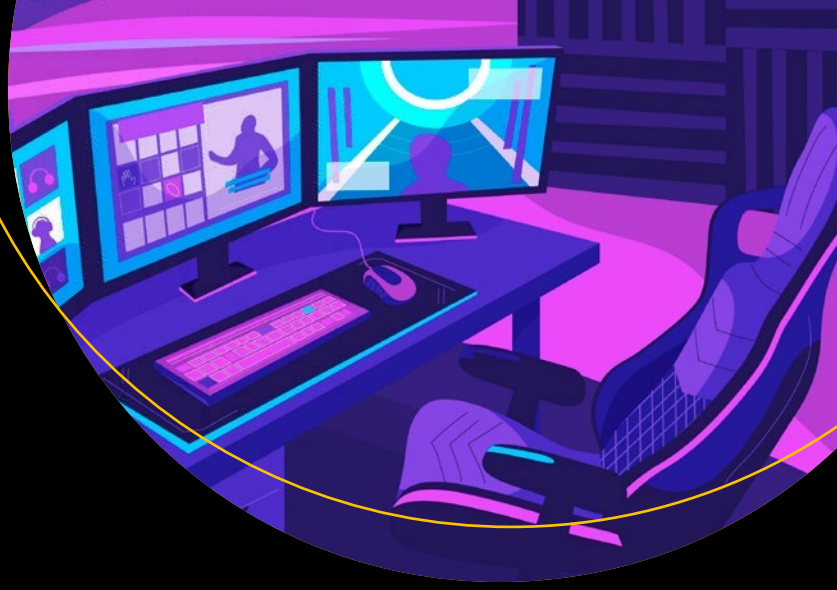

GameMaker Fundamentals

Learn GML Programming to Start Making Amazing Games

—

Ben Tyers

Apress®

GameMaker Fundamentals

Learn GML Programming to Start Making Amazing Games

Ben Tyers

Apress®

GameMaker Fundamentals: Learn GML Programming to Start Making Amazing Games

Ben Tyers
Worthing, West Sussex, UK

ISBN-13 (pbk): 978-1-4842-8712-5 ISBN-13 (electronic): 978-1-4842-8713-2
https://doi.org/10.1007/978-1-4842-8713-2

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Spandana Chatterjee
Development Editor: Spandana Chatterjee
Coordinating Editor: Mark Powers

Cover designed by eStudioCalamar

Cover image by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Apress Media, LLC, 1 New York Plaza, New York, NY 10004, U.S.A. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub (https://github.com/Apress). For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

Table of Contents

About the Author

Ben Tyers is an expert GameMaker user, developer, coder, and trainer, with over 10 years of experience in GameMaker. He has authored several books on GameMaker for game application developers. He runs a popular gaming website, which features accessible games that can be played by gamers with physical disabilities, with around one new game added every month. He has learnt GML coding to a reasonable level and has picked up many skills, tips and tricks, and methodology for making games in GameMaker.

About the Technical Reviewer

Mark Alexander has been a technical writer for the last 12 years, starting his career working for YoYo Games in Scotland, where he was in charge of the GameMaker documentation for almost ten years. After that, he moved on to work at Asobo Studio in France, where he is doing the technical documentation for the Microsoft Flight Simulator SDK.

In his spare time – apart from reading sci-fi, collecting Lego, and playing video games – he creates and publishes small indie games and also creates assets for people to use in GameMaker. Any time he has left after that, he dedicates to the administration of the GameMaker Community Forum.

Acknowledgments

Various Audio and Graphics Used in Game and Challenges: Permission Given to Use by Yoyogames.com. Please Do Not Redistribute These Assets.

Graphics:

Aeroplane: Pzuj/Opengameart.org – CC0

Animated Boy Sprites: Bevouliin.com/Opengameart.org – CC0

Arrow: oglsdl/Opengameart.org – CC0

Asteroids: Jasper/Opengameart.org – CC0

Bag: Flowly/Opengameart.org – CC0

Ball: Gothicfan95/Freepik.com

Barrel: Manenwolf/Opengameart.org – CC0

Beach Scene: Tgfcoder/Opengameart.org – CC0

Bee: Carriejanesart/Opengameart.org – CC0

Birds: Bevouliin.com/Opengameart.org – CC0 && CC-BY 4.0

Blue Character: Kenny/Opengameart.org – CC0

Boats: Kutejnikov/Opengameart.org – CC0

Bomb Sprite: Znevs/Opengameart.org – CC0

Boss Missile: Bevouliin.com/Opengameart.org – CC0

Boss Plane: Bevouliin.com/Opengameart.org – CC0

Bullets Set: Bevouliin.com/Opengameart.org – CC-BY 3.0

Buttons: Game Developer Studio/Opengameart.org – CC-BY 3.0

Crosshair: hc/Opengameart.org – CC0

Caveman: Hapiel/Opengameart.org – CC0

Cloud: Bananaowl/Opengameart.org – CC-BY 3.0

Clouds: Bevouliin.com/Opengameart.org – CC0

Crate: Rfc1394/Opengameart.org – CC0

Crocodile Character: Bevouliin.com/Opengameart.org – CC0

Dinosaur: Pzuj/Opengameart.org – CC0

Donuts: Game Developer Studio/Opengameart.org – CC-BY 3.0

Eggs: Ogreofwart/Opengameart.org – CC0

Elements: Game Developer Studio/Opengameart.org – CC-BY 3.0

Explosion: Stumpystrust/Opengameart.org – CC0
Extra Character Sprites: Bevouliin.com/Opengameart.org – CC0
Firenall: Cethiel/Opengameart.org – CC0
Firework: Jellyfish/Opengameart.org – CC0
Fuel Collectable: Freestockimages/Opengameart.org – CC0
Fuel Gauge: JM.Atencia/Opengameart.org – CC-BY 3.0
Fish: Sylly/Opengameart.org – CC0
Forest: Ansimuz/Opengameart.org – CC0
Gems: Wenrexa/Opengameart.org – CC0
Grass Sky Backgrounds Images: Segel/Opengameart.org – CC0
Green Horn Zombie Sprites: Bevouliin.com/Opengameart.org – CC0
Green Zombie: Segel/Opengameart.org – CC0
Guns & Bullets Images: Kay Lousberg/Opengameart.org – CC0
Heart/Lives Sprite: Cdgramos/Opengameart.org – CC0
Missile Enemy: Bevouliin.com/Opengameart.org – CC0
Laser: Rawdanitsu/Opengameart.org – CC0
Missile: JM.Atencia/Opengameart.org – CC-BY 3.0
Mountain Background Images: Ansimuz/Opengameart.org – CC0
Mountain Images: Ansimuz/Opengameart.org – CC0
Mouse Icons: Qubodup/Opengameart.org – CC0
Padlocks: Rawpixel.com/Freepik.com
Plane Game Background: Bevouliin.com/Opengameart.org – CC0 && OGA-BY-3.0
Rocks: Bevouliin.com/Opengameart.org – CC0
Rotating Coin Sprites: Puddin/Opengameart.org – CC0
Rpg Character: Curt/Opengameart.org – CC-BY 3.0
Sea Background: Donte/Opengameart.org – CC0
Shinny Coins: Bevouliin.com/Opengameart.org – CC0 && CC-BY 4.0
Space Background: Satur9/Opengameart.org – CC0
Space Parallax Images: Ansimuz/Opengameart.org – CC0
Spaceship: Bevouliin.com/Opengameart.org – CC-BY 3.0 & CC0
Spiky Land Monster: Bevouliin.com/Opengameart.org – CC0
Star: Jellyfish/Opengameart.org – CC0
Tileset: Buch/Opengameart.org – CC0
Top Down Zombie Images: Riley Gombart/Opengameart.org – CC0
Underwater: Game Developer Studio/Opengameart.org – CC-BY 3.0

Walking Green Monster Sprites: Bevouliin.com/Opengameart.org – CC0
Wall: Beaterator/Opengameart.org – CC0

Audio:
Beep: Thisusernameis/Freesound.org – CC0
Bird Sound: JackWilDesign/Freesound.org – CC0
Bell Sound: Se2001/Freesound.org – CC0
Coins (long) dj997/Freesound.org – CC-BY 3.0
Dance Music: Davejf/Freesound.org – CC0
Explosion Sound: Derplayer/Freesound.org – CC0
Firework: Rudmer_rotteveel/Freesound.org – CC0
Gun Sound: Mrrap4food/Freesound.org – CC0
Heavy Beet Music: Jpmusic82/Freesound.org – CC0
Machine Gun Sound: Pgi/Freesound.org – CC0
Melody Music: Drfx/Freesound.org – CC0
Menu Sound: Apenguin73/Freesound.org – CC0
Nuke Sound: Tristanluigi/Freesound.org – CC0
Ouch: Legnalegna55/Freesound.org – CC0
Ping Sound: Wilhellboy/Freesound.org – CC0
Plane Game Level Music: Rezoner/Opengameart.org – CC-BY 3.0
Plane Game Menu Music: Cynicmusic/Opengameart.org – CC0
Plane Game Sound Effects: Subspaceaudio/Opengameart.org – CC0
Random Sound 1: Mosrod/Freesound.org – CC0
Random Sound 2: Littlerainyseasons/Freesound.org – CC0
Random Sound 3: Elfstonepress/Freesound.org – CC0
Random Sound 4: Alanmcki/Freesound.org – CC0
Random Sound 5: Jeremysykes/Freesound.org – CC0
Rocket Sound: Derplayer/Freesound.org – CC0

Introduction

A note from the author:

Congratulations!
You are about to learn the basics of GameMaker and potentiallystart a career in game making.
This book is an introduction to the game-making process, an introduction to GameMaker, and other considerations when making your first game.

GameMaker is a powerful piece of software for making games. This book only covers the basics but is a great place to start. This book uses the updated 2022 edition.
Best of luck with your game-making endeavors.
Ben

Welcome

The purpose of this book is to provide you with some of the knowledge that I have acquired. I make no claim that I'm the best coder or designer, but I do have a proficient understanding that I would like to instill in other budding game makers.

The book is broken down into 29 main chapters and three appendixes: one that pulls together the knowledge you gained from the book to create a game, one offering programming challenges, and one covering how to get set up with GameMaker and its IDE.

The main chapters each introduce a separate programming concept and explain what it is used for and how it is applied, along with some examples. Each of these chapters then finishes with some assignments.

The game chapter details how to make a simple shooting game.

You are then presented with 65 programming challenges, each of which adds a new feature to the game you have just made.

Source Code

All source code in this book can be downloaded from `github.com/apress/gamemaker-fundamentals`.

CHAPTER 1

Variables

When using GameMaker, there are two main types of variables that you will use often. They are strings and numbers (reals and integers). You can do many different things with variables. They can be used for performing calculations and drawing information on the screen. You will use variables for such things as

- Keeping track of score, health, and lives
- Drawing the values of score, health and lives
- Processing data and performing actions based on their values
- Using values in calculations
- Making an object move
- Displaying values, as text or graphically
- Saving the player's progress
- Setting the difficulty of a game or level
- Sending values for online highscores
- Setting initial positions of objects and how they move
- Checking for a weapon's availability or power
- Drawing text to explain what a player must do
- And many more

Note There are several variable types such as instance, local, and global. You will focus mainly on instance and global variables.

B. Tyers, *GameMaker Fundamentals*, https://doi.org/10.1007/978-1-4842-8713-2_1

Some variables automatically have global scope, such as `health`, `lives`, and `score`.

Note It should be noted that these preceding three variables may be deprecated in the future, so you'll need to create your own variables with global scope if you wish them to be accessible game wide, for example, `global.my_score`.

Global variables have scope across the whole game and can be accessed, read, and changed from anywhere else within your game. For example:

```
global.target=100;
```

Generally you would set your global values at the very start of the game, for example, some code that is run on your game splash screen:

```
/// @description Set up global values
lives=6;
score=0;
health=100;
global.target=100;
global.enemies_to_kill=20;
room_goto(room_menu);
```

Note Failure to declare any global value prior to using it will result in an error and crash your game. As such, these are generally set at the start of the game, usually in splash screen room; this is explained later in this book.

If you want to make another variable global in scope, you can put `global` in front of the variable, for example, `global.level`.

Instance variables, for example, `x` and `y`, and `hp`, are generally accessible for the object that set them. (Actually, you can access and change instance variables from other objects, but we won't go into that just now.)

Local variables have the scope of the code block that is using them, for example:

```
var size=5;
```

A basic example usage would be the following:

```
for (var loop = 0; loop< 20; loop++)
{
     draw_text(20+(loop*25,30,loop);
}
```

which would draw the numbers 0 through 19.

An advanced example may look something like this:

```
/// @description Shooting bullet control
if can_shoot_bullet
{
     var bullet=instance_create_layer( x,y,"bullets",obj_player_bullet);
     //spawn the bullet
     bullet.speed=4;//set the speed of bullet
     bullet.direction=image_angle;//match direction to that of the plane
     bullet.image_angle=image_angle;//make the bullet point to direction of
                                    movement
     can_shoot_bullet=false;
     alarm[0]=game_get_speed(gamespeed_fps);
     global.shots_fired++;//update shot count
     audio_play_sound(snd_player_bullet,1,false);
}
```

This would mean that the game would not remember the bullet variable once the code block is completed.

Local variables will mainly be used for data and structures that are no longer required once some data has been processed, and for variables that are outside the scope of the code block. These types of variables help prevent memory leaks.

A memory leak is, for example, when your program keeps creating new items (i.e., variables, or instances) without properly removing them, gradually eating up available memory space – which will crash your game and most likely crash the device that is running it.

The basic code for drawing text is

```
draw_text(x_position,y_position,text);
```

It's good practice to remember to set the font and formatting prior to drawing any text.

Note Text formatting (font, colors, alignment) will be continued for any future drawing of text until you tell the program otherwise, so get into the habit of setting it before drawing anything.

You can create a font, as shown in Figure 1-1.

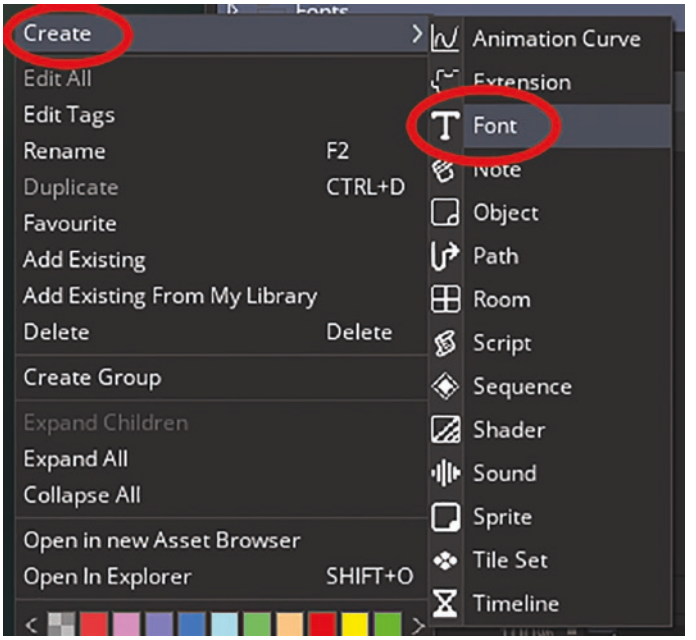

Figure 1-1. *Creating a font asset*

First, let's set up a font to use for this chapter, as shown in Figure 1-2.

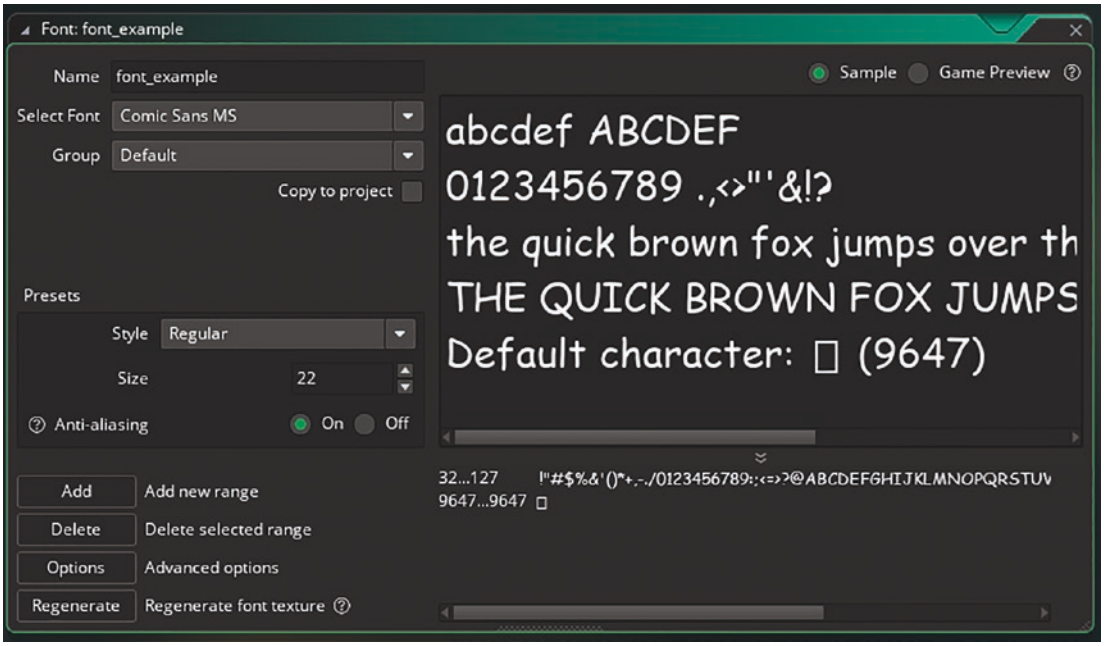

Figure 1-2. *Setting a font to be used for this chapter*

A real working example would be: To draw text "Hello World" at position 100x100. Create an object, **obj_example**, with the following code placed in the **Draw Event**, as shown in Figure 1-3.

```
/// @description Draw example text
draw_set_font(font_example);
draw_text(100,100,"Hello World");
```

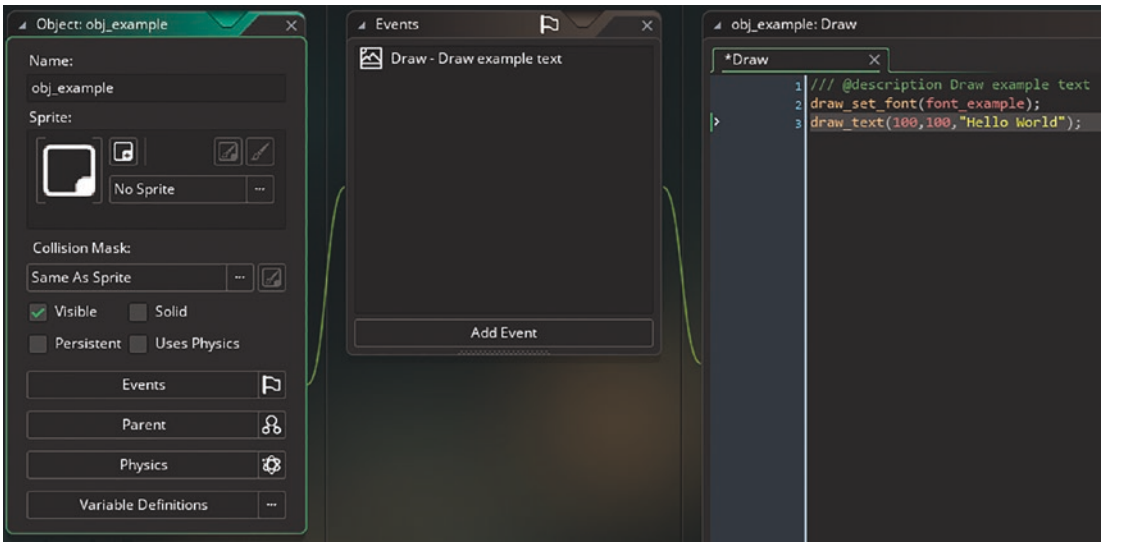

Figure 1-3. *Draw event code*

Drag an instance of this object into the room onto the **Instances** layer and click the run button, as shown in Figure 1-4.

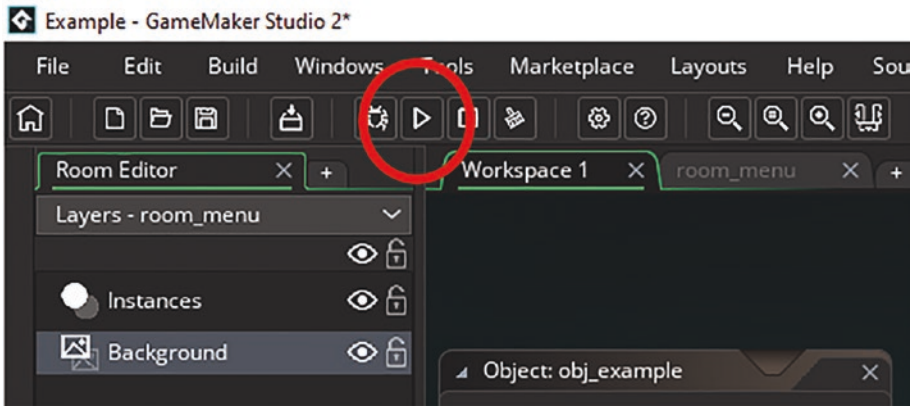

Figure 1-4. *Showing where to click*

You'll then see the text drawn onscreen, like that in Figure 1-5.

Figure 1-5. *Showing some example text drawn*

To draw a variable with a number (an integer), for example, add a **Create Event** and put the following code:

```
/// @description Set up a value
value=20;
```

And in the **Draw Event**, put the following code:

```
/// @description Draw Value
draw_set_font(font_example);
draw_text(200,200,value);
```

Test the game and check the expected value is drawn.

Next, we'll combine some variables, so change the **Create Event** to

```
/// @description Set up values
text="My Name Is Ben";
age=36;
```

Set **Draw Event** to combine these values, noting that you change the value of age to a string so they can be combined without error:

```
/// @description Draw example text
draw_set_font(font_example);
draw_text(100,100,text+" My age is "+string(age));
```

If you are drawing just text or a numerical variable as a separate statement, then the preceding is not required, so, for example, the following would work fine:

```
line="Hello World";
draw_text(30,30,line);
value=4500;
draw_text(30,60,value);
```

When run, you'll see the following on your screen, as shown in Figure 1-6.

Figure 1-6. *Drawing text and variables combined*

Numbers can be an integer, such as 5, or a real which includes decimals, such as `11.8`.

Double-click on **obj_example** in the resource tree. Change the **Create Event** code to

```
my_age=43;
```

You can of course add strings together (called concatenation), for example:

```
first_name="Ben";
last_name="Tyers";
my_name=first_name+" "+last_name;
```

You can do mathematical operations on numbers and then draw the result, for example, the following where if you were to draw the value of `total_cost` on the screen it would show as 45:

```
cakes=9;
cost=5;
total_cost=cakes*cost;
```