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

|  |
| --- |
| **Public Document** |
| N62 | 2020/11/18 |
| Source | MPAI-1 |
| Title | Draft MPAI Application Note #5 Rev. 1 - MPAI-SPG – Server-based Predictive Multiplayer Gaming |
| Target | MPAI Members |

### MPAI Application Note #5

### Server-based Predictive Multiplayer Gaming (MPAI-SPG)

**Proponents**: Marco Mazzaglia, Andrea Basso (Synesthesia)

**Description**: Gamers spend ludicrous amounts of cash on the latest GPUs and displays to get more FPS (frames per second). Indeed, a recent study [1] shows that higher frame rates have a noticeable impact on how well the gamer will play, improving a player’s kill-to-death ratio by as much as 90%.

In online gaming, there are two basic approaches.

1. Traditional online gaming: the server receives a sequence of data from the client(s) and sends an input-dependent sequence of data to the client(s) which use the data to create appropriate video frames.
2. Cloud gaming: the server receives a sequence of data from the client(s) and sends an input-dependent sequence of video frames to the client(s).

In a cloud gaming scenario, all clients run in the cloud on virtual machines.



*Figure 1 – Online and cloud gaming models*

In case the connection has temporary high latency or packet loss (in the following called network disruption) two strategies may be used to make up for missing information

1. Client-side prediction when information from the client does not reach the server or from the server does not reach the client
2. Server-side prediction when information from the client does not reach the server

In a game a finite state Game machine calculates the logic of the game from the inputs received from the game controller. The client reacts to such user input before the server has acknowledged the input and updated its Game state. If an updated Game state from the server is missing, the client predicts the Game state locally and produces a video frame that is potentially wrong. When the right information from the server reaches the client, the client Game state is reconciliated with the server Game state.

For example, in a first-person shooter game where a fast pace shooting takes place, player A aims their weapon at the position that player B held some milliseconds in the past. Therefore, when Player A fires, player B is long gone. In order to remediate this, when the server gets the information regarding the shooting, which is precise because it carries timestamps, it knows exactly where player A’s the weapon was aiming at and the past position of player B. The server thus processes the shot at that past time, reconciles all the states and updates the clients. In spite of the belated reconciliation, however. the situation is not satisfactory because player B was shot in the past and, in the few milliseconds of differ­ence, player B may have taken cover.

**Comments**: AI and ML can provide more efficient solutions than available today to compensate network disruption and improve user experience to both traditional online gaming and cloud-based gaming.

An AI machine can collect data from multiple clients, perform a much better prediction of each move of each participant and perform sophisticated reconcil­iations. Information from the game engine – inputs from clients and reconciliation info – can be used in the video encoding process to improve the encoding (i.e. motion estimation), thus making possible encoding at higher frame rates for a high-quality gaming exper­ience even on low performing hardware.

An MPAI standard can accelerate adoption of AI technologies to improve server performance by creating a horizontal competitive market of AI components.

**Examples**

Here are two examples of known game genres that illustrate how MPAI standards can feasibility and user experience.

*Example 1: Racing games*

During an online racing game, players can see lagging reactions to their moves and an overall low-quality presentation because of network disruption. Usually, the information on the screen predicted by a client is wrong if the online client information cannot reach the server and the clients involved in the online game on time. In a car racing game, the player may see at time t0 a vehicle going straight to the wall when it is reaching a curve. At time t1, after some seconds, the same vehicle is “teleported” to the actual position yielding an awful player experience.

AI can mitigate this issue, offering a better game experience to players. Data from the different online games are collected and used to predict a meaningful path or the correct behaviour in the time information does not reach the clients.

*Example 2: A zombie game*

In some traditional online video games, specific information is displayed differently on different clients because it is too onerous to compute all the outcomes of players’ actions in a physically consistent way. An example is provided by a zombie game: the result of killing hordes of zombies in each client is visually different from client to client.

A server-based predictive input can support the online architecture and enable it to provide an equal outcome on all clients. In a massive multiplayer hack&slash game, the result of the different combats among players yields the same live visual online experience to each player.

**Requirements:**

1. Standard representation of main parameters of a Game state
2. Standard representation of input signals from clients with a common set of parameters (e.g. position, speed etc.) e.g. for metrics
3. Standard interface to video encoder (e.g. to MPAI-EVC), for cloud gaming

**Object of standard**:

|  |  |
| --- | --- |
| All parameters time stamped1. Main parameters of Game state
2. Game messages
3. Data types
	1. Digital inputs (input ID, 1/0)
	2. Analogue inputs (input ID, value)
	3. Motion inputs (input ID, value)
4. Predicted game messages, e.g.
	1. New speed, position of bullet
	2. Recoil on the shooter
	3. Associated sound effect
	4. Effect on the scene
	5. Effect on the target
5. Predictions resulting from data inputs: e.g. intermediate format for digital and analogue outputs (input ID, 1/0)
6. Interface with video encoder, in par­ticular MPAI-EVC
7. Interfaces to the delivery schemes adopted by the market
 |  |

**Benefits:**

1. Better quality for the gameplay experience.
2. Improved and detailed visual outcome for all players.
3. Possibility to implement more complex and richer games.
4. Possibility to use acquired data & players’ behaviours for analysis to improve future works.

**Bottlenecks:**

1. Training the neural network with simulated games need further analysis (maybe follow the same approach used in music synthesis, i.e. quantization)
2. Network congestion is still an issue; we can mitigate it, not exclude the problem at all.

**Social aspects**:

The improvement of the quality of the overall experience can benefit all the events that involve e-Sports or equivalent environments and consolidate the technology for the market.

The potential of this use case can influence common online applications that orchestrate different inputs from different users.

**Success criteria**:

Short term: improved UX on a large set of current games due to a more performing AI based architecture.

Long term: MPAI-SPG becomes a standard used in video game engines like Unity 3D, Unreal Engine and the internal Software Development Kit (SDKs) developed by main game device manufacturers (Sony, Microsoft and Nintendo), cloud game based services (Amazon Luna, Google Stadia) and eSports platforms (FaceIt).

**References**

[1] <https://www.nvidia.com/en-us/geforce/news/what-is-fps-and-how-it-helps-you-win-games/>