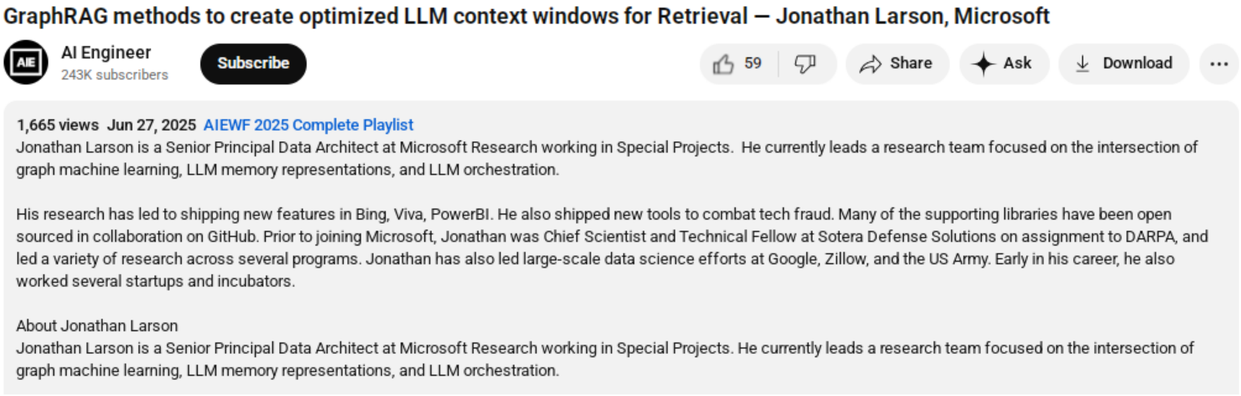


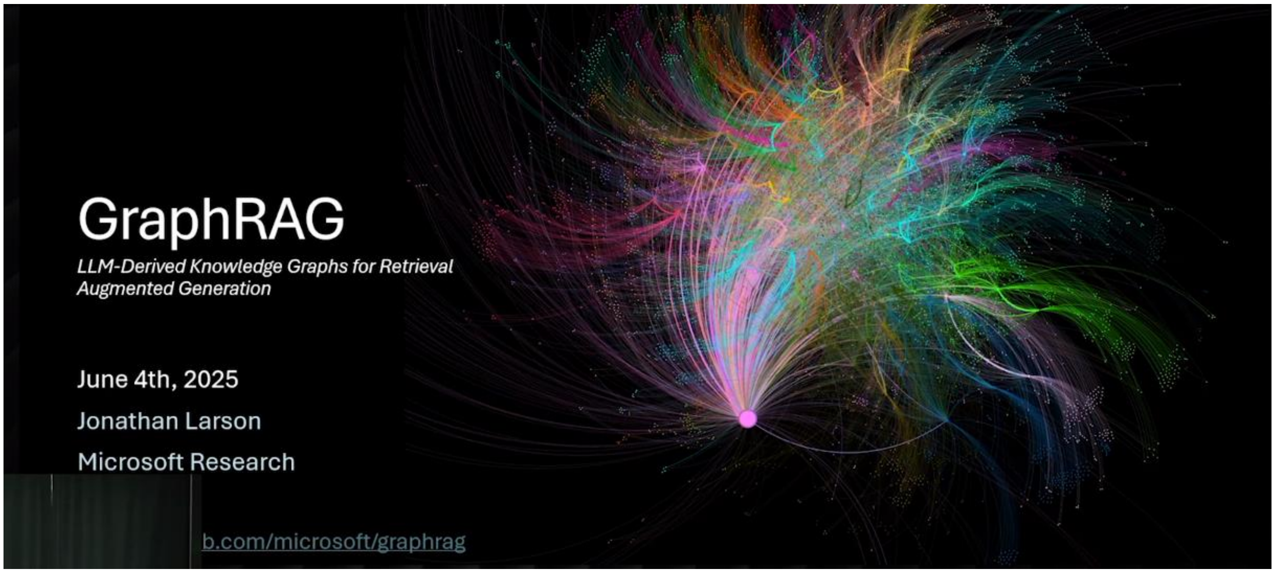
GraphRAG methods to create optimized LLM context windows for Retrieval — Jonathan Larson, Microsoft

AI Engineer
243K subscribers

1,665 views Jun 27, 2025 AIEWF 2025 Complete Playlist
Jonathan Larson is a Senior Principal Data Architect at Microsoft Research working in Special Projects. He currently leads a research team focused on the intersection of graph machine learning, LLM memory representations, and LLM orchestration.

His research has led to shipping new features in Bing, Viva, PowerBI. He also shipped new tools to combat tech fraud. Many of the supporting libraries have been open sourced in collaboration on GitHub. Prior to joining Microsoft, Jonathan was Chief Scientist and Technical Fellow at Sotera Defense Solutions on assignment to DARPA, and led a variety of research across several programs. Jonathan has also led large-scale data science efforts at Google, Zillow, and the US Army. Early in his career, he also worked several startups and incubators.

About Jonathan Larson
Jonathan Larson is a Senior Principal Data Architect at Microsoft Research working in Special Projects. He currently leads a research team focused on the intersection of graph machine learning, LLM memory representations, and LLM orchestration.

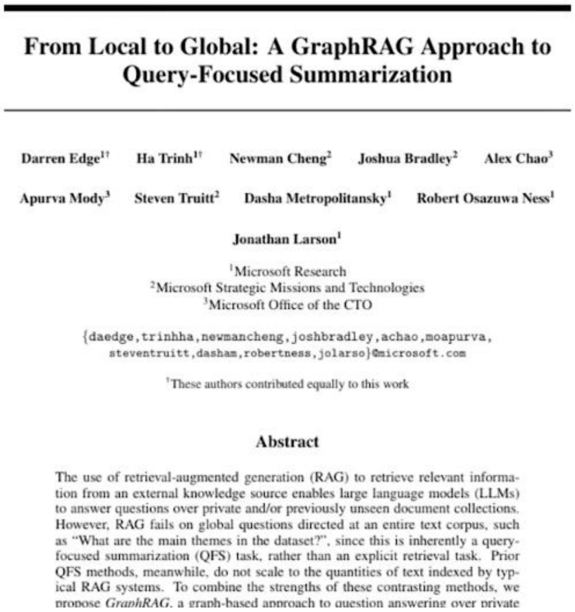


GraphRAG

LLM-Derived Knowledge Graphs for Retrieval Augmented Generation

June 4th, 2025
Jonathan Larson
Microsoft Research

b.com/microsoft/graphrag



From Local to Global: A GraphRAG Approach to Query-Focused Summarization

Darren Edge[1†] **Ha Trinh**[1†] **Newman Cheng**[2] **Joshua Bradley**[2] **Alex Chao**[3]

Apurva Mody[3] **Steven Truitt**[2] **Dasha Metropolitansky**[1] **Robert Osazuwa Ness**[1]

Jonathan Larson[1]

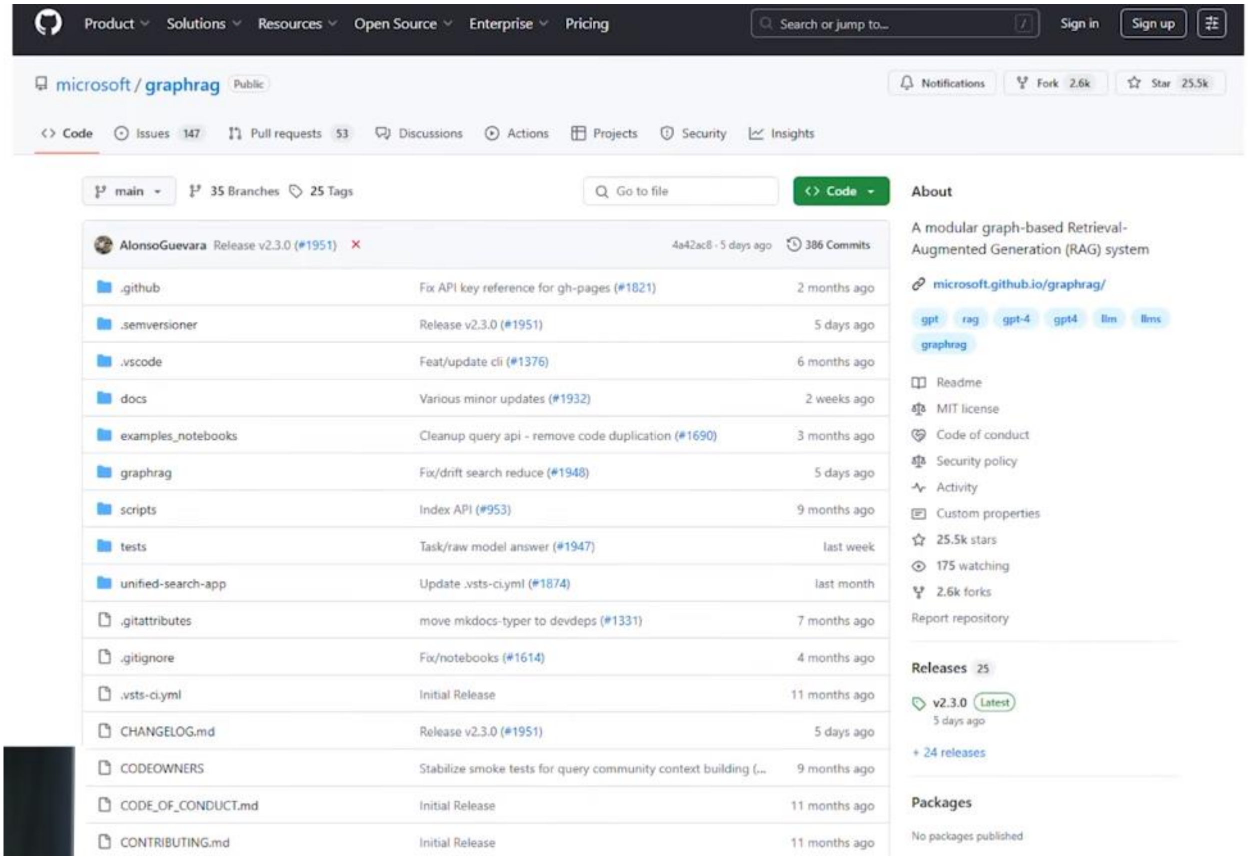
[1]Microsoft Research
[2]Microsoft Strategic Missions and Technologies
[3]Microsoft Office of the CTO

`{daedge,trinhha,newmancheng,joshbradley,achao,moapurva,`
`steventruitt,dasham,robertness,jolarso}@microsoft.com`

[†]These authors contributed equally to this work

Abstract

The use of retrieval-augmented generation (RAG) to retrieve relevant information from an external knowledge source enables large language models (LLMs) to answer questions over private and/or previously unseen document collections. However, RAG fails on global questions directed at an entire text corpus, such as "What are the main themes in the dataset?", since this is inherently a query-focused summarization (QFS) task, rather than an explicit retrieval task. Prior QFS methods, meanwhile, do not scale to the quantities of text indexed by typical RAG systems. To combine the strengths of these contrasting methods, we propose *GraphRAG*, a graph-based approach to question answering over private



microsoft / graphrag Public

AlonsoGuevara Release v2.3.0 (#1951) 4a42ac8 · 5 days ago 386 Commits

.github	Fix API key reference for gh-pages (#1821)	2 months ago
.semversioner	Release v2.3.0 (#1951)	5 days ago
.vscode	Feat/update cli (#1376)	6 months ago
docs	Various minor updates (#1932)	2 weeks ago
examples_notebooks	Cleanup query api - remove code duplication (#1690)	3 months ago
graphrag	Fix/drift search reduce (#1948)	5 days ago
scripts	Index API (#953)	9 months ago
tests	Task/raw model answer (#1947)	last week
unified-search-app	Update .vsts-ci.yml (#1874)	last month
.gitattributes	move mkdocs-typer to devdeps (#1331)	7 months ago
.gitignore	Fix/notebooks (#1614)	4 months ago
.vsts-ci.yml	Initial Release	11 months ago
CHANGELOG.md	Release v2.3.0 (#1951)	5 days ago
CODEOWNERS	Stabilize smoke tests for query community context building (...	9 months ago
CODE_OF_CONDUCT.md	Initial Release	11 months ago
CONTRIBUTING.md	Initial Release	11 months ago

About

A modular graph-based Retrieval-Augmented Generation (RAG) system

microsoft.github.io/graphrag/

gpt rag gpt-4 gpt4 llm llms graphrag

Readme
MIT license
Code of conduct
Security policy
Activity
Custom properties
25.5k stars
175 watching
2.6k forks
Report repository

Releases 25

v2.3.0 Latest
5 days ago

+ 24 releases

Packages

No packages published

Inspired many other offerings, books, and games

- Neo4j-graphrag-python
- Llamaindex graphrag
- Graphrag-lite
- Nano-graphrag
- Fast-graphrag
- Langchain-graphrag
- Tiny-graphrag
- LightRAG
- GraphRAG Local UI
- infiniflow/ragflow

- Microsoft Supercharges RAG with Knowledge Graphs

GraphRAG – The Card Game

New Horizons!

LLM memory with structure is a key enabler

- New Vertical: GraphRAG for Code
- **New OSS Release Today: BenchmarkQED**
- LazyGraphRAG Results
  - New Benchmarks
  - Azure Local
  - LazyGraphRAG for Science

GraphRAG for Code - Repo Understanding
An Example

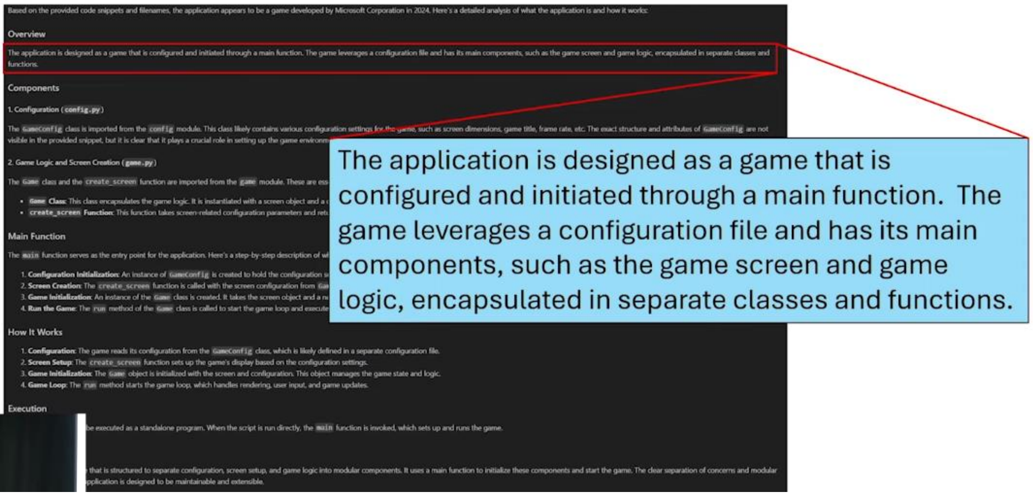
Terminal based Python game where the player earns points by avoiding obstacles and loses points if they collide.

- LLM has never seen this code before
- Small enough for human to know holistically for ground truth
- Complex enough that the LLM couldn't reason well over it
- ~200 lines of code across 7 files

[redacted]

Regular RAG – Q&A

"Describe what the application is and how it works."



Based on the provided code snippets and filenames, the application appears to be a game developed by Microsoft Corporation in 2024. Here's a detailed analysis of what the application is and how it works:

Overview

The application is designed as a game that is configured and initiated through a main function. The game leverages a configuration file and has its main components, such as the game screen and game logic, encapsulated in separate classes and functions.

Components

1. Configuration (`config.py`)

The `GameConfig` class is imported from the `config` module. This class likely contains various configuration settings for the game, such as screen dimensions, game title, frame rate, etc. The exact structure and attributes of `GameConfig` are not visible in the provided snippet, but it is clear that it plays a crucial role in setting up the game environm

2. Game Logic and Screen Creation (`game.py`)

The `Game` class and the `create_screen` function are imported from the `game` module. These are ess

- `Game` **Class**: This class encapsulates the game logic. It is instantiated with a screen object and a c
- `create_screen` **Function**: This function takes screen-related configuration parameters and retu

Main Function

The `main` function serves as the entry point for the application. Here's a step-by-step description of wh

1. **Configuration Initialization**: An instance of `GameConfig` is created to hold the configuration s
2. **Screen Creation**: The `create_screen` function is called with the screen configuration from Ga
3. **Game Initialization**: An instance of the `Game` class is created. It takes the screen object and a n
4. **Run the Game**: The `run` method of the `Game` class is called to start the game loop and execute

How It Works

1. **Configuration**: The game reads its configuration from the `GameConfig` class, which is likely defined in a separate configuration file.
2. **Screen Setup**: The `create_screen` function sets up the game's display based on the configuration settings.
3. **Game Initialization**: The `Game` object is initialized with the screen and configuration. This object manages the game state and logic.
4. **Game Loop**: The `run` method starts the game loop, which handles rendering, user input, and game updates.

Execution

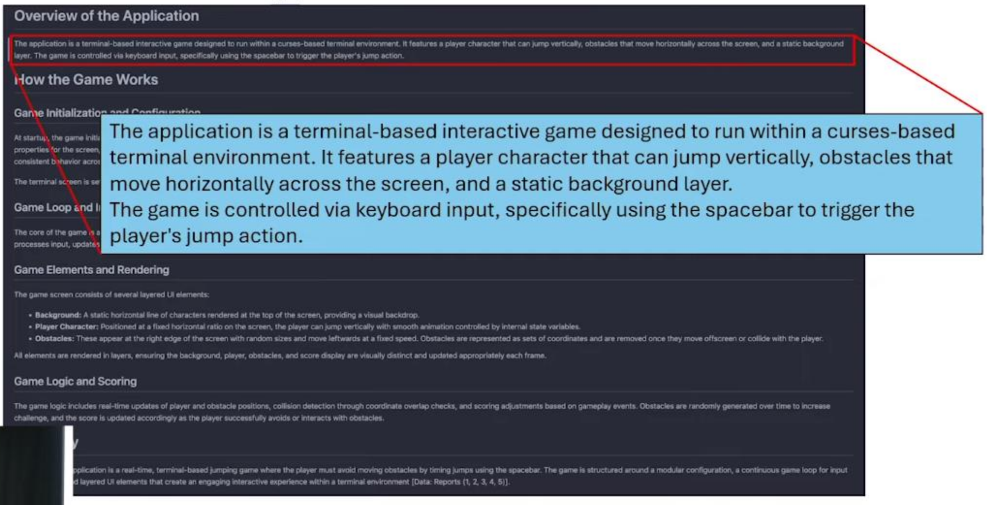
[redacted] be executed as a standalone program. When the script is run directly, the `main` function is invoked, which sets up and runs the game.

that is structured to separate configuration, screen setup, and game logic into modular components. It uses a main function to initialize these components and start the game. The clear separation of concerns and modular pplication is designed to be maintainable and extensible.

The application is designed as a game that is configured and initiated through a main function. The game leverages a configuration file and has its main components, such as the game screen and game logic, encapsulated in separate classes and functions.

GraphRAG for Code – Q&A

“Describe what the application is and how it works.”



Overview of the Application

The application is a terminal-based interactive game designed to run within a curses-based terminal environment. It features a player character that can jump vertically, obstacles that move horizontally across the screen, and a static background layer. The game is controlled via keyboard input, specifically using the spacebar to trigger the player's jump action.

How the Game Works

Game Initializatio

At startup, the game initi
properties for the screen,
consistent behavior acro

The terminal screen is se

Game Loop and I

The core of the game is a
processes input, updates

Game Elements and Rendering

The game screen consists of several layered UI elements:

- **Background:** A static horizontal line of characters rendered at the top of the screen, providing a visual backdrop.
- **Player Character:** Positioned at a fixed horizontal ratio on the screen, the player can jump vertically with smooth animation controlled by internal state variables.
- **Obstacles:** These appear at the right edge of the screen with random sizes and move leftwards at a fixed speed. Obstacles are represented as sets of coordinates and are removed once they move offscreen or collide with the player.

All elements are rendered in layers, ensuring the background, player, obstacles, and score display are visually distinct and updated appropriately each frame.

Game Logic and Scoring

The game logic includes real-time updates of player and obstacle positions, collision detection through coordinate overlap checks, and scoring adjustments based on gameplay events. Obstacles are randomly generated over time to increase challenge, and the score is updated accordingly as the player successfully avoids or interacts with obstacles.

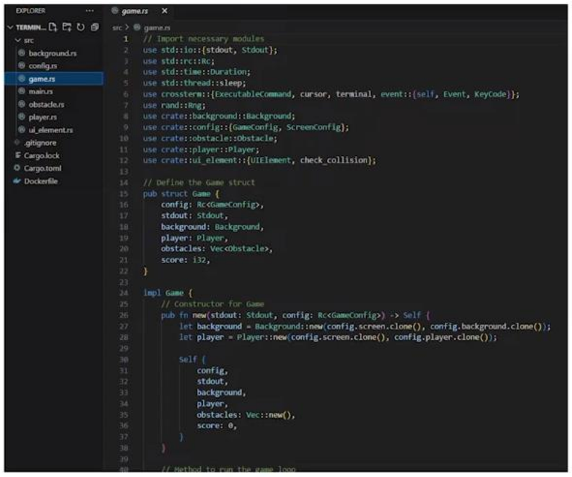
y

pplication is a real-time, terminal-based jumping game where the player must avoid moving obstacles by timing jumps using the spacebar. The game is structured around a modular configuration, a continuous game loop for input
d layered UI elements that create an engaging interactive experience within a terminal environment [Data: Reports (1, 2, 3, 4, 5)].

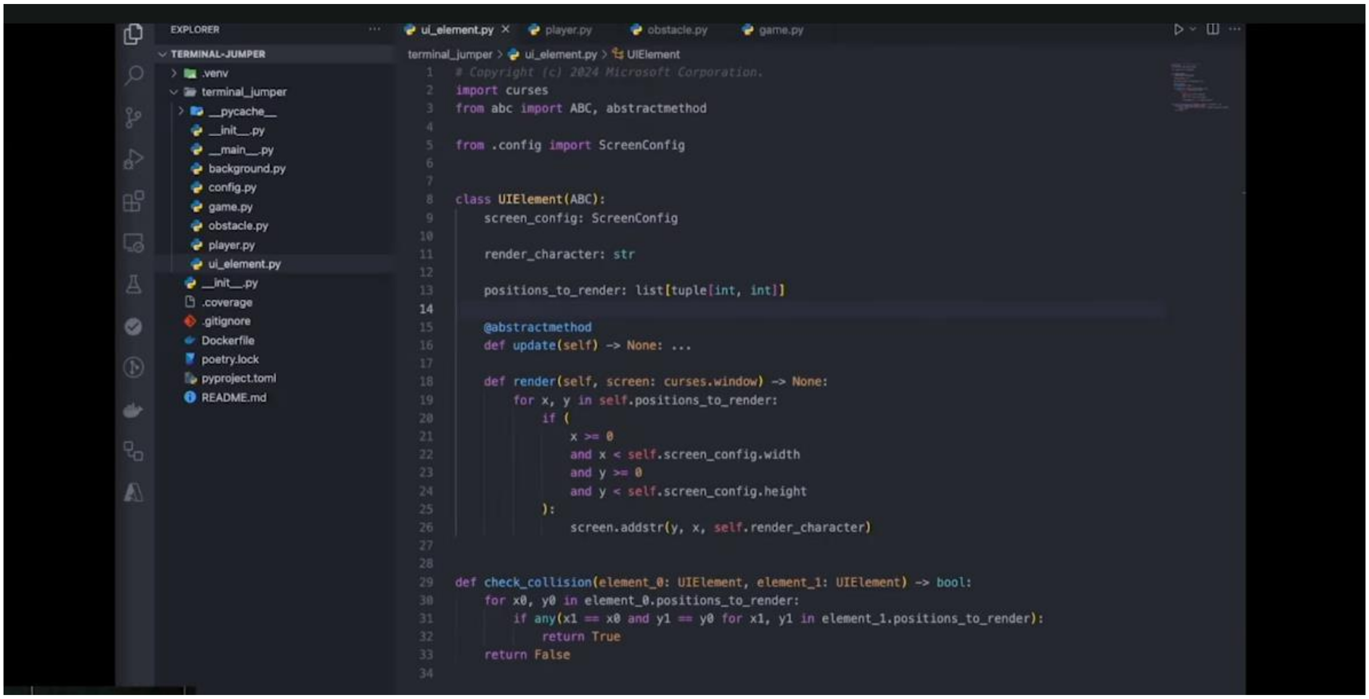
The application is a terminal-based interactive game designed to run within a curses-based terminal environment. It features a player character that can jump vertically, obstacles that move horizontally across the screen, and a static background layer.
The game is controlled via keyboard input, specifically using the spacebar to trigger the player's jump action.

GraphRAG for Code - Translation

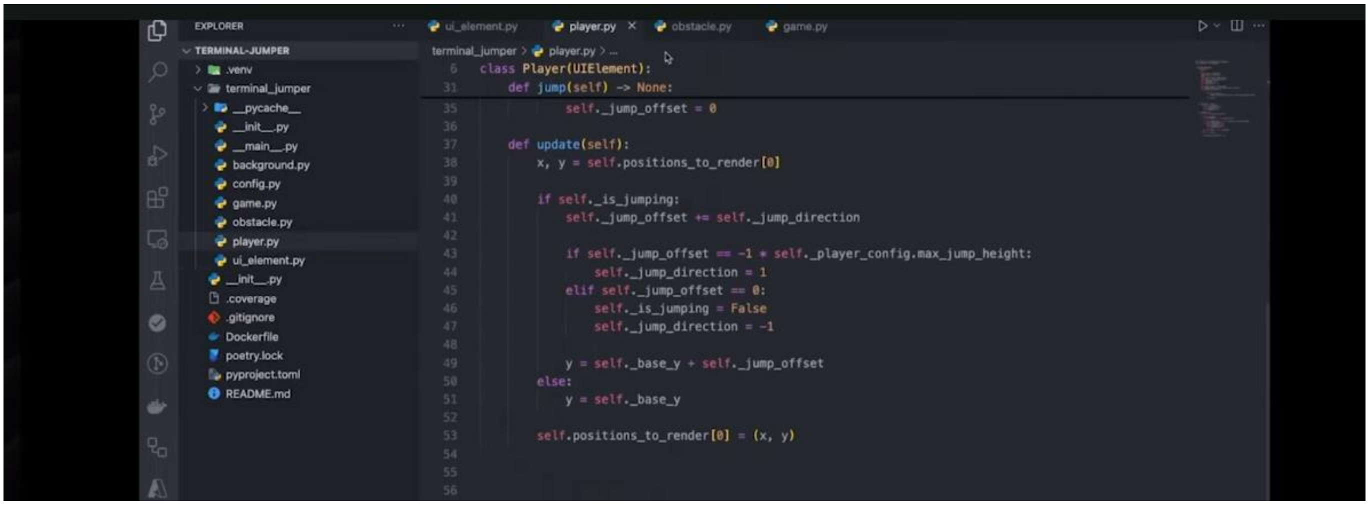
- Simply including all code directly in the context window fails
- Python -> Rust
  - Involved language specific libraries (E.g. curses)
  - Game codebase uses abstract classes
  - Rust has highly different semantics than Python
- Building Rust codebase generated **complete working code out of the box**



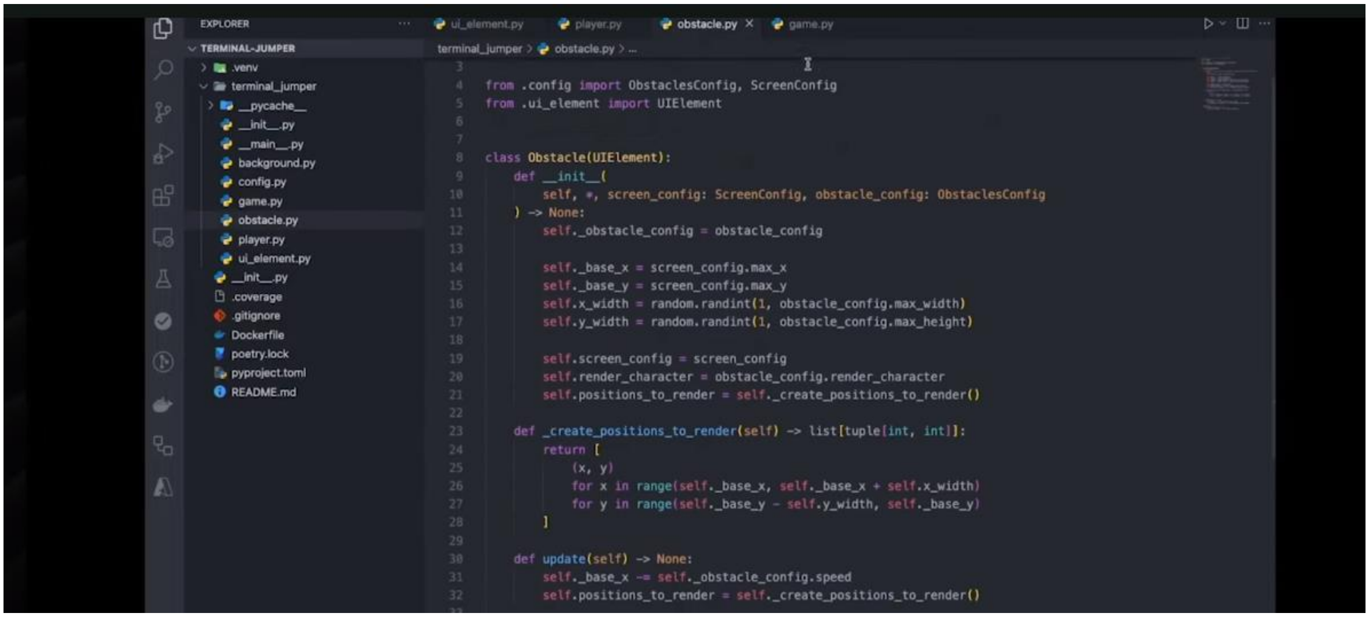
```rust
// Import necessary modules
use std::io::{stdout, Stdout};
use std::rc::Rc;
use std::time::Duration;
use std::thread::sleep;
use crossterm::{ExecutableCommand, cursor, terminal, event::{self, Event, KeyCode}};
use rand::Rng;
use crate::background::Background;
use crate::config::{GameConfig, ScreenConfig};
use crate::obstacle::Obstacle;
use crate::player::Player;
use crate::ui_element::{UIElement, check_collision};

// Define the Game struct
pub struct Game {
    config: Rc<GameConfig>,
    stdout: Stdout,
    background: Background,
    player: Player,
    obstacles: Vec<Obstacle>,
    score: i32,
}

impl Game {
    // Constructor for Game
    pub fn new(stdout: Stdout, config: Rc<GameConfig>) -> Self {
        let background = Background::new(config.screen.clone(), config.background.clone());
        let player = Player::new(config.screen.clone(), config.player.clone());

        Self {
            config,
            stdout,
            background,
            player,
            obstacles: Vec::new(),
            score: 0,
        }
    }

    // Method to run the game loop
```



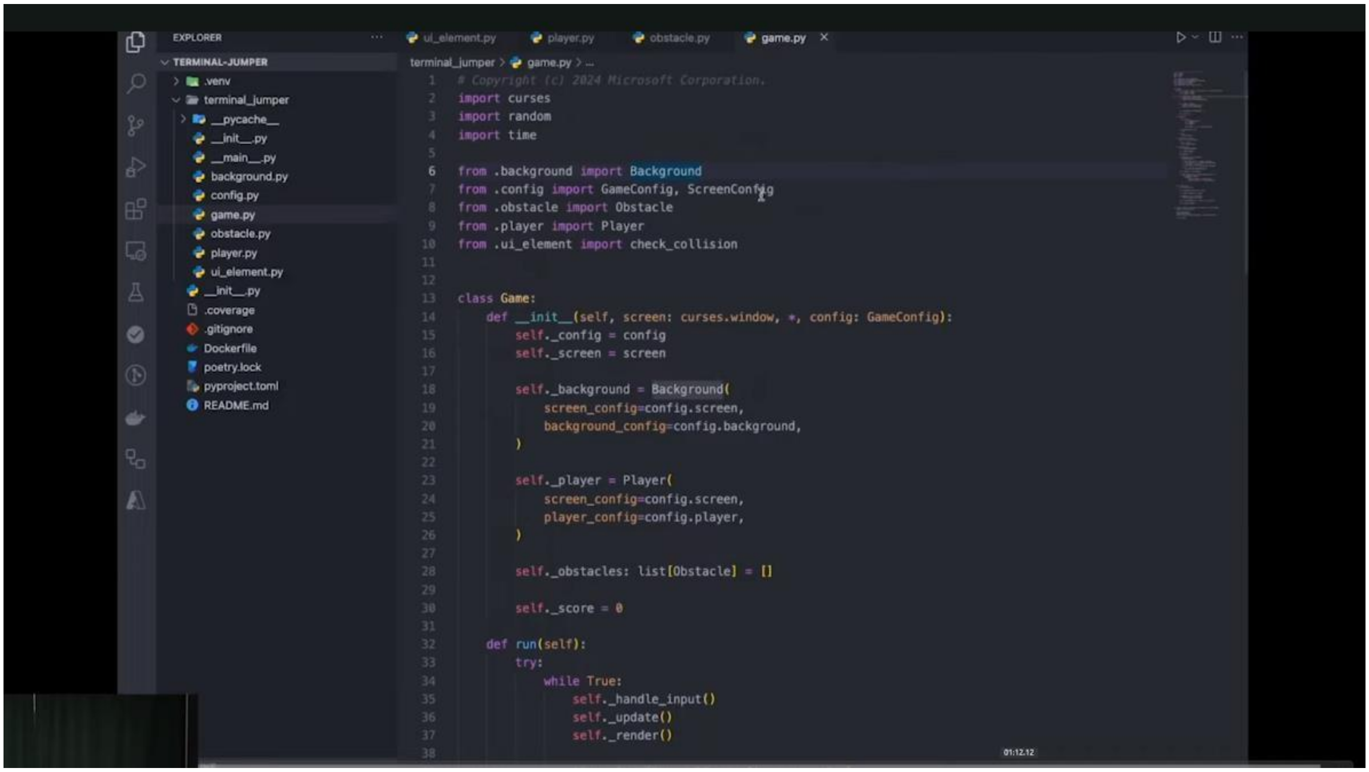
```python
# Copyright (c) 2024 Microsoft Corporation.
import curses
from abc import ABC, abstractmethod

from .config import ScreenConfig


class UIElement(ABC):
    screen_config: ScreenConfig

    render_character: str

    positions_to_render: list[tuple[int, int]]

    @abstractmethod
    def update(self) -> None: ...

    def render(self, screen: curses.window) -> None:
        for x, y in self.positions_to_render:
            if (
                x >= 0
                and x < self.screen_config.width
                and y >= 0
                and y < self.screen_config.height
            ):
                screen.addstr(y, x, self.render_character)


def check_collision(element_0: UIElement, element_1: UIElement) -> bool:
    for x0, y0 in element_0.positions_to_render:
        if any(x1 == x0 and y1 == y0 for x1, y1 in element_1.positions_to_render):
            return True
    return False

```



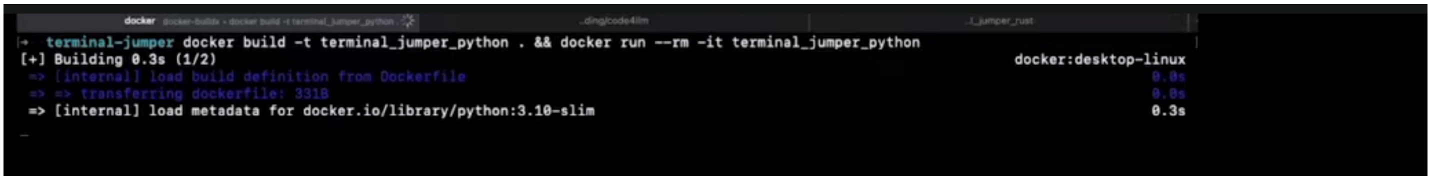
```python
class Player(UIElement):
    def jump(self) -> None:
            self._jump_offset = 0

    def update(self):
        x, y = self.positions_to_render[0]

        if self._is_jumping:
            self._jump_offset += self._jump_direction

            if self._jump_offset == -1 * self._player_config.max_jump_height:
                self._jump_direction = 1
            elif self._jump_offset == 0:
                self._is_jumping = False
                self._jump_direction = -1

            y = self._base_y + self._jump_offset
        else:
            y = self._base_y

        self.positions_to_render[0] = (x, y)
```



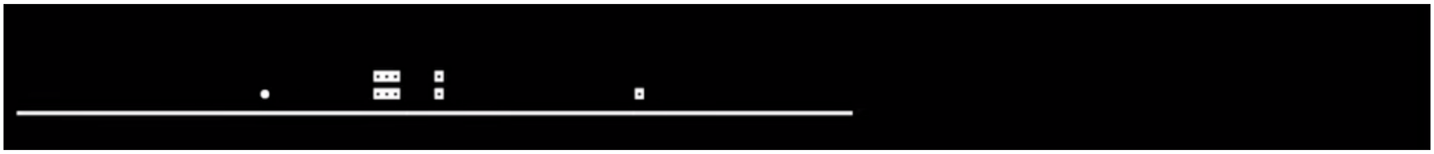
```python

from .config import ObstaclesConfig, ScreenConfig
from .ui_element import UIElement


class Obstacle(UIElement):
    def __init__(
        self, *, screen_config: ScreenConfig, obstacle_config: ObstaclesConfig
    ) -> None:
        self._obstacle_config = obstacle_config

        self._base_x = screen_config.max_x
        self._base_y = screen_config.max_y
        self.x_width = random.randint(1, obstacle_config.max_width)
        self.y_width = random.randint(1, obstacle_config.max_height)

        self.screen_config = screen_config
        self.render_character = obstacle_config.render_character
        self.positions_to_render = self._create_positions_to_render()

    def _create_positions_to_render(self) -> list[tuple[int, int]]:
        return [
            (x, y)
            for x in range(self._base_x, self._base_x + self.x_width)
            for y in range(self._base_y - self.y_width, self._base_y)
        ]

    def update(self) -> None:
        self._base_x -= self._obstacle_config.speed
        self.positions_to_render = self._create_positions_to_render()
```

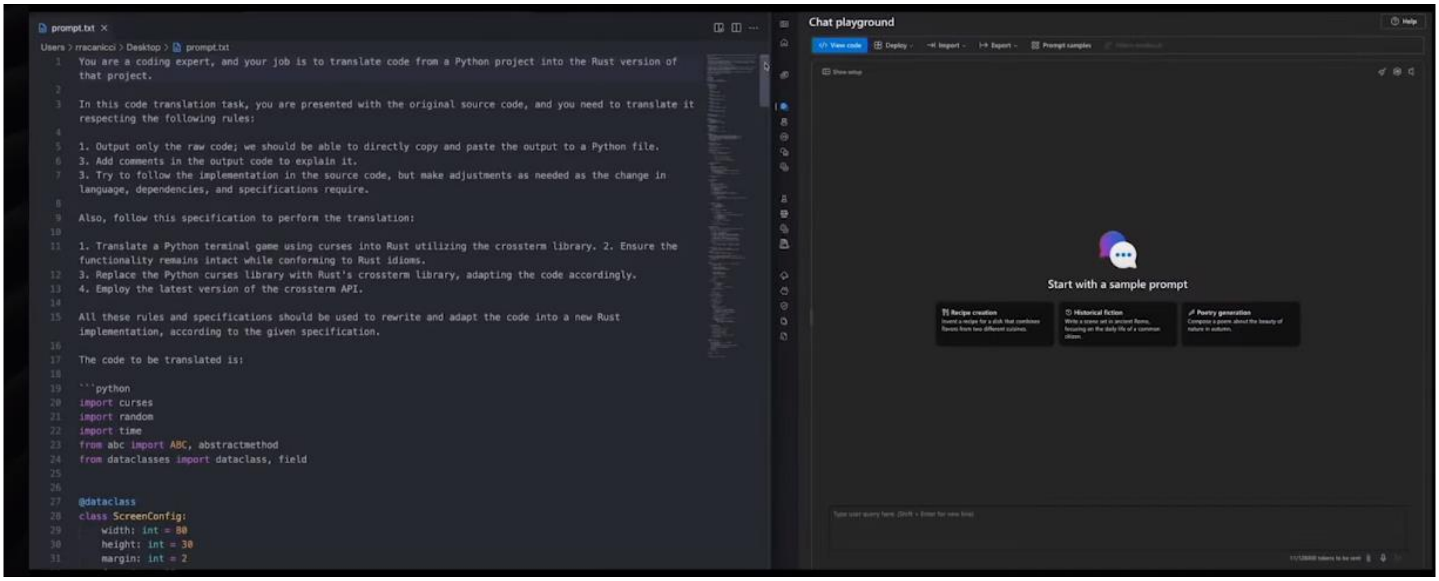


```python
# Copyright (c) 2024 Microsoft Corporation.
import curses
import random
import time

from .background import Background
from .config import GameConfig, ScreenConfig
from .obstacle import Obstacle
from .player import Player
from .ui_element import check_collision


class Game:
    def __init__(self, screen: curses.window, *, config: GameConfig):
        self._config = config
        self._screen = screen

        self._background = Background(
            screen_config=config.screen,
            background_config=config.background,
        )

        self._player = Player(
            screen_config=config.screen,
            player_config=config.player,
        )

        self._obstacles: list[Obstacle] = []

        self._score = 0

    def run(self):
        try:
            while True:
                self._handle_input()
                self._update()
                self._render()

```

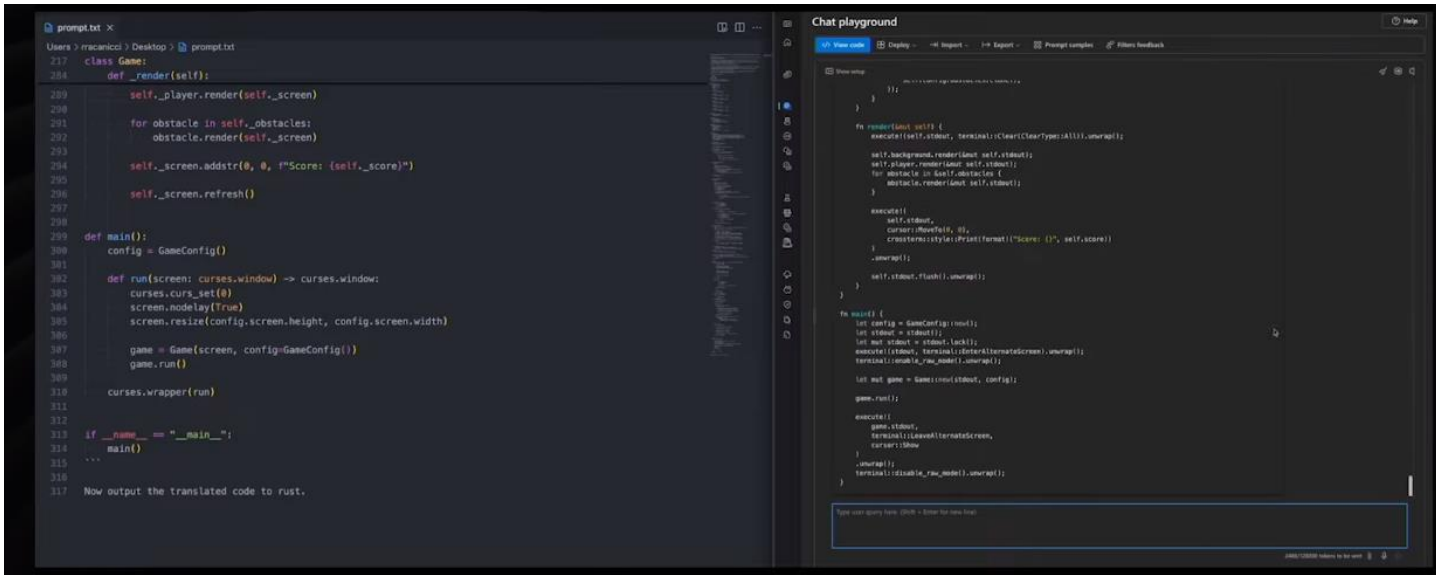


```
→  terminal-jumper docker build -t terminal_jumper_python . && docker run --rm -it terminal_jumper_python
[+] Building 0.3s (1/2)                                                                     docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                        0.0s
 => => transferring dockerfile: 331B                                                                        0.0s
 => [internal] load metadata for docker.io/library/python:3.10-slim                                         0.3s
```




prompt.txt
Users › rracanicci › Desktop › prompt.txt
Chat playground
Help
View code
Deploy
Import
Export
Prompt samples
Show setup
Start with a sample prompt
Recipe creation
Invent a recipe for a dish that combines flavors from two different cuisines.
Historical fiction
Write a scene set in ancient Rome, focusing on the daily life of a common citizen.
Poetry generation
Compose a poem about the beauty of nature in autumn.
Type user query here. (Shift + Enter for new line)

~~~
You are a coding expert, and your job is to translate code from a Python project into the Rust version of that project.

In this code translation task, you are presented with the original source code, and you need to translate it respecting the following rules:

1. Output only the raw code; we should be able to directly copy and paste the output to a Python file.
3. Add comments in the output code to explain it.
3. Try to follow the implementation in the source code, but make adjustments as needed as the change in language, dependencies, and specifications require.

Also, follow this specification to perform the translation:

1. Translate a Python terminal game using curses into Rust utilizing the crossterm library. 2. Ensure the functionality remains intact while conforming to Rust idioms.
3. Replace the Python curses library with Rust's crossterm library, adapting the code accordingly.
4. Employ the latest version of the crossterm API.

All these rules and specifications should be used to rewrite and adapt the code into a new Rust implementation, according to the given specification.

The code to be translated is:

```python
import curses
import random
import time
from abc import ABC, abstractmethod
from dataclasses import dataclass, field


@dataclass
class ScreenConfig:
    width: int = 80
    height: int = 30
    margin: int = 2
~~~



```
class Game:
    def _render(self):
            self._player.render(self._screen)

            for obstacle in self._obstacles:
                obstacle.render(self._screen)

            self._screen.addstr(0, 0, f"Score: {self._score}")

            self._screen.refresh()


def main():
    config = GameConfig()

    def run(screen: curses.window) -> curses.window:
        curses.curs_set(0)
        screen.nodelay(True)
        screen.resize(config.screen.height, config.screen.width)

        game = Game(screen, config=GameConfig())
        game.run()

    curses.wrapper(run)


if __name__ == "__main__":
    main()
```

Now output the translated code to rust.
```

```
            });
        }
    }

    fn render(&mut self) {
        execute!(self.stdout, terminal::Clear(ClearType::All)).unwrap();

        self.background.render(&mut self.stdout);
        self.player.render(&mut self.stdout);
        for obstacle in &self.obstacles {
            obstacle.render(&mut self.stdout);
        }

        execute!(
            self.stdout,
            cursor::MoveTo(0, 0),
            crossterm::style::Print(format!("Score: {}", self.score))
        )
        .unwrap();

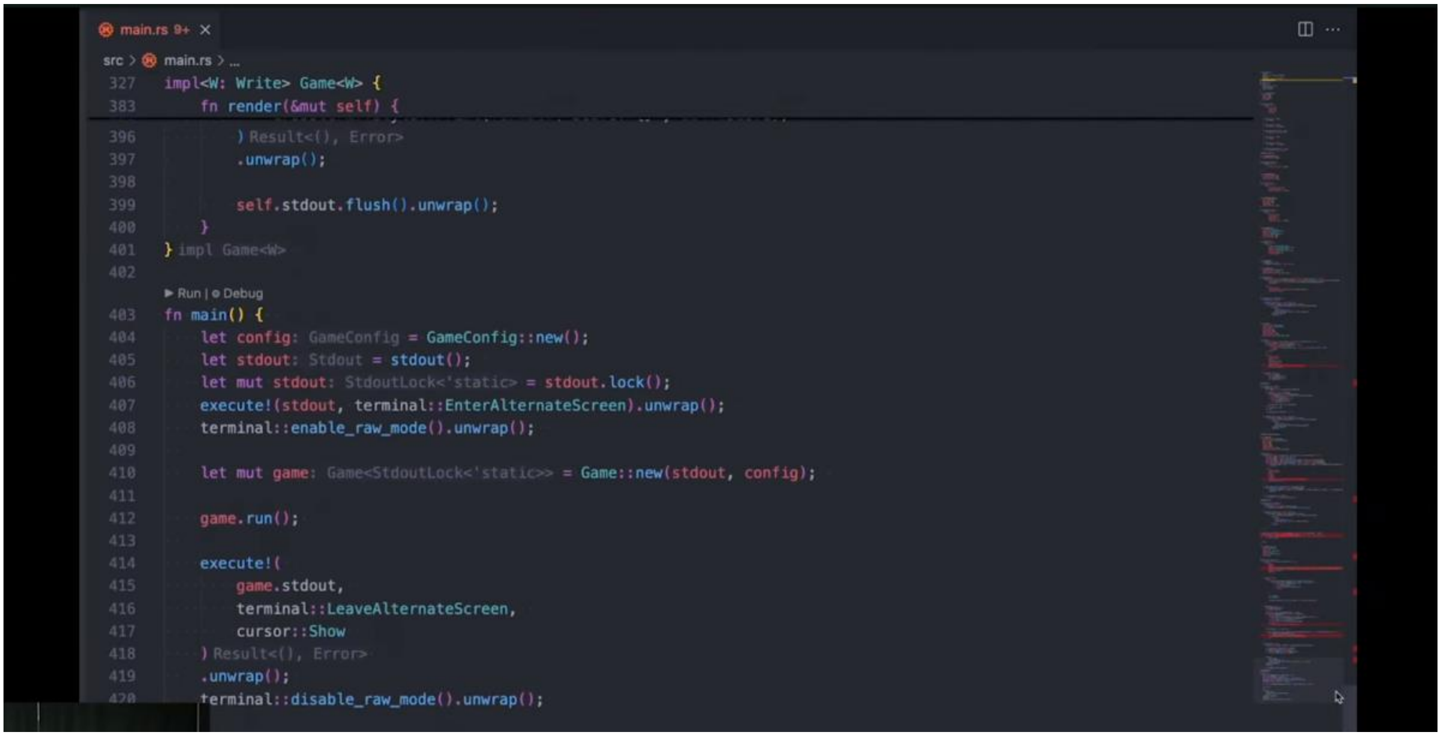
        self.stdout.flush().unwrap();
    }
}

fn main() {
    let config = GameConfig::new();
    let stdout = stdout();
    let mut stdout = stdout.lock();
    execute!(stdout, terminal::EnterAlternateScreen).unwrap();
    terminal::enable_raw_mode().unwrap();

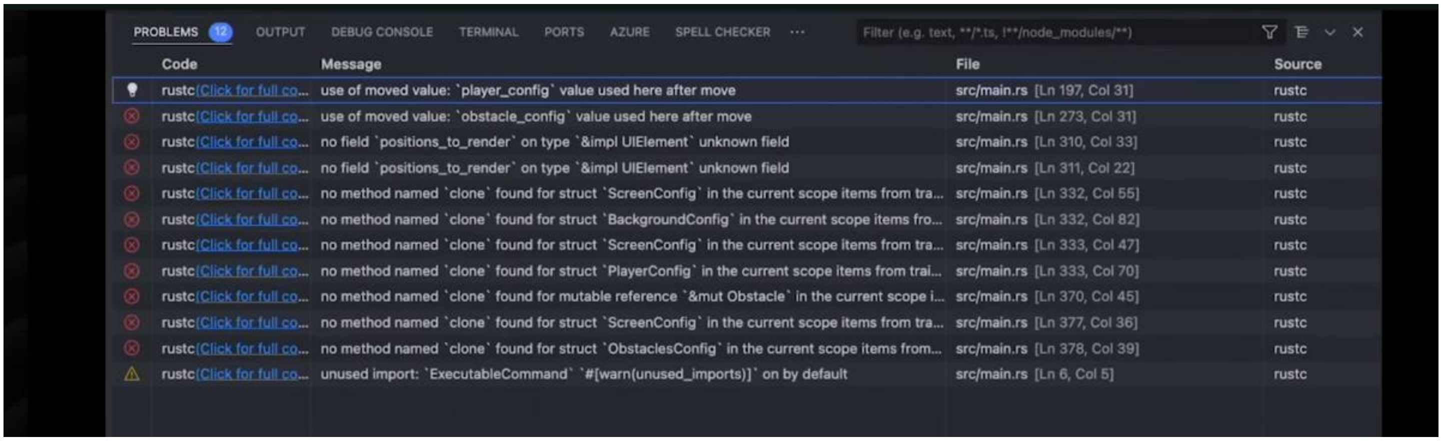
    let mut game = Game::new(stdout, config);

    game.run();

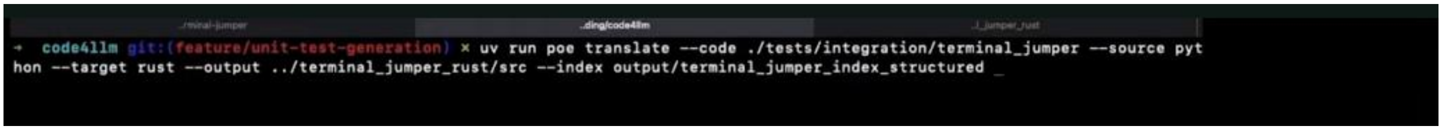
    execute!(
        game.stdout,
        terminal::LeaveAlternateScreen,
        cursor::Show
    )
    .unwrap();
    terminal::disable_raw_mode().unwrap();
}
```
```



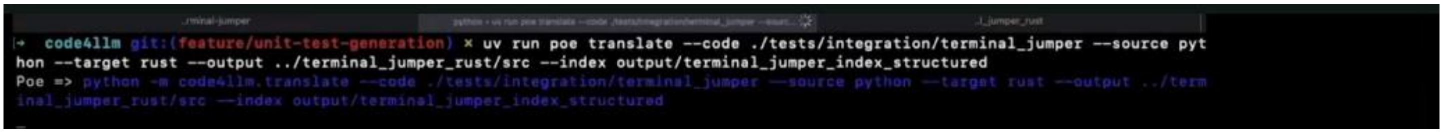
```
impl<W: Write> Game<W> {
    fn render(&mut self) {
            ) Result<(), Error>
            .unwrap();

            self.stdout.flush().unwrap();
    }
} impl Game<W>

fn main() {
    let config: GameConfig = GameConfig::new();
    let stdout: Stdout = stdout();
    let mut stdout: StdoutLock<'static> = stdout.lock();
    execute!(stdout, terminal::EnterAlternateScreen).unwrap();
    terminal::enable_raw_mode().unwrap();

    let mut game: Game<StdoutLock<'static>> = Game::new(stdout, config);

    game.run();

    execute!(
        game.stdout,
        terminal::LeaveAlternateScreen,
        cursor::Show
    ) Result<(), Error>
    .unwrap();
    terminal::disable_raw_mode().unwrap();
```



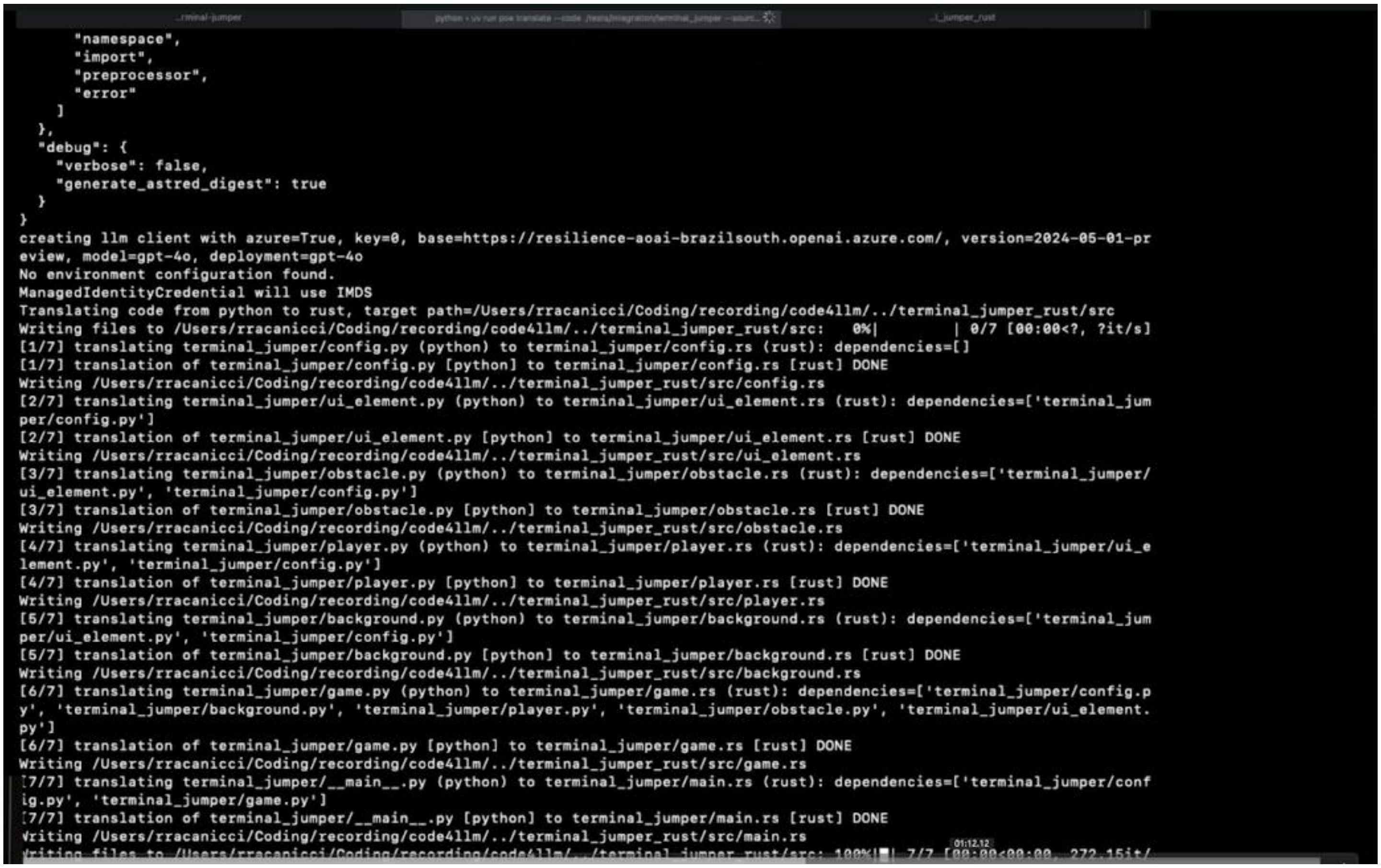
	Code	Message	File	Source
💡	rustc(Click for full co...	use of moved value: `player_config` value used here after move	src/main.rs [Ln 197, Col 31]	rustc
⊗	rustc(Click for full co...	use of moved value: `obstacle_config` value used here after move	src/main.rs [Ln 273, Col 31]	rustc
⊗	rustc(Click for full co...	no field `positions_to_render` on type `&impl UIElement` unknown field	src/main.rs [Ln 310, Col 33]	rustc
⊗	rustc(Click for full co...	no field `positions_to_render` on type `&impl UIElement` unknown field	src/main.rs [Ln 311, Col 22]	rustc
⊗	rustc(Click for full co...	no method named `clone` found for struct `ScreenConfig` in the current scope items from tra...	src/main.rs [Ln 332, Col 55]	rustc
⊗	rustc(Click for full co...	no method named `clone` found for struct `BackgroundConfig` in the current scope items fro...	src/main.rs [Ln 332, Col 82]	rustc
⊗	rustc(Click for full co...	no method named `clone` found for struct `ScreenConfig` in the current scope items from tra...	src/main.rs [Ln 333, Col 47]	rustc
⊗	rustc(Click for full co...	no method named `clone` found for struct `PlayerConfig` in the current scope items from trai...	src/main.rs [Ln 333, Col 70]	rustc
⊗	rustc(Click for full co...	no method named `clone` found for mutable reference `&mut Obstacle` in the current scope i...	src/main.rs [Ln 370, Col 45]	rustc
⊗	rustc(Click for full co...	no method named `clone` found for struct `ScreenConfig` in the current scope items from tra...	src/main.rs [Ln 377, Col 36]	rustc
⊗	rustc(Click for full co...	no method named `clone` found for struct `ObstaclesConfig` in the current scope items from...	src/main.rs [Ln 378, Col 39]	rustc
⚠	rustc(Click for full co...	unused import: `ExecutableCommand` `#[warn(unused_imports)]` on by default	src/main.rs [Ln 6, Col 5]	rustc



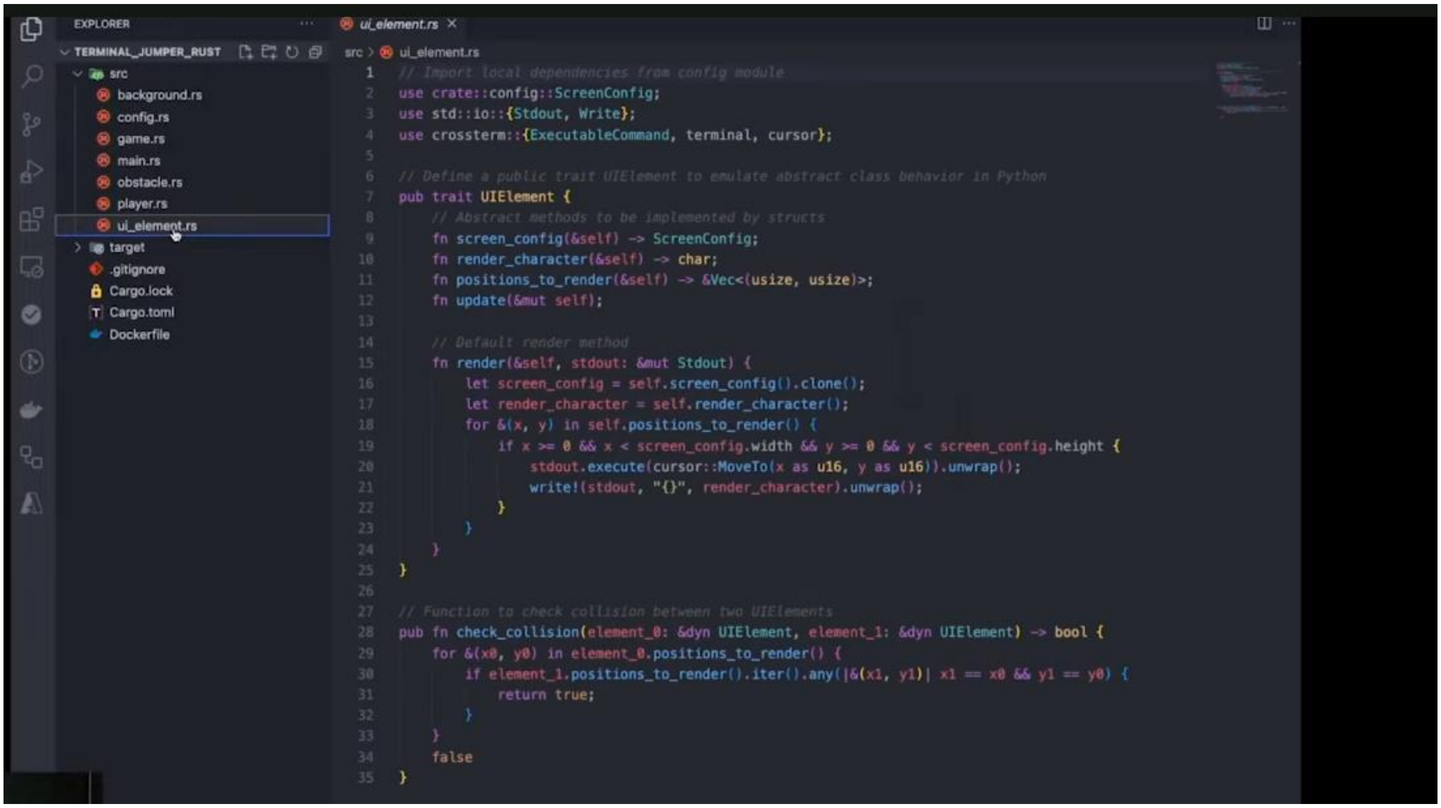
```
→  code4llm git:(feature/unit-test-generation) ✗ uv run poe translate --code ./tests/integration/terminal_jumper --source pyt
hon --target rust --output ../terminal_jumper_rust/src --index output/terminal_jumper_index_structured _
```



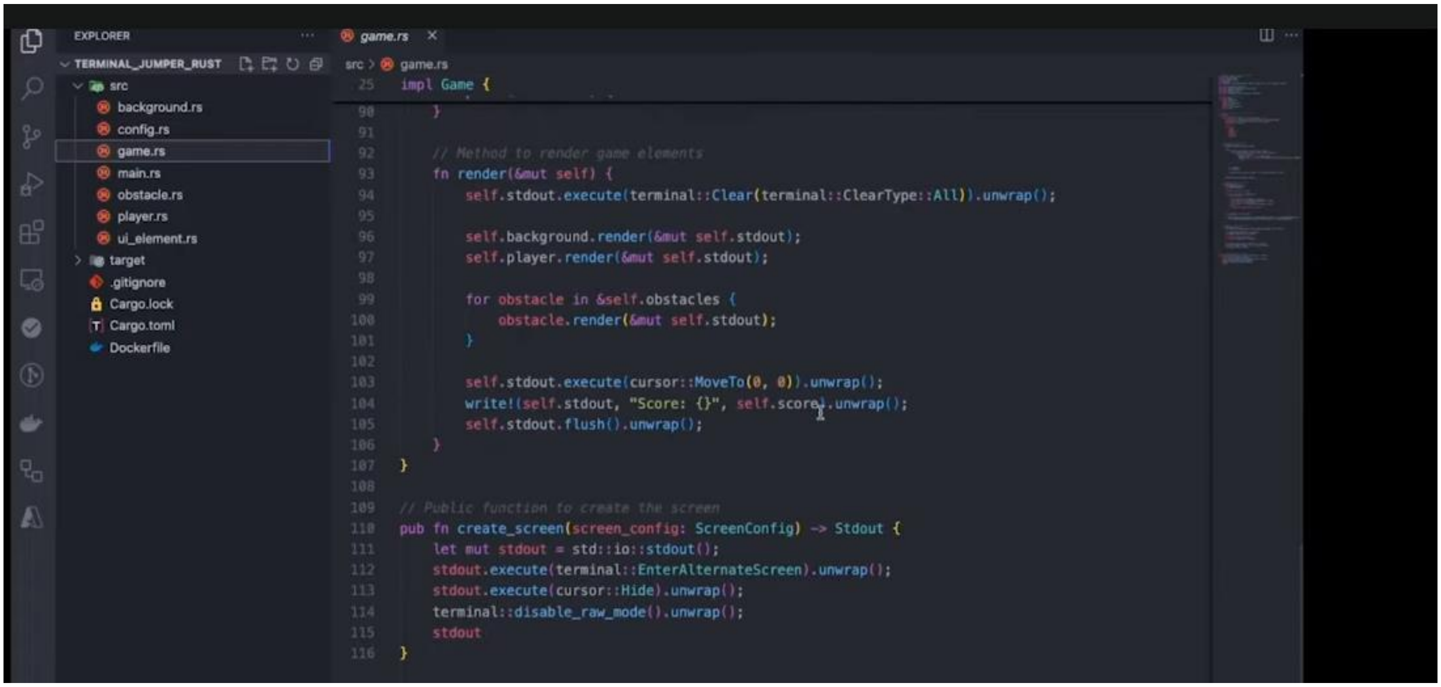
```
→  code4llm git:(feature/unit-test-generation) ✗ uv run poe translate --code ./tests/integration/terminal_jumper --source pyt
hon --target rust --output ../terminal_jumper_rust/src --index output/terminal_jumper_index_structured
Poe => python -m code4llm.translate --code ./tests/integration/terminal_jumper --source python --target rust --output ../term
inal_jumper_rust/src --index output/terminal_jumper_index_structured
_
```



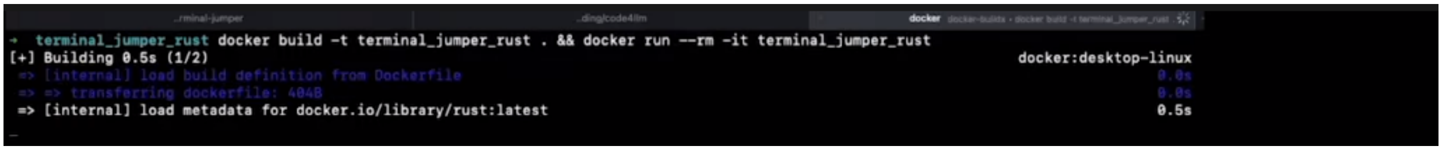
```
      "namespace",
      "import",
      "preprocessor",
      "error"
    ]
  },
  "debug": {
    "verbose": false,
    "generate_astred_digest": true
  }
}
creating llm client with azure=True, key=0, base=https://resilience-aoai-brazilsouth.openai.azure.com/, version=2024-05-01-pr
eview, model=gpt-4o, deployment=gpt-4o
No environment configuration found.
ManagedIdentityCredential will use IMDS
Translating code from python to rust, target path=/Users/rracanicci/Coding/recording/code4llm/../terminal_jumper_rust/src
Writing files to /Users/rracanicci/Coding/recording/code4llm/../terminal_jumper_rust/src:   0%|          | 0/7 [00:00<?, ?it/s]
[1/7] translating terminal_jumper/config.py (python) to terminal_jumper/config.rs (rust): dependencies=[]
[1/7] translation of terminal_jumper/config.py [python] to terminal_jumper/config.rs [rust] DONE
Writing /Users/rracanicci/Coding/recording/code4llm/../terminal_jumper_rust/src/config.rs
[2/7] translating terminal_jumper/ui_element.py (python) to terminal_jumper/ui_element.rs (rust): dependencies=['terminal_jum
per/config.py']
[2/7] translation of terminal_jumper/ui_element.py [python] to terminal_jumper/ui_element.rs [rust] DONE
Writing /Users/rracanicci/Coding/recording/code4llm/../terminal_jumper_rust/src/ui_element.rs
[3/7] translating terminal_jumper/obstacle.py (python) to terminal_jumper/obstacle.rs (rust): dependencies=['terminal_jumper/
ui_element.py', 'terminal_jumper/config.py']
[3/7] translation of terminal_jumper/obstacle.py [python] to terminal_jumper/obstacle.rs [rust] DONE
Writing /Users/rracanicci/Coding/recording/code4llm/../terminal_jumper_rust/src/obstacle.rs
[4/7] translating terminal_jumper/player.py (python) to terminal_jumper/player.rs (rust): dependencies=['terminal_jumper/ui_e
lement.py', 'terminal_jumper/config.py']
[4/7] translation of terminal_jumper/player.py [python] to terminal_jumper/player.rs [rust] DONE
Writing /Users/rracanicci/Coding/recording/code4llm/../terminal_jumper_rust/src/player.rs
[5/7] translating terminal_jumper/background.py (python) to terminal_jumper/background.rs (rust): dependencies=['terminal_jum
per/ui_element.py', 'terminal_jumper/config.py']
[5/7] translation of terminal_jumper/background.py [python] to terminal_jumper/background.rs [rust] DONE
Writing /Users/rracanicci/Coding/recording/code4llm/../terminal_jumper_rust/src/background.rs
[6/7] translating terminal_jumper/game.py (python) to terminal_jumper/game.rs (rust): dependencies=['terminal_jumper/config.p
y', 'terminal_jumper/background.py', 'terminal_jumper/player.py', 'terminal_jumper/obstacle.py', 'terminal_jumper/ui_element.
py']
[6/7] translation of terminal_jumper/game.py [python] to terminal_jumper/game.rs [rust] DONE
Writing /Users/rracanicci/Coding/recording/code4llm/../terminal_jumper_rust/src/game.rs
[7/7] translating terminal_jumper/__main__.py (python) to terminal_jumper/main.rs (rust): dependencies=['terminal_jumper/conf
ig.py', 'terminal_jumper/game.py']
[7/7] translation of terminal_jumper/__main__.py [python] to terminal_jumper/main.rs [rust] DONE
Writing /Users/rracanicci/Coding/recording/code4llm/../terminal_jumper_rust/src/main.rs
Writing files to /Users/rracanicci/Coding/recording/code4llm/../terminal_jumper_rust/src: 100%|█| 7/7 [00:00<00:00, 272.15it/
```



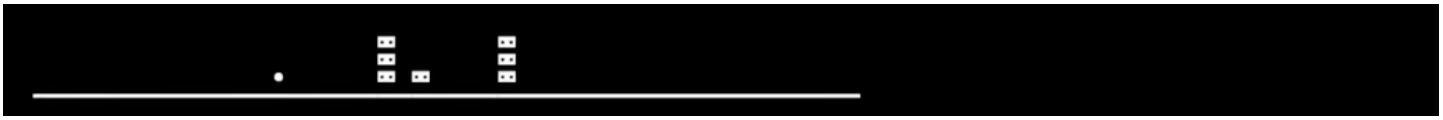
```rust
// Import local dependencies from config module
use crate::config::ScreenConfig;
use std::io::{Stdout, Write};
use crossterm::{ExecutableCommand, terminal, cursor};

// Define a public trait UIElement to emulate abstract class behavior in Python
pub trait UIElement {
    // Abstract methods to be implemented by structs
    fn screen_config(&self) -> ScreenConfig;
    fn render_character(&self) -> char;
    fn positions_to_render(&self) -> &Vec<(usize, usize)>;
    fn update(&mut self);

    // Default render method
    fn render(&self, stdout: &mut Stdout) {
        let screen_config = self.screen_config().clone();
        let render_character = self.render_character();
        for &(x, y) in self.positions_to_render() {
            if x >= 0 && x < screen_config.width && y >= 0 && y < screen_config.height {
                stdout.execute(cursor::MoveTo(x as u16, y as u16)).unwrap();
                write!(stdout, "{}", render_character).unwrap();
            }
        }
    }
}

// Function to check collision between two UIElements
pub fn check_collision(element_0: &dyn UIElement, element_1: &dyn UIElement) -> bool {
    for &(x0, y0) in element_0.positions_to_render() {
        if element_1.positions_to_render().iter().any(|&(x1, y1)| x1 == x0 && y1 == y0) {
            return true;
        }
    }
    false
}
```



```rust
impl Game {
    }

    // Method to render game elements
    fn render(&mut self) {
        self.stdout.execute(terminal::Clear(terminal::ClearType::All)).unwrap();

        self.background.render(&mut self.stdout);
        self.player.render(&mut self.stdout);

        for obstacle in &self.obstacles {
            obstacle.render(&mut self.stdout);
        }

        self.stdout.execute(cursor::MoveTo(0, 0)).unwrap();
        write!(self.stdout, "Score: {}", self.score).unwrap();
        self.stdout.flush().unwrap();
    }
}

// Public function to create the screen
pub fn create_screen(screen_config: ScreenConfig) -> Stdout {
    let mut stdout = std::io::stdout();
    stdout.execute(terminal::EnterAlternateScreen).unwrap();
    stdout.execute(cursor::Hide).unwrap();
    terminal::disable_raw_mode().unwrap();
    stdout
}
```



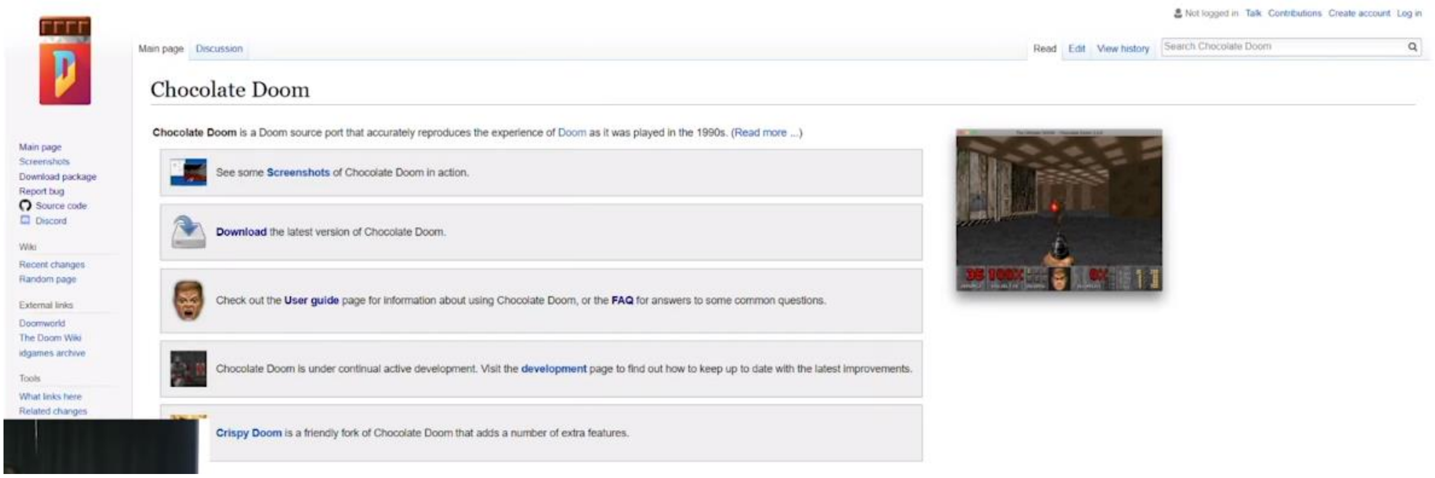
```
→  terminal_jumper_rust docker build -t terminal_jumper_rust . && docker run --rm -it terminal_jumper_rust
[+] Building 0.5s (1/2)                                                                     docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                        0.0s
 => => transferring dockerfile: 404B                                                                        0.0s
 => [internal] load metadata for docker.io/library/rust:latest                                              0.5s
```



Chocolate Doom

Original 30+ year old Doom Source updated to run on modern OS

- Doom codebase is
  - 96K+ lines of code
  - 231 files
  - 800K+ tokens



Chocolate Doom

Chocolate Doom is a Doom source port that accurately reproduces the experience of Doom as it was played in the 1990s. (Read more ...)

See some **Screenshots** of Chocolate Doom in action.

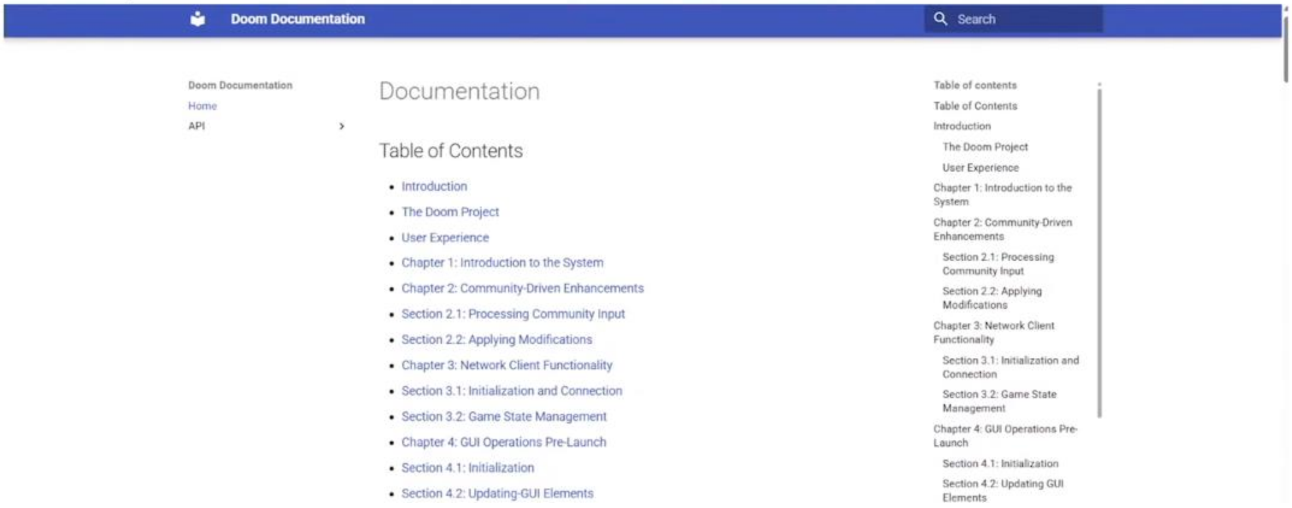
Download the latest version of Chocolate Doom.

Check out the **User guide** page for information about using Chocolate Doom, or the **FAQ** for answers to some common questions.

Chocolate Doom is under continual active development. Visit the **development** page to find out how to keep up to date with the latest improvements.

Crispy Doom is a friendly fork of Chocolate Doom that adds a number of extra features.

GraphRAG for Code - Automated Documentation



Doom Documentation

Search

Doom Documentation

Home

API

Documentation

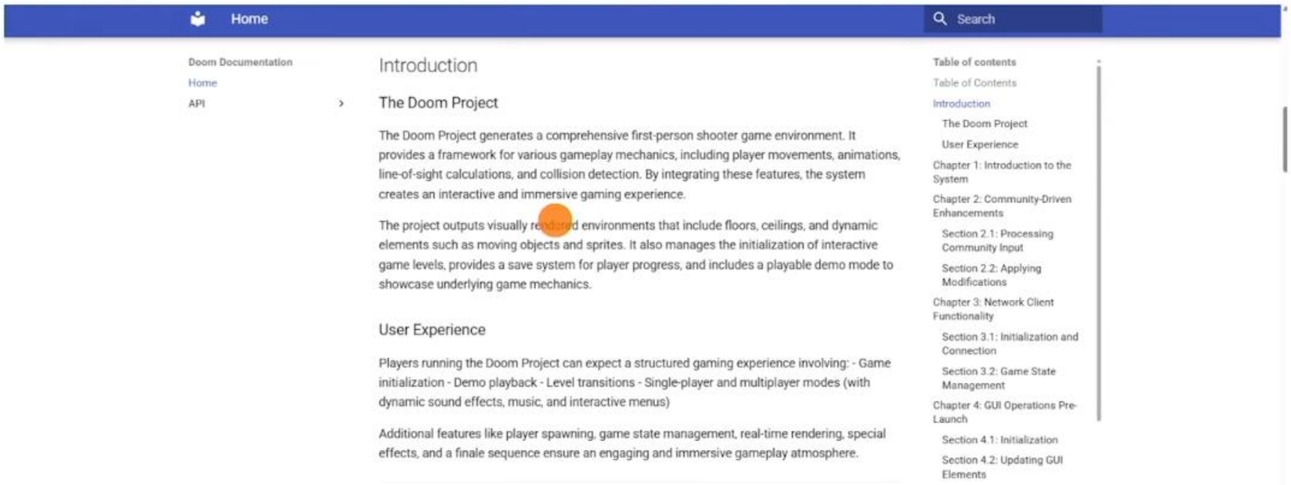
Table of Contents

- Introduction
- The Doom Project
- User Experience
- Chapter 1: Introduction to the System
- Chapter 2: Community-Driven Enhancements
- Section 2.1: Processing Community Input
- Section 2.2: Applying Modifications
- Chapter 3: Network Client Functionality
- Section 3.1: Initialization and Connection
- Section 3.2: Game State Management
- Chapter 4: GUI Operations Pre-Launch
- Section 4.1: Initialization
- Section 4.2: Updating-GUI Elements

Table of contents

- Table of Contents
- Introduction
  - The Doom Project
  - User Experience
- Chapter 1: Introduction to the System
- Chapter 2: Community-Driven Enhancements
  - Section 2.1: Processing Community Input
  - Section 2.2: Applying Modifications
- Chapter 3: Network Client Functionality
  - Section 3.1: Initialization and Connection
  - Section 3.2: Game State Management
- Chapter 4: GUI Operations Pre-Launch
  - Section 4.1: Initialization
  - Section 4.2: Updating GUI Elements

GraphRAG for Code - Automated Documentation



Home

Search

Doom Documentation

Home

API

Introduction

The Doom Project

The Doom Project generates a comprehensive first-person shooter game environment. It provides a framework for various gameplay mechanics, including player movements, animations, line-of-sight calculations, and collision detection. By integrating these features, the system creates an interactive and immersive gaming experience.

The project outputs visually rendered environments that include floors, ceilings, and dynamic elements such as moving objects and sprites. It also manages the initialization of interactive game levels, provides a save system for player progress, and includes a playable demo mode to showcase underlying game mechanics.

User Experience

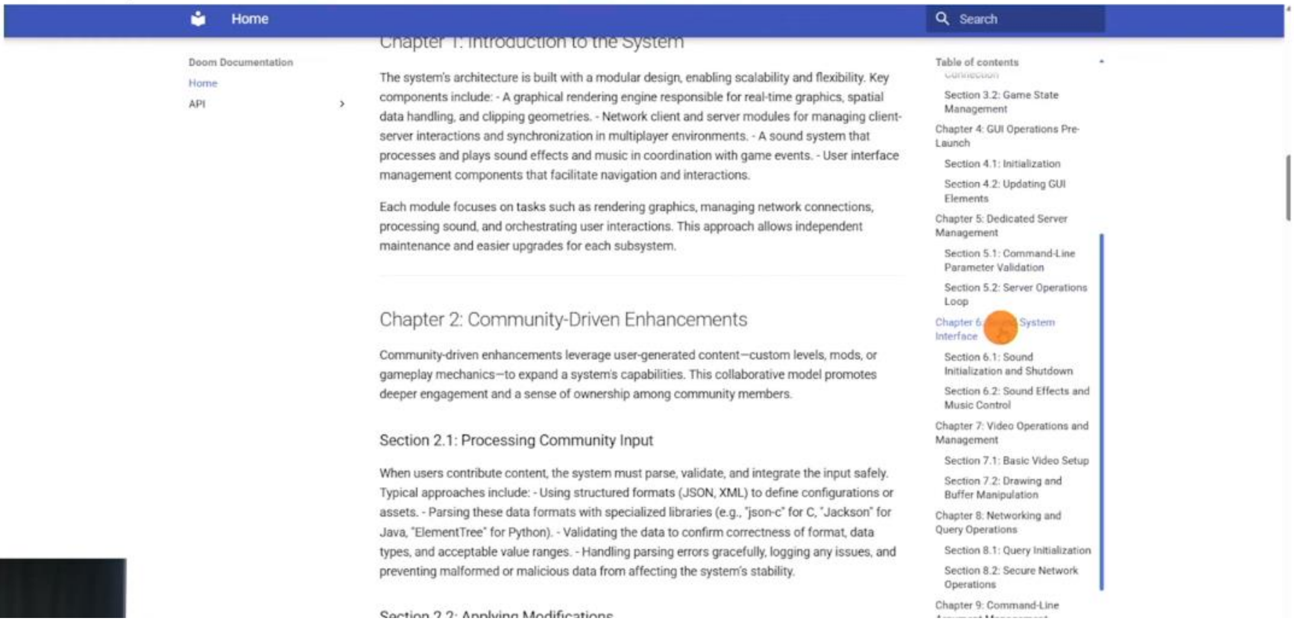
Players running the Doom Project can expect a structured gaming experience involving: - Game initialization - Demo playback - Level transitions - Single-player and multiplayer modes (with dynamic sound effects, music, and interactive menus)

Additional features like player spawning, game state management, real-time rendering, special effects, and a finale sequence ensure an engaging and immersive gameplay atmosphere.

Table of contents

- Table of Contents
- Introduction
  - The Doom Project
  - User Experience
- Chapter 1: Introduction to the System
- Chapter 2: Community-Driven Enhancements
  - Section 2.1: Processing Community Input
  - Section 2.2: Applying Modifications
- Chapter 3: Network Client Functionality
  - Section 3.1: Initialization and Connection
  - Section 3.2: Game State Management
- Chapter 4: GUI Operations Pre-Launch
  - Section 4.1: Initialization
  - Section 4.2: Updating GUI Elements

GraphRAG for Code - Automated Documentation



Home

Search

Doom Documentation

Home

API

Chapter 1: Introduction to the System

The system's architecture is built with a modular design, enabling scalability and flexibility. Key components include: - A graphical rendering engine responsible for real-time graphics, spatial data handling, and clipping geometries. - Network client and server modules for managing client-server interactions and synchronization in multiplayer environments. - A sound system that processes and plays sound effects and music in coordination with game events. - User interface management components that facilitate navigation and interactions.

Each module focuses on tasks such as rendering graphics, managing network connections, processing sound, and orchestrating user interactions. This approach allows independent maintenance and easier upgrades for each subsystem.

Chapter 2: Community-Driven Enhancements

Community-driven enhancements leverage user-generated content—custom levels, mods, or gameplay mechanics—to expand a system's capabilities. This collaborative model promotes deeper engagement and a sense of ownership among community members.

Section 2.1: Processing Community Input

When users contribute content, the system must parse, validate, and integrate the input safely. Typical approaches include: - Using structured formats (JSON, XML) to define configurations or assets. - Parsing these data formats with specialized libraries (e.g., "json-c" for C, "Jackson" for Java, "ElementTree" for Python). - Validating the data to confirm correctness of format, data types, and acceptable value ranges. - Handling parsing errors gracefully, logging any issues, and preventing malformed or malicious data from affecting the system's stability.

Section 2.2: Applying Modifications

Table of contents

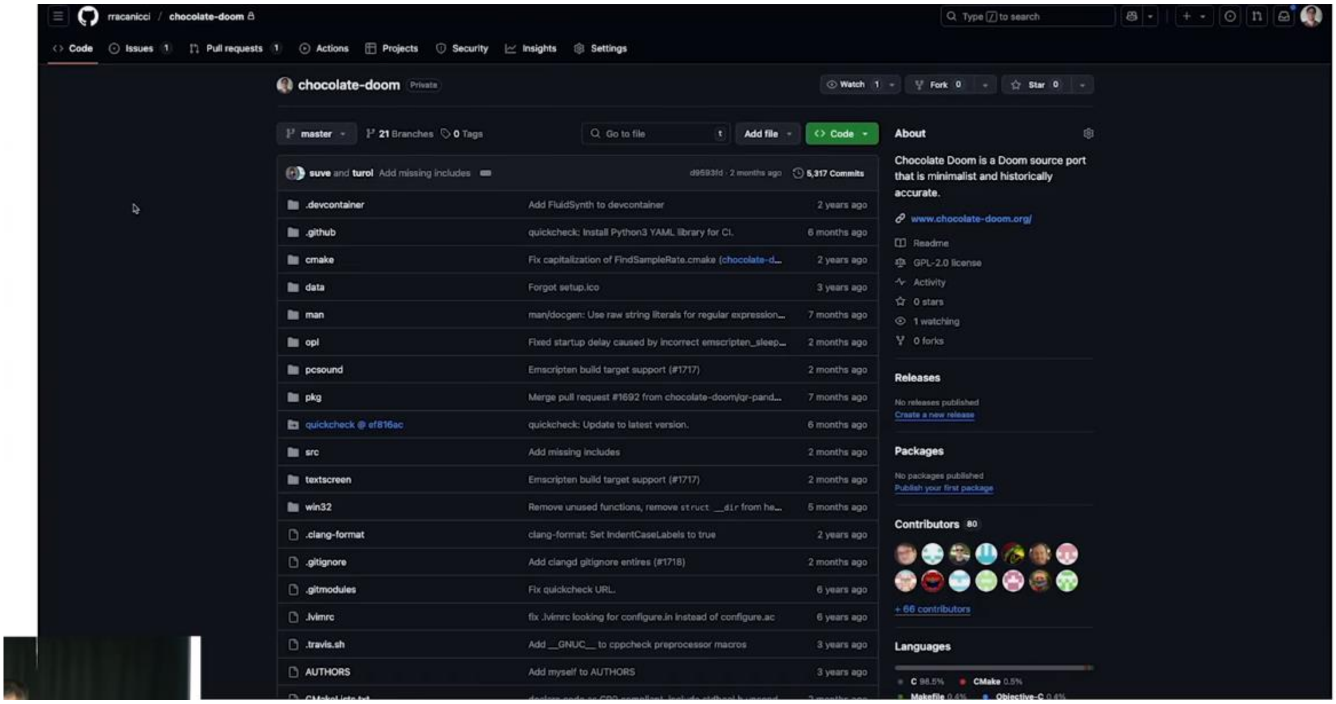
- Section 3.2: Game State Management
- Chapter 4: GUI Operations Pre-Launch
  - Section 4.1: Initialization
  - Section 4.2: Updating GUI Elements
- Chapter 5: Dedicated Server Management
  - Section 5.1: Command-Line Parameter Validation
  - Section 5.2: Server Operations Loop
- Chapter 6: [illegible] System Interface
  - Section 6.1: Sound Initialization and Shutdown
  - Section 6.2: Sound Effects and Music Control
- Chapter 7: Video Operations and Management
  - Section 7.1: Basic Video Setup
  - Section 7.2: Drawing and Buffer Manipulation
- Chapter 8: Networking and Query Operations
  - Section 8.1: Query Initialization
  - Section 8.2: Secure Network Operations
- Chapter 9: Command-Line

Code Feature Development

Feature: Add ability for players to jump in Doom

- Existing agents fail on complex tasks
  - Multi-file modifications cause problems
  - Lack of understanding of how everything fits together

This is where **GraphRAG** can actually help



chocolate-doom

suve and turol Add missing includes · d9593fd · 2 months ago · 5,317 Commits

.devcontainer	Add FluidSynth to devcontainer	2 years ago
.github	quickcheck: Install Python3 YAML library for CI.	6 months ago
cmake	Fix capitalization of FindSampleRate.cmake (chocolate-d…	2 years ago
data	Forgot setup.ico	3 years ago
man	man/docgen: Use raw string literals for regular expression…	7 months ago
opl	Fixed startup delay caused by incorrect emscripten_sleep…	2 months ago
pcsound	Emscripten build target support (#1717)	2 months ago
pkg	Merge pull request #1692 from chocolate-doom/qr-pand…	7 months ago
quickcheck @ ef816ac	quickcheck: Update to latest version.	6 months ago
src	Add missing includes	2 months ago
textscreen	Emscripten build target support (#1717)	2 months ago
win32	Remove unused functions, remove `struct __dir` from he…	5 months ago
.clang-format	clang-format: Set IndentCaseLabels to true	2 years ago
.gitignore	Add clangd gitignore entires (#1718)	2 months ago
.gitmodules	Fix quickcheck URL.	6 years ago
.lvimrc	fix .lvimrc looking for configure.in instead of configure.ac	6 years ago
.travis.sh	Add __GNUC__ to cppcheck preprocessor macros	3 years ago
AUTHORS	Add myself to AUTHORS	3 years ago

About

Chocolate Doom is a Doom source port that is minimalist and historically accurate.

www.chocolate-doom.org/

Readme

GPL-2.0 license

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

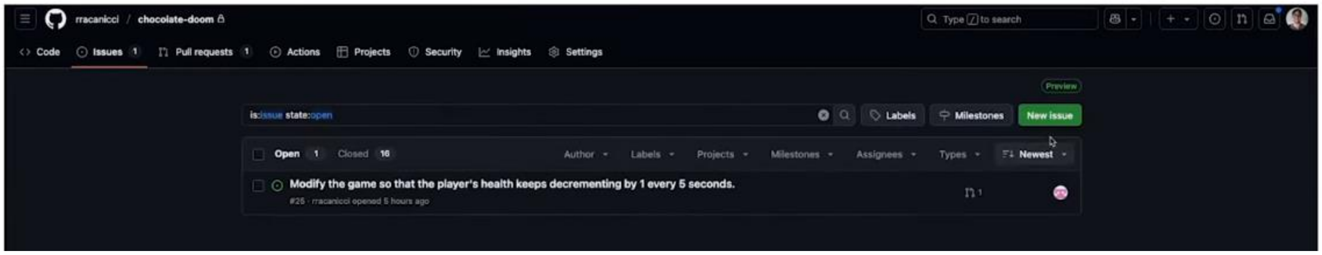
Contributors 80

+ 66 contributors

Languages

C 98.5% CMake 0.5%

Makefile 0.4% Objective-C 0.4%

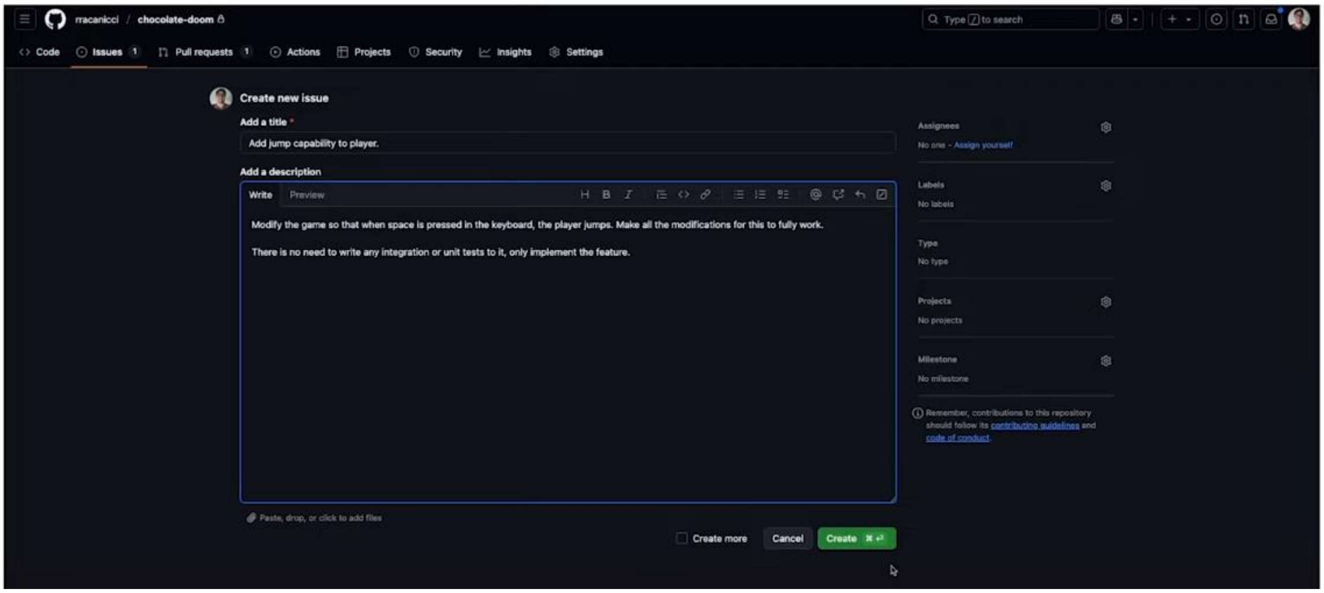


is:issue state:open

Open 1 Closed 16

Modify the game so that the player's health keeps decrementing by 1 every 5 seconds.

#25 · rracanicci opened 5 hours ago



Create new issue

Add a title *

Add jump capability to player.

Add a description

Modify the game so that when space is pressed in the keyboard, the player jumps. Make all the modifications for this to fully work.

There is no need to write any integration or unit tests to it, only implement the feature.

Assignees: No one - Assign yourself

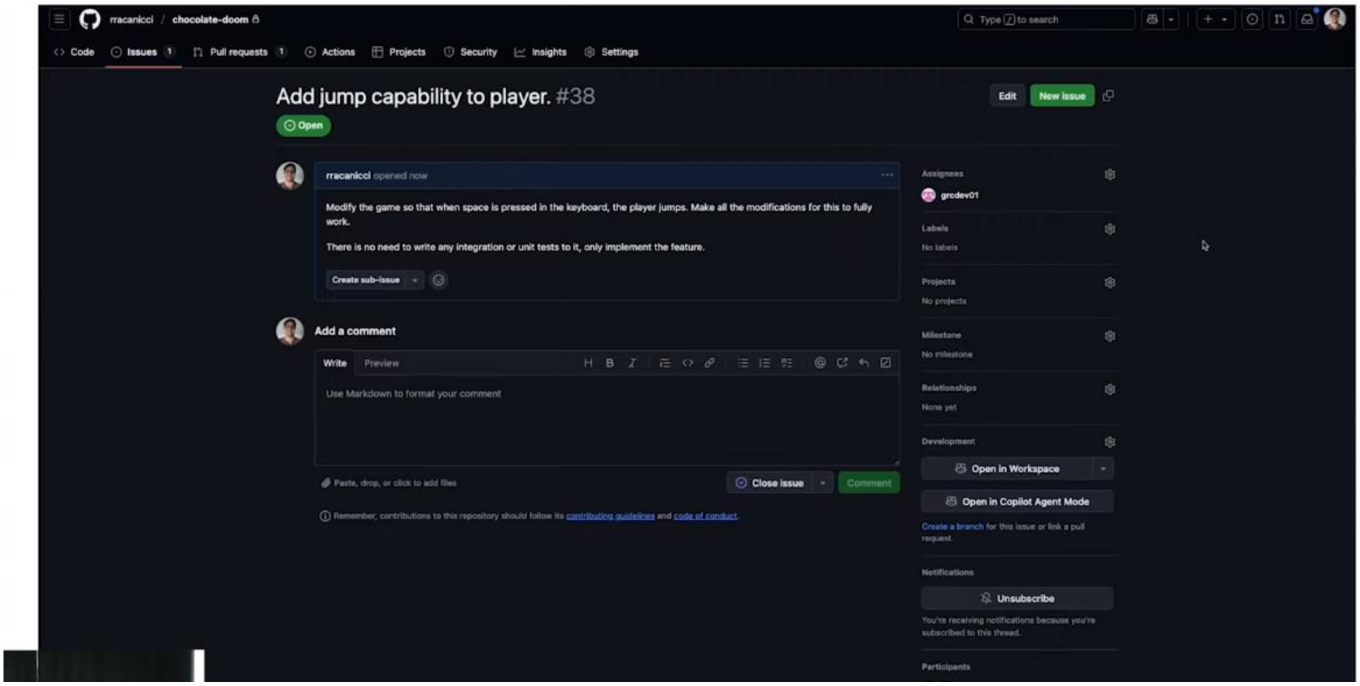
Labels: No labels

Type: No type

Projects: No projects

Milestone: No milestone

Remember, contributions to this repository should follow its contributing guidelines and code of conduct.



Add jump capability to player. #38

Open

rracanicci opened now

Modify the game so that when space is pressed in the keyboard, the player jumps. Make all the modifications for this to fully work.

There is no need to write any integration or unit tests to it, only implement the feature.

Add a comment

Remember, contributions to this repository should follow its contributing guidelines and code of conduct.

Assignees: grcdev01

Labels: No labels

Projects: No projects

Milestone: No milestone

Relationships: None yet

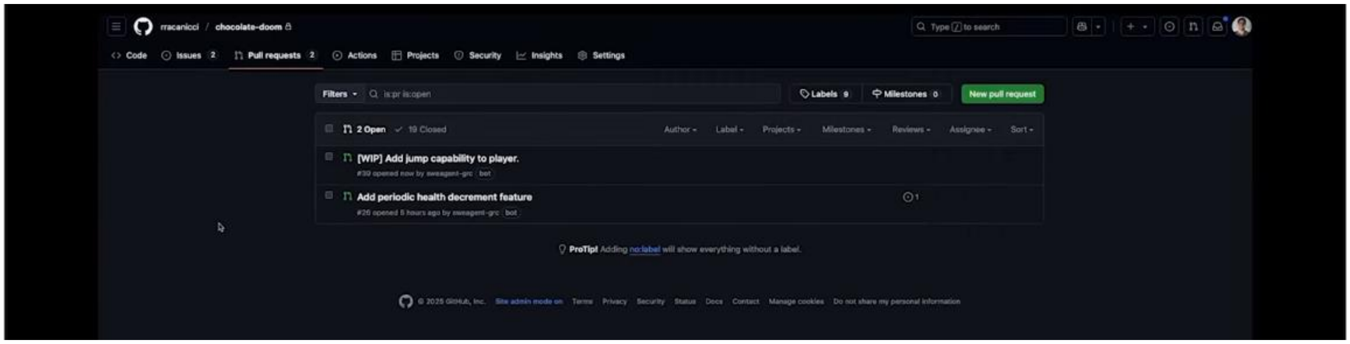
Development

Create a branch for this issue or link a pull request.

Notifications

You're receiving notifications because you're subscribed to this thread.

Participants

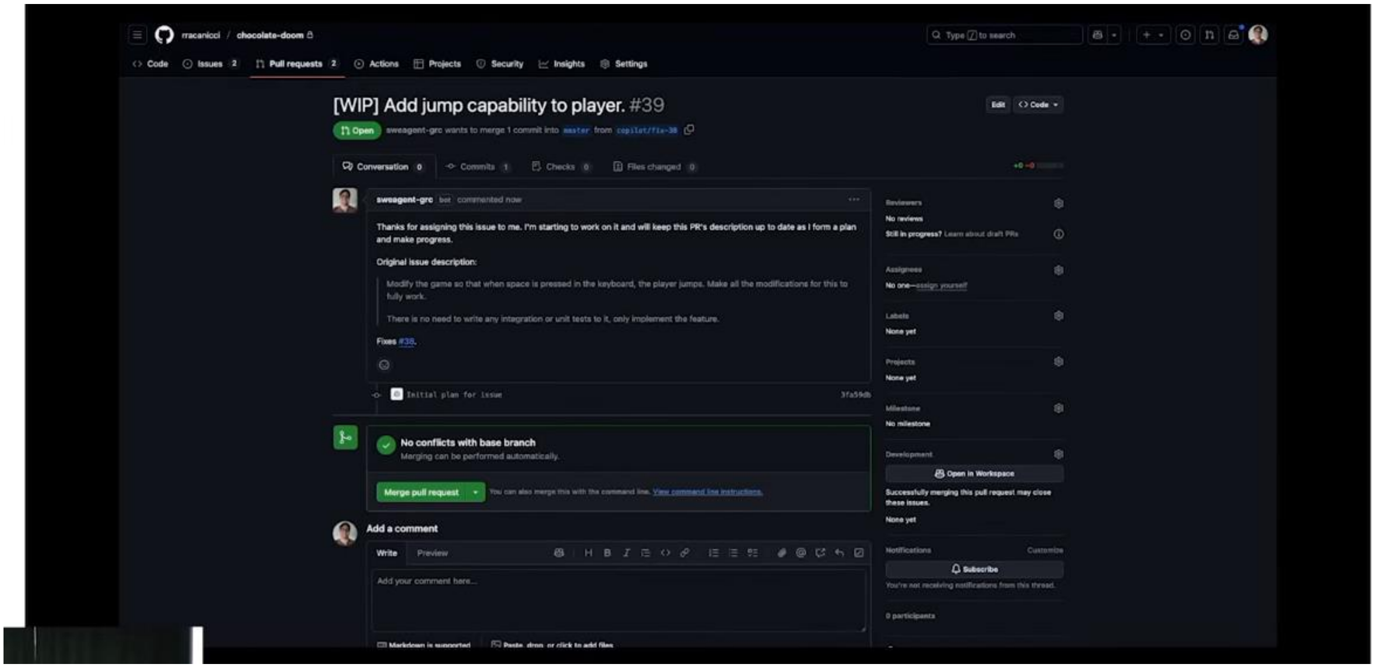


☐ [WIP] Add jump capability to player.
#39 opened now by sweagent-grc bot

☐ Add periodic health decrement feature
#26 opened 5 hours ago by sweagent-grc bot

ProTip! Adding no:label will show everything without a label.

© 2025 GitHub, Inc. Site admin mode on Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information



[WIP] Add jump capability to player. #39

Open sweagent-grc wants to merge 1 commit into `master` from `copilot/fix-38`

sweagent-grc bot commented now

Thanks for assigning this issue to me. I'm starting to work on it and will keep this PR's description up to date as I form a plan and make progress.

Original issue description:

> Modify the game so that when space is pressed in the keyboard, the player jumps. Make all the modifications for this to fully work.
>
> There is no need to write any integration or unit tests to it, only implement the feature.

Fixes #38.

`Initial plan for issue` 3fa59db

No conflicts with base branch
Merging can be performed automatically.

Merge pull request — You can also merge this with the command line. View command line instructions.

Add a comment

Reviewers
No reviews
Still in progress? Learn about draft PRs

Assignees
No one—assign yourself

Labels
None yet

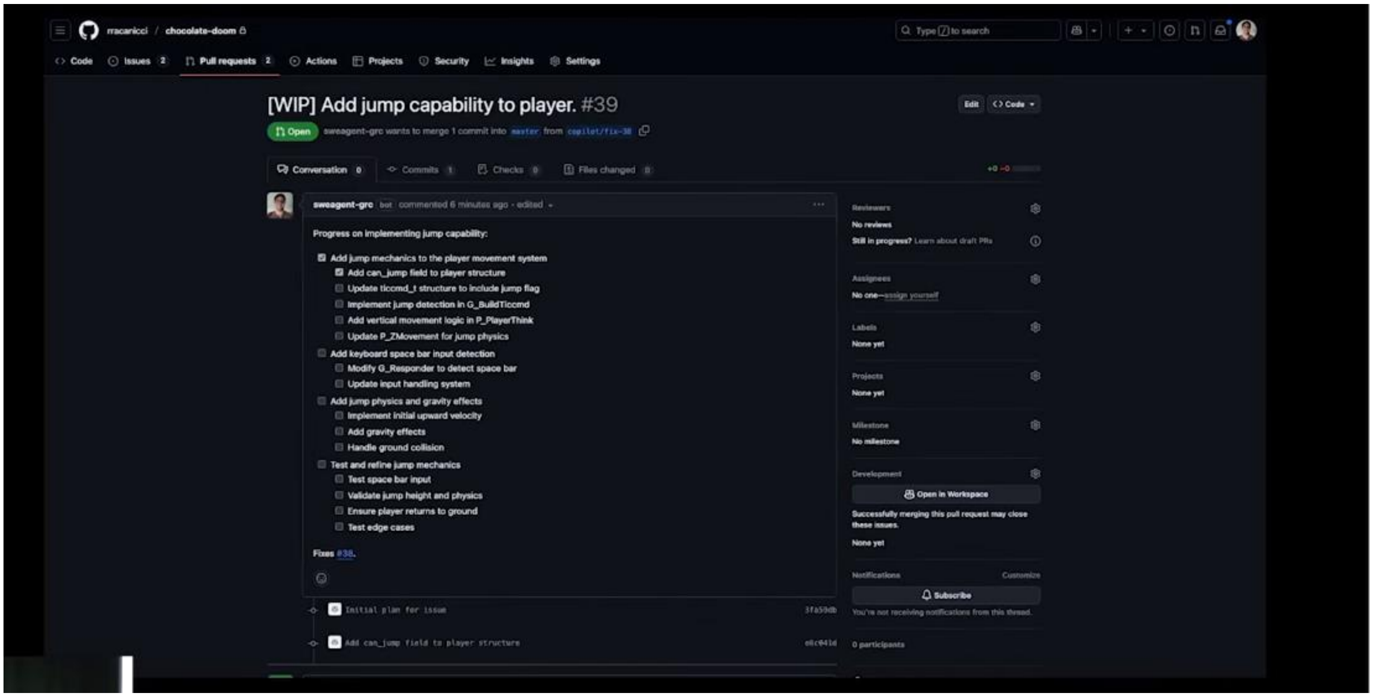
Projects
None yet

Milestone
No milestone

Development
Open in Workspace
Successfully merging this pull request may close these issues.
None yet

Notifications Customize
Subscribe
You're not receiving notifications from this thread.

0 participants



[WIP] Add jump capability to player. #39

Open sweagent-grc wants to merge 1 commit into `master` from `copilot/fix-38`

sweagent-grc bot commented 6 minutes ago · edited

Progress on implementing jump capability:

- ☑ Add jump mechanics to the player movement system
  - ☑ Add can_jump field to player structure
  - ☐ Update ticcmd_t structure to include jump flag
  - ☐ Implement jump detection in G_BuildTiccmd
  - ☐ Add vertical movement logic in P_PlayerThink
  - ☐ Update P_ZMovement for jump physics
- ☐ Add keyboard space bar input detection
  - ☐ Modify G_Responder to detect space bar
  - ☐ Update input handling system
- ☐ Add jump physics and gravity effects
  - ☐ Implement initial upward velocity
  - ☐ Add gravity effects
  - ☐ Handle ground collision
- ☐ Test and refine jump mechanics
  - ☐ Test space bar input
  - ☐ Validate jump height and physics
  - ☐ Ensure player returns to ground
  - ☐ Test edge cases

Fixes #38.

`Initial plan for issue` 3fa59db

`Add can_jump field to player structure` e6c041d

Reviewers
No reviews
Still in progress? Learn about draft PRs

Assignees
No one—assign yourself

Labels
None yet

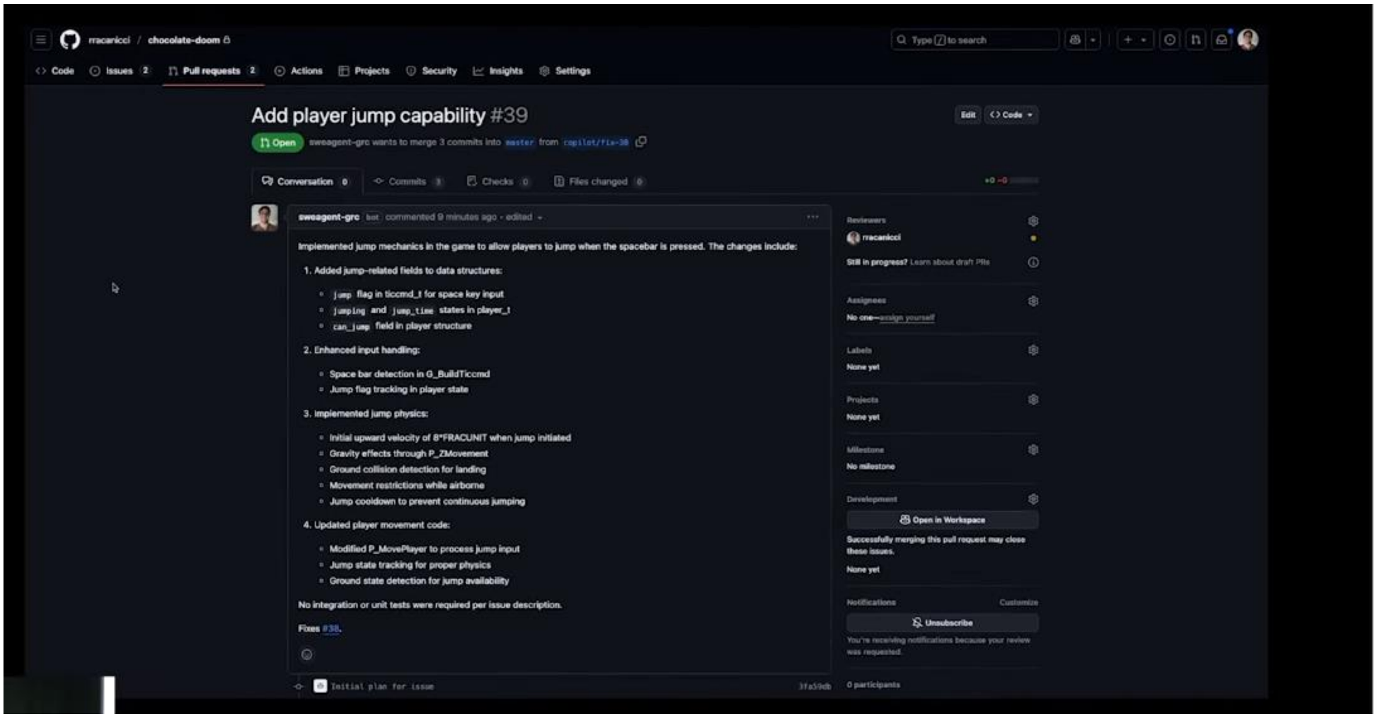
Projects
None yet

Milestone
No milestone

Development
Open in Workspace
Successfully merging this pull request may close these issues.
None yet

Notifications Customize
Subscribe
You're not receiving notifications from this thread.

0 participants



Add player jump capability #39

sweagent-grc wants to merge 3 commits into `master` from `copilot/fix-38`

sweagent-grc bot commented 9 minutes ago · edited

Implemented jump mechanics in the game to allow players to jump when the spacebar is pressed. The changes include:

1. Added jump-related fields to data structures:
    - `jump` flag in ticcmd_t for space key input
    - `jumping` and `jump_time` states in player_t
    - `can_jump` field in player structure
2. Enhanced input handling:
    - Space bar detection in G_BuildTiccmd
    - Jump flag tracking in player state
3. Implemented jump physics:
    - Initial upward velocity of 8*FRACUNIT when jump initiated
    - Gravity effects through P_ZMovement
    - Ground collision detection for landing
    - Movement restrictions while airborne
    - Jump cooldown to prevent continuous jumping
4. Updated player movement code:
    - Modified P_MovePlayer to process jump input
    - Jump state tracking for proper physics
    - Ground state detection for jump availability

No integration or unit tests were required per issue description.

Fixes #38.

`Initial plan for issue` 3fa59db

Reviewers: rracanicci

Still in progress? Learn about draft PRs

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Open in Workspace

Successfully merging this pull request may close these issues.

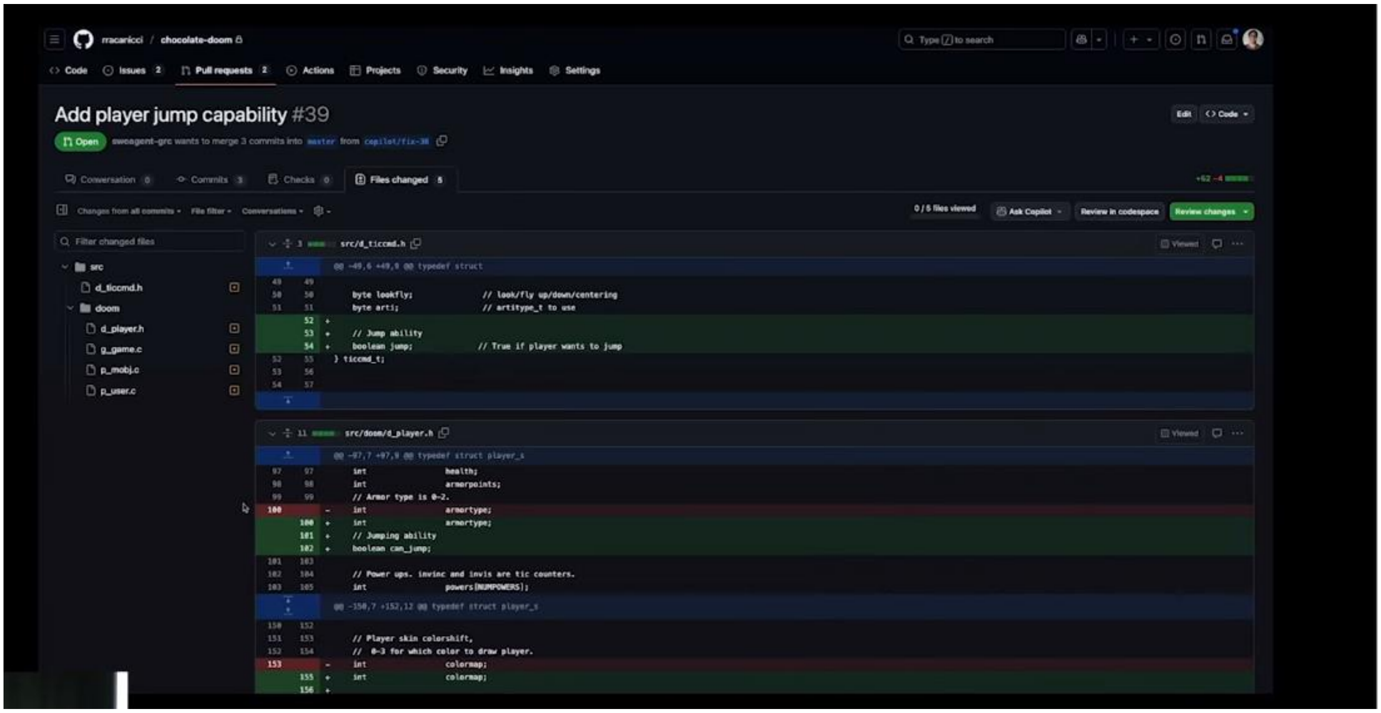
None yet

Notifications: Customize

Unsubscribe

You're receiving notifications because your review was requested.

0 participants



Add player jump capability #39

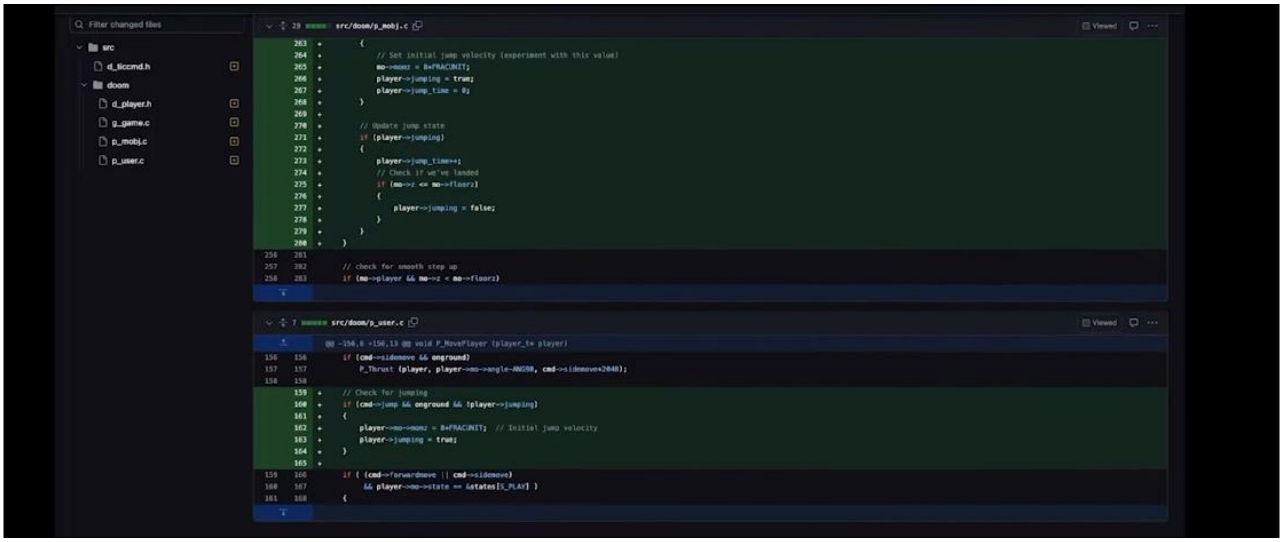
sweagent-grc wants to merge 3 commits into `master` from `copilot/fix-38`

3 src/d_ticcmd.h

```
@@ -49,6 +49,9 @@ typedef struct

     byte lookfly;            // look/fly up/down/centering
     byte arti;               // artitype_t to use
+
+    // Jump ability
+    boolean jump;            // True if player wants to jump
 } ticcmd_t;


```

11 src/doom/d_player.h

```
@@ -97,7 +97,9 @@ typedef struct player_s
     int                  health;
     int                  armorpoints;
     // Armor type is 0-2.
-    int                  armortype;
+    int                  armortype;
+    // Jumping ability
+    boolean can_jump;

     // Power ups. invinc and invis are tic counters.
     int                  powers[NUMPOWERS];
@@ -150,7 +152,12 @@ typedef struct player_s

     // Player skin colorshift,
     //  0-3 for which color to draw player.
-    int                  colormap;
+    int                  colormap;
+
```

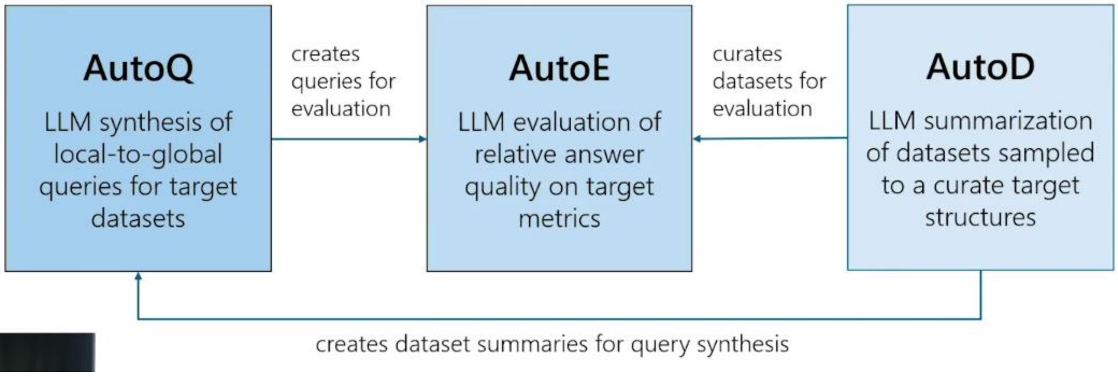


```
+            {
+                // Set initial jump velocity (experiment with this value)
+                mo->momz = 8*FRACUNIT;
+                player->jumping = true;
+                player->jump_time = 0;
+            }
+
+            // Update jump state
+            if (player->jumping)
+            {
+                player->jump_time++;
+                // Check if we've landed
+                if (mo->z <= mo->floorz)
+                {
+                    player->jumping = false;
+                }
+            }
+        }

        // check for smooth step up
        if (mo->player && mo->z < mo->floorz)
```

```
@@ -156,6 +156,13 @@ void P_MovePlayer (player_t* player)
        if (cmd->sidemove && onground)
            P_Thrust (player, player->mo->angle-ANG90, cmd->sidemove*2048);

+       // Check for jumping
+       if (cmd->jump && onground && !player->jumping)
+       {
+           player->mo->momz = 8*FRACUNIT;  // Initial jump velocity
+           player->jumping = true;
+       }
+
        if ( (cmd->forwardmove || cmd->sidemove)
             && player->mo->state == &states[S_PLAY] )
        {
```

BenchmarkQED: Automated RAG Benchmarking

Available now at https://github.com/microsoft/benchmark-qed



AutoQ

LLM synthesis of local-to-global queries for target datasets

creates queries for evaluation

AutoE

LLM evaluation of relative answer quality on target metrics

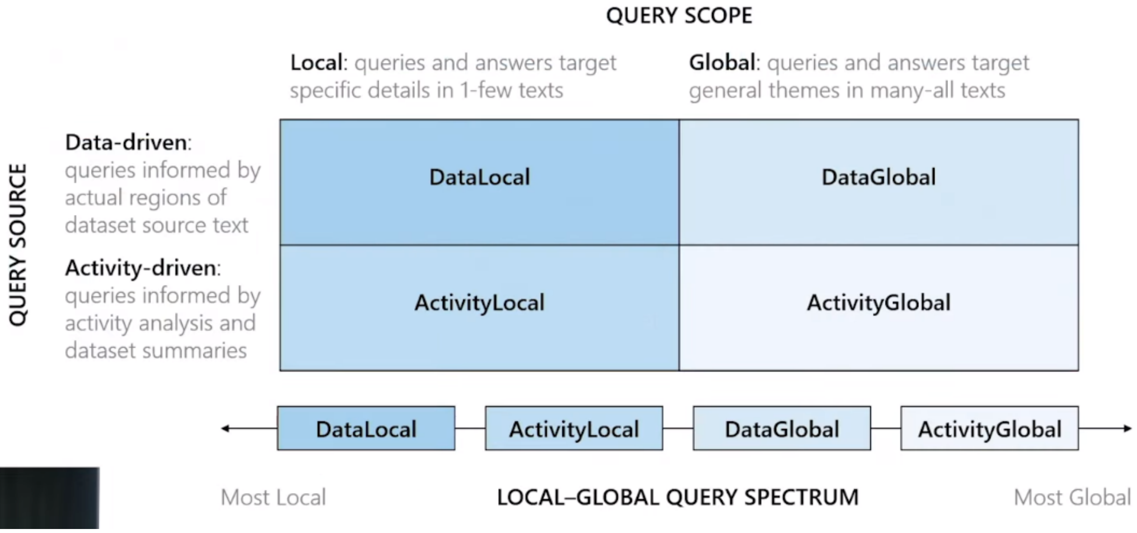
curates datasets for evaluation

AutoD

LLM summarization of datasets sampled to a curate target structures

creates dataset summaries for query synthesis

AutoQ: Synthetic queries with variable data scope



QUERY SCOPE

Local: queries and answers target specific details in 1-few texts

Global: queries and answers target general themes in many-all texts

QUERY SOURCE

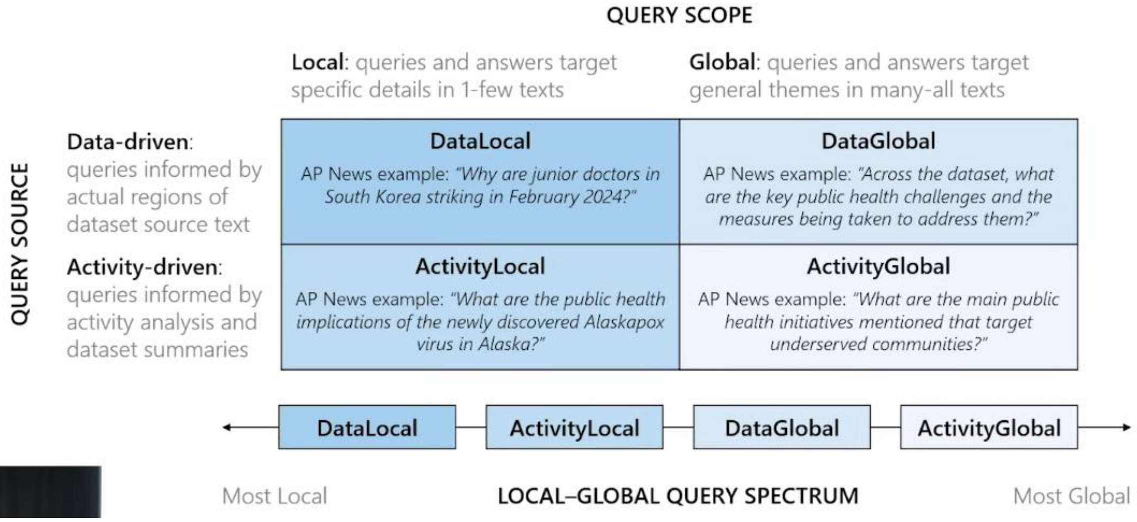
	Local	Global
Data-driven: queries informed by actual regions of dataset source text	DataLocal	DataGlobal
Activity-driven: queries informed by activity analysis and dataset summaries	ActivityLocal	ActivityGlobal

DataLocal — ActivityLocal — DataGlobal — ActivityGlobal

[redacted]

Most Local LOCAL–GLOBAL QUERY SPECTRUM Most Global

AutoQ: Synthetic queries with variable data scope



QUERY SCOPE

Local: queries and answers target specific details in 1-few texts

Global: queries and answers target general themes in many-all texts

QUERY SOURCE

	Local	Global
Data-driven: queries informed by actual regions of dataset source text	DataLocal AP News example: *"Why are junior doctors in South Korea striking in February 2024?"*	DataGlobal AP News example: *"Across the dataset, what are the key public health challenges and the measures being taken to address them?"*
Activity-driven: queries informed by activity analysis and dataset summaries	ActivityLocal AP News example: *"What are the public health implications of the newly discovered Alaskapox virus in Alaska?"*	ActivityGlobal AP News example: *"What are the main public health initiatives mentioned that target underserved communities?"*

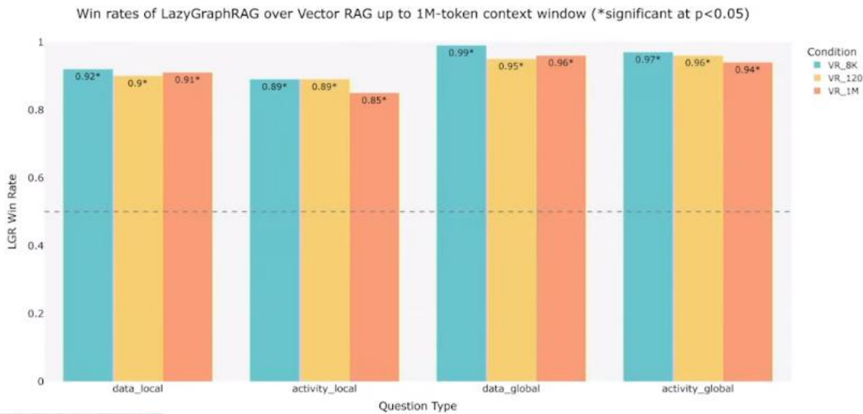
DataLocal — ActivityLocal — DataGlobal — ActivityGlobal

[redacted]

Most Local LOCAL–GLOBAL QUERY SPECTRUM Most Global

AutoE: Evaluation by query class & quality metric

Composite score of (Comprehensiveness, Diversity, Empowerment, and Relevance)



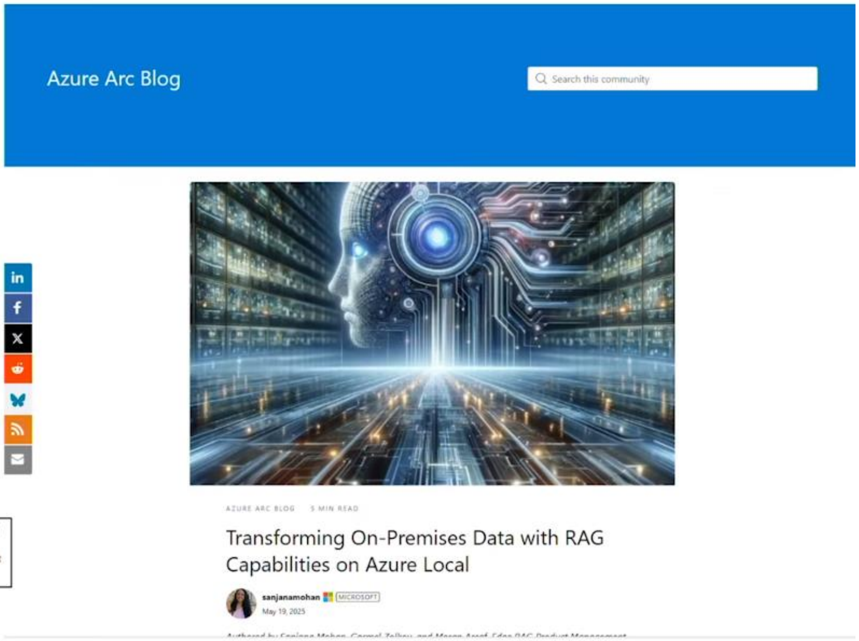
[redacted] above the 50% line represent a win for LazyGraphRAG (LGR).
· the score, the more LGR wins.

Evaluation insights

LazyGraphRAG significantly beat *VectorRAG*

- *long context* (1M tokens) makes little difference to VectorRAG performance
- *LazyGraphRAG* excels for queries spanning the local–global spectrum
- *LazyGraphRAG* was ~10% the cost of 1M token Vector RAG

Azure Local



AZURE ARC BLOG · 5 MIN READ

Transforming On-Premises Data with RAG Capabilities on Azure Local

sanjanamohan MICROSOFT
May 19, 2025

Discovering the Advanced Capabilities of RAG on Azure Local

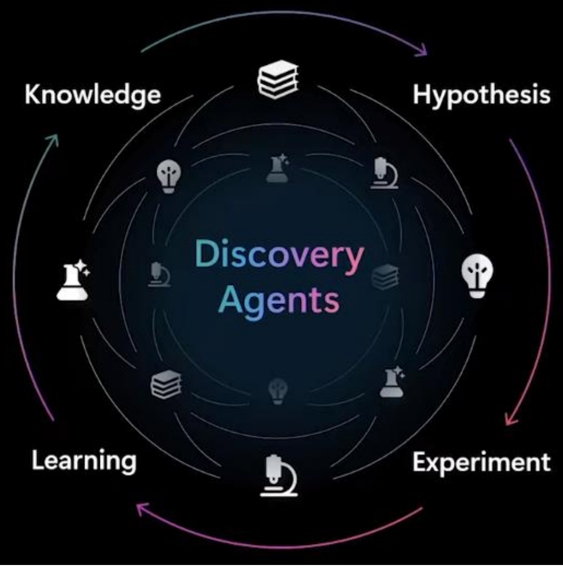
- Support for **Hybrid Search**, and soon **Lazy Graph RAG**, allowing robust, fast, low-cost indexing and providing quality and relevant answers regardless of query type.

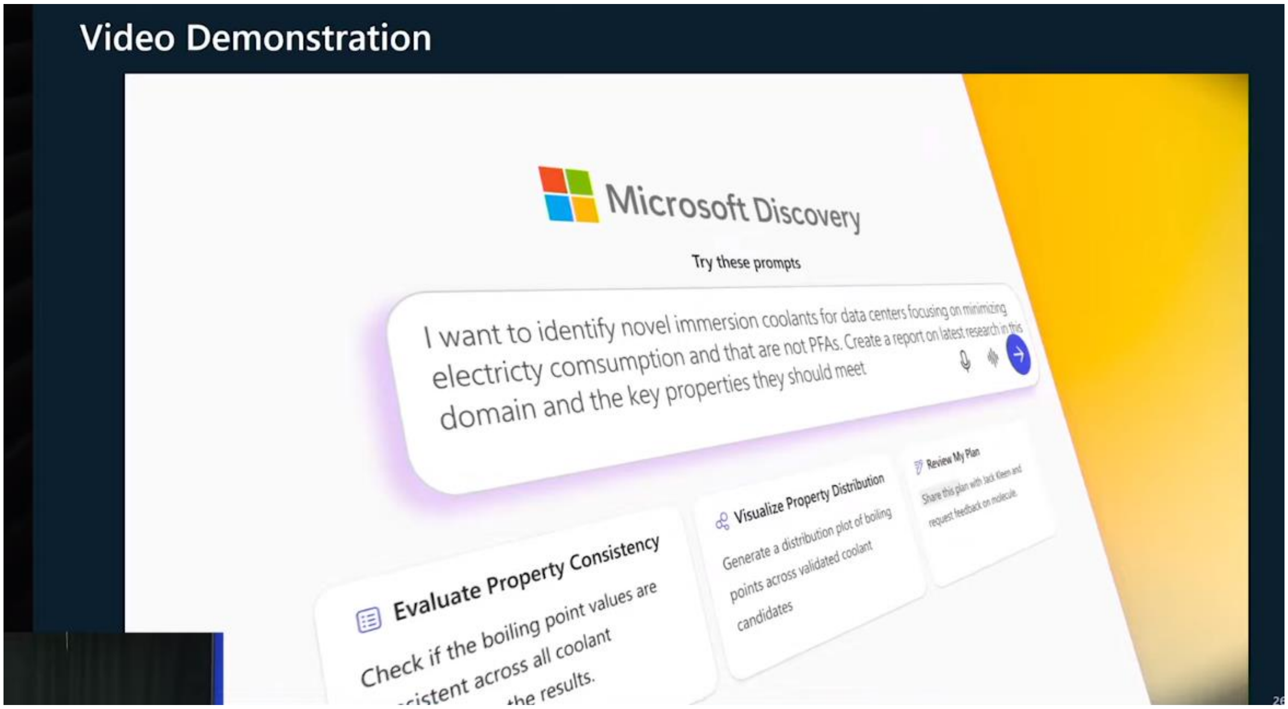
Announcing

Microsoft Discovery

Graph-based scientific co-reasoning	Specialized discovery agents for conducting research	Extensible and enterprise-ready

Transforming the scientific method with Agentic AI





Video Demonstration

Microsoft Discovery

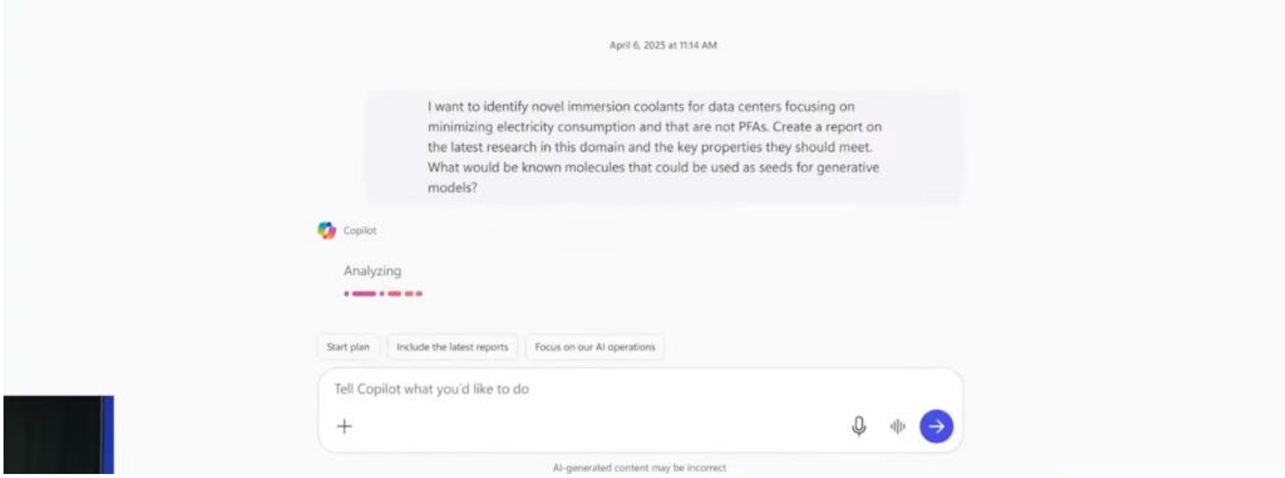
Try these prompts

I want to identify novel immersion coolants for data centers focusing on minimizing electricty comsumption and that are not PFAs. Create a report on latest research in this domain and the key properties they should meet

Evaluate Property Consistency
Check if the boiling point values are
cistent across all coolant
the results.

Visualize Property Distribution
Generate a distribution plot of boiling points across validated coolant candidates

Review My Plan
Share this plan with Jack Kleen and request feedback on molecule.



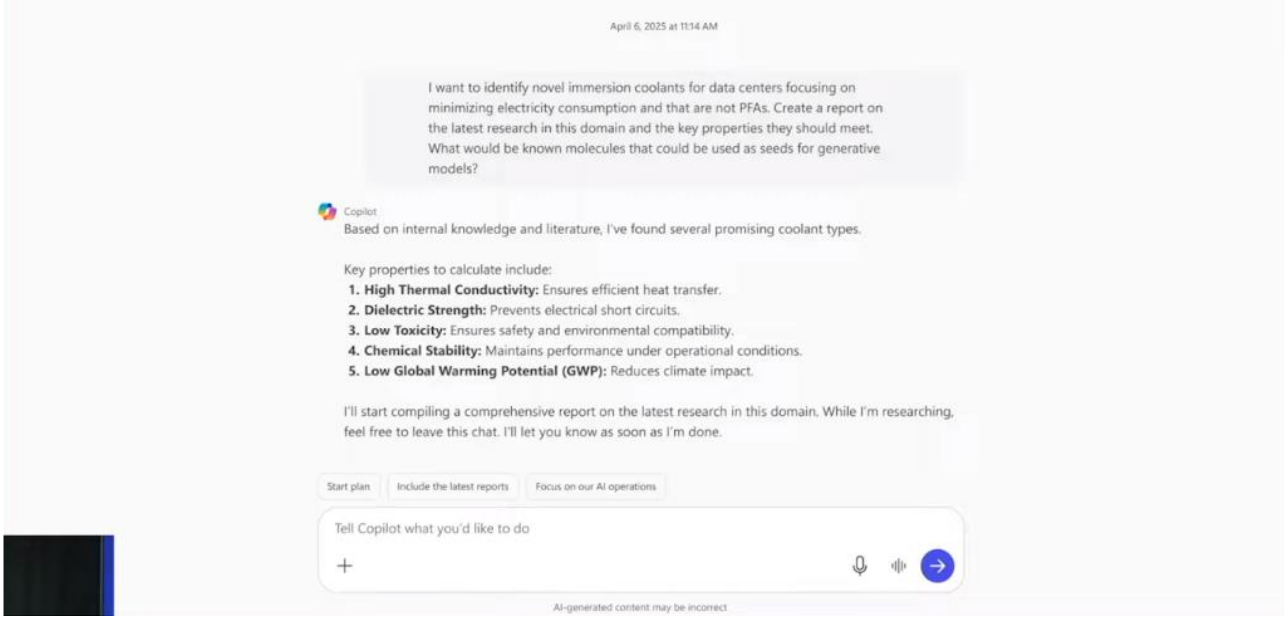
April 6, 2025 at 11:14 AM

I want to identify novel immersion coolants for data centers focusing on minimizing electricity consumption and that are not PFAs. Create a report on the latest research in this domain and the key properties they should meet. What would be known molecules that could be used as seeds for generative models?

Copilot

Analyzing

Start plan
Include the latest reports
Focus on our AI operations
Tell Copilot what you'd like to do
+
AI-generated content may be incorrect



April 6, 2025 at 11:14 AM

I want to identify novel immersion coolants for data centers focusing on minimizing electricity consumption and that are not PFAs. Create a report on the latest research in this domain and the key properties they should meet. What would be known molecules that could be used as seeds for generative models?

Copilot

Based on internal knowledge and literature, I've found several promising coolant types.

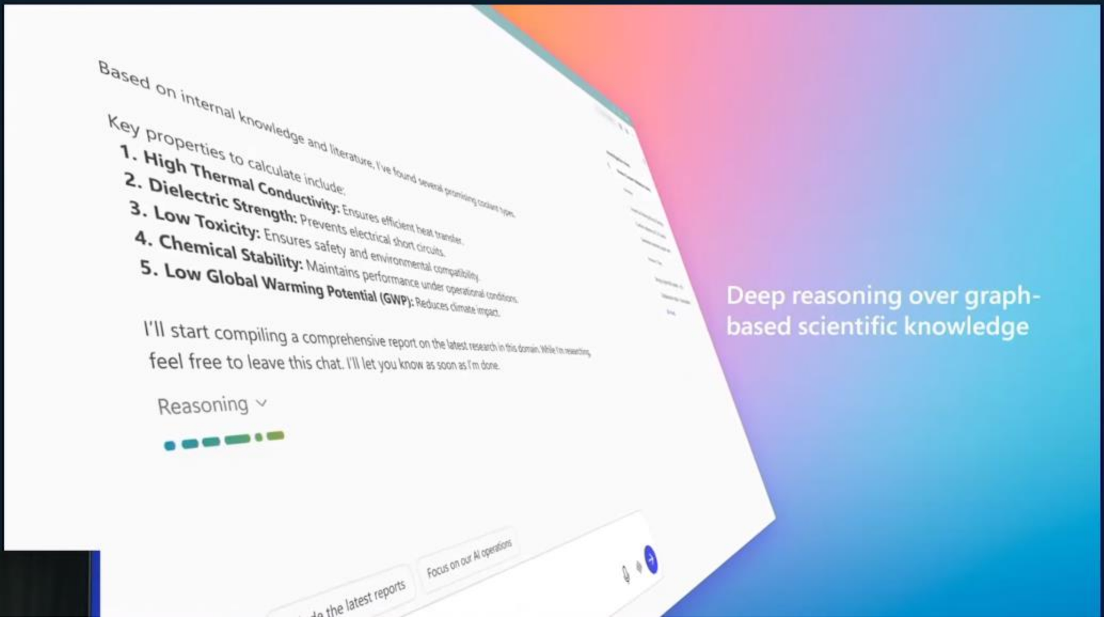
Key properties to calculate include:

1. **High Thermal Conductivity:** Ensures efficient heat transfer.
2. **Dielectric Strength:** Prevents electrical short circuits.
3. **Low Toxicity:** Ensures safety and environmental compatibility.
4. **Chemical Stability:** Maintains performance under operational conditions.
5. **Low Global Warming Potential (GWP):** Reduces climate impact.

I'll start compiling a comprehensive report on the latest research in this domain. While I'm researching, feel free to leave this chat. I'll let you know as soon as I'm done.

Start plan
Include the latest reports
Focus on our AI operations
Tell Copilot what you'd like to do
+
AI-generated content may be incorrect

Video Demonstration



Based on internal knowledge and literature, I've found several promising coolant types.

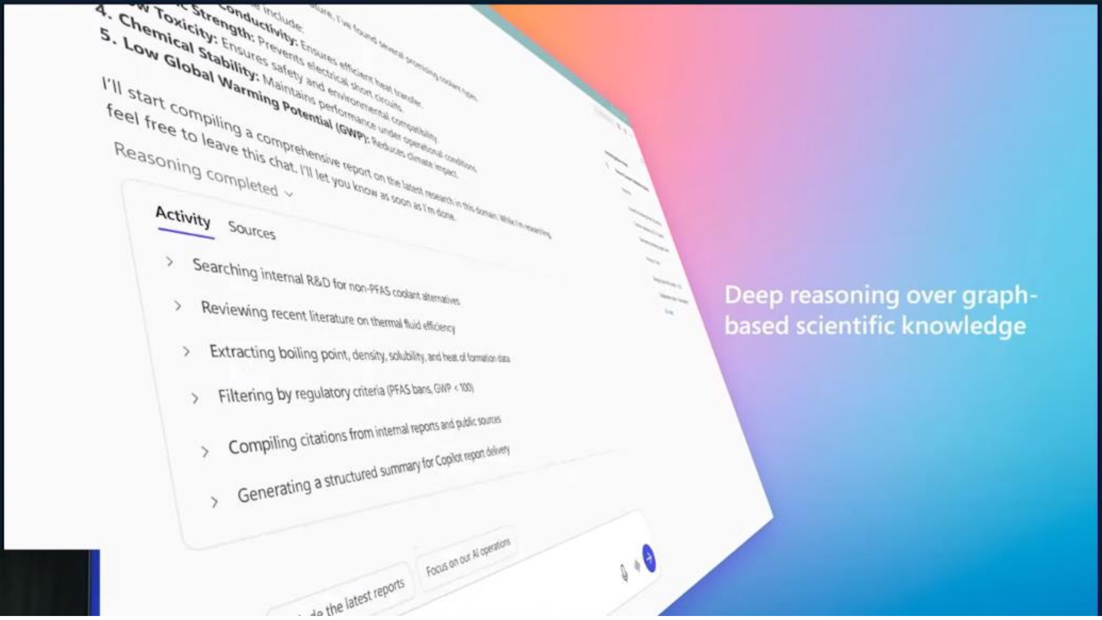
Key properties to calculate include:

1. **High Thermal Conductivity:** Ensures efficient heat transfer.
2. **Dielectric Strength:** Prevents electrical short circuits.
3. **Low Toxicity:** Ensures safety and environmental compatibility.
4. **Chemical Stability:** Maintains performance under operational conditions.
5. **Low Global Warming Potential (GWP):** Reduces climate impact.

I'll start compiling a comprehensive report on the latest research in this domain. While I'm researching, feel free to leave this chat. I'll let you know as soon as I'm done.

Deep reasoning over graph-based scientific knowledge

Video Demonstration



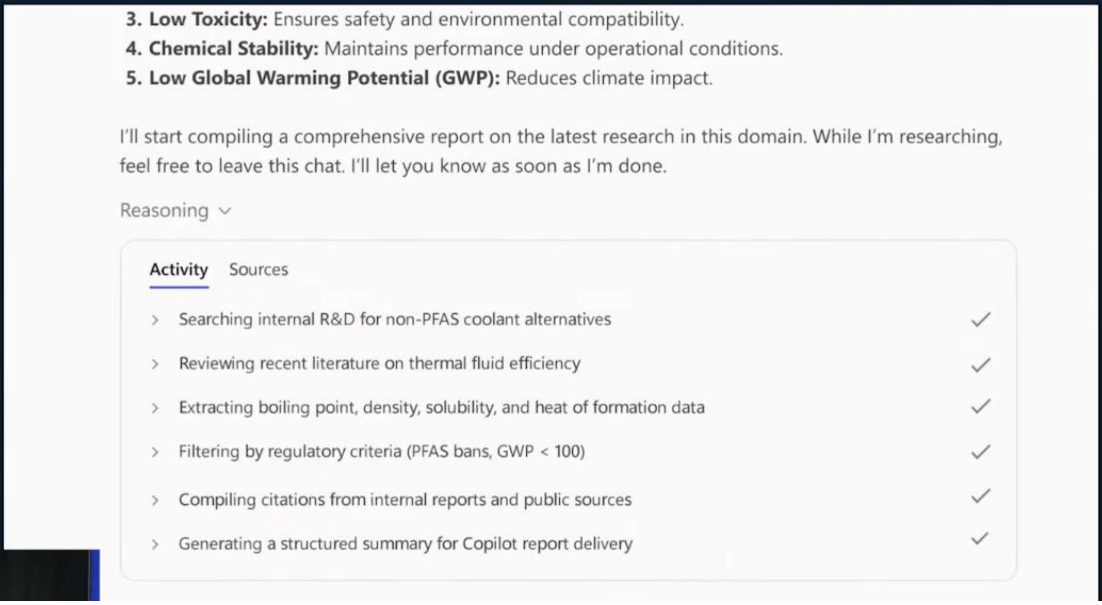
4. **Chemical Stability:** Maintains performance under operational conditions.
5. **Low Global Warming Potential (GWP):** Reduces climate impact.

I'll start compiling a comprehensive report on the latest research in this domain. While I'm researching, feel free to leave this chat. I'll let you know as soon as I'm done.

- › Searching internal R&D for non-PFAS coolant alternatives
- › Reviewing recent literature on thermal fluid efficiency
- › Extracting boiling point, density, solubility, and heat of formation data
- › Filtering by regulatory criteria (PFAS bans, GWP < 100)
- › Compiling citations from internal reports and public sources
- › Generating a structured summary for Copilot report delivery

Deep reasoning over graph-based scientific knowledge

Video Demonstration



3. **Low Toxicity:** Ensures safety and environmental compatibility.
4. **Chemical Stability:** Maintains performance under operational conditions.
5. **Low Global Warming Potential (GWP):** Reduces climate impact.

I'll start compiling a comprehensive report on the latest research in this domain. While I'm researching, feel free to leave this chat. I'll let you know as soon as I'm done.

- Searching internal R&D for non-PFAS coolant alternatives ✓
- Reviewing recent literature on thermal fluid efficiency ✓
- Extracting boiling point, density, solubility, and heat of formation data ✓
- Filtering by regulatory criteria (PFAS bans, GWP < 100) ✓
- Compiling citations from internal reports and public sources ✓
- Generating a structured summary for Copilot report delivery ✓

Takeaways:

1. LLM memory with structure is a key enabler

2. Agents can massively amplify this power