Events System: Usage

Event creation

To get started with the events system you'll need to first create an event. It's recommended to use a static class or make the event itself static where possible for ease of access. To define an event, write something like the example below:

```
public static readonly Evt OnGameOver = new Evt();
```

Note that the Evt class is a class for you'll need to make an instance of it before using it, this is the only main difference between this setup and the normal system action flow. Making the event read-only just helps us not edit it accidentally and defining it as a new event is just cleaner as it won't need to be initialized anywhere else before use.

For events with parameters there are 8 extra classes in the `Evt.cs` file that allow for it. If you need more parameters, you can make additional ones as they all follow the same setup, just with an extra generic field to use in the class. To define a parameter event, use something like the following:

```
public static readonly Evt<int> OnMoneyCollected = new Evt<int>();
```

Event subscribing

Subscribing to events is as easy as it is to do with actions. Instead of using `+=` or `-=` to subscribe, the events system has 2 simple methods, being `Add()` & `Remove()`. Note that any method subscribing to an event will need to have the same number of parameters of the same type to subscribe correctly. Some examples:

```
private void OnEnable()
{
    OnGameOver.Add(MyVoidMethod);
}


private void OnDisable()
{
    OnGameOver.Remove(MyVoidMethod);
}


private void MyVoidMethod()
```

```
{
    // Some code here...
}
```

```
private void OnEnable()
{
    OnMoneyCollect.Add(MyIntMethod);
}


private void OnDisable()
{
    OnMoneyCollected.Remove(MyIntMethod);
}


private void MyIntMethod(int amount)
{
    // Some Code Here...
}
```

Anonymous subscriptions

You can also subscribe to events without matching the parameters required using the anonymous setup. This setup has the same add and remove methods but taking in a string key for the anonymous subscription to be identified as. On removal we just pass in the key we assigned on creation. Some examples:

```
private void OnEnable()
{
    OnGameOver.AddAnonymous("MySub", () => MyIntMethod(100));
}


private void OnDisable()
{
    OnGameOver.RemoveAnonymous("MySub");
}


private void MyIntMethod(int amount)
{
    // Some code here...
}
```

Event raising/invoking

Raising events is essentially invoking the action. It is done by just calling the `Raise()` method on the event you want to call. You pass in any params the event requires in the call to pass them to all listeners.

```
private void OnSomeGameStateChange()
{
    OnGameOver.Raise();
    OnMoneyCollected.Raise(money);
}
```