Introduction

It’s a single ball game. The paddle has some special powers. So, there is a POWER SEGMENT of the bat that covers mid ten percent of the length of the bat. It’s powerful as in this region super elastic collisions occur i.e. if ball strikes in this portion, then the outgoing speed is greater than the incoming speed hence it makes it difficult for the next player. So, a player would generally try to ensure he uses this portion of bat rather than rest of the bat so that the reflected ball becomes difficult for the next player to tackle and there are significant chances of him missing the ball.

Physics equations: Let initial velocity of ball along x is vx and along y is vy, on collision with left and right walls/paddle the ball’s velocity would become -e \* vx and vy respectively, while on collision with top and bottom walls/pedals final velocity becomes vx and –e \* vy respectively, where “e” is the coefficient of restitution which will be 1 in case of elastic collision. To give the effect of friction we can just multiply velocities by a small magnitude (e.g. 0.001) on new rendering state.

Assumptions: We are treating the ball to be a point mass and paddle surface to be frictional. We assume wall collisions are perfectly elastic while paddle collisions are inelastic (or super elastic).

Initially, ball is at center and its angle with x-axis is randomized. So, it moves in any random direction at the start. Ball’s speed is changing in the game as we have frictional paddles, inelastic and super collisions

Corner cases

If ball strikes exactly the sharp point of the corner bricks we don't assume it as a fault of either player who share that corner brick. But if ball strikes the brick at any of its lateral side, its assumed equivalent to ball striking the wall. Similarly, ball colliding with the lateral part of the paddle is equivalent to colliding with the wall.

Levels of Difficulty

EASY: ball moves in straight line, starting ball speed is low, fixed speed of bot.

MEDIUM: ball moves in curves (so trajectory of ball is well defined and predictable), medium initial speed of ball, bot speed has a maximum speed cap.

HARD: ball moves in straight lines but with sharp turns and slope of line can change at any random instant, hence it's trajectory is unpredictable. Ball's initial speed is high. Ideal bot which never misses the ball.

Computer Player

It monitors ball's trajectory right from previous collisions. The ball's trajectory is extended to check if the ball will land up in it's court. Once, it has found where exactly the ball will land, it decides whether to move up/down. This calculation is done w.r.t the center of the paddle. In case of easy level, bot moves with constant speed up/down. In case of medium level, it first calculates the requisite speed, if it exceeds the maximum speed then it will move with maximum speed and it might also miss the ball if this speed isn't sufficient. In case of ideal bot, it always moves with the requisite speed. If it strikes the end of the wall, it stops there itself. So, this never misses the ball and in maximum cases strikes from the middle area. Computer player runs on all machines in a distributed fashion.

Features

1) Countdown at start

2) Special Power Bat that supports super elastic collisions

3) Three levels of difficulty

Network Communication

1) Information exchange between different machines

Coordinates & velocity of paddle and ball and actions like hit, miss, die will be exchanged over network. These actions will be mapped to corresponding functions and will be packed with other data and sent over PEER TO PEER NETWORK.

2) Maintaining Game State

Every peer is responsible for passing on the information about its own paddle. This means that, if four players are A, B, C & D, then player A is the only one able to inform where his paddle is? And where the ball is? *And did the paddle hit the ball? Or did the ball collided with his side wall?* All other players will receive messages from A informing about his actions and they will react accordingly. As a consequence, each player will see all other paddles (and their actions) and ball according to the received messages.

An important step in ensuring that every player will be able to see the same simulation accurately is the identification of relevant actions, those that change the current game state. The important actions are:

* HIT (player's paddle hits the ball)
* MOVE (player's paddle moved)
* MISS (ball collided with wall)
* DIE (player's paddle is removed since ball touched his side of wall 3 times).

3) Event Flow for network message

Map actions with

Pack data and actions-send- unpack

4) Trigger event deciding disconnection of player’s machine from the network

When a player is disconnected, then an exception is generated in the socket which can be handled by a try catch clause.

5) Response to disconnection of player’s machine

Once machine has figured out that a player's machine is no more connected, it first generates a notification instructing that the particular machine is disconnected. Now, we will give that player’s control to the computer on disconnection from the network and in case the player is reconnected to the network that player can resume with his current score to play with other players.

while not done collect player input collect network input about other players simulate player update player's state if informed it's been hit inform other clients if they've been hit move player update other player state (ammo, armor, etc.) report player state to other clients