Internal DSL for Game GUI with F# Computation Expression

Yingjun Piao

yjpark@gmail.com

2019-01-30

DSL - Domain Specific Language

- A computer programming language of *limited expressiveness* focused on a *particular domain*. [1]
- **External DSL**: is a completely *separate language* that is parsed into data that the host language can understand
- **Internal DSL**: is just a particular idiom of writing code *in the host language*

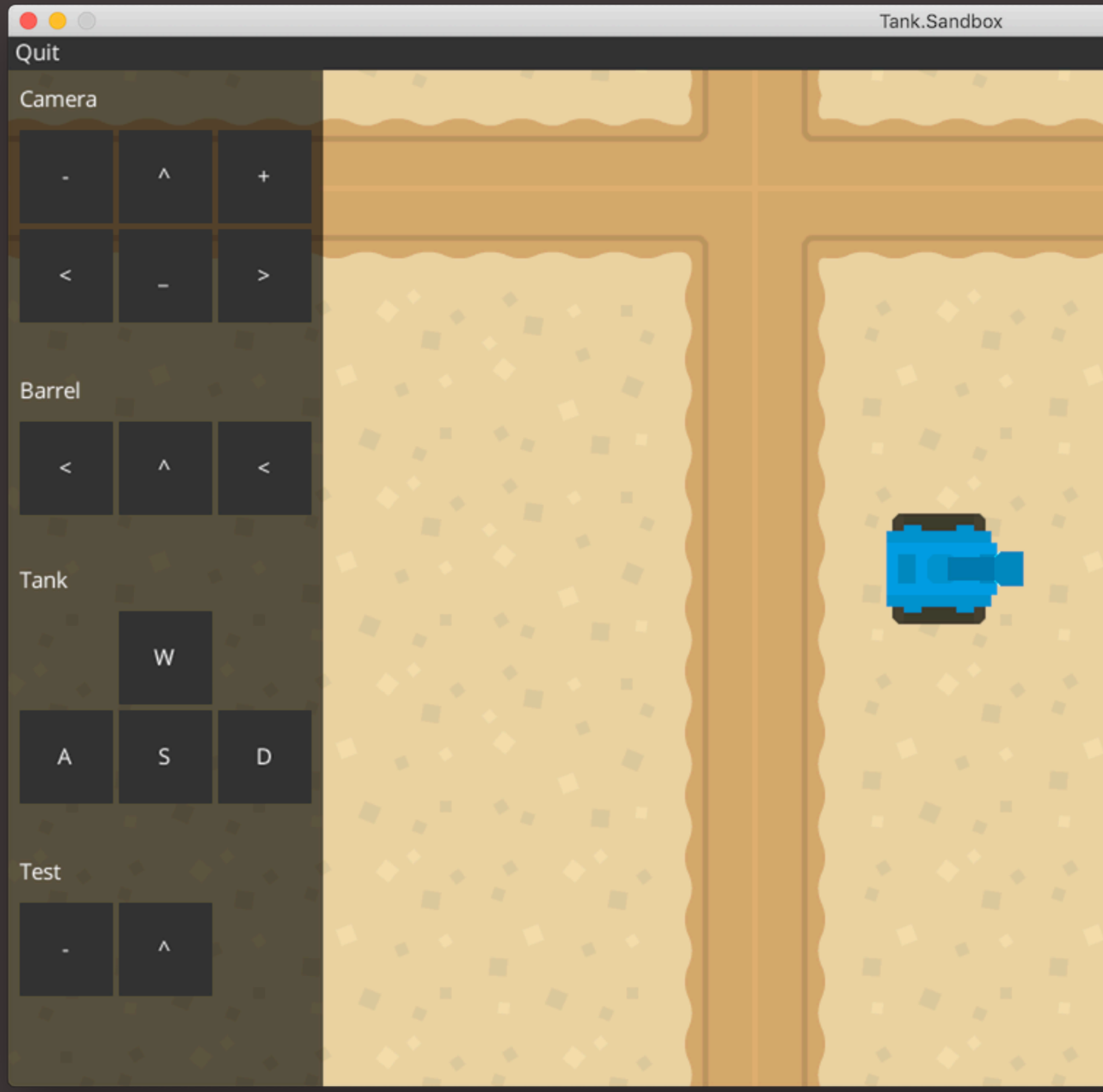
---

[1] https://martinfowler.com/dsl.html

	External	**Internal**
Syntax	Flexible	Limited
Feature	Flexible	Limited
Extend	Hard	Easy
Implementation	Hard	Easy
Editor	As Syntax	As Host

FSharp Game Tutorial

- F# on .Net Core
- on top of MonoGame
- simple 2D engine
- tank game as sample
- https://blog.yjpark.org
- https://github.com/yjpark/FSharpGameTutorial



F# Computation Expression

- F# support both functional and object oriented
- Computation expressions in F# provide a convenient syntax for writing computations that can be sequenced and combined using control flow constructs and bindings.
- Pretty powerful and complex [2]
- Mostly use `Custom Operations` [3]

---

[2] https://fsharpforfunandprofit.com/series/computation-expressions.html

[3] https://docs.microsoft.com/en-us/dotnet/fsharp/language-reference/computation-expressions#custom-operations

Internal DSL: [F# Game Tutorial] Game.Gui/Builder/Base.fs

```
type WidgetBuilder<'widget when 'widget :> Widget> () =
    member this.Yield (_ : 'a) = this.Zero ()
    abstract Zero : unit -> 'widget

    [<CustomOperation("pos")>]
    member __.Pos (widget : 'widget, l : int, t : int) =
        widget.Left <- l
        widget.Top <- t
        widget
    ...
```

Internal DSL: [F# Game Tutorial] Game.Gui/Builder/Button.fs

```
type TextButtonBuilder () =
    inherit WidgetBuilder<TextButton> ()
    override __.Zero () = new TextButton ()

    [<CustomOperation("text")>]
    member __.Text (widget : TextButton, v : string) =
        widget.Text <- v
        widget
    [<CustomOperation("onUp")>]
    member __.OnUp (widget : TextButton, onUp : EventArgs -> unit) =
        widget.Up.Add onUp
        widget
```

Internal DSL: [F# Game Tutorial] Tank.Sandbox/GuiHelper.fs

```
let boxButton (x : int) (y : int) (t : string) upAction =
    button {
        text t
        pos x y
        size 64 64
        onUp upAction
    }
```

Internal DSL: [F# Game Tutorial] Tank.Sandbox/TankGui.fs 1/2

```
let init (x : int) (y : int) (gui : IGui<Panel>) =
    let tank = gui.Game.GetAddon<TankAddon> ()
    let body = tank.Body
    gui.AddChildrenWithOffset (x, y,
        label {
            text "Tank"
            pos 0 0
        },
        boxButton 68 32 "W" <| move body 5.0f,
        boxButton 68 100 "S" <| move body -2.0f,
        boxButton 0 100 "A" <| turn body -90.0f,
        boxButton 136 100 "D" <| turn body 90.0f
    )
```

Internal DSL: [F# Game Tutorial] `Tank.Sandbox/TankGui.fs` 2/2

```
let private move (entity : IEntity) (speed : float32) =
    fun _args ->
        let r = entity.Transform.AbsoluteRotation
        let dir = Vector2 (-MathF.Sin (r), MathF.Cos (r))
        let offset = dir * speed
        entity.Transform.Position <- entity.Transform.Position + offset

let private turn (entity : IEntity) (a : float32) =
    fun _args ->
        entity.Transform.Angle <- entity.Transform.Angle + a
```

Summary

- DSL is powerful and useful in game development
- need to understand the benefits and drawbacks
- and your language regarding to DSL implementation
- choose wisely with DSL design
- and be careful with the implementation

Thank You