DOI: 10.15514/ISPRAS-2026-38(6)-34



AIGame: Architecture of the Platform for Studying AI-Agent Interaction and Immersion

E.P. Egorova, ORCID 0009-0009-5779-1791 <LizaKkiva@yandex.ru>

National Research University Higher School of Economics (HSE University), 20, Myasnitskaya st., Moscow, 101000, Russia.

Abstract. This paper addresses the problem of designing a specialized software platform for empirical research in the field of human-AI interaction. The primary objective is to study the interrelationship between the complexity of the game world model and the level of immersion experienced by the player when interacting with intelligent agents. A modular, multi-tiered microservice architecture is proposed, providing the necessary flexibility for conducting variable experiments. Key architectural characteristics include: isolated execution of agent code in Docker containers, clear separation of the Game Loop and the development/testing cycle, and the use of specialized protocols (gRPC, WebSocket) for different types of interaction. The proposed architecture allows researchers to flexibly modify both agent behavior and game world rules while ensuring deterministic data collection for subsequent analysis.

Keywords: immersion; sandbox; agent.

For citation: Egorova E.P. AIGame: Architecture of the Platform for Studying AI-Agent Interaction and Immersion. Trudy ISP RAN/Proc. ISP RAS, 2026, vol. 38, issue 6, part 2, pp. 227-238. DOI: 10.15514/ISPRAS-2026-38(6)-34.

Acknowledgements. The research was conducted with the support of the Faculty of Computer Science at HSE University in 2026.

ИИгра: Архитектура платформы для исследования взаимодействия ИИ-агентов в иммерсивных средах

Е.П. Егорова, ORCID: 0009-0009-5779-1791 <LizaKkiva@yandex.ru>

Национальный исследовательский университет "Высшая школа экономики" (НИУ ВШЭ), Россия, 101000, Москва, Мясницкая ул., д. 20.

Аннотация. Рассматривается задача проектирования специализированной программной платформы для проведения эмпирических исследований в области взаимодействия человека и искусственного интеллекта. Основная цель работы — изучение взаимосвязи между сложностью модели игрового мира и уровнем погружения игрока при взаимодействии с интеллектуальными агентами. Предлагается модульная многоуровневая микросервисная архитектура, обеспечивающая необходимую гибкость для проведения разнообразных экспериментов. Ключевые архитектурные характеристики включают: изолированное выполнение кода агентов в контейнерах Docker, четкое разделение игрового цикла (Game Loop) и цикла разработки/тестирования, а также использование специализированных протоколов (gRPC, WebSocket) для различных типов взаимодействия. Предложенная архитектура позволяет исследователям гибко изменять как поведение агентов, так и правила игрового мира, обеспечивая при этом детерминированный сбор данных для последующего анализа.

Ключевые слова: иммерсия; песочница; агент.

Для цитирования: Егорова Е.П. ИИгра: Архитектура платформы для исследования взаимодействия ИИ-агентов в иммерсивных средах. Труды ИСП РАН, 2026, том 38, вып. 6, часть 2, стр. 227–238 (на английском языке). DOI: 10.15514/ISPRAS-2026-38(6)-34.

Благодарности. Исследование подготовлено при поддержке факультета компьютерных наук НИУ ВШЭ в 2026 году.

1. Introduction

The study of immersion, as the degree of user absorption in a virtual environment, is a key challenge in human-computer interaction (HCI), game design, and artificial intelligence (AI). It is worth noting a fundamental difference between a human agent and an AI agent. For us this difference lies in how the agent perceives its environment: a human agent perceives it through its sensory organs and feelings, while an AI agent perceives it through sensors and perception systems that are modeled for it. Research into immersion during interaction with autonomous intelligent agents, whose behavior is becoming increasingly complex and human-like, is gaining particular relevance. However, as agent complexity grows, the "uncanny valley" problem intensifies, wherein minor deviations from expected behavior cause observer discomfort and disrupt the sense of presence. It is important to note that the broader field of software performance evaluation has long benefited from established benchmarking methodologies and frameworks, as comprehensively discussed in [1]. However, AIGame differs fundamentally from traditional benchmarking systems: while conventional benchmarks focus on measuring raw computational performance, our platform is designed to study human-AI interaction within a gaming context, where the independent variable is the level of immersion experienced by the player. This shift from purely technical metrics to human-centric evaluation defines the core distinction of our approach.

Specialized tools are required for the systematic study of factors influencing immersion. Existing game engines and AI training platforms are often either too rigid and do not allow flexible modification of fundamental world rules, or conversely, are low-level frameworks requiring significant effort to create a full-fledged interactive environment.

The specific scientific problem addressed in this work is the absence of a software platform that simultaneously provides: (a) a deterministic, modifiable game world model (physics and rules), (b) secure, language-agnostic execution of untrusted AI agent code, and (c) integrated tooling for

reproducible experimentation and data collection. The objective of this work is to design and evaluate such an architecture.

The input to the platform is a researcher-defined configuration (world rules, agent code, experimental protocol). The constraints include: real-time interaction with a human player (deterministic updates at ≥20 Hz), secure execution of untrusted code, and complete logging for reproducibility. The success criteria for evaluating the proposed architecture are:

- Determinism: Identical initial conditions and action sequences must produce identical final world states (quantifiable as zero standard deviation across repeated runs).
- Latency: Agent decision round-trip time must be below 100 ms to maintain real-time interaction.
- Security: Execution of untrusted agent code must not compromise the host system (demonstrable via resource isolation).
- Customization: Modification of world rules (e.g., physics parameters) must be achievable without recompiling the game server or agent code.

Thus, a scientific challenge arises: to design a software platform architecture specifically for studying the interrelationship between game world models, intelligent agent behavior, and the player's immersive experience. Unlike conventional benchmarking systems [1] that focus on raw computational performance, our platform is designed to quantitatively measure and analyze human-agent interaction within a gaming context. In doing so, we adhere to the 4R principles – Repeatability, Reproducibility, Replicability, and Reusability – as outlined in the framework for credible scientific research [2]. This requires not only deterministic simulation and secure code execution but also systematic data collection and experiment lifecycle management to ensure that findings on immersion can be independently validated and extended.

2. Definitions

"AIGame" is a project developed within our university's project group; for users, it is a platform that can be used for research and educational purposes, as well as a gaming platform.

Immersion – the degree of cognitive absorption of the player in the process of interaction with the virtual environment. Within the context of the platform, it is considered a measurable characteristic dependent on the consistency of world rules, agent behavior, and feedback quality.

Intelligent agent – an autonomous software entity that perceives the state of the game world via an API, makes decisions based on an embedded model, and acts by sending commands to the simulation. Serves as the primary object of research in the study of immersion.

Sandbox – a dynamic, configurable environment for conducting experiments within which the researcher can arbitrarily modify game world parameters, object interaction rules, and intelligent agent characteristics. Unlike static simulations, the “AIGame” platform sandbox allows flexible variation of experimental conditions while maintaining isolated code execution (containerization) for security and reproducibility of results.

Game world model – a formalized system of rules, physics, and constraints existing as an authoritative state on the game world. Serves as an independent variable in the experiment: its modification enables the study of environmental context influence on the perception of agent behavior.

Determinism of Simulation – a property of the game loop whereby identical initial conditions and identical sequences of agent actions always lead to identical results. Ensures experimental reproducibility and the ability to verify scientific hypotheses.

Reproducibility – the ability to recompute the same experimental results by re-running the platform with identical configuration, agent code, and random seeds. Reproducibility is a direct consequence of determinism and is a prerequisite for validating scientific findings in human-AI interaction studies.

Microservice architecture – an approach in which the platform is built as a set of independent, loosely coupled services with clear API contracts. Ensures modularity, the ability to replace components in isolation, and horizontal scaling of computational resources.

3. Literature Review

Considering the “AIGame” project, several approaches to constructing its architecture are apparent. The most modern approach for real-time multiplayer games is the Adaptive Client-Server Architecture, described in detail in [3]. This model assumes a server responsible for game world simulation and clients that only handle rendering and sending their state within the world. Such an approach ensures security, prevents cheating, and synchronizes the game state for all participants. Practical aspects of implementing such systems, including network latency handling and state interpolation, are detailed in [4].

However, AIGame is not merely a game being developed. We aim to combine gameplay with a foundation for creating and studying intelligent agents. Therefore, while building upon reliable networking practices, we rely on more AI-oriented design approaches examined in fundamental works on artificial intelligence in games [5-6].

3.1 Review of Specialized “Sandbox” Environments

Having identified the intersection of problems our project addresses, let us discuss existing solutions whose tasks partially overlap with ours, including both pioneering sandboxes and modern academic research platforms.

Robocode [7] is one of the oldest and most popular sandboxes. Its main contribution is the popularization of competitive programming in a simple, visual, and strictly formalized environment. From an architectural perspective, Robocode implements the isolated execution environment pattern, running agent code in a Java sandbox within the main program. However, its environment is limited to a simplified 2D arena, and agents can only be written in JVM languages. Unity ML-Agents [8] represents a more recent and powerful framework. It provides an API for communication between a Unity engine scene and external Python code where training occurs. Its key advantage is photorealistic 3D environments. However, its architectural limitation is partial monolithicity inherited from the game engine. Replacing fundamental components, such as physics, requires significant effort. Furthermore, as demonstrated in our evaluation (Section 5), Unity ML-Agents does not guarantee deterministic simulation.

OpenAI Gym [9] is a widely adopted benchmark suite for reinforcement learning. It provides a simple API for agent-environment interaction but is not designed for 3D worlds or human-in-the-loop studies. Similarly, DeepMind Lab [10] offers a 3D environment based on Quake III Arena but is primarily focused on navigation and visual recognition tasks, not on modifiable world rules or immersion research.

Project Malmo [11], developed by Microsoft Research, embeds AI agents into the Minecraft world. It offers full flexibility in world modification through Minecraft mods. However, Minecraft's underlying simulation is non-deterministic, which severely limits experimental reproducibility. Additionally, security of untrusted agent code remains a concern, as agents run within the Minecraft process.

Habitat [12] and CARLA [13] are two prominent platforms for embodied AI research. Habitat focuses on indoor navigation and rearrangement tasks in realistic 3D scenes, while CARLA is specialized for autonomous driving simulation. Both provide high-quality 3D environments and Python APIs. However, both are designed for single-agent or limited multi-agent scenarios with specific task APIs, not for general game-world modification or studying human-player immersion. Neither platform prioritizes secure execution of user-provided arbitrary code.

It is against the backdrop of these limitations that modern scientific works emerge, attempting to solve problems of scalability and security at a deeper architectural level. Key challenges for multi-

agent systems are examined in [14], proposing approaches to ensuring interaction security and scalability that resonate with our use of containerization for complete agent isolation.

3.2 Analysis of Large Research Platforms and Modern Scientific Approaches

Moving to larger-scale projects, it is necessary to consider Unity ML-Agents. This framework essentially provides a powerful API for communication between a Unity engine scene and external Python code where training occurs. However, its architectural limitation is a degree of monolithic structure, inherited from the game engine. Replacing fundamental components, such as physics, requires significant effort and is not its standard use case.

It is against the backdrop of these limitations that modern scientific works emerge, attempting to solve problems of scalability, security, and generalization at a deeper architectural level.

Elsewhere in the world, a project is being created that can compete with SIMA [15] but has significant differences, as well as results described in several scientific papers accessible to us: In the articles [16] and [17] using the example of the game “Red Dead Redemption II”.

Thus, the analysis of the existing landscape – from simple “sandboxes” to the latest scientific papers – defines an area with room for new research. The scientific community requires a tool that combines the architectural flexibility of microservices, the security of OS-level containerization, deterministic simulation, and powerful visualization capabilities inherent in modern game engines. This is precisely the niche that the “AIGame” platform is intended to fill.

4. Models and Methods

4.1 Software and Systems Engineering Methods

The design of the “AIGame” platform was carried out using a set of software and systems engineering methods, the choice of which was determined by the research nature of the project and the requirements for flexibility, scalability, and reproducibility of experiments.

The work is based on the Domain-Driven Design (DDD) approach, which enabled the identification of key bounded contexts of the subject area: “Game World Simulation”, “Agent Code Editor”, “Data Collection and Storage” and “Researcher Interface”. The clear separation of contexts ensured loose coupling of components and the possibility of their independent evolution.

The following quality assurance methods were applied during design:

- Architectural Patterns – Authoritative Server, Sandboxed Execution Environment, Event-Driven Communication;
- Formal API Description via Protocol Buffers (.proto files) for gRPC interaction, ensuring strict contract typing and automatic client code generation;
- Containerization (Docker) as a method of computational process isolation, ensuring security of untrusted code execution and environment reproducibility.

At the system design level, the following were used:

- Tiered Decomposition (Tiered Architecture) – separation into client, server, computational, and storage tiers with clear areas of responsibility;
- Interaction Protocol Design – separation of communication channels by traffic nature: WebSocket for streaming world state broadcasts, gRPC for transactional "agent-simulation" requests;
- Experiment Lifecycle Management – formalization of processes for match creation, agent container startup, log collection, and simulation termination.

The following instrumental methods were selected as core implementation tools:

- Unreal Engine – as a platform for game world visualization and simulation with native WebSocket support;
- Docker Engine API – for programmatic management of agent container lifecycles;
- Server storage – for storage of structured metadata and binary experimental artifacts.

The application of these methods enabled the design of an architecture that is not a rigid monolith but a methodological foundation – a set of engineering solutions ensuring reproducibility, security, and flexibility in conducting empirical research in the field of human-AI interaction.

4.2 Designed Architecture of the Research Platform

The architecture consists of several modules (see Fig.1 for details) and described below:

- Client Tier – Unified Researcher Interface. This is an integrated environment combining a code editor, debugging tools, and a simulation visualization window. Its task is to provide the researcher with a single-entry point for the complete work cycle: from writing agent code to analyzing the results of its interaction with the world;
- Backend Tier – Orchestration Layer. This tier acts as the “chief conductor” of the system. It does not handle resource-intensive computations but manages the lifecycle of experiments: receives requests from the client, authenticates users, manages projects, creates matches, and, most importantly, coordinates interaction among all other services;
- AI/ML Services – Computational Core. This is the most critical and high-load tier, responsible for executing intelligent agent code. Its architecture is designed for the secure and efficient execution of arbitrary, potentially unstable or hostile, user-provided code;
- Data Storage Tier. This tier ensures the persistence of all research artifacts, including experiment configurations, agent code, trained models, and match recordings. A key consideration in designing this tier is the file-oriented nature of Unreal Engine: the engine operates primarily with asset files (.uasset), maps, and configuration files. This creates specific data storage requirements – large binary files must be stored and retrieved efficiently, while experimental metadata needs to remain accessible for querying and analysis;
- World Components tier. This tier defines and exposes the logical structure of the game world to the agent development environment. It provides a description of all interactable entities, their properties, and behavior rules – the “contract” between the world and the agent. This description is delivered to the editor as a machine-readable specification (e.g., an API schema), enabling context-aware code completion, validation, and documentation for the researcher. The agent never sees the internal implementation; it only interacts through this stable, exposed interface.

The key feature of the editor is its deep integration with backend systems, ensuring a seamless “development-testing” cycle (see also Fig. 2). The list of functions of the editor below:

- Core and Intelligent Functions: The foundation is the EditorController, managing UI component interaction. Code completion (CodeCompleter) and static analysis (Linter) functions receive data not from hardcoded rules but from a dynamically loaded API Specification.json file. This file is generated during the game server build stage, guaranteeing absolute consistency between the API available to the agent and the hints in the editor.

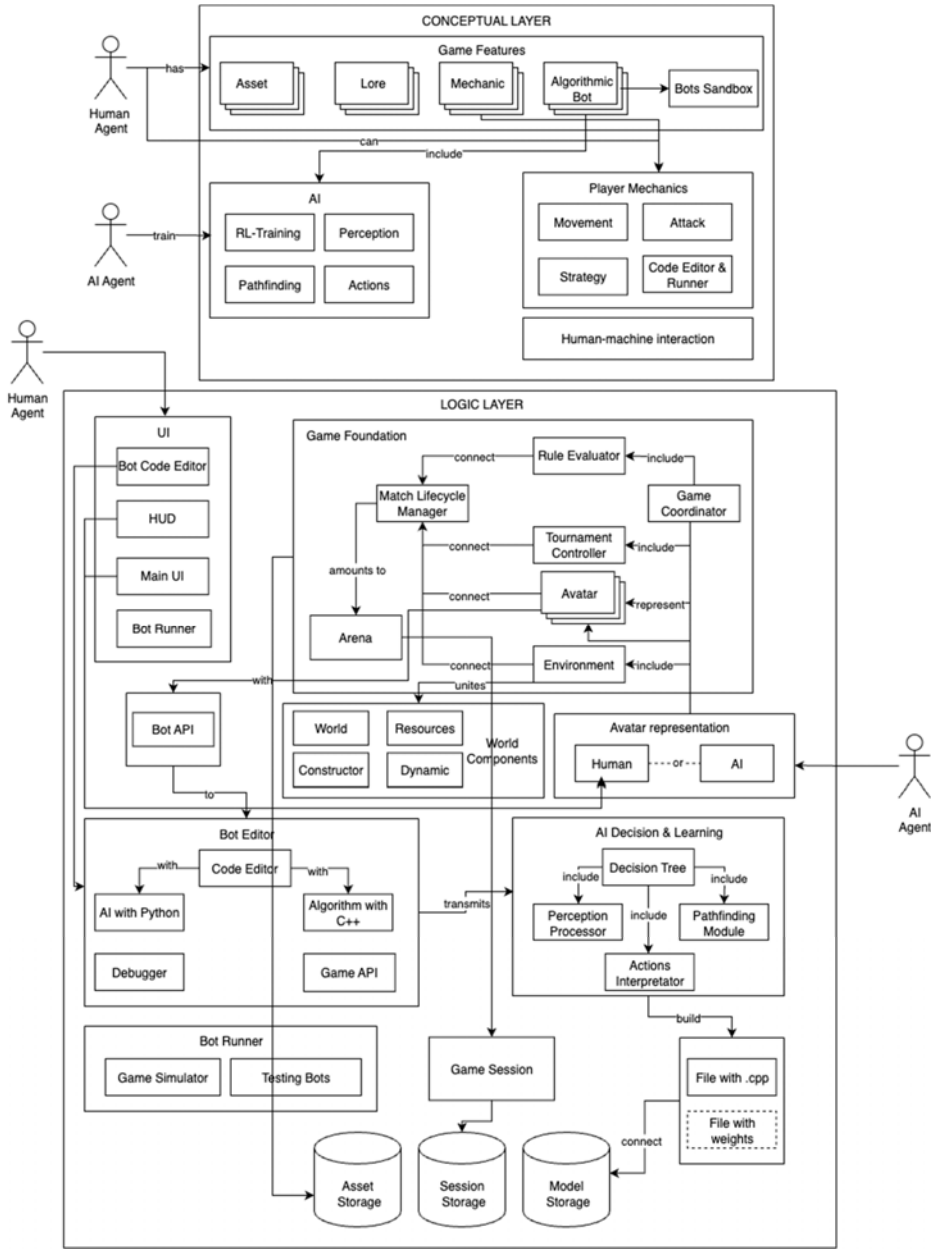


Fig. 1. Conceptual & logic Architecture Levels.

- Debugging Protocol: Instead of creating a proprietary protocol, the system uses the standard Debug Adapter Protocol (DAP). The DebugAdapterClient on the client side communicates with the DebugAdapterServer on the backend. This architectural decision makes the platform language-agnostic: to add support for a new agent programming language, it is sufficient to implement the corresponding adapter on the server without making any changes to the client editor code.
- Dual-Circuit Feedback System: When a test is launched, two independent WebSocket channels are activated:
  - World State Channel: Transmits data at high frequency for visualization in TestArenaView, ensuring smooth simulation display.
  - Logging Channel: Transmits all standard output streams (stdout/stderr) from the agent container in real-time, immediately displayed in ConsolePanel. This allows the researcher to see debug messages from their code at the very moment they are generated.

4.2.1 Detailed Architecture of the Editor Module: Researcher's Tool

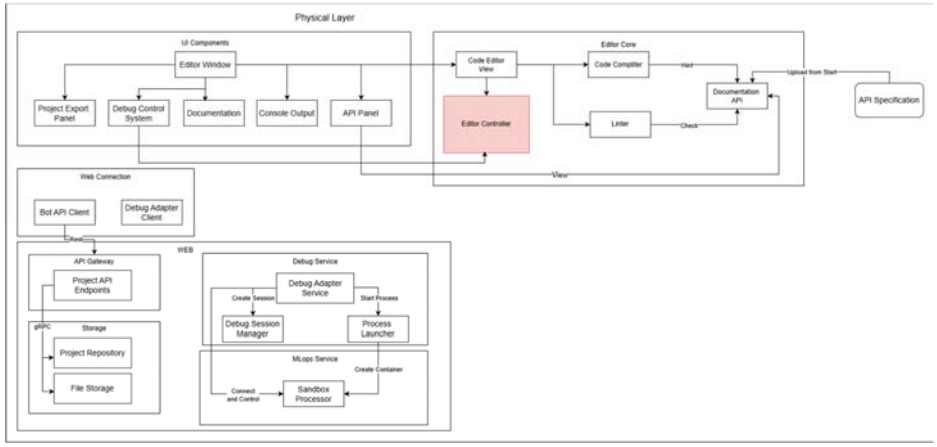


Fig. 2 Editor Physical layer.

4.2.2 Detailed Architecture of the Game Module: Deterministic Simulation

The game module is built on an authoritative server model, where World State, Rule Evaluator, and Game Entity on the server are the sole source of truth (Fig. 3).

Protocol Separation: The choice of protocols is determined by the nature of the transmitted data and the fact that Unreal Engine, which natively supports WebSocket, was chosen for implementation:

- WebSocket is used for “server-client” communication. It is an ideal protocol for the "one-to-many" pattern (server broadcasts state to all observers) and for data streaming where minimal losses are acceptable.
- gRPC is used for “game server – AI service” communication. It is a high-performance RPC framework operating over HTTP/2. It ensures strictly typed interaction (via .proto files) and is ideally suited for the transactional “request-response” pattern, critically important for the “request action – receive action” cycle, where latency and serialization/deserialization overhead (as in JSON) are unacceptable.

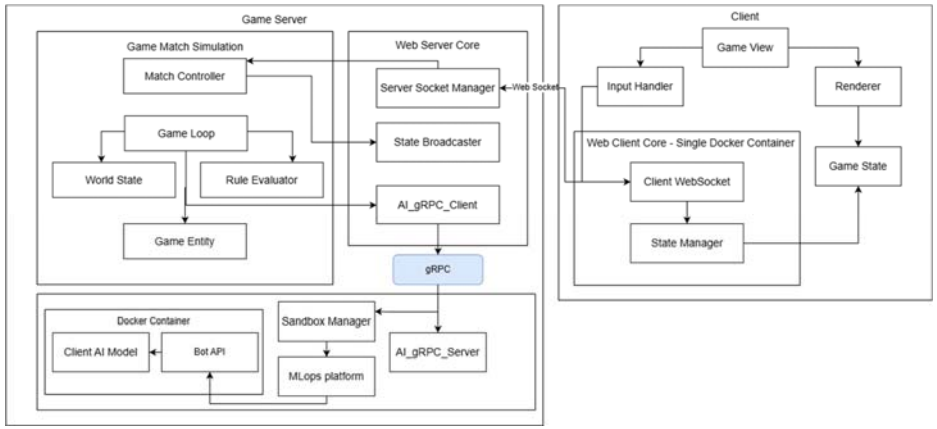


Fig. 3. Game System physical layer.

Deterministic Game Loop: GameLoop is the heart of the simulation. Each "tick" is an atomic, strictly sequential operation: request action from all agents via gRPC, apply these actions, calculate game states, update WorldState, and finally, broadcast the new state to observers.

5. Experimental Evaluation

The platform is being evaluated against four key criteria derived from our problem statement: simulation determinism, agent decision latency, secure code execution, and flexibility of world rule modification. The evaluation scenarios are built around the platform's core battle royale mode – a procedurally generated 3D world with four agents per match, consumable items, and weapon upgrades.

To cover different aspects of human-AI interaction and immersion, we have defined four scenarios that vary agent intelligence and the presence of a human player:

- S1 (Empty Agent) – measures baseline system overhead and world determinism;
- S2 (Navigating Agent) – tests pathfinding and item collection logic;
- S3 (Intelligent Battle Agent) – full combat AI with decision-making;
- S4 (Human-in-the-Loop) – a human player replaces one agent, testing real-time interaction in the editor.

Each scenario was run multiple times with fixed seeds (30 runs for S1–S3, 7 runs for S4), and preliminary results confirm that the architecture meets its core requirements: deterministic simulation, low latency, secure isolation, and rapid configuration changes. These scenarios, however, represent only a small subset of possible experimental designs. The platform is intentionally modular and configurable, allowing researchers to define many more scenarios – varying world rules, agent teams, interaction modalities, and immersion conditions – to systematically study human-agent interaction in gaming contexts.

A full-scale quantitative study, including systematic latency profiling, large-scale determinism validation, and biometric immersion measurements, is planned as future work.

6. Discussion & Conclusions

This work addresses the challenge of designing a specialized software platform architecture intended for deep and comprehensive research into the relationship between game world models, the behavior of intelligent agents, and the immersive experience perceived by the player. Unlike general-purpose

AI benchmarking systems, our platform is specifically oriented toward game-based environments and projects, with a primary focus on studying human-machine interaction and immersion in gaming contexts.

The practical implementation confirms the viability and soundness of the proposed architectural solutions, transitioning them from the conceptual plane into the realm of engineering artifacts: data and knowledge models, simulation models, and user interaction interfaces.

The scientific significance of this work lies in the proposed and practically validated architecture, which essentially constitutes a methodological foundation for conducting reproducible empirical experiments in the field of human-machine interaction – specifically, the interaction between a human player and an intelligent agent. The platform enables researchers to systematically vary both the complexity of the agent itself and the rules and physics of the world it inhabits, thereby objectively measuring the impact of these changes on the outcome of the interaction.

Despite the achieved results, it is necessary to outline a number of limitations inherent to both the proposed architecture and the current stage of project implementation.

Technical Limitations. The use of Unreal Engine as the simulation platform imposes certain hardware requirements on the server-side infrastructure. The full determinism of the game loop, stated as a key principle, is limited in practice by the use of a standard physics engine, which may introduce minor deviations when operating in a distributed network environment. Achieving absolute determinism requires additional engineering efforts, specifically the implementation of a custom fixed-point physics layer.

Methodological Limitations. The proposed architecture is focused on researching immersion within the context of interaction with one or several agents during a time-limited game session. Questions of long-term immersion, agent adaptation to changing player behavior over multiple sessions, as well as the social aspects of interaction between human players in the presence of agents, remain outside the scope of the current architectural model.

Validation Limitations. At this stage, architectural validation has primarily been conducted through its practical implementation and component testing. Full-scale empirical studies involving real users, which would enable a quantitative assessment of the impact of architectural decisions on the measured level of immersion, are a subject for future work.

The proposed architecture and the "AIGame" platform implemented on its basis are oriented toward the following usage scenarios:

- Academic Research in the fields of HCI, HMI (Human-Machine Interaction), and AI Alignment – studying the factors influencing human perception of artificial agent behavior.
- Educational Process – conducting laboratory assignments and course projects in disciplines such as "Artificial Intelligence in Games", "Software Engineering", and "Multi-Agent Systems".

References

[1]. Akinshin A. Pro .NET Benchmarking: The Art of Performance Measurement. Apress, 2019.
[2]. Hernández J. A., Colom M. Repeatability, Reproducibility, Replicability, Reusability (4R) in Journals' Policies and Software/Data Management in Scientific Publications: A Survey, Discussion, and Perspectives, 2023. DOI: 10.48550/arXiv.2312.11028.
[3]. Khan A. M., Arsov I., Preda M., Chabridon S., Beugnard A. Adaptable client-server architecture for mobile multiplayer games, in Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, ICST, Jun. 2023. DOI: 10.4108/icst.simutools2010.8704.
[4]. Glazer J., Madhav S. Multiplayer Game Programming. Addison-Wesley Longman, Incorporated, 2015.
[5]. Millington I. AI for Games. CRC Press, 2021.
[6]. Yannakakis G. N., Togelius J. Artificial Intelligence and Games. Springer, 2025.
[7]. Robocode Home. Available at: https://robocode.sourceforge.io/, accessed: 07.02.2026.

[8]. ML-Agents Overview, v3.0.0. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.ml-agents@3.0/manual/index.html (Accessed: Mar. 15, 2026).
[9]. Brockman G. et al., OpenAI Gym, 2016. DOI: 10.48550/arXiv.1606.01540.
[10]. Beattie C. et al. DeepMind Lab, 2016. DOI: 10.48550/arXiv.1612.03801.
[11]. Project Malmo, Microsoft Research. Available at: https://www.microsoft.com/en-us/research/project/project-malmo/, accessed: 15.03.2026.
[12]. Savva M. et al. Habitat: A Platform for Embodied AI Research, 2019. DOI: 10.48550/arXiv.1904.01201.
[13]. Dosovitskiy A. et al. CARLA: An Open Urban Driving Simulator, 2017. DOI: 10.48550/arXiv.1711.03938.
[14]. Du H., Gou F., Cai Y. Scalable Safe Multi-Agent Reinforcement Learning for Multi-Agent System, Jan. 2025. DOI: 10.48550/arXiv.2501.13727.
[15]. Team S., SIMA 2: An Agent that Plays, Reasons, and Learns With You in Virtual 3D Worlds, Google DeepMind. Available at: https://deepmind.google/blog/sima-2/, accessed: 07.02.2026.
[16]. Tan W. et al. Towards General Computer Control: A Multimodal Agent for Red Dead Redemption II as a Case Study, Mar. 2024. DOI: 10.48550/arXiv.2403.03186.
[17]. Tan W. et al. Cradle: Empowering Foundation Agents Towards General Computer Control, Mar. 2024. DOI: 10.48550/arXiv.2403.03186.

Информация об авторах / Information about authors

Елизавета Петровна ЕГОРОВА – бакалавр по направлению Программная инженерия. Сфера научных интересов: инженерные системы для анализа и разработки агентов внутри высокоиммерсивных сред, человеко-машинное взаимодействие, искусственный интеллект в компьютерных играх.

Elizaveta Petrovna EGOROVA – Bachelor’s degree in Software Engineering. Research interests: engineering systems for the analysis and development of agents within highly immersive environments, human-computer interaction, artificial intelligence in computer games.