VISAB - a visualization tool for CBR agent behavior in games

Moritz Fröhlich[2], Leon Römer[2], Vanessa Schriefer[2], Tim Kunold[2], Marcel Gaal[2], Pascal Reuss[1 2], and Klaus-Dieter Althoff[1 2]

[1] German Research Center for Artificial Intelligence
Kaiserslautern, Germany
`http://www.dfki.de`

[2] Institute of Computer Science, Intelligent Information Systems Lab
University of Hildesheim, Hildesheim, Germany
`http://www.uni-hildesheim.de`

Keywords: Case-based Reasoning, Software Agents, Computer Games, Visualization

1 The VISAB tool

VISAB (**VIS**ualizing **A**gent **B**ehavior) is an open-source program that is capable of receiving, analyzing and visualizing game and agent specific data of video games.[1 2] The first version was developed last year [1]. The new version of VISAB provides a generalized communication interface in form of a HTTP-Protocol based Web API. This API acts as a server, to which games communicate their data. To achieve scalability, incoming data is delegated in an event-based manner to the processing modules. Unity games [5] may use the VISABConnector library which provides a C# interface to the API.

VISABs HTTP-API is theoretically capable of receiving requests from K different machines each running m_{kn} instances of one of the N games supported by VISAB. For each of these visualizer instances views may be opened concurrently.

To provide a meaningful overview of the transmission sessions handled by the HTTP-API, a session overview displays general as well as session-specific information in separated UI elements. Each of these session elements provides the functionality to cancel it directly from VISAB side or to open the visualizers that are dynamically filled with live data from the active session.

VISABs core feature is directly accessible from the home view through a file-explorer-style control which lists the available game data files. Upon button click, the game-specific visualizer views of the selected file are loaded. VISAB currently supports "CBRShooter" [4][3] which is a real-time game and "Settlers of Catan" [2] as a round-based game..

Despite their different categories regarding their previously elaborated nature of communication frequency, both games feature the same set of visualizers to provide information to the user. We believe that the views for these games may

[1] Source code: `https://github.com/VISAB-ORG/VISAB`
[2] Video: `https://www.youtube.com/watch?v=znG2ZtcAfqA`

serve as an extendable blueprint for other games as well. Each of the game-specific views relies on the same data source, either from a saved file or from an active transmission session.

Having multiple views on the same data ensures sophisticated interpretation, each of which can highlight information differently. To achieve a highly useful illustration of the underlying data, VISAB currently contains three different visualizers with their respective that are adapted for both games: a metadata view, a statistics view, and a replay view. The first view type shows overall metadata related to the currently processed file or transmission session such as a list of players or the available items. The second view type displays important information both aggregated over the entire file as well as for game-specific logical steps. And the last view makes use of real asset visuals of the respective games and allows the user to re-watch the game flow and help to understand decisions made by the participating agents.

A dedicated workspace serves as the central point of persistence for data that is used and produced by VISAB. This includes the game data files that are generated by the transmission sessions of games sending their information to VISAB, a settings file as well as the applications log files.

Furthermore, VISAB also allows users to define their own settings. These settings determine the UI theme or modify allowed games for initiating transmission sessions with the HTTP-API. VISAB is designed to be easily extensible for new games according to a developer's needs. By making use of a reflection-based approach that creates instances of game-specific implementations for the relevant types at runtime based on a JSON config file. This assures that existing code does not need to be modified when the necessity of another visualizer arises. After the implementation of a new visualizer is accomplished, it only has to be added to the above-mentioned config file to be integrated for visualization.

References

1. Bartels, J.J., Viefhaus, S., Yasrebi-Soppa, P., Reuß, P., Althoff, K.D.: Visualizing the behaviour of cbr agents in a fps scenario. In: Trabold, D., Welke, P., Piatkowski, N. (eds.) Lernen, Wissen, Daten, Analysen. GI-Workshop-Tage "Lernen, Wissen, Daten, Analysen" (LWDA-2020), September 9-11, Online. pp. 130–141. CEUR (2020)
2. CatanGmbH: Settlers of catan overview (2021), `https://www.catan.com/game/catan`
3. Kolbe, M., Reuss, P., Schoenborn, J., Althoff, K.D.: Conceptualization and implementation of a reinforcement learning approach using a case-based reasoning agent in a fps scenario. In: CEUR LWDA 2019 Workshop Proceedings 2454. CEUR (2019)
4. Reuss, P., Hillmann, J., Viefhaus, S., Althoff, K.D.: Case-based action planning in a first person scenario game. In: Gemulla, R., Ponzetto, S., Bizer, C., Keuper, M., Stuckenschmidt, H. (eds.) LWDA 2018 - Lernen, Wissen, Daten, Analysen - Workshop Proceedings. GI-Workshop-Tage "Lernen, Wissen, Daten, Analysen" (LWDA-2018), August 22-24, Mannheim, Germany. Universität Mannheim (8 2018)
5. UnityTechnologies: Unity 3d overview (2021), `https://unity.com`