2D Action RPG Starter Kit Documentation

About this Starter Kit

This starter kit was designed to provide completely custom, and customizable examples for learning or getting a head start on building 2D Action RPG's. Read through the scripts, add on to them, and write new ones to build your own custom 2D Action RPG. All of our scripts are completely annotated and tell you exactly what is going on in each of them. If you're new to scripting this is a great way to see how things are written and simple practices for creating different behaviors in games.

All of the assets included in this file are free to use, but we recommend creating your own media to make your game unique. All of the scripts can be used as-is but we also recommend using them as a reference or build onto to create unique and great games.

Preparing Scenes for Building

By default, the scenes included in the kit should already be in the build settings list, but if for some reason they aren't, make sure you do this process below before you try to compile the game.

When adding scenes to the Build Settings, make sure the scene "loader" is at the top of the list in Build Settings. Loader is a scene that most of our kits have that contain objects that need to be carried through all of the scenes. Loader is there to be "loaded" only once and cannot be accessed again while playing the game. In the RPG Kit, things like the music manager, player, and GUI are all in Loader and will be carried over to all of the scenes when loaded.

> To do this, open the scene "loader", then in Unity go to File>Build Settings...
>
> Then click "Add Current". Once Loader is added, then do this same process with the rest of the scenes (menu, playerhouse, town, monsterarea). Order doesn't matter at this point as long as the scene "loader" is first. You can create as many scenes as you want, but you have to add them to the build settings as well.

Building for Web/Desktop

You do not need to tamper with orientation or player settings for building the kit for web/standalone builds. Make sure that the steps in "Preparing for Scenes for Building" are correct in the project before trying to build.

Inputs and Controls

The controls for the game were designed to be simple for players, but to do that a lot of effort was put into the scripts to make simplified controls work correctly.

Controls mostly reside in playercontrols.js, but also reside in other scripts. playercontrols.js manages all of the movement for the character, both WASD and arrows work for moving. Attacking with spacebar resides in playerweapons.js. Opening inventory with "e" is in playercontrols.js. Talking to an NPC with "e" is in npctalk.js. Purchasing an item with "e" is in itemforpurchase.js. Opening a chest with "e" is in chest.js.

Notice a lot of controls in the game only require one button. This is to really help simplify gameplay without need a large variety of buttons the player needs to remember when doing certain things.

Building more Scenes

To build more scenes in the game, we recommend by copying everything from the most similar scene that already exists, and then paste all of the content into the new scene and start working from there. To be able to actually load those new scenes (in this case create a new area for the RPG), you need to have a door from a previous scene that is set up to load the scene name string, when the player runs into the door.

To do this, start with the scene that has the door you want to use as the "portal" to the new scene. If you want to add a new door, either duplicate an existing door, drag and drop a prefab door from the project assets, or create drag and drop doormanager.js onto the object you want to use as the door.

Now you have to edit the public variables for the door. To do this, highlight the door or object you're using in the hierarchy, then in the inspector you will see 3 editable variables in the inspector. Change "Scene Name" to exactly the same name as the new scene you created or want to create. It must be the exact same name because the door uses the string to find the scene with that exact name. Then you must change "Teleport X" and "Teleport Z" to the position you want the player to start at in the new scene.

Then in the new scene just add another door with the same process and change all of the variables so the player can move back through the other direction (unless you want the door to be one way). Make sure to add your new scene to the build settings otherwise you'll receive an error that will tell you to do so. Take a look at "Preparing Scenes for Building" at the top of this documentation if you're unsure how to do this.

Adding Text to talking NPC's

Adding text to talking NPC's is pretty simple. You can add up to 4 lines of text for the NPC to say. Its not much text but we created it to mimic classic action rpg's.

Just duplicate an existing NPC, or drag and drop one from the prefabs list into the scene.

Highlight the new or pre-existing NPC in the hierarchy. You should see an arrow to the left of the NPC name. Click on this and it will reveal the child object named "talktrigger".

Highlight "talktrigger".

Now you should see 4 lines in the inspector that are editable. Replace or add text to the lines that you want the NPC to say. Note that you shouldn't add too much text to each line, otherwise the text will exceed the bounding box texture created for talking.

If you want to make the box larger, simply replace the current box GUI that's used for talking, then move the strings that are children of the box over to line up better with the left side.

Creating a Purchasable Item

Purchasable items are not too difficult to create, but to make them useable they must be available in the inventory. We will discuss this further in "Adding an Item to Inventory". By default all of the purchasable items in the demo are in the "blacksmith" scene.

To create a new purchasable item, either duplicate and existing item, or drag one from the Prefabs folder "Items" in the Project Assets. Something like "purchasebow" will work fine.

Then replace the texture of it, rename it to what you want, and drag it in to prefabs so you don't edit over the old item.

Highlight the new purchasable item. Notice that there are public variables that need to be changed on the "itemforpurchase.js" script in the inspector.

Change the Item Id to a unique Id Name for your new item. This must be unique because the id's are saved with playerprefs and are accessed from inventory.

Now change the price to what you want it to be.

Now change the quantity for how many of that item you want to give to the player.

Now decide whether the item is a single purchase or not. This means if the item can only be purchased once during the entire game, for example the bow can only be purchased once because once the player receives it, it's always in their inventory from then on, and they shouldn't be able to purchase it again. We recommend that most items are Single Purchase. The potion is also single purchase, but once the player uses it, the item will reappear in the store and be available for purchase, even though he already purchased it at one point.

Now locate the child object of your new purchasable item called "price" this is a GUIText object that shows the price of the item. Change it to whatever price you set for the item.

Adding an Item to Inventory

Adding an item to the inventory is easy. The selection controls will automatically notice something is available and you will be able to select the item. The item will automatically be available if you don't set an itemId. Setting an itemId for matching up the item with a purchasable item. To create a purchasable item, see "Creating a Purchasable Item" above.

To add an item to inventory, go to the scene "loader".

Navigate through "permanentobjects/GUI/inventory". You should see 16 slots and selection.

Highlight the slot you want to add an item to.

Add the texture you want to Texture in GUITexture in the inspector.

Now under the script slot manager you will see one public variable you can add to, Item Id. If you created or want to create a purchasable item, this item id needs to match with the exact same name as the id you set in the purchasable item. To create a purchasable item, see "Creating a Purchasable Item" above.

Now, to make the item useable, you'll have to add your own code to the script, playerweapons.js.

Open playerweapons.js and take a look at the code. This is where items from the inventory are managed. Even the potion. You should see code a ways down in the script like this:

```
if(PlayerPrefs.GetFloat("slotSet") == 3){
        curWeapon = wand;
        useDir();
        wand.SetActive(true);
        wand.SendMessage("magicShot",
        SendMessageOptions.DontRequireReceiver);
        audio.PlayOneShot(magicSound);
}
```

To access the slot with your new item selected you would just need to check what slot is selected like this:

```
if(PlayerPrefs.GetFloat("slotSet") == 5){
        //your code goes here
}
```

Make sure the number, like the 5 above, matches the same number as the slot you added an item to in the inventory. Then add your code in between those lines. It'd be best to add each slot in descending order in the script.

Best Practices for Creating Custom Textures

Even though you have the rights to do whatever you want with the textures given to you in the game, we highly recommend that you create your own textures for the game to really make it feel unique.

First, you should take a look at how we've created the animations for the players, npcs, and enemies to get an idea of how many frames you should create. If you want to create sprites that require a lot more frames, you'll just have to update the scripts that contain the animations to be able to house the amount of textures you desire. Basically just a lot of copy pasting.

I have two recommendations for putting your own textures into the game.

First, the easiest way to use your own textures is to simply navigate to the folder that holds all of the textures you want to replace through Explorer or Finder (not inside Unity). Create and rename all of your textures to the exact same name as the textures that you want to replace (ex. You want to replace grass with a better texture you created) and just replace them in the folder.

Second way would be to import or drag and drop all of your new textures into the Project Assets folder, then go through all of the items in the game and replace all of the textures inside Unity.

Please contact us if you have any Questions!

Our kit may be sort of strange because we've basically created a game without finishing it. It's fairly modular but we expect people to ask us questions or make suggestions if theres something stupid about our kit.

Please feel free visit http://www.cinoptstudios.com/contact/ or email us at support@cinoptstudios.com.

Thanks for your support and we hope to hear from you!