Proceedings of the Nineteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2023)

SceneCraft: Automating Interactive Narrative Scene Generation in Digital Games with Large Language Models

Vikram Kumaran, Jonathan Rowe, Bradford Mott, James Lester
North Carolina State University
{vkumara, jprowe, bwmott, lester}@ncsu.edu

Abstract

Creating engaging interactive story-based experiences dynamically responding to individual player choices poses significant challenges for narrative-centered games. Recent advances in pre-trained large language models (LLMs) have the potential to revolutionize procedural content generation for narrative-centered games. Historically, interactive narrative generation has specified pivotal events in the storyline, often utilizing planning-based approaches toward achieving narrative coherence and maintaining the story arc. However, manual authorship is typically used to create detail and variety in non-player character (NPC) interaction to specify and instantiate plot events. This paper proposes SceneCraft, a narrative scene generation framework that automates NPC interaction crucial to unfolding plot events. SceneCraft interprets natural language instructions about scene objectives, NPC traits, location, and narrative variations. It then employs large language models to generate game scenes aligned with authorial intent. It generates branching conversation paths that adapt to player choices while adhering to the author's interaction goals. LLMs generate interaction scripts, semantically extract character emotions and gestures to align with the script, and convert dialogues into a game scripting language. The generated script can then be played utilizing an existing narrative-centered game framework. Through empirical evaluation using automated and human assessments, we demonstrate SceneCraft's effectiveness in creating narrative experiences based on creativity, adaptability, and alignment with intended author instructions.

Introduction

The challenge of generating compelling, and adaptable narratives has long been a crucial component of interactive narrative design (Riedl and Bulitko 2013). While early research in narrative generation primarily focused on creating coherent sequences of events, this approach often fails to produce captivating stories (Martin et al. 2018; Kreminski, Wardrip-Fruin, and Mateas 2020; Ramirez and Bulitko 2014). In educational games, designers strive to support learning by highlighting relevant facts during gameplay, and making learning an intrinsic part of the story (Naul and Liu 2020). Additionally, as players become more familiar with a game, gameplay can become repetitive. Dynamic creation of stories around specific events becomes important for maintaining player interest. Given these challenges, we propose SceneCraft, a framework that leverages large language models (LLMs) to transform high-level author descriptions into playable episodes within a 3D virtual world, complete with engaging non-player character dialogue, gesture, and emotion. This approach simplifies how we create interactive scenes featuring non-player characters (NPCs), thereby streamlining scene authoring. SceneCraft is driven by authorial intent (Riedl and Bulitko 2013) as a story summary that is transformed into a choose-your-own-adventure style interaction that can play out in a 3D game environment. We use LLMs to generate narrative scenes for the outline provided by the author and semantically extract from the generated narrative scene emotions and gestures to display programmatically during NPC interactions.

Research in natural language processing (NLP) focusing on story generation has surged in recent years, addressing various aspects, including content control, commonsense knowledge use, understanding character actions, and creativity (Alabdulkarim, Li, and Peng 2021). Recent advances in LLMs using transformer-based models have outperformed many earlier models in generating short stories and dialogues based on human-provided narrative outlines (OpenAI 2023; Chowdhery et al. 2022; Bubeck et al. 2023). In addition, LLMs have demonstrated remarkable proficiency in tasks such as extracting semantic information, generating code based on abstract guidance, and identifying semantic features like emotions when given directive prompts, all without requiring specialized training (Liu et al. 2023). In this paper, we leverage these specific capabilities of LLMs to transform high-level outlines and instructions from an author into engaging and dynamic scripts for NPC interactions, enriching the immersive experiences within digital games.

In a 3D narrative-centered game, the story progresses as the player interacts with NPCs to advance the story arc. Each interaction must be engaging and convey the necessary information to the player to move the narrative forward. We build an end-to-end framework that enables authors to specify the topic of conversation, key issues to be addressed, NPCs' background, NPCs' appearance, scene location, and potential story branching variations. From this input, our framework generates the story and the corresponding dialogues that align with the author's vision using LLMs. Then, we extract emotions and gestures for NPCs corresponding to each exchange, translating them into a comprehensive custom game script encompassing dialogue, emotions, gestures, and story branches. For our narrative representation, we use an Ink-like scripting language. The Ink (https://www.inklestudios.com/ink) dialogue script is read by the StoryLoom engine (Mott et al. 2019), which can read the Ink script and render it as a playable Unity game episode. This script leverages existing game assets to produce an interactive game episode that can be played in a 3D virtual environment. The key contributions of SCENECRAFT narrative generation framework are the following:

- Simplifies transforming author natural language instructions into playable 3D virtual game episodes without manual intervention.
- Provides authors the ability to control the generated content by allowing authors to specify various aspects such as scene objectives (i.e., topics and context), NPC background and appearance, scene location, and narrative variations.
- Extracts emotions and gestures from generated story acts to improve the realism of NPC interaction, enhancing player engagement.

We demonstrate the effectiveness and strengths of SCENECRAFT through automated assessments and human participant evaluations.

Related Work

Story-driven games have long been a subject of interest in the interactive narrative community. A substantial body of research is dedicated to creating captivating storylines and immersive experiences for players in interactive narrative-centered games with varying degrees of success (Riedl and Bulitko 2013; Riedl and Young 2010; Kreminski et al. 2020; Stefnisson and Thue 2018). Interactive narrative-centered games are virtual environments that aim to make the player an integral part of an immersive story, where they have bounded control over how the narrative proceeds.

One way to balance player agency and narrative control is to have an experience manager (Riedl and Bulitko 2013) that revises the narrative as events unfold to maintain the storyline. Researchers have conducted multiple studies to explore how experience managers can accommodate player actions while ensuring the achievement of authorial goals by framing it as an automated planning problem (Ramirez and Bulitko 2014; Riedl and Young 2010). By representing interactive narratives as story graphs (Riedl and Young 2006), with nodes representing world states and edges representing causal transitions, experience managers can actively track the narrative's progress in the graph toward the authorial goals. Narrative control is sometimes accomplished by pruning (Ware et al. 2022) and other similar operations on the graph. We use dialogue graphs in our system to track branched interaction. In our generated script, each node of the graph represents a dialogue utterance between the player and an NPC and edges link the nodes to the corresponding response utterance by another character in the scene.

Instead of top-down control of the narrative, researchers have also designed games where the virtual agents co-create by suggesting or executing actions in the environment. Interactions between players and AI agents or NPCs move the plot forward. One approach involves sifting the generated storyline to identify compelling event patterns (Kreminski et al. 2020). Meanwhile, another strategy employs the player modifying NPC goals as a part of gameplay (Oliver and Mateas 2021). The use of AI as a co-author has typically been in the mode of making suggestions to assist authors by presenting possible actions or new goals based on the current narrative state (Stefnisson and Thue 2018; Akoury et al. 2020; Kreminski et al. 2022; Martin, Harrison, and Riedl 2016). Our framework takes a different approach by transposing the interaction model where the author provides summarized intent, and by using LLMs, we generate the scene interaction scripts. In their work, Calderwood et al. (2022) fine-tuned a large language model on a data set of *Twine* stories to create a mixed initiative platform for authoring for a text-based story game. In their work, Lin and Riedl (2021) utilized high-level author context using language models for automated story generation, our work diverges by employing the high-level author-provided context to craft interactive dialogues for individual scenes.

Advances in deep neural networks and language models have driven significant progress in the related area of automated text-based story generation. In prior versions of these language models, there was a significant challenge in exerting control over the produced content, ensuring the narrative coherence aligned with common sense, and developing compelling characters that exhibit consistent behaviors (Alabdulkarim, Li, and Peng 2021). Several systems have been built to provide the neural network with high-level plot points or events. In these models, the neural network fills in the narrative gaps between events to successfully generate coherent and interesting stories (Rashkin et al. 2020; Yao et al. 2019; Wang, Durrett, and Erk 2020). The problem of automatically generating stories has also been broken down into event generation, followed by event-to-sentence generation as a two-stage process to improve coherence and plausibility of generated stories (Ammanabrolu et al. 2020). Recent advances in pretrained LLMs have shown great promise for use by professional writers as co-authors for screenplays and scripts (Mirowski et al. 2023).

A key aspect of engaging interactive narrative experiences is the dialogue with virtual characters, which reinforces in-game events and propels the narrative forward. AI has been typically used as a planner and suggester for the next event or action in writing scripts for interactive narrative games (Stefnisson and Thue 2018; Akoury et al. 2020; Kreminski et al. 2022; Martin, Harrison, and Riedl 2016). Martin et al. (2016) created an AI tool that aids storytelling in open worlds through interactive author feedback based on a story graph. Stefnisson and Thule (2018) offer "Mimisbrunnur", an AI-assisted tool where human authors maintain control over the story but receive AI suggestions. Kreminski et al. (2022) present "Loose Ends," a game providing dynamic storytelling prompts for authors. Despite the AI-driven high-level input, the intricate dialogue and interaction predomi-

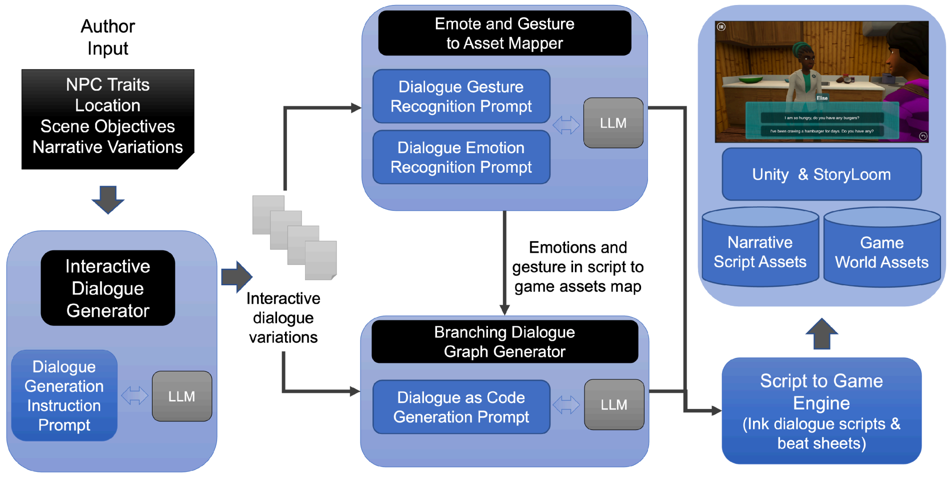

Author Input
NPC Traits
Location
Scene Objectives
Narrative Variations
Interactive Dialogue Generator
Dialogue Generation Instruction Prompt
LLM
Interactive dialogue variations
Emote and Gesture to Asset Mapper
Dialogue Gesture Recognition Prompt
Dialogue Emotion Recognition Prompt
LLM
Emotions and gesture in script to game assets map
Branching Dialogue Graph Generator
Dialogue as Code Generation Prompt
LLM
Elise
Unity & StoryLoom
Narrative Script Assets
Game World Assets
Script to Game Engine (Ink dialogue scripts & beat sheets)

Figure 1: The SCENECRAFT interactive narrative scene generation framework.

nantly originate from human authorship, as all tools focus on supporting human-led story creation. Our research utilizes LLMs to generate stimulating branching dialogues with individual non-player characters (NPCs) within a high-level scene context provided by authors, thereby flipping the usual mode of co-authorship and addressing the gap for automating detailed interaction scenes in interactive narrative generation systems.

Recent LLMs, such as OpenAI's GPT series, have showcased substantial generative potential. These LLMs have also proven their capacity to follow instructions in prompts, derive semantic understanding from natural language text, and translate instructions into code (OpenAI 2023; Chowdhery et al. 2022; Liu et al. 2023; Bubeck et al. 2023), thus providing a valuable tool for generating content. These capabilities of LLMs have been used for procedural content generation (PCG) to create game levels by fine-tuning pretrained GPT-2 models on the ASCII representation of a puzzle game, Sokoban (Todd et al. 2023). However, these PCG approaches only leverage a limited amount of the language semantic knowledge captured by the LLMs as part of pretraining.

Interactive Scene Generation Framework

The SCENECRAFT framework, shown in Figure 1, outlines key components of a structured approach for creating engaging virtual game episodes with NPC interactions derived from author input through PCG and LLM. The framework takes input from the authors on NPC traits, scene location, scene objectives, and possible narrative variations. This input is fed to an Interactive Dialogue Generator module, which generates a dialogue script using prompt base instructions on an LLM (GPT 3.5). Emote and Gesture to Asset Mapper takes this dialogue script and pairs the expressed emotions and gestures with corresponding entities from our game asset database. The Branching Dialogue Graph Generator module combines the dialogue script and any narrative variations generated to craft a story graph. The Script to Game Engine module translates the story graph, mapped emotions, and gesture assets into an Ink-like scripting language. The Ink dialogue script is read by the StoryLoom engine and rendered as a playable Unity game episode. The following sections provide additional details about various framework modules.

Interactive Dialogue Generator

We discern the author's intent by capturing four elements by presenting an author intent workbook where the author enters their choices. These elements encompass the character (including their background) with whom the player interacts, the setting or location of the interaction, the scenario, the main topics under discussion, and the tone of the conversation. The scene location and character are selected from a given list based on the character and location assets we currently support in the game. The author can choose various characters with different backgrounds. We currently support ten characters, including Kim (a female medical doctor), Quentin (a male chef), and Sam (a young gentleman). We currently support nine locations in a tropical island or a hospital, including a beach, infirmary, diner, and more. While the character and scene location is explicitly selected, the author implicitly defines the interaction scenario, central discussion topics, conversation tone, and desired outcome through their natural language story prompts. Additionally, the authors can provide alternate prompts that the generator uses to create alternate branched paths to the conversation. Unlike earlier mixed initiative author-AI interactions (Stefnisson and Thue 2018; Akoury et al. 2020), where AI offers suggestions and authors finalize the narrative, our method inverts this dynamic by having authors provide summary guidance and utilizing large language models (LLMs) to generate the dialogue.

The generator transforms authorial input into generated interaction dialogues using a predefined prompt template to facilitate the scene-generation process. This prompt instructs the LLM to generate a one-act play set in the specified location featuring the indicated character and discussing the designated topics. The prompt also directs the LLM to include

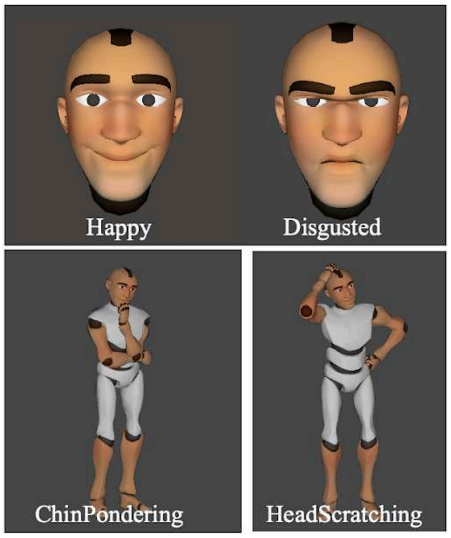


Figure 2: Examples of emote and gesture game assets.

Narrative Utterance	Emote & Gesture Map
{“dialog”: “We have the means to do it right here in this lab.”, “emotion”:“Determined”}	{“emote”:“Happy”, “scale”:0.7, “gesture”: “HandHipEngaged” }

Table 1: Emote and gesture mapping from narrative script to game assets.

descriptive details about emotions or gestures. The authors are asked to give a story prompt and one or more alternate story prompts. Using LLMs, the Interactive Dialogue Generator generates multiple dialogue variations for each prompt with utterances, emotions, and gestures (Figure 6). These dialogue scripts are used as input for subsequent modules, discussed below.

The Interactive Dialogue Generator converts author inputs into interactive dialogues and is a crucial component facilitating authorial control. We evaluate its ability to accomplish this translation through automated and human testing.

Emote and Gesture to Asset Mapper

Each NPC utterance is associated with a specific emotion and gesture from a library of Unity assets. This collection consists of a set of emotions that game characters can express. The intensity of the emotion can be controlled by a factor on a scale from 0 to 1. The prefabricated assets also consist of typical gestures that characters use to emphasize their statements. The concept of emotion and gesture building blocks has proven effective in creating realistic NPCs in previous research (Thiebaux et al. 2008). In our library of assets, we currently support six emotes (happy, sad, angry, disgusted, surprised, and fearful) and approximately 39 gestures, such as *HandsOnHips*, *ChinPondering*, and *HeadScratching*. Figure 2 illustrates emotions and gestures in our asset library. The gestures involve hand and body movements that loop with a pause between loops. The emotions are primarily expressed through facial expressions.

We employ LLMs as semantic encoders in a zero-shot classification to map the emotes and gestures for each scene utterance to the most suitable option in our asset library. This mapping is based on the name and description of the asset. The system defaults to a neutral emote, and no gesture is applied when the LLM is uncertain about the mapping. Consequently, each utterance is assigned a corresponding emote and gesture asset, which is presented along with the dialogue allowing for a realistic rendering of the NPC within the game. Table 1 shows an example of mapping from the scene script to assigned emotions and gestures. In this case, the feeling of ‘determined’ is mapped to a gesture of *HandHipEngaged* and an emotion of ‘Happy’ with an intensity scale of 0.7. The third block of Figure 3 shows a character Elise, displaying emote and gestures corresponding to being upset over farm run-off affecting fish health in the river.

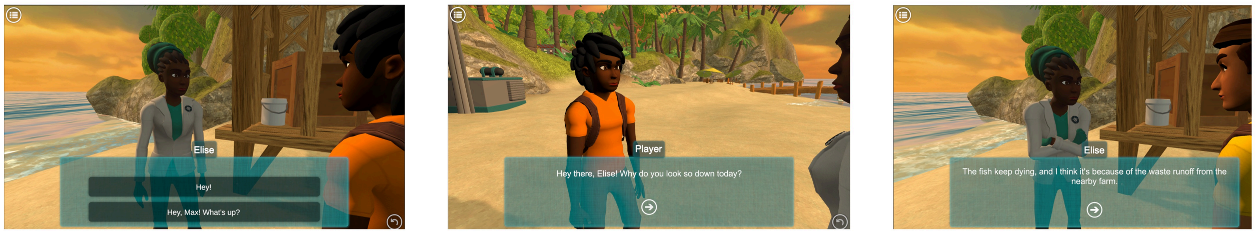
Branched Dialogue Graph Generator

The current generation of LLMs has been trained extensively on computer programs. When prompted appropriately, they have effectively translated natural language descriptions into code in many prevalent programming languages (Chen et al. 2021). We use this capability to represent the scene dialogue generated in the interactive dialogue generator as a Python object. The Python representation is a list of utterances, where each utterance is a Python dictionary with character, emotion, gesture, and dialogue content. The emotion and gesture values in the story graph nodes are from the interactive dialogue generator. We then map these values to corresponding game assets with the help of the“Emote and Gesture to Asset Mapper” discussed in the prior section. Dialogue sequences as Python objects are produced individually for every story prompt and alternate prompt supplied by the author. These multiple Python lists are combined to form the story graph.

A typical representation of a branching narrative uses a story graph representation (Riedl and Young 2006). The scene is represented as a directed graph, with each dialog utterance represented by a node and each arc corresponding to the next utterance. A path through the graph represents an unfolding conversation between the player and the NPC. The nodes with multiple edges coming out of them indicate a choice to control the progression of the conversation. Figure 4 shows an example of a branched dialogue graph. We generate such a graph from the conversation variants generated by the Interactive Narrative Generator. All occurrences of a character speaking a specific sentence in multiple variants are represented in a single node. The story graph is used as the basis to generate the scene script for StoryLoom.

Script to Game Engine

We employ a rapid prototyping tool based on the StoryLoom Architecture (Mott et al. 2019). This tool utilizes a text-based representation of the interactive narrative to script in-game interactions. The script-to-game engine relies on two types of assets: game world assets and narrative script assets. The game world assets are the building blocks phys-



Hey!

Hey, Max! What's up?

Hey there, Elise! Why do you look so down today?

The fish keep dying, and I think it's because of the waste runoff from the nearby farm.

Figure 3: Screenshots from game episodes generated by SCENECRAFT.

ically rendered in the game and correspond to entities the player interacts with. Examples of game world assets are locations, characters, and props. Narrative assets consist of dialogue scripts and beat sheets that capture the interactions between the player and the game world assets. The *dialogue scripts* represent the conversations, narrations, and branching choices presented to the player. The *beat sheet* describes the overall story and the key events and triggers when the player reaches a location, interacts with a prop, or starts a conversation with a character. A beat event or trigger represents a moment in the game narrative where something changes in the story. The idea is that the player is allowed the freedom to move around in the game, and when they interact with one of the trigger points specified in the beat sheet and if the conditions are met, an interaction is triggered. In our StoryLoom, the beat sheet is a spreadsheet identifying the character or prop that triggers a dialogue script, mapping to the dialogue script, and some rules regarding the trigger. Figure 5 represents a high-level architecture overview of the script to the game engine. The narrative progression during gameplay relies on dialogue scripts and beat sheets, which serve as the core components of each narrative episode. In StoryLoom, the creation of these scripts and beat sheets is a product of two integrated elements: The first is the 'Branching Dialogue Graph Generator', which crafts a story graph to map out potential dialogue paths. The second is the 'Emote and Gesture to Asset Mapper', which scans the narrative and identifies the necessary game assets (like characters' emotions or gestures) for each line of dialogue. Together, they generate the StoryLoom dialogue scripts and beat sheets.

The dialogue is scripted using a language that extends Ink (https://www.inklestudios.com/ink), a narrative scripting language for games. We have extensively modified the Ink scripting language. Our modified Ink version preserves its original capabilities for representing narrative dialogues and branching narratives. Additionally, it allows us to initiate Unity game engine-specific actions such as spawning characters and props, controlling the camera, activating workbooks, adding conditional triggers on game objects, and setting and modifying the game state, among other features. The scene dialogue generated in the framework's previous stages is converted into our custom Ink-like script and beat sheet that can be played in the game engine. During the conversion process, the scene location is established, and all characters from the script are moved to their identified spawn points, including both author-selected and auxiliary characters introduced by the generated script. The location, characters, emotes, and gestures are built on prefabricated game world assets. The conversion process establishes the beat sheet, initiating the conversation when the player approaches and interacts with the appropriate character. Finally, Ink script choice points and dialog path flow are set up. The Ink script is set up such that when a player decides on a conversation alternative, all the choices made by the player are remembered in the game state, and the dialogue responds to the choice as the conversation proceeds. The corresponding emote and gesture are triggered in the Ink script just before an NPC renders a dialogue. Figure 3 shows a sequence of screenshots from the generated game episode.

SCENECRAFT Implementation

Given the author's prompts, SCENECRAFT operates as follows. Figure 6(a) shows the author's initial input context obtained by the framework. It is based on the input provided by the author through the author's intent workbook. The captured details include information about all the NPC (non-player) characters and the character played by the user, along with information about the location, possible spawn points, story prompt, and possible variants. The framework identifies the spawn points and scene location as assets available in the game. The details from the author's intent are subsequently used in LLM prompts that are used to generate a scene script realized in the Unity game. Figure 6(b) shows the prompts used in sequence to translate the author's input into the scene script.

SCENECRAFT translates the author's input into a dialogue-driven scene, including dialogue flows for all variants. It then converts the generated scene script into a Python object that the subsequent framework modules can easily consume. It also uses the LLM to identify specific emotions and gestures corresponding to each dialogue. Figure 6(b) shows the emotion prompt; the gestures prompt is similar and not shown here due to space constraints. The LLM is also used to address Python compilation with the generated Python script object. As the output from the LLM is stochastic, it may sometimes be necessary to run these requests to the LLM multiple times to generate a valid Python script object. The terms in curly brackets on the first prompt in Figure 6(b): {location}, {scene}, {non_player_job} are elements captured as part of the author intent. In subsequent prompts, the variable {script} is the script generated in the previous interaction with LLM, and variables like {emote_list}

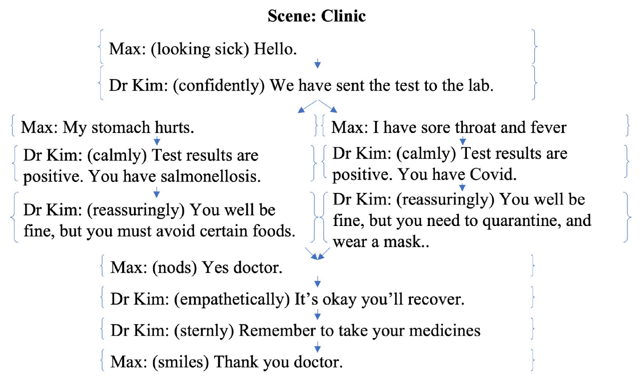

Scene: Clinic
Max: (looking sick) Hello.
Dr Kim: (confidently) We have sent the test to the lab.
Max: My stomach hurts.
Dr Kim: (calmly) Test results are positive. You have salmonellosis.
Dr Kim: (reassuringly) You well be fine, but you must avoid certain foods.
Max: I have sore throat and fever
Dr Kim: (calmly) Test results are positive. You have Covid.
Dr Kim: (reassuringly) You well be fine, but you need to quarantine, and wear a mask..
Max: (nods) Yes doctor.
Dr Kim: (empathetically) It's okay you'll recover.
Dr Kim: (sternly) Remember to take your medicines
Max: (smiles) Thank you doctor.

Figure 4: Example of a generated branched narrative graph.

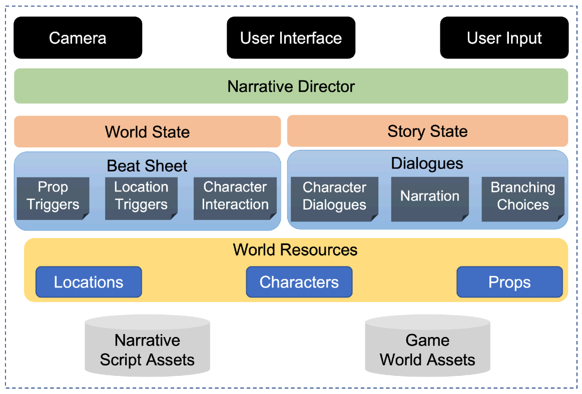


Figure 5: Architecture of script to game engine module.

are lists extracted from the Python objects created in earlier prompts. The prompts are chained together to create a script Python object.

The Python object is a list of dialogue elements. Each dialogue element in the script is a Python dictionary containing the dialogue along with the identity of the character who delivers the dialogue, the emote representing the character's emotional state, and any physical gestures to display during the dialogue. A script can have multiple variants resulting in a collection of dialogue lists, one for each variant. The Python object is converted to a story graph. The final step is the creation of the game script, which is exported as a script written for our extended Ink language. The initial sections of the Ink script specify the location, spawn points, characters, and other details to identify all the necessary game elements/assets to initialize. Each NPC with dialogues has an Ink script file illustrating how different conversation paths interact and how the user's choices can impact the conversation unfolding. The custom extension to Ink is used to indicate the emotions and gestures of characters at the appropriate time. We use state variables to track the user-chosen path, and these state variables determine how their conversation branches and progresses in the scene. StoryLoom performs several functions: it reads the script, interprets various commands and narrative paths, and constructs the game scene. The output is a playable game episode.

Evaluation

We assess the generated game episodes using automated and human evaluation methods. There are inherent challenges associated with automatic evaluation for language generative models for semantic understanding, coherence, and cohesion of long passages, which necessitates including human evaluation. For automated evaluation, we use three high-level prompts and generate ten scene scripts for each prompt using various paraphrasings. The automated evaluations are conducted based on this generated dataset.

Automated Evaluation

We measure the creativity of our system by calculating the ROUGE-L score between two generated scripts (Lin 2004). ROUGE-L is a measurement that evaluates the similarity of sentence structures by identifying the longest sequence of matching n-grams that occur in the same order. We compare the ROUGE-L metric distribution when scripts are generated from the same prompt versus those from different prompts. Although we expect higher ROUGE-L scores for scripts derived from the same prompt, we consider high variation in the distribution of ROUGE-L values as a sign of novelty in the generated script.

To assess alignment and adaptability to author input, we convert the prompt and scripts into sentence embeddings using Sentence-BERT (Reimers and Gurevych 2019). We can use the embedding to overcome the substantial size differences between prompts and generated scripts. Cosine similarity is employed to measure the similarity between these embeddings. We then plot the distribution of distances for prompt-to-generated script and prompt-to-unrelated script. Statistically distinct distributions will indicate that the generated scripts are aligned with their corresponding prompt.

Human Evaluation

Participants for this study were recruited through a purposive sampling method, utilizing the researcher's established professional and academic network. We recruited 9 participants with a broad range of game development expertise (ages 25 to 53) to use SceneCraft to generate scenes. The participants consisted of two graduate students and one research scientist. All three were familiar with video games. Additionally, it consisted of one game software engineer and two digital artists familiar with game development. Finally, we had one participant who was a math teacher, a software engineer, and one who was a scientist in biological sciences, all three with minimal exposure to video games. Each participant engaged with SceneCraft during a 45 minute session. The session started with an introduction to the framework along with an explanation of how to provide author intent for scene generation. Following the introduction, the participant used the author's intent workbook to record their intent as discussed in an earlier section. The participants waited for a few minutes as SceneCraft generated the game episode before playing. Once the game episode was generated, the participants played the episode and then responded to five questions listed below on the

(a)

```
{"player": "Max", "non_player": "Elise",
  "non_player_job": "a lab technician",
  "location": {
     "name": "island", "value": "CampCILI"
  },
  "spawn": { "scene": "Beach",
    "Max": "BeachWelcome",
    "Elise": "BeachHut"
  },
 "story_prompt": "Elise believes the lake's fish population is dying because of  water temp",
 "variants": ["Elise believes that the lake's fish   population is dying because of  waste runoff from a farm"],
  "characters": [ "Max", "Elise"]}
```

(b)

Script Generation Prompt

Write an interesting one act play about {script}. Set in a {location} {scene} with {player}(a young person) and {non_player}({non_player_job}). Include hand gestures and emotions for {non_player} in the script.

Script Variant Prompt

Given a formatted play, {script} rewrite it with the change that, {variant}. Reuse dialogues.

Emotion to Game Asset Mapping Prompt

Definition:
The system only supports, Category: Neutral, Happy, Angry, Sad, Surprised, Disgusted, Fearful. Scale: can only be between 0 and 1
Examples:
smiling, Category: Happy, Scale: 0.5; upset, Category: Sad, Scale: 0.8; nervously, Category: Fearful, Scale: 0.2
Task:
Write the following list of emotes into a python list of dictionary items, using the keys "emote", "category" and "scale": {emote_list}. emotions =

Figure 6: (a) Structured author's input used for all subsequent generations. (b) LLM prompts to generate scene scripts.

Likert scale from 1 to 7. The participants generate up to 2-3 game episodes with different prompts and record their scores for the questions. Each participant's score for each question is average across all their game plays. The data presented is the average of all the participants' scores.

These questions are based on the User Experience Questionnaire (UEQ) (Schrepp, Hinderks, and Thomaschewski 2017), a validated quantitative tool to measure psychometric properties of a product user experience. The questionnaire was created by usability experts in 2005. The UEQ has multiple versions with different sets of questions. We have used a subset of questions that apply to our specific scenario and map to the six usability measures described by UEQ: Attractiveness, Perspicuity, Efficiency, Dependability, Stimulation, and Novelty.

- How would you rate the ease of generating new narratives using the tool (Ease of Use)? From 'Complicated' to 'Easy'. SceneCraft's ultimate objective is to simplify the process of scene generation. By assessing ease of use, we can determine if SceneCraft successfully lowers the barrier to entry and allows writers and designers to generate narratives efficiently and intuitively. This question addresses UEQ measure to gauge how easy the framework is to learn and use ('Perspicuity') and if the users can accomplish their tasks efficiently ('Efficiency').
- How engaging and unexpected did you find the generated game (Creativity)? From 'Dull' to 'Creative'. This question evaluates SceneCraft's ability for originality and unexpectedness of the exchanges it creates. Based on these scores, one can judge the extent to which SceneCraft can break from conventional storytelling, thereby creating unique player experiences. This question validates that the framework fulfills the 'Novelty' criterion. It also addresses the fun aspect of the framework ('Stimulation').
- To what extent did the interaction in the game accurately represent and respond to your intent (Adaptability)? From 'Unsatisfactory' to 'Satisfactory'. As the authors specify their intent at the beginning of the process to initiate the generation of the scene, this question aims to assess the control authors perceive when using the tool. This question validates if the designer feels in control of the interaction ('Dependability').
- How would you evaluate the game's characters and storyline in terms of their believability, coherence, and engagement (Dependability)? From 'Not Interesting' to 'Interesting'. SceneCraft, as part of its generation, writes dialogues and renders emotion and gestures to characters to match the feeling of what the NPC says. This question aims to apprehend if this feature of SceneCraft was successful. This question validates that the effects of their suggestions are as reflected in the generation as expected ('Dependability').
- Overall, how pleased are you with the game generated by the tool (Satisfying)? From "Not Satisfied" to "Satisfied". This general metric offers a holistic view of SceneCraft's performance. A satisfying experience suggests SceneCraft meets expectations across all aspects of scene generation. This question validates that the overall impression of the product is positive ('Attractiveness').

Results and Discussion

We showcase the ability of our framework to generate novel game interactions aligned with the author's intent. In the following sections, we validate these capabilities through automated and human evaluation.

Creativity

As described above, we use the ROUGE-L score to measure the similarity between generated scripts. ROUGE-L is particularly useful as it accounts for the longest common subsequence between two texts. ROUGE-L is typically used to compare a generated text to a reference text, but we use it to compare two generated texts.

Figure 7 displays the number of pairs corresponding to each ROUGE-L value, comparing scripts that were generated from identical prompts with those generated from unrelated prompts. Our analysis reveals that, as anticipated, the ROUGE-L scores for generated script pairs originating

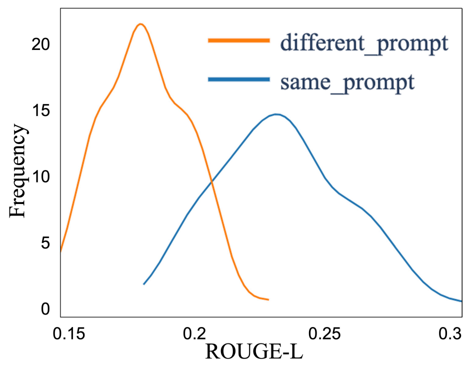


Figure 7: ROUGE-L distribution for generated scripts from the same and unrelated prompts.

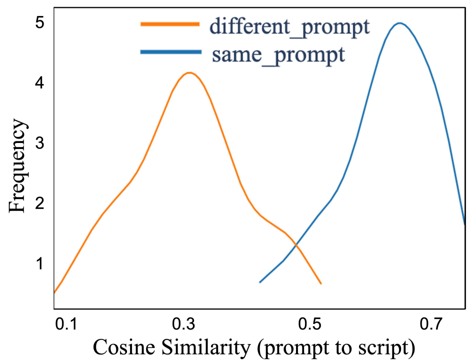


Figure 8: Cosine similarity between prompt string and generated script. Baseline (orange) corresponds to prompt string compared with unrelated script.

from the same prompt were higher than those for the baseline, which consisted of script pairs from different prompts. This outcome confirms that scripts generated from the same prompts exhibit greater similarity than unrelated prompt pairs. However, we observed an interesting phenomenon where several generated scripts from the same prompt were as dissimilar as unrelated prompt pairs, as evidenced by the overlap in the ROUGE-L score distributions.

Additionally, we noted a significantly higher variance in the ROUGE-L distribution for same-prompt script pairs, with the variance being nearly twice that of the baseline unrelated-script pair distribution. This finding suggests that while the generation process tends to produce more similar scripts when given the same prompt, it also generates a wide range of scripts with varying degrees of similarity. This variability in similarity indicates that the script generation capabilities show significant variability and novelty. We use human evaluation, discussed below, to confirm these findings.

Alignment with Author Intent

We evaluate the alignment between the author's intent and the generated script by examining the semantic similarity between the prompt and its corresponding generated script. Given the substantial word count discrepancy between the prompts and their generated scripts, we chose sentence embedding similarity over the ROUGE metric for evaluation. We expect a prompt to be most aligned with the script it generated and less aligned with unrelated scripts. Figure 8 illustrates the distribution of cosine similarity scores, indicating a statistically significant difference (p-value = 2.6e-62) in similarity distribution between the prompt and its generated script compared to an unrelated script.

To further investigate whether specific utterances within the script matched the author's intent, we evaluated the similarity between the author's input and individual sentences in the script, selecting the maximally similar sentence among them. Figure 9 displays the corresponding similarity distributions, revealing a pattern of very high similarity (median >0.7) between sentences in the prompt and those in its generated script. Interestingly, the baseline comparison with an unrelated script exhibits a bimodal distribution. This type of distribution may arise from the nature of the prompts used to generate the evaluation scripts - specifically, two of these prompts relate to sickness on the island ('fishes falling sick' and 'pandemic on the island'). These similarities in theme could lead to overlapping sentences in scripts generated from these disparate prompts.

Our quantitative analysis clearly aligns the author's intent and the generated script. We corroborate these findings with human evaluation.

Human Evaluation of SCENECRAFT

Human evaluators evaluated the ScreenCraft framework on five questions corresponding to the following metrics: 'Ease of Use', 'Creative', 'Adaptable', 'Dependable', and 'Satisfying'. As described above, 9 participants assessed each metric after generating game episodes based on 2-4 prompts. As shown in Figure 10, SCENECRAFT received an average score of 6.44 out of a maximum of 7 for 'Ease of Use,' demonstrating that participants found the framework user-friendly and straightforward for generating game episodes. The standard deviation was 0.53, showing a small spread in the scores ranging from 6 to 7. For the 'Creative' metric, SCENECRAFT received a mean score of 5.76. Even though the score is relatively high, there is still potential for improvement. The standard deviation was 0.80, implying a more diverse range of opinions. Scores ranged from 5 to 7 for the SCENECRAFT adaptability metric, with an average of 6.22, indicating the users' confidence in the reliability and stability of the framework. On the 'Dependable' metric, SCENECRAFT scored a high average of 6.06, signifying users found the characters believable, coherent, and engaging. The standard deviation was 0.77, again indicating a larger spread, with scores ranging from 5 to 7. Lastly, when asked to evaluate if the overall experience was satisfactory SCENECRAFT received a high mean score of 6.19, showing that users were satisfied with their overall experience, except for a single outlier of 4.5. The outlier score was for a case where the participant gave unrelated prompts for the main story and alternates. Overall, it was noted that participants with more extensive game development experience tended

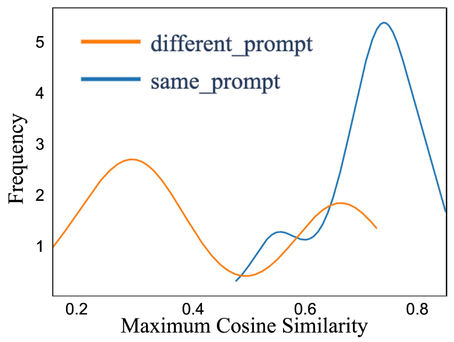


Figure 9: Maximum cosine similarity between prompt string and sentences in the generated script. Baseline (orange) corresponds to prompt string compared with unrelated script.

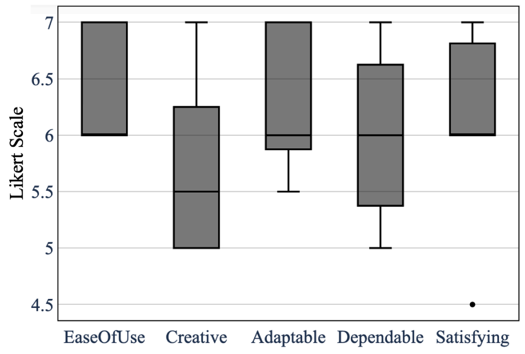


Figure 10: Score distribution for each of the human evaluation questions.

to give lower scores.

Users received the SceneCraft framework favorably, particularly concerning ease of use, satisfaction, and adaptability. The high scores for these metrics suggest that SceneCraft effectively aids authors in transforming their ideas into game episodes with minimal effort. While SceneCraft received positive feedback on creativity and character engagement, there is still room for improvement. Single-sentence story prompts provided by participants were possibly limiting the unique context required for LLMs to showcase higher creativity. Future iterations should encourage participants to provide more elaborate prompts.

Limitations

We recognize that certain factors limit the complete realization of our framework's potential, and there are specific aspects we need to address. Our framework is built on publicly available LLM implementations. As with most LLM work, it will be important to investigate how LLM output aligns and conforms to the ethical principles of the authors using the framework. In this light, future studies should identify ways to align these models' output with standard ethical guidelines. Second, our proposed framework primarily targets automating individual scene creation within a larger narrative structure. However, generating a coherent and compelling narrative requires a high-level plot that binds these individual scenes together. Addressing this limitation will involve integrating a narrative planner and experience manager to drive the higher-level plot.

Conclusion

Authoring narrative-centered games that create engaging, interactive story-based experiences that can respond to individual player choices has long been a central challenge for intelligent narrative technologies. To address this challenge, we introduced SceneCraft, a narrative scene generation framework that leverages the capabilities of LLMs to automate the generation of non-player character interactions integral to unfolding plot events. By using LLMs to extract semantic aspects of the generated script, the in-game interaction reflects the author's objective through dialogue utterances, emotes, and gestures. SceneCraft delivers engaging player experiences with branching interactions in a 3D virtual environment based on simple author instructions within a few minutes.

Our empirical evaluation has demonstrated SceneCraft's effectiveness in creating engaging narrative experiences that align with authorial intent. The evaluation incorporated a two-pronged approach integrating automated metrics and human assessment to appraise SceneCraft's capabilities. The automated evaluation using ROUGE-L score and sentence similarity shows that SceneCraft can generate diverse scripts while maintaining alignment between the author's intent and narrative episode. The effectiveness of SceneCraft was also demonstrated through human evaluation, where the system was favorably received, particularly in terms of ease of use, adaptability, and user satisfaction. These high scores indicate that SceneCraft effectively aids authors in turning their ideas into playable game episodes with minimal effort. The more moderate scores in creativity suggest room for improvement, which we hope to address in our future work. The empirical results strongly support the potential of SceneCraft as an effective tool for automating the generation of interactive narratives in games.

In future work, it will be important to build on these findings by investigating extensions to SceneCraft incorporating embodied conversational agents, generation of narrative plots, and narrative events triggered by interactions with NPCs in the game. LLMs have shown promise as zero-shot planners (Huang et al. 2022), taking a zero-shot planning approach to these problems holds considerable promise for the future. Finally, it will be important to investigate integrations of SceneCraft with narrative experience managers to further enrich the narrative game design process in order to create a broad range of player-adaptive narrative games.

Acknowledgments

This work is supported by the National Science Foundation under award DRL-2112635. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Akoury, N.; Wang, S.; Whiting, J.; Hood, S.; Peng, N.; and Iyyer, M. 2020. STORIUM: A Dataset and Evaluation Platform for Machine-in-the-Loop Story Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, (EMNLP)*, 6470–6484.

Alabdulkarim, A.; Li, S.; and Peng, X. 2021. Automatic Story Generation: Challenges and Attempts. In *Proceedings of the Third Workshop on Narrative Understanding*, 72–83. Virtual: Association for Computational Linguistics.

Ammanabrolu, P.; Tien, E.; Cheung, W.; Luo, Z.; Ma, W.; Martin, L. J.; and Riedl, M. O. 2020. Story realization: Expanding plot events into sentences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 7375–7382.

Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y. T.; Li, Y.; Lundberg, S.; Nori, H.; Palangi, H.; Ribeiro, M. T.; and Zhang, Y. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. arXiv:2303.12712.

Calderwood, A.; Wardrip-Fruin, N.; and Mateas, M. 2022. Spinning Coherent Interactive Fiction through Foundation Model Prompts. In *Proceedings of the 13th International Conference on Computational Creativity, Bozen-Bolzano, Italy, June 27 - July 1, 2022*, 44–53. Association for Computational Creativity (ACC).

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H. P. d. O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, 9118–9147. PMLR.

Kreminski, M.; Dickinson, M.; Mateas, M.; and Wardrip-Fruin, N. 2020. Why Are We Like This?: The AI architecture of a co-creative storytelling game. In *Proceedings of the 15th International Conference on the Foundations of Digital Games*, 1–4.

Kreminski, M.; Dickinson, M.; Wardrip-Fruin, N.; and Mateas, M. 2022. Loose Ends: a mixed-initiative creative interface for playful storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 18, 120–128.

Kreminski, M.; Wardrip-Fruin, N.; and Mateas, M. 2020. Toward Example-Driven Program Synthesis of Story Sifting Patterns. In *AIIDE Workshops*.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 74–81. Barcelona, Spain: Association for Computational Linguistics.

Lin, Z.; and Riedl, M. O. 2021. Plug-and-blend: a framework for plug-and-play controllable story generation with sketches. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 17, 58–65.

Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; and Neubig, G. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9): 1–35.

Martin, L.; Ammanabrolu, P.; Wang, X.; Hancock, W.; Singh, S.; Harrison, B.; and Riedl, M. 2018. Event representations for automated story generation with deep neural nets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Martin, L. J.; Harrison, B.; and Riedl, M. O. 2016. Improvisational computational storytelling in open worlds. In *Interactive Storytelling: 9th International Conference on Interactive Digital Storytelling, ICIDS 2016, Los Angeles, CA, USA, November 15–18, 2016, Proceedings 9*, 73–84. Springer.

Mirowski, P.; Mathewson, K. W.; Pittman, J.; and Evans, R. 2023. Co-Writing Screenplays and Theatre Scripts with Language Models: Evaluation by Industry Professionals. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 1–34.

Mott, B. W.; Taylor, R. G.; Lee, S. Y.; Rowe, J. P.; Saleh, A.; Glazewski, K. D.; Hmelo-Silver, C. E.; and Lester, J. C. 2019. Designing and developing interactive narratives for collaborative problem-based learning. In *Interactive Storytelling: 12th International Conference on Interactive Digital Storytelling, ICIDS 2019, Little Cottonwood Canyon, UT, USA, November 19–22, 2019, Proceedings 12*, 86–100. Springer.

Naul, E.; and Liu, M. 2020. Why story matters: A review of narrative in serious games. *Journal of Educational Computing Research*, 58(3): 687–707.

Oliver, E.; and Mateas, M. 2021. Crosston tavern: modulating autonomous characters behaviour through player-NPC conversation. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 17, 179–186.

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774.

Ramirez, A.; and Bulitko, V. 2014. Automated planning and player modeling for interactive storytelling. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(4): 375–386.

Rashkin, H.; Celikyilmaz, A.; Choi, Y.; and Gao, J. 2020. PlotMachines: Outline-Conditioned Generation with Dynamic Plot State Tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 4274–4295.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3982–3992.

Riedl, M. O.; and Bulitko, V. 2013. Interactive narrative: An intelligent systems approach. *Ai Magazine*, 34(1): 67–67.

Riedl, M. O.; and Young, R. M. 2006. From linear story generation to branching story graphs. *IEEE Computer Graphics and Applications*, 26(3): 23–31.

Riedl, M. O.; and Young, R. M. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39: 217–268.

Schrepp, M.; Hinderks, A.; and Thomaschewski, J. 2017. Construction of a Benchmark for the User Experience Questionnaire (UEQ). *International Journal of Interactive Multimedia & Artificial Intelligence*, 4(4).

Stefnisson, I.; and Thue, D. 2018. Mimisbrunnur: AI-assisted authoring for interactive storytelling. In *Proceedings of the AAAI Conference on artificial Intelligence and Interactive Digital entertainment*, volume 14, 236–242.

Thiebaux, M.; Marsella, S.; Marshall, A. N.; and Kallmann, M. 2008. Smartbody: Behavior realization for embodied conversational agents. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*, 151–158.

Todd, G.; Earle, S.; Nasir, M. U.; Green, M. C.; and Togelius, J. 2023. Level Generation Through Large Language Models. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 1–8.

Wang, S.; Durrett, G.; and Erk, K. 2020. Narrative interpolation for generating and understanding stories. *arXiv preprint arXiv:2008.07466*.

Ware, S. G.; Garcia, E.; Fisher, M.; Shirvani, A.; and Farrell, R. 2022. Multi-agent narrative experience management as story graph pruning. *IEEE Transactions on Games*.

Yao, L.; Peng, N.; Weischedel, R.; Knight, K.; Zhao, D.; and Yan, R. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7378–7385.