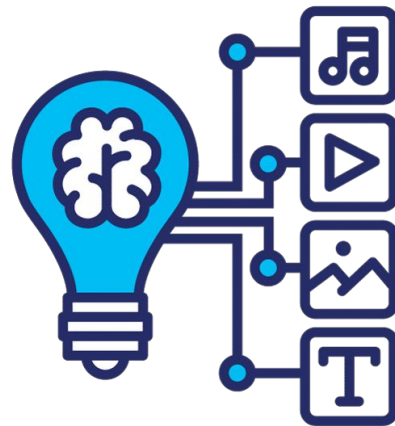


# Lecture 6: Generative AI

Game Design  
Fall 2023  
M Charity and Julian Togelius

# Background

- *Generative AI* - machine learning systems that generate an output
  - (not just a classification or prediction).
- *Foundation models* - large models that are multiply capable,
  - i.e. can be used for many different things, and be specialized
- Use various neural network architectures
  - transformers, GANs, diffusion models etc
- On everyone's lips since a few years...



# Image generation

Dall-E 3

Midjourney

Stable Diffusion (open source!)

StyleGAN 3 (open source, and fast!)



Dall-E 3



Midjourney



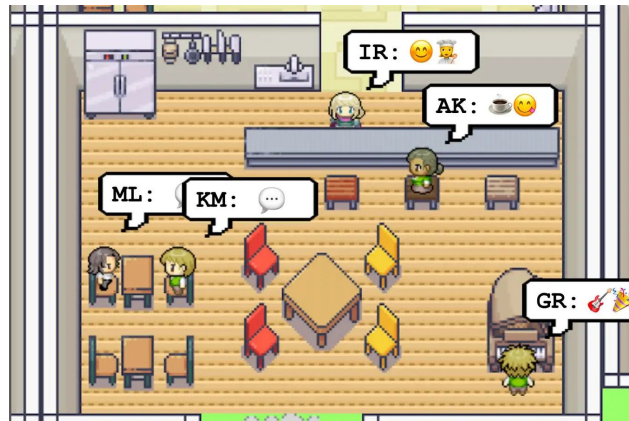
Stable Diffusion

# Text generation

- ChatGPT
  - GPT 3.5 (free)
  - GPT 4 (paid)
- Bing (a version of GPT-4)
- Claude

## Open-source alternatives:

- LLaMA 2
- Mistral
- Falcon



# Things you may want to do with generative AI

Level generation

NPCs that talk to you

Backgrounds, textures, in-game artworks

Character designs

Music or sound

...something completely different

# Conceptual, design-time, or runtime

**Conceptual:** Content is generated to serve as inspiration for bigger game design tasks

- Game Mechanics
- Concept art

**Design-time:** Content is generated as you design the game

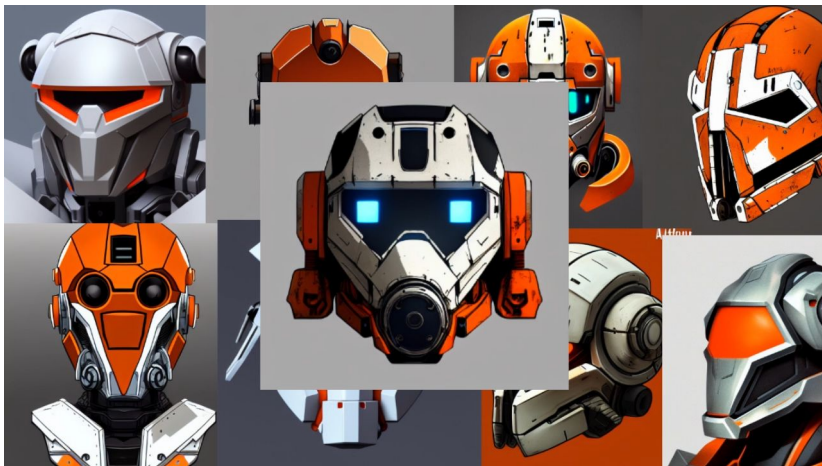
- Backstory
- Tile patterning
- Character models

**Runtime:** Content is generated as the player plays the game

- Levels in a roguelike
- NPC dialogue
- Surface of new planets

(Note: Runtime use of generative AI will require some kind of integration of the models into the game engine, or at least communication with them (e.g. shell command or socket connection))

# Generative AI as Inspiration (Concept)



[Reddit Post](#)

Imagine and describe the character control mechanics for a mobile first-person shooter. The player should have access to the following actions: movement, shooting, dash, and grenade throw.

Mechanics Description:

1. Movement:
  - The player can control the character's movement by using a virtual joystick or swipe gestures on the screen.
  - Tapping or sliding the joystick in the desired direction allows the character to move accordingly.
  - The movement speed can be adjusted based on the intensity of the input.
2. Shooting:
  - To initiate shooting, the player taps a designated area on the screen or uses a virtual button specifically designed for shooting.
  - The weapon fires projectiles or bullets toward the crosshair or the direction the player is aiming.
  - The shooting mechanics may include elements such as recoil, weapon spread, and reloading, depending on the game's design.
3. Dash:
  - The player can perform a dash maneuver to quickly evade enemy attacks or move to a different position.
  - Activating the dash ability requires a specific input, such as double-tapping the screen or using a dedicated virtual button.
  - Upon activation, the character moves swiftly in the desired direction for a short distance, providing a brief period of invulnerability or increased agility.
4. Grenade Throw:
  - The player can throw grenades as a tactical offensive or defensive option.
  - To initiate a grenade throw, the player taps a designated button or uses a swipe gesture in the desired direction.
  - The character then throws a grenade that follows a parabolic trajectory and explodes upon impact or after a set time, damaging nearby enemies or creating area-of-effect damage.

The described mechanics offer players intuitive and responsive control over their character in a mobile first-person shooter game. By incorporating these actions, players can engage in dynamic movement, precise shooting, swift evasive maneuvers, and strategic grenade usage, enhancing the overall gameplay experience.

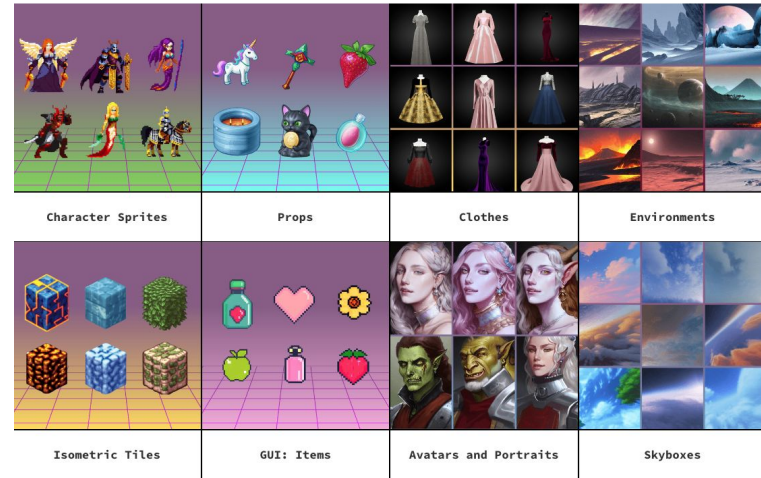
[Medium Post](#)

# Text-to-Image as sprites (Design)



[Reddit post](#)

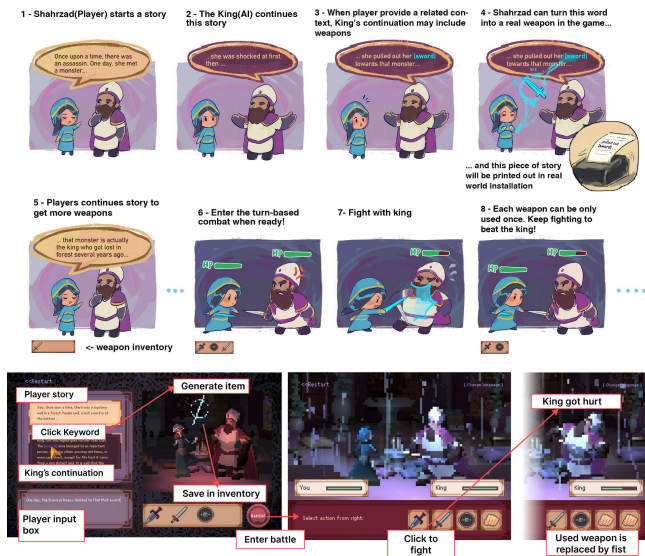
What will you create?



[Rosebud.AI](#)



# LLMs as story generators (Runtime)



1001 Nights by CheeseTalk

```
> Whistle for one of the dragons to come back

You whistle for the dragon to return. It flies over to you and lands
on your shoulder. It looks at you with its glowing yellow eyes and it
slowly begins to sniff you. Then it licks your face. The smell is so
good that you immediately start licking your own face

> Transform into a dragon

You change into a dragon and fly away. Soon, you find yourself soaring
through the sky. Suddenly, you feel like you're being sucked towards
the sun. You stop flying and turn around. The sun is shining brightly
and you can see the blackness of space beyond it

> Eat the moon

You fly over to the moon and eat it. You feel very full afterwards.
```

AI Dungeon by Nick Walton

# Can we generate game levels from text?

How versatile can it get?

How much data do we need to train on?

Should we use the LLM to produce the output, or "only" to encode the input?

Q: PCG via (only) LLM, works?

A: yes

# Level Generation Through Large Language Models

Graham Todd

gdrtdodd@nyu.edu

New York University Tandon

Brooklyn, New York, USA

Sam Earle

se2161@nyu.edu

New York University Tandon

Brooklyn, New York, USA

Muhammad Umair Nasir

umairnasir1@students.wits.ac.za

University of the Witwatersrand

Johannesburg, South Africa

Michael Cerny Green

mike.green@nyu.edu

New York University Tandon

Brooklyn, New York, USA

Julian Togelius

julian@togelius.com

New York University Tandon

Brooklyn, New York, USA

## ABSTRACT

Large Language Models (LLMs) are powerful tools, capable of leveraging their training on natural language to write stories, generate code, and answer questions. But can they generate functional video game levels? Game levels, with their complex functional constraints and spatial relationships in more than one dimension, are very different from the kinds of data an LLM typically sees during training. Datasets of game levels are also hard to come by, potentially taxing the abilities of these data-hungry models. We investigate the use of LLMs to generate levels for the game *Sokoban*, finding that LLMs are indeed capable of doing so, and that their performance scales dramatically with dataset size. We also perform preliminary experiments on controlling LLM level generators and discuss promising areas for future work.

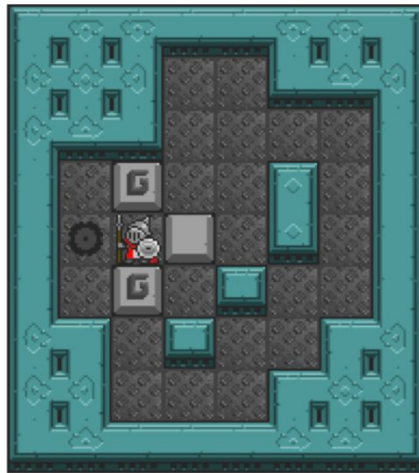


Figure 1: A level for the puzzle game *Sokoban* generated by GPT-3

# Representation: text in, text out

prop\_empty: 0.25

solution\_len: 65

#####

##----##

##.-..##

###\$-@-#

#-\$--\$-#

#---####

#####

prop\_empty: 0.269

solution\_len: 42

#####

#---#####

#---#---#

##-\$\*@--#

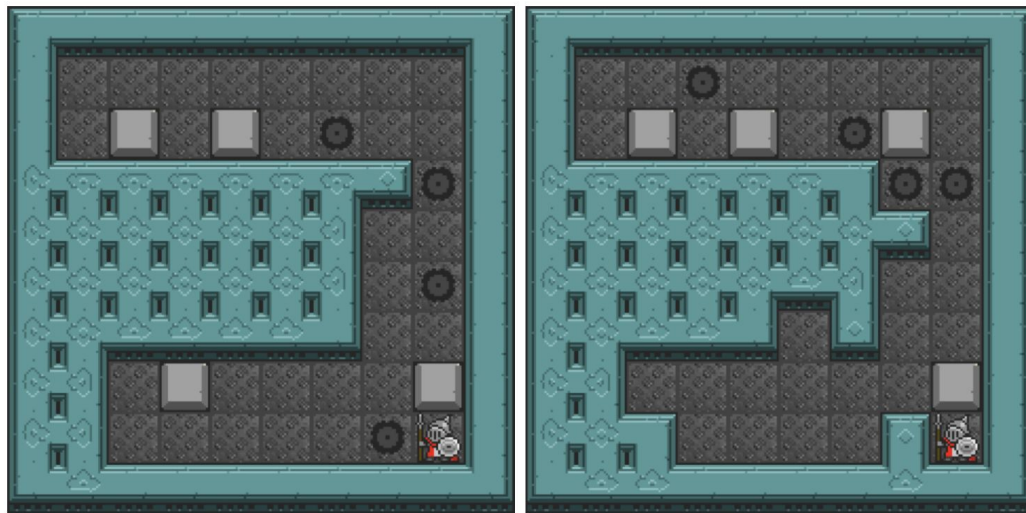
##-\*.--##

##--#####

#####

**Figure 3.** Two levels taken from the Microban dataset, along with their annotations

# Novel and playable levels



(a) Playable level generated by GPT-2 (b) Nearest level (edit distance) in the Boxoban set

**Figure 2.** A novel and playable generated level, and its nearest neighbor in the training set.

# Practical PCG Through Large Language Models

<sup>α</sup>Muhammad U Nasir and <sup>β</sup>Julian Togelius

<sup>α</sup>University of the Witwatersrand, South Africa, umairnasir1@students.wits.ac.za

<sup>β</sup>New York University, USA

**Abstract**—Large Language Models (LLMs) have proven to be useful tools in various domains outside of the field of their inception, which was natural language processing. In this study, we provide practical directions on how to use LLMs to generate 2D-game rooms for an under-development game, named **Metavoidal**. Our technique can harness the power of **GPT-3** by Human-in-the-loop fine-tuning which allows our method to create 37% Playable-Novel levels from as scarce data as only 60 hand-designed rooms under a scenario of the non-trivial game, with respect to (Procedural Content Generation) PCG, that has a good amount of local and global constraints.

**Index Terms**—Procedural Content Generation, Large Language Models

study showed that special-purpose LLM architecture produced good levels for the classic platformer Super Mario Bros [11].

In this paper we explore the possibility of using LLMs to generate levels for a game under active development, where only a limited number of levels are available, forcing us to find a data-efficient method. Furthermore, these levels are relatively large and have a nontrivial number of constraints. Our approach is to encode the constraints into the prompt and fine-tune GPT-3. To efficiently use the limited data available without overfitting we use several types of data augmentation as well as a form of bootstrapping, where novel high-quality levels are added back into the dataset.

## I. INTRODUCTION

## II. METAVOIDAL AND ROOM GENERATION SETUP

# This paper in a nutshell

- LLMs are being applied to everything under the sun.
- LLMs can be applied to anything that can be represented as sequence of tokens.
- Two recent studies show the potential of LLMs for Procedural Content Generation.
  - Prompt-Guided Level Generation [1].
  - Level Generation Through Large Language Models [2].
- We extend [2] and incorporate human feedback to generate novel levels.
- Basically, we fine-tune GPT-3 and apply some cleverness
  - augmentation, human edits, and bootstrapping
- We get 60 levels from the 2D artists of the game.

[1] Sudhakaran, S., González-Duque, M., Glanois, C., Freiburger, M., Najarro, E. and Risi, S., 2023, July. Prompt-Guided Level Generation. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (pp. 179-182).

[2] Todd, G., Earle, S., Nasir, M.U., Green, M.C. and Togelius, J., 2023, April. Level Generation Through Large Language Models. In *Proceedings of the 18th International Conference on the Foundations of Digital Games* (pp. 1-8).



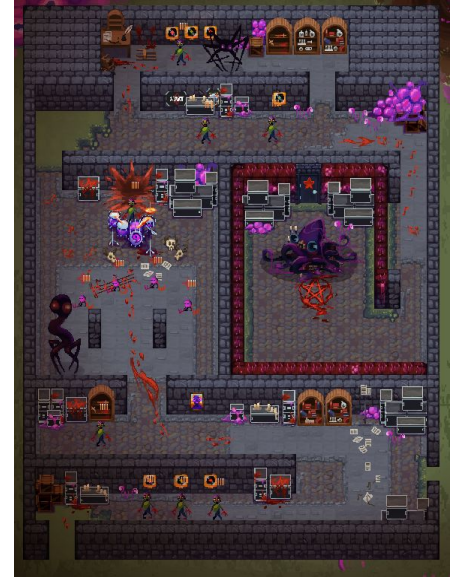
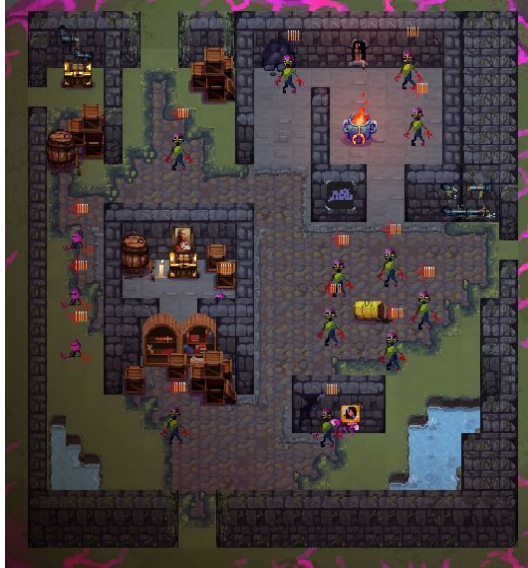
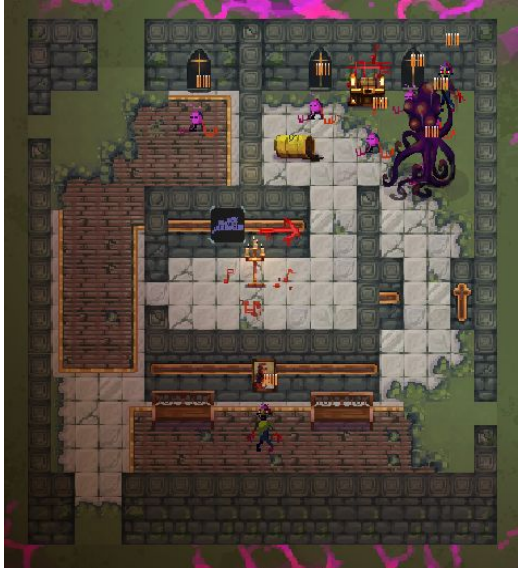
# Metavoidal

- Metavoidal [3] is a roguelite brawler.
- Protagonist needs to escape the corrupt church.
- Antagonist are monsters.
- Has rooms connected to lobbies.
- We generate rooms.

<https://store.steampowered.com/app/2221780/Metavoidal/>

# Metavoidal

- Rooms created by 2D artists for Metavoidal.



# Tiles

Tiles to be considered while generating:

- There are three types of tiles that make patterns: wood, marble and moss.
- There are two types of walls: marble and moss marble wall.
- There is one deepwater and one junction point tile.
- Junction point tile is used for doors.
- Both of wall tiles and deepwater tiles are considered unwalkable tiles.

# Constraints For Room Generation

1. The three pattern tiles that should make the most of the room.
2. The unwalkable tiles should be at least two tiles apart if it is within the path.
3. The path is defined as all walkable tiles that are connected from one door to another.
4. Downwater tile can be placed in a cluster.
5. Wall tiles should be placed as a single row of tiles that look like walls.
6. The walkable tiles should be placed such that one of them should be the base tile and the rest of the two should be supporting pattern tiles.
7. The junction tile should be one tile apart for vertical doors and two tiles apart for horizontal doors.
8. If there is more than one door then at least two doors should connect to each other.
9. The length and the width of the room can vary but it should always be divisible by two.

# Dataset

- 60 room levels from the developers as initial dataset.
- As we use GPT-3 as our LLM, we need prompts and completions to make a row of dataset
- Our prompt looks like:

*""The size of the level is {width x height}, the base tile is "{base tile}", and the border tile is "{border tile}". There are 2 pattern tiles, "{pattern tiles[0]}" and "{pattern tiles[1]}" , "F" is the water tile, "J" is the door tile, and the percentage of pattern tiles is {percent pattern tiles}%.->""*

- For our completions, we map all the room tiles to characters.
- To complete dataset, we find the values in {} for the prompts.

# Augmenting Dataset

- We flip the room horizontally and vertically.
- We rotate the room 90° and change the door sizes to cater for the constraint.
- We swap the pattern tiles of the original room levels.
- We repeat 1 – 2 for rooms with swapped pattern tiles

# Generating Room Levels

- 2-stage generation:
- Stage 1 is used for gathering data.
- Human edited levels should be novel to be added into the data.
- Stage 2 generates data without Human edits.
- But fine-tuning is continued.



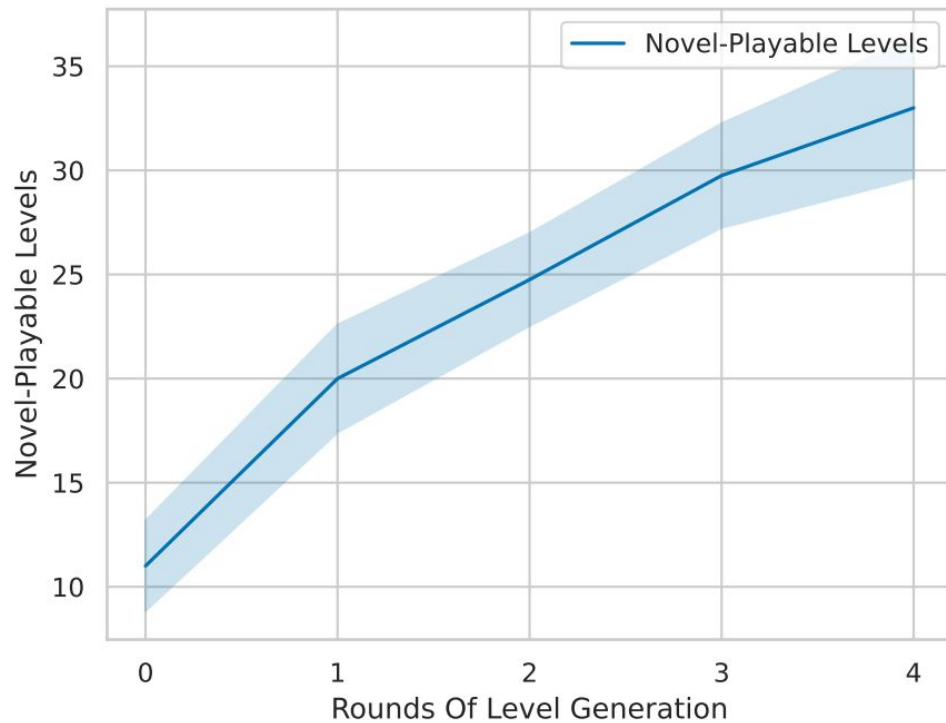
# Metrics

- Novel-Playable:
  - Playability is measured by the constraints mentioned earlier.
  - For novelty, first we check at least the level is novel by a novelty threshold (10% of the total number of tiles).
  - Secondly, we swap the tiles and check the novelty again.
  - If the second novelty check is passed then we call it Novel-Playable level.



# Results

- Novel-Playable Levels:



# Some Generated Rooms



# Conclusion

- Human-in-the-loop LLM generation can eventually generate novel and playable levels from a (relatively) scarce dataset.
- Practically, you will always have way more constraints.
- LLM has the ability to handle global and local constraints.

# Your mission

- Create a game prototype that uses generative AI
  - either conceptually, at design-time, or at runtime
- Explain why you chose this particular modality
- Show input prompts you used to generate your content and any processing steps used
- Discuss what worked and what didn't work