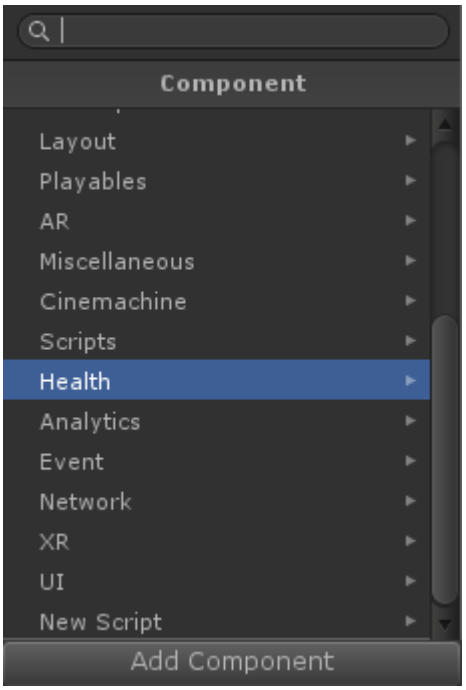
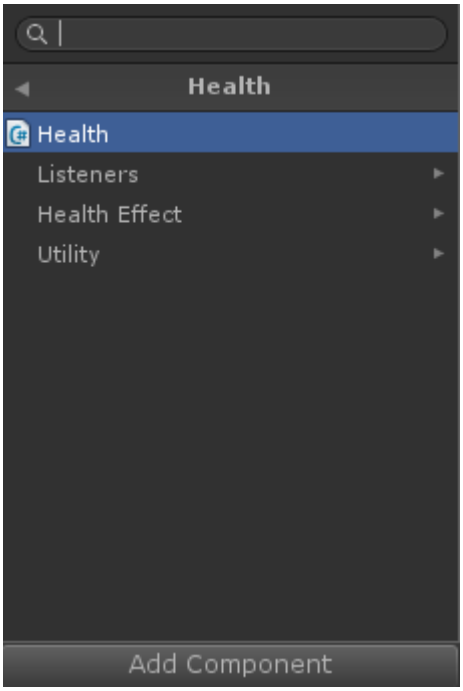
Advanced Health Component

Manual

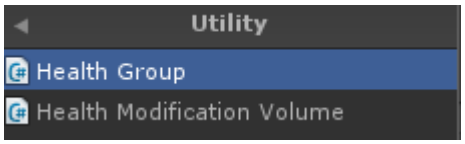
Overview

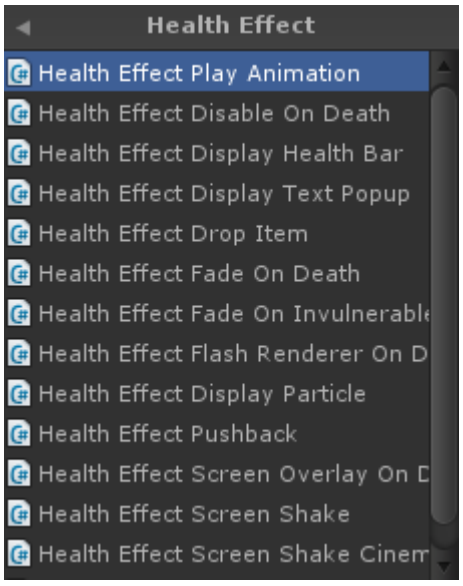
You can find all features within the Unity context menu.
All features require that you have a Health Component on a game object.



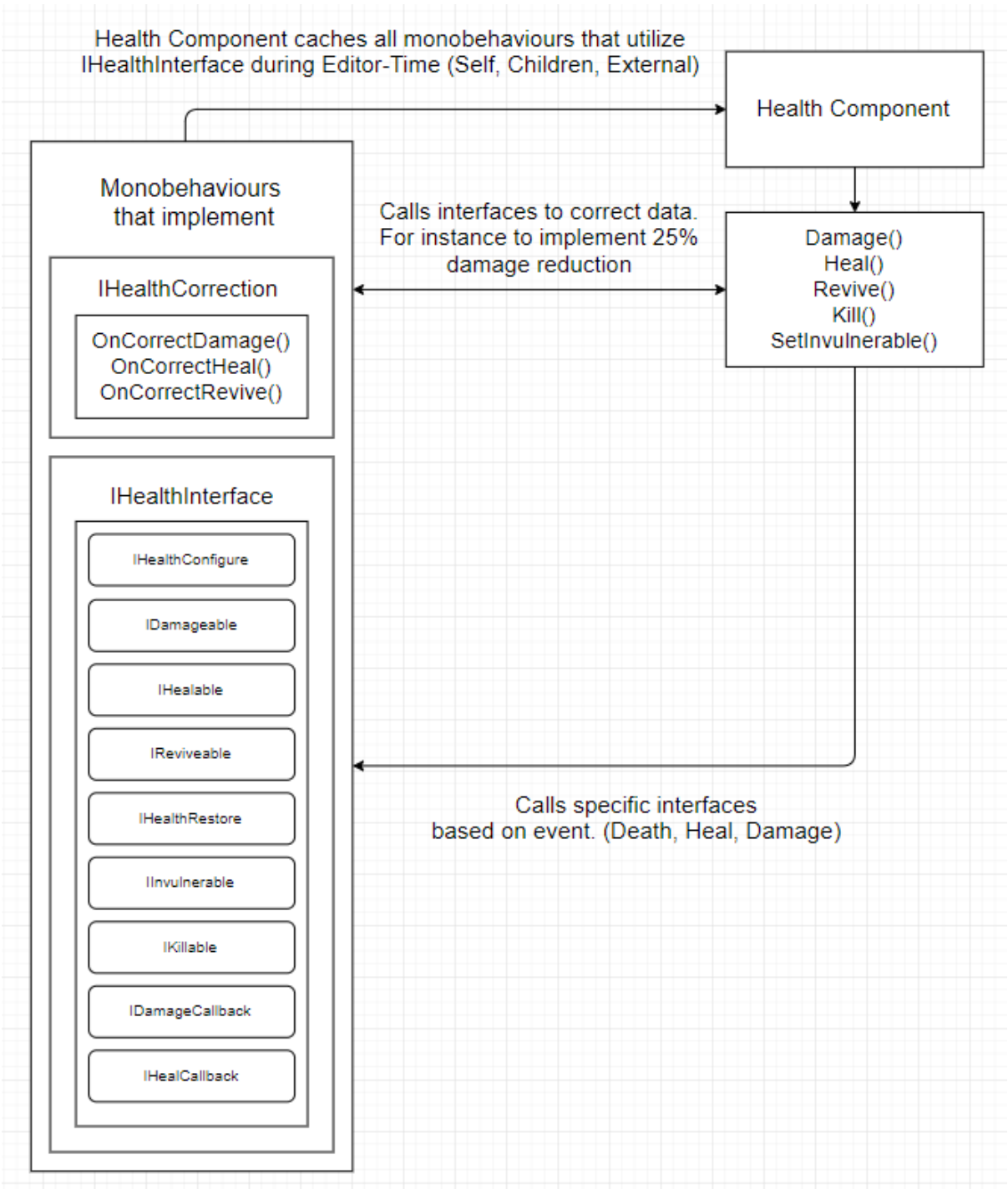


Listeners – These are Health Event Listeners that use Unity Events to communicate.
Health Effects – These are specific actions that happen based on Health Events
Utility – These are components that you can use to manipulate health.





How the Health Component works

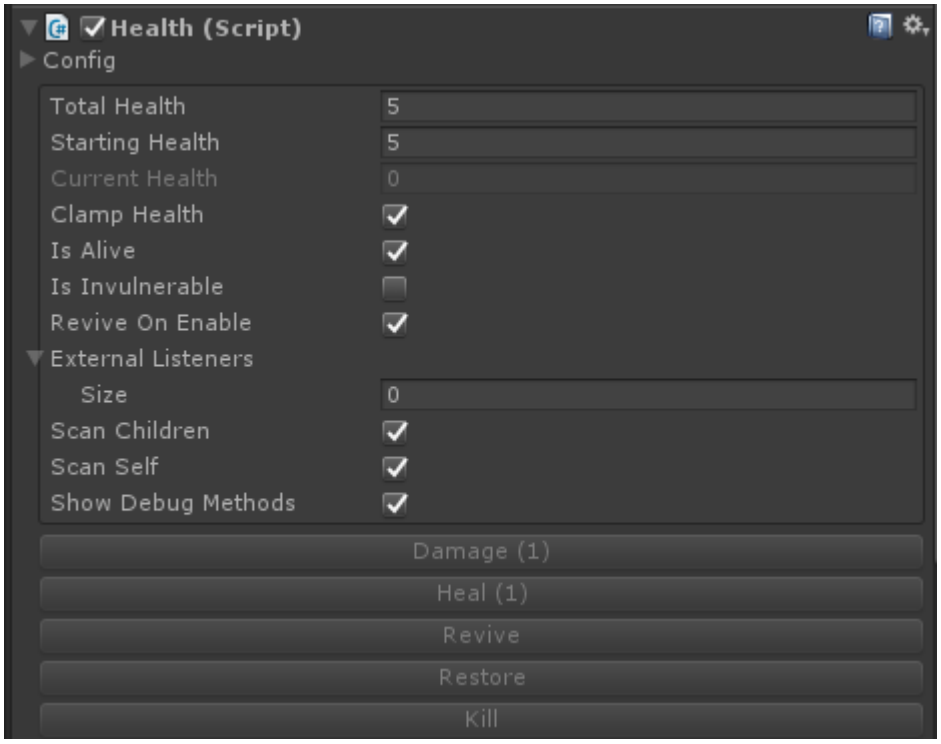


The way the Health Component handles dependencies is by obtaining interfaces.
As seen in the example above, it will search for any interface and stores it during edit time.

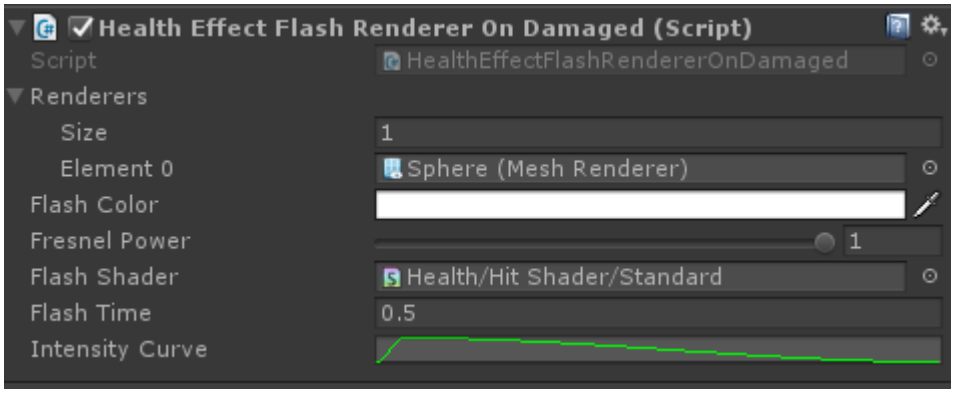
It is also possible to add them during runtime using the AddListener(IHealthInterface) method.
Which is done by implementing the

How to start: Compose your object

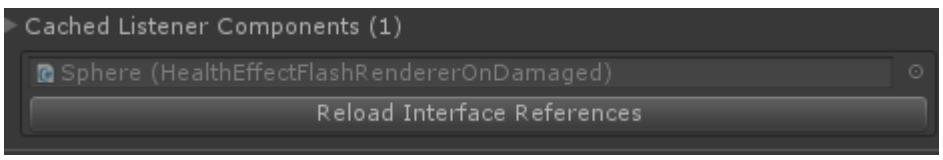
Add the health component to a game object



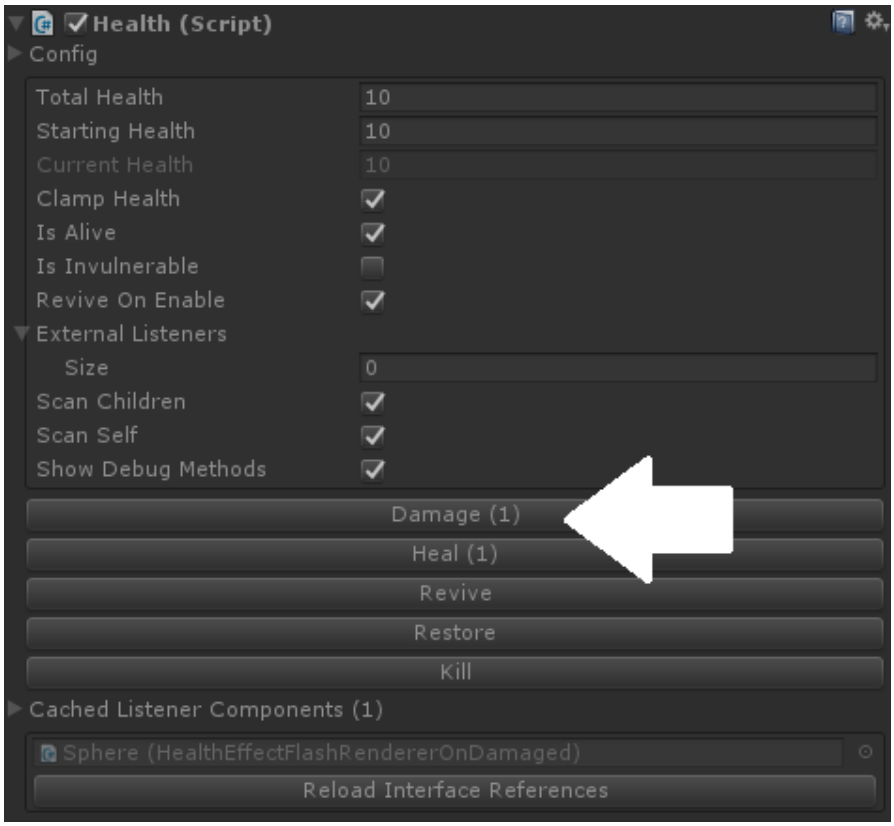
Add any effects by going to the Add Component menu shown above.
For this example I will add the Flash Renderer On Damaged effect.

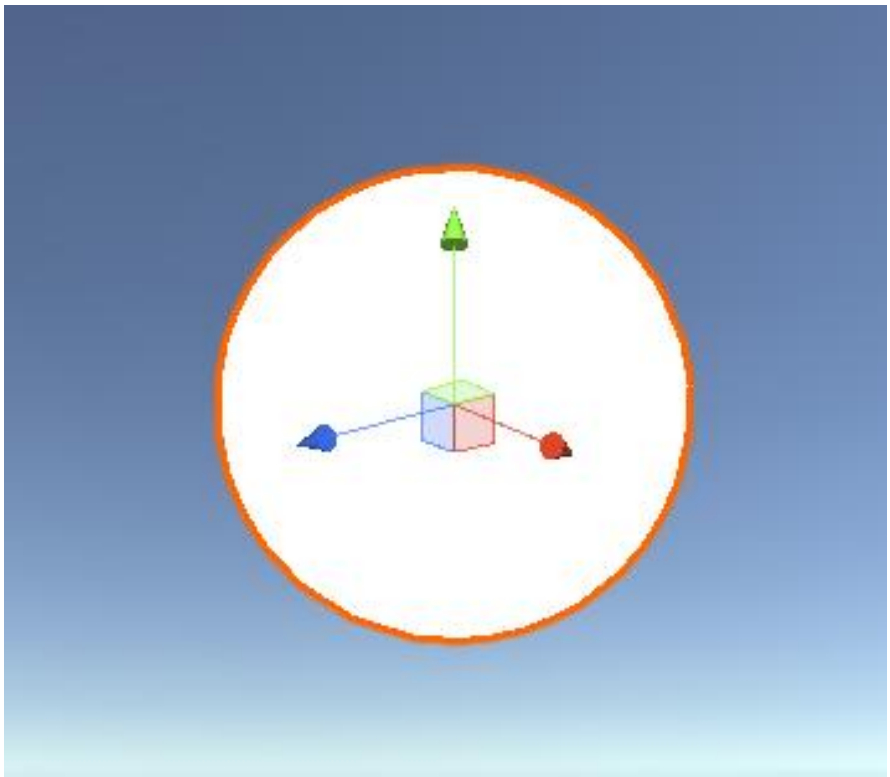


After adding the object, you will notice that the Health Component automatically fills in the references. In case it doesn't work, you can try to call a reload manually by pressing on the button.



During play mode you are able to test the effects by using the debug methods on the Health component. In this case we press the damage button and you will see the effect update.
Later in this document I will explain how to create a custom component that listens for these events.





Applying effects for gameplay - Ray Casts

Doing damage from the mouse position

```
void Update()
{
    if (Input.GetMouseButton(0))
    {
        if (cooldown > 0)
            return;

        RaycastHit hit;

        Ray screenRay = camera.ScreenPointToRay(Input.mousePosition);

        if (Physics.Raycast(screenRay, out hit, 150))
        {
            Health getStats = hit.transform.GetComponent<Health>();

            if (getStats != null)
            {
                getStats.Damage(1, hit.point, stats);
            }
        }

        cooldown = shootCooldown;
    }

    cooldown -= Time.deltaTime;
}
```

Additional information: https://docs.unity3d.com/Manual/CameraRays.html

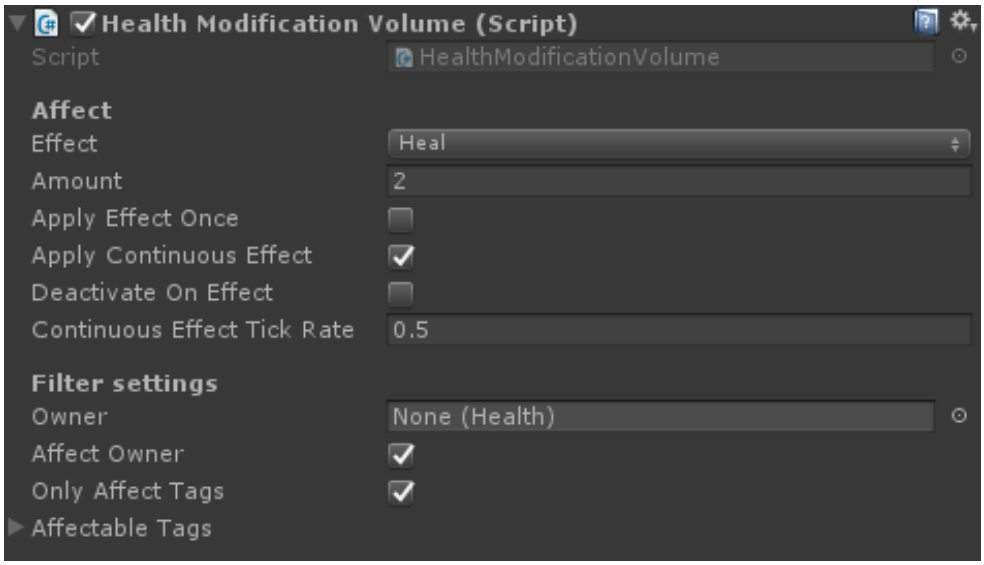
Doing damage with sphere casts, in front of a character

```
void Update()
{
    if (Input.GetKeyDown(KeyCode.Space))
    {
        RaycastHit hit;

        Vector3 origin = transform.position + charController.center;
        Vector3 direction = transform.forward;

        float radius = charController.height * 0.5f;
        float distance = 1;

        if (Physics.SphereCast(origin, radius, direction, out hit, distance))
        {
            Health getStats = hit.transform.GetComponent<Health>();

            if (getStats != null)
            {
                getStats.Damage(1, hit.point, stats);
            }
        }
    }
}
```

Additional Information: https://docs.unity3d.com/ScriptReference/Physics.SphereCast.html

Be sure to implement the Lowscope.HealthPro namespace.

Applying effects for gameplay - Health Modification Volume



You can find examples on usage of the Health Modification Volume in the “Group And Volumes” example. All fields above had tooltips attached to explain the functionality.

How to use the volume

Have it attached to a bullet prefab with a component that moves the object
and call configure. As the example below, you can access the HealthModificationVolumeBase class. And call configure with the struct below as input.

```
bullet.transform.position = this.transform.position;
bullet.Configure(shootDirection.normalized, 15);

bullet.GetComponent<HealthModificationVolumeBase>()
.Configure(new ModificationVolumeConfiguration(EHealthModificationEffect.Damage, 1, 1, health, true))

bullet.gameObject.SetActive(true);
```

```
public struct ModificationVolumeConfiguration
{
        public float Amount;
        public bool AutoDeactivate;
        public float DeactivationTime;
        public string[] TargetTags;
        public int Owner;
        public bool DeactivateOnEffect;
        public EHealthModificationEffect Effect;
```

Note
The Health Modification Volume requires a collider to function.
Ensure that you have an owner set If you want to utilize the IDamageCallback and IHealCallback

Applying effects for gameplay – Unity Event Support

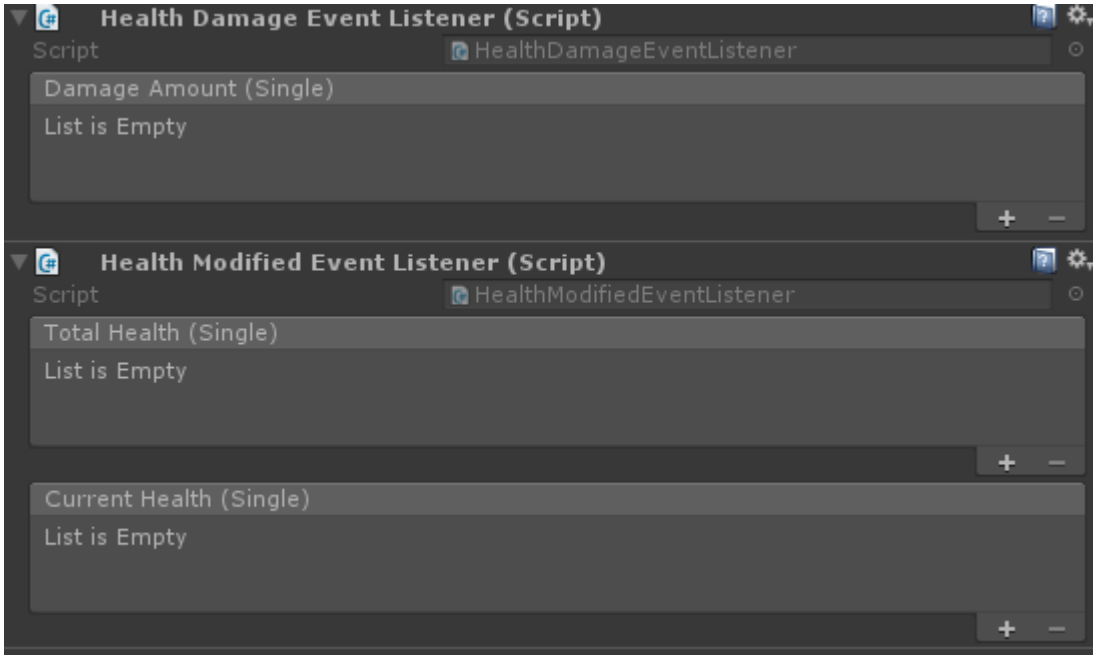
You are able to apply some of the effects to the health using UnityEvents, however this is limited. Mainly due to the shortage of parameters, this means that a damage callback would not be possible.

In case you need extra things exposed with UnityEvents, please send me a message. You can find the contact information below.

Event Listeners

The plugin contains a set of Event Listeners. They are also developed by utilizing the interfaces. Meaning that it is easy to extend or modify the functionality into new components if needed.

Example of the Health Damage Event Listener



It is recommended to place the listeners into a separate object, because they will clutter up the inspector very quickly.

Some use cases for using these listeners:
- Modifying specific objects when an entity dies or gets hit, such as opening a door
- Updating UI elements such as health sliders.

Note
Event listeners can also be used on Health Groups, because they inherit from the Health Component.

Health Correction

Depending on the type of project, you might want to modify the incoming data. Such as getting damage or getting healed. Say for example, you want to have a buff that lowers your damage by 25%

How the interface looks like, comments have been removed to make space within this document.

```
public interface IHealthCorrection : IHealthInterface
{
    int OnCorrectionOrder(int ownerIndex);

    HealthCorrectionData OnCorrectDamage(float amount, int attackerIndex);
    HealthCorrectionData OnCorrectHeal(float amount, int attackerIndex);
    HealthCorrectionData OnCorrectRevive(float percentage, int attackerIndex);
}
```

You can implement this interface in any component that is responsible for correcting the damage. For instance in a character stats component.

OnCorrectionOrder is used to define what gets checked first. Starting from low to high.
The other methods provide you with the given amount that has been affected, and the index of the attacker. In order to utilize the attacker index you can call Health.GetInstance(attackerIndex)

HealthCorrectionData is the return type, which contains a additional field called stopExecution. This can be used to ensure that further execution of the effect stops.

```
public struct HealthCorrectionData
{
    public bool stopExecution;
    public float amount;

    public HealthCorrectionData(bool stopExecution, float amount)
    {
        this.stopExecution = stopExecution;
        this.amount = amount;
    }
}
```

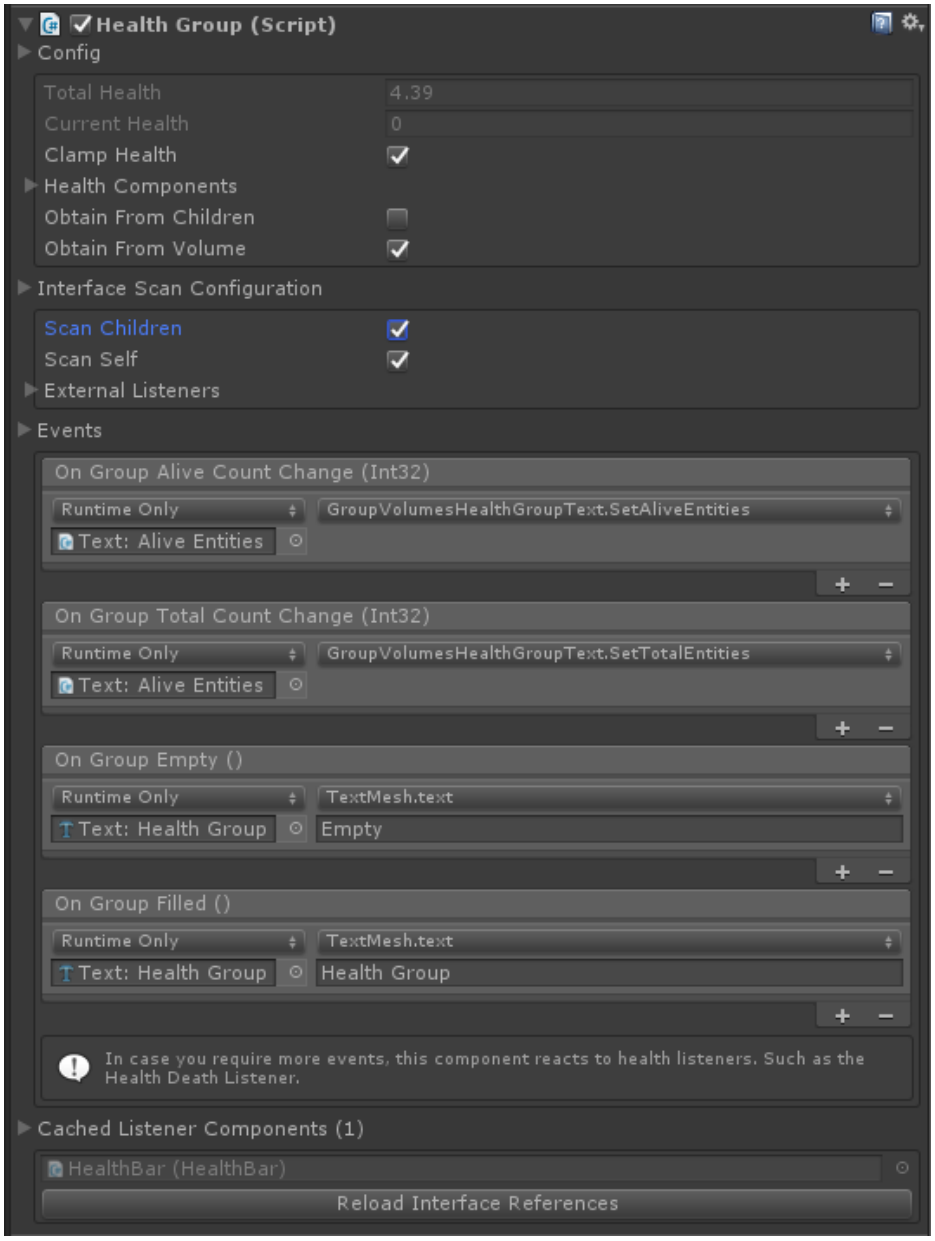
Callbacks

Sometimes you would want a callback if damage has occured. The IDamageCallback and IHealCallback are made for this. These events are invoked on the attacker once Damage() or Heal() methods are called on the Health component of the target.

An example of usage would be that you want to add score when an enemy has been killed.
The Shooter Example has code that does exactly this. (Stripped out unrelated parts)

```
public class ShooterPlayer : MonoBehaviour, IDamageCallback
{
        public void OnDamageDone(Health target, HealthInfo info)
        {
            // We verify if the target has died
            if (info.CurrentHealth <= 0)
            {
                currentScore++;
                scoreText.text = currentScore.ToString();
            }
        }
}
```

Health Groups



The Health Group class inherits from the Health class. This means they can notify any of the effects and listeners mentioned above.

Contact and support

In case you have any issues or need support. I will be glad to help!
Suggestions for the creation of

Email: info@low-scope.com