BladeSmith
Melee Combat System

Synopsis

BladeSmith is a movement-based damage applying and recieving system. Sounds fancy, but in short it allows you to set up your own weapons which hit precisely as they move, will it be through your own animation or scripting. And it's all framerate independent!

BladeSmith takes care of the damage dealing part, as well as triggering hurt and death block-stagger events, changing weapon's damage in mid-animation, as well as breathing life into the shields of your characters. All you need to bring is your own Animation Controller and animations - all the rest is right here.

As for the documentation, BladeSmith is using the Implemented Documentation method - what it means is simply that everything you need to know about any variable is written right on it, in a Tooltip. Simply mouse-over any variable to view it's detailed description, hints and how to use it. Only a few variables may reference this ReadMe File in more advanced situations.

The Systems

BladeSmith is divided into three sections: Health Systems, Animation Event System and Weapon Systems.

Weapon Systems take care of - you guessed it! - your weapons. It consists of the Weapon Marker Manager (the "Soul" of your weapon, which handles all of it's variables) and the

Markers (which are used to mark the damage-dealing zones of your weapon (like blades or hammerheads).

The Animation Events System is a script which should be attached to the object which holds all of the Attack and Shield Animations (like the Mecanim Character wih an Animator attached). We will use these functions to determine what and when your weapon and shield should do what (like enabling Markers, changing damage value etc.)

The Health Systems are responsible for the pain-recieving side of the party. It consists of the Main Health System (which is the core of the Health Systems), supporting Limb Hitbox system (which can be attached to other colliders to determine precise hitzones of the character), Shield system (which manages the shields).

Weapon Systems:

BS_Marker_Manager and BS_Markers

The BS_Marker_Manager is the primary way of making your weapons and attacks deal damage. Attach it to any object which is supposed to be an active Damage Dealer, like a sword, claws, guillotine traps and so on. You can also attach it to an empty GameObject set as a child of your weapon to keep it clean and tidy in your Hierarchy view.

When the Marker Manager is on the scene, attach the Marker Prefabs (from the BS Weapon Marker Systems folder) as children of the Marker Manager and set their hit Layer in the inspector. Please keep in mind, that the Marker Manager cannot have any other children than Markers!

Place your markers along the damage dealing zones (like along your sword or axe blade, or around the head yof your hammer), fill in the Marker Manager variables accordingly to your design and you're good to go. Your weapon doesn't need any colliders to work - it just needs to move in any way to detect hits! The more Markers you place, the more precise the hit check will be, but remember that few hundreds of enabled, active Markers on the scene may chip away a few FPS. Make sure to only use as many Markers as necessary. Yet, do not be affraid to add a few more Markers if you require some extra precision.

Animation Events System:

BS_Weapon_Animation_Events

If you plan on using BS Weapon Systems with any kind of animations, it's worth to mention that it's using Animation Events to maintain performance of your Weapons. This script should be attached to the object which holds your Attack or Shield Animations (will it be Legacy or Mecanim, like the Character with an Animator). After that, you can use various Animation Events to use functions vital for the BS_Marker_Manager and BS_Shield. The Events are:

-DisableMarkers() - Disables the child Markers of the choosen Marker Manager. While disabled, markers take up almost no performance of CPU or GPU. We will use this function mostly at the first frame of any attack animation (to make sure that the Markers weren't enabled before) and at the end of the damage-dealing phase of the attack (like right after the sword swing looses it's speed). This function also resets the Targets' List - what it means is that after this function was called, everyone who was already hit by any previous attacks can be hit again. Otherwise any Target which was already hit will be ignored in next hit checks of the weapon's Markers (so you hit everyone once, no matter how slow or fast your sword swing is).

-EnableMarkers() - Enables the child markers of the choosen Marker Manager. While active, your Markers will check if anything was hit during their movement - they use no colliders and are framerate independent. We will mostly use this Animation Event on the very start of the damage-dealing phase of your attack animation (the moment when the sword gains speed during a swing).

-***ClearTargets()*** - an optional function, which resets the Target's List without turning off the Markers. Useful if you want to chain the damage-dealing phases without turning off the markers (like a multiple-spin attack of an axe warrior). Advanced tip: This function works absolutely the same as calling DisableMarkers() and EnableMarkers() right after the first one, just with a little better performance.

-CancelStagger() - If you plan on using the Stagger function with a Mecanim Animator, keep in mind that after each time the stagger animation kicks in("Stagger" bool is set to True), it may need to be reverted back to false after it's done. We will call this Animation Event at the end of the Stagger Animation.

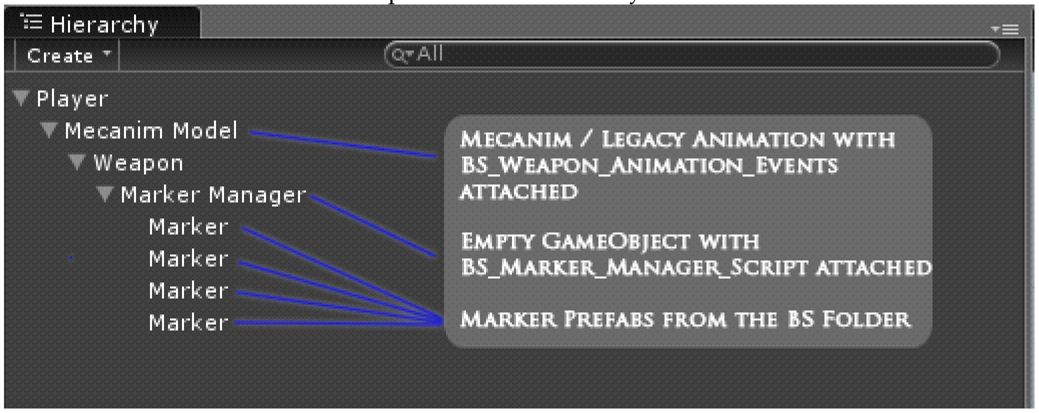
-Stagger() - This function shouldn't be used directly, as it is already called by the BS_Marker_Manager script, but if you are absolutely sure you know what you're doing - this function triggers Stagger events.

-SetDamage1() ... SetDamage9() - Sets the Damage value of the Marker Manager to given value. It is used if some of your attacks are supposed to deal more or less damage than others. We will call this Animation Event at the start of each Attack Animation to determine how much Damage should the consecutive attack deal. Keep in mind that after the Damage was set, it will not change until another SetDamage was called. You can set up to 9 different damage values for a single weapon.

-***DisableShieldCollider()*** and ***EnableShieldCollider() -*** These two functions are used to turn on and off the colliders of a BS_Shield system. We will call EnableShieldCollider() upon rising character's shield and DisableShieldCollider() on lowering it.

-***SetWeapon1MarkerManager() ... SetWeapon10MarkerManager()*** - These functions are useful if you've got animations for multiple weapons on one Animator. This function switches the MarkerManager to another (like between the Marker Manager of a sword and an axe). EnableMarkers, DisableMarkers and other Marker Manager functions will be directed to a new Marker Manager. We'll use it at the start of each attack animation, before other animation events, if we've got multiple weapons. Also, don't worry about the Markers of the previous Manager – they can be automatically disabled upon switching Managers.

Example use in the Hierarchy View:


Hierarchy
Create
All
Player
Mecanim Model
Weapon
Marker Manager
Marker
Marker
Marker
Marker
MECANIM / LEGACY ANIMATION WITH BS_WEAPON_ANIMATION_EVENTS ATTACHED
EMPTY GAMEOBJECT WITH BS_MARKER_MANAGER_SCRIPT ATTACHED
MARKER PREFABS FROM THE BS FOLDER

Health Systems:

BS_Main_Health System

The Main Health System can be applied on your characters and objects which should be able to recieve Damage of any kind - will it be an Enemy or a destructible element of the environment, like wood planks. Simply attach it to a character or Object as a new component and fill out the variables. You can also attach any 3D Collider to the same object, if you want to use simplified hitzone (like a capsule collider on a character or an explosive barrel).

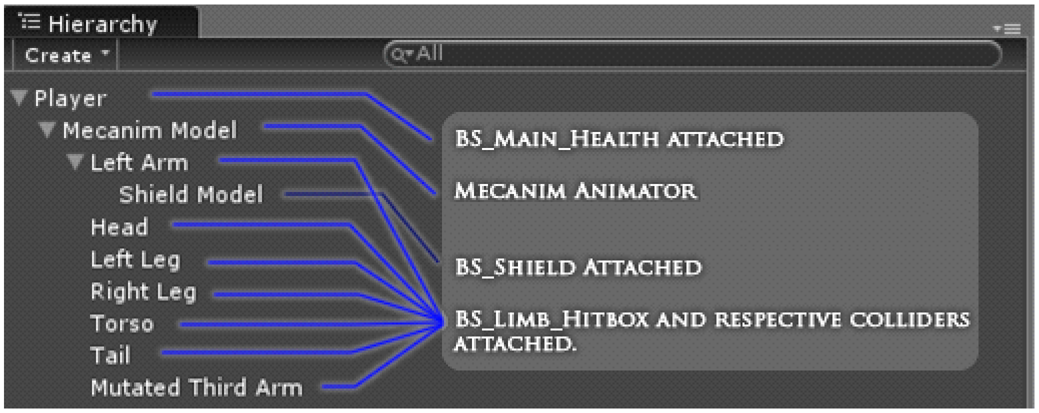
BS_Limb_Hitbox

If you wish to bring more precision into your combat and a single hitbox or hitcapsule is not enough, you can use the Limb system to manage hitboxes. Simply, if your character is using children objects as limbs (like in most 3D Character models), just attach precise colliders to all of them and add a Limb Hitbox script to each one (it doesn't need to be humanoid – the limbs can be used in any custom creatures!). In the Limb system inspector view assign the Main Health of the character and it can be hit preciesly where you want.

BS_Shield

BladeSmith allows you also to create your own reactive shields. Attach this script to any object which is supposed to block incoming damage (like a shield model, set as child of your character's arm), add up to 5 colliders, set the Edge Spots to determine where are the colliders' edges (for Advanced Shield Detection) and fill out the variables. Now you can disable and enable shield's colliders on Animation Events (like rising your shield and lowering it), as well as play blocking animations, giving you the most precise shield-play possible.

Example use of Health Systems:



WORKFLOW

1. Place the Weapon Marker Manager and weapon Markers on the weapon object. It can be a simple sword, as well as Rigidbody-driven, blade on a spinning chain – all hit detection will still be framerate independent!

2. Place the Health and/or Shield script on your characters.

3. Bring your Animation Controllers and Animation Clips for attacking, getting hurt, dying, getting staggered, blocking with your shield etc. [if needed – they are not required]

4. Place the Animation Events Script next to all required character Mecanim Animators or Legacy Animations components (if needed, same as above)

5. Set the Animation Events on your Animation Clips or set your weapon to Constant Damaging.

6. Done! Right now, your weapon will register hits and deploy different effects with the Health and Shield Systems, like: spawning blood, playing Sounds, decreaseing targets' health, SendMessaging on Hurt and on Death, playing different animations, getting staggered on shield and wall hits and many, many more!