BONE ZONES

Are you developing an FPS or turn-based game and want to let your player hit a part of the enemy? With Bone Zones, you can easily include this feature without having to split your models!

With a quick and intuitive wizard, you can divide your rigged model into different logical zones as you like, highlight the targeted area and quickly prototype your model's colliders. Save a lot of time!

Bone zones allow you to define logical zones for your models with RIG and then highlight or identify them without splitting them into sub-meshes.

Features:

- Highlight parts of an animated character without altering the model
- Highlight objects linked to the hierarchy automatically (like weapons)
- Identify the affected zone from a RaycastHit or Collider with a single line of code
- Easily associate additional information to each zone
- Simple and automated wizard, but with the possibility to check every detail in advanced mode
- Automatic recognition of Humanoid zones
- Automatic collider creation for fast prototyping
- Customization of layers and collider types
- Compatible with any model with RIG (Humanoid and non-Humanoid)
- Supports multiple SkinnedMesh in the same model
- Easily integrated with **InVector** and **Opsive UCC**
- Sources available for complete customization

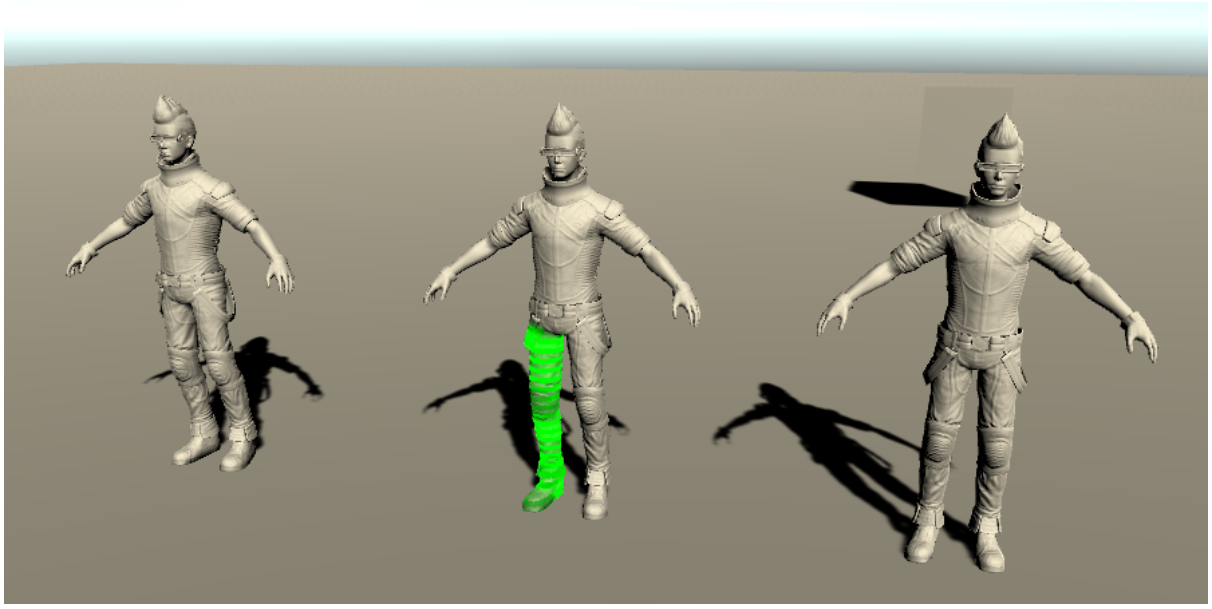
INDEX

MAIN FEATURES

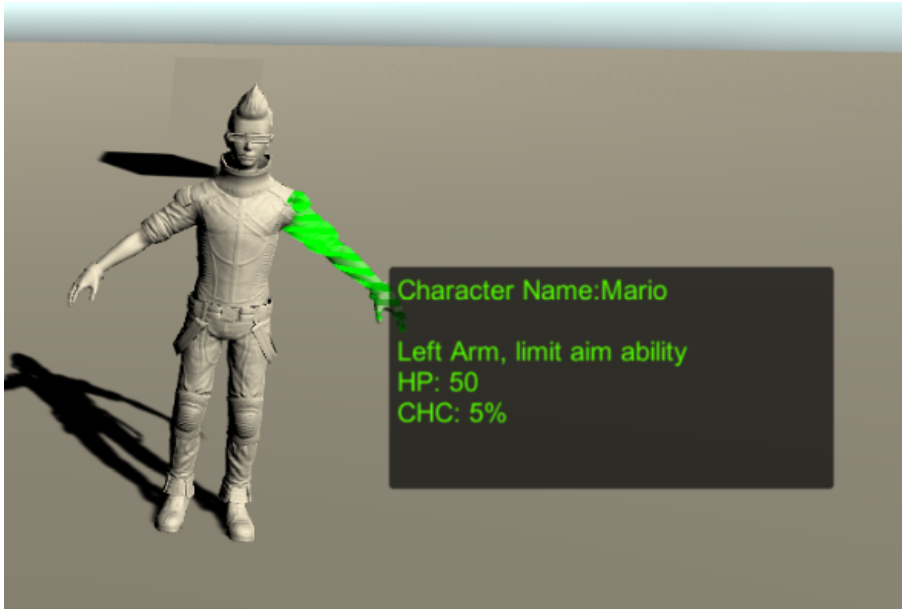
BONES ZONES lets you define **zones** based on the bone hierarchy of your characters. Based on this definition, you can activate several features like:

Highlight a zone of the model without dividing it into multiple sub-meshes.

Highlight all non-SkinnedMesh models connected to the selected zone.



Determine the zone hit by a Raycast, for example, to allow the UI to choose a character element, or to determine the zone hit by a bullet, or to determine the zone targeted by the player.



Each zone can easily be associated with additional information useful for your game.

REQUIREMENTS

Bones Zones works on models with a hierarchy of bones (RIG). The model does not have to be a humanoid character, the tool also works with generic hierarchies. In the case of humanoid models, some automatic features are still available, like the zone self-recognition.

HOW TO USE IT

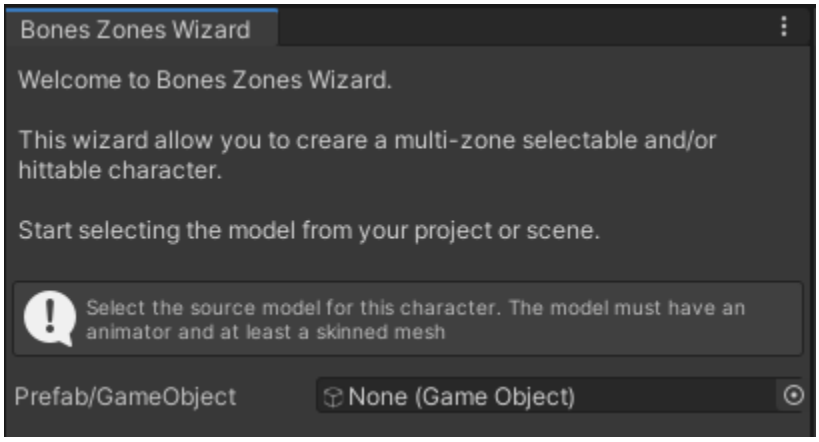
To use the functionality of BONES ZONES it is necessary to execute two steps: the preparation of models and the use of models.

MODEL PREPARATION

To prepare a model you need to use the Wizard that can be activated from

Tools\RuneHeads\Bones Zones\Wizard

First of all, you need to specify the starting model.



The starting model can be any simple 3d model or even your complex **Prefab** with all your game scripts. The wizard will not overwrite any information and add only the necessary information.

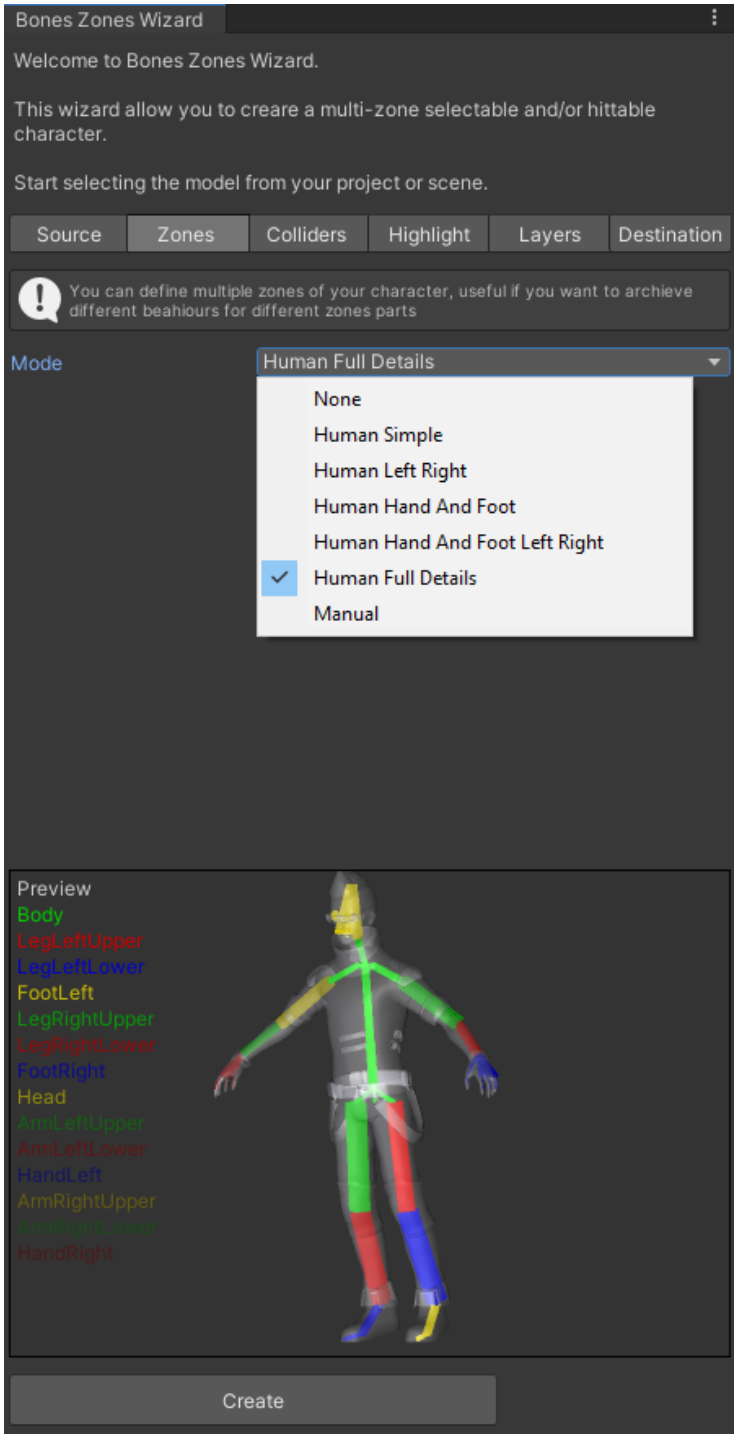
TIP: In general it is suitable to use the model with all the possible weapons/accessories that you want to insert, this will allow you to activate the automatic recognition of the wizard.

The model must have an Animator in its origin (root). The **avatar** is optional, but it is useful for humanoid characters to activate several automatisms.

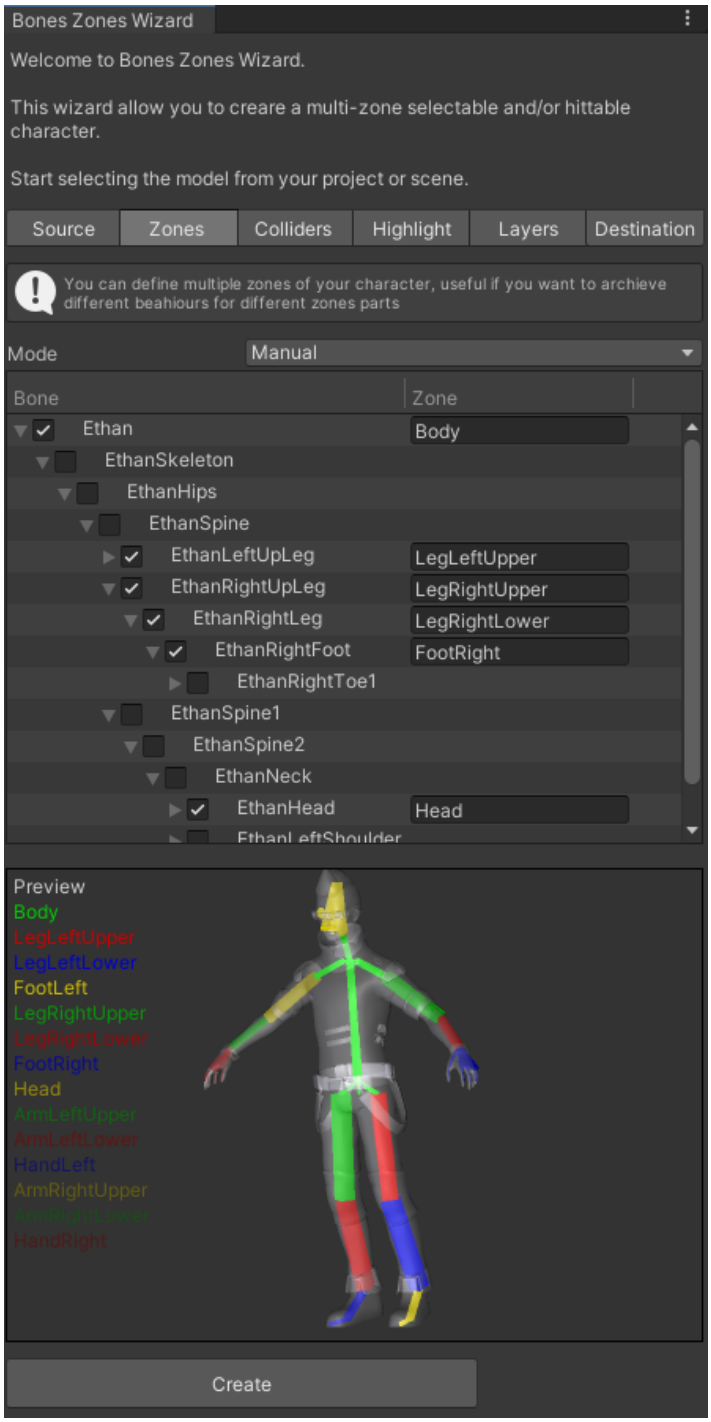
Once the model is selected, the Wizard activates the various tabs where specify all the details.

ZONES

Here you can indicate which zones to create for your model. In the case of humanoid characters there are several ready-made configurations:


Bones Zones Wizard
Welcome to Bones Zones Wizard.
This wizard allow you to creare a multi-zone selectable and/or hittable character.
Start selecting the model from your project or scene.
Source
Zones
Colliders
Highlight
Layers
Destination
You can define multiple zones of your character, useful if you want to archieve different beahiours for different zones parts
Mode
Human Full Details
None
Human Simple
Human Left Right
Human Hand And Foot
Human Hand And Foot Left Right
Human Full Details
Manual
Preview
Body
LegLeftUpper
LegLeftLower
FootLeft
LegRightUpper
LegRightLower
FootRight
Head
ArmLeftUpper
ArmLeftLower
HandLeft
ArmRightUpper
ArmRightLower
HandRight
Create

You can also switch to manual mode and work directly on the bones hierarchy:



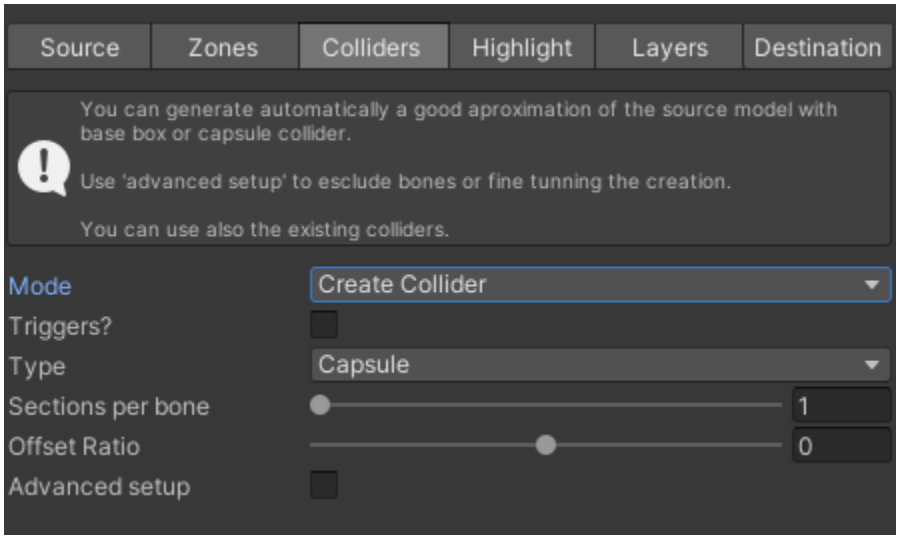
The preview image will help you understand quickly which zones you are creating.

COLLIDERS

The colliders tab allow you to define how the wizard will act about the colliders, the possible modes are:

Create Colliders

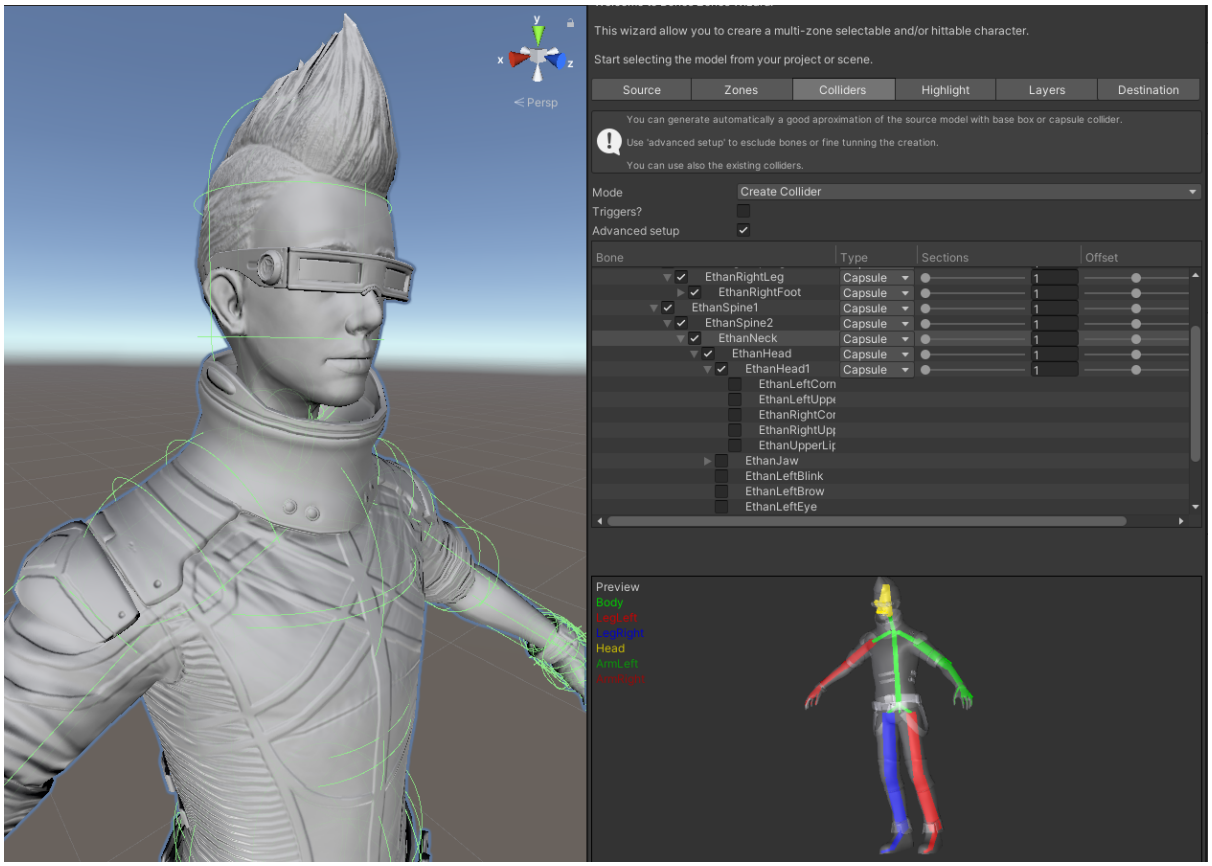
In this case, the wizard will create the colliders for you. The creation of colliders is based on an approximation of the model. This approximation will use Box collider or Capsule collider to improve performance.



Some parameters can help you create a better approximation:

Parameter	Description
Triggers?	This indicates whether the colliders will be market as Trigger
Type	Capsule or Box
Sections per bone	This indicates the number of segments per single bone. A number greater than one may allow for a better approximation. Some bones may generate unintended results if this parameter is too high. **The process involves creating N segments for the length of the bone, but if a section does not find enough vertices in the nearby model, it could create an incorrect or partial collider!**
Offset Ratio	This allows you to tighten or increase colliders around the model
Advanced Setup	Activates the advanced mode

Advanced Setup allows you to have more control per bone to further improve the model:



For example, like in the case in the picture, the creation of colliders for head detail elements is avoided by disabling the corresponding flag.

Use Existing Colliders

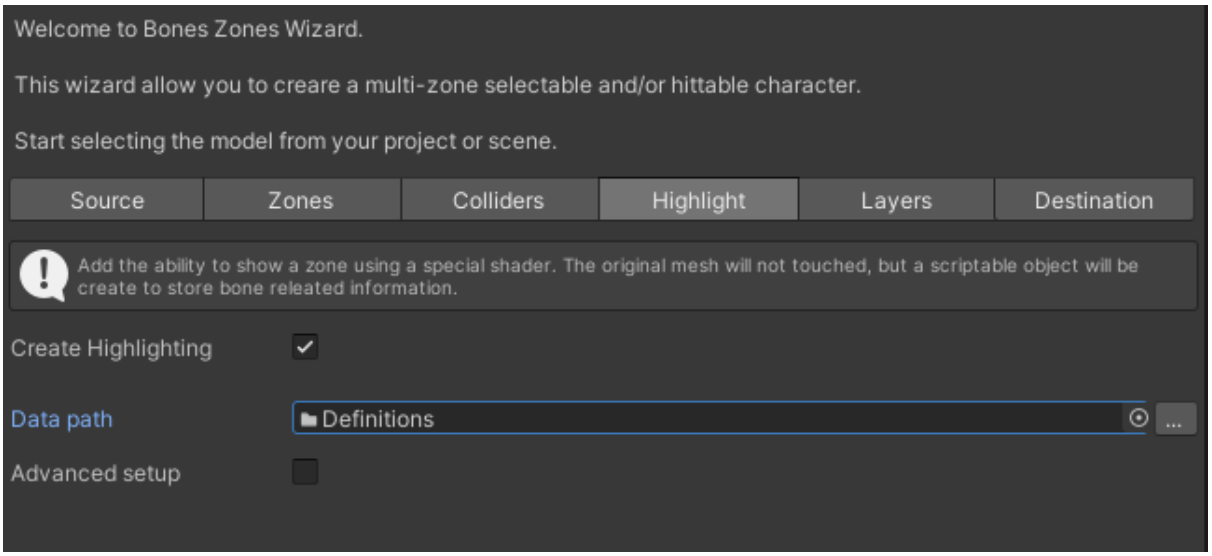
In this method, no new colliders are created. Existing colliders are identified and associated with the zone they belong to. This method requires your model to already be equipped with the appropriate colliders.

Do Nothing

This method allows you to ignore colliders altogether. The model cannot be used with RayCast functions (due to the absence of colliders) but it can still be highlighted in zones. This is convenient if you have a user interface (UI) that displays a highlighted model.

HIGHLIGHT

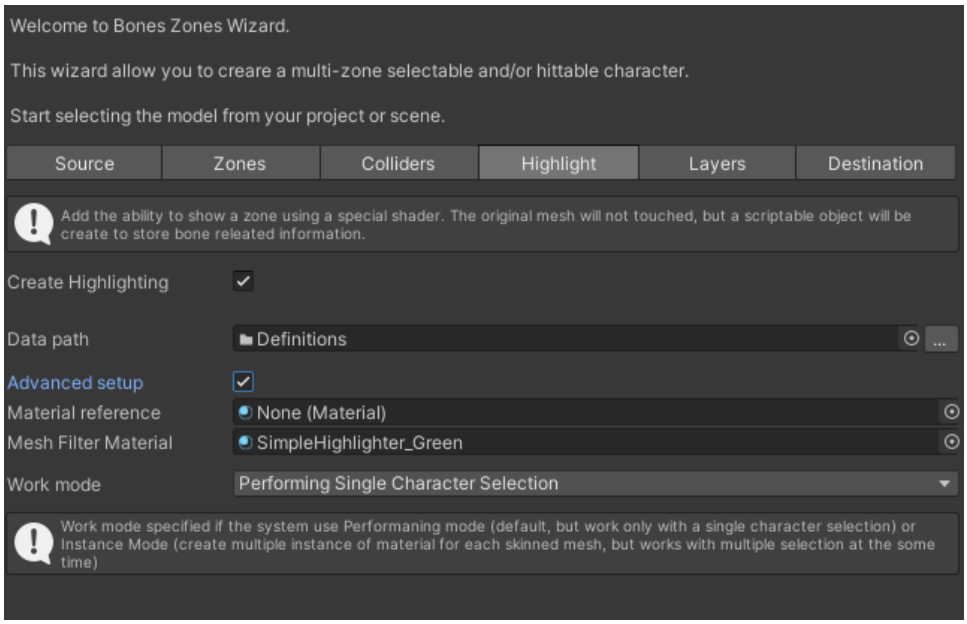
This tab allows you to specify whether to enable the highlighting feature.



The **"Data Path"** indicates where the information needed for the shader to function will be saved.

WARNING: When you replace an existing model, this information is also overwritten. If a model is NOT prefab linked to this information, it may lose its references. Pay attention to the name given in Destination to avoid overwriting existing objects. In any case, the wizard will warn you if there is a risk.

The advanced setup allows you to indicate the specific materials you want to use for lighting the model. Default materials will be used if not specified.

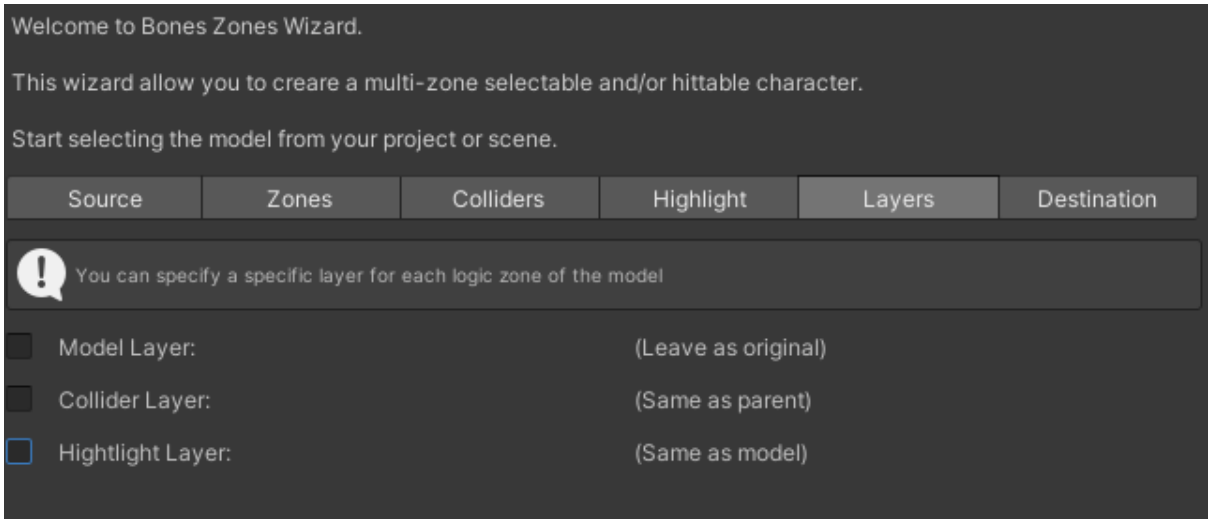


Parameter	Description
Material Reference (Optional)	Indicates the material used to light the model. If not specified, the default material is used. The given material must have the following shader **RuneHeads\BonesZonesHighlighter** Some examples can be found in **RuneHeads\BonesZones\Materials**
Mesh Filter Material	If the model has a boneless object in the hierarchy, it will be highlighted with the following: Some examples can be found in **RuneHeads\BonesZones\Materials**
Work Mode	Indicates the runtime mode of the highlighter, see next paragraph.

Work Mode allows you to specify the runtime mode of the highlighter:

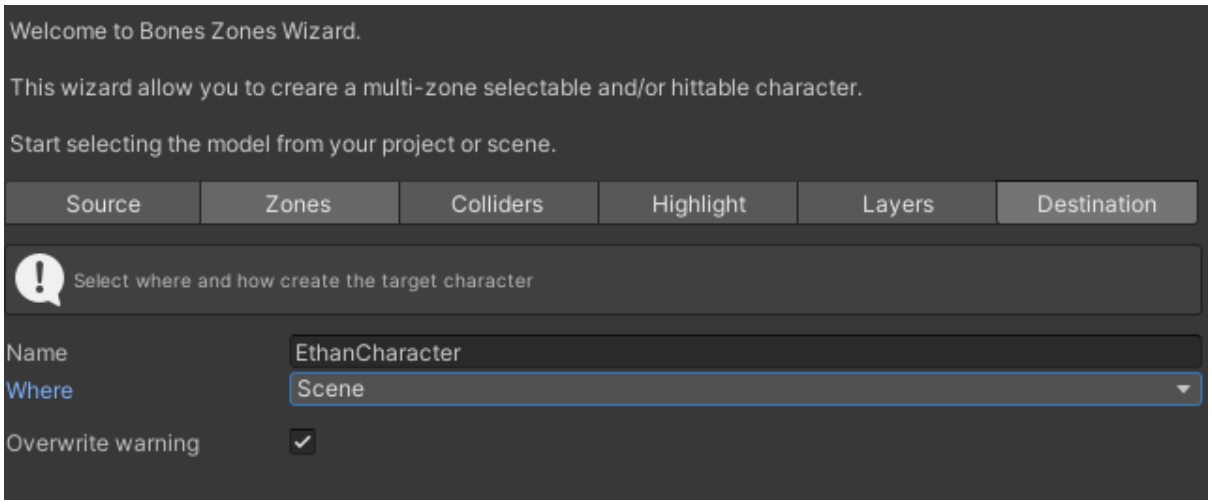
Parameter	Description
Performing Single Character Selection (default)	Only one character can be highlighted at a time in this mode. It allows higher performance because it does not duplicate material instances, but it is not possible to select two characters connected to the same definition differently.
Multiple Character Selection	In this mode, it is possible to highlight more than one character at a time, even if they are connected to the same definition (same prefab). It creates several instances of materials, so it generally less performant.

LAYER



This tab allows you to indicate which layer to associate with the different zones of the model. By default, the layers are left unaltered (the same as the original model), but it can be specified by simply checking the relative flag and indicating the desired layer.

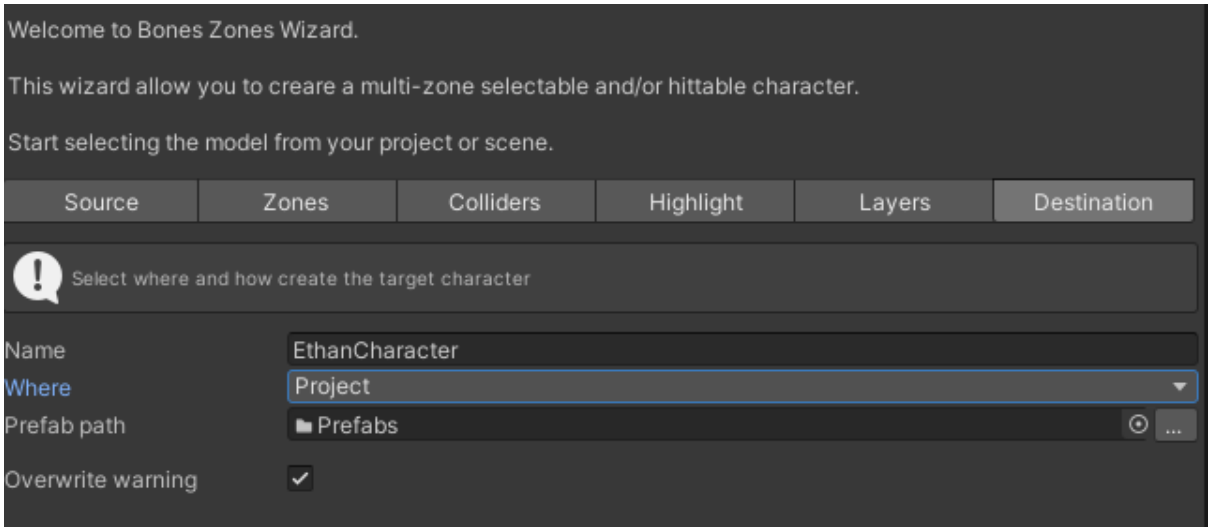
DESTINATION



In this tab, you can specify where to create the target model and its name.

Overwrite warning, turned on by default, will warn you if you are overwriting something existing. It can be turned off during the creation of a character so you can do several tests without intermediate confirmations (for example while finding the correct collider configuration).

If **Where** is set to **"Project"** you can indicate the destination folder. In the case of Project destination, a **Prefab** will be created.



USING MODELS

To use the functionality of the generated models, it is sufficient to write a few code lines. **For all the following examples, remember to always include the link to the library:**

```
using RuneHeads.BonesZones;
```

RaycastHit

In case your program runs Raycasts to find the point of impact, to retrieve the impacted zone simply call the GetZone method on the found hit.

```
RaycastHit hitInfo;
ZoneTrigger currentZone = null;
bool hitSomeThing = false;

if (Physics.Raycast(from, direction, out hitInfo, 50, layerMask))
{
     hitSomeThing = true;

     // Retrive the zone information
     currentZone = hitInfo.GetZone();
}
```

Collider

In case your program works with colliders generated by OnCollisionEnter or OnTriggerEnter, to recover the impacted zone just call the **GetZone** method on the found collider.

```
private void OnCollisionEnter(Collision collision)
{
    var zone = collision.collider.GetZone();
}
```

ZoneTrigger

The found zones will be the **ZoneTrigger** type. ZoneTrigger allows you to get some information and activate some functionality:

Field	**Description**
zonedCharacter	Returns the zone descriptor linked to the root model. It is useful to get to the root of the character
zoneName	Zone name
zoneId	Zone index (0..n-1)
Method	**Description**
Show()	Activates the highlighter for this zone
Hide()	Disable highlighter
ShowAll()	Enable highlighter on the whole model

ZoneCharacter

The root of the model will have this component that keeps track of all defined zones. It also has a link to the highlighter:

Field	**Description**
zonesName	Zone names (0..n-1)
highLighter	Possible character highlighter

HighLighter

Field	Description
skinnedMeshes	List of SkinnedMeshRender
materials	List of used materials
otherHighlighting	List of eventual other objects present in the hierarchy
workMode	Runtime mode (HIGHLIGHT)
Method	**Description**
Show(int zone)	Activates the highlighter for this zone (0..n-1)
Hide()	Disable highlighter
ShowAll()	Activates the highlighter on the whole model

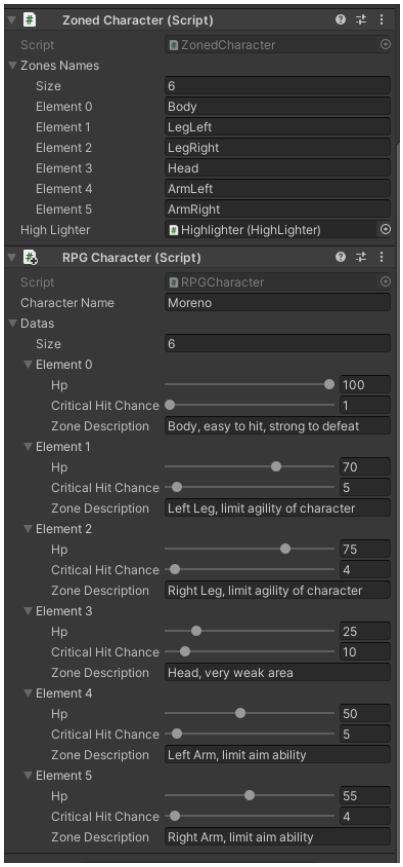
Add more information to the zones

To add more detailed information to zones you can proceed with this solution:

Create your component that contains the additional information. Create an array of this component that maps the zones defined in the ZoneCharacter to the character root, one by one.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 2 references
public class RPGCharacter : MonoBehaviour
{

    public string characterName;

    public ZoneRPGData[] datas;

}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[System.Serializable]
1 reference
public class ZoneRPGData
{
    [Range(10,100)]
    public int hp;

    [Range(1, 90)]
    public int criticalHitChance;

    public string zoneDescription;


}
```



Use the zone identifier to retrieve the additional information from the array.

```
//get custom script from character root
var rpg = currentZone.zonedCharacter.GetComponent<RPGCharacter>();

//get custom data from my custom script
var data = rpg.datas[currentZone.zoneId];
```

The sample code is in the **RuneHeads\BonesZones\Examples\UI folder**

Display the targeted enemy part

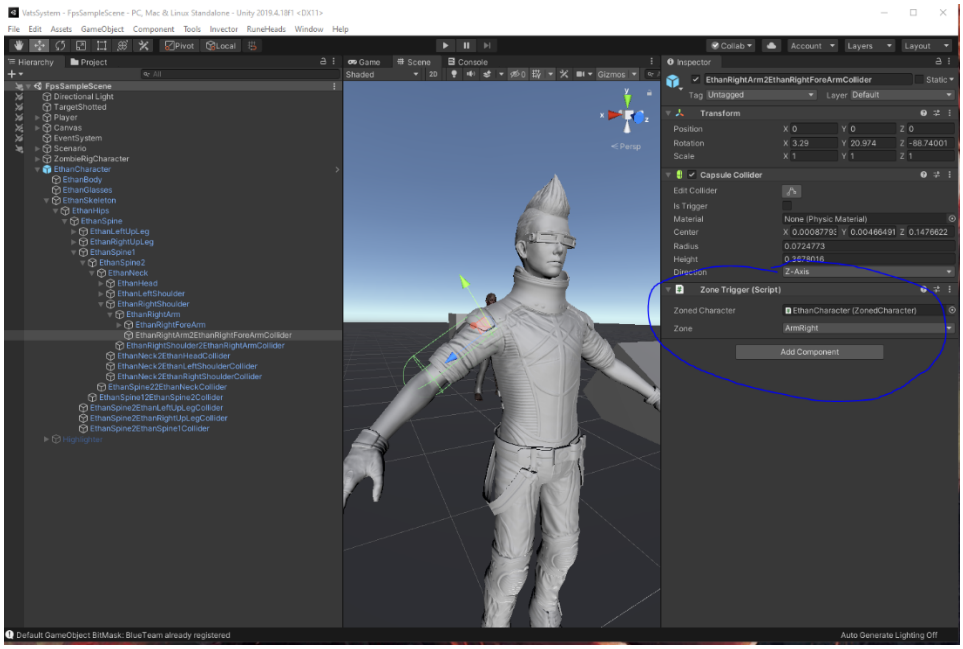
If you want to highlight the part of the enemy you are targeting, you can use the example available in the RuneHeads\BonesZones\FPS folder

In general, the strategy is to retrieve the current targeted area and then apply a code like this:

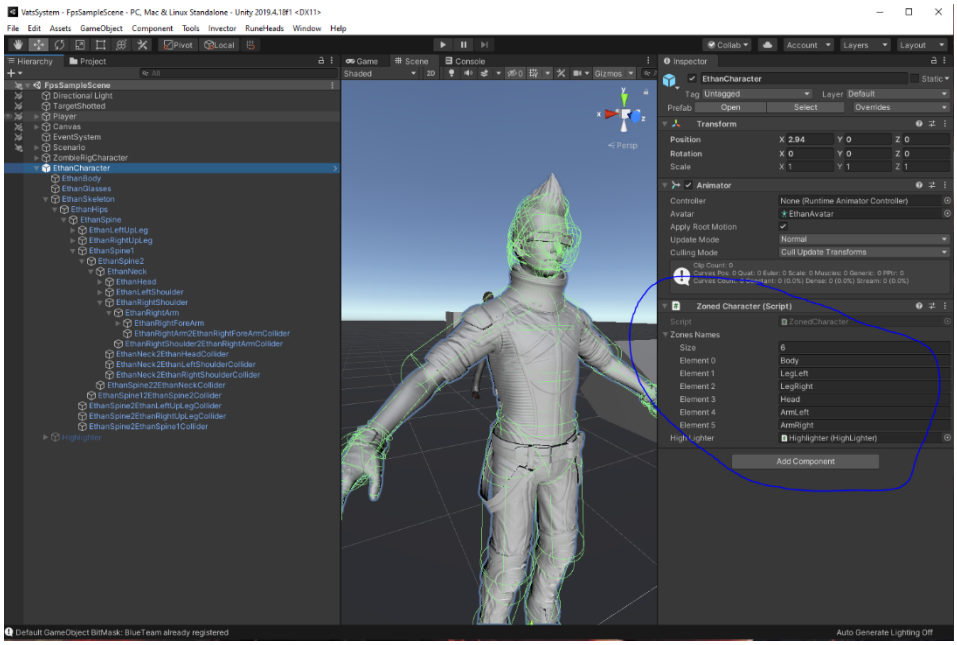
```
ZoneTrigger lastZone;

// Update is called once per frame
void Update()
{
       [...]
       var currentZone=[...]

       // Something change?
       if (lastZone != currentZone) UpdateZone(currentZone);
}

void UpdateZone(ZoneTrigger currentZone)
{
        // remove highlighting for previous object
        if (lastZone != null) lastZone.Hide();

        // highlight zone
        if (currentZone != null)currentZone.Show();

        lastZone = currentZone;
}
```

HOW IT WORKS

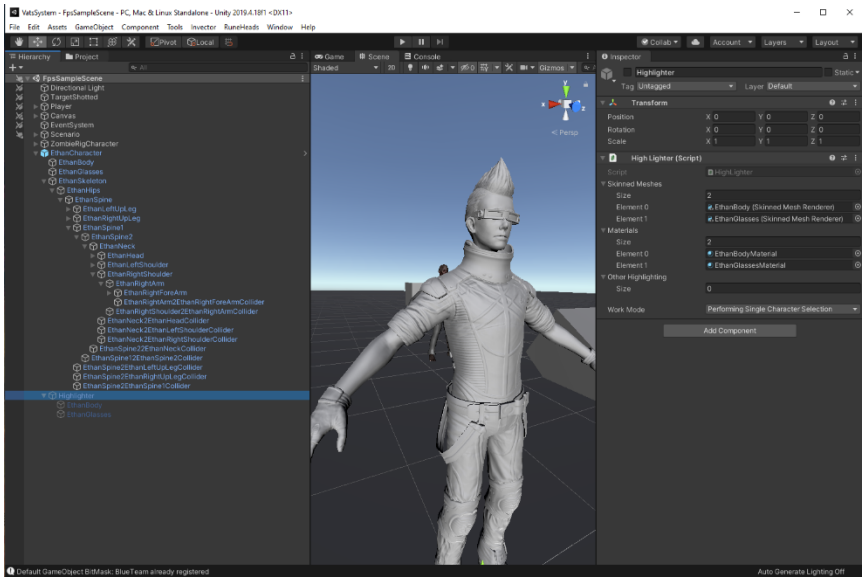
BONES ZONES uses the definition of zones associated with bones to map to colliders present in the model (or created specifically):



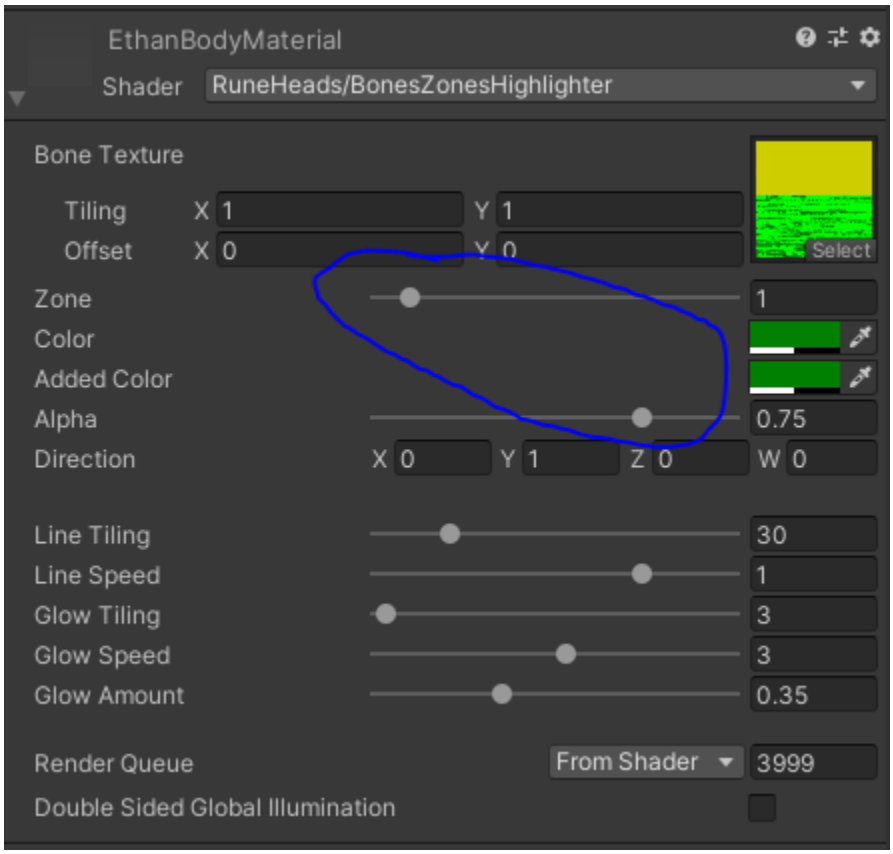
The character root contains the script that describes and collects all the zones of the character:



In this script the Highlighting system is also connected, it takes care of highlighting the object on-demand:



The highlighter uses a special shader and information saved in textures that allow to highlight either only one or all zones of a model:

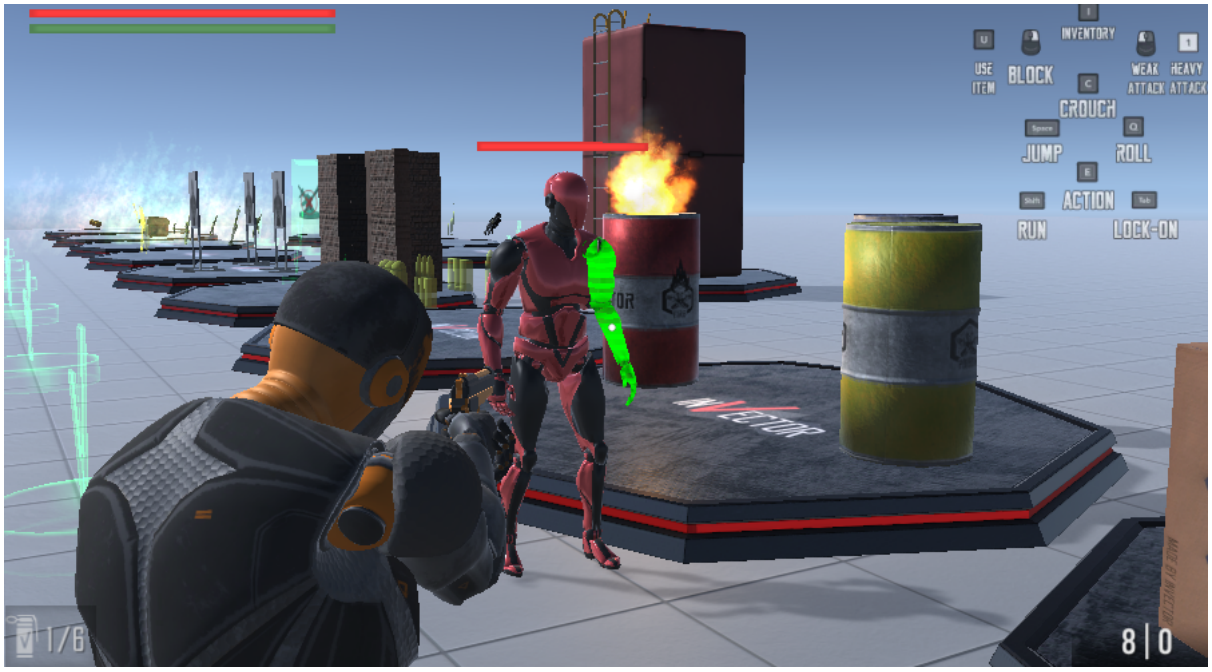


The highlighter also has a list of objects that are turned on or off in the hierarchy to highlight static models (MeshFilter) detected during creation.

INTEGRATIONS

Bones Zones can be easily integrated with tools that are already available on the market like Opsive Ultimate Character Controller and InVector.

InVector Shooter template



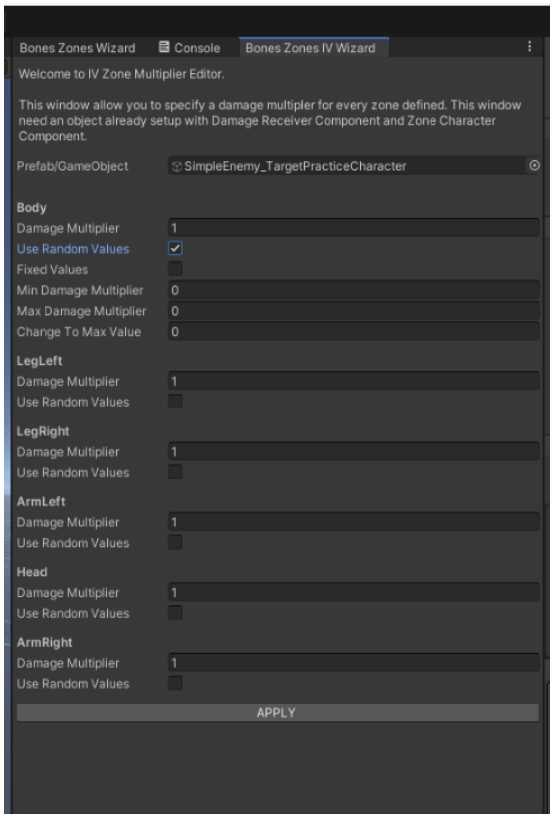
To activate this extension you need to import the RuneHeads\BonesZones\Extension\IV_Extension package.

WARNING: the InVector package must already be imported into the project otherwise you'll have compilation errors.

To integrate the characters of InVector with the zone system it is necessary to pass them through the **Model Preparation**. In this case, be careful to use existing colliders so as not to create duplicates.

Once the model is prepared, to access the zone and eventually highlight it we can use the methods described in **Use of Model**.

InVector allows you to define the damage received for each zone. Bones Zones provides a Wizard that allows you to quickly configure them, starting from the zone definition. Simply activate the wizard in the menu **Tools\RuneHeads\Bones Zones\IV Multiplier Editor** and enter the desired values:



To use the highlighter system or retrieve the hitting zone you can check the example available in **RuneHeads\BonesZones\IV\ShooterExample**

The technique remains the same, once you have recovered the **RaycastHit**, you simply have to recover the zone with the **GetZone** method.

Opsive UCC template

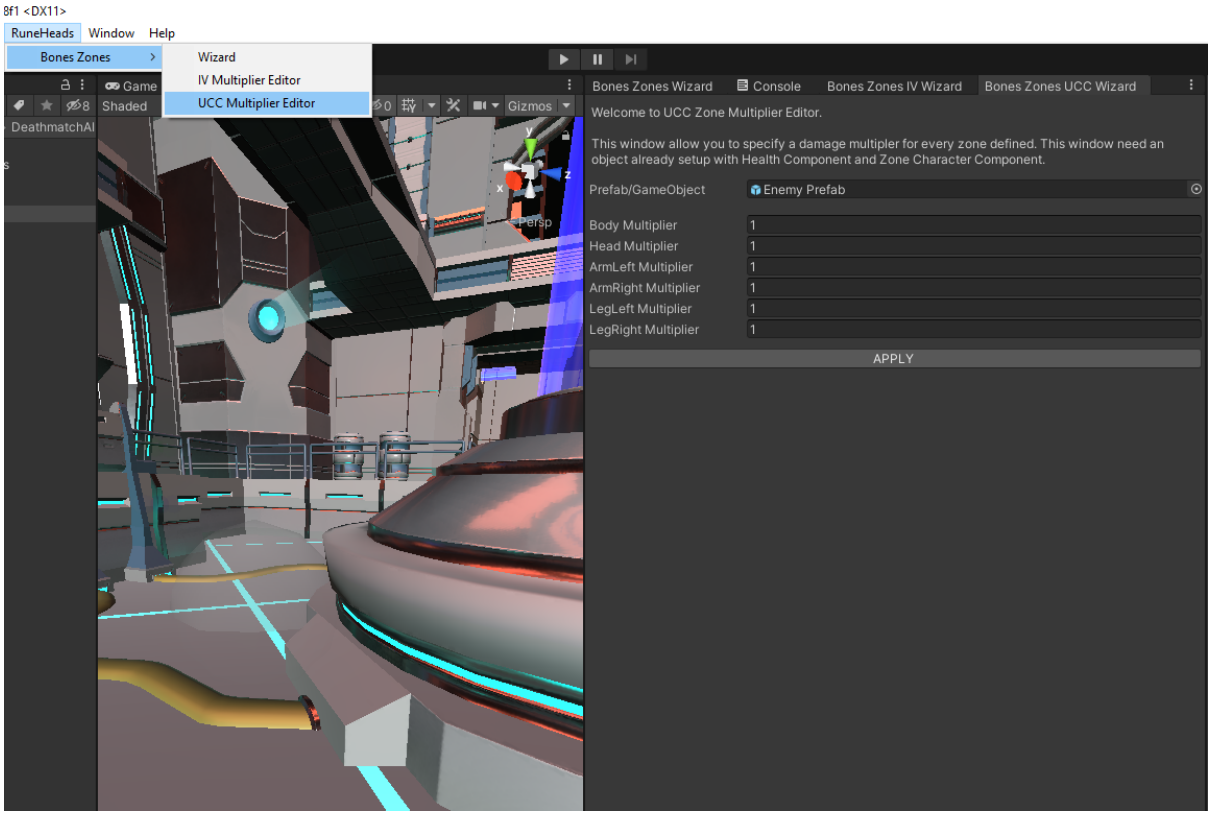
To activate this extension you need to import the RuneHeads\BonesZones\Extension\UCC_Extension package.

> ***WARNING: the UCC package must already be imported into the project otherwise there will be compilation errors.***

To integrate the UCC characters with the zone system it is necessary to run them through **Model Preparation**. In this chase, be careful to use existing colliders so as not to create duplicates.

Once the model is prepared, to access the zone and eventually highlight it we can use the methods described in **Use of Models**.

UCC allows you to define multipliers for the damage received for each zone. Bones Zones provides a Wizard that allows you to quickly configure them starting from the zone definition. Simply activate the wizard in the menu **Tools\RuneHeads\Bones Zones\UCC Multiplier Editor** and enter the desired values:



To use the Highlighter, use the **RaycastHit** obtained from the targeting system and retrieve the zone with the **GetZone** method.