Assignment 1 - Memory Training Game ENG1003, Semester 1, 2019

Due: Friday 5th April 18:00
Worth: 12% of final mark

Aim

For this assignment your team will develop a memory skill game based on the electronic game Simon, produced by Hasbro. You will create a web app that displays random sequences of colours which the user must repeat from memory. As the user continually enters correct sequences the game progressively selects longer sequences.

Background

Simon is a 40-year-old electronic game, currently produced by Hasbro (`https://en.wikipedia.org/wiki/Simon_(game)`). The game has four coloured buttons that light up and play a sound in sequence. The user then has to replicate the sequence by pressing those buttons in the same order. The sequences become progressively longer and more difficult until the user is unable to repeat the sequence and the game ends.

While such games are entertaining, the complexity and randomness of the sequences also allows the game to be used as a means of practicing memory storage techniques. There is some evidence that individuals with mild cognitive impairment can benefit from memory enhancement training[1] from games such as Simon.

[1]http://dx.doi.org/10.1080/13607860120101077

Your team has been hired by *Thoughtress*, a small firm that develops memory skill games for researchers and individuals to evaluate improvement in memory ability. Normally they develop these games as physical electronic devices, however they wish to make their services more accessible to those in rural areas. To this end your team has decided to develop the game as a mobile web app.

Background: Example sequences

Below are some example sequences that might be shown when playing the game:

Seq#/Pos	**colour 1**	**colour 2**	**colour 3**	**colour 4**	**colour 5**	**colour 6**	**colour 7**	**colour 8**	. . .
Seq1	**Red**	**Red**	**Red**	**Red**					
Seq2	**Red**	**Green**	**Blue**	**Green**					
Seq3	**Blue**	**Green**	**Yellow**	**Yellow**	**Blue**				
Seq4	**Green**	**Yellow**	**Red**	**Blue**	**Red**	**Green**	**Blue**	**Red**	

Table 1: sample colour sequences

Background: Game progression

The difficulty level (sequence length) increases after the user gets a certain number of sequences of that length correct. The number of correct sequences needed to advance is shown in the table below. (The number is always two less than the sequence length.)

Sequence length (level)	**Correct sequences needed to advance**
4 (initial)	2
5	3
6	4
7	5
8	6
etc.	etc.

Table 2: sequence length progression table

For example, at the start of the game, the player is given sequences of length 4. The player receives RRGB followed by GGBY and they get both correct. Since they have entered two sequences correctly, the game now increases to sequences of length 5. The player then receives YYGBR (a sequence of length 5) and needs to enter 3 of further correct sequences to advance to sequences of length 6.

Any time the player makes a mistake, they start again at the previous level (or reset to level 4 if they make two consecutive errors). The player cannot be given sequences shorter than 4 (as this is the starting level).

Background: Angles obtained from the phone using device orientation

There are some situations where it may be too difficult for the user to interact with the app using the buttons on the screen (for example while wearing gloves). As such, we want to allow users to control the app using the phones orientation.

When using device orientation there are three angles that you have access to: alpha, beta and gamma; where beta represents the forwards-backwards tilt of the phone. It can visualised as below:

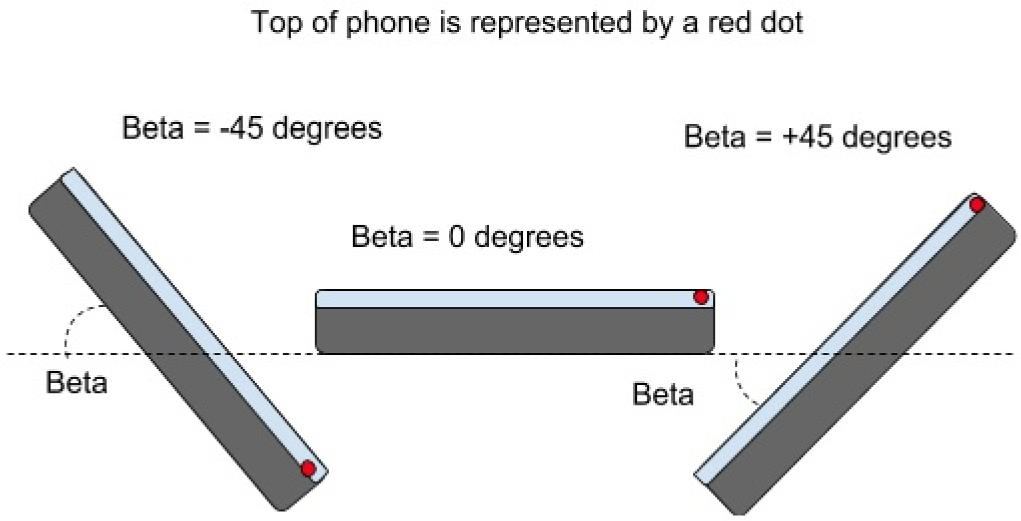

Figure 1: side view of phone at different beta angles

In figure 1, where we are looking at the phone from the side, this means that when the device is flat on a surface, screen facing up it will have a beta of 0 degrees, tilting the top of the phone down gives a beta of -45 and tilting the top of the phone up gives a beta of +45.

In figure 2, where we are looking at the bottom of the phone, being flat on a surface gives a gamma of 0 degrees, tilting the phone to the left gives it a gamma of -45 whereas tilting to the right gives it +45 degrees.

You can observe these Beta and Gamma values on the team phone using the ENG1003 Sensor Test app:

`https://eng1003.monash/apps/sensortest/`

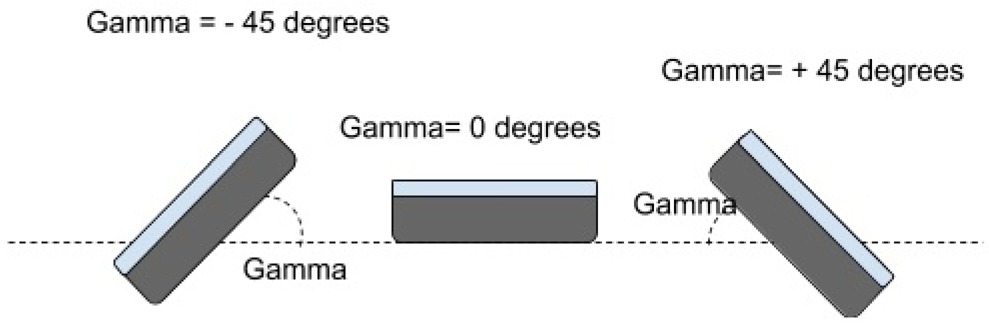

Figure 2: bottom-side view of phone at different gamma angles

What you are provided with

We have provided a skeleton version of the app, which displays the basic interface of the app and includes:

- Code that triggers the *userChoiceTimeout* function in `gameLogic.js`
  - This function is called when the user fails to make a decision after 2 seconds.
- Code to trigger the *giveNextSequence* function in `gameLogic.js`
  - The *giveNextSequence* function will be called automatically every time the user clicks on the play button.
  - This function should return a list of strings which can include the following:
    - "blue"
    - "green"
    - "yellow"
    - "red"

    For example ["blue","blue","green"] which would correspond to a sequence of **Blue** **Blue** **Green**
- Code to trigger the *sequenceHasDisplayed* function in `gameLogic.js`
  - This function is called when the sequence that has been returned by the *giveNextSequence* function has finished being shown to the user.
  - This is an opportunity to display information to the user prior to them entering a sequence.
- Code to trigger the *changeMode* function in `gameLogic.js`
  - This function will be called when the user switches input modes between touch input and tilt input. The initial input method is touch.
  - This function has a parameter mode which will contain *TOUCH_MODE* or *TILT_MODE* depending on the mode the user has just switched to.

- The *showSuccess* and *showFailure* functions
  - These functions can be called by your code to display an image of a tick or cross based on whether the user has entered a correct or incorrect sequence.
  - The tick or cross will automatically disappear after one second.
- Four coloured buttons arranged in a circle on the screen
  - These buttons are designed to be clicked (or tapped on) by the user.
  - When clicked, each of these buttons will result in a call to the *buttonSelected* function in `gameLogic.js`
  - When the buttonSelected function is called, it will be passed with one of the following strings as an argument representing which of the four buttons was clicked on:
    - "blue"
    - "green"
    - "yellow"
    - "red"
- A Play button on the app's title bar which displays the next sequence for the user to enter and results in each request to your *giveNextSequence* function.
- A *displayToastMessage* function which you can call to have a "toast" message pop up at the bottom of the screen to notify the user of some information.

Other than this, you don't need to know anything else about the skeleton to use it. You need to implement the functionality described below.

The app skeleton can be downloaded from the ENG1003 Moodle site. The app skeleton is titled **assignment1.zip** and contains a file `gameLogic.js` where you should write your submission.

You don't need to understand the other files we provide you with as part of the project skeleton – they include some concepts we haven't yet covered.

What you need to do

Getting started

1. As a team you should download a copy of the project skeleton and unzip this.
2. The `assignment1` folder contains the code of the web app.
3. Open the `gameLogic.js` file. You will implement your solution in this file.

4. The `gameLogic.js` file can be uploaded to the ENG1003 web server via the ENG1003 Assignment Uploader web page (see below) and run on your team smartphone.
5. You should discuss within your team on the breakdown of coding features and practice pair programming.
6. Plan to have steady progress rather than completing the assignment at one go, or to do it at the last minute.

Programming Tasks

The Programming component of this assignment is worth 9% of your unit mark.

For the programming tasks, we suggest you complete each part together before moving onto the next one.

Feature 1: Generating random sequences

The first step to displaying sequences on the screen is to generate those sequences. This should be done in the *giveNextSequence* function and returned as an array of strings. You should use the **random** method of the **Math** package to produce random numbers to represent any given option. Note: Your code will need to save the generated sequence for checking against the sequence entered by the user.

Feature 2: Sequence input

The *buttonSelected* function will be called every time the user clicks on one of the coloured buttons. You should keep track of which buttons the user presses and the order they were pressed in.

Once the full sequence has been entered, you should compare what the user has entered against the original sequence. If the sequences match then you should call the *showSuccess* function, otherwise you should call the *showFailure* function. (Both functions are part of the app skeleton.)

Feature 3: Increasing difficulty level

You should keep track of how many consecutive sequences the user has entered correctly at the current level. When the user gets enough correct in a row (see background, game progression (page 2)), the game should advance to the next level.

Feature 4: Resetting difficulty level

If the user enters an incorrect sequence, they should be taken back to the previous level. For example, if they provide an incorrect sequence for a sequence of length 6, they need to do sequences of length 5 again (3 of them) before they get a sequence of length 6 again.

If the user makes two mistakes in a row, they are reset to the starting sequence length of 4.

This means that you will need to track the level the player is on and the number of successful/unsuccessful sequences entered.

Feature 5: Displaying status and progress

You should show on the screen (via the label with the id ‘output’) the current progress of the user, including:

- Number of items remaining to be entered for the current sequence.
- Length of the current sequence
- Number of correct sequences entered at the current level
- Remaining additional correct sequences until they advance to the next level

You should also display general status such as whether the user is currently expected to enter a sequence or watch a sequence.

This progress information should be updated whenever any of this information changes.

Feature 6: Tilt-based input method

You should add an alternative control scheme where the user tilts the phone towards the button they wish to light up. In order to do this, the app must be able to respond to changes in the orientation of the device using the *DeviceOrientation* event. See the Background “Angles obtained from the phone using device orientation” section (page 3).

The app has a button to switch between input modes which results in a call to your *changeMode* function which has a parameter mode which will either be *TOUCH_MODE* or *TILT_MODE* depending on the mode. You may need to remember this value in your code.

You should modify your app so that it can track the orientation of the phone. When the phone is tilted toward the bottom-left of the phone this should be treated as the user attempting to select the bottom-left button (i.e., the yellow button). Likewise for the other three buttons.

There is already code in the `main.js` file (supplied with the skeleton) which triggers the buttons to light up and runs the *buttonSelected* function (that you've already implemented). Each of these functions are named for the corresponding buttons they light up: *selectYellowButton*, *selectRedButton*, *selectBlueButton* or *selectGreenButton* (also aliased as *selectBottomLeftButton*, *selectBottomRightButton*, *selectTopLeftButton* and *selectTopRightButton*).

The *DeviceOrientation* JavaScript event will notify you (via a callback function) whenever the device orientation changes, but this might be many times a second. For this reason we don't want to select a button as a result of every orientation change, but rather just store the button that would be selected, and then we can actually register this selection after the standard button press timeout occurs (i.e., a call to your *userChoiceTimeout* function). Hence, if the app is in tilt control mode when your *userChoiceTimeout* function is called you should see if the phone is tilted and if so then select the appropriate button.

Feature 7: show tilt while in tilt mode

While the app is in tilt mode, you should ensure that you display the current tilt of the phone visually. This should be done in such a way that it's clear to the user which button will end up being selected on userChoiceTimeout.

Presentation

This assignment includes a presentation component worth 3% of your unit mark.

Thoughtress have had their representatives look over the app your team has produced and are pleased with the results however despite initially agreeing with your team's suggestion of a mobile web, they now ask your team to get the app in the Google Play^{TM} store and Apple AppStore^{TM}. In other words they are asking your team to completely redesign the app as a native app for Android and iOS devices. Your team has little experience with iOS apps however you have worked closely with another organisation (*IntellAppt*) in the past with extensive experience with both Android and iOS development environments. After some negotiation, it is agreed that *IntellAppt* will handle this redevelopment while your team takes more of a support role and a handover presentation organised for three weeks time.

In your Week 6 prac class your team will deliver a handover presentation. Your team should present an overview of the functionality, design of the code and any specific hardware requirements. You should warn the new team about any current issues in your app as well as provide any suggestions for improvements. As with any good presentation, it should be prepared well in advance of the due date (including any visual aids) and it should be well rehearsed as a group and individually.

Format

Each student team will deliver a 10-minute oral presentation (in prac class) describing and demonstrating the functionality of their app and detailing any limitations of the application. Every member of the team should present for 2-3 minutes.

- The target audience for this presentation is another team who will be extending the project further.
- This presentation would be delivered in a formal business setting and so all team members should be dressed appropriately.
- This presentation must discuss the structure and functionality of the application as well as any design decisions made.

Testing the app

Upload your code to the ENG1003 server using the assignment upload page. The uploader can be found here:

`https://eng1003.monash/uploader/`

Once you have uploaded the code, you can view the uploaded assignment by selecting the appropriate assignment from the dropdown list on the assignment viewer. The viewer can be found here:

`https://eng1003.monash/view`

Resources

- Mozilla Developer Network - Math.random documentation
- Mozilla Developer Network - Detecting Device Orientation

Submission

Your team should submit their final version of the application online via Moodle. You must submit the following:

- A zip file named based on your team (e.g., "Team014.zip").
  This should contain ONLY one file named `gameLogic.js`
  This should be a ZIP file and not any other kind of compressed folder (e.g. .rar, .7zip, .tar).

The submission should be uploaded by the team leader and must be finalised by Friday 5th April 18:00 (local time). *Please note: Your entire team needs to accept the assignment submission statement individually on Moodle.*

You also need to individually complete the CATME peer assessment survey as described below.
You also need to individually complete the following tasks:

- CATME peer assessment survey
- Assignment code interview

Your presentation will be given during your practical classes in Week 6 (8th - 12th August 2018).

Marking criteria

Programming tasks

Your assignment will be assessed based on the version of `gameLogic.js` file you submit via Moodle. Before submission check your code still works with the original app skeleton, in case you have modified your copy of any of the other files. We will run it with the original app skeleton and test it on your team smartphone. We will use the same phones when marking your assignments.
Assessment criteria:

- Whether the app functionality satisfies the assignment specification
- Quality of app source code, including structure and documentation

You will be marked as a group, however your individual marks will be subject to peer review moderation based on CATME feedback and your assignment interview.

A detailed marking rubric will be available on the unit Moodle page.

CATME Peer assessment

You are expected to work together as a team on this assignment and contribute roughly equal amounts of work. Peer assessment will be conducted via the CATME online system. You will receive email reminders at the appropriate time.

Not completing the CATME peer assessment component may result in a score of zero for the assignment.

Do:

- Give your teammates accurate and honest feedback for improvement
- Leave a short comment at the end of the survey to justify your rating
- If there are issues/problems, raise them with your team early
- Contact your demonstrators if the problems cannot be solved amongst yourselves

Do NOT:

- Opt out of this process or give each person the same rating
- Make an agreement amongst your team to give the same range of mark

Assignment code interview

During your Week 6 prac class your demonstrator will spend a few minutes interviewing each team member to individually gauge the student's personal understanding of your Assignment 1 code. The purpose of this is to ensure that each member of a team has contributed to the assignment and understands the code submitted by the team in their name.

You will be assigned a score based on your interview, and your code mark will be penalised if you are unable to explain your team's submission:

Category	**Description**	**Penalty**
No understanding	The student has not prepared, cannot answer even the most basic questions and likely has not even seen the code before.	100%
Trivial understanding	The student may have seen the code before and can answer something partially relevant or correct to a question but they clearly can't engage in a serious discussion of the code.	30%
Selective understanding	The student gives answers that are partially correct or can answer questions about one area correctly but another not at all. The student has not prepared sufficiently.	20%
Adequate understanding	The student is reasonably well prepared and can consistently provide answers that are mostly correct, possibly with some prompting. The student may lack confidence or speed in answering.	10%
Complete understanding	The student has clearly prepared and understands the code. They can answer questions correctly and concisely with little to no prompting.	0%

Presentation

Students are marked individually for this assignment on their presentation skills.
Assessment criteria:

- Voice is of appropriate volume, speed and enthusiasm
- Language is appropriate for a formal context and jargon is only used where necessary (and explained if used)
- Eye contact is consistent and covers most of the audience
- Body language complements the presentation
- Explanations are clear and visual aids used appropriately

A detailed marking rubric will be available on the unit Moodle page.

Other information

Where to get help

There will be a FAQ posted in Moodle and updated periodically. You can also ask questions about the assignment on the General Discussion Forum on the unit's Moodle page. This is the preferred venue for assignment clarification-type questions. You should check this forum (and the News forum) regularly, as the responses of the teaching staff are "official" and can constitute amendments or additions to the assignment specification. Before asking for a clarification, please look at the FAQ and forum.

Plagiarism and collusion

Cheating, Plagiarism and collusion are serious academic offenses at Monash University. Students must not share their team's work with any student outside of their team. Students should consult the policies linked below for more information.

```
https://www.monash.edu/students/academic/policies/academic-integrity
https://www.monash.edu/engineering/current-students/enrolment-and-re-enrolment/
course-information/assessment-and-examinations/academic-integrity-and-plagiarism
```

See also the video linked on the Moodle page under the Assignment block.

Students involved in collusion or plagiarism will be subject to disciplinary penalties, which can include:

- The work not being assessed
- A zero grade for the unit
- Suspension from the University
- Exclusion from the University

You are required to reference code that has been obtained or provided by other sources (i.e. online), including formulas for calculating. This should be done within a comment above the code.

Late submissions

We do not accept late submissions without special consideration. Such special consideration applications should be made to the unit email address with a completed form and supporting documentation within two business days of the assignment deadline.

```
http://www.monash.edu/exams/changes/special-consideration
```

Unavailable team members

If team members are missing on the day of the presentation, the remaining members should proceed without them. Missing team members will receive a mark of zero unless they are granted special consideration. Such special consideration applications should be made to the unit email address with a completed form and supporting documentation within two business days of the presentation date.

`http://www.monash.edu/exams/changes/special-consideration`

You must also inform your team members ahead of time if you will be absent on the day of the presentation. If your team has less than three members presenting, notify the unit staff.