Bloody Mess:

Positional Data & Dismemberment

email: heavydieselsoftworks@gmail.com

Introduction

Thank you for purchasing Bloody Mess: Positional Data and Dismemberment. Within this package you will find information about Bloody Mess and links to comprehensive video tutorials that will get you started on creating characters that react to a more realistic, and satisfying, damage model. If you have any questions or problems please feel free to email us at the email address provided above.

A Word About The Web Demo

If you have played the web demo then you will know that a lot of third party assets were used to help show off Bloody Mess. We have included all the example scripts and prefabs from each demo in the zip files found in the Scene folder. In order to get them to work you will need the following assets:

For RFPS Zombie Horde: Realistic FPS Prefab by Azuline Studios, Zombie Walk by Mixamo, KY_Blood FX by Kakky.

For UFPS Zombie Horde: UFPS by VisionPunk, Zombie Walk by Mixamo, Ky_Blood FX by Kakky

For 3rd Party Effects: Exploder by Reindeer Games, Holographic Dissolve Shader by Wabo

Boss Fight: RFPS by Azuline Studios, Exploder by Reindeer Games

Project Setup

Tags

Make sure you have each of these tags listed in your tag editor:

Head
LeftHand
RightHand
RightUpper
LeftUpper
RightForeArm
LeftForeArm
RightLeg
LeftLeg
UpperBody
LowerBody
Extra1
Extra2
Extra3
Extra4
Critical

Layers

Make sure you have the following layers in the layer editor at the appropriate User Layer Number:

Player (User Layer 11)
Limb (User Layer 21)
Ragdoll (User Layer 22)
Enemy (User Layer 23)

Physics

In Edit -> Project Settings -> Physics you need to make sure the Limb layer doesn't collide with Enemy or Player layer (uncheck where they cross in the graph).

Getting Started

Bloody Mess utilizes a detailed video tutorial series to get you started. The links below will guide you through the complete setup process.

Project Setup

The following is a video example of the above Project Setup. Do this before doing any other tutorials.
https://www.youtube.com/watch?v=p9G9ZWIgt3E&feature=youtu.be

Setting Up a Character

The following are comprehensive tutorial videos for setting up a character with Bloody Mess. It is recommended that you do them in order (Basic -> Advanced)

Basic Character Setup

https://www.youtube.com/watch?v=eIYSJwNkI_Y&feature=youtu.be

Advanced Character Setup for Dismemberment

https://www.youtube.com/watch?v=zv9dc2abyvk&feature=youtu.be

Overview of Scripts

https://www.youtube.com/watch?v=PFjZu9uYv6I&feature=youtu.be

Using the Event System

The following tutorials cover how to use the two Bloody Mess event systems.

Using the Damage Event System
https://www.youtube.com/watch?v=K4KEUvQj40s&feature=youtu.be

Using the Healing Event System
https://www.youtube.com/watch?v=aFVaZ63tqHA&feature=youtu.be

3rd Party Integration

The following tutorials cover integrating 3rd party assets will Bloody Mess.

Integrating Bloody Mess with RFPS
https://www.youtube.com/watch?v=gEwmVJ-l6TE&feature=youtu.be

Integrating Bloody Mess with UFPS
https://www.youtube.com/watch?v=qXjh-oPwYRc

Integrating Bloody Mess with Easy Weapons
https://www.youtube.com/watch?v=nDusg0LN3v4

Integrating Bloody Mess with Pool Boss
Coming Soon!

Integrating Bloody Mess with Dialogue Manager
Coming Soon!

Integrating Bloody Mess with Easy Save
https://www.youtube.com/watch?v=t10n4tPWcYY

See More Tutorials At
https://www.youtube.com/channel/UCH0dmpRB8wA3esWgYzbg-qA

CharacterSetup.cs

The CharacterSetup.cs script is the heart and soul of Bloody Mess. The following will give you a detailed description of all of the variables, broken up into their own sections, that make up the script.

General Setup

Load Saved: Check this if you want to load health data from playerprefs or a save game asset. You will need to add your own hook into the CharacterSetup.cs for your save game system. There is an if statement in the Awake function that is provided for this purpose.

Target: This is used to tell the system what gameObject will be using the Damage Event System. You can leave this field blank if the scripts accessing the event system are on the same object as CharacterSetup.cs (this is recommended for ease of use)

Renderers Parent: This is where you place the top level of your Skinned mesh renderers. It is good practice to put all of your various body part renderers as a child to this.

Skeleton Parent: Place the top level of your skeletal hierarchy here.

(Both Renderers Parent and Skeleton Parent are used to globally disable your character on death therefore they are required)

Use Pooling: Set this to true if you are using an outside pooling method and want it to handle the enabling and disabling of characters.

Destruction Timer: This only shows up if Use Pooling is unchecked. This is the time it takes to destroy the whole character game object after death. You don't want to destroy it directly upon death as some functions of the CharacterSetup.cs and its event system may still be running. Instead we hide the character (as described above) and let it finish its functions before we destroy it. It is recommended to keep this value about 5.

Ragdoll Setup

Automatic Ragdoll?: Check this if you want to have Bloody Mess automatically handle the ragdoll upon health reaching 0. If you want to use death animations, an outside ragdoll plugin or just have more control over the timing of ragdoll spawning then leave this unchecked.

Ragdoll: The transform prefab that will be spawned as a ragdoll (needs to have the RagdollLogic.cs script on it)

Ragdoll Wait Time: how long after heath = 0 that a ragdoll is spawned

Destroy Ragdolls: check if you want Bloody Mess to destroy ragdolls, or uncheck if you want to let a pooling system do the work

Ragdoll Stay Time: Only shows if Destroy Ragdolls is checked. This sets the time, after spawning, a ragdoll is removed from the scene.

Health Setup

Max Global Health: the maximum health of the character, when this is 0 the character dies

Max Per Limb Health: Each of the values following Max Global Health are individual health for each limb. If you are using dismemberment, when one of these equals 0, the corresponding body part will be hidden and a dismembered limb will be spawned in its place. If you want a limb to never be dismembered you can set its health much higher than total health (or turn off body dismemberment).

Damage Multiplier Setup

These values take the incoming damage from whatever weapon system you are using and multiply it times a value on a per limb basis. By setting the value below one you can make hitting that limb give less damage towards itself and total damage.

Dismember Setup

Use Advanced Ragdoll: Check if you want the character to ragdoll itself instead of spawning a separate ragdoll. To use this you must have rigidbodies and character joints set up properly on the character. Make sure all colliders are set to trigger by default and all rigidbodies are set to isKinematic. Failure to do any of this will result in errors.

Use Head Dismember?: Check if you want to be able to dismember the head

Use Body Dismember?: Check if you want to be able to dismember individual body parts.

Use Explosive Dismember?: Check if you want to be able to do full body at once dismemberment via an external explosion script

(The following three variables are used to allow you to set what weapons can do dismemberment and on what body parts)

Weapon Type Head: what integer value of weapon type would you like to be able to dismember the head.

Weapon Type Limb: what integer value of weapon type would you like to be able to dismember limbs.

Weapon Type Torso: what integer value of weapon type would you like to be able to cut the torso in half.

Body Parts To Spawn

(only shows if use Head Dismember or Use Body Dismember is checked)

The Transform fields found here are for each body part you want to be spawned in place of a missing limb. Refer to video tutorials on how to set up body parts.

Body Part Skinned Meshes

(only shows if use Head Dismember or Use Body Dismember is checked)

The Skinned Mesh Renderer arrays found here are for you to specify skinned meshes for each body part. You need to include the body part and its children to get rendering right. For example, for Upper arm you would need to include the skinned mesh renderers for upperarm, forearm and hand (if your character includes those three objects).

Objects(Parts) To Spawn On An Explosion

Explosion Parts: This array is where you place all the body parts, or other objects you want to be spawned when a body explosion is called.

Body Part Colliders

The collider array fields found here are for you to place colliders for body parts. Just like the skinned mesh fields above you need to include children of the bodypart. Example, Left leg needs the collider for left leg and left foot.

RagdollLogic.cs

RagdollLogic is the script that makes ragdolls appear the same as the original characters. Example, if the original character is missing an arm then the ragdoll will also be missing an arm.

Use Pooling: Check this if you want ragdolls to be pooled by an external pooling system and not destroyed by RagdollLogic.cs

Is Upper Body: Check this if the current ragdoll is going to be the upper body ragdoll (when using full body dismemberment)

Is Solo Mesh: Check this if the mesh doesn't need to include children. Example, hands.

Skinned Mesh and Collider Arrays: Fill these out just like you did on the CharacterSetup.cs.

Blood Game Objects:

The Blood Game Objects are just positions where you can place a blood prefab if you wish. Upon death this blood will be turned on for each missing limb on the ragdoll.

Damage Event System

The Damage Event System is broken up into 3 different event handlers; IBloodyMessEventHandler, IBloodyMessHitEventHandler, and IBloodyMessExplosionEventHandler. You must like to one of these in your script like this:
Ex.

```
Public class YourScript : MonoBehavior, IBloodyMessEventHandler  {
        //your scripting
        //BloodyMessEvents
}
```

IBloodyMessEventHandler

public void OnLimbDeath(int limbID, GameObject spawnedLimb, GameObject removedLimb, Vector3 attackerPosition, Vector3 attackerDirection)

OnLimbDeath gets sent every time a limb dies and is dismembered. You can use it to add force, or otherwise effect or modify the spawned limb. You can also use it to turn on blood effects for a character that is still alive or even modify animation states.

public void OnDeath(Transform ragdoll)

OnDeath gets sent whenever a character dies and a ragdoll is spawned. This allows you to do any number of achievement or point related things as well as make modifications or otherwise effect the ragdoll.

IBloodyMessHitEventHandler

public void OnLimbHit(int limbID)

OnLimbHit gets sent every time a limb is damaged. It is useful for applying forces, incrementing combos or achievements.

IBloodyMessExplosionEventHandler

public void OnExplosion(Vector3 position, Vector3 direction)

OnExplosion is sent only when the ExplodeBody function is called from an outside script.

Healing Event System

Bloody Mess's healing event system allows you to heal total health, all limbs or individual limbs. This is perfect for any player character damage handling system and comes with only 1 event handler. The HealingEvents.cs script should be placed on an empty game object in your scene and have its Events activated by outside scripts. See the Gui Healing scene for an example of how to set up the system correctly

IHealingEventHandler

public void AddHealth(float amount)

AddHealth adds health points to the total health pool.

public void AddLimbHealth(float amount)

AddLimbHealth adds health points to all limbs (but not total health)

public void AddLimbHealthSelective(float amount, int limbID)

AddLimbHealthSelective adds health to only the selected limb.