Delayed Events: Scripting Api

Assembly

```
CarterGames.Cart.Modules
```

Namespace

```
CarterGames.Cart.Modules.DelayedEvents
```

Definition

```
private readonly DelayedEvt MyEvent = new DelayedEvt();
```

As DelayedEvt is just like the Evt class, you'll need to create an instance of it for use, otherwise it'll return null and do nothing.

Api

Add

Adds an listener to the delayed evt instance. Note that it must have matching parameters to correctly subscribe.

```
public void Add(Action listener);
public void Add(Action<T> listener);
public void Add(Action<T1,T2> listener);
public void Add(Action<T1,T2,T3> listener);
public void Add(Action<T1,T2,T3,T4> listener);
public void Add(Action<T1,T2,T3,T4,T5> listener);
public void Add(Action<T1,T2,T3,T4,T5,T6> listener);
public void Add(Action<T1,T2,T3,T4,T5,T6,T7> listener);
public void Add(Action<T1,T2,T3,T4,T5,T6,T7,T8> listener);
```

```
DelayedEvt MyEvent = new DelayedEvt();
```

```
private void OnEnable()
{
    MyEvent.Add(MyMethod);
}

private void MyMethod()
{
    // My logic here...
}
```

AddAnonymous

Adds an anonymous listener to the delayed evt instance. The parameters do not need to match when adding anonymous actions to run on an event. Though it is still advised to match them where possible.

```
public void AddAnonymous(string id, Action listener);
public void AddAnonymous(string id, Action<T> listener);
public void AddAnonymous(string id, Action<T1,T2> listener);
public void AddAnonymous(string id, Action<T1,T2,T3> listener);
public void AddAnonymous(string id, Action<T1,T2,T3,T4> listener);
public void AddAnonymous(string id, Action<T1,T2,T3,T4,T5> listener);
public void AddAnonymous(string id, Action<T1,T2,T3,T4,T5,T6> listener);
public void AddAnonymous(string id, Action<T1,T2,T3,T4,T5,T6,T7> listener);
public void AddAnonymous(string id, Action<T1,T2,T3,T4,T5,T6,T7,T8> listener);
```

```
DelayedEvt MyEvent = new DelayedEvt();

private void OnEnable()
{
    MyEvent.AddAnonymous("MyMethod", MyMisMatchedMethod(100));
}

private void MyMisMatchedMethod(int health)
{
    // My logic here...
}
```

Remove

Removes a listener to the delayed evt instance. Note that it must have matching parameters to correctly unsubscribe.

```
public void Remove(Action listener);
public void Remove(Action<T> listener);
```

```
public void Remove(Action<T1,T2> listener);
public void Remove(Action<T1,T2,T3> listener);
public void Remove(Action<T1,T2,T3,T4> listener);
public void Remove(Action<T1,T2,T3,T4,T5> listener);
public void Remove(Action<T1,T2,T3,T4,T5,T6> listener);
public void Remove(Action<T1,T2,T3,T4,T5,T6,T7> listener);
public void Remove(Action<T1,T2,T3,T4,T5,T6,T7,T8> listener);
```

```
DelayedEvt MyEvent = new DelayedEvt();

private void OnEnable()
{
    MyEvent.Remove(MyMethod);
}

private void MyMethod()
{
    // My logic here...
}
```

RemoveAnonymous

Removes an anonymous listener to the delayed evt instance. You only need to pass the key you assigned to the anonymous event listener to remove it.

```
public void RemoveAnonymous(string id);
```

```
DelayedEvt MyEvent = new DelayedEvt();

private void OnEnable()
{
    MyEvent.RemoveAnonymous("MyMethod");
}
```

Raise

Raises/Invokes the delayed evt which will run all the listeners currently subscribed to it. If your event takes parameters you'll need to pass through their values here.

```
public void Raise(float delay);
public void Raise(float delay, T param);
public void Raise(float delay, T1 param1, T2 param2);
public void Raise(float delay, T1 param1, T2 param2, T3 param3);
```

```
public void Raise(float delay, T1 param1, T2 param2, T3 param3, T4 param4);
public void Raise(float delay, T1 param1, T2 param2, T3 param3, T4 param4, T5 para
public void Raise(float delay, T1 param1, T2 param2, T3 param3, T4 param4, T5 para
public void Raise(float delay, T1 param1, T2 param2, T3 param3, T4 param4, T5 para
public void Raise(float delay, T1 param1, T2 param2, T3 param3, T4 param4, T5 para
```

```
DelayedEvt MyEvent = new DelayedEvt();
DelayedEvt<int> MyIntEvent = new DelayedEvt<int>();

private void OnEnable()
{
  // Raises the event after 2.5 seconds.
    MyEvent.Raise(2.5f);

    // Raises the event after 2.5 seconds with the parameter for the event.
    MyIntEvent.Raise(2.5f, 100);
}
```

Clear

Clears all the listeners from this delayed evt.

```
public void Clear();
```

```
DelayedEvt MyEvent = new DelayedEvt();

private void OnEnable()
{
    MyEvent.Clear();
}
```