Assignment 1 — TapCode reader

ENG1003, Semester 1, 2018

Due: Wednesday 4th April 2018 23:55PM (Local Campus Time)
Worth: 12% of final mark

Aim

This assignment will have you preparing an app that is capable of sending messages using a tap-code over short distances. Tap-codes are convenient forms of communication as there are only two states meaning there is less opportunity for confusion.

Contents

Background

Vicki Valentine is a dance instructor and survivalist who manages an extreme dance school in Parkmore Shopping centre (Keysborough, VIC). Unfortunately, due to declining number of available car spaces, many of her students are no longer able to attend in person.

Previously, Ms. Valentine used a tap code (Figure 2) in her classes to communicate with her students (to familiarise them with non-verbal communication). Currently, she is arranging to teach these students via video but many of them are still enamoured with her tap code and want to continue using it to communicate with her. Ms. Valentine has hired your organisation to help develop an app capable of sending and receiving her tap code between devices.

Your team was originally asked to work on the receiver app while another team develops the sender app however there were some scheduling problems in your organisation and by the time your team was ready to start, the sender app was already prepared. As such you are to prepare a receiver app which works with the existing sender app.

Tap Codes

A tap code is a way of encoding text with sequences of taps. Each element has two sequences, the first gives the row number and the second gives the column number in the corresponding matrix. For instance, a 2 by 3 tap code for the vowels[1] in english might use the following matrix:

	1	**2**	**3**
1	a	e	i
2	o	u	y

***Figure 1**: 2x3 tap code of vowels*

And the code:

* *** ** * ** **

I o u

Would correspond to *"iou"* as the first two sequences include 1 and 3 taps which is the first row and third column (i), the next two are 2 and 1 for second column and first row (o) and the final two are 2 and 2 which has u.

[1] *Note: A tap code doesn't need to represent individual letters, you could equally replace the vowels in the example above with (for instance) "yes", "no", "answer hazy", "definitely", "doubtful" and "cannot predict" and represent Magic Eight ball responses (https://en.wikipedia.org/wiki/Magic_8-Ball)*

To make best use of a tap code, it should be fairly symmetric as this will be more easily readable and generally faster in the worst case.

Below (Figure 2) we include a 5 by 5 tap code for letters in English arranged in such a way as to minimise time taken to code out the most common letters[2].

	1	**2**	**3**	**4**	**5**
1	e	t	a	n	d
2	o	i	r	u	c
3	s	h	m	f	p
4	l	y	g	v	j
5	w	b	x	q	z

Figure 2: *Vicki Valentine's 5x5 tap code*

You will notice that one of the letters (k) is missing here and there's no space. This might make it difficult to represent full sentences!

Vicki's tap code will handle this by replacing any instance of "k" with ***"qc"*** and spaces will be represented as ***"wuw"***.

Using Vicki's tap code, we would represent ***"hurt toes"*** as

" *** ** ** **** ** *** * ** ***** * ** **** ***** * * ** ** * * * *** * **"**

Of course, the duration of a tap is not fixed in stone, so in order to transmit and understand tap-codes, we need to think about how to tell the difference between 3-1 vs 1-3 vs 4 (eg. *** * vs * *** vs ****) and the difference between 4 and 1 (eg. **** vs *)
This is done by having two different kinds of gaps between taps (the half-gap and the full-gap) whose length is dependant on how long a tap goes for, see figure 3 (below)

Element	**Duration**
Tap	N (eg. 0.5 seconds)
half-gap	N (eg. 0.5 seconds)
full-gap	3N (eg. 1.5 second)

We can see that a full-gap is thrice the length of a tap (or half-gap) which means that three half-gaps immediately after one another is a full-gap.

[2] Based on frequency of letters in the complete collected works of william shakespeare

If we look at the text “is” which corresponds to “** ** *** *” we can see it is made up of the following sequence of taps (T) and gaps (G):

G	G	G	T	G	T	G	G	G	T	G	T	G	G	G	T	G	T	G	T	G	G	G	T	G	G	G

We could also represent it as so:

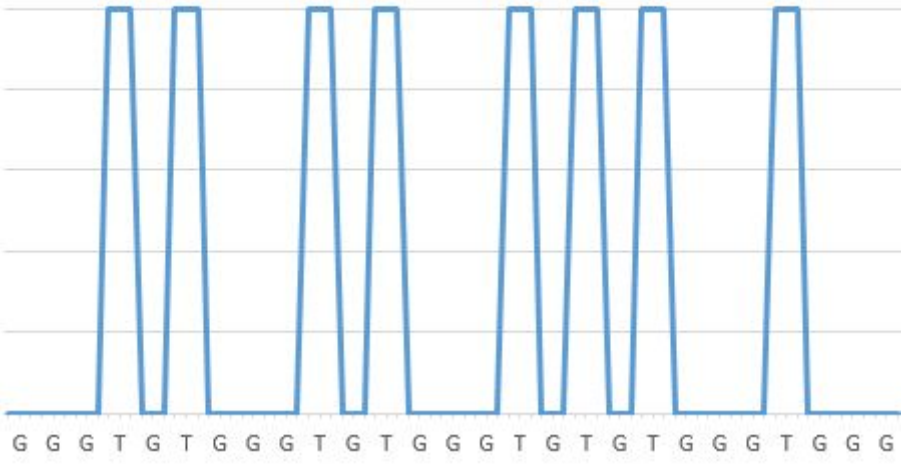
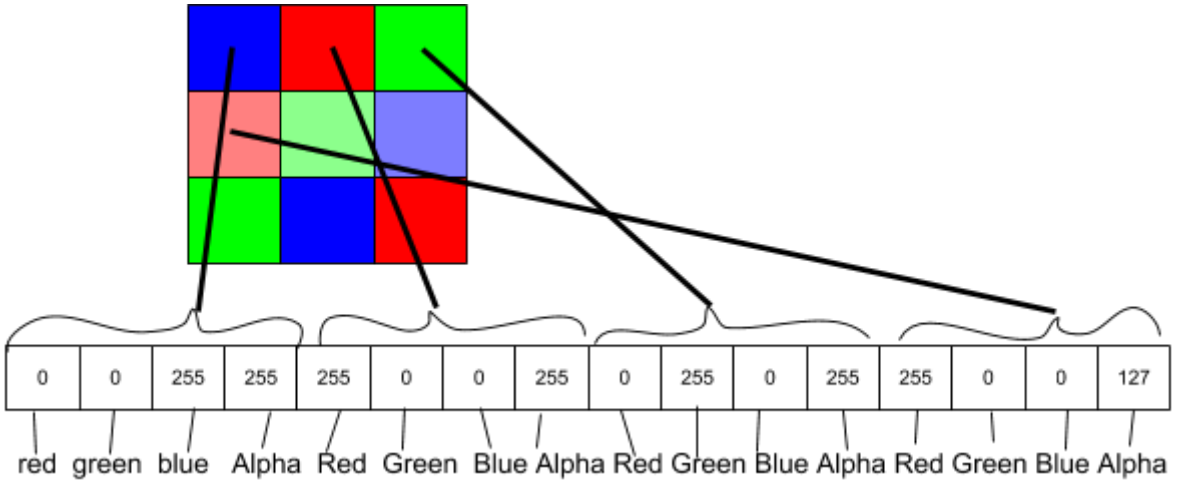


Image data format

The image data passed from the skeleton code to your function is an array of numbers. Every element in the array is a number representing a colour (in RGB) for a given pixel in the order of red, green, blue, alpha. Pixel value range between 0 and 255; for the colours the number represents how much of that colour is in that pixel (255 being the most, 0 being none at all) for instance violet would be 127(red), 0(green), 255(blue). Alpha represents the opacity of the pixel, so 0 means completely transparent and 255 would be completely opaque.

Every set of four consecutive elements is a particular pixel (the red, green, blue and alpha components)

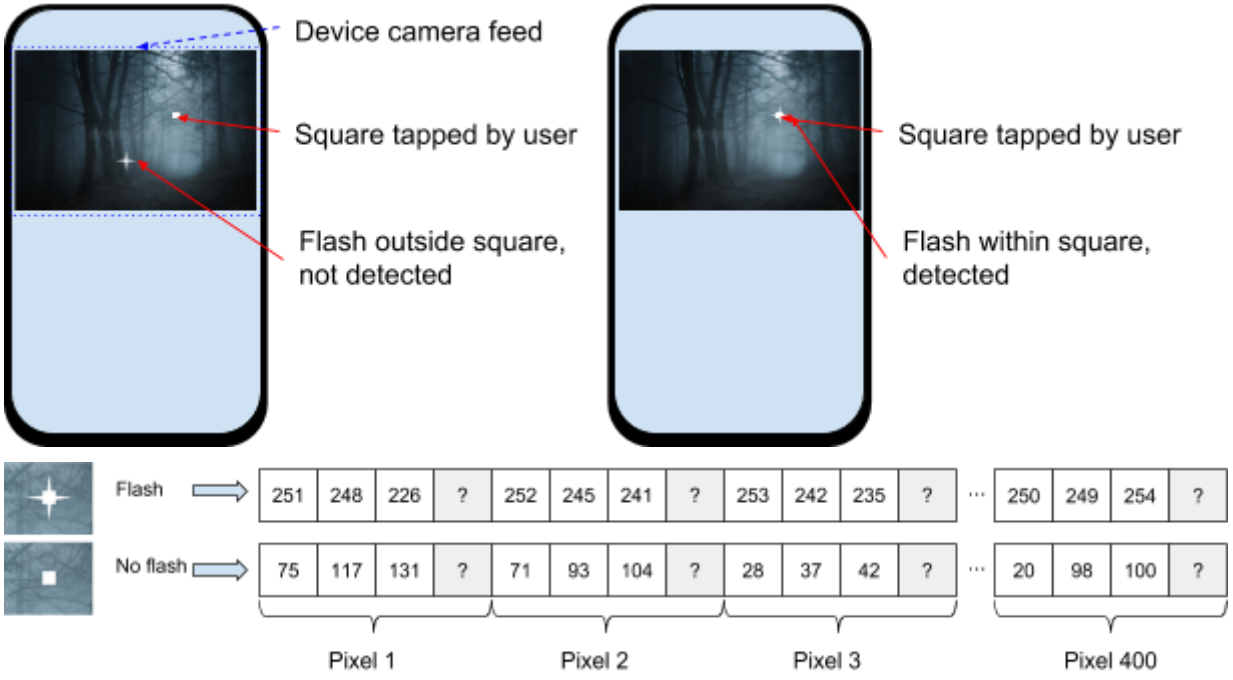
For example the array value for the first four pixels of an image are shown below:



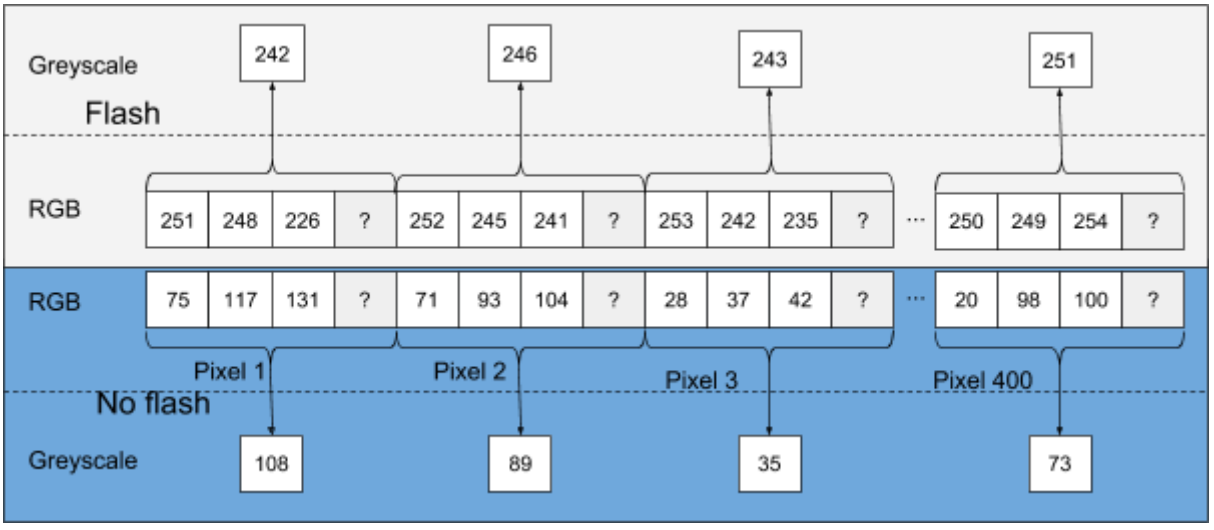
So accessing Image[0] would give the amount of red in the first pixel, Image[2] would give the amount of blue in the first pixel (in this case 255) and Image[4] would give the amount of red in the second pixel.

Using video feed as a light sensor

For this assignment you will be receiving image data *(see background - image data format)* from a video stream from your team phone that corresponds to the mark on the screen (tap to choose where it appears).



The square on the screen is a 20x20 pixel area meaning you will receive an array of 1600 positions corresponding to the 400 pixel area. In order to use this effectively as a light sensor, you will need to convert each pixel to grayscale, to do this you will average the red, green and blue values for each pixel and use this as an analogue for brightness (with 255 being white and 0 being black). For instance:



What you are provided with

We given you an initial skeleton version of the app. This code runs and displays a screen with a camera feed and area for the encoded message (“rx-code”) and decoded message (“rx-translated”). It further includes a button to clear the current code (an X), to reload the camera image (see menu top right) in case the camera goes to sleep, and record/stop recording button (an eye) which starts and stops the triggering of the *_listen* function. You need to implement the further functionality described below.

The app skeleton can be downloaded from the ENG1003 Moodle site. The app skeleton is titled *assignment1.zip* and contains some files and folders. For this assignment, you will only need to edit the file ‘`codeprocessor.js`’. We outline some functions below that you will need to fill in.

This file includes empty function definitions for

- CodeProcessor._listen (See: Steps 1, 2, 3, 5)
    - This should processes the pixel data as it gets updated on the video player and store the results in appropriate form for later access.
    - It will be called on repeatedly as the camera feed updates and will receive an object of the form

      {
      detail: {
      data: [*array of RGBA values*]
      }
      timeStamp: [*an instance of the Date type corresponding to the date/time of the image from the video*]
      }
- CodeProcessor.clear (See: Step 7)
    - This should reset the current state message so that it is as if no message has been sent or received yet.
- CodeProcessor.translate (See: Steps 4, 6, 7)
    - This should display the current message in both coded and decoded forms to the screen into the areas with IDs: “rx-code” and “rx-translated”.

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 '`codeprocessor.js`' file. You will implement your solution in this file.
4. The 'assignment1' folder can be uploaded to the ENG1003 web server via the ENG1003 Assignment Uploader web page (see below) and run on your team smartphone.

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. It is strongly recommended that you practice pair programming rather than trying to split up the coding and attempting parts individually.

Step 1: Conversions between tap code and plain text

Create a data structure that allows you to represent the relationship between characters in plain text and characters in Vicki Valentines 5x5 tap code (figure 2).

Step 2: Convert to grayscale

In the ***CodeProcessor._listen*** function you should begin by converting the received RGB array into grayscale (remembering to ignore the alpha values). This should leave you with the grayscale values for each pixel in the square tapped on the screen.

Step 3: White/Black duration

The ***CodeProcessor._listen*** function will be called every 20ms with a new array of pixels which could correspond to part of a tap or part of a gap (according to the brightness of the square[3]). While you won't know how long a tap or gap lasts initially (as it can be different for each message) you can assume that it won't change while the message is being sent.

Write code that is able to determine whether the current square corresponds to a tap or a halfgap and use the timestamp given with each array of pixels to determine how long a tap corresponds to. You may need to do some investigation to decide what classifies as a tap or a halfgap.

Hint: Assume that the sequence of images starts and ends with a long period of darkness.

[3] Your team will have to determine how bright the pixel array should be to correspond to a tap or gap.

Step 4: Full-gap vs Half-gap

Once your code can reliably determine whether a given square is a tap or a half-gap you should setup code in the ***CodeProcessor._listen*** function capable of determining the difference between a half-gap and a full-gap. This should allow you to generate the coded version of the message (eg. “** ** *** *”).

Step 5: Convert to characters

Setup code in the ***CodeProcessor.translate*** function which can take the coded version of the message and (using your structure from step 1) convert the coded version of the message into characters (plain-text).

Step 6: Handle early termination of message

We expect standard use of the receiver app to be to start recording the screen before transmission begins and stop recording after the transmission ends however it’s possible that the user may accidentally stop recording before the transmission finishes. In this case the last screen of the message might be light (rather than dark). Make sure your code from the ***CodeProcessor._listen*** function is capable of handling this situation.

Step 7: Convert special characters

As the tap code was only capable of representing 25 characters, we used qc to represent k and wuw to represent spaces. Ensure that the ***CodeProcessor.translate*** function correctly updates the plain-text version of your message to replace these with the correct characters.

Step 8 Displaying output and clearing

Finally, by this point you should have a program which can convert between the tap-code and plain text appropriately. Complete the ***CodeProcessor.translate*** so that it uses the “rx-translated” and “rx-code” areas to display the **converted** and **coded** versions of the message respectively. At your choice, you may update this message as each character is converted or you may display it once the full message is obtained.

You should also make sure the ***clear*** method is correctly defined so that you can run the app multiple times without needing to reload the page.

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/

We provide a tap code transmitter application that you can use to test your application with. You can find it at:

> https://eng1003.monash/apps/tapcodetransmitter/

Presentation Task

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

Ms. Valentine contacted your team's line manager to see how much progress has been made and is not optimistic about the app being ready within the time frame she envisioned; as such, she is paying your organisation for the work done so far and bringing in *Glaven and Frink* (a rival software development firm) to complete the final product. She has ask your team to give a presentation to the team from *Glaven and Frink* as part of the handover of the project.

In your week 6 prac class you will give 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.

Format

Each student team will deliver a 10 minute oral presentation (in prac class) describing and demonstrating their app and detailing any issues they encountered. 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.

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.

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 *‘codeprocessor.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 the deadline specified on Moodle and at the top of this file.
Your entire team must accept the assignment submission statement individually on Moodle. If you are unable to accept the statement online you may complete an offline form as a team and send it via email to eng1003.clayton-x@monash.edu with the subject “Assignment One Submission Statement”. You just need to type your names on the signature line digitally. Form available here: https://goo.gl/ZSDmRY.
No replies will be sent in response to these emails.

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

You also need to individually complete the following tasks (described below):

- CATME peer assessment survey
- Assignment code interview

Your presentation will be given during your practical classes in week 6.

Marking criteria

Programming tasks

Your assignment will be assessed based on the version of '`codeprocessor.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%
Good 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. 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.

Plagiarism and collusion

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 policy linked below for more information.

https://www.monash.edu/students/academic/policies/academic-integrity

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

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 unit staff.