Consistency requirements of games

Alessandro Monticelli, Jóhannes Helgi Tómasson, Jonas Stahl April 3, 2026

Envisage a multi-user game where players move figures around a common scene. The state of the game is replicated at the players' workstations and a server. The server contains services controlling the game, such as collision detection. Updates are multicast to all replicas. Consider the following conditions:

1. The figures may throw projectiles at one another, and a hit debilitates the unfortunate recipient for a limited time. What type of update order is required here? Consider the 'throw', 'collide' and 'revive' events.

Listing 1: Causal ordering of actions as seen by server and clients

```
1 P1: W(throw1)                        R(hit2)              R(revive2)
2 P2:                                  R(hit2)              R(revive2)
3 S:              R(throw1)  W(hit2)            W(revive2)
```

The events form the causal chain:

$$throw1 \rightarrow hit2 \rightarrow revive2$$

The chain of events can be demonstrated with causal consistency ordering. $P1$ throws at $P2$ the server reads the throw, and multi-casts to all players that $P2$ has been hit. Some time later, the server revives $P2$ and multi-casts the information to all players. No process can read a hit preceding a throw and no revive preceding a hit. Note that not every throw leads to a hit, and not every hit leads to a revive, but hits are causally dependent on a throw and revives are causally dependent on a hit. Independent throws by different players are concurrent, and do not need to be observed in the same order in all processes.

2. The game incorporates magic devices a player may pick up to assist them. What type of order should be applied to the 'pickup-device' operation?

Listing 2: Total ordering of actions as seen by server and clients

```
1 P1: W(pickup)a                                    R(give)a
2 P2: W(pickup)b                                    R(give)a
3 S:              R(pickup)a  R(pickup)b  W(give)a
```

Two players concurrently attempt to pickup the same item. Since an item can only be owned by a single player, the system must ensure all players agree on who picked up the item first. The pickup must therefore be totally ordered. A total order guarantees every player observes pickup events in the same sequence. The server orders events as:

$$(pickup)a \rightarrow (pickup)b$$

The server reads pickups a and b from both players and orders that player 1 was the first to pickup. The server broadcasts to all players that player 1 has the item. The second player's request fails, since the item has already been assigned. This ensures all replicas of the game state remain consistent.