LAG ERASURE FOR ZERO-LAG DISTRIBUTED SIMULATION

DISSERTATION DEFENSE 17 FEB 2026

LOREN PEITSO

Abstract

The need for simulations larger than a single host or even cluster can manage is becoming common, and simulations which require specialized hardware from geographically distributed sites are everyday occurrences. Communications latency generates lag within distributed simulations, creating inaccuracies and event causality violations that at best may put simulation progression and final results into doubt, and always prevents repeatable, verifiable results. This problem is catastrophically bad for building trust in Artificial Intelligence-enabled systems via simulation, the costs and risks to replace simulation with live training may be unpalatable. Decades old ubiquitous engineering assumptions about latency in networked simulation communications are treated as inescapable in seminal papers and textbooks, preventing any progress on eliminating the lag problem. This work has discovered that those engineering assumptions are not absolute, they are implementation dependent. This enables a new hybrid simulation architecture that is not subject to inescapable negative lag effects. The Lag Erasure simulation architecture enables lag-free distributed simulations which are uniquely repeatable and verifiable, this enables a level of confidence and trust in simulation results impossible to date. The scope of this advance is widespread, lag effects crosscut all distributed shared worlds including: Command and Control (C2) systems, Live Virtual Constructive (LVC) training, multi-platform weapons systems, multi-user virtual/augmented (VR/AR) useful for both training and field operations, the online gaming community, especially the massively multiplayer online games and battle arenas, and development of shared massively multi-user AR & VR applications that constitute the long-anticipated metaverse. Lag is their enemy no longer.

Defense.key - February 13, 2026

Abstract - Highlights

- **Communications latency generates lag within distributed simulations, creates inaccuracies and causality violations**
  - Always prevents repeatable, verifiable results
  - This is unworkable for training simulations and building trust in Artificial Intelligence-enabled systems
- Discover that existing engineering assumptions locking-in latency and lag are implementation dependent
- **New assumptions, new hybrid simulation architecture that is not subject to negative lag effects.**
  - Repeatable and verifiable simulations. Scope affected is widespread
    - Command and Control systems, Live Virtual and Constructive training, multi-platform weapons systems, Multi-user virtual/augmented (VR/AR) for training and field operations
    - Online gaming, especially the massively multiplayer online games (MMOG) and battle arenas (MOBA)
    - Shared massively multi-user AR & VR applications that constitute the long-anticipated metaverse

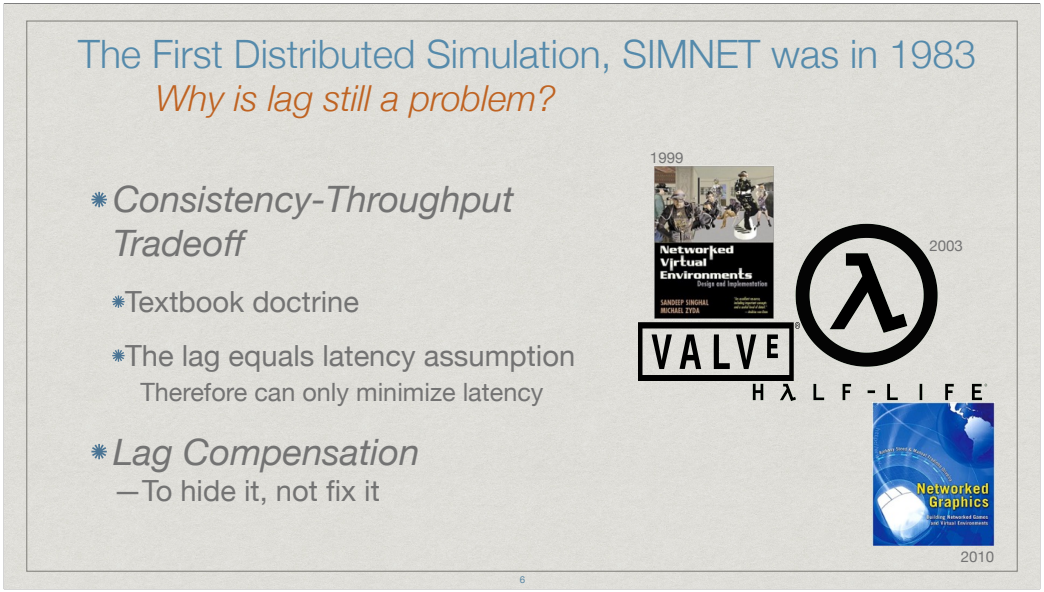


Meme screenie credit to internet, Skyrim and Toy Story sources

Lag as a Misalignment of Timing Expectations

Breaking Cause and Effect

- Player-**A** initiates an action sequence **`moveHandTo(target)`** and sends an action message to Player-**B**.
  - This may occur before or after Player-**B** initiates the **`releaseBalloon()`** action, either way neither player's message is received prior to taking their initial action
  - On Player-**A**'s display their hand moves and **the balloon in Player-B's hand pops** after a split-second
- Player-**B** initiates an action sequence **`releaseBalloon()`** and sends an action message to Player-**A**
  - This may occur before or after Player-**A** initiates the **`moveHandTo(target)`** action, either way neither player's message is received prior to taking their initial action
  - On Player-**B**'s display they see **the balloon fly away**, Player-**A** misses low by a split-second



S&Z: It is impossible to allow dynamic shared state to change frequently and guarantee that all hosts simultaneously access identical versions of that state.

S&O: This is a fundamental property of NVEs and other realtime systems if they run in what is called an asynchronous mode, meaning that they evolve their state without waiting for the network.

SIMNET 1983

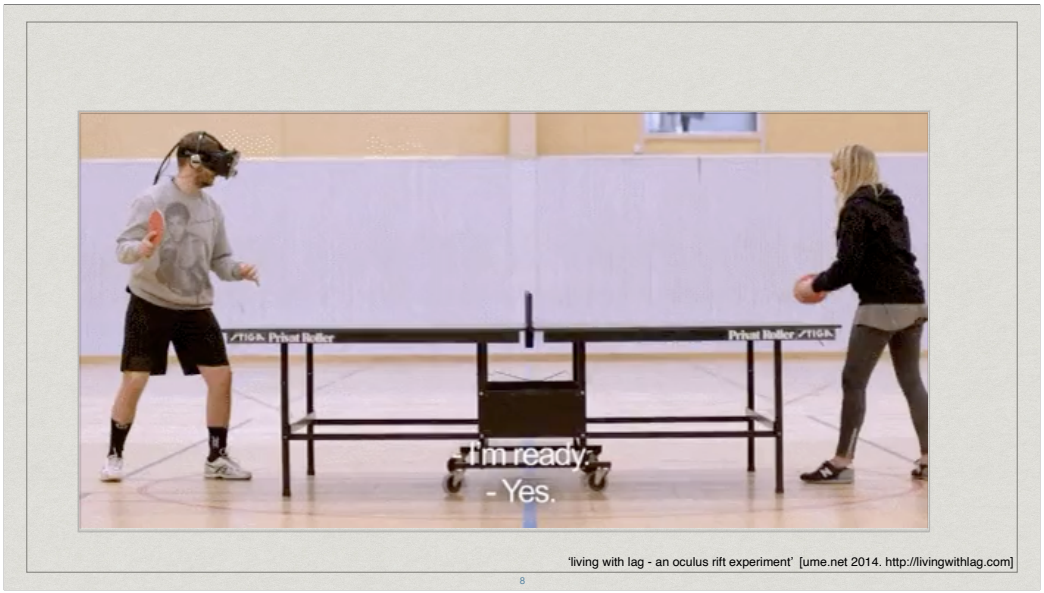
How to Train Your AI-Tactical Assistant?

- The only sane answer is simulation
- First, purely virtual
- Graduate to LVC simulations
  - Participants distributed all over the place
  - We already “bet the farm” on LVC for large-scale training
    - So use it, ***but fix it!***
- Nobody sane can accept training an AI in an environment as broken as the next slide…



7

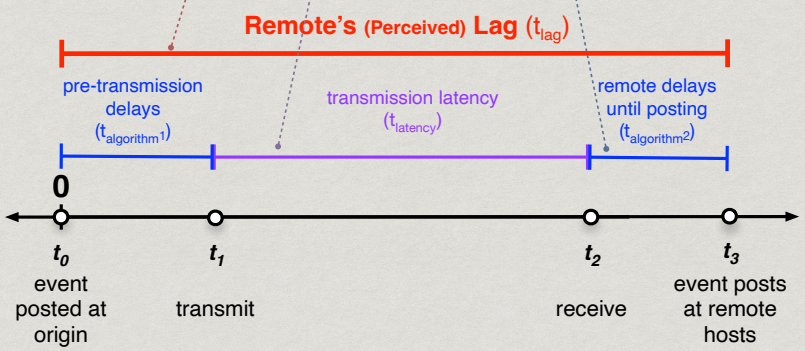
Live Virtual Constructive



⅓ of a sec lag injected via an external server, ume.net is a cable television delivery provider using the marketing claim they could beat this lag with their low lag networks

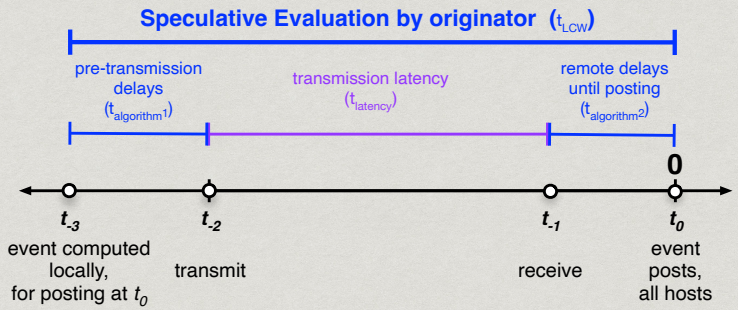
Kobayashi Maru - The No-Win Version

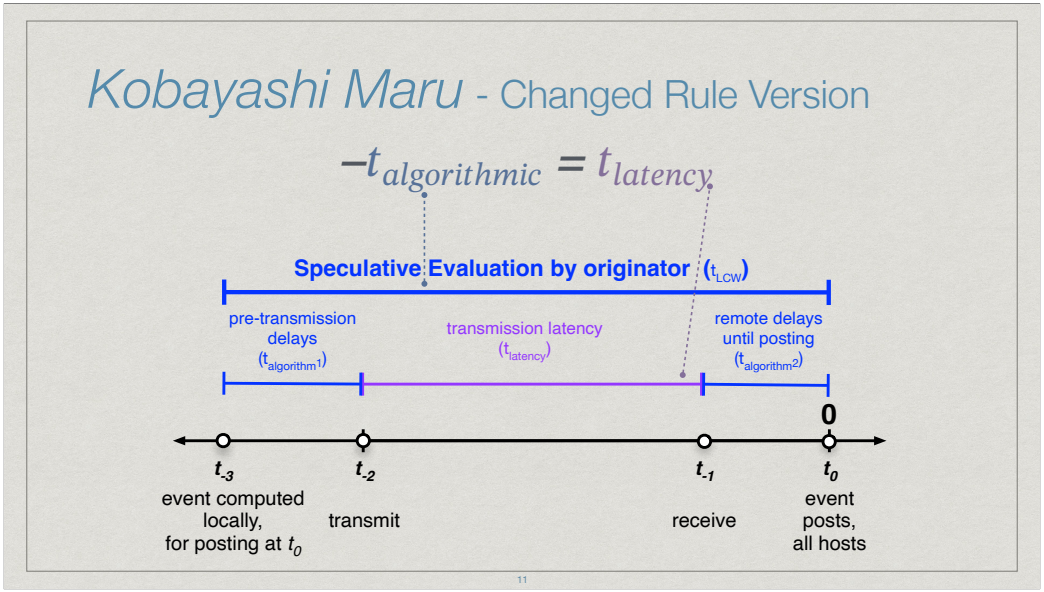
$$t_{lag} = t_{latency} + t_{algorithmic}$$



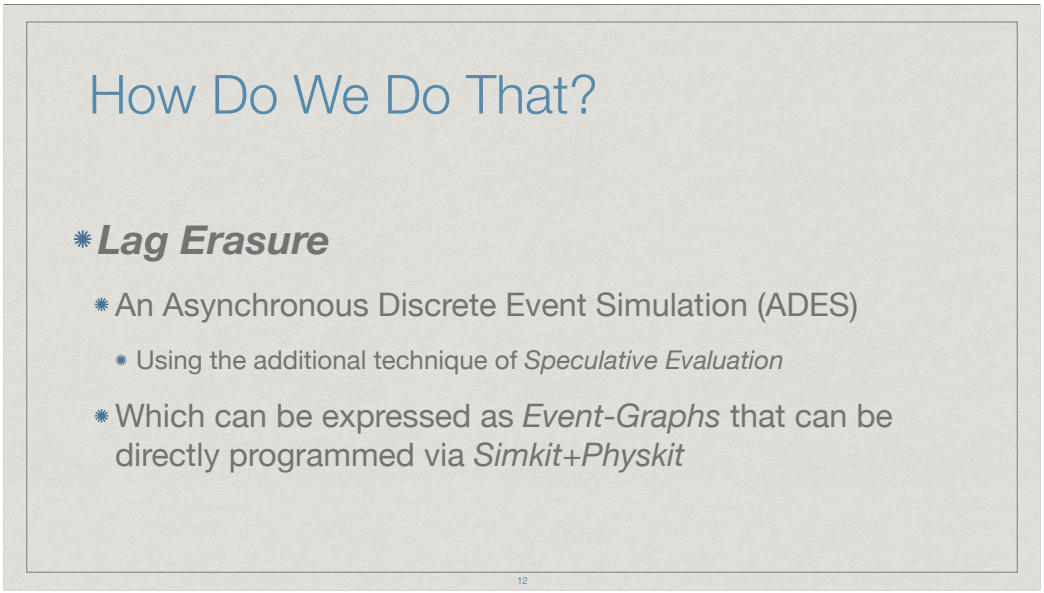
Kobayashi Maru - Changed Rule Version

$$0 = t_{latency} + t_{algorithmic}$$





the idea is simple, how to get there from the starting point of time-step simulations is not obvious

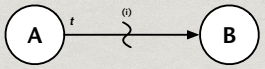


Speculative Evaluation has been around since 1979 at least

Physkit inherits from SimEntityBase and BasicEventList

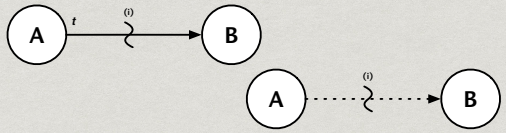
Event-Graphs

- Fundamental
- Cancelling
- Iteration
- Turing-Complete system



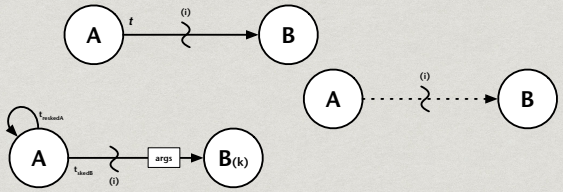
Event-Graphs

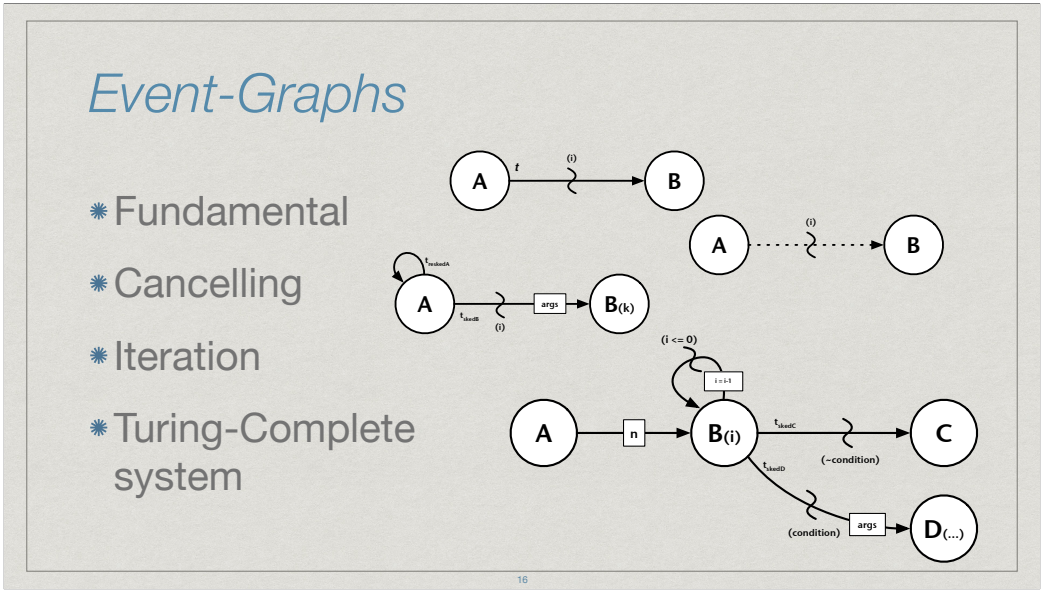
- Fundamental
- Cancelling
- Iteration
- Turing-Complete system



Event-Graphs

- Fundamental
- Cancelling
- Iteration
- Turing-Complete system



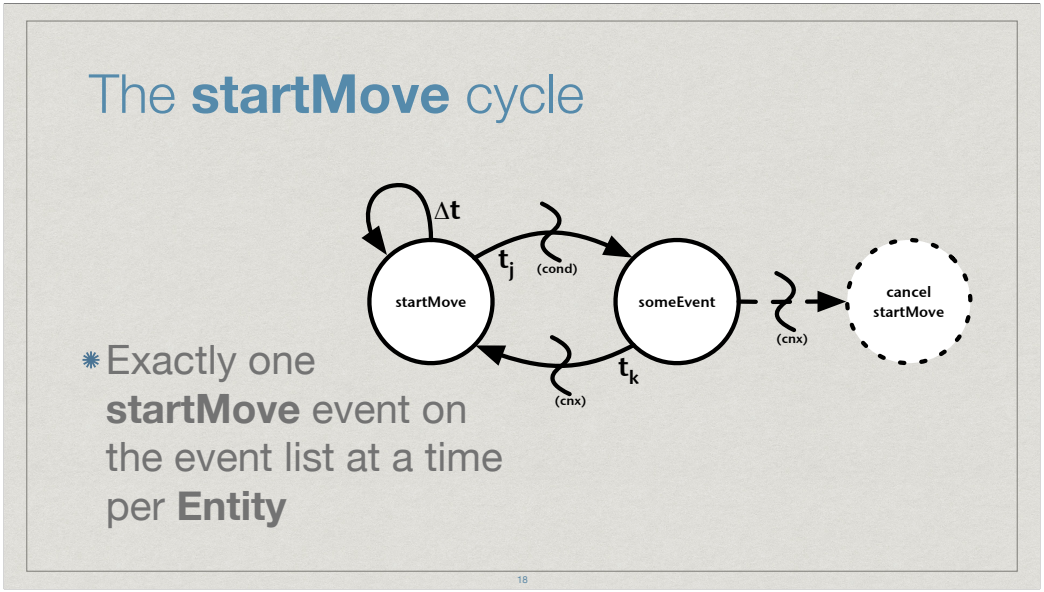


Turing-Complete for operations equality for a Turing Machine, time remains outside the concept

Event-Graph Patterns for Physics Sims

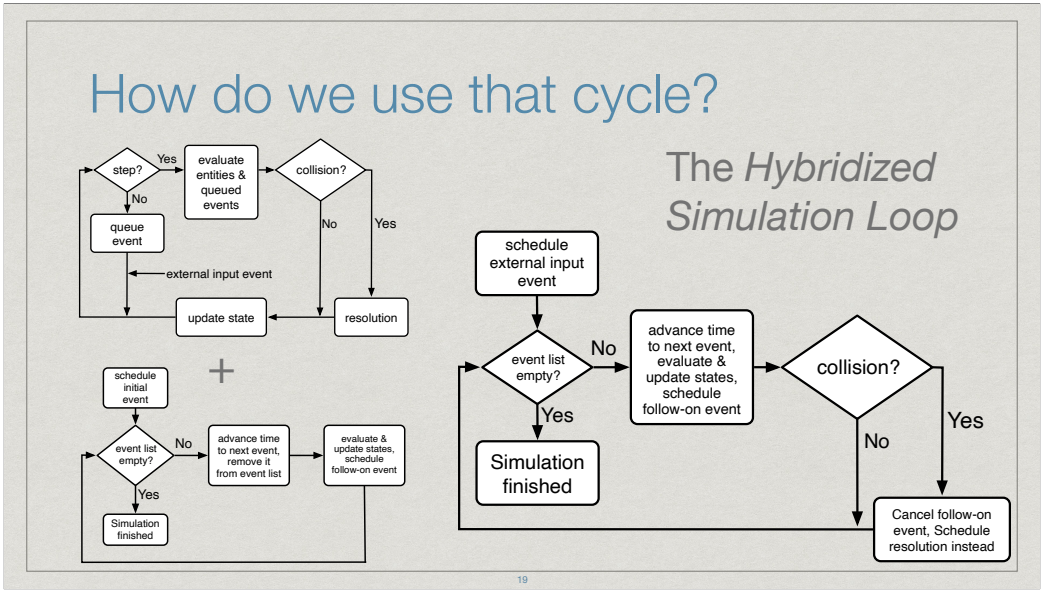
- Events put into *Simkit* event list
- *Physkit* extends two *Simkit* classes, adding physics and collision detection support
- Next: the basic pattern and how to detail it out

Physkit inherits from SimEntityBase and BasicEventList



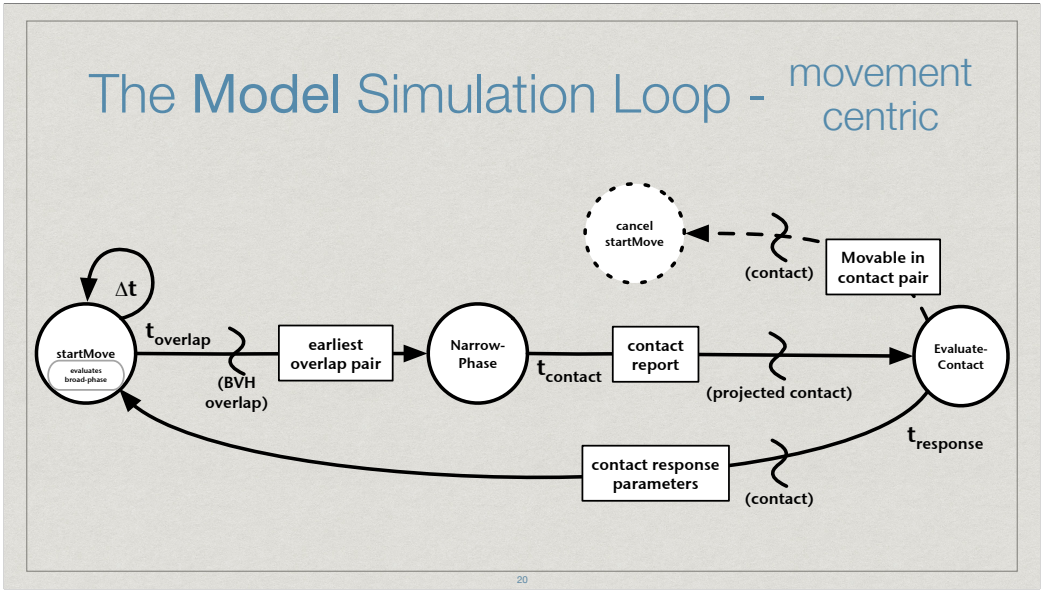
on evaluation an event is commit to simulation state, and it has exactly the one startMove following it on the EventList

very useful for avoiding event list searches when unnecessary

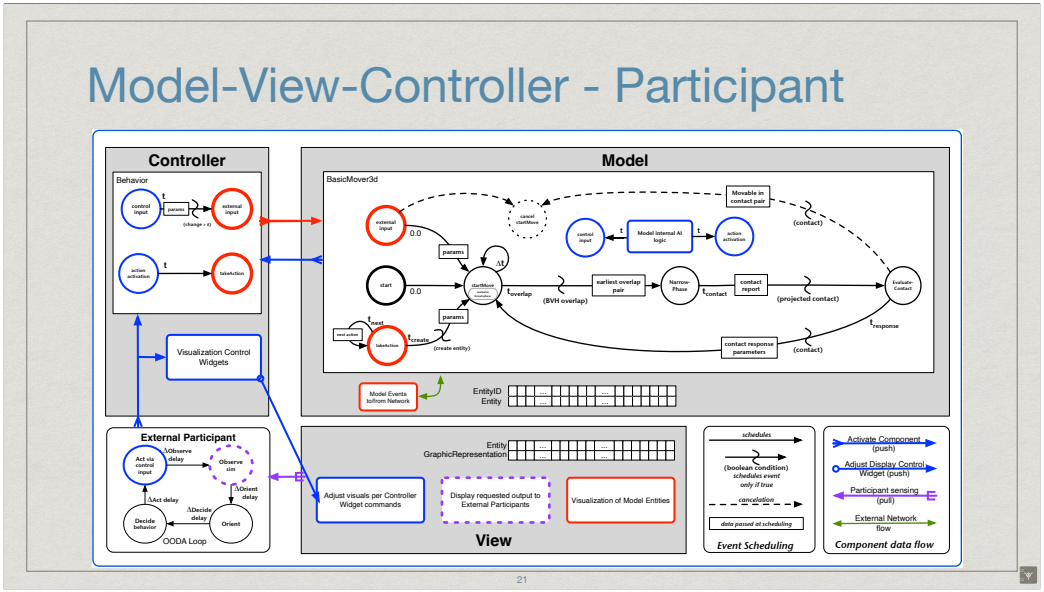


fixed time step loop +

Next-event loop == Hybrid loop


The Model Simulation Loop - movement centric
cancel startMove
(contact)
Movable in contact pair
Δt
startMove
evaluates broad-phase
t overlap
(BVH overlap)
earliest overlap pair
Narrow-Phase
t contact
contact report
(projected contact)
Evaluate-Contact
t response
contact response parameters
(contact)
20

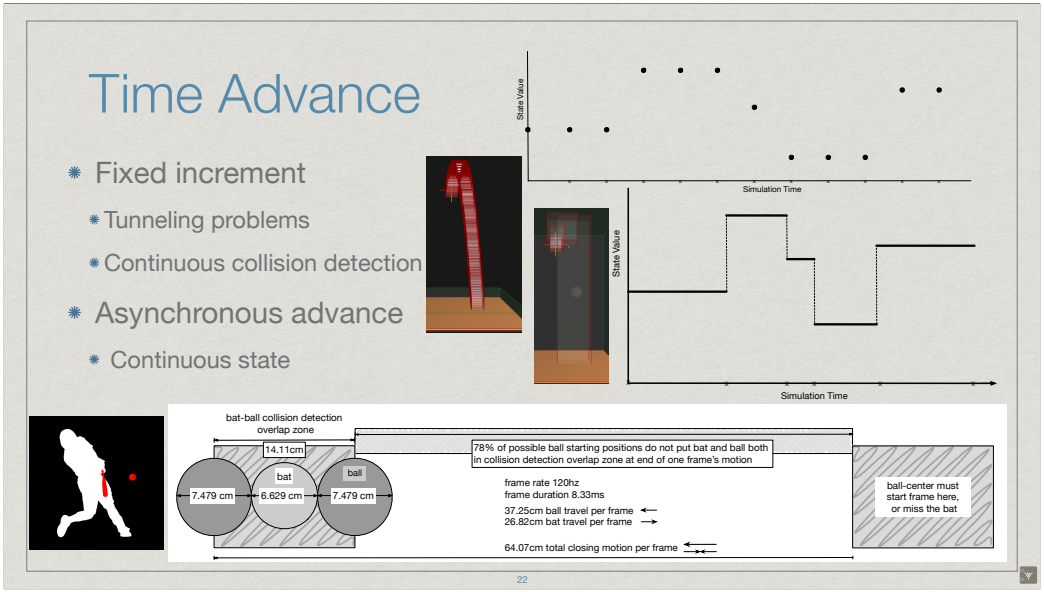
Per Entity



Based on the original MVC design pattern

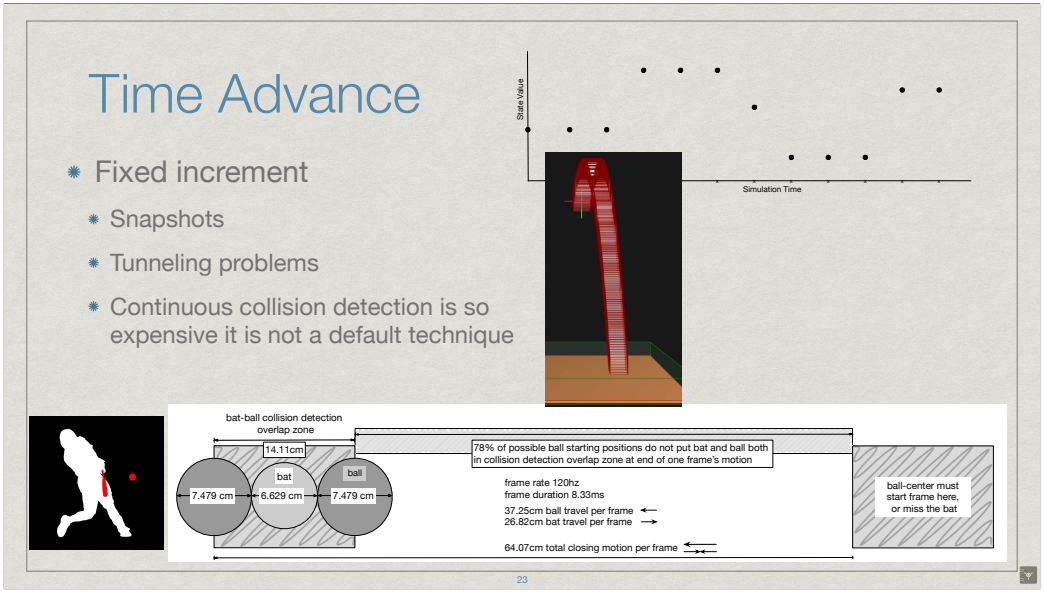
external participant **Action**: example - perform action that produces a detectable signal (not movement)

coherence of distributed state

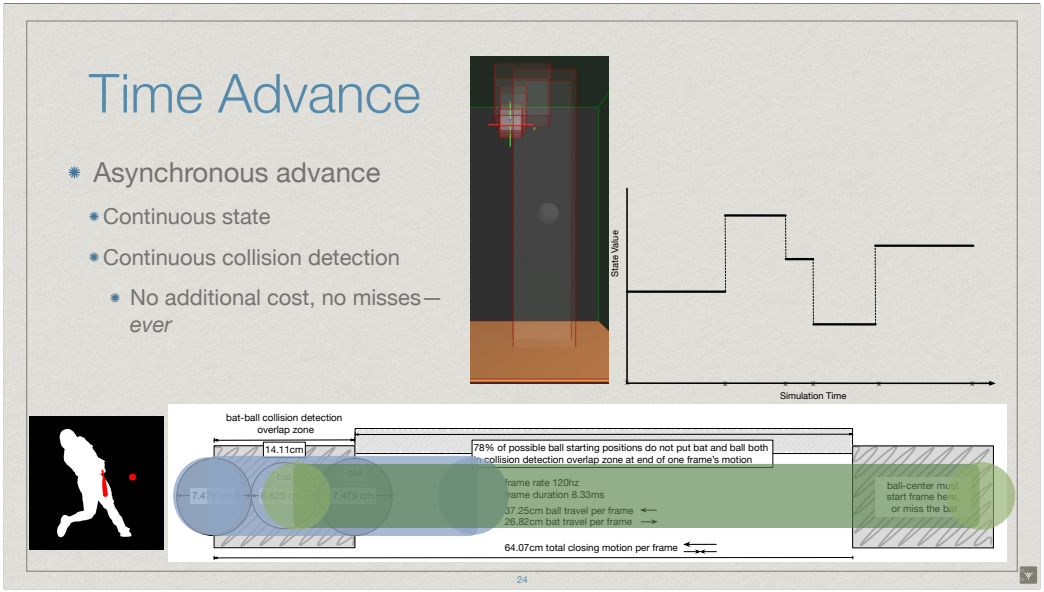


both state diagrams approximate the same changes

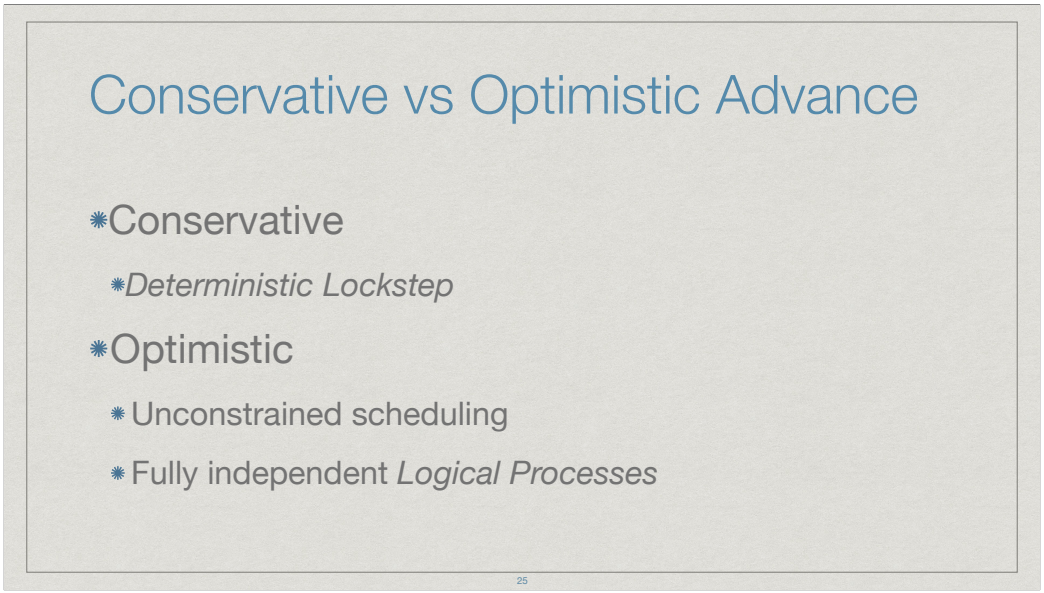
fixed step snapshots intentionally do not line up exactly



bat and ball tunnel through each other



asynchronous advance is **necessary** to get longer advances than frame update-rate fixed steps



Approaches/Methodologies not flavors

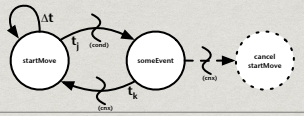
optimistic is subject to getting too far ahead, requiring rollback to fix

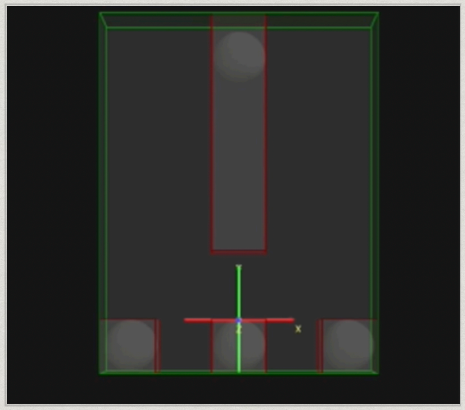
The *Logical Process* (*LP*)

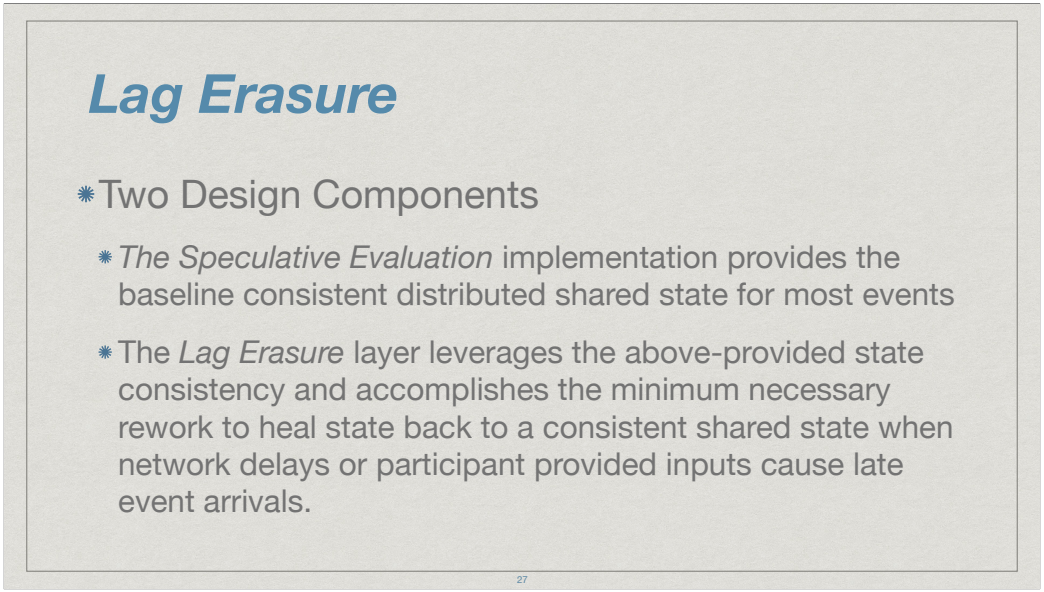
- Represents a Physical Process
- *LP* and virtual time
- Threading and full independence

  not really…

- The **startMove** cycle is the *LP* in *Lag Erasure*







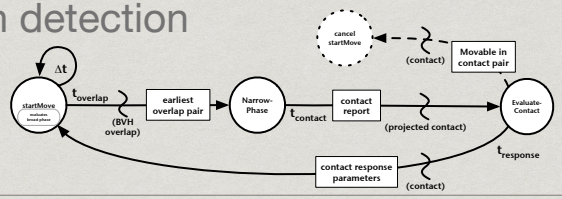
erasing in a bounded time window — computationally tractable for all distributed participants

The *Speculative Evaluation* ADES

- Typical **`eventList`**
- **`startMove`** event cycle
- Complex behavior support for collision detection

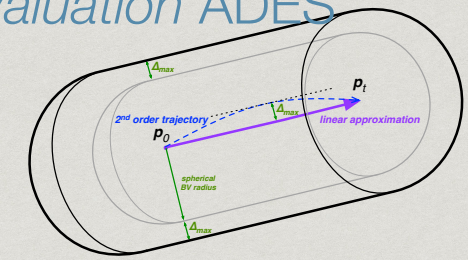
```
** Event List 1 --  **
0.010          StartMove

1.798e+308     StartMove
 ** End of Event List -
```

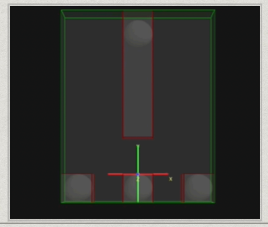


The *Speculative Evaluation* ADES

- **`MotionWitness`**
  - Continuous movement representation

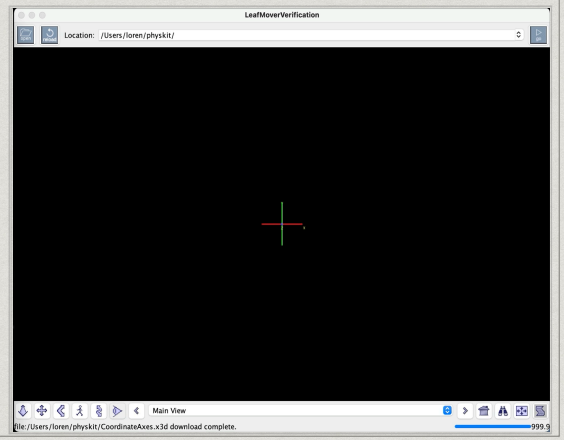


- **`R*-Tree`**
  - Culling for collision detection broad phase



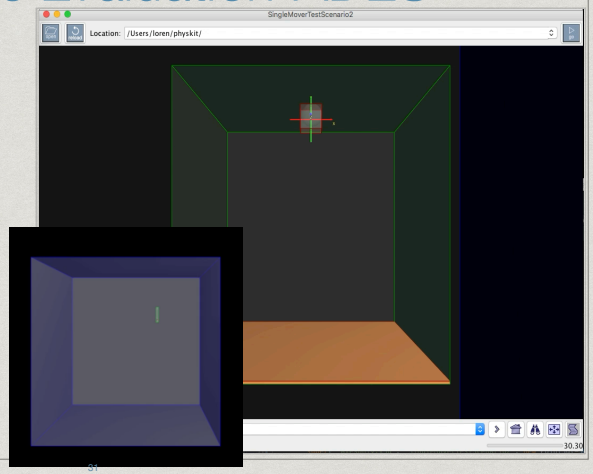
The *Speculative Evaluation* ADES

- **R*-Tree3D**
  - Basic collision detection in action
  - Adaptive step growth at beginning
  - Max step stable at h where sign of slope changes



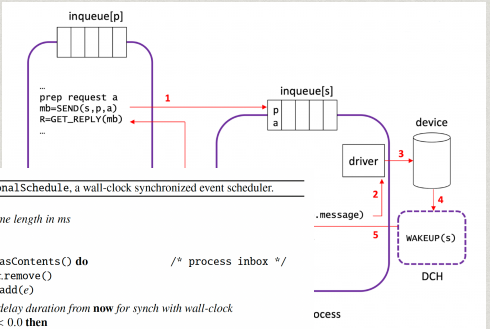
The *Speculative Evaluation* ADES

- **`R*-Tree`** in Operation
  - Intermediate Node leaves-only (green)
  - Red boxes are the leaves wrapping a **`MotionWitness`**
    - they are artificially retained for this demo video
  - Root/Directory Node (blue)



The *Speculative Evaluation* ADES

- Multi-thread simulation
  - **Region**s as protected service process
  - **RegionalSchedule**

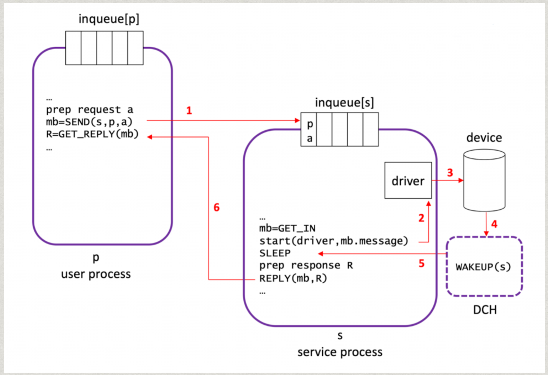


Algorithm 1: The `RegionalSchedule`, a wall-clock synchronized event scheduler.

```
1  begin
2      oneFrame_ms ← frame length in ms
3      while running do
4          do
5              while inbox.hasContents() do                    /* process inbox */
6                  e ← inbox.remove()
7                  EventList.add(e)
8              maxSleep ← delay duration from now for synch with wall-clock
9              if maxSleep < 0.0 then
10                 rework()                                    /* see Algorithm 2 */
11             if maxSleep > oneFrame_ms then                  /* wall-clock sync */
12                 SLEEP(maxSleep)              /* also wakes on inbox.add() */
13         while maxSleep > oneFrame_ms
14         CurrentEvent ← EventList.pop()
15         CurrentEvent.evaluate()
```

The *Speculative Evaluation* ADES

- Multi-thread simulation
  - **Region**s as protected service process
  - RegionalSchedule



peter's slide

The *Speculative Evaluation* ADES

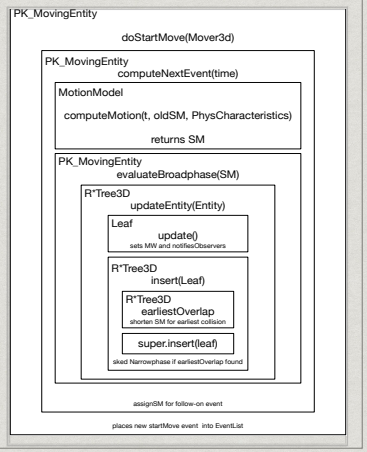
- Multi-thread simulation
  - **Regions** as protected service process
  - **RegionalSchedule**

Algorithm 1: The `RegionalSchedule`, a wall-clock synchronized event scheduler.

```
1  begin
2      oneFrame_ms ← frame length in ms
3      while running do
4          do
5              while inbox.hasContents() do                    /* process inbox */
6                  e ← inbox.remove()
7                  EventList.add(e)
8              maxSleep ← delay duration from now for synch with wall-clock
9              if maxSleep < 0.0 then
10                 rework()                                    /* see Algorithm 2 */
11             if maxSleep > oneFrame_ms then                  /* wall-clock sync */
12                 SLEEP(maxSleep)              /* also wakes on inbox.add() */
13         while maxSleep > oneFrame_ms
14         CurrentEvent ← EventList.pop()
15         CurrentEvent.evaluate()
```

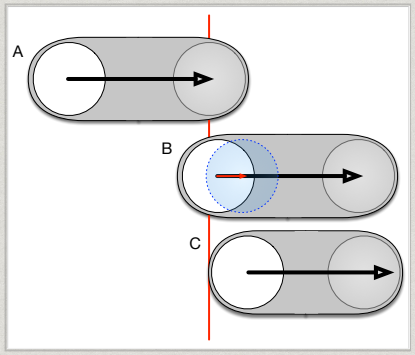
The *Speculative Evaluation* ADES

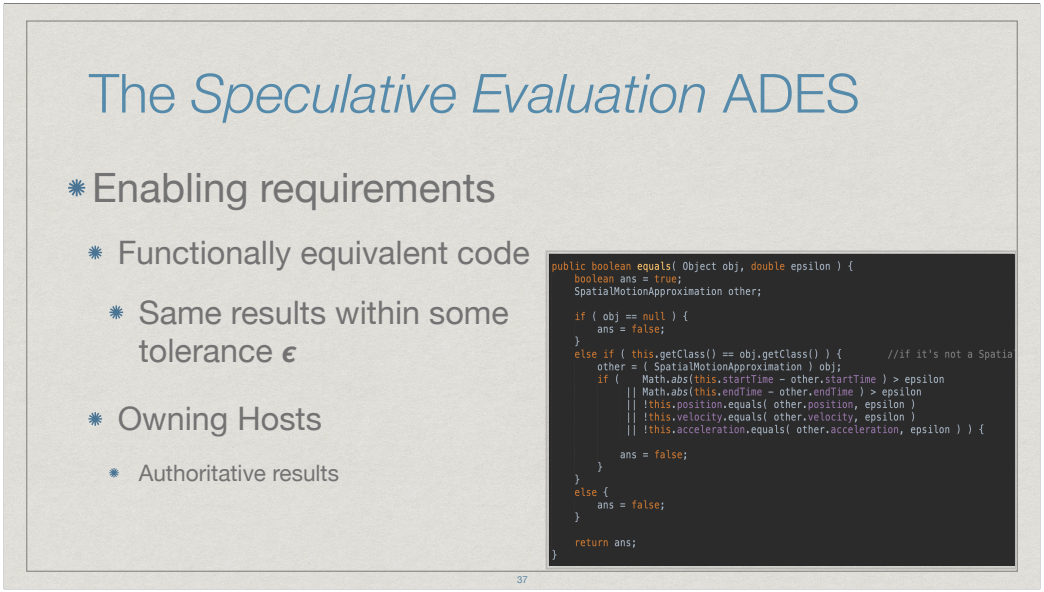
- Multi-thread simulation
  - Interlude for a live 4 thread simulation scenario



The *Speculative Evaluation* ADES

- Moving between **`Region`**s
  - Entities only in one event list—ever
- **`Liminal`**s
  - Passed via **`Region.inbox`**
  - Not put into incoming event list until situation-**`C`**

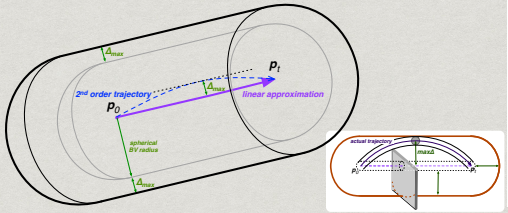




equivalent results across physics-related computations
implementation / language / platform independent

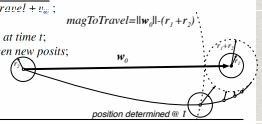
The *Speculative Evaluation* ADES

- Broad phase (culling)
  - Exit Collision Detection ASAP
  - Capsule Geometry



Algorithm 4: Time of Bounding Sphere to Bounding Sphere Contact

```
1  Function timeAtContact( p1, p2, v1, v2, a1, a2, r1, r2 )
2  w0 ← p1 − p2, v ← v2 − v1, a ← a2 − a1, vw ← v · w0, aw ← a · w0;
3  if aw and vw <= 0 then                                   /* opening motion */
4      return −∞;
5  else if ||a|| = 0 then                                   /* non-accelerated closure */
6      if ||v|| = 0 then                                    /* parallel motion */
7          return −∞;
8      else                                                 /* velocity-only closure */
9          t ← compute standard linear time-of-CPA;
10         compute new posits for both objects at time t;
11         distance ← compute distance between new posits;
12         if distance > (r1 + r2) then
13             return −∞;
14         else if distance < (r1 + r2) then
15             return t − (√((r1 + r2)² − distance²)/||v||);
16         else
17             return t;
18 else                                                     /* accelerated closure */
19     magToTravel ← ||w0|| − (r1 + r2);
20     velocityMag ← √(2 · ||a|| · magToTravel + vw²) − vw;
21     t ← (velocityMag − vw)/||a||;
22     compute new posits for both objects at time t;
23     distance ← compute distance between new posits;
24     if distance > (r1 + r2) then
25         return −∞;
26     else
27         return t;
```



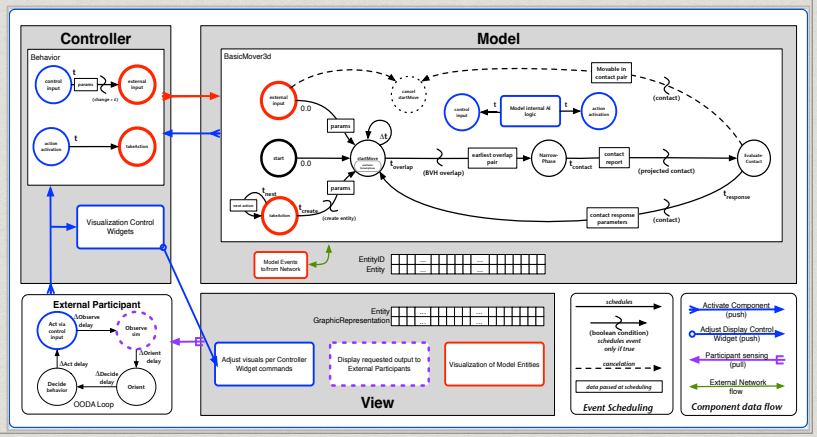
Speculative Evaluation and Latency Hiding Limits

- The ***Lag Erasure*** architecture targets simulation models and events at **human scales** intended to support **real-time** human/equipment/world-in-the-loop applications
  - Not designed for high-frequency updates
- Not a substitute for a bad network



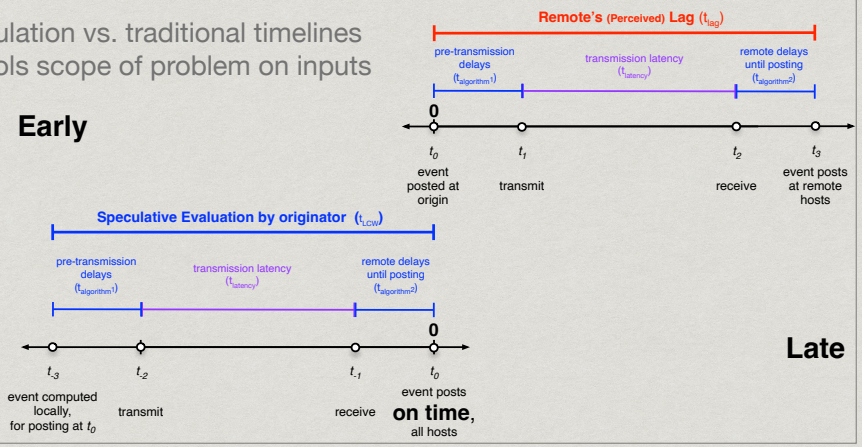
The ***Lag Erasure*** Layer

- Necessary due to external inputs & participants or excess network jitter



The ***Lag Erasure*** Layer

- Speculation vs. traditional timelines controls scope of problem on inputs



The ***Lag Erasure*** Layer - Rollback vs. Rework

- Rollback
  - Unconstrained
  - Network communications
  - Not compatible with real-time simulations
- Rework
  - Highly constrained scope due to the *Latency Compensation Window* (LCW)
  - Local resolution
  - **This. Erases. Lag.**

Algorithm 2: The rework algorithm.

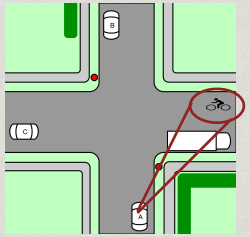
```
1  begin
2      if authoritativeEvent matches local version then
3          assign authoritative binary starting values to local event
4      else
5          reworkTime ← authoratativeEvent.startTime
6          reworkList ← shallowCopyOf(archivedEvents >= reworkTime)
7          for eachEvent in reworkList do
8              entity.reworking ← True
9              entity.evaluateEvent() {
10                 if required then cancel existing follow-on event(s)
11                 if required then schedule new follow-on event
12             }
13             entity.reworking ← False
```

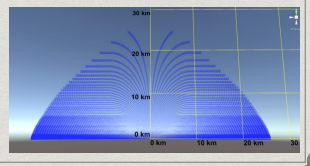
Use Case Possibilities - Shared Worlds

- *Perfect Playback*
- Digital Twin reality
- AR in the cockpit (sensor dynamics)
- AR for collaboration
- Everyday AR for traffic safety

None of the above are workable

- Multiplayer gaming (MMOG/MOBA)
  bad latency effects, professional competitions over in-person LAN only





massively multiplayer online game
multiplayer online battle arena

Contributions

Big Picture

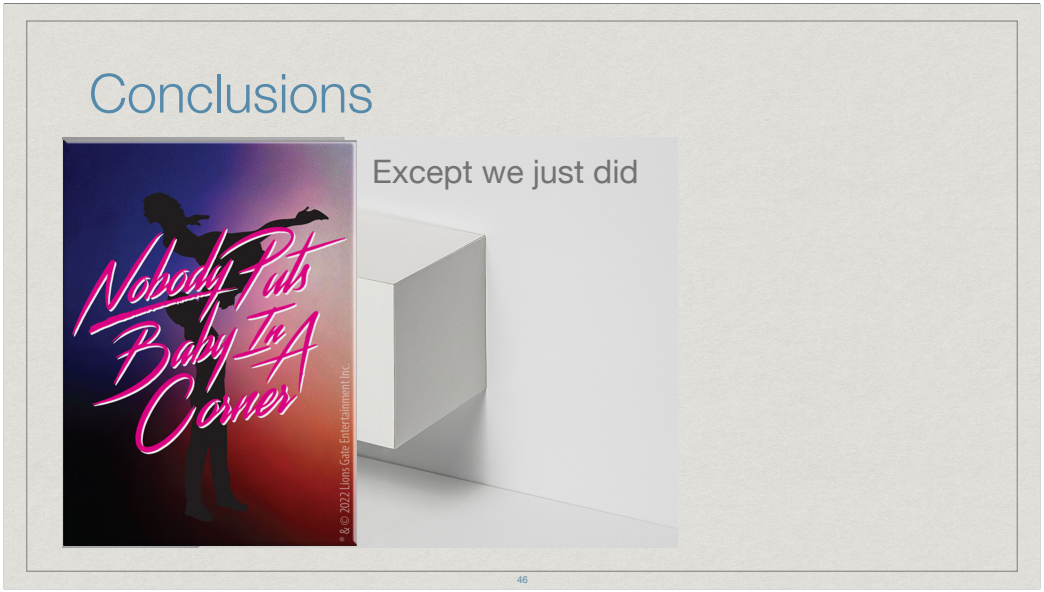
- This dissertation's primary original contribution is the *Lag Erasure* set of algorithms and architecture which provide users a lag-free distributed simulation.
- The *Lag Erasure* architecture includes novel handling of time offsets to decouple network latency (including jitter) effects from maintaining distributed shared state.

Enabling Details

- Defining and implementing the **`Latency Compensation Window`**
- Providing the *Speculative Evaluation* of physics-based simulation events
- The result of a consistent shared distributed state (previously documented as impossible)
- Enabling consistent LVC and C2 interoperability through this consistent state and physically-based simulations

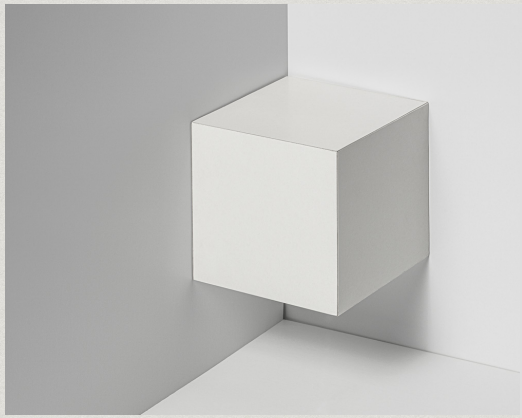
Conclusions



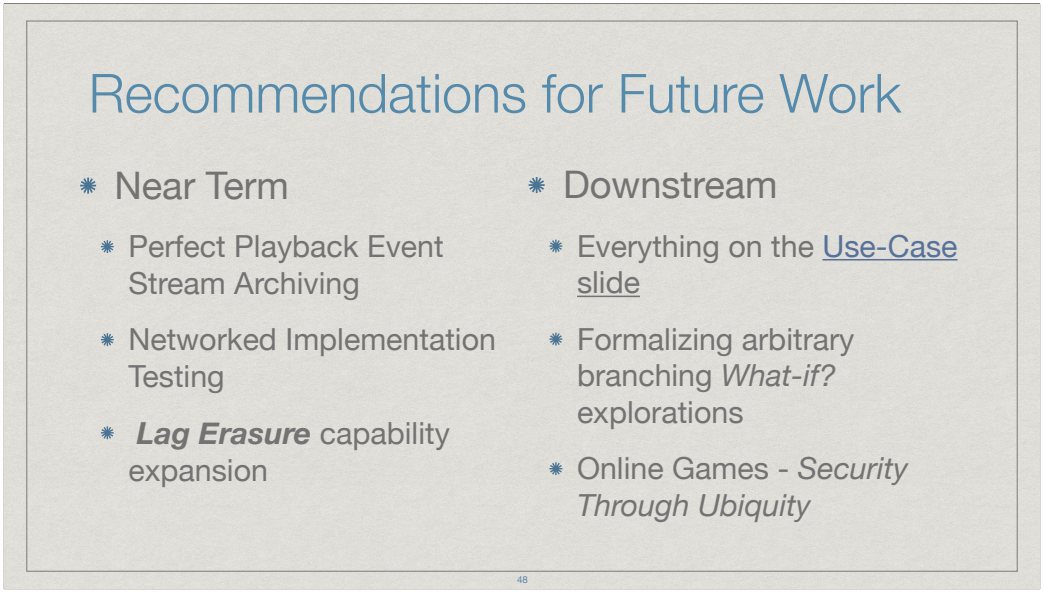


movie graphic with copyright lettermark —used under free use
box: royalty free https://stockcake.com/i/warehouse-corner-storage_1860145_1291652

Conclusions



- The *Consistency-Throughput Tradeoff* remains relevant
- But the impossibility of dynamic shared state maintenance is put in a box in the corner of a now greatly enlarged simulation operating environment
- Consistent distributed shared state is now possible



Doctor Strange style explorations of a problem space

A Few Relevant MoVES Dissertations & Theses:

A Locus of Simulation, Physics, Human Training, and AI Integration

- **1994 Brutzman**: A Virtual World for an Autonomous Underwater Vehicle
- **1995 Macedonia**: A Network Software Architecture for Large Scale Virtual Environments
- **2011 Alrowaie**: The Effect of Time-Advance Mechanism in Modeling and Simulation
- **2016 Cheng**: Hybrid High Fidelity Modeling of Radar Scenarios Using Atemporal, Discrete Event, and Time Step Simulation
- **2020 Maroon**: Predicting Opponent Position and Modeling Uncertainty
- **2022 Black**: Exploiting Kasparov's Law: Enhanced Information Systems Integration in DoD Simulation-Based Training Environments
- **2024 Rowan**: Model-Based Assessment of Adaptive Automations Unintended Consequences
- **2024 Black**: Mastering the Digital Art of War: Developing Intelligent Combat Simulation Agents for Wargaming Using Hierarchical Reinforcement Learning

- **2009 Walthen**: Dynamic Scalable Network Area of Interest Management for Virtual Worlds
- **2018 Tan**: Radar Envelope Visualization
- **2020 Knobeloch**: Military Mission Planning Using Augmented Reality Technology
- **2024 Marquez Bautista**: Adapting Interoperability in Geographically Distributed Simulations: A Novel Approach Using Cloud Computing

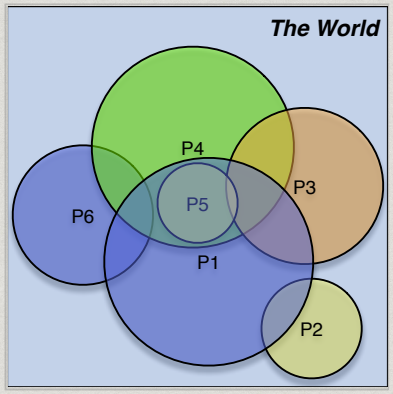


free use : https://www.pngkey.com/detail/u2q8a9y3o0a9u2u2_doctor-who-tardis-drawing/

Desktop image capture from internet, Artist Mike Horn for SF Fleet Week, capture from “I Left My Star Destroyer in San Francisco” 2009

Distributed Compute vs Distributed Sim

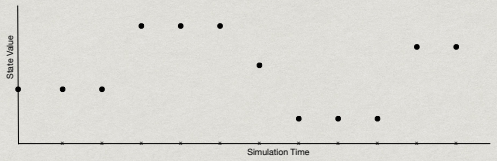
- Similar seeming problems
- Dynamism of continual change
- Not a database problem
- Discrete states vs. continuous state

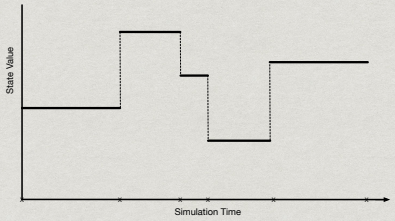


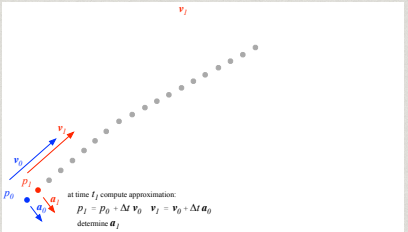
Time

- Virtual time
- Hardware clock synchronization
  - NTP
  - PTP

Time Advance







at time t_1 compute approximation:

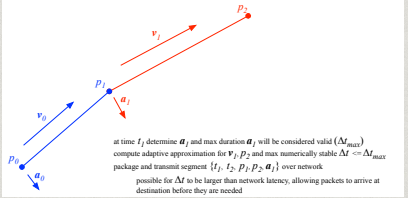
$p_1 = p_0 + \Delta t\ v_0 \quad v_1 = v_0 + \Delta t\ a_0$

determine a_1

package and transmit $\{t_1, p_1, v_1, a_1\}$ over network

typically network transmission latency is significantly larger than Δt so many timesteps pass before packet arrives at destination

repeat every Δt

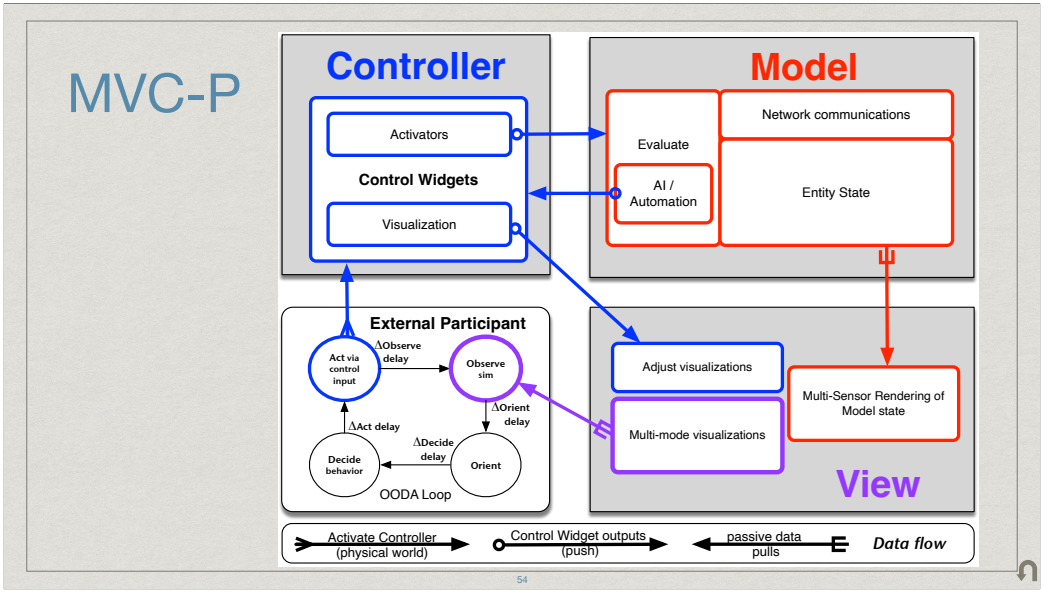


at time t_1 determine a_1 and max duration a_1 will be considered valid (Δt_{max})

compute adaptive approximation for v_1, p_2 and max numerically stable $\Delta t <= \Delta t_{max}$

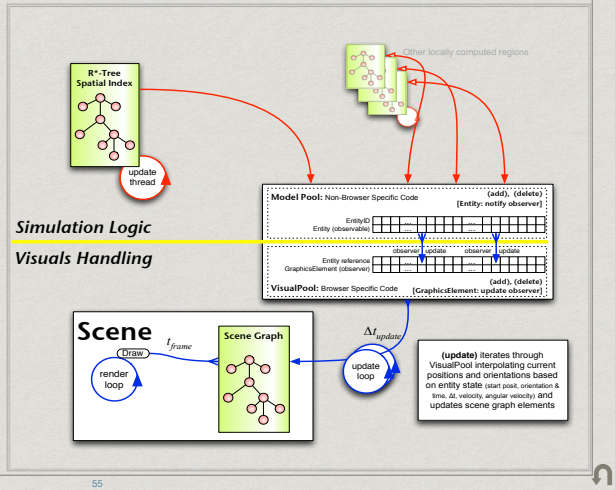
package and transmit segment $\{t_1, t_2, p_1, p_2, a_1\}$ over network

possible for Δt to be larger than network latency, allowing packets to arrive at destination before they are needed


MVC-P
Controller
Activators
Control Widgets
Visualization
Model
Evaluate
AI / Automation
Network communications
Entity State
External Participant
Act via control input
ΔObserve delay
Observe sim
ΔOrient delay
Orient
ΔDecide delay
Decide behavior
ΔAct delay
OODA Loop
Adjust visualizations
Multi-mode visualizations
Multi-Sensor Rendering of Model state
View
Activate Controller (physical world)
Control Widget outputs (push)
passive data pulls
Data flow
54

MVC-P

- Isolation of the **`Model`** and **`View`** data flow



Sphere-Swept Volumes

