



Mostly Codeless Game Development

New School Game Engines

—
Robert Ciesla

Apress®

Mostly Codeless Game Development

New School Game Engines



Robert Ciesla

Apress®

Mostly Codeless Game Development: New School Game Engines

Robert Ciesla
Helsinki, Finland

ISBN-13 (pbk): 978-1-4842-2969-9
DOI 10.1007/978-1-4842-2970-5

ISBN-13 (electronic): 978-1-4842-2970-5

Library of Congress Control Number: 2017948735

Copyright © 2017 by Robert Ciesla

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Cover image designed by Freepik

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Pramila Balan
Development Editor: Matthew Moodie
Technical Reviewer: Nakul Verma
Coordinating Editor: Prachi Mehta
Copy Editor: Kim Wimpsett
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. 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 rights@apress.com, or visit www.apress.com/rights-permissions.

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 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 via the book's product page, located at www.apress.com/9781484229699. For more detailed information, please visit www.apress.com/source-code.

Printed on acid-free paper

*This book is dedicated to Dennis Ritchie (1941-2011),
the creator of the C programming language*

and

my friend Tuomas Mäkelä.

Contents at a Glance

About the Author	xix
About the Technical Reviewer	xxi
Acknowledgments	xxiii
Introduction	xxv
■ Chapter 1: Getting Ready!	1
■ Chapter 2: Game Engine Museum.....	11
■ Chapter 3: A Game Maker’s Lexicon: Level 1.....	21
■ Chapter 4: Commercial Game Engines	41
■ Chapter 5: Freeware Game Engines.....	97
■ Chapter 6: Audiovisual Assets.....	123
■ Chapter 7: Selling Your Game	145
■ Chapter 8: Knowing Your Old-School Games.....	161
■ Chapter 9: Game Developer’s Battle Station.....	181
■ Chapter 10: A Game Maker’s Lexicon: Level 2.....	187
■ Chapter 11: The Mostly Codeless Challenge.....	211
Index.....	213

Contents

About the Author	xix
About the Technical Reviewer	xxi
Acknowledgments	xxiii
Introduction	xxv
■ Chapter 1: Getting Ready!	1
Who Does What in the Video Game Industry	2
Producer	2
Designer	3
Programmer.....	3
Visual Artist.....	4
Sound Designer/Musician.....	4
Tester	4
Writer/Copywriter	5
Common Pitfalls for New Developers.....	5
The Motivational Hole	5
The Ugly Date Syndrome	6
The Wrong Game Engine Issue	6
The Prequel Syndrome	7
The No Testing Needed Approach.....	7
Perfectionism Quest	8
A Programmer's Mind.....	8
In Closing.....	9

■ Chapter 2: Game Engine Museum.....	11
The Quill (1983) by Gilsoft	11
Pinball Construction Set (1983) by Electronic Arts	11
Adventure Construction Set (1984) by Electronic Arts	12
Garry Kitchen's GameMaker (1985) by Activision.....	13
Shoot-'Em-Up Construction Kit (1987) by Sensible Software	14
Arcade Game Construction Kit (1988) by Broderbund Software	15
STOS BASIC (1988)/AMOS BASIC (1990) by Mandarin Software	16
Allegro (1990) by Shawn Hargreaves and the Allegro Developers	17
3D Construction Kit (1991) by Domark	18
RSD Game-Maker (1991) by Recreational Software Designs.....	19
Zillions of Games (1998) by Zillions Development Corp.	20
■ Chapter 3: A Game Maker's Lexicon: Level 1.....	21
General Terminology.....	21
4X	21
AAA	21
Algorithm	22
Application Programming Interface (API).....	22
Bits, Bytes, and Binary.....	22
Chiptune	23
DLC/Microtransactions	23
FPS	24
MMOG/MMORPG.....	24
Particles (i.e., Particle Effects).....	24
Ping.....	24
Pixel.....	25

Polygon.....	25
Pong.....	25
Primitive	26
Resolution.....	26
Shader	26
Shovelware.....	26
Sprite.....	27
Steam	27
Rendering and Prerendering.....	28
Resources.....	28
Sandbox Game.....	28
Skybox	29
Vertical Sync (V-sync).....	29
WASD.....	29
The Fundamentals of Programming.....	29
Programming Language	29
How to Talk to Your Computer.....	30
High-Level vs. Low-Level Languages.....	31
Compiled vs. Interpreted Languages.....	31
Control Flow: Nonstructured vs. Procedural Programming	31
Procedures	32
A Few Lines of Code	32
Variable.....	33
Object-Oriented Programming (OOP).....	33
Classes and Objects (i.e., Classes and Instances)	34
Inheritance.....	34
Methods.....	35
Abstraction and Encapsulation	35

Common Programming Languages: A Primer	36
BASIC.....	36
C	37
C++	37
C#	37
Java	37
JavaScript.....	37
ActionScript.....	38
GML	38
Python.....	38
Lua.....	38
Some Words About Optimization	38
In Closing.....	39
■ Chapter 4: Commercial Game Engines	41
Before You Embark.....	42
Your First Game Engine	43
Your First Game	43
GM Classes and Objects	43
Game Engine Reviews.....	51
GameMaker Studio Professional 1.4 by YoYo Games	51
001 Game Creator (formerly known as 001 Engine) by Mike Weir	58
GameGuru by The Game Creators.....	61
Case Study: Halflight by Soiree Games.....	65
Shoot-'Em-Up Kit by Tall Studios.....	68
Leadwerks Game Engine 4.3 by Leadwerks Software	70
CopperCube 5/CopperCube 5 Pro by Ambiera	73
RPG Maker VX/RPG Maker VX Ace by Kadokawa Games/Enterbrain	76
RPG Maker MV by Kadokawa Games/Enterbrain.....	79
Clickteam Fusion 2.5 by Clickteam.....	81

Game Salad 1.25 (Mac)/Game Salad 1.00 (Windows) by GameSalad Inc.....	85
S2 Engine HD 1.4.6 by Profenix Studio SRLS	88
Tyranobuilder Visual Novel Studio by STRIKEWORKS	91
RTS Creator by Infotread, LLC.....	93
■ Chapter 5: Freeware Game Engines.....	97
Unity 5.5 by Unity Technologies.....	98
A Beginner and His Unity Experience: Developer Interview.....	100
Unity Tips	101
Unity License Options	102
Construct Classic/Construct 2 by Scirra.....	103
Ren'Py 6.99.12.3 by Tom Rothamel and His Team	105
Gameloper by Oyun Döngüsü Ltd	111
Stencyl 3.4 by Stencyl, LLC	113
Godot 2.1 by Juan Linietsky and Ariel Manzur	116
■ Chapter 6: Audiovisual Assets.....	123
The Basics of Digital Audio.....	123
MIDI	124
Lossy Audio Formats (i.e., Delivery Formats).....	124
Nonlossy Audio Formats (i.e., Source Formats).....	125
Plug-in	125
The Fundamental Concepts of Audio Processing	126
Decibels (dB)	126
EQ	127
Common Types of EQ Filters	127
How to EQ Your Material	128
Dynamics (Compression and Limiting)	129
Clipping.....	129
Normalization	129

Reverb	130
Free Audio Resources	130
Some Tools of the Audio Trade	130
Digital Audio Questions	133
Software for Game Visuals	134
Lossy Image Formats (i.e., Delivery Formats).....	134
Nonlossy Image Formats (i.e., Source Formats).....	134
Transparency	135
Tools for 2D.....	135
Tools for 3D.....	140
■ Chapter 7: Selling Your Game	145
Product.....	145
Price	145
Psychological Pricing.....	146
Penetration Pricing	146
Honeymoon Pricing.....	146
Premium Pricing	146
Economy Pricing.....	146
Product Bundling	146
Free-to-Play (Also Pay-to-Win and Freemium).....	147
Microtransactions.....	147
Place (Distribution)	147
Steam (steampowered.com)	147
Amazon Appstore (amazon.com/appstore).....	148
Amazon Digital Game Store (amazon.com/gamedownloads).....	148
Google Play (play.google.com)	148
Apple App Store (apple.com)	149
Apple Mac App Store	149

Good Old Games (gog.com)	150
Itch.io (www.itch.io).....	150
Humble Store (humblestore.com).....	151
IndieGameStand (indiegamestand.com).....	151
Playism (playism-games.com)	151
Promotion	152
Web Site	152
Route A: Custom Domain and Hosting.....	152
Route B: No Budget	153
Screenshots.....	153
Video Trailer	153
Social Media	154
In-Game Advertising (IGA).....	155
Festivals	155
Indiecade (indiecade.com).....	155
Independent Game Festival (igf.com)	155
Assembly Summer (assembly.org)	156
Business and Finance	156
Return on Investment (ROI).....	156
Economies of Scale/Economies of Scope.....	157
Securities.....	157
Securities and Exchange Commission (SEC)	157
Nonaccredited Investor.....	157
Accredited Investor.....	157
Crowdfunding	158
Kickstarter (kickstarter.com)	158
Fig (fig.co).....	159
Indiegogo (indiegogo.com)	159
Gambitious (gambitious.com).....	160

Chapter 8: Knowing Your Old-School Games	161
1977	162
Atari 2600	162
1982	163
Atari 5200	163
Commodore 64	164
1983	165
Nintendo Entertainment System (NES)	165
The Great Video Game Crash of 1983	166
1985	166
Commodore Amiga	166
Atari ST	168
1986	169
Sega Master System.....	169
Atari 7800	170
1987	172
PC Engine (TurboGrafx-16) by NEC Corporation	172
1988	173
Sega Genesis (Megadrive).....	173
1990	174
Super Nintendo.....	174
1994	175
Sony PlayStation (PS1 or PSX).....	175
2000	176
PlayStation 2 by Sony	176
2001	177
Gamecube by Nintendo	177
Xbox by Microsoft	178
The Homebrew Market.....	179

■ Chapter 9: Game Developer’s Battle Station	181
Resources.....	181
CPU.....	181
Hard Drive.....	183
Random Access Memory (RAM)	183
Video Card	184
Your Hardware Needs as an Indie Developer	184
Option 1: Windows 7/8/10 PC	184
Option 2: iMac (Previous Generation), Mac Pro, or Mac Mini.....	185
The Ecological Imperative	186
A Few Words on Displays	186
■ Chapter 10: A Game Maker’s Lexicon: Level 2	187
Digital Units of Measurement.....	187
32-Bit/64-Bit Architecture	187
Hard Drives Revisited	188
Advanced Visual Terminology.....	188
Antialiasing (AA)	189
Billboard	189
Cel Shading	189
Clipping Plane.....	189
Fog.....	189
Viewing Frustum.....	190
Z-buffer.....	191
Shader Languages.....	191
Pixel Shaders (Fragment Shader).....	191
Vertex Shaders	191
Geometry Shaders	192
How to Implement Shaders	192

Texture	192
Texture Atlas	193
Texture Mapping	193
Bump Mapping	194
Normal Mapping	194
Environment Mapping	194
Interpolation	195
Texture Filtering	195
Mipmaps	195
Transform and Lighting (T&L)	196
Raytracing	196
Bloom (Glow)	196
Depth of Field (DOF)	196
Gradient Noise	196
Parallax Scrolling	197
Voxel	197
Delta Time (Δt)	198
Finite State Machine (FSM)	199
UML and FSM, Best Buddies	199
Machinima	199
The Golden Age of Arcade Video Games	200
Grinding	200
Konami Code	200
More on Programming	200
Variables and Operators	201
Data Structures	202
Flow Control	203
Pseudocode and Code Comments	204

More on Physics 207

- Newton’s Laws of Motion 207
- Euler’s Method..... 208
- Ragdoll..... 208
- Rigid-Body Dynamics 208
- Soft-Body Dynamics 208
- Physics Engines Rundown..... 208

■ Chapter 11: The Mostly Codeless Challenge..... 211

Index..... 213

About the Author

Robert Ciesla is a freelance writer from Helsinki, Finland. He earned a bachelor of arts degree in journalism and has a knack for writing urban fiction and directing short films. Robert has worked on many video games on several platforms since being a kid in the mid-1990s. His latest venture is Soiree Games, a burgeoning games company specializing in products with a socially aware slant. Robert's personal web site is at robertciesla.com.

About the Technical Reviewer

Nakul Verma is a professional game developer and currently works as a senior unity developer at Aquimo Sports Pvt Ltd. He has worked in a variety of game genres using multiple technologies. Specifically, he has worked on casual puzzle games, an endless runner, an endless casual game, card games (rummy on Cocos2d-JS and an African game), and a physics simulation sample and is currently working independently on his own game that will be hitting stores soon. He is proficient in game technologies such as Unity, Cocos2d-x/JS, Construct, and Allegro. Gaming has always been one of his favorite hobbies along with sports, music, and break dancing. His favorite game genres are first-person shooters, platformers, and puzzlers. When he is not making or playing games, he is working out, break dancing, or messing around with some gadget.

He earned a bachelor of technology degree from PEC University of Technology in the field of electronics and electrical communication.

Acknowledgments

I'd like to thank the entire Apress editorial team for their support and constructive criticism. Their input greatly helped shape my vision for this book for the better. As Robocop once so eloquently put it, thank you for your cooperation.

Introduction

Why would anyone get into the video game industry? I'll give you two pretty good reasons.

- As of 2017, the global games market was estimated to be worth more than \$100 billion.
- It won't stop at \$100 billion.

From the rectangles on the TV that used to excite people in the 1970s to the painstakingly drawn pixel art of the 1980s and the 1990s to the 3D revolution of the 2000s, video games have been increasingly influencing the aesthetic enterprises around them. Now, in 2017, we have truly reached the Golden Age of the Video Game. The line between cutting edge and retro has never been this blurry. It's hip again to be pixelated. Unlike those early millennial times, a gigantic software team is no longer a must-have prerequisite for success. Thanks to some new tools and digital delivery systems, the one-person operation is back.

Not only do we have the technical know-how to produce nearly movie-quality game experiences, but some very powerful pieces of game making-software are becoming both numerous and widely available. In addition to having top-notch hardware support, this type of software is finally becoming accessible to all. There are tools for every budget and every skill level. And the end results can be more impressive than the audiovisuals in Hollywood. Finally, after 40 years or so, making game creation software is a truly viable option for investors and programmers alike. This is where we, the small-time entrepreneurs, cash in. This is it.

But as great as all of these tools are, this book is not about productivity software per se; it's more about every future game visionary out there. This book is about you, my friend, and together we will take *your* ideas from *your* consciousness to all those little screens in the world. Video games are no longer a mere industry. They are a culture, and you're part of it. Now more than ever.

—Robert Ciesla
CEO, Soiree Games

CHAPTER 1



Getting Ready!

Before you felt the urge to create your own games, you probably were a consumer of games for quite some time. You've played a lot of them over the years, and you have intuitive ideas of what appeals to gamers like you.

Prior to getting serious about game development, you should consciously seek factors that make a game great. The biggest overall successes in entertainment software history all share common traits. To a degree, you are wise to emulate them. Let's take look at these titles, shall we?

- *Tetris for mobile devices by Alexey Pajitnov & EA*: More than 100 million copies sold. It was originally released in 1984 for the Electronika 60, a then-hip computer from the Soviet Union.
- *Wii Sports by Nintendo*: More than 82 million copies sold.
- *Minecraft by Mojang*: More than 70 million copies sold.
- *Grand Theft Auto V (GTA V) by Rockstar Games*: More than 52 million copies sold.
- *Super Mario Bros by Nintendo*: More than 40 million copies sold.

Now let's take a look at the indie contestants.

- *Minecraft by Mojang*: More than 70 million copies sold
- *Super Meat Boy by Team Meat*: More than 40 million copies sold
- *Fez by Polytron Corporation*: More than 40 million copies sold
- *World of Goo by 2D Boy*: More than 40 million copies sold
- *Bastion by Supergiant Games*: More than 3 million copies sold

What do all of these titles have in common? They're multiplatform. In some cases, they're very, very multiplatform. One of them, *Wii Sports*, came bundled with a cool new system—see if you can get on that bandwagon! These ten games are intuitive to grasp and control, offering a smooth gaming experience. There are also memorable main characters with highly merchandisable gimmicks—Italian plumbers, anyone? And there's a lot of

violence, in the case of *GTA V* (and many hit games that didn't quite make the list), which is a gangbanger simulation of the highest caliber. For some reason, the people of this planet really, really enjoy their extreme violence.

So, the following features are what sells:

- *Flawless game mechanics*: The player doesn't have to struggle with controls or grasping the idea of the franchise. The basics are simple, and they work at all times.
- *Lasting challenge*: The game is virtually unbeatable (like *Tetris* and *Wii Sports*) and/or provides tons of replayability.
- *Deployment for multiple platforms*: Keep those Windows-only games to a minimum.
- *Memorable, merchandisable characters*.
- *Conflict and violence*. In general, this planet loves it. In games, it certainly helps.

Now, as a small developer, it's likely you will need to wear many hats. Let's take a look at these roles in the video game industry next. Working with modern video game-making software, it's unlikely you will need to ever dwell very deep into more complex areas of programming, but it's still a good idea to get acquainted with some common industry job titles.

Who Does What in the Video Game Industry

Many of the following development team roles are increasingly becoming specialized as video games rival blockbuster movies in their armies of creative people working on them.

Producer

Responsible for keeping the whole project together, a video game producer benefits from both hands-on experience in as many related fields as possible and a sense of overall vision. The number of a producer's creative responsibilities varies within development teams. In some cases, a producer solely works as management, solving conflicts and keeping a team going.

This may be an unnecessary post in smaller projects, however. A tiny operation obviously doesn't benefit much from a hired producer.

Some typical producer duties include the following:

- Building and maintaining a functional team
- Contracting out work and delegating responsibilities
- Media relations

Designer

Video game designers come up with the conceptual part of a product. Good game design is timeless. Think of chess: it was designed in the sixth century and is still going strong. Creating balanced game dynamics and a low enough learning curve are the designer's job. Also, as sprawling 3D games are all the rage, level designers are very much in demand. Modern level designers usually work with dedicated software, sometimes provided by the programmers in the team.

Programmer

Programmers handle perhaps the most diverse bunch of duties within game production. There are numerous specialized programming fields needed in creating a competitive product. In the early days of the 1980s, being a programmer meant you were the sole person behind a title. Not so much in modern times, although there are exceptions (*Minecraft*, for one, is a one-person operation). Being a programmer can mean these things and much more.

- *Game core creator*: This is what people usually mean by programmer. The game core creator is responsible for game mechanics, main visuals, and player controls.
- *Artificial intelligence developer*: This person is responsible for making smart enemies within a game.
- *Problem solver*: If you're really good, you may be hired as a mercenary programmer to solve a development team's issues within a project.
- *Physics expert*: This person is responsible for creating realistic maps/levels for games with a set of artificial laws of physics governing the game world.
- *Networking specialist*: Many games are run online these days as multiplayer war zones. This creates a whole host of challenges to a project.

Usually, being a good programmer requires a strong sense of logic and/or mathematics. The importance of math is somewhat exaggerated in most programming literature, but it always helps. Some fields of programming work, such as physics, simply do require strong math skills.

Many programmers have a "pet language," which is one they are most comfortable with. Make sure yours is one of the more useful ones, such as C++ or Java.

Visual Artist

Video games used to mostly feature simple on-screen shapes for visuals. Since the advent of 3D graphics in the mid-to-late 1980s, visual artists span an increasingly large group of subfields. These include the following:

- *2D artist*: This includes duties such as presentation and, in the case of 2D games, in-game visuals.
- *3D artist/3D modeler*: These artists create 3D objects with software such as Blender or Maya.
- *3D animator*: This may in many cases be the job of the modeler also. An animator works with the 3D objects created by the modeler and crafts fetching animated sequences, such as a 3D human walking, running, or fighting.
- *Texture artist*: In essence, 3D objects need a coat of digital paint on them to make them look less bland and more realistic, which is the duty of the texture artist.
- *Environmental artist*: Most 3D games need compelling vistas to make them draw the players in.
- *Conceptual artist*: Especially bigger projects benefit from unified art direction. Concept drawings on whatever media help with this goal.

Sound Designer/Musician

If you are an experienced musician, you can in theory make it as a video game composer. Earlier on, as in the 1980s, computer musicians were required to create their compositions pretty much using programming skills. As of late, as long as the output is in digital form, your audio work can be quickly incorporated into a video game project.

Sound designers may or may not also be musicians. What they need is the ability to create audio usable in a video game context, meaning mostly sound effects and atmospheres.

Tester

Testing is a very important phase in a video game's life. If you are a one-person developer, you should put considerable effort into testing your products thoroughly before release.

There are roughly two stages of game testing: alpha (in-house) and beta. Beta testing refers to the public at large volunteering to spot issues in your game. Beta testing can be either by invitation only or in public.