Realizing Bullet Time in Multiplayer Games with Local Perception Filters

Jouni Smed, Henrik Niinisalo, Harri Hakonen
Turku Centre for Computer Science (TUCS)
Department of Information Technology,
University of Turku, Finland

Bullet time

- movies: visual effect combining slow motion with dynamic camera movement
- computer games: player can slow down the surroudings 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 (LPFs)

- introduced by Sharkey, Ryan & Roberts (1998)
- a method for hiding communication delays in networked virtual environments
- exploits the human perceptual limitations by rendering entities slightly out-of-date locations based on the underlying network delays
  - causality of events is preserved
  - rendered view may have temporal distortions
  - rendered view ≠ real view

Rules of LPFs

1. Player should be able to interact in real-time with the nearby entities.
2. Player should be able to view remote interactions in real-time, although they can be out-of-date.
3. Temporal distortions in the player's perception should be as unnoticeable as possible.

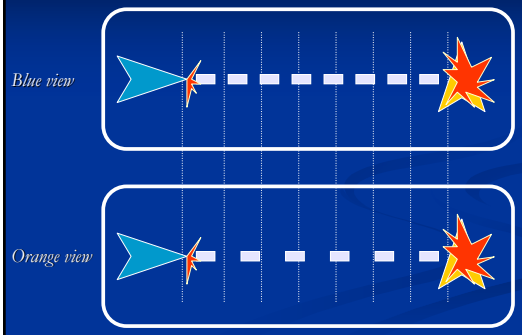
Entity types

- active: indeterministic, unpredictable (humans)
  ⇒ *players*
  - local: residing in the same computer
  - remote: connected over a network
- passive: deterministic, predictable (projectiles, buildings etc.)
  ⇒ *entities*

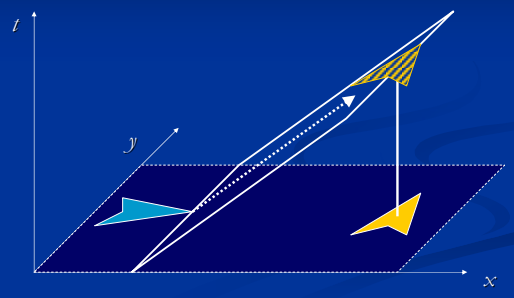
Interaction between players

- interaction = communication between the players
  - local players: immediate
  - remote players: subject to the network latency
    - time frame = current time – communication delay
- interaction = players exchanging passive entities
  - passive entities are predictable ⇒ they can be rendered in the past (or in the future)
- a passive entity can change its time frame dynamically
  - the nearer to a local player, the closer it is rendered to the current time
  - the nearer to a remote player, the closer it is rendered to its time frame

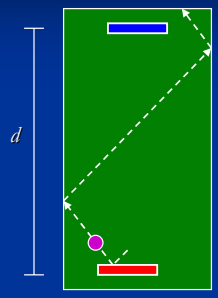
Example: Temporal distortion



Temporal contour (from the blue player's perspective)

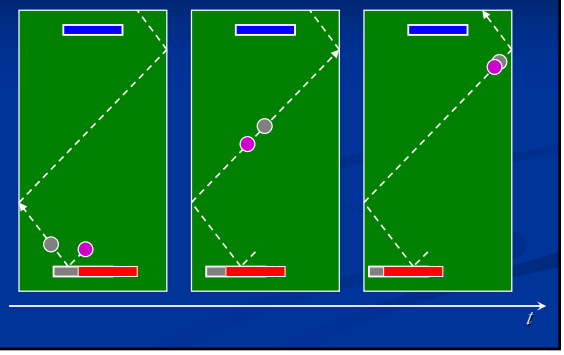


Example: Pong

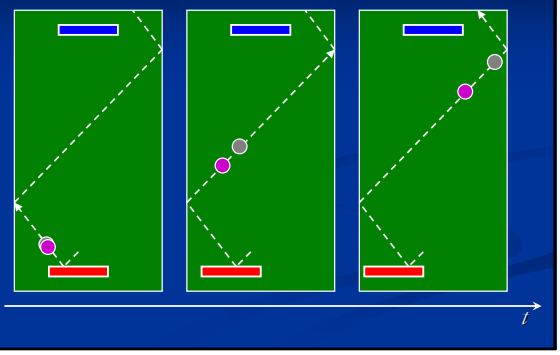


- two active entities: paddles
  - movements unpredictable
- one passive entity: ball
  - movements predictable
- latency of *d* seconds

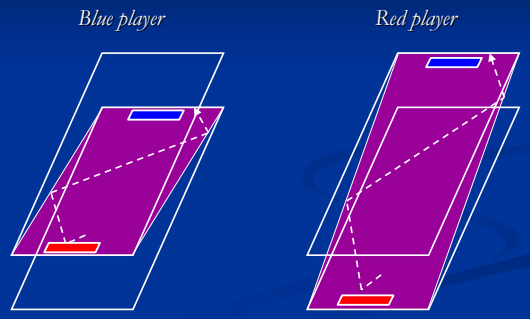
The view of the blue player



The view of the red player

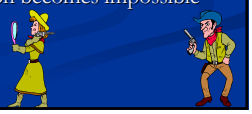


Temporal contours in Pong

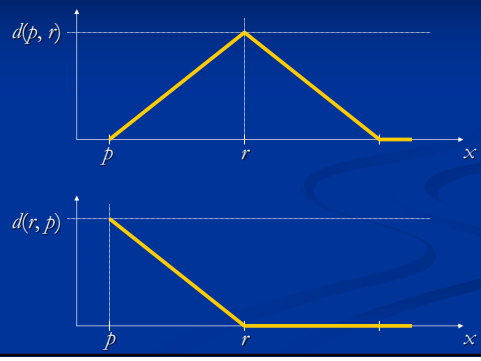


Worth noting

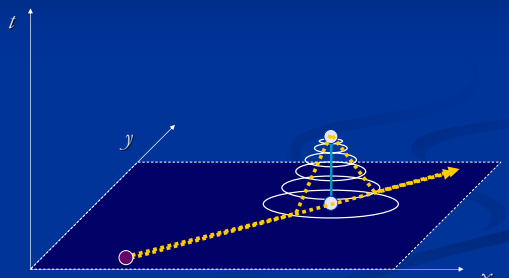
- original paper: continuous temporal contours; here: simple linear functions
- LPFs are the 'opposite' of dead reckoning
  - no prediction for remote players
- the closer the players get, the more noticeable the temporal distortion becomes
  - in critical proximity interaction becomes impossible
  - no mêlée



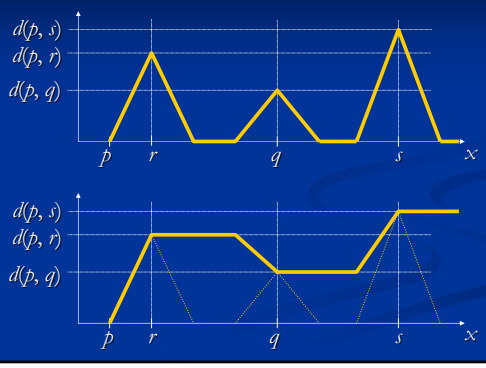
Example: Temporal contours



2½-dimensional temporal contour



Many players



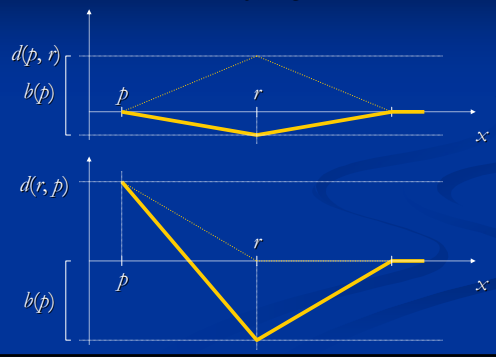
Problems

- original approach: visual disruptions on impact ⇒ shadows (see the paper for details)
- sudden changes in the player's position or delay can cause unwanted effects
  - if a player leaves the game, what happens to the temporal contour?
  - third party instrusion: someone with a high delay 'blocks' the incoming entities
  - jitter: entities start to bounce back and forth in time

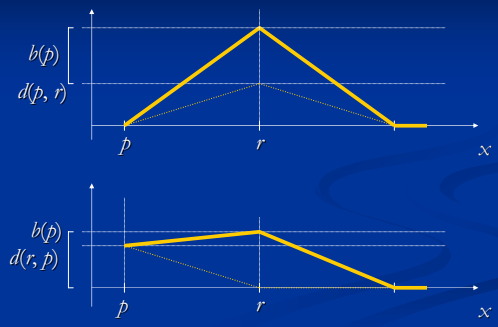
Adding bullet time

- 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

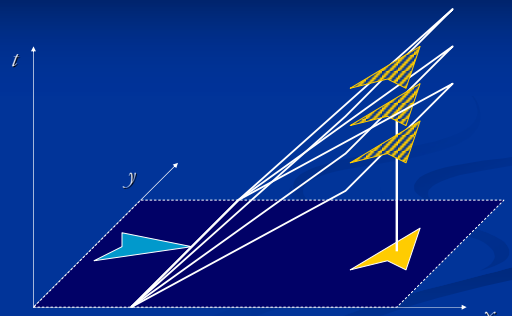
p shoots *r* while *p* is using bullet time



p shoots *r* while *r* is using bullet time

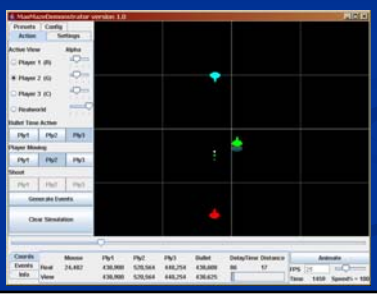


2½-dimensional temporal contour and bullet time



MaxMazeDemonstrator

http://staff.cs.utu.fi/staff/jouni.smed/mmd/



Future work

- 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?