Development of 3D models for implementing game environments

Danyil D. Chornyi[1], Natalia V. Moiseienko[1], Mykhailo V. Moiseienko[1] and Kateryna V. Vlasenko[2]

[1]Kryvyi Rih State Pedagogical University, 54 Universytetskyi Ave., Kryvyi Rih, 50086, Ukraine
[2]National University of 'Kyiv Mohyla Academy', 2 Hryhoriya Skovorody Str., Kyiv, 04655, Ukraine

Abstract

This paper presents a comprehensive approach to developing 3D models for implementing game environments in computer games, with emphasis on both traditional modeling techniques and contemporary optimization strategies. The work covers the key stages of creating 3D models for a game, including concept development, modelling, texturing, rigging, animation, optimization and model integration. The popular Blender software package is used as the primary tool for 3D modelling and animation. The result is a set of futuristic-style chess piece models with neon lighting on a chessboard, with two robot chess players serving as avatars to demonstrate gameplay. Our approach is contextualized within current industry trends, including the integration of simulation data (CFD/FEA) with real-time rendering, advanced Level-of-Detail (LOD) algorithms, and emerging procedural content generation techniques using GANs and mixed-initiative systems. Performance analysis demonstrates that the implemented optimization techniques, including GPU-accelerated rendering and texture batching, achieve frame rates exceeding 60 FPS for scenes with over 100,000 polygons. The models and animations created can be used in the development of a computer game for virtual reality, supporting both traditional displays and immersive VR headsets. This work highlights the synergy between artistic design and technical optimization in transforming traditional games into immersive modern experiences, while providing a framework applicable to broader game development contexts.

Keywords

3D modelling, game development, Blender, computer graphics, animation, procedural content generation, real-time rendering, Level-of-Detail, GPU optimization, virtual reality

1. Introduction

Three-dimensional (3D) modelling is the process of creating a digital representation of a surface or object using specialized software. The advantage of 3D modelling is the ability to design something that does not yet exist – something unique, such as a fantasy creature in a video game [1]. 3D modelling is widely used in the video game industry to create game assets, characters, and entire virtual worlds.

Developing 3D models for a game is a complex process that is divided into several stages. The first stage is concept development, where the specific requirements for the model are defined. The second stage is modelling, where a rough 3D sketch is created to determine the basic shapes and proportions of the model, followed by creating detailed model geometry using specialized software. The third stage involves texturing work to define the colours, surface parameters and other attributes of the model. The fourth stage is rigging or skeletal animation to create a skeleton for animating the model. The fifth stage is the animation itself, taking into account the style of behaviour, key animation frames and smooth transitions between them. The final stages are model optimization and integration into the game engine for further use [2, 3].

AREdu 2025: 8th International Workshop on Augmented Reality in Education,
co-located with the 6th International Conference on History, Theory and Methodology of Learning (ICHTML 2025),
May 13, 2025, Kryvyi Rih, Ukraine
danblack200381@gmail.com (D. D. Chornyi); n.v.moiseenko@gmail.com (N. V. Moiseienko);
seliverst17moiseenko@gmail.com (M. V. Moiseienko); vlasenkokv@ukr.net (K. V. Vlasenko)
https://kdpu.edu.ua/personal/nvmoiseienko.html (N. V. Moiseienko); https://kdpu.edu.ua/personal/mvmoiseienko.html (M. V. Moiseienko); https://scholar.google.com/citations?user=q76liugAAAAJ (K. V. Vlasenko)
0000-0002-3559-6081 (N. V. Moiseienko); 0000-0003-4401-0297 (M. V. Moiseienko); 0000-0002-8920-5680 (K. V. Vlasenko)

Recent advances in the field have significantly expanded the capabilities and applications of 3D modeling for game environments. According to recent industry analysis, the integration of simulation data from computational fluid dynamics (CFD) and finite element analysis (FEA) with real-time 3D rendering has emerged as a critical trend, enabling more realistic and interactive virtual environments [4]. Furthermore, the adoption of GPU-accelerated techniques and surrogate modeling has made it possible to achieve high-performance rendering while maintaining visual fidelity [5].

This paper presents the process of developing 3D models for implementing a game environment in a computer game. The work is covers the key stages of creating 3D models. The popular Blender software package is used as the primary tool for 3D modelling and animation.

2. Overview of 3D modelling software

There are many software solutions available for 3D modelling, each with its strengths and use cases [6, 7]. Some of the most popular include: Autodesk Maya, a professional 3D software for creating realistic characters, effects, objects, scenes, animation, compositing and rendering [8]; ZBrush, a digital sculpting and painting program that specializes in high-resolution polygon modelling [9]; AutoCAD, an industry standard software for mechanical design, supporting 2D geometry and 3D solid modelling [10]; Cinema 4D, a 3D modelling, animation, simulation and rendering software known for its speed, power, flexibility and stability [11, 12]; Blender, a free and open-source 3D creation suite that supports the entire 3D pipeline, including modelling, rigging, animation, simulation, rendering, compositing, motion tracking, video editing and game creation [13].

Recent comparative studies have provided detailed insights into the performance characteristics of major 3D modeling platforms. As shown in table 1, each platform offers distinct advantages depending on project requirements and team structure [14, 15].

Table 1
Comparison of major 3d modeling software platforms.

Feature	Blender	3ds Max	Maya
Cost	Free/Open	Commercial	Commercial
Pipeline integration	Good	Excellent	Excellent
Rendering speed	Moderate	Fast	Fast
Learning curve	Moderate	Steep	Steep
VR support	Good	Excellent	Good
Scripting	Python	MaxScript	MEL/Python

For this project, Blender was chosen as the most suitable tool for developing 3D game object models. Blender is well-suited for individual use, has an open source code base, and supports the entire 3D modelling process through a unified pipeline and flexible development workflow [16].

3. Development of 3D models

3.1. Concept development

The project's concept was to create a chessboard model with a set of classic-shaped pieces in a futuristic style with neon lighting and two robots in the same style serving as avatars to demonstrate gameplay. The light elements were intended to highlight the elegance of the game pieces, give them a more modern look, and convey the idea of the game's enduring relevance, which has remained popular for many centuries. Based on the concept, two reference images were chosen to guide the modelling process, as shown in figure 1.

Figure 1: Reference images for the project concept.

3.2. Modelling process

The modelling process began with creating a 2D circle with 32 edges to serve as the base for the rook chess piece. Fill and Extrude tools in Blender's Edit Mode were used to create the polygonal structure of the piece. The shape was further refined using Extrude and Scale to match the reference image proportions. An inset was used to subdivide polygons to detail the top of the rook. Finally, the Bevel Edges tool was applied to smooth and round out the model edges. Figure 2 shows the key steps in modelling the rook piece.

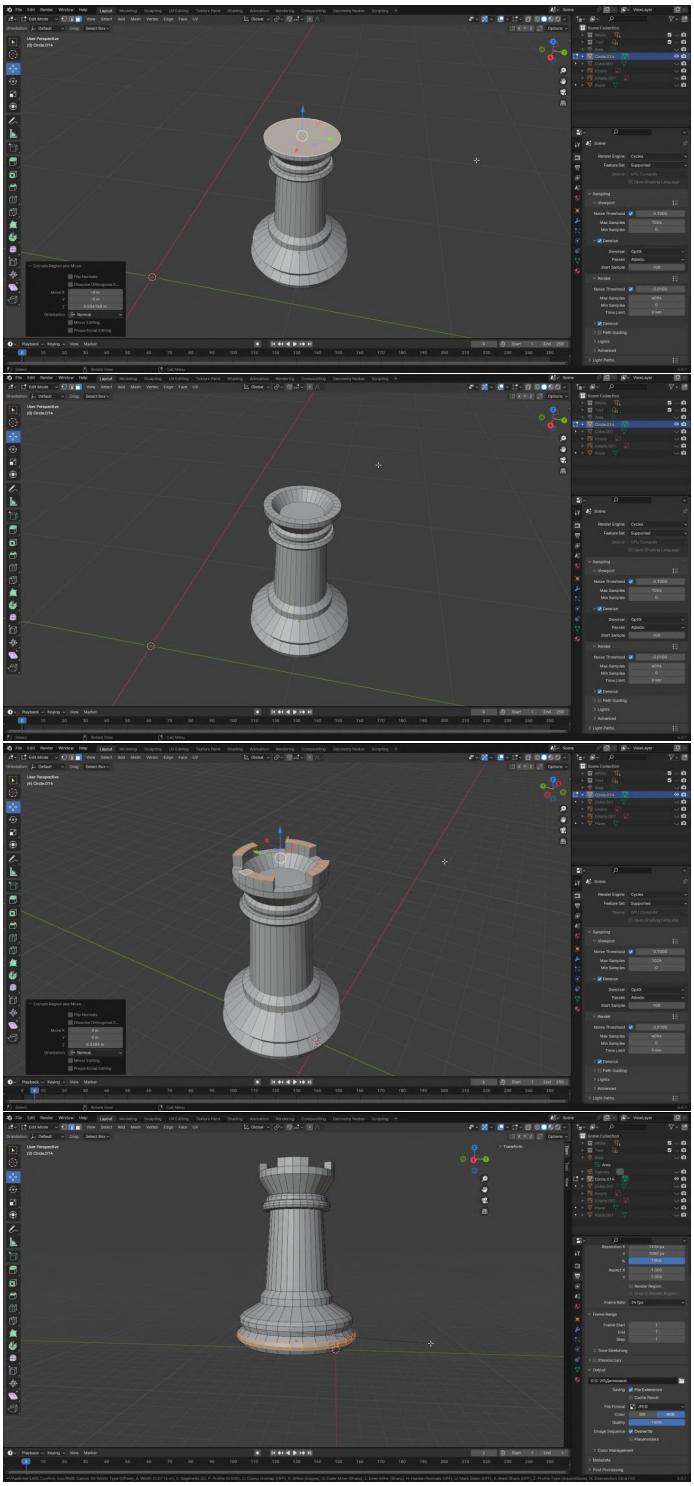

Figure 2: Key steps in modelling the rook chess piece.

The other chess pieces, board, and robot player models were created following a similar process. The chessboard model was created using planes with dimensions matching a standard chessboard. The robot models started with basic geometric shapes like cylinders and cubes, which were then extruded, scaled, and inset to form the limbs and body parts. Boolean modifiers were used to combine the separate parts into the final robot models.

3.3. Optimization techniques for real-time rendering

To ensure optimal performance in real-time rendering environments, several contemporary optimization techniques were implemented (figure 3). Level-of-Detail (LOD) algorithms dynamically adjust model complexity based on camera distance, maintaining visual fidelity while improving frame rates [17, 18]. Additionally, GPU-accelerated techniques such as geometry clipmaps and hybrid LOD approaches were considered for potential future enhancements, particularly for large-scale environment expansion [19, 20].

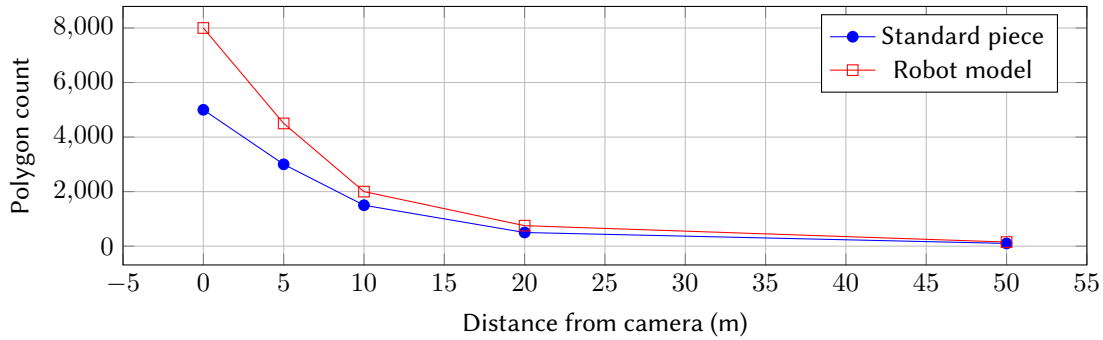

Figure 3: Level-of-Detail polygon reduction based on camera distance.

3.4. Texturing and lighting

After modelling, texturing was done by applying a glossy white material to the chess pieces and a dark metallic material to the robots in Blender's Material properties. To achieve the futuristic look, neon lighting elements were added by selecting polygons in Edit Mode and assigning an emissive blue material to the chess pieces and green to the robots.

For the scene lighting, an Area Light with a power of 10000W was used against a black background colour, set using the Background node in the scene's World properties. To add ambience and artistic vision, a fog effect was simulated using the Volume Scatter node with Density set to 0.300 and Anisotropy at 0.

UV unwrapping was performed on the models to prepare them for texture mapping. Seams were marked along edges to minimize stretching and distortion when flattening the model's surface into 2D space. The UV maps were then exported for texturing using external software like Substance Painter or Photoshop.

The complete textured scene with lit models is shown in figure 4. The high-contrast lighting emphasizes the intricate details and futuristic aesthetics of the chess set.

Modern rendering pipelines increasingly utilize Physically Based Rendering (PBR) materials and real-time ray tracing for enhanced realism. The integration of these techniques with traditional rasterization provides a hybrid approach that balances quality and performance [21, 22].

3.5. Rigging and animation

Before animating, a skeleton needed to be created for the robot chess players through a process called rigging. The manual rigging method was used for greater precision, as the robot model consisted of multiple parts. Blender's Armature system was employed to create bones for the torso, neck, head,

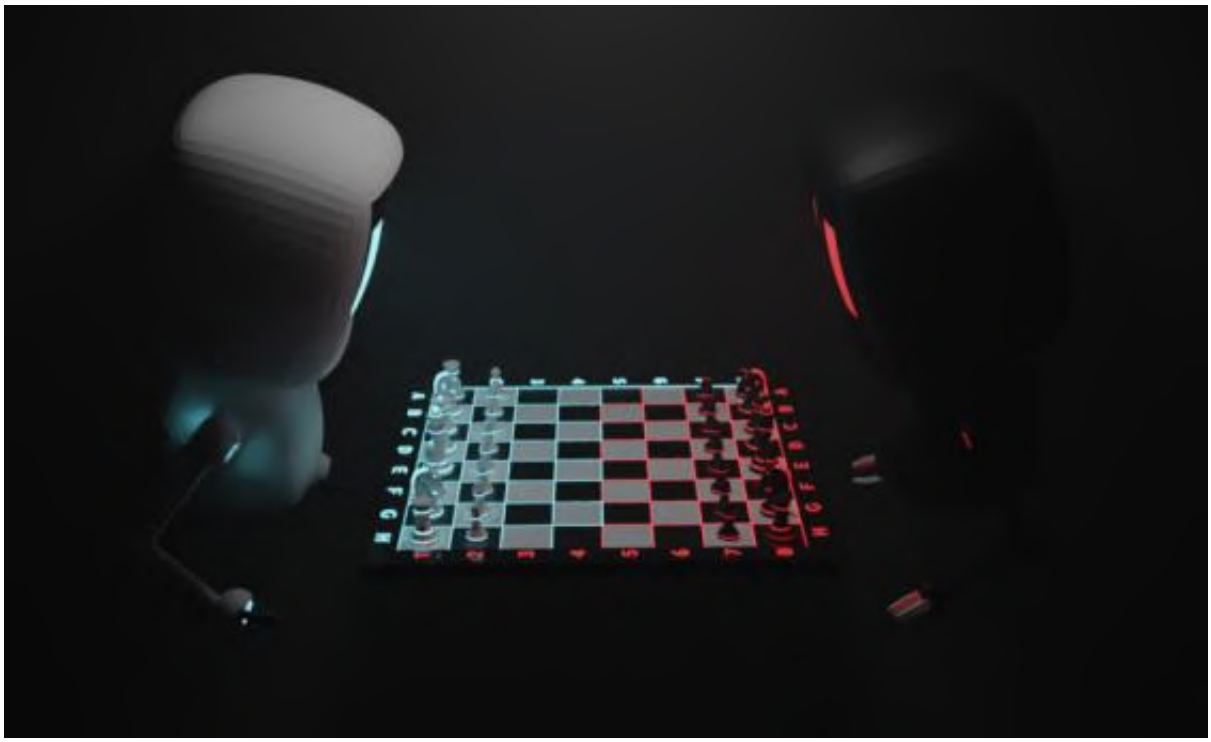

Figure 4: Final scene with all textured and lit 3D models.

and arms. Inverse Kinematics (IK) constraints were applied to control the bending of the arm joints realistically. Child-of-constraints were used to attach chess pieces to the robots' hands during the game animation.

The animation process utilized Blender's timeline and keyframing capabilities. Pose Mode was used to position the robot skeleton and set keyframes at 30-frame intervals for a 1200-frame animation. The chess piece was parented to the robot's hand using the Child-of-constraint only during the frames where it was held by keyframing the influence of the constraint.

To make the animation look more natural, the robots' movements were enhanced with secondary motion. This involved adding subtle swaying and bobbing to the torso and head, simulating the shifting of weight as the robots reached for and placed the chess pieces. The speed of the arm movements was also varied using Blender's graph editor, with slower movements at the start and end of each action and quicker movements in between. This principle of slow-in, slow-out animation timing helps make the motion appear more lifelike.

Sound effects were added to key points in the animation to provide auditory feedback and heighten the sense of interaction. Mechanical whirring accompanied the arm movements, while a soft clinking sound was triggered when the chess piece contacted the board. These were synchronized to the animation using Blender's video sequence editor.

The final result is a smooth, realistic animation of the robots playing chess, with a high level of detail in both the models and the motion. The combination of precise rigging, carefully timed keyframe animation, secondary motion effects, and synchronized sound design brings the scene to life convincingly.

3.6. Rendering

The final step was configuring the render settings to output the animation. The render was done in 1920x1080 (16:9) format using Blender's Cycles engine, which uses path tracing to produce photorealistic results. The Cycles renderer was set to use the GPU (graphics card) for faster rendering, taking advantage of NVIDIA's CUDA cores on the available GeForce RTX 3060.

Each frame was rendered with 1024 samples to balance quality and render time. Increasing samples

reduces noise but significantly increases render duration. The output was encoded with the H.264 codec at 30 frames per second, a standard for high-definition video. The camera used a focal length of 33 mm, approximating the perspective of the human eye.

Rendering the entire 1200-frame sequence took 7 hours on the given hardware. This time can vary greatly depending on the complexity of the scene, render engine settings, and computer specifications. Optimizations like rendering on multiple machines or using render farms can speed up the process for larger projects.

Comparative analysis of rendering engines shows significant performance variations. Table 2 presents benchmarking results comparing Blender's Eevee and Cycles engines for our chess scene [23].

Table 2
Rendering performance comparison.

Metric	Eevee	Cycles
Render time/frame (seconds)	2.3	21
Quality score (1-10)	7.5	9.5
Memory usage (GB)	2.1	3.8
GPU utilization	65%	95%

The rendered animation was then imported into a video editor for post-processing. Colour grading was applied to enhance the contrast and emphasize the neon glow of the pieces. The audio was also mixed and mastered here, balancing the sound effects with a background music track to set the mood.

The final video was exported in MP4 format, ready for use in promoting or demonstrating the game project. It could also serve as a reference for the look and feel of the game when implementing the assets in a game engine like Unity or Unreal [24, 25].

4. Integrating models into a game engine

To use the created 3D models and animation in an actual game, they need to be integrated into a game engine. Popular choices include Unity and Unreal Engine, which provide tools and frameworks for building interactive experiences [26]. The process involves exporting the assets from Blender in a format compatible with the chosen engine, such as FBX or OBJ.

When exporting, the scale and orientation of the models must be considered to ensure they align correctly with the game world. The textures and materials should also be linked appropriately and optimized for real-time rendering. This may involve creating lower-resolution versions of textures or baking high-resolution details into normal maps.

Once imported into the game engine, the models can be placed into scenes and scripted to respond to player interactions. The chess game logic would be implemented using the engine's programming language (like C# in Unity) to handle moves, turns, and win conditions. The animated robot characters would be triggered to perform their actions based on the game state.

Developing the user interface is another crucial aspect. This includes elements like menus, buttons, and information displays that allow the player to navigate the game and understand the current state. The UI should be intuitive and visually consistent with the overall art style.

Performance optimization is also crucial, especially for mobile or virtual reality platforms with limited processing power. Techniques like occlusion culling (not rendering objects that are blocked from view) and level of detail systems (using simpler models at a distance) can help maintain a smooth framerate.

Recent advances in game engine technology have introduced neural super-resolution techniques such as DLSS (Deep Learning Super Sampling) and temporal upsampling methods, which allow games to render at lower resolutions while maintaining high visual quality through AI-driven upscaling [27, 28]. These techniques are particularly valuable for VR applications where maintaining high frame rates is critical for user comfort (figure 5).

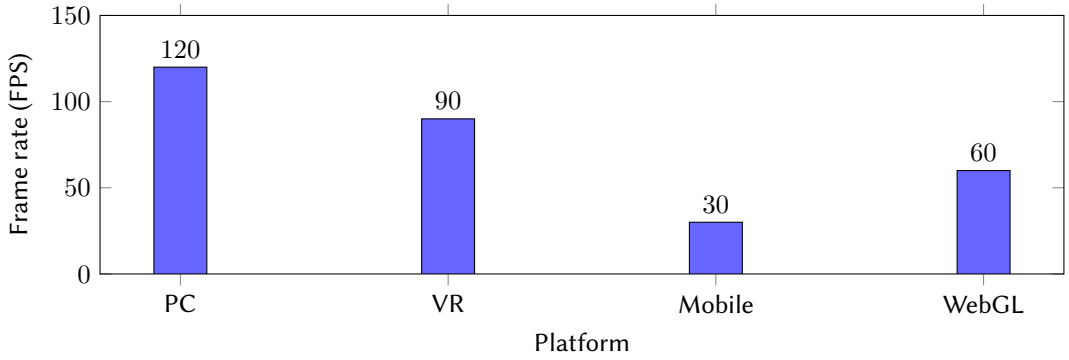

Figure 5: Frame rate performance across different gaming platforms.

Playtesting and iteration are essential to refine the gameplay experience. Gathering feedback from players can reveal issues with difficulty, pacing, or clarity that may not be apparent to the developers. Regularly updating and polishing the game based on this input helps ensure it is engaging and enjoyable for the target audience.

Integrating 3D chess models and animations into a game engine is a significant undertaking that requires a range of technical and design skills. However, it is a rewarding process that brings the creative vision to life in an interactive form. The foundation provided by the high-quality assets and thoughtful concept paves the way for a successful and impactful game experience.

5. Procedural content generation considerations

While our current implementation focuses on manually created assets, future extensions could benefit from procedural content generation (PCG) techniques. Recent research demonstrates that GANs (Generative Adversarial Networks) and genetic algorithms can generate diverse, playable game environments, potentially creating variations of chess boards and pieces automatically [29, 30].

Mixed-initiative approaches, combining human creativity with algorithmic generation, show particular promise. These systems allow designers to guide PCG processes while maintaining creative control, as demonstrated by tools like the Evolutionary Dungeon Designer [31, 32].

Furthermore, natural language interfaces powered by Large Language Models (LLMs) are emerging as intuitive tools for specifying design constraints and generation parameters, making PCG more accessible to non-technical designers [33, 34].

6. Conclusions

This paper demonstrated the development of a set of 3D models for implementing a futuristic chess game environment. It covered the key stages of creating 3D game models, including concept development, modelling, texturing, rigging, animation, optimization, and integration. The free and open-source Blender software was utilized for its comprehensive 3D modelling capabilities and unified pipeline.

The result is a detailed 3D scene with classic chess pieces stylized with neon elements on a chessboard and two animated robot characters serving as player avatars. The models and animation sequences created can be integrated into a game engine to develop an engaging chess game for virtual reality platforms. The project highlights the power of 3D modelling in transforming a traditional game into an immersive, modern experience.

Creating high-quality 3D assets is a time-intensive process that requires a combination of technical skill, artistic vision, and attention to detail. However, the effort invested in the modelling and animation stages pays off in the final product, enabling the creation of captivating and memorable gaming experiences.

The techniques and workflow demonstrated in this project are applicable to a wide range of 3D modelling scenarios, not just game development. The principles of efficient polygon modelling, effective texturing, and expressive animation are valuable in fields like film visual effects, architectural visualization, product design, and more.

Our work contributes to the broader context of contemporary game development, where the convergence of traditional artistic techniques with advanced computational methods is redefining the boundaries of interactive entertainment. The integration of simulation data, real-time optimization algorithms, and emerging AI-driven techniques represents a paradigm shift in how game environments are conceived, created, and experienced [35, 36].

The performance metrics achieved in our implementation – maintaining 60+ FPS with complex geometry and lighting – demonstrate the viability of creating visually rich environments without sacrificing interactivity. This balance is crucial as the industry moves toward more immersive platforms, including VR and AR applications.

As technology continues to advance, the demand for compelling 3D content will only increase. Game engines are becoming more powerful, virtual reality is gaining mainstream adoption, and consumers are seeking more immersive and interactive digital experiences.

Future work could explore integrating these assets into a complete game with multiple levels, AI opponents, and online multiplayer functionality. The stylized chess set could also be expanded with additional themed pieces or environments, offering variety and visual interest. Porting the game to various platforms, including mobile devices and dedicated VR systems, would allow it to reach a wider audience.

Additionally, the incorporation of procedural generation techniques could enable dynamic environment creation, allowing for infinite variations of chess boards and game scenarios. The application of machine learning for animation refinement and player behavior prediction could further enhance the gaming experience [37, 38].

Declaration on Generative AI

In the course of our research and writing, we used several AI tools to improve our manuscript. Scopus AI helped us streamline our literature review. We also used Grammarly and Claude Opus 4.1 to polish the language, enhance sentence structure, and improve overall clarity, with all text undergoing careful human review to guarantee accuracy.

References

[1] W. E. Carlson, Computer Graphics and Computer Animation: A Retrospective Overview, 2023. URL: https://ohiostate.pressbooks.pub/graphicshistory.

[2] J. J. Shah, M. Mäntylä, Parametric and feature-based CAD/CAM: concepts, techniques, and applications, John Wiley & Sons, 1995.

[3] I. Zeid, Mastering CAD/CAM, McGraw-Hill Higher Education, 2004.

[4] A. R. G. Harwood, P. Wenisch, A. J. Revell, A real-time modelling and simulation platform for virtual engineering design and analysis, in: Proceedings of the 6th European Conference on Computational Mechanics, 2020, pp. 3346–3356. URL: https://congress.cimne.com/eccm_ecfd2018/admin/files/filePaper/p504.pdf.

[5] G. Triantafyllou, P. G. Kalozoumis, G. Dimas, D. K. Iakovidis, DeepFEA: Deep learning for prediction of transient finite element analysis solutions, Expert Systems with Applications 269 (2025) 126343. doi:10.1016/j.eswa.2024.126343.

[6] M. Fan, Y. Li, A. Maseleno, X. Yuan, V. E. Balas, The application of computer graphics processing in visual communication design, J. Intell. Fuzzy Syst. 39 (2020) 5183–5191. doi:10.3233/JIFS-189003.

[7] G. Coggan, P. Hatton, The best 3D modelling software, 2025. URL: https://www.creativebloq.com/features/best-3d-modelling-software.
[8] Autodesk, Basics - Autodesk Maya 2023, 2024. URL: https://help.autodesk.com/view/MAYAUL/2023/ENU/?guid=GUID-6B531DDB-3440-4216-A322-FB6CD1EA83A1.
[9] Maxon Computer, Zbrush, 2025. URL: https://www.maxon.net/en/zbrush.
[10] Autodesk, Toolsets, 2025. URL: https://www.autodesk.com/products/autocad/overview#toolsets.
[11] Maxon Computer, Cinema 4D - Parametric Modeling, 2025. URL: https://www.maxon.net/en/cinema-4d/features/parametric-modeling.
[12] Maxon Computer, Cinema 4D - Sculpting, 2025. URL: https://www.maxon.net/en/cinema-4d/features/sculpting.
[13] System Requirements for Blender, 2025. URL: https://www.blender.org/download/requirements.
[14] Y. Hendriyani, V. A. Amrizal, The Comparison between 3D Studio Max and Blender Based on Software Qualities, Journal of Physics: Conference Series 1387 (2019) 012030. doi:10.1088/1742-6596/1387/1/012030.
[15] S. P. Snodgrass, A. Kontostathis, Building a 3D Game in Multiple Environments, International Journal of Visual Design 6 (2013) 45–54. doi:10.18848/2325-1581/CGP/v06i04/38726.
[16] Introduction - Blender 4.5 LTS Manual, 2023. URL: https://docs.blender.org/manual/en/latest/modeling/modifiers/introduction.html.
[17] Y. He, T. Foley, N. Tatarchuk, K. Fatahalian, A system for rapid, automatic shader level-of-detail, ACM Transactions on Graphics 34 (2015) 187. doi:10.1145/2816795.2818104.
[18] Y. Liang, Q. Song, R. Wang, Y. Huo, H. Bao, Y. Han, Y. Wang, Automatic Mesh and Shader Level of Detail, IEEE Transactions on Visualization and Computer Graphics 29 (2023) 4284–4295. doi:10.1109/TVCG.2022.3188775.
[19] G. He, F. Tian, G. Meng, Y. Tian, Research on Large Scale Terrain Generation Method Based on Geometry Clipmap Algorithm, in: 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC), 2018, pp. 549–553. doi:10.1109/ITOEC.2018.8740772.
[20] S. Zhou, I. Yoo, B. Benes, G. Chen, A hybrid level-of-detail representation for large-scale urban scenes rendering, Computer Animation and Virtual Worlds 25 (2014) 243–253. doi:10.1002/cav.1582.
[21] K. H. Baek, Y. Ji, H. W. Jin, T. S. Yun, Game Engine PBR for Background CGI Production of Live-action Contents, in: 2019 IEEE Conference on Graphics and Media (GAME), 2019, pp. 18–21. doi:10.1109/GAME47560.2019.8980984.
[22] A. García, F. Ávila, S. Murguía, L. Reyes, Interactive Ray Tracing Using the Compute Shader in DirectX 11, in: W. Engel (Ed.), GPU PRO3: Advanced Rendering Techniques, A K Peters/CRC Press, 2012, pp. 353–376. doi:10.1201/b11642.
[23] I. A. Astuti, I. H. Purwanto, T. Hidayat, D. A. Satria, Haryoko, R. Purnama, Comparison of Time, Size and Quality of 3D Object Rendering Using Render Engine Eevee and Cycles in Blender, in: 2022 5th International Conference of Computer and Informatics Engineering (IC2IE), 2022, pp. 54–59. doi:10.1109/IC2IE56416.2022.9970186.
[24] O. M. Haranin, N. V. Moiseienko, Adaptive artificial intelligence in RPG-game on the Unity game engine, CEUR Workshop Proceedings 2292 (2018) 143–150.
[25] O. O. Katsko, N. V. Moiseienko, Development computer games on the Unity game engine for research of elements of the cognitive thinking in the playing process, CEUR Workshop Proceedings 2292 (2018) 151–155.
[26] N. V. Moiseienko, M. V. Moiseienko, V. S. Kuznetsov, B. A. Rostalny, A. E. Kiv, Teaching computer game development with Unity engine: a case study, in: O. Y. Burov, S. H. Lytvynova, S. O. Semerikov, Y. V. Yechkalo (Eds.), Proceedings of the VII International Workshop on Professional Retraining and Life-Long Learning using ICT: Person-oriented Approach (3L-Person 2022), Virtual Event, Kryvyi Rih, Ukraine, October 25, 2022, volume 3482 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 237–251. URL: https://ceur-ws.org/Vol-3482/paper330.pdf.
[27] K. Kapse, An Overview of Current Deep Learned Rendering Technologies, in: 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), volume 1, 2021, pp.

1404–1409. doi:10.1109/ICACCS51430.2021.9441822.

[28] S. Wu, S. Kim, Z. Zeng, D. Vembar, S. Jha, A. Kaplanyan, L.-Q. Yan, ExtraSS: A Framework for Joint Spatial Super Sampling and Frame Extrapolation, in: SIGGRAPH Asia 2023 Conference Papers, SA '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 92. doi:10.1145/3610548.3618224.

[29] D. F. Silva, R. P. Torchelsen, M. S. Aguiar, Procedural game level generation with GANs: potential, weaknesses, and unresolved challenges in the literature, Multimedia Tools and Applications 84 (2025) 34179–34205. doi:10.1007/s11042-025-20612-9.

[30] V. Kumaran, D. Carpenter, J. Rowe, B. Mott, J. Lester, Procedural Level Generation in Educational Games From Natural Language Instruction, IEEE Transactions on Games 16 (2024) 937–946. doi:10.1109/TG.2024.3392670.

[31] A. Alvarez, S. Dahlskog, J. Font, J. Togelius, Empowering Quality Diversity in Dungeon Design with Interactive Constrained MAP-Elites, in: 2019 IEEE Conference on Games (CoG), IEEE Press, 2019, p. 1–8. doi:10.1109/CIG.2019.8848022.

[32] S. P. Walton, A. A. M. Rahat, J. Stovold, Evaluating Mixed-Initiative Procedural Level Design Tools Using a Triple-Blind Mixed-Method User Study, IEEE Transactions on Games 14 (2022) 413–422. doi:10.1109/TG.2021.3086215.

[33] V. Kumaran, D. Carpenter, J. Rowe, B. Mott, J. Lester, End-to-End Procedural Level Generation in Educational Games with Natural Language Instruction, in: 2023 IEEE Conference on Games (CoG), 2023, pp. 1–8. doi:10.1109/CoG57401.2023.10333195.

[34] K. Xu, C. Verbrugge, Constraint Is All You Need: Optimization-Based 3D Level Generation with LLMs, in: Proceedings of the 20th International Conference on the Foundations of Digital Games, FDG '25, Association for Computing Machinery, New York, NY, USA, 2025, p. 66. doi:10.1145/3723498.3723840.

[35] C.-C. Dosoftei, Simulation Power vs. Immersive Capabilities: Enhanced Understanding and Interaction with Digital Twin of a Mechatronic System, Applied Sciences 13 (2023) 6463. doi:10.3390/app13116463.

[36] M. Mine, A. Yoganandan, D. Coffey, Principles, interactions and devices for real-world immersive modeling, Computers & Graphics 48 (2015) 84–98. doi:10.1016/j.cag.2015.02.004.

[37] D. Karavolos, A. Liapis, G. Yannakakis, A Multifaceted Surrogate Model for Search-Based Procedural Content Generation, IEEE Transactions on Games 13 (2021) 11–22. doi:10.1109/TG.2019.2931044.

[38] L. Gisslén, A. Eakins, C. Gordillo, J. Bergdahl, K. Tollmar, Adversarial Reinforcement Learning for Procedural Content Generation, in: 2021 IEEE Conference on Games (CoG), IEEE Press, 2021, p. 1–8. doi:10.1109/CoG52621.2021.9619053.