UPDATE? NOT EVEN ONCE

Introducción a Reactive Programming en Unity

VALERIA COLOMBO
Technical Owner

RODRIGO REARDEN
Technical Owner

Roadmap

- **¿Qué es Rx? ¿Para qué sirve Rx?**
- Patrón Observer
- Ejemplo
- Práctica

https://github.com/etermax/RxBirdGame

¿Qué es Rx?

Programación Reactiva es una forma cómoda de programar usando flujos de datos asincrónicos.

Abreviaremos Programación Reactiva a **Rx**, y llamaremos a los flujos **pipes, data streams**, o simplemente **streams**.

Por un lado, nuestro programa reacciona a la llegada **eventos** que llegan por diferentes **streams**.

Por otro, a los **streams** se los puede transformar mediante **operadores** de filtrado, conversión, combinación, unión, y muchos otros para adaptarlos al requerimiento de la aplicación.

¿Para qué sirve Rx?

En general, estamos acostumbrados a programar de forma sincrónica:

```
void Update()
{
    if (thingIsDone)
        DoThisOtherThing();
}
```

Si bien estas líneas son simples, más temprano que tarde ese **if** va a tener otras condiciones y chequeos de ocurrencias de otros eventos anidados, complejizando la lectura.

¿Para qué sirve Rx?

Por ejemplo, si quisiera llamar solo una vez a **DoThisOtherThing()**

```
void Update()
{
    if (thingIsDone && !alreadyDidOtherThing)
    {
        DoThisOtherThing();
        alreadyDidOtherThing = true;
    }
}
```

¿Para qué sirve Rx?

Una primera opción es resolverlo con una corrutina.

```
IEnumerator DoThisOtherThingCoroutine()
{
    yield return new WaitUntil(() => thingIsDone)
    DoThisOtherThing();
}
```

Nos sacamos de encima un nivel de anidación, pero realmente no es un código del que uno pueda estar demasiado orgulloso, el boilerplate de usar **yield** e **IEnumerator** es grande, y si uno no conoce las corrutinas… es hasta confuso.

¿Para qué sirve Rx?

Una solución en Rx podría ser

```
void DoThisOtherThingReactive()
{
    Observable.EveryUpdate()
        .TakeWhile(_ => !thingIsDone)
        .Last()
        .Subscribe(_ => DoThisOtherThing());
}
```

Parece más código, pero esto **inicia** y **finaliza** todo el flujo. A diferencia de con Corutinas/Updates, Start/Stop Coroutine, o llenar el código de chequeos de estado (if, if, if, switch, if) no son necesarios.

¿Para qué sirve Rx?

Si **thingIsDone** fuera un evento reactivo en lugar de un flag....

```
void DoThisOtherThingReactive()
{
    Observable.EveryUpdate()
        .TakeUntil(ThingIsDone)
        .Last()
        .Subscribe(_ => DoThisOtherThing());
}
```

El cambio es menor, pero agrega un poco de legibilidad, veremos TakeWhile y TakeUntil en más detalle en un rato.

Roadmap

- ~~¿Qué es Rx? ¿Para qué sirve Rx?~~
- **Patrón Observer**
- Ejemplo
- Práctica

https://github.com/etermax/RxBirdGame

Patrón Observer

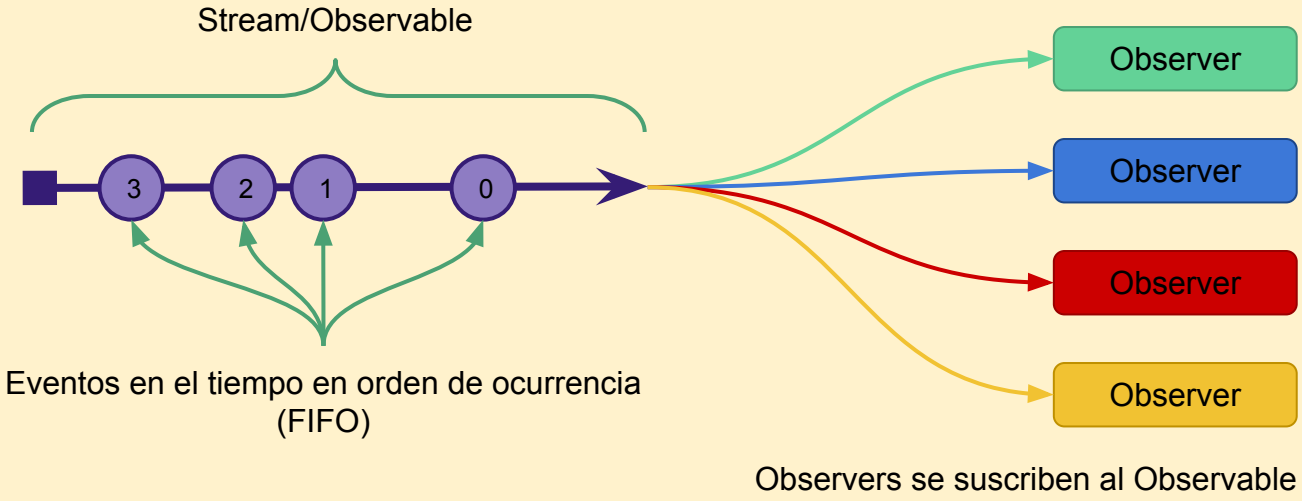
El **Patrón Observer** define la relación entre **observers** y **observable**. Cuando el **observable** emite un evento, todos los **observers** que se suscribieron al mismo son notificados y reaccionan automáticamente.

Sirve para mantener a los dos roles desacoplados entre sí, agregar un nuevo **observer** no implica cambio alguno en el código del **observable**.

Rx es una implementación de este patrón.

¿Que es un Observable?

En Rx, **observable** y **stream** son sinónimos.



Operadores básicos de observable

Un stream es una secuencia de eventos ordenados en el tiempo. Puede emitir tres cosas diferentes: un valor (de cualquier tipo), un error o una señal de completitud.

Algunos operadores básicos que se le pueden aplicar a un stream:

- **map (Select)**: es la transformación más clásica. Permite tomar cada elemento del stream y realizar una transformación al mismo.
- **filter (Where)**: sirve para *filtrar* elementos dentro un stream.
- **last**: obtiene el último elemento una vez que el observable termine.
- **takeUntil**: termina el observable cuando otro observable emita un elemento.

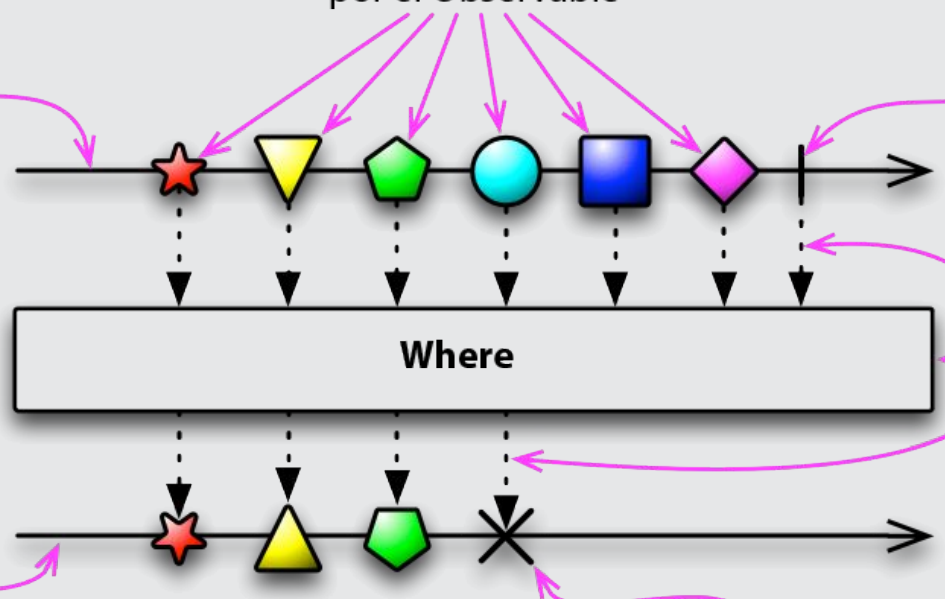
¿Que es un Observer?

En **Rx**, un **observer** se suscribe a un **observable**.

Un **observer** reacciona a los elementos o secuencias de elementos que el **observable** emite.

Esto facilita el uso de operaciones concurrentes ya que no necesita bloquearse mientras espera a que el observable emita elementos.

Resumiendo...



Esta es la timeline del Observable. El tiempo pasa de izquierda a derecha

Estos son items emitidos por el Observable

Esta linea vertical indica que el Observable se completo con exito

Estas lineas punteadas y esta caja indican que una transformacion es aplicada al Observable. El texto dentro de la caja muestra la naturaleza de la transformacion

Este Observable es el resultado de la transformacion

Si por alguna razon el Observable termina de forma anormal, con un error, la linea vertical se reemplaza por una X

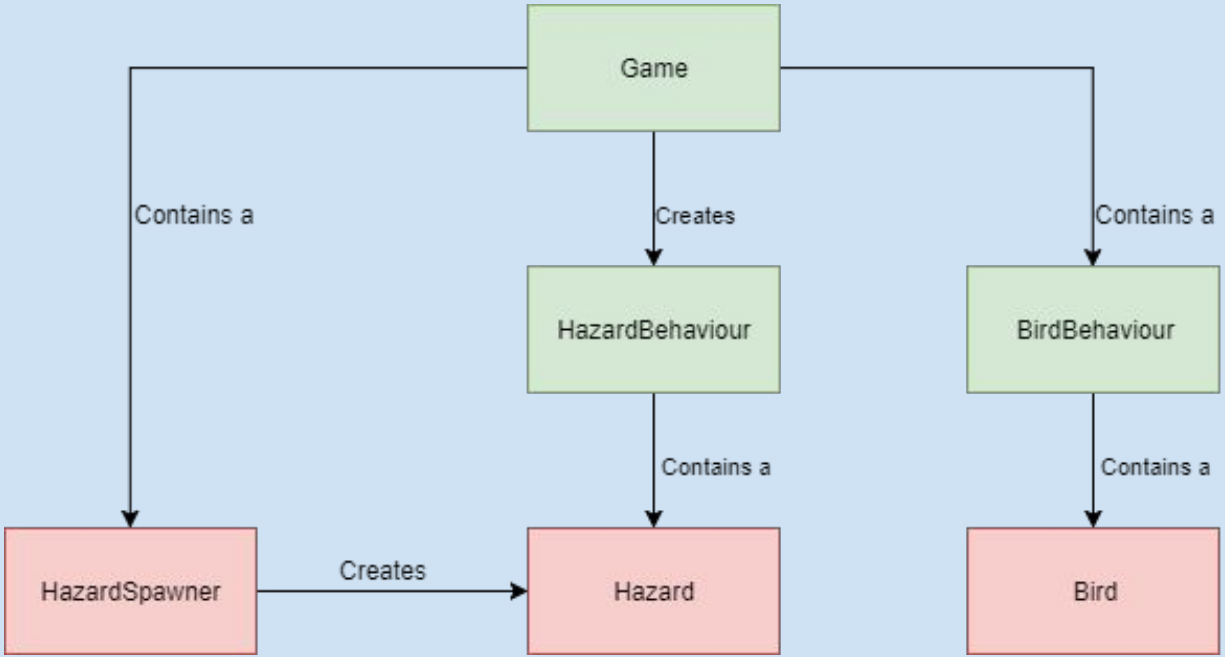
Roadmap

- ~~¿Qué es Rx? ¿Para qué sirve Rx?~~
- ~~Patrón Observer~~
- **Ejemplo**
- Práctica

https://github.com/etermax/RxBirdGame

Vamos a verlo en acción...

Hicimos un Flappy Bird sin usar Update ni Corrutinas... Solo con Rx...



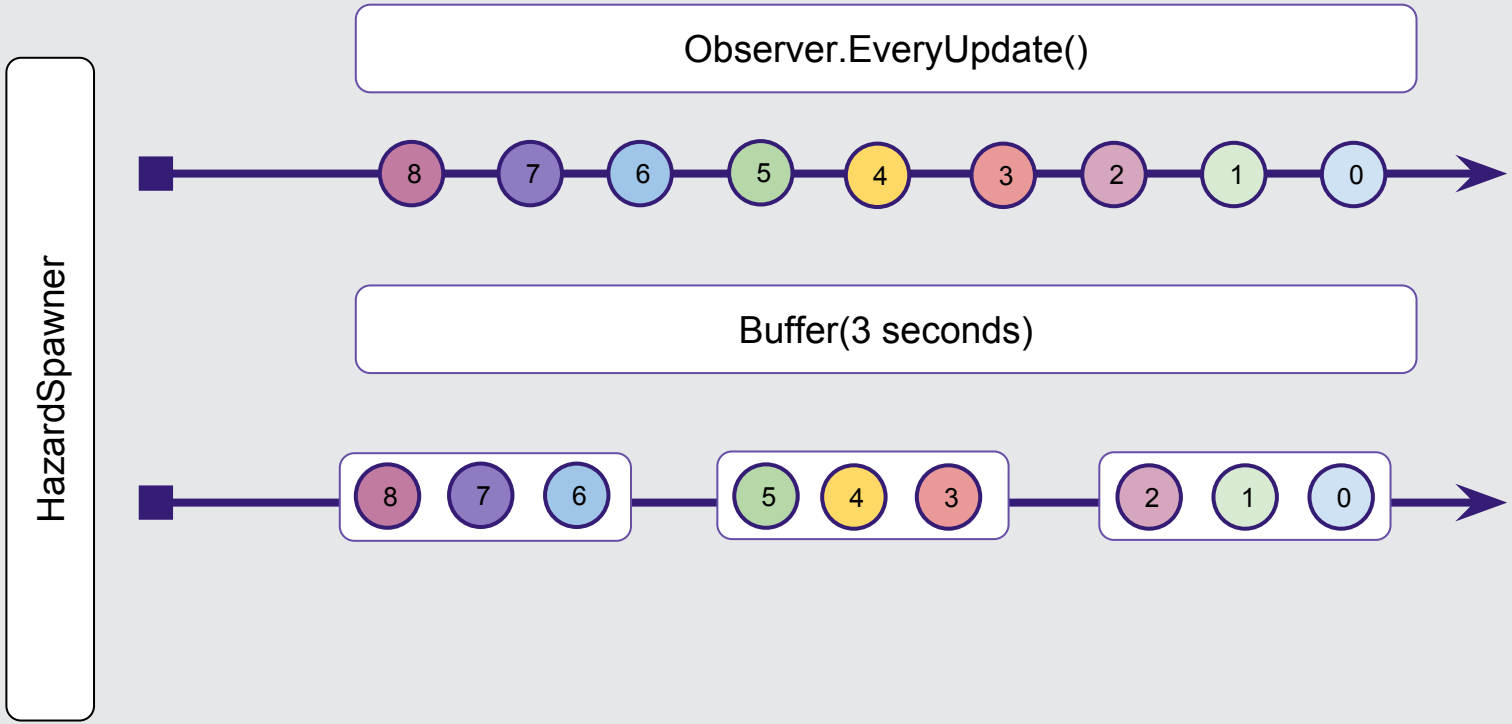
https://github.com/etermax/RxBirdGame

Creación de los obstáculos

Observer.EveryUpdate()

HazardSpawner

Creación de los obstáculos



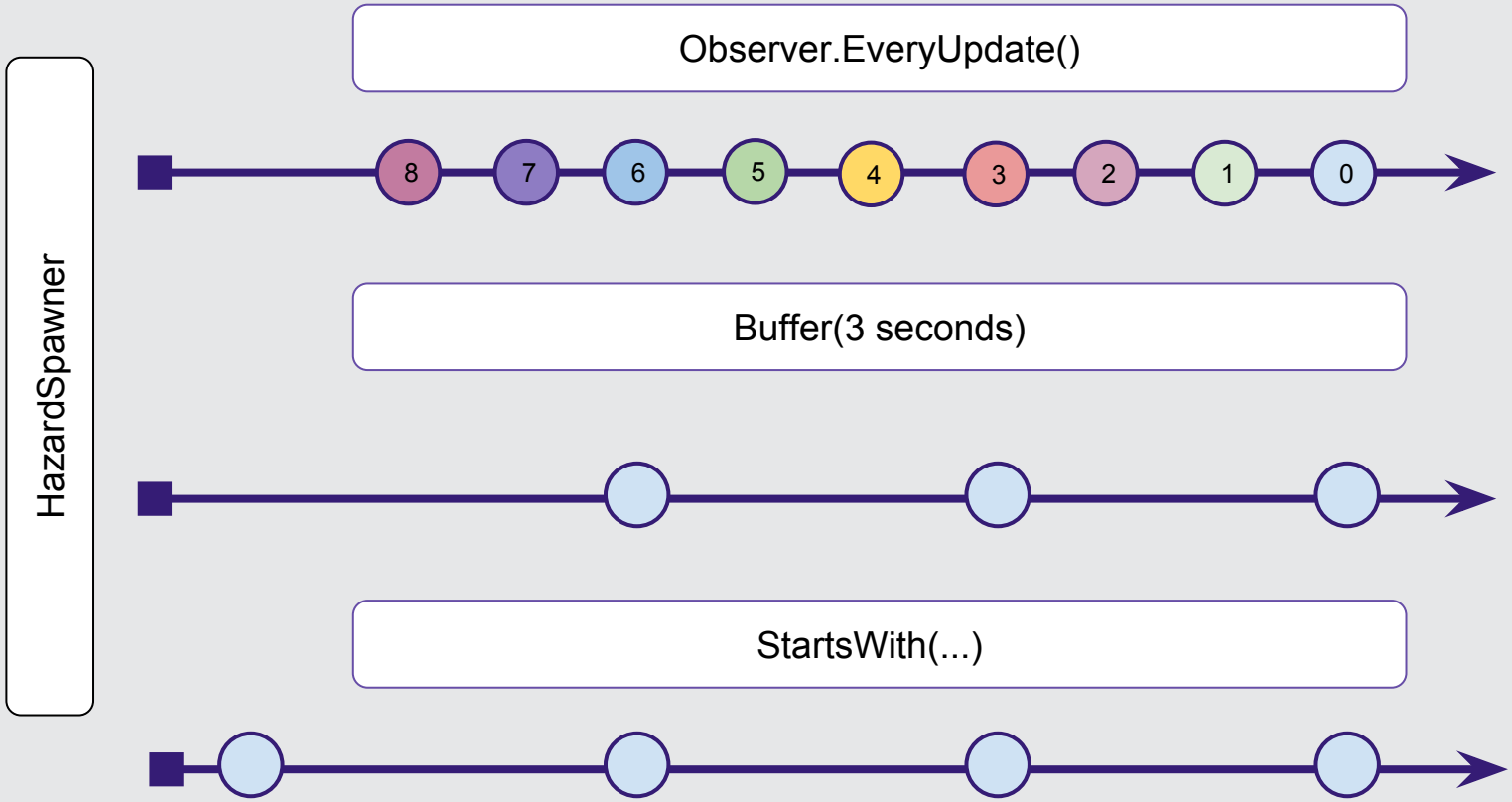
Creación de los obstáculos

HazardSpawner

Observer.EveryUpdate()

Buffer(3 seconds)

Creación de los obstáculos



Creación de los obstáculos

HazardSpawner

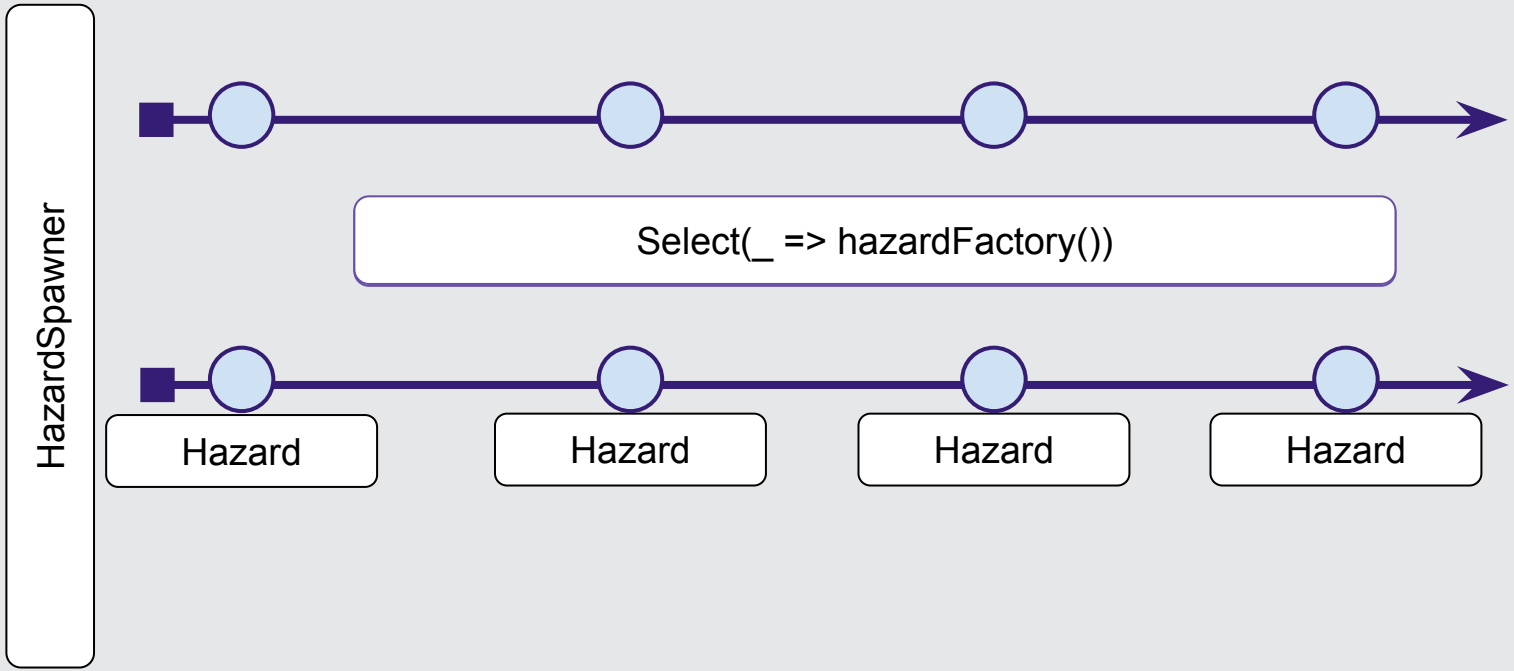
Observer.EveryUpdate()

Buffer(3 seconds)

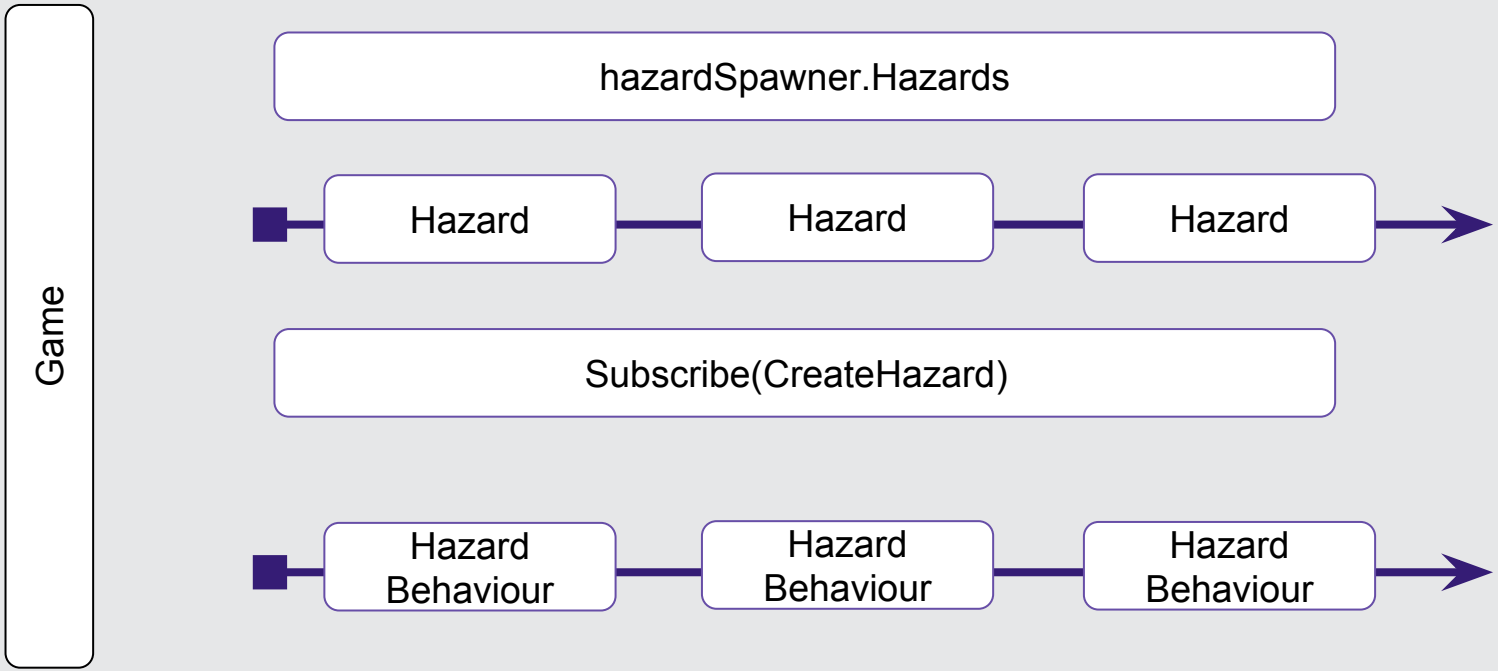
StartsWith(...)

Select(_ => hazardFactory())

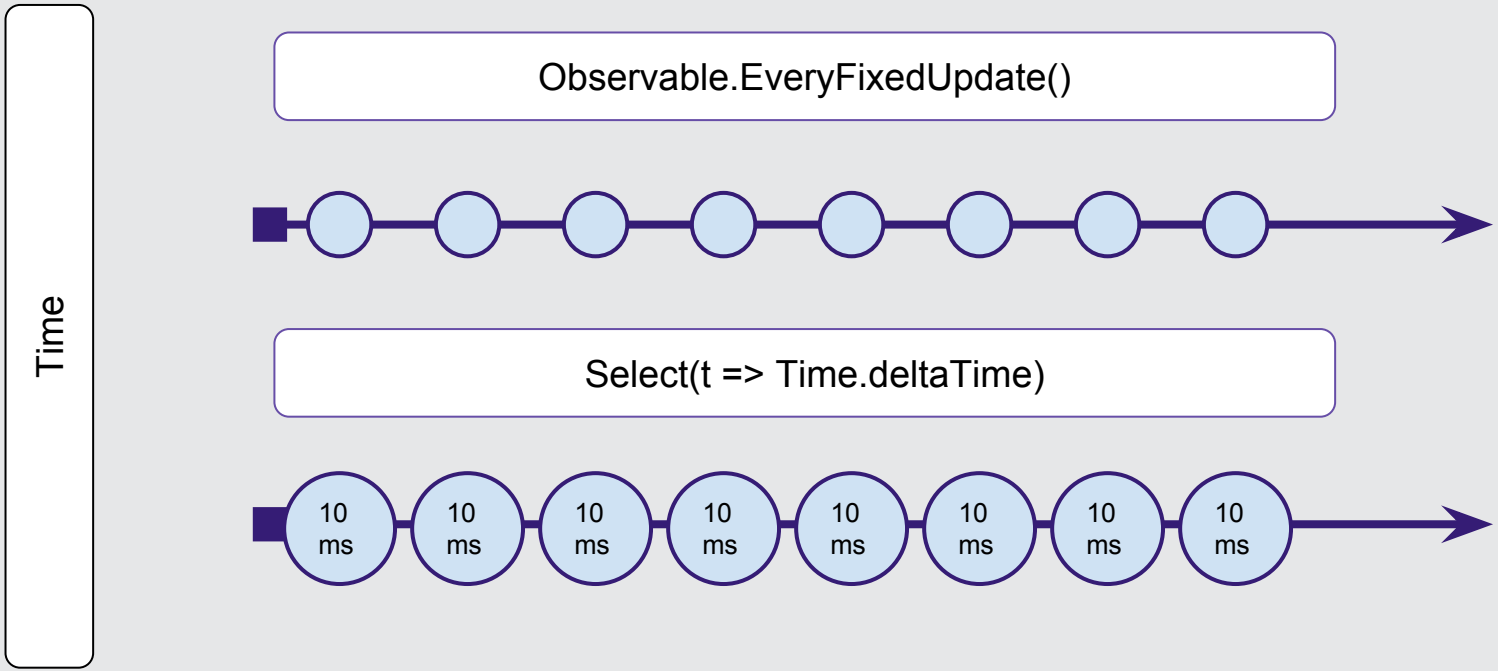
Creación de los obstáculos



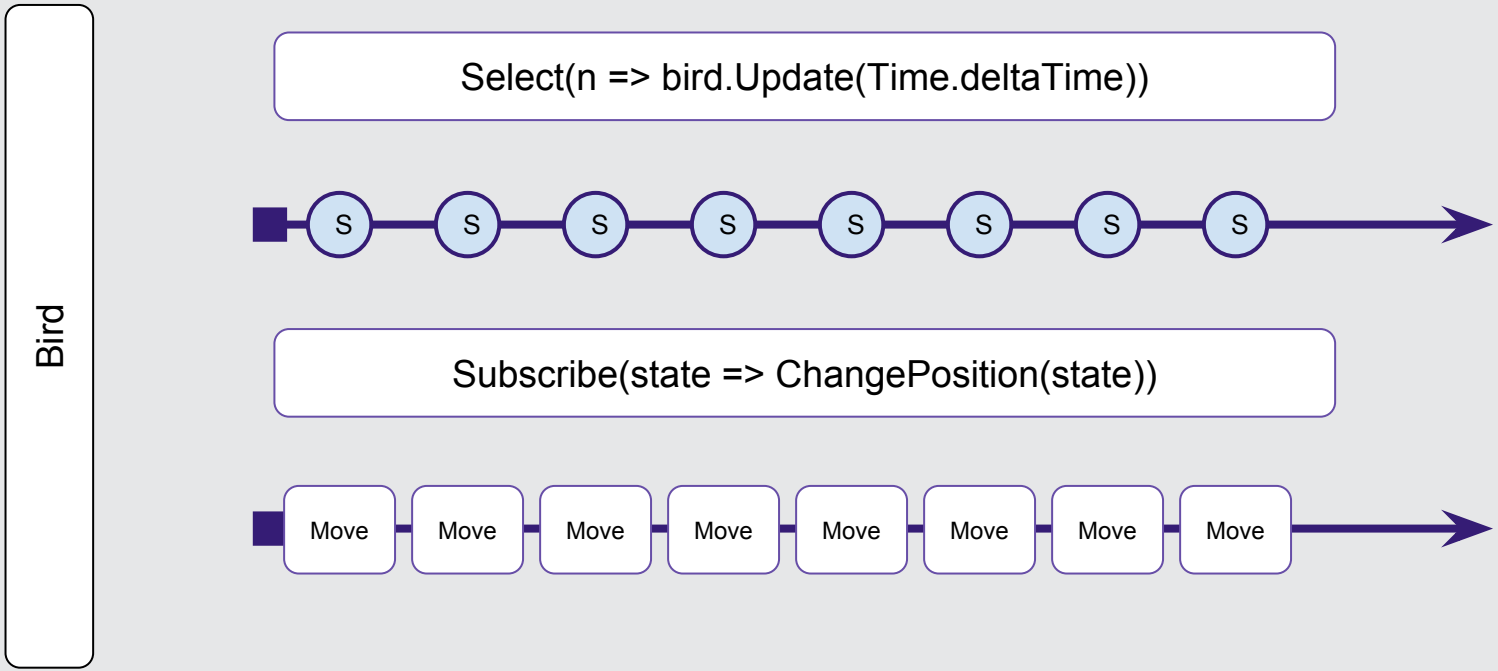
Creación de los obstáculos



Movimiento del pajarito



Movimiento del pajarito



Roadmap

- ~~¿Qué es Rx? ¿Para qué sirve Rx?~~
- ~~Patrón Observer~~
- ~~Ejemplo~~
- **Práctica**

https://github.com/etermax/RxBirdGame

Pero... La clase bird no es completamente Rx!!!

Ahora les toca a ustedes...

- *Reactivizar* Bird

- Mostrar puntaje en pantalla: cada vez que el pajarito pasa un obstáculo, se incrementa 1 punto

- Cuando el pajarito sube, se inclina hacia arriba, cuando baja, se inclina hacia abajo.

Links útiles

UniRx (Reactive Extensions for Unity)

https://github.com/neuecc/UniRx

The introduction to Reactive Programming you've been missing

https://gist.github.com/staltz/868e7e9bc2a7b8c1f754

Documentación de Rx en varios lenguajes

http://reactivex.io

¿PREGUNTAS?

¡GRACIAS!

VALERIA COLOMBO
valeria.colombo@etermax.com

RODRIGO REARDEN
rodrigo.rearden@etermax.com