Paradigm Statement

In crafting my Snake game, I adopted the procedural programming paradigm, which aligns seamlessly with the capabilities of C, the language used for this project. The game is structured through distinct functions such as `init\_screen`, `update\_screen`, `step`, and `handle\_input`. Each function serves a specific purpose, illustrating the procedural approach of breaking down complex tasks into simpler, well-defined procedures. For example, the `step` function is solely responsible for controlling the snake's movement and growth.

A notable feature of this paradigm in my project is the use of global variables like `board`, `head`, `dir`, and `score`. These variables maintain the state of the game and are modified by various functions throughout the code. This approach is typical in procedural programming, where a shared state is common, and it fits well with C’s focus on direct memory manipulation.

The core of the game lies in the main loop within the `main` function, which is a direct representation of procedural programming’s iterative nature. This loop is responsible for rendering the game state, processing user input, and updating the game mechanics in a sequential manner, emphasizing the paradigm’s focus on step-by-step execution.

Control structures such as `switch` and `if` statements, particularly in the `handle\_input` and `step` functions, are crucial in dictating the game's behaviour in response to various inputs and scenarios. These structures demonstrate the procedural paradigm's emphasis on controlling the flow of the program.

In summary, the development of the Snake game in C showcases the strengths of procedural programming. This approach facilitated an organized and straightforward development process, adeptly handling the game's logic and state. While object-oriented principles offer a different approach to structuring, the procedural paradigm proved to be a perfect fit for the nature of this project and the features of the C language.