Surreal VR Pong: LLM approach to Game Design

Jasmine Roberts
Microsoft & UCSD
jar072@ucsd.edu

Andrzej Banburski-Fahey
Microsoft

Jaron Lanier
Microsoft

Abstract

The increase in complexity from 2D to 3D game design makes it fascinating to study from a computational creativity perspective. Generating images given text descriptions using models like DALL-E has recently increased in popularity. However, these models are limited to generating 2-dimensional outputs. While outputs of these models can be used to stylize 3d objects with variable textures, they cannot produce mesh-level interactions. We introduce Codex VR Pong as a demonstration of controlled non-deterministic game mechanics leveraging generative models. We are proposing that prompt-based creation can become part of gameplay rather than just part of game development.

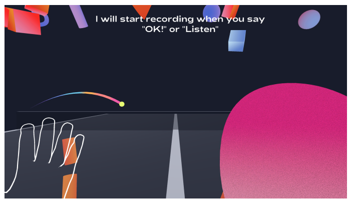
1 Introduction

The recent progress in Large Language Models (LLMs), like GPT-3 [1], has enabled surprising code generation capabilities in the form of models like OpenAI Codex [2] or AlphaCode [3]. Initial investigations have shown that these models can write short programs and match the performance of average competition coders [3], as well as solve university-level math problems [4]. In this work, we demonstrate the potential for real-time gaming applications of such generative models. While previous work has shown that it is possible to create NPCs capable of following natural language orders [5] or text adventure games [6], here we investigate the potential for creative interactions in Virtual Reality between game objects that the developers would not be able to plan for beforehand.

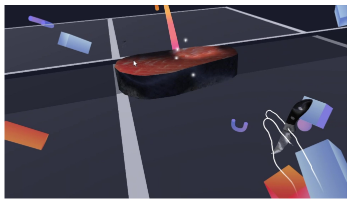
Traditional 3D game design is significantly more challenging than 2D, and this is even more pronounced when attempting to generate content at runtime. While recent models like DALL-E 2 [7], Midjourney [8] or Stable Diffusion [9] could conceivably be used to generate 2D textures that could be animated with other tools [10], 3D content generation from scratch is very nascent [11]. This leads us to integrating Codex with the Unity game engine [12], which has the potential for unbounded creativity from the user/player, but introduces nondeterminism that carries the risk of removing all sense of agency from game design. In this work, we choose to study a restricted semi-deterministic setting with some implemented hard-coded mechanics while trying to not compromise our investigation of LLMs in-game. In homage to the first video game made [13], we developed a surreal game of VR tennis for two, in which players can both change the ball as well as their paddles to arbitrary 3D objects and allow for semantically relevant interactions between these morphed objects.

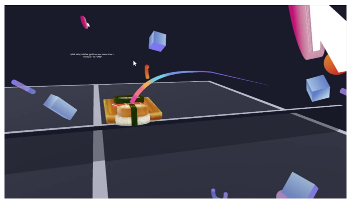
2 Surreal tennis game

To create a multimodal 2-player game of surreal tennis we need several different technologies working together. The first of these is naturally the integration of Codex with Unity. While Codex is most capable in Python and Javascript, it has been fine-tuned on majority of GitHub, which includes $C\#$ code used in Unity projects, as can be evidenced by simple experiments with the OpenAI API. The immediate obstacle one encounters is that $C\#$ code needs to be compiled, and by default Unity does not ship with a compiler that would allow evaluation of code generated at runtime. Fortunately, the



(a) Default game scene


(b) Ball $\rightarrow$ salmon, paddle $\rightarrow$ knife


(c) Collision results in sushi

open source Roslyn $C\#$ compiler has been successfully implemented in Unity previously [14]. To integrate the OpenAI API, we expanded a previous attempt to build a .NET wrapper [15].

We used both text generation through GPT-3 and code generation through Codex to achieve the experience. To enable creative interactions, we used the *text-davinci-002* model with temperature $T = 0.5$, and provided it with a contextual prompt: "This is a magical game like ping pong, in which the players can change both the ball and their paddles, when the transformed ball object hits the transformed paddle, it changes the ball according to how you'd expect those two objects to interact." Working in the few-shot regime, we provided several examples and then prompted the actual interaction within the game, e.g. "When water collides with fire, it spawns". From this sentence, GPT-3 predicts the most likely following word (in this example, it returns "steam"). See the table for more examples. We then separately used *code-davinci-002* with $T = 0$ to obtain scripts implementing model downloading, replacements and rescalings (possible due to Codex's real world knowledge), and any other requests the user might have. While Web extraction tools have perfected algorithms for harvesting 2D media like images and video, for our game we needed a database with a sizable number of 3D assets. Therefore, we dynamically loaded models at runtime via the Sketchfab API (which contains over 3 million models). When the user voices the command to change objects, the models that contain the spoken object are searched ordered by their like count (determined by Sketchfab users) and then a randomized object with the lowest vertex count gets selected as the model to download. This way, download speed and model quality are being optimized.

For user input we used voice commands via Azure Cognitive Services Speech to Text [16]. The system continually listens for phrases to initialize user input: "Change ball" changes the model of the ball/projectile, "Change Paddle" swaps out a paddle model affixed to the user's hand with an object the user specifies, with the resulting code completion displayed on a panel overhead the user. We use the UltraLeap Stereo IR 170 for tracking players' hands. Photon is used for multiplayer functionality.

Object 1	Object 2	**Output**
salmon	knife	**sushi**
fried egg	time	**rotten egg**
fire	ice	**water**
family	time	**memory**
memory	disaster	**PTSD**
pineapple	banana	**smoothie**
apple	tennis racket	**apple pie**

3 Discussion & Conclusion

When developing games it may be necessary for more elaborate implementations than Codex can provide using improved API calls, wrapping logic into classes/functions, and better kinematic algorithms. This is where human game developers still surpass AI models like Codex. Without additional fine-tuning, Codex often produces sub-optimal game mechanics in many cases.

World mythology often includes shapeshifter figures that transform as part of their interactions with each other and the human world. Pre-digital themed gaming, such as dungeons and dragons, often included more transformative gameplay than has become typical of digital gaming, and the advances shown here might help to address that gap. Furthermore, some of the early hopes for social VR when it was first demonstrated included allowing people to spontaneously create transformations, in the hopes of bringing about speculative future modes of communication, which were sometimes called "post-symbolic" [17, 18].

Ethical considerations

While application of LLMs has the potential to be transformative to gaming, it does come with certain risks. The usual concern of automation displacing creators is not much of a concern in the near future, as these tools are still far away from end-to-end development. Of bigger concern is the potential for users to generate sensitive or harmful content. While Sketchfab has some moderation on the submitted models, and OpenAI filters inputs and outputs on potentially dangerous or harmful content, it is possible for users to circumvent these filters in creative ways. Additionally, the integration of Codex with Unity allows the user to in principle generate arbitrary visual scenes as well generate unsafe code. While there needs to be more research done into how to moderate global scene generation, at least we can automatically check for unsafe code with the Roslyn compiler to avoid obviously dangerous code from being evaluated.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[3] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *arXiv preprint arXiv:2203.07814*, 2022.

[4] Iddo Drori, Sarah Zhang, Reece Shuttleworth, Leonard Tang, Albert Lu, Elizabeth Ke, Kevin Liu, Linda Chen, Sunny Tran, Newman Cheng, et al. A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level. *Proceedings of the National Academy of Sciences*, 119(32):e2123433119, 2022.

[5] Ryan Volum, Sudha Rao, Michael Xu, Gabriel A DesGarennes, Chris Brockett, Benjamin Van Durme, Olivia Deng, Akanksha Malhotra, and Bill Dolan. Craft an iron sword: Dynamically generating interactive game characters by prompting large language models tuned on code. In *The Third Wordplay: When Language Meets Games Workshop*, 2022.

[6] Nick Walton. Ai dungeon, https://play.aidungeon.io/.

[7] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[8] David Holz et al. Midjourney, https://www.midjourney.com/.

[9] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[10] Chenfei Wu, Jian Liang, Xiaowei Hu, Zhe Gan, Jianfeng Wang, Lijuan Wang, Zicheng Liu, Yuejian Fang, and Nan Duan. Nuwa-infinity: Autoregressive over autoregressive generation for infinite visual synthesis. *arXiv preprint arXiv:2207.09814*, 2022.

[11] Nasir Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Text to mesh without 3d supervision using limit subdivision. *arXiv preprint arXiv:2203.13333*, 2022.

[12] Unity Technologies. Unity game engine, https://unity.com/.

[13] Joseph O. Harrison and M Barrett. Computer-aided information systems for gaming. 1964.

[14] Trivial Interactive. Roslyn c - runtime c compiler, https://forum.unity.com/threads/released-roslyn-c-runtime-c-compiler.651505/.

[15] William Welsh. Williamwelsh/openai.net: An unofficial .net wrapper for the openai api to access gpt-3, https://github.com/williamwelsh/openai.net.

[16] Microsoft Azure. Speech to text – audio to text translation: Microsoft azure, https://azure.microsoft.com/en-us/products/cognitive-services/speech-to-text.

[17] Kevin Kelly, Adam Heilbrun, and Barbara Stacks. An interview with jaron lanier. *Whole Earth Review: Saulsalito*, 1989.

[18] Jaron Lanier and Frank Biocca. An insider's view of the future of virtual reality. *Journal of Communication*, 42(4):150–172, 1992.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] There are however further considerations in terms of scaling this to consumer-facing games that go beyond the scope of this paper
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Ethical considerations section
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The main hyperparameter was the temperature of the OpenAI models called
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cited the Unity packages we based this work on
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Prompt to GPT-3 for creative collisions

This is a magical game like ping pong, in which the players can change both the ball and their paddles, when the transformed ball object hits the transformed paddle, it changes the ball according to how you'd expect those two objects to interact.

When a spawned loaf of bread collides with spawned cheese it spawns A sandwich object.

When a spawned pen collides with spawned paper it spawns a notebook object.

When spawned meat collides with a spawned clock it spawns a bacteria object.

When a music note object collides with a cube object it spawns an instrument.

When water object collides with air object it spawns ice.

When a tree collides with a clock, it spawns a dead tree.

When an egg collides with a clock, it spawns a chicken.

When a cube collides with a wheel, it spawns a car.

When an egg collides with a frying pan, it spawns a fried egg.

When a balloon collides with a pin, it spawns a popped balloon.

When a bread collides with a clock, it spawns a moldy bread.

When a caterpillar collides with a clock, it spawns a butterfly.

When water collides with fire, it spawns steam.

When seed collides with water, it spawns a plant.

When egg collides with clock, it spawns