

11 Shipping and Releasing

Contents

[Overview 3](#_Toc508201775)

[11.1. Core Concepts 4](#_Toc508201776)

[Deployment Process 4](#_Toc508201777)

[Benefits of Cooking 4](#_Toc508201778)

[Packaging Configurations 5](#_Toc508201779)

[Distribution and Deployment 5](#_Toc508201780)

[Patching 6](#_Toc508201781)

[11.2. Common Mistakes 7](#_Toc508201782)

[“When It’s Shipped, We Can Start the New Project” 7](#_Toc508201783)

[Sending the Right Build 7](#_Toc508201784)

[When It’s Done for One Console, It’s Not Done for All 7](#_Toc508201785)

[Exercises 8](#_Toc508201786)

[Exercise 11A: Packaging a Game 8](#_Toc508201787)

[Exercise 11B: Deploying to a Mobile Device 9](#_Toc508201788)

# Overview

**PURPOSE**

The purpose of this tutorial is to provide a basic understanding of the shipping and deployment phases of project development. The content delves into the process of cooking and deploying a project to various platforms.

**SCOPE**

This section will provide an introductory overview of the following:

* Building
* Project settings
* Packaging and cooking
* Platform development

**PREREQUISITES**

It is assumed that you have worked through previous tutorials such that you have a sound understanding of how to use the Unreal Editor.

## 11.1. Core Concepts

### Deployment Process

When a project is ready for distribution, it must go through several steps before it can be given to testers, customers, or potential publishers. Following are some of the steps that will be taken during the packaging process:

1. When a project has a custom source code, this code will first be compiled.
2. All required content will be converted, or **cooked**, into a format that can be used by the target platform.
3. Compiled code and cooked content will be bundled into a distributable set of files, such as an installer for Windows. This is called **packaging**.

### Benefits of Cooking

As a game is developed, it accrues a large amount of ancillary data that will never need to be in the shipped product. This data is often stored in file formats that are more conducive to development than to the final release. A process known as “cooking” is used to organize the build into a state that’s ready for distribution.

The cooking process generally includes the following operations:

* Unreferenced content is removed from the game.
* Textures are analyzed, according to their groups, and mipmaps above the maximum size are removed.
* Content is converted into platform-specific formats.
* Where possible, content is compressed into formats that are memory efficient and/or more performant.
* Content related to the editor is identified and removed.
* Files are consolidated into a single .pak file (this operation is optional).

The result is a distribution build that is smaller in size and less taxing on runtime resources such as the CPU, GPU, and RAM.

### Packaging Configurations

#### Debug

The Debug configuration contains symbols for debugging. This configuration builds both the engine and game code in debug configuration. If you compile your project using the Debug configuration and want to open the project with the Unreal Editor, you must use the “-debug” flag to see the code changes reflected in your project.

#### DebugGame

The DebugGame configuration builds the engine as optimized but leaves the game code in a debug state. This configuration is ideal for debugging only game modules.

#### Development

The Development configuration is equivalent to Release. Unreal Editor uses the Development configuration by default. Compiling your project using the Development configuration enables you to see code changes made to your project reflected in the editor.

#### Shipping

The Shipping configuration is used for optimal performance and shipping your game. This configuration strips out console commands, stats, and profiling tools.

#### Test

The Test configuration is the same as the Shipping configuration with the addition of console commands, stats, and profiling tools.

### Distribution and Deployment

#### PC (Non-Steam)

Packaging and deploying a project for PC is the most straightforward form of deployment. It simply produces a folder with your final game in it. You can zip this folder to share with friends or upload the game to a website like itch.io for people to download.

#### Steam

To publish your game to Steam, you must first become a Steam Partner. Teams can sign up for partner status through **Steam Direct**. This service does require a fee, and there are content guidelines that define what you can and can’t publish to the platform.

To prepare your game for deployment to Steam, you simply need to package it like you would for a non-Steam release. After that, Steam’s content packaging requires you to write specific scripts to determine what content from your game needs to be uploaded into a “depot.” Every time you make a new release, you must run this script to upload your content to Steam. These tools come as part of the **Steamworks SDK**, which you can download once you become a Steam Partner.

#### Android

Packaging for Android is a slightly more involved process than for PC, as different devices require some assets to be in a specific format. This requirement is due to the differences in the graphics chips and how they render various textures.

To package and deploy for Android, you must ensure that Codeworks for Android is installed. Codeworks is bundled as part of Unreal Engine 4 to help make the process a little easier.

The packaging produces several different kinds of output files, including the following:

* An **APK file** containing your binary code (that is, the engine itself, any C++ code you have written, and additional supporting libraries for the platform)
* An **OBB** file containing all the actual art assets for your game

Google Play has file size limits on APK files, which is why OBB files of content are provided separately. Publishing to Google Play requires you to sign up for a Google Play account for a one-off fee.

#### iOS

To develop for iOS, you need an Apple Developer account. Once you are signed up as an Apple Developer (requires an annual fee), you must go through the process of generating certificates and provisioning profiles before your game can be deployed to a device.

In addition, you must have iTunes installed before you can package and deploy your content to the device. It is also worth noting that if you have written any C++ code, you require a Mac to compile any code you have written for final use.

iOS packaging produces a single IPA file containing all the binary code and art assets that your game needs for distribution.

#### Consoles

Consoles, such as the Nintendo Switch, PlayStation 4, and Xbox One, each have their own specific process and rules related to packaging. Unlike other platforms, you must be an approved development partner with the company to obtain any specific hardware, agreements, and software that may be required for development.

Each console has similar processes for releasing a product, which generally involves a submission and review phase.

Consoles also have a technical requirements checklist (TRC), which is a list of functionalities that your game must include in order for it to pass review. The contents of these documents are confidential, but they generally include technical, usability, and error-handling requirements.

### Patching

Your project may require post-release updates to fix bugs or add new features and content. These post-release updates are known as “patches.” During patching, the engine compares the content in the released version with the content in the new version to determine which parts need to be updated. In this way, only the parts that have changed need to be included in the patch. This results in a smaller file, which reduces the time and bandwidth required to distribute and apply the patch.

## 11.2. Common Mistakes

### “When It’s Shipped, We Can Start the New Project”

Once a project has shipped, the maintenance period begins. This period typically involves all team members as gameplay balancing, stability fixes, upgrades, gameplay tracking, and other issues are addressed.

### Sending the Right Build

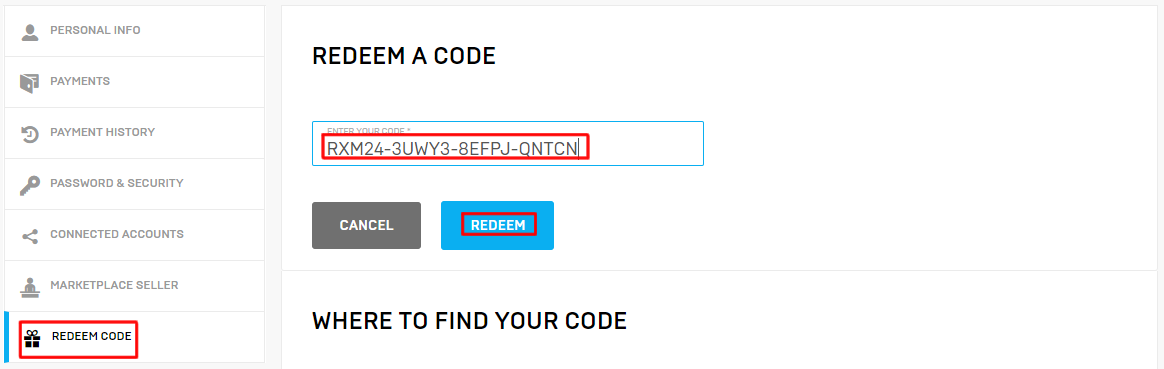
When a build is in development, it’s important that build management principles are preserved. For instance, a build going to testers should include information and toolsets to analyze and clarify potential game issues, whereas a build going to a reviewer probably shouldn’t. Sending a default build to everyone, although easier, should be done with a considered approach.

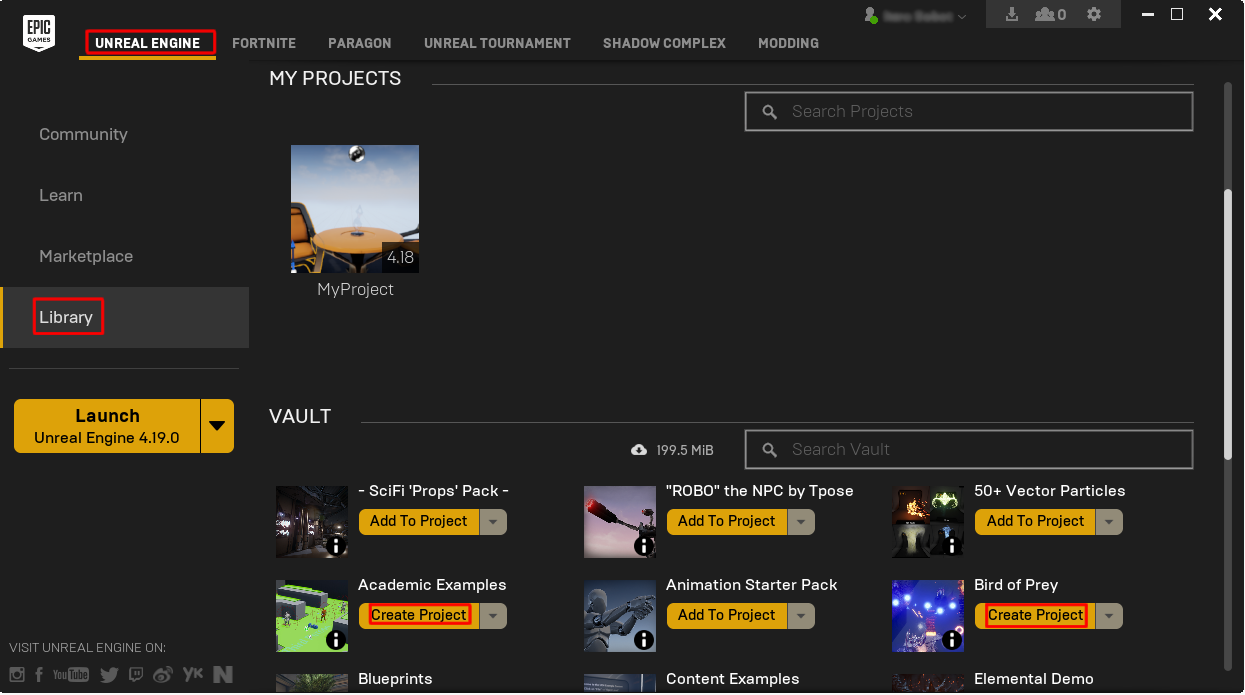
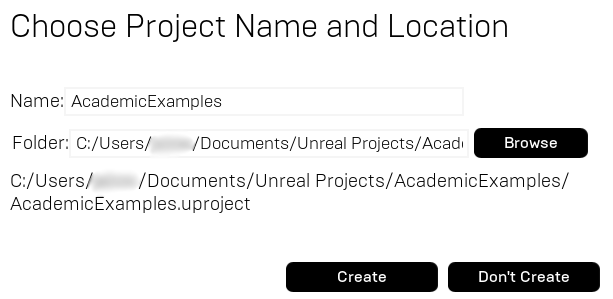
### When It’s Done for One Console, It’s Not Done for All

It’s common for new teams to assume that the similarities between consoles will make simultaneous development easier. This generally isn’t the case, as optimizing for one system may be very different from optimizing for another. Development needs to consider each console as it progresses, and if a simultaneous release is planned, a large amount of slack time needs to be considered to ensure feature parity.

## 11.3 Supplementary Resources

You will need these files to complete the Exercises on this page. These files are hosted on the Unreal Engine Marketplace and can be accessed from your Launcher by following the directions below.

1. On the [Code Redemption tab](https://www.unrealengine.com/dashboard/code-redemption) (<https://www.unrealengine.com/dashboard/code-redemption>) of your Epic personal dashboard, enter the code RXM24-3UWY3-8EFPJ-QNTCN, then click REDEEM and CONFIRM  


2. Open the Epic Games Launcher (you can download the [Launcher](https://www.unrealengine.com/download) [here](https://www.unrealengine.com/download)if you need it), navigate to the Vault section of the Unreal Engine Library tab, and click Create Project on Academic Examples or Bird of Prey  
  
  
3. Choose a project name and location and click Create  


And that's it! If you have any further questions about how to download these products from the Epic Games Launcher, please contact [marketplace-support@unrealengine.com](mailto:marketplace-support@unrealengine.com).

# Exercises

## Exercise 11A: Packaging a Game

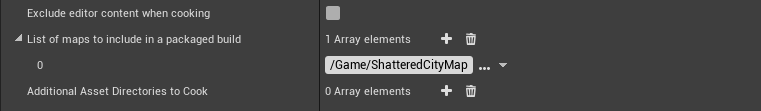
**Deliverables:** Packaged game files (as a .zip with student details)

**Instructor Task:** Discuss and demonstrate the typical build process for Unreal Engine. Have the students package a project for deployment on Windows.

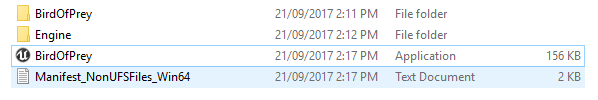
**Student Task:** Package the Bird of Prey project for deployment on Windows.

**Directions**

1. Download the Bird of Prey example from the Marketplace. See above for more information.
2. Once loaded, play through the example (in the editor) to make sure everything works.
3. Click the **Build** button and check to make sure there are no errors in the project. Check the log for any build errors and correct if needed.
4. Read through the process for packaging your project here: <https://docs.unrealengine.com/latest/INT/Engine/Basics/Projects/Packaging/index.html>.
5. Make sure to add all of the maps you wish to cook to the list found in “Advanced Packaging Settings.”



1. Package your project for **Windows 64-bit.** Make sure to choose a directory that is easy to find.
2. Locate and run your freshly packaged game. The directory should look similar to the image below.



1. Save your files into a zipped folder with the following naming convention: “*yourname\_*EX11A.zip”.

## Exercise 11B: Deploying to a Mobile Device

**Deliverables:** Packaged game files (as a .zip with student details)

**Instructor Task:** Discuss and demonstrate the typical build process for Unreal Engine. Have the students investigate the Android/iOS build process for Unreal Engine 4.

**Student Task:** Investigate the packaging process for Android/iOS using the Third Person template.

**Directions**

1. Create a simple project based upon the Third Person template.
2. Load your project and play through it (in the editor) to make sure everything works.
3. Read through the process for packaging your project here: <https://docs.unrealengine.com/latest/INT/Platforms/Android/index.html>.
4. Package the game for Android and inform your instructor.
5. Save your files into a zipped folder with the following naming convention: “*yourname\_*EX11B.zip”.