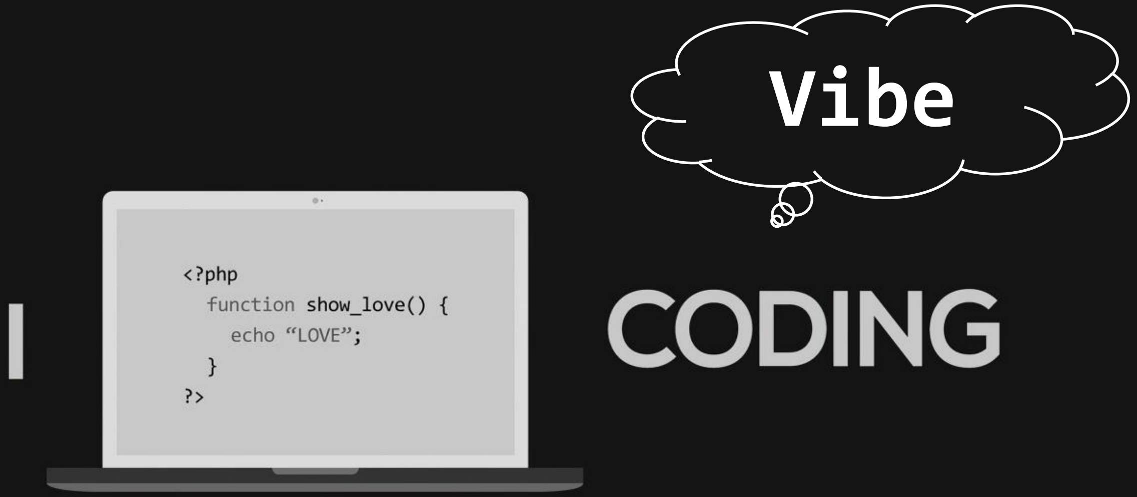
Vibe Coding with AI
Building a Simple Game from Prompts

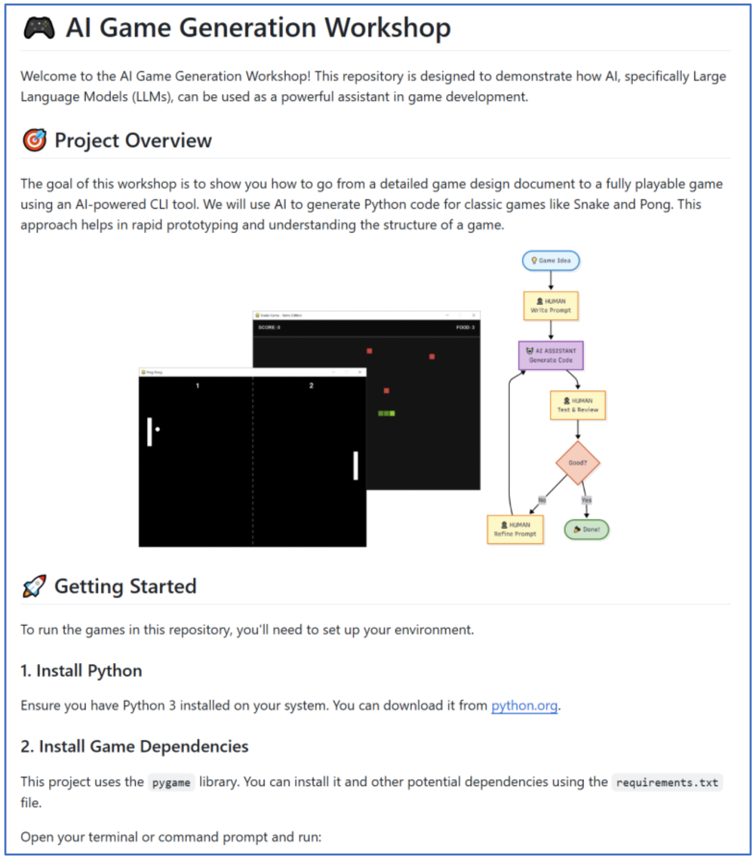


```
<?php
  function show_love() {
    echo "LOVE";
  }
?>
```

Department of Math & Computer Science
Dr. Tianyu Wang

Agenda

- What is Vibe Coding
- The Art of the Prompt
- Design a Game: Classic Snake
- Vibe Coding: Live Demo!



🎮 AI Game Generation Workshop

Welcome to the AI Game Generation Workshop! This repository is designed to demonstrate how AI, specifically Large Language Models (LLMs), can be used as a powerful assistant in game development.

🎯 Project Overview

The goal of this workshop is to show you how to go from a detailed game design document to a fully playable game using an AI-powered CLI tool. We will use AI to generate Python code for classic games like Snake and Pong. This approach helps in rapid prototyping and understanding the structure of a game.

🚀 Getting Started

To run the games in this repository, you'll need to set up your environment.

1. Install Python

Ensure you have Python 3 installed on your system. You can download it from python.org.

2. Install Game Dependencies

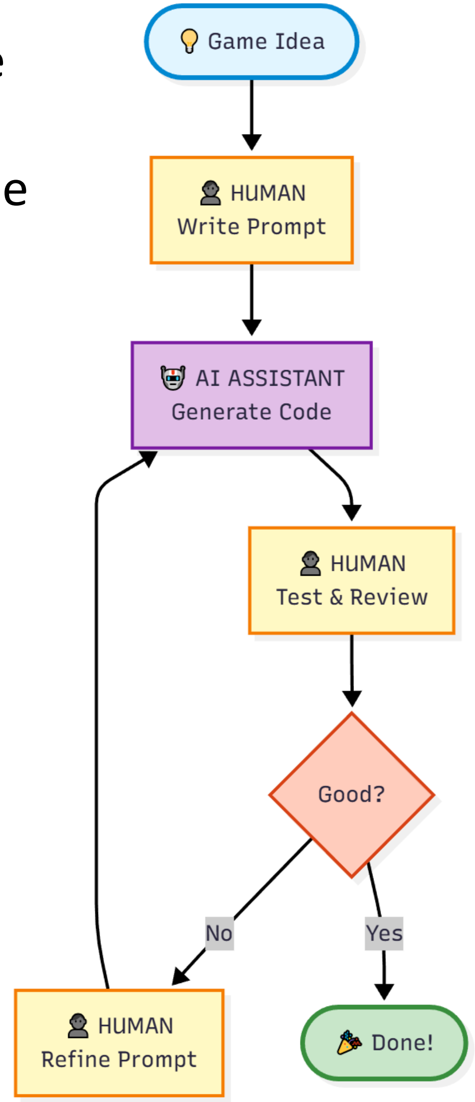
This project uses the `pygame` library. You can install it and other potential dependencies using the `requirements.txt` file.

Open your terminal or command prompt and run:

Workshop GitHub Repository:
https://github.com/tisage/game-llm/

What is "Vibe Coding"

- **Vibe Coding**: an AI-assisted software development practice where developers describe desired functionality in natural language, and an AI generates, refines, and debugs the code based on those prompts

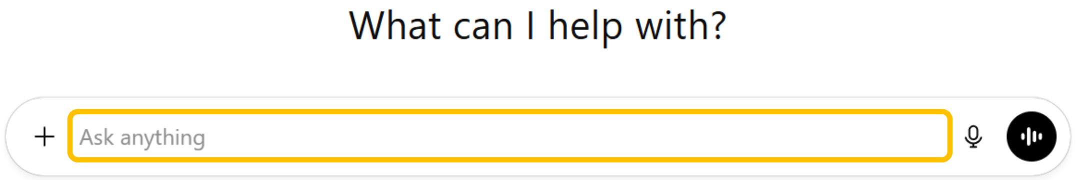


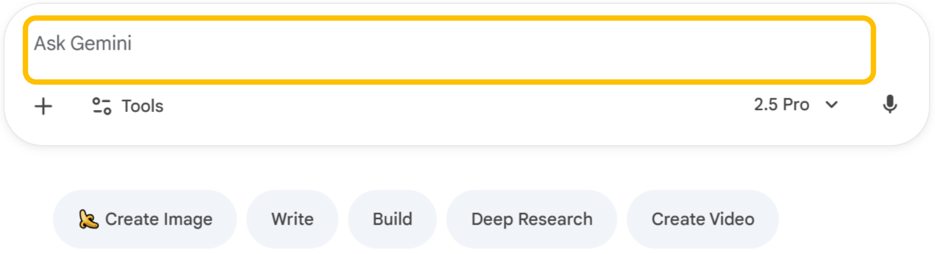
The Art of the Prompt

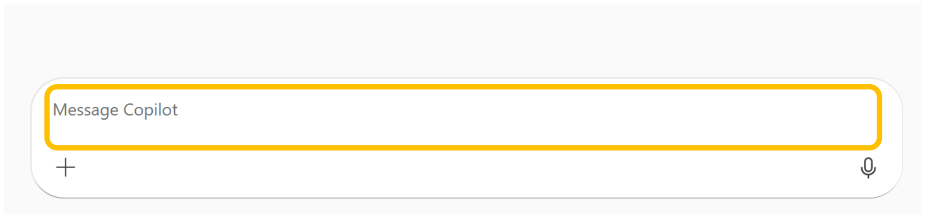
Prompt: A prompt is a user's input, such as a question, instruction, or command, given to an AI model to guide it in generating a specific response

Good prompts are:

- Clear
- Specific
- Structured

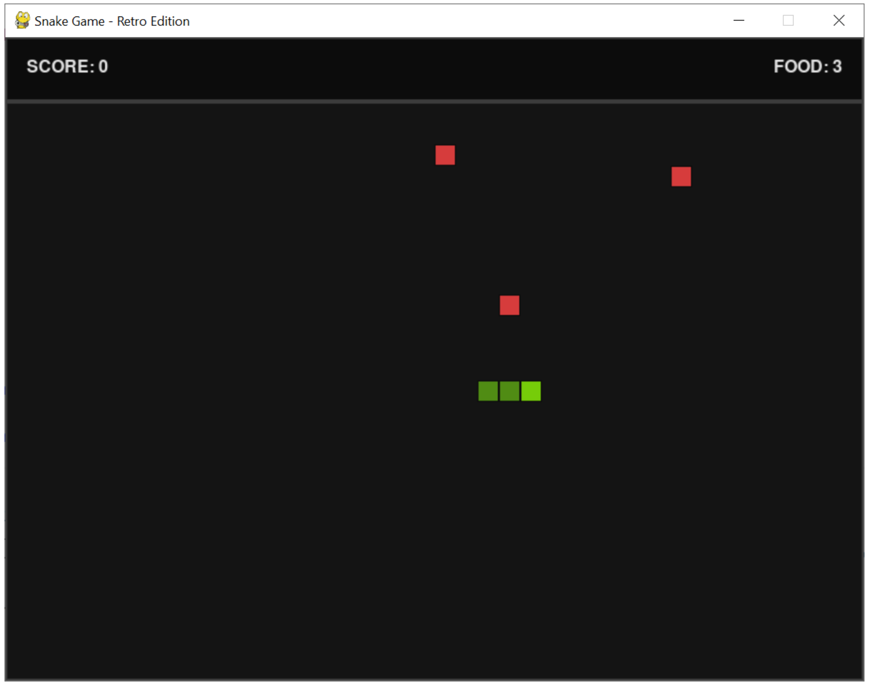






Design a Game: Classic Snake

- Snake Game: Snake is a genre of action video games where the player maneuvers the end of a growing line, often themed as a snake. The player must keep the snake from colliding with both other obstacles and itself, which gets harder as the snake lengthens.



Design a Game: Classic Snake

$$\begin{tikzcd}[row sep=small]
\text{Objective} \arrow[d] \\
\text{Gameplay} \arrow[d] \\
\text{UI} \arrow[d] \\
\text{Implementation} \arrow[d] \\
\text{Delivery}
\end{tikzcd}$$

Prompt File Content

- Purpose
- What/How to Play
- User Interface
- Technical Implementation
- Expected Delivery

`game_prompt.md`

Step 1: The Objective

Our Objective:

- High-level goal
- Sentence summary of the project

```
## 1. Objective
Create a complete, single-file Python application for a classic Snake game using the
Pygame library. The game should be easy to run and control, featuring a clean and
visually appealing interface.
```

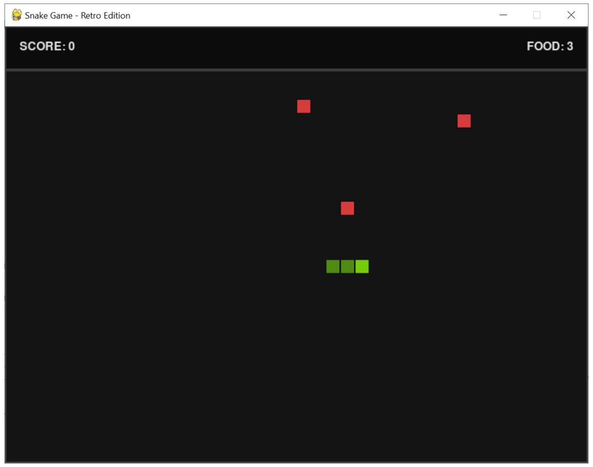
Step 2: Gameplay Mechanics

- **Game Board**: The play area. (e.g., 800x600 window, 20x20 grid)
- **The Player (Snake)**: How does the player move and interact? (e.g., Starts in center, controlled by WASD, can't reverse).
- **The Goal (Food)**: What does the player try to achieve? (e.g., Randomly appearing food, snake grows when it eats).
- **Rules (Scoring & Game Over)**: How do you win or lose? (e.g., +10 points per food, game ends on collision).
- **Game Flow**: How does the game state change? (e.g., Restart with 'R' key).

```
## 2. Core Gameplay Mechanics
 **Game Board:**
    *   The game should be played on a grid-based window.
    *   Window dimensions: 800x600 pixels.
    *   Grid size: 20x20 pixels.
 **Snake:**
    *   The snake starts in the center of the screen, moving horizontally.
    *   The snake is composed of square segments, fitting the grid.
    *   The snake's head should be a distinct color from its body.
    *   The snake continuously moves in its current direction.
    *   The player can change the snake's direction using WASD keys.
…
```

Step 3: GUI Design

- **User Interface (UI)**: What information does the player need? (e.g., Display the score in the top-left corner).
- **Graphics & Art Style**: Define the aesthetic. Simple and clean is a great starting point. (e.g., Green snake, red food, black background).
- **Sound**: Don't forget audio! For our example, we'll keep it simple. (e.g., No sound effects).



```
## 3. GUI
**Graphics:**
  *   Use simple, clean, 2D graphics.
  *   The snake should be green, with a brighter green for the head.
  *   Food items should be visually distinct, with a 3D-like effect (e.g., a
circle with a shadow and highlight).
  *   The background should be black.

**User Interface (UI):**
  *   Display the current score in the top-left corner of the screen.
  *   Display the number of food items currently on the screen below the score.
```

Step 4: The Implementation Plan

- **Language and Library**: Be explicit. (e.g., Use Python 3 and the **pygame** library).
- **Code Structure**: This is key for good software engineering. We can ask the AI to follow best practices.
  - Use a class to organize the code.
  - Use constants for game parameters.
  - Use separate methods for input, updates, and drawing.

```
## 4. Code Structure and Implementation

*   **Language and Library:**
  *   Use Python 3.
  *   Use the `pygame` library.

*   **Structure:**
  *   Organize the code within a `SnakeGame` class to encapsulate all game-related logic and data.
  *   Use constants for key game parameters like window dimensions, grid size, colors, etc.
  *   Create separate methods for handling user input, updating the game state, and rendering the game.
  *   Include a main game loop that controls the flow of the game.
…
```

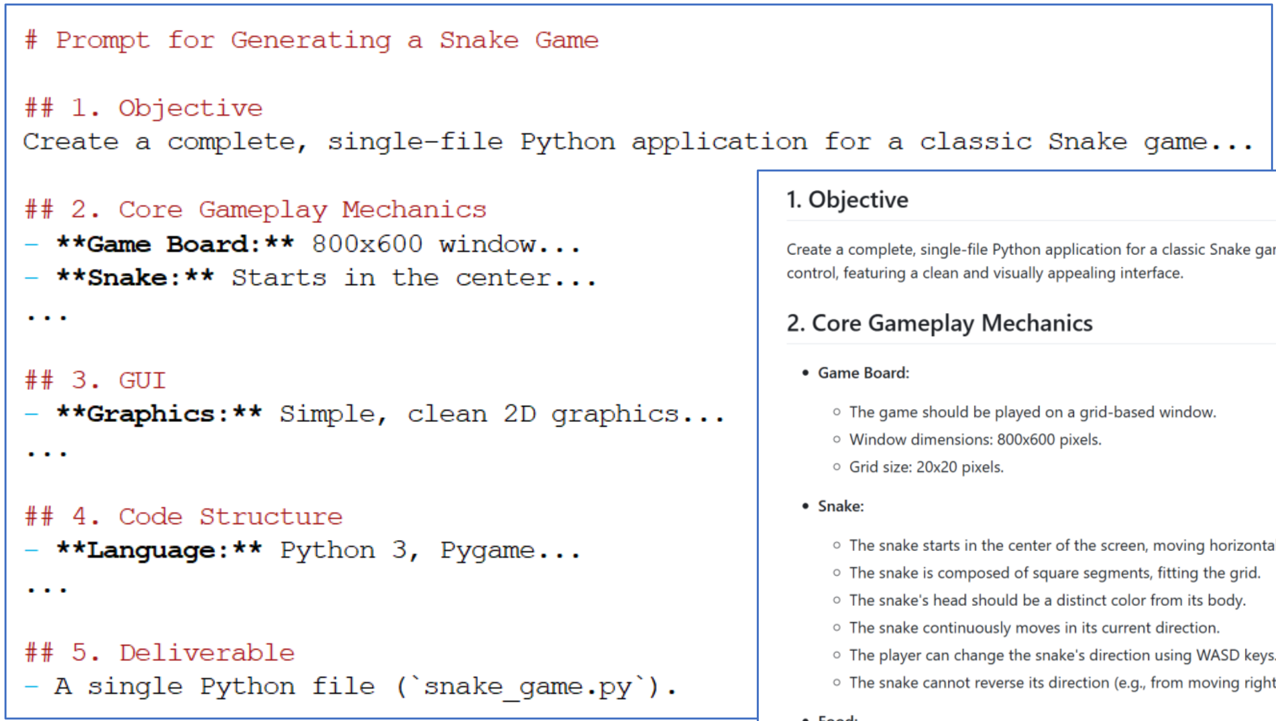
Step 5: The Deliverable

- For a simple game, a single file is often best.
- Our Deliverable:

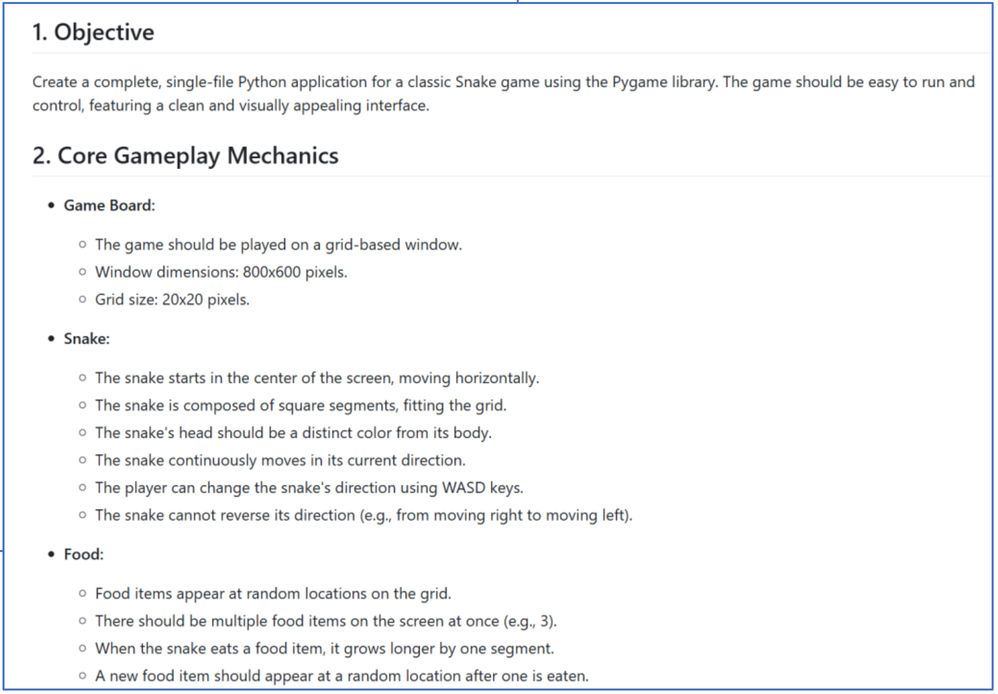
```
## 5. Deliverable

A single Python file (`snake_game.py`) containing the complete, runnable game.
```

The Final Prompt

- Final Prompt File
- (`prompts/game_design_prompt_snake.md`)



```
# Prompt for Generating a Snake Game

## 1. Objective
Create a complete, single-file Python application for a classic Snake game...

## 2. Core Gameplay Mechanics
- **Game Board:** 800x600 window...
- **Snake:** Starts in the center...
...

## 3. GUI
- **Graphics:** Simple, clean 2D graphics...
...

## 4. Code Structure
- **Language:** Python 3, Pygame...
...

## 5. Deliverable
- A single Python file (`snake_game.py`).
```



1. Objective

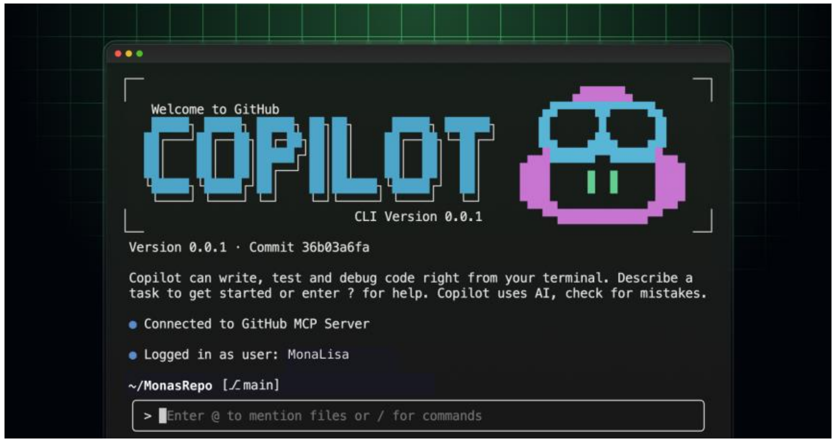
Create a complete, single-file Python application for a classic Snake game using the Pygame library. The game should be easy to run and control, featuring a clean and visually appealing interface.

2. Core Gameplay Mechanics

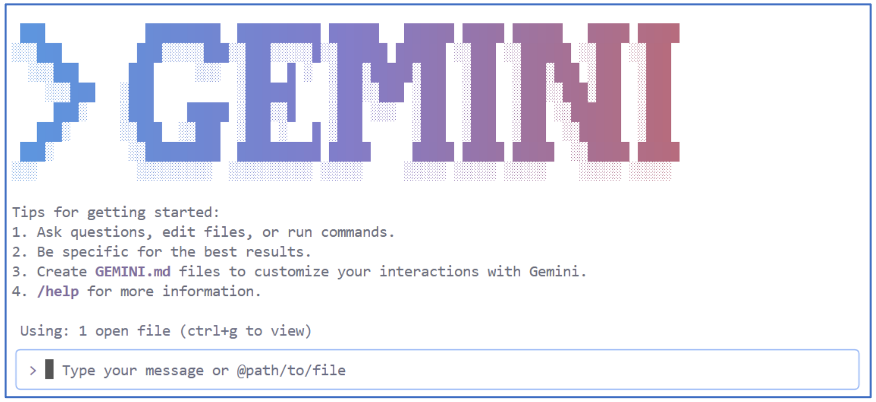
- **Game Board:**
  - The game should be played on a grid-based window.
  - Window dimensions: 800x600 pixels.
  - Grid size: 20x20 pixels.
- **Snake:**
  - The snake starts in the center of the screen, moving horizontally.
  - The snake is composed of square segments, fitting the grid.
  - The snake's head should be a distinct color from its body.
  - The snake continuously moves in its current direction.
  - The player can change the snake's direction using WASD keys.
  - The snake cannot reverse its direction (e.g., from moving right to moving left).
- **Food:**
  - Food items appear at random locations on the grid.
  - There should be multiple food items on the screen at once (e.g., 3).
  - When the snake eats a food item, it grows longer by one segment.
  - A new food item should appear at a random location after one is eaten.

Vibe Coding: Live Demo!

- Recommended AI CLI Tools for This Workshop:
  - **GitHub Copilot CLI** - Free for students with GitHub Education
  - **Gemini CLI** - Free tier available
- Both tools work great for game prototyping. Choose whichever you prefer!



```
Welcome to GitHub
COPILOT
CLI Version 0.0.1
Version 0.0.1 · Commit 36b03a6fa

Copilot can write, test and debug code right from your terminal. Describe a
task to get started or enter ? for help. Copilot uses AI, check for mistakes.

● Connected to GitHub MCP Server

● Logged in as user: MonaLisa

~/MonasRepo [⎇main]

> Enter @ to mention files or / for commands
```



```
> GEMINI

Tips for getting started:
1. Ask questions, edit files, or run commands.
2. Be specific for the best results.
3. Create GEMINI.md files to customize your interactions with Gemini.
4. /help for more information.

 Using: 1 open file (ctrl+g to view)

> Type your message or @path/to/file
```

More Game Design Prompt Templates

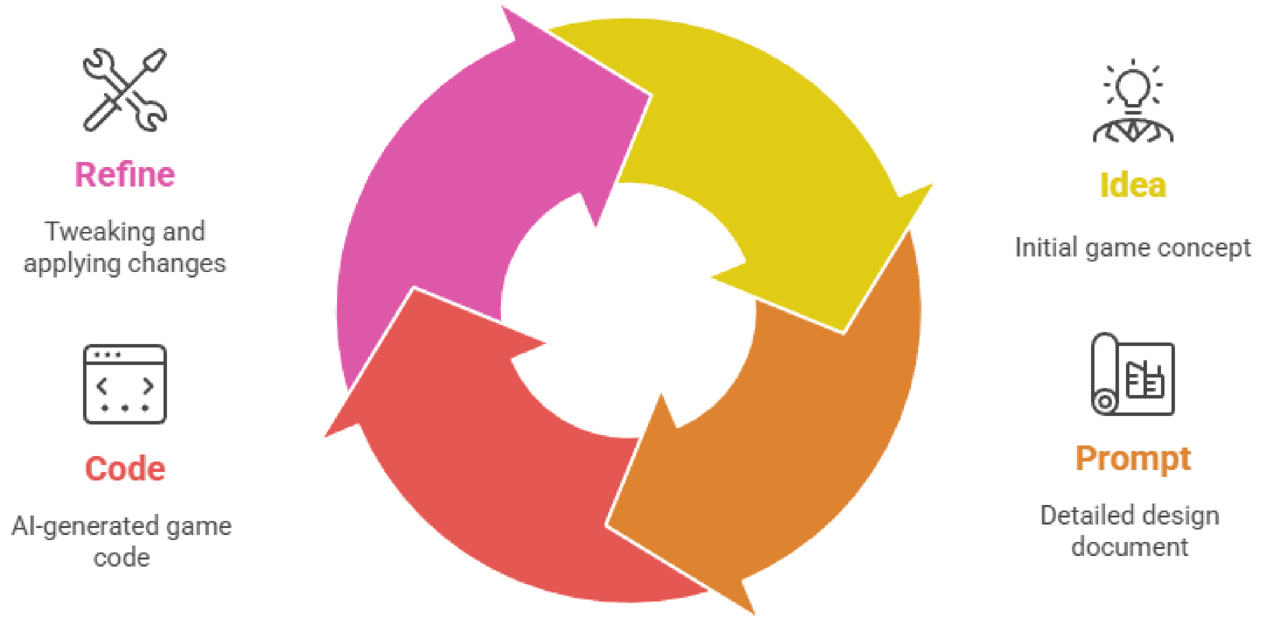
- More game prompts are available in **`/prompts/`**:
  - `game_design_prompt_snake.md` - Classic Snake game
  - `game_design_prompt_pingpong.md` - Two-player Pong game
  - `game_design_prompt_breakout.md` - Brick breaker game
  - `game_design_prompt_flappybird.md` - Flappy Bird style game
  - `game_design_prompt_spaceshooter.md` - Vertical space shooter
  - `game_design_prompt_mazerunner.md` - Maze navigation game

- *Each prompt includes: Objective, Gameplay Mechanics, GUI Design, Implementation Plan, and Deliverable*

`Workshop GitHub Repository:`
[`https://github.com/tisage/game-llm/`](https://github.com/tisage/game-llm/)

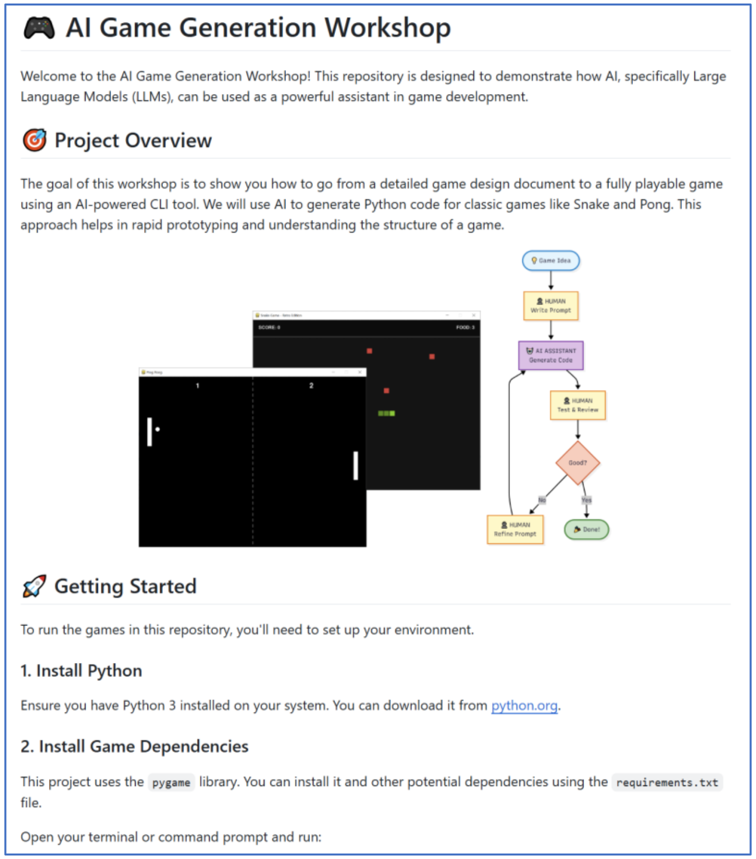
Recap

- **Idea**: We had an idea for a game.
- **Prompt**: We translated that idea into a detailed, structured design document.
- **Code**: The AI generated the code based on our blueprint.
- **Refine**: We can easily tweak the design and have the AI apply the changes.



Q & A

- **Thank You!**



🎮 AI Game Generation Workshop

Welcome to the AI Game Generation Workshop! This repository is designed to demonstrate how AI, specifically Large Language Models (LLMs), can be used as a powerful assistant in game development.

🎯 Project Overview

The goal of this workshop is to show you how to go from a detailed game design document to a fully playable game using an AI-powered CLI tool. We will use AI to generate Python code for classic games like Snake and Pong. This approach helps in rapid prototyping and understanding the structure of a game.

🚀 Getting Started

To run the games in this repository, you'll need to set up your environment.

1. Install Python

Ensure you have Python 3 installed on your system. You can download it from python.org.

2. Install Game Dependencies

This project uses the `pygame` library. You can install it and other potential dependencies using the `requirements.txt` file.

Open your terminal or command prompt and run:

Workshop GitHub Repository:
https://github.com/tisage/game-llm/

New Course CISC395 (Spring 2026)

New Course:

- **Applied Generative Artificial Intelligence and Large Language Model Applications**
- **Elective Course:** Special Topic
- CISC395 CRN 15738, Spring 2026

Key Course Highlights:

- **Practical Focus**: Mastery of state-of-the-art tools.
- **Core Skills**: Prompt engineering, AI-assisted coding (vibe coding), and building Retrieval-Augmented Generation (RAG) systems.
- **Hands-On Labs**: Develop real-world applications using industry-standard tools like Google AI Studio, Copilot CLI, Streamlit, and Gradio.