Easy Weapons

Version 1.0

Easy weapons include two sets of fantasy medieval style weapons, one is a ready to use set that's been pre-colored; the other is a RGBA colored set that can be used to recolor the meshes.

This pack requires:

- Animation Rigging (included in unity registry)
- Mathematics (also in unity registry)

Table of contents:

Getting Started

In sample scene 1, there are two sets of each weapon, a regularly colored one, and the other is just pure red, green, blue, and black. The regular colored one is an example which has a prefab ready to be implemented wherever you need it. However, it is likely that you may not be satisfied with the limited color range of the provided examples.

This is where the RGBA variation comes in. The idea is that while in play mode, you can pick any of the RGBA prefabs or scene objects (I would use the prefabs since they are pre centered), add it to the Prefab To Recolor field of the RECOLOR TOOL, pick the relevant colors in the field below (Alpha channel corresponds to smoothness in the shader), press the Recolor Mesh(s) button, if satisfied press the Save as New Prefab button. After that you can save the new prefab, and generate meshes anywhere you wish.

Note: some of the prefabs have a clone that is slightly larger and has slightly different colors as a child. This is the enchanted variation of the weapon, when this variation is disabled the base weapon is revealed. When recoloring the RGBAs it's important that the enchant variation is enabled. Otherwise it will be copied as an empty mesh.

Note 2: when you press the 'Save as New Prefab,' and the save window appears, no matter what you type in to save it as, it will always save as the objects name in the scene hierarchy

Listed Procedure for recoloring:

- Enter play mode
- Find the RGBA prefab of the item you want to recolor
- Select the RECOLOR TOOL in the scene hierarchy
- Add the prefab to the 'Prefab To Recolor' field
- Adjust colors as desired (Alpha becomes smoothness)

- Change the ‘Enchantment Glow Mult’ depending on intended lighting and volume settings
- Press the ‘Recolor Mesh(s)
- New Object should appear as a child of the RECOLOR TOOL in hierarchy
- Rename the object in the hierarchy to desired file name
- If happy with result, press ‘Save as New Prefab
- Pick a folder to save prefab and its accompanying meshes.
- Exit playmode, and now you should have a new prefab

Compatibility

By default the Easy Weapons pack is compatible with Unity URP 2020.1 and above. In order to make this package compatible with HDRP, all you would have to do is switch the Active Target from universal, to HDRP in the graph settings of the VertexColoredMetal shader in the shader graph. Also, when switching to HDRP, sometimes having the exposure override enabled in the fog volume will force you to turn the glow from enchantments up to 1000000+. Which will require making new Prefabs, and will be super over powering in dark lighting.

Vertex Shader

The only shader this pack uses is the VertexColoredMetal (VCM) shader. The only input this shader takes in is a greyscale smoothness map, and the vertex colors of the mesh it’s applied to. This allows the use of only one material(Metal_VertexColorMat) for all the objects in this pack, except for the bow string which uses the standard Unlit shader. Anyway, within the VCM shader the vertex colors are used as follows: Red, Green, Blue determine the color of the mesh, and Alpha determines the max smoothness value, with the lowest always being 0. Where

the highest and lowest smoothness values are applied is determined by the ‘SmoothMap’ texture, where white always represents the highest possible smoothness, and black represents 0 smoothness.

Note: This pack comes with three different possible tileable smoothness textures, with ascending levels of detail and scale. They can be substituted with each other at any time.

Note 2: If one or more of the RGB values exceeds one, the shader will interpret that vertex as giving off an emission of ONLY the values that exceed 1.

Scripts

Recoloring Tool

The ‘RecoloringTool’ script is in charge making new variations of prefabs that have different color values as the original, and then converting this new variation into its own unique prefab, making new Mesh assets in the process. In general it does this by making an instance of the original Prefab, swapping the original colors for the new ones in this instance, then saving the new Meshes as individual Assets, then saving the overall object as a prefab.

ReColor() - Creates a new instance of designated Object, loops through every Color within every active mesh within the instance, then remaps pure RGBA colors to new Colors.

- R (1,0,0,0) gets recolored to the blade Color parameter
- G (0,1,0,0) gets recolored to the hilt Color
- B (0,0,1,0) gets recolored to the handle Color
- A (0,0,0,1) gets recolored to the enchantment Color

LockInAndCreateNewPrefab() - loops through all the active meshes within the recolored instance, then converts all of them to mesh assets, then it converts the instance into a prefab at

the desired path. Desired file name should be given to the object in the hierarchy, no matter what is put in the ‘save as’ section, it won’t stick.

Recoloring Editor Controller

Simply in charge of making the buttons ‘Recolor Mesh(s)’ and ‘Save as New Prefab’ appear in the recoloring tool inspector.

Weapon Controller AA

Weapon Controller is a placeholder substituted for whatever you will need to control melee weapon colliders and such. It also serves as the base script for the bow, gun, and shield scripts by taking advantage of the inheritance system.

UseWeapon() - currently blank, but could be used to enable damaging colliders, creating particle effects, or sound effects.

StopWeapon() - currently blank, but would be used to disable everything that happened with UseWeapon().

UseEnchantment() - if the item this script is applied to has an enchanted model, then this will enable that model, or it can be used for throwing, although that part is blank.

StopEnchantment() - this disables everything that UseEnchantment() enables.

Note: making all the other weapon controllers inherit from this one means that when you are wanting to reference any of the weapon controllers, you can simply reference them as WeaponControllerAA, and call the same base functions, even though they do different things.

Bow Weapon Controller AA

Does everything the Weapon controller does, but it also controls how the bow animates back and forth when firing, and also controls the string of the bow.

UseWeapon(), StopWeapon(), UseEnchantment(), StopEnchantment() - currently do WeaponControllerAA()'s same functions, but can be edited easily (Still Placeholders).

Update() - like default, this is called every frame and is in charge of positioning the string line renderer, and controls the Multi Position Constraint within the bow armature. If you have questions about any of the Animation and Rigging components, please check the animation and rigging documentation.

SetStringMaterial(Material stringMat) - sets the string material whenever the bow is instantiated, or at the start.

rotateToStartPosition() - This is part of the bow creation process, but should not be used anymore. It would be safe to delete, or to leave it.

NockArrow(GameObject inputArrow) - empty placeholder function for loading an arrow, would recommend using the BowDrawPoint transform as a base point for the arrow, since that is where the middle of the line renderer for the string will be.

DeleteArrow() - placeholder for deleting an arrow, either after firing, or when released from bow.

OnDestroy(), OnDisable(), OnEnable() - makes bow string match bow when bow is deleted, set inactive, or active.

Note: if you don't want to execute any of the Weapon Controllers functions, simply get rid of the corresponding base.XYZ() within the function.

Note: bow string won't appear until you enter playmode.

Gun Weapon Controller AA

Empty place holder for gun control. If you don’t want to execute any of the Weapon Controllers functions, simply get rid of the corresponding base.XYZ() within the function. Inherits from WeaponControllerAA.

Shield Weapon Controller AA

Like the other weapon controllers, shield inherits from the WeaponControllerAA. However while it contains the same override voids they do not perform any of the same actions, but can be called from the same reference.