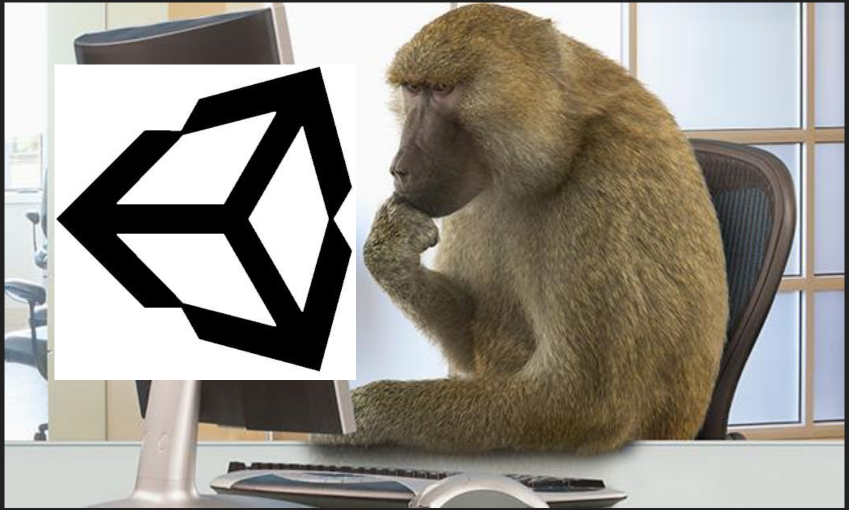
Unity & VR Best Practices



A long-winded discussion-lecture where I talk a lot and maybe someone learns something but probably not.

~By Victor Mouschovias~

Who am I?

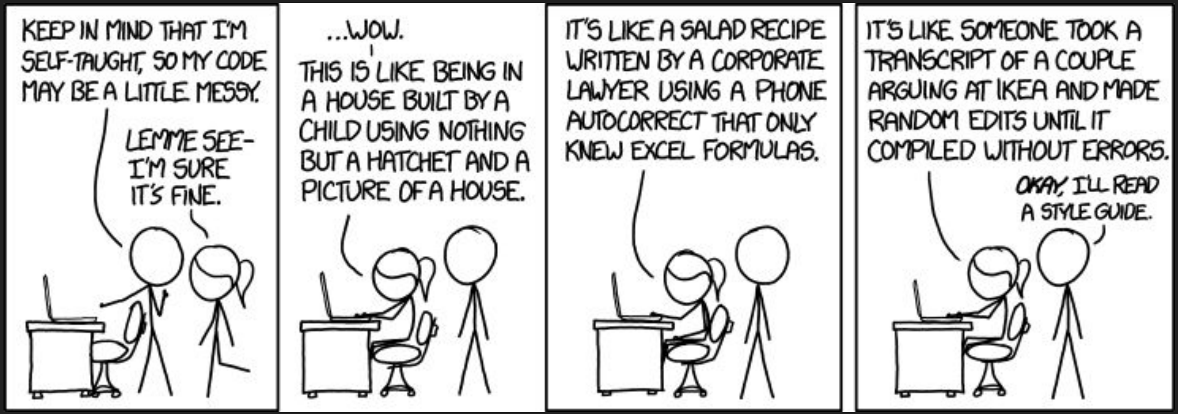
- 24601

- GameBuilders Chair

- Future Psyonix Dev!

Why are we here?

- Unity encourages *really friggin' awful* solutions to problems.
- People are super good at making super bad VR.
- Nobody reads the Oculus Best Practices Guide :'(


KEEP IN MIND THAT I'M SELF-TAUGHT, SO MY CODE MAY BE A LITTLE MESSY.
LEMME SEE– I'M SURE IT'S FINE.
...WOW.
THIS IS LIKE BEING IN A HOUSE BUILT BY A CHILD USING NOTHING BUT A HATCHET AND A PICTURE OF A HOUSE.
IT'S LIKE A SALAD RECIPE WRITTEN BY A CORPORATE LAWYER USING A PHONE AUTOCORRECT THAT ONLY KNEW EXCEL FORMULAS.
IT'S LIKE SOMEONE TOOK A TRANSCRIPT OF A COUPLE ARGUING AT IKEA AND MADE RANDOM EDITS UNTIL IT COMPILED WITHOUT ERRORS.
OKAY, I'LL READ A STYLE GUIDE.

Friggin' Awful Solutions

Steve is a Minecraft character. He is making a 3D puzzle game in Unity. His game has one scene with a single puzzle. When players solve the puzzle, Steve wants to play some special effects baked into the environment (stars twinkle, confetti cannons erupt, etc.).

Steve has a ***PuzzleScript*** that handles all puzzle logic, including checking for the victory condition.

Steve is a computer science student, but he's already ***paid an art student*** to make all of his effects. How can Steve accomplish his goal?

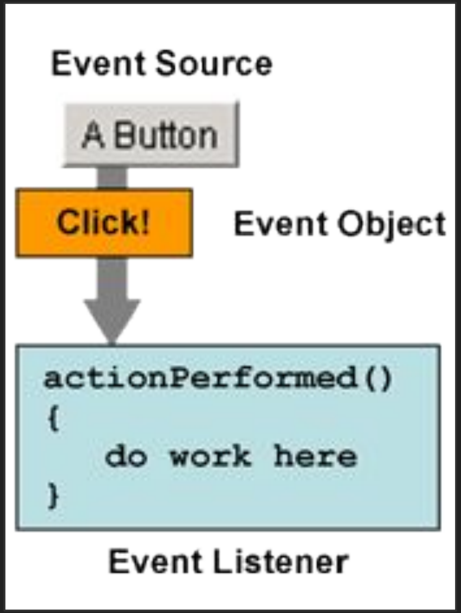
Friggin' Awful Solutions

Steve is still a Minecraft character. Now he wants to add 10 more scenes, each with the same puzzle and ***PuzzleScript***. Each scene requires its own, unique victory effects.

How can Steve trigger each of these effects from one universal script?!?

Our Hero: Event-Based Programming

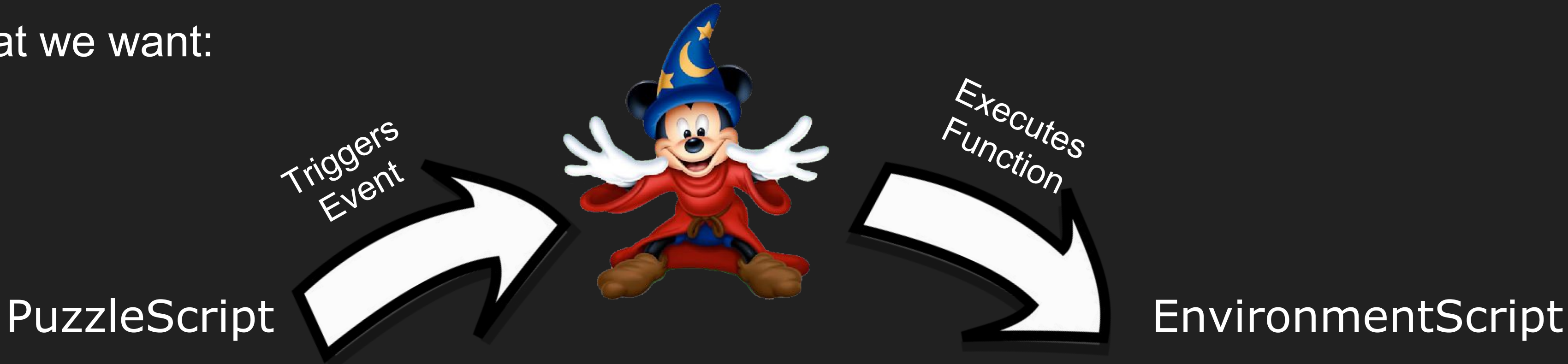
- A programming paradigm where functions are called by *event managers* in response to events.
- Very common in mobile & web development
- Unity has some “rigid” events
  - OnCollisionEnter
  - Start
  - Update



```
actionPerformed()
{
   do work here
}
```

We need a more Flexible Structure!

What we want:



PuzzleScript

Triggers Event

Executes Function

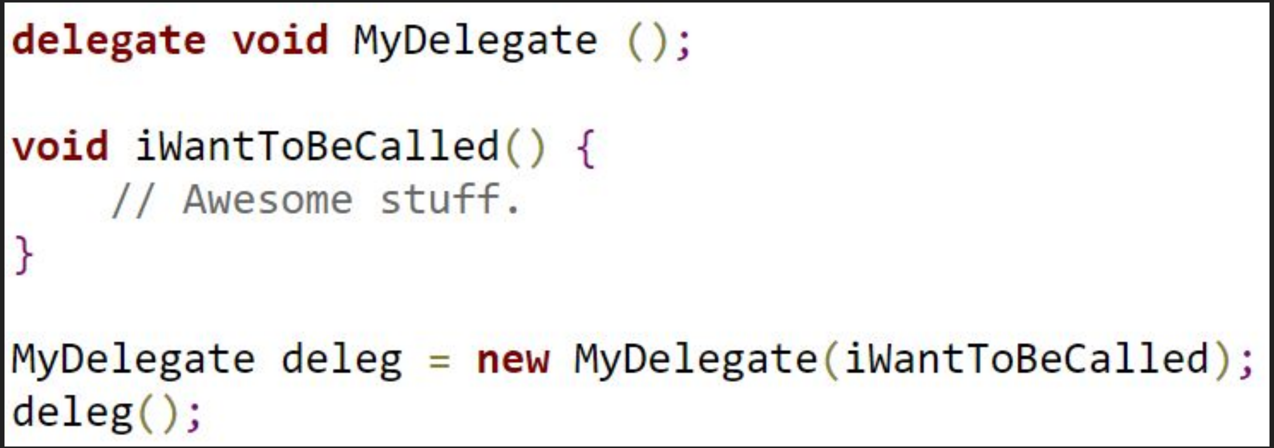
EnvironmentScript

What we have:

```
delegate int SomeDelegate(int x);
```

Delegates

- “Points” to a certain type of function.
- Can be assigned like any variable.
- Can “remotely” call a function.



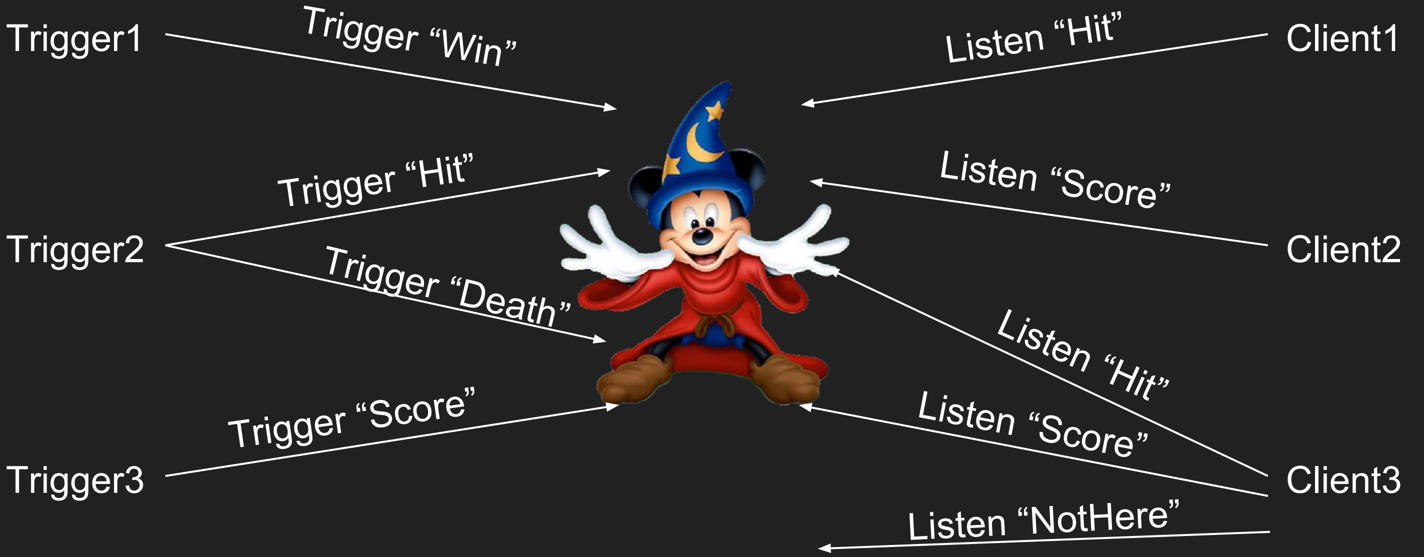
```
delegate void MyDelegate ();

void iWantToBeCalled() {
    // Awesome stuff.
}

MyDelegate deleg = new MyDelegate(iWantToBeCalled);
deleg();
```

The Callback Machine

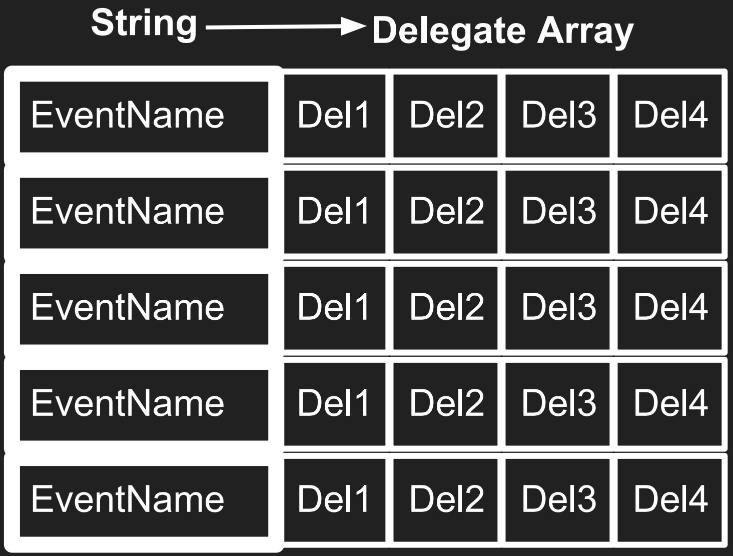
- We can do better than storing a single function to callback...



The Callback Machine

Register Listener

- Args:
    - EventName (String)
    - Callback (Delegate)
- Place delegate in the dictionary under “EventName”



String → **Delegate Array**

EventName	Del1	Del2	Del3	Del4
EventName	Del1	Del2	Del3	Del4
EventName	Del1	Del2	Del3	Del4
EventName	Del1	Del2	Del3	Del4
EventName	Del1	Del2	Del3	Del4

Trigger Event

- Arg:
    - EventName (String)
- Iterate through “EventName” entry and call each delegate.
    - How could we add argument support?

That’s not all, Folks!

We’ve just solved a surprising amount of problems…

- Non-Blocking control flow
- Too many Singletons
- Need for Global Data
- Messy code

Other Dangers

- “Fluid” Component Structures
  - Strict class hierarchies have their benefits.
- Loading resources as needed, dynamic resolutions
  - Resources.Load, GameObject.Find, etc.
- Using basic Unity for *everything*
  - Right tool for the right job.
- Networking

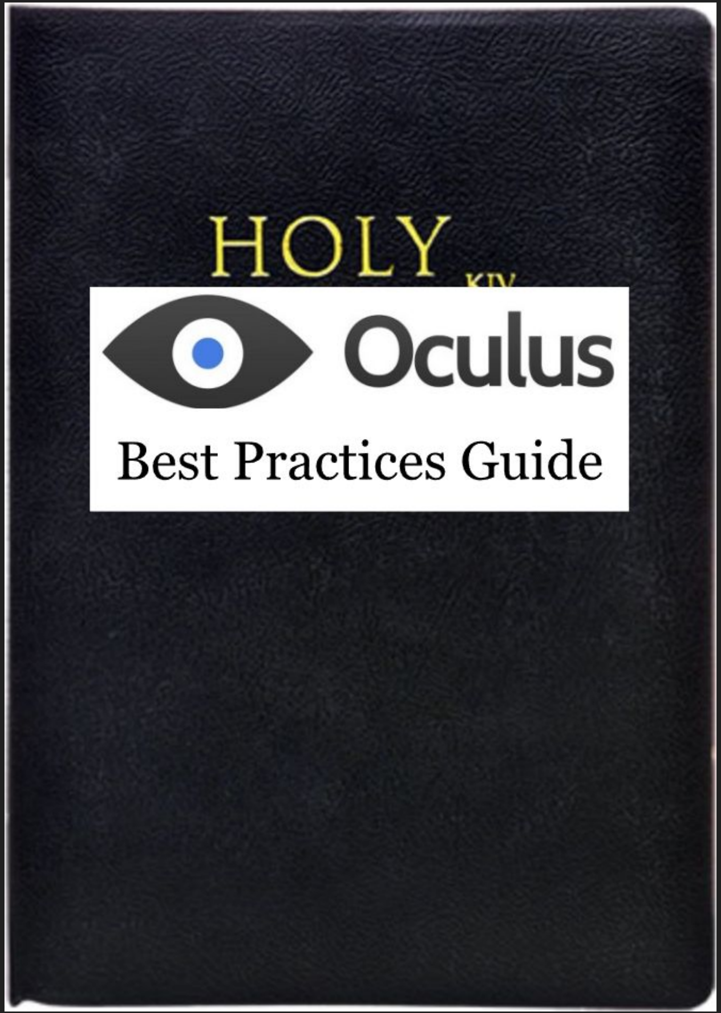
Let’s Talk “Bad” VR

[illegible]

i totally dont remember showing you this

Let’s Talk “bad” VR

- High Production Value != Good Design
    - Testing is expensive!
    - Testers usually don’t identify issues directly.
- We need to make better designs...


HOLY
KJV
Oculus
Best Practices Guide

Minimizing Latency

- FPS can't drop *no matter what*.
- Games with sandbox elements might have some issues...

Accelerations

- Vection, vestibular system, blah blah blah.
- Easy to forget about:
  - Rotations
  - Teleportation effects
  - Preparing users for motion
    - More on this soon

Accelerations, Field of View

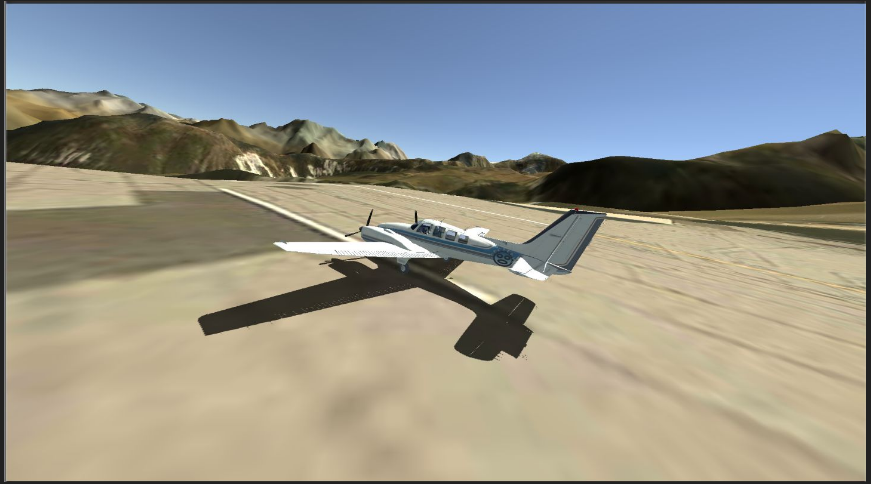
- The less they see, the less they feel!
- ***Very, very, very*** useful to provide constant frame of reference.
- Users may have to move their head more, so watch out.

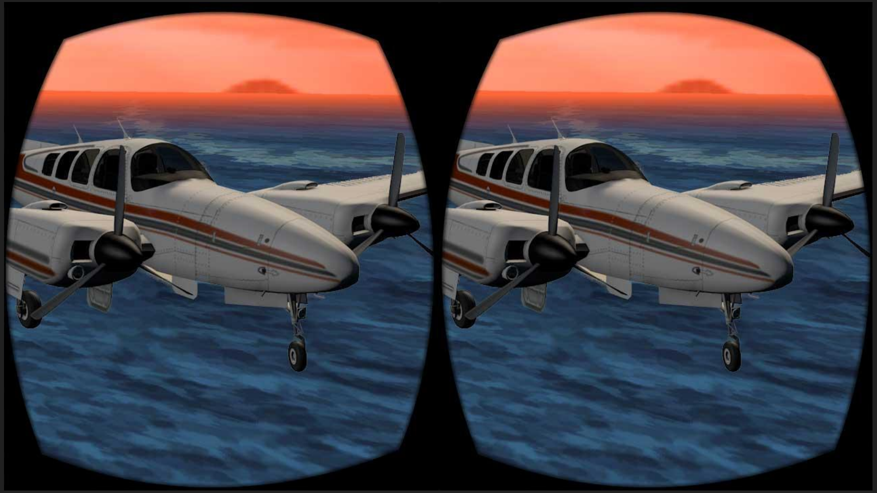
Accelerations, Movement

- Movement in-line with the viewing direction is optimal.
- Preparing the body for movement goes a long way...

Third-Person Cameras

- Subject to the same accelerations issues as first-person.
  - Camera swings!
- We lose some Field of View control.
- We can decouple camera and avatar movement!
  - Flight sims can benefit from this!





User Interface

- Part of the 3D world.
  - **NOT RIGIDLY ATTACHED TO USER'S HEAD! >:(**
- Sits 2-3 meters in front of eyes.
- Doesn't require eye-swivels.
  - Put UI in middle ⅓ of viewing area.
  - Or allow head movements to examine UI features.

Other Stuff

- Sound cues
- Content
    - Don’t rely on stereoscopic vision
- Altitude
    - “*Visual flow*” of pixels

Questions? How are your projects going?

FRIDAY'S