GPR 430
Conner Root
Cormac MacKinnon

Final Project Plan

Gameplay Description

You and other players control a cube that you roll around a grid. When your cube touches a cell in the grid, the cell changes to your color.

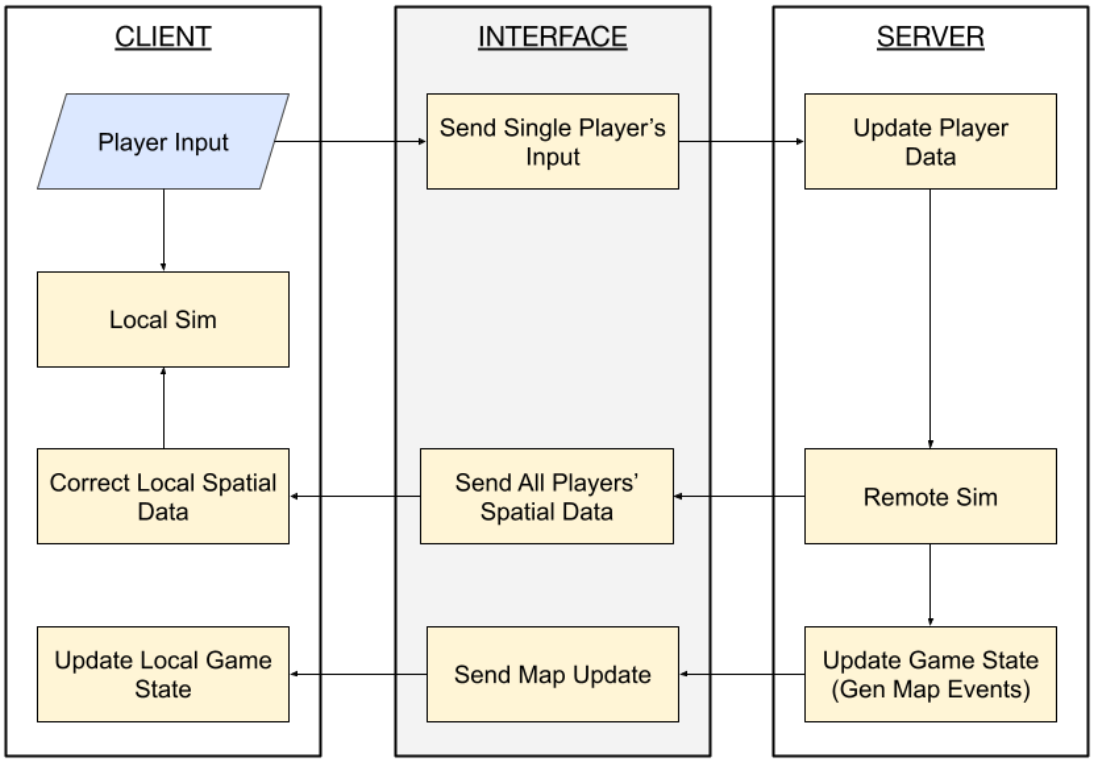
Networking Architecture

We use the new Unity MLAPI as a base and to handle joining servers (**async**)

Server-client interactions (Server-Authoritative)

- The controller for a client sends its inputs to the server (**remote agents/sim**)
- Server does its own simulations using the inputs it's sent
    - The server sends input and spatial data to the other clients (**sync**)
- Clients do their own simulations using dead-reckoning
    - Based on the input it was given and spatial data
    - When a client receives a position/rotation from server it overwrites the current one it has for that client (including self) (**remote correction**)

The diagram is slightly out of date but encompasses most of it



Packing the Data

All Packets have **`4 bits`** for the message type packed in first

Agent Data

Will send **`4 bits`** for the 'NetworkObjectID' (something unity assigns an object) to denote the client the information is for.

- This gives us space for 16 players, but we cap it at 4

Input

Will just send **`4 bits`** for the input flags (WASD)

- This means a client can do the entire simulation themselves with no error introduced by compression (only error is if you lose packets)
- This also means we don't *need* physics data, because its derived from the inputs
    - It would likely be necessary for other use cases, but the cubes don't move fast enough to require tracking velocity or anything

Total data for an input packet: `12 bits` *(4 + 4 + 4)*

Spatial Data

Will compress the vector3 position and rotation (euler angles) into **`39 bits`** (`13 bits` for each component)

- In our testing, it was only ever off by <0.002 for both position and rotation
    - Not really noticeable in our case

Total data for a spatial packet: `86 bits` *(4 + 4 + 3*13 + 3*13)*

World State

Map Event

Will pack the index of the cell changed using **`8 bits`**

- We have a 16x16 grid, so it fits perfectly (only 256 cells)

Will then pack the player id using **`4 bits`** that claimed the cell

Total data for a map event packet: `12 bits` *(8 + 4)*

Compression

We compress everything using the following quantization function (the algorithm has been stripped slightly to make it more general):

```
// compresses val to a value that can fit w/in the
// provided number of bits
private ulong CompressValue(float val, int bits)
{
```

```
    int maxBitVal = (2 << (bits - 1)) - 1; // get max value for bits
    float serial = (val - valMin);         // serialize to 0->total range
    float ratio = serial / totRange;       // divide by total range

    // convert to integer in 0->maxBitVal range
    ulong compressed = (ulong)Mathf.RoundToInt(ratio * (float)maxVal);
    return compressed;
}
```