VL Concepts + Patterns II

Tebjan Halm + Elias Holzer NODE17

Intro

- Wanted to do:
- Top-Down view on everything
- Small simple examples
- See some libs that use a certian concept extensively
- Software Patterns

Examples Overview

-

Adaptive and Generic

Idea

- Strict type system comes with a draw back see vvvv GetSlice, +, ...
- Both solve the same problem: One node in node browser → work for many types → simpler, more flexible patching
- But with different ideas behind them:
- Generic: same definition for all types → GetSlice
- Adaptive: make specific definition for type → +, Lerp or ComputeArea for Rectangle or Circle

Generics

- Defines nodes that work for all types or a subset of types → try to write it for the most 'super' type
- One definition → One node in node browser → compiler generates concrete implementation when needed
- Often build with other generic operations
- First/Last slice example patch

Generic Types

- Generic Types have the same concept → Type parameters written with <T>
- Genric types of generic types is possible and handled by the compiler for example: Spread<PointWithColor<Vector3>>

Adaptive

- A specific signature gets registered to the system
- Everyone can implement it with same name + same pins, also member nodes → only signature counts
- Generic pins are allowed!
- Many definitions → One [Adaptive] node in node browser (+ hidden concrete ones) → compiler tries to find right implementaion
- Double click adaptive node to replace with concrete one or see what is available
- Adaptives also help to define generic nodes

Delegates

Idea

- Ad hoc operation definition
- Not registered in node browser but sent over link
- Call with Invoke node → see basic example
- Invoke can feed in data and enclosing operation can link in data
- Invoke will get the result → Invoke is (probably) in an operation that cannot do everything on its own
- Node with delegate input can be created as region → sugar for delegate + link
- Can have many delegate inputs → multiple layers → see assign menu → lifetime manager
- Loops are also just nodes with delegate input + slicer / accumulator feature

Basic Use Case

- Help an operation to complete a task → First/Last with selector patch
- Put type specific parts of the operation in an Invoke and let user provide delegate because user knows about the intended type
- In FirstAndLast it makes sense to pass it on to an input

Custom Layer System

- Well known from vvvv → Group joines layers into one new layer
- Collect all quads into one big list in the right order
- Layer gets the list and adds its data → called with the list/place where to add the data
- Group just passes the list on by calling the incoming layer → and so on
- Check Custome layer patch

Collections and LINQ

Collection Types

- Sequence
- Spread
- SpreadBuilder
- List
- Array
- Dictionary
- HashSet
- String → Array of characters

Collection Operations

- GetSlice vs GetItem
- Spread operation overview
- SpreadBuilder has almost the same
- SpreadBuilder for small and many local apread ops

LINQ

- Defines a set of operators which can work on different kind of data streams
- Two kinds escpially useful for us: Sequence and Observable
- Seqeuence → works for almost all collections
- Most nodes have one delegate to do type specific work → operation itself handles the collection handling → complicated programming is cast away → well tested over years and works!
- Needs Memoize for multi sinks because of how it works internally → reason for dev lib, would need wrappers like reactive nodes
- Helpful examples

- Where → filter elements
- Project, Select → change type
- OrderBy → select element to order
- Build helper types on the fly → tuple or small record

Reactive

Idea

- LINQ is pull data, RX is push data → turns the idea around → events → data stream oder time
- All regions get called when event happens
- Easy async/threading
- Can send everything, preferrably records

Reactive Operations

- Sources: Input devices, Interval
- In patch: ToObservable (and Sequence)
- Event modifiers → ForEach (and Keep)
- All LINQ operators

Large Collections

Large Object Heap

- .NET limitation
- Everything above 85 kib gets moved to the large object heap
- Slows down garbage collection → even stops the whole program
- Show particle example in CraftLie
- Problem is dynamic counts → cannot create/delete big memory per frame

Double Buffer

- To avoid LOH collection one needs to create one big place in memory and hold that
- Spreadbuilder can do that, it doubles its memory when more elements than capacity
- For dynamic instances like particles use two builders → add and read from A, clear and put result into B → end of frame swap the references → next frame the results are in A again because B is now A → like ping pong
- Fix particle example

Upcoming

Imports

- Drag’n’drop libs → play with them (WHILE RUNNING!!!)
- Opens up to all .NET libs you can find online + Framework
- Enables everyone to extend the library → looking forward to new vl packs

Interfaces

- Manage different types in one collection
- Call different type in the same way
- Build UIs
- Build a 3D engine → scenegraph

Video Tutorials

- From vl4vvvv beginner workshop
- From this reference workshop
- From CraftLie
- From Game Project