8. Runtime Variables

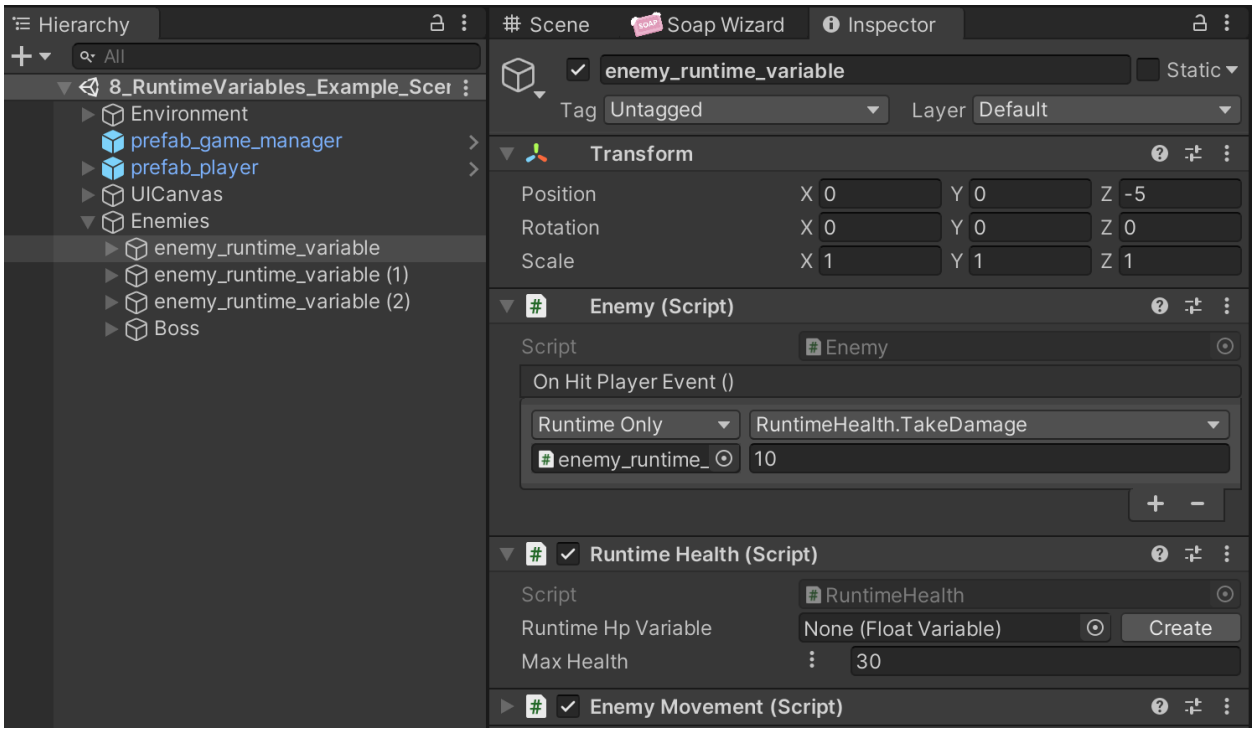
Table of Content

Set up

In this example, I have multiple enemies, each with a health bar. The health and health bar of enemies are generic components that rely on a Scriptable Variable float (Health) for their logic.

When you play the game, each time an enemy hits the player, the enemy will lose health. You can see their HP bar updating. When an enemy dies, it will disable itself.

Let's select an enemy to understand how this works. Go to the scene hierarchy and click on an enemy.



The enemy script is very simple; it detects collisions and calls a method on the RuntimeHealth.cs component. This component is the one we are interested in. You will notice that the "Runtime HP variable" field is left empty! Now, if you open the script RuntimeHealth.cs, you can see that in Start(), we create a runtime variable if the field is null.

```
private void Start()
{
    //Create the runtime variable instance. It's better to do it in Start because of the execution order:
    //Awake-> SceneLoaded (values are reset) -> Start
    if (_runtimeHpVariable == null)
        _runtimeHpVariable = CreateRuntimeVariable<FloatVariable>(name: $"{gameObject.name}_Hp");

    _runtimeHpVariable.Value = _maxHealth.Value;
    _runtimeHpVariable.OnValueChanged += OnHealthChanged;

    //Initialize the health bar only after the variable has been properly set.
    //You can use events to decouple components if your health bar is in another scene (UI scene for example).
    //In this case, as it's a local Health bar, a direct reference is fine.
    GetComponentInChildren<HealthBarSprite>().Init(_runtimeHpVariable);
}
```

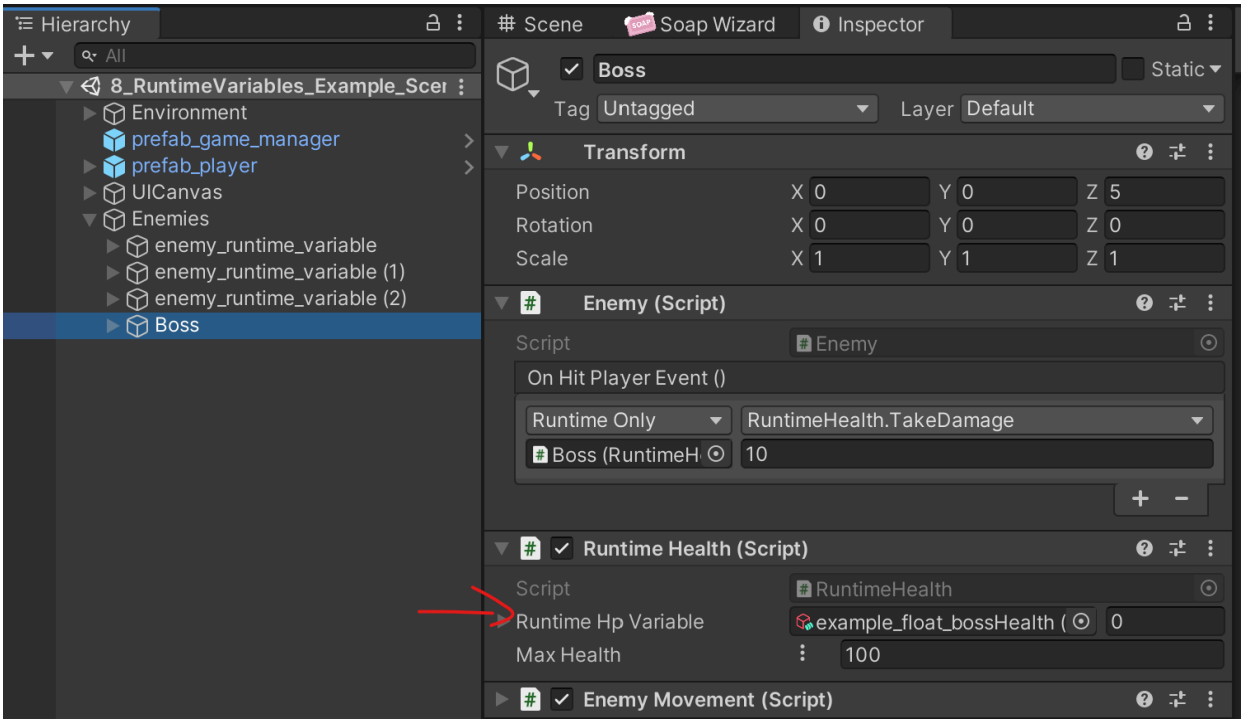
```
private T CreateRuntimeVariable<T>(string name) where T : ScriptableVariableBase
{
    var runtimeVariable = ScriptableObject.CreateInstance<T>();
    runtimeVariable.name = name;
    return runtimeVariable;
}
```

Once the variable instance is created, we pass it to the health bar. Because the HealthBarSprite is a child of the 'prefab', we can safely use a direct method call. However, if the Health bar were in 2D on a UI Canvas (maybe in another scene), it would be better to use events to decouple these systems.

The execution flow is extremely important when using runtime variables. You will notice that I am not relying on Start() for the Health bar component. Doing so might cause a race condition between the Start() of the RuntimeHealth (which creates the instance) and the Start() of the Health Bar that registers to it.

Flexibility

Now, let's go back to the scene hierarchy and select the Boss.



You notice that the Boss's RuntimeHealth component has an existing Scriptable Variable in its HP field. For most enemies, we leave the field empty, and HP variables will be created at runtime. However, for the Boss, we can use the same components and logic but override them with a variable that exists in the project. By doing so, we can connect the Boss's HP variable to other elements of our game (quests, trigger events, save system, etc.).

We could also add a field for Scriptable Events like OnDeath, which would be empty for regular enemies but filled for the Boss (like our HP variable).

I hope that with this example, you can grasp the essence of runtime variables. It is not a traditional pattern, but it can work in some games. I'll let you experiment with it and see if you enjoy it.