

---

**Daniel Ashton**

Student number: 10632874

Course code: SOFT051

Date of submission: 09/05/19

# Snake game project

---

## Contents:

- Background
- Method
- Analysis
- Design
- Implementation
- Testing
- Conclusions
- References

## Background

### Application domain:

This project is a version of the classic snake game developed using JavaScript and C# in visual studio.

### Why is it needed:

The outcome of this project is a take on the classic snake game coded using JavaScript and C#. This is aimed to provide entertainment for the user and allow them to express their competitiveness by trying to beat high scores.

### Aims and objectives:

This project will test the user's reaction time and give them a sense of nostalgia as they play this modern version of the game created decades ago.

Functionality aims:

- Allow users to change the settings of the game such as, speed of snake and ability to travel through walls and appear the other side.
- Store the top 10 scores in each difficulty setting (easy, medium, hard) in a database.
- Allow user to control the snake and make it grow in length as they guide it in the path of randomly appearing object.
- Allow user to navigate between different pages such as settings and scoreboard.

## Method

### Techniques and technology used:

This project was created using visual studio and coded in JavaScript and C#. Originally, only C# was going to be used but Javascript seemed like an easier language to use to code a game like this.

Referring to previous lecture slides helped aid this projects progress as some of the code had similar functionality but required modifying a bit to create this snake game.

## Analysis

### Problem domain:

```
function Collision(obj1, obj2) {  
    var snake0 = obj1.getBoundingClientRect();  
    var food0 = obj2.getBoundingClientRect();  
    var eat = document.getElementById("eataudio");  
    if (snake0.left < food0.left + food0.width && snake0.left + snake0.width > food0.left &&  
        snake0.top < food0.top + food0.height && snake0.top + snake0.height > food0.top) {  
  
        food1.style.top = Math.random() * gamespace.clientHeight + "px";  
        food1.style.left = Math.random() * gamespace.clientWidth + "px";  
  
        score = score + 1;  
        parScore.innerHTML = "Score: " + score;  
        eat.play();  
    }  
}
```

This project encountered many problems during its development. Many of which were solved from gathering information from previous lecture slides and online. One of the problems encountered was getting the food to move to a new random position within the gamespace when the snake encounters it. The above code is implemented to do this operation. I get the size and position of the snake and food elements by using `getBoundingClientRect()` and create an if statement that states if they encounter each other then place the food in a random place within the gamespace, add 1 to score and play appropriate eating sound. I used online sources to help me find out how I could determine if two div elements collided and adapted the code I found so I could use it for my own code and get it to work.

```
function Highscore() {  
    window.location.href = "Scoreboard.aspx?currentscore=" + score;  
}
```

```
void Page_Load(){  
    String cs = "Provider=Microsoft.ACE.OLEDB.12.0;" +  
        "Data Source=" + Server.MapPath("Scoreboard.accdb") + ";";  
    OleDbConnection cn = new OleDbConnection(cs);  
    OleDbCommand cmd;  
    OleDbDataReader r;  
    String i;  
    string score1 = Request.QueryString["currentscore"];  
    String sql;  
  
    if (score1 != null)  
    {  
        sql = "INSERT INTO [Scores] " +  
            "(Score) VALUES" + "(" + score1 + ")";  
        cmd = new OleDbCommand(sql, cn);  
        cn.Open();  
        cmd.ExecuteNonQuery();  
        cn.Close();  
    }  
  
    cmd = new OleDbCommand("SELECT * FROM Scores ORDER BY Score DESC", cn);  
    cn.Open();  
    r = cmd.ExecuteReader();  
  
    i = "";  
  
    while(r.Read()){  
        i = i + r["Score"];  
        if(score1 == r["Score"].ToString())  
        {  
            i = i + "<-----";  
        }  
        i = i + "<br />";  
    }  
    cn.Close();  
    parScore.InnerHtml = i;  
}
```

1st  
2nd  
3rd  
4th  
5th

5

Homepage

Creating a database to have a scoreboard was a problem this project encountered, but was able to be implemented by using a jQuery string to pass the score variable from the JavaScript game to the C# scoreboard page. The Scoreboard page has a Microsoft access database connected to it and the players score is stored in there and tells them where they placed in the leader board. The database was successfully implemented by referring to previous lecture slides and using the help of the lecturer to guide through addressing the problems with the code. Using the access database query design helped understand how to write the code to connect the database and have it be able to be updated.

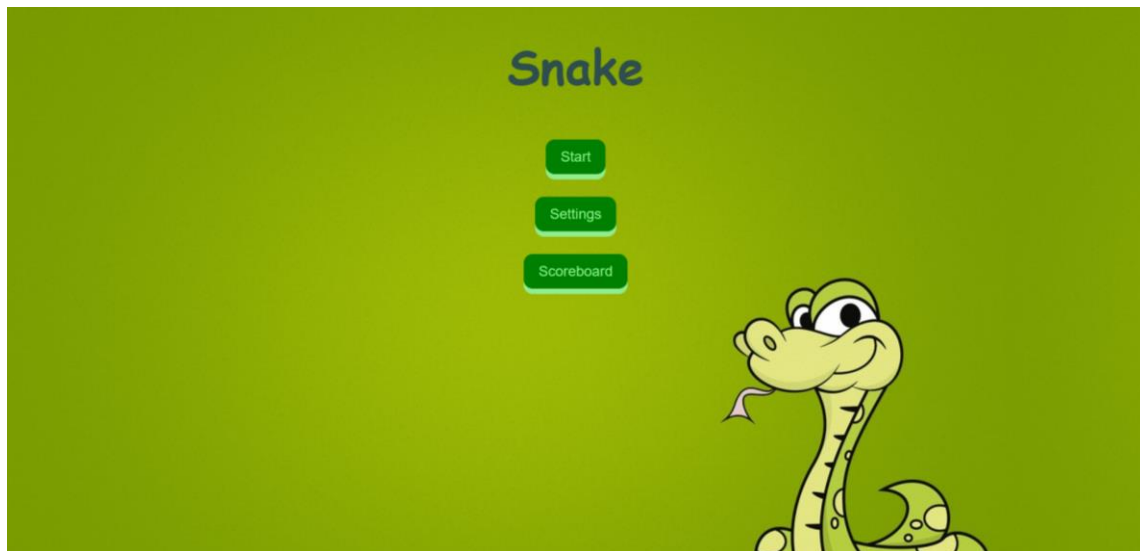
## Design

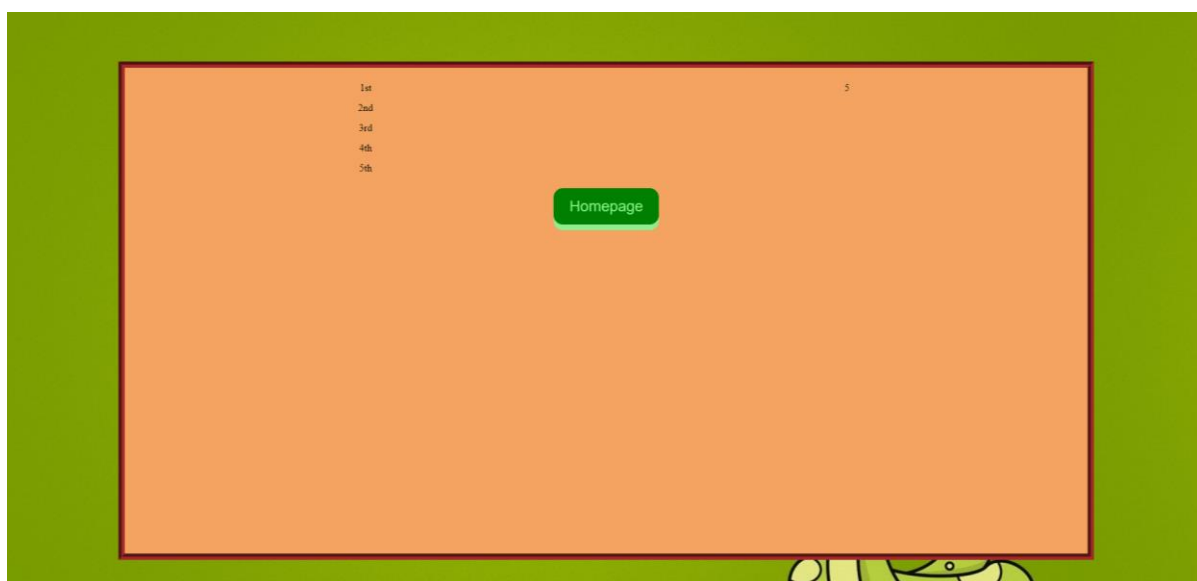
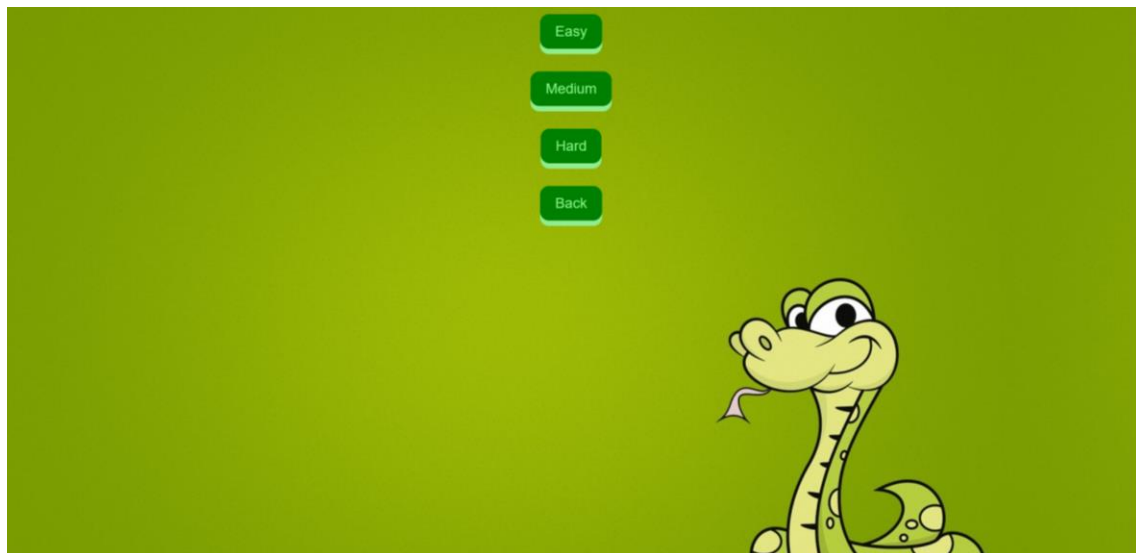
### Functionality:

The different functionalities of this project are;

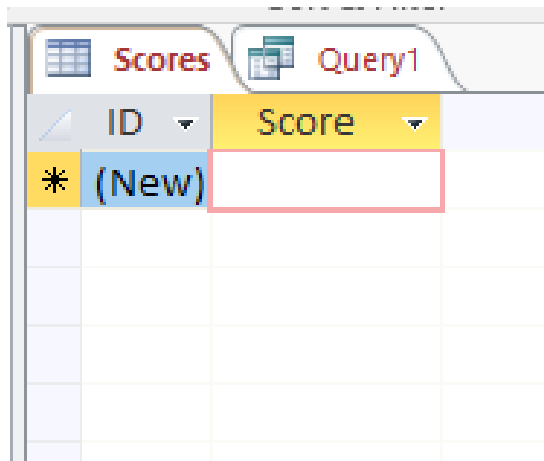
- Ability to navigate between pages using buttons.
- Display score on a scoreboard.
- Move snake using arrow keys.
- Have snake eat object, subsequently making the score increase.

### User interface:





### Data:



ID	Score
* (New)	<input type="text"/>

This is the database used for the scoreboard. The scoreboard inserts into the score column and displays the entire column in a paragraph. It sorts the scores into descending order and highlights what score the previous player achieved by putting an arrow next to it.

### Code:

```
<style>
.button {
  padding: 15px 25px;
  font-size: 24px;
  text-align: center;
  cursor: pointer;
  outline: none;
  color: lightgreen;
  background-color: green;
  border: none;
  border-radius: 15px;
  box-shadow: 0 9px;
}
.button:hover {
  background-color: forestgreen;
}
.button:active {
  background-color: forestgreen;
  box-shadow: 0 5px;
  transform: translateY(4px);
}
</style>
```



These are some css classes that are used throughout this project on multiple pages. They give the buttons a good design to add an appealing style to the game and make the users experience better.



## Implementation:

```
function window_onLoad() {
    Startpositionsnake(snake1);
    Startpositionfood(food1);
    window.setInterval("Main()", x);
}
function document_onKeyDown() {
    switch (window.event.keyCode) {
        case 37: // Left
            xInc = -1;
            yInc = 0;
            break;
        case 39: // Right
            xInc = 1;
            yInc = 0;
            break;
        case 38: // Up
            yInc = -1;
            xInc = 0;
            break;
        case 40: // Down
            yInc = 1;
            xInc = 0;
            break;
    }
}

function MoveSnake() {
    var gamesound = document.getElementById("gamesound");
    gamesound.play();
    var gameover = document.getElementById("eataudio");
    var mx;
    mx = parseInt(snake1.style.left) + xInc;
    if (mx < 0 || mx > gamespace.clientWidth - snake.clientWidth) {
        gamesound.pause();
        gameover.play();
        Highscore()
    } else {
        snake1.style.left = mx + "px";
    }
    var my;
    my = parseInt(snake1.style.top) + yInc;
    if (my < 0 || my > gamespace.clientHeight - snake.clientHeight) {
        gamesound.pause();
        gameover.play();
        Highscore()
    } else {
        snake1.style.top = my + "px";
    }
}
```

The above code allows the user to control the snake using the arrow keys. Inside the window\_onload() function is a set interval that continuously recalls the Main() function at the rate of the set value of x (which is determined in the settings page). Document\_onkeydown() function sets activates the arrow keys to be used in that page. Each time an arrow key is pressed it will change the value of one of two variables (xInc or yInc); these variables are used in the Movesnake() function to move the snake in all four directions. In Movesnake() function the variable mx is the distance the snake is from the left side of the gamespace and the value of xInc. If this value is less than 0 or greater than the width of the gamespace then the soundtrack will stop, and an appropriate sound will play to indicate the player lost. The Highscore() function is then called which takes the user to the scoreboard to see how they ranked against other users. If the player keeps the snake within the gamespace then the snake will move around the screen in the direction the user chooses. The same code is used for the up and down arrow keys in the same function.

```
function Highscore() {
    window.location.href = "Scoreboard.aspx?currentscore=" + score;
}
```

```
void Page_Load(){
    String cs = "Provider=Microsoft.ACE.OLEDB.12.0;" +
                "Data Source=" + Server.MapPath("Scoreboard.accdb") + ";";
    OleDbConnection cn = new OleDbConnection(cs);
    OleDbCommand cmd;
    OleDbDataReader r;
    String i;
    string score1 = Request.QueryString["currentscore"];
    String sql;

    if (score1 != null)
    {
        sql = "INSERT INTO [Scores] " +
              "(Score) VALUES " + "(" + score1 + ")";
        cmd = new OleDbCommand(sql, cn);
        cn.Open();
        cmd.ExecuteNonQuery();
        cn.Close();
    }

    cmd = new OleDbCommand("SELECT * FROM Scores ORDER BY Score DESC", cn);
    cn.Open();
    r = cmd.ExecuteReader();

    i = "";

    while(r.Read()){
        i = i + r["Score"];
        if(score1 == r["Score"].ToString())
        {
            i = i + "<-----";
        }
        i = i + "<br />";
    }
    cn.Close();
    parScore.InnerHtml = i;
}
```

This code is used for the scoreboard to store the players score in a database. The first image shows that the score variable is being passed to the C# scoreboard page by using querystring. This is then requested in the scoreboard page and displayed in the database. The database is connected to this C# page by using the code shown in the picture, the code inserts the players score into the score column into descending order and shows the player what score they achieved and how it compares to previous scores.

## Testing:

This project has undergone tests by different users and has proven to be a playable game although it doesn't meet some of the specifications that were proposed in the plan. The navigation between pages is working as it should however the setting page navigates to the game but without changing the difficulty. The scoreboard page is storing users scores and sorting them into descending order so they can compare to other scores. The gameplay is functioning but not exactly as intended, the user is able to control the snake using the arrow keys and guide it towards the object which then disappears and reappears in a random location on the screen but the snake never grows when the object is eaten.

## Conclusions:

Progress and plan:

Week	From	Description
1	4 Feb	Prepare proposal
2	11 Feb	Submit proposal
3	18 Feb	Receive feedback on proposal Proposal approval
4	25 Feb	Annalise other similar games for ideas, speak to target users Initial specification
5	4 Mar	Design user interface
6	11 Mar	Research how to get the core functionality working and what it entails
7	18 Mar	Phase 1 – Allow user to navigate between pages
8	25 Mar	Phase 2 – Get snake and randomly appearing object inside the game space.
9	1 Apr	Phase 3 – Allow user to control the snake
10	8 Apr	Phase 4 – Get the gameplay functioning correctly (as described above)
11	15 Apr	Phase 5 – Implement different difficulty levels and allow user to select them and turn walls on or off
12	22 Apr	Phase 6 – Create a database to store top 10 scores (of each difficulty level)
13	29 Apr	Verification – Unit Testing Validation – User Testing Software Modification if required
14	6 May	Write Project Report Submit Project Report (9 <sup>th</sup> May)
15	13 May	Demonstration

This is the plan I submitted with my proposal, I have highlighted the objectives I achieved in green and the ones I didn't in red.

Phase 4 was to get the gameplay functioning correctly and whilst the game created, is a playable game it does not operate like as intended because the snake does not grow when it eats the food as it is supposed to.

Phase 5 was to implement different difficulty levels and give the user an option to turn the walls on or off. A settings page is used in the project but the initial idea of making the snake move faster or slower depending on the difficulty selected doesn't work and the snake goes the same pace on all the difficulty options.

Reasons for differences:

This project now just has the moving snake eat the object and not grow. The reason for this is because I couldn't figure out the code to make the div element grow and follow itself. Changes made were to not allow the user the ability to travel through walls because otherwise there would be no lose function.

If this project was to be made again I think the best way to implement the snake would be to use a canvas element and draw the snake on the canvas element within the JavaScript code rather than creating a div element and trying to manipulate its position and size within another div element.

Review:

From doing this project I have learnt how to navigate between pages both in JavaScript and C# and also how to pass variables between the two using querystrings.

I have also learnt how to add css classes to different elements to give things such as buttons and div forms style to make them more appealing and make the user want to use them.

## References:

Wallpapercave.com. (2019). *Snake Game Wallpapers - Wallpaper Cave*. [online] Available at: <https://wallpapercave.com/snake-game-wallpapers> [Accessed 8 May 2019].

W3schools.com. (2019). *How To Animate Buttons With CSS*. [online] Available at: [https://www.w3schools.com/howto/howto\\_css\\_animate\\_buttons.asp](https://www.w3schools.com/howto/howto_css_animate_buttons.asp) [Accessed 9 May 2019].

GeeksforGeeks. (2019). *SQL | ORDER BY - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/sql-order-by/> [Accessed 9 May 2019].