





# Contents

<b>1. Introduction</b>	<b>1</b>
<b>2. Related Work</b>	<b>3</b>
<b>3. Technical Foundation</b>	<b>9</b>
3.1. Game Design Document . . . . .	9
3.2. LangChain . . . . .	10
3.3. LangGraph . . . . .	12
3.4. Gemini . . . . .	13
<b>4. Creating Game Design Documents with LLMs</b>	<b>15</b>
4.1. Prompt Engineering . . . . .	16
4.2. Architectural workflow . . . . .	17
4.3. Agent Architecture . . . . .	18
4.4. GDDState Structure . . . . .	21
4.5. Phased Questioning and GDD Templates . . . . .	22
<b>5. Evaluation</b>	<b>23</b>
5.1. Experimental Setup and Methodology . . . . .	23
5.2. Evaluation Metrics and Questionnaire Design . . . . .	24
5.3. Statistical Analysis and Inference . . . . .	25
5.4. Discussion and Qualitative Findings . . . . .	26
<b>6. Conclusion</b>	<b>29</b>
<b>Bibliography</b>	<b>31</b>
<b>A. Appendix</b>	<b>35</b>



# 1. Introduction

Gaming is, with good reason, the largest and highest-grossing entertainment sector in the world. The fascinating properties of games can help humans learn and safely gain experiences without the fear of real-world consequences. Through gameplay, humans can explore emotions and experiment with different strategies, which can help them master the real world. A Game Design Document (GDD) is used by the game designer to communicate with the development team to ensure everyone is on the same page. A GDD is created at the start of a game's lifecycle and iterated upon until the game is released. It contains not only the theoretical mechanics and gameplay loops of the game but also handles how the game should look and feel, as well as the technical engine, database, and programming language the game will be developed in. Because of this, it is necessary to have a GDD as early as possible to communicate the game designer's needs to the team.

Coming up with a Game Design Document is not an easy task. It is essentially conceiving a game idea, fleshing it out, and figuring out all the intricate details of how you want your game to look, feel, and what experience you want the players to have. This is a formidable task, especially for newer game developers. Because of this time-intensive and complex process, people seek ways to simplify it. With the rise of AI and Large Language Models (LLMs), new possibilities have been created to streamline the ideation and GDD writing process. As in most creative activities, LLMs can be used as a tool to help create games and make them more immersive. For example, in-game, LLMs can be used for adaptive storylines and quest structures in which NPCs can react with more freedom to the player's actions. They find even more utility in the production phase of video game creation. LLMs can be used for repetitive tasks, like coding with Copilot, helping in writing storylines for the world and characters, and creating concept art and in-game assets with generative models like Midjourney. In almost every step, you can enhance your performance with LLMs.

This thesis seeks to investigate how effectively LLMs can be used for game design by creating ideas and planning a GDD for video games. Therefore, as little human input as possible is given to the LLMs to test their skills. The central focus of this work is to determine to what degree LLMs are capable of generating a fully developed GDD with minimal human intervention. These three research questions were chosen, which all address different qualities of GDDs. The first addresses the originality and uniqueness of the ideas and mechanics; the second investigates if the AI can produce structurally sound and consistent documents. The third research question is about readability and how easy it is for humans to understand the ideas and mechanics. This is essential for a tool which is used for communication.

- **RQ1:** To what extent can an AI agentic workflow produce original game concepts

## 1. Introduction

that are perceived as innovative rather than derivative?

- **RQ2:** How effective is the AI at maintaining internal logic and technical consistency compared to the fleshed-out details of professional human blueprints?
- **RQ3:** How does the communicative quality and professional writing style of AI-generated GDDs compare to human-authored documents?

The results show that the AI process can create more innovative and creative game ideas compared to humans. However, the structural integrity and clarity of the documents are still lacking. AIs tend to use keywords for mechanics without context, resulting in fluff-heavy sentences. Additionally, repeated runs with similar inputs tend to converge and produce similar results. While it does not replace the creative nature of humans, the framework can be used to generate prototypes and starting points for their own projects. While LLMs are generally good at keeping consistency and structure, the findings suggest that full automation currently lacks the divergent thinking required for high-level creativity. Furthermore, by giving LLMs full control, it raises ethical questions about the loss of human-centric authorship. The resulting erosion of original creative intent could potentially hinder the long-term evolution of the medium.

## 2. Related Work

To understand whether LLMs can generate complete, functional Game Design Documents (GDDs), this section first analyzes the broader academic context of AI in game creation before examining the methodological advances of creativity in LLMs, including autonomous agents, multi-agent systems, and persona-based prompting. This outlines the current state of research and highlights the gap that this thesis addresses. Games and Large Language Models have a bilateral relationship regarding mutual improvement. On the one hand, humans use games to better understand LLMs; researchers have begun using games as testbeds to evaluate the capabilities of these models. They provide controlled and well-defined environments suitable for developing and benchmarking AI capabilities.

Sidji investigated different prompt engineering strategies for the game Codenames to improve game scores. While different prompt engineering strategies did not significantly alter performance, the playstyle and tone of the LLM could be drastically changed [26]. Another example is the work of Maisto, who explored the linguistic styles of AI based on fully AI-generated roleplay sessions. This research reveals that AI uses a different linguistic style and sentence structure than humans. AI mostly utilizes the present tense, whereas human roleplaying sessions contain significantly more past-tense sentences. They also analyzed contextual cohesion and found that it was deficient, revealing logical errors [21]. With the game Werewolf, researchers have shown that LLMs can overcome their intrinsic bias. Often in models, randomness is not evenly distributed, but with the help of reinforcement learning and Werewolf as a testing ground, this machine error has been mitigated. This leads to human-level play and the outperformance of other LLMs playing Werewolf [30]. There is a wide array of games used to test the skillset of LLMs, such as Tetris and Super Mario Bros. These studies demonstrate that games often test multiple capabilities of AI at once, which makes them perfect for overall performance measurement. Researchers performed reinforcement learning across multiple games, which also increased model proficiency in unseen games and external tasks [10]. This demonstrates how games can have a positive impact on understanding and testing Large Language Models.

On the other hand, LLMs can perform various tasks to assist game designers throughout the development pipeline. In nearly every development phase, designers can use AI tools to improve their workflow. A foundational approach by Lim shows that for ideation, LLMs can produce a vast amount of diverse content used for rapid idea generation and brainstorming. The authors conducted a rapid ideation process with 21 participants and found that LLMs produce varied, high-quality ideas in co-creation with humans. They did not, however, use AI to evaluate these ideas [17]. Another method for idea generation is shown in the work of Gu et al., which uses combinatorial creativity to outperform baseline approaches by approximately 10 percent. A retrieval system can combine connections from cross-domain knowledge to create novel ideas. This demonstrates that a guided

## 2. Related Work

framework can enhance LLMs; however, it carries the potential risk of amplifying biases present in the reference texts [8]. For evaluating creativity, Kim and Oh compared the performance of AI to human evaluation. They evaluate creative tasks focusing on specific criteria such as originality, fluency, and flexibility. The results show that LLMs can provide objectively consistent evaluations but face limitations in recognizing cultural specifics and context-dependent aspects. These findings suggest that further research and more robust AI models are needed to bring them to a human level of evaluation [12]. Kumar et al. introduce an "LLM-as-a-judge" framework that uses personality traits and expert attribution to evaluate creative text. It demonstrates that this approach consistently outperforms supervised fine-tuning, aligning more closely with human evaluation. However, it focuses only on general text creativity and does not investigate specialized domains [14].

The most obvious use for a text-based AI model is generating game scripts. Quests, dialogues, and background stories can all be quickly prototyped by AI models. In recent years, many studies have shown interest in not only using pre-written AI stories but also utilizing LLMs procedurally in-game to generate fully dynamic dialogues, quests, and background narratives. Buongiorno et al. researched Procedural Artificial Narrative using Generative AI (PANGeA) in role-playing games (RPGs). PANGeA is a framework consisting of a memory system, validation system, Unity plug-in, and a server connecting any game component with a local, private LLM. This enables a system where RPGs can dynamically generate levels, key items, and non-player characters (NPCs). This positions PANGeA as a versatile tool for game designers supporting narrative-driven content. However, it lacks anti-cheat mechanisms; therefore, players might be able to derail the game narrative via malicious input [4].

To mitigate the possibility of changing the narrative space, Lu et al. developed an AI-bridged Interactive Narrative (IN) authoring system called WhatELSE. With this tool, creators can define and control narrative possibility spaces, which are perceived and shaped through three different views: narrative instance, outline, and variants. WhatELSE additionally unfolds those narrative spaces into executable game plots. This study focuses on the authors without incorporating the player experience; future work could incorporate player feedback to further understand the system [18]. Another way to sharpen procedural stories is to incorporate emotional arcs as a structural backbone in narrative generation. By using emotional arcs as templates, generative systems can ensure that humans recognize these patterns, making the story objectively more engaging. Wen et al. focus on two emotional patterns which were implemented in a prototype action role-playing game. Through human sentiment analysis and interviews, the authors show that an emotional arc enhances engagement and emotional impact. The current implementation focuses on the narrative without incorporating gameplay systems, which could lead to ludonarrative dissonance between the story and the gameplay [29].

For image creation, multiple AI methods can be used to generate procedural imagery. As Mao et al. show, Generative Adversarial Networks (GANs) can be applied to a wide variety of applications, ranging from game environments and high-quality characters to sound. However, they suffer from significant training instability and the risk of mode

collapse, where the generator produces limited varieties of output. Consequently, ensuring a balanced optimization between the generator and discriminator remains a primary challenge in applying GANs to complex narrative or procedural tasks. Transformers excel in handling long sequential data, making them better suited for movie production or game level generation. Diffusion-based models, on the other hand, learn by adding random noise to existing data and then reversing that process to transform the noise into structured output. They excel in human motion, texture, and material generation [22]. Until now, only 2D art has been discussed. For the creation of 3D models with AI, Begemann and Hutson explored a hands-on approach, utilizing different AI tools for 2D concept art creation and then transforming these images into 3D models. These AI-generated models still lag behind, often being less detailed or optimized than required for use in games. They specifically lack in topology, complexity, and optimization. Further research and more promising AI models are needed to make 3D model generation a viable tool for game designers [2].

Another phase of game design where AI tools can be applied is in prototyping and playtesting. In the context of game design, iterative development serves as the fundamental mechanism for refining the player experience during this phase. This is generally a very slow process because, after each iteration, one should ideally change only one element at a time [11]. Therefore, many methods have been developed to reduce the time this task takes. Li et al. introduced an LLM-based pipeline for card game prototyping. Through the integration of a graph-based representation of game mechanics, a game code generation system, and a scalable gameplay AI, they provide designers with a system that enables faster card game prototyping cycles. As a next step, different game genres could be the subject of research, expanding into puzzle or narrative-driven games [15]. In the work conducted by Anjum et al., they tested how effective AI models are at prototyping and implementing full functional samples. In this research, ChatGPT was used not only to theoretically conceptualize game mechanics but also to implement them afterward. Three genres were chosen in which the LLM was tasked with designing three mechanics. After the AI implementation, a user study was conducted comparing the AI-generated features to human-crafted ones. It demonstrated that the theoretical features are at least as good as the human features. In contrast, during implementation, the AI-designed systems and features lagged behind human-made games. They lacked an understanding of important context, and their ability to discern what "feels good" to a player was poor [1].

Recent research has increasingly examined the creative capabilities of LLMs and how they compare to human creativity. Empirical studies suggest that LLMs are able to perform competitively with the average human on certain creativity-related tasks, particularly those involving divergent thinking and idea generation. Bellemare et al. used different methods to compare human capabilities with AI. The results show that AI models perform better on the Divergent Association Task (DAT) and possess nearly average human writing abilities. Their findings also indicate more creative output when using a higher temperature value for input. It also suggests that even small prompt changes can produce a significant increase in creative task performance [3]. In contrast to studies that benchmark LLMs directly against human performance, the theoretical work

## 2. Related Work

of Kortenbach et al. suggests that LLMs and human creativity operate differently in the creative act. Specifically, LLMs are positioned as complementary amplifiers of human cognition that contribute to creative processes without replicating human intentions or subjective creative capacities [13]. Mehrotra et al. also illustrate how interdisciplinary analogies can be used to enhance LLMs. Through associative thinking—a process of linking unrelated concepts known to be effective in humans—the study proves that such methods are equally applicable to LLMs. This was limited by context dependency, where the models hallucinate in certain contexts. Another side effect was the over-representation of nature themes in random objects, which can be called an "associative rut," where certain concepts lead to the activation of already known ideas. Distinct from the aforementioned context dependency, AI models tend to rely on a limited subset of concepts even when operating within a single domain. This suggests a lack of conceptual diversity, as the model repeatedly favors specific patterns over the full knowledge base [23].

In the paper by Zhao et al., the authors discuss how approaches such as agent-based systems, personas, and prompt engineering are used to guide LLMs toward more creative, diverse, or coherent outputs. These methods play a central role in practical applications of LLM creativity. The paper also points out that Large Multimodal Models (LMMs) and their ability to process input and output beyond verbal question-and-answer formats are important for future research. Zhao et al. argue that an open question in the field remains whether LLMs inhabit real creativity or simply imitate human creativity by learning from vast amounts of content. As creativity in LLMs depends on multiple controllable factors, researchers increasingly focus on methods that influence model behavior directly [32].

While prompt engineering and agent-based approaches are closely related, they address different levels of control. Prompt engineering focuses on the design of input instructions, which guide a single model response. These instructions can be in natural language and guide the model toward task-oriented outputs, influencing factors such as reasoning style, creativity, and output structure. The most prominent prompt engineering mechanic is Chain-of-Thought (CoT). Wei et al. identified that by eliciting a series of intermediate steps, the model's performance is boosted on logical problems [28]. Sahoo et al. identify multiple prompt engineering styles, including zero-shot, few-shot, and chain-of-thought prompting. Zero-shot prompts prioritize efficiency but offer limited control, whereas few-shot and chain-of-thought prompting improve coherence and problem-solving accuracy. Despite these advances, prompt engineering still faces challenges such as biases and factual inaccuracies [24].

To reduce hallucinations, various methods have been created to mitigate the risk of incorrect information. One approach is named Chain-of-Verification (CoVe). This approach by Dhuliawala et al. first drafts responses before planning to fact-check and verify the questions independently. This ensures the questions are not biased by previous answers. The findings indicate that by simply asking the models to verify the output, they achieve substantial performance gains. Nevertheless, the risk of hallucination is not completely removed, only reduced [6].

Another important component of prompt engineering is the use of personas. According to Tseng et al., personas are primarily implemented through role-based prompts that

assign LLMs a specific identity or role. This changes the tone, reasoning style, and output structure. Tseng et al. emphasize that personas enable effective control over model behavior without modifying model parameters. One challenge to be resolved in the future is a standardized benchmark for testing personas. Currently, a diverse array of task-specific metrics is needed to test all skills [27]. Subsequent research has shown that the formulation of persona prompts matters. LLMs often reflect societal biases or stereotypes; for instance, studies analyzing demographic representations show that while AI can simulate various groups, it may default to specific patterns unless prompted otherwise. Systematic evaluations of sociodemographic persona prompting find that different role adoption strategies and demographic priming can significantly influence how LLMs simulate human groups and reduce stereotyping under certain configurations. The study is limited by the fact that it only focuses on gender and race [20]. Research indicates that when assigning racial or ethnic personas, LLMs can exhibit performance variances. For example, some benchmarks show that without specific guidance, models may produce more stereotypical content for Hispanic (up to 12 percent higher frequency in some tests) or Middle Eastern personas compared to neutral "assistant" roles.

Jiang et al. investigated the behavioral effects of assigning personas to models. This is demonstrated by the Big Five Inventory (BFI) test, used to analyze different AI personas. The results show that the assigned personalities match their BFI answers and produce text with the characteristics of the assigned trait. This demonstrates that persona assignments produce greater variability in LLM responses compared with neutral controls and influence both objective and subjective aspects of generation across diverse tasks [5]. To evaluate agent personas, Samuel et al. introduced PersonaGym, where LLM personas are evaluated using PersonaScore as a metric to assess how faithfully these agents adhere to their personas in diverse task environments [25].

In contrast, agent-based systems treat LLMs as autonomous entities with persistent goals. They can have different memory types, such as short-term or long-term memory, to store inputs and outputs. Interaction rules and tool utilization enable sustained and adaptive behavior across multiple steps. As a result, prompt engineering primarily affects local output generation, whereas agents and agent systems influence long-term behavior and decision-making processes [19]. Ye uses a dynamic graph-based memory tree to store and retrieve information, which transforms LLMs into more robust revision-aware agents. This Task Memory Engine (TME) builds a dynamic task graph that maps the input to multiple subtasks and enables dependency-tracked revisions. This reduces the risk of hallucination and misinterpretation compared to standard LLM systems [31]. A different memory type is used in LEGOMem, a modular procedural multi-agent framework where task agents learn from earlier executions by splitting full-task and subtask components. Han et al. use a memory retrieval and allocation system accompanied by a memory orchestrator agent to obtain more reliable and reusable solutions [9]. To further diminish hallucinations, Liang et al. introduced Multi-Agent Debate (MAD), a framework where multiple agents discuss a topic while a judge manages the debate to obtain a final solution. This addresses the problem of Degeneration-of-Thought (DoT), which occurs when a model loses logical coherence and is unable to reflect on its flaws. This setup still has limitations,

## 2. *Related Work*

such as maintaining coherence in long-context scenarios [16]. Multidimensional Multi-Agent Debate extends the MAD framework by adding multiple dimensions for evaluation. With this, the judge can compare ideas based on multiple criteria like accuracy, coherence, and originality. This reduces the risk of oversimplification and improves interpretability. However, open questions remain about how well robustness scales when many dimensions are used for evaluation [7].

## 3. Technical Foundation

This chapter provides an overview of the technical and theoretical foundations upon which this thesis is built. To construct Game Design Documents fully autonomously, a foundational understanding of their structure and how agent systems can be utilized to generate these documents is needed. This chapter starts with an introduction to Game Design Documents, why they are a necessity in game design, and how they are structured. After that, the software architecture of LangChain and LangGraph is explained to provide insight into the building blocks of this thesis. The chapter demonstrates the capabilities and constraints of LangGraph-based multi-agent workflows, how they can be constructed, and how to use them effectively. In conclusion, the last section takes a look at Gemini, the Large Language Model which acts as the central intelligence for creating the document.

### 3.1. Game Design Document

A Game Design Document is the central hub for communication in the development of all modern digital games. It serves as a comprehensive description of a game's components, containing concepts, mechanics, systems, and future content. It provides a shared reference for all stakeholders involved in the development process. These stakeholders typically include everyone who is part of the game design process in any capacity, such as designers, software developers, artists, producers, and other contributors. The primary purpose of a GDD is to provide everyone working on the game with a consistent vision through the entire lifecycle. Normally, the game director is responsible for creating and managing this document. Due to the iterative process of game design, the document will change significantly during the development period. This is beneficial because, with every playtest and iteration, the team gains new insights into what works and what does not for the specific experience. Because of the creative nature of games, the overall structure of a GDD is not set in stone and should be tailored to the needs of the specific game.

A Game Design Document typically begins as a concise, high-level description during the early stages of development, where it primarily serves to define the core concept of the game. As development progresses and the game becomes more refined, the GDD evolves accordingly, gradually expanding in scope and detail. The document is generally considered complete only when the game reaches its final development stage. Defining the experience players should have and what they will learn through the game is key in every GDD. It also describes the genre, target audience, required technologies, and intended platform. The first section establishes the overall identity and goals of the game. After that, the document usually defines the core gameplay mechanics, such as player interactions, progression systems, rules, and feedback loops. These mechanics form the

### 3. *Technical Foundation*

foundation of the player experience and are therefore described in detail.

In addition to gameplay mechanics, many Game Design Documents include narrative and world-building elements. Even in games with minimal narrative focus, contextual elements are often described to support consistency in visual design and, by extension, player immersion. This can include story arcs of the main characters, descriptions of the game world and its lore, and the narrative framing of gameplay systems. Additionally, the art style and theme of the game can be described in detail in a GDD. This can be presented in the form of text, images, concept art, or any other medium to better illustrate certain design decisions. It can describe the style, color palette, level of realism or abstraction, and overall aesthetic direction. This can drastically change how the player perceives and experiences the game. From a technical perspective, a GDD may outline system architectures, artificial intelligence behavior, level design principles, and user interface concepts. While it does not replace technical design documents or implementation specifications, it provides a conceptual framework to build upon. As a result, the level of detail within a GDD can vary significantly depending on the project scope, team size, and development methodology.

## 3.2. LangChain

LangChain is a high-level framework utilized in this thesis. It is open-source and designed to facilitate the development of applications based on Large Language Models (LLMs) with agents. First released in 2022 by Harrison Chase, it provides abstractions and tools that enable the creation of workflows around LLMs. By offering standardized components for prompt handling, model interaction, memory management, and tool integration, LangChain supports the development of complex and reliable AI-driven systems. LangChain supports both Python and TypeScript as implementation languages.

A central architectural characteristic of LangChain is the organization of workflows in the form of Directed Acyclic Graphs (DAGs). Within this framework, a workflow is defined as a graph consisting of nodes and edges. Nodes represent executable units, such as LLM calls, tool invocations, or custom logic functions that agents can perform. These nodes can be placed in sequence, creating "chains," which is a core focus of LangChain. These chains transform LLMs—which typically generate isolated text—into reasoning engines capable of executing complex, multi-step workflows. Crucial to this is the acyclic nature of LangChain; data flows in one direction without loops or recursions to a previously visited node. This approach is useful for tasks requiring structured output, but it lacks iterative cycles and circular logic which are essential for creative refinement and the generation of a Game Design Document.

While chains are static in nature, specific LangChain agents use an LLM as a reasoning engine rather than just for text generation. Based on user inputs and context, agents can autonomously decide which tools to use to reach their goals. For this, the principle of ReAct (Reason + Act) prompting is used: the model first generates a "thought" and then carries out a certain "action" via LangChain tools. If tasks become too complex for a single agent, Multi-Agent Systems (MAS) can be used to divide tasks between

specialized entity agents. This modular approach utilizes the principle of Separation of Concerns, where each agent has a distinct role. This also enables communication between them, allowing domain-specific expertise to be exchanged, which diversifies the output and prevents models from suffering contextual dilution. Sub-agents can also be employed, where a supervisor agent decides which agent is invoked next, a process that can be parallelized for faster results.

Tools are defined as interfaces that enable LLMs to interact with external functions. They can retrieve real-time data, query external databases, execute code, and take actions in the physical or digital world. To define a tool, a decorator is required, specifying a name, a description, and an input schema for the expected parameters. During execution, an LLM can select and invoke a tool by generating a structured request that matches the tool's specification. Tools are most effective when they can access the state, memory, and runtime context of the agents, enabling context-aware decisions and personalized responses.

LLMs can be utilized through agents or as standalone models for simple tasks where no agent framework is needed. LangChain supports over 40 different AI model providers, including OpenAI, Google, and Ollama. Multiple parameters can be used to configure model behavior. With temperature, the randomness of the output can be controlled; a higher value grants more creative output, while a lower value makes the output more deterministic. Other parameters include maxTokens, which limits the response length, or maxRetries, which specifies the number of attempts an agent takes if a request fails. Certain models also possess the ability to process and return non-textual data types like images, audio, or video. Messages are the fundamental context for agents, representing the state of a conversation through content and metadata. Messages include a "role" (such as system, human, or AI), the actual content, and optional metadata like message IDs or token usage. The system message instructs the model on how to behave, human messages represent user interaction, AI messages represent model-generated data, and tool messages represent the output of tool calls.

Access to memory is a crucial part of agent interaction, allowing for the storage of previous interactions, feedback, and user preferences. For tasks with high user interaction, retrieving information from memory is essential for efficiency and satisfaction. Without this short-term memory, every LLM call would be stateless, meaning the model would forget everything from the previous turn. To implement this, a checkpointer must be specified when creating an agent. Memory is stored in the agent's state and updated after a completed step or tool call. To customize memory, long messages can be trimmed, deleted, or summarized to ensure they do not exceed the model's context limit. Another utility available to developers is Human-in-the-loop (HITL) middleware. This allows the code to halt for human approval when an agent requires manual review. Each tool can check a configurable policy to issue an interrupt. The human can then approve the action, reject it with feedback, or edit the tool call before it runs.

While LangChain provides powerful abstractions for building LLM-driven workflows, it primarily supports linear or branching execution patterns. For more complex, iterative workflows, additional orchestration mechanisms are required. This limitation led to

### 3. Technical Foundation

the development of LangGraph, which extends the LangChain ecosystem by enabling graph-based, cyclic interaction models.

#### 3.3. LangGraph

LangGraph is a framework designed to be used in conjunction with LangChain components, enabling the construction of iterative, graph-based workflows for LLMs. It is a low-level framework that focuses on agent orchestration, supporting complex execution patterns that extend beyond linear chains or simple agent loops. Within this framework, a workflow is defined as a graph consisting of nodes and edges. Edges define the possible transitions between nodes and can be conditional, allowing execution paths to vary based on the current state or model output. This architecture facilitates the complex cyclic iterations required for developing a GDD.

With cyclic execution, directed edges in a workflow graph can loop back to earlier nodes. These iterations continue until explicit termination conditions are fulfilled or the token length limit is reached. These conditions are defined through conditional edges or state-based checks. This execution model makes the control flow explicit, as all possible cycles and transitions are defined within the graph structure. As a result, iteration behavior is deterministic and traceable. The execution can be inspected and modified by changing the graph definitions rather than altering individual agent personas.

Subgraphs can be used to build multi-agent systems that reuse a set of nodes across multiple graphs. A subgraph is essentially a graph used as a node within another graph. This modularity allows multiple developers to work on different parts of the graph independently, provided the input and output schemas are respected. The most straightforward implementation involves invoking a graph from within a node of another graph. If required, each subgraph can maintain its own memory via the `checkpoint` option.

Through LangGraph streaming, intermediate outputs can be displayed while a graph is running. Instead of returning only a final result after full execution, the streaming interface produces partial outputs from individual nodes. These contain streamed graph states, such as updates and values. Additionally, LLM token streams and custom data from tool functions can be captured. These events are generated in execution order, reflecting the current progression of the graph. During streaming execution, each node activation can produce one or more streamable outputs exposed as structured events. The shared state is updated incrementally, and changes may also be streamed as they occur. This greatly improves the user experience, allowing consumers of the graph execution to process intermediate results, monitor the execution flow, or terminate execution based on streamed signals.

LangGraph, in combination with LangChain, is a powerful toolset that elevates basic LLM usage to a new level. Through the use of agents, multiple personas can be created and utilized to produce more creative output. With iterative workflows and multi-agent setups, AI agents can discuss, communicate, and select answers dynamically. This enables humans to apply AI to increasingly complex and creative tasks, such as the creation of a

Game Design Document.

### 3.4. Gemini

Google Gemini 3 was released on November 18, 2025, and is currently one of the leading LLMs for following complex instructions, agentic use cases, and high-level reasoning. It is based on a decoder-only transformer architecture and is designed as a multimodal model, allowing it to process and understand different types of input data—such as images, videos, text, and code—simultaneously. This capability enables sophisticated cross-modal reasoning.

One of the key features of Gemini 3 is its advanced reasoning capabilities, which include various "thinking levels" that allow users to adjust the depth of the LLM's logic. Gemini 3 Deep Think is a specialized configuration that runs multiple internal reasoning traces to reduce the risk of hallucinations. It achieves superior results on benchmarks measuring domain knowledge, mathematics, and logical reasoning. In contrast, Gemini 3 Flash offers extremely low latency, making it the most viable configuration for multi-agent setups where speed is critical.

Another area where Gemini 3 excels is in so-called "vibe coding." This allows the model to generate and debug code autonomously while remaining tightly integrated with developer tools and environments. Through vibe coding, Gemini 3 can generate, debug, refactor, document, and iteratively improve complete software systems. Function calling also received a significant upgrade; Gemini 3 can now produce structured JSON files with a higher probability of success and without format hallucinations. A critical technical requirement for Gemini 3 agents is the use of Thought Signatures, which must be passed back through the conversation history to maintain the model's exact train of thought during multi-step tool use.

In conclusion, the Gemini 3 family is increasingly focused on agentic systems. By offering different model configurations, users and designers can further customize their experience. Specifically, Gemini 3 Flash provides the flexibility needed for latency-critical use cases like multi-agent setups. This provides a robust foundation for using Gemini 3 as the core AI model for this thesis.



## 4. Creating Game Design Documents with LLMs

This chapter presents the overall structure and design process of the automated GDD creation implementation. At first, the general workflow is examined before analyzing the more specific and intricate details, such as the agent prompts and the GDDStruct. The goal of the Game Design Document is not to contain clearly defined algorithms or ready-to-use code. It should simply provide the creative baseline from which a game could be developed, in a written format.

The implementation of the GDD generation is, at its core, an agent-orchestrated Python script with cyclic feedback loops. This enables a setup where a combination of different agent personas can be used to generate more complex outcomes. The process of document creation can be split into three parts. The first part consists solely of a research agent. This agent's mission is to create a "human spark" of imagination by generating random words, which must be used by the other agents in some way. Generating the whole text in one go would overwhelm the AI without creating context-rich and dynamic ideas. Because of that, the second part of the workflow contains the iterative core loop, where multiple agents communicate to answer specific game design questions one at a time. This is the intellectually intensive and token-expensive part of the pipeline, as each design question is iterated upon multiple times. The third phase is the editorial phase, where the questions are edited into a clean and formal structure: the Game Design Document. Then, concept and gameplay images are generated as the last step in the design process by using the fully answered design questions as input for the concept drawings.

To utilize Gemini 3 to its fullest, this implementation uses various temperature settings for the LLM. The designer agent uses a temperature of 1.0 to achieve the most creative thinking and overcome general AI tropes. The temperature indicates the degree of randomness, where high numbers indicate a more divergent outcome, whereas lower numbers indicate more deterministic and logically consistent output. The artist, researcher, and evaluator all use a temperature setting of 0.5, and the programmer agent uses a minimal setting of 0.1, which better suits this agent for monitoring the feasibility and technical constraints of the project. This helps ensure that the agents act as both creative input and technical filters.

The only time the user can input anything to control the outcome of the GDD is at the start of the execution through a Command-Line Interface. Through this entry point, users can define two variables. The first is the overall topic, where the genre or a small description of the desired game can be entered. The second is called constraints, where the user can enter the budget, team size, and timeline. All remaining creative reasoning is delegated entirely to the AI agents. The longer and more precise the descriptions are,

## 4. Creating Game Design Documents with LLMs

the more granular and precise the results usually become.

```
1 """
2     "topic": "An open-world action-adventure game where the player
3     awakes from a coma, fighting rival gangs and a corrupt corporation.",
4     "constraints": "Multi-million dollar AAA production; Timeline: 3
5     years; Team Size: 200+ employees; Target: Xbox, PlayStation, PC",
6 """
```

Listing 4.1: Example for human input prompt

### 4.1. Prompt Engineering

Initial development of this workflow began as a monolithic single-prompt system. It started by using a zero-shot prompting strategy with the Gemini web interface to generate a full document at once. This revealed a central flaw in AI-generated documents: a lack of creative variance. When prompted with identical inputs, the system produced nearly deterministic results that lacked creative depth. An iterative approach was needed to counter this lack of originality. Tests were conducted using two instances of Gemini, copying results back and forth. In this prototyping phase, both models already had different personas and iterated over one question at a time. Unfortunately, this manual approach showed that automation was necessary because the operational overhead was unsustainable.

An open-source Ollama Python script was found where multiple agents could communicate in a fixed sequence. However, it was unable to create complex iterative graphs, meaning all questions had to be inputted at once, which diluted the agents' focus. Furthermore, the logic for moving to the next question had to be handled manually by the agents; only when every agent said "FITS" could a moderator agent introduce a new question. This was error-prone, and because it used a local Ollama LLM, the results were not significantly better.

A framework was needed that supported agentic orchestration and native support for a persistent global state. LangGraph and CrewAI were identified as leading open-source frameworks. While CrewAI excels at autonomous role-based collaboration where agents choose how to interact, LangGraph is more structured, offering explicit state-machine logic. Since state management is only implicitly possible in CrewAI, LangGraph was chosen. First, the iterative core was designed using multiple agents and one evaluator. Later, agent sizes and personas were established through multiple testing rounds. After this refinement, the editorial and image generation agents were implemented. To solve the remaining homogeneity in the output, the research agent was injected to generate more diverse results.

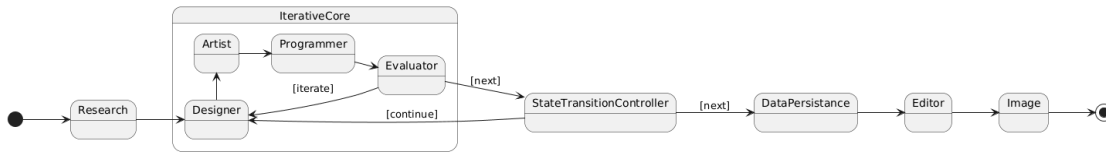


Figure 4.1.: Node setup

## 4.2. Architectural workflow

The first node called in every run is the researcher agent, who plays an important creative role in the set of agents used. AI models are often not proficient at creative tasks because they rely too heavily on probabilistic patterns rather than genuine innovation. To avoid having each Game Design Document degenerate into generic AI tropes, the research node introduces randomly selected words, which enables more creative freedom through divergent mechanical constraints. This has been implemented with the `RandomWords()` library, which generates random English words. These words act as seeds for the LLMs, as they prevent them from taking the path of least resistance. In doing so, they are forced to use combinations that are not found in generic datasets. To ensure that these words are not only used as "flavor text" for game mechanics without any real meaning behind them, the researcher provides the agents with definitions and ways to incorporate these words into the mechanical structure of the game. The research node is only called once in the design process, so the three words will remain consistent throughout the full development pipeline. This agent provides the necessary entropy in an otherwise rigid system, enabling contextually more creative documents.

In contrast to the researcher, the next four agents are part of the iterative core. Their job is to work through a set of predefined questions. To achieve this, the whole iterative core is activated multiple times per active question. These questions start with fundamental design philosophies and then become increasingly specific. The first agent in this quadruple cycle is the designer agent. He serves as the creative anchor. He is the idea generator, meaning he does not iterate over old ideas but generates new ones at the beginning of each question. As input, the designer agent must also utilize the results of the researcher. Instead of generating only one idea, he proposes multiple ideas with fundamentally unique approaches for every question. This marks the building block from which all other ideas are derived.

By having multiple ideas, the next two agents—the artist and programmer—have something to compare against. Their job is to rate the ideas and keep iterating on them. Both also act as active filters. This means the artist evaluates for artistic expression and how the concept could set itself apart from other games. The programmer, on the other hand, pays attention to the constraints given by the user. He focuses on the feasibility of the project and how certain concepts can be implemented technically. They do not only provide specific feedback; they can also give suggestions on how to make these ideas better. These multiple perspectives act as rigorous stress tests for the designer's initial concepts.

#### 4. *Creating Game Design Documents with LLMs*

After the initial idea generation, idea validation is needed. This is conducted by the Evaluator Agent, who acts as the Lead Game Designer. He decides if an answer is good enough and has therefore received enough polish to be put into the Game Design Document, or if another iteration should be started to refine certain elements. After reading all the ideas from the designer, artist, and programmer, he has to make a binary decision between STATUS: CONTINUE or STATUS: FINAL. If the result is satisfactory or the maximum number of iterations per question is reached, he summarizes the final decision and stores it in the GDDState struct. If the answers are not sufficient, the evaluator can give the team follow-up questions regarding what he wants more specifically. To ensure that a question will not be indefinitely refined, a termination constraint is added to the evaluator. This forces a FINAL decision after the third round of iteration. This is implemented with a conditional edge where a function checks if it is the last round. The evaluator enforces forward progress, rejects vague ideas, and manages the token budget by maintaining a fixed round limit.

If the decision is finalized, another conditional edge is needed to determine whether another question is in the queue or if it was the last question. When the last question is completed, the editor agent node is called. At this stage, all design answers are stored in a text file without any structure. The editor agent converts this into a readable Game Design Document. To ensure that every document looks the same, a specific structure is given to the agent to ensure a consistent style. This style is saved as a Markdown file, which is human-readable and easily modifiable by LLMs for further steps. The last node is the image agent. This node generates images based on the full document answers. It produces one gameplay concept art image, which is then stored in the GDD. For this to work, it utilizes another LLM to generate images.

### 4.3. Agent Architecture

This section discusses the anatomy of each agent, as they are not merely interfaces for LLM calls, but are structured in their layout to enable complex and creative output. The goal was to simulate a real-life game design group with different-minded people, enabling more input and a more creative outcome. At their technical level, each agent receives input through context injection. Most of the agents use GDDState to get their required information. In their main body, agents define the input which is then sent to the LLM. While normal text can be used, LangChain's SystemMessage is better for longer prompts because it explicitly separates high-level instructions from user-specific data. This helps separate, for example, the formatting guidelines from the task content. On the other hand, HumanMessage is used for giving the agent dynamic user input. This separation is used in most of the agents to provide better context and clarity. At the end of each agent, the invoke method triggers, the LLM is executed, and it returns an AIMessage containing the response.

The research agent's prompt receives three random words and then generates a dictionary definition for them. After that, the prompt "Explain how this could be used in a game" helps the other agents brainstorm how to integrate these concepts. Finally, two additional

lines help the agent focus on the literal meaning of the words rather than using them as mere decorators.

```

1  """
2  You are the Game Artist.
3  Your job is to make a FUN game, which challenges stereotypes and which is
   new on the market.
4  Tell which Approach you like the best, whilst and how you can make it
   better in a few sentences.
5  You want the ideas to be as creative as possible.
6
7  Research Data:
8  {research}
9  Research Laws:
10 - Strict word ban: Do NOT use the name of the research words, only their
   concepts.
11 - Functional Translation: You must translate the literal dictionary
   definition into the game.
12 - Approach Structure: For each approach, explicitly state which
   Functional Logic from the research is used.
13
14 - Do NOT contradict already accepted questions.
15 - Do NOT restate already accepted ideas.
16 - Do NOT summarize previous discussion.
17 - Do NOT praise other agents.
18 - Produce only new, concrete contributions.
19 - Be concise and specific with detailed solutions.
20 -Be precise and answer as short as possible.
21
22 ANSWER this question every time:
23 - How does it fit the core experience?
24 - What sets it apart from different games?
25 - What is the "Anti-Trope" choice, and can it be incorporated?
26
27 GAME TOPIC:
28 {state['topic']}
29 CONSTRAINTS:
30 {state['constraints']}
31 ALREADY ACCEPTED:
32 {state["decisions"]}
33 """

```

Listing 4.2: Prompt snippet of the SystemMessage from the artist agent

The most advanced prompt is used by the iterative core. The messages of the game designer, artist, and programmer all follow the same structural composition. They all contain one SystemMessage and one HumanMessage to split the functionality. The SystemMessage starts with the persona guidelines for each agent. These are high-concept definitions of who they are, what they must do, and how to behave in different situations. This can range from how many sentences they should output to the specific tone of their voice. After the persona is defined, the research data is integrated into the text. Along with the data, design "laws" for the research are included so the LLM uses them correctly. The model needs to know that it should not use the random word names merely as flavor

#### 4. *Creating Game Design Documents with LLMs*

text, as they often use them without meaning. They might take a word and use it as a game mechanic without considering its definition or how it could be implemented. To ensure that the research truly influences the design, three laws were necessary. The first is that they cannot type the word itself, which avoids lazy association by the LLM. The second law is that they must translate the literal dictionary definition into the game, which helps them derive original ideas from the concept of the word. The last law is that they must state in each of their approaches which functional logic from the research is being used, ensuring the agents actively apply the concepts in their conversations.

After the research input, general constraints are described so that the agents behave accordingly. The rules "Do NOT praise other agents," "Do NOT summarize previous discussion," and "Be concise and specific with detailed solutions" are technical optimizations necessary to reduce token length. By explicitly banning summaries and praise, the agents must use their computational budget for novel content creation. The rules "Do NOT restate already accepted ideas" and "Do NOT contradict" are required so that the global state of already accepted answers serves as a fixed boundary, ensuring consistency and preventing agents from overriding established design rules.

After these general guidelines, each of the four core agents has specific constraints and formatting rules to support their persona. With "Answer with APPROACH 1:", the designer agent follows a structural guideline that helps convert unstructured language into semi-structured data. This helps the other agents identify where one approach ends and the next begins. The constraint "State how each researcher concept can be applied" further improves the rate at which the designer uses the researched words as input. The artist follows more creatively focused constraints. By answering "- How does it fit the core experience?", the artist agent must consider previous decisions, producing a more coherent creative idea. By asking the model "- What sets it apart from other games?" and "- What is the 'Anti-Trope' choice, and can it be incorporated?", it thinks about games in general, helping to generate ideas that deviate from the norm. The programmer agent utilizes a different strategy by assigning a feasibility score for each approach using the labels [CRITICAL / RISKY / FEASIBLE]. This assists the evaluator agent in assessing the answers. This is further accomplished by having the programmer answer two questions: "How can this be achieved technically?" and "Which can be achieved easiest technically?".

The evaluator agent has the most specialized rules due to his function of finalizing or continuing the questions. To save the current answer, the evaluator's message must contain "FINAL" to indicate satisfaction with the response. When "CONTINUE" is contained in the response, the agents iterate again until the round limit is reached. Rules such as "CONTINUE if the answer is too broad or combines too many directions" support the decision-making process. With "YOU MUST FOLLOW THIS FORMAT EXACTLY. STATUS: FINAL or CONTINUE", the system ensures the output has the exact required format. When the last round of a question is reached, a `HumanMessage` is appended, instructing the evaluator to stop iterating and only output "STATUS: FINAL" alongside the last decision. All other agents in the core loop receive the same `HumanMessage` containing the responses of the other agents to facilitate communication.

The editor agent does not require as many rules as the iterative core, as its only job is

to summarize the answers and organize them into a structure. Three different sections are needed for a high-performing editor. First is a description of the input material and the task. "Do NOT invent new features" and "Do NOT contradict decisions" are essential so the editor remains faithful to the core decisions. The second section refers to the style the editor uses to create the GDD, stating the use of professional, full sentences without conversational language. "- Improve wording, flow, and structure" is used to polish the GDD so it does not rely on the exact phrasing of the evaluator. The last section is the GDDSTRUCTURE, which is a predefined template without answers, ensuring every document follows the same format. The style is saved as a Markdown file, which is human-readable and easily modifiable. The image agent represents the multimodal synthesis and does not alter the text of the GDD. Style and context are defined to generate well-suited results. The style ensures a digital gameplay image with UI elements showing moment-to-moment gameplay. The context includes all design decisions finalized by the evaluator, with additional focus given to visual answers.

## 4.4. GDDState Structure

The GDDState is the shared brain of the multi-agent setup. It stores all information needed for the agents and acts as a centralized blackboard. GDDState inherits from TypedDict, which ensures data integrity—meaning every agent receives a predictable payload with fixed value types. At the start of execution, the GDDState is initialized with a set of empty variables.

```

1 class GDDState(TypedDict):
2     topic: str
3     constraints: str
4
5     randomWord: str
6     wordDefinitions: str
7
8     questionIndex: int
9     currentQuestion: str
10
11     designerNotes: str
12     artistNotes: str
13     programmerNotes: str
14     evaluatorNotes: str
15
16     decisions: Dict[str, str]
17     conversationLog: list[str]
18
19     roundLimit: int
20     images: Dict[str, list[str]]

```

Listing 4.3: GDDState definition

The topic and constraints variables are used for user input, storing the intended project parameters and operational boundaries. The randomWord variable is used by the research agent to receive random input. He stores his research and definitions of the words in

#### 4. *Creating Game Design Documents with LLMs*

wordDefinitions, which is then utilized by the iterative core. The variable questionIndex is required for storing the current question number as an integer, while currentQuestion stores the textual representation of the current task.

Agent-specific notes are stored as strings, but they do not contain the entire dialogue from all previous questions; they only store the input for the current question to ensure the information does not become diluted. This leads to fewer hallucinations and less recursive logic. To provide agents with the answers from previous rounds—which is vital for consistent output—a decision structure is used. The key is used to store the questions, while the value stores the generated answers from the evaluator agent. The conversationLog is used solely for debugging and stores the entire conversation as a single string. roundLimit defines the maximum number of iterations for a single question, whereas the image structure stores any generated images. In summary, the GDDStruct is the central structure for accessing and storing information throughout the entire execution pipeline.

### 4.5. Phased Questioning and GDD Templates

To ensure that the agents model a complete Game Design Document, 50 questions were required. This forces the models toward atomic reasoning and better outcomes. These questions are separated into seven phases to handle different aspects of game design independently. The first phase, titled "Core Idea and Goals," lays the foundation for all subsequent phases. In this phase, the core experience, player motivations, and the cultural impact of the game are discussed. Phase 2 focuses on Gameplay, handling questions such as whether the game is single-player or multiplayer, and establishing challenges, core gameplay loops, and level structures.

The next section focuses on player interaction, with questions like "Who is the player character?" or "What input methods and control schemes are used?". Phase 4 concretizes the setting and the world in which the game is set, asking questions about non-playable characters (NPCs), antagonists, and the conflict the player faces. Visual and audio-related questions are answered in Phase 5, providing insights into the visual tone, atmosphere, UI/UX design, and how the music supports the theme. Phase 6 focuses on technical details, such as specifying the game engine, mod support, and infrastructure. The final section, "Monetization and Release," contains questions regarding marketing and monetization plans before and after release. All answered questions are then translated by the editor agent into the predefined GDD structure. All phases are converted into separate chapters, except for marketing and monetization, which are included as a subsection in the first chapter.

## 5. Evaluation

### 5.1. Experimental Setup and Methodology

To evaluate the Game Design Documents created by AI, a questionnaire was developed to test the AI-generated GDDs against human-authored ones. A blind test was conducted with three human-authored documents against three AI-generated GDDs. To present them objectively, standardizations were applied to eliminate aesthetic bias. The human documents were first converted to Markdown files, and then the editor agent formatted the text into the same categories as the AI-generated ones. It was essential that the language and sentence structure remained unchanged so they could be accurately compared. Additionally, all images were removed to ensure each document looked the same at first glance. Finally, some names in the human documents were altered to prevent participants from recognizing them, which would bias the results.

For the human-authored documents, the internet was searched for suitable examples. The selection strategy was to obtain three documents with maximum diversity in genre and technical scope to test the versatility of the agentic workflow. Once found, these served as the foundation for the experimental group. The inputs for the three AI-generated documents were based on the core parameters of the human versions, including budget, team size, complexity, and genre. This mirroring ensured the evaluation was not influenced by the complexity of the game itself.

The first document is from Star Raiders 2, a space combat simulator released for Atari 8-bit systems in 1986. As the game is older and the target audience for the questionnaire is younger, no name changes were made. The topic and constraints in the AI-generated version were kept identical to the Star Raiders 2 baseline. These constraints act as a consistent frame of reference that forces the AI to operate within realistic development boundaries rather than achieving a perfect historical reconstruction.

```
1 """ "topic": "A space combat simulator. The player must navigate the
    galactixy to intercept invading fighter wings and destroy enemy base
    ships.",
2 "constraints": "Budget: $85,000; Timeline: 10 Months; Team Size: 2, 1
    Programmer/Designer, 1 Artist/Musician); Target: Atari 8-bit" """
```

Listing 5.1: Input parameters for the AI-generated Star Raiders 2 counterpart.

The next document is from the original Diablo. This is an isometric action RPG developed by Blizzard and published in 1996. Due to its popularity, all mentions of the game's name were replaced by "Hellfire." Listing 5.2 shows the prompt used by the AI agents to generate a Diablo-like GDD. This game was chosen to see how effectively the agents could translate a dark gothic fantasy atmosphere into game mechanics.

## 5. Evaluation

```
1 """ "topic": "An action-oriented dungeon crawler set in a gothic dark
    fantasy world.",
2 "constraints": "Budget: $3,000,000; Timeline: 12 Months; Team Size:
    15-20; Target: Windows 95" """
```

Listing 5.2: Input parameters for the AI-generated Diablo counterpart.

The final game used for evaluation is Saints Row 2, a "AAA" open-world action-adventure game published in 2008. This title represents the sandbox genre, where multiple mechanics like physics and NPC behavior interact. To anonymize the document, the title was replaced by "Undercover." These constraints provide a modern AAA production environment to test the scalability and structural integrity of the generated GDD.

```
1 """ "topic": "An open-world action-adventure game where the player awakes
    from a coma, fighting rival gangs and a corrupt corporation.",
2 "constraints": "Multi-million dollar AAA production; Timeline: 3 years;
    Team Size: 200+ employees; Target: Xbox, PlayStation, PC" """
```

Listing 5.3: Input parameters for the AI-generated Saints Row 2 counterpart.

Participants were selected from the gaming community, specifically targeting long-term gamers, individuals active in game development, or those with a background in computer science. This ensured that evaluators possessed the necessary domain literacy to understand the intricacies of game design documents.

### 5.2. Evaluation Metrics and Questionnaire Design

To quantify the quality of AI-generated game design documents, a structured survey was developed. This section explains the different questions used and how the participants could rate them. The evaluation utilizes a 4-point Likert scale with options from "Strongly Disagree," "Disagree," and "Agree" to "Strongly Agree." An even-numbered scale was used to remove the central tendency bias. With this scale, the neutral midpoint is removed, which results in participants having to choose a more definitive response.

Each participant evaluated the six game design documents based on three criteria: Creativity, Cohesion, and Clarity. These three categories were chosen to reflect the core requirements of most professional game design documents.

**Dimension 1: Creativity** The creativity statements are used to assess the originality of the core concept. They measure whether the AI agents generate unique ideas or if they only copy tropes from existing games. The following four statements are used in the questionnaire:

- The game concept feels original rather than a derivative of existing titles.
- The narrative or world-building elements are engaging and immersive.
- The mechanics described feel innovative in the genre.
- The aesthetic vision of the game is clearly conveyed through the descriptions.

**Dimension 2: Cohesion** Cohesion is used as a metric to test the AI models for hallucinations and context retention. Often, AI agents lose track of constraints and technical limitations, especially when working with long inputs and text strings. These statements are designed to highlight and measure these discrepancies:

- The game’s logic is internally consistent.
- The level of detail is appropriate (neither too vague nor over-explaining).
- The game described feels like it could actually be built and released.
- There are no visible contradictions in the rules or features described.

**Dimension 3: Clarity** This category evaluates the communicative effectiveness and professionalism of the writing. If a game design document cannot be read or is not easy to understand, any creative idea is wasted. These statements measure if the generated text can truly serve as a functional blueprint for a professional game design team.

- The core gameplay loop is clearly defined and easy to understand.
- The descriptions are precise enough to prevent misunderstandings.
- The writing style is professional and avoids ambiguous or fluffy language.
- The mechanics are described clearly enough that I can imagine how they work.

To capture specifics which the numerical data might overlook, the questionnaire includes two qualitative questions at the end of each GDD. These two questions serve as a diagnostic tool to identify where the agentic workflow was superior to the human-made document and where the AI documents might hallucinate.

- What was the single most impressive part of this document?
- What was the single most confusing part of this document?

### 5.3. Statistical Analysis and Inference

To evaluate the outcome of the questionnaire, a quantitative analysis with descriptive and inferential statistics was conducted. The primary objective was to determine if a significant difference exists between the AI and human-made documents in the three categories: Creativity, Cohesion, and Clarity. The following hypotheses were formulated:

- **Null Hypothesis ( $H_0$ ):** There is no statistically significant difference in the perceived quality (Creativity, Cohesion, and Clarity) between AI-generated Game Design Documents and human-authored Game Design Documents.

## 5. Evaluation

- **Alternative Hypothesis ( $H_1$ ):** There is a statistically significant difference in the perceived quality (Creativity, Cohesion, and Clarity) between AI-generated Game Design Documents and human-authored Game Design Documents.

To prepare the data for statistical analysis, each Likert scale response was assigned a number from 1 (Strongly Agree) to 4 (Strongly Disagree). The analysis was performed in Microsoft Excel, where a Welch’s t-test was conducted. This test was necessary because the variances of the two sample groups were not equal. A two-tailed distribution was used with a significance level of  $\alpha = 0.05$ , which detects significant differences in both directions. Additionally, Cohen’s d was calculated to provide context to the  $p$ -value, indicating whether the results are practically meaningful in real-world game development. The results of the inference tests are as follows:

Table 5.1.: Statistical Comparison of AI-Generated vs. Human-Authored GDDs

Dimension	AI Mean	Human Mean	$p$ -value	Cohen’s $d$
Creativity	1.89	2.73	$4.26 \times 10^{-13}$	0.99
Cohesion	2.33	1.83	$1.67 \times 10^{-6}$	0.63
Clarity	2.70	1.98	$2.90 \times 10^{-10}$	0.85

Note: Mean scores based on a 4-point Likert scale (1 = Strongly Agree, 4 = Strongly Disagree).

All  $p$ -values are lower than 0.05, so the Null Hypothesis  $H_0$  is rejected in all three dimensions. This demonstrates that there are statistically significant differences in all categories between the AI and human-made GDDs. This is further supported by a high Cohen’s d in every category, suggesting that the observed differences are not only statistically significant but also hold substantial practical importance for game development workflows.

### 5.4. Discussion and Qualitative Findings

The resulting data shows that there are differences in each of the three categories. Interestingly, the directions of the differences provide a nuanced look at automated GDD generation.

Surprisingly, the documents generated by AI were perceived as significantly better than the human-made ones in terms of creativity. This shows that AI is able to generate highly conceptual ideas which feel more innovative and original. While the creative side of the AI workflow outperformed the human documents, the dimensions of clarity and cohesion were significantly outperformed by the humans. This suggests that the capacity to establish internally consistent rules and professional blueprints remains a domain of human superiority.

The qualitative feedback also highlighted specific diagnostic differences. The most impressive parts for many participants in the AI documents were the interesting core ideas, the world-building, and the atmosphere. This showcases the AI’s proficiency in creative conceptualization.

#### 5.4. Discussion and Qualitative Findings

- The player experience and aesthetics chapter clearly conveyed a vision and feel for the game that I could instantly imagine.
- The idea behind it seems very innovative, specifically having neurosurgery as a mechanic of permanent change in your character.
- The atmosphere.

However, the qualitative answers often noted how AI models use new keywords for ideas and mechanics without specifying what they are really about. This highlights how AI uses "black box" words where the internal logic is never unpacked. It shows how capable AI generation is for high-level concepts, but how lacking the models are at generating user-centric documentation.

- Also the extensive and specific vocabulary that includes terms that are seldom used in day-to-day speech.
- The text is written in detail, explaining everything needed, but the way it is written and used is not clear or easy to understand.
- I didn't understand the whole thing about "orbital crossing" and how the alphanumeric codes are used for multiplayer.

For the human-made documents, many participants favored their understandable descriptions, their simplicity, and their fleshed-out details.

- The detail of the implementations of the movement and combat of the character.
- Understandable description of the elements of the game mechanics.
- The clear vision of the game.

Those qualitative statements help to contextualize the numerical decisions of the participants. Nevertheless, there are also some limitations to this study.

To ensure a standardized test, all images, diagrams, and UI mockups were removed to ensure all documents looked the same. This hindered the human documents from demonstrating their full potential, which traditionally relies on more than text. This could decrease their perceived creativity because the best way to communicate atmosphere is often with images and concept art.

Another limitation was the use of older human-made game design documents. Due to the unavailability of newer documents, older ones had to be part of the questionnaire. This creates a chronological gap between the documents which could potentially change the results. Games like *Star Raiders 2* or *Diablo* were huge milestones in the history of gaming. Their work was so influential that many subsequent games copied their mechanics, which participants could perceive as less creative because they are familiar with the genres those games helped to define.

These statistical and qualitative findings provide the basis for answering the research questions which were established at the beginning of this study.

## 5. Evaluation

- To what extent can an AI agentic workflow produce original game concepts that are perceived as innovative rather than derivative?
- How effective is the AI at maintaining internal logic and technical consistency compared to the fleshed-out details of professional human blueprints?
- How does the communicative quality and professional writing style of AI-generated GDDs compare to human-authored documents?

RQ1 answers the question of whether a fully automated game design document creation process can produce original concepts. AI workflows are exceptionally strong in creating interesting themes, core ideas, and atmosphere, which is reflected by the statistical and qualitative analysis. This shows how valuable AI can be for game designers iterating together on ideas or for brainstorming sessions.

RQ2 seeks to determine if AI GDDs can create and maintain the logical consistency and mechanical details needed for creating games. The study shows that with this implementation of automated GDD creation, the AI often reduces mechanics and advanced concepts to mere keywords. Therefore, no logical consistency could be established.

RQ3 focuses on the quality of the GDD. Many participants and the statistical data demonstrate that the AI documents are not precise enough to compete with human documents, even if text is the only communication medium used. This is also influenced by the inability to make strict decisions rather than relying on keywords. With truly fleshed-out mechanics and structures, AI could write about them effectively instead of relying on unnecessary fluff.

Further research must be put into the generation of precise mechanics and concepts. This could be accomplished by AI-generated questions which could be fed back into the design document. That would help to specialize the document to be more suited to the game's genre. For example, the evaluator agent could generate a question to clearly define a keyword used by the other agents. This would make the structure more dynamic and would lead the agents to truly explain their mechanics. Additional tools, like modeling languages or code, could be used in the workflow to solidify any suggested ideas. To achieve a more professional writing style, more prompt engineering is needed to produce a more user-oriented, fluff-avoiding game design document.

## 6. Conclusion

This thesis set out to explore how LLMs can be used for generating GDDs. Using the low-level frameworks of LangChain and LangGraph, a multi-agent workflow was set up to generate game design documents through agent orchestration and prompt engineering. With the use of specialized agent personas, it was possible to create single-purpose agents that communicate sequentially to generate and iterate over a set of design questions.

The evaluation was conducted with a 4-point Likert scale, where participants answered 12 questions across the dimensions of Creativity, Clarity, and Cohesion. Two qualitative questions were additionally asked. The results show that human-made documents are still superior in clarity and cohesion. The AI documents contain fluffy language and use interesting-sounding mechanics and ideas without clearly defining them. On the other hand, the AI documents are statistically significantly better in the originality dimension. Based on the results, they produce more interesting core ideas and aesthetic visions than the human-made ones.

Further work is needed to make AI-generated documents more human-like. A key area would be generating game-specific questions to eliminate keywords without meaning. To truly achieve human-like documents, image generation must be more integrated into the agent system. This could be done by feeding the image output as input to the designer agent, enabling images to be used as actual reference points. Furthermore, future research could focus on more complex workflows and more intricate agent personas and prompting to obtain more diverse and creative GDDs.



# Bibliography

- [1] Asad Anjum, Yuting Li, Noelle Law, M Charity, and Julian Togelius. The ink splotch effect: A case study on chatgpt as a co-creative game designer, 2024.
- [2] Andrew Begemann and James Hutson. Empirical insights into ai-assisted game development: A case study on the integration of generative ai tools in creative pipelines. *Metaverse*, 5(2):2568, 2024.
- [3] Antoine Bellemare-Pépin, François Lespinasse, Philipp Thölke, Yann Harel, Kory Mathewson, Jay A. Olson, Yoshua Bengio, and Karim Jerbi. Divergent creativity in humans and large language models. *arXiv preprint arXiv:2405.13012*, 2024.
- [4] Steph Buongiorno, Lawrence Jake Klinkert, Tanishq Chawla, Zixin Zhuang, and Corey Clark. Pangea: Procedural artificial narrative using generative ai for turn-based video games, 2024.
- [5] Pedro Henrique Luz de Araujo and Benjamin Roth. Helpful assistant or fruitful facilitator? investigating how personas affect language model behavior. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3605–3627. Association for Computational Linguistics, 2024.
- [6] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models, 2023.
- [7] Zhaopeng Feng, Jiayuan Su, Jiamei Zheng, Jiahan Ren, Yan Zhang, Jian Wu, Hongwei Wang, and Zuozhu Liu. M-MAD: Multidimensional multi-agent debate for advanced machine translation evaluation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7084–7107, Vienna, Austria, July 2025. Association for Computational Linguistics.
- [8] Tianyang Gu, Jingjin Wang, Zhihao Zhang, and HaoHong Li. Llms can realize combinatorial creativity: Generating creative ideas via llms for scientific research, 2025.
- [9] Dongge Han, Camille Couturier, Daniel Madrigal Diaz, Xuchao Zhang, Victor Rühle, and Saravan Rajmohan. Legomem: Modular procedural memory for multi-agent llm systems for workflow automation, 2025.

## Bibliography

- [10] Lanxiang Hu, Mingjia Huo, Yuxuan Zhang, Haoyang Yu, Eric P. Xing, Ion Stoica, Tajana Rosing, Haojian Jin, and Hao Zhang. *lmgame-bench: How good are llms at playing games?*, 2025.
- [11] Sharp John and Colleen Macklin. *Iterate*. The MIT Press, 1 edition edition, 2019.
- [12] Sungeun Kim and Dongsuk Oh. Evaluating creativity: Can llms be good evaluators in creative writing tasks? *Applied Sciences*, 15(6), 2025.
- [13] Tom Kortenbach, Yuan Yin, Milene Guerreiro Gonçalves, Rebecca Anne Price, and Pan Wang. The relation between humans and llms in the creative act. *Journal of Creativity*, 36(1):100114, 2026.
- [14] Vanya Bannihatti Kumar, Divyanshu Goyal, Akhil Eppa, and Neel Bhandari. Curiosity-driven llm-as-a-judge for personalized creative judgment, 2025.
- [15] Danrui Li, Sen Zhang, Sam S. Sohn, Kaidong Hu, Muhammad Usman, and Mubbasir Kapadia. Cardiverse: Harnessing llms for novel card game prototyping, 2025.
- [16] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, Miami, Florida, USA, November 2024. Association for Computational Linguistics.
- [17] Gionnieve Lim and Simon T. Perrault. Rapid aideation: Generating ideas with the self and in collaboration with large language models, 2024.
- [18] Zhuoran Lu, Qian Zhou, and Yi Wang. Whatelse: Shaping narrative spaces at configurable level of abstraction for ai-bridged interactive storytelling, 2025.
- [19] Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, Rongcheng Tu, Xiao Luo, Wei Ju, Zhiping Xiao, Yifan Wang, Meng Xiao, Chenwu Liu, Jingyang Yuan, Shichang Zhang, Yiqiao Jin, Fan Zhang, Xian Wu, Hanqing Zhao, Dacheng Tao, Philip S. Yu, and Ming Zhang. Large language model agent: A survey on methodology, applications and challenges, 2025.
- [20] Marlene Lutz, Indira Sen, Georg Ahnert, Elisa Rogers, and Markus Strohmaier. The prompt makes the person(a): A systematic evaluation of sociodemographic persona prompting for large language models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 23212–23237, Suzhou, China, November 2025. Association for Computational Linguistics.
- [21] Alessandro Maisto. Collaborative storytelling and llm: A linguistic analysis of automatically-generated role-playing game sessions, 2025.

- [22] Xinyu Mao, Wanli Yu, Kazunori D Yamada, and Michael R. Zielewski. Procedural content generation via generative artificial intelligence, 2024.
- [23] Pronita Mehrotra, Aishni Parab, and Sumit Gulwani. Enhancing creativity in large language models through associative thinking strategies, 2024.
- [24] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications, 2025.
- [25] Vinay Samuel, Henry Peng Zou, Yue Zhou, Shreyas Chaudhari, Ashwin Kalyan, Tanmay Rajpurohit, Ameet Deshpande, Karthik R Narasimhan, and Vishvak Murahari. PersonaGym: Evaluating persona agents and LLMs. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 6999–7022, Suzhou, China, November 2025. Association for Computational Linguistics.
- [26] Matthew Sidji and Matthew Stephenson. Prompt engineering chatgpt for codenames. In *2024 IEEE Conference on Games (CoG)*, pages 1–4, 2024.
- [27] Yu-Min Tseng, Yu-Chao Huang, Teng-Yun Hsiao, Wei-Lin Chen, Chao-Wei Huang, Yu Meng, and Yun-Nung Chen. Two tales of persona in llms: A survey of role-playing and personalization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16612–16631, Miami, Florida, USA, 2024. Association for Computational Linguistics.
- [28] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.
- [29] Yunge Wen, Chenliang Huang, Hangyu Zhou, Zhuo Zeng, Chun Ming Louis Po, Julian Togelius, Timothy Merino, and Sam Earle. All stories are one story: Emotional arc guided procedural game level generation, 2025.
- [30] Zelai Xu, Chao Yu, Fei Fang, Yu Wang, and Yi Wu. Language agents with reinforcement learning for strategic play in the werewolf game. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 55434–55464. PMLR, 21–27 Jul 2024.
- [31] Ye Ye. Task memory engine: Spatial memory for robust multi-step llm agents, 2025.
- [32] Yunpu Zhao, Rui Zhang, Wenyi Li, and Ling Li. Assessing and understanding creativity in large language models. *Machine Intelligence Research*, 22(3):417–436, 2025.



## A. Appendix

```
1 DESIGN_QUESTIONS = [  
2  
3 # Phase 1      Core Idea and Goals  
4 "What is the central idea of the game?",  
5  
6 "What human abilities will the players learn from playing it?",  
7  
8 "What player motivations does the game address?",  
9  
10 "What genre does the game belong to?",  
11  
12 "How will the game stand out in this genre",  
13  
14 "What is the social and cultural impact of this game in the real world?",  
15  
16 "What does moment to moment gameplay look like?",  
17  
18 "What age group and player types are targeted?",  
19  
20 "On which platforms and devices will the game be available?",  
21  
22 "What is the games name",  
23  
24  
25 # Phase 2      Gameplay  
26  
27 "Is the game singleplayer, multiplayer, or both?",  
28  
29 "What is the core gameplay loop?",  
30  
31 "How is challenge introduced and increased?",  
32  
33 "What are the primary player goals and objectives?",  
34  
35 "What rules govern the gameplay experience?",  
36  
37 "What systems limit or enable player actions?",  
38  
39 "Are there levels or missions structured?",  
40  
41 "How does the difficulty curve evolve over time?",  
42  
43  
44 # Phase 3      Player Interaction  
45
```

## A. Appendix

```
46 "Who is the player character?",
47
48 "How does the player control the game?",
49
50 "What input methods and control schemes are used?",
51
52 "How does the user interface support player actions?",
53
54 "How does the game communicate progress to the player?",
55
56 "What feedback systems (visual, audio, rewards) exist?",
57
58 "What types of player-to-player interactions exist?",
59
60
61 # Phase 4      World, Story, and Setting
62
63 "Where does the game take place?",
64
65 "What kind of world or universe does the game feature?",
66
67 "Is there a story or narrative?",
68
69 "Who are the main NPCs and antagonists?",
70
71 "What is the main conflict or narrative goal?",
72
73 "Do player choices affect the story or ending?",
74
75
76 # Phase 5      Aesthetics and Audio
77
78 "What art style does the game use (realistic, cartoon, pixel, 2D, 3D)?",
79
80 "What visual tone and atmosphere should the game convey?",
81
82 "What emotions should the visuals evoke?",
83
84 "What mood should the music create?",
85
86 "How does sound design support gameplay and immersion?",
87
88 "How are menus, HUD, and notifications presented?",
89
90 "How does the UI/UX design improve clarity and accessibility?",
91
92
93 # Phase 6      Technical Foundation
94
95 "What is the target platform and its hardware limitations?",
96
97 "Which game engine and tools will be used?",
98
```

```

99 "What technical constraints exist per platform?",
100
101 "What is the target frame rate",
102
103 "How are save/load systems handled?",
104
105 "Is networking or multiplayer infrastructure required?",
106
107 "How does the game handle world loading and asset streaming?",
108
109 "Is mod support or extensibility planned?",
110
111
112 # Phase 7      Monetization and Release
113
114 "What monetization model is used (premium, free-to-play, in-game
      purchases)?",
115
116 "How does monetization avoid harming player experience?",
117
118 "How will the game be marketed and branded?",
119
120 "What makes the game appealing in trailers and screenshots?",
121
122 ]

```

Listing A.1: GDD Questions.

```

1 GDDSTRUCTURE = ""
2 # GAME DESIGN DOCUMENT
3
4 ## 1. Concept & Vision
5 ### 1.1 Overview
6 - **Game Title:** [The game's name]
7 - **Genre:** [Genre and how it stands out]
8 - **Core Concept:** [Central idea and core gameplay loop]
9 - **Target Audience:** [Age group, player types, and motivations]
10 - **Intended Experience:** [Human abilities learned, emotions evoked]
11 - **Social & Cultural Impact:** [Impact in the real world]
12
13 ### 1.2 Market & Platform
14 - **Platforms:** [Devices and platform availability]
15 - **Monetization:** [Model used and player experience protection]
16 - **Marketing & Branding:** [Branding strategy and trailer appeal]
17
18 ---
19
20 ## 2. Gameplay Mechanics
21 ### 2.1 Core Systems
22 - **Player Goals:** [Primary objectives and rules]
23 - **Moment-to-Moment:** [Description of gameplay]
24 - **Action Economy:** [Systems that limit or enable actions]
25 - **Progress & Feedback:** [Communication of progress, rewards, and
      feedback systems]

```

## A. Appendix

```
26
27 ### 2.2 Difficulty & Progression
28 - **Challenge:** [How challenge is introduced]
29 - **Difficulty Curve:** [Evolution of difficulty over time]
30 - **Structure:** [Level/Mission structure]
31
32 ---
33
34 ## 3. The Player Experience
35 ### 3.1 Interaction & Controls
36 - **Player Character:** [Identity and role of the player]
37 - **Control Scheme:** [Input methods and control game logic]
38 - **User Interface:** [HUD, menus, and how UI supports actions]
39 - **UX Design:** [Clarity, accessibility, and notifications]
40
41 ### 3.2 Social Interaction
42 - **Multiplayer:** [Singleplayer/Multiplayer status]
43 - **Player-to-Player:** [Types of interactions]
44
45 ---
46
47 ## 4. World & Narrative
48 ### 4.1 Setting & Universe
49 - **The World:** [Location, universe, and atmosphere]
50 - **Visual Tone:** [Art style and emotional visuals]
51
52 ### 4.2 Story & Characters
53 - **Narrative:** [The story, main conflict, and goals]
54 - **Cast:** [Main NPCs and Antagonists]
55 - **Agency:** [Impact of player choices on story/ending]
56
57 ---
58
59 ## 5. Aesthetics & Audio
60 ### 5.1 Visuals
61 - **Art Style:**
62 - **Atmosphere:**
63
64 ### 5.2 Audio
65 - **Music:** [Mood and composition goals]
66 - **Sound Design:** [Support for gameplay and immersion]
67
68 ---
69
70 ## 6. Technical Foundation
71 ### 6.1 Systems & Performance
72 - **Engine:** [Game engine and tools]
73 - **Technical Constraints:** [Platform-specific limits]
74 - **Performance:** [Resolution and FPS targets]
75
76 ### 6.2 Infrastructure
77 - **Data:** [Save/Load systems]
78 - **Networking:** [Multiplayer infrastructure]
```

```
79 - **Extensibility:** [Mod support and future-proofing]  
80 """
```

Listing A.2: GDD Markdown Template