LEVEL 0
INTRODUCTION ......................................................... 1

CHAPTER 1 — WELCOME TO LEVEL 0 ......................... 7
    What you’ll learn in this book........................................ 7

CHAPTER 2 — BUILDING YOUR FIRST GAME ............... 11
    Bubble Pop Madness............................................ 11
    Starting a new project.................................... 13
    Add some objects.......................................... 14
    Project properties and layers............................ 19
    Adding Events............................................... 23
    Adding behaviors to your objects.................... 27
    Spawning multiple objects............................... 29
    Adding Scoring to the Game............................. 34
    Adding Lives to the Game................................ 40
    Creating an End Game Condition....................... 43
    Bring your game to life with sounds................. 50
    Changing the Game Metrics.............................. 54

CHAPTER 3 — I SPY ............................................... 55
    Start a new project........................................ 56
    Adding your first touch event......................... 65
    Adding a countdown timer............................... 68

CHAPTER 4 — JUMBLE ART ..................................... 85
    Get Started.................................................. 86
    Setting up The Start Screen............................ 87
Creating the Puzzle Game Mechanics ........................................... 92
Snapping Objects .......................................................................... 104
Object Jumble .............................................................................. 111
Adding a Winning Condition ......................................................... 117

CHAPTER 5 — PLANE ................................................................. 127
Get started! .................................................................................. 128
Import Your Assets ...................................................................... 128
Player Movement ......................................................................... 131
Spawning Random Missiles ............................................................ 133
Creating Lives ............................................................................. 135
Creating a Win/Lose Condition ....................................................... 137
Adding Sound and Music ............................................................... 139

CHAPTER 6 — SOUND BOARD.................................................. 149
Start a New Project ....................................................................... 150
Creating A Menu ........................................................................... 152
Creating Your Game ..................................................................... 158
Adding Sound ............................................................................... 167
Variables ...................................................................................... 169

CHAPTER 7 — PUBLISHING YOUR GAME ............................. 175

CHAPTER 8 — WHAT’S NEXT? ................................................. 193
Take it a step further ..................................................................... 193
Plan your dive and dive your plan .................................................. 194

ABOUT THE AUTHORS .............................................................. 197
Introduction

If you’re an avid gamer and have always wanted to dive down below the screen and create an interactive experience of your own, then you’ve come to the right place. This book will introduce you to the world of mobile game creation, walk you through five sample projects and give you the tools you need to unlock your imagination and your true creative potential.

Did you know that today, the video game industry, which includes console games, computer games and mobile gaming makes more than $30 billion annually? That’s more than movies and music combined!

Believe it or not… it’s a lot easier than you think. You don’t have to be a C++ programmer or a graphic artist to build your own games, and you don’t need to be a computer scientist with a master’s degree. You don’t even need to know any programming language at all! With the information in these few pages, you’ll be able to use powerful apps that do most of the work for you, and you’ll be creating games in no time.

This isn’t a book about learning code or even how to design cool graphics. This handbook will give you the power to put your creativity to work — to give your imagination wings so that you can soar toward whatever you can conceive. There is nothing more satisfying than seeing something you’ve imagined come to life, and the modern gaming landscape gives you virtually unlimited possibilities!

Game creation is easy

Gaming has come a long way since Pacman and Space Invaders. By today’s 3D photo-realistic standards, these games were little better than dinosaurs by comparison. If games are much more complex now, aren’t they also much more complex to build?
When you get to the core of actually programming and designing a game from scratch, yes. However, while games have gotten more complex, they’ve also inspired a host of tools to make building them easier.

In the 1990s, a new type of application began to take shape known as the game engine. Essentially, programmers got together and built a core video game module. This contained all of the logic and the programming to run the gameplay environment. Then they developed the graphics, sound effects (“FX”), music and even game levels in separate applications. The main video game engine allowed non-programmers to, on a limited scale, make their own games based on the core operating principles.

The first of these was a somewhat simple first-person shooter game called Wolfenstein. Wolfenstein was a ray-trace game — meaning that you walked around buildings in what looked like a three-dimensional environment shooting up Nazis in a World War 2 setting. For the early 90s, Wolfenstein was a revolutionary concept in computer gaming, and because it used the Wolfenstein engine, gamers could get free level builders and could use 3rd party graphics programs to do more than just create new levels; they could change the textures that defined how the walls looked, create their own weapons and even alter the characters in the game.

While you were still limited to the parameters of the Wolfenstein engine and couldn’t really change sound effects or music, you could create your own environment. This continued with the release of Doom and Doom2 — similar to Wolfenstein but with a far more complex engine. In Doom, you could create game levels with 3D environments — stairs, platforms and even elevators. The game engine had more complex level creation, sprite (character) graphics, and you could add your own music and sound effects.

If you remember such games as Unreal, Quake and even Sid Meyer’s Civilization games, all of these came with environment creation tools. However, you were still very much limited by the game engine’s core programming. You could modify the look, but not the function.

Even the simplest games are more complex than they seem. Say you want to create an arcade game like Donkey Kong the old fashioned way. If you have ever played the classic Donkey Kong, you know that it’s pretty simple. You have
a two-dimensional board where you climb from the bottom to the top and avoid being hit by barrels that Kong is throwing down at you. Your goal is to rescue the girl he’s holding.

If you’ve ever gotten good at *Donkey Kong*, you’ll remember that there are actually a fair number of game boards — different levels. They get more complex as you move up, including such obstacles as conveyer belts, elevators and as always, Kong’s barrels. It’s a great game that can take up an amazing number of hours, even when compared to today’s enormously complex titles.

So you may think that creating such a simple game would be easy as compared with something like *Call of Duty*, right?

Yes — in a way. Because a game like *Call of Duty* requires an entire team to create. There are storytellers, voiceover actors, 3D modelers, graphic artists and of course, an entire team of programmers to make it all work.

The advantage that these folks have over you creating *Donkey Kong* all by yourself is that they have pre-built tools to help them. They have a game engine to work with. You have only your computer and a C++ IDE — independent development environment — if you’re lucky — so you have to create everything yourself in code. This includes the game logic, the graphics, the sounds, everything. It’s your job to conceptualize everything and to write the complex code that makes it all work. While the end game is simpler than those of today, the development process is much more difficult.

All of that has changed.

Modern game engines not only let you alter the look and feel of your game, they provide you with the ability to actually encode logic and events into your game, and they do this without you having to know any code at all.

**Free tools that make things even easier**

Remember a little earlier on when we talked about game engines for *Wolfenstein*, *Doom* and *Quake*? Now you can simply drag and drop your game elements and create a fully functional mobile game without having to write one line of code. That’s the power of today’s game engines.

There are quite a few of these game engines out there, including *Construct 2*, *GameMaker*, *GameSalad* and *Unity*. In this book, we’ll focus on the one that is your best bet to get started.
Scirra’s Construct 2 allows you to create games for iOS and Android alike. It’s free, and it provides you with very powerful drag and drop tools to build completely functional mobile games. We’ll be using Construct 2 for all five of the game tutorials in this book.

Scirra’s Construct 2 is a true multi-platform development system. You can build your games and then publish them virtually anywhere. In addition to iOS and Android, you can publish your game to be played on a website, Facebook, for Windows or Mac computers and more.

We’ll get more into the specifics of this engine later. For now, there is only one thing you need to know about developing a game in Scirra’s Construct 2 — it’s a real blast!

Just the act of building is fun, and it’ll challenge your creativity, open up your imagination and even release your inner child!

**Why we decided to write this book...**

We believe strongly in giving people tools that they can use to unlock the true power of their imaginations. Game creation is one of the best methods for doing this because it can be done from anywhere, and with today’s technology, it opens up an almost unlimited world of possibilities. Sure, it’s fun to play video games — but when you play one you built yourself, it’s a whole new level of satisfaction.

Our imaginations make us who we are. Why did we build the Pyramids or create a great civilization like Rome? Why did Columbus sail off into the unknown 500 years ago? Why did we send men to the moon?

Because we imagined we could, and we needed to make what we imagined come to life. Every human being has this potential, and if this book helps you to get creative and express your imagination, then we’re proud to be a part of it.

Here’s a little example from Ankur Prasad’s own experience:

“When I was a kid in the 1980s and the first Nintendo came out, I loved playing Super Mario Brothers. I’d already developed a friendship...
with Mario by playing Donkey Kong, and when SMB came out, I was blown away.

Up until then, game systems were pretty simple by comparison. The Nintendo’s 8-bit system was like nothing I’d ever seen. There was a whole new range of colors and sounds and a complexity of gameplay that I’d never imagined.

Before Super Mario, I enjoyed playing video games. After Super Mario, I was hooked. Not just on playing — I wanted to build games like this myself. I wanted to be a part of this — to create cool video games that others would play and fall in love with, too.

As I got a little older, I imagined myself working for EA — Entertainment Arts — creating awesome games. I even tried my hand at game creation by myself using BASIC and then moving up to C. As you might expect, I hit a roadblock.

While even the original Nintendo’s games like Super Mario Brothers, Castlevania, Final Fantasy and even Tetris are simple by today’s standards, they’re very complex to build, especially for one teenage kid at his home computer.

I was quickly overwhelmed by it all and never really got to make my first game. I still had the bug, though. So when I grew up and was working for Amazon, I began hearing about these new powerful game engines.

A friend introduced me to Scirra’s Construct 2, and I was blown away! It was so easy. I didn’t need to write any code, just drag and drop components and create a series of events, and so on.

The first game I made was from a first-person shooter tutorial. It was so awesome when I was done. My character moved when I pushed the arrow keys. I’d done it! I’d finally created my very own video game, and it actually worked.

As you may imagine, I was hooked. That’s the power of the Scirra game engine, and that’s what we’re going to teach you in this book. You’ll learn what you need to know by creating example games, and then you can even try your hand at creating one yourself."

If you’ve ever had even a passing desire to create your own game, this book is going to give you the keys to unlocking a whole new world. For some of you,
it’ll become a fun hobby, and for others it may very well be the launching pad for an exciting new career.

Now it’s your turn. You can finally create your own games with your own story lines and characters — it’s your turn to show the world just how much imagination you really have!
Welcome to Level 0

Well, you have to start somewhere, right? The best place to start is at the beginning with a clean slate and an unlimited number of possibilities!

Remember, this book is designed not just to teach you how to create a couple of mobile games by following step-by-step instructions. That’s the easy part. We are going to get your creative juices flowing and introduce you to a powerful tool that will allow you to put your imagination to work.

What you’ll learn in this book…

By following the five game tutorials we’ve created for you, you’ll be introduced to Construct 2 and how it works, why it’s effective and how you can expand on it and go even further if you want to.

In a way, you’ll be learning game programming without having to actually write your own code. Construct 2 follows the same game building methodologies that you’d use if you sat down and wrote your own game from scratch in C++. Luckily, Construct 2 does the hard work for you — you just have to tell it what to do when, how you want your game to look and how it should behave.

Check out some of the cool features that Scirra has built into their game development engine:

---

**A powerful event system**

Imagine your game character walks over a special part of the board. When he does this, an event is triggered that reveals a hidden door or staircase. Perhaps an enemy character appears, or he’s transported to another level or part of the game.

Events determine cause and effect — what happens whenever something else happens. With Construct 2, you can create and re-use events for a single action, level or throughout the entire game. You create an event sheet that dictates what happens when. You are, in essence, creating game code but in a simple and easy-to-understand language.
Whether you just want to have fun creating games or whether this is your first step into becoming a future programmer, the Construct 2 event system is going to help prepare you. It’s part of the process of harnessing your creative thoughts and putting them into order to achieve a goal.

Flexible and time-saving behaviors

This is an awesome and time… as well as frustration… saving feature of Construct 2.

Behaviors decide what objects do — how they behave. When you assign a behavior to an object, the game engine does the event work for you. The Construct 2 behaviors include such things as 8-direction movement, shooting, jumping, and speed. For example, let’s say that you assign the platform behavior. Then you can move your character — or sprite — across your game board and let him jump and run on solid ground. You can also dictate how fast, how high and how strong the gravity is in a specific area. Then there’s the car behavior, which is used to create any vehicles from cars to space ships to submarines and anything in between.

Behaviors make life so much easier, even if you’re an experienced game maker. They can be combined with events to create unlimited game action, and you can do it in minutes or seconds rather than days of writing code!

Stunning visual effects

Construct 2 provides an array of visual effects you can apply to your games. The design environment makes everything incredibly easy, and even if you’re not a graphic designer or an artist, you can make your games look spectacular.

Additionally, the engine has fallback visual options. These let you stack up a couple of effects so that even if a player has a computer or phone that doesn’t support something, the game will try to substitute that visual effect with something it does support. This preserves your game play and allows your game to work on many different platforms.

The Construct 2 engine also has a really cool particles feature. This creates and moves a lot of tiny images to create effects such as rain, sparkles, smoke, water and other moving environmental effects that add a whole new dimension to your game.

We discovered a long time ago that most people’s biggest learning block is their own frustration. They read some text and then are a bit overwhelmed by all the
things they feel they should know. Just relax, have fun and let the knowledge come to you.

If you just forget about how fast you seem to be getting it and simply believe that sooner or later, you’ll master the skill — it’ll happen much faster. It’s amazing and it really works!

So get ready to start your imagination and to create your first Construct 2 video game!
Are you excited?

You’re about to build your first mobile game. Before we get started, there is one little chore that you need to take care of.

Go to www.scirra.com and register a free account to download the free version of Construct 2. You’ll see that there is a paid version, but don’t worry about that now. The free version has everything you need to work through the five game tutorials in this book.

Of course, if this is something you want to do on a regular basis, the full version of Construct 2 isn’t a bad idea. It really opens up the possibilities and has all sorts of cool add-ons and features.

Okay, now that you’ve gotten Construct 2 set up, let’s jump right into your first game! We will learn the ins and outs of Construct 2 along the way.

Bubble Pop Madness

For your first foray into mobile game creation, we’re going to keep things relatively simple. In this tutorial, you’re going to build a game that’s both fun and easy to make.

Basically, Bubble Pop Madness is a game in which bubbles rise from the bottom of the player’s screen. The goal is to tap each bubble and pop it before it reaches the top of the screen. For each bubble that does reach the top, your player loses a life.

Sounds simple enough, right?
Well, it does get a little more complex. As time goes on, your player will be faced with more bubbles to pop as well as an increase in their speed. The longer you play, the harder it gets.

Ready? Then let’s begin!
Starting a new project

Before you can build a game, you’ll need to start a new project in the Construct 2 environment. A project is a series of elements that make up a game. Think of it as your game space.

To start a new project, click on the gear icon at the top left corner of your screen. A dropdown menu will appear, and you need to select “New.”

Construct 2 has a lot of templates to choose from. For example if you want to make a flappy bird or angry birds-style game, there is a template for that. For this game, select a new empty project.
Great! Now you’ve got a new project started and you can begin creating your game. Here is what your screen should look like:

Add some objects

An object is anything in a game — like the character, background, sound bite, function to detect mouse input, and so on. You’re going to be working with a lot of objects as we go along, so in order to get you started on the right foot, start by creating a “sprite.”

No, it’s not a soda! A sprite is any image displayed on the screen. So in this game all the bubbles the player will tap are sprites. Firstly, for the purpose of this tutorial, go to [http://bit.ly/bubblepopassets](http://bit.ly/bubblepopassets) and download all the
images and sounds (sprites) and save it into a folder. We will refer to this folder as the projects folder moving forward.

Before you get started,

If you want to add things like characters, enemies, bullets or backgrounds inside the game, you will have to insert a sprite. Right click or double click anywhere inside the white space of your project and select “Insert new object”.

From the Insert New Object box, select “Sprite.”
When you click on the sprite image, a cross-hair will show up. Simply click the screen again, and you will see the sprite options.

Of course, you want your sprite to actually be some kind of graphic. So in order to do this, click on the folder icon and grab the bubble.png images from your projects folder. Now that you have your sprite open, crop it using the crop tool.

It’s important to crop or trim your images to avoid unwanted extra space, which takes up more memory on your player’s computer and affects how smoothly the game works. Removing extra space also makes the bubbles actually appear to touch each other when the images collide, which brings you to your next step.
The last thing you need to do is adjust the bubble’s collision. A collision is an area around a sprite that checks to see if it’s been touched or is colliding with another object. Construct 2 automatically assigns a collision to your sprite. You may or may not want to use what it gives you. In this case you will be checking to see if the player has clicked on the bubble. If the collision is off, the bubble won’t pop when a player clicks on it, which makes an unhappy player. So you must make sure the collision is surrounding the entire bubble sprite object.
Right click on the image and click “set to bounding box.”

Great now you have your bubbles implemented in the game.

Now it’s time to focus on the environment of the game itself and add the background. To do this, create another sprite like you did before. Locate the Bathtub image in your objects folder, and add it to your background sprite.
Project properties and layers

So far so good, right? Nothing too complex or scary. Game building is just a step-by-step process. It’s like that old saying, “How do you eat an elephant?”

One bite… or in our case, byte… at a time!

At this stage of your game you should have a bubble sprite and a background sprite that looks like a bathtub. You’ve got the game environment and the game objects ready to go.
But there’s one thing you have to think about… how will the game look on different web browsers, or different phones or tablets?

To account for this, resize your game so that it can be played at an optimal resolution for all web browsers. Click on the background and look to the left of the screen for a size property. Set the size of the background to 1366, 768. Every size in Construct 2 starts off with the width first and height second.

Now set your layout size to be the same size as your background. To do this, click anywhere in the layout and you’ll see the layout size property on the left of the screen.
Last but not least you need to set your play area or “Window Size” to the same dimensions: 1366, 768. Click on “project properties” right below “effects” and change the window size to match.

You have now set up your layout and it is ready for play… or is it? It looks like your bubbles are behind the background. To make sure everything is layered correctly, create a new layer by clicking on the “layers” tab and hitting the plus button.
Now drag layer 0 above layer 1 and name layer 1 “Background.” This is just for organization and clarity — you can name it anything you want to.

Nothing will change unless you put your sprites on those new layers you just made. Click on the bathtub and change its layer to “background.”

You may notice that you still can’t see the bubble sprite. Click on the layer tab and change layer 0 to transparent. This will get rid of the generic white background overlapping your actual background sprite.
You’re almost there, but not quite. Now you have to add some actual game playing functionality to your game. You want the bubbles to pop when the player touches them, so next you’re going to add a touch object and use this to add functionality.

**Adding Events**

By default the touch object will not do anything. So you now have to add an event so that when you click on the bubble, it gets destroyed.

This is where the Event Sheet comes into play.
An Event Sheet is where all of your scripting code will go to make your game interactive. After you make the bubbles pop, you’ll add a scoring function.

Double click on the Event Sheet.

Below is what your Event Sheet should look like. It should be empty without any events. To add your first event, double click inside the Event Sheet and click “Add event.”
Double click on the touch event and select “Touch.”

The “on touch object” event checks to see if an object has been touched. This check happens every frame. There are other events that have to do with touch, but for now we will only need “on touch object.” Construct 2 will ask you which object should be used. Target the bubble, and you’re done.

Before you go any further, make sure you name your sprites something that makes sense. As you get deeper into the project, the word “sprite” can get confusing and will slow down your workflow if they’re not all named intuitively. Name your bubble sprite “bubble” and your bathtub background “bathtub.”

Now that you’ve added a condition, you have to add an action. Without actions the game will not do anything. What you just added was a condition that will trigger an action whenever a condition has been met. In your case, when the bubble has been touched it will be destroyed. Click “add action,” select the bubble and destroy it.
TIP: If you want to find an action faster, type it in the search box at the top right of the action box. Prefill the box with the term “destroy” to access it without having to scroll down. This will increase your overall workflow speed. Game creation is fun, but it’s even better when you can do it faster!

This is what your screen should look like. Now you get to test it and see if it works. Click on the run layout (play button) at the top of the project and click on the bubble. It should destroy when touched.
Adding behaviors to your objects

Believe it or not you are halfway through. Of course, you have to make the game actually work. The next step is to make your bubbles float upward from the bottom of the screen. In order to make this work properly, you’ll need to add spawners, a bullet behavior and a few events.

Begin by clicking on the bubble, then go to “Behaviors,” select the plus button and add a bullet behavior to the bubble sprite.

To save time, type in “bullet” in the search box and select the bullet behavior. If you run the game now, the bubble will move in the direction of its angle. Even though it says 0 degrees, the bubble will move to the right by default. Zero is right, 180 is left, 90 is down and –90 is up. If this is confusing to you,

Construct 2 has a handle that represents the direction of the object. You can simply rotate this in the direction you desire. As you want the bubbles to rise from the bottom up, rotate the handle upwards, or type in –90.
Hit “Run” again, and the bubble should float up to the top. If it is not floating in the desired direction, make sure your angle is correct.

So far so good, right? Except you have a little problem right now. If you touch the bubble, it’s destroyed forever and that’s the end of the game. So you’d better do something about it…
Spawning multiple objects

Now that your bubble floats upward, you can spawn more of them and have a playable game. All you need to do now is add a score and a game over screen. First add a sprite to your game and call it “spawner.” Create five of them and spread them evenly at the bottom of the project. Take the paint bucket and color them any color. They will just be used to spawn the bubbles with events.

Remember how the angle will be zero by default? Well if you leave your spawners like this the bubbles will spawn to the right of the spawner sprite. Set all of the spawners to have an angle of -90 so the bubbles will spawn going upward.
Great! Now click on the Event Sheet and create a new event. This event will be a system “Every X seconds condition.” Set it to create bubbles at the target location every 1 second.
Next add another system action. This time create a bubble at the spawner’s X and Y coordinates. An object’s X and Y coordinates correspond to its location in relation to the top left corner of the screen as measured in pixels. So, an object whose X and Y is 100 X 100 is 100 pixels to the right of the left side of the screen, and 100 pixels down from the top.
You are done with the hard part. Click “run layout” and your game should look like this.

The game is doing exactly what you told it to do. You may be wondering why it is only spawning through one spawner instead of all of them. Turns out you didn’t give Construct 2 a specific enough routine to run. This time, add some randomness to it and hide your spawners.

You can add more than one condition to an event. Click on the event and right click it to add another condition.
Select the system event and click on “pick random instance.” An instance is a variation — or copy — of an object that is on the screen. Each of your spawners is an instance of the original spawner. You duplicated it four times, and each of those sprites is an individual instance.

You will be picking a random instance every 1 second. This will spawn a bubble randomly and have it float to the top of the screen.
Make your spawners invisible and run the layout. Your core game mechanic is just about finished!

Adding Scoring to the Game

A video game is no fun without a score. Adding a score to your game is really simple. All you need is a text object and a variable. Right click in your project and add a new text object. A text object is used to display text and dynamic text on the screen. Dynamic text is text that is changed using scripts, or events in this case.
Name the text “score,” change its color to white, and its size to bold, 24 px.

Click on the dropdown arrow next to the font to change the font size. You can also change the color to white by clicking on the “…” icon next to color. Pick any color you want. Just remember that you want a color that contrasts with the background and any other images so that it shows up clearly.
NOTE: There is a big difference between the object’s name and the text property. In the previous step, you changed the name to “score” but not the property name. The property name will be changed dynamically in the game as the player collects more and more bubbles. You can just leave it as “Text.” Every frame you want to set the text property to the player’s score. Right click inside the event sheet and create a new global variable called “score.”
Nothing would happen if you ran the layout right now. The score is 0, and you still haven’t set your text to reflect the player’s score. This part is fairly simple. Create an event that sets the player’s text to [“Score:” & Score] every tick.

**NOTE:** You can simply set the text to the variable “score,” but if you add a string you can actually display “Score: [player score]” on the screen. A string is a type of variable that contains text — a “string” of characters and numbers. The “” creates a string inside of a variable to be displayed. The “&” tells Construct 2 that you want to add another expression. An expression is anything that calculates or displays information on screen.
So now every frame your text will display the player’s current score. The player has no way to increase her score, so it will display “Score: 0.”

In order for the player to start increasing her score, you need to add an action to one of your conditions that you already have. Locate the touch event for the bubble sprite and add an “add to global variable” action. You can make the score increase by however many points you like. For now, choose 10 points.
Ta-da! You have a game with a score that is playable. To make things a little more challenging, add some strategy to the game by punishing the player if the bubbles escape the game. Similar to the score, you will add a “lives” text, and it will be roughly the same procedure. The only difference is that you’ll be subtracting instead of adding.
Adding Lives to the Game

First add a text object called “lives” and give it the same properties as the score. White, bold, and size 72 px. Then add a variable called “lives” in the event sheet. Give it a default of 3 lives.

Since you already have an “every tick” condition, just use that one to add your lives set text action. Set the text to [“Lives:”&Lives].
In order to determine if the player is going to lose a life, you have to check and see if the bubbles are outside of the layout. To do that, add a new event and select the bubble.

Select “bubble is outside of layout.” Then add an action that destroys the bubble and subtracts 1 from the variable lives.
Run the layout and see how far you can get without running out of lives. Once you reach 0, you may notice that the lives start going backwards into the negative. In video games there is rarely such a thing as negative lives. There are two solutions to this problem. The first is to create a condition that stops subtracting from the lives. The second is to stop the game once the lives are less than or equal to zero.

To end the game if the lives variable = 0, your condition will look like this:

[If lives <= 0 set endgame to 1]

An endgame variable would work really well in this case. In the next section you will be adding the end game mechanics.
Creating an End Game Condition

Now that you have a working game, you want it to eventually stop and make the player able to restart the game if she loses. You’ll create an end game variable that is triggered once the score is equal to zero. You will then do things such as hide the score and disable the bubbles’ clickability.

First add another text and call it “Game Over.” This time, change the property of the text to an actual string. This will not be dynamic text. It will only be shown once the player has lost the game.

Set the “game over” text to invisible. You only want to show this once the player has lost.

Now add a variable called “game over” inside of the Event Sheet. This variable will be used to tell Construct 2 if the game has ended.

It’s condition time again!
Create a new event and check to see if lives is <= 0. Then create an action that sets game over to 1.
GREAT!

If you were to run this right now, nothing special would happen. When game over = 1, you have to turn off certain things. For instance the spawner should
not spawn any more bubbles, and the score and lives should disappear. Not to mention you have to unhide your “game over” text.

Add a new condition to check and see if game over = 0 inside of the spawner event. If the player has 0 lives, it will not spawn any more bubbles.

To save time, copy and paste that same condition on the “bubble outside of layout” event. If a stray bubble escapes when lives = 0, it won’t keep subtracting from the lives variable.
Copy that same condition and add it to the “on touch event.” You don’t want the player to still be able to play while she has 0 lives, do you? How else are you going to get her to keep feeding you quarters?

You’re almost done.

In the same event, hide the lives and score text with a condition that checks to see if game over = 1. Use the search tool to search for the action “visible” if you get stuck. Set both to invisible. Now set the game over text to visible.
Lastly add an event that restarts the game two seconds after the game over screen is shown.
In order to allow the game to restart from scratch, simply reset all global variables, and the game will be complete! Run the layout and see how far you can get.
Now that your game is finished, you can add some extra details to the game such as sound.

http://www.superflashbros.net/as3sfxr/ is a great place to get video game sound files. You can generate sounds on the right and save the one you like as a .wav.

**Bring your game to life with sounds**

I chose the power-up sound and exported it to my computer as a .wav file. Construct 2 will only import .wav and .mp3 files.

Import your sound in Construct 2 by right clicking on “sound” and importing the .wav file.
Now that the sound has been imported, right click anywhere inside the project and add a sound object. Construct 2 will not be able to find the audio file if it doesn’t have the sound object added to the project.
Finally, add an action that plays the sound when the player clicks on the bubble.
Since you only have one sound, it will pre-populate the sound file you just imported. The tag section is where you can give your audio file a reference name just in case you want to deactivate it or mute it. This is usually used to toggle music on and off. Click on “run layout” and you now have a fully functional game with sound!
Changing the Game Metrics

You can mess around with the speed at which the bubbles ascend to make a more challenging game. For instance, try changing the bubble speed to 600-700. You can also decrease the rate at which the spawners create bubbles on screen. So instead of the spawners creating bubbles every 1 second, you can change it to every 0.5 seconds. This makes the game insanely harder to beat. Have fun!

Congratulations!

You’ve now completed your first Construct 2 tutorial. It wasn’t so hard, was it? In the next tutorial, you’ll kick it up a notch and create a game that’s slightly more involved.

Before you do, though, it’s a good idea to get in there and tweak your game a little bit. Save your project and then save it as a different name, like “Bubbles 1.”

Once you’ve done this, go back through and play with the variables, the angles, the speeds and even add some more sounds, like background music. Don’t be afraid to try something new. Even if you make a mistake, you can always reload the original project file and try again.

There’s no substitute for this kind of learning — actually getting in there and playing with the settings. If you feel you’re ready, move on to your next game project!
Now that you’ve successfully completed the Bubble Pop Madness tutorial, you’re ready to move on to something a little more advanced. You’re going to build on what you’ve learned so far and expand just a little by creating a hidden object game.

The goal of I Spy is to locate the hidden object that the game tells you to find. If the player finds the object, he gets a point. If he doesn’t spy the object within 10 seconds, the game goes to the next object and the player is not rewarded any points. The faster the player finds the object the higher the score.

Sounds pretty straightforward, right?

For this project you’ll be using interesting images of creatures, animals and other objects for the player to find. So that you don’t have to create the images yourself, we’ve provided all the game assets that you’ll need for this project. Just visit this link and download them: http://bit.ly/lspyassets
Start a new project

As you now know from the previous chapter, you have to start a new project in order to build a new game. So just like before, click the gear icon, and “New” and then “New empty project.”
Before you start adding images to your game, set the layout size to 1366, 768. Click “view” next to “project properties” and do the same for the window size.

Sound familiar?
Add a sprite object and import the image with all of the animals and objects. Name the image anything but “sprite.” For this tutorial, name it “background image.”

Since the goal of the game is to click on the correct object, you need to create at least 10 different sprites that will be invisible in the game. Create 10 different sprites with names that match what you see in the images below. Also make sure they are invisible when the game is played.
Place each sprite on the corresponding name. For example, if your sprite is named “elephant,” place that sprite on top of the elephant on the background picture. Resize the sprites as best you can. Here’s how mine looks:
Now add a text object, call it “Finder” and set the text size to bold, 24 px.
If the text suddenly disappears after you change the size, grab the handles on the text object and drag it outward until it’s visible.

Right click anywhere on the screen and insert a new touch object so you can detect when the player has clicked on a sprite.
Click on the Event Sheet and create a global variable called “CharacterNumber.”

Create a condition that checks to see if CharacterNumber = 0.
For the action, you are going to set your text to “Find the Elephant.”

Currently this does nothing. But once you add your touch event, you’ll add 1 to CharacterNumber and have the text display the next object you want the player to find.
Adding your first touch event

Once you are able to locate one object, you can simply duplicate the event until you have ten. Create a touch event for the elephant sprite.
Create an action that adds 1 to CharacterNumber.
You also have to make sure that CharacterNumber = 0 or the player will be able to click on the elephant anytime he wants instead of when you want him to. You don’t want people to be able to click on your elephant anytime they want!

Believe it or not, the hard part is over. To show that this works, copy and paste both events and change the CharacterNumber to 1 and the object touched to the pig. The events below are separated with a comment below so you can see the difference.

The above image is what you should currently have. You can duplicate these events eight more times, but wait — there are some more things you have to add to make the game complete. Once everything is added you can duplicate the rest of the events. It’s time to add your score and a countdown timer.
Adding a countdown timer

Now you can expand the game’s parameters by setting up a time constraint. This limits the player’s available time to find the hidden object and makes the game a bit more challenging.

Since you are already inside the Event Sheet, add a new global variable called “Timer.” Set its initial state to “start from 10” since it will be counting down.

Create a new text object and name it “Timertxt.” Place it right under the “Finder” text and set its size to bold, 24 px.
Now that you have a timer text, set it to display: “Timer:”&Timer. Remember that anything inside the parentheses is treated as a string. The “&” tells Construct 2 that another parameter is coming. Lastly, leaving the timer outside of the parentheses tells Construct 2 that you want to display a variable. Setting text every tick updates the screen to display data in real time.

For the actual timer, create an “every (x) seconds” event and decrease the timer variable by 1 every 1 second.

Currently your timer keeps going after it reaches zero. This is a problem. In order to make this work you need a condition to trigger once the timer has hit zero. Create an event that adds 1 to CharacterNumber and sets the timer back to 10.
This is how you’ll move on to the next object the player has to find.

It works! Try it out. Did you notice that when you clicked on the elephant the timer did not reset? You need to fix that. Copy and paste the “Set Timer to 10” event and place it wherever you have a touch event.

You are nearly finished with the basic setup of your game. The rest will just be cosmetics — making it cooler.
Create a new text object called “Scoretxt” and create a corresponding global variable called “Score.” In the “every tick” event create a new action and set the Scoretxt to: “Score:”&Score.

Create a new system action and “Add 10 to Score” on both touch events.
Finally, add a music sound when the player clicks on the correct animal. To make your game look more interesting, add a green checkmark sign that lets the player know he clicked on the correct object.

Insert an audio object to your game and import both audio files: Miss.wav and Found.wav.
Nice! You killed two birds with one stone. Head over to the Event Sheet and play the “Found” sound when the player has touched both objects.

Import the checkmark sprite and give it a fade behavior. If you forgot how to add behaviors to sprites, check out Chapter 2 and the Bubble Pop game tutorial.
Go to the Event Sheet and add an action that spawns the checkmark sprite from the object. So for instance if the player clicks on the yellow elephant sprite, a green checkmark will spawn there.

Locate the timer event and create a new action that plays the Miss.wav sound. A cool trick is to use the search box at the top of Construct 2 to find important code that is lost in the sea of events.
Click the “Clear Search” button to get back to the rest of the events. Make sure you do this so you won’t get confused trying to find missing events. They are still there.

Now it’s time to start duplicating events since you have everything set up. Duplicate the events eight more times until every object has a corresponding touch event and CharacterNumber event.

When copying and pasting make sure you change the proper settings or it will not work the way you expect. This means that each duplicated object, or instance, must have its own unique name.
The highlighted events and actions are the things you need to change. You can leave everything else the way it is.

Now you have a fully functional hidden object game! Play it a few times and make sure everything works the way it should. You’ll now add the final touches to it: a game start screen and a game end screen that displays the player’s score.

Create a new layout called “Menu” with an Event Sheet. Rename your “Layout 1” and “Event sheet 1” both to “Game.”
Just like the game layout, set the layout size and window size to 1366, 768.

Create a sprite and call it “Playbtn.” After that, create a text object and call it “PLAY.” Put it right on top of the sprite.
Drag the background image onto the screen and add some directions for the player.

Go to the Event Sheet and add an event that takes the player to the Game screen when he presses the blue PLAY sprite.
Finally, add a game over screen that shows the player’s current score. Create a new layout like before and call it “game over.” Set the layout and window size to 1366, 768.

Create a new text object called “YouWintxt.” Set the text size to bold, 49 px. Center it toward the middle of the screen.
Go to the “game over” Event Sheet and create an “every tick” event that sets the text to: “Congratulations! You Completed The Game. Your High Score Is:” & Score.

Last but not least, add a condition inside of the “game event” sheet that sends the player to the game over screen when CharacterNumber = 10. Because remember, you have 10 characters from 0-9, and if the player has clicked them all, the CharacterNumber will be 10, 9 being the very last object.
You are just about finished! You need a way for the player to get back to the menu. Copy and paste the button you created for the menu, and create an event that sends the player back to the beginning of the game. The game will continue to loop if you don’t reset the variables. Create an action that resets the global variables.

Congratulations! You Completed The Game. Your High Score Is: 100
Congratulations, you just finished your second game!

That probably seemed a little easier than Bubble Pop Madness, didn’t it? The truth is, neither of these games was very complex, although I Spy certainly had a little more to do. However, because you’ve already gone through some of this, it probably seemed easier this time around, didn’t it?

That’s how it works — game building is like putting up a building. You start with the foundation and then add more stories until you’re done. As you create more
games, you’ll find it easier, and it’ll also be easier to add more complexity as your range of skills increases.

As in your first game, we encourage you to play around with this one. Change the text, the timers and so on to see how your changes take shape.

When you’re ready, move on to Chapter 4!
Now that you have a couple of games under your belt, it’s time to move on to something a bit more complex. In this tutorial, you’re going to create a jumble art game. It’s a step up from bubbles and I Spy — but you can do it!

A jumble art game is a puzzle game in which your player has to put the pieces together as quickly as possible. When all the pieces are in place, he wins.

This tutorial is going to help you understand how to use animation frames as well as instance variables in a productive and efficient way. By the end of this chapter, you’ll start to see how to get the most functionality possible while using as few events as possible.

The more efficient you are at building your games, the better they’ll operate.

Before you begin, you can download the assets for this project here: http://bit.ly/Jumbleart
Get started!

To build any game, you’ll need to start a new project. If you recall in the first tutorial, a project is your game space. Start a new project by clicking on the gear icon in the top left of the screen. When a dropdown appears, select New. In “Layout properties” set the layout size to 1366, 768. Make sure the window size is the same size.
Setting up The Start Screen

Now that you’ve started a new project, rename the layout and Event Sheet “Start Screen.”

Right click inside the white canvas and import the Background image. Place it right in the middle of the screen and change its name from “sprite” to “Background.”
Right click outside the background area and insert a text object. The text should read “Art Jumble” in big letters. Set the color to white and the text size to bold, 100 px.

In the black box area, create instructions for the player. Insert a text object that reads “Learn cool art trivia while solving puzzles before your time runs out!”
Make the text white and the size 24 px, then set the horizontal alignment to centered.

To finish the Start Screen, you need a button for the player to press to send him to the game board. Before you add the start button, you need a layout first. Create a new layout and Event Sheet called “puzzle.”
On your Start Screen, import the start button sprite, place it in the middle of the black area and call it “Startbtn.”

Right click anywhere on the blue area and insert a touch object.
Now that your touch object is loaded, go to the Start Screen’s Event Sheet and create an event that sends the player to the “Puzzle1” layout.
Creating the Puzzle Game Mechanics

Click on the Puzzle1 layout and change the layout and window size to 1366, 768.

Drag the background image to the center of the screen.
Import the black background sprite and move it to the far left. Change its size to 206, 768.

Import the “PuzzlePiece” sprite and place it at the top left of the background.
Insert a text object and have it read: “Can you un-jumble Mona Lisa before the time runs out?”

Now insert the full Mona Lisa painting and put it in the upper right corner.
Create another text object and call it “timertxt.” Put this at the bottom left and make it white, size 72 px. You’ll return to this later.

Import the “EmptyBox” sprite and change its size to 144, 117.
Copy and paste the empty box eight more times to make nine empty boxes total. Don’t worry about how they are positioned… you will change that next.

These boxes are really messy. To snap to the grid to position them perfectly, click on the “view” tab up top and turn on “snap to grid” and “show grid.”
Align all of the empty boxes. Once everything is aligned, uncheck the snaps in the “view” tab.

Import the “mona1” sprite and change its size to 130, 102. Call the sprite MonaPieces. You will be importing all the pieces in one sprite.
Double click on the “MonaPieces” sprite and import the rest of the frames by right clicking inside the Animation frame window and clicking on “Import frames” then “From files...”.
Set the animation frame speed to zero by clicking on the default button. You don’t want these to play when the game starts.

Copy your MonaPieces sprite eight more times just like you did for the empty boxes sprite. You don’t have to worry about aligning these properly.
Currently all of the images are on animation zero, which is why they all look the same. Each piece of the Mona Lisa has its own frame — a way to determine what part of the whole image appears.

Each one of these images can look completely different by changing the animation frame. Click on the second MonaPieces sprite and change its frame to 1.
Change the third frame to animation 2, the fourth to animation 3, and so on. Do this until every sprite has its proper animation frame.

Don’t worry, you will jumble these images up using events a little later. You still have a few more things to do. To make these images draggable, add a drag behavior on the MonaPieces sprite.
You ultimately want these pieces to snap into place using events, which requires letting Construct 2 know which pieces go with which empty box sprites. To do this you will use instance variables and compare them to the sprite’s animation frame. If they match it will snap into place.

Remember, an instance is essentially a copy of an object. So instead of creating nine individual sprites, you create one object and then copy it and all of its associated features. Each copy of an object is given a unique name, or instance name.
Click on the empty box sprite and create a new instance variable called “instance.”
Give each empty box sprite an instance variable starting from zero all the way to eight. This process is similar to how you changed the animation frames on the MonaPieces sprite. The only difference is that you will be changing the instance variable on each. I added numbers on the screenshot to show you how you should be numbering each one.

Snapping Objects

Click on the Puzzle1 Event Sheet and create an event that checks to see if MonaPieces has collided with the empty box sprite.
You also want to compare to see if MonaPieces’s animation frame is 0.

Now create a condition by right-clicking on the event and choosing “add new condition” that checks to see if the empty box that you collided with has an instance variable parameter of zero.
Now that the conditions are met, add an action that drops the draggable Mona-Pieces sprite.

Lastly, create another action that positions the MonaPieces sprite directly in the middle of the empty box sprite. To do this set the position of MonaPieces to “EmptyBox.X” and “EmptyBox.Y”.

Test your game in the browser and watch how the first MonaPieces sprite snaps perfectly in the middle of the first empty box.
Before you start to copy and paste your events, add some sound effects for when the player places the piece in the proper spot. You will add music to the game as well.

Insert a sound object by right clicking anywhere on the blank canvas and importing the success, fail, and background music audio files. Notice in the image that you’re not creating a sound sprite, but simply clicking on the sounds folder and then adding the sound files to it.

Go back to the Puzzle1 Event Sheet and add an action that plays a success sound. In order to do this, you’ll need to put your sound effect into its own sound effects sprite and give it a unique name.
Add a new event that plays the background music when the game starts with an “On start of layout” condition.

Copy and paste the snapping event eight times and change the animation frames and instance variables accordingly.
You should now have a game that snaps perfectly like the image below.

Object Jumble

Right now, your game is far too easy. You need to jumble the images each time the player starts the game. To do this, you will make the images bounce around really fast for half a second before the game starts.

Create a sprite called “border” and duplicate it four times to create a border around the black area.
Give the border sprite a solid behavior. You have to do this because bullets can bounce off of solids, and you want to contain these images inside the black box while they bounce around.

Click on the MonaPieces sprite and give it a bullet behavior. Change the setting “Bounce off solids” from “No” to “Yes.”
If you play the game right now, the images will bounce back and forth. To make it jumble, you have to change the angle of motion of the sprite so it can go in a random angle each time the game starts. When it stops you will set its angle back to zero.

Create an “On created” event, and set the MonaPieces sprite’s angle of motion to random(0,360). The random expression will randomize any two numbers inside of a parentheses with zero being its normal angle and 180 being completely flipped upside-down.
You don’t want the pieces to twirl around forever. Create a system action that waits one second then sets the bullet speed to zero and sets the angle back to zero.
Pretty cool, right? There are a few tweaks you can add to make this even better.

First, you are still able to drag the item while it’s jumbling. This is a problem. Set the initial state of the sprite to “disabled” and enable it after the one second has passed.
Set the border to invisible.
Adding a Winning Condition

In order for the player to win, he needs to put all the pieces in their proper places. But how does Construct 2 know if the pieces are in the correct positions? You will have to establish the exact location of the object. The easiest way to do this is to create a test object that displays the MonaPieces sprite’s position when the player clicks on it.

Create a text called “position” and place it at the bottom. Make it white so you can see it inside the game.

Go to the Event Sheet and create an event that checks to see if the player has clicked on the MonaPieces sprite. If so you want to set the text to: (MonaPieces.X&","&MonaPieces.Y).
As you can see in the picture above, when you have all the images snapped you can click on them individually to see in what current position the image is located, 320 being the X position and 288 being the Y position.
Create an event that checks to see if the MonaPieces sprite with the animation frame of 0 is at position 320, 288. Your coordinates are probably slightly different from mine, so click on the first image to see where it’s positioned.

Add 1 to this variable whenever an object is in the correct position. Once all nine objects are in the correct positions, you will check to see if complete = 9. If so, you will send the player to the winning screen.

To begin, create a variable called “complete.”
Now set “add 1 to complete” inside of the event you just created.

Add a “trigger once while true” system condition so it won’t keep adding 1 while true.

Do this for the other eight images.
Now copy and paste this event eight more times and change the respective X, Y, and animation frames. Remember that each item in the same column will have the same Y coordinates and each item in the same row will have the same X coordinates.

Create a new text object called “You Win” and place that inside of the black box area. Make it invisible. You will unhide it when the player has completed the game.
Create an event that checks to see if complete = 9. If so, make the “You Win” text object visible. After that, create an action that waits two seconds and sends the player back to the main menu.
The last thing you need to add is the “you lose” part of the game with a fail sound. You already have your timer text set up, remember? Create a variable called “Timer” and set it to 15.

Create an “every tick” event and set the “Timertxt” to the Timer variable.
Now create an event that subtracts 1 from timer every 1 second.

Create an event that checks to see if the timer is less than or equal to zero. If so, play the fail sound and set the “You Win” text to visible and change the text to “You Lose.” After that add a “wait” action for 2 seconds and send the player back to the start screen. Destroy the timer text. Make sure you trigger this only once.
Play the game in a browser, and it should take you to the start screen if you let the time run out. Finally, you need to reset the global variables when the game starts.

Scroll up until you see the “On start of layout” event and create an action that resets the global variables.
Congratulations!

You’ve now completed your Jumble Art game. Did you notice how handy it was to create instances of one object and its associated events so that you didn’t have to re-create them for all nine?

In programming, this is called “object-oriented programming.” This means that you add as much functionality into the parent object as you can so that when you create an instance — or copy — of that object, all of the functionality is inherited with it.

Go back and play around with this game. You can even add more pieces to make it more difficult, or change the timers. When you’re ready to move on, go to your next tutorial!
Now that you’ve gone through three very different games, you’re ready to add a little more action.

In this game, your player’s goal is to fly her plane through a shower of missiles and try to reach the airbase. If she can do this without getting hit more than three times, she wins.

Of course, every time a missile touches the plane, the player loses a life.

Before you begin, you can download the assets for this project here: http://bit.ly/Planeassets
Get started!

By now you know what to do. Start a new Construct 2 project and choose “new empty.” As before, in layout properties, set the layout size to 1366, 768. Make sure the window size is the same.

You should now have a blank canvas as shown on the right.

Import Your Assets

Before starting any new project, import all of your assets ahead of time so you won’t have to backtrack. To import sprites, right click anywhere on the canvas and select “Sprite.” To import a tiled background, follow the same steps and select the tiled background image you want.
Import the following assets into your scene: Background.png(TiledBackground), explosion1.png(Sprite), explosion 2.png(Sprite), home.png(Sprite), plane.png(Sprite), tracker.png(Sprite), missile.png(Sprite), left-arrow.png(Sprite), and right-arrow.png(Sprite).

You will import the sound files after you are nearly finished with your game. For now, import the sprites. If you need a recap on how to import sprites, go to the bubble game tutorial in Chapter 2.

For all of the objects that will be colliding, such as the plane and the missiles, make sure their collision boxes are “set to bounding box”. If you need a recap on how to make collisions for your sprites, refer to the bubble game tutorial in Chapter 2.
Now rearrange your canvas so it looks like the picture below. Spread out your tiled background so it expands across the entire game field.

After that, create three layers that contain the proper items. Place the background on layer 0. Place all of the in-game items on layer 1.

Finally place all of the HUD items on layer 2. If you’re confused about whether or not an item is a HUD object or an in-game object, just make sure it is above the background layer so it can be visible. You can organize your layers any way you choose. We prefer three. Lock your background so it doesn’t slide around while you’re working.
Player Movement

Now that everything is properly arranged, begin adding some game functionality. Add a touch object to your game. You will use it to check and see if the player has touched on the left or right arrow, which moves the plane accordingly.
Now that you have a touch object, give the plane an “8 Direction” behavior so it can move inside of your game. Set the “Set Angle” property to “no” so it won’t flip when it moves.

Create an event that checks to see if the player has touched the left arrow button. After that, simulate the player pressing left. This will allow the plane to move left. Copy and paste that event and change it for when the right button has been pressed.
Now that the player can move, make sure she doesn’t move outside the layout. Add a “bound to layout” behavior to the plane so it cannot leave the play area.

Spawning Random Missiles

Now that your player can move, it’s time to create the falling missiles. The player will have to avoid these to win the game. If you forgot how to spawn random objects, refer to the bubble tutorial in Chapter 2. Create six spawners and spawn a missile from one of them every 1 second. Your missiles should also be pointing downward with a bullet behavior attached to them. Make sure the yellow spawn boxes are invisible after you’re finished.
Great! The core of your game is complete. Everything else you add after this point is simply cosmetic, like the score, sounds, lives and a winning/losing conditions.
Creating Lives

At this point, if a missile hits the player, nothing happens. You have to show the player that she is indeed losing health as her plane comes into contact with the dangerous missiles. In the bubble tutorial you added a text-based representation of the player’s lives. In this tutorial, you will represent each of the player’s lives as a small plane on the top right of the screen. Each time the player gets hit, one of the three planes will get destroyed until there are no more planes remaining. So far you only have one of the three. Import two more planes and name them “life2” and “life3.”

You will use a variable to compare how many times the player has been hit. Create a variable called “lives” in the Event Sheet window. Set lives to three.
Make an event that checks to see how many lives the player has and if the player has been hit. If so, destroy the missile and subtract 1 from lives. Destroy a corresponding life as well.

Create an “on collision” event between the plane and the missile. Next check to see how many lives are left. Finally subtract one from lives, destroy the missile and corresponding life. Repeat this three times.

Play test your game and make sure everything is working correctly. At this point the lives should be disappearing as the player gets hit. We will come back to these events to add sound and a game over condition a bit later.
Creating a Win/Lose Condition

In order for the player to win she has to make it to the base without getting hit. Now you will add that functionality to the game. This part is fairly simple. Add a bullet behavior to that small plane and set its speed to 5. Once it reaches the base, you will take the player to a win screen.

Create a new layout called “win” with its own Event Sheet, and take the player to this new layout once the small plane touches the home sprite.
Now create another layout called “Lose” with its own Event Sheet like you did for the win.

Go into your game Event Sheet and create an event that checks to see if the players lives = 0. If so, send the player to the lose layout.
Adding Sound and Music

One of the most important, and yet seemingly minor, elements that really makes a game fun is sound and music. It adds another dimension to the game play.

To add any type of audio to your game, right click anywhere on the canvas and add the audio object.
Whenever the player gets hit, add an explosion sound with a nice explosion sprite. Import the explosion sound, then go to the Event Sheet and create an action that plays that sound.
Before you add the explosions, you have to change the background to be invisible. Currently it has a black background, which you do not want. Click on the explosion sprite and set its blend mode to additive.
Give both of the explosions a fade behavior so they can disappear after a few seconds once they’re spawned.

Now add an explosion for each plane and missile collision. To get this to work, create an action that spawns the explosion on layer 1, which is your “In Game” layer. Whenever you spawn an object it will default to spawning on layer 0. Always change it. Do this for all three events.
Import the music.

Create an “on start of layout” event and play the music. Select “looping” and create a tag called “music” just in case you want to refer to that audio file in the future to mute, pause, or play the music with events.
Currently you have no way for the player to get back to the game after she wins or loses. Create a play again button, and when it’s pressed send the player back to the game. You cannot create an event that checks to see if the player has clicked on a text object, so create a long sprite. Make it invisible.

Click on the “win” Event Sheet and add an event that takes the player to the “Game” layout after she clicks the play button.
Now that you have the winning screen all set up, do the same steps for the lose screen. The only difference is putting “You Lose” at the top.
Create an event inside the “Lose” Event Sheet that sends the player to the game after she clicks the play button like you did earlier.

The final thing you need to do is reset your global variables when you go back to the game screen. If you don’t, it will continue to go to the win/lose screen since the lives will still be at 0. Go to the Game Event Sheet and reset all global variables when “on start of layout.”
Congratulations!

You’ve now created your first “real” video game, meaning it has all of the elements of a classic video game — a character (the plane), enemies (missiles), a goal (make it to the base), and lives that are taken when the player is hit. You’ve also added sound effects and music to give it that real arcade feeling.

Now that your plane game works, go back and tweak some of the features. Change the speeds of the plane and missiles, add more or reduce the number of missiles, and so on.

Get creative. You could add another dimension by including “power up” objects that your players can fly over. Maybe they get a shield that protects them for 20 seconds. You can add random parachutes that give the player points when flown over. When they get X points, they could even get another life. Let your imagination play around.

The sky is the limit — so have fun and when you’re done, move on to your final tutorial!
Sound Board

Are you ready to create a sound board game? This kind of game does not have an end goal. The player will be able to click on various instruments to play a corresponding sound. For this tutorial you will be using a drum set.

Before you begin you can download the assets for this project here — http://bit.ly/Soundboardassets
Start a New Project

You’re already a pro at this step. Don’t forget to set the layout size and window size to 1366, 768.
Name your layer and Event Sheet “Game.”

Create a new layout with an Event Sheet and name it “Menu.” As before, set the layout and window size to 1366, 768.
Creating A Menu

To create a menu, first import the black background sprite onto your canvas by right clicking anywhere on the screen and adding the sprite object. Name the sprite “MenuBackground.”
Drag the Menu Background image and place it in the middle of the screen the best you can. Just make sure there is no visible white space.

Open up the “Drum” image as a sprite and place it in the center of the canvas. Name this image “MenuDrum.” You will be using the same image inside of your game, so naming it “MenuDrum” will let you know this particular sprite is only for the menu.
You’ll need to edit this image some more and add a slight grey overlay that goes across the bottom of the image. Double click on the MenuDrum sprite. You should now be inside the sprite editor.

Click on the “Alpha” area and set the alpha to 50. The alpha changes the opacity, 255 being fully opaque.
Click on the minus button to zoom out.

Lastly, click on the rectangle tool and draw a box on the bottom of the sprite. Since the alpha is currently at 50, it should be translucent. If it’s not, make sure you’ve turned down the alpha and that you have selected the black color.
Right click anywhere outside the canvas and insert a text image called “Drum-Set.” Place it over the opaque area. Make sure the text color is white and that the font is at least 24 px.
Now create a text at the bottom of the screen that says “Touch or click on the drum set to play.”

Right click anywhere outside the play area and insert a touch object.
Create an event that sends the player to the game screen as soon as he clicks/touches the drum sprite. It should take you to a white screen since you haven’t worked on the “Game” layout yet.

Creating Your Game

Click on the “Game” layout and import the background as a sprite. It’s 1366, 768 so it should fit snugly on your canvas. Name the layer “background.”
Create a new layer called “Game.” All of your interactive game objects will go here.

Right click anywhere outside the canvas and insert a new sprite. Color it dark blue. Remember if the color is faded, make sure to change the alpha back to 255. Name the sprite “Panel.”
Duplicate the panel sprite three more times to make a total of four.

They may not all be aligned, so click on all of the panel images by holding down the CTRL button. Set the X position on all of them to 748. This will make sure they are lined up properly. You can then separate them the best you can vertically.
This is looking very good. Next add a music button to each of the panels. Right click anywhere outside the canvas to insert a new sprite. Select the “playbtn.”

The play button’s collision is not how it should be, so think back to the very first tutorial in which you changed the collision area to bounding box. Do that here by right clicking inside the sprite and selecting “set to bounding box.” Now, when the player clicks on the play button, he will have a big enough collision area to click on because of all the space you have made available.
Copy and paste the play button three times and place them on the panel spritesw.

Right click anywhere on the empty part of the canvas and insert a text object. Set the color to white and the size to 24 px. Name it “instrumenttxt” and give the property text value “Drum.” There is a difference between the name of the text object and the property text value. The name of the object simply names it for your own purposes, and the property text value is what the game displays on the screen during game play.
Duplicate the text object three more times and change the property text value on each to match what’s below. The second text should say "snare," the third, "cymbal," and the fourth, "hit."

Insert the drum sprite and place it next to the play icon. Name the sprite “instrument.”
We will be using the same sprite while taking advantage of Construct 2’s animation frames. Double click on the drum and select “add frame” in the animation frame window.
Do this two more times and insert the remaining snare and hit.

By default the game will automatically run the animations, and you do not want that. Click on the default animation button and change the frame speed to 0.
Exit the animation screen and duplicate the instrument three more times.

Now set the other three animations to different frames. Since you imported three more sprite images within the same sprite, you can tell it which frame (or image) to display. Click on the second image, scroll down and change its initial frame to 1.
Now change the third instrument to frame 2 and the fourth instrument to frame 3.

Adding Sound

Now it’s time to add some sound to your game. Right click anywhere on the blank part of the canvas and import the four sound files located in the assets folder.
Now that your sound object is added, go to the Event Sheet and create an event that plays the drum sound.

Variables

Now you need to create an instance variable to distinguish each play button. A variable is a container that can hold a number or boolean value. A boolean value is simply a true or false statement. An instance variable is simply a variable nested inside an object. You have to create an instance variable so you can give each play button its own number. This way you can say, “Play button with the instance variable 1 should play instrument sound when clicked.”

Click on the play button and give it an instance variable called “instance.”
By default the value of the variable will be 0. That is perfectly fine. For the second button change it to 1, for the third change it to 2, and for the fourth change it to 3.
Now go back to the Event Sheet and add the condition to check and see if the play button that was clicked has an instance variable of 0 since that is your first instrument on the list.
Duplicate the event three times by clicking on the event, pressing CTRL+C to copy it, and then pressing CTRL+V to paste it. Change the instance variables accordingly.
Currently the three other events are still set to play the drum instrument. Change them so they match what is displayed in the game. The order should be snare, cymbal, and hit.

Congratulations! You just made a sound board game inside of Construct 2!

Let's add a few finishing touches to this game. Create a sprite called “GoBack” and place it over the left arrow button so the player can go back to the main menu just in case you wanted to add another sound board. Make the sprite invisible.
Create an event that sends the player back to the main menu.

You can pat yourself on the back now, because you just completed your final tutorial.
Introduction

The next step in your evolution is to publish your game and make it available to millions of potential customers around the world. Given that the process is free and easy, we will use the Amazon Appstore as an example to introduce you to creating a developer account and publishing your game.

We will show you how to take your Android game project, test it with Intel XDK and then export it to the Amazon App Store.

Section 1: Registering on the Amazon App Store

Let’s visit developer.amazon.com/public

Once you are there it will bring you this page:
In the upper right click ‘**SIGN IN or CREATE FREE ACCOUNT**’ once there follow the account creation process or if you already have an Amazon account, sign-in!

We will come back to this page later! Now that your account has been made and you are logged in, let’s make a quick Construct 2 project!

**Section 2: Exporting our game for testing with Intel XDK and exporting for the Amazon Store**

Before you begin this section download **Intel XDK and register an account**. The software is free for Mac, Linux and Windows and is the easiest way to build our Construct 2 file. You can download it here: [https://software.intel.com/en-us/intel-xdk](https://software.intel.com/en-us/intel-xdk)

When you open up Construct 2 and make a New Project (Ctrl-N) you are given a box with a search option. In this box type the word ‘**Mobile**’ and **double-click** on the first project called ‘**Template: Infinite Jumping**’.
Once you import it you will have a game like this.

This game is already finished for us so instead of having to make our own (feel free to do so) all we have left is to fill out the project file properties. If you don’t see the properties window then click anywhere on Layout 1 and it will pop up. Otherwise, click the ‘View’ tab and make sure to check the ‘Properties’ tab. Once it is open, click on the Project Properties ‘View’ button.

Once in the properties make sure to fill out all of the ‘About’ information. To export, the minimum requirements are the Description and the ID. Fill out the description with your own game name, version, description and ID. For the ID make sure you enter it with a ‘reverse domain’ name. So if your website is construct2.com then call your ID com.construct2.main. You can pick another name instead of ‘Main’ such as ‘Game’ or your username.
NOTE: Before this step make sure you save your project to a location you can easily locate. After you save the project to your desired location, at the top of Construct 2 click ‘Export Project’.

Once you click export it will bring up this window with a whole bunch of export types. For this example we are going to select the Amazon Appstore and hit ‘Next’.

When you hit the ‘Next’ button we must pick the folder to export to. Your original save location and this location should be different. Make a new folder that you can easily locate and hit next!

The next step in this export process is to make sure the type is set to ‘Packaged app’ specifically for the Amazon Appstore. You will need to make sure Java 8+ is installed. When this is selected hit ‘Export’!

Once you hit the ‘Export’ button the project will load.

When it finishes exporting ‘Open the destination folder’.
With the ‘Destination Folder’ open we are going to need to create a .zip of it for our Amazon build.

To do this we will select everything, right click and send to compressed .zip file.
If you have free software such as WinRar you can also add this to archive and select the .zip file. By creating this .zip file you are creating the final build ready to upload to Amazon.

Rename this .zip file and leave it for now, we will come back to it when we upload to Amazon.

Let’s open up Intel XDK. You will have to sign in with your account. Once you do, locate ‘Import Your HTML5 Code Base’. This software will allow us to test our game on a mobile device emulator. You also have the option to build your game to Android, IOS or Windows straight from XDK.
Navigate to the folder in which you saved your exported project. Intel XDK is going to act as our way to test our game on our mobile device.

Once you hit ‘Ok’ type in the name of your project and hit ‘Create’.
Hit yes to both of these then press ‘Continue’. Intel XDK will take care of all the importing and add its own files to our project.

After hitting continue you will be taken to the index.html page in your ‘Destination Folder’. To make sure our project is working let’s click on the ‘Emulate’ tab.
In the ‘Emulate’ tab we are able to view our game and play it as if it were on a mobile device. This is great because with this you can save your project in Construct 2 and hit the refresh button in the top-left to constantly test the latest version of your game.

The ‘Devices’ drop-down gives you options as to which device you are developing with to test aspect ratio and compatibility.
For this example you can use the arrow keys in the ‘Emulator’ tab to control the game but where XDK really shines is with it’s accelerometer tab. If you scroll down and click on the ‘Accelerometer’ tab you will see a device and a list of its positions. Put your mouse over the device and click and drag to move it left to right. This will let you test to see if your ‘tilt’ controls are functioning correctly!

Another perk of using XDK is to test on your mobile device. You can either plug your device in via USB or use the same WIFI to host your test build. In the top-left corner you can see a ‘Mobile’ and a ‘Wifi’ button. If Mobile is selected, plug your device into your computer and hit push files. However, none of this can be achieved without the companion app.
Intel App Preview is a free app for both Android and Windows devices. Install this onto your device and make sure you are on the same wifi. Even if you use the USB connection, you need this app to run your game. Open the app on your device and with XDK hit push files on the ‘Test’ tab. You will see a play button, select that and your game will run on your device.
Section 3: Uploading to the Amazon Appstore

Now that we have gone over how to test your game, let’s go back to Amazon and upload it to the app store. The .zip file we made is going to be our final build which is why it is important to thoroughly test your game through XDK to make sure everything is working the way you intend it to.


When you sign in you are given a Dashboard that has a button to ‘Add a New App’. Drop the menu down and select Mobile Web. Select this since we built a packaged app for Amazon from Construct 2.
Once you hit ‘Mobile Web’ the page your are brought to is this. Amazon wants the basic information for your game and the category it belongs in. Make sure to fill these out and hit ‘Save’.

The next tab is for our pricing and publishing. Select where you want Amazon to publish your app and the pricing you would like. In this instance we have the app for free in every available Amazon market.
If you selected paid, you will need to pick a base list price so Amazon can convert it for other markets. When you decide hit ‘Save’.

This next tab is the information for your app. It’s where you get to create your Amazon page and therefore it is extremely important. Make sure you take your time to write the best description for your application!
The next tab is to upload all of our artwork for our Amazon page. We need to upload our icons, screenshots and any trailers you may have made for your game! Hit ‘Save’ when you are done.
Next we must give our game a content rating. If your game has any violence or intense themes then make sure you select that and follow the process Amazon suggests. Otherwise, select ‘None’ if your game doesn’t have the items listed.
Finally, upload your .zip file in the last tab. Make sure this is the final build of your game (best to .zip everything when you are finished) and leave the Launch path as ‘/index.html’.

Once you pick all the devices you know your game runs on, leave a note to the Amazon testers in case there are things you want to explain. They will go over quality assurance and let you know how your game fares among testing. Hit ‘Save’ to continue.
NOTE: Don’t create the zip file until you are ready to upload. Then, when you are ready make re-export so you have a clean version without the XDK files included.

Now that you have been through the process of taking a mobile game from Construct 2 to exporting it as a packaged app for the Amazon Appstore and using Intel XDK to test each build, you are ready to publish! Hit ‘Submit App’ and enjoy your hard work!

As we talked about earlier, we used publishing to the Amazon Appstore as an example because it is free to create a developer account for the Amazon Appstore and publish your game. However, you can follow similar processes to publish to other app stores like Apple App Store, Google Play and Windows Store. For example, you can follow the steps outlined at this link to publish the same game to Windows Store. [http://bit.ly/PublishToWindows10](http://bit.ly/PublishToWindows10)
Now that you’ve completed five successful game tutorials, we want to take this opportunity to congratulate you. It’s no small feat to have gotten this far, and no matter how quickly or slowly you got through it, you made it to the finish line.

By now, you should have a fairly solid understanding of Construct 2 and how to create virtually any game you want. If you’re still interested in moving forward, then it may be a good idea to get the paid version of Construct 2.

Whether you’re going to make game building a hobby or are considering it as a career, you’ve certainly walked through a door and have expanded your imagination. Once expanded, your imagination can never shrink back again!

Take it a step further

We’d like to propose that you go and work through a 6th tutorial. However, in this tutorial, there will be no help, no walk through and no helpful screen shots to keep you on track!

For your 6th tutorial, try and come up with something from scratch and then make it happen with what you’ve learned in Construct 2. You can do it. You can always use this book as a reference as well as the projects you’ve now created.

So what game should you build?

Well, that’s up to you. You may want to start by creating a copy of a game you like to play now. Pick a fun game and try and make your own version of it with your game builder. This is a good first “on your own” method because at least you have a template to follow.

Maybe you want to try and make a Space Invaders game, which is not unlike our Plane game, right? Maybe you’d like to create a Donkey Kong level, or even a maze game in which the player walks a character around and tries to find her way out.
Then again — you may want to start from scratch. If so, here’s how to plan a game before you sit down and design it.

**Plan your dive and dive your plan**

The Navy SEALS use this motto, and it’s a good one. When you’re doing something complicated, it’s a good idea to sketch out exactly what you want to do before you sit down and try and implement it.

**Step 1 — What will your game be?**

First, think of the kind of game you want to build. Will it be a puzzle, a side-scrolling adventure game, a flying game, a submarine shooting game? Maybe you want to create a small version of a 2D adventure game like Final Fantasy.

The point is, you need to decide what type of game to build. Then you can move on to what functionality, or game play, your new game will feature.

**Step 2 — Gameplay**

Now that you know what type of game you’re building, you need to figure out how it will work. For example, will it have a character, and if so, what will the character do? What are the challenges? Are there different levels, scoring, lives?

Jotting down every conceivable action and reaction will help you to keep track of everything.

**Step 3 — Game assets**

Now, you may or may not be a graphics artist. If you aren’t, that’s okay. There are lots of places online where you can download sprites, backgrounds, animations and more. Additionally, there are also many resources for sound effects (FX) and music, so don’t worry — you should be able to give your game the look, feel and sound that you want.

**Step 4 — Plan your layouts**

Now think about all of the screens that you’re going to need to create for your game. Perhaps you’ll need a start screen, a win screen and a lose screen. Of course, there is the game screen, or screens, too.
Why not jot these down? Use this as a visual reference to help you keep everything straight. Then, you can list the functionality, texts and other things that go in each layout.

**Step 5 — Start designing**

Now get going! Start with the simple things first, such as start, win and lose screens. Then move on to your game itself. Think it through and test often.

Always try to remember the OOP (object-oriented programming) rule that you never want to write the same code twice. Just play around and get creative and you’ll be amazed at what you invent!

We hope you’ve enjoyed this short introduction to mobile and web game creation. Our sincere hope is that you’ve been excited by the possibilities and will continue to use your imagination and the Construct 2 — or any other — game engine to bring your creativity to life.
About the authors...

In this book three game and app developers have collaborated to set their decades of combined experience, insights and expertise at your fingertips.

These three men share a love of creativity. As the true goal of this book is to open up your mind and give you a powerful tool with which you can bring your imagination to life, they’ve put more than simple information into this work. They’ve poured a lifetime of passion and hope into the recipe, too.

Ankur Prasad

Ankur Prasad wants to live in a world filled with innovative tech startups, next-gen mobile applications and Pumpkin Spice Chai Tea Lattes (yes that a real thing!) Presently, he is the head of partner marketing for the Amazon Appstore. Rob Pulciani, head of consumer and developer marketing for the Amazon Appstore called him one of his “smartest, most creative, industrious and disruptive (in the best way) colleagues”. An expert in the mobile application and game industry, he has a proven track record in startups and developers of all levels of experience.

He has a Masters in Management Sciences with a focus on strategy and entrepreneurship from Stanford University. As Vice President of the Business Association of Stanford Entrepreneurship Students (BASES), he founded the Entrepreneurs Bootcamp where he brought together top startup founders, venture capitalists and technology industry experts to educate Stanford students on how to launch successful startups. At Microsoft, he was the head of student developer relations and outreach and led a team of campus ambassadors in 100 universities to introduce students to mobile app and game development. He re-invented the Microsoft Imagine Cup, the largest university technology competition in the world, into a start-up accelerator to identify, educate and fund high potential student startups. Suffice to say, most of his waking hours (and sometimes sleeping) were consumed by the, then, only fledgling world of start-ups and enabling budding talent to achieve their entrepreneurial ambitions.

Following his tenure at Microsoft, he started the developer education and training practice for the Amazon Appstore. He launched the largest Android pro-
graming course in the world with over 250k enrollments to date. Recently, he is working closely with top app development platforms, ad networks and online education platforms to help developers and startups realize their visions and build successful ventures. When not dreaming up the next ‘unicorn’ in the world of apps, he is known for his many attempts at trying to achieve a single digit body fat percentage (17th times the charm), his love for discovering esoteric anime and pretending to be a gourmet chef.

**Allen Wu**

Allen Wu is the founder of Wu Tango Media, a mobile app development firm that has published 58 mobile games with more than 3.5 million downloads on iOS, GooglePlay and Amazon. Wu also leads the development experience for Microsoft game and app developers. He will tell you his career started in front of his Nintendo, but his professional career started with a degree in project management from the University of Washington. He then led the advertising campaigns for the Amazon app store and received the coveted and prestigious “Just Do It” award, which was presented to him by none other than Jeff Beazos himself along with the entire Amazon leadership team. He then moved into Amazon’s Game Studios department where he led the user acquisition for second-party mobile games and shipped four triple-A game titles. Wu’s work has been published in PC World, WebProNews, The Next Web and GeekWire.