Ahmed Almoamen

Snake Game Project Report/Tutorial

Introduction

This report outlines the development of a classic snake game using web technologies including HTML, CSS, JavaScript, Node.js, and MySQL. The game features a snake that grows when it eats apples, with added complexities like golden apples, reverse direction power-downs, and obstacles.

Technologies Used

- HTML: Structured the game's layout.
- CSS: Styled the game, including the canvas and instruction panel.
- JavaScript: Powered the game's logic, including snake movements, collision detection, and score handling.
- Node.js: Provided the server-side environment for running the game and handling HTTP requests.
- MySQL: Used for storing and retrieving high scores, demonstrating data persistence.

Game Mechanics

- Controls: The snake is controlled using arrow keys or WASD. The game responds to keyboard inputs for navigation.
- Scoring System: Players earn points by eating apples. Each apple increases the score by 1 point.
- Special Items:
  - Golden Apple: Appears occasionally with a 10% chance, increasing the score by 3 points and adding 3 segments to the snake.
  - Reverse Power-Down: Appears with a 20% chance after eating an apple, temporarily reversing the control scheme.
- Obstacles: Saw blades act as static obstacles. Colliding with them ends the game.

Development Process

Challenges and Solutions

- Implementing the Reverse Control Feature
  - Challenge: A significant challenge in this project was the implementation of the reverse control power-down. The main difficulty lay in temporarily reversing the game's control scheme without affecting the overall game logic and user experience.

- Solution: This was addressed by introducing a state variable isReverseActive to track whether the reverse power-down was in effect. The changeDirection function, responsible for handling user inputs, was modified to check this state. If active, the function would invert the control mappings (e.g., pressing the left arrow key moves the snake right). A timer was used to revert the controls to normal after a set duration, ensuring the effect was temporary.

Database Structure

The Snake game utilizes a MySQL database to store player scores and related information. The database structure is simple, consisting of a single table designed to record the scores achieved by players.

Table: **scores**

The **scores** table is used to keep track of each player's score. Below are the details of the table structure:

- **id**: An auto-incremented integer that uniquely identifies each score entry. It serves as the primary key for the table.
  - Type: **INT**
  - Properties: **AUTO_INCREMENT**, **PRIMARY KEY**
- **playerName**: A variable character field that stores the name of the player.
  - Type: **VARCHAR(255)**
  - Properties: **NOT NULL**
- **score**: An integer field that records the player's score.
  - Type: **INT**
  - Properties: **NOT NULL**
- **timestamp**: A timestamp field that records the date and time when the score was achieved. It defaults to the current timestamp at the time of record insertion.
  - Type: **TIMESTAMP**
  - Properties: **DEFAULT CURRENT_TIMESTAMP**

Constraints and Relationships

The **scores** table does not have explicit foreign keys or complex relationships with other tables, as it serves as a standalone table to store game scores. There are no complex constraints beyond the **NOT NULL** constraints on the **playerName** and **score** fields.

Conclusion

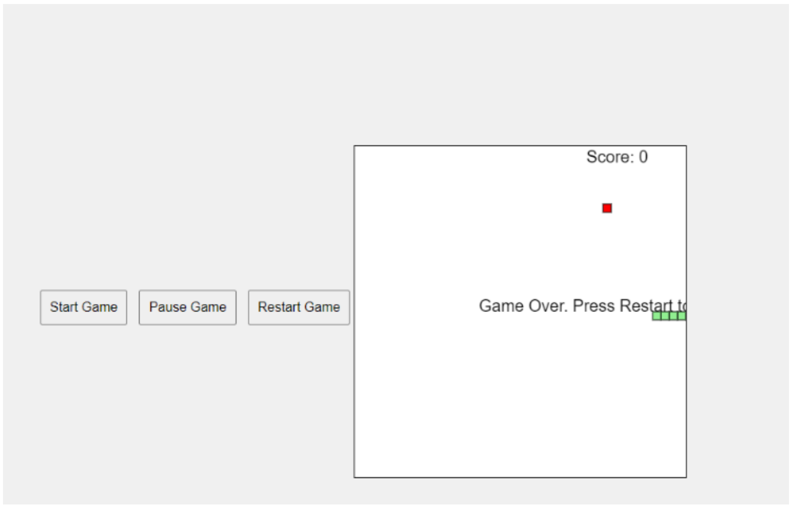
This project successfully resulted in the creation of a classic snake game using web technologies. The primary goals of building a functional game with basic features like snake movement, apple collection, and score tracking were achieved.

The process involved routine coding tasks and addressing common development challenges, such as implementing specific game features and ensuring smooth gameplay. This project provided an opportunity to apply basic programming concepts and tools.

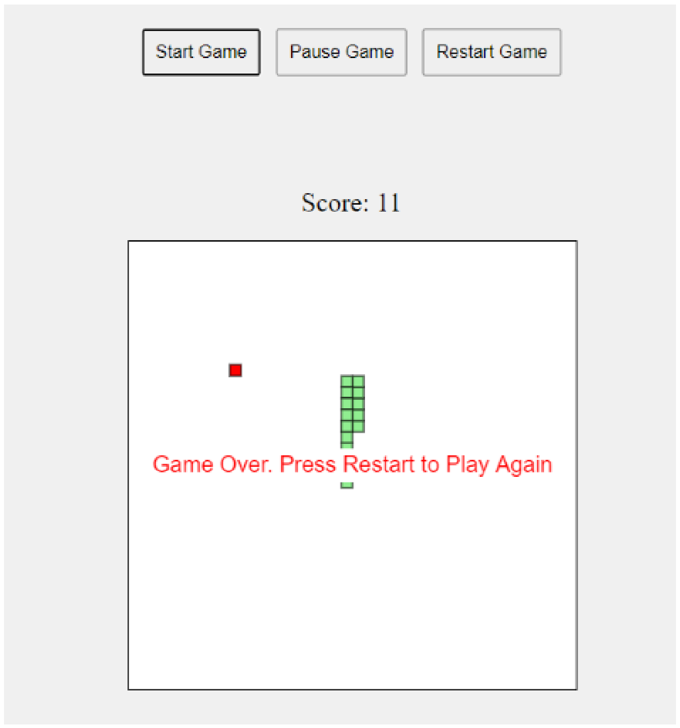
Future improvements might include adding new features or refining the existing ones. Maybe some fun new features including a 2-player mode or adding new effects.

Appendix

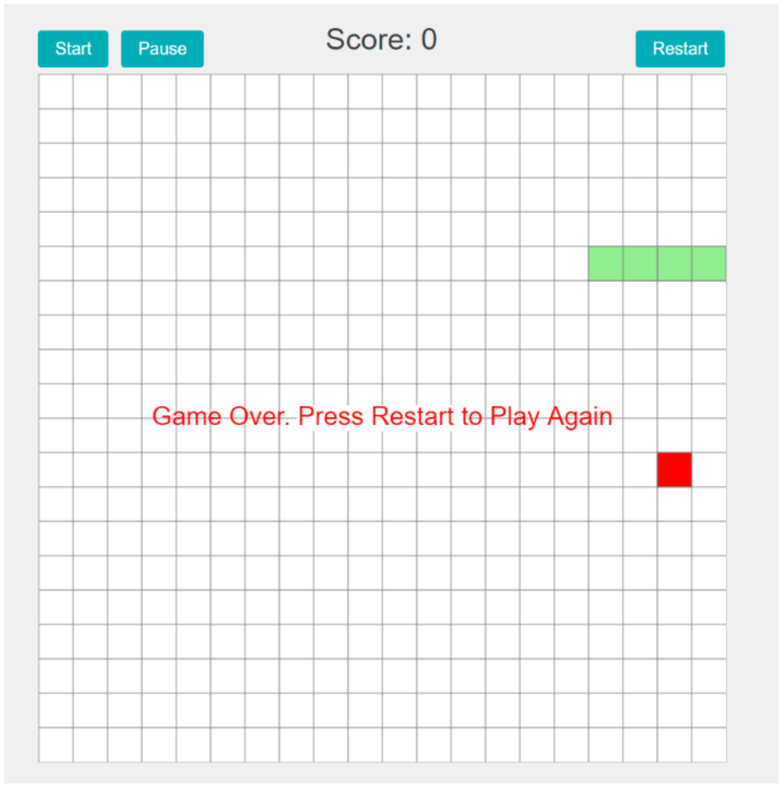
Early Development



Ahmed Almoamen

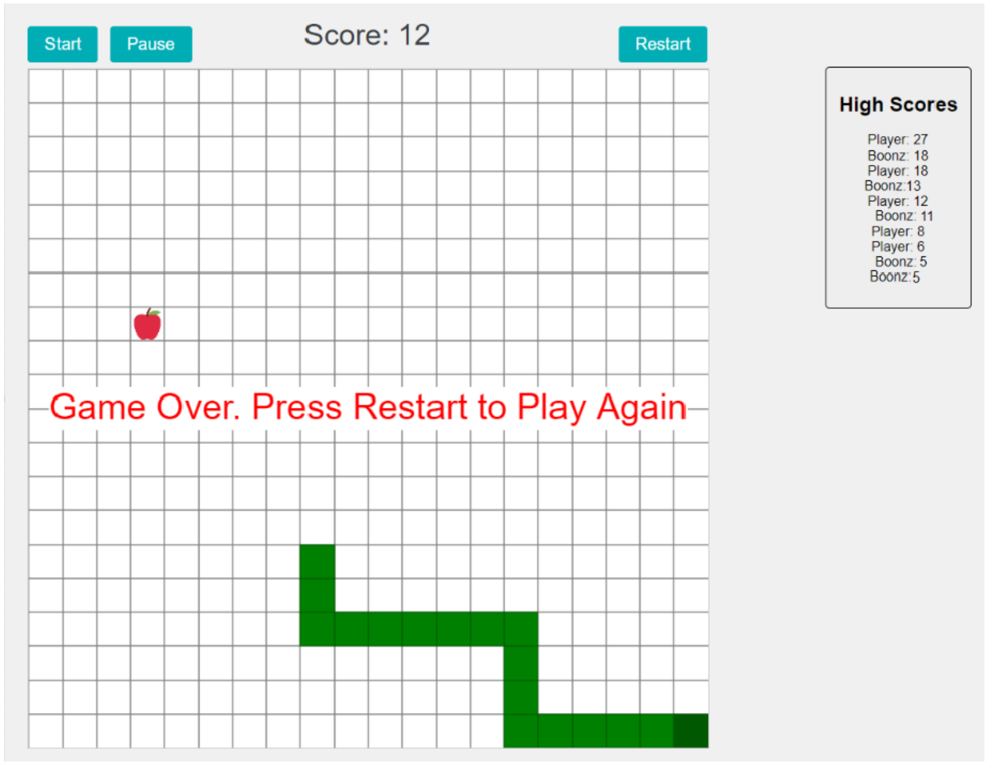


Some CSS/Overlay fixing

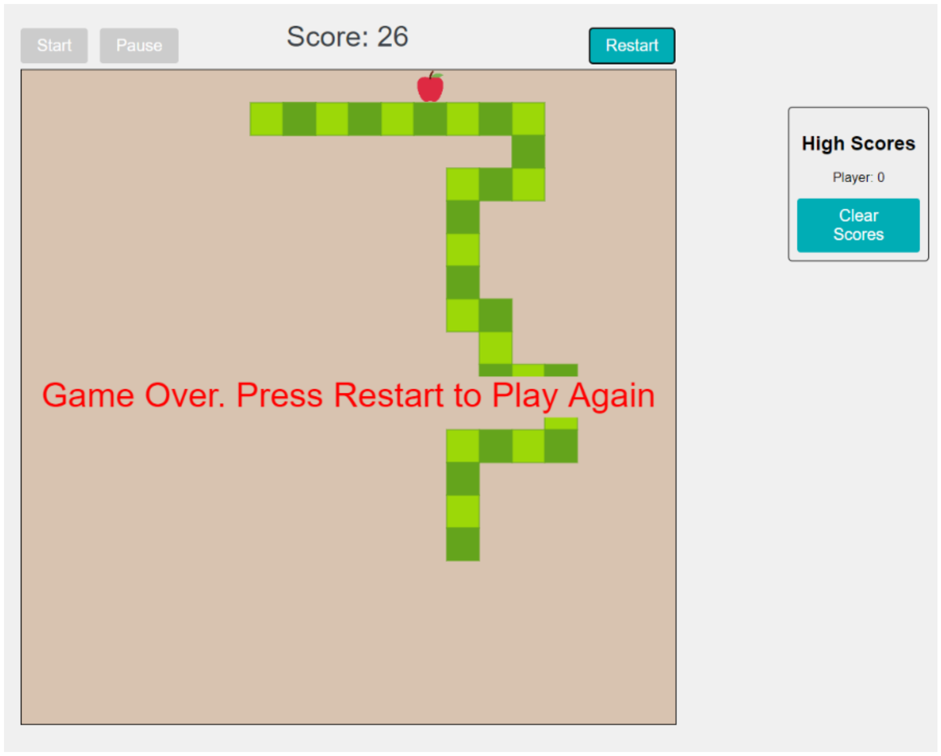


Ahmed Almoamen

Implemented DB and more Styling

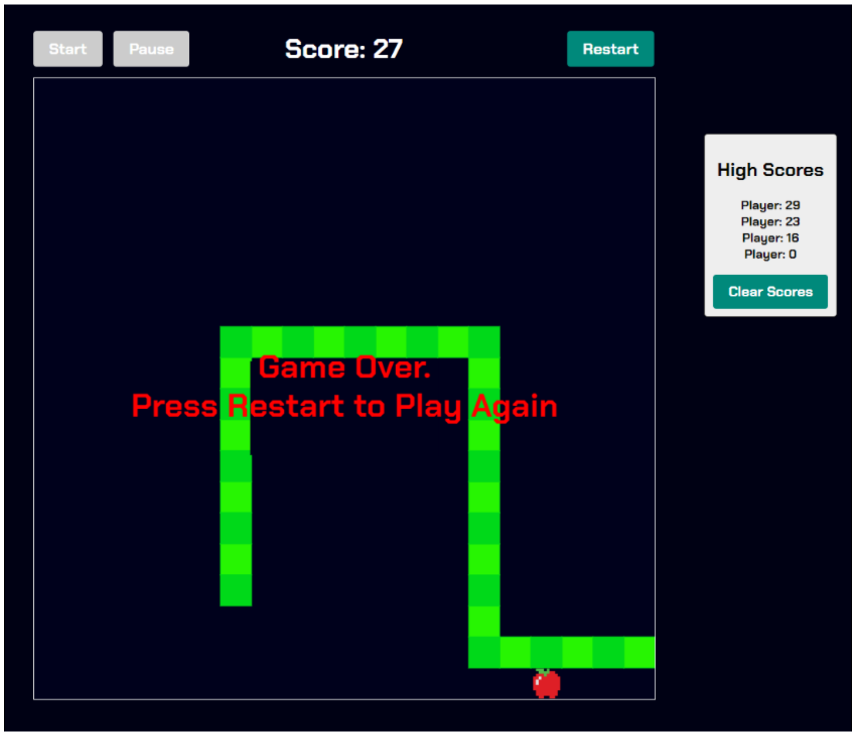


More styling

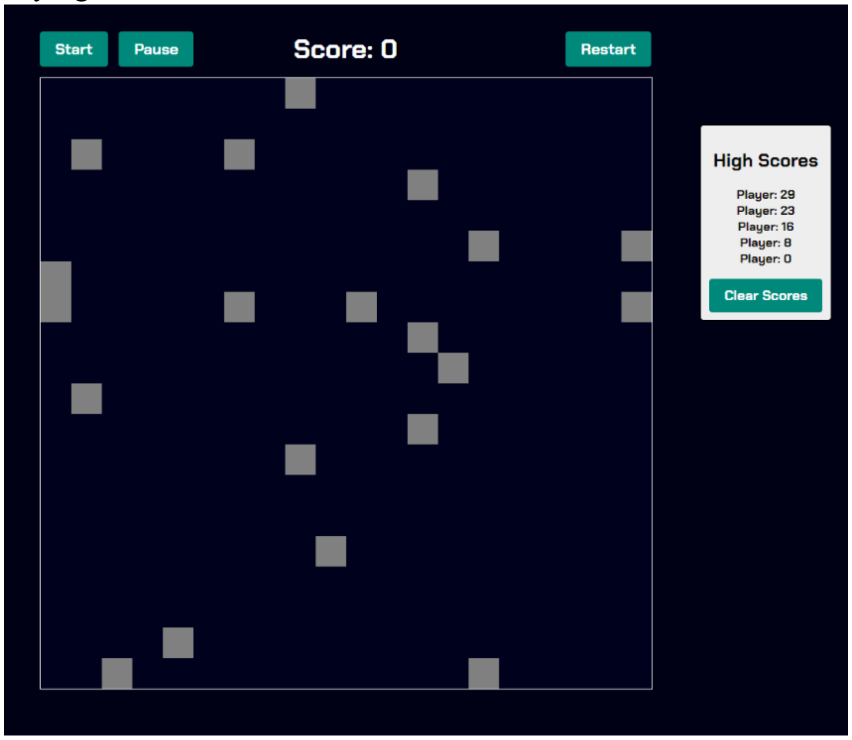


Ahmed Almoamen

A lot of styling changes here.

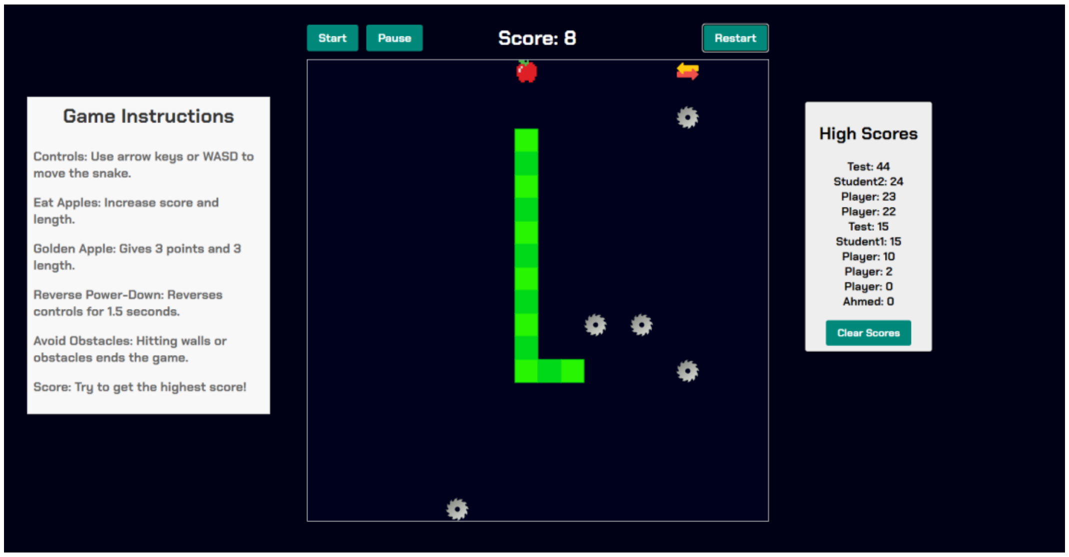


Trying to add obstacles



Ahmed Almoamen

Final Project



Game Instructions

Controls: Use arrow keys or WASD to move the snake.

Eat Apples: Increase score and length.

Golden Apple: Gives 3 points and 3 length.

Reverse Power-Down: Reverses controls for 1.5 seconds.

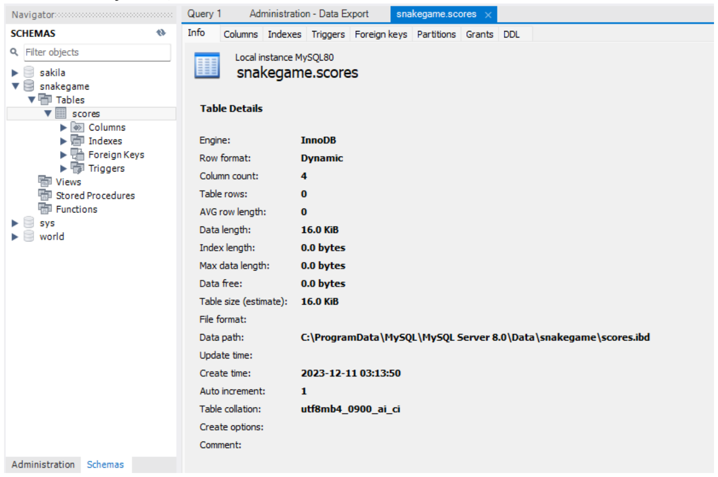
Avoid Obstacles: Hitting walls or obstacles ends the game.

Score: Try to get the highest score!

High Scores

Test: 44
Student2: 24
Player: 23
Player: 22
Test: 15
Student1: 15
Player: 10
Player: 2
Player: 0
Ahmed: 0

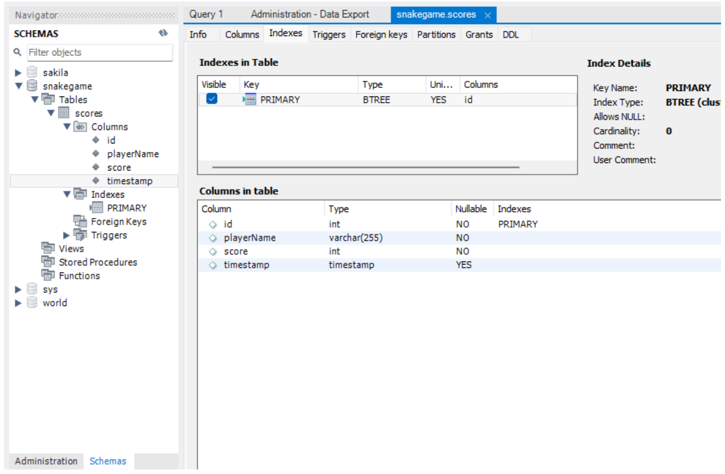
Some MySQL Stuff



Local instance MySQL80
snakegame.scores

Table Details

Engine:	**InnoDB**
Row format:	**Dynamic**
Column count:	**4**
Table rows:	**0**
AVG row length:	**0**
Data length:	**16.0 KiB**
Index length:	**0.0 bytes**
Max data length:	**0.0 bytes**
Data free:	**0.0 bytes**
Table size (estimate):	**16.0 KiB**
File format:	
Data path:	**C:\ProgramData\MySQL\MySQL Server 8.0\Data\snakegame\scores.ibd**
Update time:	
Create time:	**2023-12-11 03:13:50**
Auto increment:	**1**
Table collation:	**utf8mb4_0900_ai_ci**
Create options:	
Comment:	



Indexes in Table

Visible	Key	Type	Uni...	Columns
✓	PRIMARY	BTREE	YES	id

Index Details

Key Name:	**PRIMARY**
Index Type:	**BTREE (clus**
Allows NULL:	
Cardinality:	**0**
Comment:	
User Comment:	

Columns in table

Column	Type	Nullable	Indexes
id	int	NO	PRIMARY
playerName	varchar(255)	NO	
score	int	NO	
timestamp	timestamp	YES	