Text2VRScene: Exploring the Framework of Automated Text-driven Generation System for VR Experience

Zhizhuo Yin [*]
The Hong Kong University of Science and Technology (Guangzhou)

Yuyang Wang [†]
The Hong Kong University of Science and Technology (Guangzhou)

Theodoros Papatheodorou [‡]
The Hong Kong University of Science and Technology (Guangzhou)

Pan Hui[§]
The Hong Kong University of Science and Technology (Guangzhou)

Abstract

With the recent development of the Virtual Reality (VR) industry, the increasing number of VR users pushes the demand for the massive production of immersive and expressive VR scenes in related industries. However, creating expressive VR scenes involves the reasonable organization of various digital content to express a coherent and logical theme, which is time-consuming and labor-intensive. In recent years, Large Language Models (LLMs) such as ChatGPT 3.5 and generative models such as stable diffusion have emerged as powerful tools for comprehending natural language and generating digital contents such as text, code, images, and 3D objects. In this paper, we have explored how we can generate VR scenes from text by incorporating LLMs and various generative models into an automated system. To achieve this, we first identify the possible limitations of LLMs for an automated system and propose a systematic framework to mitigate them. Subsequently, we developed Text2VRScene, a VR scene generation system, based on our proposed framework with well-designed prompts. To validate the effectiveness of our proposed framework and the designed prompts, we carry out a series of test cases. The results show that the proposed framework contributes to improving the reliability of the system and the quality of the generated VR scenes. The results also illustrate the promising performance of the Text2VRScene in generating satisfying VR scenes with a clear theme regularized by our well-designed prompts. This paper ends with a discussion about the limitations of the current system and the potential of developing similar generation systems based on our framework.

Index Terms: Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Virtual reality; Human-centered computing—Human computer interaction (HCI)—Interaction techniques—Text input

1 Introduction

The popularity and easy accessibility of commercial head-mounted displays (HMDs) have encouraged a rising demand for immersive and high-quality virtual reality (VR) content creation. With the surging needs, how to massively produce high-quality VR experiences becomes a valuable problem.

However, the VR experience is a cohesion of creativity from diverse fields, which requires collaboration among various disciplines. In the creative disciplines, a high-quality VR experience generally consists of 3D models and 2D textures and images. The creation of these digital contents is labor-intensive even for proficient creators. Moreover, the collaboration between creative artists and engineers is also important since expressing aesthetic themes with the above digital content should be via the medium of computer codes. The above features of the VR experience make it a complex undertaking and hard to automate for massive production.

Recently, some advances in artificial intelligence shed light on the possibility of automating such complex collaborative processes. The rapid emergence of digital assets generation techniques such as diffusion models [12, 36, 38], Generative Adversarial Networks (GANs) [22, 25, 35], etc. has enabled automatic generation of digital contents such as 2D images and 3D models using natural language. These studies showcase the potential for automatically and efficiently creating high-quality digital content. However, they still fail to generate large-scale virtual scenes with massive interrelated objects due to limited ability to conduct logic reasoning and analyze the spatio-temporal relations among different objects.

The breakthroughs in Large Language Models (LLMs), like OpenAI's GPT-4, have achieved the advanced capability to comprehend natural language [3, 19, 27], generate a reasonable, and rational response or even decent coding [8, 9, 15, 17, 32]. We notice that with the extraordinary capability [19, 27] of LLMs to comprehend natural language and conduct logical reasoning, the LLMs can be utilized to comprehend the user prompt about the desired VR scene, instruct the generation of digital assets through text prompts, and generate the source code that organizes different digital contents with reasonable storytelling. We gain inspiration and further propose that LLMs can be utilized as a core in an automated text-driven generation system and is able to organize all kinds of digital assets to construct creative expression. In such LLMs-based system, developers should only manually define and fine-tune the prompts for LLMs. Once the prompts are confirmed, the generation process is fully automated by LLMs without extra interference.

However, several concerns about the limitation of existing LLMs such as randomness in response, inconsistent capability in reasoning tasks, and hallucination have been raised by recent literature [3, 5, 29, 47], posing a significant threat to the reliability of existing LLMs for developing automated system.

To address the above problem, we first conduct a qualitative research study on the limitations of LLMs as a component in an automated system. We combined the study with the analysis of existing literature to reach a more systematic insight for identifying the limitations of LLMs. Then we propose a pioneering LLM-based automated system framework for increasing reliability and optimizing the utilization of LLMs.

With the proposed LLM-based automated system framework, we have meticulously crafted and engineered a VR scene generation system based on the above principles called Text2VRScene. The Text2VRScene system is capable of robustly generating immersive virtual experiences based on a simple text. To validate the capabilities of the Text2VRScene system and evaluate the effectiveness of the proposed framework, we have conducted a series of case studies, including a reliability study and performance analysis.

All studies demonstrate the effectiveness of our proposed framework in increasing the reliability of an automated system. Meanwhile, these studies also demonstrate the effectiveness of the crafted prompts in our system in generating expressive VR experiences.

[*]e-mail: zyin190@connect.hkust-gz.edu.cn
[†]e-mail: yuyangwang@hkust-gz.edu.cn (Corresponding author)
[‡]e-mail: theodoros@ust.hk
[§]e-mail: panhui@ust.hk (Corresponding author)

In this paper, we identified the potential of LLMs in an automated generation system, which prompts text-based AIGC models and organizes the generated assets with code. The contributions of this paper can be summarized as follows:

- We summarize the major pain points of utilizing LLMs for automated code generation by performing a review of existing literature and a semi-structured interview with experts on LLMs. Then we propose a pioneering LLM-based automated system framework that aims to address the summarized pain points from existing literature and a semi-structured interview. The framework can mitigate the pain points of using ChatGPT for an automated generation system.

- We developed Text2VRScene based on our proposed framework, an LLMs-based VR experience generation system, by meticulous prompt tuning. Text2VRScene is able to depict the scene from a user prompt, determine digital assets needed, prompt AI generation models, generate source code to organize the scene based on the previously determined scene, and thus generate the VR scene the user desires.

- We conducted a series of case studies on our developed Text2VRScene, which validate the effectiveness of our proposed automated system framework in improving the reliability and consistency of LLMs and demonstrate the feasibility of our Text2VRScene in generating expressive VR scenes.

2 Related Work

2.1 Generation Models

Diffusion models [20,38,42,43] employ a UNet backbone to enable image generation, by adding noise as a forward diffusion process and reversing this process to recover the data and learn the latent distribution of the data sets. [40] treat diffusion model for image generation and transformers as a language encoder. By generating images based on encoded text, it achieves state-of-the-art performance in text-to-image synthesis tasks. Furthermore, advances in diffusion models have also spurred the development of text-to-3D methods [11,26,36,49]. [22] is a text-to-3D synthesis method, which is capable of generating latent vectors of 3D models after a two-stage training in an encoder and diffusion model. Then the latent vectors can be rendered as both textured meshes and neural radiance fields.

Generative models [12, 22–24, 48] are based primarily on the framework of generative adversarial networks (GAN) [18]. GANs consist of a generator that produces items and a discriminator that distinguishes between real and generated inputs. After training, the generator becomes capable of generating realistic items. In image generation tasks, [23] proposes using a mapping network to generate style and a synthesis network to generate images, both under the structure of a progressive growing GAN (PG-GAN). The results demonstrate the high quality of the generated images.

All of the above asset generation models require a language model to encode the natural language into a latent vector. The language model is a classic type of AI model with a long history [7,39]. For the task of language generation, it is typically implemented as a conditional probabilistic model that predicts the next token of an incomplete sentence based on previous tokens: $c_i \sim p(c_i|c_1,\ldots,c_{i-1};\theta)$, where c_i represents the i-th token in the text sequence, and θ denotes the parameters of the language model.

LLMs are a specific kind of language model that possesses over a hundred billion parameters, such as GPT-3, Gopher [37], LaMDa [46], and others. LLMs demonstrate powerful capabilities in understanding natural language, logic analysis, and reasoning [27], benefiting from the emergent effect [50]. To fully exploit the capabilities of LLMs, prompt engineering [3,51] is required.

In addition to generating natural language responses, LLMs can be equipped with automatic techniques such as code generation [9,15], program repair [15,21], and code summarization [1]. These techniques further enhance the ability of LLMs to manage and organize automated generation systems.

2.2 Content generation in VR

Content generation is of paramount importance for virtual reality, as it lays the foundation for immersive experiences that define the essence of this technology. The current generation of VR experiences is heavily based on manual operations [30] within software development tools such as Blender [4], 3ds Max [2], Unity [45], and others. This process necessitates collaboration among 3D modeling artists, texture artists, painters, programmers, and script designers.

Although some studies [10,16,28,52] are focused on generating 3D scenes, which is applicable to VR scenarios, these methods are limited to generating static indoor scenes without dynamics. Consequently, they fail to deliver the fantastic and immersive experiences that users have yet to witness, and they are unable to express creativity through the dynamic of objects. [10] is a 3D scene generation system with natural language. It utilizes a probabilistic model to parse natural language and employs a spatial knowledge base to infer the arrangement of objects. However, this system can only generate indoor scenes limited to several pre-defined objects such as tables, desks, chairs, etc., which will cause limited diversity and unsatisfying topics of the scene to users.

In more recent research [13,14,44], attempts have been made to generate VR scenes. [14] propose to use the indoor environment to generate the virtual world, by manipulating the database of Unity's default VR world. [13] propose a method that generates staircases according to the physical world that allows users to walk continuously on them. However, the range of scene types of these methods remains constrained in the Unity database and cannot be customized by users, resulting in a gradual loss of freshness.

As a result, these studies fall short of meeting the growing demand for diversified VR experiences that users are looking for. We propose that the main reason for the lack of diversity is that existing studies cannot generate arbitrary digital assets that users require. In our proposed system, with the help of text-prompted digital content generation models, the digital models that users require can be generated in real-time and reasonably organized to express the story.

3 Framework of LLMs-based Automated System

In this section, our goal is to identify the main limitations of LLMs in an automated system. Then we will propose a framework to mitigate the above-identified limitations and ensure the reliability of LLM in automated VR generation systems.

Recently, some literature [3,5,29,47] discredits current LLMs due to some limitations, such as their poor performance in multi-hop reasoning, potential bias, inability to process long-term dependencies, etc. However, in the context of the deployment of LLMs in automated systems, these limitations may not perfectly match the requirements of automated systems, as the conclusions of the existing limitations of LLMs are derived primarily from experimental dialogues with researchers instead of machines. As a result, we conducted qualitative research as a domain-specific reference combined with the analysis of the existing literature, which helped us to gain a more specific vision to identify the influential limitations of LLMs in the automatic generation system.

To mitigate the effect of the limitations identified in the interview, we propose a framework with several designed techniques such as multi-stage sub-tasks division, error detection, retry and error reporting, and information passing techniques, which can effectively mitigate the limitations outlined above as validated by experiments.

3.1 Expert Interviews

Consequently, considering the aforementioned literature, we conducted semi-structured interviews with LLMs and/or ChatGPT experts (3 males, 1 female, Age: M=26/SD=2.16 years, experience: M=3.63/SD=1.38 years) to explore other potential weaknesses of LLMs that could impede the reliability and performance of automatic generation systems.

3.1.1 Interview Process

The interviews were conducted in the experts' native language, with translated citations provided in English. Participants were recruited through the laboratory's research network. For the interview questions, we first asked experts about their overview of LLMs' weaknesses in an automated system. We summarize the major concerns about the use of LLMs through the experts' responses[1] – (1) ability to address complex questions, (2) stability of input and output, (3) long conversation situation, (4) possibility of full automation through self-instruction. This summarizing is also supported by a review of the relevant literature [3, 5, 29, 47].

During the interview, our objective was to gain comprehensive insights into the perspectives of domain experts on the potential challenges associated with the implementation of Language Models (LLMs) in automated systems. To begin each session, we provide an introduction explaining the concept of LLM-based automated systems. The interviews were conducted via Zoom, and prior to the sessions, the participants consented to the recording. These recordings were securely stored and processed in accordance with the General Data Protection Regulation (GDPR). The average duration of the interviews was 9.28 min (SD = 1.9 min).

We employed a thematic analysis methodology, as outlined by Braun and Clarke [6], to examine the participants' responses. Each response was individually labeled with distinct codes, which were subsequently clustered into various themes in section 3.2.2.

3.1.2 Results

From interviews and existing analyses in the literature, we summarize four main aspects that might affect the reliability and correctness of the LLM-based automated system.

Lack of Ability in Dealing with Multi-hop Reasoning In the interview, all experts (N=4) expressed their concern about the ability of LLMs to correctly understand and respond to multi-hop reasoning questions such as *"... It has the ability to deal with simple questions; however, according to my experience, it performs still unsatisfactory toward complex questions that require multi-hop logic analysis..."*. Though LLMs have exceptional capability in general natural language understanding and responding, it is still doubtful whether LLMs are able to generate an in-depth and high-quality response to multi-hop reasoning tasks. Correspondingly, existing work [3] also argues that LLMs can rarely accomplish multi-hop reasoning tasks. This deficit might lead to an illogical and unreasonable arrangement of the generated content, causing unsatisfying outputs.

The Randomized Response from LLMs In the interview, most experts (N = 3) complained that the response of LLMs might not be stable enough in an automated parsing process like *"... If you want to incorporate LLMs as part of a system by formatting its input and output, it is important to be aware that its randomness may cause system crashes..."*. It might affect the reliability of the whole system since the unparsable response might cause a systematic breakdown. Some experts also mentioned that the randomness of LLMs might also cause wrongly generated programming code, leading to an unsatisfactory experience or even system breakdown. The unreliability of LLMs in generating expected code is also reported in [47].

Memory Loss For a system that requires long conversations to address complex tasks. Many experts (N=3) expressed their concerns about the ability of LLMs to remember the information from existing conversations, and utilize them in the current dialog: *"...However, LLMs may suffer from forgetting or conflating previous dialogue history in long-term conversations, leading to issues in understanding and responding to the context. This can result in the model providing incorrect or inconsistent answers, or being unable to accurately address specific questions within the conversation..."*, *"...As the conversation continues, the model may fail to understand the task requirements from earlier ones. This can result in the model being unable to effectively accomplish the task in long conversations..."*, and they think it is a disastrous problem for an automatic generation system since it causes unsatisfying results or system breakdown.

[1]Anonymized transcripts can be accessed in the github page https://github.com/Williamy946/Text2VRScene

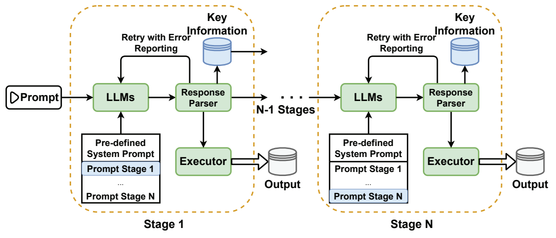


Figure 1: Illustration of Proposed Framework

Hardness for Self-Instruction Since LLMs can automatically parse text and generate responses, it is natural to consider requesting LLMs to analyze a given question, generate the required steps to address the question and execute the steps sequentially. Such a fully automated manner is attractive but faces several challenges resulting from the limited capability of current LLMs. According to the literature [3, 31], sophisticated prompt engineering is encouraged to maximize the ability of LLMs. Similarly, some experts (N=3) proposed that the prompt generated by LLMs might not be well-designed enough to ensure the quality of the LLMs' response generated. *"...Because ChatGPT's responses are sensitive to the prompts it receives, it requires explicit instructions to generate higher-quality responses..."*. Moreover, for an automated system that requires long conversations with LLMs, those experts said that *"...As the conversation progresses, the semantics between questions and answers may change, causing the model to generate responses that are inconsistent with the original task or question requirements...."*,*"...If there are errors, ambiguities, or semantic drift in the formatted output generated by large-scale language models, parsing such output may lead to erroneous or inaccurate results during task execution....."*, it means that once a misunderstanding happened in the middle of the long conversation, the results might largely differ from the expectation.

3.2 Framework Design

In this section, we present several techniques to address the aforementioned problems associated with LLMs, thereby enhancing the reliability of the system. Subsequently, we introduce a framework for developing LLMs-based automated generation systems for complex creative digital content, as illustrated in 1.

Multi-Stage Sub-tasks Division LLMs have shown acceptable performance in simple tasks such as causal and analogical tasks. However, they fail to handle complex tasks, particularly those that require multi-hop reasoning proposed by existing work [3, 34] and expert interviews. To fully exploit the capabilities of LLMs, we propose manually dividing the complex task into several simple tasks will be more suitable for an LLMs-based system. By doing so, we can develop well-designed prompts for each simple task, thereby ensuring the quality of the response from LLMs for each sub-task. Finally, by integrating each sub-task together, the automated system will reach a satisfying result purely generated by LLMs.

The effectiveness of this technique relies on the task division by humans, particularly their domain knowledge regarding how to effectively divide the targeted complex task into several simple tasks.

Retry and Error Report Prompting The formatted response plays a crucial role in automating LLMs as it facilitates automated parsing of the response for subsequent operations. However, due to the inherent randomness of LLMs, even when a formatted response is requested at the prompt, the generated response often fails to be fully formatted, posing a challenge to the reliability of the system. To mitigate this issue, we present a retry and error report prompting technique. Whenever LLMs produce an unparsable response, the system should treat the incorrect response as historical information and provide a reminder prompt such as "Please check the correctness of your answer and respond in the required format."

This technique offers additional guidance to LLMs by treating the unparsable response as a negative instance, aiding them in generating responses in the correct format.

Key Information Message Passing As concluded in the expert interview, the memory loss issue in LLMs can lead to incorrect output, particularly in tasks involving the comprehension of lengthy conversations. To tackle this issue, we propose the implementation of a key information message-passing technique. This technique aims to alleviate the risk of memory loss by distilling crucial information from the previous conversation and integrating it into the subsequent prompt. By doing so, LLMs can maintain a more comprehensive understanding of the ongoing discourse, thus improving the accuracy and reliability of their responses.

Prompt Engineering Based on the findings derived from previous interviews and literature [3, 31], it is unlikely that a fully automated self-instruction-based system is feasible due to the inherent limitations of LLM in understanding ambiguous instructions. Consequently, in order to attain an automated generation process while preserving the reliability and accuracy of the system, this paper proposes to utilize a predefined system prompt at each stage.

Practically, the prompt in each stage will be meticulously crafted to properly leverage the key information passed from the historical stages and achieve the specific requirements of each stage, thereby ensuring that the responses generated by the LLMs meet the requirements for the system's automatic parsing and processing.

4 The VR Experience Generation System

This section presents a detailed design of the Text2VRScene system shown in 2, which is based on the aforementioned framework.

To showcase the advanced performance of LLMs, we employ ChatGPT-3.5 [33] as the backbone of our Large Language Models(LLMs). Regarding the executor component, we utilize the Shap-E model [22] as the 3D object generator and use the Blockad Skybox as the text-based skybox image generator. For the generation of the virtual reality (VR) scene, we leverage A-Frame, a web framework specifically designed for building immersive VR experiences, as the VR experience engine in our system.

For prompt engineering, our designed prompts in each stage are based on the JSON template as follows for the convenience of task determining and key information passing:

```
{
 'Task'': ''{Determine the task and passed
information that should be inferred}'',
 'Key Information1'': ''{Passed Information}'',
  ..
 'Constraints'': ''{Pre-defined Task
Constraints}'',
}
```

In the template, we first define the task and mention the possible useful key information passed from historical conversations. Then list all the key information one by one. Finally, we add the pre-defined task constraints to regulate the response of ChatGPT in this step. Specifically, in the task constraints of each step, we require ChatGPT to respond in JSON format as follows.

```
{
    ''Responses'':
    { 'Response1'': ''{Response Content}'', ...}
}
```

After receiving the responses, we utilize a response parser to automatically extract the information for the next stage. Following the source code of Auto-GPT, a self-managing AI agent, the response parser employs a regular expression to capture the outermost curly braces, ignoring tedious explanations in ChatGPT's response, and then uses the json.loads() function in python to parse the JSON formatted content within. In this process, the parser provides some degree of robustness to the automatic system.

4.1 Overall Workflow

Following the principles of the *Multi-Stage Sub-tasks Division*, we split the VR scene generation task into seven sub-tasks to handle the complexity of the whole task. The structure of prompts for each step is also given below.

Scene Description: In the first task, ChatGPT is required to determine the topic of the scene described by the user based on the user prompt. Then ChatGPT will be required to expand the topic to a detailed description that includes several key elements such as background, main characters, main objects, etc. In this step, the key information is the user prompt.

```
{
 'Task'': ''Generate the main topic of the User
Prompt, and generate the description follows the
constraints.'',
 'User Prompt'': ''{User Prompt}'',
 'Constraints'': ''{Pre-defined constraints}'',
}
```

Skybox Generation: This sub-task involves generating a description of the sky-box for the scene. The generated description will be passed to the skybox generator to generate the background of our VR scene. As a result, it is essential for our prompt design to regulate the response of ChatGPT and ensure that it can be comprehended by the skybox generation model. In this task, the scene description response from the last task is added to the key information set.

```
{
 'Task'': ''Respond the short description of
the skybox based on the Scene Description based
on the content in Literature, following the
Constraints.'',
 'Scene Description'': ''{Previous Responded
Scene Description}'',
 'Literature'': ''{Name of Determined
Literature}'',
 'Constraints'': ''{Pre-defined Constraints}'',
}
```

3D Model Generation: In this subtask, ChatGPT should determine the main characters and objects in the scene according to the Scene Description generated in the first stage, then generate the text description of these 3D models. These descriptive texts will be passed

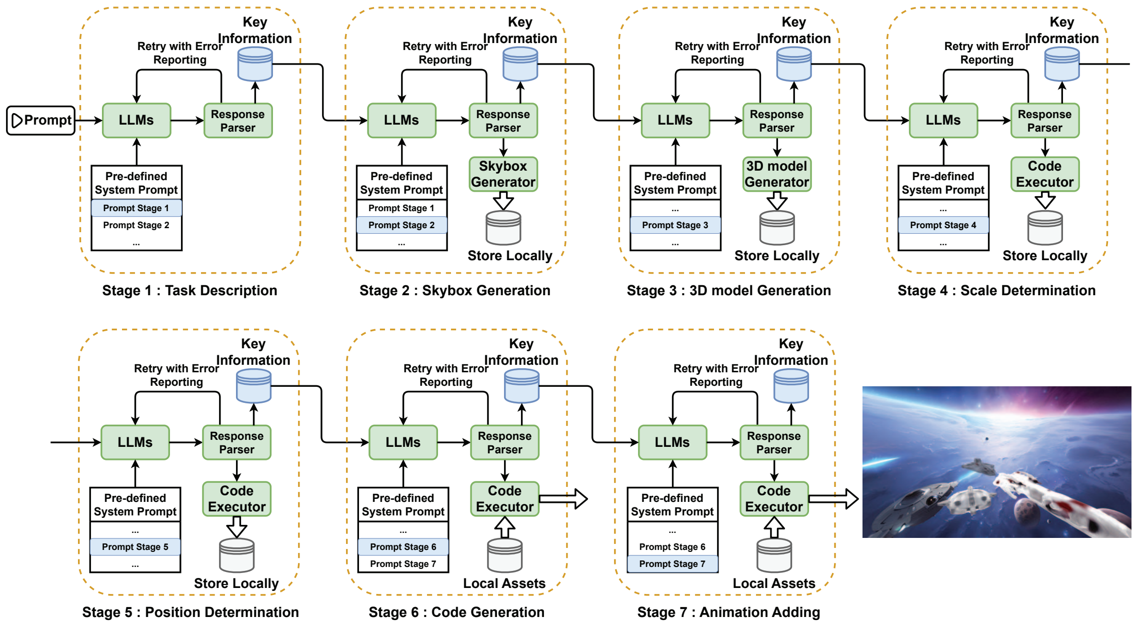

Prompt
LLMs
Response Parser
Key Information
Retry with Error Reporting
Pre-defined System Prompt
Prompt Stage 1
Prompt Stage 2
Prompt Stage 3
Prompt Stage 4
Prompt Stage 5
Prompt Stage 6
Prompt Stage 7
Skybox Generator
3D model Generator
Code Executor
Store Locally
Local Assets
Stage 1 : Task Description
Stage 2 : Skybox Generation
Stage 3 : 3D model Generation
Stage 4 : Scale Determination
Stage 5 : Position Determination
Stage 6 : Code Generation
Stage 7 : Animation Adding

Figure 2: Workflow of Text2VRScene System

to the text-driven 3D model generator, Shap-E, to generate the corresponding 3D models. As a result, the format of the generated descriptive texts is carefully crafted in our constraint to avoid sentimental unalignment between the ChatGPT and the Shap-E model.

```
{
 'Task'': ''Based on the Scene Description and
the description of the skybox. Respond the
mentioned and possible characters and objects
STRICTLY following the Constraints.'',
 'Scene Description'': ''{Previous Responded
Scene Description}'',
 'Skybox Description'': ''{Previous Responded
Skybox Description}'',
 'Literature'': ''{Name of Determined
Literature}'',
 'Constraints'': ''{Pre-defined Constraints}'',
}
```

Scale Determination: With all the digital assets generated, this sub-task requires ChatGPT to determine the scale of the generated 3D models according to the scene descriptions and their physical size. Key information includes the generated models and the scene descriptions. The determined scales will be set as the scale attributes of the corresponding 3D models in the generated codes.

```
{
 'Task'': ''Based on the given scene description,
you should generate the SIZE of all characters
and objects in the given Model List STRICTLY
following the Constraint.'',
 'Scene Description'': ''{Generated Scene
Description}'',
 'Model List'': ''{Generated Models}'',
 'Constraints'': ''{Pre-defined Constraints}'',
}
```

Position Determination: In this sub-task, ChatGPT should determine the position of each 3D model in the scene according to the scene description in the previous response. It is crucial to take the previous scale into consideration during the generation of positions to prevent overlapping between different 3D models.

```
{
 'Task'': ''Considering the scale of all
characters and objects in the given Scales
and considering the relative position of all
characters and objects described in the given
Scene Descriptions, generate the position of all
characters and objects listed in given Models
STRICTLY following the Constraints below.'',
 'Scene Description'': ''{Generated Scene
Description}'',
 'Model'': ''{Generated Models}'',
 'Constraints'': ''{Pre-defined Constraints}'',
}
```

Code Generation: After generating all the digital assets needed and determining the scales and positions of all 3D models, in this task, ChatGPT is required to organize all the digital assets into a scene using programming codes to express the main theme of the VR experience the user desires. To avoid the influence of randomness in generating arbitrary source codes or generating source codes section by section, we regulate the response of ChatGPT by providing a code template in our designed constraints. In this task, the key information is the scene description and the attributes of all the digital assets generated in the previous steps.

Animation Adding: After generating the source code, in this section, ChatGPT is required to polish the source code by adding animation to each 3D model according to the description of the scene.

```
{
 'Task'': ''Based on the given Constraints,
generating the source code USING the 3D objects
in given Models, skybox image file in given
Skybox.  Set the scale and position of the above
models with the given Scale and Position.  Then
output the AFrame code based on the given AFrame
template in the given constraints.'',
 'Models'': ''{Generated Models}'',
 'Skybox'': ''{Generated Skybox Image}'',
 'Scene Description'': ''{Generated Scene
Description}'',
 'Scale'': ''{Generated Model Scale}'',
 'Position'': ''{Generated Model Position}'',
 'Constraints'': ''{Pre-defined Constraints}'',
}
```

```
{
 'Task'': ''Based on the given Constraints,
generating the source code USING the 3D objects
in given Models, skybox image file in given
Skybox.  Set the scale and position of the above
models with the given Scale and Position.  Then
output the AFrame code based on the given AFrame
template in the given constraints.'',
 'Scene Description'': ''{Generated Scene
Description}'',
 'Generated Code'': ''{Generated Code}'',
 'Constraints'': ''{Pre-defined Constraints}'',
}
```

4.2 Framework Implementations

The above Text2VRScene system is developed based on our proposed framework. In this section, we will elaborate on how the framework is implemented in our Text2VRScene system.

It is obvious that the above system follows the principles of the **Multi-Stage Sub-tasks Division** techniques to divide the complex VR experience generation task into several subtasks.

For the **Retry and Error Report Prompting** technique, we have set the maximum number of retries for each sub-task to five. This means that if ChatGPT's response could not be parsed, we would initiate a retry and add an error report prompt to the new prompt, tailored to the specific error type encountered. Furthermore, we have implemented a response parser that automatically identifies the outermost curly brackets in the response, which further improves the reliability of the entire system.

Regarding the **Key Information Message Passing** technique, we can see from the prompt design above that we store the response of the historical subtask as the key information. Then we pass the needed key information to the prompt of the following subtasks as the implementation of key information message passing.

In terms of **Prompt Engineering**, the formatted prompt design and the pre-defined constraints of each task are the main parts of our prompt engineering. Moreover, in the text of constraints, we utilized the few-shot prompting technique to enhance the reliability of ChatGPT's response.

5 Case Study

Based on the above framework, we developed the Text2VRScene system[2], which is able to automatically generate the VR experience with a clear theme based on a simple text. In this section, we will evaluate the effectiveness of the proposed framework in increasing the reliability and performance of the system with several case studies. The developing environment is Python 3.10 on Ubuntu Server. The 2D Skybox Image generation task is conducted by calling the commercial text-based skybox generation API of Skybox AI in BlockAde Labs. The 3D model generation method is Shap-E [22] implemented on an Ubuntu server with 1xNVIDIA GeForce RTX 3090 Ti.

[2]The source code of this system is available at https://github.com/Williamy946/Text2VRScene

5.1 Evaluation Metrics

To evaluate the reliability of the system and the quality of the generated VR scenes, we employ an Error Rate (ER) metric as [41].

Considering the inherent randomness of LLMs, we believe that the reliability of an LLMs-based generation system is crucial, and the possibility of generating eccentric experiences or encountering system breakdowns is a necessary metric for evaluating the reliability and robustness of the system.

Hence, we define the Error Rate (ER) in our system as the probability of failure, such as generating target-unrelated experiences or experiencing system breakdowns. Furthermore, to accurately assess the reliability of the system, we exclude failures caused by external factors such as network errors or overload errors originating from OpenAI servers or generator servers.

5.2 Comparison Systems

To validate the effectiveness of our proposed framework, we introduce several comparison systems by discarding one of the techniques from our framework for each comparison system. Moreover, we also introduce the Web-based ChatGPT as the baseline method.

- **Text2VRScene-SinglePrompt**: This variant discards the multi-stage technique and use a single prompt to ask LLMs to generate the description, sky-box, 3D models, and source code.
- **Text2VRScene-NoRetry**: This variant discards the retry technique and terminates the system if the response cannot be parsed.
- **Text2VRScene-NoPromptEngineering**: In this variant, we deprecate the pre-defined constraints, which are primarily used for regulating the behavior of ChatGPT. ChatGPT in this variant is required to generate responses based solely on the stage's requirements and the need for a formatted response, without regulating their behavior through well-designed prompts.
- **ChatGPT-Web**: In this variant, we directly use the ChatGPT website to generate the source code of a web page based on the given prompt. We employ this variant as a baseline to illustrate the capabilities of LLMs in generating VR experiences.

Moreover, except for these variant systems, the system that deprecates the key-information messaging technique is not included in the comparison variants. The main reason is that after the deprecating of the key information messaging technique, the system is unable to generate executable source code that organizes the generated digital assets based on the determined scene due to the memory loss problem. As a result, the key-information messaging technique is essential for the feasibility of our system.

5.3 Reliability Study

For the ChatGPT-based virtual reality scene generation system, ensuring system reliability is of paramount importance due to the inherent randomness of ChatGPT, which may lead to unexpected system breakdowns. Therefore, we begin by conducting a reliability study to compare the performance of the Text2VRScene system with other variants. In this case study, we select the **Text2VRScene-SinglePrompt**, **Text2VRScene-NoRetry**, and **Text2VRScene-NoPromptEngineering** variants as the comparison methods for Text2VRScene. The reason why **ChatGPT-Web** is not selected is that we manually parse the HTML code for this variant, so it is meaningless to compare the reliability between an automated system and a human-labor-based system.

We utilize the Error Rate (*ER*) as the performance metric. Additionally, we perform in-depth analysis to identify the causes of system breakdowns or unsatisfactory experiences, categorizing errors into three types. First, the *Unparsable Error* occurs when the system fails to parse the response and cannot continue, typically caused by the randomness of LLMs. Second, the *Code Bug Error* occurs when the generated code cannot be executed due to bugs. Finally, the *Eccentric Topic Error* occurs when the generated scene is unrelated to the topic requested by the user. The *Eccentric Topic Error* can be defined as the user cannot recognize the generated VR scene after the user inputs the text.

5.3.1 Study Settings

This study consists of two types: stability and robustness. For the stability assessment, we repeat the following four prompts 50 times and calculate the Error Rate (*ER*) for each system.

- **Prompt 1**: “Generate a VR scene about the movie Star Wars.”
- **Prompt 2**: “Generate a VR scene about the song My Heart Will Go On.”
- **Prompt 3**: “Generate a VR scene about the novel Lord of the Rings.”
- **Prompt 4**: “Generate a VR scene about the movie Porco Rosso.”

For robustness evaluation, we randomly request the system to generate scenes for 50 different literature or songs. We then calculate the overall Error Rate (*ER*) value for each system.

5.4 Results

Table 1 clearly demonstrates that all variant systems exhibit higher error rates compared to Text2VRScene, highlighting the effectiveness of the Retry technique, Message Passing technique, and Prompt Engineering in enhancing the system’s reliability and robustness.

Methods	Pro. 1	Pro. 2	Pro. 3	Pro. 4	Ave.	Robust
Text2VRScene-SinglePrompt	18%	26%	24%	18%	21.5%	24%
Text2VRScene-NoPE	34%	36%	24%	28%	30.5%	32%
Text2VRScene-NoRetry	10%	32%	14%	20%	19%	24%
Text2VRScene	2%	8%	8%	4%	5.5%	10%

Table 1: Reliability Error Rate (ER) Comparison

Notably, the variant without prompt engineering exhibits the lowest reliability and robustness among all systems, emphasizing the necessity of prompt engineering. Additionally, Table 1 also presents varying distributions of error types among the different variant systems, reflecting the distinctive characteristics of these variants and the advantages provided by the discarded techniques.

To further investigate the effect of different techniques, we analyze the distribution of error types of the four systems in Fig. 3.

Among the three types of errors, both *Unparsable Error* and *Code Bug Error* can lead to system breakdown, reducing the overall reliability and robustness of the system, while *Eccentric error* results in unsatisfactory user experiences.

In the variant system **Text2VRScene-NoPE** where pre-defined constraints that regulate the behavior of LLMs are discarded, we observe a significant decline in reliability and correctness, as indicated by the extremely high error rate in Table 1. Furthermore, Fig. 3 illustrates that the major types of errors in this variant are *Unparsable Error* and *Code Bug Error*, which separately account for nearly 40% of the errors.

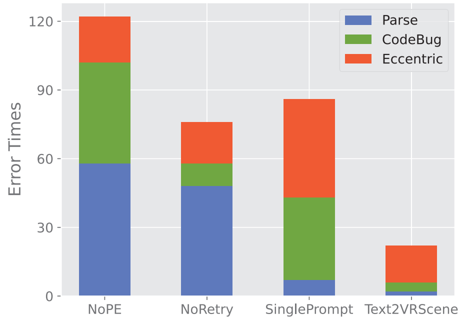


Figure 3: Frequency and Distribution of Different Error Types

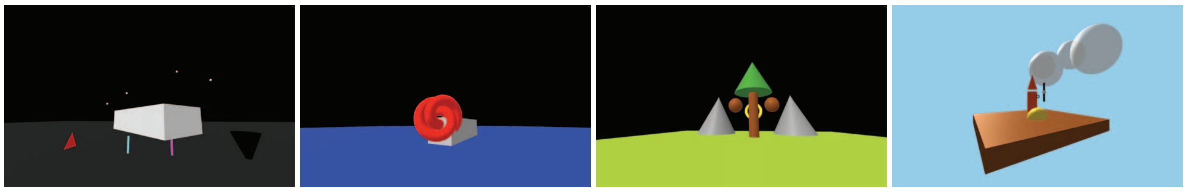
The occurrence of *Unparsable Error* can be attributed to the lack of emphasis on the required response format due to the absence of prompt engineering. Although the format is included in the prompt, without prompt engineering, the LLMs may fail to strictly adhere to the specified format, leading to *Unparsable Error*. On the other hand, *Code Bug Error* is influenced by the randomness of LLMs. By incorporating a basic template of source code in the pre-defined constraints, prompt engineering helps ensure that the generated source code is executable. Without prompt engineering, the generated source code may include incorrect packages or deprecated functions, resulting in a system breakdown.

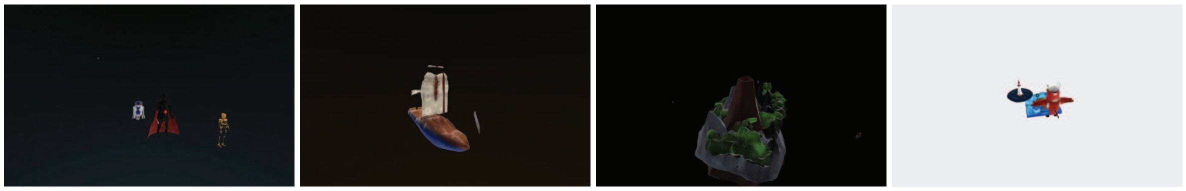
In the variant system **Text2VRScene-SinglePrompt**, we observe that it has the most similar performance in the parable error rate compared to the Text2VRScene system. It can be linked to the elimination of the randomness of ChatGPT by the one-turn response in this variant. Fig. 3 indicates that the major types of errors in this variant are *Eccentric Error* and *Code Bug Error*. Upon investigating the causes of these two errors, we find that ChatGPT might have limited attention to the given long prompt. It means that ChatGPT might ignore some requirements in the prompt and focus on satisfying the demand in the other part of the prompt. As a result, if the code requirements are not the focus of ChatGPT, ChatGPT might generate codes that cannot be executed. While once the attention of ChatGPT is not about the position and scale of models, then the object might overlap, causing unsatisfying experience.

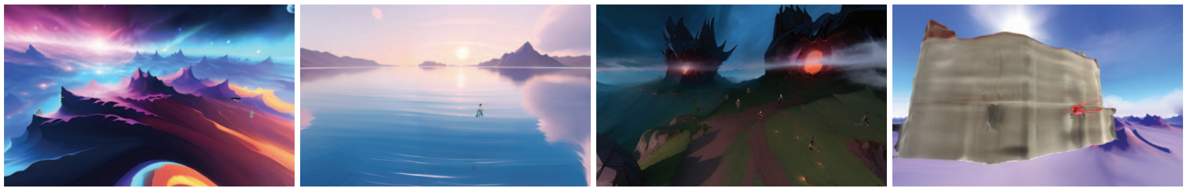
In the variant system **Text2VR-NoRetry**, Fig. 3 demonstrates that the primary error type is *Unparsable Error*. Surprisingly, though with the repeated emphasis on response format in the pre-defined constraints, format issues still persist as the main cause of errors in this system. In comparison to the complete system **Text2VRScene**, where most format errors can be addressed by retries with error report prompts such as ”You should check the correctness of the JSON format of your response again,” this finding highlights the inherent randomness of LLMs and emphasizes the necessity of the *Retry with Error Report Prompting* in enhancing the reliability and robustness of LLMs-based generation systems.

Indeed, the above analysis reports that the system **Text2VRScene** has the lowest error rate. Moreover, it is worth noting that the major error type in this system is the *Eccentric Error*, which primarily affects the user experience rather than causing system breakdown. This finding validates the effectiveness of our proposed framework in enhancing the reliability and robustness of LLMs-based systems.

In conclusion, based on the above analysis, we can affirm that our proposed framework has the capability to significantly increase the reliability and robustness of LLMs-based automated systems.


(a) ChatGPT-Web


(b) Text2VRScene-no Prompt Engineering


(c) Text2VRScene-Single

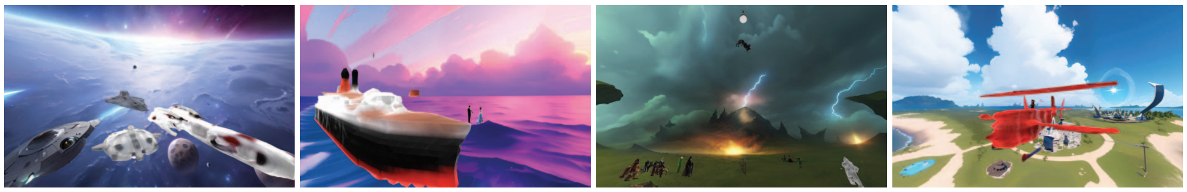

(d) Text2VRScene

Figure 4: Performance Illustration. For each system, we generate four scenes as illustrations: Star Wars, My Heart Will Go On, Lord of the Rings, and Porco Rosso. The above illustrations show that the Text2VRScene system generates the scenes that best meet the requirements of the user.

5.5 VR Scene Illustration

Fig. 5 presents the screenshots of the generated VR scenes, providing a visual representation of the performance of our Text2VRScene system compared to the selected variants. The variants **Text2VRScene-SinglePrompt**, **Text2VRScene-NoPromptEngineering**, and **Text2VRScene-Manual** were chosen as the comparison methods to showcase the performance differences. And **Text2VRScene-NoRetry** is not selected as the comparison method because once a parse error occurs in this system, it will break down, while if no parse error occurs, the output of this variant system will be the same as the complete **Text2VRScene** system. So we believe that it is meaningless to compare the illustration performance of **Text2VRScene-NoRetry** with **Text2VRScene**.

For each screenshot shown in Fig. 5, we have executed the VR scene generation system ten times and selected the most representative images of the system for the following analysis.

The results in Fig. 5 demonstrate that our Text2VRScene system is capable of capturing the essence of the desired scene. The generated sky-box and 3D models align with the theme specified in the prompt, and the scale and position of the models are accurately set. Although not visible in the figure, the animations of the 3D models also express the topic of the prompt through the dynamics of 3D objects.

In Fig. 5, we first illustrate the performance of the variant **ChatGPT-Web**. In this study, we appended the phrase ”using primitives” to the prompt. Otherwise, the generated source code would contain references to skybox images and 3D models that do not exist in the local database. We can observe that ChatGPT possesses the ability to comprehend the relative positions and scales of 3D objects, enabling them to combine various objects to convey a theme. To further enhance the expressiveness of the ChatGPT, it is natural to consider the collaboration between text-based digital asset generation models and the ChatGPT, as is evident in other variants.

In the **Text2VRScene-NoPromptEngineering** variant, where only the response format and stage tasks are retained, it is evident that the generated VR scenes lack environmental information, as the skybox appears as a plain color. This is primarily due to the absence of any instructions in the prompt regarding the use of skybox images. Consequently, the generated source code utilizes pure colors as the background instead. Furthermore, the positioning of objects in these scenes is often illogical, failing to convey the intended theme.

In the **Text2VRScene-SinglePrompt** variant, we have integrated the prompts from each stage into a single prompt file, which is then provided to the system. However, upon examining the resulting VR scenes, it becomes apparent that ChatGPT struggles to comprehend and generate responses and fully satisfy the complex requirements encompassed within a single response. These requirements include scene description, skybox depiction, identification of potential 3D models, and the generation of source code to arrange and activate these models in a cohesive manner. Consequently, the generated 3D model scenes suffer from issues such as overlapping and undersized objects, which hinder the expression of the user prompt.

Both of the aforementioned variants overlook one or some of the crucial techniques in our proposed framework, leading to a noticeable decline in performance in terms of understanding the logical structure of the described scene and accurately translating that logic into source code. This highlights the effectiveness and necessity of the multi-stage technique and prompt engineering in our proposed framework for developing a reliable and expressive generation system based on ChatGPT.

In the comprehensive system, **Text2VRScene**, we integrate all the techniques from our proposed framework and showcase the generated results in Fig. 4(d). It is evident that the **Text2VRScene** system successfully associates literature or song names with representative scenes, followed by generating detailed descriptions of these scenes. Furthermore, the system demonstrates the ability to determine the external environment of the scene and identify the presence of mentioned objects. Importantly, by referring to the descriptions and real-world knowledge, the system exhibits an understanding of the scale, relative positions, and dynamics of the mentioned objects. Subsequently, the system accurately translates this understanding of the scene into the correct source code for execution.

6 Discussion

The rise of VR as an art form has brought about a revolutionary new way of immersing ourselves in the realm of creativity and expression. It has given birth to a synthesis of technology and fine art, providing artists with a new, boundless canvas where they can create multi-sensory, interactive, and immersive experiences. This reimagining of artistic expression is opening new frontiers in how we create and experience art, deepening our connection with the world of imagination and abstraction. The prevailing commodity head-mounted displays (HMDs) have democratized the VR experience, leading to a surge in demand for immersive and high-quality VR content. However, this demand has not been fully satisfied due to limited production capabilities.

To explore the automated generation of VR experience, several recent studies have focused on generating or reconstructing 3D scenes [10, 14, 28, 44, 52], but few of these studies have been able to automate the massive production of satisfying VR experiences. On the one hand, certain methods [10, 28, 52] focus mainly on the generation of 3D models of indoor scenes, thereby limiting the expression of creativity through dynamic interactions between objects. On the other hand, other methods [14, 44] are restricted by the available databases and the capabilities of the underlying models, which hinders the generation of diverse and fantastic scenes customized for users. These limitations significantly affect the user experience due to a lack of freshness, which causes much unsatisfactory.

Therefore, generating creative digital content is a challenging task that requires consideration from multi-aspects. However, the recent development of generative models [22, 33, 38] opens new pathways for the automated generation of creative VR experiences and even other forms of creative digital content.

To explore the methodologies of developing an LLMs-based VR experience generator, our study first investigates the possible restrictions of existing LLMs to automatically solve a task. Then, we propose a pioneering framework to organically mitigate the impact of these restrictions and enhance the reliability of the automated generation system. To the best of our knowledge, we are the first to explore the general framework of LLMs-based creative digital content generation systems.

Finally, we developed Text2VRScene, according to our framework, which addresses the problems of existing VR scene generators. The Text2VRScene system is able to generate VR scenes without clear boundaries from a simple text and express creativity through the dynamics of different 3D objects. The user can generate any experience he/she wishes through a short description of the scene or even the name of the literature from which the scene originates.

Limitation: However, our study design has certain limitations. The first is the methodology employed to identify the weaknesses of LLMs in automated systems. We acknowledge that relying solely on the four short expert interviews in a qualitative analysis may inadvertently overlook important features and limitations of LLMs in automated systems. To address this concern and ensure a comprehensive evaluation, we combined the analysis of existing literature with our qualitative study to cover as many main limitations as possible. This approach aimed to compensate for any potential limitations of the expert opinions. We would like to argue that with the above process, the main obstacles that affect the feasibility of achieving an LLMs-based automated system should be identified.

Another limitation is the quality of our generated VR scene with respect to the lack of interactivity and the imbalanced quality between the 2D background image and 3D models. The lack of interactivity will reduce users' sense of involvement in the dynamic of the scene, the main reason is that AFrame, the engine employed for rendering VR scenes,does not natively support interaction with objects, which would be addressed by the development of the AFrame or using another engine. The imbalanced quality of 2D background and 3D models will reduce users' sense of immersive in the scene, the main reason is the imperfect performance of the existing text-based 3D model generation algorithms. Such limitations will be addressed with the advancement of text-based 3D model generation algorithms. Despite the limitation in scene quality, we would like to argue that the main target of our work remains achieved. The main contribution of this paper is to explore the possible framework of the LLMs-based automated generation system, whose effectiveness and reliability have been validated by the proposed Text2VRScene.

Our research will persistently leverage the ongoing advances in LLMs and digital content generation models to advance the LLMs-based generation system. Adhering to the proposed framework, we intend to delve into the creation of generation systems for various forms of digital content, such as animation, manga, and potentially even movies. Moreover, taking into account the prospective progress and enhanced capabilities of increasingly powerful LLMs, we will persistently investigate the feasibility of a universal framework to facilitate the development of LLMs-based automated systems.

7 Conclusion

In this work, we identify the critical limitations of LLMs in an automated system based on expert interviews and literature reviews. Then we proposed a general framework to provide a systematic approach to applying LLMs to automated systems. According to the proposed framework, we develop the Text2VRScene system, which generates expressive VR scenes based solely on text. Furthermore, we conduct a series of case studies to validate that the proposed framework provides a systematic guarantee of reliability. Case study results also illustrate the promising performance of the Text2VRScene system in generating expressive VR scenes regulated by meticulously designed prompts.

In general, the work presented in this paper proposes a system that can generate complex VR scenes based solely on natural language. This paper explores the general framework for developing LLMs-based automated systems, providing valuable guidance for researchers interested in developing generative systems toward other kinds of complex and creative digital content.

8 Acknowledgement

The work is supported by the Bureau of Science and Technology of Nansha District, Guangzhou, Grant No.2022ZD012.

References

[1] T. Ahmed and P. Devanbu. Few-shot training llms for project-specific code-summarization. *arXiv preprint arXiv:2207.04237*, 2022.

[2] Autodesk. 3DMAX Software. `https://www.autodesk.com/products/3ds-max/`, 2023.

[3] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung, et al. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*, 2023.

[4] Blender. Blender Software. `https://www.blender.org//`, 2023.

[5] A. Borji. A categorical archive of chatgpt failures. *arXiv preprint arXiv:2302.03494*, 2023.

[6] V. Braun and V. Clarke. *Thematic analysis.* American Psychological Association, 2012.

[7] E. Cambria and B. White. Jumping nlp curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, 9(2):48–57, 2014.

[8] D. Castelvecchi. Are chatgpt and alphacode going to replace programmers? *Nature*, 2022.

[9] S. Chakraborty, T. Ahmed, Y. Ding, P. T. Devanbu, and B. Ray. Natgen: generative pre-training by "naturalizing" source code. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 18–30, 2022.

[10] A. X. Chang, M. Eric, M. Savva, and C. D. Manning. Sceneseer: 3d scene design with natural language. *arXiv preprint arXiv:1703.00050*, 2017.

[11] R. Chen, Y. Chen, N. Jiao, and K. Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. *arXiv preprint arXiv:2303.13873*, 2023.

[12] Z. Chen, G. Wang, and Z. Liu. Text2light: Zero-shot text-driven hdr panorama generation. *ACM Transactions on Graphics (TOG)*, 41(6):1–16, 2022.

[13] J.-H. Cheng, Y. Chen, T.-Y. Chang, H.-E. Lin, P.-Y. C. Wang, and L.-P. Cheng. Impossible staircase: Vertically real walking in an infinite virtual tower. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pp. 50–56. IEEE Computer Society, 2021.

[14] L.-P. Cheng, E. Ofek, C. Holz, and A. D. Wilson. Vroamer: generating on-the-fly vr experiences while walking inside large, unknown real-world building environments. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 359–366. IEEE, 2019.

[15] A. M. Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. Jiang. Github copilot ai pair programmer: Asset or liability? *Journal of Systems and Software*, p. 111734, 2023.

[16] M. Denninger and R. Triebel. 3d scene reconstruction from a single viewport. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII*, pp. 51–67. Springer, 2020.

[17] D. Fried, A. Aghajanyan, J. Lin, S. Wang, E. Wallace, F. Shi, R. Zhong, W.-t. Yih, L. Zettlemoyer, and M. Lewis. Incoder: A generative model for code infilling and synthesis. *arXiv preprint arXiv:2204.05999*, 2022.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, oct 2020. doi: 10.1145/3422622

[19] P. Hämäläinen, M. Tavast, and A. Kunnari. Evaluating large language models in generating synthetic hci research data: a case study. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–19, 2023.

[20] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[21] N. Jain, S. Vaidyanath, A. Iyer, N. Natarajan, S. Parthasarathy, S. Rajamani, and R. Sharma. Jigsaw: Large language models meet program synthesis. In *Proceedings of the 44th International Conference on Software Engineering*, pp. 1219–1231, 2022.

[22] H. Jun and A. Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023.

[23] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

[24] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8110–8119, 2020.

[25] Y. Li, Z. Gan, Y. Shen, J. Liu, Y. Cheng, Y. Wu, L. Carin, D. Carlson, and J. Gao. Storygan: A sequential conditional gan for story visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6329–6338, 2019.

[26] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 300–309, 2023.

[27] H. Liu, R. Ning, Z. Teng, J. Liu, Q. Zhou, and Y. Zhang. Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*, 2023.

[28] R. Ma, A. G. Patil, M. Fisher, M. Li, S. Pirk, B.-S. Hua, S.-K. Yeung, X. Tong, L. Guibas, and H. Zhang. Language-driven synthesis of 3d scenes from scene databases. *ACM Transactions on Graphics (TOG)*, 37(6):1–16, 2018.

[29] K. Mahowald, A. A. Ivanova, I. A. Blank, N. Kanwisher, J. B. Tenenbaum, and E. Fedorenko. Dissociating language and thought in large language models: a cognitive perspective. *arXiv preprint arXiv:2301.06627*, 2023.

[30] T. Mullen. *Mastering blender*. John Wiley & Sons, 2011.

[31] NeuralMagic. The ChatGPT Cheat Sheet. `https://www.kdnuggets.com/publications/sheets/ChatGPT_Cheatsheet_Costa.pdf/`, 2023.

[32] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong. A conversational paradigm for program synthesis. *arXiv e-prints*, pp. arXiv–2203, 2022.

[33] OpenAI. Chatgpt: Optimizing language models for dialogue. *OpenAI Blog*, 2022.

[34] S. Ott, K. Hebenstreit, V. Liévin, C. E. Hother, M. Moradi, M. Mayrhauser, R. Praas, O. Winther, and M. Samwald. Thoughtsource: A central hub for large language model reasoning data. *arXiv preprint arXiv:2301.11596*, 2023.

[35] X. Pan, A. Tewari, T. Leimkühler, L. Liu, A. Meka, and C. Theobalt. Drag your gan: Interactive point-based manipulation on the generative image manifold. *arXiv preprint arXiv:2305.10973*, 2023.

[36] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[37] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[38] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.

[39] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278, 2000.

[40] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[41] R. Shi, N. Zhu, H.-N. Liang, and S. Zhao. Exploring head-based mode-switching in virtual reality. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 118–127. IEEE, 2021.

[42] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

[43] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[44] M. Sra, S. Garrido-Jurado, C. Schmandt, and P. Maes. Procedurally

generated virtual reality from 3d reconstructed physical space. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*, pp. 191–200, 2016.
[45] U. Technologies. Unity Software. `https://unity.com/`, 2023.
[46] R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.
[47] H. Tian, W. Lu, T. O. Li, X. Tang, S.-C. Cheung, J. Klein, and T. F. Bissyandé. Is chatgpt the ultimate programming assistant–how far is it? *arXiv preprint arXiv:2304.11938*, 2023.
[48] G. Wang, Y. Yang, C. C. Loy, and Z. Liu. Stylelight: Hdr panorama generation for lighting estimation and editing. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XV*, pp. 477–492. Springer, 2022.
[49] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation, 2023.
[50] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.
[51] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*, 2023.
[52] H. Yi, C.-H. P. Huang, S. Tripathi, L. Hering, J. Thies, and M. J. Black. Mime: Human-aware 3d scene generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12965–12976, 2023.

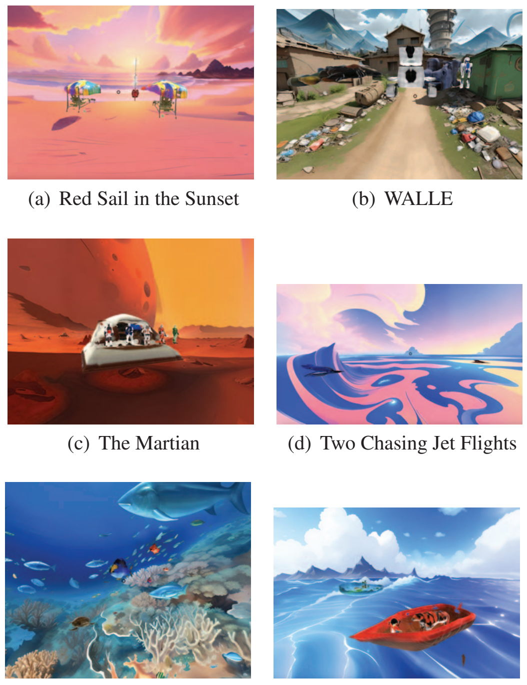

(a) Red Sail in the Sunset (b) WALLE (c) The Martian (d) Two Chasing Jet Flights (e) Finding Nemo (f) Life of Pi

Figure 5: Extra Examples of Generated VR Scene from simple sentences.

A EXTRA EXAMPLES

Due to the insufficient space for comparing more generation cases with different methods, we add some extra cases generated by the full-version Text2VRScene system to illustrate the generality of the system. Following the prompt settings of Section 5.3.1 in the main article, we only changed the content of the requirements, including literature, movie, and direct scene descriptions. The used prompts are listed as follows:

- **Prompt 1**: "Generate a VR scene about the song Red sail in the Sunset."
- **Prompt 2**: "Generate a VR scene about the movie WALLE."
- **Prompt 3**: "Generate a VR scene about the movie The Martian."
- **Prompt 4**: "Generate a VR scene about the scene two chasing jet flights."
- **Prompt 5**: "Generate a VR scene about the movie Finding Nemo."
- **Prompt 6**: "Generate a VR scene about the movie Life of Pi."

B FINE-TUNED PROMPTS

In this section, we will elaborate on the patterns of prompt engineering fine-tuning in this method. Due to the limited pages, we will select the fine-tuned prompt for the task determination component as an example for analysis. Other prompts are similar in structure and available at https://github.com/Williamy946/Text2VRScene.

As illustrated in the box, the pre-defined task constraints consists of two parts, the Constraints part and the Response Format part. In the constraint part, the detailed rules of the given task and the requirements about the response will be provided. Moreover, to fine-tune the performance of LLMs, some empirical methods will be adopted to raise the extra attention of LLMs to the specific requirement such as using upper case alphabets, some hinting words like ”ATTENTION!”, ”WARNING!”, etc.

In the Response Format part, the detailed response format of each stage will be given for LLMs' in-context learning. Moreover, for the compatibility with the limited text-understanding ability of existing text-based generation methods, a few examples about the descriptions about the 2D skybox and 3D models will be given as few-shot information for LLMs.

```
CONSTRAINTS:
1.  No user assistance,
2.  Determine the literature or name of songs in the User prompt in "topic", if
no literature or song is mentioned, generate the main theme of the User prompt
in "topic".,
3.  YOU SHOULD Generate the description of a scene about the most expressive
event related to the topic.,
4.  In the description, you should include the introduction of background, the
description of objects and characters, the description of the start position
and size of these objects and characters, and descriptions about their physical
movements in the event.,
5.  ATTENTION!:  YOU SHOULD SPECIFY THE DETAILS OF COLORS, AND APPEARANCE ABOUT
EVERY CHARACTERS AND OBJECTS IN THE DESCRIPTION.,
6.  You should only respond in JSON format as described below and do not add
extra explanations excluding the JSON response,
7.  Ensure the response can be parsed by Python json.loads,
Response Format:
{
    "responses":
    {
        "Topic":  "The topic in the user prompt",
        "Description":
        {
        "Background":  "description of the background of the scene, eg.  <Ocean
with hues of orange and pink at sunset>.",
        "Characters":  "full-body description of the characters in the scene,
eg.  <Rose, a young woman with flowing auburn hair, wears an elegant blue
gown.>, <Jack, with disheveled brown hair, is dressed in a simple white shirt
and trousers.>",
        "Objects":  "description of the objects itself in the scene without
mentioning other objects, eg.  <Titanic, a majestic ship with red strips.>, <an
iceberg, with the color of dark blue.> ",
        "Scale":  "description of relative scales of ALL characters and objects
in the scene, eg.  <the Titanic is in very large scale compared with human>,
<the iceberg is two times large than the Titanic>",
        "Start position":  "description of start positions of ALL characters
and objects in the scene, eg.  <Jack is standing behind Rose at the bow of the
Titanic ship>, <the Titanic is start sailing from the left side>",
        "Movements":  "description of the movements of ALL characters and
objects in the event.  You must include the start position and end position of
characters and objects.  eg.  <the Titanic is translating from left to right,
heading toward the iceberg>, <Jack and Rose are on the Titanic, moving together
from left to right>, <The iceberg is floating up and down and rotating>"
        },
    }
}
```