Exploring the potential use of video games to improve focus and memory, according to game testers.

Student: Martin Engström
Class: TE23s
Year: 2025/2026
Supervisors: (Francisco Sabino || Julia Komljenovic)

Abstract

Detta projekt siktar på att utforska om datorspel kan ha potentialen att förbättra fokus och minne. Målet kommer att vara att skapa ett “Simon says"-spel, testa om spelet har potentialen att förbättra fokus och minne samt undersöka om spelet lyckades med sitt mål eller inte. För att ta reda på svaret kommer ett digitalt Google formulär att förberedas och sedan skickas ut till slumpmässiga speltestare av olika åldrar. När alla speltestare har besvarat sina formulär kommer resultaten att visas upp i form av bilder under rubriken “Results”. Under rubriken “Analysis & Discussion” kommer resultaten även att diskuteras kring varför resultaten blev som de blev.

Table Of Contents

1. Introduction

All over the world people are struggling with focus in their daily lives. This can be anything from struggles with sitting still, listening to school lectures or just having difficulties understanding instructions. This project aims to create a “Simon Says” game that can help people learn to focus. What’s most important when creating this type of experiment is making the experience enjoyable for the user. If the user forgets that the game is meant as an exercise, the probability of the game helping the user will increase as well.

1.1 Purpose

The purpose of this project is to see if people can experience improvements by playing a video game. The project starts by making a Google Form asking users to rate their focus and memory. They’ll be asked how they feel before trying the game and then once more after their testing is done. When all game testers have sent in their feedback, it will be determined whether the video game helped them or not.

1.2 Limitations

Limitation 1.

- For the testing period to not take too long, the Google form will only explore the participants' belief of what the game can achieve, meaning no concrete proof of anyone actually improving. The results will solely contain feedback on how the participants experienced the game and if the game has the potential to improve their cognitive abilities.

Limitation 2.

- To ensure the thesis along with the video game gets finished on time, the project will be making a single game, a “Simon Says” game.

Limitation 3.

- For the coding process to ever become finished, no further game improvements will be made once the participants have started their testing period. An opportunity to provide improvement ideas will however still be available in the Google Form, in case the participants come up with an idea that’s worth using in future projects.

1.3 Earlier Plan

In the beginning it was planned to make three small games. One Simon Says, one Memory game and lastly a 2D running game. After finishing the “Simon Says” it was quickly understood that there would be no more time to make another and so the thesis was instead modified towards presenting a single game more thoroughly. Out of the three main aspects Focus, Memory and Decision making, the last one had to go due to it not fitting in with Simon Says’ relaxed play style.

1.4 Why Simon Says was chosen for this project

In the current project, the chosen creation is a “Simon Says” game. In the absolute beginning of the project, figuring out what to make was tricky. After consulting with the web development teacher Mr Sabino, a suggestion was given stating that anything could be done as long as the arguments behind the project can claim that your creation has the potential for helping individuals in some way. Mr Sabino continued by opening up his computer to show various simple videogames. He proceeded by explaining that while the games themselves are very simple, with a solid reasoning behind them, they could be seen as something just and well thought out.

As mentioned in the “1.3 earlier plan”, the original idea was to make three simple video games. One Simon Says, one Memory game and lastly a 2d running game. In the other section it was explained that the Simon Says game was picked due to it being closest to completion and while true it is not the full story. Out of the various games that Mr Sabino showed off, there was a Simon Says video game. Thanks to that, a general understanding of what the game had to look like was acquired even before the styling period had taken place. As a final conclusion, The inspiration for this project, while not intentional, ended up being one of the simple games that Mr Sabino showed off.

1.5 Research Questions

- Did the game come out as enjoyable even though it was meant as an exercise?

- Did the participants feel that the game had potential for improving their focus and memory? If so, did the two cognitive abilities have the potential for improving equally as much?

- Would the participants consider playing a game like this again in the future or do we need to adapt our idea towards something?

2. Background

For this project, focus and memory were chosen as the goals for improvement. All over the world people struggle with focus within their daily lives and to improve that, this project aims to create an exercise in the form of a Simon Says game. Studies have shown that usage of digital media for long sessions, on a regular basis can have negative impacts on a person's ability to focus. However, that doesn't mean that it can't be used for good as well. On another note, studies have shown digital media combined into study material can instead lead to positive outcomes. Games often require players to concentrate, remember patterns, react quickly, and make decisions under pressure, which are all skills related to cognitive abilities such as focus and memory. If the project succeeds with making the video game, may it therefore possibly have the power to improve people's cognitive abilities as well.

2.1 Materials

This project will be using three coding tools to create the game "Simon Says". HTML, CSS and JavaScript. HTML (HyperText Markup Language) is used for outlining the structure of a webpage. In this case it creates elements such as all clickable buttons on the webpage, as well as all containers. CSS (Cascading Style Sheets) is used for designing the elements. First HTML was used for creating the buttons and then the CSS is applied for designing them, meaning the CSS assigns the button's height, width, background color and text. JavaScript is the practical coding language out of the three. It is used for handling game logic and makes sure that the buttons react when pressed on. When a button is clicked it is in the JavaScript where the code instructs what to do in that scenario.

2.2 Game Design

Simon Says is a game where four buttons split into four quarters of a circle are displayed on a circular game board. When the game starts, a random pattern is displayed on the board which the player then has to imitate by pressing the four buttons. For every time the sequence is completed, the pattern is increased by one additional blink of a button.

2.3 Method / Project Planning

This project will be using three stages of a plan. Step 1, make the Simon Says game. Step 2, Have randomly selected individuals test the game and answer a Google Forms document. Step 3, use the game tester's feedback to determine if the game helped them or not.

Making the game:

- Estimation: The game will be finished during the winter break. It will take about 6 weeks in total with 2-3 hour long sessions each day.

Making the Google Forms:

- Just like the Simon Says game, the Google Forms will be finished during the winter break. Expectingly, learning the software along with writing the questions will take around 2 weeks.

Finding game testers:

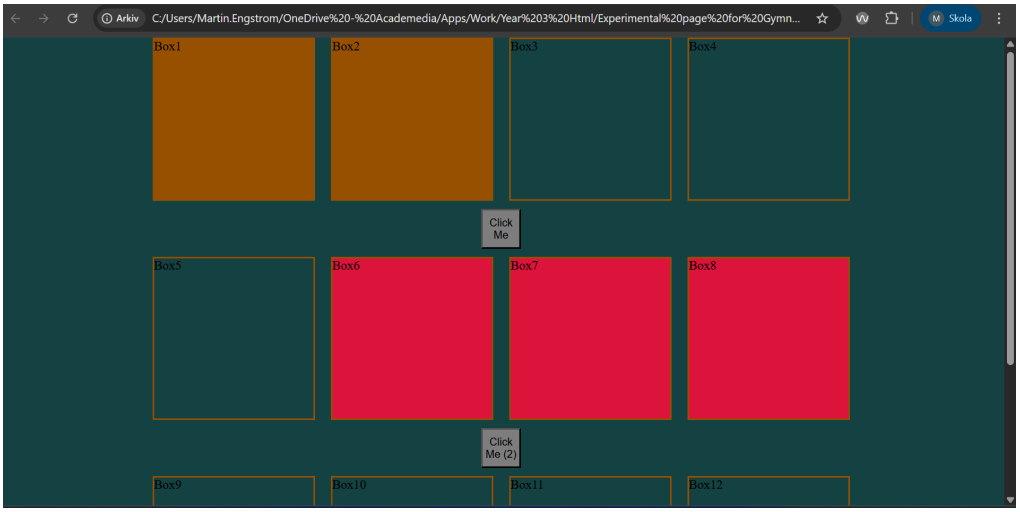
- The game along with the google forms will be expected to be done around new years eve. After that, focus will solely be on finding people that want to test the game. The goal of the project is to have 10-15 people try the game so that a proper analysis can be made by the end of the project. Deadline: 09/02-2026

3. Learning Javascript

For this project to get started it first required the knowledge of three things. HTML for creating elements, CSS for styling the visuals and JavaScript for coding the game's functionality. HTML & CSS already felt comfortable thanks to previous experiences and so JavaScript was the only one needing attention. The journey for gathering knowledge started by looking into the virtual assistant ChatGPT for an introduction to Javascript. There it taught

the most basic of things. How to select DOM elements, how to change the background color of said DOM element and later how to create functions and variables. With the information gathered, a side project was set in motion where every time a new thing was learnt, it was put into a self made website as a practice until it was mastered and utilized in the main Simon Says game project.

An image of one such exercise where the goal was to make blinking sequences.



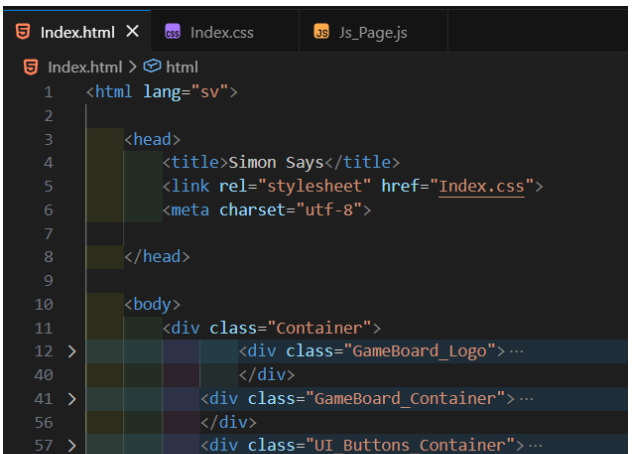
Source used in the project:

https://chatgpt.com/

4. The creation of Simon Says

4.1 HTML & CSS

4.1.1 Making Containers

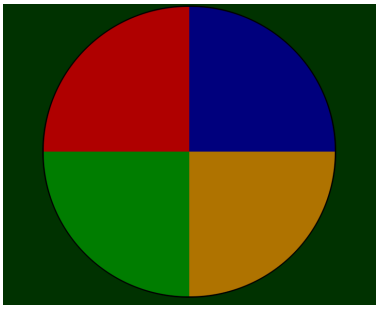


This webpage will be using four containers. One containing absolutely everything on the page, one containing the game board, one containing the game

board logo and one containing a highscore and round tracker.

4.1.2 Adding :root pseudo class

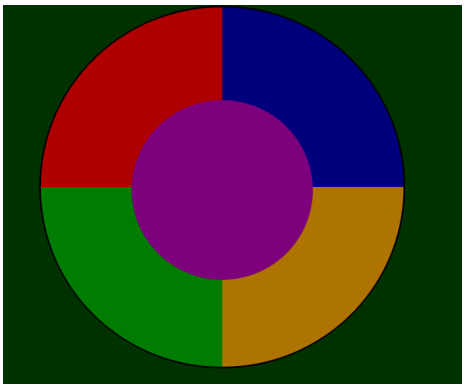
A :root is a position in the css where all colours can be stored. The reason why this is done, is to make it easier in the future in case the colours will need to be changed. It's also favorable because if a variable with a certain colour is changed, all items on the webpage using that variable's colour will be changed as well. This saves a lot of time compared to changing the colour of every item individually. On this step the styling of the gameboard was added as well, excluding the logo.



```
:root{
    --Body_Background-color: rgb(1, 53, 1);
    --TopLeft_Button_bg-color: rgb(177, 2, 2);
    --TopRight_Button_bg-color: rgb(0, 0, 128);
    --BottomLeft_Button_bg-color: rgb(0, 128, 0);
    --BottomRight_Button_bg-color: rgb(179, 117, 2);
    --GameBoard_Logo_OnOffButton_bg-color: grey;
    --GameBoard_Logo_Button_bg-color: rgb(71, 69, 69);
}
```

4.1.3 Styling the logo

```
.GameBoard_Logo{

    height: 200px;
    width: 200px;

    background-color: purple;

    border-radius: 100%;

    z-index: 999;


    position: absolute;
    top: 115;

    display: flex;
    justify-content: center;
    align-items: center;
    overflow: hidden;
```



For the logo to be in front of the game board, it was given the property of "position:

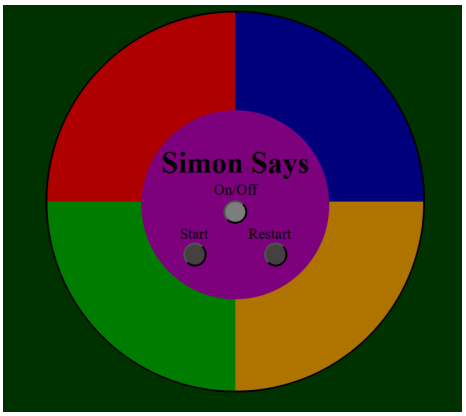
absolute;". This means that no matter what, the logo is going to be in its assigned position, which in this case is in the center of the game board. For the logo to not push any other object away, it was given the property of "z-index: 999;", which puts it on top of everything else on the page.

4.1.4 Styling the logo's interface

After the logo had been positioned, styling the logo's interface was set in motion. At the top a title was positioned displaying the title of the game, "Simon Says". Below the title a combination of a text and button was added displaying the text "On/Off". Below the button was the same combination used two more times but this time displaying the text "Start" and "Restart", respectively.

The intention with these buttons is for them to do what their text says they will do. One button for turning on and off the system, one for starting the game and lastly one for restarting the game in case of a game over. With the help of these buttons, the user will be able to move the game along at their own pace instead of having to keep up with an automated system that keeps moving forward

```
<div class="GameBoard_Logo">
    <table>
        <tr id="GameBoard_Logo_Top">
            <td><h1>Simon Says</h1></td>
        </tr>

        <tr>
            <td>On/Off</td>
        </tr>

        <tr>
            <td><Button class="GameBoard_Logo_Button OnOffButton"></Button></td>
        </tr>

        <tr id="GameBoard_Logo_Bottom">
            <td><p>Start</p></td>
            <td><p>Restart</p></td>
        </tr>

        <tr id="GameBoard_Logo_Bottom">
            <td><button class="GameBoard_Logo_Button StartButton"></button></td>
            <td><button class="GameBoard_Logo_Button RestartButton"></button></td>

        </tr>

    </table>
```

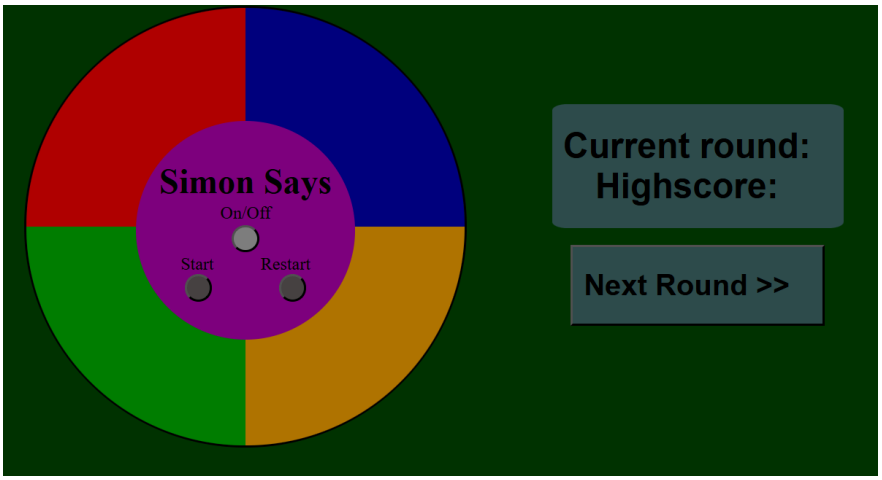


4.1.5 Round Counter, Highscore tracker and a button

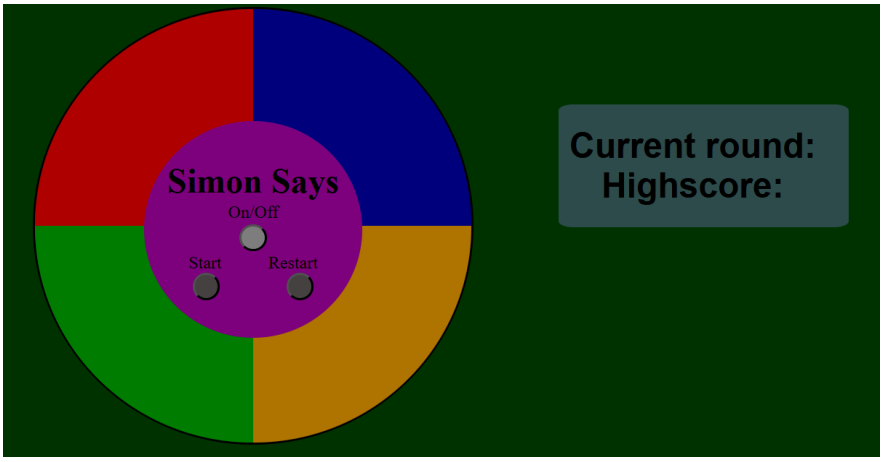
On this step a visual display for the current round and highscore was created, along with a button used for continuing the game after either losing or winning a round. On the right side of the game board, all three elements were placed inside the container named "UI_Buttons_Container". Originally, the container was given a position of "relative" but due to the container pushing the gameboard to the left, it was given the position of "absolute". What position absolute does is keep the element in its assigned position, no matter what.

The button in the container is named “NextOrRestart_Button”. As the name suggests, the button has two functionalities, Continuing the game and restarting the game. As default, the button is invisible and whenever a round has finished, the button shows up displaying either “Next” or “Restart”, depending on the round’s outcome.

When the players loses or wins a round:



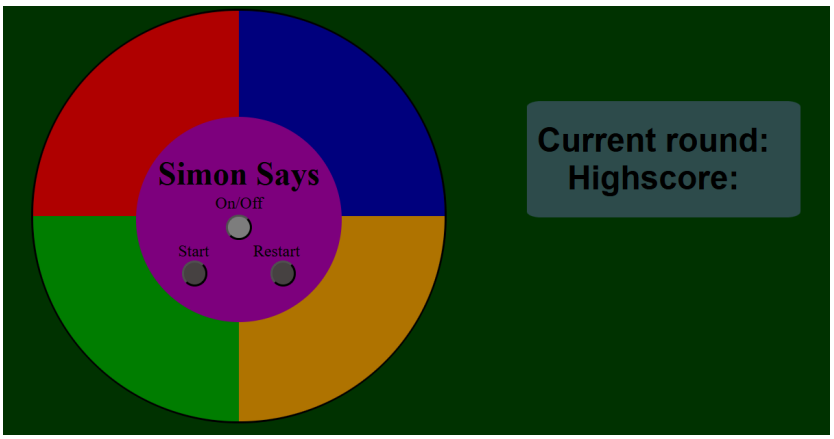
Default:



4.1.6 Media Queries

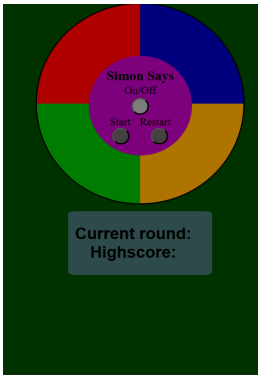
Media queries are instructions given in css that adapt the webpage’s elements depending on what screen size the participant is using. If the user’s screen has a width of above 1190px, the webpage’s content will be displayed with the original design and measurements. If the width is below 1190px, the UI_Buttons_Container will be positioned underneath the gameboard. If the width goes any lower than that, all objects on the page will gradually get smaller and smaller until it runs out of space.

Above 1190px



Below 1190px

Below 390px



4.2 Javascript

4.2.1 Setting Up a structure

To not get lost in the code, a structure is set up dividing the different programming elements. At the top are DOM-elements, used for selecting elements from the html. Below them are variables. These are used to store data such as strings of text or numbers. Below them are event listeners. They are the location where instructions are given for when a button is pressed. At the bottom of the structure are the functions. If there is a piece of code that's used in multiple locations, it can be put into a function for easier use. When the piece of code is in the function you just have to write the function's name where you need to use it. This makes the code simpler, non-repetitive and easier to use.

Structure Example

```
// DOM-element
const TopLeft_Button_Test = document.querySelector(".TopLeft_Button");

//_____________________________________________________________________

// Variable
const Message = "TopLeft_Button was pressed";

//_____________________________________________________________________

// Event Listener
TopLeft_Button_Test.addEventListener('click', function() {
    console.log(Message);
    Data();

});

//_____________________________________________________________________

// Function
function Data()
{
    console.log("This information was in the data function");
    console.log("This information was also inside the data function");
}
```

The real DOM-elements

```
const GameBoard_Container = document.querySelector(".GameBoard_Container");

const TopLeft_Button = document.querySelector(".TopLeft_Button");
const TopRight_Button = document.querySelector(".TopRight_Button");
const BottomLeft_Button = document.querySelector(".BottomLeft_Button");
const BottomRight_Button = document.querySelector(".BottomRight_Button");

const OnOffButton = document.querySelector(".OnOffButton");
const RestartButton = document.querySelector(".RestartButton");
const StartButton = document.querySelector(".StartButton");

const RoundDisplayer = document.querySelector(".RoundDisplayer");
const HighScore_Displayer = document.querySelector(".HighScore_Displayer");


const NextOrRestart_Button = document.querySelector(".NextOrRestart_Button");
```

The real variables

```
let OnOffButton_BootUp_ShutDown = false;
let OnOffButton_Disabled = false;
let StartButton_Disabled = true;
let RestartButton_Disabled = true;


let playerTurn = false;
let playerStep = 0;


let RoundCounter = 0;
let HighScore_Tracker = 0; // Latest

let gameStarted = false; // Changed


let WonRound = false; // NextOrRestart
let LostRound = false; // NextOrRestart


let RestartHasBeenPressed = false;



const names = ["TopLeft_Button", "TopRight_Button", "BottomLeft_Button", "BottomRight_Button"];
```

One of the eight real event listeners

```
OnOffButton.addEventListener('click', function() {
    if (OnOffButton_Disabled) return;

    OnOffButton_BootUp_ShutDown = !OnOffButton_BootUp_ShutDown;

    if (OnOffButton_BootUp_ShutDown === true)
    {
        BootUP();

    }

    else
    {
        console.log("Console has been turned off");
        ShutDown();
    }
});
```

Two of the fifteen real functions

```
function GameButtonsUnlocked()
{
    console.log("GameButtons unlocked!");
    OnOffButton_Disabled = false;
    StartButton_Disabled = false;
    RestartButton_Disabled = false;

    OnOffButton.style.backgroundColor = "";
    StartButton.style.backgroundColor = "var(--GameBoard_Logo_OnOffButton_bg-color)";
    RestartButton.style.backgroundColor = "var(--GameBoard_Logo_OnOffButton_bg-color)";
}


function GameButtonsLocked()
{
    console.log("GameButtons locked!");
    OnOffButton_Disabled = true;
    StartButton_Disabled = true;
    RestartButton_Disabled = true;

    OnOffButton.style.backgroundColor = "var(--GameBoard_Logo_Button_bg-color)";
    StartButton.style.backgroundColor = "";
    RestartButton.style.backgroundColor = "";
}
```

4.2.2 Coding the On/Off button.

```
function BootUP()
{
    console.clear("");
    console.log("System booting up...");
    GameButtonsLocked();

    setTimeout(() => {
        TopLeft_Button.style.filter = "brightness(150%)";
    }, 500);

    setTimeout(() => {
        TopRight_Button.style.filter = "brightness(150%)";
    }, 750);

    setTimeout(() => {
        BottomRight_Button.style.filter = "brightness(150%)";
    }, 1000);

    setTimeout(() => {
        BottomLeft_Button.style.filter = "brightness(150%)";
    }, 1250);



    setTimeout(() => {
        TopLeft_Button.style.filter = "brightness(300%)";
        TopRight_Button.style.filter = "brightness(300%)";
        BottomRight_Button.style.filter = "brightness(300%)";
        BottomLeft_Button.style.filter = "brightness(300%)";
    }, 1500);

    setTimeout(() => {
        TopLeft_Button.style.filter = "";
        TopRight_Button.style.filter = "";
        BottomRight_Button.style.filter = "";
        BottomLeft_Button.style.filter = "";

        GameButtonsUnlocked()
        console.log("System has booted up");
    }, 1750);

}
```

Whether the player turns the system on or off, a blinking sequence is played to signal the player of which it is. For a boot up, the buttons light up one after another in a clockwise manner before they all turn off again once all buttons have been lit up. For a shut down the four buttons light up two at a time. First the top two and then the bottom two before they all turn off again as well. The Boot up function

The Shut down function

```
function ShutDown()
{
    console.clear();
    console.log("System shutting down...");
    GameButtonsLocked();

    setTimeout(() => {
        TopLeft_Button.style.filter = "brightness(150%)";
        TopRight_Button.style.filter = "brightness(150%)";
    }, 500);

    setTimeout(() => {
        BottomLeft_Button.style.filter = "brightness(150%)";
        BottomRight_Button.style.filter = "brightness(150%)";
    }, 750);


    setTimeout(() => {
        TopLeft_Button.style.filter = "";
        TopRight_Button.style.filter = "";
        BottomLeft_Button.style.filter = "";
        BottomRight_Button.style.filter = "";

        OnOffButton_Disabled = false;
        OnOffButton.style.backgroundColor = "";

        console.log("System has been shutdown");

    }, 1000);
}
```

The event listener that determines whether the system is booting up or shutting down

```
OnOffButton.addEventListener('click', function() {
    if (OnOffButton_Disabled) return;

    OnOffButton_BootUp_ShutDown = !OnOffButton_BootUp_ShutDown;

    if (OnOffButton_BootUp_ShutDown === true)
    {
        BootUP();

    }

    else
    {
        console.log("Console has been turned off");
        ShutDown();
    }
});
```

4.2.3 Making a random generator

For the Simon Says game to be able to have a random order of blinking buttons, it requires a name generator. For a random name generator to work it needs to grab the randomness from something, whether that be numbers, words, letters, etc… For this generator it is the names of the game's four blinkable buttons.

```
const names = ["TopLeft_Button", "TopRight_Button", "BottomLeft_Button", "BottomRight_Button"];
```

After the names have been generated, they need to be handled somewhere. In this instance it is in a function named ”PlaySequence”. What this function does is that it accepts the sequence or the list of names provided and makes sure to go through each individual item with an interval. If the names were to directly be thrown into the sequence, nothing would happen because the names are just pieces of text. For the sequence to be able to understand the generated names, a connection is made under the variable named “ButtonMap”. What this variable does is that the names are translated so that the sequence can understand which buttons they want to do something with. To clarify, the const names could contain any text or numbers without an issue, they’re just pieces of text. It’s the connection in the variable “ButtonMap” that lets the sequence know what DOM-elements they’re trying to reach.

```
function playSequence(sequence)
{
    playerTurn = false;
    let i = 0;
    GameButtonsLocked();

    const interval = setInterval(() => {
        let buttonName = sequence[i];
        FlashButton(buttonName);
        i++;

        if (i >= sequence.length)
        {
            clearInterval(interval);

            setTimeout(() =>
            {
                playerTurn = true;
                playerStep = 0;
                GameButtonsUnlocked();

                StartButton_Disabled = true;
                StartButton.style.backgroundColor = "";

                console.log("Player's turn: Active");
            }, 600);
        }

    }, 800);
}
```

```
const ButtonMap = {
    "TopLeft_Button" : TopLeft_Button,
    "TopRight_Button" : TopRight_Button,
    "BottomLeft_Button" : BottomLeft_Button,
    "BottomRight_Button" : BottomRight_Button
};
```

The function named “PlaySequence” accepts and handles the list of names randomly generated, but for the function to be able to make the buttons blink, another function is required. This function is called “FlashButton” and as the name suggests, it causes the buttons to flash. As mentioned before, a function is often used for making code simpler and that is precisely what is done here. Each time the function is called, it lights up the button that’s currently active and then turns it off again after 400ms.

4.2.4 Making the player able to imitate the sequence

For this section the function named “HandlePlayerInput” was added. As the name suggests, the function handles the player input. The input handler works in the way that when one of the four blinkable buttons get pressed, it sends a check through the function to see if it matches with what the game had in its sequence.

In the function the possibility of being able to continue the game was added if the player lost or won the round. In a round of Simon Says there are two possible ways of continuing the game.

Outcome 1

1. The player wins the round. The four buttons blink twice and a clickable button shows up on screen displaying the text “Next Round >>”.

Outcome 2

2. The player loses the round. The four buttons make one long blink before a clickable button shows up in the same place as the last, but this time displaying the text “Reset Game”. After you’ve pressed the reset button, the text changes to “Start Game” which if pressed starts the game again from the beginning with a new randomized blinking pattern.

Apart from the game logic, a highscore and a round counter was added. When the “Next Round” button is pressed, the round counter increases by one so that the player knows which round they’re on.

The highscore tracker works in the way that if the round counter is the same as the highscore’s highest number, then the two of them increase together. If the game is lost then the round counter is reset to zero and the highscore remains on the highest reached round.

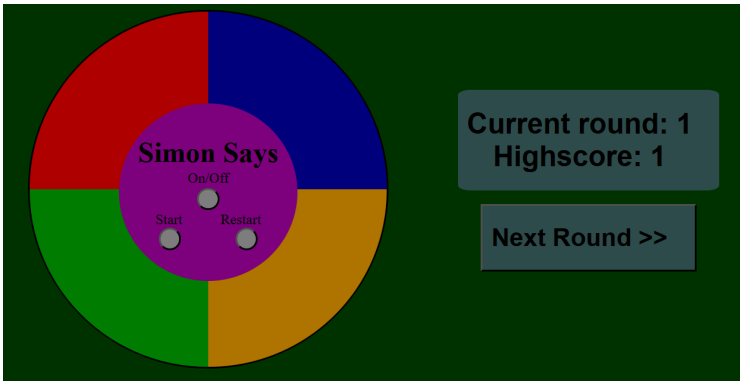
The input handler

```
TopLeft_Button.addEventListener('click', () => HandlePlayerInput("TopLeft_Button"));
TopRight_Button.addEventListener('click', () => HandlePlayerInput("TopRight_Button"));
BottomLeft_Button.addEventListener('click', () => HandlePlayerInput("BottomLeft_Button"));
BottomRight_Button.addEventListener('click', () => HandlePlayerInput("BottomRight_Button"));
```

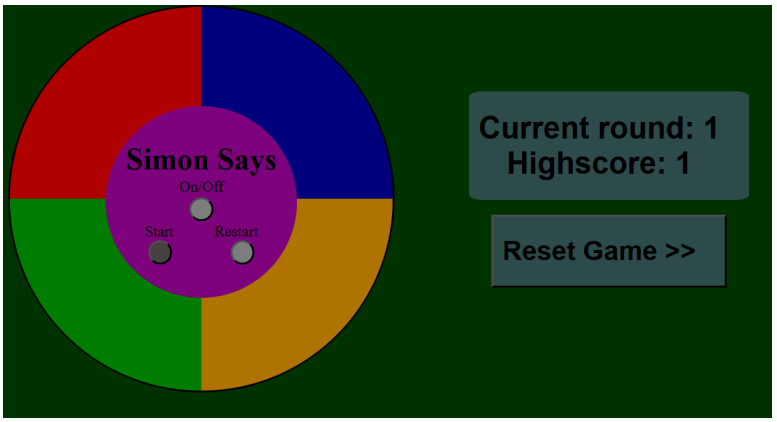
The function

```
function HandlePlayerInput(buttonName)
{
    if (!playerTurn) return;

    FlashButton(buttonName);

    if (buttonName === roundOne[playerStep])
    {
        playerStep++;

        if (playerStep === roundOne.length)
        {
            console.log("Player completed the sequence!");
            GameButtonsLocked();
            NewWaveSequence();
            SequenceCompleted();
        }
    }

    else
    {
        SequenceFailed();
        console.log("Oh... That's not the one... Game Over");
        StartButton_Disabled = true;
        StartButton.style.backgroundColor = "";
        playerTurn = false;


    }

}
```

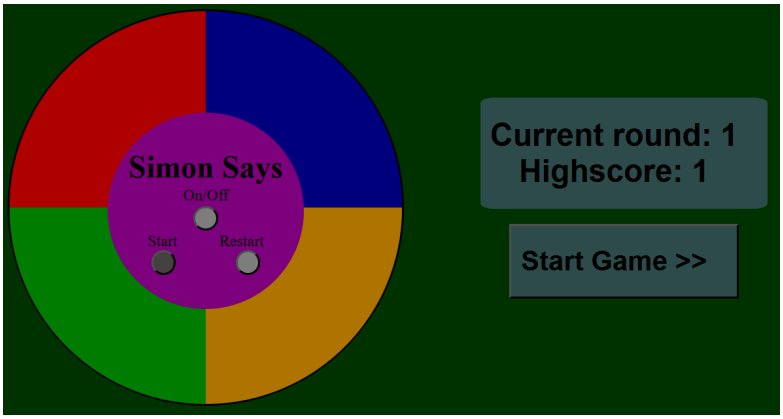
Displaying of the Next Round button



Displaying of the Reset Game button



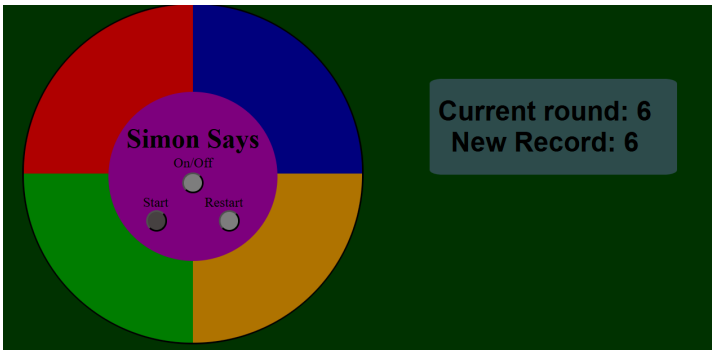
Displaying of the start game button



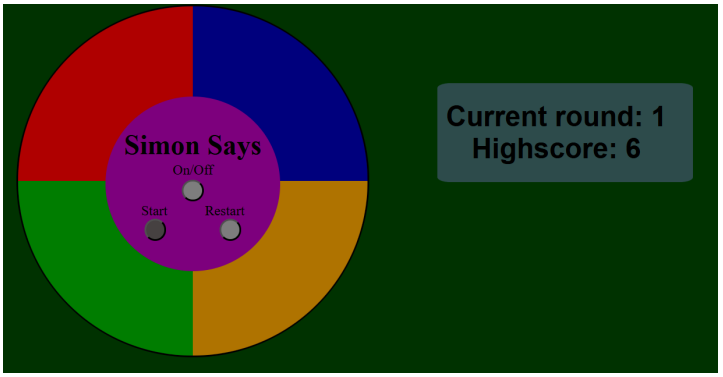
The Round Counter and the Highscore tracker's code

```
else
{
    if (HighScore_Tracker === 0)
    {
        HighScore_Tracker = RoundCounter;
        HighScore_Displayer.textContent = `Highscore: ${HighScore_Tracker}`;
    }

    if (RoundCounter > 0 && RoundCounter > HighScore_Tracker)
    {
        HighScore_Tracker = RoundCounter;
        HighScore_Displayer.textContent = `Highscore: ${HighScore_Tracker}`;
    }
```

The highscore and round counter increasing together



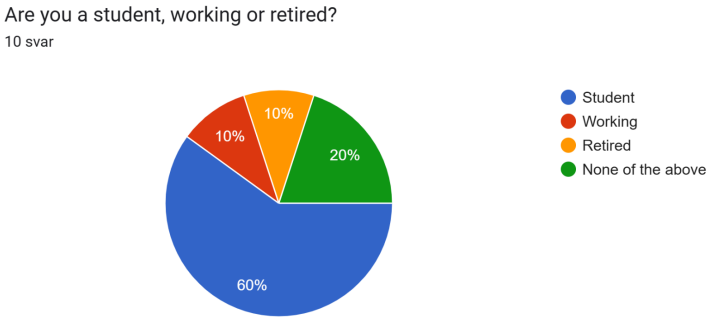
When the game gets restarted and the highscore remains at the highest reached round



5. Google Form Results

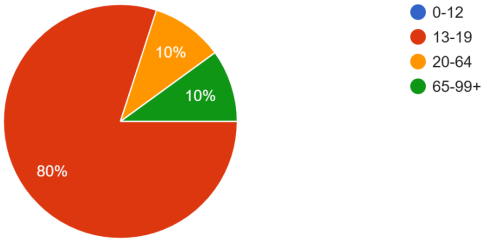
5.1 Feedback Statistics

5.1.1 Before testing the games



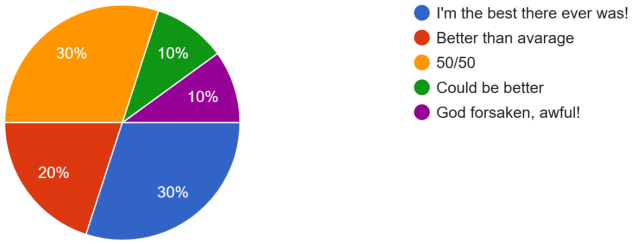
Which of the age groups below do you belong to?

10 svar



This game aims to improve focus. Would you consider your focus to be good?

10 svar



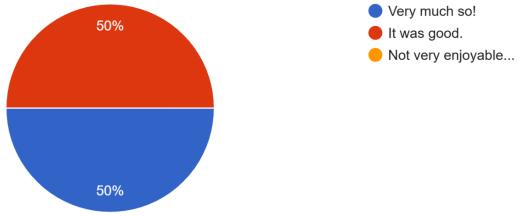
How good would you rate your ability of memorizing things?

10 svar

5.1.2 After testing the games

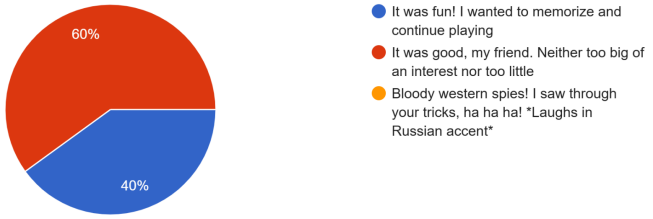
Did the video game come out as enjoyable?

10 svar



Memory... This game aimed towards making the user want to memorize things even though the underlaying purpose was to make the user exercise their mind. How was your experience?

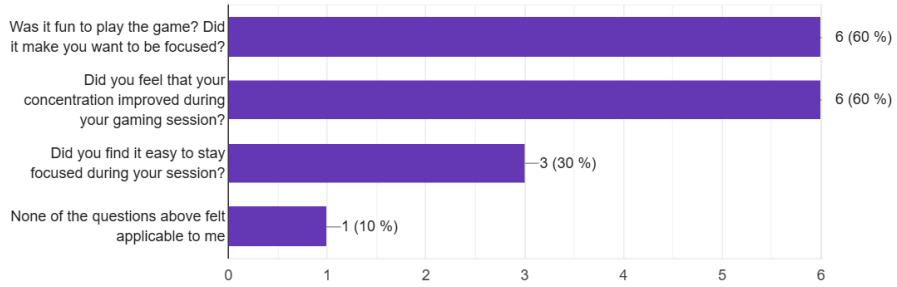
10 svar



Focus... just like the question above, how was the experience?

Kopiera diagram

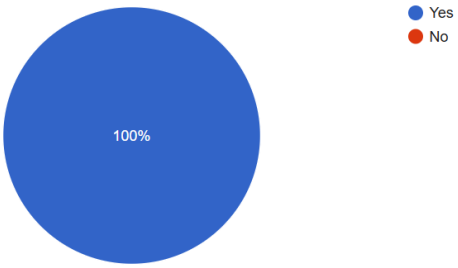
10 svar

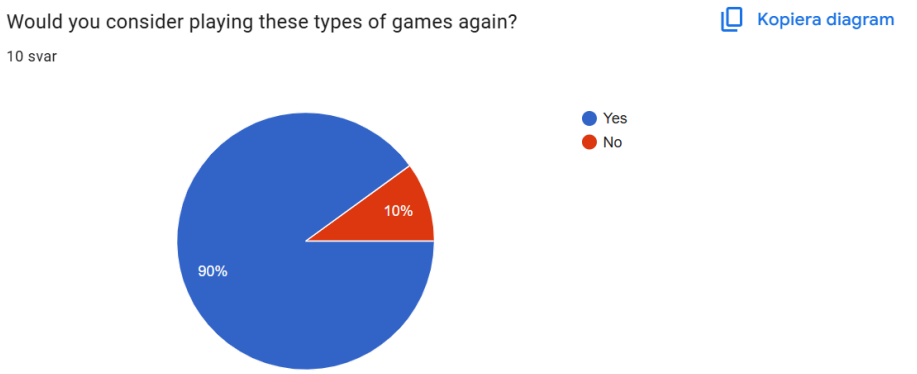


Do you believe games like these could help improve ones mind capabilities in everyday life?

Kopiera diagram

10 svar





6. Analysis & Discussion

6.1 Reflection on only creating a single game

As mentioned in the project's beginning, the original plan was to make 3 small games. One Simon Says, one Memory game and lastly a 2D Running game. Due to time constraints, the project was modified towards only presenting one of these games, which in this case ended up being the Simon Says.

6.2 Reflection on the amount of participants gathered.

During the project, it was estimated that 10-15 individuals would be willing to participate in testing the Simon Says game. Because of other studies the thesis was put on hold until the winter break where there finally was time to finish the Google Form. The Form was finished just after new years eve but due to an unexpectedly heavy winter, going out in traffic was prohibited by the media. Online links were given out to friends and family but for the sake of finishing this thesis on time, the feedback's waiting time was stopped at 10 responses on 09/01-2026.

6.3 Results On Research Questions

- **Did the game come out as enjoyable even though it was meant as an exercise?** Absolutely. Out of the 10 participants, 100% responded with a yes, whether that was with the response of the game being good or the game being great.

- **Did the participants feel that the game had potential for improving their focus and memory? If so, did the two cognitive abilities have the potential for improving equally as much?** On this question, there was a tiny difference between focus and memory. Just like the question above, when asked about their focus, all 10

participants responded with a solid yes. No matter which option the participants picked, they all centered around the game helping them in some way.

When asked the same question regarding memory, there was a deviation. Out of the 10 participants, one user responded that the game did not have the possibility of improving their memory. Due to not foreseeing such a tiny deviation from the majority, the Google Form did not have a section where the participants could explain their reasoning. In future research, such a section will be implemented but for this project the deviation will be left open for self speculation. As a conclusion, the majority of the participants deemed the game to have the possibility of improving their cognitive abilities with a ratio of 9 : 1.

- **Would the participants consider playing a game like this again in the future or do we need to adapt our idea somehow?** Out of the 10 participants, 9 responded with a yes, they would consider playing these types of games again. Since the majority responded with an interest in trying these types of games out again and no section had been prepared for a reasoning from the user, the games will continue being made with the same core idea as from the start.

7. Conclusion

Once the game had been tested and the feedback had been written, it was concluded that the game had indeed potential for improving focus and memory. On a scale of 0-100, all 10 participants responded with yes, the game was indeed enjoyable. The game made them want to stay focused and they all agreed that games like these have the potential for improving their cognitive abilities. As a final note, 9 out of the 10 participants responded with a resounding yes towards playing these types of games again in the future, which as a conclusion will mark the game as a solid success.

As stated earlier in the limitations section, due to time constraints an analysis was made examining the potential of a video game improving cognitive abilities such as focus and memory. This means that all conclusions leading towards the game having the potential to mentally strengthen individuals, is solely based on the participants opinions and beliefs, not concrete proof or evidence.

Sources

[1] Open AI (2026) ChatGPT [Large Language Model]
https://chatgpt.com/

[2] Engström, M (2026) Simon Says Game [Student Project]
https://m4ri7i42.github.io/SimonSays_Game/

[3] Engström, M (2026) Simon Says Game, Google Form [Online Form]
https://forms.gle/9hemMJcYKjmTHhGj9