Replicating Chaos

Vehicle Replication in Watch Dogs 2

Matt Delbosc
Team Lead Programmer
Ubisoft Toronto

GAME DEVELOPERS CONFERENCE® | FEB 27–MAR 3, 2017 | EXPO: MAR 1–3, 2017 #GDC17

UBM


Neutralize pat
Neutralize
132m
5:16
LB

Network architecture

- 4-player peer-to-peer
  - No single server
- Lots of entities to replicate
- Distributed authority

Why are vehicles hard?

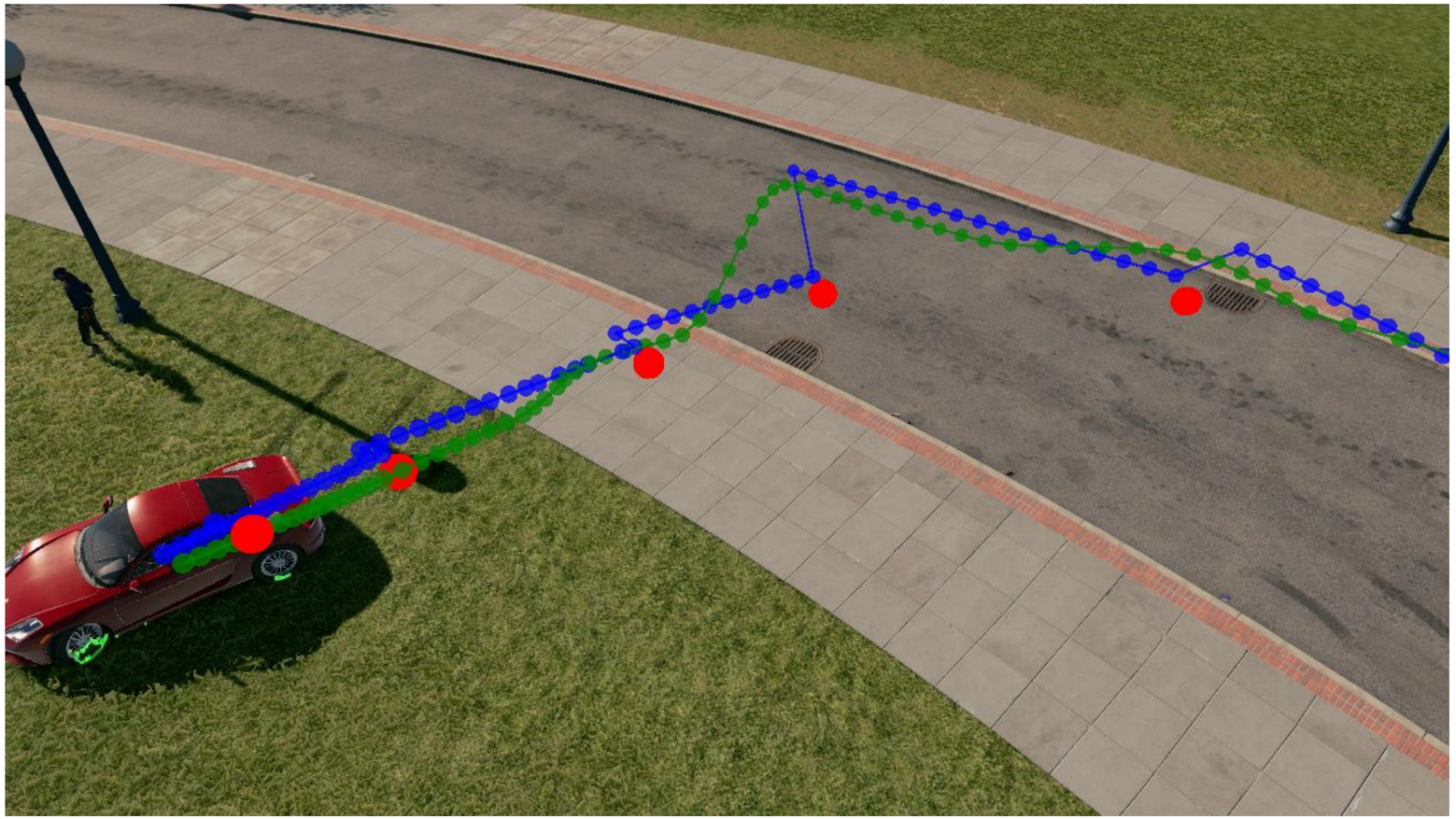
- They drive fast
  - 100ms lag could mean 2-3m difference
- They collide with each other
  - Lots of traffic in the Watch Dogs 2 open world
- The human eye is very sensitive to irregular velocities

What we will cover

- Debugging trajectories
- Standard techniques: Projective Velocity Blending (with a twist)
- Extrapolate and/or interpolate with Snapshot Buffers
- Dealing with collisions by blending with the physics simulation
- Some unsolved problems
- Future research directions

Trajectory Debugging

- Path Taken
- Snapshots
- Prediction



Projective Velocity Blending

Maths from Wikipedia

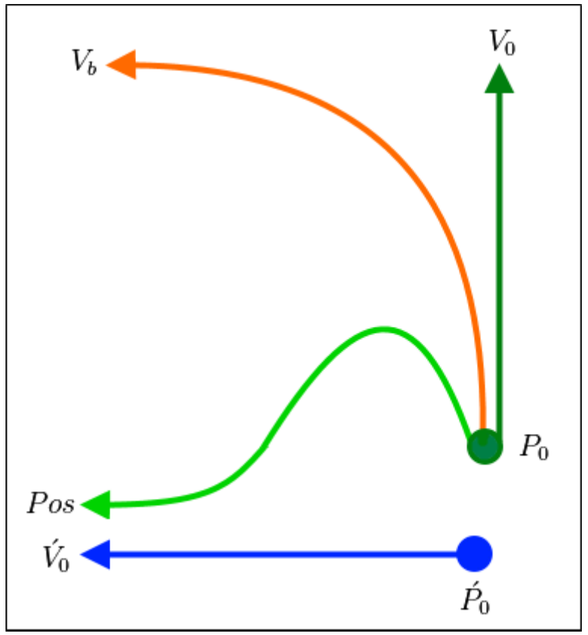
$$V_b = V_0 + \left(\acute{V}_0 - V_0\right)\hat{T}$$

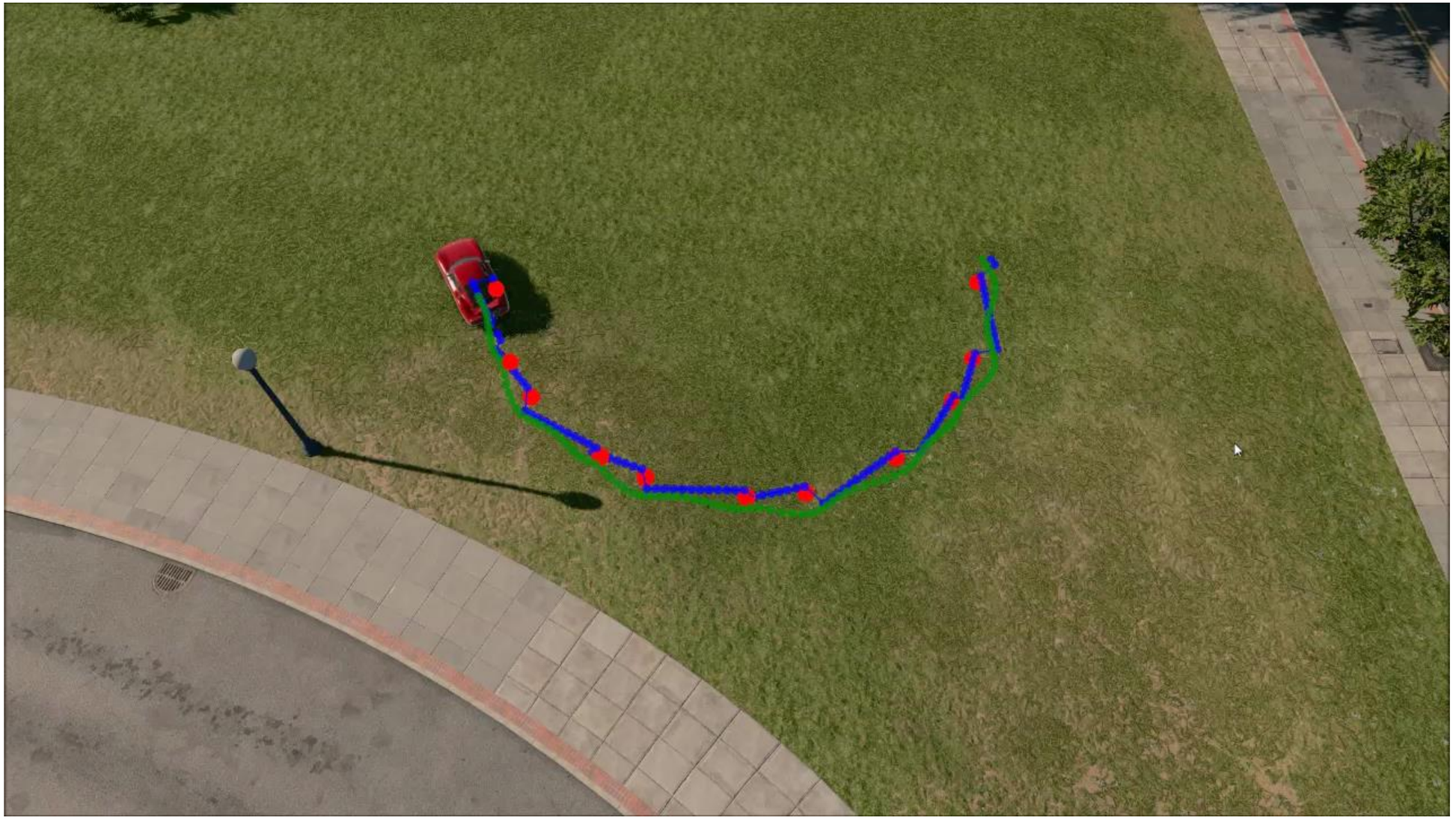
$$P_t = P_0 + V_b T_t + \frac{1}{2}\acute{A}_0 T_t^2$$

$$\acute{P}_t = \acute{P}_0 + \acute{V}_0 T_t + \frac{1}{2}\acute{A}_0 T_t^2$$

$$Pos = P_t + \left(\acute{P}_t - P_t\right)\hat{T}$$

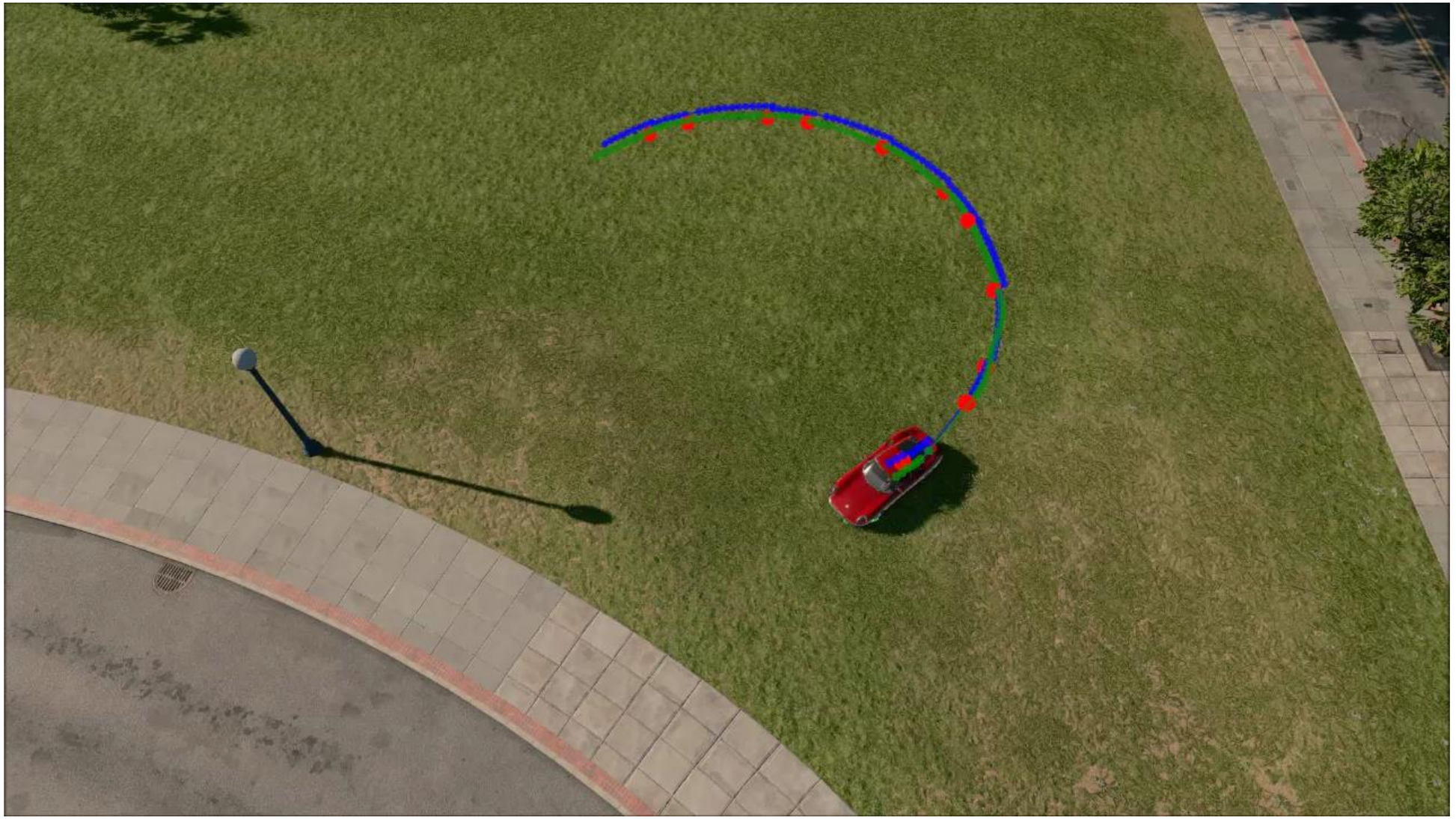
What this looks like





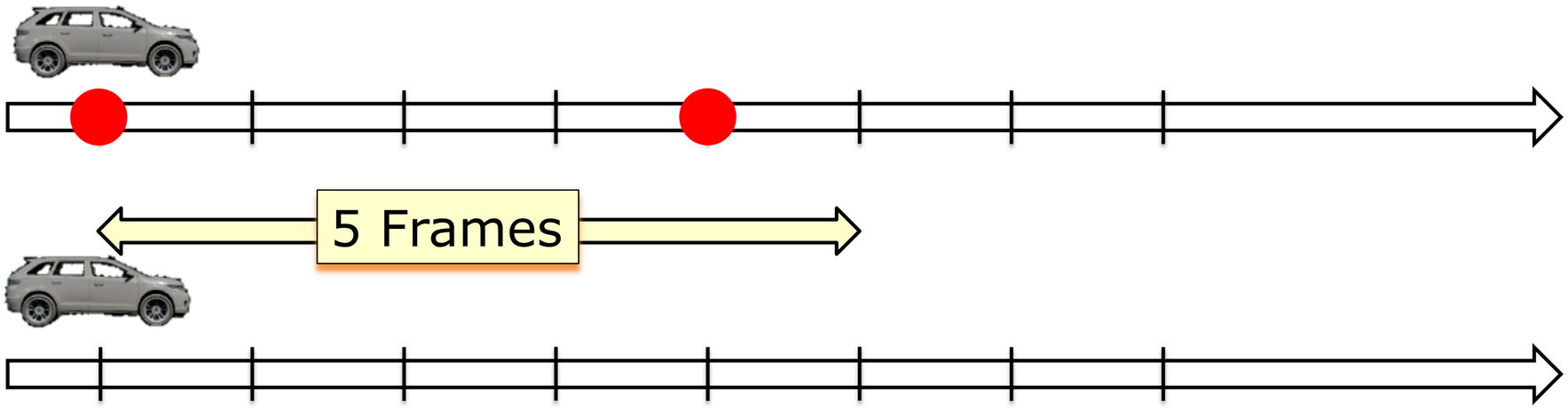
Turn Prediction

- Extrapolating a turning car with a straight line is not ideal
- We transmit the wheel steering angle
  - But the vehicle physics simulation is complex
  - Cannot predict an accurate turning trajectory from this
- We already transmit vehicle's position, velocity and angular velocity
  - Use yaw from angular velocity to predict turn
  - Doesn't work for drifting cars, but not an issue for our game



Snapshot Buffer

- Buffer snapshots and interpolate between last 2
- Pro: More accurate trajectory
- Con: Rendering objects further back in time



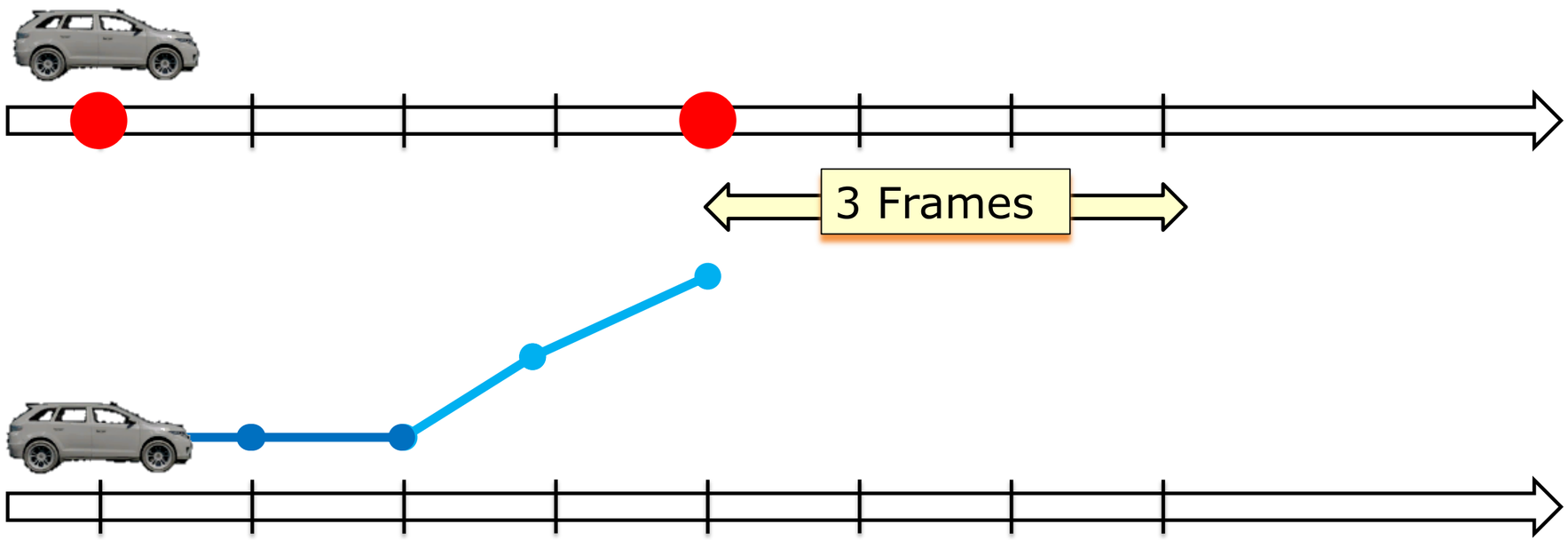
Too much interpolation

- Rendering objects that are further back in time

Extrapolation vs Interpolation

- We don’t have to interpolate
  - Can still extrapolate from last received snapshot
- Introduce a “time offset”
  - How far back in time do we render this car
  - Look at (Current time) – (Time offset):
    - Time falls between received snapshots => interpolate
    - Time ahead of last received snapshot => extrapolate
  - Snapshots are timestamped

How it looks



Time Offset

- Choosing the time offset is not easy
  - Too much interpolation causes missed collisions
  - Too much extrapolation causes discontinuous trajectories
  - Balance between the two
- Used (average measured lag) + (constant value)
  - Damped
  - Clamped at 300ms
  - Constant value in the 100ms – 200ms range
    - Proportional to vehicle speed

Vehicle collisions

- Dead reckoned trajectories are now looking good
- Let’s crash some cars


SPEED LIMIT
35
Q1U 1C2


01U 1C2

What happened?

- Both cars should stop, or push each other out of the way less forcefully

- Look at it frame-by-frame
  - Internal tool to help with this


Watch_Dogs 2[9540] -nofullscreen -xpos=1920 -ypos=0 -resolution=1280,720 -matchkey=mdk -world=san_francisco -login=mdelbosc-1 -nouplayflow -publicCoopSession -useworldstartpoint -disableonlinetimeout -no...

Can we solve this?

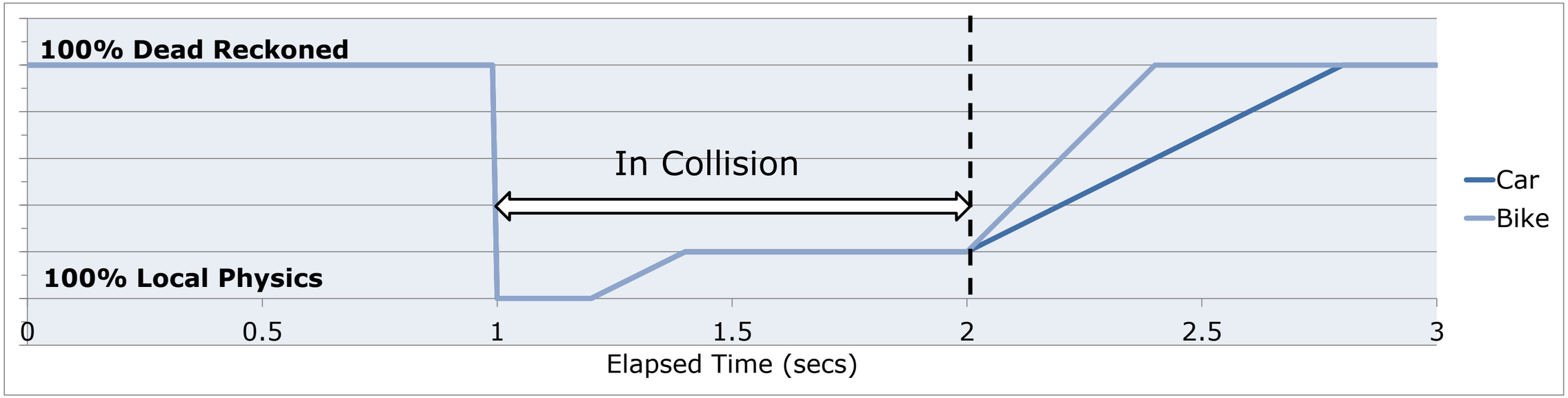
- The replica that pushed you doesn’t know about the collision yet
  - They will keep driving forward until a new packet arrives
- This is an unsolvable problem!
  - No one has full authority over the collision
- Gee I wish I had a server ☹

Physics Simulation Blending

- The collision has happened whether we like it or not
- Give local physics a chance to simulate the collision
- Then blend back into to the dead reckoned trajectory
- Tunable blend between two velocities:
  - Velocity applied by the rigid body simulation
  - Velocity to reach the dead reckoned position
- Representing something believable locally, then blend into snapshots that take the collision into account

Blend Factor Tuning

- Play out lots of collisions and choose what looks good
  - Max blend factor to use while in a collision
  - How quickly to go back to full dead reckoning afterwards



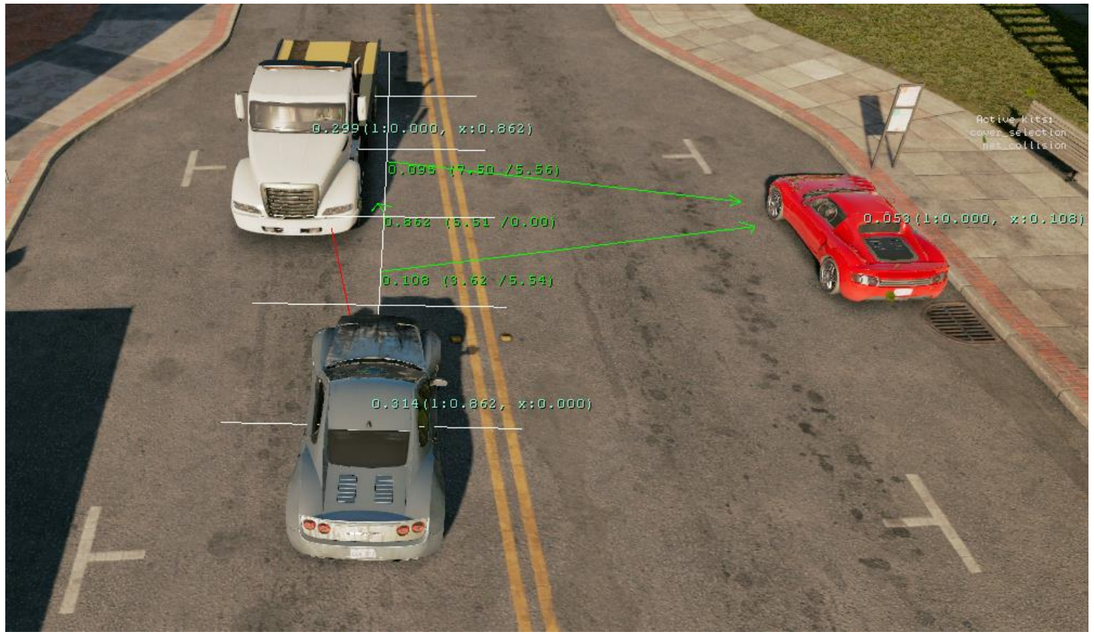

RAUMSTEIGER
X4W 7N7

Collision Unpredictability

- Locally simulated collisions won’t play out the same on both peers
- Blending replica out of locally simulated collision into dead reckoned trajectory could be jarring
- Before the collision starts, try to bring vehicles closer to their master positions
  - Make them more likely to play out the same

Collision Prediction

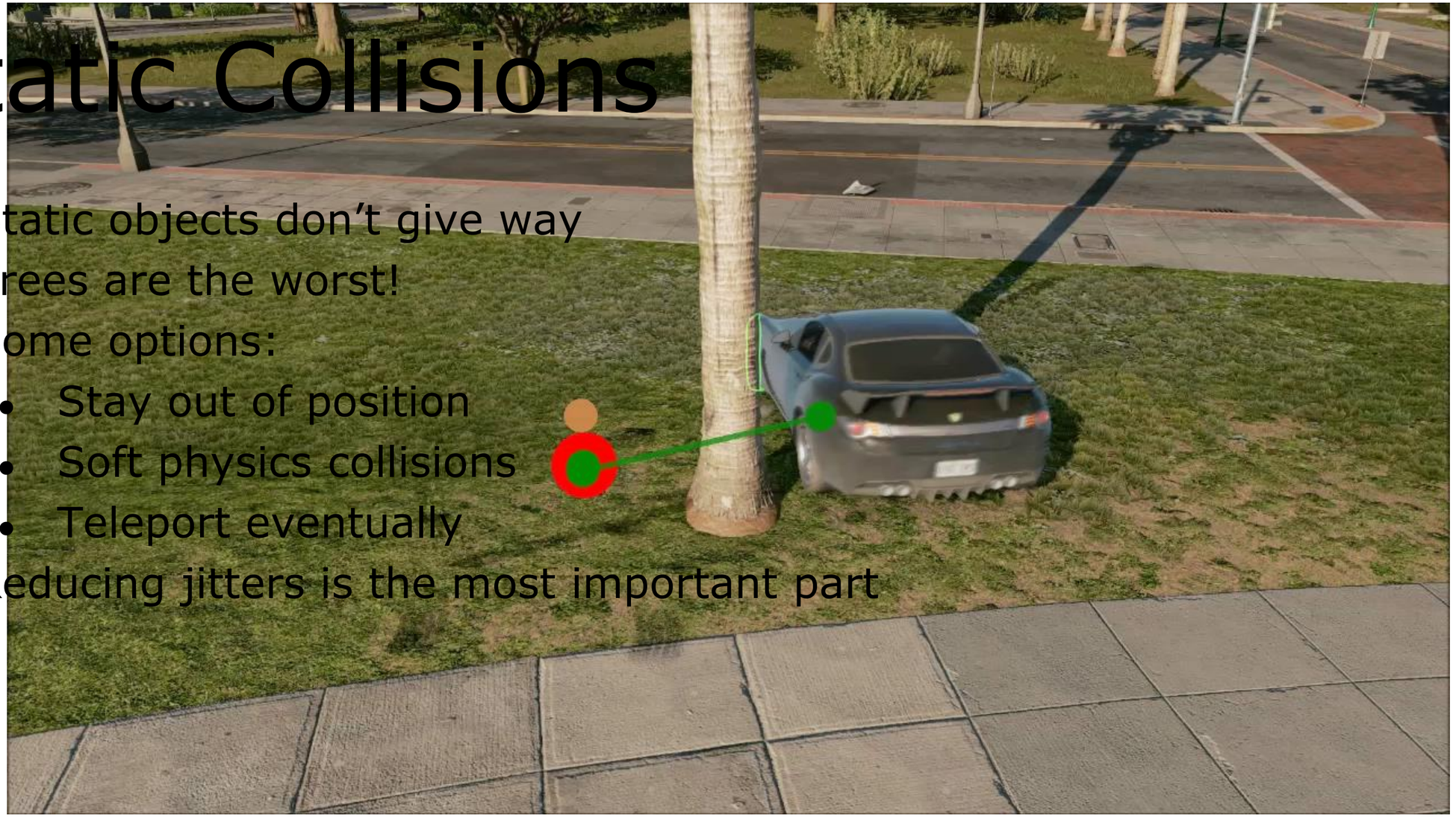
- Predict imminent collisions and extrapolate more in those cases
  - Unlikely to steer out of the way
  - Use existing algorithm from AI systems



Unsolved Problems

- Some of many outstanding problems:
  - Collisions with static geometry
  - Time offset differences
  - The uncanny valley

Static Collisions



- Static objects don’t give way
- Trees are the worst!
- Some options:
  - Stay out of position
  - Soft physics collisions
  - Teleport eventually
- Reducing jitters is the most important part

Different Time Offsets

- Pedestrians prefer interpolation

- Similar issue for breakables, but less gameplay impact

The Uncanny Valley

- Trajectories are correct, but it still doesn’t feel right
  - Minor weight shifts lost in replication
  - Cars don’t rotate around their centre of mass
  - Human eye is sensitive to even minor inconsistencies
- Apply some post-process smoothing
  - Smooth velocities and angular velocities, not positions / rotations
  - Has to co-exist with Projective Velocity Blending

Future Investigations

- Machine-learning based algorithms
  - Self-tweaking variables!
- Smoothing algorithms
  - Kalman filters

Take-aways

- Debug your trajectories
  - Record everything
- Somehow there is still a lot of art to this science

UBISOFT

MORE QUESTIONS?

MEET ME ON THE UBISOFT LOUNGE

TODAY from 2PM to 3PM

WEST HALL, 2ND FLOOR

#UBIGDC

matt.delbosc@ubisoft.com