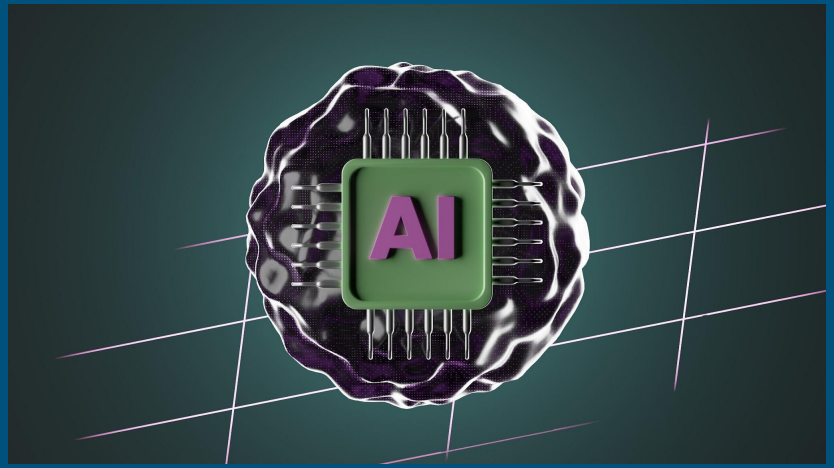
Chat to Game Generator Interface

By Christopher Figueroa

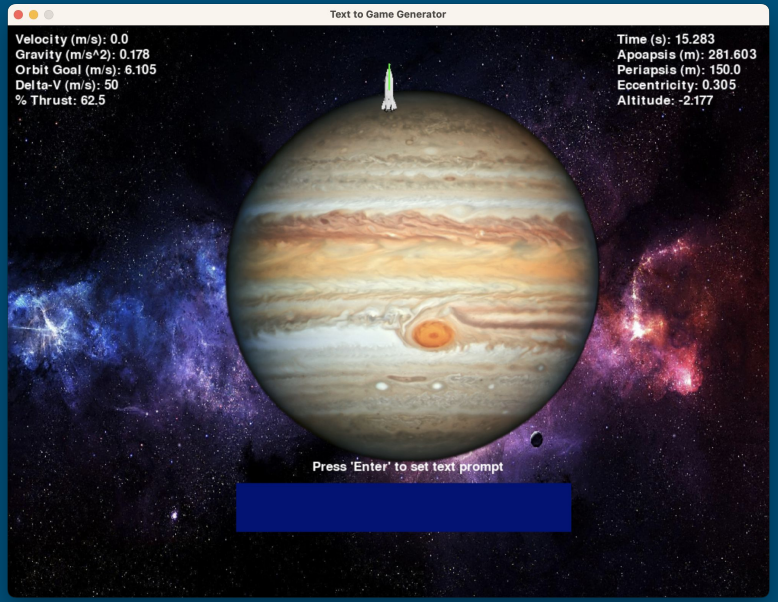
Abstract

- Inspired by ChatGPT and Rix AI
- Generates a game using a text interface
- Goals of game generator:
  - Real-time video game editing
  - Direct compilation of AI code
  - Dynamic editing of existing code
- Uses Python for game generation
- Well suited for 2D game generation



Motivation

- Compare the 2D game generation methods
- Three generation methods:
  - Hard-coded instructions
  - OpenAI interface
  - Dynamic code editing
- Demonstrate feasibility of a prompt-based 2D game engine
- Keep number of libraries to a minimum



Hard-coded planet orbiter game

Related Research - ChatGPT

- OpenAI, founded in 2015 by Elon Musk and Sam Altman.
- ChatGPT generates natural language text, pre-training on large text an unsupervised manner.
- Pre-training involves predicting the next word in a sequence, allowing the model to learn language patterns, syntax, grammar, and semantics.
- Fine-tuning on specific tasks is done with smaller labeled datasets, updating the model for tasks like text classification or question-answering.
- GPT-1 had 117 million parameters and demonstrated impressive results in various natural language processing tasks.
- GPT-4 is a large multimodal language model accepting both image and text inputs, generating text outputs.

Related Research - Alpaca

- Developed by Stanford University researchers, Alpaca is an open-source language model based on Meta's Llama.
- Not available for commercial use, but small businesses can use it for chatbot development.
- Tested against ChatGPT in tasks like email creation, social media, and productivity tools.
- Alpaca won 90 times, while ChatGPT won 89 times.
- Applicable in real-world scenarios, aiding researchers in ethical AI and cybersecurity (e.g., scam detection).
- Alpaca allows for training sophisticated language models without expensive hardware concerns.

Python Libraries Used

- 2D Python game engine
- Base game libraries:
  - Pygame
  - Math
  - Numpy
- ChatGPT libraries:
  - Sys (already included)
  - OpenAI
- Compiled in command prompt
- Python 3.11.5

Three Game Generation Methods

- Hard-coded instructions:
  - Uses the Planet Orbiter game as default base game
  - Text-based user input to edit existing planets and newly added planets
  - Procedurally generated planets and background
- OpenAI interface:
  - Takes in input from player to send prompts to ChatGPT
  - Uses OpenAI library to run the output of ChatGPT in Python
- Dynamic code editing:
  - Uses the game Pong as the default base game
  - Stores the existing game code as a string
  - Functions and if-then statements search for variables to edit in the string code

Hard-Coded Instructions

- Runs 2D physics Planet Orbiter game using Pygame
- Composed of modular programs:
  - Images.py: handles planet and background images
  - Orbits.py: traces orbit of spacecraft
  - Planet.py: represents each generated planet object
  - TextGame.py: the main program file
- Main program applies user input to enact changes to game

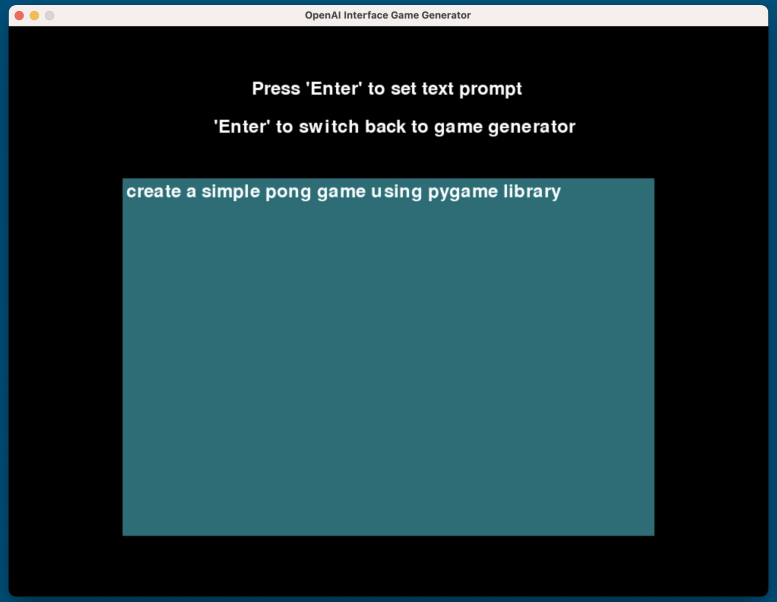
Mars moved 300 pixels to the left

Functions in Main Program

- Main program sections:
  - Planet and spacecraft variables
  - While running loop
  - Spacecraft running variables
  - Physics calculations
  - Hard-coded planet alteration instructions
- Planet object 'Planet.py' file instances are saved into a list of planets
- The spacecraft's orbital parameters are based off first planet on the list
- The list of hard-coded instructions applied to every planet
  - Move planet up, down, right, or left
  - Alter radius of planet
  - Alter mass of planet
  - Planets can be referenced as planet 1, 2, 3… etc

OpenAI Interface

- Uses OpenAI library to send prompts to ChatGPT
- Uses API key to access ChatGPT
- Functions in program:
  - Return_code variable
  - TextChatCode()
  - Is_valid_code()
  - Text box user input
- Accompanying programs:
  - Invalid code program ‘Invalid.py’
  - Pong test game ‘PongGame.py’



Prompts ChatGPT to create a simple Pong game

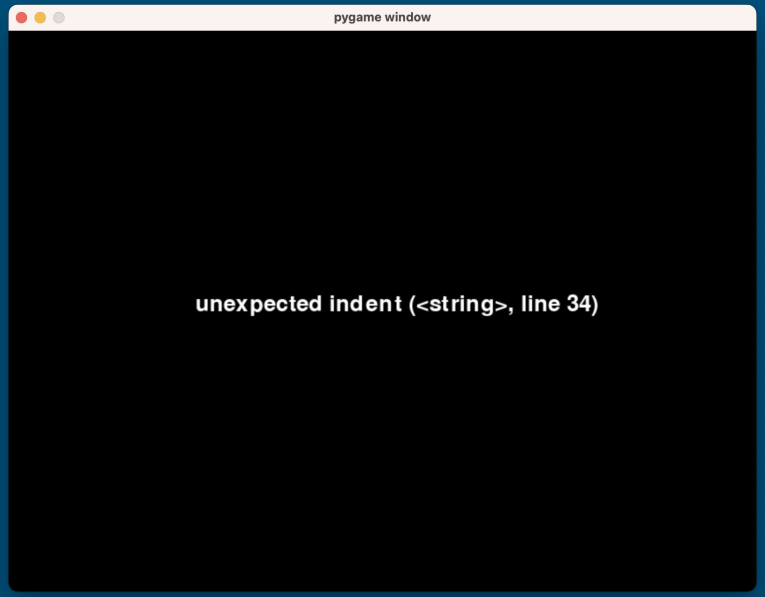
Return Code Variable

```
# allows for user to return to text game when return button is clicked
return_code = '''
    # for key clicks where one by one key pressing is preferred
    for event in pygame.event.get():
        # if the user quits the game
        if event.type == pygame.QUIT:
            running = False
        if event.type == pygame.KEYDOWN:
            # checks to see if the enter key has been pressed
            if event.key == pygame.K_RETURN:
                # reads the contents of TextInput.py
                file_path = "TextInput.py"
                old_code = ''
                with open(file_path, "r") as file:
                    old_code = file.read()
                # returns to text box game
                exec(old_code)
'''
```

- Return code initialized as multi-lined string
- Allows generated game to return to TextInput.py file with ‘Enter’ key
- Appended to game generated game string ‘response’
- Exec() command used to run the modified game

```
# adds return code to end of current game code
response = response + '\n' + return_code
# warning, this code does not account for programs that end with 'pygame.quit()'
```

Invalid Code Program



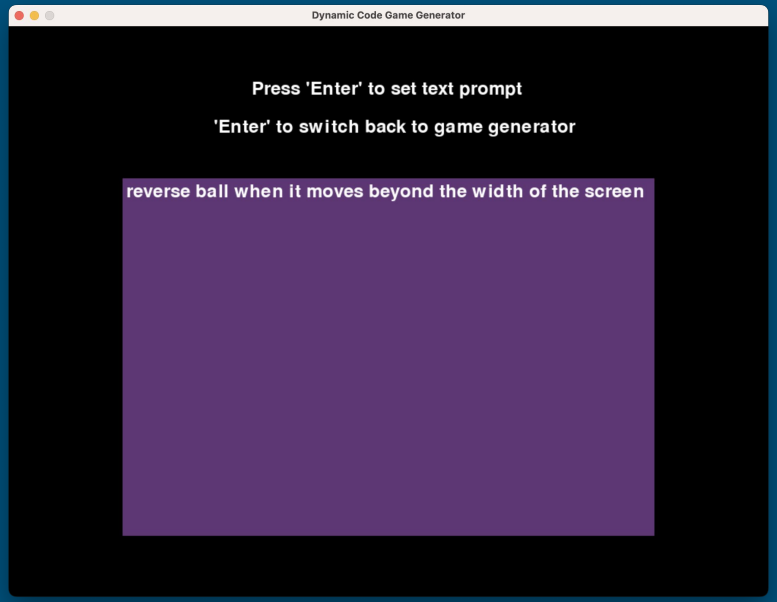
- Invalid class code contained in ‘Invalid.py’ file
- Functions in class:
  - Set_error(): sets the error message string
  - Run(): runs a Pygame instance that displays the error message string
- Runs when the generated game has invalid code

```
# invalid game screen
from Invalid import InvalidCode
invalid = InvalidCode()

# saves invalid error message
error_str = ''
```

Dynamic Code Editing

- Loads existing Pong game as a mutable string
- Functions in program:
  - Return_code variable
  - Old_code (Pong) variable
  - Is_valid_code()
  - Text box user input
  - Game editing instructions
- Accompanying programs:
  - Invalid code program ‘Invalid.py’
  - Pong test game ‘PongGame.py’



Adding a condition to bounce ball off right wall

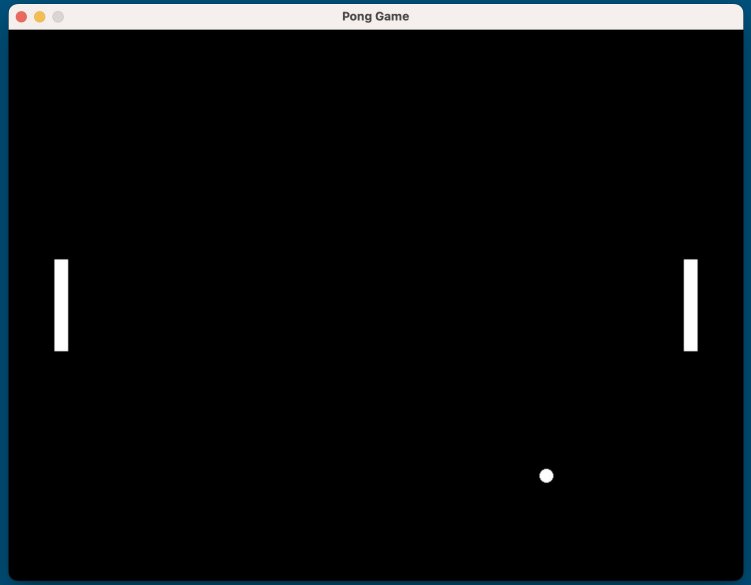
Instructions in Dynamic Code

- The return code variable is added to the base Pong game before modifications
- The list of hard-coded instructions
  - Move racket up, down, right, or left
  - Change color of background, racket, or ball
  - Alter velocity of ball or rackets
  - Add variables or modify existing variables
- The user input is saved as a string variable called ‘user_text’
- One for-loop searches for the game object (racket, ball, etc) that will be modified
- Every instruction is assigned a pre-fabricated code snippet
- Another loop applies the code snippets to the appropriate lines

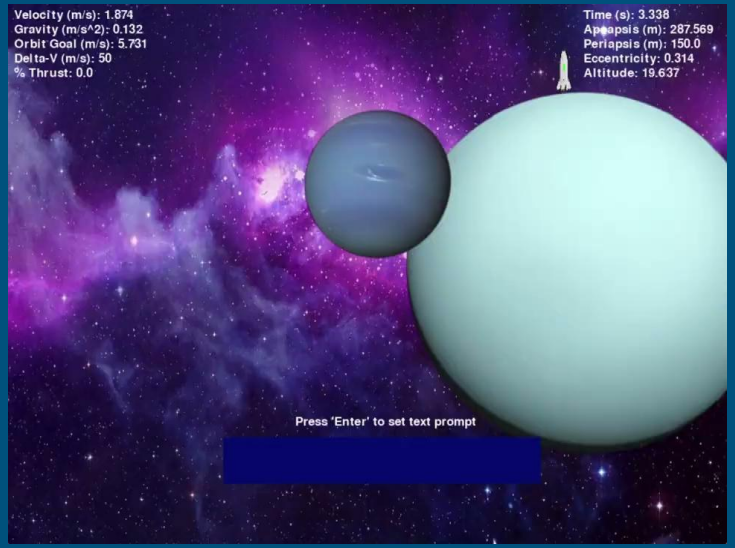
Pong Base Game

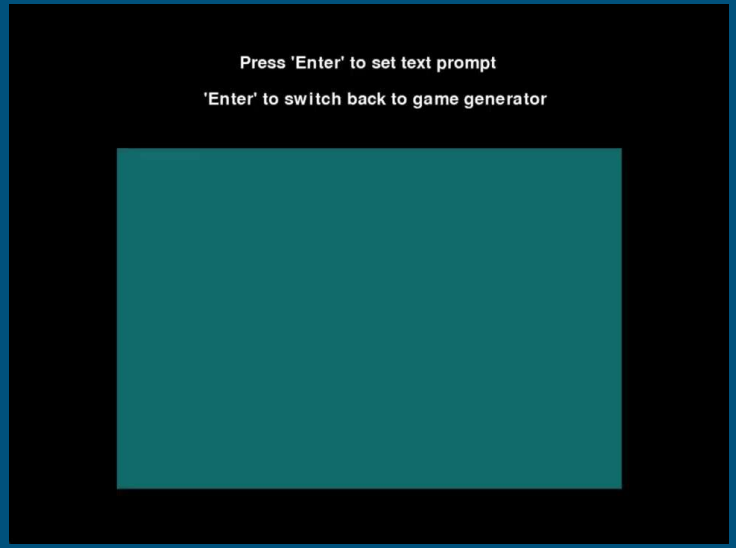
- Default game for dynamic code generation engine
- Test game for OpenAI interface generation
- Contained in ‘PongGame.py’ file
- Loaded in as string and then modified as needed

```python
# reads the contents of PongGame.py
file_path = "PongGame.py"
old_code = ''
with open(file_path, "r") as file:
    old_code = file.read()
```

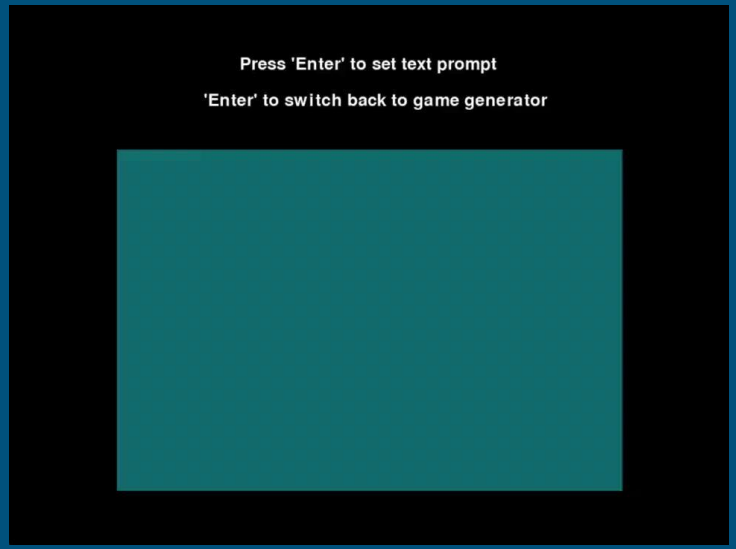


Hard-Coded and Open-AI Demos





Dynamic and Invalid Game Demos



Current Progress

- The hard-coded game generator works reliably
- The OpenAI game generator works for the most part
- Dynamic code progress
  - Edits code based on keywords
  - Struggled to make large scale edits
- Room for improvement
  - Alter dynamic code generation to allow for insertion of new functions
  - Find alternate API system that does not require payment

CS591 — -bash — 88×27

```
    response = TextChatCode(prompt)
               ^^^^^^^^^^^^^^^^^^^^
  File "/Users/Chris/Desktop/desktop folders/documents/CS591/TextInput.py", line 11, in
TextChatCode
    response = openai.Completion.create(
               ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/Chris/anaconda3/lib/python3.11/site-packages/openai/api_resources/complet
ion.py", line 25, in create
    return super().create(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/Chris/anaconda3/lib/python3.11/site-packages/openai/api_resources/abstrac
t/engine_api_resource.py", line 153, in create
    response, _, api_key = requestor.request(
                           ^^^^^^^^^^^^^^^^^^
  File "/Users/Chris/anaconda3/lib/python3.11/site-packages/openai/api_requestor.py", li
ne 298, in request
    resp, got_stream = self._interpret_response(result, stream)
                       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/Chris/anaconda3/lib/python3.11/site-packages/openai/api_requestor.py", li
ne 700, in _interpret_response
    self._interpret_response_line(
  File "/Users/Chris/anaconda3/lib/python3.11/site-packages/openai/api_requestor.py", li
ne 765, in _interpret_response_line
    raise self.handle_error_response(
openai.error.RateLimitError: You exceeded your current quota, please check your plan and
 billing details.
(base) MacBook-Pro-3:CS591 Chris$
```

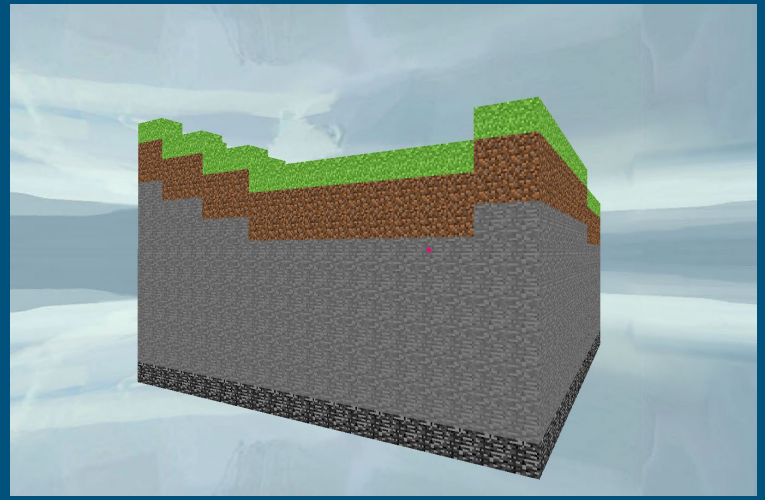
ChatGPT API use limit reached

Conclusion

- Three generation methods:
  - Hard-coded instructions
  - OpenAI interface
  - Dynamic code editing
- The hard-coded game generation program is the least versatile method
- The OpenAI interface generation is the most versatile but requires payment
- Pros/Cons of dynamic code editing:
  - Allows for editing of existing code
  - More versatile than hard-coded instructions
  - Can be used without any paid API plans
  - Downside, does not use an openAI to generate games

Future Work

- Plans to expand to 3D game ChatGPT generation
- Use a combination of an API interface and dynamic code
- Use Python for game engine
- 3D Python game libraries
  - Ursina
  - PerlinNoise
  - OpenAI



Minecraft clone made using Python's Ursina library

Sources

- [1] Orbital physics equations: https://oer.pressbooks.pub/lynnanegeorge/chapter/chapter-1/
- [2] Apoapsis and periapsis: https://orbital-mechanics.space/the-orbit-equation/orbital-nomenclature.html?utm_source=hashnode&utm_medium=hashnode+rix&utm_campaign=rix_chatbot_answer
- [3] Input textbox: https://www.geeksforgeeks.org/how-to-create-a-text-input-box-with-pygame/
- [4] ChatGPT API: https://blog.enterprisedna.co/how-to-use-chatgpt-for-python/
- [5] Alpaca AI Chat program: https://www.listendata.com/2023/03/open-source-chatgpt-models-step-by-step.html
- [6] ChatGPT research article: https://www.sciencedirect.com/science/article/pii/S266734522300024X