# Construct 2 Multiplayer Engine - WebRTC

The Construct 2 platform's multiplayer engine is capable of exchanging up to 30 messages per second, approximately one message every 33 ms. This rate is enough for games that run at 30 fps. For those which run in 60 fps, the engine allows you to send data every 2 frames.

All Construct 2 platform predictions are based on the premise that the peers have the same physics engine. The game must be built so that the peers share the same objects with the same attributes. Thus, they can exchange messages to update statuses if predictions fail or are inaccurate.

According to the platform's API, one of the peers is responsible for controlling the exchange of messages between all the peers that make up a game match, especially message forwardings and broadcasts when the game consists of more than 2 pairs. We call that “host”.

Also, the multiplayer engine automatically adds 80ms of artificial delay between pairs. This value is much less than the expected response time for a human being and does not hinder the simplest games. It uses these 80 ms to create a buffer for synchronizing the actions between peers.

# Application description

The Pong application consists of two screens, which load all the necessary operations in the simplest game mode. On the first screen, the player provides a username or alias. With this, the game can connect to the signaling server provided by Scirra to then find other peers.

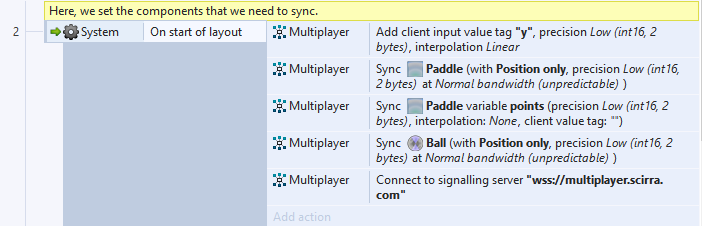
The game layout handles all signaling and connection tasks. The variables GAME\_NAME, INSTANCE\_NAME, and ROOM\_NAME allow players to connect using Scirra's WebRTC signaling server.

To run the game, just put them on a web server and open it in two different web browsers. Alternatively, you can open the game on <http://issonaocainaprova.com.br/games/Pong/>. Enjoy

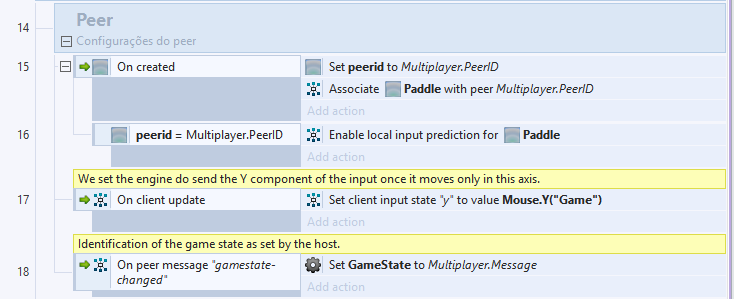
# Compliance with requirements

## The control should have fast response, independent of network latency, using client-side prediction.

Peers can exchange messages every 2 frames. The game engine uses messages to exchange commands and/or positions transparently to the developer through Object Syncing. In the figure below, we configure the synchronization command for component Y, as the paddle moves only on that axis. In addition, it is possible to observe the synchronization of points of the match, and the position of the ball.



Here, we set the state of the customer's command in terms of the Y axis of the paddle.



To give consistency to the game, one of the peers, the host, becomes the reference for predictions and adjustments. When peer A moves his paddle, the game sends the movement command to the host. Using the same physical engine, the host moves the paddle of peer A according to the characteristics of the command sent. After making the move, the host uses the messages to check the position of the paddle of peer A.

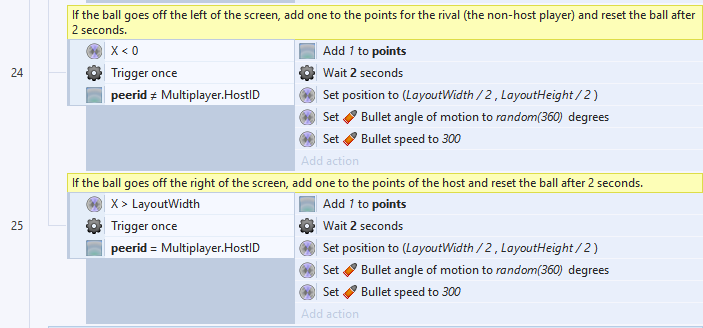
If there are differences between the position of the paddles of the two pairs, the host performs the adjustment. Given that the two pairs share the physical engine, errors are expected to be minimal or even non-existent. Also, the pairs use the 80 ms buffer so that adjustments can be made smoothly, when necessary.

## The game should also use client-side prediction to predict the ball position based on the network latency, allowing the player to be able to correctly hit the ball even under moderate network latency.

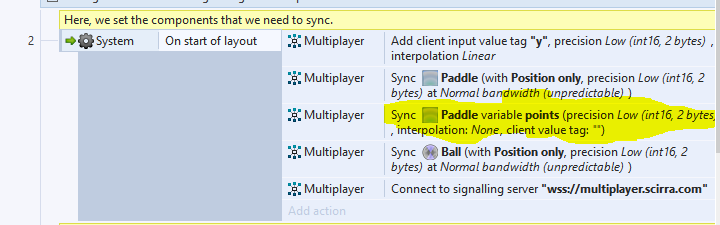
The position of the ball does not depend on players' commands. Therefore, the Construct engine recommends that the host become the reference for this movement. Thus, each peer performs the mechanics normally according to the built-in physics and the initial command given by the host. Using the messages, the host sends a command with the current position of the ball. The peer compares his predictions of

## The game state (ball position, score, and adversary paddle position) should not go out of sync between players.

The construct engine performs automatic state synchronization of game objects using one of the peers, the host, as a reference. The figure below shows the match points update. When the ball exceeds the left limit of the layout, peer adds up to one point. On the contrary, when the ball exceeds the right limit of the layout, the host adds a point.



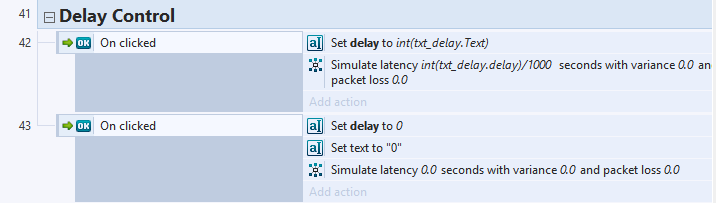
Object synchronization leverages object synchronization as in line 2.



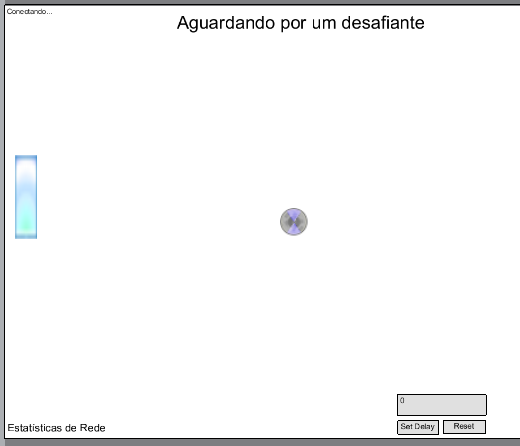
## The game needs to offer a way to simulate variable latency and should be demonstrated with latency that varies between 150 and 250 milliseconds randomly during the match.

We added a feature, enabled only on the host, to test latency in the game. According to the Construct 2 engine, the informed value is added to the real latency of the network. For this, we add a button and a text field.

Initially, the two elements are disabled and hidden. When the two pairs are connected, the elements appear only on the host screen. The value entered in the text field is added to the network latency.



The simulated delay is displayed on the bottom of the screen. The tester needs to type the delay in the box and press “set delay”. When testing ends, one needs to press “Reset”.



## Source code must be provided.

The game and source code are attached to the email.

Pong-game.zip → game

Pong-src.zip → source code

# Sources

Official Construct 2 Manual → <https://www.scirra.com/manual/174/multiplayer>

MULTIPLAYER TUTORIAL 1: CONCEPTS → [https://www.construct.net / en / tutorials / multiplayer-tutorial-concepts-579](https://www.construct.net/en/tutorials/multiplayer-tutorial-concepts-579)

MULTIPLAYER TUTORIAL 3: PONG → <https://www.construct.net/en/tutorials/multiplayer-tutorial-pong-626>