Section 3 - Krazy Karts

These are the slides that accompany the Unreal Multiplayer course.

Looking for something, try searching our GitHub repo.

Enjoy your stay!

Sam & Ben
GameDev.tv

Live Google Slides at bit.ly/multiplayerslides

z

Introduction to Krazy Karts

Creating A Go-Kart Pawn

Make It Move!

- Add an appropriate camera
- Hook up the forward axis
- Use to set a velocity member variable
- Update position in tick.

Understanding Forces And Movement

How Things Move

$$F = m * a$$

$$a = F / m$$

Live Google Slides at bit.ly/multiplayerslides

Acceleration

$$dx / dt = v$$

$$dx = v * dt$$

$$dv / dt = a$$

$$dv = a * dt$$

Provide The Driving Force.

- Where should it come from?
- Make sure that it is configurable.
- Test!

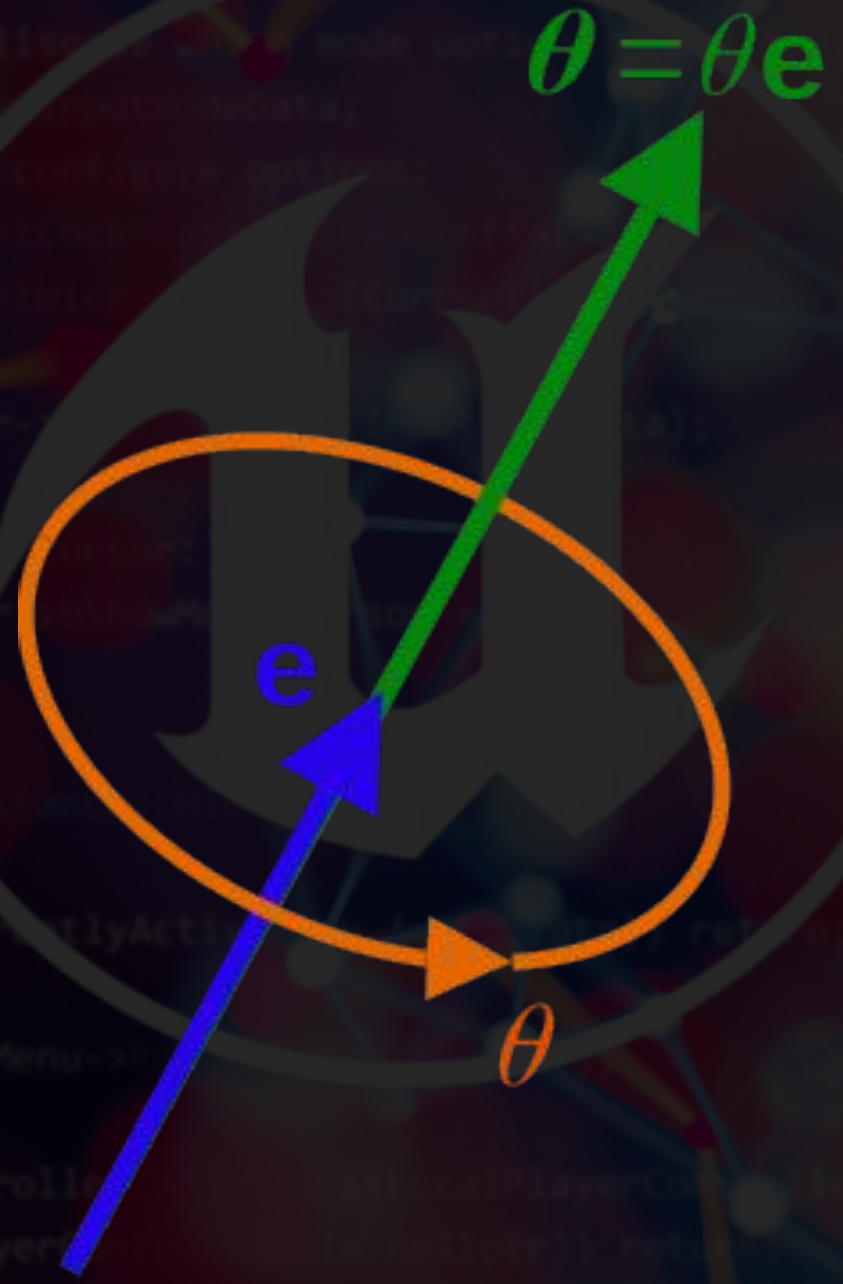
Blocking Movement Without Physics

Reset The Velocity

- Read up on your `FHitResult`
- Check if there was a collision
- Reset velocity
- Refactor for clarity.

Rotations With Quaternions

Angle Axis Rotations

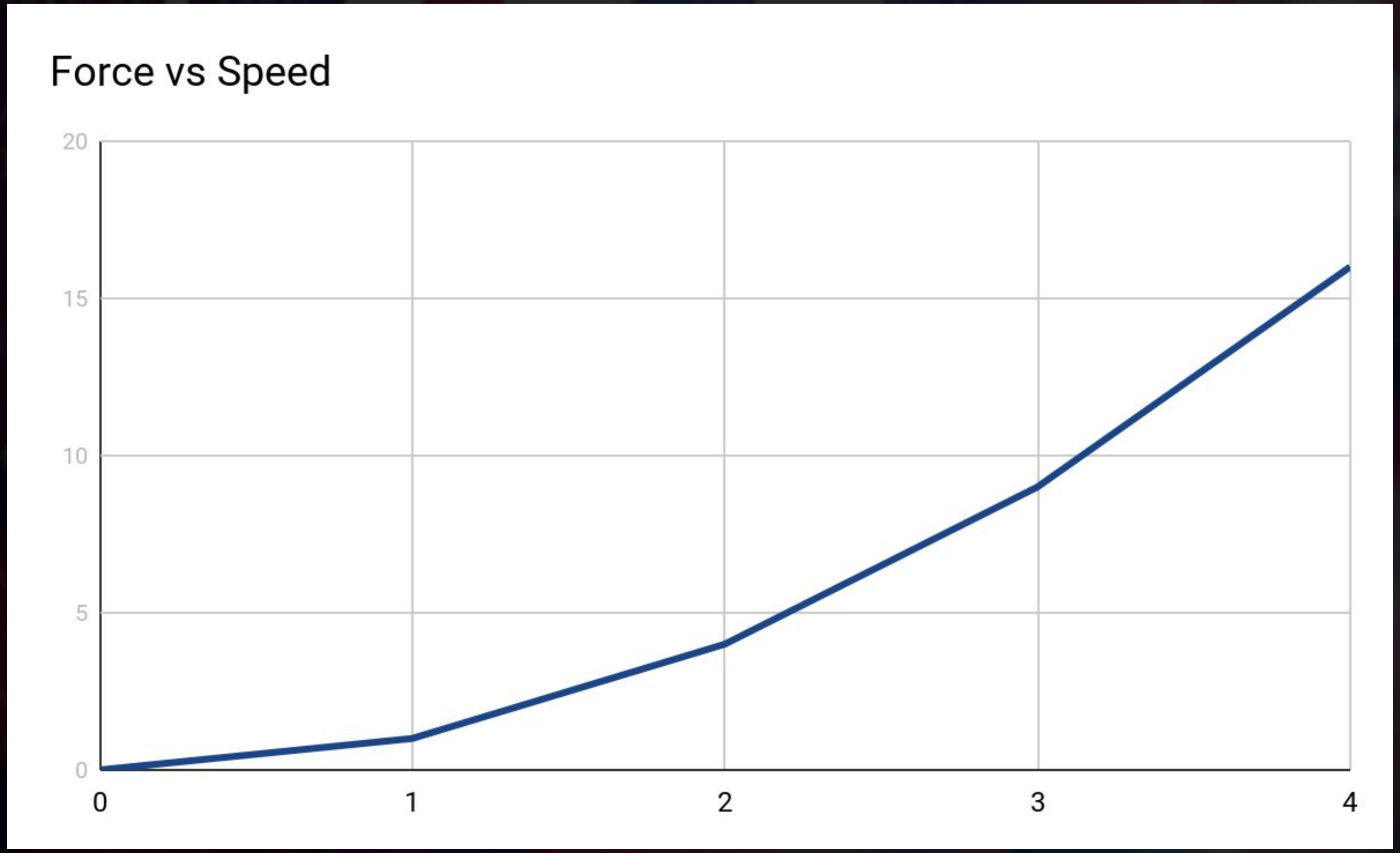


Simulating Air Resistance

Rotate Velocity

- Read the FQuat documentation
- Update the velocity
- Test
- Refactor.

Drag



Air Resistance Formula

$$\text{AirResistance} = -\,\text{Speed}^2 \times \text{DragCoefficient}$$

Add The Air Resistance

- Create a function to calculate resistance
- Use the formula from last slide
- Which direction should it be in?
- Sum the forces on the car.

Calculating Drag Coefficient

$$\text{AirResistance} = -\,\text{Speed}^2 \times \text{DragCoefficient}$$

$$\text{AirResistance} / \text{Speed}^2 = \text{DragCoefficient}$$

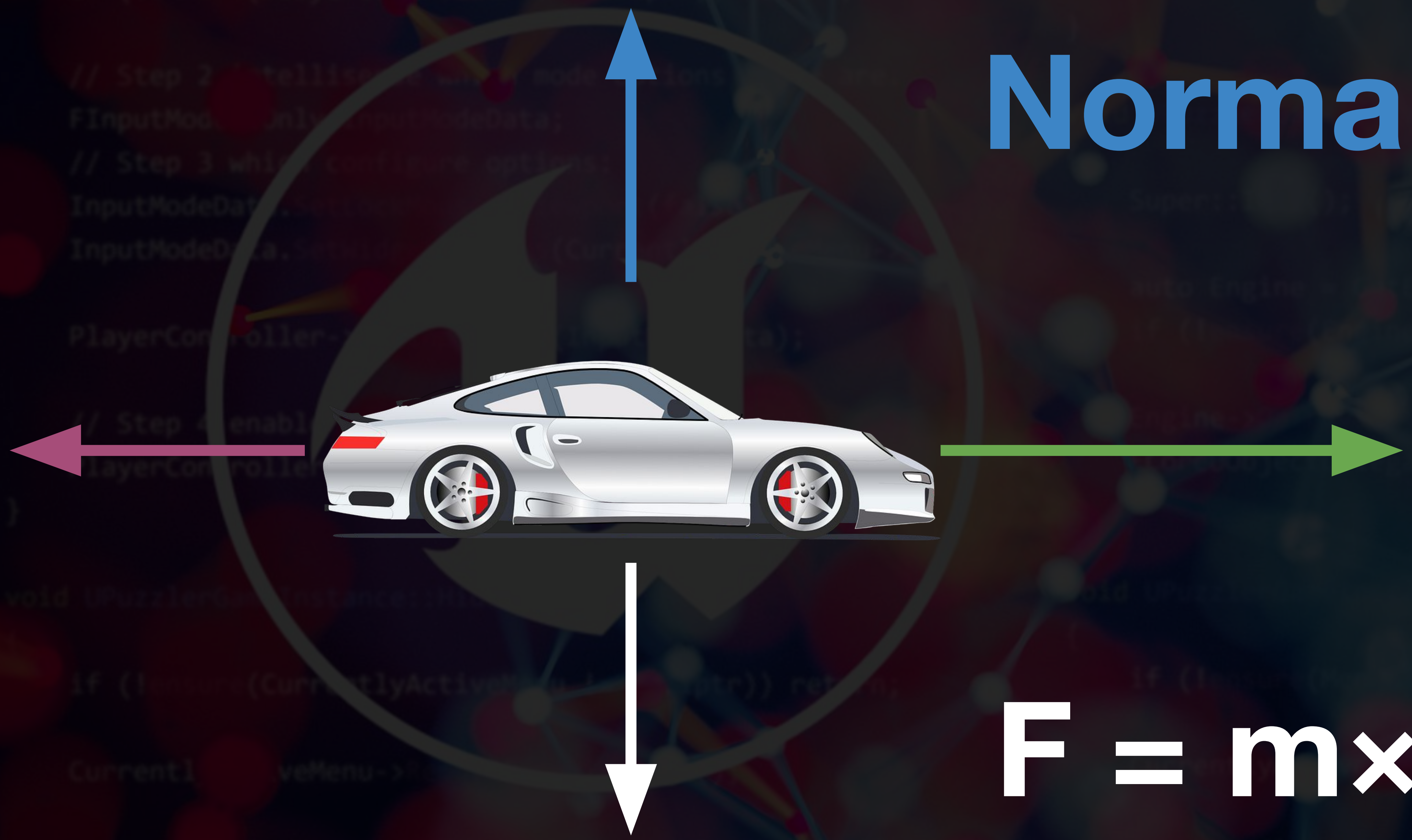
$$10{,}000 / 25^2 = \text{DragCoefficient}$$

Simulating Rolling Resistance

Rolling Resistance

$$\text{RollingResistance} = \text{RRCoefficient} \times \text{NormalForce}$$

Gravity



NormalForce

$F = m \times g$

Live Google Slides at bit.ly/multiplayerslides

Add Rolling Resistance

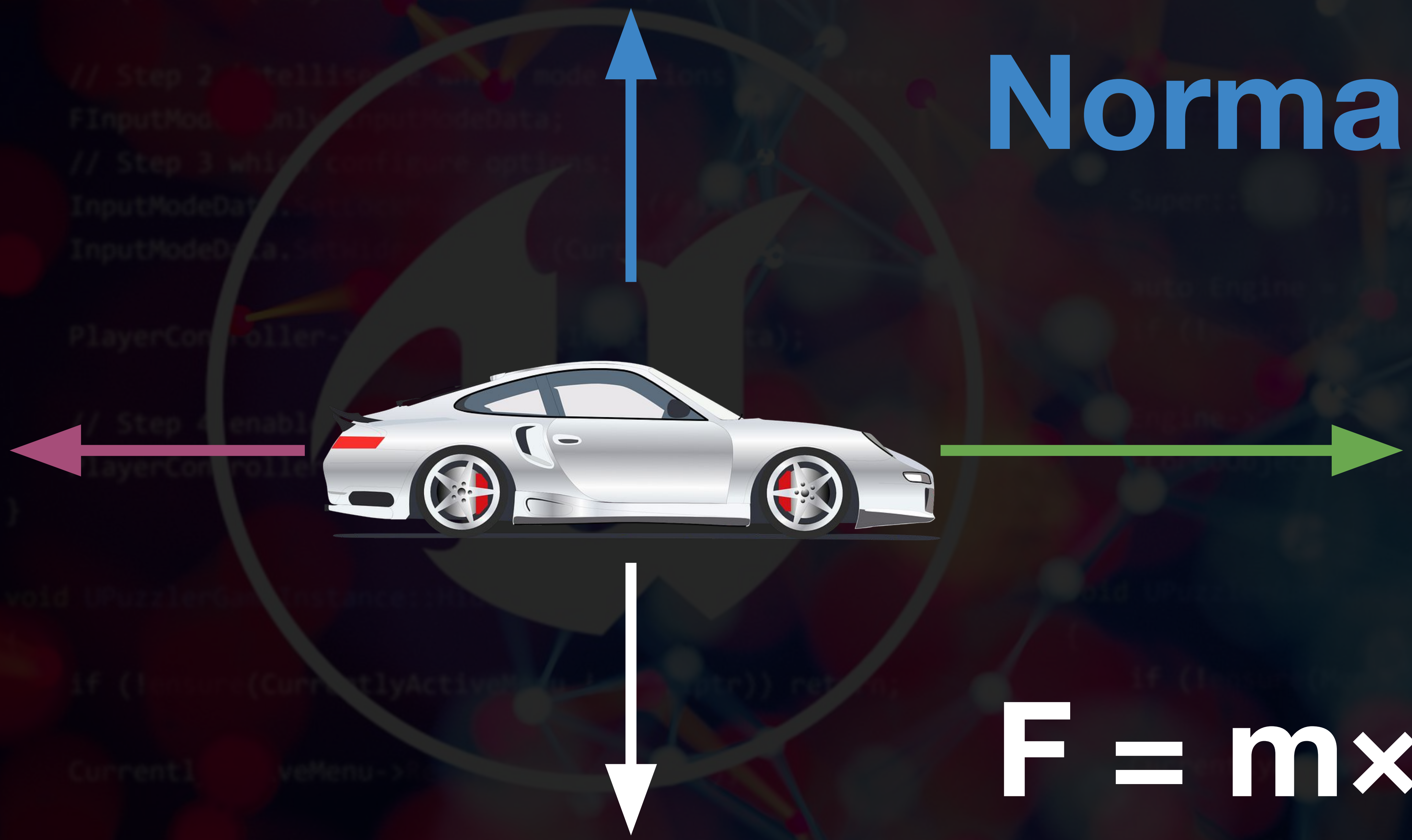
- Try different coefficients
- How long does it take to stop?
- Tweak!

Simulating Rolling Resistance

Rolling Resistance

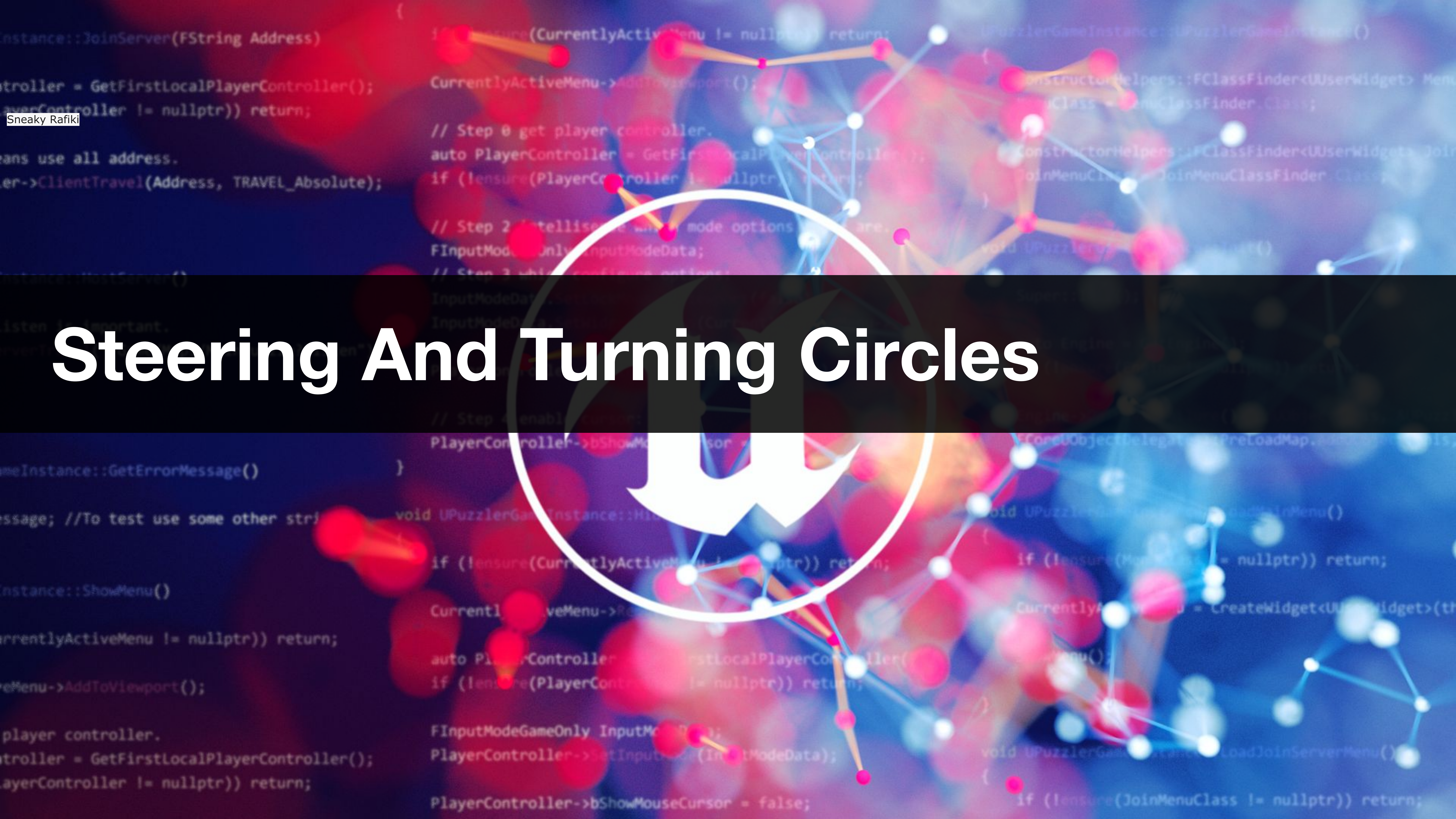
$$\text{RollingResistance} = \text{RRCoefficient} \times \text{NormalForce}$$

Gravity



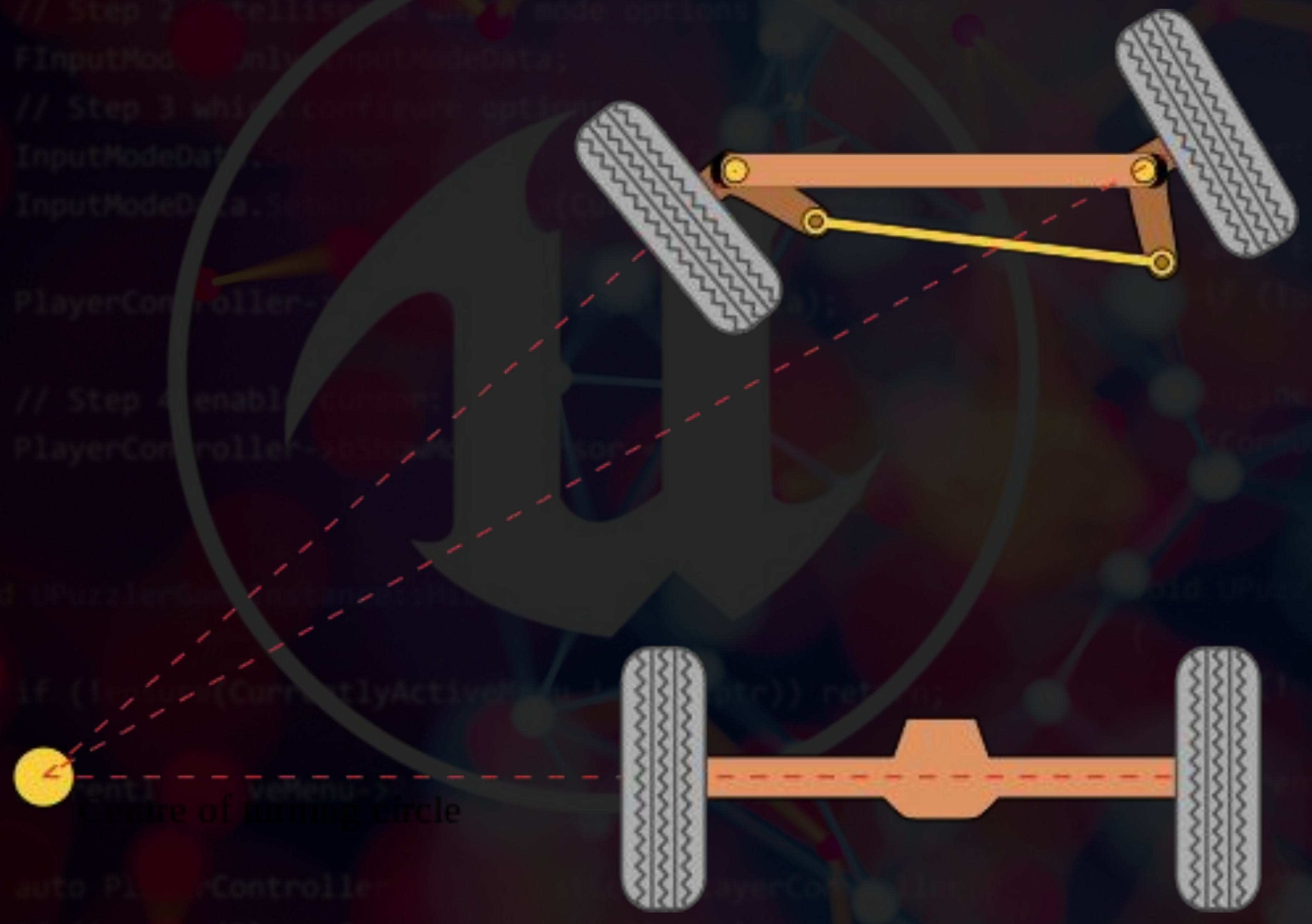
$F = m \times g$

Sneaky Rafiki

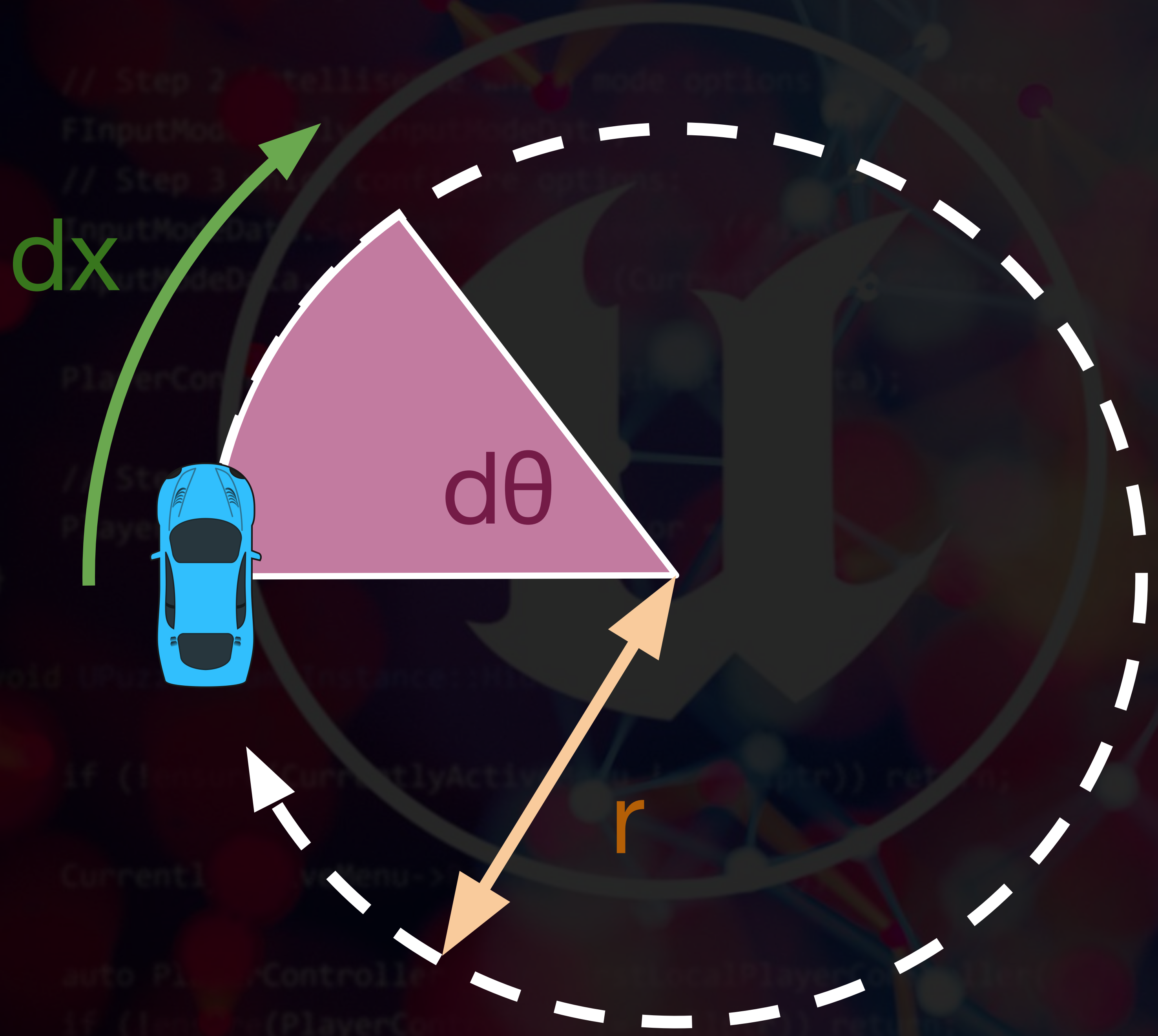


Steering And Turning Circles

How To Steer



Turning Circle

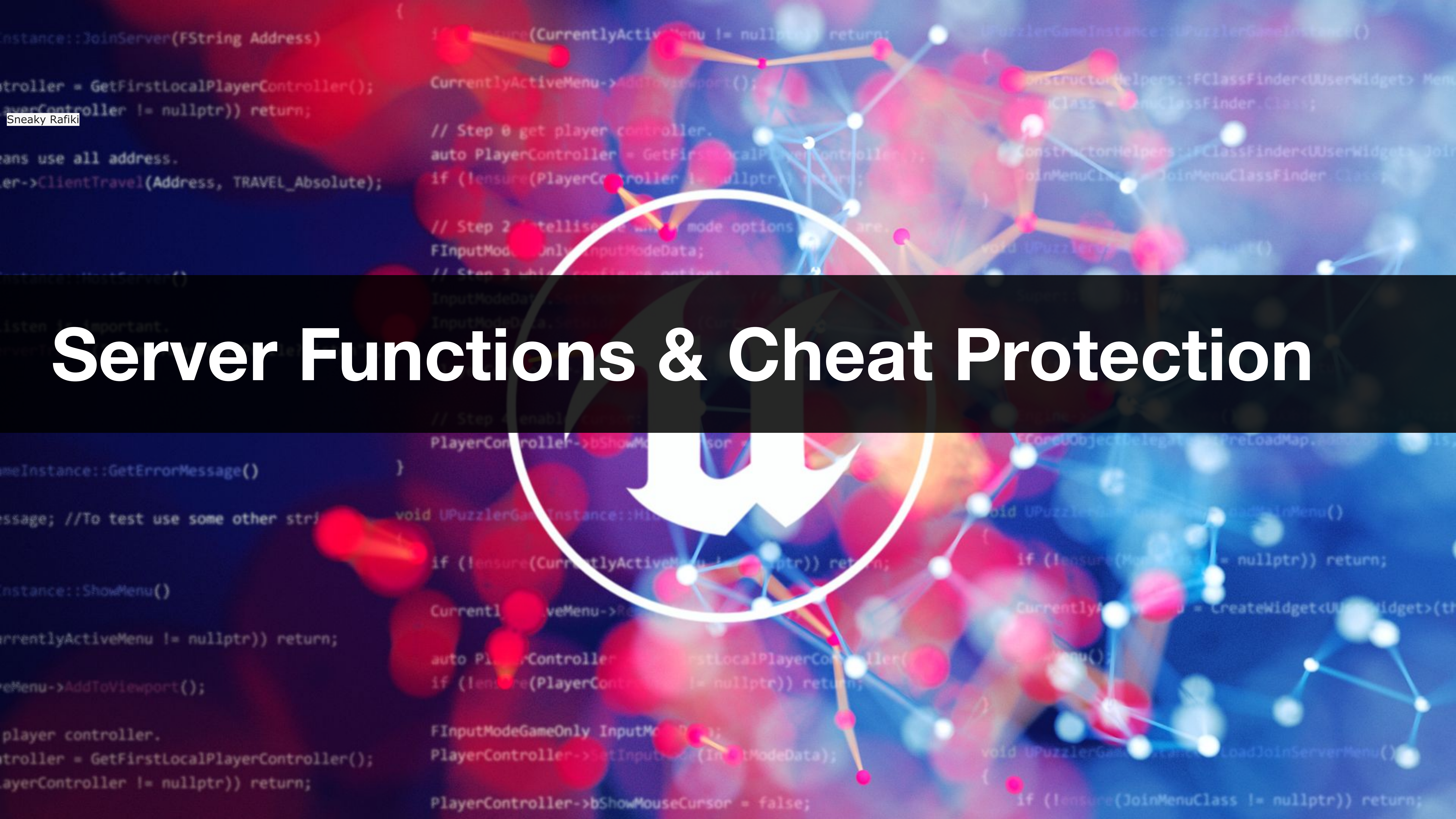


$$dx = d\theta \times r$$

Add A Turning Circle

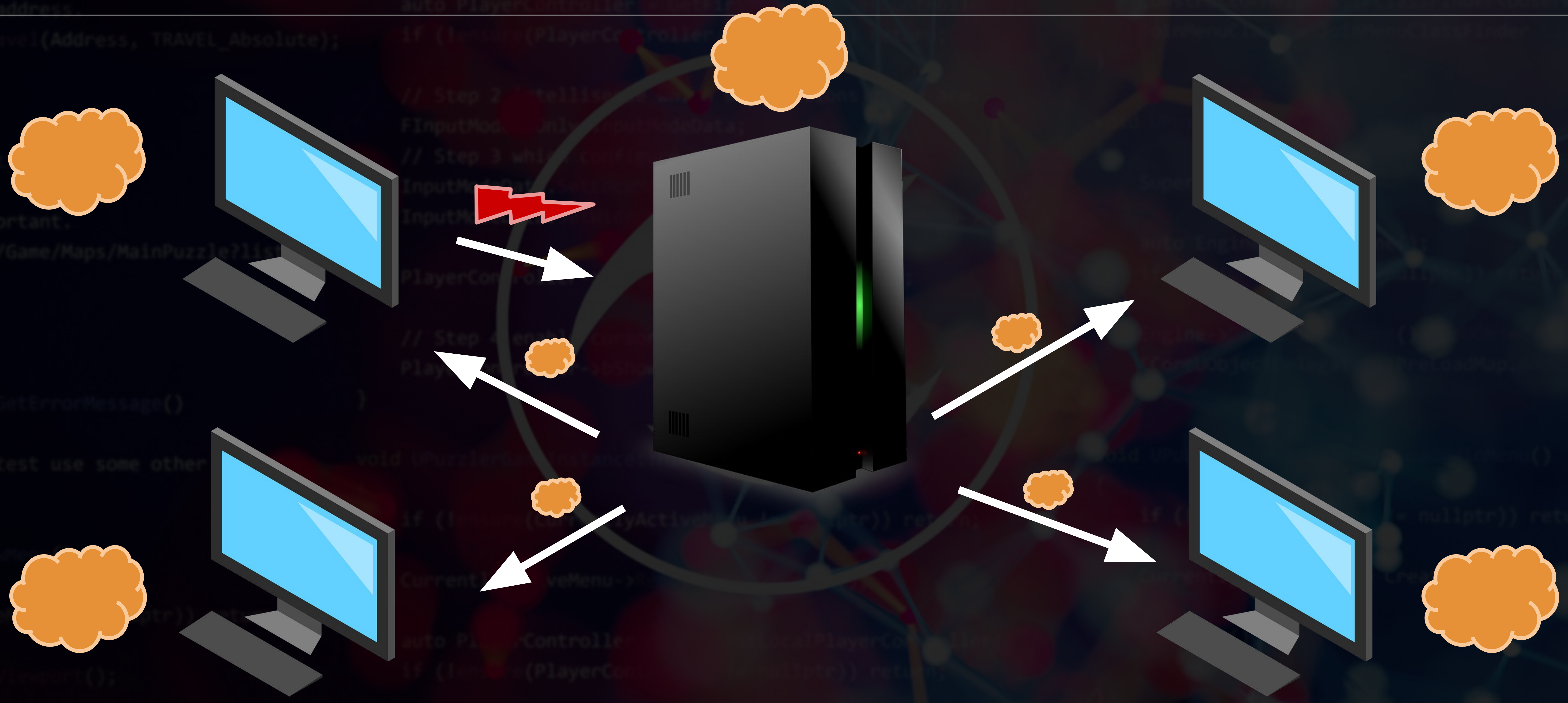
- Make the circle radius configurable
- Calculate the angle from the speed and radius
- What should happen when we aren’t moving?
- What about reversing?

Sneaky Rafiki



Server Functions & Cheat Protection

Client-Server



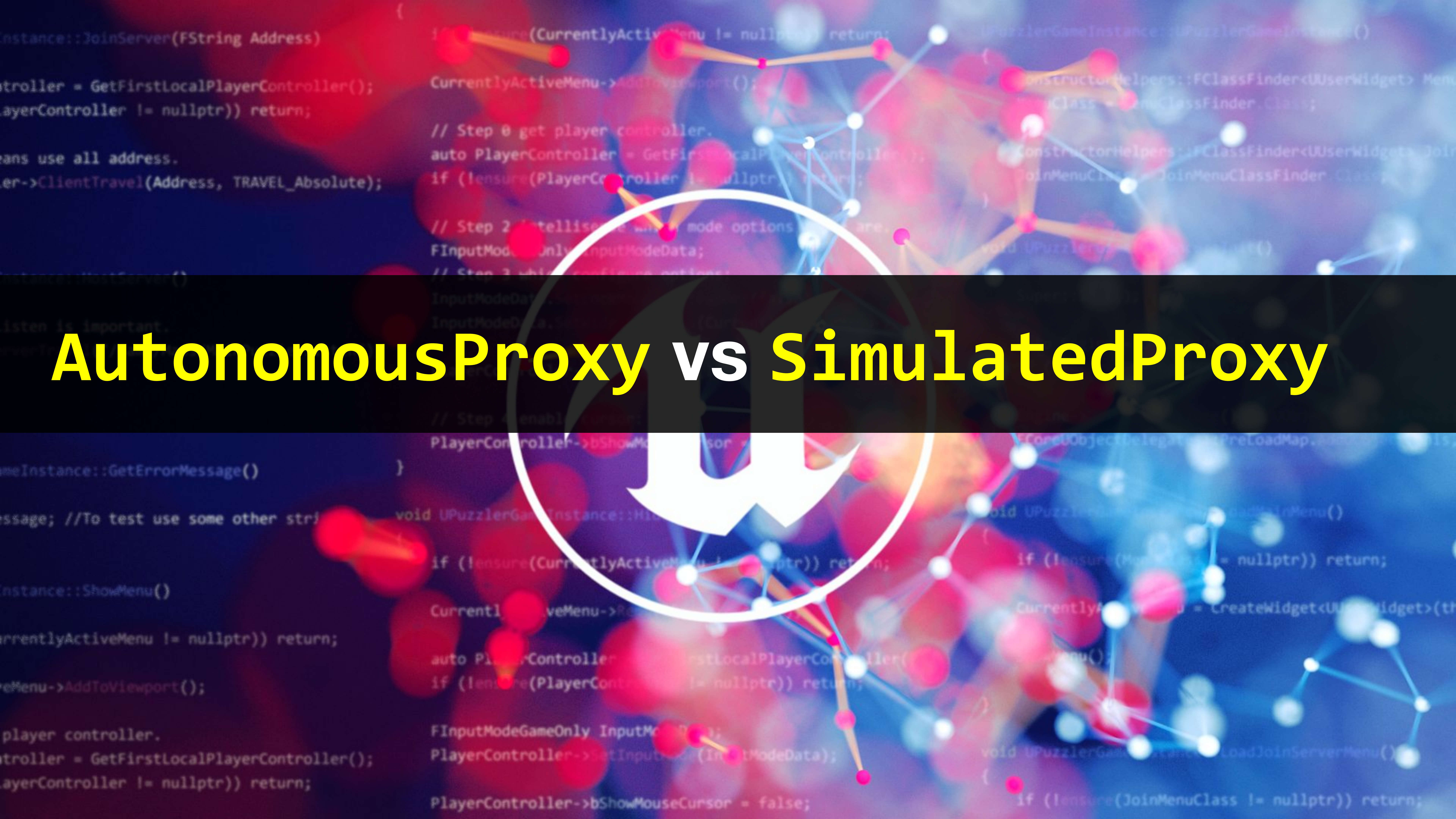
Live Google Slides at bit.ly/multiplayerslides

Cheat Protection

✓	✗
SetThrottle()	SetLocation()
Jump()	SetEnemyHealth()
Attack()	KillEnemy()

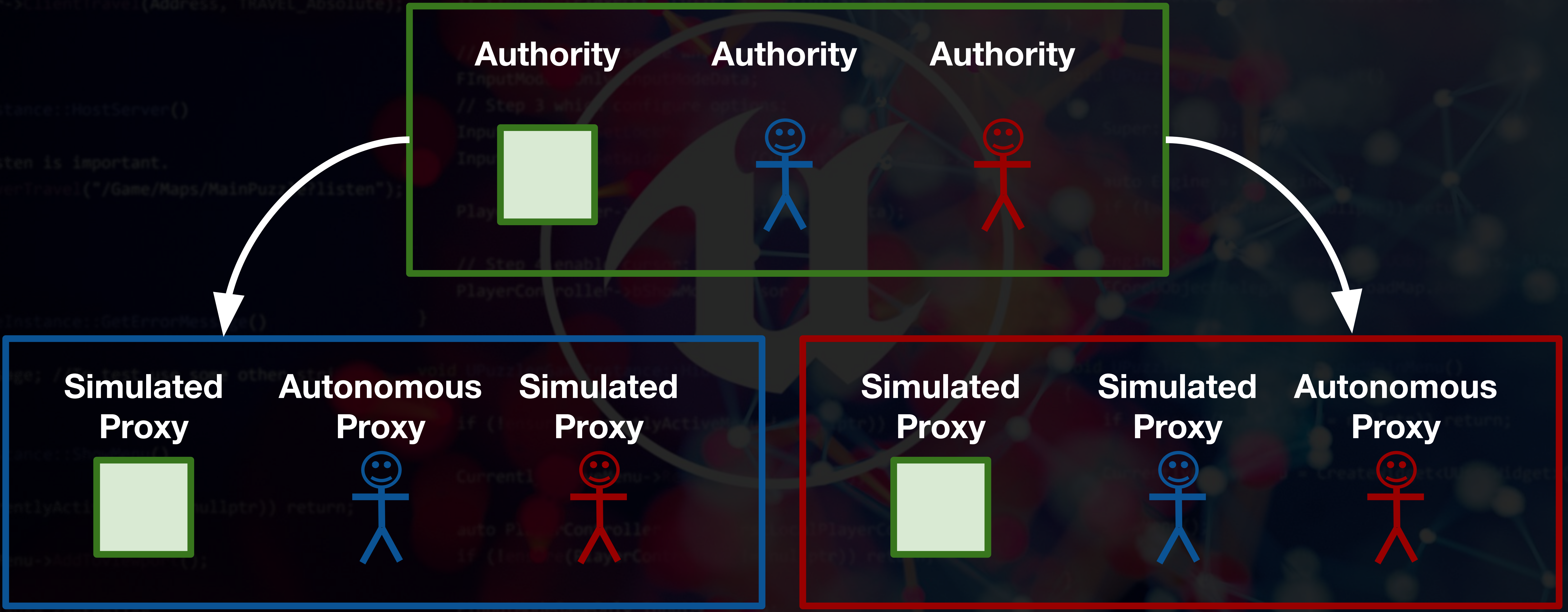
Cheat Protect The Game

- Implement steering
- Think of how the client might cheat
- Implement validation
- What happens if the validation fails?



AutonomousProxy vs SimulatedProxy

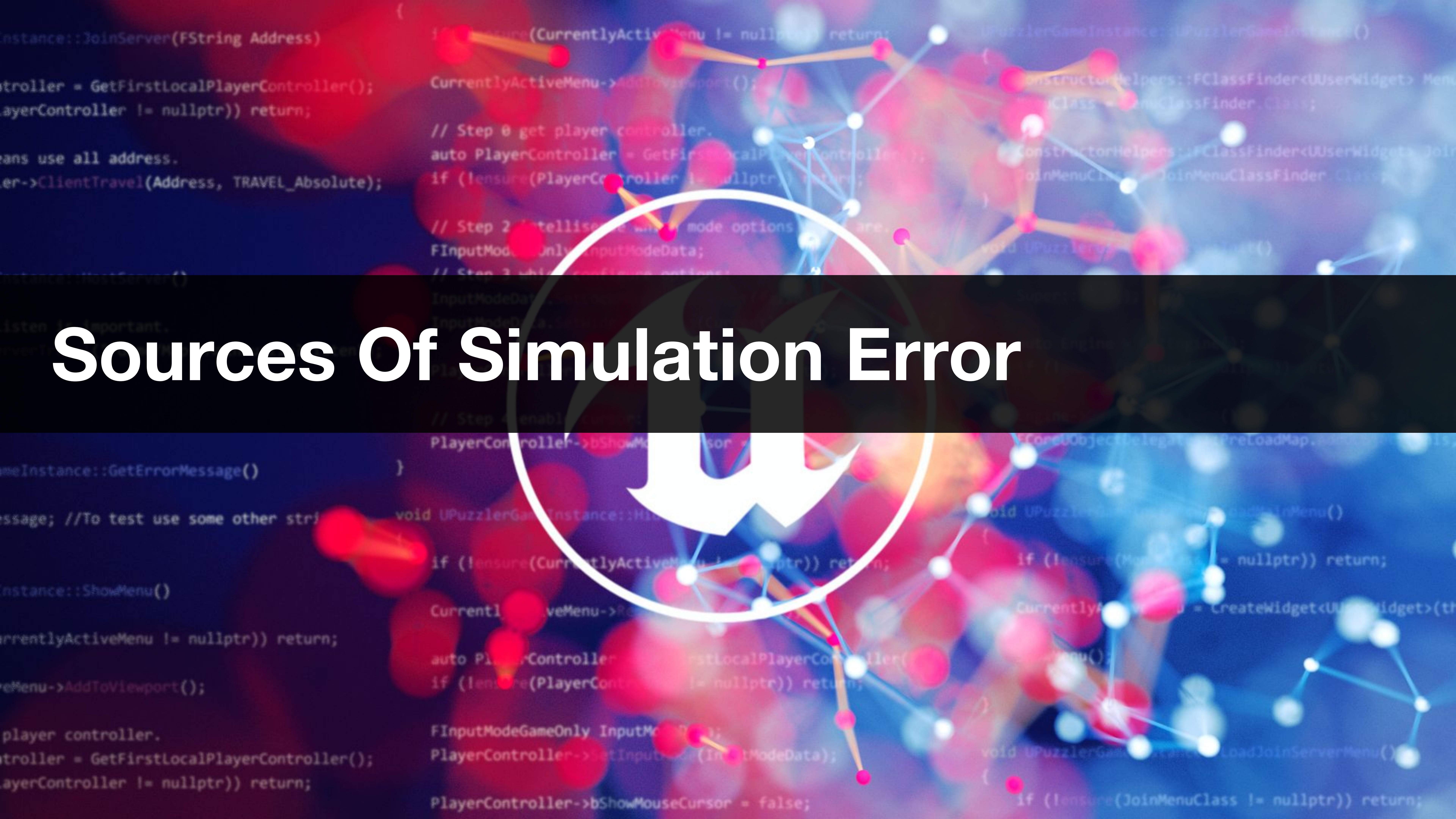
Actor Roles



Live Google Slides at bit.ly/multiplayerslides

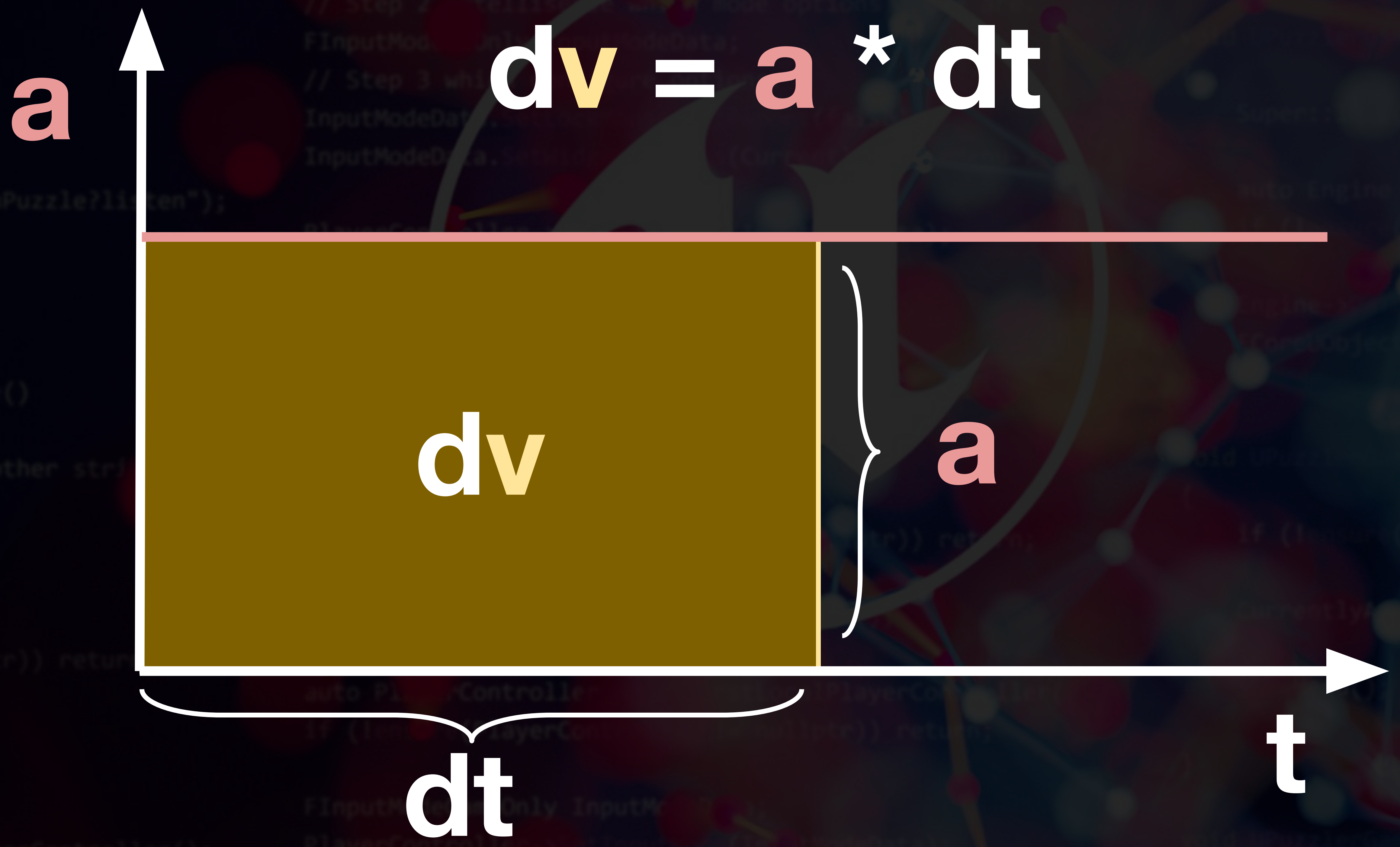
Update The AutonomousProxy

- Handle the bindings locally first
- Then pass them up to the server
- Test to check your positions coincide.

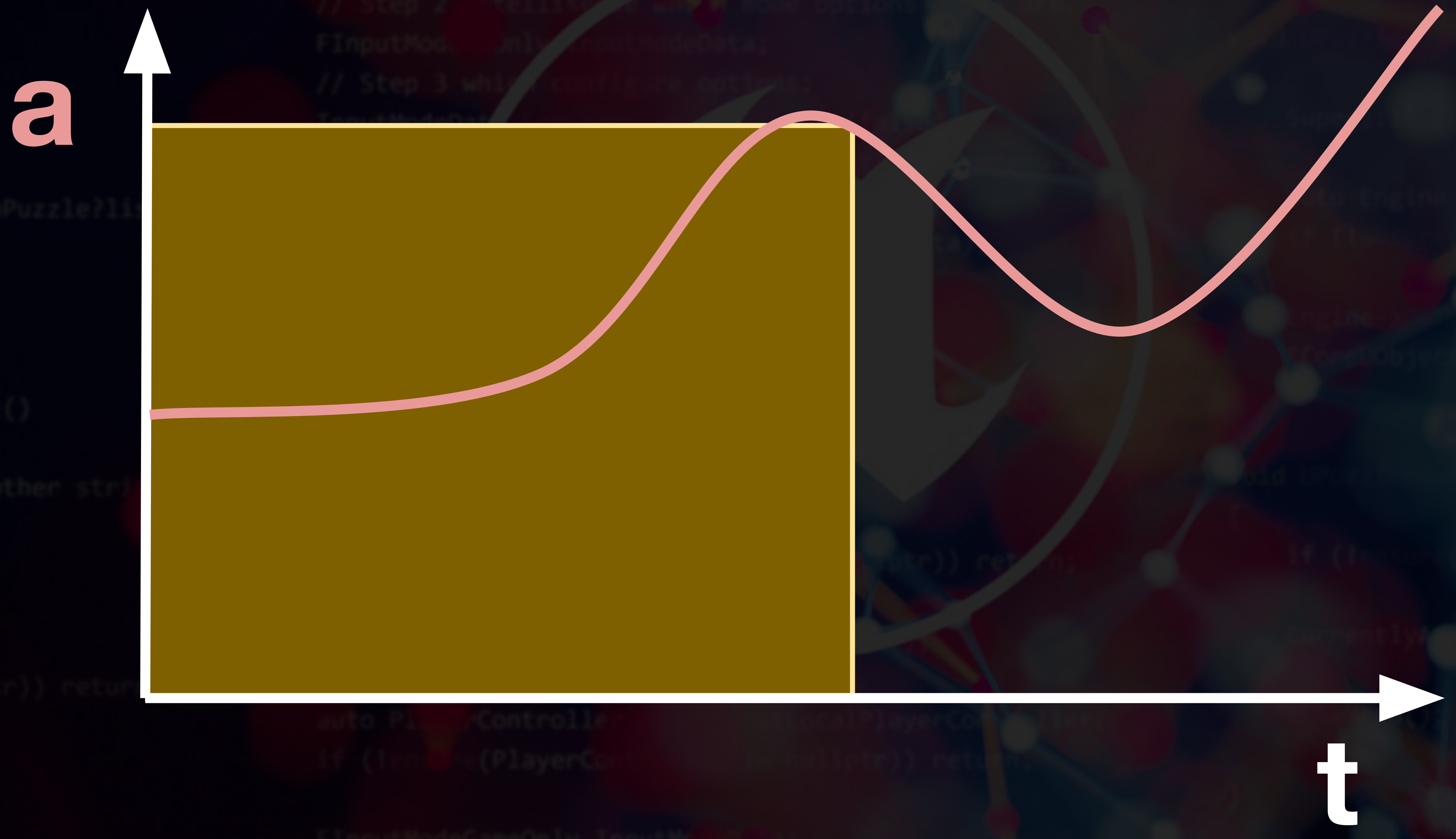


Sources Of Simulation Error

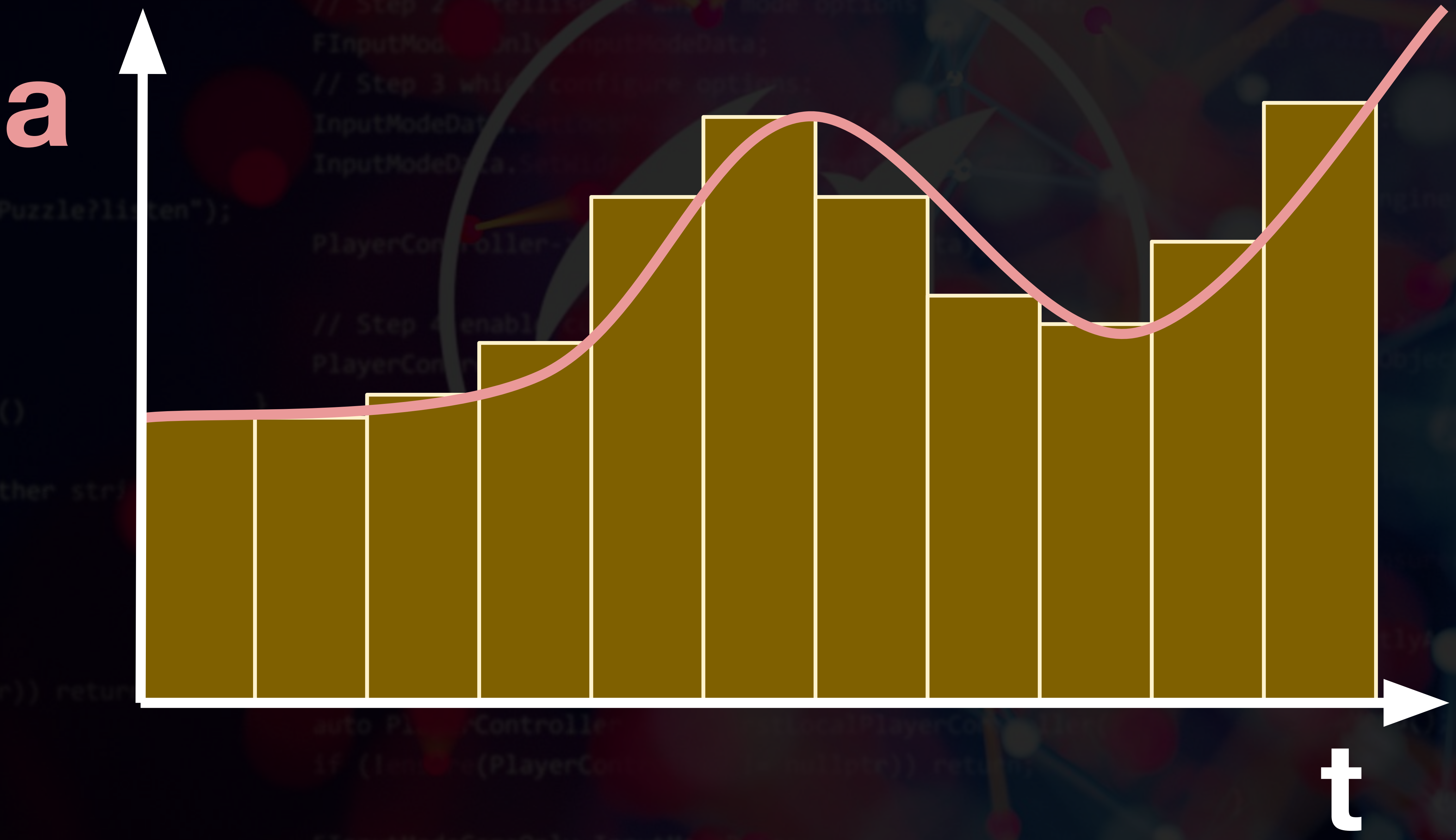
Numerical Integration



Numerical Integration: Errors

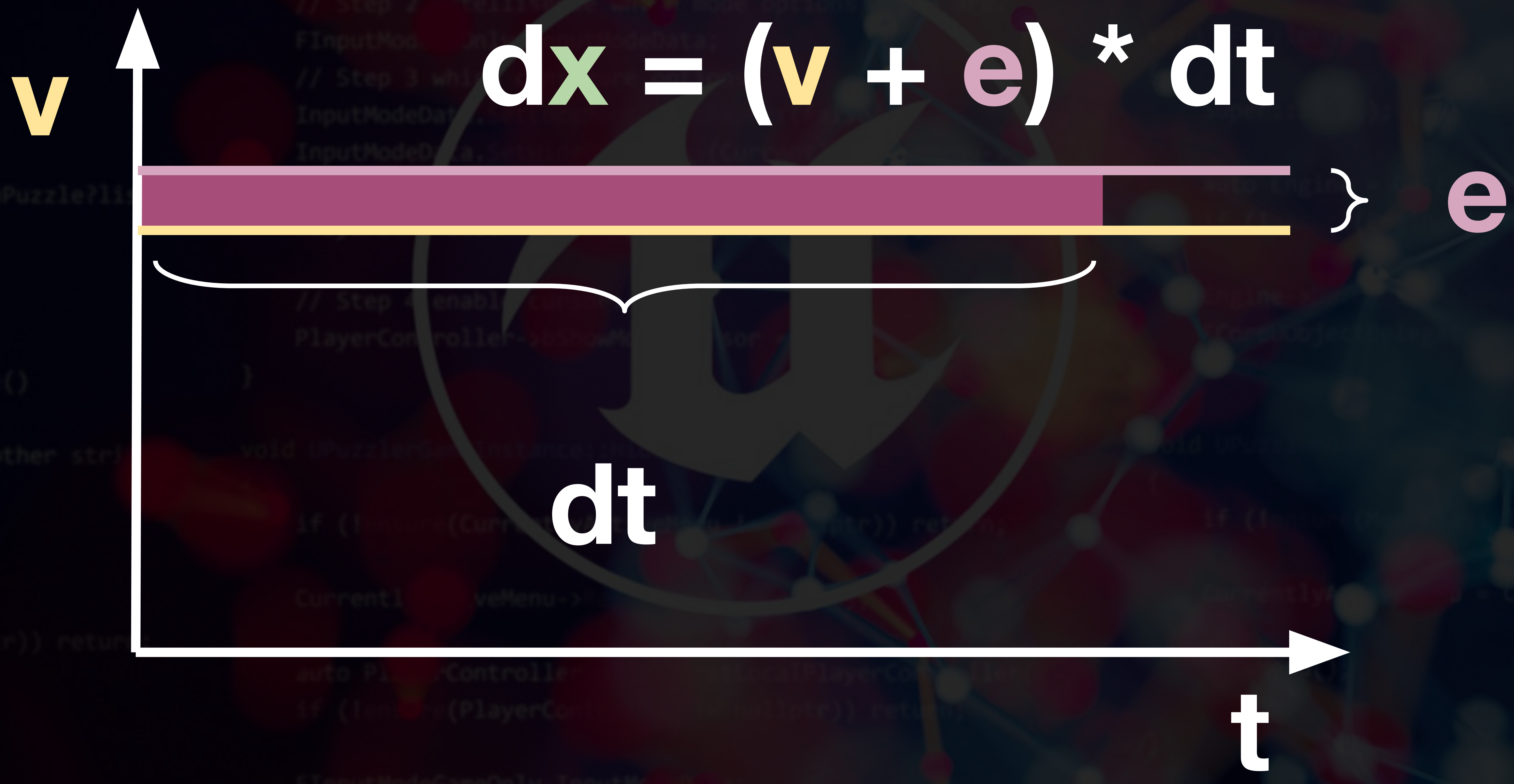


Numerical Integration: Errors

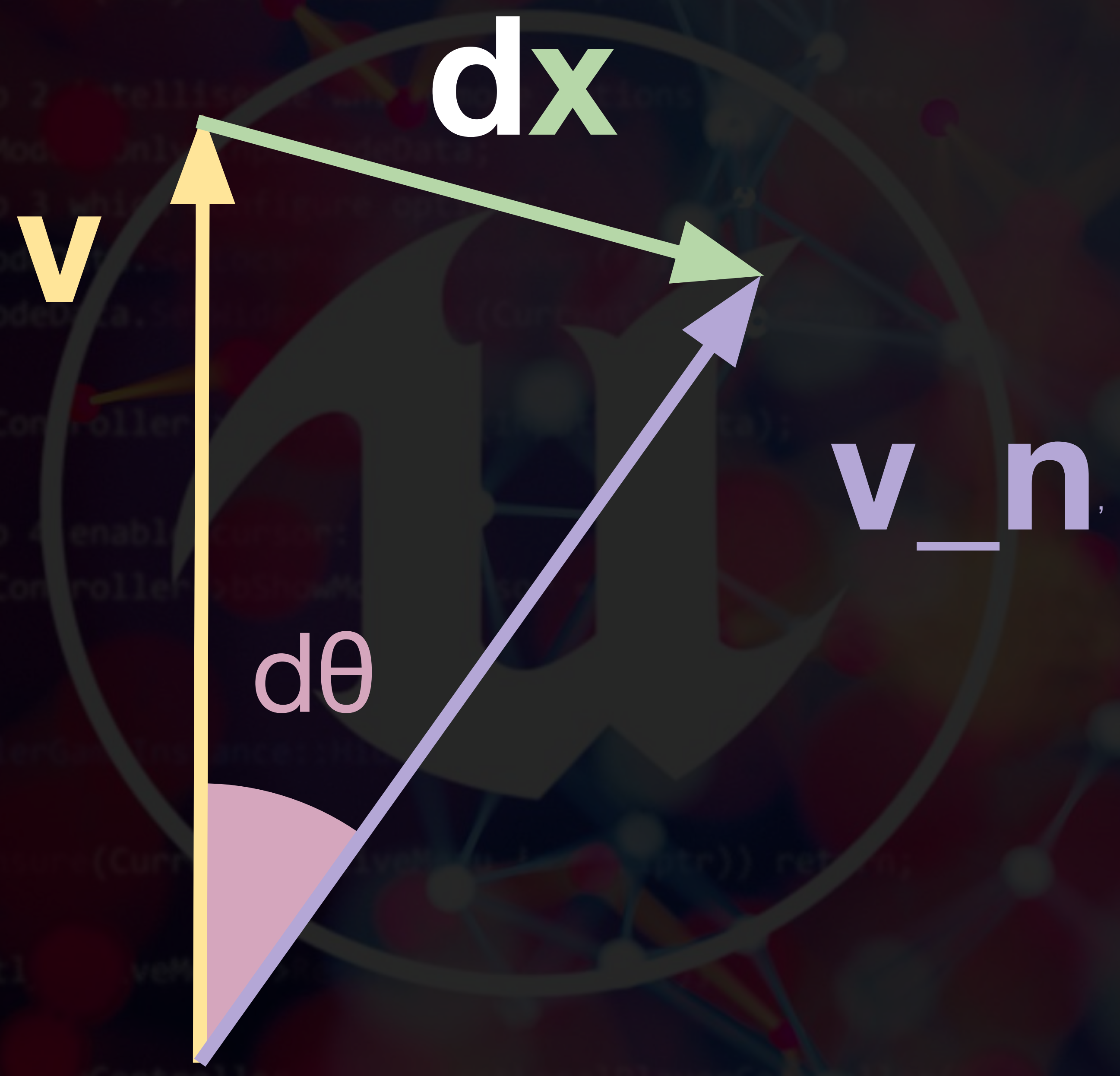


Integration: Error Magnification

$$dx = (v + e) * dt$$



Rotation: Error Magnification



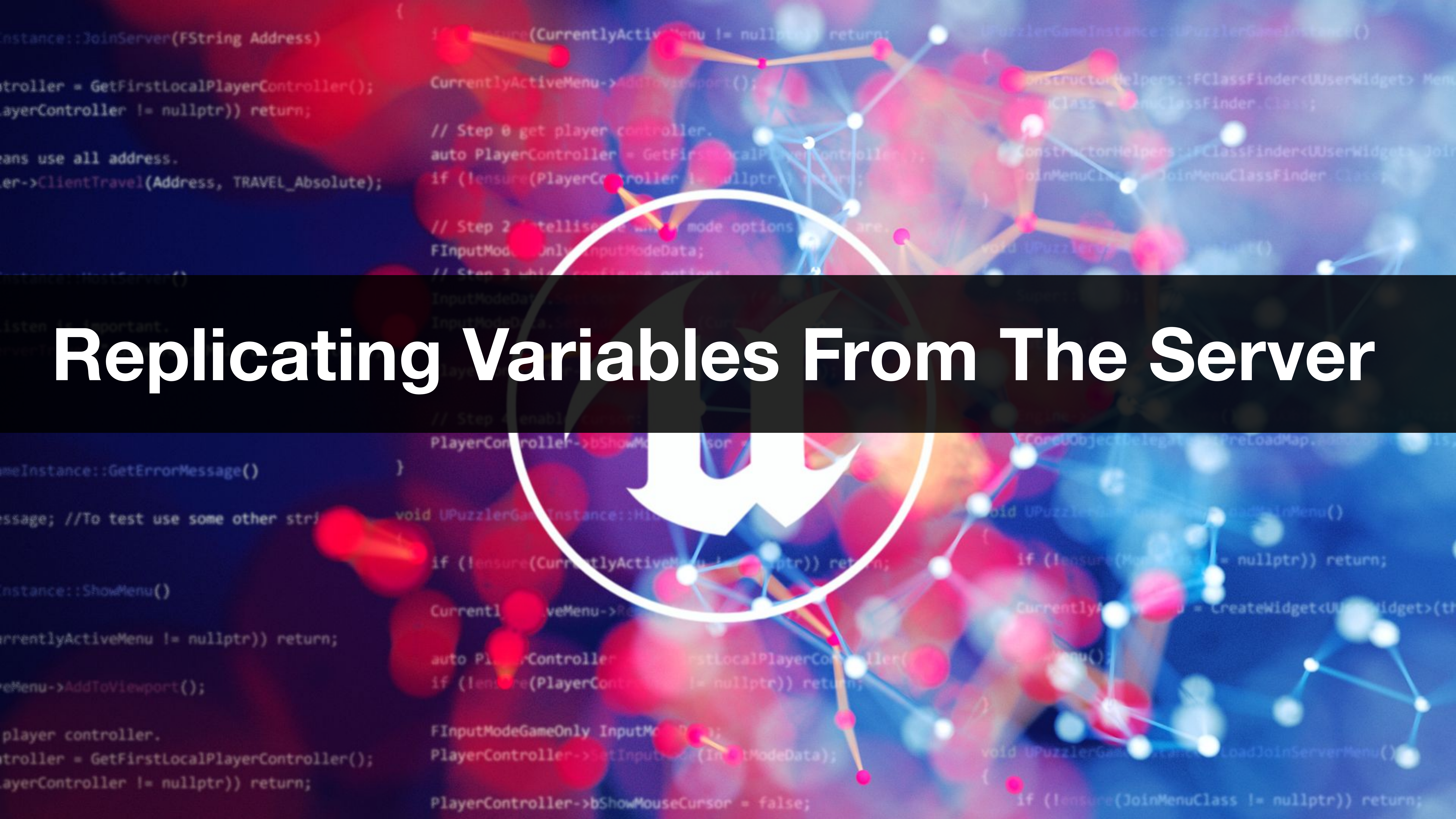
Live Google Slides at bit.ly/multiplayerslides

How Could You Solve it?

- Review the sources of errors.
- Make a list of 3 potential solutions.
- Write the pros and cons of each.
- Share on the community site.
- Comment on somebody else’s share.

Approaches

1. Synchronise velocity from the server
2. Overwrite location and rotation from the server
3. Simulate with a fixed time step.



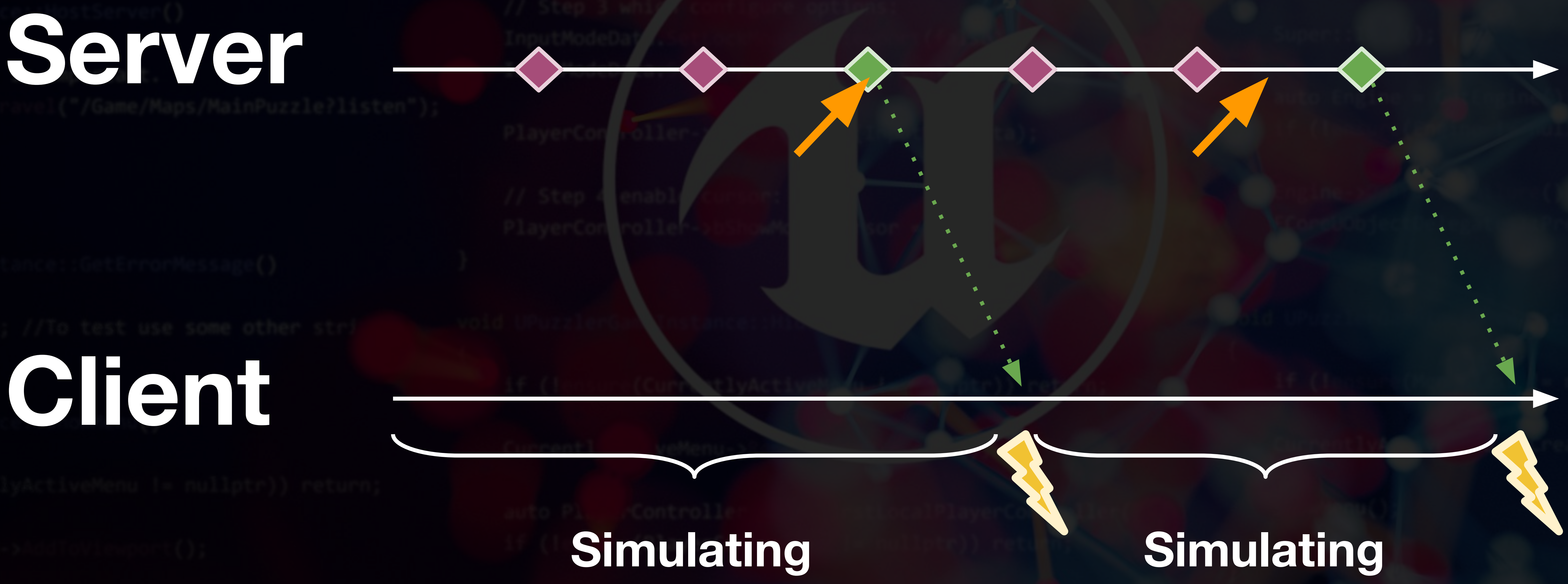
Replicating Variables From The Server

Set And Get The Position

- Set on the server
- Get on all clients
- Test!
- Repeat for rotation.

Triggering Code On Replication

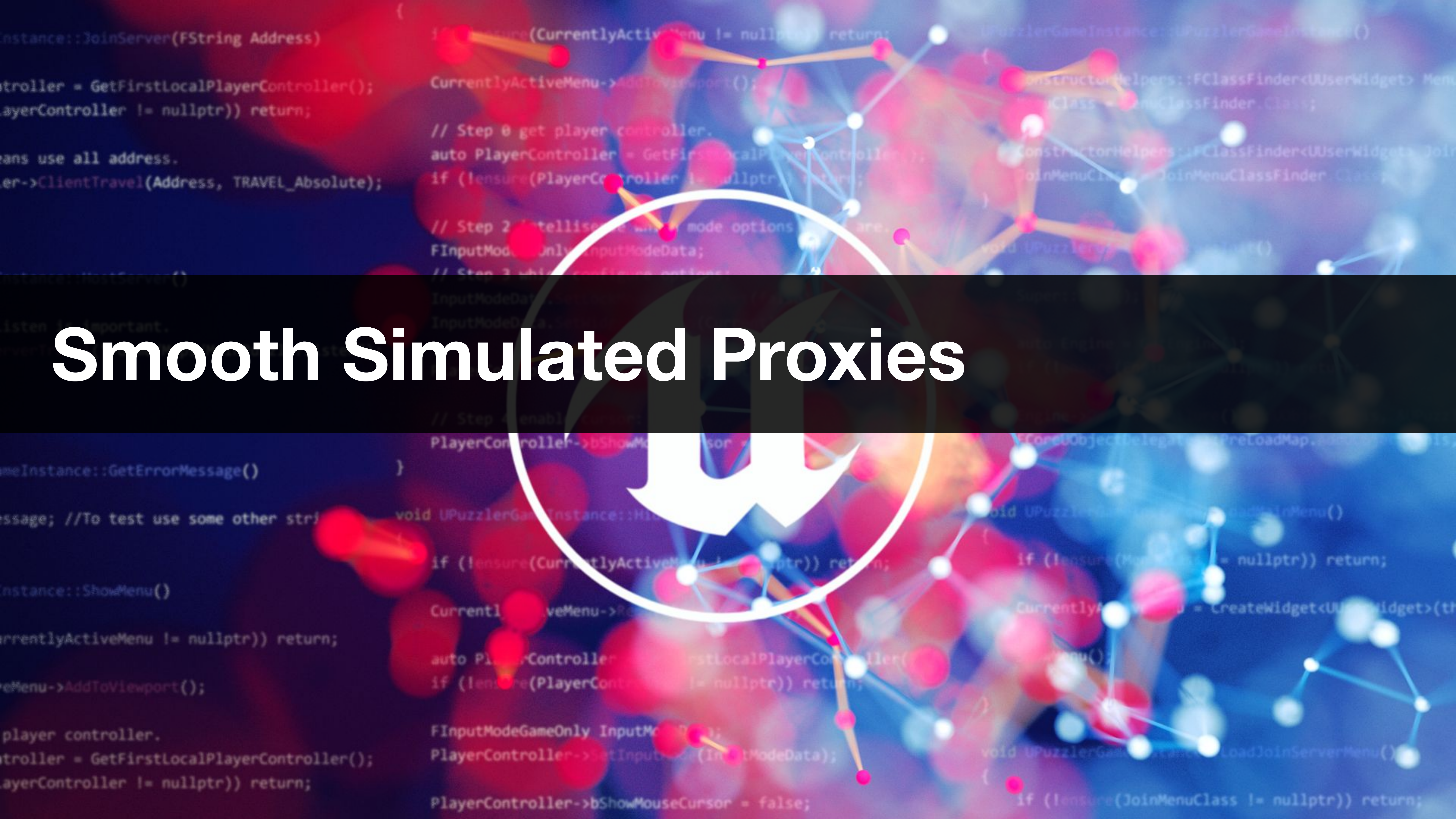
How Replication Works



Live Google Slides at bit.ly/multiplayerslides

Simulate Between Updates

- Replicate the transform for simplicity
- Ensure we are simulating locally
- Only overwrite when there is an update.



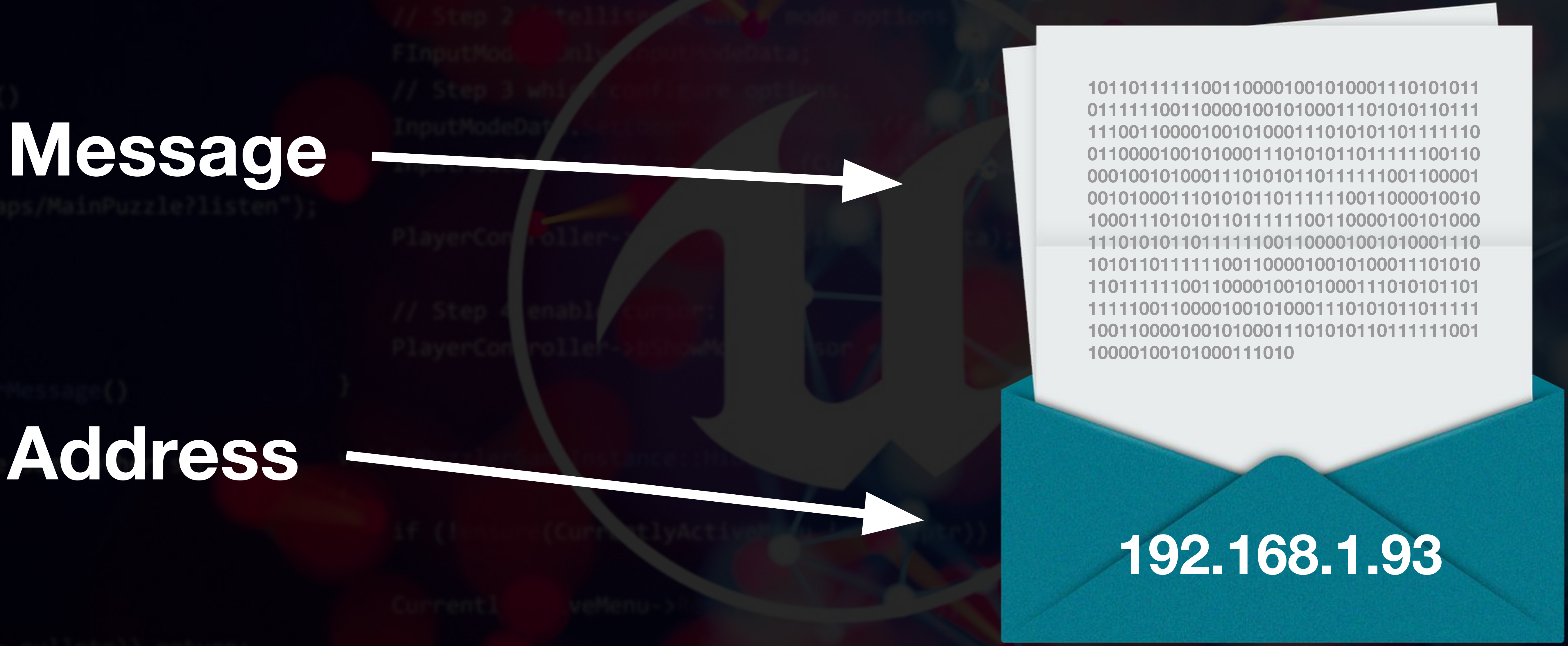
Smooth Simulated Proxies

Make It Smooth

- Why is the motion still jerky?
- What else could you replicate?
- Can you do any better?

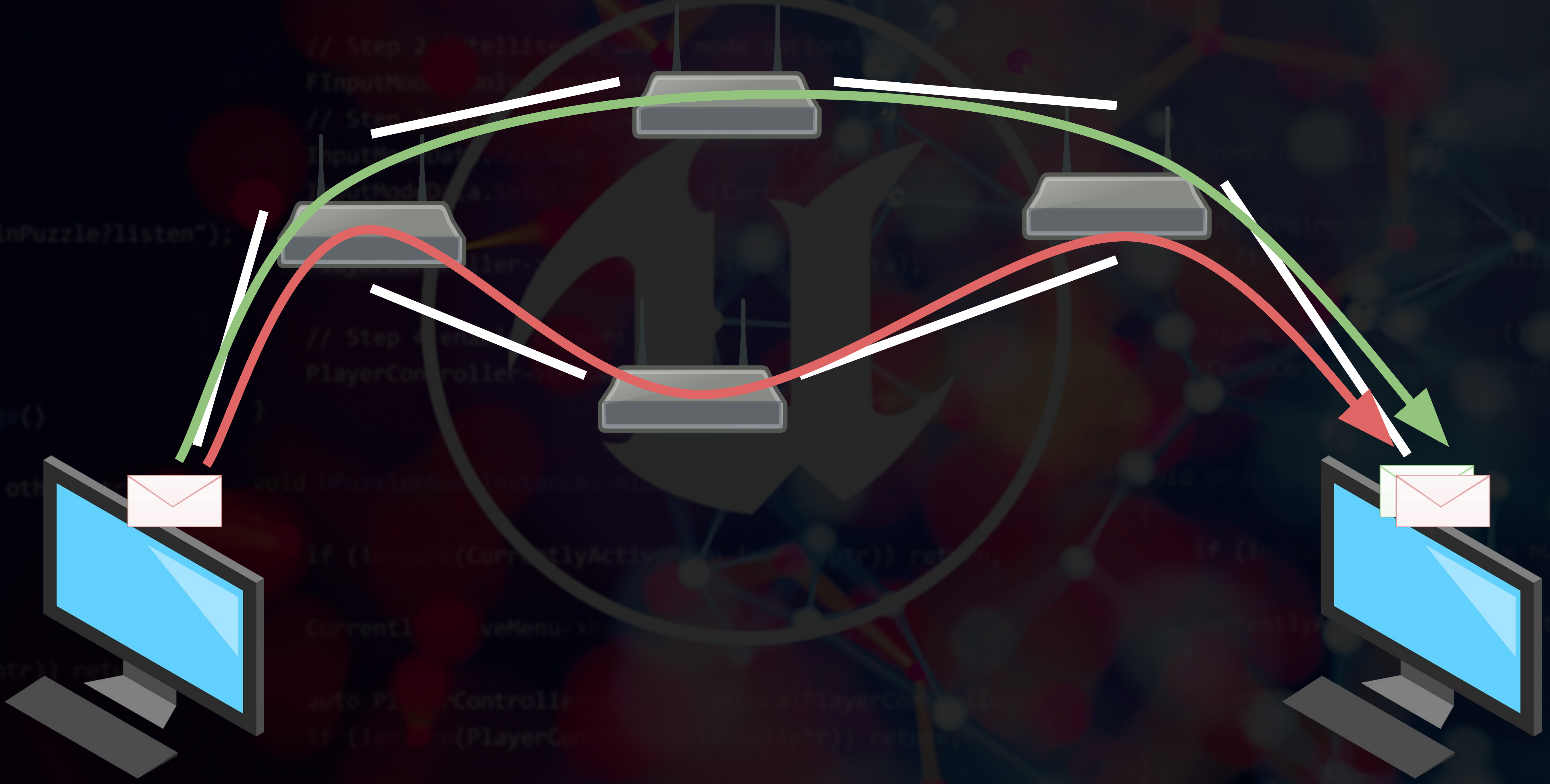
Simulating Lag And Packet Loss

Packets



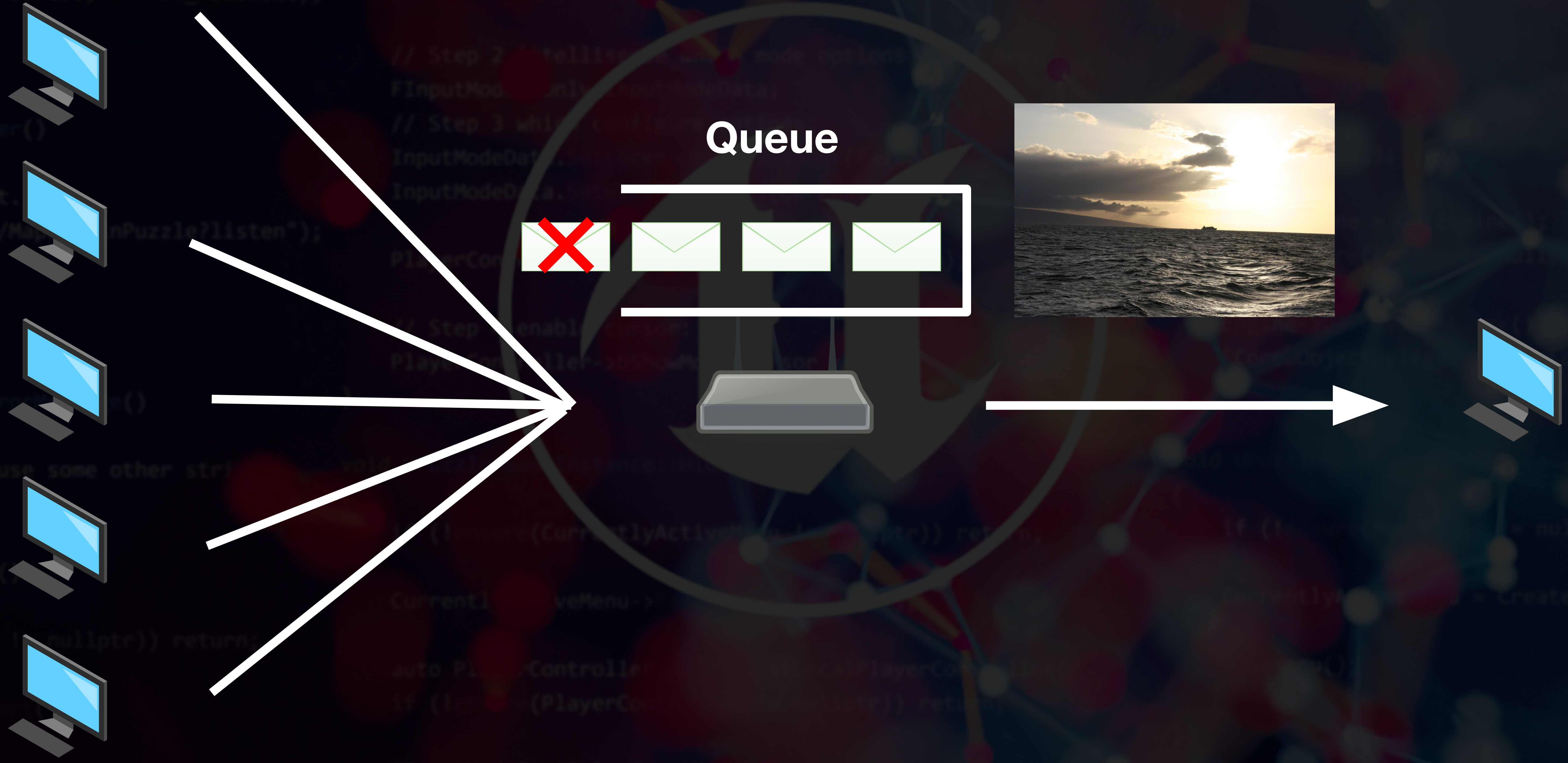
Live Google Slides at bit.ly/multiplayerslides

Sending Packets

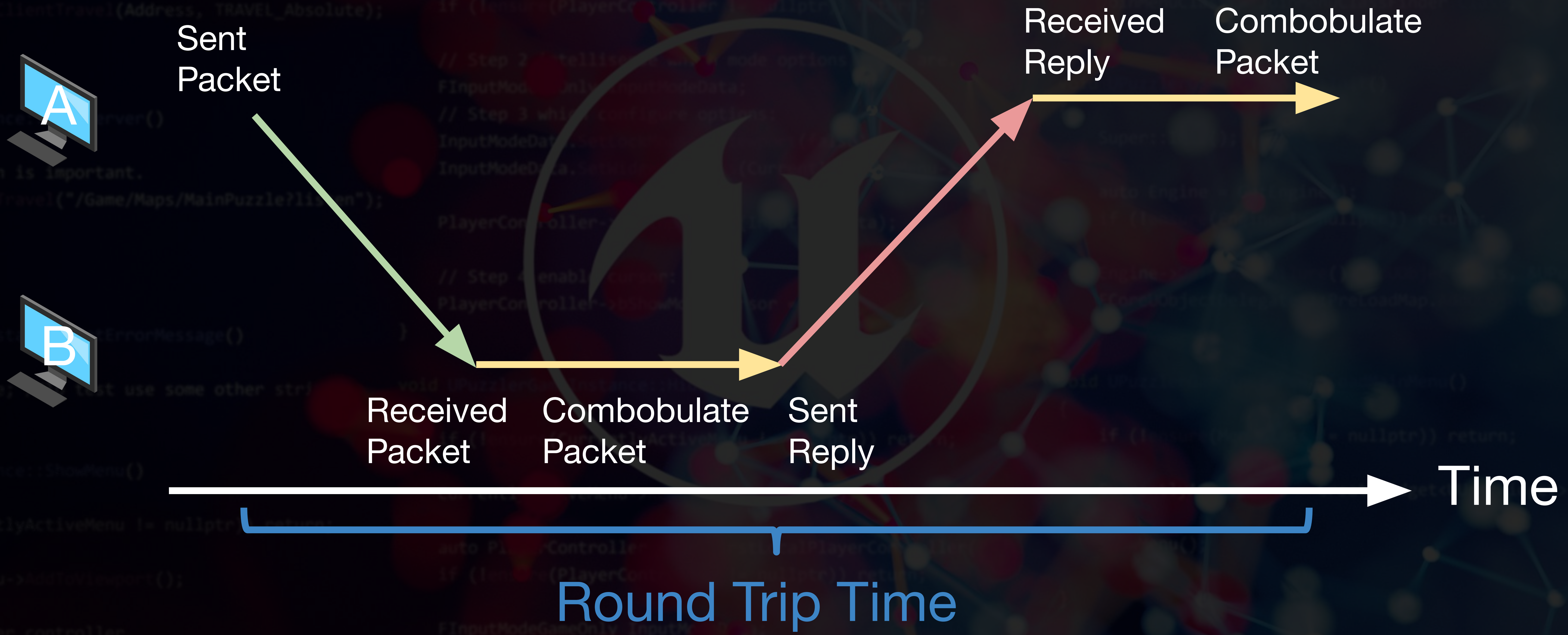


Live Google Slides at bit.ly/multiplayerslides

Best Effort Delivery



Lag

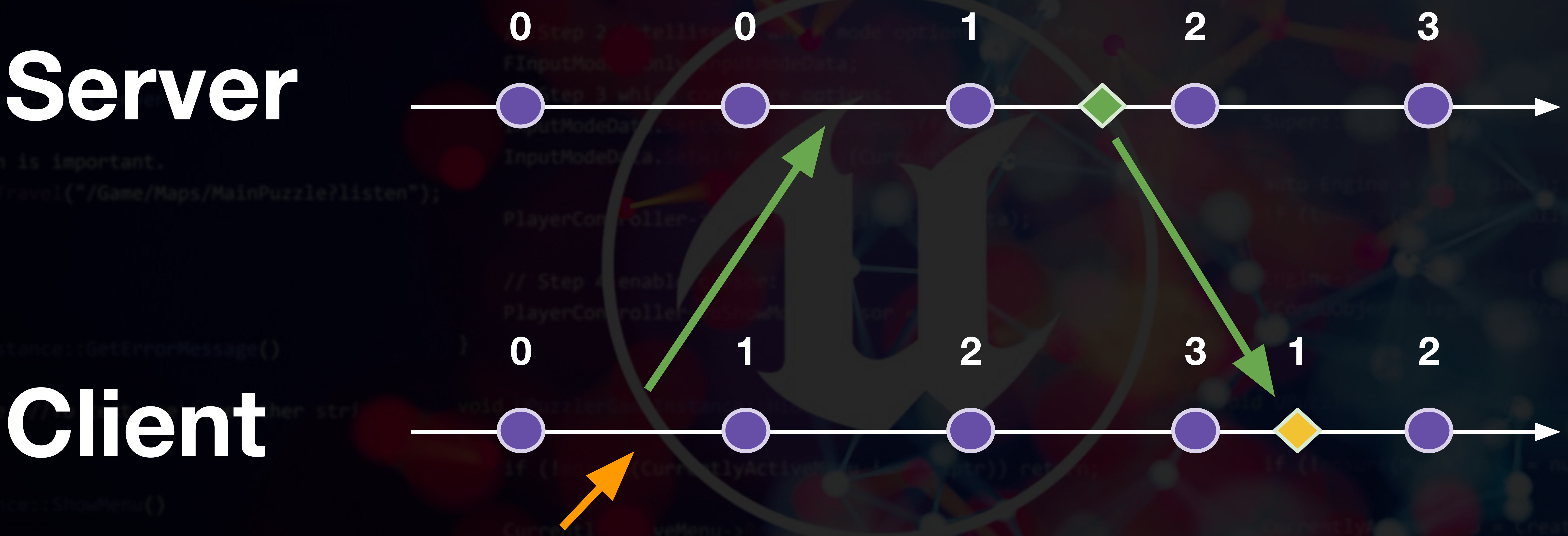


Live Google Slides at bit.ly/multiplayerslides

Why is the car glitching?

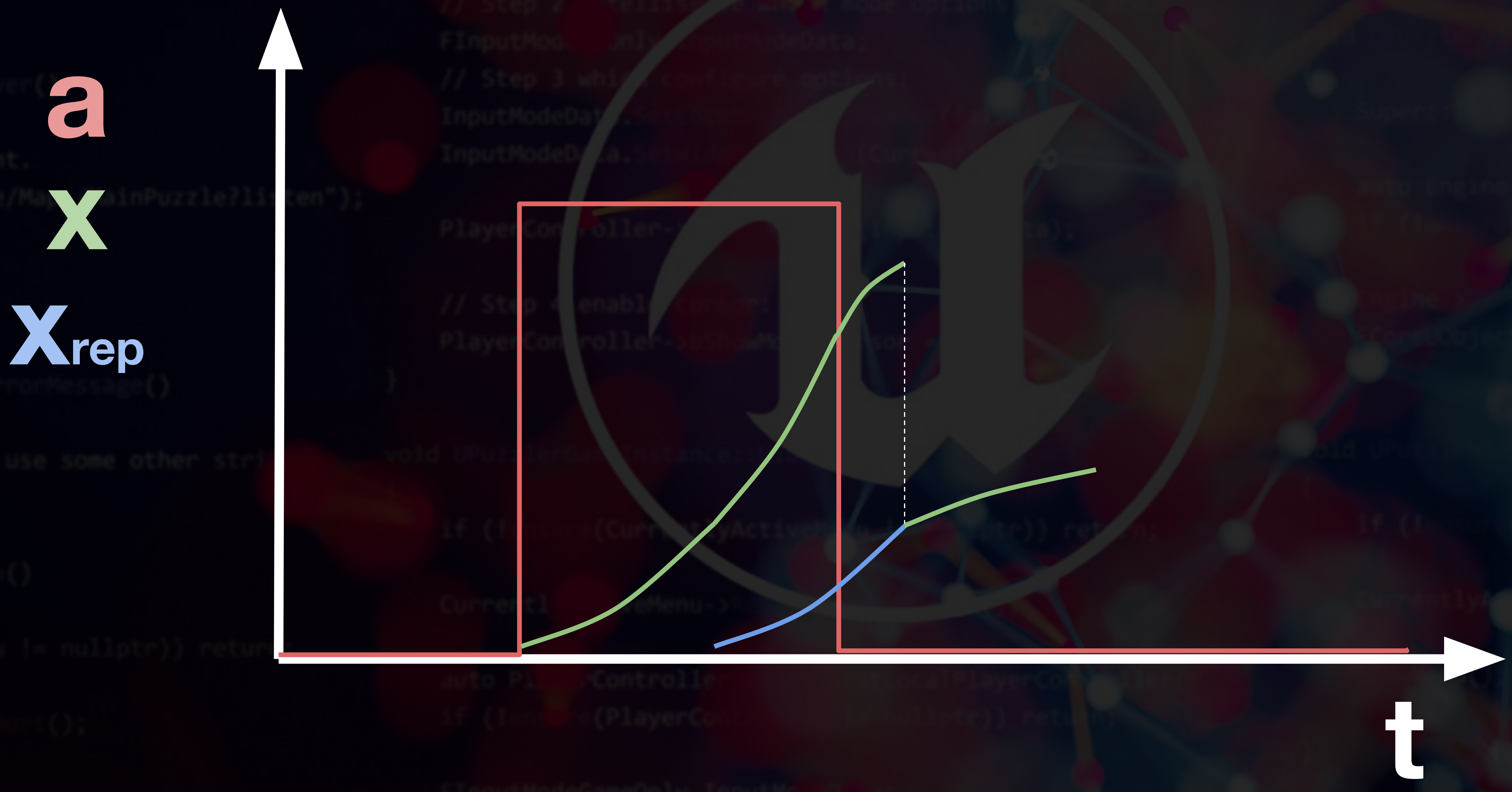
- Enable lag for yourself
- Play around with the game
- Take a look at what the server is seeing
- Can you explain why the game is glitching?
- Write up your explanation

Lag Glitching



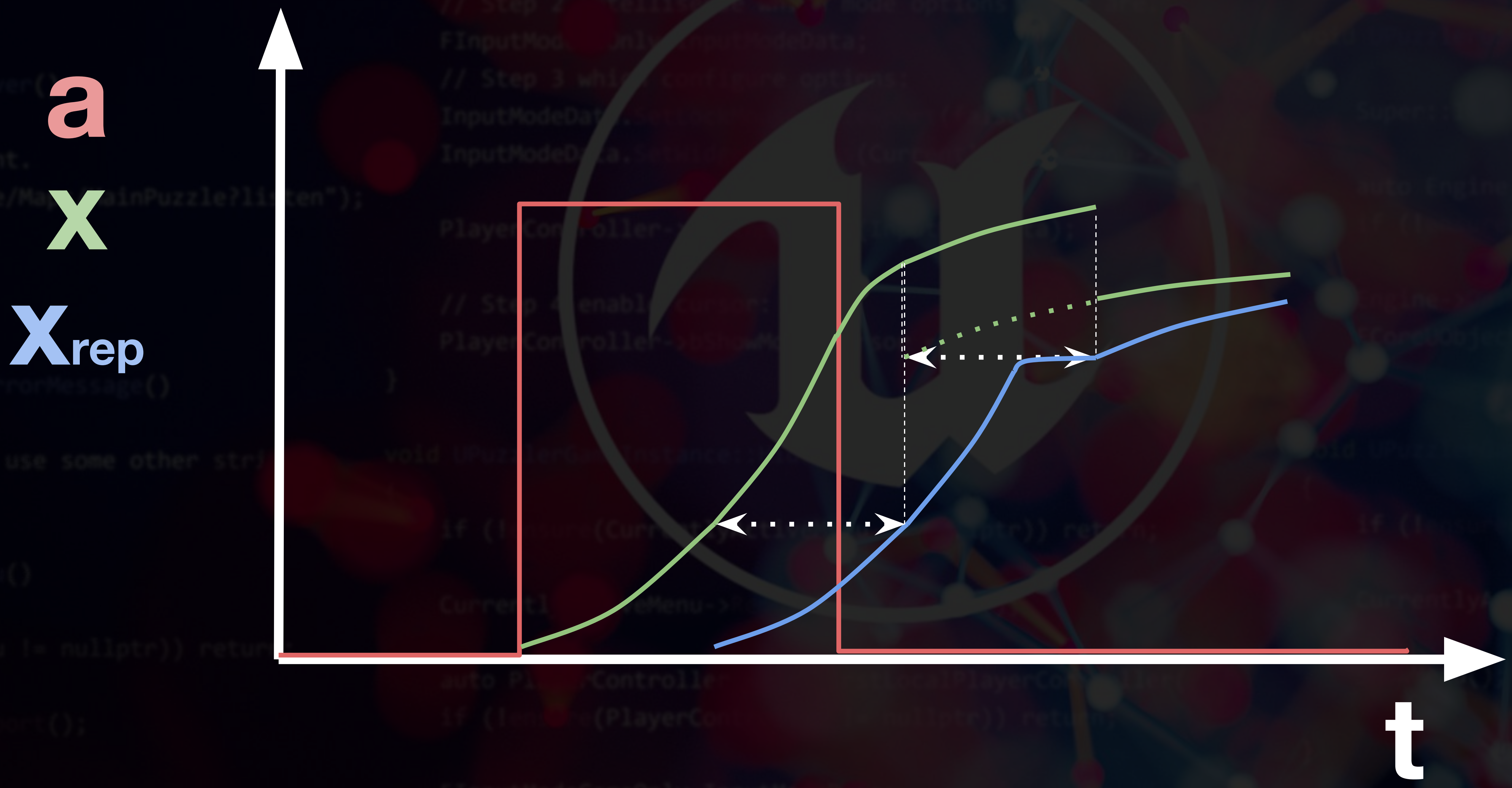
Replay Autonomous Moves

Acceleration With Lag



Live Google Slides at bit.ly/multiplayerslides

Keeping Ahead Of The Server



Comparison Of Methods

	v1
Problem	Not smooth
Information sent to server	Throw
Between updates	Do nothing
Information received	Transform, Velocity
On Receipt	Overwrite local

Comparison Of Methods

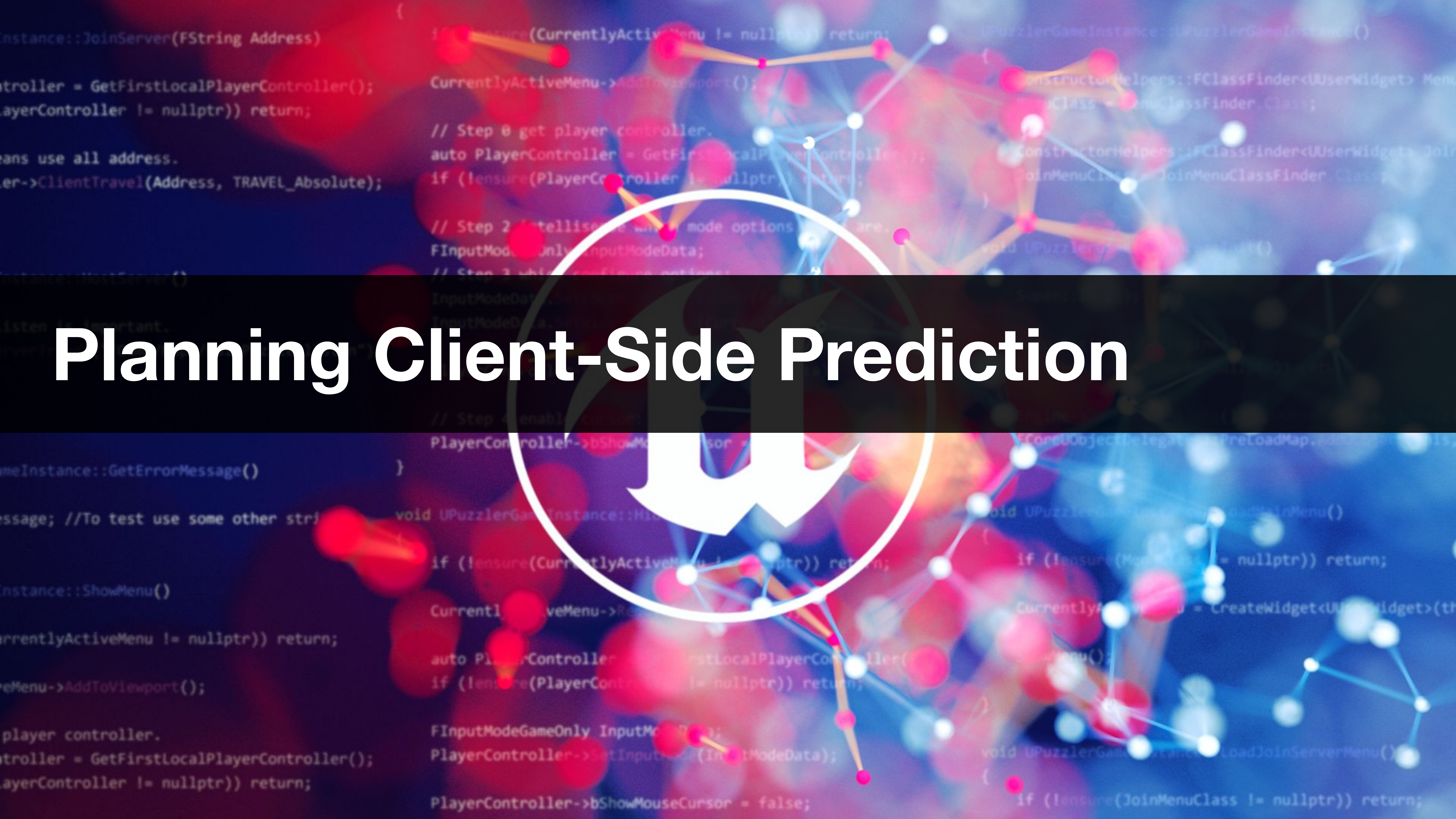
	v1	v2
Problem	Not smooth	Lag
Information sent to server	Throw	
Between updates	Do nothing	Simulate
Information received	Transform, Velocity	
On Receipt	Overwrite local	

Fill Out The Table

- What information needs to be sent?
- What do you do between updates?
- What information comes back?
- What do you do on server update?

Comparison Of Methods

	v1	v2	v3
Problem	Not smooth	Lag	-
Information sent to server	Throw		
Between updates	Do nothing	Simulate	
Information received	Transform, Velocity		Transform, Velocity, ServerTime
On Receipt	Overwrite local		Replay controls since ServerTime



Planning Client-Side Prediction

The High Level

- OnTick:

  Create a move and send to the server.
- OnReceiveMove:

  Simulate it on the server.
- OnReceiveServerState:

  Replay local moves on top.

Pseudocode: OnTick

1. Create a new Move,
2. Save to a list of unacknowledged moves,
3. Send the move to the server,
4. Simulate the move locally.

Pseudocode: OnReceiveMove

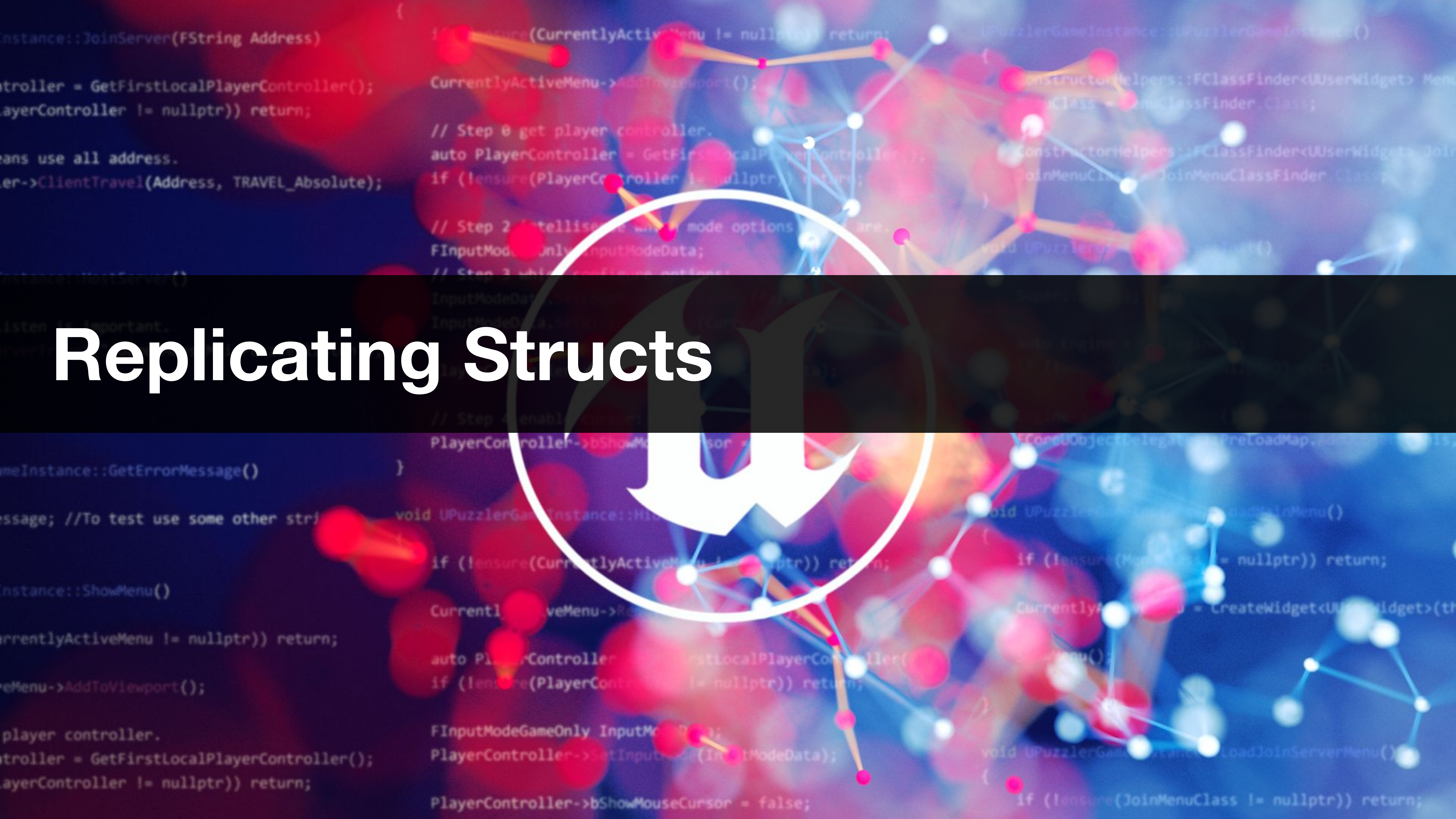
1. Check that the move is valid, (No cheating!)
2. Simulate the move,
3. Send the canonical State to the clients.

Pseudocode: OnReceiveServerState

1. Remove all moves included in state,
2. Reset to server state,
3. Replay/simulate unacknowledged moves.

Create The Structs

- Review the pseudocode
- What data does the Move and State include?
- Create structs for both
- No need to use them yet.



Replicating Structs

Pseudocode Overview

OnTick

1. Create a new Move,
2. Save to a list of unacknowledged moves,
3. Send the move to the server,
4. Simulate the move locally.

OnReceiveMove

1. Check that the move is valid, (No cheating!)
2. Simulate the move,
3. Send the canonical State to the clients.

OnReceiveServerState

1. Remove all moves included in state,
2. Reset to server state,
3. Replay/simulate unacknowledged moves.

What We Already Have (Sort Of)

OnTick

1. Create a new Move,
2. *Save to a list of unacknowledged moves,*
3. Send the move to the server,
4. Simulate the move locally.

OnReceiveMove

1. Check that the move is valid, (No cheating!)
2. Simulate the move,
3. Send the canonical State to the clients.

OnReceiveServerState

1. *Remove all moves included in state,*
2. Reset to server state,
3. *Replay/simulate unacknowledged moves.*

Consolidate The RPC

- Use only the Move struct
- Where should the struct be created?
- Which client should create it?
- What should the server do with the data?

Simulating A Move

Pseudocode

OnTick

1. Create a new Move,
2. *Save to a list of unacknowledged moves,*
3. Send the move to the server,
4. Simulate the move locally.

OnReceiveMove

1. Check that the move is valid, (No cheating!)
2. Simulate the move,
3. Send the canonical State to the clients.

OnReceiveServerState

1. *Remove all moves included in state,*
2. Reset to server state,
3. *Replay/simulate unacknowledged moves.*

Implement SimulateMove

- Pull out of Tick.
- Where will you get the input data?
- Where will you get DeltaTime?

Unacknowledged Move Queue

Pseudocode

OnTick

1. Create a new Move,
2. *Save to a list of unacknowledged moves,*
3. Send the move to the server,
4. Simulate the move locally.

OnReceiveMove

1. Check that the move is valid, (No cheating!)
2. Simulate the move,
3. Send the canonical State to the clients.

OnReceiveServerState

1. *Remove all moves included in state,*
2. Reset to server state,
3. *Replay/simulate unacknowledged moves.*

Prune The Queue

- Write ClearAcknowledgedMoves
- Call it from OnRep_ServerState
- Use the TArray documentation to help you
- What makes a move stale?

Simulating Unacknowledged Moves

Pseudocode

OnTick

1. Create a new Move,
2. *Save to a list of unacknowledged moves,*
3. Send the move to the server,
4. Simulate the move locally.

OnReceiveMove

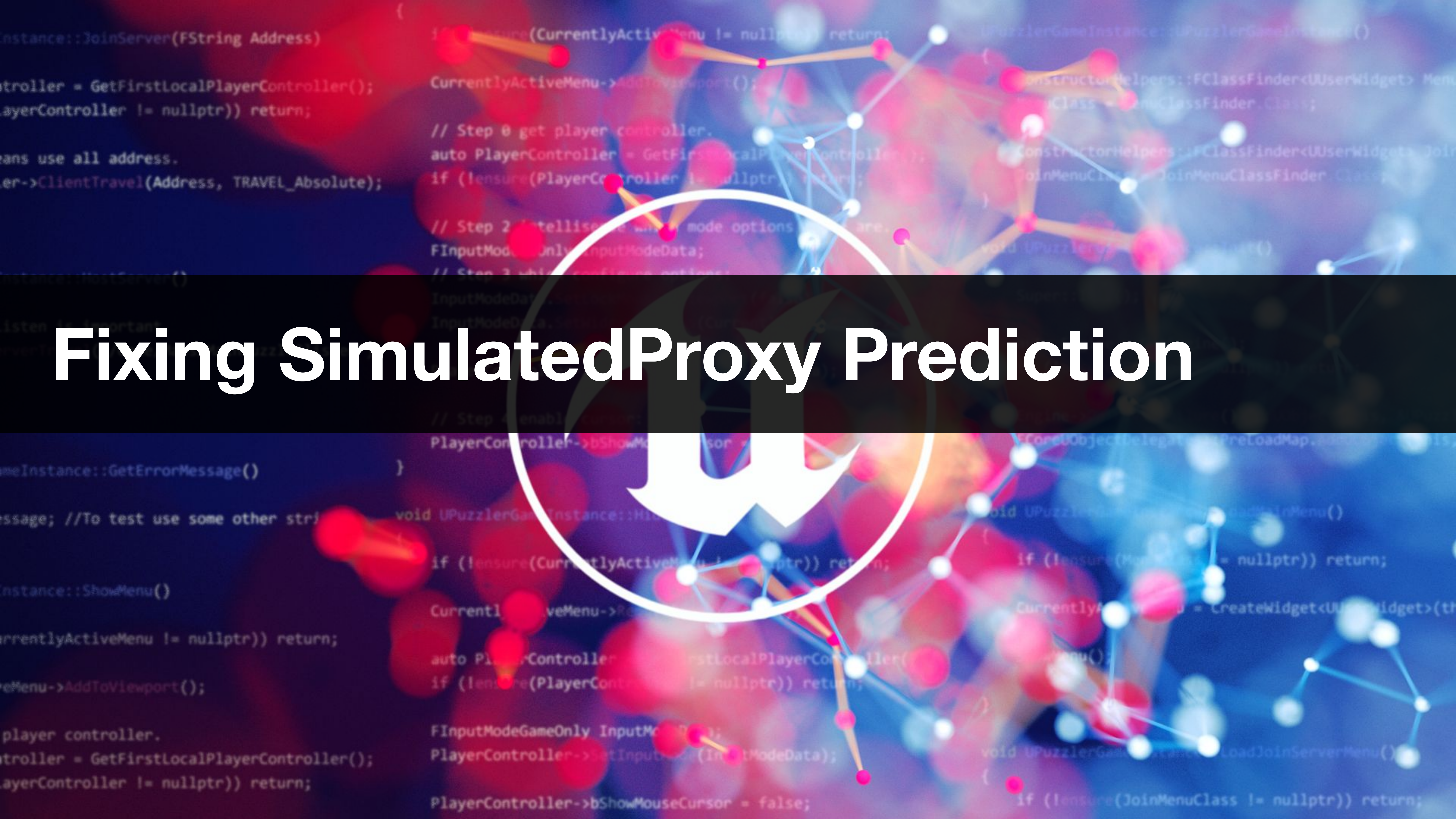
1. Check that the move is valid, (No cheating!)
2. Simulate the move,
3. Send the canonical State to the clients.

OnReceiveServerState

1. *Remove all moves included in state,*
2. Reset to server state,
3. *Replay/simulate unacknowledged moves.*

Make It Glitch

- Glitching is still possible if we disagree
- Brainstorm why we might disagree
- Share on the forum
- Try to reproduce a glitch.

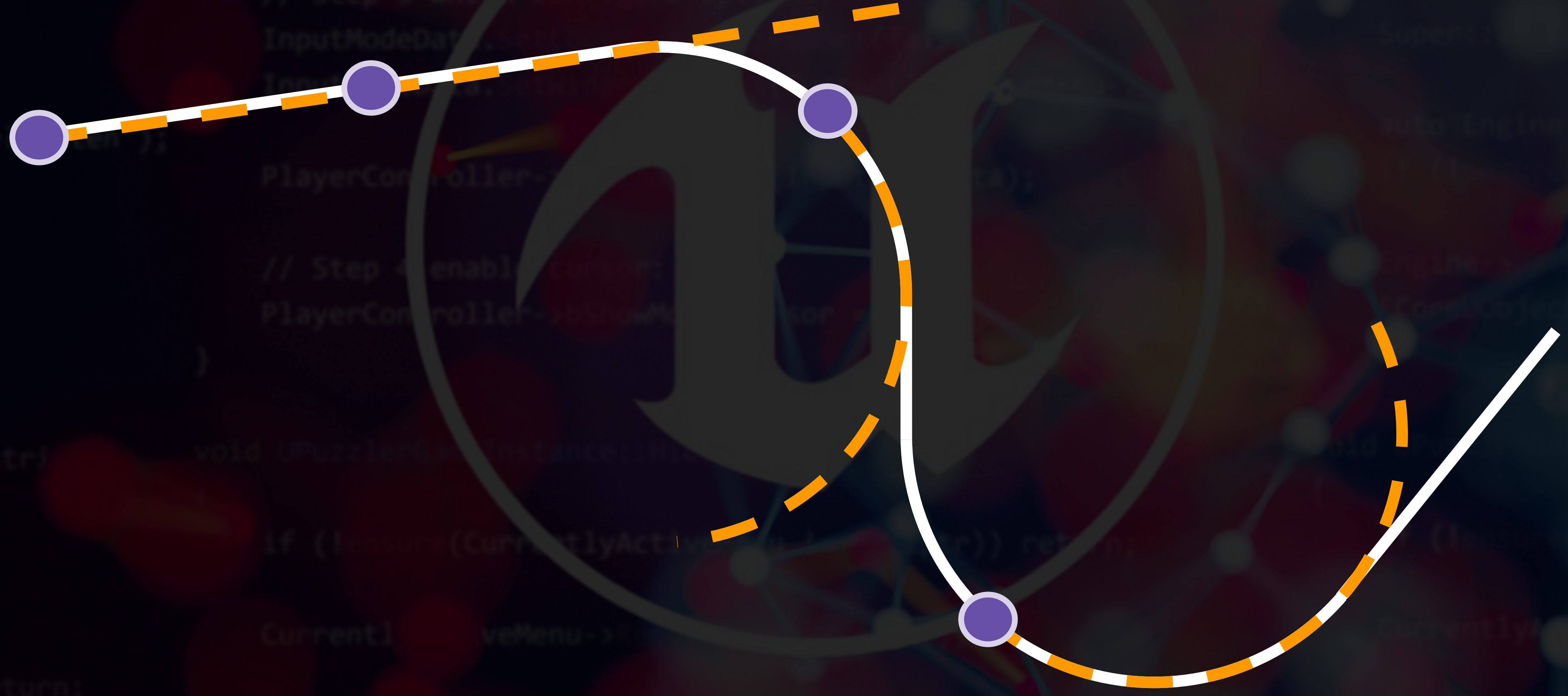


Fixing SimulatedProxy Prediction

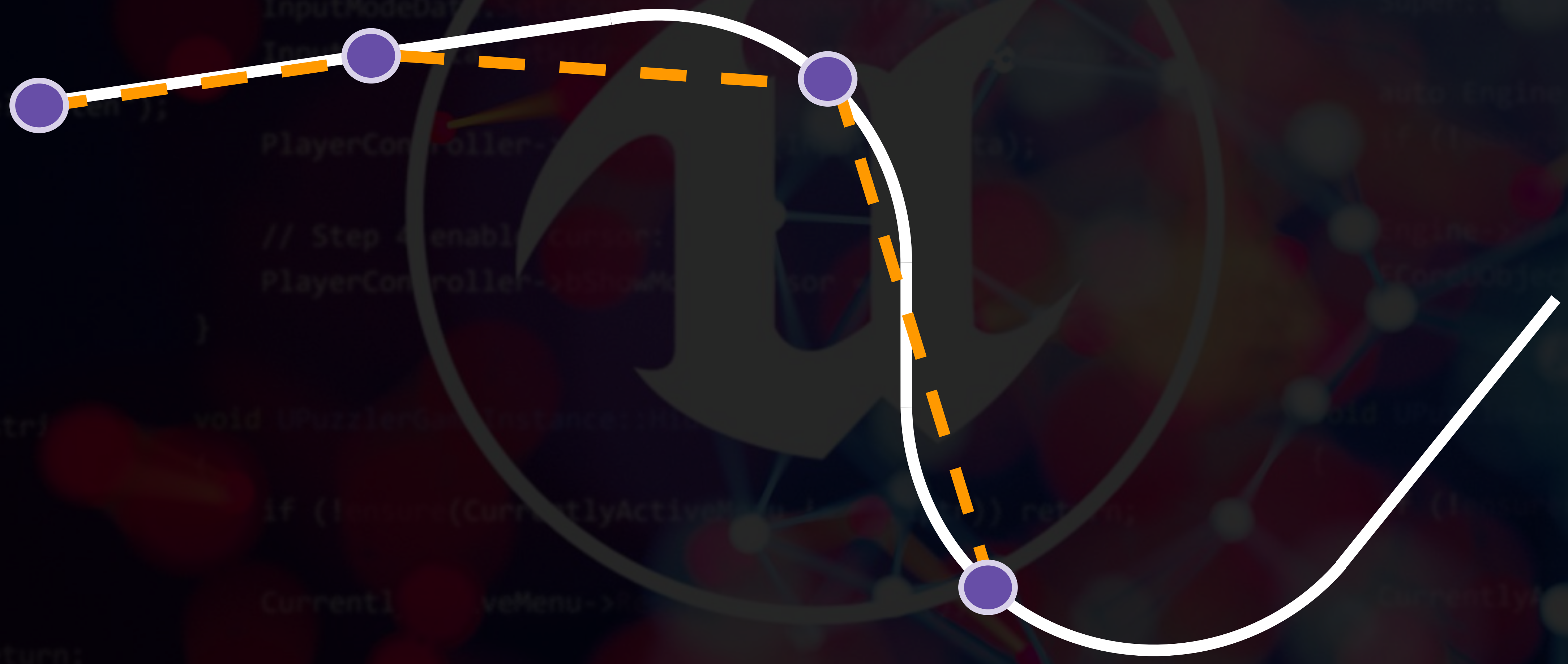
Fix Client Prediction

- Why is the client jumping?
- What could we do between updates?
- Implement your solution.

Our Current Solution

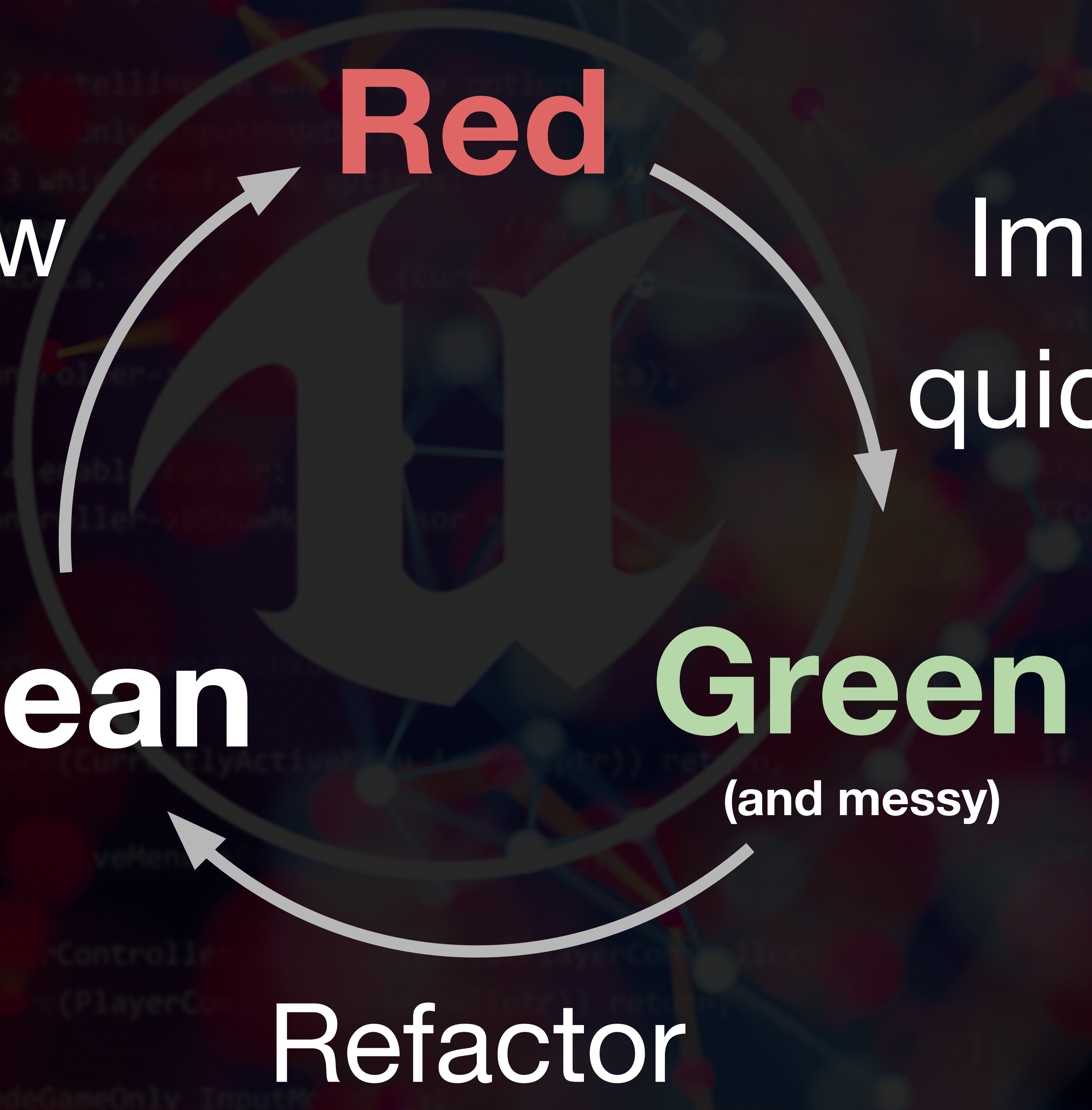


Interpolating Client Position



Refactoring Into Components

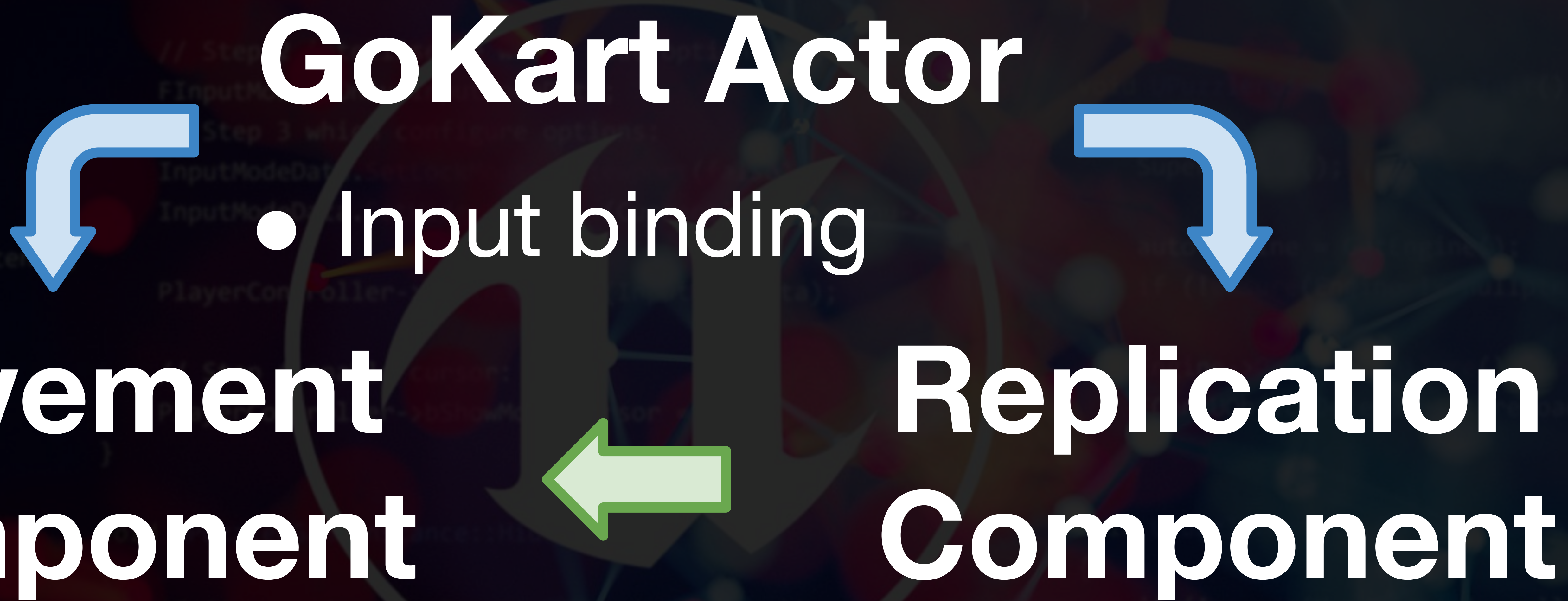
Red-Green-Refactor Loop



Plan The Refactor

- Identify the “code smell”
- Choose a refactor
- Plan that refactor.

Refactor Plan



GoKart Actor

- Input binding

Movement Component

- Simulating physics

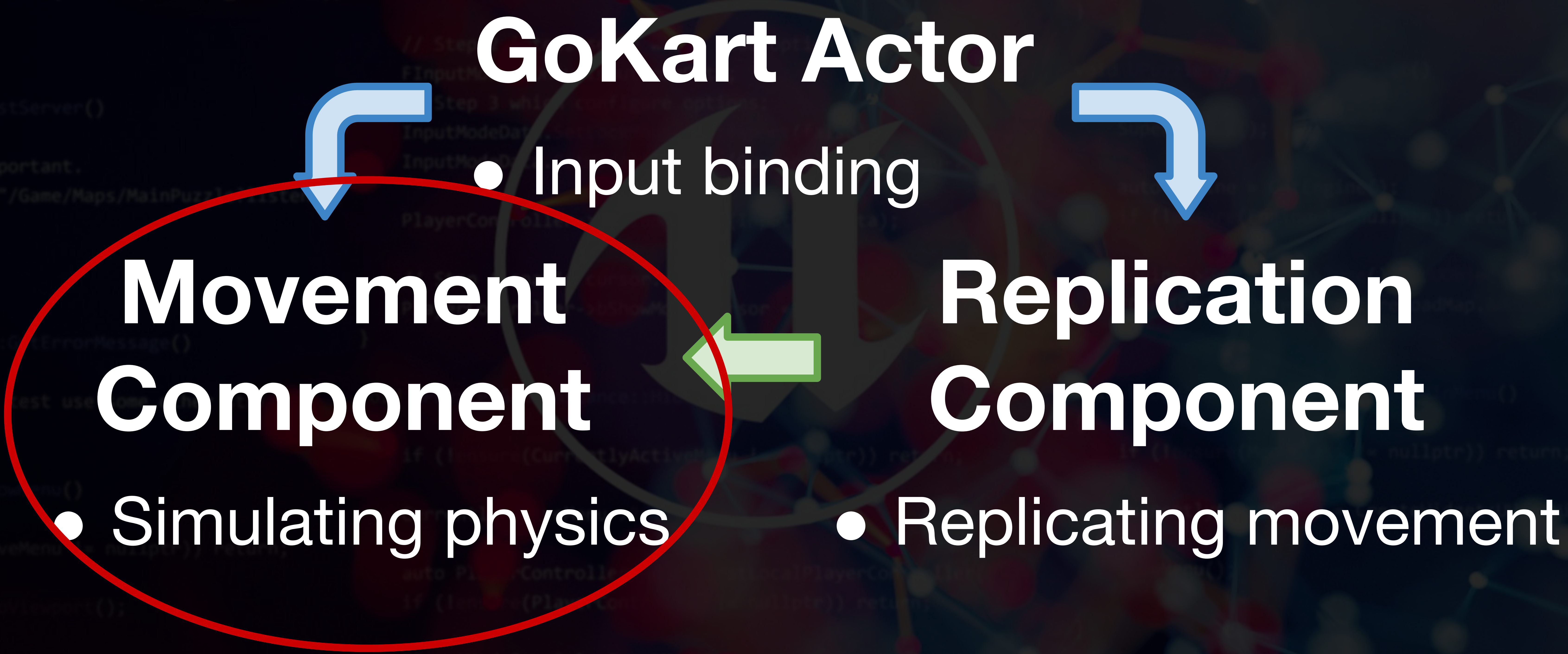
Replication Component

- Replicating movement

Live Google Slides at bit.ly/multiplayerslides

Extracting A Movement Component

Refactor Plan

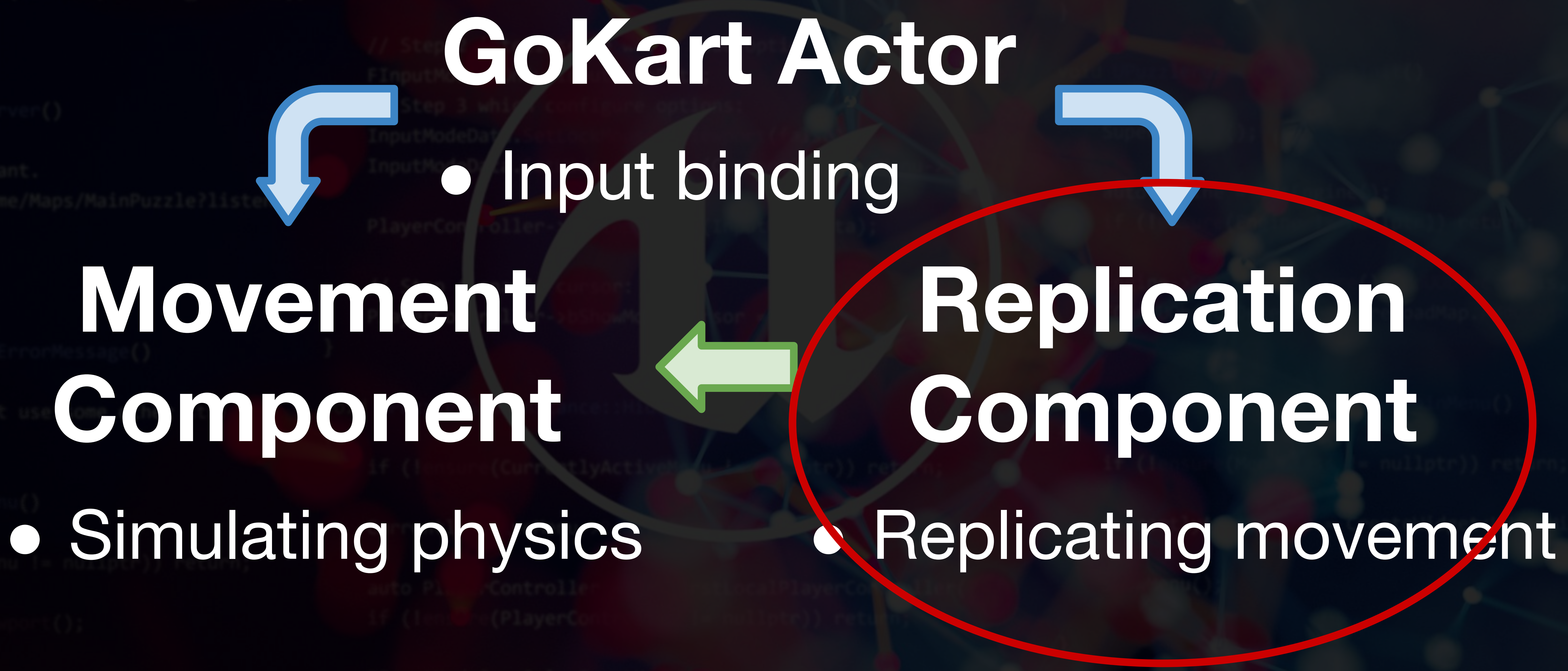


Make It Build

- Pull across the method implementations
- Change the class namespace
- Prefix all calls to Actor functions with:
  `GetOwner()->`
- Fix references in GoKart.cpp
- Create any accessor methods you need.
- Fix other build errors.

Extracting A Replication Component

Refactor Plan

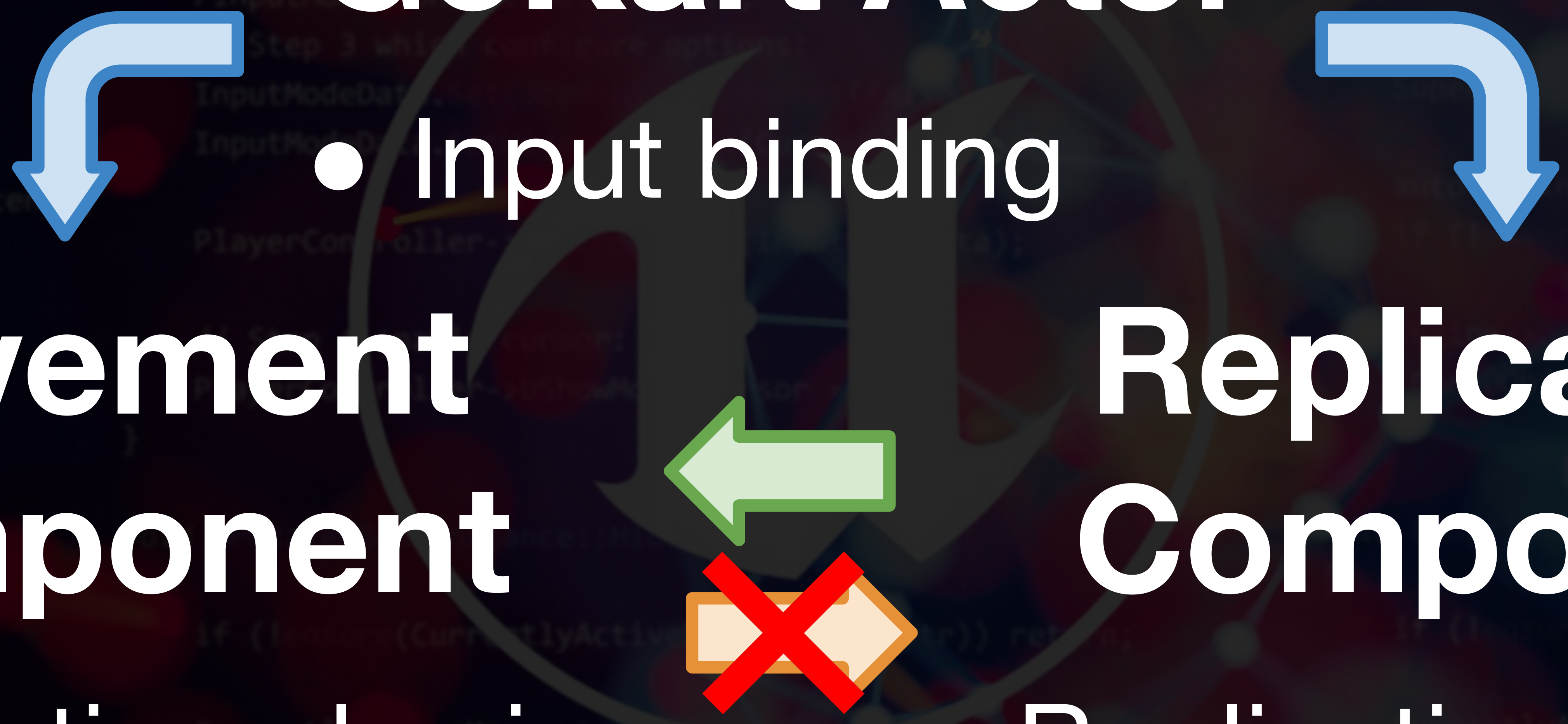


Refactor!

- Pull across the method implementations
- Change the class namespace
- Prefix all calls to Actor functions with: `GetOwner()->`
- Fix references in GoKart.cpp
- Create any accessor methods you need.
- Fix other build errors.

Decoupling Movement & Replication

Refactor Plan



GoKart Actor

- Input binding

Movement Component

- Simulating physics

Replication Component

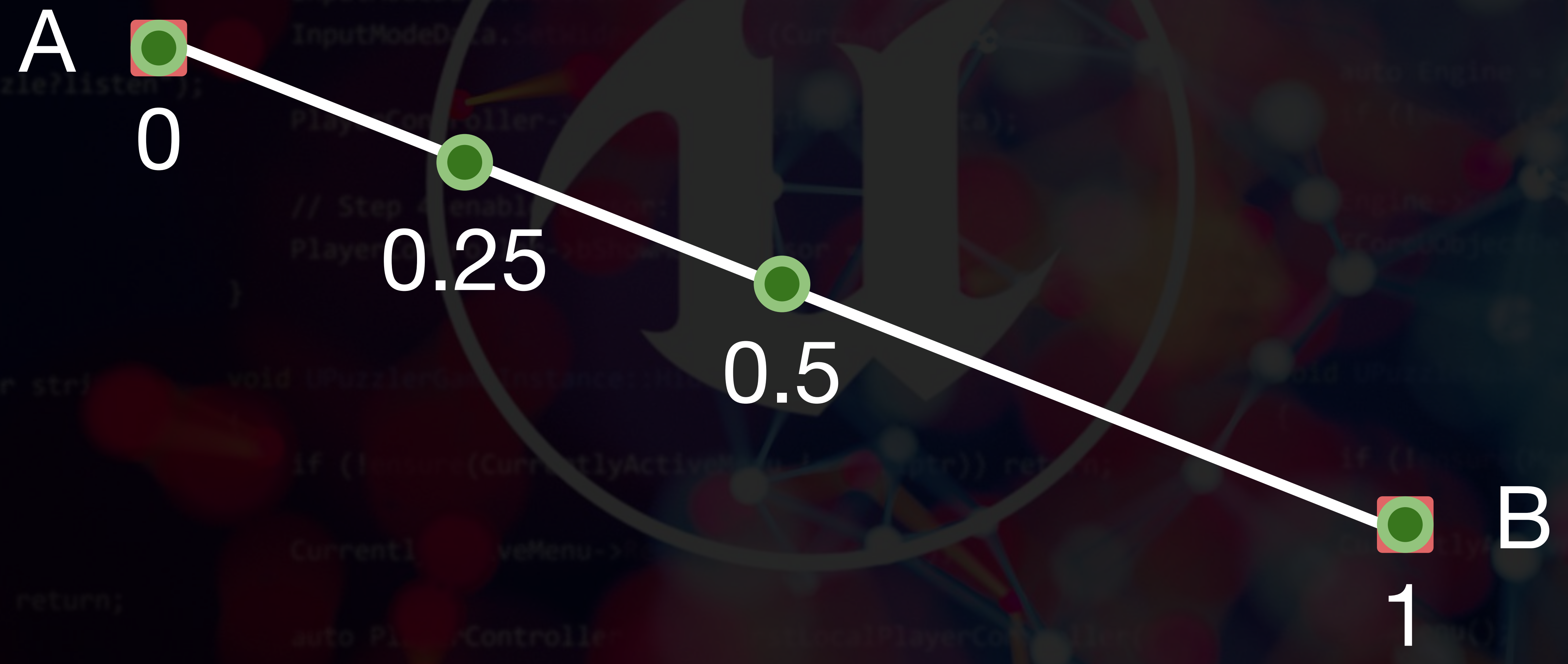
- Replicating movement

Make Replication Work

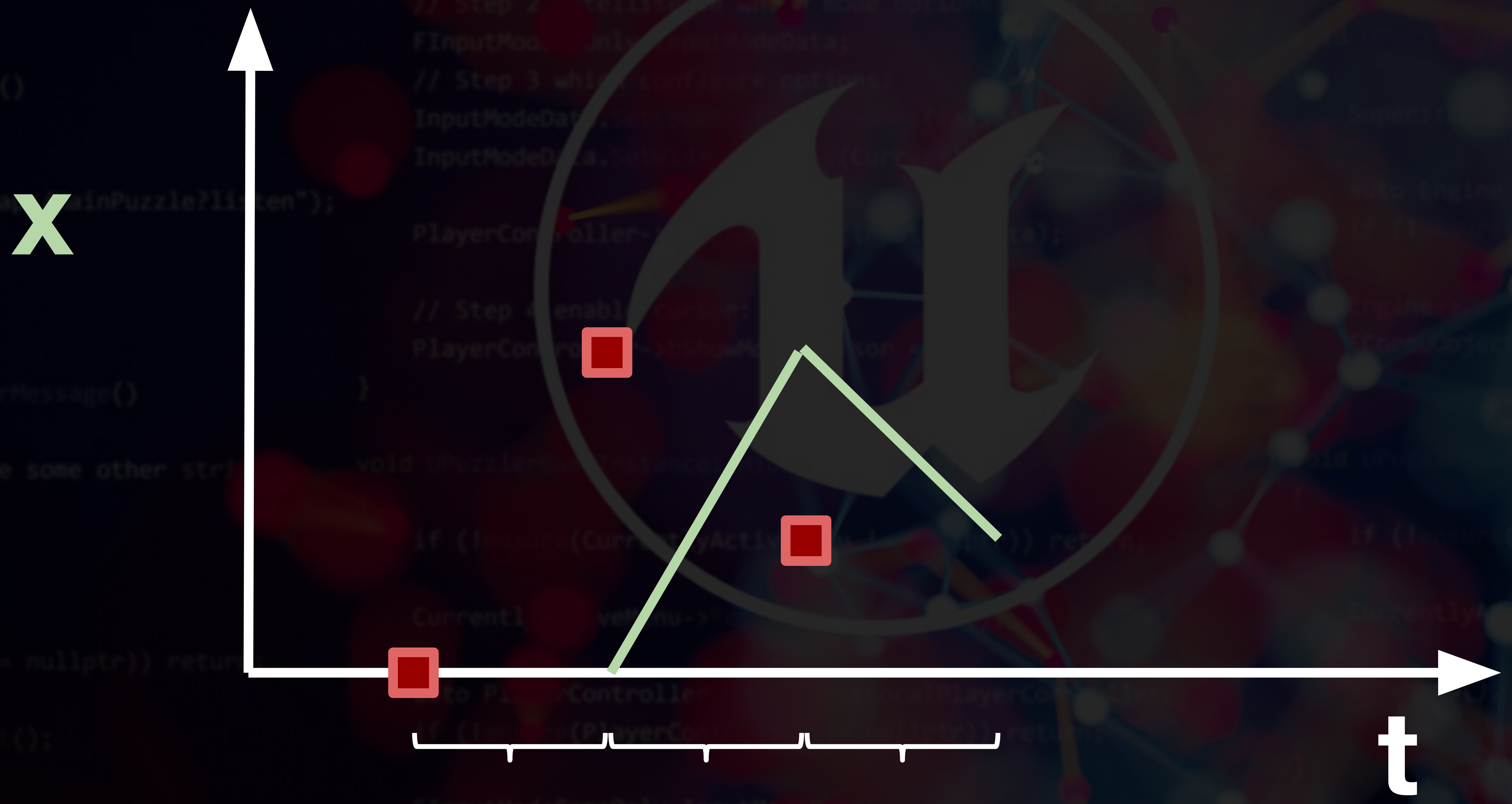
- Expose the last move
- Replace the simulating code in Replicator
- How to change Server_SendMove()?
- Can me make some methods private now?
- When shouldn't the MC simulate?

Linear Interpolation For Position

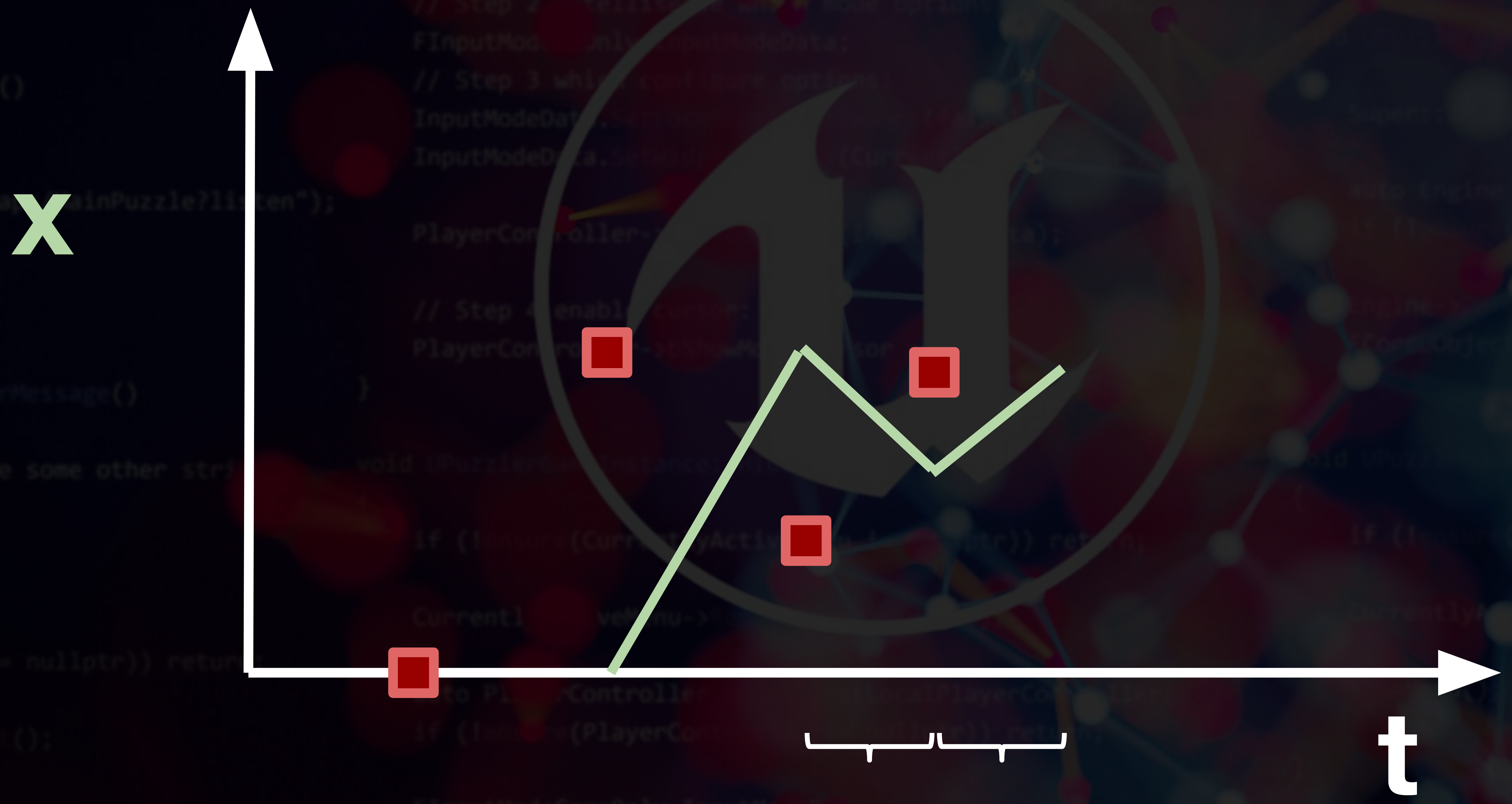
Linear Interpolation (Lerp)



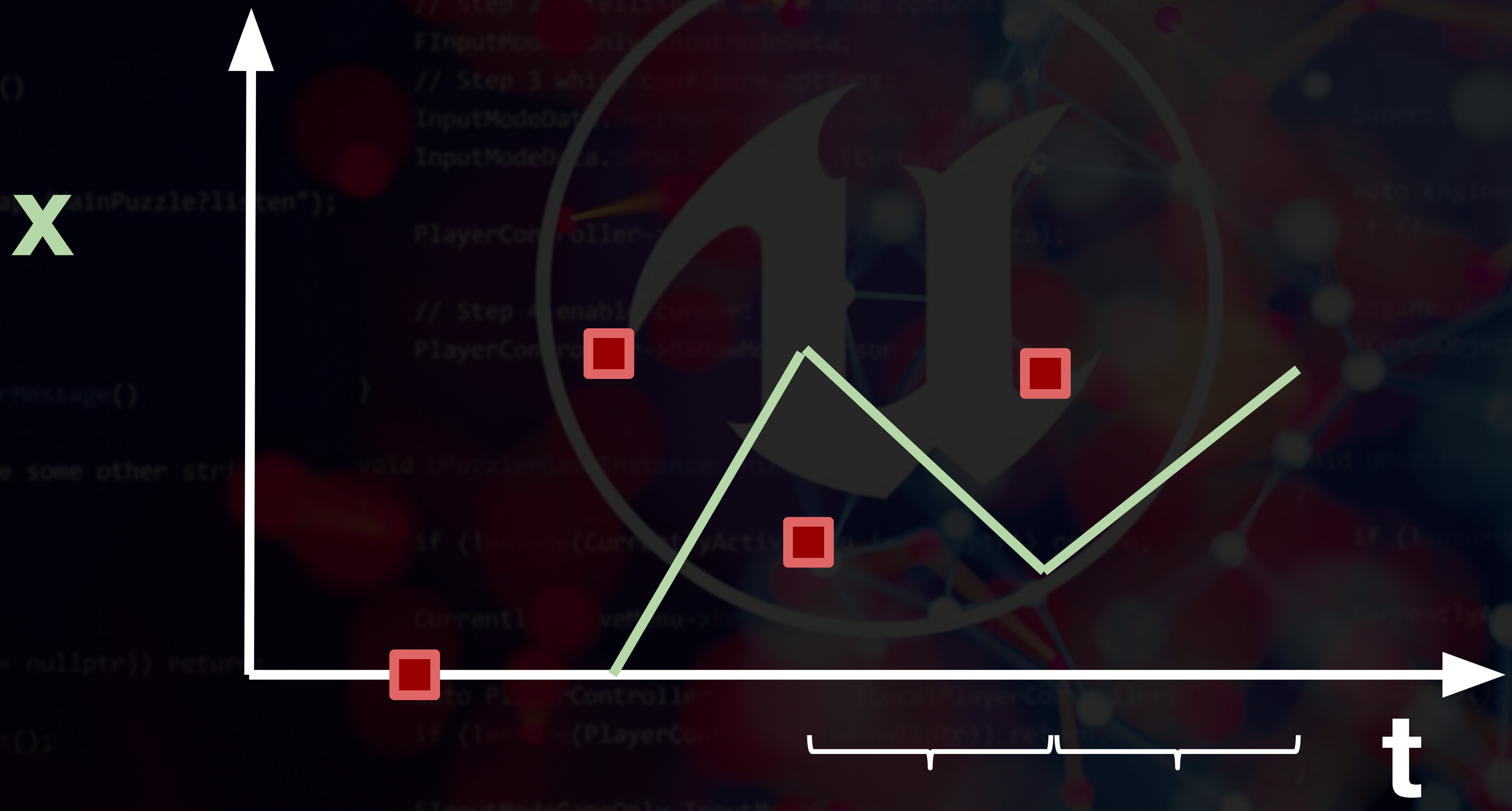
Client Interpolation



Early Updates



Late Updates



Write The Pseudocode

- Consider what you will do on tick
- How about OnRep?
- User the Lerp function:

```
Lerp(A, B, alpha)
```

- What data do you need to store?

Pseudo Code

OnTick:

```
TargetLocation = ServerState.Location
LerpRatio = TimeSinceUpdate / TimeBetweenLastUpdates
NextLocation = Lerp(StartLocation, TargetLocation, LerpRatio)
SetLocation(NextLocation)
```

OnRep:

```
StartLocation = GetLocation()
```

FMath::Lerp For Client Interpolation

Pseudo Code

OnTick:

```
TargetLocation = ServerState.Location
LerpRatio = TimeSinceUpdate / TimeBetweenLastUpdates
NextLocation = Lerp(StartLocation, TargetLocation, LerpRatio)
SetLocation(NextLocation)
```

OnRep:

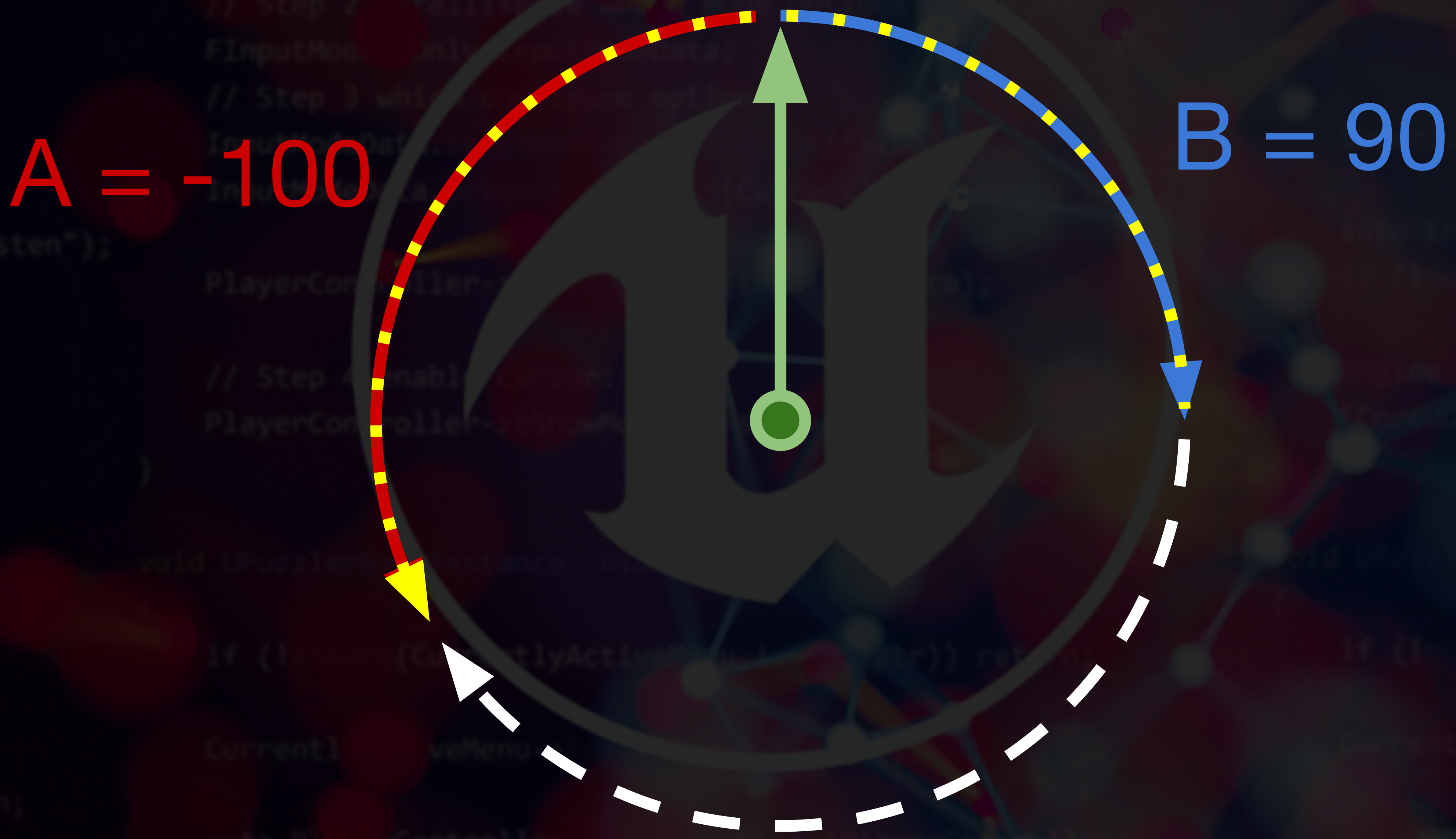
```
StartLocation = GetLocation()
```

Implement The Pseudocode

- Which member variables are missing?
- Implement Tick.
- Implement OnRep.
- Keep it neat.

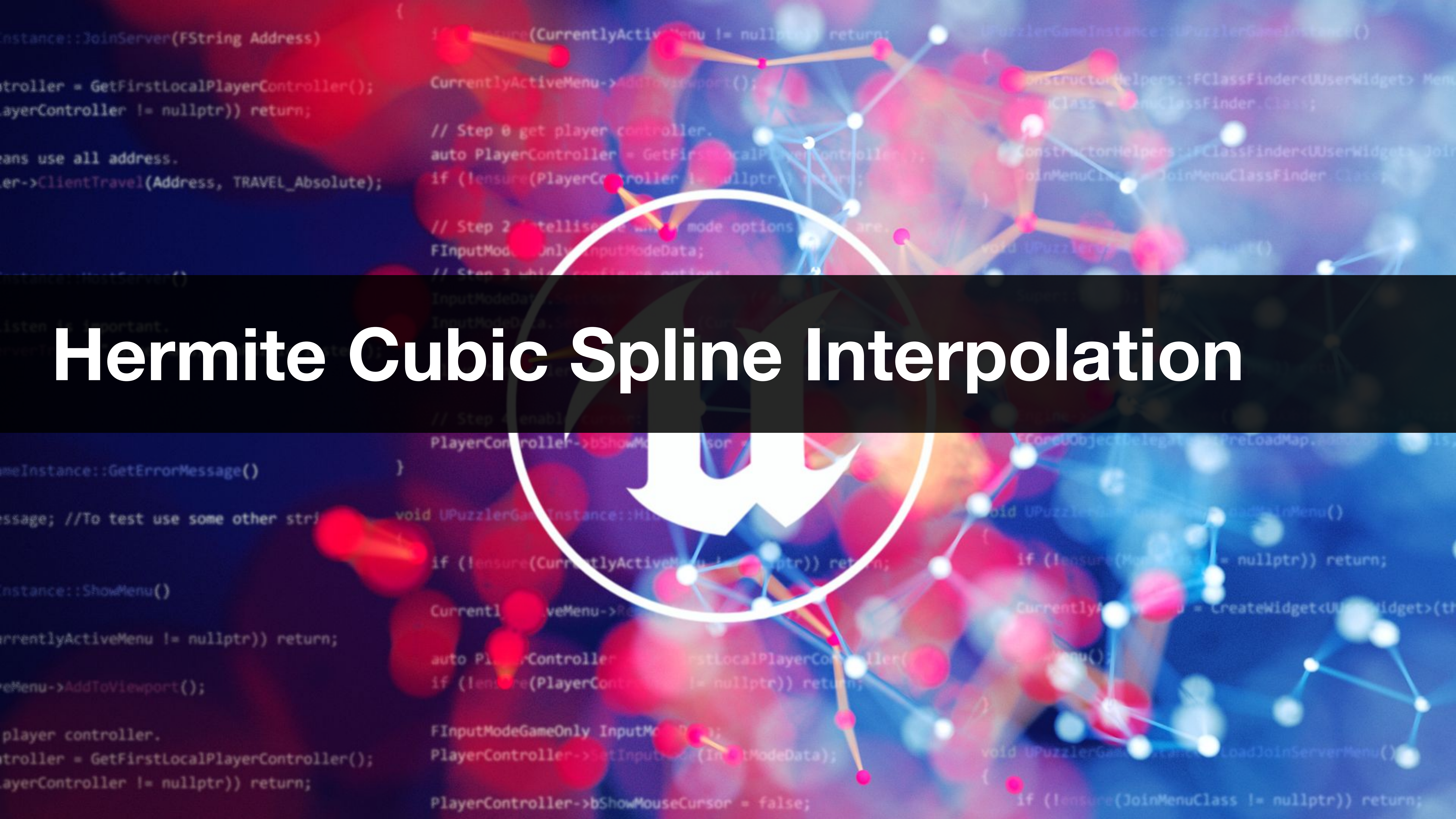
FQuat::Slerp For Rotation

Why Can't We Lerp?



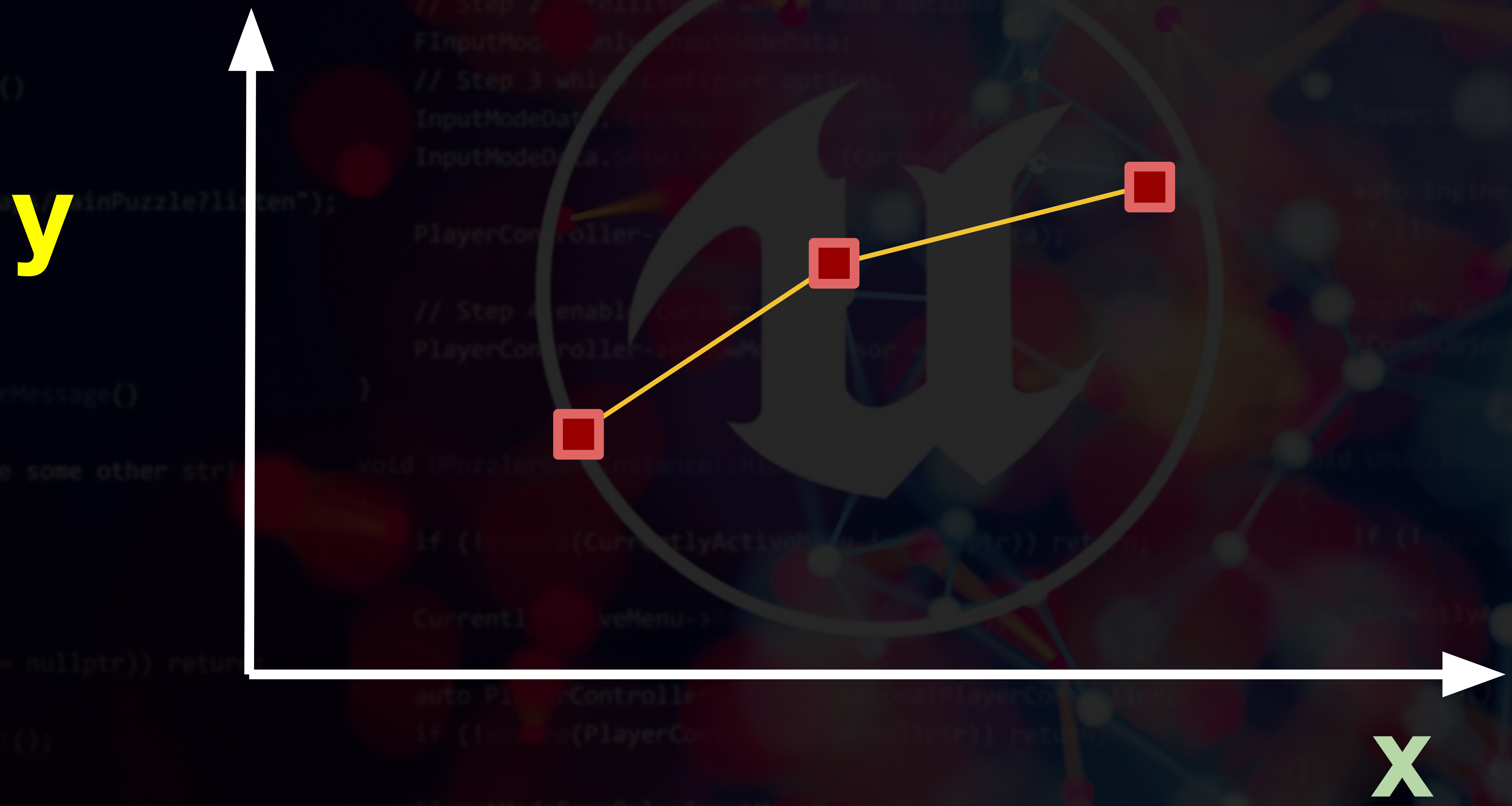
Slerp The Rotation

- Repeat the pattern
- Use `FQuat::Slerp`
- Test!

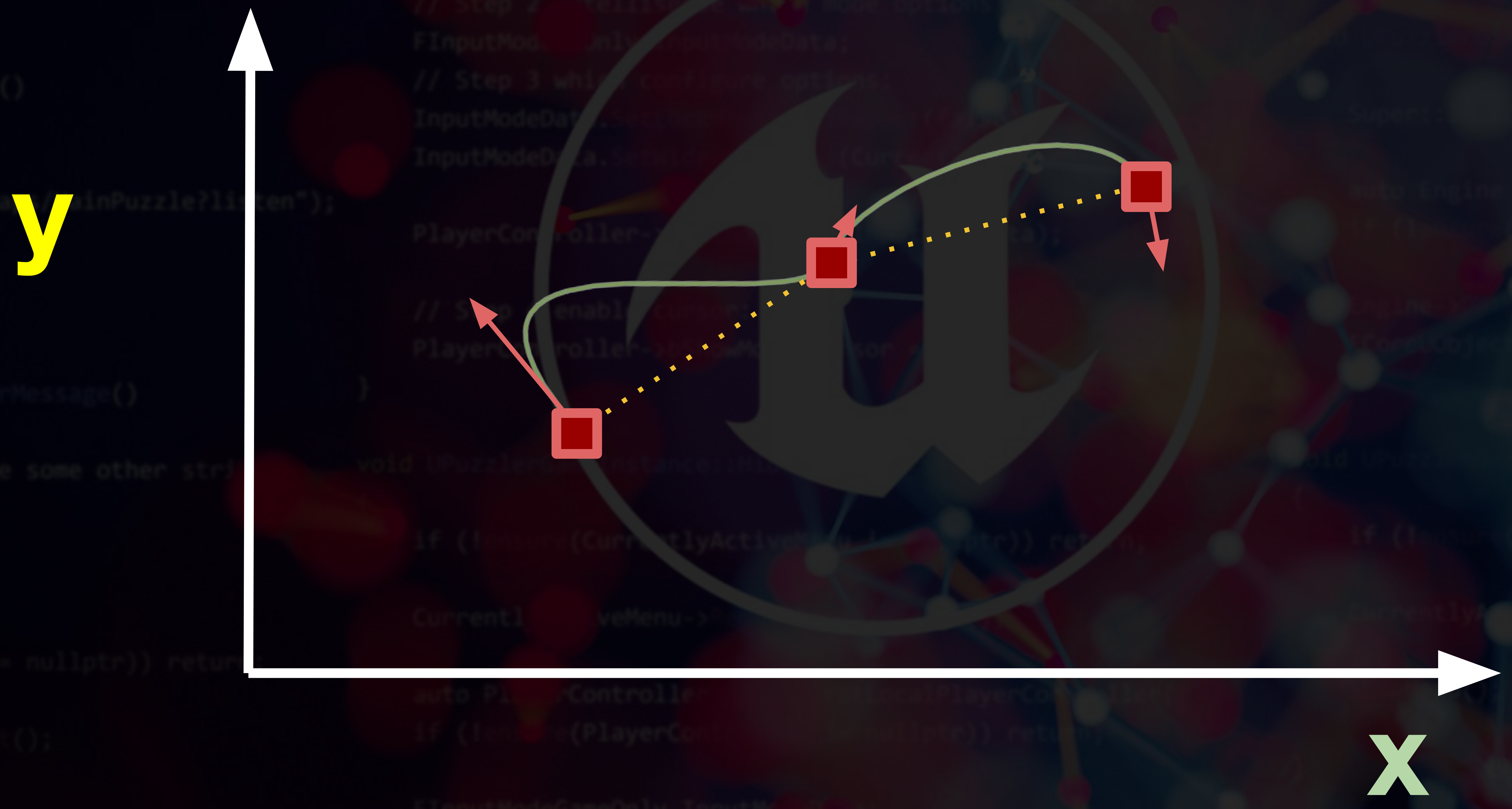


Hermite Cubic Spline Interpolation

Problem: Jarring Movement

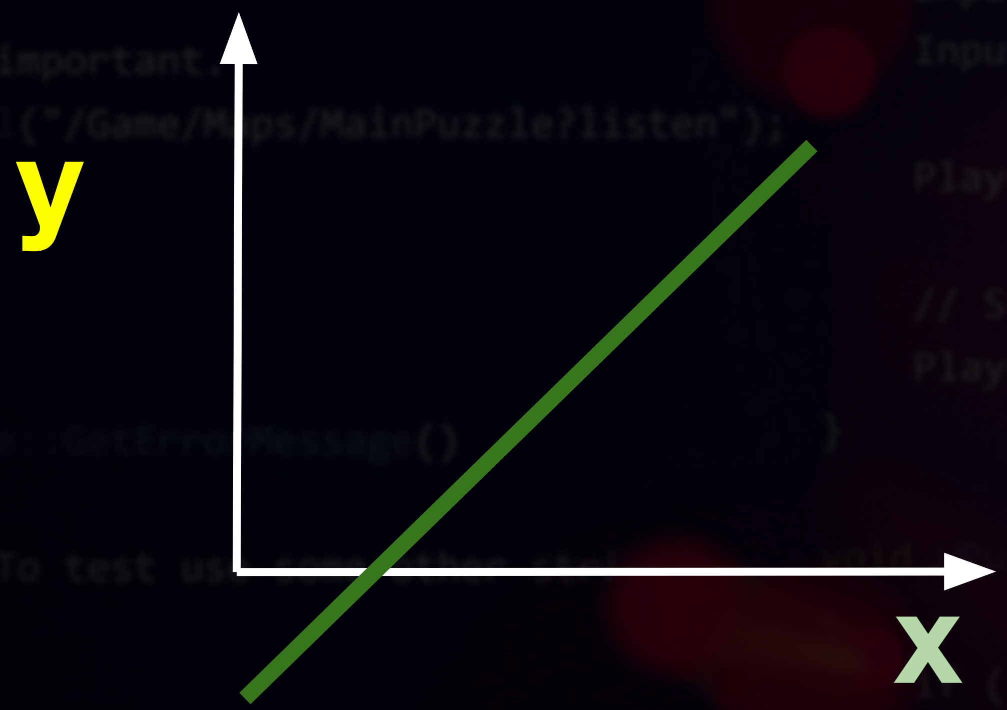


Incorporate Velocity?

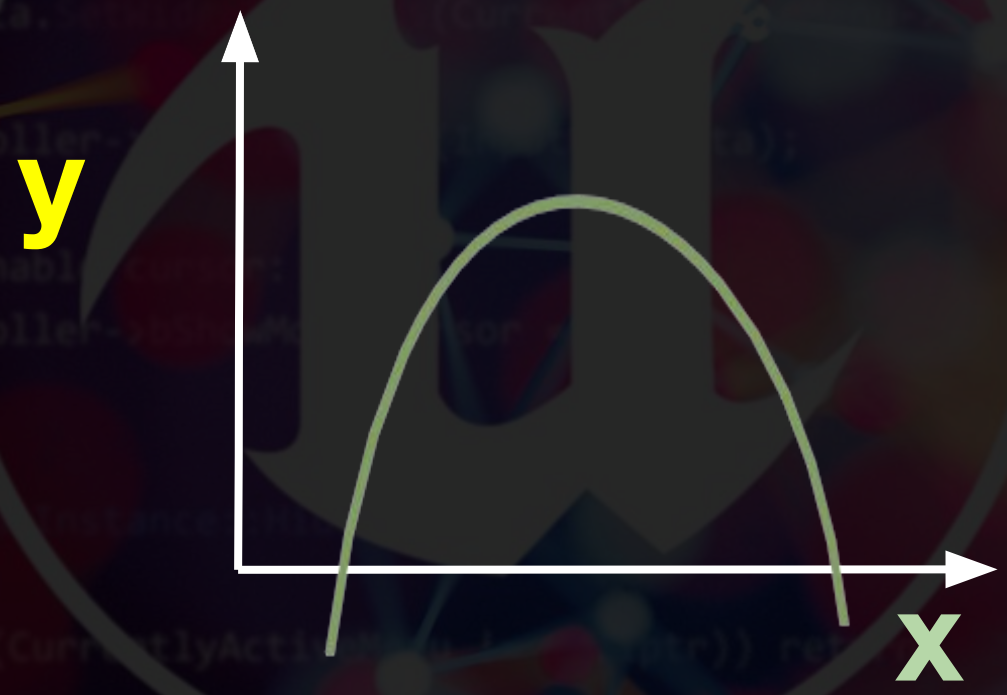


New Tool: Polynomials

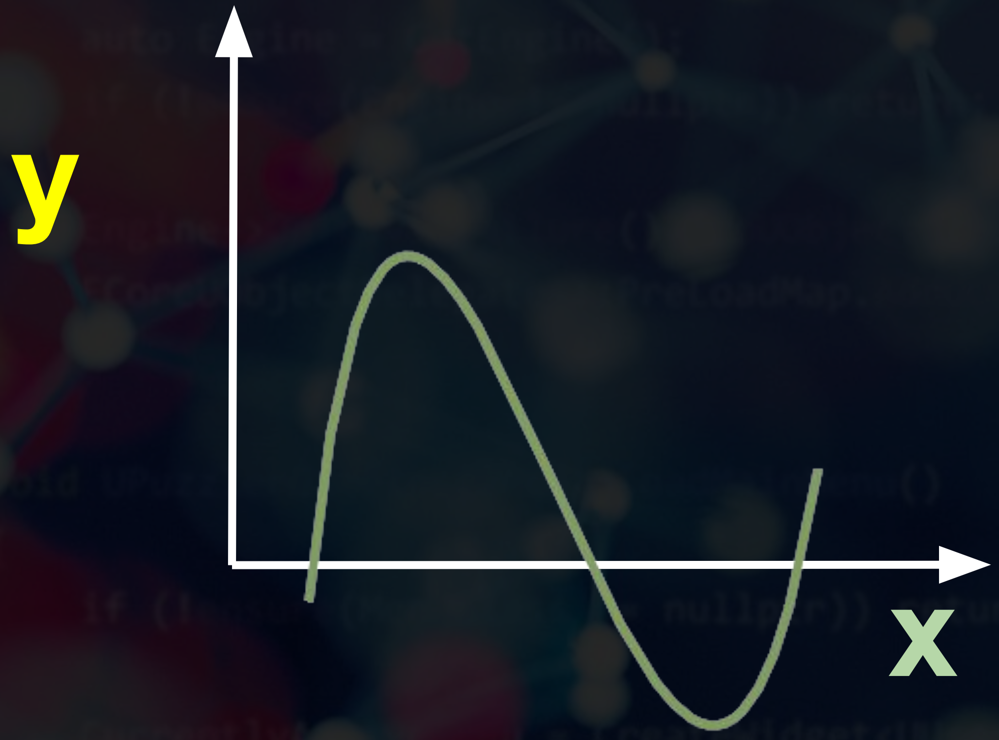
Linear



Quadratic



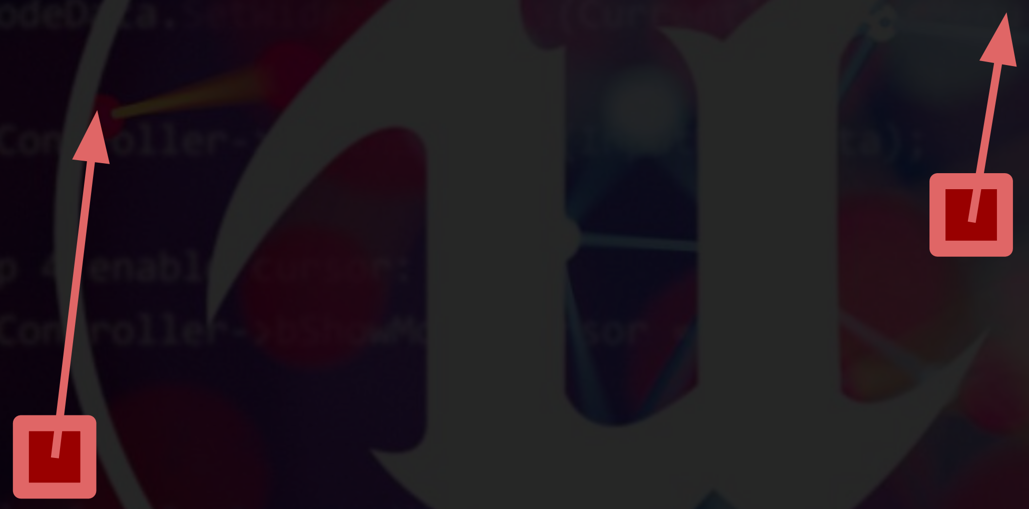
Cubic



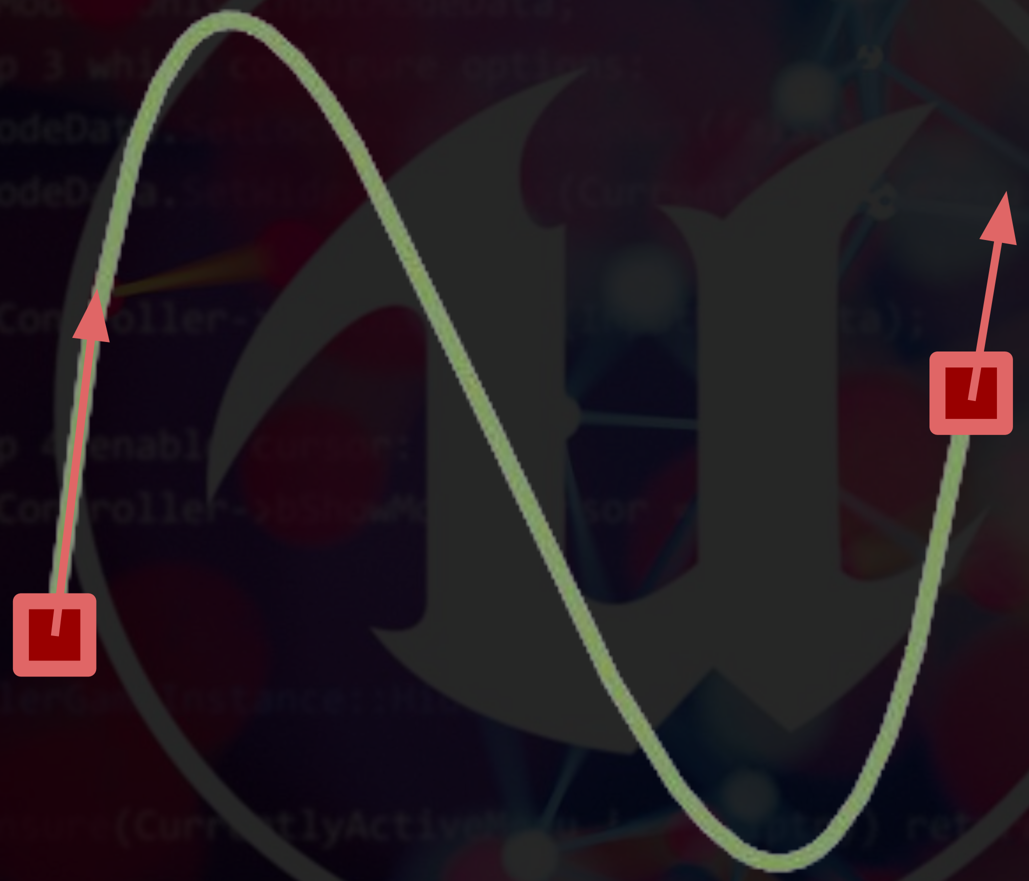
More Complex

Live Google Slides at bit.ly/multiplayerslides

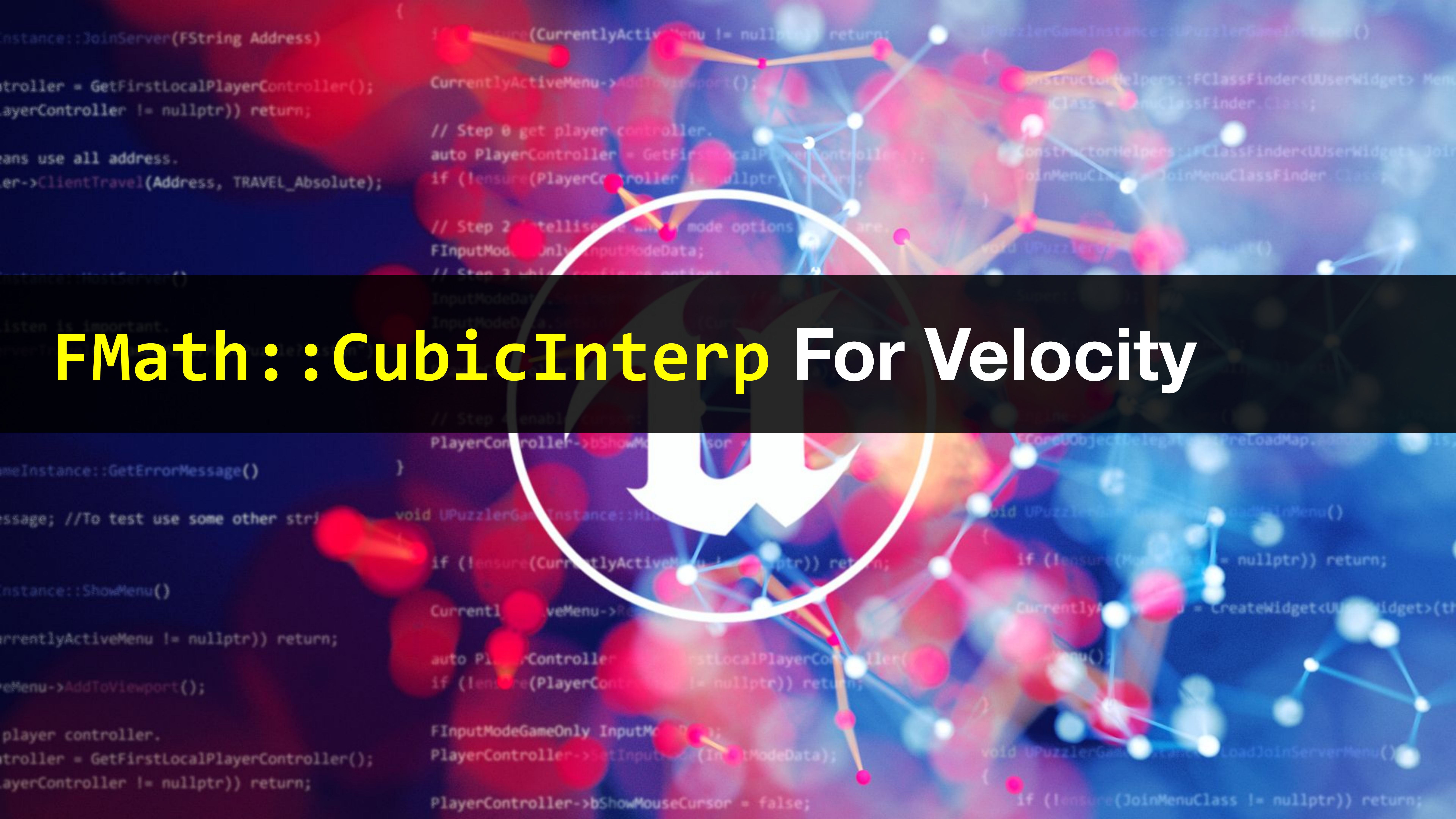
Which Is The Simplest Curve?



Which Is The Simplest Curve?

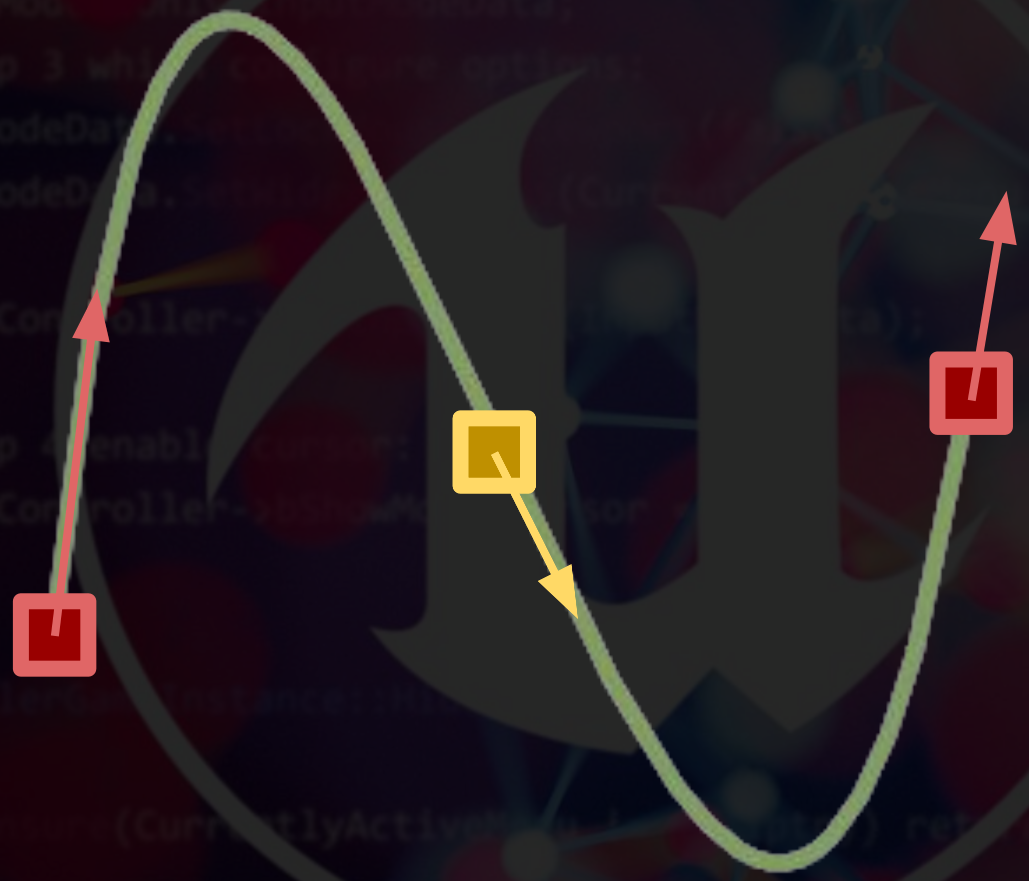


Cubic!



FMath::CubicInterp For Velocity

Cubic Interpolation And Velocity



Slope, Derivative And Velocity

```
Slope = Derivative
      = DeltaLocation / DeltaAlpha

Velocity = DeltaLocation / DeltaTime
DeltaAlpha = DeltaTime / TimeBetweenLastUpdates
Derivative = Velocity * TimeBetweenLastUpdates
```

Interpolate Velocity

- Research `FMath::CubicInterpDerivative`
- Convert this to velocity
- Set the velocity on the movement component
- Test!

Refactoring With Structs

Pull Out Some Methods

- Pull out `CreateSpline`,
- `InterpolateLocation`,
- `InterpolateVelocity`,
- `InterpolateRotation`

Client Interpolation Mesh Offset

Manipulate The Offset

- Reset the actor location on rep
- Elsewhere, use the component location
- Do the same for rotation.

Advanced Cheat Protection

Prevent the DeltaTime Cheat

- Track the simulated time
- Ensure it's less than the server time
- Prevent multiplying moves
- Prevent long delta times.

End Of Course Wrap-up

That’s all for now!

Slides after this point are being recorded as you read this. Comment on them to have your say!