Journal of Innovations in Business and Industry

Vol. 02, No. 02 (2024) 61-68, doi: 10.61552/JIBI.2024.02.001 - http://jibi.aspur.rs

GAME DEVELOPMENT TOOLS AND AI BASED ALGORITMS FOR CREATION OF GAMEWORLD

Hrvoje Puskaric[1]

Received 05.01.2023.
Revised 10.10.2023.
Accepted 12.11.2023.

Keywords:

Game development, game world, AI-based algorithms, procedural content generation, game engines, level design, conversational AI, game AI.

Original research

A B S T R A C T

Game development is a complex and time-consuming process that requires the use of specialized tools and technologies to create immersive and engaging game worlds. Game world design theory involves creating a virtual environment that is immersive, engaging, and meaningful for players. In recent years, the use of AI-based algorithms has become increasingly popular in game development, as they offer the ability to generate dynamic and varied game worlds that adapt to the player's actions. In this manuscript we will discuss the main issues connected with game development tools and AI based algoritms for creation of gameworld.

1. INTRODUCTION

Game world design theory involves creating a virtual environment that is immersive, engaging, and meaningful for players (Bormann & Greitemeyer 2015). The game world design is a critical component of a game, as it sets the tone, atmosphere, and overall experience for the player (Pagulayan et al. 2002).

Creating the game world requires multiple process which should be interconnected (Grizioti & Kynigos 2021).

One of the main principles of game world design theory is consistency. A game world should be consistent in its design, with elements that are logically and thematically connected (Rupp et al. 2010). This helps create a sense of believability and immersion for players. By creating a world that is internally consistent, designers can help players become fully immersed in the game world (Rapp 2017).

Another important principle of game world design theory is exploration. The game world should be designed in a way that encourages exploration and discovery (Flanagan et al., 2005). This can include hidden areas, secrets, and rewards that player can discover as they navigate the game world. By creating a sense of discovery, designers can help players feel more engaged and invested in the game world.

Interactivity is another key principle of game world design theory. The game world should be interactive, with elements that can be manipulated or interacted with by the player. This can include objects that can be picked up, doors that can be opened, or enemies that can be defeated. By creating a world that is interactive, designers can help players feel more engaged and invested in the game world (Reeves & Read, 2009).

In addition, game world design theory emphasizes the importance of challenge. The game world should provide challenges and obstacles for the player to overcome (Bellotti et al., 2010; Lopes & Bidarra, 2011).

[1] Corresponding author: Hrvoje Puskaric
Email: xpboje@gmail.com

These challenges should be designed in a way that is fair and balanced, but also challenging enough to keep the player engaged. By providing challenges, designers can create a sense of achievement and satisfaction for players.

Another important principle of game world design theory is storytelling. The game world should tell a story through its design and elements (Wei et al., 2010). This can include environmental storytelling, where the game world itself tells a story through its design and placement of objects. By using storytelling techniques, designers can create a world that is more engaging and meaningful for players.

Finally, game world design theory emphasizes the importance of player agency. The game world should provide the player with a sense of agency and control over their experience. This can include multiple paths or options for completing objectives, or the ability to make choices that affect the game world and its outcome. By giving players a sense of control, designers can help create a more engaging and immersive game world.

Overall, game world design theory is about creating a virtual environment that is engaging, immersive, and meaningful for players. By following these key principles, game designers can create game worlds that are memorable and enjoyable for players.

2. GAME ENGINES FOR WORLD CREATION

Game world creation engines are software tools that game developers use to create and design game worlds. These engines provide a wide range of features and tools that help developers create immersive and engaging game worlds more efficiently.

2.1 Unity engine

Unity is a powerful game engine that is used by game developers around the world to create a wide variety of games and interactive experiences (figure 1). It offers a range of features and tools for game development, including 3D modeling and animation tools, physics simulations, and real-time rendering apabilities (De Macedo & Formico Rodrigues, 2011; Jitendra et al., 2021).

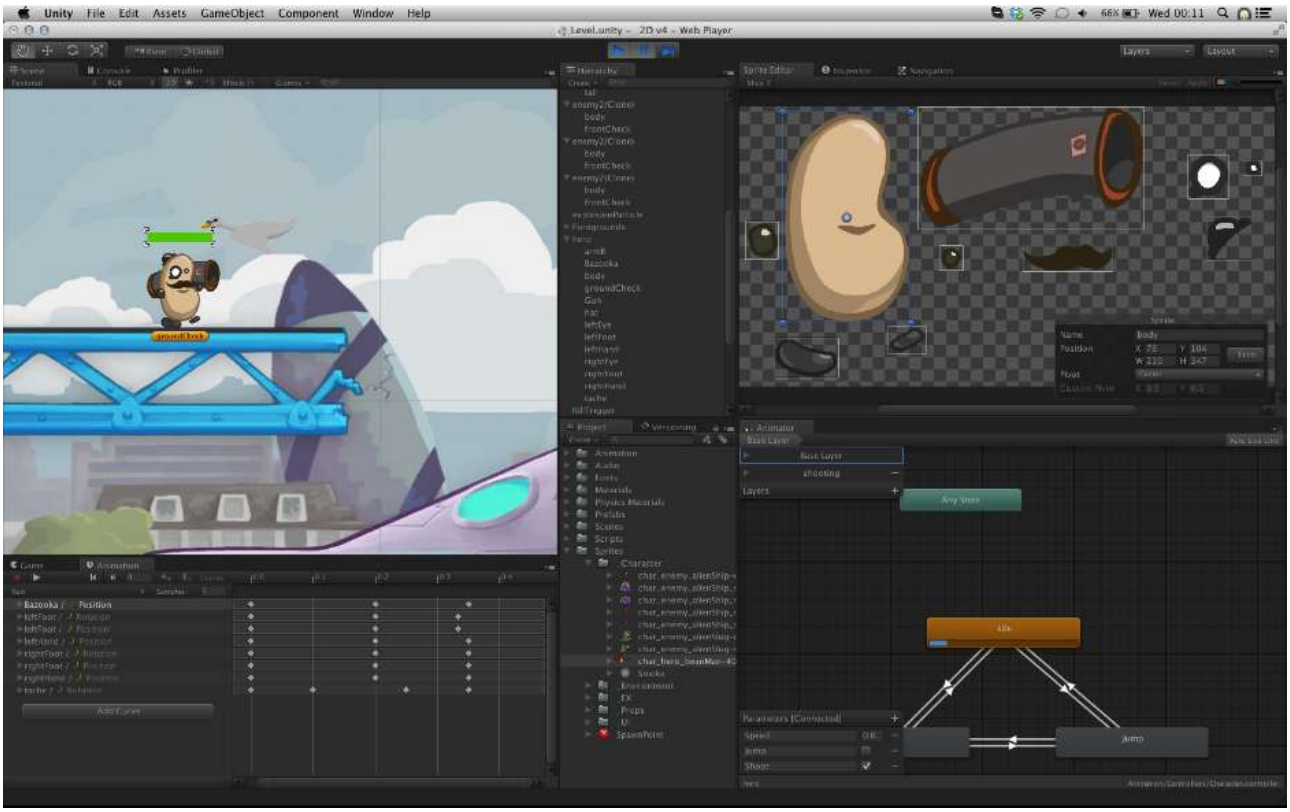

Figure 1. Unity engine interface (source https://techcrunch.com/2013/08/28/unity-game-engine-to-get-official-2d-game-support-and-a-built-in-ad-service/)

One of the key features of Unity is its cross-platform support. Games developed in Unity can be easily deployed to a wide range of platforms, including mobile devices, PCs, consoles, and even virtual and augmented reality devices. This makes it a popular choice for developers who want to reach a large audience with their games.

Unity also includes a visual editor that allows developers to create and design game worlds without having to write code. This editor includes a range of

tools for creating terrain, adding objects to the world, and configuring lighting and effects. It also includes a powerful scripting system that allows developers to create custom behaviors for their game objects using C# or UnityScript (a variant of JavaScript).

In addition to its core features, Unity also offers a range of add-ons and extensions that can be used to extend its functionality. These include tools for creating user interfaces, networking and multiplayer support, and AI and machine learning integration.

Another key advantage of Unity is its active and supportive community. There are a wide range of resources available for Unity developers, including tutorials, documentation, and forums where developers can ask questions and share their knowledge.

Overall, Unity is a versatile and powerful game engine that offers a range of features and tools for game development. Its cross-platform support, visual editor, and scripting system make it a popular choice for both beginner and experienced game developers.

2.2 Unreal engine

Unreal Engine is a popular game engine developed by Epic Games that is used by game developers to create high-quality, AAA games for a range of platforms, including consoles, PCs, and mobile devices (Lee, 2016; Satheesh, 2016). The engine is known for its advanced graphics capabilities, powerful tools, and flexibility (figure 2).

Figure 2. Unity engine interface (source https://techcrunch.com/2013/08/28/unity-game-engine-to-get-official-2d-game-support-and-a-built-in-ad-service/)

One of the key features of Unreal Engine is its advanced rendering capabilities. The engine uses a real-time rendering system that allows developers to create highly detailed and realistic environments, characters, and effects. It also includes a range of tools for creating advanced lighting and materials, as well as support for high dynamic range (HDR) rendering, which allows for a wider range of colors and brightness levels.

Unreal Engine also includes a powerful visual editor that allows developers to create and edit game worlds in real-time. This editor includes a range of tools for creating terrain, adding objects to the world, and configuring lighting and effects. It also includes a Blueprint visual scripting system that allows developers to create custom behaviors for their game objects without having to write code.

In addition to its core features, Unreal Engine also offers a range of add-ons and extensions that can be used to extend its functionality. These include tools for creating user interfaces, networking and multiplayer support, and physics simulations.

One of the key advantages of Unreal Engine is its active and supportive community. There are a wide range of resources available for Unreal Engine developers, including tutorials, documentation, and forums where developers can ask questions and share their knowledge.

Overall, Unreal Engine is a powerful and flexible game engine that offers a range of advanced features and tools for game development. Its advanced graphics

capabilities, powerful tools, and active community make it a popular choice for AAA game development.

3. ARTIFITIAL INTELIGENCE IN GAME DEVELOPMENT

Artificial intelligence (AI) is used extensively in gaming development to enhance the gaming experience and provide more realistic and immersive gameplay (Carpenter, 2023; ChatGPT successor creates entire video games in minutes). Here are some ways AI is used in gaming development:

Non-Player Characters (NPCs): AI is used to create intelligent NPCs that can act and react realistically in the game world. NPCs can have various behaviors and personalities, such as aggressive, defensive, passive, or social, and they can learn from the player's actions and adapt to different situations.

1) **Enemy AI:** AI is used to create intelligent enemies that can challenge the player and provide a more realistic gaming experience. Enemies can have different abilities and behaviors, and they can learn from the player's actions and adapt to different situations.
2) **Procedural Content Generation (PCG)**: AI is used to generate game content, such as levels, environments, and items. PCG can create unique and dynamic content that adapts to the player's actions and provides a more engaging gaming experience.
3) **Pathfinding:** AI is used to help characters navigate the game world by finding the best path to reach their destination. Pathfinding algorithms can calculate the shortest or safest route, avoid obstacles, and adjust to changing terrain.
4) **Natural Language Processing (NLP):** AI is used to understand and interpret player input, such as voice commands or text messages. NLP can provide more immersive and interactive gameplay by allowing players to communicate with NPCs or control the game using natural language.
5) **Machine Learning (ML):** AI is used to analyze player data and behavior to improve the game's performance and provide a more personalized gaming experience. ML can learn from player actions and adjust the game's difficulty, pacing, or content to match the player's skill level and preferences.

4. EXAMPLE OF ALGORITHMS IN GAME DEVELOPEMNT

An AI chat bot can be used in Unreal Engine to generate a game world by using natural language processing techniques to understand the user's preferences and then using procedural content generation algorithms to generate the game world. The process can be broken down into several steps.

First, the inputs need to be defined. This involves defining the types of inputs that the chat bot will receive, such as the user's preferences for terrain type, object density, quest types, and NPC personalities. These inputs will be used by the procedural content generation algorithms to generate the game world.

Next, the natural language processing (NLP) algorithms need to be trained. This involves providing the chat bot with a large dataset of text input and corresponding outputs. The chat bot will use this dataset to learn how to recognize and interpret natural language input from the user.

Once the NLP algorithms are trained, the chat bot can be integrated into the Unreal Engine game development environment. The chat bot will be programmed to listen for user input and respond with generated game world data based on the input received.

The generated game world data can then be used to create the game world in Unreal Engine. For example, the terrain generation algorithm might use the user's preference for mountainous terrain to generate a hilly landscape with large rock formations. Similarly, the object placement algorithm might use the user's preference for a dense object population to generate a forest with lots of trees and wildlife.

In summary, the process of using an AI chat bot in Unreal Engine involves defining the inputs, training the NLP algorithms, integrating the chat bot into the game development environment, generating game world data based on user input, and using the generated data to create the game world.

4.1 Use of Natural Language

In essential AI chat bot could be used to help create a game world in Unreal Engine.

User inputs a command such as "Generate a new game world".

The AI chat bot (ChatGPT successor creates entire video games in minutes) responds with a series of questions to help determine the parameters of the game world (e.g. size, terrain type, number of biomes, etc.).

The user responds to each question, providing the necessary information.

The AI chat bot then generates a game world using the parameters provided by the user, using a variety of algorithms and techniques such as Perlin noise, Voronoi diagrams, and L-Systems.

The user can then explore the generated game world in Unreal Engine and make adjustments as necessary.

Here is an example conversation between a user and an AI chat bot:

User: "Generate a new game world."

AI chat bot: "Sure! What size should the game world be? (e.g. 10x10, 20x20, etc.)"

User: "20x20."

AI chat bot: "What type of terrain do you want? (e.g. mountains, forests, etc.)"

User: "Mountains and forests."

AI chat bot: "Okay. How many biomes do you want to include? (e.g. one, two, three, etc.)"

User: "Two."

AI chat bot: "Great. Using the parameters you provided, I will now generate a new game world. This may take a few minutes, so please be patient."

The AI chat bot would then use the parameters provided by the user to generate a new game world in Unreal Engine, using a variety of algorithms and techniques such as Perlin noise, Voronoi diagrams, and L-Systems. Once the game world has been generated, the user can then explore it in Unreal Engine and make adjustments as necessary (Artificial intelligence. n.d.).'

4.2 In Code Ai Algoritms in Unreal Engine

AI chat bots could be used to create a game world in Unreal Engine:

Terrain Generation: An AI chat bot could ask the user to specify the type of terrain they want (e.g. mountains, plains, forests, etc.), the size of the game world, and any other relevant parameters. Based on the user's input, the AI chat bot could use algorithms such as Perlin noise, Voronoi diagrams, and fractal terrain generation to generate a unique and realistic game world (Code list 1).

Code list 1. Terrain Generation

```
// Prompt the user to specify the type of terrain they want
string terrainType = chatbot.promptUser("What type of terrain do you want?");

// Prompt the user to specify the size of the game world
int worldSize = chatbot.promptUser("What is the size of the game world?");

// Use Perlin noise to generate the terrain
for (int x = 0; x < worldSize; x++) {
    for (int y = 0; y < worldSize; y++) {
        float noiseValue = PerlinNoise(x, y);
        float terrainHeight = noiseValue * MAX_TERRAIN_HEIGHT;
        terrain.setVertexHeight(x, y, terrainHeight);
    }
}
```

Object Placement: An AI chat bot could ask the user to specify the type of objects they want to place in the game world (e.g. trees, rocks, buildings, etc.), the density of the objects, and any other relevant parameters. Based on the user's input, the AI chat bot could use algorithms such as Poisson disk sampling and random walks to intelligently and realistically place objects in the game world (Code list 2).

Code list 2. Object Placement

```
// Prompt the user to specify the type of objects they want to place
string objectType = chatbot.promptUser("What type of objects do you want to place?");

// Prompt the user to specify the density of the objects
float objectDensity = chatbot.promptUser("What is the density of the objects?");

// Use Poisson disk sampling to place objects
for (int x = 0; x < worldSize; x++) {
    for (int y = 0; y < worldSize; y++) {
        if (PoissonDiskSampling(x, y, objectDensity)) {
            Object* newObj = ObjectFactory::createObject(objectType);
            newObj->setPosition(x, y);
            gameWorld.addObject(newObj);
        }
    }
}
```

Quest Generation: An AI chat bot could ask the user to specify the type of quests they want to include in the game world (e.g. fetch quests, kill quests, puzzle quests, etc.), the difficulty level, and any other relevant parameters. Based on the user's input, the AI chat bot could use techniques such as procedural content generation and natural language processing to generate unique and engaging quests for the player to complete (Code list 3).

Code list 3. Quest Generation

```
// Prompt the user to specify the type of quests they want to include
string questType = chatbot.promptUser("What type of quests do you want to include?");

// Prompt the user to specify the difficulty level
int difficultyLevel = chatbot.promptUser("What is the difficulty level?");

// Use procedural content generation to generate quests
Quest* newQuest = ProceduralContentGenerator::generateQuest(questType, difficultyLevel);
gameWorld.addQuest(newQuest);
```

NPC Generation: An AI chat bot could ask the user to specify the type of non-playable characters (NPCs) they want to include in the game world (e.g. merchants, warriors, mages, etc.), the personality traits of the NPCs, and any other relevant parameters. Based on the user's input, the AI chat bot could use techniques such as decision trees and Markov chains to generate realistic and dynamic NPCs that interact with the player in unique and interesting ways (Code list 4).

Code list 4. NPC Generation

```
// Prompt the user to specify the type of NPCs they want to include
string npcType = chatbot.promptUser("What type of NPCs do you want to include?");

// Prompt the user to specify the personality traits of the NPCs
string personalityTraits = chatbot.promptUser("What are the personality traits of the NPCs?");

// Use decision trees to generate NPCs
NPC* newNPC = DecisionTreeGenerator::generateNPC(npcType, personalityTraits);
gameWorld.addNPC(newNPC);
```

Overall, AI chat bots have the potential to revolutionize game world creation in Unreal Engine by providing an intelligent and user-friendly interface for generating unique and engaging game worlds.

5. CONCLUSION

In conclusion, AI algorithms have greatly influenced the creation of game worlds in the gaming industry. The use of AI algorithms in game development has enabled game developers to create immersive and dynamic game worlds that can adapt to the player's preferences and behavior. With the use of AI algorithms such as machine learning, natural language processing, and procedural content generation, game developers can generate game worlds with minimal human input, saving time and resources.

The Unreal Engine and Unity Engine are two of the most popular game design tools that game developers use to create game worlds. These game engines come equipped with various AI tools and features that enable game developers to incorporate AI algorithms into their game development process.

AI chat bots can be used in Unreal Engine to generate game worlds by using natural language processing techniques to understand the user's preferences and generating game world data based on the input received. Additionally, AI algorithms can be used in Unreal Engine to generate game world data for terrain, object placement, and NPC behavior, among other things.

In summary, the use of AI algorithms in game development has revolutionized the way game worlds are created, making them more immersive, interactive, and adaptable to player preferences. As AI technology continues to advance, we can expect to see even more innovative and creative use of AI in game development, leading to more engaging and immersive game worlds for players to enjoy.

References:

Artificial intelligence. (n.d.). Retrieved April 10, 2023, from https://docs.unrealengine.com/5.1/en-US/artificial-intelligence-in-unreal-engine/

Bellotti, F., Berta, R., & De Gloria, A. (2010). Designing effective serious games: opportunities and challenges for research. *International Journal of Emerging Technologies in Learning (iJET)*, 5(2010), 23-35.

Bormann, D., & Greitemeyer, T. (2015). Immersed in virtual worlds and minds: effects of in-game storytelling on immersion, need satisfaction, and affective theory of mind. *Social Psychological and Personality Science*, 6(6), 646-652.

Carpenter, N. (2023, March 22). Can AI really make a video game? *Polygon*. https://www.polygon.com/ai-artificial-intelligence/23650693/chatgpt-generative-ai-video-game-development

ChatGPT successor creates entire video games in minutes. (2023, March 15). The Independent. https://www.independent.co.uk/tech/chatgpt-gpt-4-ai-video-games-b2301358.html

De Macedo, D. V., & Formico Rodrigues, M. A. (2011). Experiences with rapid mobile game development using unity engine. *Computers in Entertainment*, *9*(3), 1–12. DOI: 10.1145/2027456.2027460

Jitendra, M. S. N. V., Srinivas, A. S., Surendra, T., Rao, R. V., & Chowdary, P. R. (2021). *A study on game development using unity engine*. 040001. DOI:10.1063/5.0066303

Flanagan, M., Howe, D. C., & Nissenbaum, H. (2005, April). Values at play: Design tradeoffs in socially-oriented game design. In Proceedings of the SIGCHI conference on human factors in computing systems (pp. 751-760).

Grizioti, M., & Kynigos, C. (2021, June). Children as players, modders, and creators of simulation games: A design for making sense of complex real-world problems: Children as players, modders and creators of simulation games. In Proceedings of the 20th Annual ACM Interaction Design and Children Conference (pp. 363-374).

Lee, J. (2016). *Learning unreal engine game development: A step-by-step guide that paves the way for developing fantastic games with Unreal Engine 4*. Packt Publishing.

Lopes, R., & Bidarra, R. (2011). Adaptivity challenges in games and simulations: a survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(2), 85-99.

Pagulayan, R. J., Keeker, K., Wixon, D., Romero, R. L., & Fuller, T. (2002). User-centered design in games. In The human-computer interaction handbook (pp. 915-938). CRC Press.

Reeves, B., & Read, J. L. (2009). Total engagement: How games and virtual worlds are changing the way people work and businesses compete. Harvard Business Press.

Rupp, A. A., Gushta, M., Mislevy, R. J., & Shaffer, D. W. (2010). Evidence-centered design of epistemic games: Measurement principles for complex learning environments. *The Journal of Technology, Learning and Assessment*, 8(4).

Rapp, A. (2017). Designing interactive systems through a game lens: An ethnographic approach. *Computers in human behavior*, 71, 455-468.

Satheesh, P. V. (2016). *Unreal Engine 4 game development essentials: Master the basics of Unreal Engine 4 to build stunning video games*. Packt Publishing.

Wei, H., Bizzocchi, J., & Calvert, T. (2010). Time and space in digital game storytelling. *International Journal of Computer Games Technology*, 2010, 1-23.

Hrvoje Puškarić
Academy of Professional Studies
Sumadija,
Kragujevac, Serbia
hpuskaric@asss.edu.rs
ORCID: 0000-0002-9239-1867