University of Colorado Boulder

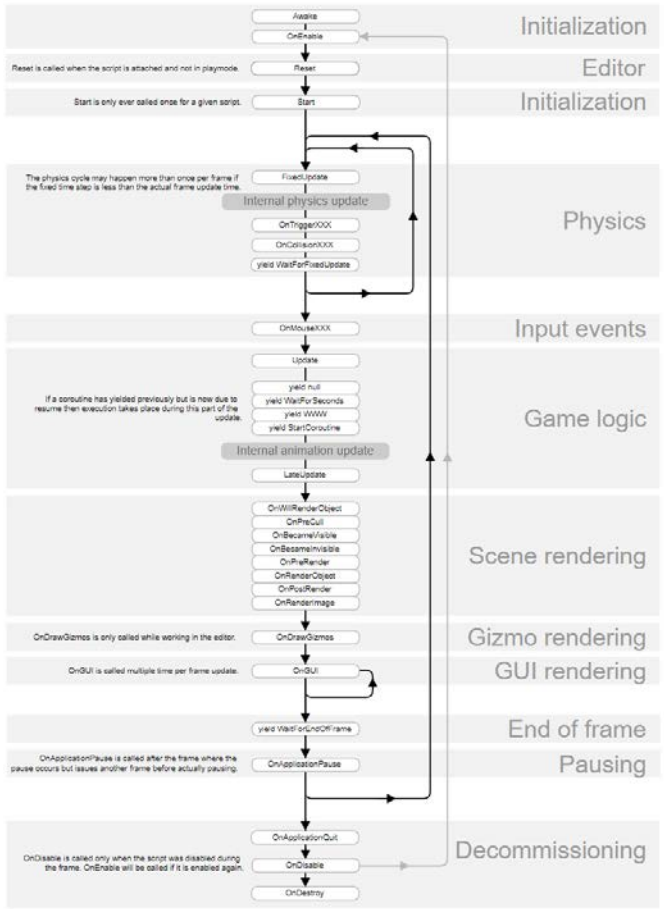
Introduction to Virtual Reality

Timing and Engine Optimization

Professor **Dan Szafir**

Computer Science & ATLAS Institute
University of Colorado Boulder

Engine Overview



Process input

Update state

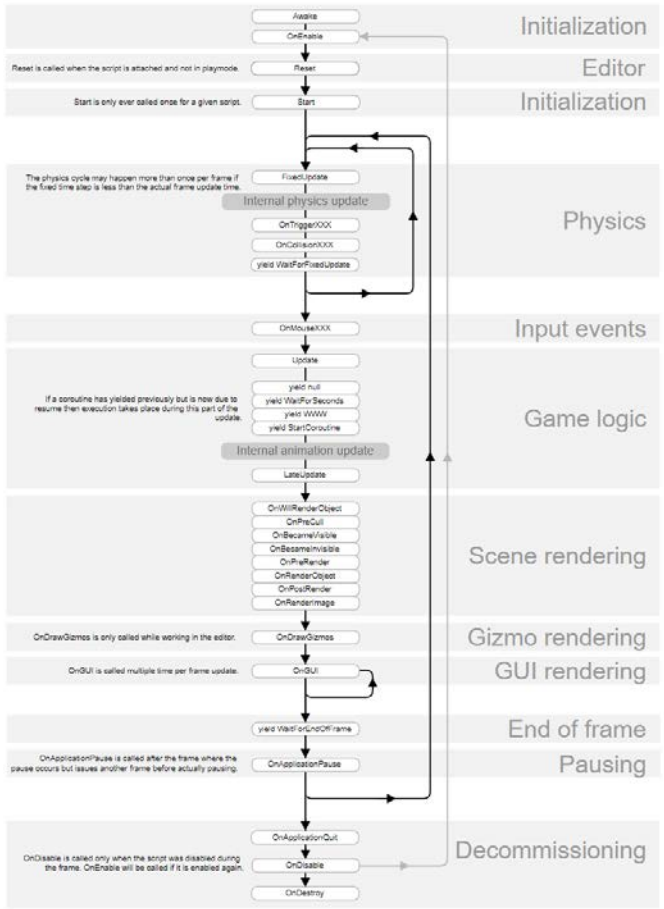
Render a frame

Application

Graphics Pipeline (or “Rendering Pipeline”) using CPU / GPU

Display

Engine Overview



Process input

Update state

Render a frame

Application

Graphics Pipeline (or “Rendering Pipeline”) using CPU / GPU

Display

Asynchronous vs Synchronous

Synchronous: things happen on the main thread as the computer's internal clock ticks

```
void Update () {
    if(Input.GetMouseButtonDown(0))
    {
        Debug.Log("Mouse button pressed");
    }
}
```

Asynchronous: things happen on a separate background thread and/or in response to something unexpected (e.g., an *event* like a button press or mouse click)

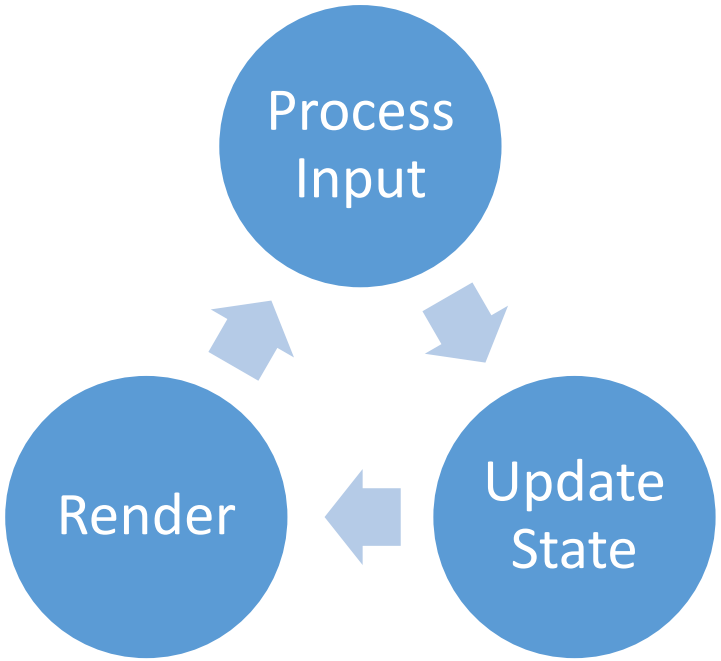
- Unity has coroutines for background operations (will cover more later)

MVC Controller: Process Input

- Usually you want *asynchronous communication* (why?)
- Program using *callbacks*
  - A form of “event-driven programming”
  - Other examples: robotics controller driven by sensor input
  - …but not implemented for input in Unity (experimental in 2018.2)
    - Can just use conditionals and pass messages manually if you need
- See Javascript example on Moodle

Timing

Game clock



vs

How much has to be calculated each frame?

What is the hardware?

Real time

Frame Rate

Frequency at which frames are displayed

Measured in Frames per second (FPS)

Examples:

- 10 – 12 fps: individual images
- 16 – 24 fps: early silent films (but variable as they used hand cranks)
- 24 fps: standard with introduction of sound
- 30 fps: standard in US TV (NTSC)
- 48 fps: experimental in cinema (first film: The Hobbit)

Frame Rate vs Refresh Rate

Frame rate: how many frames can we draw in a second

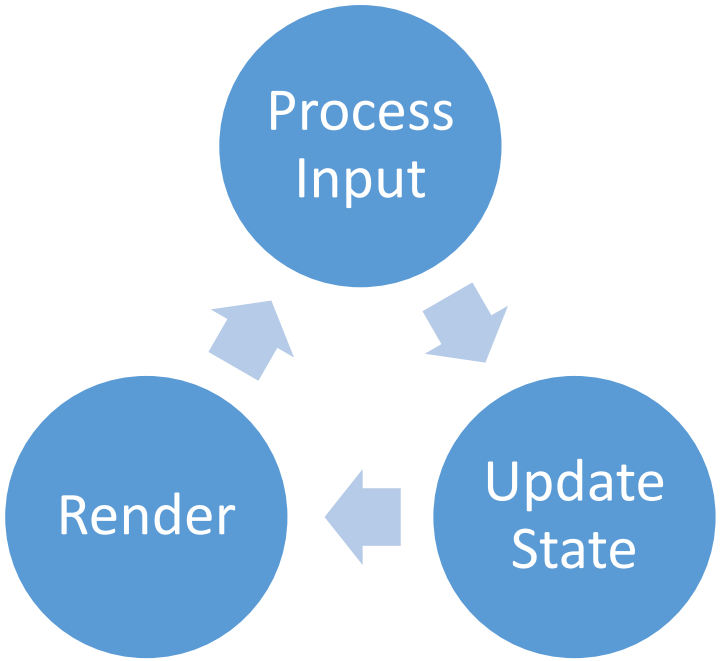
Refresh rate: how often are we showing something on the display (might be repeated drawing of identical picture)

Refresh rate measured in Hz

- -60Hz: US TV (NTSC), my monitor
- -90Hz: Oculus Rift, HTC Vive
- -120Hz: Active stereo (60 Hz per eye)

Timing

Game clock vs Real time



How much has to be calculated each frame?

What is the hardware?

Frame Rate

Problem: we want constant FPS!

(or at least constant simulation/gameplay regardless of FPS)

Managing the rate of gameplay

Fixed time step with no synchronization: do nothing (render as fast as possible)

Fixed time step, no synchronization

First GameLoop

Let's add an FPS counter

Managing the rate of gameplay

Fixed time step with no synchronization: do nothing (render as fast as possible)

- **Problem: variable rate of play**

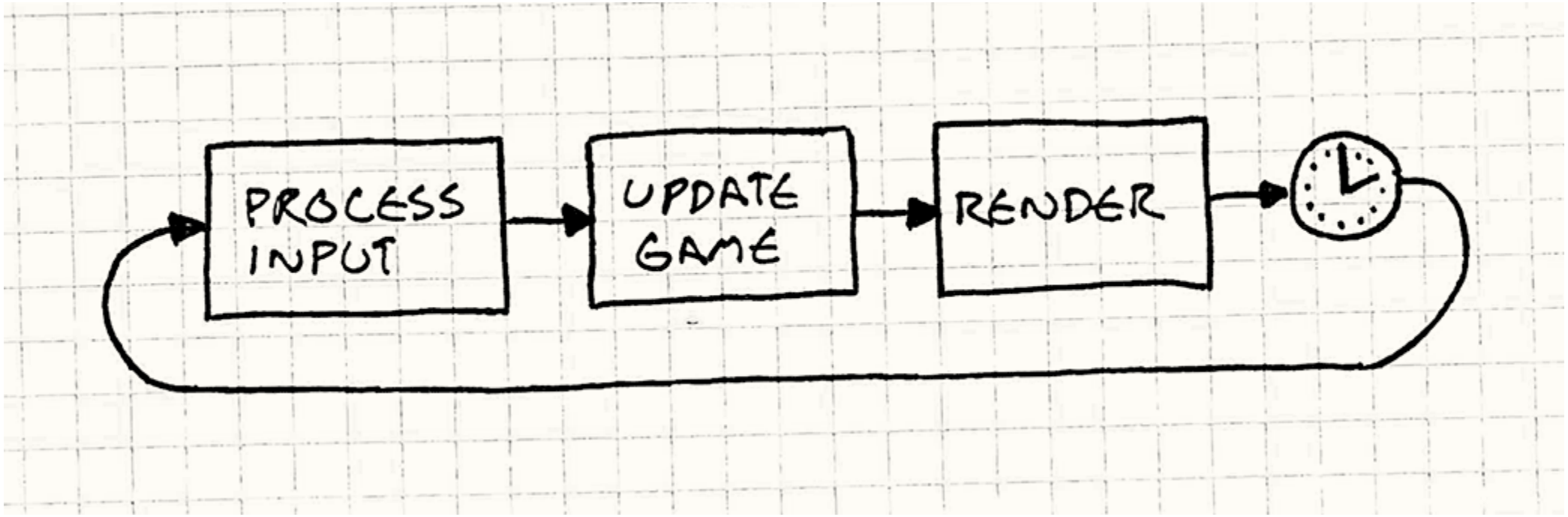
Managing the rate of gameplay

Fixed time step with no synchronization: do nothing (render as fast as possible)

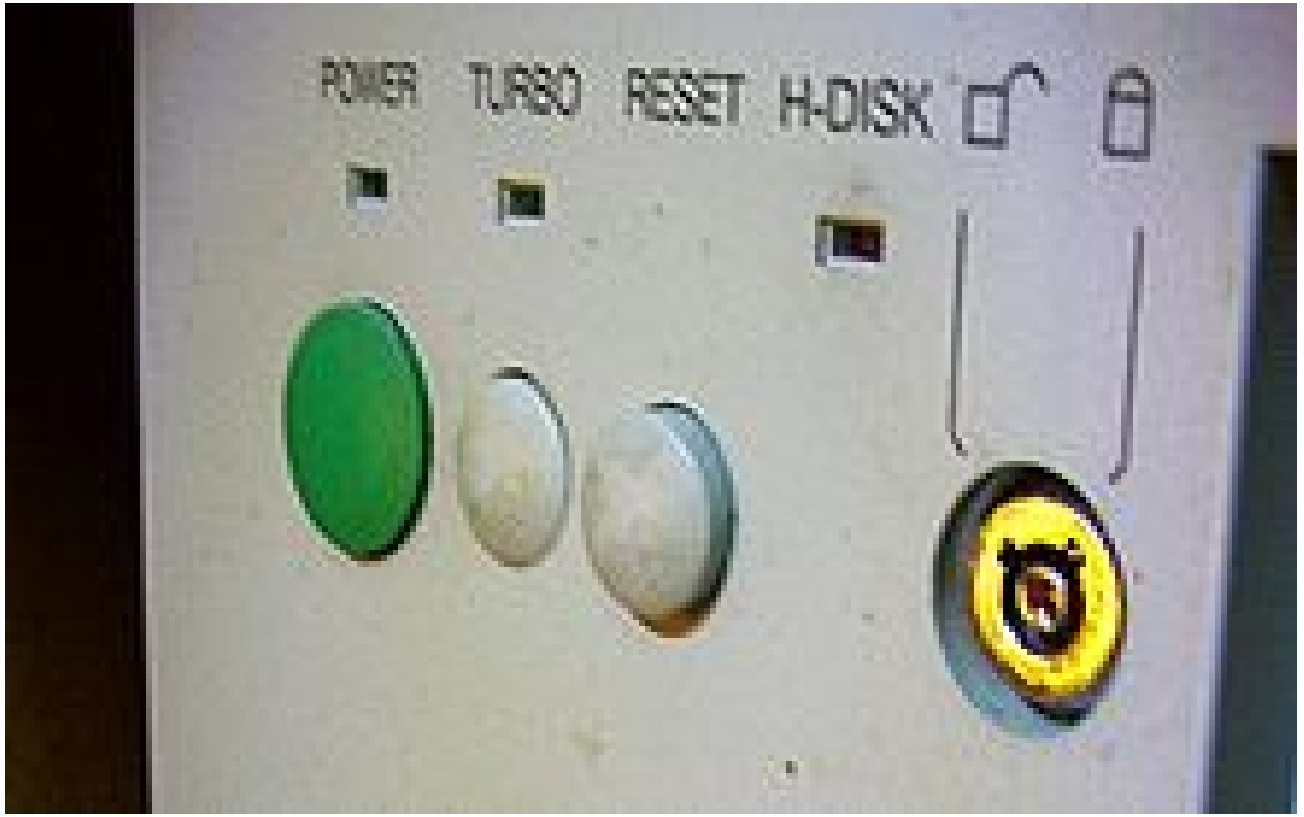
- Problem: variable rate of play

Fixed time step with synchronization: wait after rendering

Fixed Time Step with Synchronization



Turbo Button



https://www.youtube.com/watch?v=p2q02Bxtqds

Managing the rate of gameplay

Fixed time step with no synchronization: do nothing (render as fast as possible)

- Problem: variable rate of play

Fixed time step with synchronization: wait after rendering

- **Fixes too fast rendering**
- **Problem: what if our FPS is < Refresh rate? (almost always will be)**

Managing the rate of gameplay

Fixed time step with no synchronization: do nothing (render as fast as possible)

- Problem: variable rate of play

Fixed time step with synchronization: wait after rendering

- Fixes too fast rendering
- Problem: what if our FPS is < Refresh rate? (almost always will be)

Variable time step: adapt update to FPS

Variable Time Step

```
Main loop:
        ...
        delta = currentTime – lastRenderTime
        update(delta)
        ...

Update(delta):
        position += velocity * delta
```

Managing the rate of gameplay

Fixed time step with no synchronization: do nothing (render as fast as possible)

- Problem: variable rate of play

Fixed time step with synchronization: wait after rendering

- Fixes too fast rendering
- Problem: what if our FPS is < Refresh rate? (almost always will be)

Variable time step: adapt update to FPS

- **Enables constant FPS regardless of hardware**
- **Problem: Physics problems, Non-deterministic**

Managing the rate of gameplay

Fixed time step with no synchronization: do nothing (render as fast as possible)

- Problem: variable rate of play

Fixed time step with synchronization: wait after rendering

- Fixes too fast rendering
- Problem: what if our FPS is < Refresh rate? (almost always will be)

Variable time step: adapt update to FPS

- Enables constant FPS regardless of hardware
- Problem: Physics problems, Non-deterministic

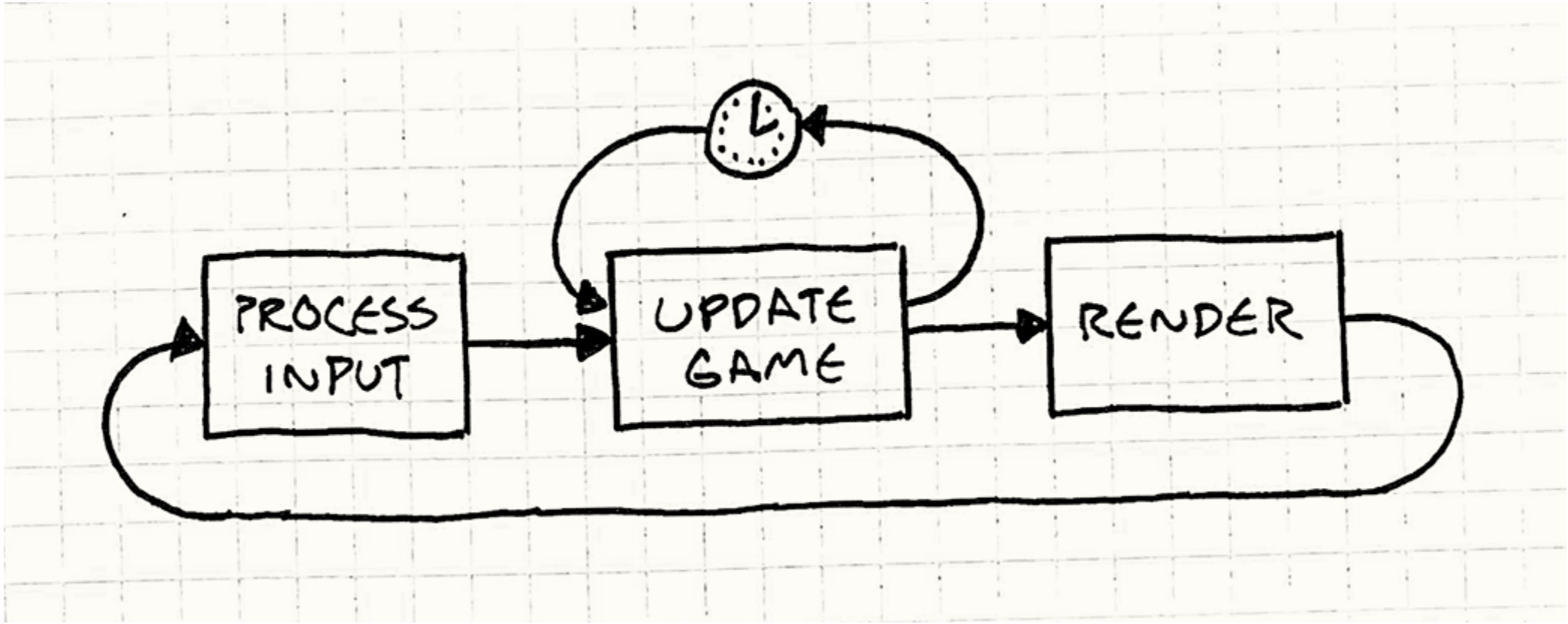
Fixed update time step, variable rendering

Fixed update time-step, variable rendering

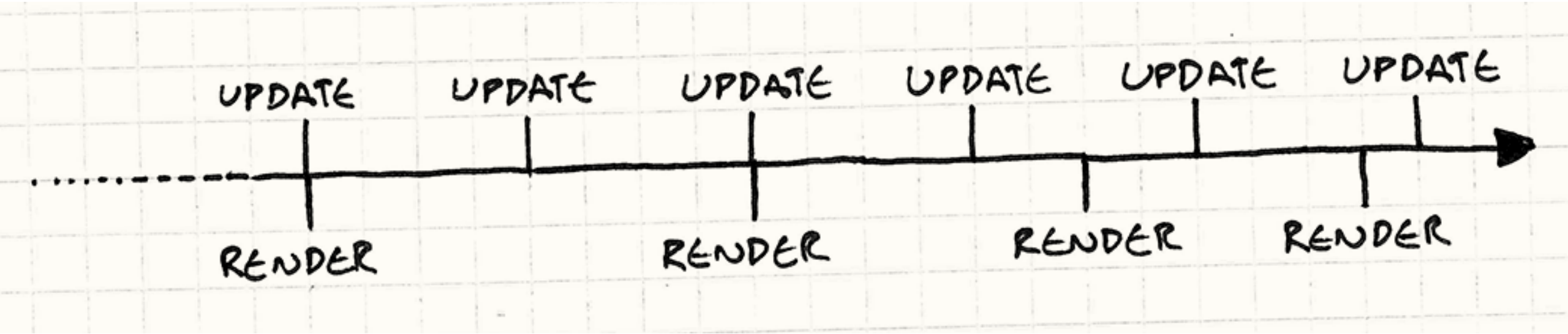
Assignment 4!

This is why Unity has both Update() and FixedUpdate() functions!

Separate Updating from Rendering



Rendering in the middle of updates



Managing the rate of gameplay

Fixed time step with no synchronization: do nothing (render as fast as possible)

- Problem: variable rate of play

Fixed time step with synchronization: wait after rendering

- Fixes too fast rendering
- Problem: what if our FPS is < Refresh rate? (almost always will be)

Variable time step: adapt update to FPS

- Enables constant FPS regardless of hardware
- Problem: Physics problems, Non-deterministic

Fixed update time step, variable rendering

- **Separates updating from rendering (model from view)**
- **Simulates at a constant rate, visible scene may render at variable rate**
- **Need to carefully select time step size (must be > time to update)**
- **Problem: spiral of death if time step < update time**
- **Fix: adjust amount of work in update if frame rate drops**

Managing the rate of gameplay

Fixed time step with no synchronization: do nothing (render as fast as possible)

- Problem: variable rate of play

Fixed time step with synchronization: wait after rendering

- Fixes too fast rendering
- Problem: what if our FPS is < Refresh rate? (almost always will be)

Variable time step: adapt update to FPS

- Enables constant FPS regardless of hardware
- Problem: Physics problems, Non-deterministic

Fixed update time step, variable rendering

- Separates updating from rendering (model from view)
- Simulates at a constant rate, visible scene may render at variable rate
- Need to carefully select time step size (must be > time to update)
- Problem: spiral of death if time step < update time
- Fix: adjust amount of work in update if frame rate drops

University of Colorado
Boulder

THANKS!

Professor **Dan Szafir**

Computer Science & ATLAS Institute
University of Colorado Boulder