Unreal Engine RPG Game Design & Development Blueprint Tutorial Overview

Introduction

The Unreal Game Engine RPG Game Design & Development Blueprint Template by Kidware Software is an ideal starting point for creating your very own RPG game. While you can import all the Blueprints and content to another project it is recommended creating a new project from the Unreal Game Engine RPG Game Design & Development Blueprint Tutorial and using it as a base for your own project.

Before you Get Started

The Unreal Game Engine RPG Game Design & Development Blueprint Tutorial uses a lot of Blueprints to implement the given system. If you have no prior experience with the Blueprint system it is not recommended that you get started with it right away. Epic has quite a lot of tutorials which will teach you a lot of the basic stuff. This documentation assumes that you have at least a basic knowledge about Blueprints.

If you are completely new to the Unreal Engine and haven't used it at all, you should start here: Get Started with UE.

This guide assumes that you are using the **5.6** version of the Unreal Game Engine RPG Game Design & Development Blueprint Tutorial.

Blueprints Overview

This section gives a quick overview over all the Blueprints that come with the Unreal Game Engine RPG Game Design & Development Blueprint Tutorial.

Character Blueprints

Inside "Blueprints/Characters" there are a few different Blueprints which provide different functionality. The Blueprints inside the "Blueprints/Characters/AI" are covered in the "AI/Enemy Blueprints" section below.

Blueprint Class	Explanation
BP_PoseableChar	This character is not meant for gameplay. Its sole purpose is to visualize something, e.g. a class. An example is the "Character Selection" screen where you can choose between two different classes/characters.
BPI_Character	This is the Blueprint Interface for the character. It introduces a few functions which each player character has to implement. Further information on interfaces can be found here.
RPGCharacter	The RPGCharacter is the one Blueprint which includes most of the functionality.
RPGPlayerController	The PlayerController for the main player character. It isn't that important, as it doesn't contain a lot of functionality.

AI / Enemy Blueprints

The Blueprints inside the "Blueprints/Characters/AI" are the ones who drive all NPCs in the game regardless of being an ally or an enemy to the player.

Blueprint Class	Explanation
BPI_Enemy	The Blueprint Interface for the enemy. It makes sure that every enemy implements the most needed functionality, e.g. like a TakeDamage function.
BPI_EnemyController	This Blueprint Interface is used for communication between NPCs. The StandardEnemy has the ability to alert other enemies if he has spotted a target or has heard something. This Blueprint Interface is used to find all nearby enemies and alert them.
BTDecorator_ and BTTask_	These are custom Decorators and Tasks for the Enemy Behaviour Trees. More info on behavior trees: https://docs.un realengine.com/latest/INT/Engine/AI/BehaviorTrees/
EnemyController	The controller for the enemy. Both the melee and the ranged enemy use the same controller as they share almost the same functionality. Their different behavior results in a different configuration. The controller implements the Behavior Tree which is using a set of custom Decorators and Tasks.
QuestGiver	The QuestGiver is a friendly NPC. It shows an example of how to use the Quest- and the Dialogue System.
RangedEnemy	The Ranged Enemy. It uses the EnemyController (as AIController) and a Behaviour Tree to implement basic AI behavior.
StandardEnemy	The Standard Melee Enemy. It uses the EnemyController (as AIController) and a Behavior Tree to implement basic AI behaviour.

Gameplay Blueprints

Blueprint Class	Explanation
AdvancedCamera	The advanced camera has some extra functionality added. It is based on the standard camera.
BP_GameModeStandard & BP_MainMenuGameMode	As the name suggests these two Game Modes are used to differentiate between the actual Gameplay (BP_GameModeStandard) and the menu mode (BP_MainMenuGameMode).
BP_QuestComponent	This is a custom component which can be added to a Pawn to enable the stat system for that Pawn. The RPGCharacter implements this component to showcase how it can be used.
BP_StatsComponent	Much like the QuestComponent the StatsComponent is a custom-built component. It implements the functionality for a Pawn to use stats. The RPGCharacter showcases how to use this system. The RPGCharacter implements stats like Strength, Agility, Intelligence through this system. The StatsComponent is only used for the management of these values. The visual representation is handled through a UMG widget.
BPS_SingleStat	The StatComponent consists of a variable number of SingleStat objects. Each SingleStat comes with a name and a value.

Item System Blueprints

Class	Explanation
ActionInterface	This interface is used so that the player can interact with an item. It is used to picked up an item, but this could also be used to trigger special events or implement any kind of custom behavior.
BPI_Trigger	This interface is used to let the player know that he can interact with an item (e.g. displaying its name or a pickup message)
InventoryStruct	The inventory struct defines what information is stored for an item. At the moment these are the actor (for mesh representation), the item image, the item name and the item type.
Item	Main class for items.
Item_Weapon	Subclass of item with custom behavior for weapons.
ItemType	The items are divided into different categories. For example, there is made a differentiation between items that the player can equip (weapons, armor) or items that can be consumed (e.g. healing potion).

UMG Blueprints

Blueprint Class	Explanation
CharacterSelection	The CharacterSelection Blueprint is used in conjunction with the main menu. It contains the overlaying info and the buttons to choose from different character classes. Like the main menu it is not activated by default. To enable it, have a look at the level blueprint. There is a comment on how to enable the main menu and the character selection menu. The character selection screen uses a separate scene which is loaded via level streaming. After the player selects a character the scene is discarded.
CharacterStatsBox	The CharacterStatsBox is a sample that shows how to make us of the stats system. It displays the stats that are setup and saved in the RPGCharacter.
Dialouge	The dialogue is responsible for displaying the dialogue between the player and a NPC. It includes a "PlayerChoiceButton" (see below) to give the player multiple options to choose from.
Inventory	The inventory blueprint displays the items the player currently has, the CharacterStatsBox (for stats like strength, etc.) and a player model with equipment slots to visualize the currently equipped items.
LostScreen	The LostScreen widget simply displays the "You are dead" message and a button to restart the whole game.
MainMenu	The MainMenu displays various buttons to begin a game, exit the game. It handles the transition to the CharacterSelection screen.
PickupText	The pickup text is used to indicate that the player can pick up an item and displays the name of the item.
PlayerChoiceButton	The PlayerChoiceButton is part of the Dialogue System. It includes one or more button to give the player various options in the dialouge.
PlayerHUD	The PlayerHUD displays everything that is permanently displayed on the screen. It is responsible to display the players health and mana bar.

Animations

The Unreal Game Engine RPG Game Design & Development Blueprint Tutorial comes with a set of animations that are used by the RPGCharacter, the NPCs and the enemy skeletons. You can find all the animations in the subfolder "Animations".

Attack Animation and Combos

There are two attack animations which are used by the combo system of the RPGCharacter. The animations have matching frames. This means that the first frame of the first attack animation matches with the first frame of the idle animation. The last frame matches with the first frame of the second attack animation. The last frame of the second attack animation matches with the first frame of the idle animation. That way a smooth transition between the attack/idle and between the two attack animations (for a combo) is possible without the need of animation blending.

The Animation Graph for the RPGCharacter comes with some comments and demonstrates how such a system can be setup.

The Animation Graph itself isn't very complex. Almost all of the logic for the combo system is included in the RPGCharacter. The Animation Graph is using Animation Notifies (which are setup in the animations itself) to let the RPGCharacter know when a certain position in the current animation is reached (e.g. end of a swing).

Extending the Unreal Game Engine RPG Game Design & Development Blueprint Tutorial

The Unreal Game Engine RPG Game Design & Development Blueprint Tutorial is built in a modular fashion to make it easier to extend the systems. This chapter discusses a few possibilities on how to extend the system.

Adding new Actor Components

Actor components can simply be added to any actor. This makes it easy to add additional functionality which is being by different actors. E.g. adding a system that displays the damage taken in a fight above the head of the actor. All logic should be handled in this component. That way it is easy to add this to the existing characters like the RPGCharacter and the Ranged/Standard Enemy. If the new component just needs a single function to be called (e.g. something like "DisplayDamageTaken") this could just be called in the TakeDamage function.

Following this scheme, a lot of new functionality can be added.

Modify existing classes

If you want to modify existing classes to add or change some functionality it is always recommend to create a child class and add/edit anything in here. That way you can still make use of updates to the Unreal Game Engine
RPG Game Design & Development Blueprint Tutorial. However, if you overwrite existing classes, you either lose your changes to that class when updating or you will not be able to apply any updates.

To add additional functionality for a class, e.g., the RPGCharacter, create a child of that blueprint. Open up the child class and now you will be able to overwrite all the functions that exist in the parent class.

Adding your own content

Adding your own content is easy. The Unreal Game Engine RPG Game Design & Development Blueprint Tutorial uses the guidelines provided by Epic. That way there shouldn’t be any circumstances that prevent you from adding your own content.

Adding new characters & animations

At the moment the Unreal Game Engine RPG Game Design & Development Blueprint Tutorial is using the pre 4.8 skeletons. If your own character is using the 4.8 rig you have to retarget the animations or use your own animations. For retargeting Epic is providing a detailed tutorial. The same goes for custom animations that are added to the project.

Switching out a character is quite simple. To switch out the character of the RPGCharacter just select a different mesh and a corresponding animation blueprint. If you do not want to create a new animation blueprint you can just retarget the currently existing animation blueprint. For further information about animation blueprints have a look at the Animation System Overview.

Adding new items

The easiest way is to just copy an existing item and adjust the necessary values. As an example, try copying the sword and open up the copy. Have a look at the default values like name, description and the mesh.

For a weapon mesh to function properly there are a few requirements that need to be met. The origin of the FBX has to be set to the point where the weapon will be attached to the characters hand. Moreover, for the weapon trails to function correctly there are two sockets that need to be added to the mesh - a start and an end point for the weapon trails. Have a look at the sword mesh to see a correct setup of these sockets.

Creating a new item type

Currently the items are organized into a few different categories: Weapon_1Handed, Weapon_2Handed, Armor_Shield, Armor_Helmet, Consumable. These items types are used to differentiate to do what an item upon use. E.g. a weapon gets attached to the character hand and the character can now attack. A helmet gets attached to the characters heads and so on.

If you want to create new item types, open up the Enum "ItemType" inside the "Blueprints/ItemSystem" folder. Here you can simply click the "New" button on the upper right corner.

To make use of the newly added you will need to the edit the RPGCharacter blueprint. Inside the "EventGraph_Inventory" there is a custom event called "ItemUsed". It includes a switch based on the ItemType. After adding a new item type you can now implement your custom logic here.

Conclusion

In this chapter, we have presented a quick introduction of all the Blueprints that come with Unreal Game Engine RPG Game Design & Development Blueprint Tutorial. In the next chapter, we will explain how to integrate custom 3D Character Model into your RPG game and how to re-target animations for the custom 3D Character Models.