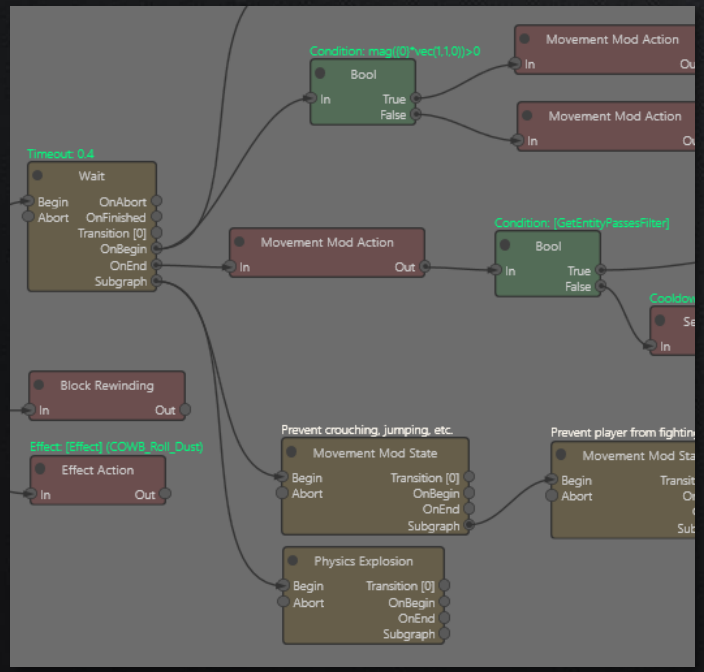
Networking Scripted Weapons and Abilities in Overwatch

Dan Reed
Senior Gameplay Engineer
Blizzard Entertainment
dreed@blizzard.com


Condition: mag([0]*vec[1,1,0])>0
Bool
In
True
False
Movement Mod Action
In
Out
Movement Mod Action
In
Timeout: 0.4
Wait
Begin
Abort
OnAbort
OnFinished
Transition [0]
OnBegin
OnEnd
Subgraph
Movement Mod Action
In
Out
Condition: [GetEntityPassesFilter]
Bool
In
True
False
Block Rewinding
In
Out
Effect: [Effect] (COWB_Roll_Dust)
Effect Action
In
Out
Prevent crouching, jumping, etc.
Movement Mod State
Begin
Abort
Transition [0]
OnBegin
OnEnd
Subgraph
Physics Explosion
Begin
Abort
Transition [0]
OnBegin
OnEnd
Subgraph


0%
0%
CONTESTING POINT!
ARYA 2
ANA
LUCIO
TORBJORN
ATTACK
MCCREE


Zenyatta
Lucio
Bastion 2
Torbjörn
Soldier: 76
Soldier: 77

Overview

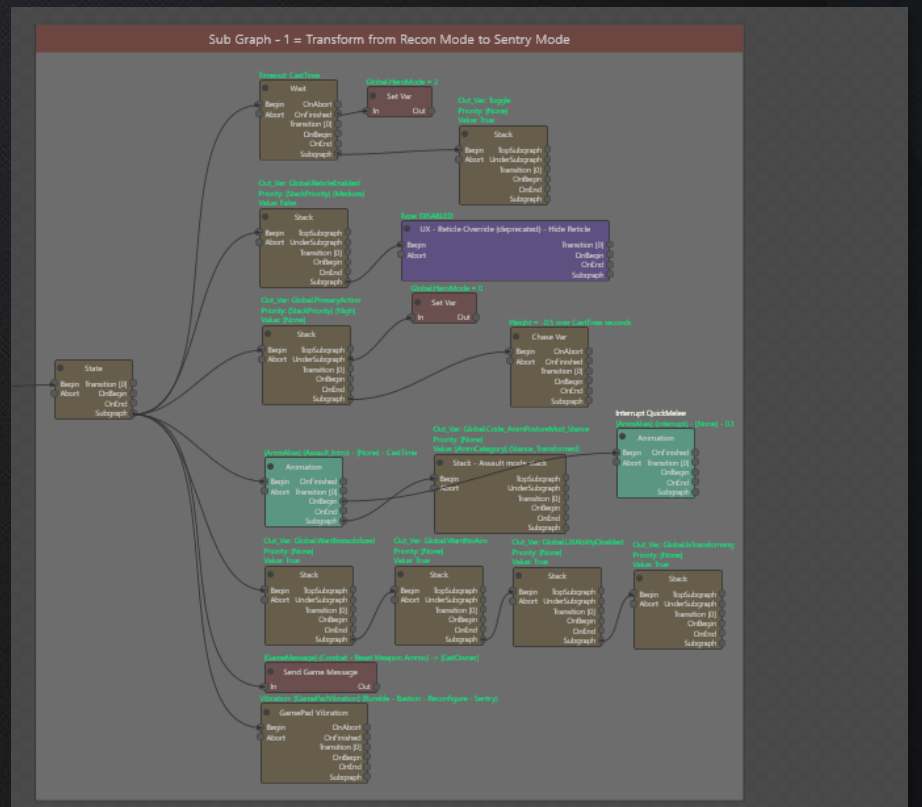
- Overwatch implements its high-level logic in **Statescript**
- What this talk is about
  - Why, what, and how *(~15 min)*
  - Networking requirements and solutions *(~30 min)*
  - Benefits and challenges *(~5 min)*
- What this talk is *not* about
  - Projectiles
  - Hit registration
  - Tech for specific abilities

Why Statescript

- Non-programmers need to create high-level logic
- Define game state (don’t just react to it)
- Encourage modular code
- Painless synchronized state machines
- Works with the rest of our engine

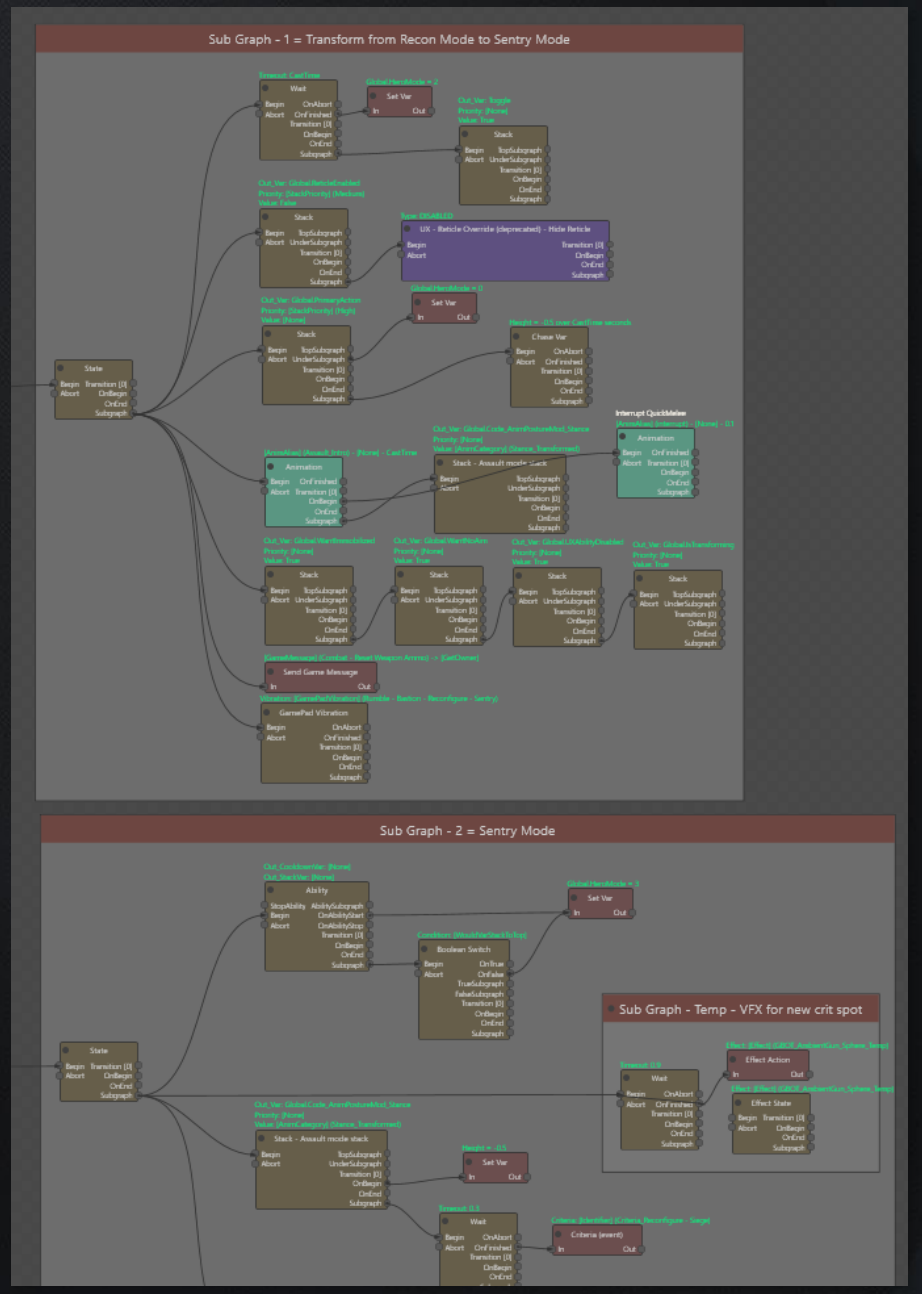
What is Statescript

- Visual scripting language
- Each script is a graph of nodes describing a chunk of gameplay
- Examples
  - Tracer’s Recall ability
  - Lucio buff
  - Common UI elements

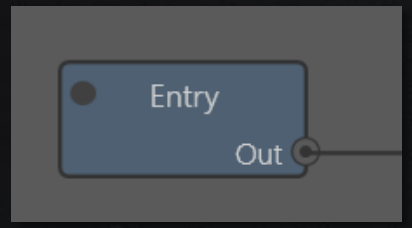


What is Statescript

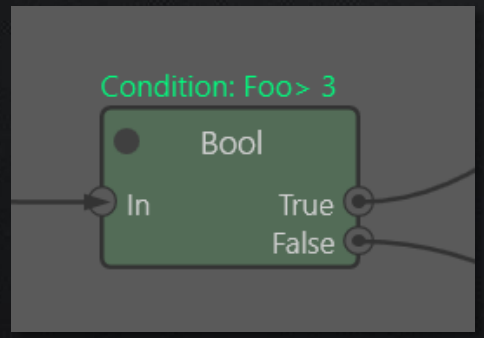
- When a script is played, it creates a Statescript **Instance**
  - An Instance is owned by an **Entity**
  - Instances can be added and removed dynamically
  - Multiple Instances of the same script are allowed



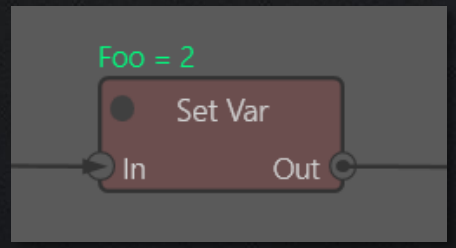
Statescript Nodes



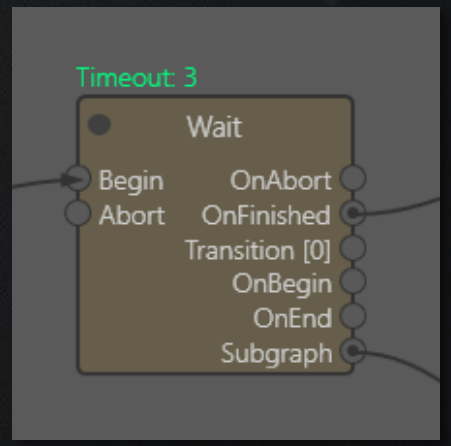
Entry



Condition



Action



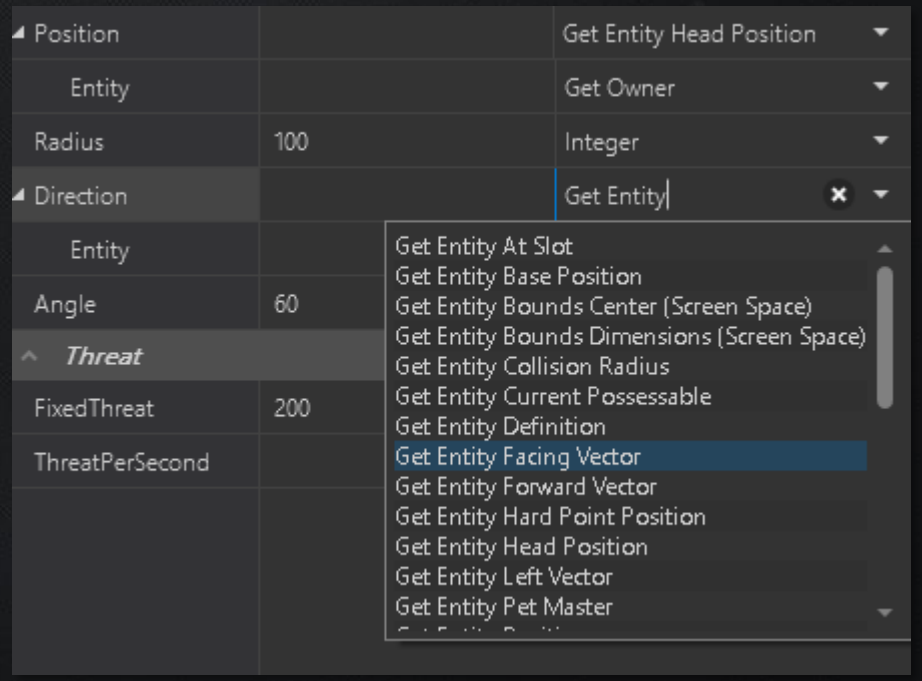
State

Statescript Variables

- Statescript provides bags of **Variables**
  - **Instance** Variables
  - **Owner** Variables
- Primitive value or array of primitive values
- Variables can be “state-defined”

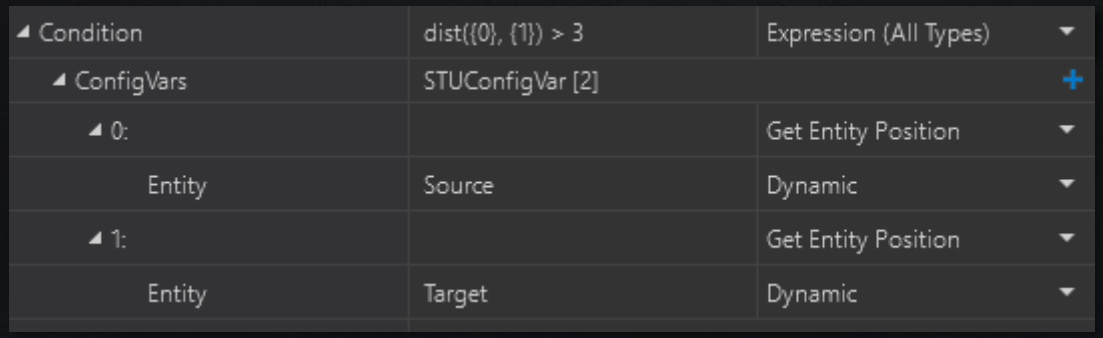
Statescript Properties

- Node behavior is defined by properties
- The scripter chooses from a list of **Config Vars**
- Config Vars may contain nested properties
- Each type is implemented by a C++ function



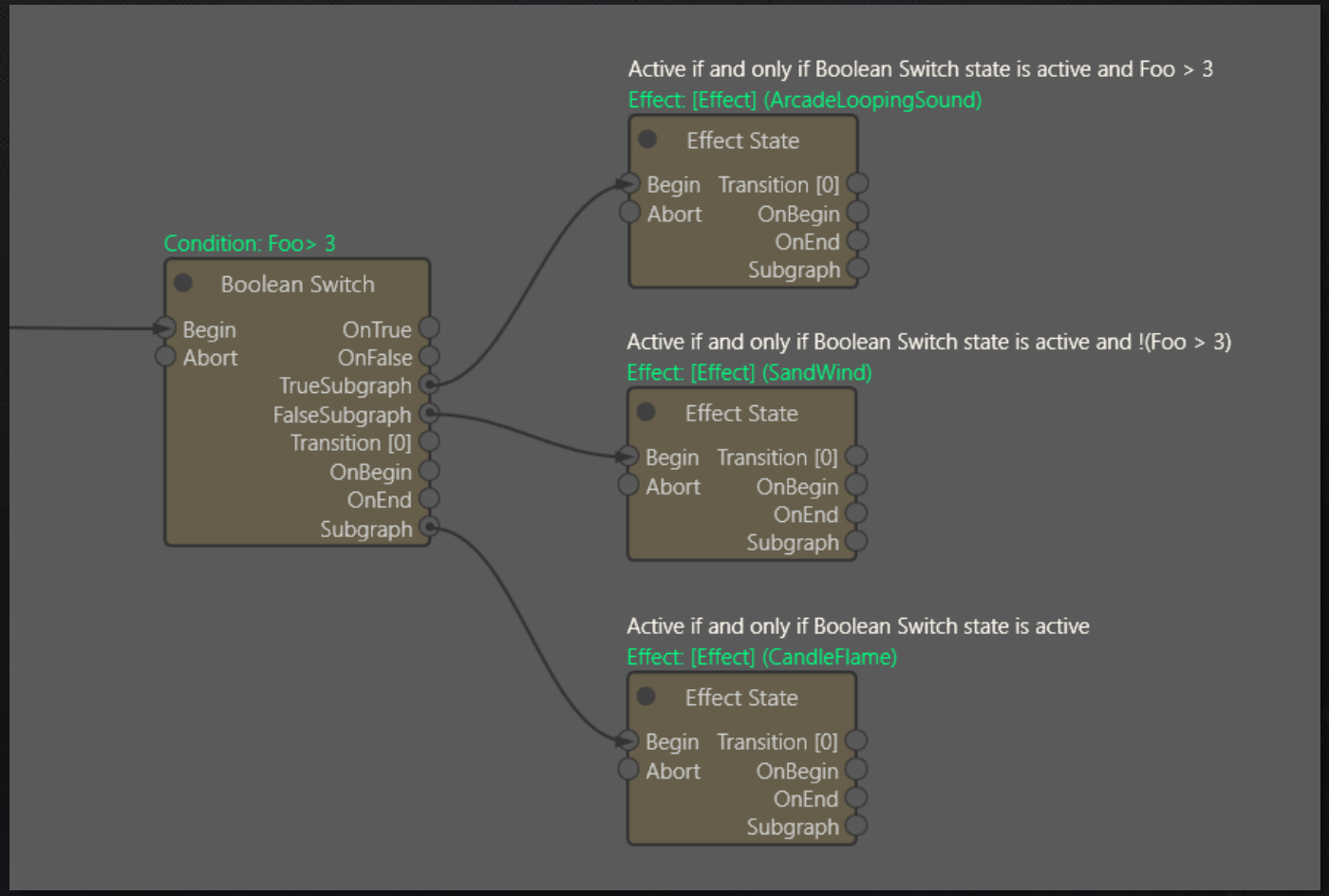
Statescript Properties

- Examples of Config Vars
  - Literals (Bool, Int, Float, Vec, String, etc.)
  - Variables
  - Utilities (Is Hero Select Active, Get Entity Position, etc.)
  - Expressions
    - Foo > 3
    - dist({0}, {1}) > 3



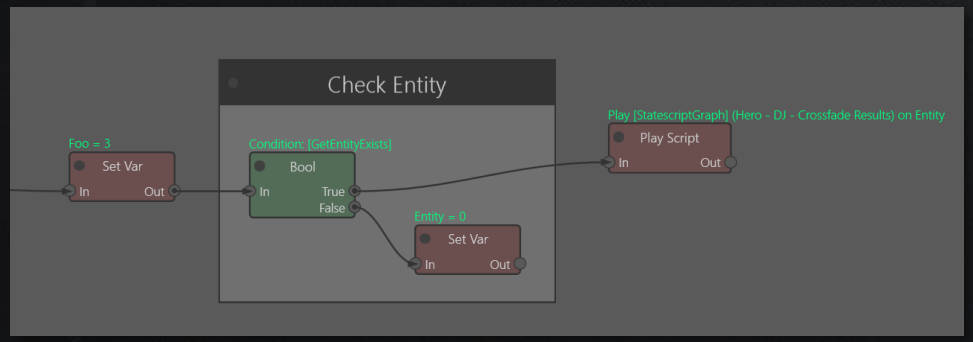
Other Statescript Features

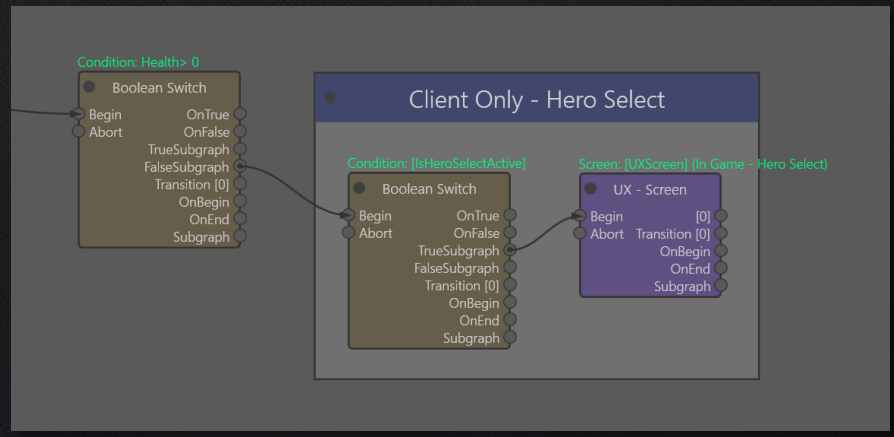
Subgraphs

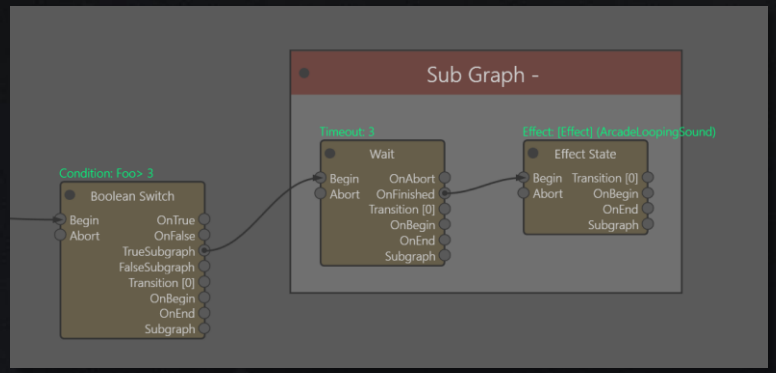


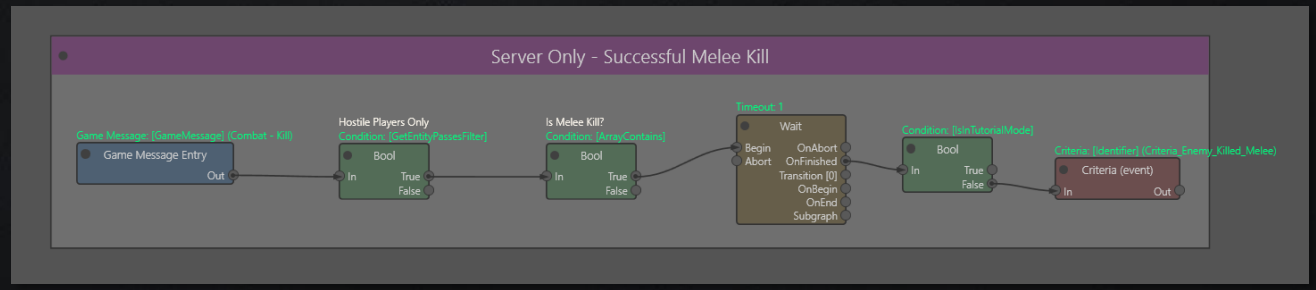
Other Statescript Features

Containers









Statescript Themes

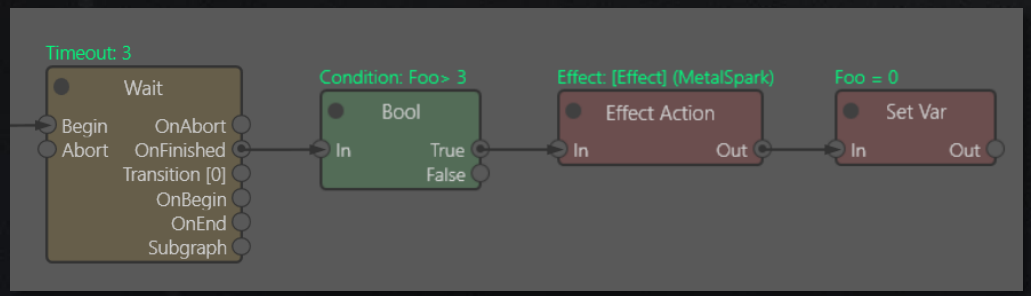
Lifetime guarantees

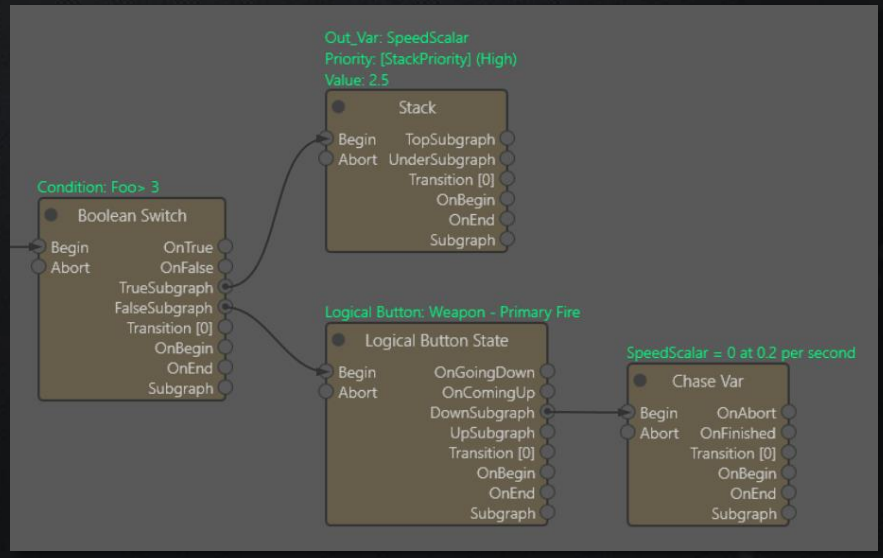
```
virtual void OnDeactivate(DeactivateReason reason) override
{
    Super::OnDeactivate(reason);

    if (m_sprayAdmin)
    {
        u64 key = __BuildKey();
        m_sprayAdmin->GetSingletonContact()->StopMaintainingContact
            (ECSingletonContact_C::Contact::MAINTAINED_USER_STATESCRIPT, key);

        Entity* sprayEntity = m_sprayAdmin->GetEntity(m_sprayHandle);
        if (sprayEntity)
            sprayEntity->ScheduleDestruction(false);
    }
}
```

Statescript Themes

Logic style

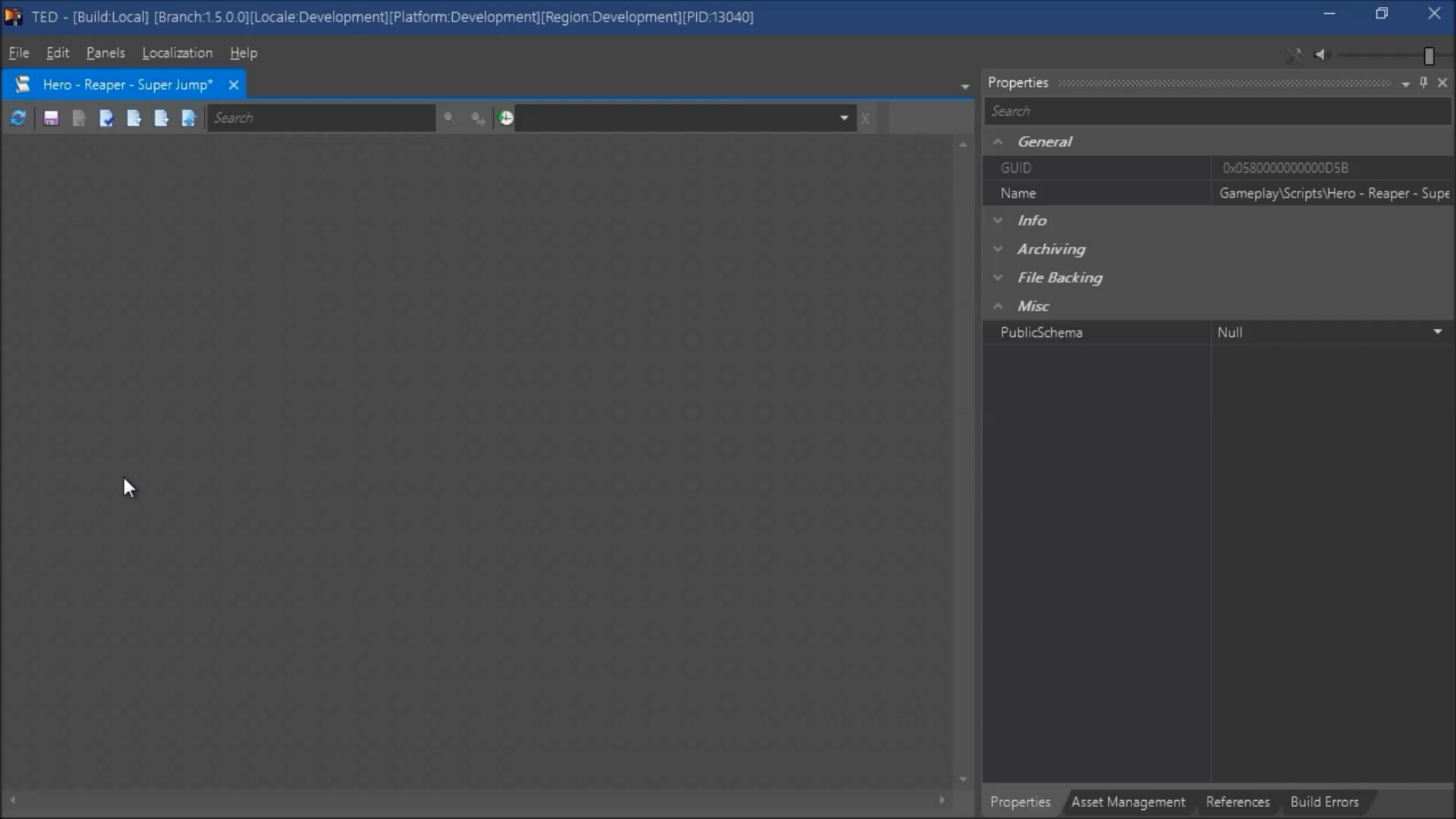
- Imperative (do this, then check this, then do this)
- Declarative (whenever this is the case, then this should be the case)





Reaper Secondary Fire Ability

Reaper Secondary Fire Ability

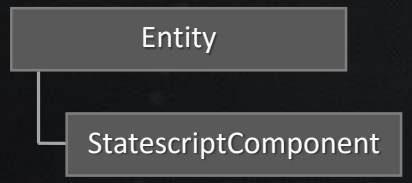

TED - [Build:Local] [Branch:1.5.0.0][Locale:Development][Platform:Development][Region:Development][PID:13040]
File
Edit
Panels
Localization
Help
Hero - Reaper - Super Jump*
Search
Properties
Search
General
GUID
0x0580000000000D5B
Name
Gameplay\Scripts\Hero - Reaper - Supe
Info
Archiving
File Backing
Misc
PublicSchema
Null
Properties
Asset Management
References
Build Errors


View: Server
LOD: Instance Summary
Entity ID: SE:{40000014} (yourself)
Entity Def: Hero - Reaper - Gameplay
Instance: Hero - Reaper - Super Jump (ID: 26)
Instance Variables
ReadyToLaunch: false
...
Wait
Begin
Abort
OnAbort
OnFinished
Transition
OnBegin
OnEnd
Subgraph
SetVar
In Out
Camera3P
Begin
Abort
Transition
OnBegin
OnEnd
Subgraph
LogicalButton
Begin
Abort
OnGoingDown
OnComingUp
DownSubgraph
UpSubgraph
Transition
OnBegin
OnEnd
Subgraph
Bool
In
True
False
MovementMod
In
Out
SetVar
In Out
CF: 10593


View: Server
LOD: Instance Summary
Entity ID: SE:{80000038} (yourself)
Entity Def: Hero - Tracer - Gameplay
Instance: Hero - Tracer - Automatic Pistols (ID: 10)
Instance Variables
AmmoClipMax: 40
AmmoInClip: 40
BlockFiring: 0
IsFiring: 0
RecoveryDue: 0
ReloadAmmoAtAnimPercent: 0.7000
ReloadRequested: 0
ReloadTime: 1
SpreadMax: 3.5000
SpreadMin: 0.0000
SpreadProgress: 0.0000
SpreadRecovery: 0
WeaponEntityLH1: (1, 10, 0, 0)
WeaponEntityRH1: (1, 10, 1, 0)
Owner Variables
AbilityLock: 0
BaseHealth: 150
BeingBuffed: 0
BeingHealed: 0
CanResurrect: 0
ClearDoTs: 0
ClientReticleEnabled: 0
ConnectedPlayer: 0
CurWeaponIndex: 1
DesiredWeaponIndex: 1
...
WeaponEntityRH1: (1, 10, 1, 0)
WeaponEntityLH1: (1, 10, 0, 0)
Owner.WeaponSpreadAngle: 0.0000
CF: 232167

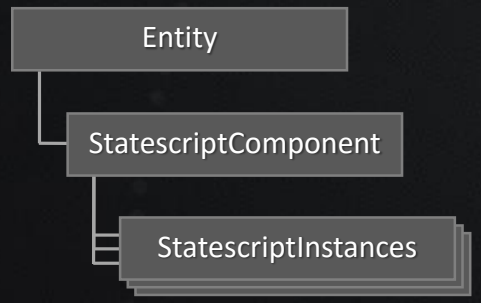
How Statescript is Implemented

Core Runtime



- Timing is based on **Command Frames**
- Playing scripts requires a **StatescriptComponent**

Core Runtime



- **StatescriptComponent**

```
int m_internalCommandFrame;
Array<StatescriptInstance*> m_instances;
StatescriptVarBag m_ownerVars;
StatescriptSyncMgr m_syncMgr;
```

- **StatescriptInstance**

```
int m_instanceID;
const STUStatescriptGraph* m_stuGraph;
Array<StatescriptState*> m_states;
List<StatescriptEvent*> m_futureEvents;
StatescriptVarBag m_instanceVars;
```

States

- **StatescriptState**
  - Base class provides utility functions

```
GetBool
GetInt
GetFloat
GetVec
GetEntityID
```

```
EnqueueTimerEvent
EnqueueAbortStateEvent
EnqueueFinishStateEvent
SetFrameTickingEnabled
```

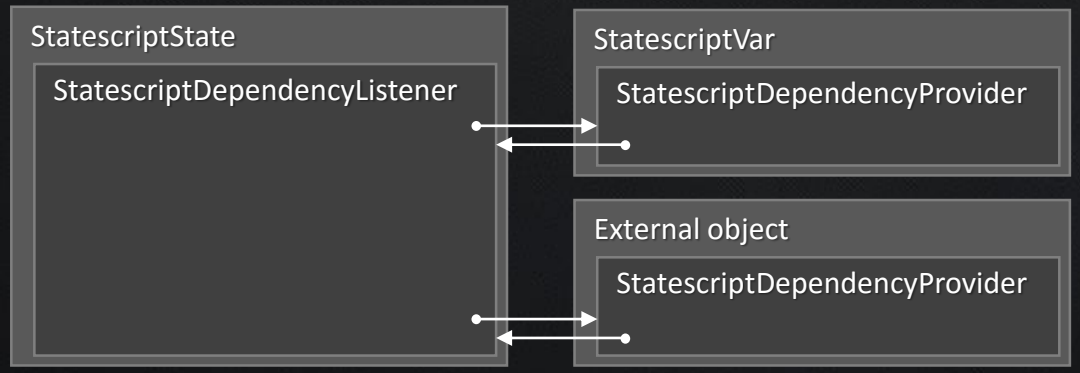
States

- **StatescriptState**
  - Base class provides utility functions
  - Base class provides virtual functions

```
OnActivate
OnDeactivate
OnTimerEvent
OnFrameTick
GetStateDefinedValue
OnInternalDependencyChanged
OnBecameActiveThisTick
OnBecameInactiveThisTick
OnActivationChangedThisTick
```

States

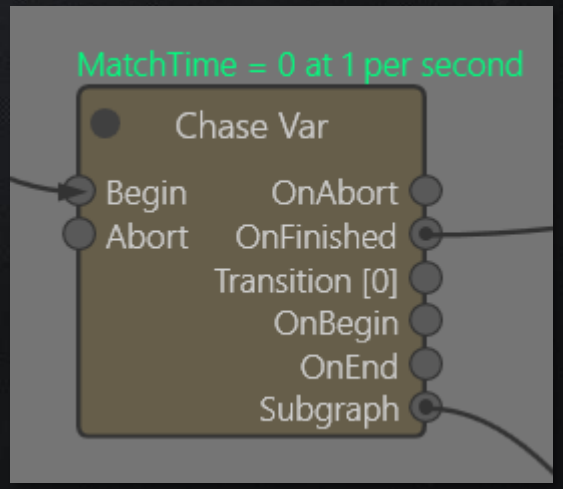
- **StatescriptState**
  - **StatescriptDependencyListener <–> StatescriptDependencyProvider**



- Listener is populated lazily
- Providers call **OnInternalDependencyChanged** on states

Variables

- **StatescriptVarBags** contain a table of **StatescriptVars**
  - Key is a 16-bit **Identifier**
- **StatescriptVar**
  - **StatescriptVarPrimitive**
  - **StatescriptVarPrimitiveArray**
- Primitives are stored using a 128-bit union type
- StatescriptVars are dependency providers
- StatescriptVars optionally reference a StatescriptState



Structured Data

- Assets defined by **Structured Data** (stu)
  - Compile .stu files to generate code
  - Supports attributes and runtime reflection

Wait State Data Schema

.stu file

```
class STUStatescriptStateWait : STUStatescriptState
{
    [Constraint(typeof(STUConfigVarNumeric))]
    embed<STUConfigVar> m_timeout;

    embed<STUStatescriptOutputPlug> m_onAbortPlug;
    embed<STUStatescriptOutputPlug> m_onFinishedPlug;
};
```

Wait State Runtime Implementation

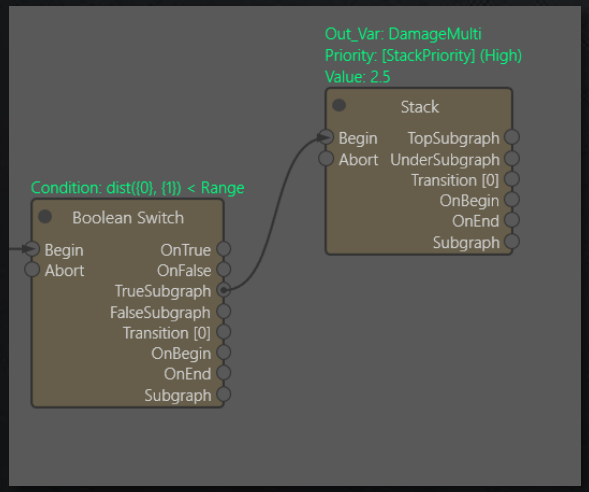
.cpp file

```
class StatescriptStateWait : public StatescriptState
{
    DECLARE_STATESCRIPT_RTTI(StatescriptStateWait, StatescriptState)

    public:
        virtual void OnActivate(ActivateReason) override
        {
            const STUStatescriptStateWait* stu =
                GetSTUStatescriptStateT<STUStatescriptStateWait>();

            EnqueueFinishStateEvent(GetFloat(stu->m_timeout));
        }

        STATESCRIPT_IMPLEMENT_TRAVERSE_ABORT(STUStatescriptStateWait, m_onAbortPlug)
        STATESCRIPT_IMPLEMENT_TRAVERSE_FINISHED(STUStatescriptStateWait, m_onFinishedPlug)
};

IMPLEMENT_STATESCRIPT_FACTORY_AND_RTTI(StatescriptStateWait, STUStatescriptStateWait)
```

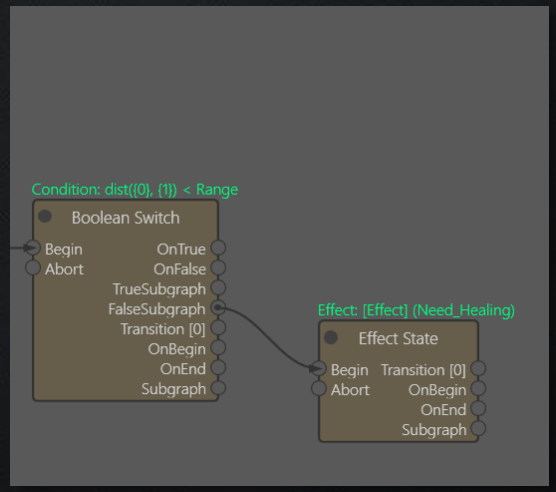
Networking Requirements

Usability

It must be non-invasive to the scripter, abstracting away networking details



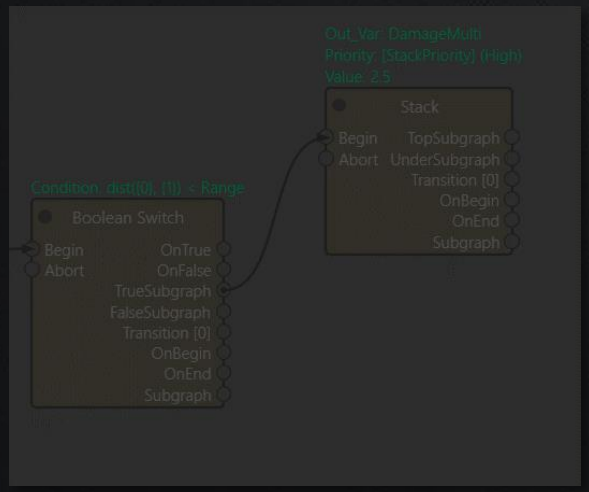
Server?



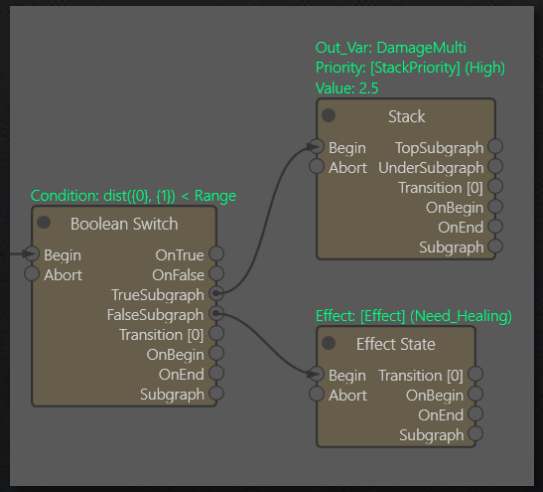
Client?

Usability

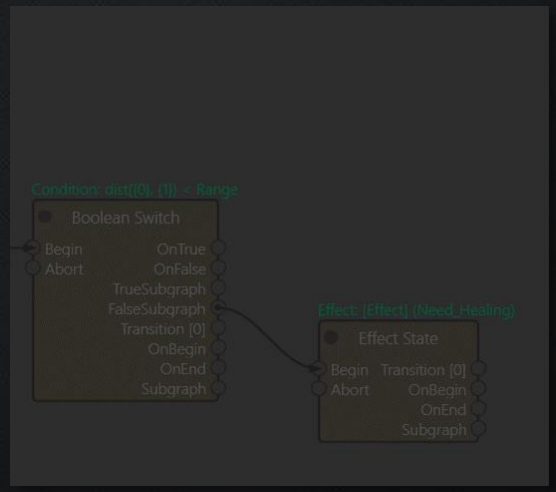
It must be non-invasive to the scripter, abstracting away networking details



Server?



Synchronized



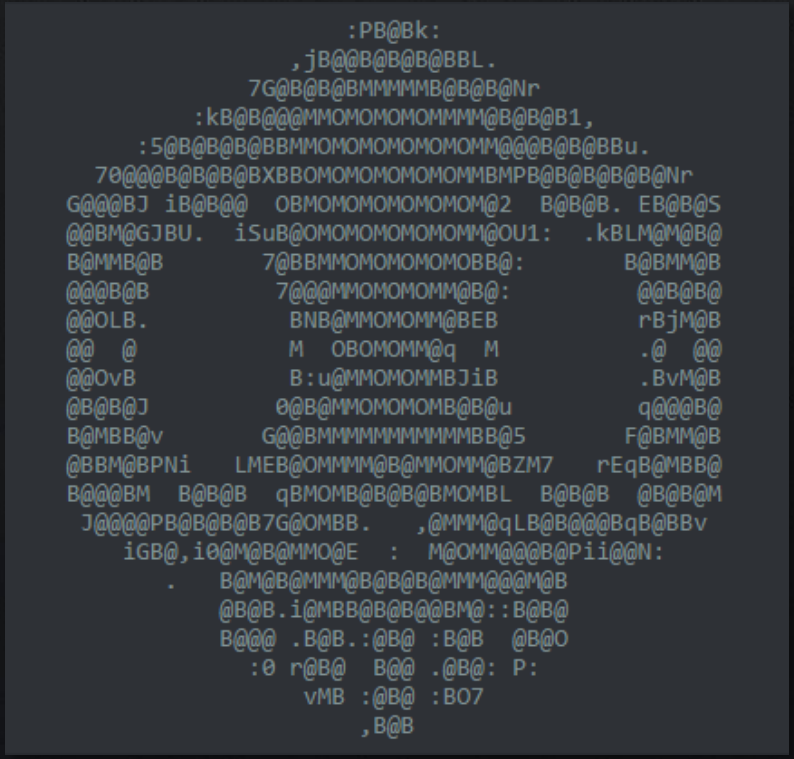
Client?

Responsiveness

It must accommodate responsive gameplay

Security

It must be secure, preventing hackers from influencing server behavior



Efficiency

It must be efficient, allowing the game to function on lower-quality networks

Seamlessness

It must be seamless, minimizing noticeable side effects from networking

Networking Solutions

Synchronized Instances

- Synchronized Instances
  - Server and client describe the same Instance
  - Eventually consistent
- Unsynchronized Instances
  - Receive messages from Synchronized Instances
  - Read variables off of Synchronized Instances
  - Independent

Synchronized Instances

- Scripts played synchronized
  - Weapons
  - Abilities
  - Emotes
  - Game modes
  - Map Entities
- Scripts played unsynchronized
  - Menus
  - Hero Gallery
  - End-of-match flow
  - Music

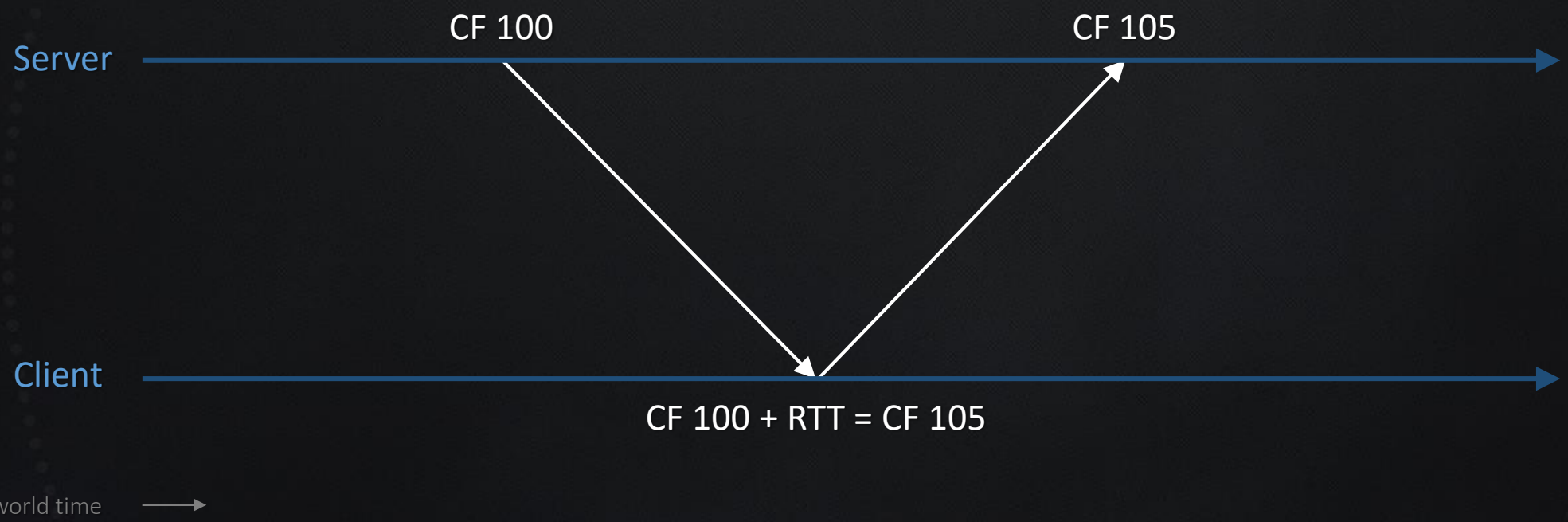
Local and Remote Instances

- A single networked Entity is allowed to be controlled by the player
  - This Entity is **local**
  - All other networked Entities are **remote**
- The server keeps track of which Entities are local to which clients

Server Authority

- Statescript is server-authoritative
  - Communication is mostly server-to-client
  - Button input and aim is client-to-server

Client Input

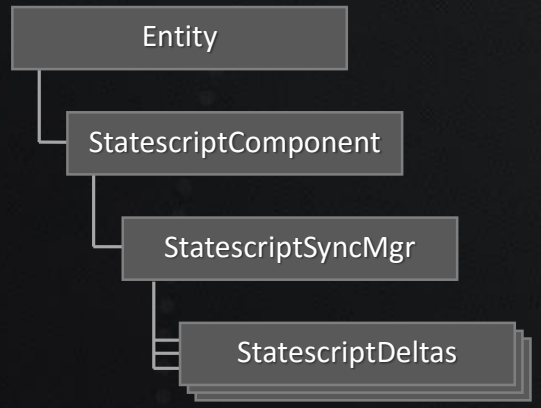


Round-Trip Time (RTT) = 5 Command Frames (80 ms)

Server Synchronization Overview

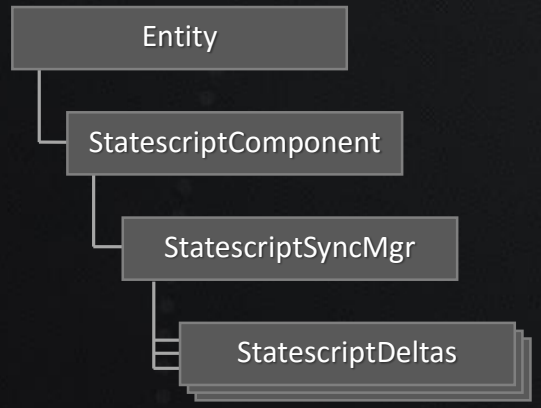
- Gather input from clients
- Simulate
  - Store changes in **StatescriptDeltas**
- Send the deltas to the clients

StatescriptDeltas



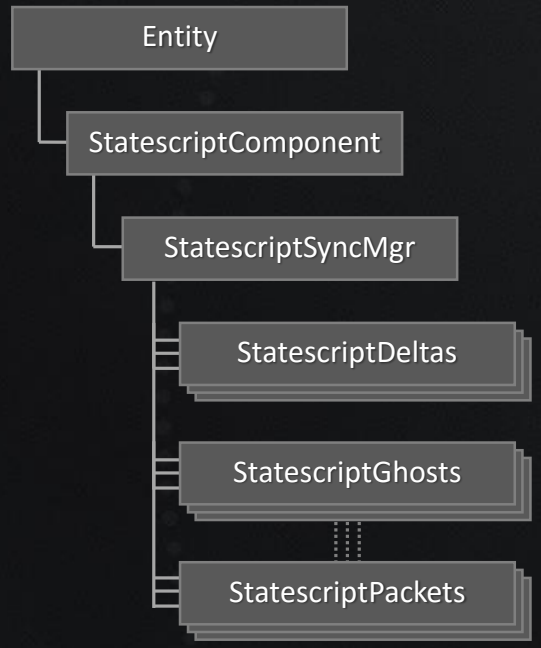
- **StatescriptDeltas**
  - Command Frame
  - Array of all synchronized Instances that changed
    - Instance ID
    - Flags for creation/destruction
    - Array of all Instance Variables that changed
      - Identifier
      - For arrays: Affected index range
    - Array of indices of States that changed
    - Array of indices of Actions that executed
  - Array of all Owner Variables that changed

StatescriptDeltas



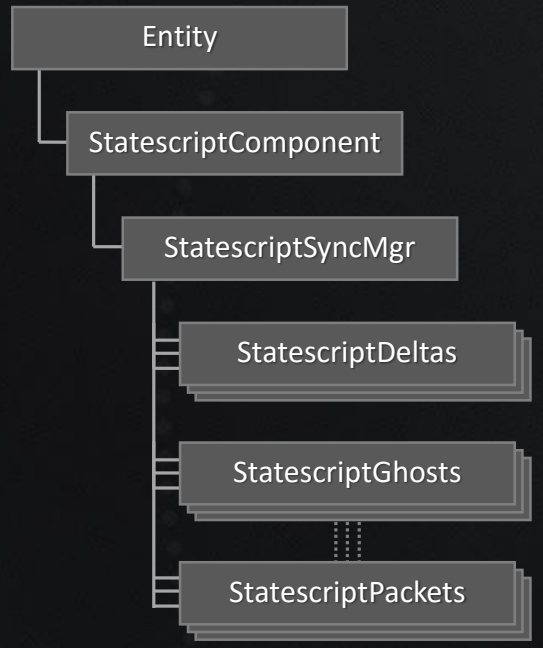
- A StatescriptDelta is stored until all clients have acknowledged receipt of its Command Frame

StatescriptGhosts



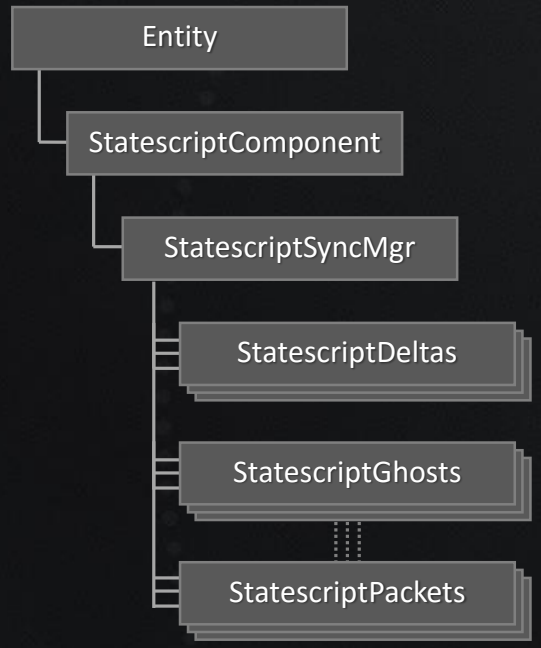
- **StatescriptGhost**
  - Client ID
  - Command Frame last acknowledged
  - Array of pointers to outstanding **StatescriptPackets**
- A StatescriptGhost exists until its client disconnects

StatescriptPackets



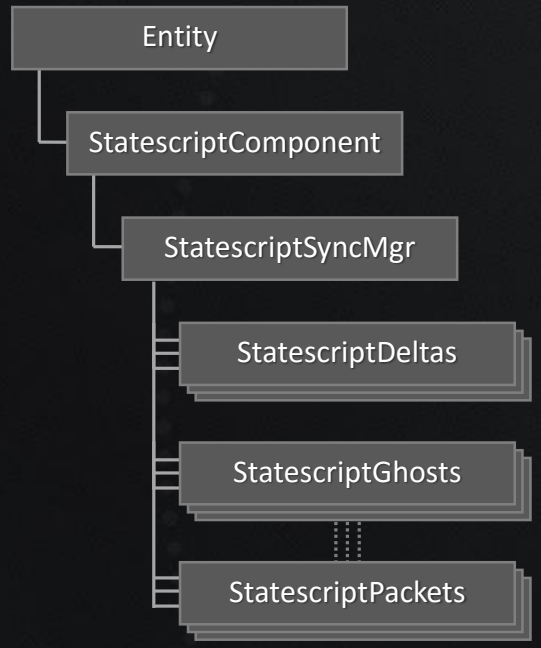
- **StatescriptPacket**
  - Local/Remote Flag
  - Command Frame range (start and end)
  - Payload to transmit
    - Create a union of all StatescriptDeltas in the Command Frame range
    - Serialize the current values of the objects referenced by this union
      - If the Command Frame range starts at 0, then just send the current values of all objects.
- StatescriptPackets are stored and reused

StatescriptPackets



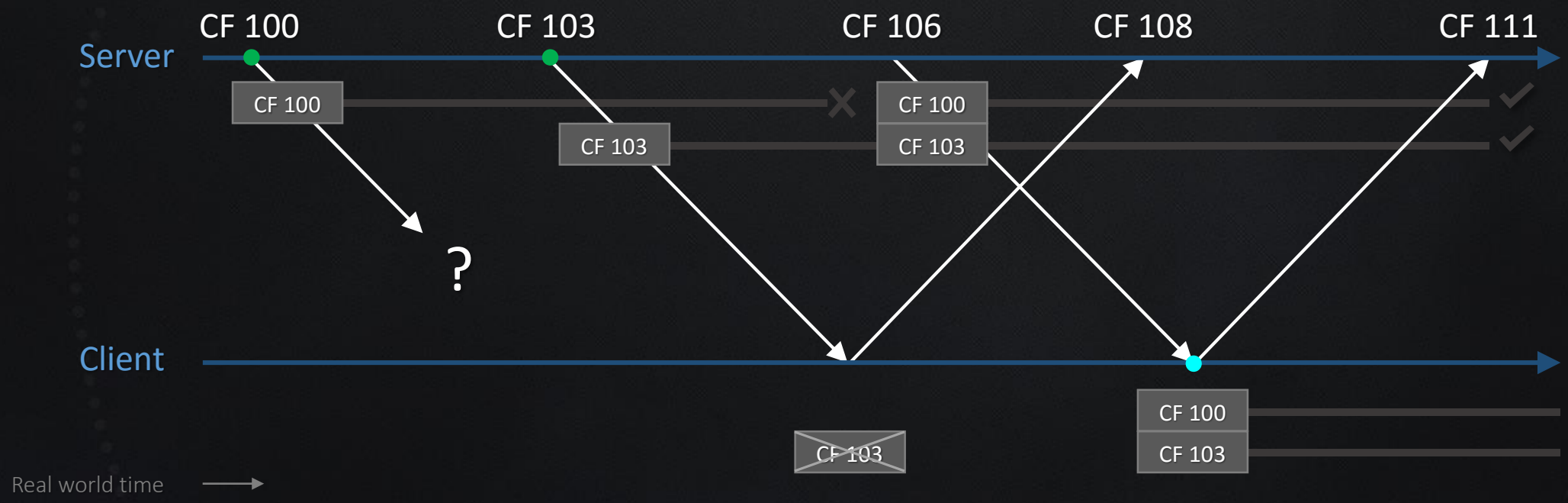
- A StatescriptPacket is stored until all clients have acknowledged receipt of its end Command Frame

Server Synchronization Recap



- **StatescriptDeltas**
- **StatescriptGhosts**
- **StatescriptPackets**

Server Synchronization Demo



Round-Trip Time (RTT) = 5 Command Frames (80 ms)

Green = Simulation
Cyan = Replication

Client Synchronization Overview

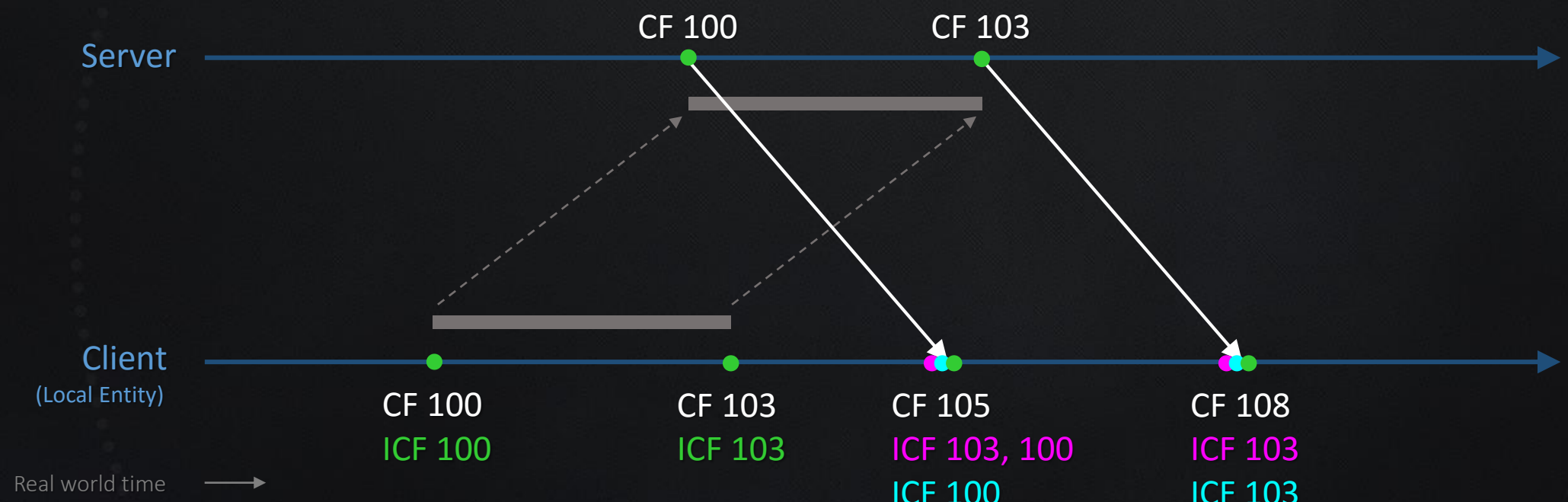
- The local entity stores input and predictions
- When receiving a StatescriptPacket
  - Send acknowledgement
  - Ignore packet if redundant or out-of-order
  - If remote
    - Replicate
  - If local
    - Rollback
    - Replicate
    - Simulate

Receiving a Remote StatescriptPacket

Round-Trip Time (RTT) = 5 Command Frames (80 ms)

Green = Simulation
Cyan = Replication

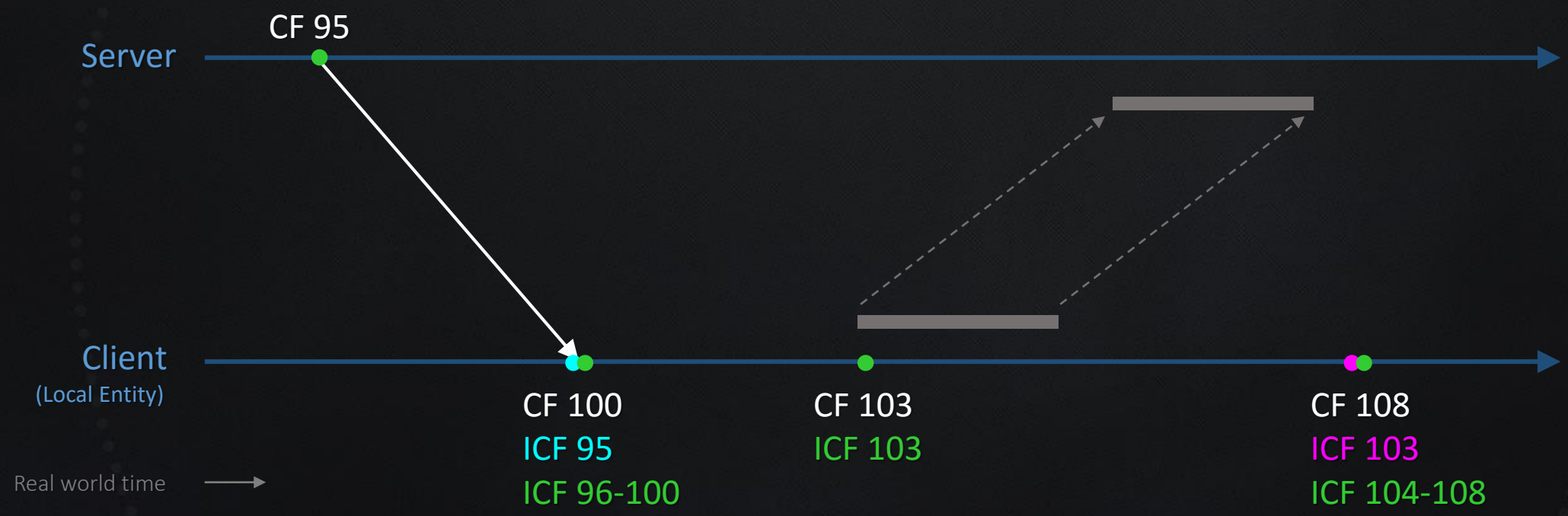
Receiving a Predicted StatescriptPacket



Round-Trip Time (RTT) = 5 Command Frames (80 ms)

Green = Simulation
Magenta = Rollback
Cyan = Replication

Receiving a Mispredicted StatescriptPacket



Round-Trip Time (RTT) = 5 Command Frames (80 ms)

Green = Simulation
Magenta = Rollback
Cyan = Replication


View: Server
LOD: Instance Summary
Entity ID: SE:{40000014} (yourself)
Entity Def: Hero - Reaper - Gameplay
Instance: Hero - Reaper - Super Jump (ID: 26)
Instance Variables
ReadyToLaunch: false
...
Wait
Begin
Abort
OnAbort
OnFinished
Transition
OnBegin
OnEnd
Subgraph
SetVar
In Out
Camera3P
Begin
Abort
Transition
OnBegin
OnEnd
Subgraph
LogicalButton
Begin
Abort
OnGoingDown
OnComingUp
DownSubgraph
UpSubgraph
Transition
OnBegin
OnEnd
Subgraph
Bool
In
True
False
MovementMod
In
Out
SetVar
In Out
CF: 20302

Rollback/Replicate/Simulate C++ APIs

- Synchronization Utilities for StatescriptStates

```
OnActivate(ActivationReason reason)
OnDeactivate(DeactivationReason reason)
IsRollingBack()
IsReplicating()
IsRollingForth()
GetActivationSummary()
OnBecameActiveThisTick()
OnBecameInactiveThisTick()
PutUpdate(DataStore* dataStore)
GetUpdate(DataStore* dataStore)
```

Rollback/Replicate/Simulate C++ APIs

- Synchronization Utilities for StatescriptActions

```
DoRollback()
GetRollbackStorage()
```

Rollback/Replicate/Simulate Mirrored Data

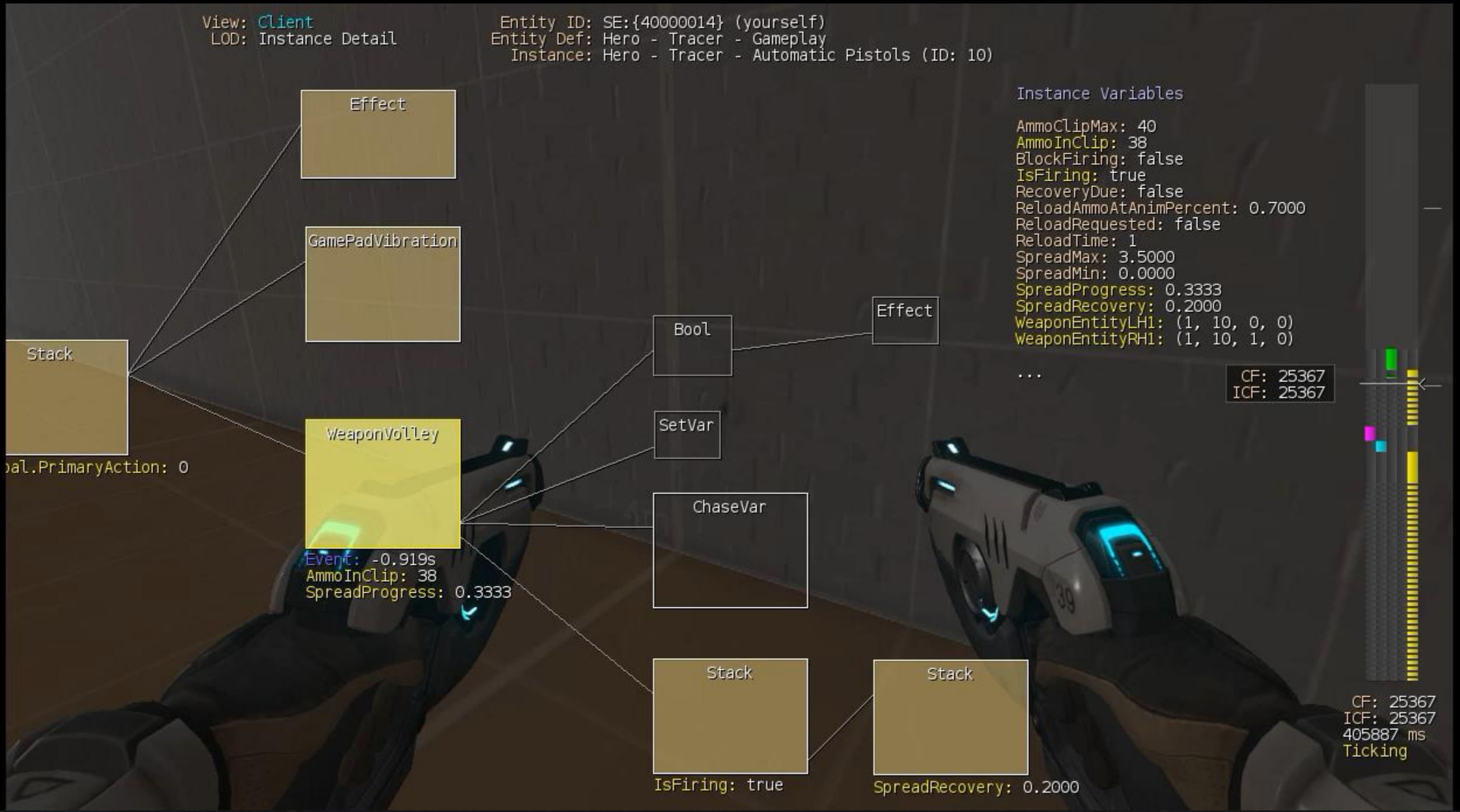
~~PutUpdate(DataStore* dataStore)~~
~~GetUpdate(DataStore* dataStore)~~
WriteMirroredData()
ReadMirroredData()

.stu file

```
mirror MStatescriptStateChaseVar
{
    Vec3A m_curVal;
    s64 m_lastTickTime;
    s32 m_timeRemainingMS;
    bool m_isVec;
    bool m_hasDuration;
    bool m_hasReachedDestination;
};
```

.cpp file

```
MStatescriptStateChaseVar* data = WriteMirrorData();

Vec3A curVal = data->GetCurVal();

bool isVec;
Vec3A destination = __BuildVal(
    stu->m_destination,
    stu->m_destinationBone,
    stu->m_destinationObject,
    &isVec);

data->SetCurVal(destination);
data->SetTimeRemainingMS(0);
data->SetHasReachedDestination(true);
```


View: Client
LOD: Instance Detail
Entity ID: SE:{40000014} (yourself)
Entity Def: Hero - Tracer - Gameplay
Instance: Hero - Tracer - Automatic Pistols (ID: 10)
Effect
GamePadVibration
Stack
bal.PrimaryAction: 0
WeaponVolley
Event: -0.919s
AmmoInClip: 38
SpreadProgress: 0.3333
Bool
Effect
SetVar
ChaseVar
Stack
IsFiring: true
Stack
SpreadRecovery: 0.2000
Instance Variables
AmmoClipMax: 40
AmmoInClip: 38
BlockFiring: false
IsFiring: true
RecoveryDue: false
ReloadAmmoAtAnimPercent: 0.7000
ReloadRequested: false
ReloadTime: 1
SpreadMax: 3.5000
SpreadMin: 0.0000
SpreadProgress: 0.3333
SpreadRecovery: 0.2000
WeaponEntityLH1: (1, 10, 0, 0)
WeaponEntityRH1: (1, 10, 1, 0)
...
CF: 25367
ICF: 25367
CF: 25367
ICF: 25367
405887 ms
Ticking

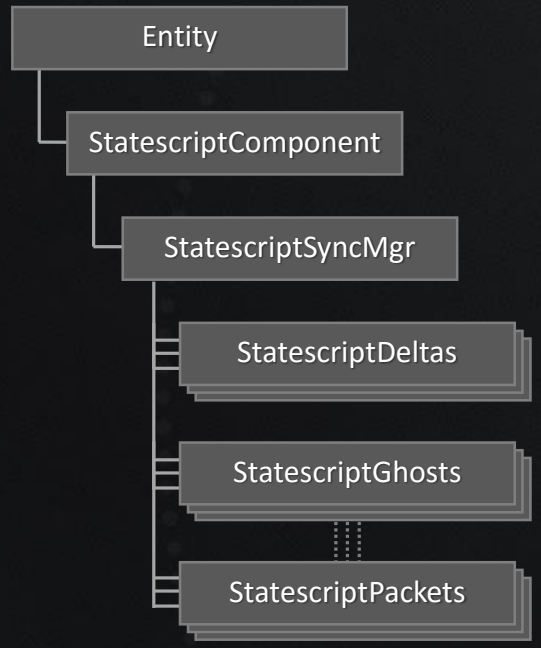
Rollback/Replicate/Simulate Historical Data

- Historical data
  - All variables and states on the local Entity
  - Button input and aim
  - Positions and poses for all Entities
- No historical data
  - Variables and states on remote Entities
  - Data from other Entity Components (such as for health and filtering)

Networking Recap

- ✓ Usability
- ✓ Responsiveness
- ✓ Security
- • Efficiency
- ✓ Seamlessness

Efficiency Overview



- **StatescriptDeltas**
- **StatescriptGhosts**
- **StatescriptPackets**

Efficiency Overview

- Statescript analyzes scripts to discover synchronization requirements
- Local Entities
  - StatescriptPackets must contain everything
- Remote Entities
  - Remote Entities do not simulate Statescript Instances
  - StatescriptPackets only need
    - States and Actions that remote Instances care about
    - Variables referenced by those States and Actions

Efficiency Attributes

- **[STUStatescriptNode::SYNC_ALL]**

.stu file

```
[STUStatescriptNode::SYNC_ALL]
class STUStatescriptStateRaycast : STUStatescriptState
{
    [Constraint(typeof(STUConfigVarVecBase))] embed<STUConfigVar> m_startPosWS;
    [Constraint(typeof(STUConfigVarVecBase))] embed<STUConfigVar> m_dirWS;
    [Constraint(typeof(STUConfigVarNumeric))] embed<STUConfigVar> m_castLength;
    [Constraint(typeof(STUConfigVarEntityID))] array<embed<STUConfigVar>> m_ignoreEntities;
    [Constraint(typeof(STUConfigVarNumeric))] embed<STUConfigVar> m_castRadius;
    embed<STUConfigVarFilter> m_castFilter;
    embed<STUConfigVarDynamic> m_out_hitSomething;
    embed<STUConfigVarDynamic> m_out_hitEntity;
    embed<STUConfigVarDynamic> m_out_hitPointWS;
    embed<STUConfigVarDynamic> m_out_hitNormal;
};
```

Efficiency Attributes

- **[STUStatescriptNode::SYNC_ALL]**
  - With attribute:
    - StatescriptStates transmit locally and remotely
    - StatescriptActions transmit locally and remotely
    - Runtime objects may opt out of transmitting
  - Without attribute:
    - StatescriptStates only transmit locally
    - StatescriptActions do not transmit

Efficiency Compiling

- Determine nodes that need to be sent to remote clients
- Determine which variables these nodes reference
- Determine how many bits are needed to reference these nodes and variables in a StatescriptPacket

```
array<internal<STUStatescriptNode>> m_remoteSyncNodes;
array<inline<STUStatescriptSyncVar>> m_syncVars;

s32 m_nodesBitCount;
s32 m_statesBitCount;
s32 m_remoteSyncNodesBitCount;
s32 m_syncVarsBitCount;
```


View: Server
LOD: Instance Summary
Entity ID: SE:{80000021}
Entity Def: Hero - Tracer - Gameplay
Instance: Hero - Tracer - Automatic Pistols (ID: 10)
Instance Variables
AmmoClipMax: 40
AmmoInClip: 0
BlockFiring: false
IsFiring: 0
RecoveryDue: false
ReloadAmmoAtAnimPercent: 0.7000
ReloadRequested: true
ReloadTime: 1
SpreadMax: 3.5000
SpreadMin: 0.0000
SpreadProgress: 0.0000
SpreadRecovery: 0.0000
WeaponEntityLH1: (1, 10, 0, 0)
WeaponEntityRH1: (1, 10, 1, 0)
...
AmmoInClip: 40
BlockFiring: true
BlockFiring: false
Top
ReloadRequested: true
Owner.PrimaryAction: 0
ReloadRequested: false
Playing 131115
Event: -0.189s
Owner.PrimaryAction:
Event: -0.489s
RecoveryDue: false
False
False
True
True
Top
Owner.PrimaryAction: 0
AmmoInClip: 38
SpreadProgress: 0.3333
RecoveryDue: true
SpreadRecovery: 0.0000
IsSpreadRecovery: 0.2000
IsFiring: 0
SpreadRecovery: 0.0000
Owner.WeaponSpreadAngle: 0.0000
SpreadProgress: 0.0000
SpreadProgress: 0.0000
CF: 18760

Efficiency Numbers

	Local	Remote
States	463 bits	190 bits
Instance Vars	519 bits	246 bits
Owner Vars	54 bits	54 bits
Events	431 bits	0 bits
Other	561 bits	316 bits
Total	2028 bits 1.0 Kb/s	806 bits 0.4 Kb/s

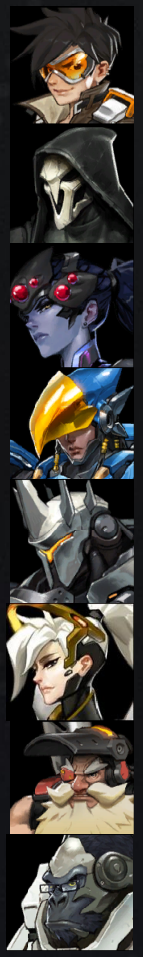
From Tracer firing a full clip and reloading (2 seconds exactly)

Benefits

Rapid Iteration

- Flexible, iterative workflow for designers

Rapid Iteration



State, Wait, Set Var, Chase Var, Boolean Switch, Cosmetic Entity, Logical Button State, Data Flow Mapping, Invisible, Stack, Weapon Volley, Movement Mod Action, Blink Effect, History Builder, History Rewinder, Health Pool, Animation

Track Targets, Modify Health, Target Effect, Phase Out, Movement Mod State, Teleport, Find Ground, Camera 3P
Send Game Message, Effect State, Effect Action, Switch, Watch, Eject Cosmetic Entity

Aim Speed, Camera Transform, Extending Ledge Finder, Move To, Apply Aura, UX - Screen, UX - View Mode,
UX - UI In World, Destroy Entity

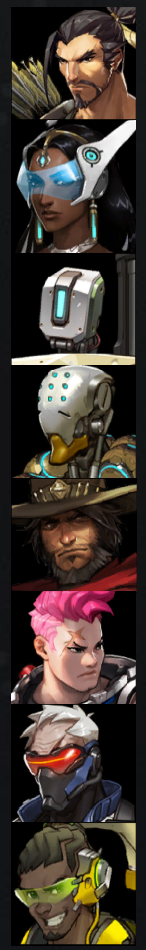
Animated Camera, Steer, Track Slamming, Camera Look At, Play Script, Create Entity State, Shockwave
Suppress Movement Prediction

Beam Effect, Resurrect, Create Entity Action

Override Model Look, Object Placement Validation, Pet

Track Movement

Rapid Iteration

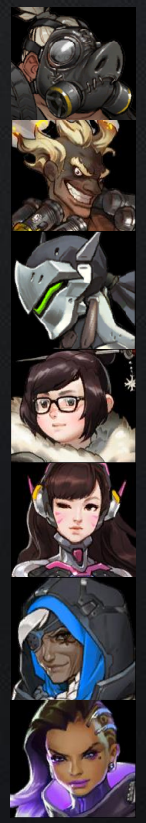


Wall Move

Limit Aim

Combat Mod Filter, Restart Anim

Track Ray, Contact Set State



Raycast Action

Possess Control, Possess View

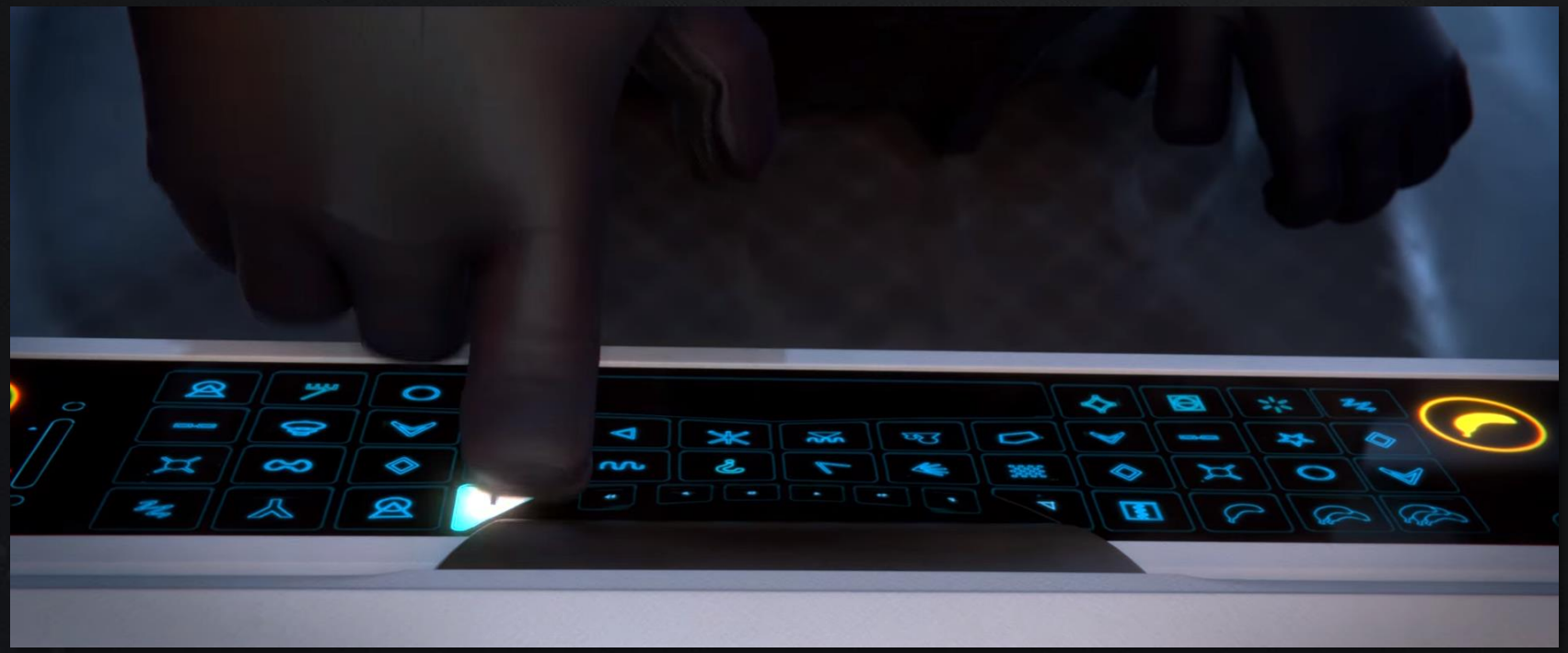
Deflect Projectiles, Deflect Projectiles Effects, Pause Anim

Raycast State

Track Validated Position, Teleport Over Duration

Automatic Syncing and State Machines

- New gameplay features can be added without writing synchronization or state machine logic in code



Fewer Lifetime and Desync Bugs

- Certain types of bugs are less common, including lifetime issues and server-client desyncs

Fewer Lifetime and Desync Bugs

- Certain types of bugs are less common, including lifetime issues and server-client desyncs

Challenges

Up-Front Cost

- Hefty up-front implementation cost for the runtime, an editor, and a debugger

Learning Curve

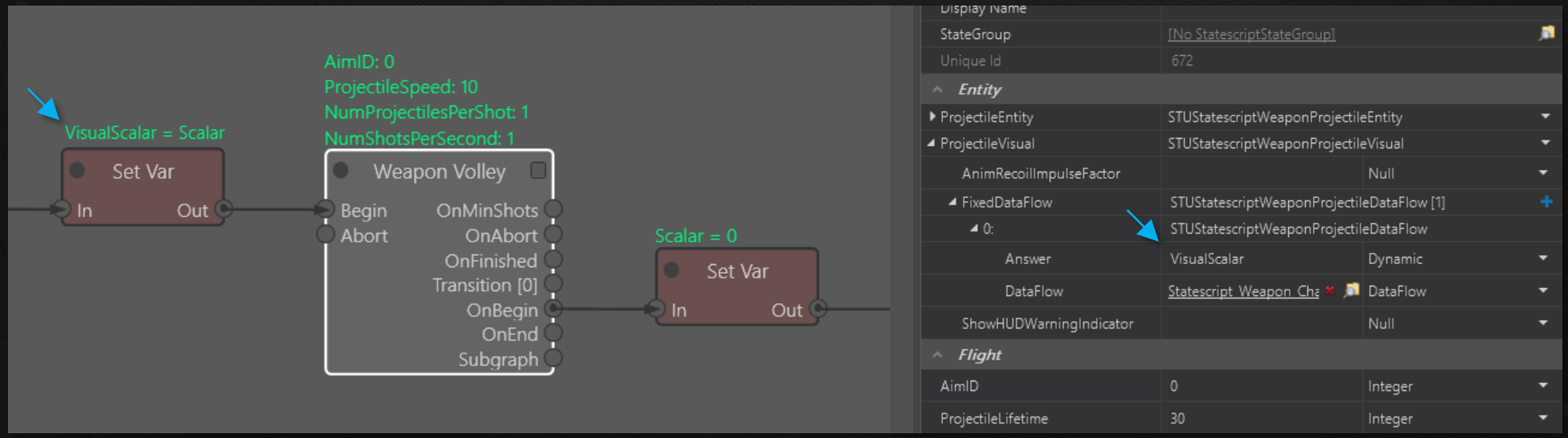
- Learning curve for engineers, particularly when deciding which parts of a feature belong in code versus script

Occasional Workaround

- The eventual-consistency networking model does not provide a perfect blow-by-blow replication

Occasional Workaround

- The eventual-consistency networking model does not provide a perfect blow-by-blow replication



Conclusion

Q&A