KOKKU

Gameplay Network Engineer Test

Specifications:

I have met the requirements of this test implementing a game using Unreal Engine 4.25.1. The implementation consists solely of blueprints as there was no need for direct C++ coding for this project.

I have used the authoritative networking model with a listener server (no support for dedicated server). The implementation does not include any port-bypass algorithms such as NAT punchthrough or UPnP. Therefore, to play using WAN networking it will be necessary to port-forward the 7777 UDP port (default from Unreal Engine). To play locally one can use the default localhost redirect IP 127.0.0.1 to join a hosted server.

Requirements:

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

The implementation uses interpolation of “dummy/fake” actors to give the player a feeling of smooth controls independent of network latency. This dummy actor cannot interact with the ball and the “true” actor that is received by the server is displayed in green once debug mode is enabled. As the server player never has latency with himself, this interpolation is only done for the client player controlling the paddle on the left side of the screen.

The interpolation of the client paddle is done in the following manner:

- The client player presses Up or Down arrows to move the paddle.
- Automatically, the client “dummy/fake” paddle position is updated.
- The input is sent to the server, which validates the position of the paddle and updates the “real” paddle position.
- When the input is released, the now idle position of the “dummy/fake” paddle is sent to the server for synchronization.
- The server validates this position and interpolates the “real” paddle to match position with the client’s “dummy/fake” paddle.

- **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 client also uses a “dummy/fake” ball that cannot make goals or bounce on paddles but can bounce on map boundaries. This interpolation is done in the following manner:

- Smoothly move towards the last known “real” ball position received by the server.
- Adjust velocity to the last known “real” ball velocity received by the server.
- If the “real” ball position surpasses a distance threshold, teleport directly to the “real” ball.
- Path prediction to interpolate to the expected path of the “real” ball was ***NOT*** implemented.

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

The interpolations mentioned above fulfill this requirement. Furthermore, important RPCs such as updating the score for the clients were marked as “Reliable” making the server resend the updated score packet in the case of packet loss (requires ACK by client).

- **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.**

Pressing the “F” keyboard button enables latency mode, which simulates a random 150 ms to 250 ms latency. Furthermore, a debug mode was implemented that allows visualization of “real” actors received by the client from the server in green. This mode and can be toggled by pressing “D” keyboard button

- **Source code must be provided.**
- **The candidate should write a document explaining how they met the requirements.**