Bullet Time

- movies: visual effect combining slow motion with dynamic camera movement
- computer games: player can slow down the surroundings to have *more time* to make decisions
- easy in single player games: slow down the game!
- how about multiplayer games?

Bullet Time in Multiplayer Games

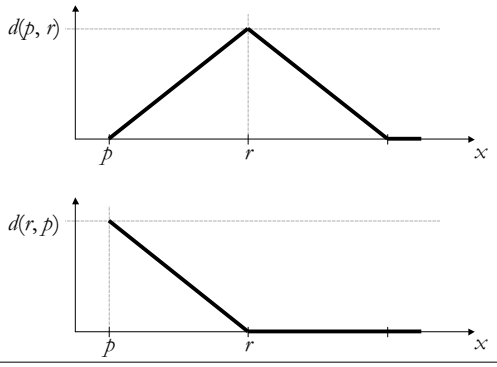
- two approaches:
  - speed up the player
  - slow down the other players
- if a player can slow down/speed up the time, how it will affect the other players?
  - localize the temporal distortion to the immediate surroundings of the player
- but how to do that?

  ⇒ local perception filters!

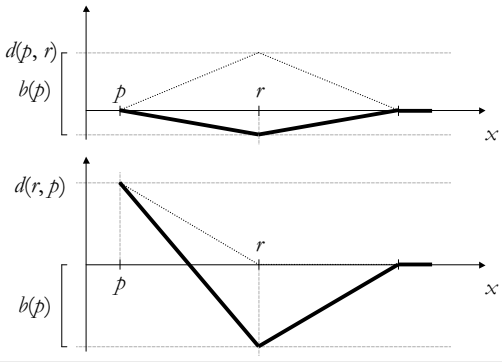
Adding Bullet Time to LPFs

- player using the bullet time has more time to react
  ⇒ the delay between bullet-timed player and the other players increases
- add artificial delay to the temporal contour

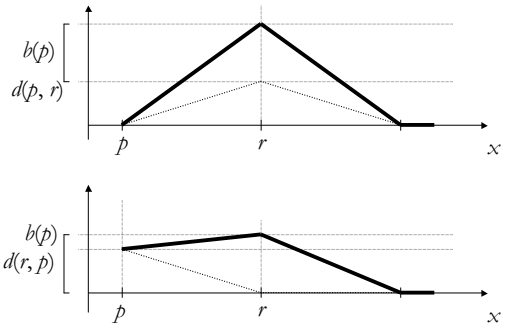
p Shoots *r* Without Bullet Time



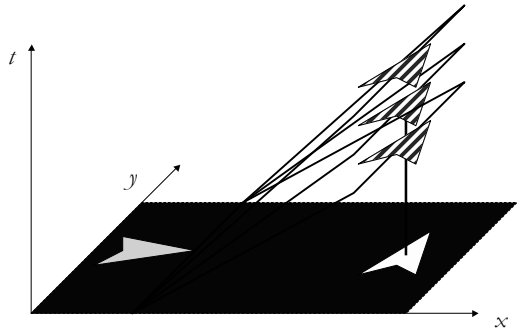
p Shoots *r* While *p* Is Using Bullet Time



p Shoots *r* While *r* Is Using Bullet Time



2½-Dimensional Temporal Contour and Bullet Time



Open Questions

- non-linear temporal contours
  - how to compute quickly?
  - noticeable benefits (if any)?
- numerical evaluation
  - measuring the distortion and its effects
- practical evaluation
  - how well does it work?
  - does it allow new kinds of games?

§9.5 Synchronized Simulation

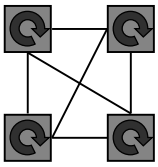
- used in *Age of Empires* (1997)
- command categories:
  - deterministic: computer
  - indeterministic: human
- distribute the indeterministic commands only
- deterministic commands are derived from pseudo-random numbers
  → distribute the seed value only
- consistency checks and recovery mechanisms

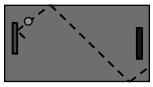
Synchronized Simulation in *Age of Empires*

- *Age of Empires* game series by Ensemble Studios
- Real-time strategy (RTS) game
- Max 8 players, each can have up to 200 moving units
  ⇒ 1600 moving units
  ⇒ large-scale simulation
- Rough breakdown of the processing tasks:
  - 30% graphic rendering
  - 30% AI and path-finding
  - 30% running the simulation and maintenance

Synchronized (or Simultaneous) Simulation

- Large simulation ⇒ a lot of data to be transmitted
- Trade-off: computation vs. communication
  - 'If you have more updating data than you can move on the network, the only real option is to generate the data on each client'
- Run the *exact* same simulation in each client



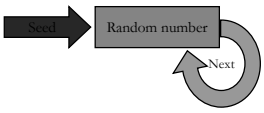


Handling Indeterminism

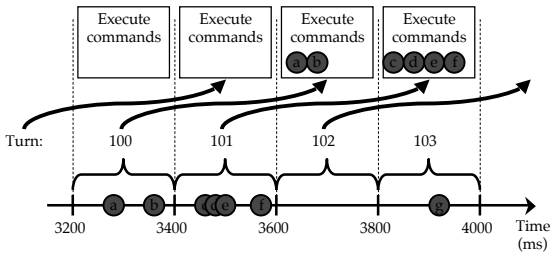
- 'Indeterministic' events are either
  - predictable (computers) or
  - unpredictable (humans)
- Only the unpredictable events have to be transmitted
  ⇒ communication
  - apply an identical set of commands that were issued at the same time
- The predictable events can be calculated locally on each client
  ⇒ computation

- Pseudo-random numbers are deterministic
- All clients use the same seed for their random number generator
  - disseminate the seed

Pseudo-random number generator

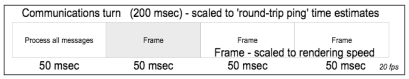


Communication Turns

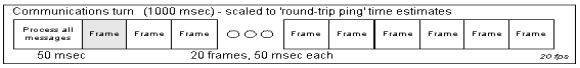


Division of the Communication Turn

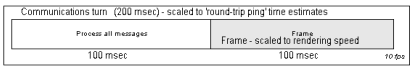
Single communication turn



High Internet latency with normal machine performance



Poor machine performance with normal latency



Features

- Guaranteed delivery using UDP
  - message packet:
    - execution turn
    - sequence number
  - if messages are received out of order, send immediately a resend request
  - if acknowledgement arrives late, resend the message
- Hidden benefits
  - clients are hard to hack
  - any simulation running differently is out-of-sync
- Hidden problems
  - programming is demanding
  - out-of-sync errors

Lessons Learned

- Players can tolerate a high latency as long as it remains constant
  - for an RTS game, even 250–500 ms latencies are still playable
- Jitter (the variance of the latency) is a bigger problem
  - consistent slow response is better than alternating between fast and slow
- Studying player behaviour helps to identify problematic situations
  - hectic situations (like battles) cause spikes in the network traffic
- Measuring the communication system early on helps the development
  - identify bottlenecks and slowdowns
- Educating programmers to work on multiplayer environments