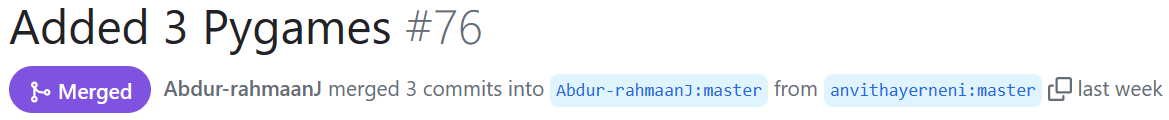
**CONTRIBUTIONS TO AN OPEN SOURCE SOFTWARE PROJECT**

**For Issues 1, 2 & 3:** [**Pull Request Link**](https://github.com/Abdur-rahmaanJ/hooman/pull/76) **,** [**Issue Link**](https://github.com/Abdur-rahmaanJ/hooman/issues/43)

The issue is to create any unique pygame using hooman library. I have created 3 unique pygames and raised request in a single pull request for all three of them

Status: **Accepted**



**Issue 1: Create a pygame using Hooman Library - Asteroid Dodger**

**Overview of the project:**

This Pygame-based game(Asteroid Dodger) features a simple yet engaging concept where the player controls a spaceship to dodge incoming asteroids. The spaceship, maneuverable left and right, must avoid collisions with randomly generated and descending asteroids of varying sizes. Each successful dodge increases the player's score, enhancing the game's challenge and engagement. The game employs straightforward graphics, with the asteroids and spaceship rendered as basic geometric shapes. The difficulty incrementally rises as more asteroids appear. The game ends when the spaceship collides with an asteroid, making quick reflexes and strategic movements key to achieving a high score in this captivating space adventure.

**Problems Faced:**

1. Understanding Pygame and Hooman Libraries: Learning the basics of these libraries and how they interact with each other and understanding how to handle user inputs, game loops, and rendering in Pygame.
2. Collision Detection & User Interface and Experience: Implementing efficient and accurate collision detection between the spaceship and asteroids and ensuring that the collision logic correctly handles different sizes of asteroids. Designing a simple yet intuitive user interface. Making the game engaging and responsive to player inputs.
3. Debugging and Testing: Identifying and fixing bugs, such as issues with collision detection or game logic.Testing the game across different scenarios to ensure stability and consistency.

**New Skills Acquired:**

1. Advanced Pygame Techniques: Gained deeper insights into Pygame’s capabilities, such as custom event handling and animation techniques. And earned hooman library functionalities.
2. Efficient Resource Management: Learned how to manage game resources like memory and processing power, especially important for dynamic object creation like asteroids.
3. Problem-Solving and Debugging: Enhanced problem-solving skills, especially in a dynamic environment like a game. Improved debugging skills for real-time applications.
4. Adaptability and Learning: Adapted to new challenges and quickly learned the nuances of game development with specific libraries.

**Issue 2: Create a pygame using Hooman Library - Ocean Cleaner**

Status: **Accepted**

**Overview of the project:**

This submarine game, made with Pygame and Hooman, was a basic start for a fun arcade game. Players controlled a submarine to dodge dangers and pick up trash. The gameplay was simple. The game looked simple with basic shapes and designs to fit the underwater theme. The game kept score and got harder as you did better.

**Problems Faced:**

1. Gameplay Balance: Adjusting the game's difficulty to be challenging but not too hard.
2. Graphics and Animation: Creating attractive and thematic graphics with limited resources.
3. Bug Fixing: Finding and fixing game bugs can be time-consuming.
4. Player Engagement: Keeping players interested and coming back to the game.
5. Cross-Platform Compatibility: Making the game work well on different platforms.

**New Skills Acquired:**

1. Game Design Principles: Acquired knowledge in game design, including level design, gameplay mechanics, and player engagement strategies.
2. Graphic Design: Developed skills in creating and manipulating game graphics, animations, and visual effects.
3. Programming and Logic: Improved programming skills, particularly in Python, with a focus on logical thinking and problem-solving.
4. Project Management: Developed project management and time management skills, balancing various aspects of game development.

**Issue 3: Create a pygame using Hooman Library - Pong game**

Status: **Accepted**

**Overview of the project:**

The Pong game I developed is a classic table tennis simulation in a digital format, using Pygame for the graphics and mechanics. Set in an 800x600 pixel window, it features a stark black background with contrasting white paddles and a ball, reflecting the original Pong's retro style. The game includes two paddles, one for the player on the right and an AI-controlled opponent on the left. Players control their paddle using the up and down arrow keys. The ball moves across the screen, bouncing off the top and bottom edges and the paddles. Scoring occurs when the ball passes a paddle, with scores displayed at the top. This simple yet engaging game encapsulates the essence of the early era of video games.

**Problems Faced:**

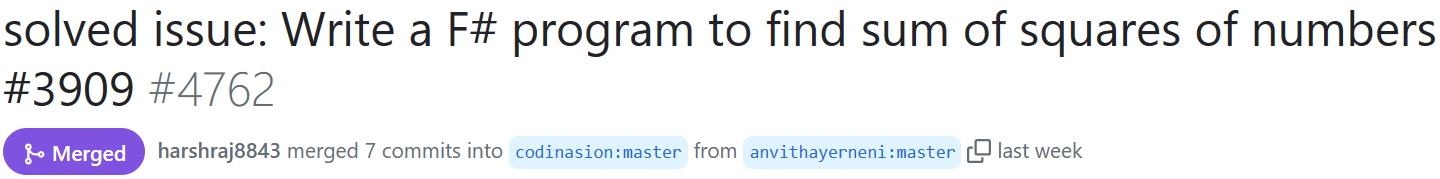
1. Balancing Gameplay: Adjusting the speed of the paddles and the ball to ensure the game was neither too easy nor too difficult was a critical challenge.
2. AI Implementation: Creating a basic AI for the opponent paddle that was challenging but not unbeatable required fine-tuning.
3. Collision Detection: Ensuring accurate collision detection between the ball and the paddles to properly reflect the ball's movement was complex.
4. User Input Handling: Implementing smooth and responsive controls for the player's paddle was essential for a good gameplay experience.
5. Scoring Logic: Developing a scoring system that accurately tracked and displayed scores without errors was crucial.

**New Skills Acquired:**

1. Collision Detection: Learned how to implement effective collision detection, a crucial aspect in games where objects interact, such as the ball bouncing off paddles.
2. Basic AI Implementation: Acquired skills in creating a simple AI for the opponent paddle, involving basic logic to react to the game's state.
3. Responsive User Input Handling: Gained experience in processing user inputs, ensuring smooth and responsive control over game elements like the player's paddle.
4. Scoring System Development: Learned how to design and implement a scoring system that updates and displays scores within the game.

**Issue 4: Write a F# program to find sum of squares of numbers**

Status: **Accepted**

****

**Overview of the project:** [**Pull Request Link**](https://github.com/codinasion/codinasion/pull/4762) **,** [**Issue Link**](https://github.com/codinasion/codinasion/issues/3909)

This F# code exemplifies functional programming by defining a `sumOfSquares` function to calculate the sum of squares of a number list. It demonstrates F#'s proficiency in list processing, user interaction, and string manipulation. The script reads user input, converts it into integers, applies the function, and displays the result, showcasing F#'s ease in handling data sequences and mathematical computations, making it a practical choice for algorithmic tasks.

**Problems Faced:**

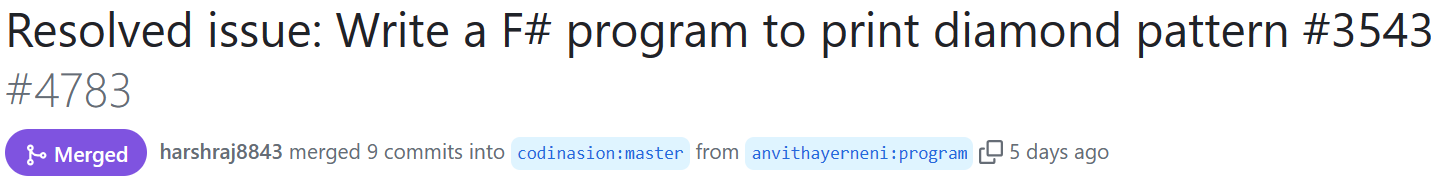
1. New Language Learning: As a new language to learn, F# presented an initial barrier with its unique syntax and functional programming principles, which were significantly different from imperative programming languages.
2. Understanding Functional Paradigms: Grasping concepts like immutability, first-class functions, and higher-order functions was challenging, requiring a shift in mindset from traditional programming approaches.
3. Debugging and Testing: Adapting to functional-specific debugging and testing methods in F# was a steep learning curve, especially understanding how to effectively test stateless functions and immutability.
4. Performance Optimization: Learning to optimize performance in F#, particularly understanding the implications of recursion, lazy evaluation, and efficient data processing, was a key area of focus.

**New Skills Acquired:**

1. Functional Programming Mastery: Developed expertise in F# features, including advanced techniques and domain modeling.
2. .NET Interoperability: Learned to integrate F# with other .NET languages and libraries.
3. Community Engagement and Reflection: Enhanced skills through community involvement and regular code reflection.
4. Learning a New Language: Adapted to the unique syntax and paradigms of F# as a new programming language.
5. Console Input/Output Handling: Gained experience in handling console input and output, essential for interactive programs.Understood how to read user input and display output in a structured format.

**Issue 5:Write a F# program to print diamond pattern**

Status: **Accepted**

****

**Overview of the project:** [**Pull Request Link**](https://github.com/codinasion/codinasion/pull/4783) **,** [**Issue Link**](https://github.com/codinasion/codinasion/issues/3543)

The F# code features a `printDiamond` function designed to generate and display a diamond-shaped pattern in the console. This function takes an integer `n` as input, representing the size of the diamond. Inside `printDiamond`, there's a nested helper function named `printLine`, which is responsible for printing individual lines of the diamond. Each line comprises a combination of asterisks and spaces, with the number of asterisks centered and increasing to the midpoint, then decreasing symmetrically. The code also includes a user input section, where it prompts for and accepts the diamond size, converting it to an integer to construct the pattern. The output is a visually symmetrical diamond made of asterisks, exemplifying basic loop and string manipulation in F#.

**Problems Faced:**

1. Loop Logic: Ensuring correct loop bounds and steps was crucial to form the proper diamond shape.
2. String Calculations: Accurate calculations for the number of spaces and asterisks were needed to maintain symmetry.
3. Off-by-One Errors: Common in patterns, these errors could easily disrupt the diamond's shape.
4. Input Validation: Handling non-integer or negative inputs was necessary to prevent runtime errors.
5. Console Limitations: Large diamonds often exceeded the console window size, causing display issues.

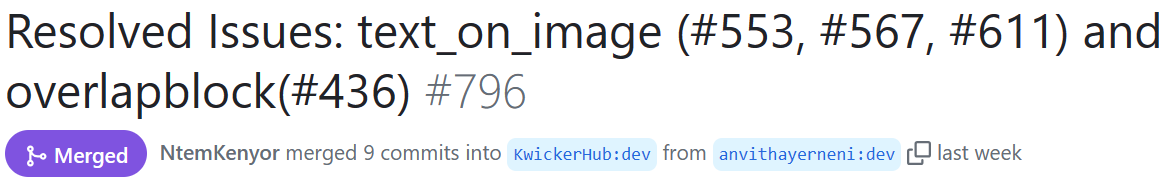
**New Skills Acquired:**

1. Advanced String Manipulation: Became proficient in string operations such as concatenation, repetition, and slicing, crucial for creating the diamond pattern. Understood how to dynamically generate strings based on loop counters and conditional logic.
2. Loop Control and Iteration Techniques: Mastered the use of for-loops, including setting correct loop bounds and step values, essential for the symmetric pattern of the diamond. Learned about loop control structures and how they govern the flow of a program.
3. Problem-Solving and Algorithm Development: Enhanced problem-solving skills by translating a visual pattern into a logical algorithm.Learned to break down a complex problem (diamond pattern creation) into smaller, manageable parts.

**Issue 6: Creating HTML Pages( Text on Image, Overlap Block)**

Status: **Accepted**

I have resolved 4 issues in a single PR . Three from text\_on\_image([#553](https://github.com/KwickerHub/WebCraftifyAI/issues/553), [#567](https://github.com/KwickerHub/WebCraftifyAI/issues/567), [#611](https://github.com/KwickerHub/WebCraftifyAI/issues/611)) and one from overlapblock([#436](https://github.com/KwickerHub/WebCraftifyAI/issues/436)) .



**Overview of the project:** [**Pull Request Link**](https://github.com/KwickerHub/WebCraftifyAI/pull/796)

Based on the templates given in each issue Html pages are created accordingly using HTML structure, CSS styling and text styling.

**Problems Faced:**

1. Positioning Issues: Aligning text over an image can be tricky. Beginners might struggle with understanding positioning properties (relative, absolute, fixed) and how they work together.
2. Cross-Browser Compatibility: Different browsers may render HTML/CSS differently, leading to issues with layout and appearance that a beginner might not anticipate.
3. Responsiveness: Ensuring that the text remains properly aligned and readable on different screen sizes and devices is a common challenge. This requires a good grasp of responsive design principles and media queries.
4. Interactivity Issues: If the overlapping blocks contain interactive elements (like buttons or links), ensuring they are accessible and functional can be a challenge.
5. Learning Curve: HTML and CSS have many properties and nuances that can be overwhelming for beginners like me.

**New Skills Acquired:**

1. Advanced CSS Positioning: I mastered the use of CSS positioning techniques (relative, absolute, fixed) for complex layouts and overlapping elements.
2. Responsive Web Design Skills: I developed proficiency in creating flexible, device-friendly designs using media queries and fluid layouts.
3. HTML Semantics and Structure: I learned to use HTML5 semantic elements and structured content logically for improved accessibility and SEO.
4. Understanding of Design and Accessibility Principles: I gained an understanding of UI/UX design aesthetics, including color schemes, typography, and ensuring web accessibility for all users.
5. Cross-Browser Compatibility and Debugging Skills: I acquired skills in testing, debugging, and ensuring consistent website performance across different browsers and devices.

**Overall Experiences- Knowledge Applied through the entire project**

1. For developing a pygame using hooman library:
   1. Programming Basics: Utilizing variables, loops, functions, and control structures.
   2. Pygame Framework: Handling events, rendering graphics, and managing the game window.
   3. Game Mechanics: Implementing collision detection, randomness in spawning, and score tracking.
   4. Math and Geometry: Calculating positions and movements, and using bounding boxes for collision detection.
   5. Performance Optimization: Controlling frame rate and managing resources efficiently.
   6. User Interface and Experience: Creating simple visuals and responsive controls for a good user experience.
2. For writing a #F program:
   1. Functional Programming: Grasped concepts like immutability, pure functions, and higher-order functions.
   2. F# Syntax and Features: Mastered F#'s syntax, pattern matching, and unique data types.
   3. Asynchronous Programming: Implemented async workflows and parallel processing.
   4. NET Framework Integration: Utilized the .NET libraries and interoperated with other .NET languages.
   5. Code Organization: Effectively used modules, namespaces, and managed dependencies with NuGet.
3. For developing a front end page using HTML & CSS:
   1. HTML Proficiency: Understood HTML syntax, tags, and semantic HTML.
   2. CSS Skills: Mastered selectors, responsive design, and layout techniques like Flexbox and CSS Grid.
   3. Design Principles: Acquired basic knowledge of color theory, typography, and UI/UX best practices.
   4. Responsive Web Design: Created adaptable layouts for various devices and screen sizes.
   5. Cross-Browser Compatibility: Ensured consistent rendering across different browsers.
   6. Performance Optimization: Applied techniques to improve page load times.
   7. Debugging: Utilized browser developer tools to troubleshoot HTML and CSS.