
GDC
Back to the Future!
Working with deterministic simulation in 'For Honor'
Jennifer Henry
Gameplay Programmer at Ubisoft Montreal
GAME DEVELOPERS CONFERENCE
MARCH 18–22, 2019 | #GDC19

Jennifer Henry

Gameplay Programmer

Systems team

This talk is about…

- ☑ Making a deterministic game
- ☒ Dedicated Servers
- ☒ Network
- ☒ Gameplay
- ☒ Input Delay

Design constraints

- Precise
- Fast-paced
- Multiplayer

Technical constraints

- Peer-to-peer
- Low bandwidth
- Fair

Lockstep
Deterministic
Simulation

Buffered
Deterministic
Simulation

8 Frames in 16ms: Rollback Networking in 'Mortal Kombat' and 'Injustice 2'

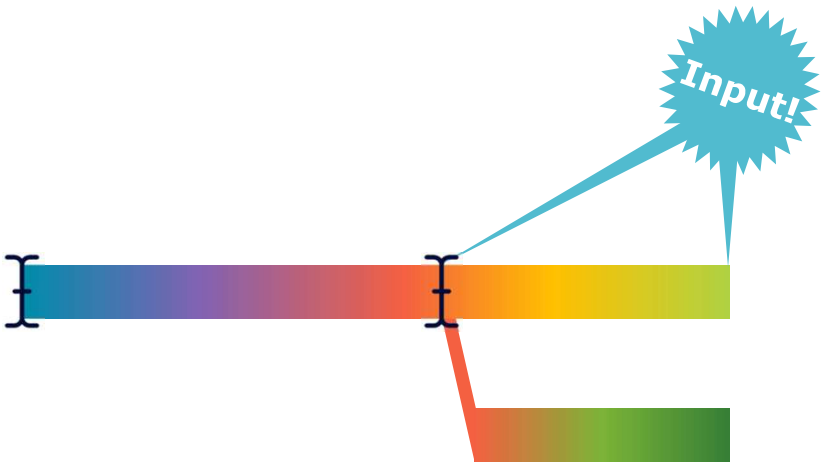
by Michael Stallone

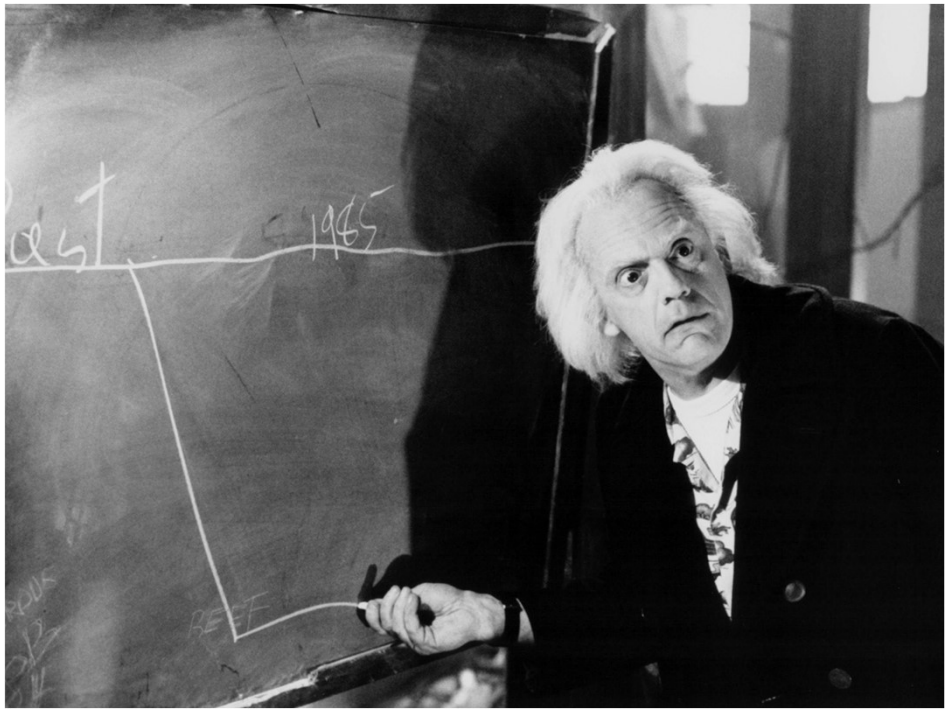
Deterministic simulation?

- Sending only inputs on the network
- Every peer simulate the result
- No authority

DETERMINISM
DESYNCH

Buffered?





History buffer 5 seconds


Determinism
is
hard

Don't assume that your engine is deterministic

Don't try to make everything deterministic

Don't underestimate desynchs

Don't assume that your engine is deterministic

- Floating point
- Physics
- Multithreading
- Time
- Loading / dynamic spawning
- Random

```
__m128 Rsq = _mm_rsqrt_ps(vecW);

__m128 Rsq = _mm_div_ps(_mm_set_ps1(1.0f), _mm_sqrt_ps(vecW));
```

Don't try to make everything deterministic

- Animations
  - Not essential

Don't try to make everything deterministic

- Visual FX, UI, Sound
  - Observers
  - Immediate vs Finalized

Don't underestimate desynchs

"We'll be careful"

"It won't happen that much"

"Must not be that hard"

Step #1 Know when a desynch happens

"DESYNCH!!!"

Track and compare everything all the time

Step #1 Know when a desynch happens

- Gameplay update at time 2000
  - Manager #1 changes state
  - Player #2 changes velocity
  - Bot #6 changes state
  - Bot #6 launches projectile
  - Footman #200 deactivates collision

=> Record set

Step #1 Know when a desynch happens

- Record set validation every 100ms
- CRC sent to a peer and compared
  - If still different after the "finalization time"
    - Pause the game

Step #2 Gather some information

- Output record set to XML file ("trace")
  - Up to 150MB
  - CRC for state changes

Step #2 Gather some information

```
<GameplayUpdate TimeMS="2000" Type="Main">
    <GameplayUpdate TimeMS="2000" Type="GameplayObjectsMT">
        <ProxyUpdate Type="Manager" Index="1">
            <SendUserMessage SenderType="Manager" SenderIndex="1" TargetType="Agent" TargetIndex="1" MessageId="0xfcd02609" MessageDataCRC="0x4caf94a0" />
            <ManagerStateChange ManagerID="1" ManagerState="N/A" ManagerStateCRC="0x34679bfb" />
        </ProxyUpdate>
        <ProxyUpdate Type="Agent" Index="2">
            <SendUserMessage SenderType="Agent" SenderIndex="2" TargetType="Agent" TargetIndex="200" MessageId="0x4854982a" MessageDataCRC="0x8b4d1797" />
            <AgentMovementStateChange TimeMS="2000" AgentIndex="2" Displacement="Local Velocity (0.12, 1.75, 0.00)" Rotation="Lock target 'Actor1_(Actor_Replica_
        </ProxyUpdate>
        <ProxyUpdate Type="Agent" Index="201" />
    </GameplayUpdate>
    <UserMessages>
        <UserMessageBatch TargetProxyType="Agent" TargetProxyTypeUniqueID="1">
            <UserMessage SenderProxyType="Manager" SenderProxyTypeUniqueID="1" MessageId="0xfcd02609" MessageDataCRC="0x4caf94a0" />
            <AgentStateChange AgentID="6" AgentState="N/A" AgentStateCRC="0xc22fa401" />
            <SystemMessages>
                <SystemMessage TargetProxyType="Agent" TargetProxyTypeUniqueID="6" SenderProxyType="Agent" SenderProxyTypeUniqueID="6" MessageType="AddProjectile
                    <ProjectileOperation TimeMS="2000" Type="UserAdd" AgentIndex="6">
                        <Projectile StartTime="2000" ExpirationTime="3500" ProjectileDescriptionObjectID="0x32e3eb00" InitialPosition="[-31.922534943 ,  -42.564
                    </ProjectileOperation>
                </SystemMessage>
            </SystemMessages>
        </UserMessageBatch>
        <UserMessageBatch TargetProxyType="Agent" TargetProxyTypeUniqueID="200">
            <UserMessage SenderProxyType="Agent" SenderProxyTypeUniqueID="2" MessageId="0x4854982a" MessageDataCRC="0x8b4d1797" />
            <AgentPhysicsParametersChange TimeMS="2000" AgentIndex="200" Parameters="capsuleRadius: 0.40, capsuleHeight: 1.40, capsuleYOffset: 0.00, capsuleZOffs
        </UserMessageBatch>
    </UserMessages>
</GameplayUpdate>
```

Gameplay update at time 2000

Manager #1 changes state

Player #2 changes velocity

Bot #6 changes state

Bot #6 launches projectile

Footman #200 deactivates collision

Step #2 Gather some information

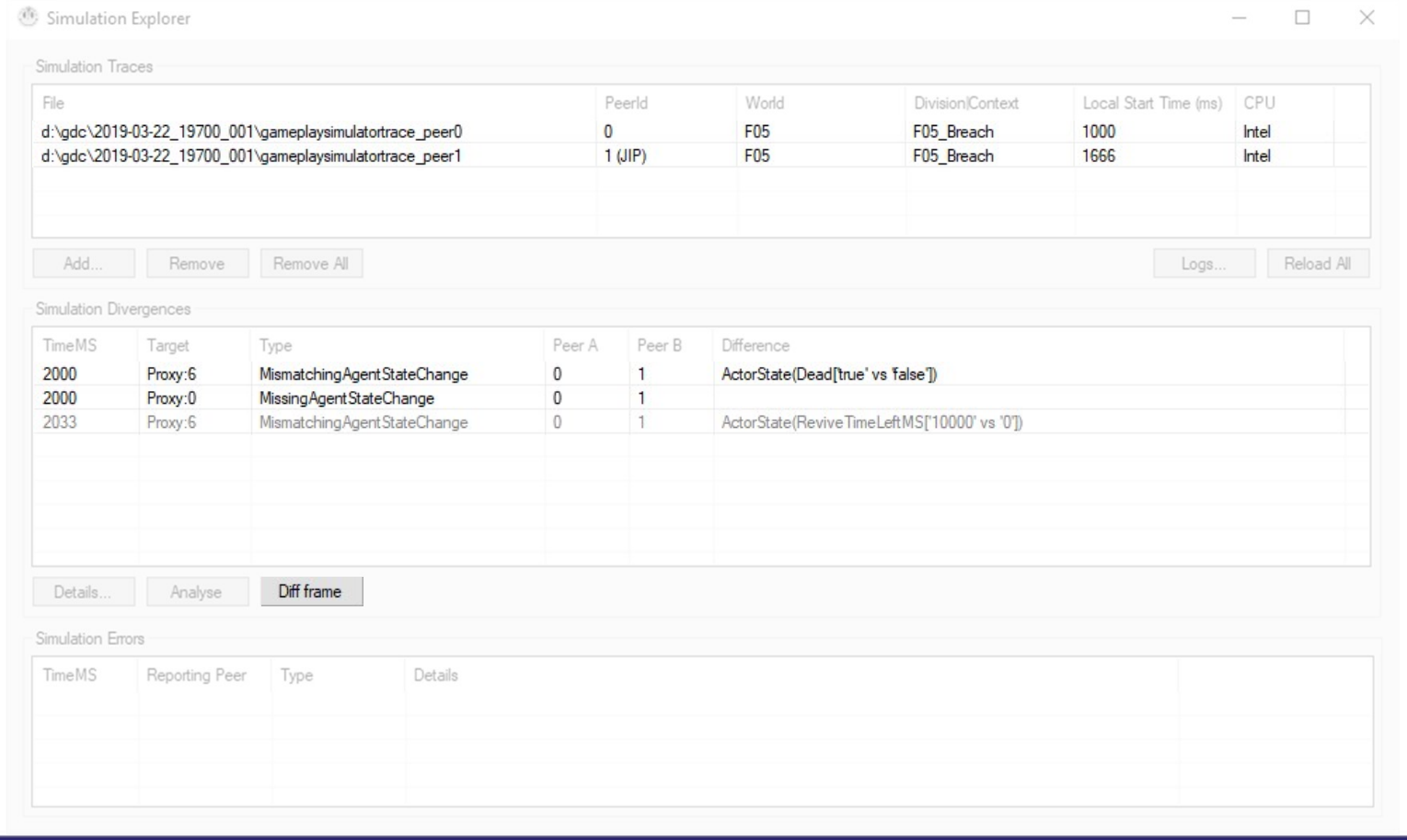
- Ask disagreeing peer to send the details of the operations

- Extract the differences
  - Missing operations
  - Mismatching operations
  - Output the diff to an XML file

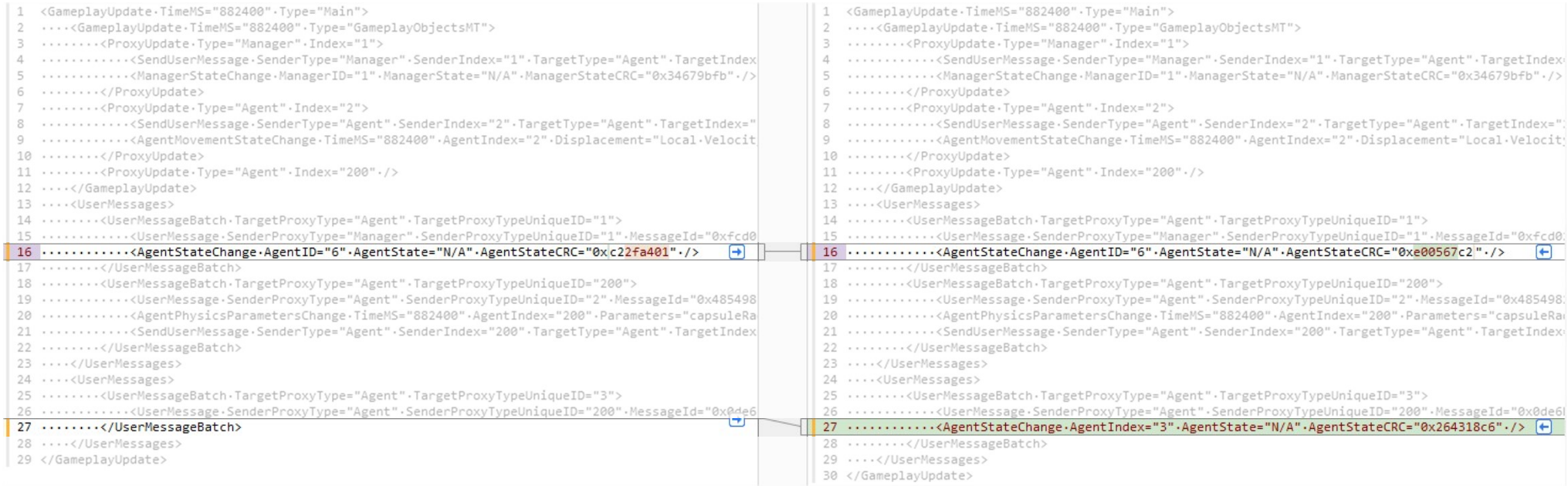
Step #2 Gather some information

```
<?xml version="1.0" encoding="utf-8"?>
<Simulation StartTimeMS="1000" JoinTimeMS="1000" LocalPeerId="0" WorldName="F05" ContextName="F05_Breach" ChipManufacturer="Intel">
    <Divergence Type="MismatchingAgentStateChange" PeerId="1" TimeMS="2000" ProxyIndex="6" Summary="ActorState(Dead['true' vs 'false']) ">
        <SubState Name="ActorState">
            <Difference Property="Dead" Local="true" Remote="false" />
        </SubState>
    </Divergence>
    <Divergence Type="MissingAgentStateChange" PeerId="1" TimeMS="2000" ProxyIndex="0" />
    <Divergence Type="MismatchingAgentStateChange" PeerId="1" TimeMS="2033" ProxyIndex="6" Summary="ActorState(ReviveTimeLeftMS['10000' vs '0']) ">
        <SubState Name="ActorState">
            <Difference Property="ReviveTimeLeftMS" Local="10000" Remote="0" />
        </SubState>
    </Divergence>
</Simulation>
```

Step #3 Visualize this information



Step #3 Visualize this information



```
<GameplayUpdate TimeMS="882400" Type="Main">
    <GameplayUpdate TimeMS="882400" Type="GameplayObjectsMT">
        <ProxyUpdate Type="Manager" Index="1">
            <SendUserMessage SenderType="Manager" SenderIndex="1" TargetType="Agent" TargetIndex
            <ManagerStateChange ManagerID="1" ManagerState="N/A" ManagerStateCRC="0x34679bfb" />
        </ProxyUpdate>
        <ProxyUpdate Type="Agent" Index="2">
            <SendUserMessage SenderType="Agent" SenderIndex="2" TargetType="Agent" TargetIndex="
            <AgentMovementStateChange TimeMS="882400" AgentIndex="2" Displacement="Local Velocit
        </ProxyUpdate>
        <ProxyUpdate Type="Agent" Index="200" />
    </GameplayUpdate>
    <UserMessages>
        <UserMessageBatch TargetProxyType="Agent" TargetProxyTypeUniqueID="1">
            <UserMessage SenderProxyType="Manager" SenderProxyTypeUniqueID="1" MessageId="0xfcd0
            <AgentStateChange AgentID="6" AgentState="N/A" AgentStateCRC="0xc22fa401" />
        </UserMessageBatch>
        <UserMessageBatch TargetProxyType="Agent" TargetProxyTypeUniqueID="200">
            <UserMessage SenderProxyType="Agent" SenderProxyTypeUniqueID="2" MessageId="0x485498
            <AgentPhysicsParametersChange TimeMS="882400" AgentIndex="200" Parameters="capsuleRa
            <SendUserMessage SenderType="Agent" SenderIndex="200" TargetType="Agent" TargetIndex
        </UserMessageBatch>
    </UserMessages>
    <UserMessages>
        <UserMessageBatch TargetProxyType="Agent" TargetProxyTypeUniqueID="3">
            <UserMessage SenderProxyType="Agent" SenderProxyTypeUniqueID="200" MessageId="0x0de6
        </UserMessageBatch>
    </UserMessages>
</GameplayUpdate>
```

```
<GameplayUpdate TimeMS="882400" Type="Main">
    <GameplayUpdate TimeMS="882400" Type="GameplayObjectsMT">
        <ProxyUpdate Type="Manager" Index="1">
            <SendUserMessage SenderType="Manager" SenderIndex="1" TargetType="Agent" TargetIndex
            <ManagerStateChange ManagerID="1" ManagerState="N/A" ManagerStateCRC="0x34679bfb" />
        </ProxyUpdate>
        <ProxyUpdate Type="Agent" Index="2">
            <SendUserMessage SenderType="Agent" SenderIndex="2" TargetType="Agent" TargetIndex="
            <AgentMovementStateChange TimeMS="882400" AgentIndex="2" Displacement="Local Velocit
        </ProxyUpdate>
        <ProxyUpdate Type="Agent" Index="200" />
    </GameplayUpdate>
    <UserMessages>
        <UserMessageBatch TargetProxyType="Agent" TargetProxyTypeUniqueID="1">
            <UserMessage SenderProxyType="Manager" SenderProxyTypeUniqueID="1" MessageId="0xfcd0
            <AgentStateChange AgentID="6" AgentState="N/A" AgentStateCRC="0xe00567c2" />
        </UserMessageBatch>
        <UserMessageBatch TargetProxyType="Agent" TargetProxyTypeUniqueID="200">
            <UserMessage SenderProxyType="Agent" SenderProxyTypeUniqueID="2" MessageId="0x485498
            <AgentPhysicsParametersChange TimeMS="882400" AgentIndex="200" Parameters="capsuleRa
            <SendUserMessage SenderType="Agent" SenderIndex="200" TargetType="Agent" TargetIndex
        </UserMessageBatch>
    </UserMessages>
    <UserMessages>
        <UserMessageBatch TargetProxyType="Agent" TargetProxyTypeUniqueID="3">
            <UserMessage SenderProxyType="Agent" SenderProxyTypeUniqueID="200" MessageId="0x0de6
            <AgentStateChange AgentIndex="3" AgentState="N/A" AgentStateCRC="0x264318c6" />
        </UserMessageBatch>
    </UserMessages>
</GameplayUpdate>
```

Step #4 Keep track of desynchs

MissingManagerStateChange - BattlefieldManager

MismatchingAgentStateChange: ActionState.SprintTimeMS

MismatchingAgentProjectile: initialVelocity

Step #5 Debug desynchs

- First investigation - SimulationExplorer
  - Recently touched gameplay system?
  - Repro steps
  - Quick look at the code
- ...

Live validation

- Snapshots taken every second
- CRC sent to a peer and compared
  - If still different after the "finalization time"
    - Try to recover
    - Kick the diverging peer if in minority
    - Disband the session
- Visibility

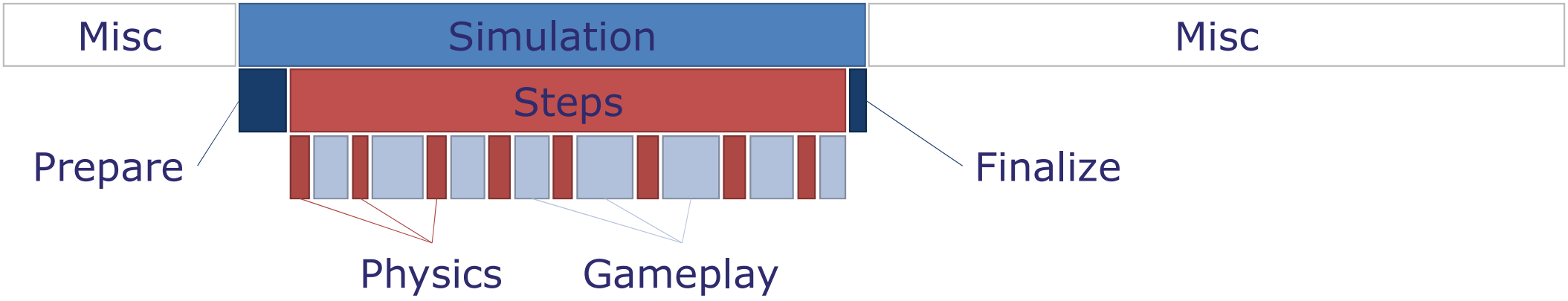
Join in Progress

- Naïve implementation: replay all inputs
  - 10 minutes = 18 000 steps!
- Our implementation : load snapshot
  - 10 minutes = average of 60 steps


Find
creative ways
to
optimize

- Run 8 times the gameplay in a single frame
- Content is always growing
- Communicate constraints to game design

Structure of a frame



- Multithread (deterministically!)
  - Double buffering
  - Messages
- Reorganize the rest of the frame to make room for gameplay
- Remove useless code & variables


FOR
HONOR
MARCHING FIRE

	Dominion	Breach
Players / Bots	8	8
NPCs	100 footmen	60 pikemen & archers 4 bots
Ingredients	20	60

Castle siege fantasy

Same CPU budget!

Initiative #1 Suspension

- Breach is a phase based gamemode
- Remove actors and ingredients from the simulation
  - No more in history buffer, no more updated
- Unsuspend at initial state

Initiative #2 Sleeping

- A lot of idle time
- Dynamic update rate
- CPU cost reduced by 90%

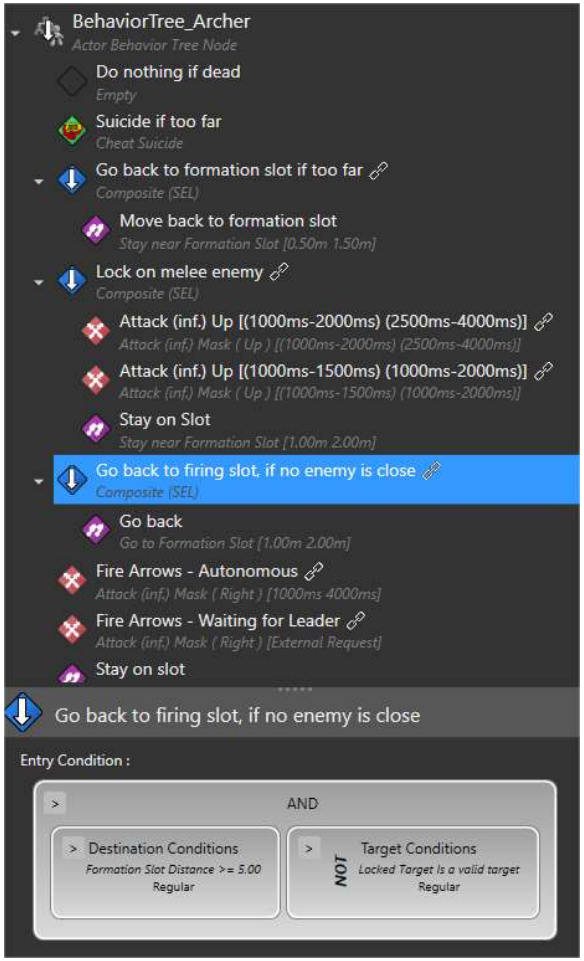
Initiative #3 Junior bots

- Bots with behavior updated only every 200ms
  - 6 times less costly than regular bots
  - Lower complexity

- Issues with ladders

Initiative #4 BehaviorFSM

- Lego-block data-driven system
- Created one class per type of soldier
  - Cost of the behavior divided by 20
  - 20% in the MT update of each soldier
- Low impact on designers velocity



Initiative #5 Physics

- Stepping was taking 0.5ms
- Guidance system
  - Plus a simple collision solver
- Saved 15% of the frame

Know where you stand

- Analytics tools and tests are your best friends
  - Benchmark: always 250ms resimulations
  - Reports
  - Automated tests

+ableau

Group
- Gameplay
- ManagerSimulationProxies
- Physics Character & Others
- Physics ContactManager & HandleMessa...
- Update Gameplay & GPU Frame

Tag	World										
	S01	S02	S03	S04	S05	S06	S11	S12	S21	S31	S41
AIManagerSimulationProxy	0.0328	0.0335	0.0318	0.0326	0.0328	0.0330	0.0330	0.0340	0.0341	0.0331	0.0330
AreaOfEffectManagerSimulationProxy	0.0232	0.0236	0.0232	0.0227	0.0231	0.0238	0.0229	0.0236	0.0239	0.0231	0.0230
BattlefieldManagerSimulationProxy	0.0698	0.0692	0.0667	0.0655	0.0665	0.0667	0.0667	0.0678	0.0702	0.0654	0.0661
ContactManagerSimulationProxy	0.0016	0.0017	0.0018	0.0017	0.0018	0.0019	0.0016	0.0019	0.0020	0.0019	0.0019
ConversationManagerSimulationProxy	0.0092	0.0093	0.0094	0.0090	0.0092	0.0095	0.0090	0.0096	0.0097	0.0094	0.0093
MapIngredientManagerSimulationProxy	0.0682	0.0860	0.0450	0.0715	0.0697	0.0493	0.0513	0.0471	0.0798	0.0415	0.0736
MatchManagerSimulationProxy	0.0374	0.0365	0.0370	0.0364	0.0369	0.0360	0.0346	0.0372	0.0369	0.0350	0.0359
NavigationManagerSimulationProxy	0.0116	0.0115	0.0116	0.0118	0.0123	0.0113	0.0109	0.0119	0.0120	0.0111	0.0117
ProjectileManagerSimulationProxy	0.0512	0.0516	0.0513	0.0501	0.0512	0.0526	0.0506	0.0518	0.0521	0.0506	0.0508
RespawnManagerSimulationProxy	0.0240	0.0243	0.0281	0.0237	0.0241	0.0259	0.0239	0.0247	0.0247	0.0261	0.0240

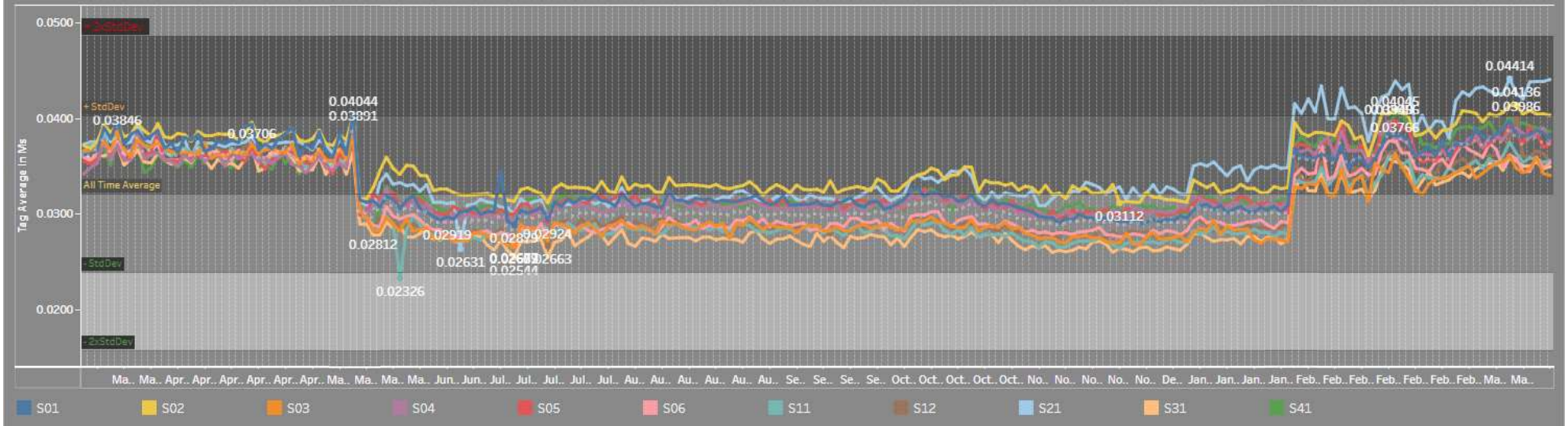
Version Date Range
3/6/2018 — 3/12/2019

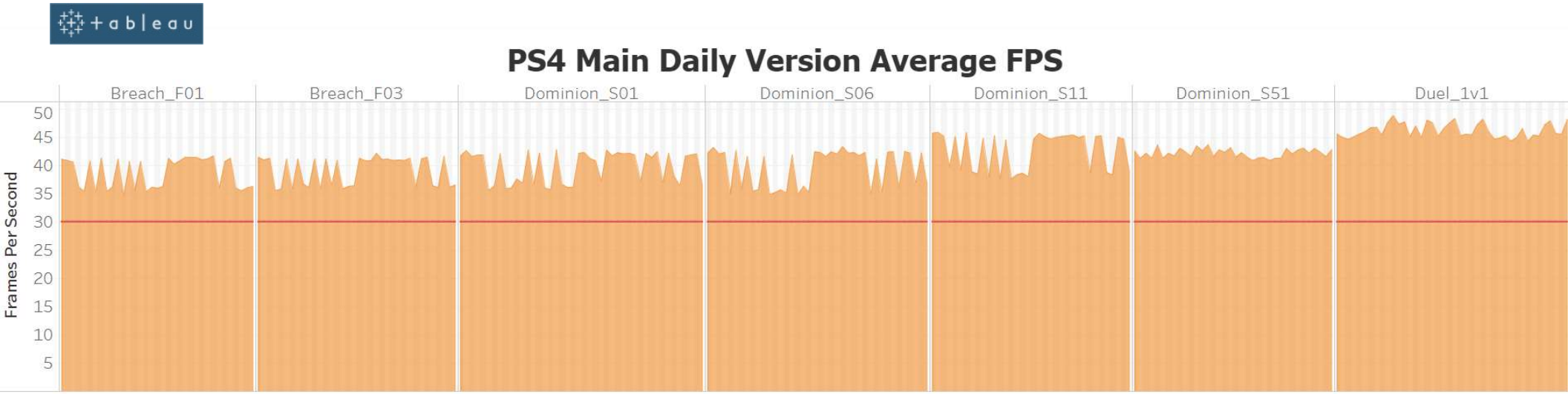
Version Date
(All)

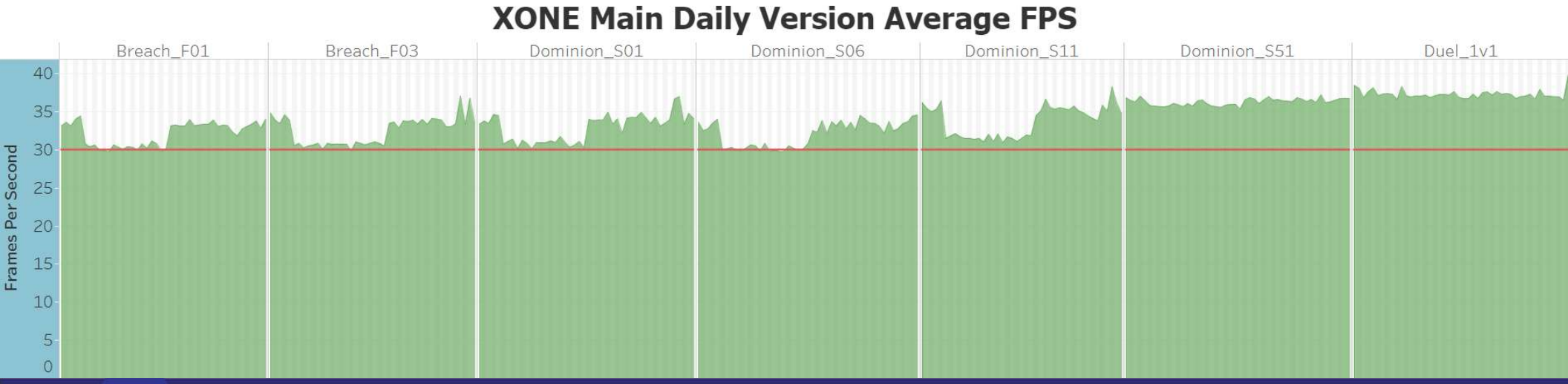
Parse Time
(Multiple values)

Version
(All)

Per World - Tag:AIManagerSimulationProxy, AreaOfEffectManagerSimulationProxy, BattlefieldManagerSimulationProxy and 7 more




PS4 Main Daily Version Average FPS
Breach_F01
Breach_F03
Dominion_S01
Dominion_S06
Dominion_S11
Dominion_S51
Duel_1v1
Frames Per Second
50
45
40
35
30
25
20
15
10
5


XONE Main Daily Version Average FPS
Breach_F01
Breach_F03
Dominion_S01
Dominion_S06
Dominion_S11
Dominion_S51
Duel_1v1
Frames Per Second
40
35
30
25
20
15
10
5
0


Prevent people
from
repeating
mistakes
TODAY
YOUTH JAILED

Usual suspects - 80% of desynchs come from the same 3 sources

- Cosmetic
- Multithread access violation
- Random code mistakes

Usual suspect #1 Cosmetic

- SessionTime, visual
- Pathfinding, raycast, random, skeleton, camera

- Validation macros in unsafe functions

```
#define ValidateCosmeticScope() popIgnorableAssert(!GameplaySimulatorProxy::GetInstance()->IsSimulationThread()
                                                   || (GameplaySessionManager::GetSimulatedTimeMS() == InvalidSessionTimeMS),
                                                   "Cosmetic Scope Validation Failed - Simulation is updating!")

ubiU32 Actor::GetTimeSinceDeathMS() const
{
    ValidateCosmeticScope();
    if (m_SimulationProxy)
        return m_SimulationProxy->GetTimeSinceDeathMS(TimeManager::GetSessionTimeMS());
    return 0;
};
```

Usual suspect #1 Cosmetic

- Desynchs created by wrong data setup

Usual suspect #2 Multithread access violation

- Validation macros in every state accessor

```
#define ObjectProxy_ValidateAccess() popIgnorableAssert(!GetSimulator()->IsInMultithreadedUpdate()
                                                        || GetSimulator()->IsProxyExecutingOnCurrentThread(this),
                                                        "Simulation Proxy: Multithread Access violation!")


inline const ActorSimulationState& GetCurrentSimulationState() const {return m_CurrentSimulationState;}


inline ActorSimulationState& GetNextSimulationState() {ObjectProxy_ValidateAccess();
                                                       return m_NextSimulationState;}
inline const ActorSimulationState& GetNextSimulationState() const {ObjectProxy_ValidateAccess();
                                                                   return m_NextSimulationState;}
```

Usual suspect #3 Random code mistakes

- Uninitialized variables
- Static variables
- Pointers
- Missing break;

¯_(ツ)_/¯

Usual suspect #4 Validation code

- Written by hand at the beginning
- Code generation for gameplay states & messages

```
simstruct ActorDeathState
{
    ubiBool Dead = false;
    ubiU32 RevivingTimeLeftMS = 0;
    simstruct HitInfo[2] LastHitInfos;
}

simstruct HitInfo
{
    ubiFloat Damage = 0.0f;
}
```

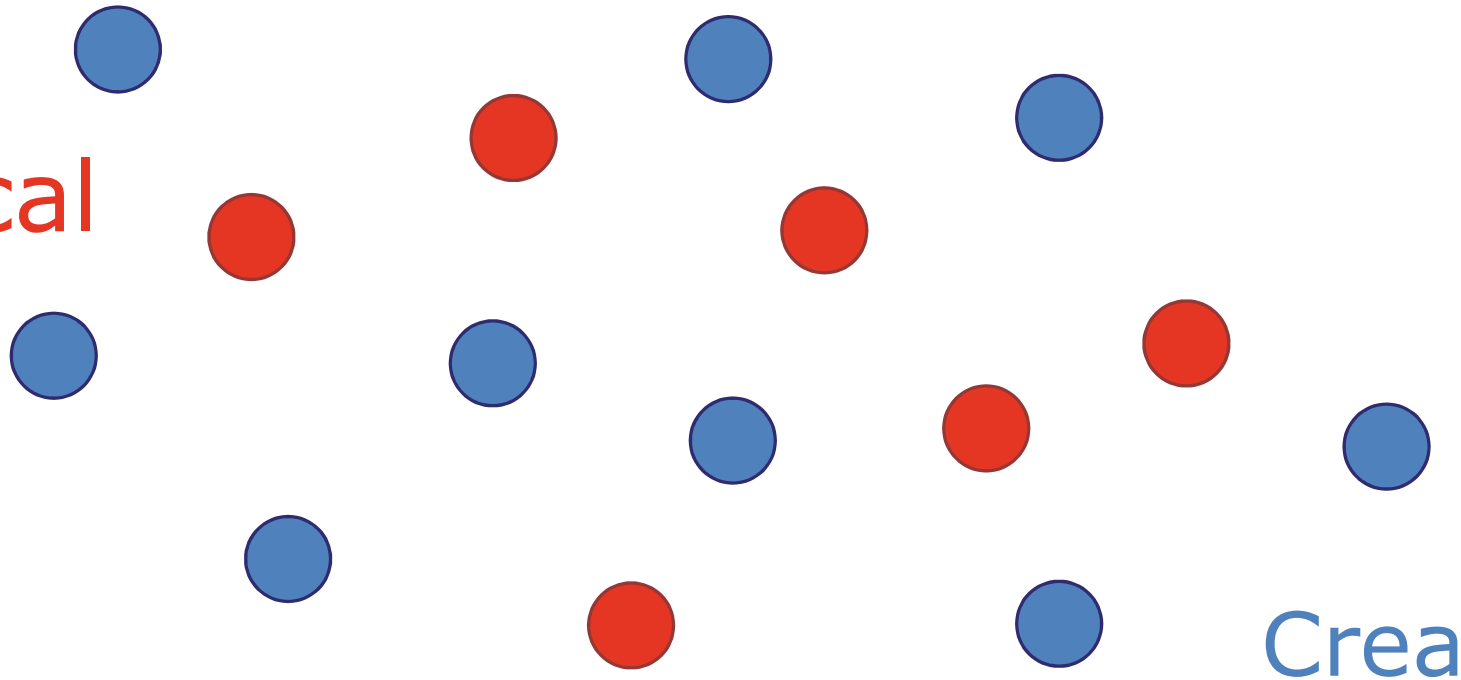
```
#if defined(POP_USE_GAMEPLAY_SIMULATION_DIFFERENCE_REPORT)
void ActorDeathState::ReportDifferences(GameplaySimulationDifferenceReport& _report, const void* _structA, const void* _structB)
{
    const ActorDeathState* simStructA = (const ActorDeathState*)_structA;
    const ActorDeathState* simStructB = (const ActorDeathState*)_structB;

    for (ubiU32 i = 0; i < 2; ++i)
    {
        if (simStructA->m_LastHitInfos[i] != simStructB->m_LastHitInfos[i])
        {
            _report.BeginSubState(StringFormat("LastHitInfos[%u]", i));
            HitInfo::ReportDifferences(_report, &simStructA->m_LastHitInfos[i], &simStructB->m_LastHitInfos[i]);
            _report.EndSubState();
        }
    }
    if (simStructA->m_RevivingTimeLeftMS != simStructB->m_RevivingTimeLeftMS)
    {
        _report.ReportDifference("RevivingTimeLeftMS", simStructA->m_RevivingTimeLeftMS, simStructB->m_RevivingTimeLeftMS);
    }
    if (simStructA->m_Dead != simStructB->m_Dead)
    {
        _report.ReportDifference("Dead", simStructA->m_Dead, simStructB->m_Dead);
    }
}
#endif
```

- Define clear rules…
  - …and remove all exceptions
- Enforce those rules in your code architecture
- The more you wait, the harder it becomes


Nourish
your
team
culture

Technical



Creative

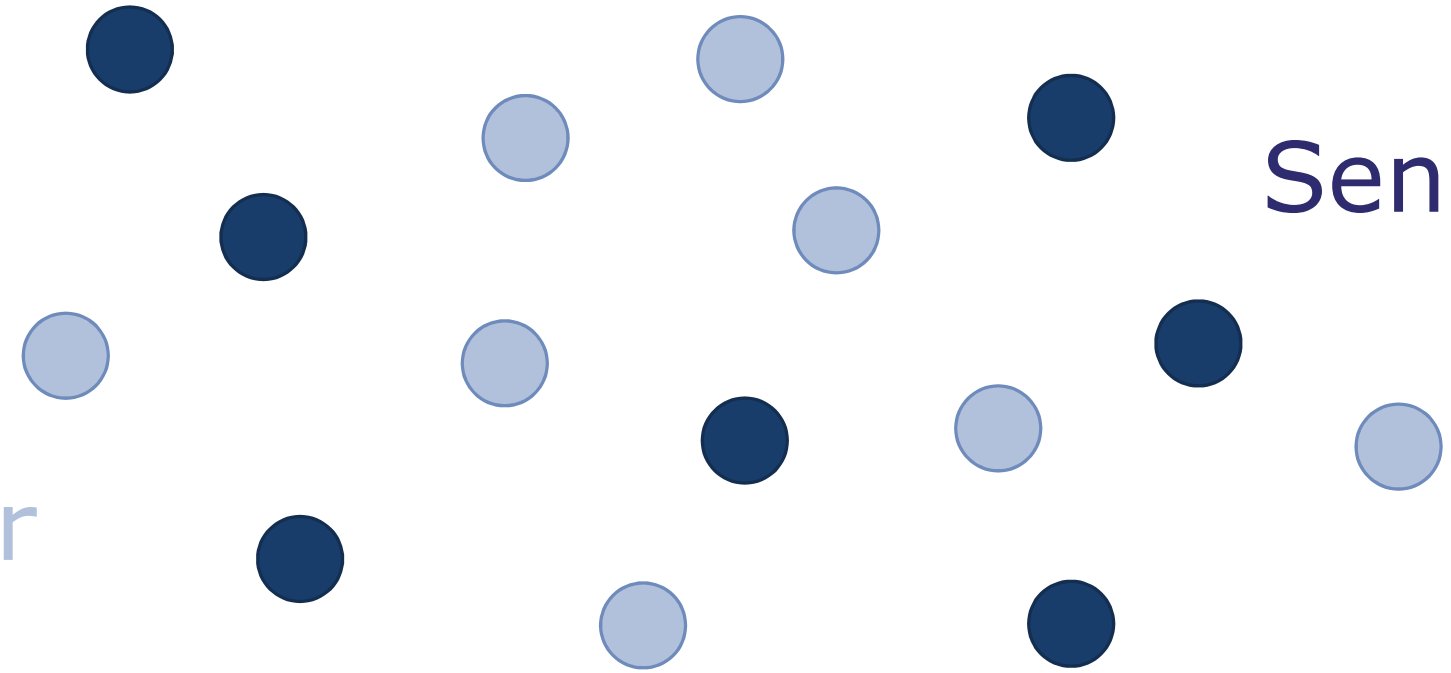
Determinism is a good fit with this team

- Fragile - it takes discipline
- Less "networky"

"Multiplayer first"

- Design-wise 'For Honor' is a multiplayer game
  - Very few purely solo content

- Code-wise Each feature has to work in multiplayer
  - Lego-blocks based gameplay
  - Sometimes easy to forget
  - No shortcuts


Senior
Junior

- Simulation 101
- Write documentation
- Create redundancy
- Spread new habits

- It's all about intuition

```
MyType* myPointer = NULL;
myPointer->DoSomething();
```

- Involve QC
  - Share a common vocabulary
  - How to treat a desynch
  - Multiplayer first
  - Edge cases


Let
it
go

- You could work on your pipeline and tools forever
  - Choose your battles
- Keep your tech aligned with what your game needs

- Some people will care more than others
- People WILL do mistakes
- You will never reach 0% desynchs
- Let gameplay programmers focus on gameplay

Thank you!

jennifer.henry@ubisoft.com
www.linkedin.com/in/henry-jennifer

Image sources:
Back to the Future. Dir. Robert Zemeckis. Perf. Michael J. Fox, Christopher Lloyd. Universal Pictures, USA, 1985.
Back to the Future II. Dir. Robert Zemeckis. Perf. Michael J. Fox, Christopher Lloyd. Universal Pictures, USA, 1989.
Back to the Future III. Dir. Robert Zemeckis. Perf. Michael J. Fox, Christopher Lloyd. Universal Pictures, USA, 1990.


ARE YOU READY TO
CREATE THE UNKNOWN?
jobs.ubisoft.com
LEARN MORE AT OUR BOOTH!
WEST HALL - FLOOR 2
UBISOFT