|  |  |
| --- | --- |
|  | Moving Picture, Audio and Data Coding by Artificial Intelligencewww.mpai.community |

|  |  |
| --- | --- |
| **N408** | 2021/10/27 |
| **Source** | Requirements (GSA) |
| **Title** | MPAI-GSA Status report  |
| **Target** | MPAI Members |

# Project status

The month was devoted to continuing the development of the Pong (*Figure 1*) prototype based on the MPAI-SPG architecture.



Figure - Pong game

It started with an off-line version that was extended and brought online.

To comply with the specifications of MPAI-SPG (*Figure 2*), the type of online architecture chosen was "client/authoritative server".

Two programs were developed, one representing the game server and the other representing the clients. At runtime there will be two different instances (process C – C1 & C2) of clients both connected to the game server (process S). The clients are interfaced to digital controllers: two directions and one fire button.



Figure - MPAI-SPG Architecture

Legenda:

Green: new improvements

Red: TODO

Black: general status of the project

Both clients and server use Unity 3D as core game engine.

Features implemented so far:

* **Client**:
	+ Process C can send controller data to Process S
	+ Defined a CSV template to make the client log file
	+ Process C can obtain the ownership of one paddle sending an explicit request to the Process S.
	+ Process C can send notifies to Process S through the so-called “RPCs” (Remote Procedure Calls), to specify some actions or some sort of communication
	+ Process C implements Client-side Prediction related to his paddle.
	+ Process C implements interpolation (paddle position fixed with Lerp function)
	+ Process C implements server reconciliation (tick number for each data request)
* **Server**:
	+ Data exchange explanation between Game State Engine and game engines (Physics, Rule and Behavior)
	+ Defined a CSV template to make the server log file (game messages and game states)
	+ Game Server use the Photon architecture as space to instantiate each game
	+ Process S is a service program, running in “Batch/Headless mode” as Unity instance
	+ Process S can receive data from process C and acknowledge receipt of this data
	+ Process S instantiate both paddles and ball, then send (through RPCs) their ID to the clients so that they have a reference to those object
	+ Process S can handle the data (CD) sent by the process C in order to update the GS
	+ Process S is able to manage the physics both of the ball and the client paddles, sending the resulting data (GS) to both clients.
	+ Process S is able to synch both ball and client paddles.
	+ Better management of lag compensation
	+ Process S has
		- Game State Engine (GSE)
		- Physics Engine (PE)
		- Behaviour Engine (BE)
		- Rules Engine (RE)
	+ Game State Engine can send data to other three engines (PE/BE/RE)
	+ The three engines process the data correctly and send data to GSE
* **AI:**
* ML Agents developed, to simulate games in order to train the Neural Networks, using two techniques:
	+ Imitation Learning
	+ Reinforcement Learning
* Imitation learning Idea:
	+ Build a Demo with human input (Teacher)
	+ Define config.yaml with hyperparameters specifics (GAIL = GAN like learning)
	+ Start learning. Agent acts as student and learns how to behave as similar as his Teacher. Time required is significantly shorter than other methods (in terms of minutes not hours)
	+ NN produced will be the brain of our Client in Pong Game Online
* Integration of “ml-agents” framework inside Pong online version. This way will be possible to move paddle in an automatic way using inference
* Game simulation tests produce coherent data (log file) usable in any Neural Network
* Reinforcement Learning Idea:
	+ A set of observed data is chosen (player position, opine position, ball velocity…)
	+ Using its NN under development and observed data as input, the player executes an action and gets a reward. The reward is positive if it is correct in order to win the game, negative if is wrong.
* After multiple matches executed, paddle will be able to hit the ball
* **PLAN:**
* Learning optimization (both Imitation and reinforcement learning) in terms of:
	+ Physics accuracy
	+ Hyperparameters selection
* Introduction of a raycasting system in order to improve Reinforcement Learning:
	+ When the ball collides with a paddle, a series of rays is drawn on the game field representing ball trajectory ending on the opponent’s side of the field. This way the opponent will know where the ball will go and will ben able to execute the correct action
* Offline training of agents:
	+ Agents are going to be trained on a pong offline version creating two types of Neural Networks (one made using Reinforcement Learning and the other one from Imitation Learning) the will be used to make inference for automatic movement on online version and create log files