**Paradigm Statement**

In the development of my Snake game using the Go programming language, I have chosen to follow the procedural programming paradigm, which significantly shapes the way my code is structured and how it functions. Procedural programming is characterized by the organization of functions and the step-by-step flow of the program's logic.

I have taken a deliberate approach to design functions, each with a specific role in the game. Functions such as `playSnake`, `startEventLoop`, `doMove`, and `getStudentID` have distinct purposes, contributing to the overall functionality of the game. This adherence to the procedural paradigm emphasizes the importance of breaking down the program into smaller, well-defined functions responsible for manipulating data and executing specific operations.

Furthermore, I have utilized imperative programming techniques, which involve explicit instructions and control flow, to drive the game's behaviour. Loops and conditional statements are essential for managing the game loop, handling user input, and overseeing collision detection. These imperative constructs ensure precise execution and decision-making within the game.

While my Go implementation leans more towards procedural programming, it highlights the adaptability of the Go language, which can seamlessly accommodate multiple programming paradigms. Go's simplicity and efficiency in handling low-level operations make it a practical choice for game development, even when emphasizing procedural programming.

To sum it up, my Snake game in Go effectively applies the principles of procedural programming, featuring modular functions and imperative coding structures. This approach enhances code organization, maintainability, and the overall control of the game's flow, showcasing Go's versatility as a language capable of accommodating various programming paradigms.