LLMaker: A Game Level Design Interface Using (Only) Natural Language

Roberto Gallotta
Institute of Digital Games
University of Malta
Msida, Malta
roberto.gallotta@um.edu.mt

Antonios Liapis
Institute of Digital Games
University of Malta
Msida, Malta
antonios.liapis@um.edu.mt

Georgios Yannakakis
Institute of Digital Games
University of Malta
Msida, Malta
georgios.yannakakis@um.edu.mt

***Abstract*—In this demo paper we present *LLMaker*, a video-game level design tool that operates solely via natural language instructions. Unlike existing level design tools, *LLMaker* allows the designer to express their intent in an intuitive way, interacting with a cognitively undemanding interface, while still ensuring the generated levels adhere to specific domain and playability constraints.**

***Index Terms*—creativity support tools, design assistant, mixed-initiative tools, large language models**

I. Introduction

Design tools [1], ranging from simple CAD programs to video-game editors such as Super Mario Maker, are powerful programs that let users express their creativity by interacting via their interfaces. However, designing an interface that is intuitive and does not stifle user creativity is a challenge in itself. Most of these tools rely on sliders, buttons, and adjustable knobs as a simple solution to this problem. Explaining an idea or a requirement in natural language is instead more intuitive, lowering the level of expertise required to interact with these tools. With the recent surge of large language models (LLMs) in almost all aspects of everyday life, it is not difficult to imagine a future where tools integrate LLMs as a faster medium to communicate user intentions to the program [2]. To showcase the power of this approach, we introduce *LLMaker*, a video-game level design tool that leverages both LLMs and foundation models (FDs) to generate levels for a dungeon crawler game, complete with ad-hoc graphical assets. Games provide an ideal domain for this kind of applications, as they exhibit both constraints and a highly creative design process. In this demo, we aim to explore how a natural language-only application is compared to traditional tools, and whether *LLMaker* provides the basis for a more unconstrained creative process. During the demo session, we will let attendees create levels alongside our advice and support. This will also highlight issues with usability in unconstrained settings, with creatives that may wish to explore more exotic themes than the ones in our internal tests.

This project has received funding from the European Union's Horizon 2020 programme under grant agreement No 951911.

979-8-3503-5067-8/24/$31.00 ©2024 European Union

II. The LLMaker tool

LLMaker allows the user to design a level for a hypothetical dungeon crawler video-game, in the vein of Darkest Dungeon[1]. Generation is driven exclusively via natural language instructions. A LLM interprets the request and, via function calling [3], generates as response the function name and parameters that will be executed on our back-end system. Parameters for the function are filled out either via extrapolation from the user request, or are generated by the LLM directly. For example, the user can specify the "name" of an enemy to create, and the LLM will use it in the function call, while generating the enemy's "description" accordingly. As we leverage a back-end system, we can enforce constraints that the LLM is forced to adhere to. These constraints also impact the the value range of generated parameters and, for textual parameters, we condition their values by means of prompting. The back-end functions, which the LLM can call, affect the level by: (1) adding, removing, or modifying a room; (2) adding or removing a corridor; and (3) adding, removing or altering an enemy, a trap, or a treasure chest. Once the function is executed, the LLM provides a short summary of the changes to the user. In case a function fails to execute, a functional error [4] is returned to the LLM, which can decide whether to try calling the function again with different parameters, or simply inform the user of the problem.

While the level is described in its entirety in a structured text format (JSON), we also generate assets (backdrops for the room or corridor, and sprites for enemies, traps, and treasure chests) using Stable Diffusion models [5]. We employ a fine-tuned model, A-Zovya RPG [6], to generate images from the descriptions provided by the LLM.

We can identify three main components the user can interact with in the main *LLMaker* interface, as shown in Figure 1. On the right side of the interface, we can find the "Chat Area". This is where the user can ask questions about the current level, or request changes (as mentioned above). To the right of the interface, the current room or corridor is displayed as its image background along with any enemies, traps, and treasure chests. Hovering over any of these entities will make a tool-tip appear, summarising the properties of the entity. Finally,

[1] Red Hook Studios, 2016

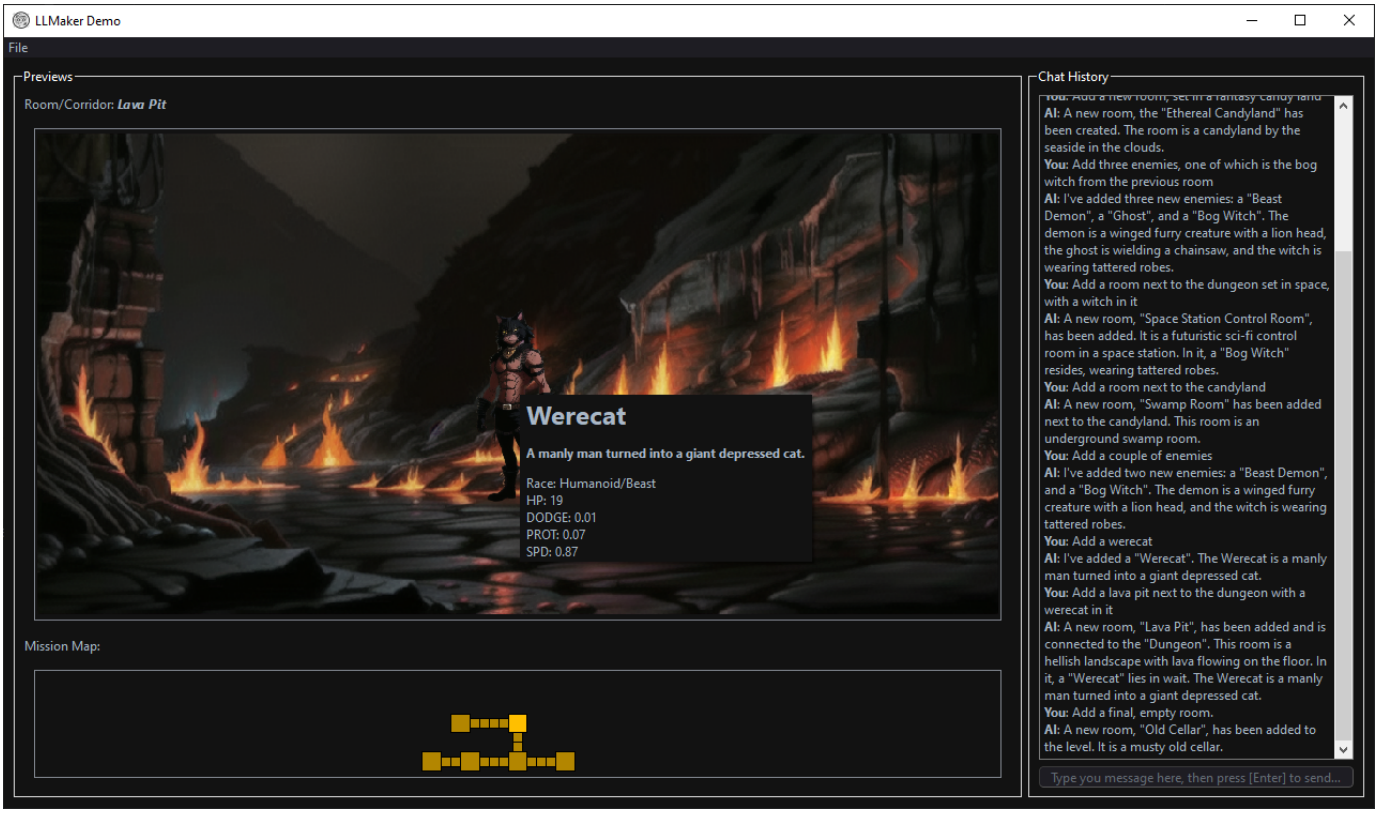

Fig. 1. A screenshot of our chat-based level design interface, *LLMaker*. On the upper left pane, the preview of the currently selected room. On the lower left pane, the generated level layout, with rooms (larger squares) and corridors (smaller squares). On the right pane, the chat area with the conversation between designer and LLM.

a mini-map at the bottom of the interface shows rooms and corridors on a tile-based grid. Users can move from one room or corridor to another simply by clicking on the mini-map.

The entire application is built with Python using PyQt for the user interface. *LLMaker* is powered by GPT-3.5-Turbo-1106, a proprietary LLM provided by OpenAI. *LLMaker* interacts with the model via API calls using Python. The functions in our back-end system allow for creation, removal, or editing of rooms, corridors, enemies, treasures, and traps. The description of these methods, necessary for function calling, is included in each API call following YAML formatting, as recommended in the OpenAI API developer guidelines.

III. Discussion and Conclusion

LLMaker currently generates content as both text (the level descriptors) and images (the level background and entity sprites). We intend to broaden the capabilities of the application by generating ancillary resources such as background music, synthesised taunts, and a voiced narrator.

Future research will focus on the integration of a mission objective for the work-in-progress level, with checks for completion, as well as a simulation-based evaluation of level difficulty and player engagement.

Finally, while the current language model follows the typical "AI Assistant" paradigm, which falls under the "support" role [7], it would be interesting to change the behaviour of the language model from assistant to something more proactive, following the recently introduced user-centered framework for human-AI co-creativity [8].

References

[1] S. Deterding, J. Hook, R. Fiebrink, M. Gillies, J. Gow, M. Akten, G. Smith, A. Liapis, and K. Compton, "Mixed-initiative creative interfaces," in *Proceedings of the CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 2017.

[2] R. Gallotta, G. Todd, M. Zammit, S. Earle, A. Liapis, J. Togelius, and G. N. Yannakakis, "Large language models and games: A survey and roadmap," *arXiv preprint arXiv:2402.18659*, 2024.

[3] OpenAI, "Function calling." https://platform.openai.com/docs/guides/function-calling, 2023. Accessed 27 Feb 2024.

[4] E. Buoannno, "Functional error handling," in *Functional Programming in C#*, ch. 6, Manning Publications Co., 2017.

[5] 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*, 2022.

[6] Zovya, "A-Zovya RPG Artist Tools." https://civitai.com/models/8124/a-zovya-rpg-artist-tools, 2023. Accessed 16 April 2024.

[7] M. L. Maher, "Computational and collective creativity: Who's being creative?," in *Proceedings of the International Conference on Innovative Computing and Cloud Computing*, 2012.

[8] C. Moruzzi and S. Margarido, "A user-centered framework for human-ai co-creativity," in *Proceedings of the Conference on Human Factors in Computing Systems*, 2024.