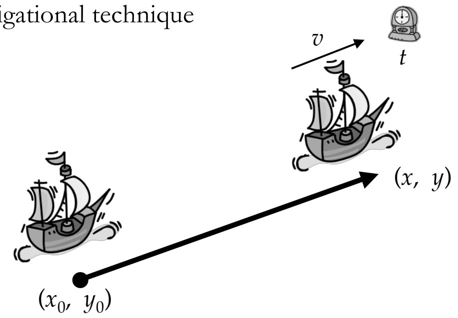
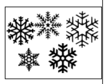
§9.3 Dead Reckoning

- navigational technique

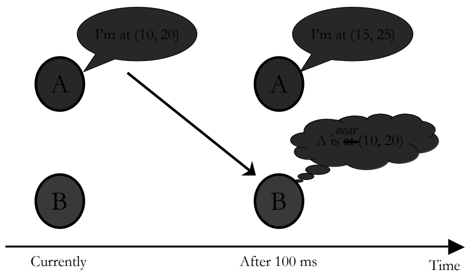


Dynamic Shared State



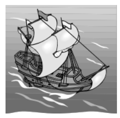
- Dynamic shared state constitutes the changing information that multiple nodes must maintain
  - participants, their locations and behaviours
  - environment itself, all objects, weather, natural laws,...
- In a highly dynamic environment, almost all information about the game world may change ⇒ needs to be shared
- Accuracy is fundamental to creating realistic environments
- Makes an environment available to multiple users
  - without dynamic shared state, each user works independently (and alone)

Example of Dynamic Shared State

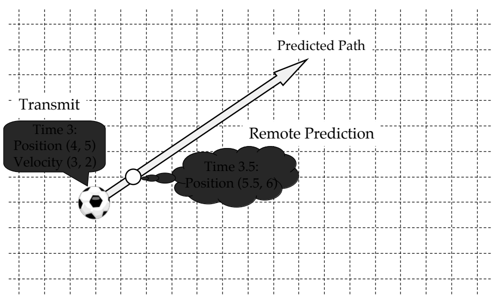


Dead Reckoning of Shared State

- Transmit state update packets less frequently
- Use received information to *approximate* the true shared state
- In between updates, each node predicts the state of the entities



Dead Reckoning: Example

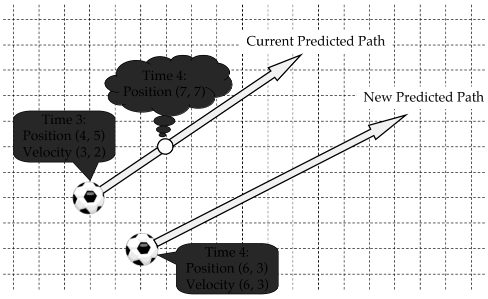


Dead Reckoning Protocol

DR protocol consists of two elements:

- prediction technique
  - how the entity's current state is computed based on previously received update packets
- convergence technique
  - how to correct the state information when an update is received

Prediction and Convergence



Prediction Using Derivative Polynomials

- The most common DR protocols use derivative polynomials
- Involves various derivatives of the entity's current position
- Derivatives of position
  1. velocity
  2. acceleration
  3. jerk

Zero-Order and First-Order Polynomials

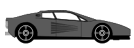
- Zero-order polynomial
  - position p
  - the object's instantaneous position, no derivative information
  - predicted position after t seconds = p
- First-order polynomial
  - velocity v
  - predicted position after t seconds = $vt + p$
  - update packet provides current position and velocity

Second-Order Polynomials

- We can usually obtain better prediction by incorporating more derivatives
- Second-order polynomial
  - acceleration a
  - predicted position after t seconds $= \frac{1}{2}at^2 + vt + p$
  - update packet: current position, velocity, and acceleration
  - popular and widely used
  - easy to understand and implement
  - fast to compute
  - relatively good predictions of position

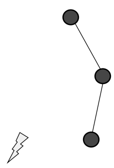
Hybrid Polynomial Prediction

- The remote host can dynamically choose the order of prediction polynomial
  - first-order or second-order?
- First-order
  - fewer computational operations
  - good when acceleration changes frequently or when acceleration is minimal
  - prediction can be more accurate without acceleration information



Position History-Based Dead Reckoning

- Chooses dynamically between first-order and second-order
- Evaluates the object's motion over the three most recent position updates
- If acceleration is minimal or substantial, use first-order
  - threshold cut-off values for each entity
- The acceleration behaviour affects to the convergence algorithm selection
- Ignores instantaneous derivative information
  - update packets only contain the most recent position
  - estimate velocity and acceleration
- Reduces bandwidth requirement
- Improves prediction accuracy in many cases



Limitations of Derivative Polynomials

- Add more terms to the derivative polynomial—why not?
- With higher-order polynomials, more information have to be transmitted
- The computational complexity increases
  - each additional term requires few extra operations
- Sensitivity to errors
  - derivative information must be accurate
  - inaccurate values for the higher derivatives might actually make the prediction worse

$p(t) = \frac{1}{2}at^2 + vt + p$

Limitations of Derivative Polynomials (cont'd)

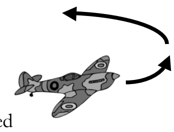
- Hard to get accurate instantaneous information
  - entity models typically contain velocity and acceleration
  - higher-order derivatives must be estimated or tracked
  - defining jerk (change in acceleration):
    - predict human behaviour
    - air resistance, muscle tension, collisions,…
  - values of higher-order derivatives tend to change more rapidly than lower-order derivatives

⇒High-order derivatives should generally be avoided

- The Law of Diminishing Returns
  - more effort typically provides progressively less impact on the overall effectiveness of a particular technique

Object-Specialized Prediction

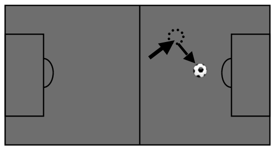
- Derivative polynomials do not take into account
  - what the entity is currently doing
  - what the entity is capable of doing
  - who is controlling the entity
- Managing a wide variety of dead reckoning protocols is expensive



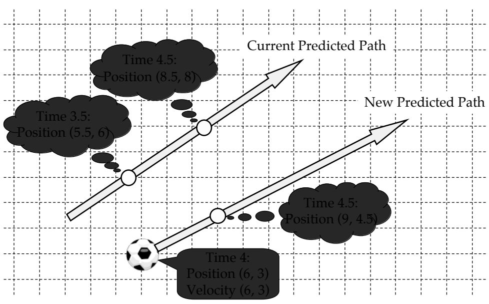
- Aircraft making military flight manoeuvers
  - constant acceleration and instant velocity ⇒ position trajectory
  - the aeroplane's orientation angle
- All information does not need to be transmitted
  - dancing is relevant not the footwork, fire not the flames,…
- In general, precise behaviour would be nice but overall behaviour is enough

Convergence Algorithms

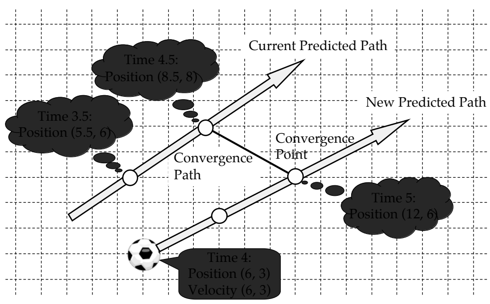
- Prediction estimates the future value of the shared state
- Convergence tells how to correct inexact prediction
- Correct predicted state quickly but without noticeable visual distortion



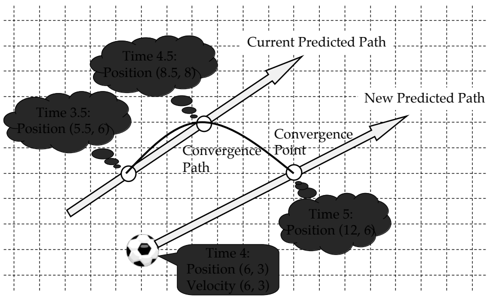
Zero-Order Convergence (or Snap)



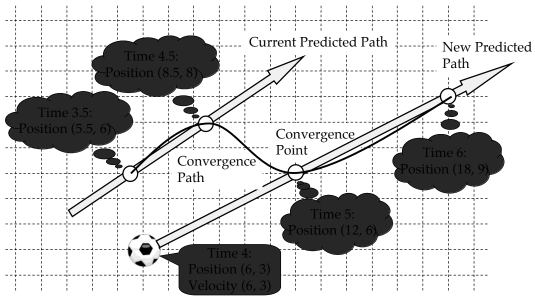
Linear Convergence



Quadratic Convergence



Convergence with Cubic Spline



Nonregular Update Generation

- By taking advance of knowledge about the computations at remote host, the source host can reduce the required state update rate
- The source host can use the same prediction algorithm than the remote hosts
- Transmit updates only when there is a significant divergence between the actual position and the predicted position

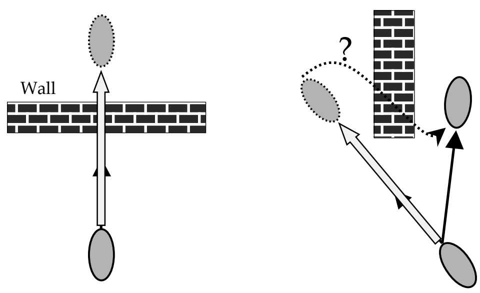
Advantages of Nonregular Transmissions

- Reduces update rates, if prediction algorithm is reasonable accurate
- Allows to make guarantees about the overall accuracy
- The source host can dynamically balance its network transmission resources
  - limited bandwidth ⇒ increase error threshold
- Nonregular updates provide a way to dynamically balance consistency and responsiveness based on the changing consistency demands

Lack of Update Packets

- If the prediction algorithm is really good, or if the entity is not moving significantly, the source might never send any updates
- New participants never receive any initial state
- Recipients cannot tell the difference between receiving no updates because
  - the object's behaviour has not changed
  - the network has failed
  - the object has left the game world
- Solution: timeout on packet transmissions

Environmental Effects



Dead Reckoning: Advantages and Drawbacks

- Reduces bandwidth requirements because updates can be transmitted at lower-than-frame-rate
- Because hosts receive updates about remote entities at a slower rate than local entities, receivers must use prediction and convergence to integrate remote and local entities
- Does not guarantee identical view for all participants
  - tolerate and adapt to potential differences
- Complex to develop, maintain, and evaluate
- Dead reckoning algorithms must often be customized for particular objects
- Are entities predictable?