# Contents

## Acknowledgement

We would like to express our special thanks of gratitude to our teacher (MS. SREELAKSHMI K ma’am), who gave us the golden opportunity to do this wonderful project on the topic ( SNAKE GAME USING JACK), which also helped us in doing a lot of Research and we came to know about so many new things. We are thankful for the opportunity given. We would also like to thank our group members, as without their cooperation, we would not have been able to complete the project within the prescribed time.

## List of Figures

## ABSTRACT

The creation of a classic Snake game with the Jack programming language is shown . The main goal was to produce an engaging and playable game that illustrates key ideas in game creation, such as processing user input, managing game state, and generating visuals. The game is divided into various sections, each of them manages certain aspects of the gameplay, including the generation of random numbers, the movement and growth of the snake, and the main game loop.

The process comprised creating the game's architecture, putting Jack's reasoning into practice, and compiling the result into VM code that could be run on the Hack platform. Smooth snake movement, food production, score monitoring, increasing difficulty as the snake develops,increasing levels by eating food ,and the creation of a scoreboard at the end of the game are some of the game's key elements. The end result is a Snake game that is completely functional, works well on the Hack platform, and offers an entertaining user experience. Test results show that there are no visible performance problems with the game, and it runs smoothly. Future upgrades might include improving the obstacles ,life's for the game to restart , and power up's.

## INTRODUCTION

The goal of the Snake game project was to create a classic action game on the Hack platform exclusively utilizing the Jack programming language. This project illustrates important ideas in game design and software development, acting as an instruction tool. The primary goals are to manage the game state, process user inputs, and render images within a modular framework that improves scalability and maintainability. The project showcases the useful uses of the Jack language and the capabilities of the Hack platform by developing a pleasant and entertaining game, offering a thorough introduction to computer science concepts and programming methods.

## METHODOLOGY

The design, implementation, and testing stages of the Snake game's development were divided into multiple distinct sections. Six primary   files make up the project; each one is in charge of particular features that come together to make the entire game.Each file and its function in the project are described in detail below.

### Main.jack

- Functionality: The Main class serves as the entry point for the program.The Main class essentially coordinates the initialization, execution, and cleanup of the Snake game. It ensures that the game starts, runs, and shuts down properly
- Implementation:
  Initialization
  In the main function, we create a new SnakeGame object called game by using its constructor.
  Game Execution
  We call the run() method on the game object to start the game.The game loop, where the snake moves and the player uses controls to interact with the game.

The loop continues until the game is over.

Cleanup

- After the game ends, we call the dispose() method on the game object to clean up any resources it used, like memory or files.

## Random.jack

- Functionality: **Random .jack**in the Jack programming language, which implements a pseudo-random number generatorImplementation setSeed:

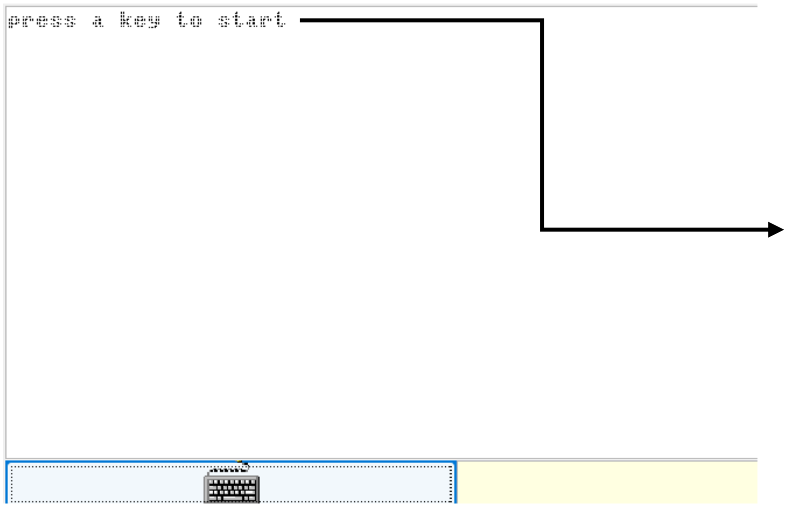
  Sets the seed value for generating random numbers,Updates the seed to the new value provided.

  rand:

  Generates a random number between 0 and 32767.Adds 20251 to the current seed. If the new seed is negative, it subtracts 32768 from it. The updated seed value is then returned as the random number.

  randRange:

  Creates a mask to ensure the number stays within the specified range. It uses the rand() function to get a number and applies the mask. If the number exceeds the specified range, it repeats the process until a valid number is generated. The valid random number is then returned.



the part that involves displaying "Press any key to start" Specifically, there should be a section of code responsible for initializing the game environment and prompting the player to start.

**Fig(2.1.1) . Key to start**

## RandSeed.jack

- Functionality: The random number generator's seeding is controlled by this file.
- Implementation:

  Initializes two local variables, seed and key, to 0, prompts the user to press a key to start, and continuously increments the seed value until a key press is detected. If seed reaches 32767, it resets to 0 to prevent overflow.

  Clears the screen after a key press is detected and returns the generated seed value as an integer.

## Snake.jack

- Functionality: This file manages the snake's properties and behaviors, including movement, growth, and collision detection.
- Implementation:

  Initialization:

The constructor new() initializes various properties of the snake, such as its position, length, and growth rate. It also draws the initial snake on the game grid.

Disposal:

The method dispose() releases memory allocated for storing the snake's history and deallocates the snake object.

History Management:

The method checkRewriteHistory() checks if the snake's history needs to be rewritten due to reaching the maximum cycle count.

The method rewriteHistory() updates the snake's history when necessary.

Accessors:

Methods like posX(), posY(), getLength(), getDir(), and getLastDir() provide access to the snake's position, length, and direction.

Direction Control:

Methods like setDir(), rememberDir(), and tryMove() manage the snake's direction and movement.

Growth:

The method grow() handles the snake's growth by drawing the new head and updating its length.

The method eatFood() increases the amount the snake has left to grow when it eats a food pellet.

Drawing and Clearing:

Methods like drawHead() and clearTail() manage drawing and clearing the snake's head and tail segments on the game grid.

## SnakeGrid.jack

- Functionality: This file manages how food is arranged and how the game grid is represented.
- Implementation

  Initialization:

  The constructor new() initializes the grid based on the specified pixel size, calculating the grid size accordingly and initializing the grid array.

  Disposal:

  The method dispose() releases memory allocated for storing the grid and deallocates the SnakeGrid object.

  Food Placement:

  The method placeFood() selects a position for the food pellet on the grid, ensuring it is not placed on top of a snake piece.

  The method drawFood() draws the food pellet on the screen.

  Grid Initialization:

  The method initGrid() initializes the grid by creating a two-dimensional array and setting all grid cells to unoccupied.

  Accessor Methods:

  Methods like sizeX(), sizeY(), foodX(), and foodY() provide access to various properties of the grid.

  Grid Access Methods:

  Methods like setOccupied(), checkOccupied(), and checkFood() allow for checking and updating the occupancy status of grid cells.

Debugging and Drawing:

The method debugPosition() displays debug information on the status line regarding the cycle, current X/Y position, etc.

Methods like drawSnakeBit() and clearSnakeBit() draw and clear blocks on the grid respectively, marking them as occupied or unoccupied.

The method drawStatus() draws the status line on the screen, including information about the level, score, snake length, and whether the game is paused.

## SnakeGame.jack

- Functionality: This file combines all parts and controls the overall logic of the game, including the score, length, stages, scoreboard, and toggling pause features.
- Implementation:

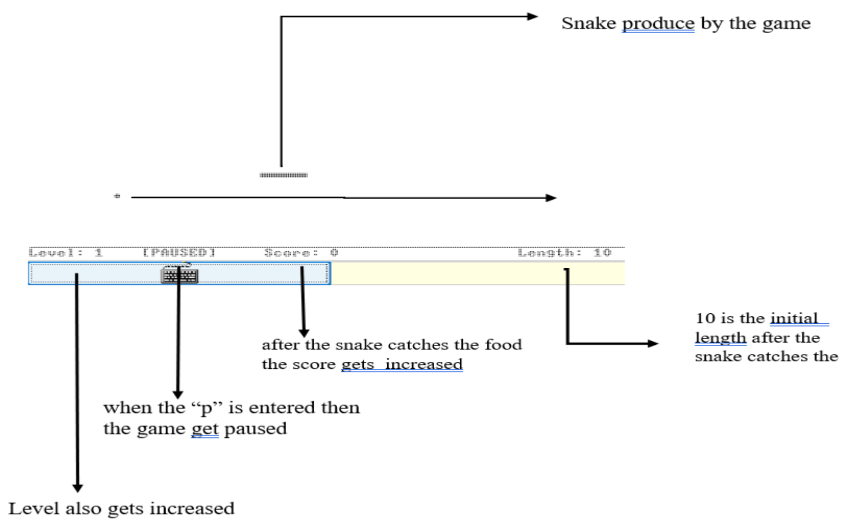
Levels: As the player eats more food, the game gets harder. The snake moves faster through each level, and new obstacles could appear.

Score tracking: Depending on how many food items are ingested, it keeps track of and changes the player's score.

Length Management: The snake's length is tracked in the file, which is updated as it grows longer with each meal it eats.

Scoreboard: The player's total score, the snake's length, and the greatest level they have reached are all displayed on the scoreboard at the end of the game.

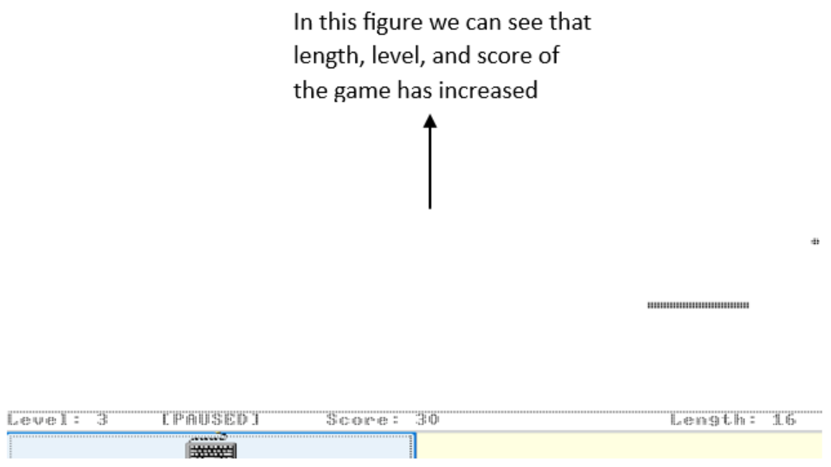
Toggle Pause: The game has a pause feature that lets the user stop and start playing at any moment. This feature makes gaming more flexible, which improves the user experience.



Snake produce by the game
10 is the initial length after the snake catches the
after the snake catches the food the score gets increased
when the "p" is entered then the game get paused
Level also gets increased

```
Level: 1     [PAUSED]      Score: 0                     Length: 10
```

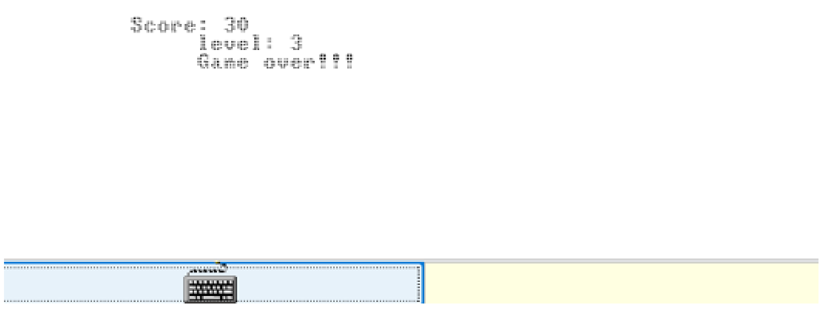
**Fig(2.6.1). Initial of the game**



In this figure we can see that length, level, and score of the game has increased

```
Level: 3      [PAUSED]       Score: 30                  Length: 16
```

**Fig(2.6.3). Display of Scoreboard**



```
Score: 30
     level: 3
     Game over!!!
```

**Fig(2.6.3). Display of Scoreboard**

The six primary components of the Snake game's development plan each have a specific role to play. Together, these components ensure that the game functions properly and is enjoyable to play. This kind of labor breakdown makes it easy to introduce new features and fix bugs. This method produces a more cohesive and fun-to-play game while also streamlining the game-building process.

## RESULTS

How the Game Works and How to Play It

**Launching the Game and Controls**: "Press any key to start the game" appears on the output screen when the code executes in the virtual machine emulator. Pressing any key on the keyboard launches the game. The player can then use the arrow keys or other predefined controls to control the snake's direction. Using the 'P' key, you can pause and resume the game.

**Gameplay Mechanics**: The player sees food items on the grid and a moving snake when the game first launches. The snake never stops moving as it searches for and eats food. The snake grows longer after consuming a food item, and a new food item is created at random. The game level rises as the snake eats more food, which causes it to move quicker and increases the challenge. As the snake consumes the food, the score updates in real-time, giving the user instant feedback.

**Scoreboard and Game Over**: The game is not over until the snake strikes one of the grid's edges. Following a collision, the player's total score, the level they completed, and the word "GAME OVER!!!" are displayed on a scoreboard.

The information on the pause feature and the real-time score update is included in this updated section, which improves the user experience overall by offering quick feedback and flexible control. The Jack programming language on the Hack platform is effectively showcased in the Snake game, which offers an interesting and immersive gaming experience through the combination of dynamic gameplay advancement, intuitive controls, and unambiguous end-of-game feedback. This thorough review demonstrates the thoughtful design and implementation decisions made to guarantee that users have a fluid, responsive, and entertaining gaming experience.

## CONCLUSION AND FUTURE WORK

The Snake game project effectively illustrates how basic programming ideas can be applied to the Hack platform by utilizing the Jack programming language. The game is painstakingly divided into six primary files, each of which handles a particular set of functionalities. This allows the project to achieve a modular and maintainable architecture.

The game's features, which improve the entire gameplay experience, include real-time score updates, level progression, and a pause function. It also starts with a prompt for the player and uses simple keyboard controls. The player is kept interested and pushed by the dynamic gameplay, which sees the snake grow with each food item consumed and the game speed up with each level. A thorough scoreboard that offers concise feedback on the player's performance appears at the end of the game.

All things considered, the project is a prime example of how to combine game state management, visual rendering, and user input processing into a scalable and organized system. In addition to showcasing the useful uses for the Jack programming language, it provides users with an enjoyable and instructive experience. The potential of the Jack programming language and the Hack platform are successfully demonstrated through this project, highlighting the value of modular design and extensive testing in the game development process.

## BIBLIOGRAPHY

https://github.com/SeaRbSg/nand2tetris/tree/master/jf647/09/Square

https://github.com/wendyyuchensun/snakeGame

https://github.com/RakovBogdan/Snake

https://github.com/RakovBogdan/Snake