When the host closes the Remote Preview dialog, the remote preview ends. Clients will see a notification that the host disconnected. Clients can continue to run the project (they are not cut off), but if they reload the project will no longer be available.
- When the host starts or stops video, the client will be notified.

Remote Preview allows you to instantly share your project to anyone in the world with an Internet connection. This is a great way for your students to quickly share their games with you or their peers, either for testing, reviewing or just for fun!

Talking of testing, you can use the remote preview URL on the same device for cross-browser testing, such as using Remote Preview to test your project in Firefox while Construct 3 runs in Chrome. In this case, data is not sent over the Internet and is only transferred across the local system.

PREVIEW SETTINGS

By default, starting a preview opens a pop-up window. You may see a message that the pop-up was blocked. Clicking **Try again** normally works, but to permanently prevent the message appearing you may need to change your browser's settings. Usually, an icon or message will appear somewhere in the browser UI indicating a pop-up was blocked; clicking this should provide a way to always allow pop-ups for the current website.

In Menu → Settings, you can choose different preview modes. The three options are:

Pop-up window: as described above, opens a pop-up window to run the project in.

Browser tab: opens a new browser tab to run the project in.

Dialog: opens a dialog inside the Construct 3 UI to run the project in. This does not use a new browser window so is not subject to pop-up blockers and does not include other browser UI features like the address bar. However, it cannot appear larger than, or outside of, the Construct 3 window.

If you select **Preview** again with a preview already running, the existing preview window or dialog will restart and begin previewing the latest version of your project.

Let's say you want to test multiplayer, and you need a couple of preview windows open. Your best options are either:

- Have the project open in multiple browsers – this also allows you to see if there are any differences when running your game in Firefox vs Chrome, for example.
- Or, if you're using the full version, you can use Remote Preview



EXPORTING AND PUBLISHING

EXPORTING FOR WEB	107
Publishing to the Construct Arcade	108

EXPORTING FOR MOBILE	109
----------------------------	-----

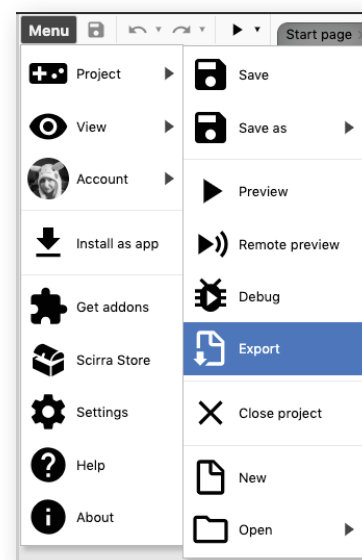
EXPORTING FOR DESKTOP	110
-----------------------------	-----

Construct 3 allows you to export to a range of platforms, such as the web (HTML5), Android and iOS (via **Cordova**), desktop apps (via **NW.js** and **WebView wrappers**). Each platform has an accompanying tutorial to guide you through how the exporter works and covering any exporter-specific settings – you can find all of these in the tutorials section on **construct.net**. There is also a publishing course that houses the more important web, desktop and mobile publishing tutorials in one place.

Bear in mind that in some cases fees may be involved, such as for developer subscriptions to allow uploading to app stores, or for hosting web content if you don't already have a server.

To publish a project, use the Menu → Project → Export option. Every kind of export has three common settings:

- **Deduplicate images** will search the entire project for identical images and remove the duplicates. This helps save memory and reduce the download size by removing redundant images.
- **Recompress images** will recompress all the PNG images in the project with enhanced compression. This can take a long time, but often significantly reduces the download size of the exported project. This step is lossless, so it is guaranteed to preserve the quality of all your artwork.
- **Minify script** will obfuscate and compress the main JavaScript file for your exported project. This also helps reduce the download size, improves load time, and makes it significantly more difficult to reverse-engineer the project.



Each option helps optimise the exported project but can make the export take longer. It is recommended to enable all three when exporting the final finished project for publishing. However, if you are simply doing a trial export, you may wish to disable them to speed up the process.

Most exporters will prompt you for additional settings specific to that exporter. For example, the Android exporter has a setting for the minimum supported Android version.

When the export finishes, you will be provided with a link to download a zip file containing the final exported project. You can also view some statistics about the export, such as how much download size was saved by image recompression (if it was enabled). The Export Manager can also be used to find the last few exported projects and download them again.

EXPORTING FOR WEB

Construct projects are based on HTML5, which makes them perfectly suited for web publishing.

To publish to web, use the Web (HTML5) option when exporting. When the export finishes, you'll get a zip file which contains all the files to upload.

You will need a web server to upload your game to. If you don't have one, there are many services where you can rent a cheap static file host. The specific steps for uploading files to your server depend on the host.

There are lots of services out there you could use. Two possibilities are **Netlify Drop** and **Static.app**, both of which allow you to drag and drop in some files for instant hosting. There are also websites like **itch.io** which allow users to create accounts and then host their games on the site just by uploading the exported zip file.

It's common to make a mistake while uploading your files. If you forget to upload a file or folder, the game could break. If your game does not seem to work once uploaded, check for browser errors and hopefully there will be a useful message (e.g. "myfile.png returned 404 not found", indicating you forgot to upload myfile.png).

Do bear in mind though that games will not work if you run them from disk.

If you extract the exported zip to your computer and double-click **index.html** to run it, the game will start using a **file:/// URL** instead of a **http://** or **https:// URL**, since the game is on disk and not on a server. For security reasons browsers have some tight limitations on what can be done in a web page from a **file:/// URL**, and these usually stop Construct games from working.

The best workflow is to test by previewing in Construct during development, and then immediately publish the game to the web after exporting. If you run a game from disk, you might see a messagebox reminding you of this ("exported games won't work until you upload them").

If you really need your game to run from the desktop, use a desktop export option instead. If you need to test on mobile, you can use Remote Preview instead of having to export repeatedly.

PUBLISHING TO THE CONSTRUCT ARCADE

An easy way to publish to the web is to use the Construct Arcade! We host the game for you, so you don't need to find your own web server.

To ensure players have a good experience on the Construct Arcade, some features are blocked. For example, your game cannot navigate to another page, nor open any pop-up windows. If you wish to provide a link to a website or to your game in an app store, you can add them elsewhere (we'll talk about this in a bit!)

There are also some types of content that aren't allowed on the Arcade. Please refrain from publishing games that contain:

- **Real Money or Cryptocurrency Prizes**
- **Pay to play or In-app purchases**
- **Chat rooms or multiplayer chat functionality**

Bear in mind you'll need to give your game an age rating, so read them carefully when you come to publish! If your game has been published with something that shouldn't be on the Arcade, we may take it down without warning.

To export for the Construct Arcade, simply choose the Construct Arcade option when exporting. You'll get a zip file when the export finishes just as you would with a standard HTML5 export.

You then need to upload this file to the Construct Arcade and we'll walk you through the information you need to fill in.

Firstly, you'll need to make sure you're logged into your Construct Account - you can't upload games as a guest!

On the main Arcade page is the option to Upload a Game, click this to start the process. You'll need to fill in some information about your game before it can be uploaded:

- Game Language
- Game Name
- A short Description - this should be more than 100 characters
- Category - pick which best describes your game, is it a puzzler or an action game for example?
- Status - is your game complete, a test or a demo of a larger game?
- Age Rating - be sure to read the ratings thoroughly and categorise your game accordingly, else your game could be removed, and your account banned from publishing
- Supported Devices - have you designed/tested your game for Desktop, Mobile or both?

Next, you'll be able to upload your game file. You can also choose to upload the source file if you want. Once you've uploaded the zip file, you can view its arcade page - but the game will be unpublished.

Once uploaded you can access new pages including the Basic Details, Settings and App Store Links pages.

On the Basic Details page, you can add more information like a fuller description (including any external links, perhaps to your website) and instructions for how to play the game. You can also adjust any of the information you added before uploading your files.

The Settings page allows you to tweak certain aspects of the game's page. This is where you'll find the privacy settings so you can decide if your game is public or not. You can also turn off commenting and public ratings if you like, and you can decide if you want to show the game's stats or live player counts. Plus, if you're updating your game in the future, there is a setting in here to enable or disable the ability to play older versions of your game.

If your game is classed as a Full Game or a Demo you'll see the App Store Links option as well. This is where you can add links to Apple's App Store, Google Play or Steam.

Under the Game Files section of the menu bar, you'll see links to the Main Game Image, Game Icons and Screenshots pages. This is where you can upload graphical assets for your game. If you've created a great piece of cover art for your game, or want to show off some excellent gameplay screenshots, you can upload them in the relevant tabs.

When you're happy with the information available and the game's settings, you can hit publish, agree to the terms and conditions, and your game will be available for everyone to play!

EXPORTING FOR MOBILE

There are several ways you can export and build mobile apps from Construct 3.

Also, it should be noted that you should test regularly on mobile from the very start of your project! If you do not, you may not find out that your game is too slow or difficult to use on mobile until the end, and it will be very difficult to make changes by then.

When you export your project, you can choose the Android or iOS options. Each platform has different build options and specifics to be aware of.

For exporting to Android, your export options are quite varied:

- **Cordova project:** Export a Cordova CLI project. You can submit this to third-party build services. Advanced users can also use this to manually build Android apps.
- **Android Studio project:** Export an Android Studio project. This option is for advanced users who are already familiar with Android Studio. Refer to Google's documentation on Android Studio for more information.
- **Debug APK:** Use Construct 3's mobile app build service to build an Android app (APK) for you. This option is for testing your app during development.
- **Signed release APK/AAB:** Use Construct 3's mobile app build service to build a signed Android app (APK) for you. Signed apps are ready for publishing.
- **Unsigned release APK/AAB:** This option is only suitable for advanced users. Use Construct 3's mobile app build service to build an unsigned release APK, and then manually sign it yourself.

You'll notice that signed and unsigned releases can be built as either APK or AAB files. Both are Android app files, however AAB is a format designed only for publishing to an app store – they cannot be directly installed onto an Android device.

For exporting to iOS there are fewer export options to choose from – either a Cordova Project or an Xcode Project. To publish an iOS app, Apple require that you use Xcode on macOS. They also have stringent requirements on how apps are tested and signed. Because of this, Construct's export options for iOS ultimately produce an Xcode project, where you then continue testing or publishing with Apple's tools.

Both iOS and Android exports can produce a Cordova project which can be used with the Cordova CLI tool. This is an advanced, command-line based tool that allows use of Cordova plugins and offline builds. We recommend using the built-in build service for the majority of mobile app builds, but for more advanced students looking for a challenge, learning the command-line tool could be quite useful.

EXPORTING FOR DESKTOP

Construct has several options for exporting desktop apps. If you're looking for a traditional desktop app, you can use **NW.js**, which is basically a standalone version of the Google Chrome browser. In other words, it's very much like having your project run in Chrome, but without the user needing to have Chrome installed on their computer, and without the browser parts showing like the address bar so it looks like a native app.

NW.js exports Linux, macOS and Windows desktop apps. Both 32-bit and 64-bit variants are available for Linux and Windows. macOS only provides a 64-bit build, since all modern Mac systems are already 64-bit.

Since **NW.js** bundles a copy of the entire Chromium browser engine, the size of the exported files will be relatively large, adding perhaps 80-90mb per platform. For smaller exports, Construct offers two WebView based export options – one for Windows and one for MacOS.

The Windows wrapper is based on the WebView2 runtime, which is part of Windows. You don't need to ship a full copy of a browser engine with the app, which means it has a much smaller file size overhead (under 1mb). The WebView2 runtime also auto-

updates so you don't need to publish updates to your app just to update the browser engine. Note the WebView2 runtime is also based on the Chromium browser engine (which is used by both Microsoft Edge and Google Chrome).

If you have students that wish to take their game further and look at another language while they're at it, there is a C++ Extension SDK for the Windows WebView2 export that is designed for integration with storefronts like Steam and Epic Games.

The macOS wrapper is based on WKWebView, which is based on Safari's browser engine (WebKit). This updates with macOS system updates. The version available is based on the version of Safari that originally shipped with macOS. For example, on macOS 10.14, WKWebView is based on Safari 12, on macOS 10.15 it's based on Safari 13, and so on. Note updating the Safari app does not affect this: only updating the entire macOS system will update the version of WKWebView in use.

More information on Construct's various export options can be found in the [Publishing Tutorial Course](#) on our website.



WHAT TOPICS HAVE WE COVERED?

SELECTION AND CONDITIONAL LOGIC 113

ITERATIVE TESTING AND DEBUGGING 113

VARIABLES AND DATA TYPES..... 113

LICENSING AND COPYRIGHT 113

LEVEL DESIGN..... 114

ORGANISATIONAL SKILLS..... 114

Creating games using Construct is a great way to teach computer science and programming principles through a compelling and creative medium, but what have we looked at in this short tutorial?

SELECTION AND CONDITIONAL LOGIC

Selection works by testing a condition. The test gives a Boolean result – TRUE or FALSE. If the result is TRUE, the program follows one path – otherwise it follows another.

Construct's event system is built on Boolean or conditional logic – an event can be boiled down to 'If this is true, do this' which is a good starting point for learning about selection. Events can then be made more complex by using OR blocks or Else blocks – further demonstrating aspects of Boolean logic.

Sub-events then allow you to teach the concept of nesting – a sub-event is 'nested' under the higher-level event, meaning that its conditions will only be checked if the top-level event's conditions are true.

ITERATIVE TESTING AND DEBUGGING

More often than not, a program will not work correctly when it's first built, so learning how to test, debug and iterate on those results is a key skill for developers.

When you first put together your platform level and give your player the platform behavior, it probably didn't quite work right when you tested it. So, you tweak the behavior properties, change the player size, move platforms around – all of this is part of the debugging process even if it's not specifically related to code. Combine this with testing tables to really drive home the ideas behind finding, fixing and retesting issues in your projects.

VARIABLES AND DATA TYPES

The standard definition for a variable is 'something that is used to store information to be referenced and manipulated in a computer program.' In our short tutorial, you've used a string instance variable to store the data for the enemy's movement direction and a global variable to store the value for the player's current score.

This offers a good opportunity to discuss different data types and how this data can be used in the scope of a project. Construct can introduce the concept of String, Number (these can be integer or float values and Construct contains expressions like int, round and float to set numbers to whole or decimal values) and Boolean data types for variables. And while instance variables are more of a Construct feature, global variables are commonly found in programming languages as variables that are available throughout an entire program.

LICENSING AND COPYRIGHT

This project uses assets with a Creative Commons license, but sometimes, it's really easy to just grab a couple of bits from Google and put them in. What's the harm, right? The ability to import artwork into Construct allows for discussion around the use of copyrighted materials, even in personal games.

In both the UK's current GCSE syllabus and the United States' AP Computer Science Principles curriculum, ethical and legal considerations around computing are part of the required knowledge. Discussing license types and what assets can or cannot be used in games is a great starting point for this topic.

LEVEL DESIGN

By creating your platformer level from scratch, your students get an insight into how to build a fun level. It's far more than a case of just popping some blocks on a screen and hoping for the best. For instance, a level might look fantastic, but some jumps might not be possible, or the player might be too big for gaps between platforms. Or in some cases, it might just be too easy, or too hard. Level design is often an iterative process and this tutorial begins to teach that concept.

For an extension activity, why not have students mock up a platformer level on squared paper before they then build it in Construct? Planning on paper is a great way to visualise their ideas and squared paper simulates a tilemap very well!

ORGANISATIONAL SKILLS

While the game created in this tutorial may not be very complex, it's still got enough objects to begin discussing things like naming conventions to keep your project organised. Having 50 sprites all named Sprite1, Sprite 2 etc is a recipe for disaster further down the line. There are plenty of naming conventions used in programming including snake_case, camelCase and PascalCase. You could get students to discuss the differences between the conventions and which they'd prefer to use.

If you wanted to take project readability a step further, you can encourage your students to add Comments to their code. Comments were mentioned in the UI chapter of this book, and they're a fantastic way to not only make sure someone else knows what the code does when looking at the project, but also that your students fully understand what they're code does.

Lots of functionality in Construct has a counterpart in a traditional text-based programming language too. This is one of the things that makes Construct ideal for teaching the broader principles before diving into a specific syntax.

WHAT NEXT?

Hopefully now you have a basic understanding of how the Construct engine works and can now start passing this knowledge to your students. If you want to explore more with the game created in the tutorial, why not try some stretch goals (either for yourself, or your students!)

- The player has a health variable, but it's not displayed anywhere. Try implementing a visual feedback system for the player's health.
- You've implemented collectibles, so as an alternative to the Star Count, try creating a Win Condition for when the player collects all of the items in a level.
- Your game so far only has one level, why not try creating more?

The best way to learn about Construct is to use it! Try creating small projects to get used to how the engine works and what it can do. If you want to introduce it to your students quickly, there are also the in-built Guided Tours that provide a step-by-step walkthrough of several key features including Timelines and JavaScript as well as a platform game like the tutorial in this book.

USEFUL RESOURCES

There are plenty of resources to help you further with Construct 3 - all available on the Construct website.

The **Construct 3 User Manual** is the complete documentation for C3 and is updated whenever there is a new stable release. It contains information on all of Construct's functionality including behaviors, UI and detail of the conditions, actions, and expressions available for all of Construct's plugins.

We have a **free set of lesson plans** aimed at Middle School students, but they are available to download from our website to use or edit as you need to for your classroom. This resource comes with 12 lesson plans with accompanying worksheets and is fully aligned to CSTA standards.

We also have **two sets of game specific resources** from our partner, Digital Schoolhouse. The first is a workshop pack themed around the game Starlink: Battle for Atlas and contains a teaching guide for running the workshop, worksheets and slides for accompanying unplugged activities and the assets and slide pack for creating the game itself.

The second is a maze game which is based on the DSH Part-Baked Games concept. The resource contains several versions of the same game to allow students to try and build the game themselves and check progress as they go or add to the existing game to improve it.

Talking of partners, **Digital Schoolhouse** and our other education partners **STEMFuse** have their own resources available for teachers. DSH offer free workshop packs similar to Starlink on a variety of topics both with and without Construct, while STEMFuse sell a full curriculum for a variety of skill levels all using both Construct 3's event system and JavaScript.

For less education-focused resources, be sure to check out our **tutorials** and **courses**, all available for free on our website. The resource bank contains content written both by Scirra and the Construct Community and can either be used by students when they're looking for help with their own projects or as the basis for your own lesson plans.

If you're teaching more advanced students or those working together on projects, we recommend looking at our tutorial about **Collaborating using GitHub**. GitHub is a tool used widely in industry and using it with Construct is a great way to learn how it works. The tutorial covers all the steps for using it properly. Plus, for those wanting to stretch into JavaScript, we have a 13-part tutorial set on **Learning JavaScript in Construct 3**. And for those already familiar with the language, a **Quickstart Guide** to cover some of the particulars of using the language in the engine.

We also have an active community that we encourage you to be a part of! **The Forum** on our website has an Education and Construct section where you can discuss all things from how you use Construct in the classroom to issues you have with getting something to work. If you have more specific questions about teaching with Construct, we also have our **Teacher Advisor Program** - a group of teachers who are already using Construct with their students and are more than happy to help you with your queries.

THANK YOU!

Thank you for taking the time to read this book and we do hope you found it useful! Construct can be a great tool for teaching students computer science and programming principles as well as game development skills.

We always love hearing what people get up to with Construct in their classrooms, so keep us posted via our X (Twitter) account - [@ConstructEdu](#).

Best of luck with your classes and we thank you again for choosing Construct!

Copyright © 2024

All rights reserved.

Cover and Book Design by Davina Hopping

No part of this book can be reproduced in any form or by written, electronic or mechanical, including photocopying, recording, or by any information retrieval system without written permission in writing by the author.

Printed by Book Printing UK www.bookprintinguk.com

Remus House, Coltsfoot Drive, Peterborough, PE2 9BF

Printed in Great Britain

Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of information contained herein.