



Community Experience Distilled

Mastering CryENGINE

Use CryENGINE at a professional level and master the engine's advanced features to build AAA quality games

Sascha Gundlach

Michelle K. Martin

[PACKT]
PUBLISHING

Mastering CryENGINE

Use CryENGINE at a professional level and master the engine's advanced features to build AAA quality games

Sascha Gundlach

Michelle K. Martin



BIRMINGHAM - MUMBAI

Mastering CryENGINE

Copyright © 2014 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: April 2014

Production Reference: 1040414

Published by Packt Publishing Ltd.

Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78355-025-8

www.packtpub.com

Cover Image by Berker Siino (berkersiino@gmail.com)

Credits

Authors

Sascha Gundlach

Michelle K. Martin

Reviewers

Hendrik Polczynski

Ross Rothenstine

Sheetanshu

Acquisition Editor

Owen Roberts

Content Development Editor

Neeshma Ramakrishnan

Technical Editors

Pragnesh Bilimoria

Pooja Nair

Nikhil Potdukhe

Copy Editors

Alisha Aranha

Roshni Banerjee

Gladson Monteiro

Adithi Shetty

Project Coordinator

Priyanka Goel

Proofreaders

Simran Bhogal

Maria Gould

Ameesha Green

Paul Hindle

Indexers

Mariammal Chettiyar

Monica Ajmera Mehta

Graphics

Ronak Dhruv

Disha Haria

Yuvraj Mannari

Abhinash Sahu

Production Coordinator

Adonia Jones

Cover Work

Adonia Jones

Shantanu Zagade

About the Authors

Sascha Gundlach has been working in the games industry for over a decade and started his career as a script programmer in a small game studio in the early 2000s. He worked for Crytek for eight years, working on games such as *Crysis*, *Crysis: Warhead*, and *Crysis 2*.

He is a CryENGINE expert and has provided countless training sessions and individual training to CryENGINE licensees in the past years.

In 2013, he founded his own game development company, MetalPop Games, together with his partner and Crytek veteran Michelle K. Martin in Orlando, Florida.

He spends his days working on video game projects and provides consulting work for other game projects.

Michelle K. Martin is a software engineer in the game industry, specializing in animation systems. She started her career with the German developer, Crytek, working on projects such as *Crysis* and *Crysis 2*. During her career, Michelle has helped develop and improve CryENGINE's animation system with several features. Being an expert in CryENGINE, she has provided a lot of support and training to CryENGINE licensees over the years, helping their team to get the most out of the engine.

In 2013, she founded MetalPop Games together with her partner and Crytek veteran Sascha Gundlach. It is an indie game development studio and they are currently working on their first title.

When she's not in front of the computer programming, she is most likely to be in front of the computer playing games.

More about Sascha and Michelle's company MetalPop Games can be found at www.metalpopgames.com.

About the Reviewers

Hendrik Polczynski is a software developer from Germany. He has been working on software development for over 10 years. He likes to take on a variety of fields, from the automation industry to web, UI, and game development. You can find his open source projects on github.com/hendrikp or on his YouTube channel. Hendrik is currently maintaining a handful of open source projects around the CryDev community using CryENGINE 3 FreeSDK. When he is not working, he is working on his Bachelor thesis or helping out in the development of *Miscreated* by Entrada Interactive, which is a post-apocalyptic, survival-based MMORPG; it is unlike anything you've played before.

I would like to thank the following people who have helped me review specific chapters of this book:

Victor Duarte, Simon Hambly, and Chris Ioakeimoglou

Ross Rothenstine has been interested in game development from the instant he sat in front of a computer. Studying all engines, from self-made to commercial, he loves to find ways to tinker with these massive systems and push them to their core, thereafter presenting his findings to universities and teaching courses wherever he may. Game development may be an intimidating task, but with books like these, he's sure you can do it!

Sheetanshu is a professional developer who resides in the metro city of Gurgaon, India. He is currently working to obtain an Engineering degree at the Guru Gobind Singh Indraprastha University. He fell in love with programming during his childhood and since then there was no turning back. From the beginning of his bachelor's degree in engineering, he has been an active developer. He had already contributed a lot to the web community when he further got involved in game development at his brother's request. He has over a year's worth of experience working with game engines such as Unity 3D, CryENGINE 3.5, and UDK. Presently, as the final phase of his Engineering degree, he is working on his industrial internship with 4play as the Chief Game Officer and is also working as a research assistant with Dr. Aynur Unal from Stanford, Palo Alto.

www.PacktPub.com

Support files, eBooks, discount offers, and more

You might want to visit www.PacktPub.com for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface	1
Chapter 1: Setting Up the Perfect Pipeline	7
What is a production pipeline?	7
Importance of a strong pipeline	8
Version control for CryENGINE projects	8
What version control does for you	9
Production without version control	9
Working from a shared folder	9
Selecting a VCS for CryENGINE projects	10
Setting up version control for CryENGINE	11
Sandbox	11
Identifying CryENGINE project files to be excluded from version control	16
Automated builds and build scripts	16
Creating nightly builds	17
Setting up a build server	17
Operating systems	18
What the build scripts should do	18
Creating your custom build script	19
Writing your own script	20
Wrapping it up	29
Scheduling automated builds	30
Automated performance tests	34
Using level statistics to profile the performance of a level	35
Build integration	36
Integrating a new version of CryENGINE	36
The engine depot	37
Project branch	37
Integration	37

Quality assurance processes	38
QA pipeline in larger scale teams	39
QA pipeline in smaller teams	40
Working without a QA pipeline	40
Understanding issue tracking in CryENGINE	40
Summary	41
Chapter 2: Using the CryENGINE Input System – Keyboard, Mouse, and Game Controller	43
The CryENGINE input system	43
A layer of abstraction	43
The input	44
Game actions	45
Action Maps	46
Multiple Action Maps	47
Creating a new Action	49
Setting up an Action event	50
Adding an Action mapping	50
Reacting to Action events	57
Action events in code	57
Action events in FlowGraph	62
Filtering Actions	63
Creating Action Filters	63
Using Action Filters	64
Reacting to Input events	65
Code	65
FlowGraph	67
User profiles	68
Modifying user profiles	68
DLCs and patches	70
The input event names reference	70
Keyboard	70
Mouse	72
Xbox 360 controller	72
PS3 controller	73
Summary	74
Chapter 3: Building Complex Flow Graph Logic	75
Who uses the flow graph system?	75
A more complex application of the flow graph	76
Revisiting the basics of flow graphs	76
Types of nodes	76
Entity nodes	76
Component nodes	77

Flow graph data format and storage	78
The entity nodes with dynamic targets	82
What happens if we input the wrong EntityId?	83
A more complex application of dynamic EntityIds	84
Let's take a shortcut	84
Q – node quick search	85
F/G – link highlighting	85
Ctrl + Shift + V – paste nodes with links	86
Embedding the flow graphs	86
GameTokens	89
The GameToken libraries	89
Reaction to a game token changing its state	90
The GraphTokens variable	91
Accessing the Lua functionality	91
Creating nodes in C++ and Lua	92
Adding ports to an entity flow graph node	92
Creating flow graph nodes using the Lua script	93
Creating flow graph nodes using C++	94
Summary	94
Chapter 4: Morphs and Bones – Creating a Facial Setup for Your Character	95
Creating a facial setup for a character	96
Exporting the character's head	96
Using facial expression libraries	98
Creating a new facial expression library	99
Mapping the library	103
Creating expressions	104
Facial animation	106
Creating facial sequences	107
Using facial sequences in the engine	108
Inside the TrackView editor	109
Inside the FlowGraph node	110
Using expressions	111
The Lip Sync feature	112
Manual lip synching	112
Automatic phoneme extraction	112
Lip sync playback	113
Quality – phoneme extraction	115
Quality – visemes and phonemes	115
Quality – adding emotions	116
Summary	116

Chapter 5: Mastering Sandbox	117
Don't stop getting better	117
Getting faster with keyboard shortcuts	118
Thinking about hand placement	119
Object editing modes	120
Test it!	120
Aligning objects	121
Using the deep selection feature	122
Using the Goto Selection feature	124
Using camera tag-points	124
Top five shortcuts	125
Customizing Sandbox	126
Customizing the Sandbox window layout	127
Saving and loading layouts	127
Working with cameras	127
Camera targets	128
Switching cameras	129
Exploring Sandbox custom commands and macros	129
Sandbox custom commands and macros	130
Looking at some lesser-known features	131
Video recording	131
Mesh editing	132
Managing PAK files	133
Renaming multiple objects	133
Summary	134
Chapter 6: Utilizing Lua Script in CryENGINE	135
Understanding the relevance of the Lua script in CryENGINE	136
Lua-based entities	137
Creating a new Lua-based entity	137
Assigning a 3D object to an entity	140
Using entity slots	141
Setting up physics	143
Making an entity multiplayer-ready	144
Understanding the dataflow of Lua entities in a multiplayer environment	145
The Client/Server functions	147
The Remote Method Invocation definitions	148
Using the state machine	151
Using script binds	152
Calling script binds	153
Creating new script binds	153

Using engine callbacks	154
Using the Lua debugger	155
Summary	157
Chapter 7: Animating Characters	159
<hr/>	
The CryENGINE animation system	159
Introducing CryMannequin	160
Splitting up the game and animation logic	161
Understanding CryMannequin's purpose	162
Selecting animations	162
Starting animations	164
Fragments, Fragment IDs, and Tags	165
Extending the state machine	167
Understanding the state machine hierarchy	168
Creating a new state	170
Triggering the new state	172
Playing animations without CryMannequin	174
TrackView	174
Multiple animation layers	174
CryMannequin tracks	177
Triggering animation from FlowGraph	177
The PlayAnimation node	178
Other animation nodes	179
The code	180
Summary	181
Chapter 8: Mastering the Smart Objects System	183
<hr/>	
What are SmartObjects?	183
Where the Smart Objects system is used	184
Smart Objects categories	185
Environmental SmartObject	185
A time-based SmartObject	185
Navigational SmartObject	185
The concept of the SmartObject system	185
The SmartObjects editor	186
The Window layout of the SmartObject editor	187
The Rules and Tasks windows	188
The Rules List window	188
The Rule Properties window	188
The SmartObject library	189
Creating a new SmartObject rule	189
Preparing the level	190
Creating the SmartObject rule	191

Creating the SmartObject classes	192
Creating the SmartObject states	194
Creating a SmartObject state pattern	195
User and Object	195
The state pattern	196
Creating an AIAction	198
Selecting actions	198
Creating the action	199
Setting up the action and state changes	200
Getting the level ready	201
Testing the SmartObject rule	202
Troubleshooting	203
Debugging SmartObjects	203
Debugging AIActions	204
Changing states from Lua	204
Summary	205
Chapter 9: Eye Candy – Particles, Lens Flares, and More	207
Types of eye candy	207
Particle effects	208
Working with particle effects in CryENGINE	208
The particle editor	208
Creating a new particle effect	209
Customizing the particle parameters	211
Tweaking the effect	212
Particle effects at runtime	213
Lens flares	214
The lens flare editor	214
Creating a new lens flare effect	214
Assigning a lens flare effect to a light	216
Lens flare effects caused by the sun	217
Postprocessing effects and the flow graph	217
Using material FX graphs	218
Creating a custom material FX graph	219
Testing the new material effect	222
Debugging material effects	222
Postprocessing in TrackView	222
Using effect tracks in a TrackView sequence	223
Using track events	224
Performance considerations	225
Overdraw	226
Draw calls	226
Summary	226

Chapter 10: Shipping the Build	227
Getting your game ready to ship	227
Optimizing performance	228
Optimizing levels	228
Optimizing shadows	229
Vegetation	231
Layers	231
Testing and QA	231
Errors and warnings	233
Log verbosity	233
Tackling legal issues	234
Copyright	234
Credits	235
CryENGINE license	236
Things to consider for indies	236
MobyGames	237
Preparing your build	237
Building a release candidate	238
Auto-loading the first level	238
PAK files	239
Removing all debug features	240
Reducing your build size	241
Shaders	242
Creating an installer	244
A ZIP file	244
Selecting an installer	245
Dependencies	246
An icon for your executable	246
Summary	247
Index	249

Preface

Today, making games is easier than ever before. There are a plethora of game engines available for developers, and most of them can even be tried out free of charge or used to release games noncommercially. So, irrespective of whether you are modifying an already released game, building your own indie game, or maybe working on a big AAA production, the chances that you will be using a licensed 3D engine such as the popular CryENGINE are pretty big.

The times where development teams would write their custom game engine to produce a game are mostly over. The use of licensed 3D engines is very common and saves developers and publishers a lot of money. Using a licensed 3D engine instead of building a custom solution allows developers to focus on making a great game instead of developing and maintaining their own technology.

A result of this continually advancing technology development, however, is that it has become very difficult for developers to really master all aspects of a 3D engine. Engines such as CryENGINE are not simply rendering programs that are capable of drawing beautiful content on the screen in real time. Animation systems, physics simulation, AI behaviors, or particle systems are just a few parts of what makes up the CryENGINE. However, with the increasing complexity of game engines, it has become more difficult for today's game developers to stay on top of the technology.

This is where *Mastering CryENGINE* comes in. This book focuses on the professional CryENGINE developer and tries to provide an inside scoop on how to produce games at an AAA production level. Getting the most out of the engine and becoming a highly productive CryENGINE developer requires knowledge of the multitude of subsystems that CryENGINE offers.

The goal of this book is to provide you with valuable information about the most important aspects of CryENGINE production as well as guide you through the most common technical problems encountered when developing game content with the engine.

What this book covers

This book covers a wide range of topics that are closely related to making games with CryENGINE at a professional level. Basic elements such as setting up the engine, building simple environments, or other topics that might be of interest for beginners might be touched upon, but they will not be covered in too much depth. Instead, this book focuses on arming you with in-depth knowledge of the core systems of CryENGINE that are necessary to build high-quality content.

Chapter 1, Setting Up the Perfect Pipeline, focuses on one of the most important aspects of game production: a stable and flexible pipeline. This chapter covers the tailoring of the perfect pipeline for your project as well as the important aspects of setting up a new pipeline.

Chapter 2, Using the CryENGINE Input System – Keyboard, Mouse, and Game Controller, provides an overview of the CryENGINE input systems. You will learn how to create new action maps and handle user profiles as well as how to react to input events in code and flow graphs.

Chapter 3, Building Complex Flow Graph Logic, focuses on the more advanced features of the flow graph system. Nested flow graphs as well as graph tokens will be explained in detail and will be used to build a practical game example.

Chapter 4, Morphs and Bones – Creating a Facial Setup for Your Character, covers all the steps necessary to create a complete facial setup for a character. You will learn about facial libraries as well as how to get a character ready for lip syncing and procedural blinking.

Chapter 5, Mastering Sandbox, focuses on increasing your production speed, efficiency, and productivity when working with CryENGINE. Hidden features, important shortcuts, and relevant engine settings will be discussed in this chapter.

Chapter 6, Utilizing Lua Script in CryENGINE, teaches you how to use the Lua scripting language to build more sophisticated gameplay elements. The creation of new script binds and modification of the existing entities will be covered here.

Chapter 7, Animating Characters, will explain the principles of CryMannequin, the high-level animation system of CryENGINE. You will also learn how to extend the state machine to trigger your own mannequin animation sequences. The chapter will also cover other methods of triggering animations.

Chapter 8, Mastering the Smart Objects System, will provide an insight on how to use the SmartObject system. This system will be used to build a gameplay example of a security guard AI behavior. Furthermore, navigational SmartObject systems will be used to set up AI characters that can climb over walls.

Chapter 9, Eye Candy – Particles, Lens Flares, and More, focuses on adding some eye candy to your game. The setup and usage of particle effects as well as the brand new Lens Flare editor will be covered.

Chapter 10, Shipping the Build, focuses on getting your game ready to ship. It will cover how to prepare the build for release, remove unwanted source files, and reduce the overall build size.

What you need for this book

In order to make best use of the examples in this book, you should use the latest version of CryENGINE. Although much of the knowledge provided in this book can still be applied to older versions of the engine, it is recommended that you use this book with CryENGINE 3.5.2 or above.

The jump to CryENGINE 3.5

The CryENGINE technology has been around for over 10 years and the engine has undergone a lot of changes and improvements over those years.

Crytek released the latest version of the engine, CryENGINE 3, in 2009, which introduced a lot of improvements (for example, a deferred rendering pipeline) and brought the engine to XBOX 360 and PlayStation 3. Along with the addition of countless new rendering features, the Sandbox editor also underwent a big facelift.

Within the lifespan of CryENGINE 3, there has been one big transition: the jump from Version 3.4.5 to Version 3.5.

With the upgrade to Version 3.5, all the changes and improvements made during the development of the critically acclaimed games, *Crysis 3* and *Ryse*, found their way into the CryENGINE 3 SDK.

All the new features that made *Crysis 3* and *Ryse* look so stunning became available to developers with this upgrade. An improved rendering pipeline and the new animation system CryMannequin, which replaced AnimationGraph, are two of the biggest changes done to the engine in Version 3.5.

Most of the topics covered in this book will still be valuable for you if you are working with an older version of the engine. However, some of the newer features discussed in this book, for example the LensFlare editor, might not be available for you if you are working with an older version of the engine.

Other required software

In order to follow the examples in this book, we recommend that you obtain the following software:

- CryENGINE SDK 3.5.2 or above
- Photoshop Version 4 or above
- Notepad++
- Visual Studio 2010
- 3D Studio Max 2010

Who this book is for

This book is aimed at an experienced CryENGINE developer. Although it is certainly possible to use this book as a beginner who is unfamiliar with the CryENGINE technology, it will be much more efficient when a certain level of experience with the engine is there.

Whether you are a CryENGINE enthusiast looking to turn your hobby into a full-time profession or you've just started working with CryENGINE on a professional project, this book will provide you with valuable information and deep insights into the engine. This is invaluable to produce content at a professional level.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "The OnReset () function will be called every time the entity script is reloaded."


A block of code is set as follows:


```
if (slot == 0) then
    self:DrawSlot(0, 1);
    self:DrawSlot(1, 0);
else
    self:DrawSlot(0, 0);
    self:DrawSlot(1, 1);
end
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
if (slot == 0) then
    self:DrawSlot(0, 1);
    self:DrawSlot(1, 0);
else
    self:DrawSlot(0, 0);
    self:DrawSlot(1, 1);
end
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "Clicking on **Show Log File** will open the respective logfile for you automatically."

 Warnings or important notes appear in a box like this.

 Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book – what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to feedback@packtpub.com, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.packtpub.com>. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books – maybe a mistake in the text or the code – we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/submit-errata>, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at questions@packtpub.com if you are having a problem with any aspect of the book, and we will do our best to address it.

1

Setting Up the Perfect Pipeline

Before the actual work on any new project can begin, you as a developer have to think about your production pipeline. Time spent on designing a robust pipeline is always time well invested. The larger the project ahead of you, the more important it is to set up a stable pipeline. In this chapter, we will discuss the following topics:

- Production pipeline setup for CryENGINE projects
- Using version control in CryENGINE projects
- Setting up automated builds and build scripts
- Integration of CryENGINE builds and versions

The goal of this chapter is to provide you with information and best practices on building a stable and flexible CryENGINE production pipeline.

What is a production pipeline?

In simple words, a CryENGINE production pipeline could be described as a series of operations you are performing with the engine in order to create your product. Things like exporting a 3D asset or compiling C++ code, for example, are parts of a typical CryENGINE pipeline. Our production pipeline also defines how and to what standards you perform all project-related tasks.

When working with CryENGINE, those project-related tasks can include:

- Exporting a 3D asset
- Compiling code
- Creating automatic builds

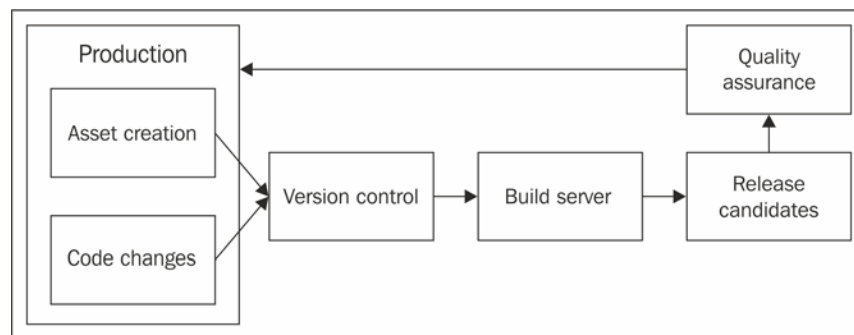
- Processing bug reports
- Checking files into your version control system

A pipeline is basically a number of rules and guidelines you set for yourself and your team to work on your project. If those rules make sense and fit your project, your life will become a lot easier. If they do not fit your project or if they do not exist yet, you will have a higher chance of running into all kinds of problems.

Importance of a strong pipeline

No matter what CryENGINE project you are about to start, you will have weeks or possibly months of work ahead of you. Being as prepared as you can for this should be your goal. Planning and preparing your pipeline will help you save time and work more efficiently during the lifespan of your whole project.

A well thought out pipeline, which is standardized and enforced within your team, will increase production speed significantly. In this chapter, we will discuss the most important aspects of a CryENGINE production pipeline.



Overview of the production pipeline

Version control for CryENGINE projects

The decision of which **version control system** (VCS) to use is one of the most important pipeline decisions to make when you are planning your CryENGINE project.

It is also one of the first things that should be discussed, since a lot of other aspects of your production will depend on it.

What version control does for you

Version control is incredibly useful in any area of game production. What a VCS basically does is keep track of changes made to your files and allows you to review those changes and even helps you revert to the older versions of your files.

This means if you make a mistake or delete or lose a file, it is generally very easy to recover whatever you have lost. In addition to this, using a VCS makes it a lot easier to collaborate since you will always have an overview of what changes your team members made to the project files.

Production without version control

Even today, where VCSs such as SVN, Perforce, or Git have become very affordable, or even free, there are still teams out there working without the safety net of a version control system.



Not using any version control whatsoever is always a bad decision for larger projects.

Working from a shared folder

One method often used by less experienced mod teams is to work out of a shared folder, which is accessible for everybody from the network. While it might seem simple and easy to work this way with everybody just copying their files into the shared folder, there are a lot of things which can go wrong, which are as follows:

- Files can get overwritten accidentally
- Code and script conflicts cannot be caught and resolved easily
- Tracing back older changes becomes extremely difficult

With low cost version control systems being widely available today, there is no reason even for small teams to work this way. Setting up and maintaining a version control solution will of course consume a certain amount of time, but it is always time well invested.

Let's have a look at a real-life example. You are working on a CryENGINE game project and you discover a game breaking bug. Let's say someone on your team submitted something which broke the game. Now it is up to you to identify and fix the issue. Having no access to either the file history or changes done to the individual files will make it very difficult for you to solve the issue. However, in a project environment with a VCS setup, you could simply step backwards through the submitted changes to identify the file which was responsible for breaking your game.

Selecting a VCS for CryENGINE projects

When it comes to deciding which VCS to use for your CryENGINE project, your decision will be determined by your budget, the scale of your project, and possibly your personal preference.

You will have to choose between a centralized and distributed VCS. While a centralized VCS keeps all files on a central server, a distributed VCS mirrors the whole repository on each client. Both systems come with different upsides and downsides, but for CryENGINE, it makes no difference which type of system is used.

There are many VCSs available today, and they come in many flavors. Most commonly used VCSs for CryENGINE projects are as follows:

- **Perforce:** This is sometimes also called P4 and is a professional centralized VCS, which mostly is the tool of choice for professional and larger size game teams. Perforce licenses are generally not free, but there are various license options which allow indie and mod teams to make use of the software without spending much money. CryENGINE has native support for Perforce and allows you to check in/out files directly from Sandbox.
- **SVN:** This is also called Subversion and is a free, open source centralized VCS. It is widely used by smaller teams without a big budget, since it can be used without any cost.
- **Git:** This is also a free to use open source VCS. It differs from Perforce and SVN by using a distributed architecture. In direct comparison to Perforce and SVN, Git can be quite difficult to use, especially for developers with a nontechnical background.

Setting up version control for CryENGINE

Once you have made your decision regarding which version control system to use for your project, it is time to set up your CryENGINE environment. Depending on your role in your game's production, certain aspects of this setup might be more or less interesting to you. For example, if you are a programmer, you might be less interested to learn about setting up your Photoshop or 3ds Max and skip ahead to the relevant coding topics.

Sandbox

Being able to check out levels, layers, or materials files directly from Sandbox without switching to your version control client is very comfortable and will speed up your workflow considerably.

Support for Perforce version control is integrated into the Sandbox editor. Sandbox will automatically check out the corresponding files when they are being modified.

When using SVN, Git, or any other system, files cannot be directly checked in/out from Sandbox. In this case, no further setup is necessary.

Perforce setup

The first step to setting up Sandbox to work with Perforce is to enable version control. This is done in the Sandbox preferences as follows:

1. Open the **Preferences** window from the **Tools** menu.

