Patterns

How to implement certain design patterns from C# with U#

- Observer Pattern

Observer Pattern

> *Since we don't have access to delegates, how can we effectively implement an observer pattern in UdonSharp?*

I currently have 2 stable projects that have an observer pattern in them:

VUdon - EventDispatcher

VUdon - EventDispatcher

EventDispatcher is a "*singleton*" prefab for dispatching update events *(FixedUpdate, Update, LateUpdate and PostLateUpdate).*

You add and remove handlers *(a.k.a. "subscribers")* by calling the following public API methods:

```
[PublicAPI]
public void AddHandler(VRCUpdateEventType eventType, IUdonEventReceiver handler)
{
  switch (eventType)
  {
    case VRCUpdateEventType.FixedUpdate:
      AddFixedUpdateHandler(handler);
      break;
    case VRCUpdateEventType.Update:
      AddUpdateHandler(handler);
      break;
    case VRCUpdateEventType.LateUpdate:
      AddLateUpdateHandler(handler);
      break;
    case VRCUpdateEventType.PostLateUpdate:
      AddPostLateUpdateHandler(handler);
      break;
  }
}
```

```
[PublicAPI]
public void RemoveHandler(VRCUpdateEventType eventType, IUdonEventReceiver handler)
{
  switch (eventType)
  {
    case VRCUpdateEventType.FixedUpdate:
      RemoveFixedUpdateHandler(handler);
      break;
    case VRCUpdateEventType.Update:
      RemoveUpdateHandler(handler);
      break;
    case VRCUpdateEventType.LateUpdate:
      RemoveLateUpdateHandler(handler);
      break;
    case VRCUpdateEventType.PostLateUpdate:
      RemovePostLateUpdateHandler(handler);
      break;
  }
}
```

These methods will then call the methods for directly adding and removing the handler to and from the array:

```
[PublicAPI]
public void AddFixedUpdateHandler(IUdonEventReceiver handler)
{
  hasFixedUpdateHandlers = (fixedUpdateHandlerCount = (fixedUpdateHandlers =
fixedUpdateHandlers.AddUnique(handler)).Length) > 0;
}

[PublicAPI]
public void RemoveFixedUpdateHandler(IUdonEventReceiver handler)
{
  if (fixedUpdateHandlerCount > 0) { hasFixedUpdateHandlers = (fixedUpdateHandlerCount =
(fixedUpdateHandlers = fixedUpdateHandlers.Remove(handler)).Length) > 0; }
}
```

When a handler has been added, EventDispatcher will proceed to call the named methods on the handler behaviours:

```
private const string
    FixedUpdateHandlerName = "_FixedUpdateHandler",
    UpdateHandlerName = "_UpdateHandler",
    LateUpdateHandlerName = "_LateUpdateHandler",
    PostLateUpdateHandlerName = "_PostLateUpdateHandler";

private void FixedUpdate()
{
  if (hasFixedUpdateHandlers)
  {
    for (int i = 0; i < fixedUpdateHandlerCount; i++)
    {
      fixedUpdateHandlers[i].SendCustomEvent(FixedUpdateHandlerName);
    }
  }
}

private void Update()
{
  if (hasUpdateHandlers)
  {
    for (int i = 0; i < updateHandlerCount; i++)
    {
      updateHandlers[i].SendCustomEvent(UpdateHandlerName);
    }
  }
}

private void LateUpdate()
{
  if (hasLateUpdateHandlers)
  {
    for (int i = 0; i < lateUpdateHandlerCount; i++)
    {
      lateUpdateHandlers[i].SendCustomEvent(LateUpdateHandlerName);
    }
  }
}
```

```
public override void PostLateUpdate()
{
  if (hasPostLateUpdateHandlers)
  {
    for (int i = 0; i < postLateUpdateHandlerCount; i++)
    {
      postLateUpdateHandlers[i].SendCustomEvent(PostLateUpdateHandlerName);
    }
  }
}
```

Script references from: https://github.com/Varneon/VUdon-EventDispatcher/blob/main/Packages/com.varneon.vudon.event-dispatcher/Runtime/Udon%20Programs/EventDispatcher.cs

VUdon - Udonity

VUdon - Udonity

Udonity is a runtime replica of the Unity Editor for VRChat, and by its nature it requires an observer pattern to be in place for efficient runtime operation.

All fields within Udonity implement an observer pattern for receiving an event when the value of the field is changed by the user:

```
[SerializeField, HideInInspector]
private UdonSharpBehaviour valueChangedCallbackReceiver;

[SerializeField, HideInInspector]
private string valueChangedCallbackMethod;

[SerializeField, HideInInspector]
private int fieldId = -1;

public void RegisterValueChangedCallback(UdonSharpBehaviour callbackReceiver, string
methodName)
{
  valueChangedCallbackReceiver = callbackReceiver;
```

```
  valueChangedCallbackMethod = methodName;
}

public void RegisterValueChangedCallback(UdonSharpBehaviour callbackReceiver, string
methodName, int id)
{
  fieldId = id;

  RegisterValueChangedCallback(callbackReceiver, methodName);
}

public void OnValueChanged()
{
  if (valueChangedCallbackReceiver != null)
  {
    if(fieldId >= 0)
    {
      valueChangedCallbackReceiver.SetProgramVariable("activeFieldId", fieldId);
    }

    valueChangedCallbackReceiver.SendCustomEvent(valueChangedCallbackMethod);
  }
}
```

Script reference from: https://github.com/Varneon/VUdon-Udonity/blob/main/Packages/com.varneon.vudon.udonity/Runtime/Udon%20Programs/Fields/Abstract/Field.cs