

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/388457666>

# Automation of Game Development using AI and Other Tools for AAA Video Game Production

Conference Paper · January 2025

---

CITATIONS

0

READS

61

6 authors, including:



Paul Toprac

University of Texas at Austin

34 PUBLICATIONS 466 CITATIONS

SEE PROFILE

# Automation of AAA Game Development using AI

Paul Toprac, Branden Heng, Harsheni Siddharthan, Allison Tseng, Sarah Abraham, Etienne Vouga

**Abstract**— The goal of this project was to evaluate and document the capabilities and limitations of AI tools for empowering small teams to create high budget, high profile (AAA) 3D games typically reserved for large studios. Two teams of novice game developers attempted to create two different games using AI and Unreal Engine 5.3. First, the teams evaluated 60 AI art, design, sound, and programming tools in terms of capability, ease of use, cost, and license restrictions. Then, the teams used a shortlist of 12 AI tools for game development. During this process, the following tools were found to be the most productive:

- ChatGPT 4.0 for both game and narrative conceiving and documentation.
- Dall-E 3 and OpenArt for concept art.
- Beatovent for music drafting.
- ChatGPT 4.0 and Github Copilot for generation of simple code and to complement human-made tutorials as an additional learning resource.

While current generative AI seems impressive, the assets they produce fall short of AAA industry standards. Generative AI tools are helpful when brainstorming ideas such as concept art and basic storylines, but they still cannot replace human input or creativity at this time. For coding, AI can generate simple code and act as an additional learning resource. Thus, it is recommended to use AI as a tool to enhance developer productivity rather than as a system to replace developers.

**Keywords**— AAA Games, AI, Automation Tools, Game Development

## I. INTRODUCTION

TOOLS leveraging artificial intelligence (AI) have garnered attention for their potential to empower and perhaps someday replace human centered software development. *Word (doc, docx)* versions. Integration of AI tools into game development can be revolutionary, but it remains challenging as games are complex pieces of software involving code, art, sound, and level design integrated into a coherent experience. In the gaming industry today, surveys have shown that 50% of developers use AI tools in their work, particularly in finance, community management, and production management, but not game development [1]. Although only a minority of developers use AI in content generation, the number of developers choosing to use AI will only increase as the number of available tools and their effectiveness increases. One of the common areas of AI usage include development of music, writing, and dialogue. For instance, SuckUp! [2] uses a large language model to simulate dialogue between players and non-playable characters using the player's voice input, clothing, and mannerisms. Other examples include NVIDIA's recent demo

The authors are with the Department of Computer Science at The University of Texas at Austin.

Paul Toprac (corresponding author, e-mail: topcat@cs.utexas.edu).

Branden Heng (email: branden.heng@gmail.com).

Harsheni Siddharthan (email: hsiddharthan@utexas.edu).

featuring dynamic NPCs that respond to players with generated dialogue and voices and can interact with the scene by picking up objects or navigating the world [3].

Using two teams of computer science majors with limited game development experience, the goal was to make two AAA games. The AI tools were used to accomplish this goal, along with other game development tools such as automated procedural generation software, which uses algorithms or equations to generate content like maps, items, or designs as opposed to being manually hand crafted. For example, procedural generation can be used to automatically create/generate variable cities quickly instead of creating a city manually.

The two games that were developed: *Same-Day Delivery*, a parkour action game with extensive open world levels, and *The Lab*, an atmospheric horror game with enemy AI and puzzles. During development, the development teams tried to leverage AI tools in the following areas: player interaction, text generation, 3D modeling, music, and level design. The authors documented the successes and failures of these tools, relying on non-generative techniques and easily accessible assets as fallbacks.

## II. GAME DEVELOPMENT METHODOLOGY

Both games were developed in the Unreal Engine 5.3 [4] game engine, a software program designed for video game development, with each game focused on its unique genre based needs in terms of player interaction, AI behavior, level design, 3D modeling, text generation, and music. Both games were also built using C++ to determine whether AI-generated code could be directly integrated into an existing code base in Unreal Engine 5.3. The teams investigated how readily usable and up to date this code was with the current API, and whether it could accomplish its objective without further human engineering. The areas that required the most code generation were player interaction in both games, and the inventory system in *The Lab*.

The authors also investigated AI ability to generate assets and level designs. Asset creation is one of the most time and labor intensive parts of game development. Therefore, the current limits of 3D model generation tools in both games were evaluated. The development teams determined the effectiveness of the AI generated models by evaluating the polygon-count and quality of generated assets to see whether they could feasibly be used on a large scale without being detrimental to game performance. For level design, the teams evaluated

Allison Tseng (email: allison.tseng@utexas.edu).

Sarah Abraham (email: theshark@cs.utexas.edu).

Etienne Vouga (Email: evouga@cs.utexas.edu).

Funding provided by The University of Texas at Austin's AI Pathways sponsored by the College of Natural Sciences.

whether AI could create traversable maps for *Same-Day Delivery*, and claustrophobic and atmospheric levels for *The Lab*.

Only *The Lab* used text generation for its puzzle design, and only *The Lab* implemented music and sound effects. For sound, the teams judged the mix quality of generated music by checking for the balance of tracks and the presence of undesirable properties like clipping, and realism for sound effects.

### III. ANALYSIS OF TOOLS

In January and February of 2024, the authors canvased the Internet to find the longlist of appropriate AI tools for game development. This was somewhat challenging because AI tools would be announced or updated as the search was conducted. As shown in the Appendix, 60 separate AI programs were evaluated: for the creation of art (33 AI tools), design (8), programming (12), and sound (11), with some programs being in multiple categories, e.g ChatGPT 4. Although an extensive search was conducted for the available AI tools for game development, Keyword Studios announced that they evaluated 400 AI tools for an internal project like this study called Ava [5].

Subsequently, each tool was evaluated in terms of capability, ease of use, cost, and license restrictions. From this analysis, a shortlist of twelve programs were identified, as shown in Tables I-III below:

TABLE I  
ART AND DESIGN AI TOOLS

Name	Developer Organization	Type	License	Cost	Evaluation
<b>Animate Anything</b>	Anything World Ltd.	Automatic rigging and animations	You own the models you generate	• Free • Premium	• Difficult to use • Varying quality of automatic rigging
<b>Dall-E 3 with ChatGPT 4.0</b>	OpenAI	Text to Image Generator	You own the images you generate	• \$20/month	• Easy to use web interface • Good for concept art • Limited level design
<b>Houdini PDG</b>	SideFX	3D Modeling Software	Commercial	• Free for students • Enterprise	• Highly advanced results • Complex interface • Scalable and modular
<b>Mixamo</b>	Mixamo	Automatic rigging and animations	Royalty free	• Free	• Adds smooth pre-existing animations to 3D models • Good automatic rigging
<b>OpenArt</b>	OpenAI	Text to Image, Sketch-to-image, and a stock art transformer	Commercial, with attribution	• \$10/month for 5,000 credits	• Additional features to Stable Diffusion • Custom models
<b>PCG (Procedural Content Generation)</b>	Epic Games	Procedural environment creation	Free-to-use for learning	• Free for students • Enterprise	• Excellent for prototyping • Easy to use

TABLE II  
PROGRAMMING AI TOOLS

Name	Developer Organization	Type	License	Cost	Evaluation
<b>ChatGPT 4</b>	OpenAI	LLM Generative Chatbot	MIT license	• \$20/month • \$25 for team	• Improved model from ChatGPT3.5 • Improved boilerplate code • Bad for complex interfaces
<b>Copilot</b>	GitHub	Code completion	You own the code you generate	• \$10/month	• Useful for boilerplate • Convenient interface as IDE extension

TABLE III  
AUDIO AI TOOLS

Name	Developer Organization	Type	License	Cost	Evaluation
<b>AIVA</b>	AIVA Technologies	Music Generator	Commercial Options	• Free • \$33/month	• Simple and intuitive • Includes a digital audio workspace • Highly customizable • Can build around input
<b>Beethoven</b>	Beethoven	Text-to-Music Generator	Commercial Purposes	• Free • \$20/month	• Good at sound generation • Solid ambient music • Easy customization
<b>Sound Effects</b>	Eleven Labs	Text-to-Sound, Effects Generator	Different plans for personal and commercial use	• Early Access	• Convincing sound effects • Quality is prompt dependent
<b>Soundraw</b>	Soundraw	Music Generator	Royalty free, Lifelong License	• Free • 16.99/month	• Only good for pop music • Limited list of genres

### IV. SAME-DAY DELIVERY

The game, *Same-Day Delivery*, is a platforming/parkour game. Platforming or parkour refers to jumping across, over, and around obstacles. Players play as a delivery person for a megacorporation. The goal of the game is to deliver dozens of packages within one minute in a bustling city to satisfy upper management. Players must parkour through traffic, obstacles, and rooftops to meet their quotas. Utilizing roads and sidewalks is not enough to deliver all packages, instead the player must use their parkour skills and strategic planning to find shortcuts through alleyways and rooftops.

The idea for *Same-Day Delivery* emerged without the assistance of AI. Instead, AI extended the *Same-Day Delivery* concept. ChatGPT 4.0 [6] was used to organize, refine, and format the initial idea and draft of the *Same-Day Delivery* concept into game and technical design documents. These documents specify how the game was to be created and the ideas behind certain mechanics. ChatGPT 4.0 was effective at this task as it sped up the process significantly.

The development of *Same-Day Delivery* relies on three main components. The first core component is asset creation. Here, the team explored concept art generation, generative 3D model methods to see whether they were ready for game development, including for animation. Second, the game needed strong platforming controls, such as player jumping and vaulting, and a way to deliver packages to destinations. The final and most important, the map must be both accommodating and restricting

of the player's movement to be challenging/puzzling. These were the core evaluation metrics of *Same-Day Delivery*, as this tested the capability of AI rigorously.

#### A. Concepting and Design

The first step in the development process was to generate concept art to facilitate the preproduction phase. Subsequently, level designs were generated that could be used whole-clothed, partially, or to facilitate the creation of the level design by providing inspiration. Below describes the process and outcomes of our investigation.

##### Concept Art

*Same-Day Delivery* benefited from the concept art that was generated by OpenAI's Dall-E 3 [7], and OpenArt [8]. The variety of styles from Dall-E 3 helped standardize the look and feel for *Same-Day Delivery*, as shown in Fig. 1. Generating AI art for concepts was extremely fast and allowed this team to focus on other aspects of the game. For a specific design, the best approach was to search for reference images that contain the items and keywords necessary to prompt Dall-E 3 or OpenArt. Using this method resulted in precise prompts that could be utilized repeatedly.



Fig. 1 Concept Art generated by Dall-E

##### Level Design

For *Same-Day Delivery* it was also imperative for the map to be well designed with sensible and traversable layouts to allow for compelling gameplay. The ideal map would be a tightly knit, close-quarters urban level that conforms to the movement of the player. A less than ideal level design would have flat buildings, a suburban/rural layout as these elements do not support our game requirements for platforming/parkouring.

To test AI's level design prowess, Dall-E 3 and OpenArt were used to generate concepts of towns and cities. The team immediately discovered that these tools will not be effective in

producing 3D assets or playable levels for our 3D game. The team then sought to use the AI generative art tools for a different purpose: to generate concept maps capable of being recreated in the game engine. Unfortunately, the generated concept maps were inconsistent, unusual, and lacked proper Euclidean geometry, one example can be seen in Fig. 2. While Fig. 2 was stylized and visually impressive and followed the team's design philosophy, on closer inspection there are inconsistencies in the geometries and flow of pathways. Overall, the design was difficult to follow and was unusable in the final design; however, it was useful for inspiration. That is, the designed assets matched modern, urban architecture commonly found in metropolitan areas like New York City. Ultimately, the team had to rely on hand-designed levels or content that was procedurally generated, as described below.

It was determined that intentional, deliberately designed levels would improve the feel for the game and increase enjoyability for the players. The team abandoned using AI and instead decided to use procedural generation tools to create the level. Two tools were used to mechanically generate levels: Unreal Game Engine's procedural [4] content generation and Houdini's [9] procedural data generation. Unreal's procedural content generation tool is a plugin that allows for quick and efficient prototyping for generating huge cities at desired specifications. Houdini, a 3D modeling software, has a similar tool that produces the same result but with exceptional quality. However, Houdini's learning curve is exceedingly high. Unreal's procedural content generation was selected over Houdini's procedural data generation since speedy prototypes were valued over quality at the time of development of *Same-Day Delivery*. The resulting level was found to be acceptable for early development.



Fig. 2 Concept art for a city block, generated by Dall-E 3

#### B. Content Creation

After conceiving and designing, the development team attempted to use AI tools to generate 3D art assets. For *Same-Day Delivery* it was imperative that the 3D art assets be animated so that players can see their character run, jump, and

parkour throughout the environment. In this section, the process and outcomes of using AI tools for art generation and animation are described.

### 3D Art Generation

3D assets are a crucial focus in the production of both games as asset production is one of the most time intensive aspects of 3D game production. The team performed a cursory evaluation of several generative 3D AI programs such as TripoAI [10], but found that the polygon count (polycount) of produced assets were significantly high. The team found that these programs all work in a similar way by creating point clouds from 2D images that are then transformed into a 3D asset. For 3D models, often polygons are represented as triangles. At the time of research, TripoAI was the most accessible program as it was capable of easily generating models that could be used in the game engine. The team found that TripoAI generates 3D models with fifty-thousand triangles, regardless of the simplicity or complexity of the intended artifact, as shown in Fig. 3.

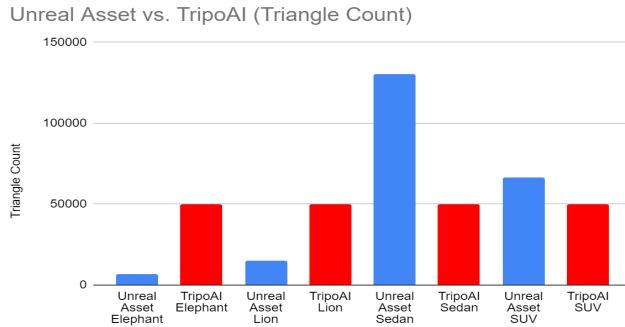


Fig. 3 Graph showing the triangle polygon count for varying models

The team was concerned that the relatively high polycount of these models would affect game performance. An experiment was conducted comparing performance between an elephant model acquired from the Unreal Marketplace [11] and a 3D generated elephant from TripoAI. The system used in this experiment was an AMD Ryzen 7 7800X3D, Radeon RX 7900 XT, 32GB DDR5-6000 CL30 desktop computer, which could be considered a high-end desktop computer. The test found the framerate or frames per second (FPS) based on the number of elephant entities instantiated in Unreal Engine 5.3 from 1 to 16,384.

There was a substantial drop in framerate with the TripoAI elephant in comparison to the elephant acquired from the Unreal Marketplace. As shown in Fig. 4, for both assets the system performed well around 120 frames per second when there are around 100 entities of each elephant asset. However, there was an immediate drop in frames when the number of elephant entities reached 250 elephant entities. The Unreal Asset Elephant, the asset with a lower polycount, continued to outperform the TripoAI Elephant from 100 elephant entities onward. Whilst generative 3D models like the TripoAI elephant are easy to create, they can reduce the overall performance of games due to their unnecessarily high polycounts. This issue would be exacerbated on lower performing systems and is not

healthy for game performance at scale.

Entity Quantities Impact on Framerate

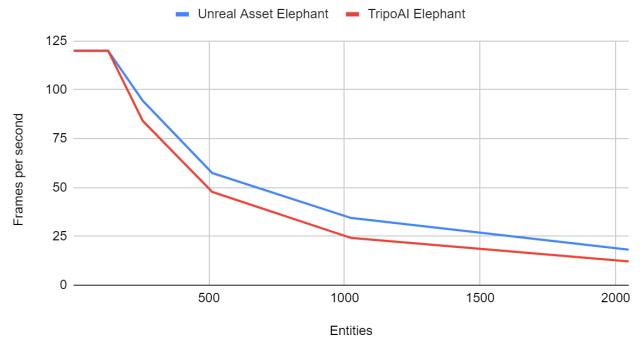
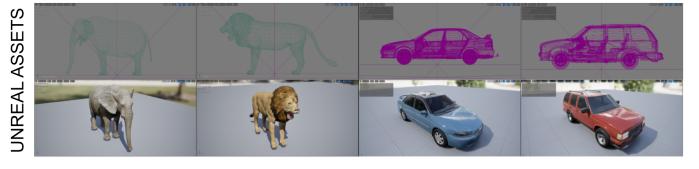


Fig. 4 Graph displaying the frames per second difference between two models of differing polycounts

In Fig. 5, while generative 3D models' aesthetics are sufficiently pleasing, they have an unnecessary number of polygons as shown in the elephant and lion comparisons. The Unreal Asset Store models contain a fraction of the polygons of the TripoAI while maintaining the look of an elephant and lion. Lower geometric models are advantages because they allow for better control of the animated model, i.e. the visual actions that models can perform, such as walking, running, and jumping.



Less dense, more geometric models are preferred.

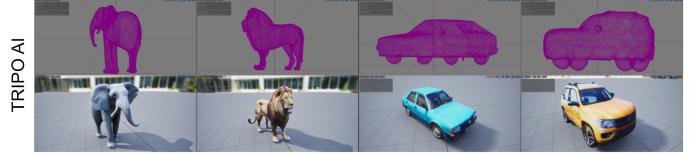


Fig. 5 Comparison of Unreal Marketplace and TripoAI models

The exception is the Unreal Engine Marketplace car model, which does have extremely high polycount compared to the TripoAI car models. This is due to the Unreal Engine Marketplace car models being designed for players to be inside of the vehicle, as seen by the highly detailed steering wheel and seats. The TripoAI models do not contain any interior detail.

Although the generative AI 3D models have excessive polygon counts that can lead to reduced game performance, it is possible for developers to remedy this by using digital art editors to decimate the models, i.e. reduce the polycount. However, manually decimating every model is time intensive and results in lower quality than opting for readily available assets. Because of this, the art pipeline was modified to instead use art assets from services such as Unreal Engine Marketplace and itch.io [12], as these assets are carefully designed with the intention of being used in video games.

In summary, simple, stylized aesthetics for the art was

desired for *Same-Day Delivery*. Current generative 3D AI tools were not capable of recreating this style and meeting AAA standards, so it was decided to use art assets from the Unreal Marketplace and itch.io for *Same-Day Delivery*.

### 3D Animations and Rigging

Since the game featured a third-person perspective of the character parkouring, high quality animations for the main character were essential. To provide the player model with animations, characters needed a “skeleton” of bones (e.g. arms, legs, neck), i.e. rig, for the 3D model. The authors found a site called Anything World’s Animate Anything [13] that purported to be a web-based AI model rigging and animations tool. Using this service, two static player models were submitted and rigged for testing: one with an A-pose, the other with a T-pose, as shown in Figs. 6 and 7. The T-posing model fared better than the A-posing model, i.e. the AI was able to place the bones more correctly for the T-pose model than the A-pose model. However, the bone placement for both models did not meet industry standards. In the end, the bones of each model had to be adjusted by hand. Unfortunately, the bones were not labeled, so it was difficult to determine what bones went where on the character. This resulted in unusual skeletons that could not be fully adjusted and were far below AAA standards. For instance, Fig. 6 shows our playable character model with slumped shoulders and arms that extend past their intended locale. Fig. 7 squishes the playable character’s head and has a lazy arm.

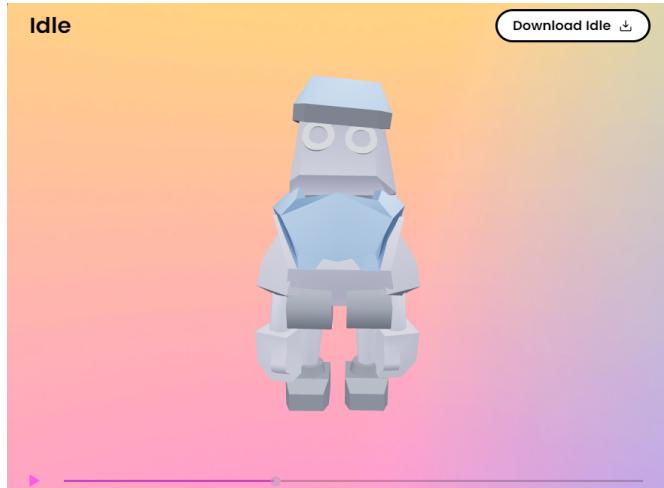


Fig. 6 The result of a A-pose model in Animate Anything



Fig. 7 The result of a T-pose model in Animate Anything

Due to the disappointing model riggings, the team decided to compare Adobe’s Mixamo [14] automatic rigging process to Animate Anything’s. Adobe Mixamo’s AutoRigger applies machine learning to determine where the limbs of a 3D model are and to insert the rig. After uploading the T-posing player model, Mixamo provided an easy, interactable interface that allowed for placement points where the chin, wrists, elbows, knees, and groin should reside as shown in Fig. 8. After rigging the player model, Mixamo offered hundreds of free animations, such as a walk cycle, that could be used for characters, though even those animations are often not up to AAA standards.

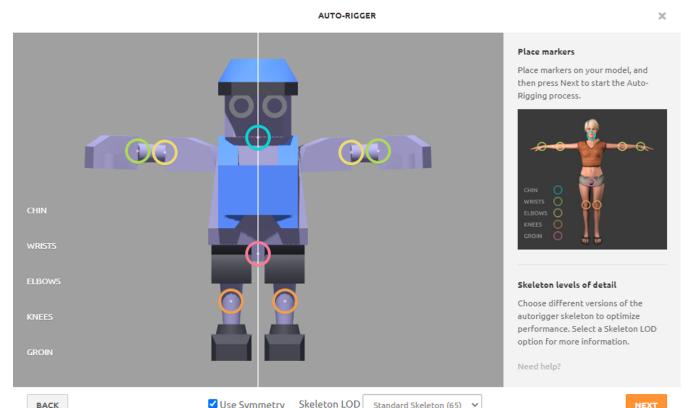


Fig. 8 Mixamo’s Auto-Rigger interface

The team found that the player model rig created by using Mixamo exceeded the quality of Animate Anything. There were significantly fewer strange artifacts in the Mixamo rigging compared to that of Animate Anything’s rigging. As shown in Figs. 8 and 9, Mixamo was able to preserve the shape of our player model, something Animate Anything was unable to do. The playable character’s torso in Fig. 8 is completely disfigured and chaotic. The playable character’s model in Fig. 9 maintains the appropriate geometry during a run cycle.



Fig. 9 Animate Anything's AI running animation

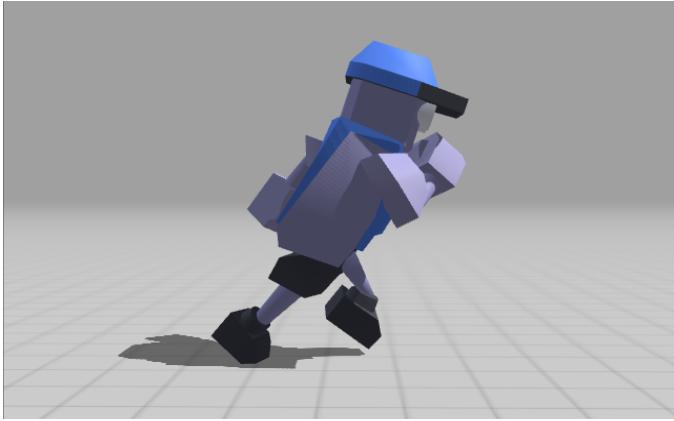


Fig. 10 Mixamo's automatic running animation

In summary, the authors found the model, skeleton, and animations of Fig. 9 (Animate Anything) to be unacceptable, whereas Mixamo provided acceptable animations, though still needing professional touch-up to meet AAA standards.

### C. Programming

To have an interactive game, it is necessary to program the gameplay and events in the games. Parkouring and platforming comprises the core gameplay of *Same-Day Delivery*. Thus, the ability to move the character was tantamount. The team attempted to create this ability by generating coding using AI.

To start, a character package was needed that contains the essentials to move, turn, and control the playable character. ChatGPT 4.0 [6] was prompted to create this package. Upon trying to implement the code provided by ChatGPT 4.0, we discovered that the code, written in C++ (a programming language used by the Unreal Engine 5.3) lacked a connection between files when trying to generate a header file and a source code file. Both files needed to share attributes, but the generated code failed to make those connections, leading to a failed compile. For inexperienced programmers, this mistake could easily be overlooked and cause problems. ChatGPT 4.0 was informed about this lack of connection, which it agreed with, and then created code that fixed this problem.

Oftentimes, ChatGPT 4.0 would default to using Blueprints, a visual coding system used in Unreal Engine [4]. The team tried to avoid using Blueprints, as Blueprints can be under-

performative especially for AAA games compared to C++. To obtain C++ code, the team had to specifically ask ChatGPT 4.0 to generate C++ code rather than Blueprints, sometimes even asking it to convert Blueprints to C++ code.

Once a simple character package was created through several attempts, the team tested the character package in-game. Unfortunately, ChatGPT 4.0 failed to initially provide information about creating a gamemode, which was necessary to test the character package. It was necessary to ask ChatGPT 4.0 to provide this information, a detail novice developers could miss.

The team found the character package to be sufficient after some testing and debugging; however, the time to create the character package from ChatGPT 4.0 took significantly more time compared to using human-made tutorials. ChatGPT 4.0's code required debugging, as the code would not work occasionally when inputted into Unreal Engine 5.3. For a non-programmer, this issue would take significantly more time to fix. The team decided that using human-made tutorials to acquire the appropriate code to be more reliable and required less time overall. In addition, the tutorials served as learning tools that facilitated the development of the game.

The team found that ChatGPT 4.0 was useful for inquiring about complex topics that some programmers could find difficult to understand. ChatGPT 4.0 could explain logic and code to coders with a wide range of understanding. For example, ChatGPT 4.0 often gave more simple and easy-to-understand answers when the phrase “explain [topic] to me like I’m five” was included in the prompt, as shown in Fig. 11. This is quite useful when the developer is relatively new to a topic. However, on default, ChatGPT 4.0 answers assume prior knowledge about the topic of discussion and give answers that force the developer to fill in gaps in logic.

**You**  
Can you explain video-game shaders to me like I'm five?

**ChatGPT**  
Sure! Imagine you have a coloring book with lots of different pictures. You can use crayons to color them in any way you like, right? In video games, shaders are like the crayons that color and light up everything you see on the screen.

There are different types of shaders. One type works like the sun, making things look bright or casting shadows. Another type is like putting stickers on your drawings to add details like shiny windows or bumpy roads. And some shaders are like filters on a camera, changing how everything looks by making it brighter, darker, or even like it's underwater!

So, shaders help make the video game worlds look real and interesting, just like how your crayons help make your coloring book come to life!

Fig. 11 ChatGPT explanation of complex topic at low knowledge level

## V. THE LAB

The horror game, *The Lab*, focuses on building a complex narrative through layered puzzles within the immersive environment. The story revolves around a faceless and nameless researcher who finds themselves in a 90's era research lab trying to unravel the mysteries of who they are and what is happening around them. The player must explore the abandoned lab while fending off strange beings that seem to be

connected to the new technology that was being developed there.

Ambiance is a very crucial part of a horror game as it sets the tense mood and enhances immersion. Likewise, having a compelling storyline along with an immersive environment allows for a suspenseful and eerie psychological horror game. As such, the grounds for evaluation of generative tools are unlike those of *Same-Day Delivery* due to the different requirements of a more narrative than mechanics focused game.

The core gameplay centers around navigating a maze-like environment to look for clues while trying to avoid monsters. Generative techniques were utilized across all categories with evaluation focusing on their adeptness in producing atmospheric elements that carry the narrative forward. Therefore, a lot of the focus was on sound design and text generation as they are especially important elements in building an immersive horror game experience.

For generated text, the team wanted to determine whether AI tools could follow a given storyline and produce clues such as lab documents, email logs, and text messages, etc. These clues would be dispersed in such a way as to make them seem more realistic to the setting and allow the player to feel like they are piecing together the story as they progress. As such, producing these different clues require different tone and style all while following the storyline through different threads.

For generated sound, the team wanted to determine whether AI could generate convincing background music and sound effects. Background music contributes greatly to the overall mood of the game and should be used to influence the player's feelings as they progress through the story. Sound effects also contribute greatly to the immersiveness of games and help give players a sense of reality. For this reason, sounds that evoke a feeling of anxiousness and fear were necessary for the success of the game.

Therefore, our main goal for this game involved using generative AI to build immersive, maze-like environments and to program key game elements like password puzzles and an inventory system to collect clues.

#### A. Concepting and Design

The concept for *The Lab* originally came from asking ChatGPT 4.0 [6] for game ideas. ChatGPT 4.0 recommended making a horror game, and upon further prompting, pushed us towards the idea of the player waking up in what seems to be a terrifying dreamscape and being chased by monsters around a maze. From there, the idea was refined through a combination of the team's ideas and generative AI to create the fleshed-out storyline and game design. Below is a description of the process and results for using generative AI in this project.

##### Concept Art

Upon deciding the main story for the horror game, the team had to determine its overall visual aesthetic. When brainstorming for concept art, Dall-E 3 [7] showed promise, particularly for level and character design, because of its ability to generate eerie imagery that dips into the "uncanny valley." The uncanny valley is a term coined by Masahiro Mori of the

Tokyo Institute of Technology, which describes "a feeling of strangeness, a sense of unease, and a tendency to be scared or freaked out" by robots that aim to appear human [15]. This same phenomenon can be observed in AI generated imagery as they seem normal at first glance, but reveal unnatural or warped elements upon further inspection.

As shown in Fig. 12, when prompted to create an unsettling 90's era computer lab with many wires, Dall-E 3 provided concept art with a grotesque amalgamation of cables, wires, and computer screens in a somewhat normal office space. Although the image may appear rather realistic upon first glance, the overly rendered smoothness of all the objects contribute to the disturbing, uncanny valley feel of the scene, a characteristic of many AI generated images.



Fig. 12 Concept art for the horror game, generated by Dall-E 3

With *The Lab* focusing on the dangers of reckless advancements in technology, the unsettling features of AI generated concept art contributed well to the overarching theme and provided a good idea on how to begin building spaces and assets within the game.

##### Level Design

One of the key game mechanics of the horror game involves an enemy AI that chases after the player, adding a sense of urgency as they try to solve the many puzzles of the level. To further elevate this experience, the office is organized like a maze with countless winding corridors and similar looking scenery. When prompted for a "maze-like floor plan" based on the generated concept art, Dall-E supplied top down floor plans for single story buildings that were on theme with previously generated concept art. However, while it was able to incorporate key design elements like maze-like hallways and computer labs, it did not maximize the given space as it included many completely enclosed sections that could have been used as more rooms or hallways, as seen in Fig. 13.

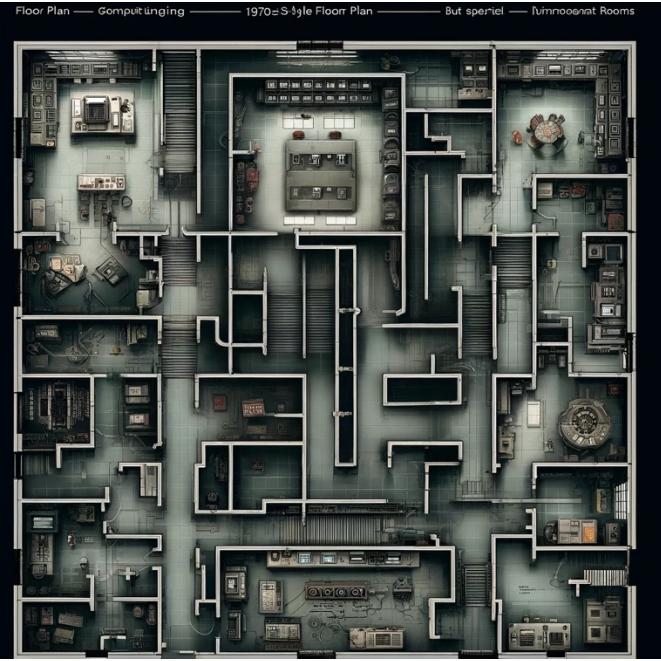


Fig. 13 Level concept art for the horror game, generated by Dall-E 3

Dall-E's layout also tended to include stairs that went to nowhere, even when prompted to switch to a top-down view or explicitly told not to. These issues were due to Dall-E 3 lacking a comprehensive sense of spatial awareness [16]. Although the labyrinthine nature of the AI generated floor plans may appear fitting for the level design of the game, making the environment traversable was of bigger concern, so these layouts were scrapped in favor of human designed ones and the AI generated level design was used as concept art instead.

For these reasons, the team found AI generated designs were helpful in providing a general schema rather than a detailed level design, as they lack the spatial awareness to provide intentional and strategic designs.

### B. Content Creation

#### 3D Art Generation

With the game taking place in an abandoned lab, many of the necessary assets were office furniture and technology like desks, computers, cables, servers, etc. The level design involved a large computer lab with winding corridors of servers that can have upwards of 100 computers in them. However, as Figs. 14 and 15 demonstrate, the polygon count of TripoAI [10] generated models is much higher than a comparable, human-made asset, which can severely decrease the performance of the game, as discussed previously in the *Same-Day Delivery* section. Mesh decimation could solve this problem, especially with objects containing a lot of flat surfaces like computers, but for better optimized and visually appealing mesh, human input is currently necessary. Also, as discussed previously, decimation takes additional time.

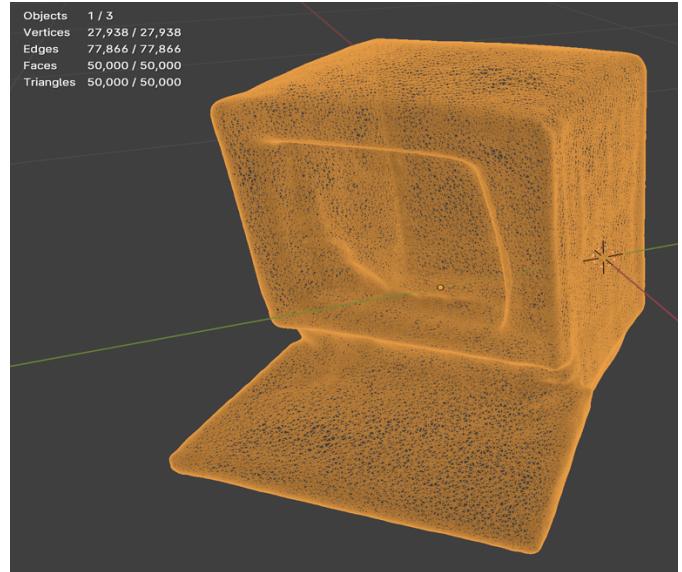


Fig. 14 TripoAI generated model of 90's era computer

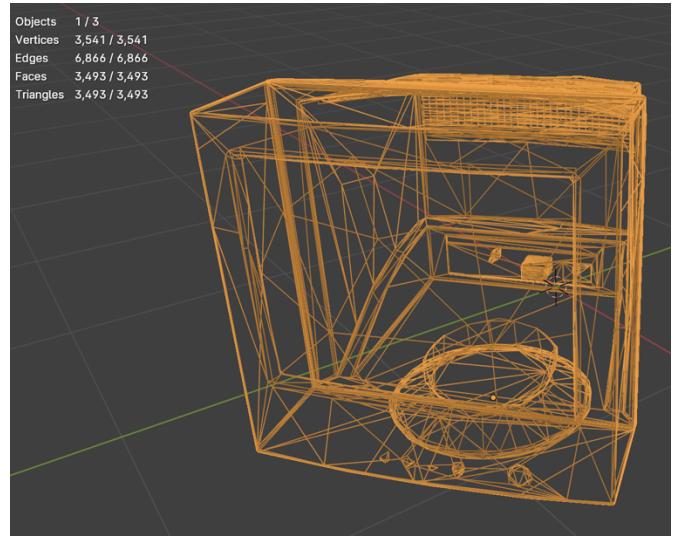


Fig. 15 Free asset from the Unreal Marketplace of a PC

Apart from possible performance issues, generated models also had warped proportions and irregular shapes, as shown in Figs. 16 and 17, that do not meet the visual standards of AAA games. Evident in the TripoAI generated model, key components like the keyboard are deformed and not true to life, rendering the model visually unappealing and unusable. As such, it was determined that AI generated 3D models to be unsuitable for making AAA games.



Fig.16 TripoAI generated 3D model requires extensive artist modifications to be usable (see keyboard)



Fig. 17 Free asset from the Unreal Marketplace without modification

### *Sound Design*

With the visual aesthetic set, the next task was determining the audio aesthetic of *The Lab*. Due to the technology driven themes of the horror game, a primarily electronic soundtrack was appropriate. The team tested three music generation programs: AIVA [17], Beatoven [18], and Soundraw [18], as well as one sound effect generation program - Eleven Labs [20].

Initial testing revealed that the music generator Soundraw was not a good fit for this project since it could only generate music based on a limited list of genres, most of which had a very pop based feeling, which did not align with the goals of the horror game soundtrack. Additionally, Soundraw generated music can only be downloaded as a singular .wav or .mp3 file, which makes additional editing difficult.

AIVA is another music generation tool that claims to be able to generate music based on a style from their preset style list or a given midi file with a chord progression or musical influence.

A midi file only stores what notes are being played in a piece, making it a useful tool for composing individual tracks within a larger piece. Considering that the midi files used in most professionally produced songs are not available to the public, AIVA's music generation features would require that the user already knows how to compose a midi file or knows how to transcribe a song they have heard onto a midi file. Both these cases would require the user to have prior musical experience, making it difficult for the average developer to use AIVA. AIVA also includes a simple in-built digital audio workstation (DAW) that could be used for increased customization, but many of the tools it includes would likewise require prior musical production knowledge.

Text to music generators like Beatoven were more effective in creating atmospheric music that could contribute to the unsettling mood of the game because text prompts could be easily edited to produce specific types of music beyond the mainstream. Beatoven generates one-minute-long pieces that are split into multiple instrumental tracks. The program allows users to regenerate specific parts of a piece and manually add volume automation. Although pieces are only minimally mixed and mastered, individual tracks can be downloaded and put into digital audio workspaces for further fine tuning. These tools are more accessible to novices who have little to no experience with music production as they quickly produce pieces of music.

Although these AI tools created mixes with no glaring problems, such as clipping, the generated music often sounded somewhat dynamically flat or repetitive. However, balance between separate tracks could be improved through further fine tuning of volume, panning, etc. using a professional digital audio workstation.

In summary, AI generated music might be acceptable for some games, as is, but needs more human manipulation to be suitable for a AAA game. That said, the process of using AI generated music and then manipulating it to achieve the desired results may be much faster than manually composing the music from scratch.

The team also tested Eleven Lab's latest text to sound effect generation tool. Upon prompting, Eleven Labs takes a few seconds to produce four clips approximately one to three seconds in length. The clips vary in quality as some sound rather clear while others sound static or electronic. Although this may be useful in producing specific sounds that are not easily recorded or manually produced, free online resources like freesound.org can find hundreds of copyright free sounds for use at the click of a button. In addition, it is difficult for AI generated sounds to sound as natural as actual recordings or professionally produced. Thus, in summary, free online resources for sound effects were preferable to the use of Eleven Lab's AI generated sound effects.

### *Text Generation for Storytelling*

Much of the immersion of the game relies on how the overarching story is presented. As *The Lab* is just as much a puzzle solving game as it is a horror game it was necessary to design reasonable clue-based puzzles that could allow players to gradually unlock parts of the story as they play. In the game,

text documents were hidden within computer objects that could be collected using the player's inventory system. Documents varied in format from emails to research notes to text messages with the player needing to piece together to learn both the clues and the overarching story.

To generate these text documents, an outline of the main story was fed into ChatGPT 4.0 [6] and asked to "write a clue note for the given game concept." As shown in Fig. 18, ChatGPT 4.0's clues tended to directly reword the entire plot rather than taking pieces of the plot to use as a singular clue. Also, AI generated clues defaulted to writing in an overly formal tone with lots of unnecessary flowery wording that slightly muddled the overall message of the clues. However, when asked for email correspondence between two concerned authors, ChatGPT 4.0 was able to convey clues using a formal tone without the excessive use of florid language.

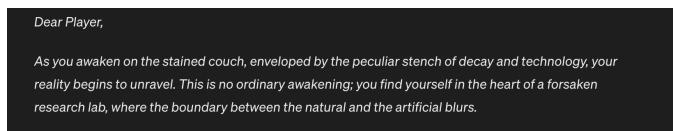


Fig. 18 Story clue generated by ChatGPT

ChatGPT 4.0 was also prompted to generate text messages for a more casual interaction between close friends. The tone and language of the generated content contained modern slang, emojis, and the more relaxed structure that texts tend to have, as Fig. 19 shows. However, some messages sounded slightly awkward and lacked conversational flow, especially with embed clues.

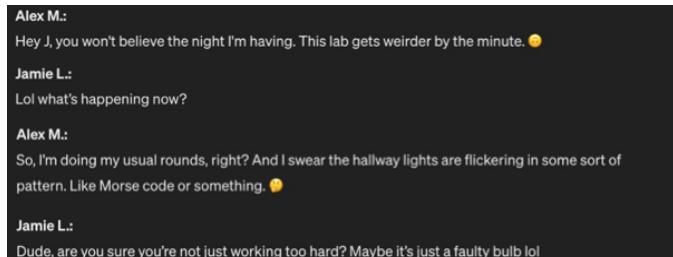


Fig. 19 Story clue generated by ChatGPT

In summary, ChatGPT 4.0 generally just regurgitated the narrative it was given. Rather than a summary of the given narrative, the team wanted it to provide pieces of the plot that could be pieced together for one overarching puzzle, and it largely failed in this aspect. That said, in some limited cases like text messages between close friends ChatGPT 4.0 was more successful.

### C. Programming

At the start of the project, ChatGPT 4.0 [6] gave effective information on how to set up an Unreal [4] project using Blueprints. However, much of the instructions largely mimicked what could be found within the Unreal Engine documentation. On the other hand, the setup for coding in C++ did not run as smoothly. The initial code it provided was full of bugs, and ChatGPT 4.0 was unhelpful in debugging this code.

To use ChatGPT 4.0 to debug, the user must have prior

knowledge on the issue they are asking about, because simply asking "what is wrong with the code" will only provide regurgitated prior answers or altogether faulty logic. Part of this gap in knowledge can also be attributed to the staleness of AI model versions compared with the more frequent release of new game engine versions. For this reason, ChatGPT 4.0 can sometimes be outdated and less useful than official documentation or web forums. Oftentimes, generated instructions can even lead users to settings that have been moved or completely deprecated in newer versions. However, ChatGPT 4.0 was useful in understanding some of the basic C++ code libraries and the semantics of the version of C++ used in Unreal as these tend to change at a slower rate than the Unreal Engine 5.3 editor.

Regarding creating the player package with Blueprints, ChatGPT 4.0's instructions were concise and effective. As shown in Figs. 20 and 21, ChatGPT 4.0 explained input bindings almost identically to those in the Unreal Engine 5.3 documentation. ChatGPT 4.0 apparently was trained on what nodes are usable in Unreal Engine, so it was able to accurately pinpoint the nodes that had to be used and how to connect them. The main difference between the two is that the Unreal Engine 5.3 documentation included screenshots that visually show how nodes should be connected while ChatGPT 4.0 only describes these connections verbally as shown in Figs. 22 and 23.

2. In the Bindings Category click the + sign next to Axis Mappings to add the following Axis Mappings.		
Axis Mapping	Key Value	Scale
MoveForward	W	1.0
	S	-1.0
MoveRight	D	1.0
	A	-1.0
Turn	Mouse X	1.0
LookUp	Mouse Y	-1.0

Fig. 20 Unreal Engine 5.3 documentation for setting up the player package with Blueprints

- **Move Forward/Backward:** Create a new Axis Mapping called 'MoveForward'. Add two entries under it:
  - Key: 'W', Scale: '1.0' (moves forward)
  - Key: 'S', Scale: '-1.0' (moves backward)
- **Move Right/Left:** Create a new Axis Mapping called 'MoveRight'. Add two entries:
  - Key: 'D', Scale: '1.0' (moves right)
  - Key: 'A', Scale: '-1.0' (moves left)
- **Look Up/Down (Mouse):** Create an Axis Mapping called 'LookUp'. Bind it to 'Mouse Y'. You might want to invert the Scale value depending on your preferences for mouse look direction.
- **Turn Right/Left (Mouse):** Create an Axis Mapping called 'Turn'. Bind it to 'Mouse X'.

Fig. 21 ChatGPT 4.0 instructions for setting up player package using Blueprints

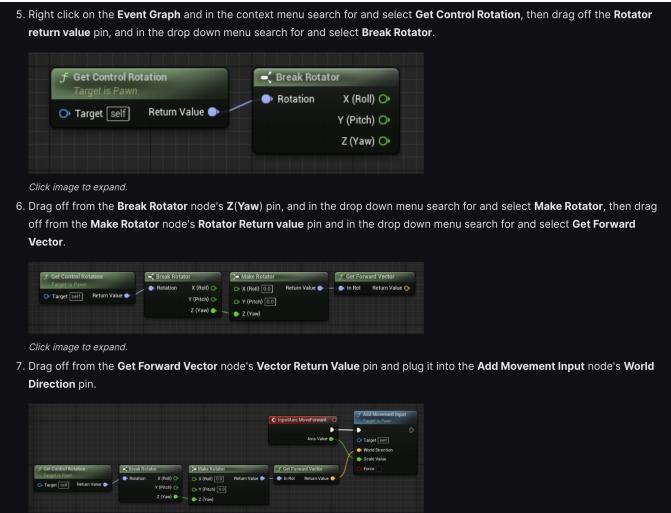


Fig. 22 Unreal Engine 5.3 documentation for setting up character movement with Blueprints

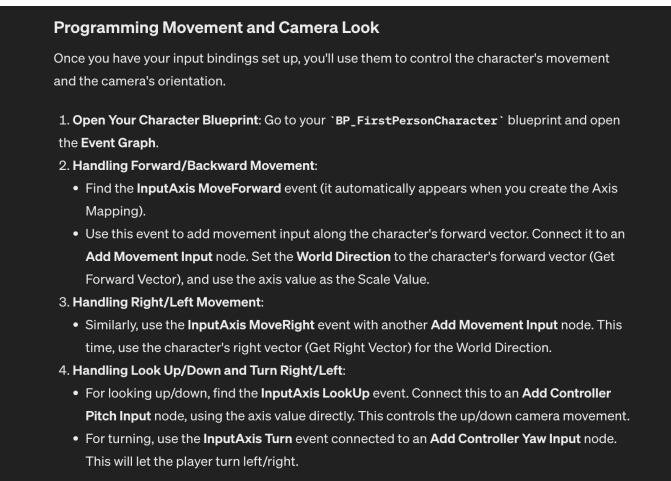


Fig. 23 ChatGPT 4.0 instructions for setting up player movement using Blueprints

However, one notable failure of ChatGPT 4.0's player package instructions was the lack of reference to character rotation, a rather important part of the player package that is always touched upon in human-made tutorials. Although not much of a problem when making a first-person game, ChatGPT 4.0 does not provide much useful information about setting up character animation as it does not know what particular character models and animations are being used. Traditional tutorials typically teach novices how to implement character animation by giving them an existing set of animations including walking, running, and or attacking, and showing how to put this all in a Blend Space. Blend Spaces are assets that blend multiple animations. For example, Blend Spaces could be used to make the character go from walking to running as they speed up. ChatGPT 4.0 cannot provide premade animations or visually show how to put animations into a Blend Space. Learning how to do an extremely visually based task like configuring character animation with only verbal instructions is very difficult for novices that are unfamiliar with how the many components within Unreal Engine fit together. ChatGPT 4.0

fell short due to lacking visual guidance and structured instructions, something traditional tutorials performed well.

For C++ code generation, ChatGPT 4.0 was able to mimic template code found in starter projects in Unreal Engine 5.3 and gave instructions on how to set up the key bindings as well. Copilot [21], which utilizes ChatGPT 4.0 and publicly available code, was also used as a code generation tool starting with the player package and provided comparable results to ChatGPT 4.0. The framework of both code segments closely resembles each other except for the new Enhanced Input functionality of Unreal Engine 5.3 that changes some of the syntax for key bindings. ChatGPT 4.0 and Copilot were a little out of date with the new functionality, but this was easily supplemented by human-created tutorials and the official documentation regarding this update.

The main downside of following the AI generated instructions was that ChatGPT 4.0 could not include screenshots taken from Unreal Engine 5.3 to show one how to find certain buttons to select like most human-made tutorials do. ChatGPT 4.0's greatest strength at this point in the project was its ability to explain why certain steps had to be taken. Some readily available human-made tutorials on YouTube and the like often only listed the steps to achieve a certain goal, without going into detail over why such steps had to be taken. While learners cannot ask for immediate clarification while watching video tutorials, they can always ask ChatGPT 4.0 the exact reasoning for why they had to do certain things for their Blueprints to work.

The next major component in creating *The Lab* was the inventory system, as most of the storyline is told through various types of research documentation, emails, photographs, notes, writeups, etc., that the player would collect as they progress through the levels. Once again, both Blueprints and C++ versions were tested using ChatGPT 4.0 and Copilot, with both yielding similar results. As Fig. 24 shows, initial querying was only able to provide boilerplate code, but the C++ code was missing an interface component that allows for better integration of all the different elements of the inventory system. Most human-made tutorials recommend including this interface component to avoid repetitive code when implementing the inventory system, so by excluding this in its instructions, ChatGPT leaves users to do more work and create less organized code.

```

#pragma once

#include "CoreMinimal.h"
#include "Components/ActorComponent.h"
#include "InventoryTypes.h"
#include "InventoryComponent.generated.h"

UCLASS( ClassGroup=(Custom), meta=(BlueprintSpawnableComponent) )
class YOURGAME_API UInventoryComponent : public UActorComponent
{
    GENERATED_BODY()

public:
    UInventoryComponent();

    UFUNCTION(BlueprintCallable, Category = "Inventory")
    void AddItem(const FInventoryItem& Item);

    UFUNCTION(BlueprintCallable, Category = "Inventory")
    void RemoveItem(const FInventoryItem& Item);

private:
    TArray<FInventoryItem> Inventory;
};

```

Fig. 24 Code provided by ChatGPT for an inventory system.

To improve the inventory system, simply asking ChatGPT 4.0 to elaborate on previous prompts was not enough and instead required having prior knowledge on how to implement an inventory system to directly nudge ChatGPT 4.0 to the right direction for generating code. The caveat to this is that even when asking about possible holes in ChatGPT 4.0's logic, it tends to apologize and agree with the user without explaining why the user is correct. ChatGPT 4.0 reacts in this manner even if the user's words are not correct, making it difficult to determine whether its explanations in these cases are trustworthy or not.

One example of this was when asked how to check if an item is in range to be picked up by the player. ChatGPT 4.0's provided solution was to calculate the distance between the player character and all the items in the scene to see if any item was available for pickup, as shown Fig. 25. However, this can easily lead to excessive computation and impact performance in a large-scale game, which is why collisions are preferred as a tool optimized for the same function. By putting a collision box the size of the range around an object, the player will be prompted to pick up the item if they simply collide with the collision box. Likewise, without further prompting, ChatGPT 4.0 solution did not check whether the player was facing the item. Thus, ChatGPT provided sub-optimal code.

#### You

how would we check if a user is close enough to pick up an item

#### ChatGPT

To check if a player character is close enough to pick up an item in the game, you typically calculate the distance between the player character and the item. If the distance is within a certain range, you consider the player to be close enough to pick up the item.

In this example:

- We use `UGameplayStatics::GetAllActorsOfClass` to find all instances of the `AItem` class (or whatever class represents your items) in the game world.
- We then iterate over the found actors and calculate the distance between the player character and each item using `FVector::Distance`.
- If the distance to the item is within the `PickupDistance` threshold, we call the `PickupItem` function to pick up the item.

You can call the `CheckItemPickup` function periodically or in response to specific events (such as player input or collision events) to check if the player is close enough to pick up items. Adjust the `PickupDistance` value according to your game's requirements.

Fig. 25 Solution provided by ChatGPT to check for possible pickups in a scene.

Copilot worked well in debugging small issues in existing code, one file at a time, like missing headers or small logic errors. Copilot was also nice about following user set guidelines like naming conventions in new functions or variables it provided.

In summary, it was more useful to turn towards human-made tutorials supplemented by generative AI, as ChatGPT 4.0 and Copilot were only able to provide a starting point for more complex systems and required prior knowledge for further assistance that novices may not have. In terms of code generation, ChatGPT 4.0 and Copilot proved effective when used in conjunction with human-made tutorials. The tutorials provided overall structure and information on breaking apart features, which is helpful for beginners working with the Unreal Engine. ChatGPT 4.0 answered specific questions about how Unreal works and basic coding implementations. Copilot effectively supplemented user-written code by debugging small issues such as missing headers or logic errors and adhered to user guidelines, including naming conventions for new functions and variables.

Regarding enemy AI, the enemy monster character in the game has two main activities: randomly roaming around the level and chasing after the player once they enter its line of sight. As the enemy AI did not require many special functions, the team opted to use Unreal Blueprints with no behavior trees to keep things simple. Upon prompting, ChatGPT 4.0 provided a simple explanation of how to set up the character Blueprint and AI Controller for the enemy AI. Although there is nothing inherently wrong about doing this, many tutorials on the topic demonstrate that it is possible to configure AI movement in a simpler fashion: using functions like the AIMoveTo node within the enemy AI's character Blueprint, which is something that may be easier for beginners to follow. Thus, as found previously, existing tutorials can be more useful than using ChatGPT 4.0.

ChatGPT 4.0 instructions up to this point may have not been optimal but were reasonable; however, to make the enemy AI usable, it must include character animations for walking around

the scene and in pursuit of the player. Because ChatGPT 4.0 cannot show any visual examples on how to create an animation Blueprint or anything of that nature, novice game developers would likely opt to follow video tutorials or Unreal Engine 5.3 documentation so they can see exactly what they should be doing. As such, generative AI would not be advisable to use when working with character classes where animation (or any requiring visual context) is key to its success.

In summary, ChatGPT 4.0 failed to be more useful than existing human-made tutorials for the creation of enemy AI behavior.

## VI. SUMMARY OF RESULTS

Generative AI tools are helpful when brainstorming ideas such as concept art, level design, and basic storylines, but they cannot replace human input or creativity at this time. The authors also found that, while generative technologies have improved greatly with the advent of deep learning, the assets they produced for AAA games fall short of industry standards. That said, the sound assets that were generated were closer to professional quality than 3D art and animation. With regards to programming, generated code still falls far below industry standards as it is riddled with bugs and lacks strong logic when implementing more complex game mechanics. The rest of the section describes in detail the summary of results.

### A. Conceiving and Design

The authors found ChatGPT 4 [6] to be helpful at compiling, progressing, and refining ideas while brainstorming. *Same-Day Delivery* utilized this ability of ChatGPT 4 to create and refine its game design and technical design documents. *The Lab* was originally conceived from a list of game ideas generated by ChatGPT 4 and then expanded upon manually.

The authors concluded that Dall-E [7] was useful for concept art for both games. For *Same-Day Delivery*, it was beneficial for creating compelling concept art generated by Dall-E 3 and OpenArt [8]. And, particularly for *The Lab*, the inherent ability of Dall-E 3 to generate art in the “uncanny valley” [15] worked well with the concept of a horror game.

However, the AI generated level designs did not work well at creating traversable in-game maps. Due to a lack of spatial awareness, AI generated levels were nonsensical designs without logical use of space. Designs contained inconsistencies in the geometry and the flow of pathways, and were only useful as inspiration for human designs. So, the *Same-Day Delivery* team ultimately had to rely on hand-designed levels or procedurally generated content using Unreal Game Engine’s [10]. Although AI generated maps were also unusable for the same reasons in *The Lab*, they provided inspiration for the mood of the level map.

### B. Content Creation

In assessing 3D art and animation AI generation tools, such as TripoAI [10] and Animate Anything [12], the authors found them unable to produce visually appealing assets in comparison to human-made counterparts and requiring extensive human intervention to even make them usable. The polygon count of

TripoAI generated assets was too high and required substantial additional post-processing by the developer to even begin to compete with free assets. So, instead readily available free assets from the Unreal Marketplace [11] and itch.io [12] was used. Similarly, Animate Anything provided very poor animations in comparison to Mixamo [14], which was used instead.

The authors found that AI generated texts using ChatGPT 4.0 could be used for in-game clues because the overly formal tone of its wording fit the uneasy tone of the horror game. However, these AI generated clues generally just regurgitated the game lore that was imputed into it, leaving little room for inductive reasoning on the player’s part as to what the game could be about, i.e. the clues were too heavy-handed.

Generative AI was successful in generating music as they created palatable pieces that achieved their aimed purpose but not for generating sound effects. AI generated adequately mixed pieces with no glaring production issues but lagged human creations in their ability to keep audiences engaged through compelling use of form and other musical techniques. For sound, Beethoven [18] was used to generate background music because this text to music generator is competent at creating unstructured, ambient music, which fit the needs of the horror game. Comparatively, tools like AIVA [17] and Sounddraw [19], which generate music based on a set list of genres are more limited in the music they can create. For sound effects, pre-made sound effects were used rather than any AI generated ones from programs like Eleven Labs [20] because recordings and professionally produced sound effects are more realistic.

### C. Programming

In terms of code generation, ChatGPT 4.0 and Copilot proved effective when used in conjunction with human-made tutorials. The AI generated tutorials provided overall structure and the high-level logic behind most in-game features, which is helpful for beginners working with Unreal Game Engine. ChatGPT 4.0 was useful as it can answer specific questions about how Unreal works and basic coding implementations, while Copilot is able to pull on public repositories to provide a solid framework of starter code to build off and debug existing code.

AI generated code succeeded in producing simple or boilerplate code, which was helpful for the player package, but fell short when prompted for more complex game components such as AI behavior. In these cases, generative AI tended to output overcomplicated code with various holes in logic and often contained many bugs.

## VII. RECOMMENDATIONS AND BEST PRACTICES

The first challenge any game developer will face is the determination of what AI tools to use. The authors evaluated 60 tools, which was daunting; however, Dinsdale [5] reported on an industry project that reviewed 400 AI tools. At this time, there is constant barrage of new AI tools being developed and released that could be used for game development. Furthermore, these tools appear to be in a constant state of evolution. Although determining which AI tools to use is

probably the most important step, the authors do not have recommendations about how to do this currently. Perhaps there will be a culling of the number of tools available in the future, which would make the job of the game developer easier. In the meantime, the game developer has either the sub-optimal choice of selecting a subset of the AI tools available for evaluation or the herculean task of evaluating hundreds of AI programs. And, even then, the process will need to be repeated on a somewhat continuous basis due to the constant evolution in AI tool technology and effectiveness.

#### A. Concepting and Design

For early stage concepting, the authors recommend using generative AI as an aid for brainstorming ideas such as mood, color palette, and art direction. However, the visual inaccuracies of AI generated images for design does not lend well for late stage concepting. Currently, the authors recommend using tools such as ChatGPT 4.0, Firefly [22], Gencraft [23], and Midjourney [24] for early-stage concepts. However, the authors recommend that professionals use the AI generated concept artifacts as the initial iteration and not as the final product.

The authors also recommend using AI to collect, brainstorm, and organize thoughts. AI chatbots like ChatGPT 4 [6] can accelerate traditional brainstorming methods. AI large language models take abstract ideas and turn them into points that users can build upon. ChatGPT 4.0 was used to come up with game ideas, which is how *The Lab* was first initialized. *Same-Day Delivery* was an initial idea that was successfully extended and organized by ChatGPT 4.0. This can be an effective use-case for developers who find the appropriate verbiage to describe their thoughts and ideas to be time consuming.

For level design, the authors do not recommend using generative AI. Generative AI like Dall-E [7] lacks spatial awareness, leading to nonsensical maps that poorly utilize available space, resulting in layouts that are neither logical nor easily navigable. Thus, the authors recommend using AI generated level designs during the concepting phase for inspiration, but the creation of level designs should be hand crafted. The exception is the use of automation tools like Unreal Game Engine's procedural content generation and Houdini's [9] procedural data generation to create the initial level and then handcrafting to make them more usable.

#### B. Content Creation

The authors do not recommend using AI generated 3D art assets because of their low quality. Instead, the authors recommend obtaining premade assets from the Unreal Marketplace or other online asset libraries. The authors recommend professionally crafting these existing premade assets to fit the needs of the project. Alternatively, make the 3D assets from scratch created by professionals. For animation, the authors recommend using Mixamo [13] or something similar rather than using pure AI tools and professionally refining them. Or, rigging and animating them from scratch.

Generated music using AI programs such as Beatoven [18] could be used in game soundtracks if a team lacks a skilled

music producer and does not need AAA quality music because they create adequately mixed music and at a much faster rate than humans can. However, the authors do not recommend using AI generated sound effects because free sound effects are readily available online and tend to be of much higher quality than whatever can be generated by AI. Like 3D art assets, the authors also recommend that professionals craft AI generated music and free sound effects to the needs of the project or create them from scratch.

Overall, the recommendation is for professionals to use these AI tools to generate initial assets that need to be modified accordingly or used for prototyping but not the final product.

#### C. Programming

The authors recommend using AI generative code in tandem with user-generated code by experienced programmers. AI generated code can be wildly difficult to interpret and work with if users are inexperienced with programming. Having programmers who are knowledgeable of the AI-generated code and its applications can accelerate the programming process because AI can generate simple snippets of code. It is highly recommended to learn the basics of programming and the tools before prompting generative AI tools like ChatGPT 4.0 or Copilot [21] for code. New and inexperienced programmers can utilize AI chatbots as a learning tool to understand programming; however, following a structured tutorial created by humans for creating the code is recommended.

The authors feel it is best to be overly cautious when working with the information these AIs provide users. There is a chance the information is incorrect. Many minor errors while using AI generated code were experienced.

Overall, as with the other recommendations, AI code and instructions can be the initial kernel of work, but professionals are needed to complete the job. That said, ChatGPT 4.0 can provide additional instructional material beyond human-made tutorials.

### VIII. STUDY LIMITATION AND FUTURE RESEARCH

There are several limitations that affect the validity of our study. For instance, because of budget and time limitations, the number of tools that we were able to test was not comprehensive. Dinsdale [5] reported on a project performed by Keyword Studios, called Ava, where they attempted to build an AAA game using only AI. Keywords reviewed 400 AI tools as a part of their efforts to complete this project. In comparison, the research team was only able to analyze 60 AI programs. Thus, a tool that would have been highly productive for making the games may have been overlooked. That said, Keywords Studios also found that current AI tools cannot replace humans for game development. Also, AI tools are rapidly evolving and so this study's results will probably be rapidly invalid. This implies that a research agenda that continuously explores the use of AI in game development is necessary to keep our understanding of this space up to date.

In addition, this study is not a formal experiment—the two teams did not create games with and without AI for comparison. Rather, the games were developed with AI tools to explore and

learn how to use them. Because of this, the authors are unable to definitively ascertain the degree to which AI was or was not useful in game development. More work can be done to explore how using AI could speed up the pipeline and increase productivity, and perhaps creativity.

Furthermore, only two 3D games were developed for this study: one horror game and one parkour game. There could exist other game genres more conducive for AI and procedurally generated game content and programming. For instance, 2D video games would seem to be a better fit for using AI based. More research developing games in other genres is warranted.

Lastly, Unreal Engine 5.3 [4] was used as the game engine. Perhaps better results could be obtained using another game engine such as Godot [25] and Unity [26]. More research can be done using these different tools to determine whether different results can be obtained.

## IX. CONCLUSION

Recent AI technology has improved greatly, but based on this study, AI alone does not produce results that meet the expectations for an AAA game. Generative AI tools are helpful when brainstorming ideas such as concept art and basic storylines but cannot replace human input or creativity at this time. Also, AI is not an adequate tool for content generation yet. For instance, current AI generative tools are not viable for creation of 3D assets. Similarly, AI generated music is often rough and repetitive and cannot replace the need for professionals to mix and master AAA quality pieces. Generative AI succeeds in writing simple snippets of code but falls short in writing complex code both in Unreal Blueprints and C++. Human-made tutorials provide better systemic understanding, and tend to be more up to date, while the AI is useful for clarifying specific topics. Thus, AI can enhance programmer productivity, including acting as an additional resource for learning more about syntax, technologies, and errors; and generating simple code.

Thus, the authors found generative AI is complementary to human driven game development rather than a replacement for human craftsmanship. The development of this study's games could not have been completed without extensive human involvement. Currently, AI is a tool to increase the productivity of game developers at best.

## APPENDIX

### COMPLETE LIST OF ART AI TOOLS EVALUATED

Name	Developer Organization	Type	License	Cost	Evaluation
<b>3dfy</b>	3DFY.ai Ltd.	Text to 3D	You own the images you generate	• \$5/ download, 10 generations	• Seems limited to couches, swords, and kitchen utensils
<b>Animate Anything</b>	Anything World Ltd.	Automatic rigging and animations	You own the models you generate	• Free • Premium	• Difficult to use • Varying quality of automatic rigging
<b>Artbreeder</b>	Studio Morphogen	Poser, Text-to-Image, Mixer, Expand	Commercial	• Free	• Mixer tool combines images • Item preset poses available
<b>Bing Image Creator (Dall-E 3)</b>	OpenAI	Text-to-Image Generator	You own the images you generate	• Free	• Can access Dall-E for free
<b>Dall-E 2</b>	OpenAI	Text to Image Generator	You own the images you generate	• 15 free images per month; \$15 per 115 requests	• Useful for concept art • Good sci-fi & cartoon style
<b>Dall-E 3 with ChatGPT 4.0</b>	OpenAI	Text to Image Generator	You own the images you generate	• \$20/month	• Easy to use web interface • Good for concept art • Limited level design
<b>Deep Dream Generator</b>	Deep Dream Generator	Text to Image Generator	Commercial	• \$19 – 99/month	• Limited in styles • Can't do realism
<b>DeepAI</b>	DeepAI	Text to Image Generator	You own the images you generate	• Free	• API & web access • Can't make humans
<b>DiffusionBee</b>	DiffusionBee	Text to Image Generator	GNU Affero General Public License	• Free	• Mac only • Runs offline • Like Stable Diffusion
<b>Firefly</b>	Adobe	Text to Image; Generative fill & recolor; 3D to Image	Commercial	• \$5/month for 100 credits per month (and included with various Adobe subscriptions)	• Can match other styles • Best text-to-image • Varied responses
<b>GenCraft</b>	GenCraft	Text to 2D image or video	Company owns the assets that are generated	• Free • Additional pricing options	• Allows for video generation

Genie	Luma Labs	Text to 3D model	Commercial	• Free	• High-res post-processing • Like Tripo AI
<b>Houdini PDG</b>	SideFX	3D Modeling Software	Commercial	• Free for students • Enterprise	• Highly advanced results • Complex interface • Scalable and modular
<b>Leap AI</b>	Stable Diffusion and other open source models	Text to Image	You own the images you generate	• Free • Additional pricing	• Can train custom models • Integrated with Zapier
<b>Lumen5</b>	Lumen5	Text to Video Generator	You own the images you generate	• Free	• Useful for marketing and brand content
<b>Magic 3D</b>	NVIDIA	Text to 3D model, prompt-based editing	N/A	• Unavailable	• Better than Dreamfusion
<b>Meshy</b>	Meshy	Text to 3D model, text to texture	CC by 4.0 License	• Free • Additional pricing options	• Like Tripo AI
<b>Midjourney</b>	Midjourney	Text to Image Generator	You own the images you generate	• \$8 – 48/month	• Discord interface • Great features & details
<b>Mixamo</b>	Mixamo	Automatic rigging and animations	Royalty free	• Free	• Adds smooth pre-existing animations to 3D models • Good automatic rigging
<b>Motorica</b>	Moterica	Generative animation	CC Noncommercial 4.0 Attribution International License	• Free	• Detailed documentation
<b>MV Dream</b>	Bytedance Inc.	Text to 3D model	Apache-2.0 license	• Free	• Requires 20GB of VRAM • Runs locally
<b>NightCafe</b>	Stable Diffusion, DALL-E 2, CLIP-Guided Diffusion, VQGAN-CLIP	Text to Image	You own the images you generate	• \$6/month for 100 credits (enough for ~1,240 images per month)	• Large selection of models & styles
<b>NVIDIA Canvas</b>	NVIDIA	Artwork Generator	You own the images you generate	• Free	• Difficult learning curve
<b>OpenArt</b>	OpenAI	Text to Image, Sketch-to-image, and a stock art transformer	Commercial, with attribution	• \$10/month for 5,000 credits	• Additional features to Stable Diffusion • Custom models
<b>PCG (Procedural Content Generation)</b>	Epic Games	Procedural environment creation	Free-to-use for learning	• Free for students • Enterprise	• Excellent for prototyping • Easy to use
<b>Pikazo</b>	Pikazo, Inc.	Image Filters	You must own the images you use in this program	• Free	• iOS only • Low-quality results
<b>Polyhive</b>	Polyhive	Text to AI texturing tool	You own the images you generate	• Custom	• Can be integrated in Unity

<b>Shap-E</b>	OpenAI	Text to 3D model, image to 3D model,	MIT License	• Free	• Fast, low-quality models
<b>Stable Diffusion</b>	Stability AI	Text to Image Generator	Creative ML OpenRAIL-M	• Free	• Good for generating concept ideas
<b>Synthesia</b>	Synthesia Limited	Text to Video Generator	You own all your content for commercial and non-commercial use with limitations on the usage of stock avatars	\$22/month	• Realistic • multi-language support
<b>Threestudio extension</b>	Google, UC Berkeley	3D content creation from text prompts, single images, and few-shot images, by lifting 2D text-to-image generation models	Apache-2.0 license	• Free	• NErF Technology • Low-quality models
<b>Tripo AI</b>	Tripo	Text to 3D model, image to 3D model	Royalty free	• Free	• Post-processing is good
<b>Vidon</b>	Vidon AI Ltd.	Text to Video Generator	You own the images you generate	• Free • \$38/month	• Useful for marketing and brand content
<b>Volograms</b>	Volograms Limited	3D Model generator	You do not own the images you generate	• Free	• Aimed towards social media

COMPLETE LIST OF PROGRAMMING AI TOOLS EVALUATED						
Name	Developer Organization	Type	License	Cost	Evaluation	
<b>AlphaCode</b>	DeepMind, Google	Natural Language Processing to Code	Apache-2.0 license	• Free • Additional pricing options	• Has the ability to be useful, requires time	
<b>Coffee</b>	Berkeley Vision	Deep Learning Framework	BSD Framework	• Free	• Not useful for game development	
<b>ChatGPT 3.5</b>	OpenAI	LLM Generative Chatbot	MIT license	• Free	• Decent at basics • Poor at advanced concepts	
<b>ChatGPT 4</b>	OpenAI	LLM Generative Chatbot	MIT license	• \$20/month • \$25 for team	• Improved model from ChatGPT3.5 • Improved boilerplate code • Bad for complex interfaces	
<b>CodeT5</b>	Salesforce	Code Completion	BSD-3 license	• Free	• Difficult for beginners • Open source	
<b>Copilot</b>	GitHub	Code completion	You own the code you generate	• \$10/month	• Useful for boilerplate • Convenient interface as IDE extension	
<b>Deepcode</b>	Deepcode AG and Smyk	Code completion	MIT license	• Free • \$52/month	• Useful for fixing code	
<b>Ludo.ai</b>	Ludo.ai	Adaptive Gameplay and Player Analysis	MIT license	• \$15/month	• System adjusts on player analysis for dynamic gameplay	
<b>Open AI Codex</b>	OpenAI	LLM-trained Programming GPT	MIT license	• Like ChatGPT services	• N/A	
<b>Perplexity</b>	perplexity.ai	LLMs	MIT license	• Free	• Good at basic coding	
<b>Poe</b>	Poe	Multiple LLM Chatbots	Multiple LLM Chatbots	You own the code you generate	• \$199.99/year	• Some chatbots are decent at coding
<b>Tabnine</b>	Codota	Code completion	You own the code you generate	• Free • \$12/month, • \$40 enterprise rate	• Odd code suggestions when starting a code base	

COMPLETE LIST OF DESIGN AI TOOLS EVALUATED					
Name	Developer Organization	Type	License	Cost	Evaluation
<b>ChatGPT 3.5</b>	OpenAI	LLM Generative Chatbot	MIT license	Free	• Decent generated content
<b>Poe</b>	Poe	Multiple LLM Chatbots	You own the images you generate	\$199.99 per year	• A lot of variety • Expensive
<b>Perplexity</b>	perplexity.ai	LLM Generative Chatbot	MIT license	Free	• Provides sources, less extravagant
<b>Jasper</b>	Jaspersoft	LLM Generative Chatbot	Allowed for commercial use	\$39 per month	• N/A
<b>Ludo</b>	Ludo AI	Image Generation / Search	MIT license	\$15 month (indie); \$250 month (studio)	• Built specifically for game development
<b>WriteSonic</b>	writesonic	Text Generation	Allowed for commercial use	Free, additional pricing options	• Useful for design process • Variety of uses
<b>Promethean AI</b>	Promethean AI	3D World Building	Allowed for commercial use	Free, or \$10/month	• N/A

COMPLETE LIST OF AUDIO AI TOOLS EVALUATED						
Name	Developer Organization	Type	License	Cost	Evaluation	
<b>AIVA</b>	AIVA Technologies	Music Generator	Commercial Options	• Free • \$33/month	• Simple and intuitive • Includes a digital audio workspace • Highly customizable • Can build around input	
<b>Beethoven</b>	Beethoven	Text-to-Music Generator	Commercial Purposes	• Free • \$20/month	• Good at sound generation • Solid ambient music • Easy customization	
<b>Lovo</b>	Lovo	Text-to-Voice	You own the content you generate; Lovo has non-exclusive rights	• Free • \$75/month	• Decent at voice generation	
<b>Murf.AI</b>	Murf	Text-to-Voice	Commercial Purposes	• Free • \$99/month	• Quality varies • Overall decent	
<b>Overdub</b>	Descript	Voice Cloning	Different plans for personal and commercial use	• Free • \$24/month	• Average	

<b>Play.ht</b>	PlayHT	Text-to-Voice	Different plans for personal and commercial use	• \$5.40 - \$59.40/month	• Quality is like Murf.AI
<b>Sound Effects</b>	Eleven Labs	Text-to-Sound, Effects Generator	Different plans for personal and commercial use	• Early Access	• Convincing sound effects • Quality is prompt dependent
<b>Soundful</b>	Soundful	Music Generator	Different plans for personal and commercial use	• Free • \$60/month	• Very limited access • Difficult to access
<b>Soundraw</b>	Soundraw	Music Generator	Royalty free, Lifelong License	• Free • 16.99/month	• Only good for pop music • Limited list of genres
<b>Voice Lab</b>	Eleven Labs	Text-to-Voice, Voice Cloning	Different plans for personal and commercial use	• Free • Additional pricing options	• Highly realistic • Huge variety of voices
<b>Voicemod</b>	Voicemod	Text-to-Music Generator	Commercial Purposes	• Free	• Average sound generation • Limited choices

- [16] J. Betker, G. Goh, L. Jing, T. Brooks, J. Wang, L. Li et al, Improving image generation with better captions, *Computer Science*, 2023, 2(3), 8, <https://cdn.openai.com/papers/dall-e-3>.
- [17] AIVA, AIVA, (*AI music generator*), Feb. - Apr. 2024, <https://www.AIVA.ai/>.
- [18] Beatooven, Beatooven, (*AI music generator*), Feb. - Apr. 2024, <https://www.beatoven.ai>.
- [19] Soundraw, Soundraw, (*AI music generator*), Feb. - Apr. 2024, <https://soundraw.io/>.
- [20] Eleven Labs, Eleven Labs Sound Effects, (*AI sound effect generator*), Feb. - Apr. 2024, <https://elevenlabs.io/sound-effects>.
- [21] GitHub. (2024). GitHub Copilot (Feb - Apr version) [Large language model], <https://copilot.github.com>.
- [22] Adobe, Firefly, (*AI image generator*), 2024, <https://www.adobe.com/products/firefly.html>.
- [23] GenCraft, GenCraft, (*AI art Image and video generator*), <https://genCraft.com>.
- [24] Midjourney, Midjourney, (*AI art generator*), 2024, <https://www.midjourney.com/home>.
- [25] Godot Foundation, Godot, (*game engine*), 2024, <https://godotengine.org/>.
- [26] Unity Software Inc., Unity, (*game engine*), 2024, <http://unity.com>.

## ACKNOWLEDGMENT

The authors thank Kevin Evans and Arjun Hegde for their help during the game development process.

## REFERENCES

- [1] J. Carter, “Near half of game devs say they’re working with generative AI tools,” *Game Developer*, 15 Feb. 2024, [www.gamedeveloper.com/business/near-half-of-game-devs-say-they-re-working-with-generative-ai-tools](http://www.gamedeveloper.com/business/near-half-of-game-devs-say-they-re-working-with-generative-ai-tools).
- [2] Dechichi, “Exploring AI characters design with ‘Suck Up’!”, *Reddit*, May 2024, [https://www.reddit.com/r/indiegamedevforum/comments/18ihixq/exploring\\_ai\\_characters\\_design\\_with\\_suck\\_up/](https://www.reddit.com/r/indiegamedevforum/comments/18ihixq/exploring_ai_characters_design_with_suck_up/).
- [3] V. Acovino and C. Intagliata, “Games made by soulless machines’: tech sparks debate over AI stories in video games,” *NPR*, 15 Mar. 2024, [www.npr.org/2024/03/15/1238111971/video-games-ai-artificial-intelligence-nvidia](http://www.npr.org/2024/03/15/1238111971/video-games-ai-artificial-intelligence-nvidia).
- [4] Epic Games. Unreal Engine 5.3, (*game engine*), Feb. - Apr. 2024, Retrieved from <https://www.unrealengine.com>.
- [5] R. Dinsdale, “Experimental video game made purely with AI failed because tech was ‘unable to replace talent,’” *IGN*, Mar. 2024, [www.ign.com/articles/video-game-made-purely-with-ai-failed-because-tech-was-unable-to-replace-talent](http://www.ign.com/articles/video-game-made-purely-with-ai-failed-because-tech-was-unable-to-replace-talent).
- [6] OpenAI, ChatGPT, (*large language model*), Feb. - Apr. 2024, <https://chat.openai.com/chat>.
- [7] OpenAI, DALL-E 3, (*text-to-image generator*), Feb. - Apr. 2024, <https://labs.openai.com>.
- [8] OpenArt, OpenArt, (*text-to-image generator*), Feb. - Apr. 2024, <https://openart.ai>.
- [9] Side Effects Software, Houdini, (*procedural 3D animation software*), Feb. - Apr. 2024, <https://www.sidefx.com>.
- [10] Tripo, TripoAI, (*text/image-to-3D model AI*), Feb. - Apr. 2024, <https://www.trip03d.ai/login>.
- [11] Epic Games, Unreal Engine Marketplace, (*3D asset acquisition site*), Retrieved from <https://www.unrealengine.com>.
- [12] L. Corcoran, itch.io, (*publishing platform and asset acquisition site*), Feb. - Apr. 2024, <https://itch.io>.
- [13] Anything World, Animate Anything, (*3D model-to rig and animate AI*), Feb. - Apr. 2024, <https://app.anything.world/animation-rigging>.
- [14] Adobe, Miximo, (*automated character animation*), Feb. - Apr. 2024, <https://www.mixamo.com/#>.
- [15] M. Mori, K. MacDorman, and N. Kageki, “The uncanny valley [from the field],” *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, June 2012, pp. 98–100, <https://doi.org/10.1109/mra.2012.2192811>.