COSC 345 Assignment 1

Nathan Kennedy 1006384, Kat Lilly 8867318, Tobyn Packer 5003352, and Daniel Davidson 375621

April 16, 2018

What we are going to build

We intend to build an app containing a suite of games that can be played without the use of one's hands. Entry to the app as well as control of the games will be by voice command.

We have begun with a tetris game as a way to get started, to familiarize ourselves with speech recognition software, and to learn how to write an Android app. Tetris is a game that has very simple keyboard inputs, and for which there are many online tutorials for writing the game, so there shouldn't be significant challenges beyond capturing the voice commands.

After this we intend to add at least one additional game. We are exploring the possibility of games that may be more interesting to control by voice than tetris is, where the game-play is improved or changed by being voice controlled. For example, there is a Mario-style game called "Scream Go" `https://www.youtube.com/watch?v=IyvjpgS9TAs`, where a character navigates a world with walking and jumping movements, which are controlled by pitch and loudness of the player's voice. Another example is "Full Voice Throttle" `https://kushulain.itch.io/full-voice-throttle`, a motorbike racing game where acceleration is achieved by increasing the pitch of the player's voice, and the gear is changed by starting again at a lower pitch.

The working title of our app is 'nktd', the most pleasing permutation of our initials. `http://github.com/katlilly/nktd`

Intended users

We spoke to the manager of the disability support unit here at Otago, and talked with her about what the real needs of students and staff here are. Her main point was that mental health issues are the biggest issue, and so we briefly explored the possibility of a project aimed at supporting Otago students in this area. However, none of us were particularly keen to take on such a weighty issue as this, and we also felt that a project like this would involve a lot of work writing content and doing research about non CS topics, which is not how we would prefer to spend most of our effort.

Instead we have chosen to write games that can be controlled without the use of hands. None of us had written games before, and we were all keen for that particular challenge. In addition, we believe this will be a project where there is a very achievable minimum product that we can get started with, as well as lots of opportunity to expand on it within the timescale of one academic year. We feel that this will be a good project for learning a lot of the skills we want to get out of this software engineering course.

Our target users are anyone who does not have full use of their hands, either permanently or temporarily. The app will be launched with the voice control in the operating system, and the user will be able to select a game and play it using only voice commands.

Our intended users are defined not by a particular illness, injury or other source of disability, but are unable to use their hands, and we are focusing on this particular interaction with a device. Our users may include amputees, quadriplegics, or people with a hand tremor, a broken arm, arthritis, and so on.

Of course our users will not be restricted to those who are unable to use their hands, and we hope that our games will be fun to play for anyone without an impairment.

A particular user might be a child who has broken both their wrists and is in pain. We know from personal experience that playing games is not only a solution to boredom but can be really good for coping with pain.

We assume that our users have already solved the problem of holding and unlocking their device, and are using the operating system's voice recognition features. We will use the system-provided Voice Actions to allow users to open our app by a voice command such as "OK Google, play tetris".

No specialised hardware will be necessary, just the built in microphone that all phones and tablets have.

We are beginning with voice control as a substitute for key presses and touch screen control, but also want to explore the possibility of using eye tracking for some aspects of our app. It is likely that this will be more than we can get working well within the timeframe of this project, but we will make the call after further research, and not at the expense of getting voice control working well.

Platforms

We will build an Android app that runs on both smartphones and tablets. If one or more of the games we write is suitable for a very small screen we will also consider including a smartwatch version.

We intend to have an optional login function in our app, that would allow the user's saved settings and high scores to be transferred between devices.

How we will build it

The game logic and the Android app will be written in Java. We are using open source voice recognition software CMUSphinx from `https://cmusphinx.github.io`. More specifically, we are using sphinx4.

The speech recognition system requires three parts:

- An acoustic model consisting of acoustic properties for each 'phone' or unit of sound.
- A dictionary mapping combinations of phones to words.
- A grammar model that tells the system valid combinations of words it will hear.

The dictionary and the acoustic model can be shared among all games we make, but we will need to write a new grammar model for each game.

We will be using one of CMUSphinx's pre-built acoustic models downloaded from `https://sourceforge.net/projects/cmusphinx`. Our dictionaries will be generated for us by CMUSphinx's LMTool here: `www.speech.cs.cmu.edu/tools/lmtool-new.html`. The grammar models we need to write ourselves in JSFG - Java Speech Grammar Format

The speech recognition will only work accurately on audio of the same sample rate/number of channels as the audio its acoustic model was trained on. The CMUSphinx model was trained on 16Khz, 16bit Mono audio. This means that potentially we will need to down-sample the audio being received from the user's microphone before it is interpreted. We can do this using the Java Sound API.

We will also need to learn how to incorporate pitch and volume information into our game control, this won't be done with CMUSphinx, and we have not yet done any real research into this aspect of the project.

Team member roles

Nathan Android app development, designing user interface and graphics.

Kat Writing reports and meeting minutes, writing game logic, some testing.

Tobyn Graphics, ideas for new games and coding them. Writing user documentation

Daniel Lead test developer, working with open source voice recognition software, speech recognition and audio inputs.

Managing the workflow

We will manage our project using the Scrum framework, currently working with two week sprints. We have a regular team meeting on Mondays (more often when needed), where we catch each other up on what we have achieved over the previous week and what needs doing next. So far these meetings have mostly been about evolving our idea of what we are actually trying to make. We are using Google Drive to host a spreadsheet with our Scrum workflow.

We have a Slack workspace for our project that is linked to the github repository, which we use to communicate about the project between meetings.

We are using the github wiki to record what was discussed at meetings and for notes about how aspects of the project work.

We intend to use a continuous integration tool to keep our code in a non-broken state as much as possible, but don't yet have this functioning.

Project Timeline

Our alpha release (Assignment 2) will be an app in which the user can play tetris purely with voice commands, and will work on both Android phones and Android tablets.

Improvements in our beta release (Assignment 3) will include the addition of a second game, and the ability to log in to the app and keep your preferences and high scores between devices.

Our finished product (Assignment 4) will be a thoroughly tested, less buggy version than Assignment 3, with faster and more accurate interpretation of voice commands as well as the possible addition of a third game and/or a smartwatch version.

See Figure 1 for our Semester One timeline, and Figure 2 for a broad brush timeline for Semester Two.

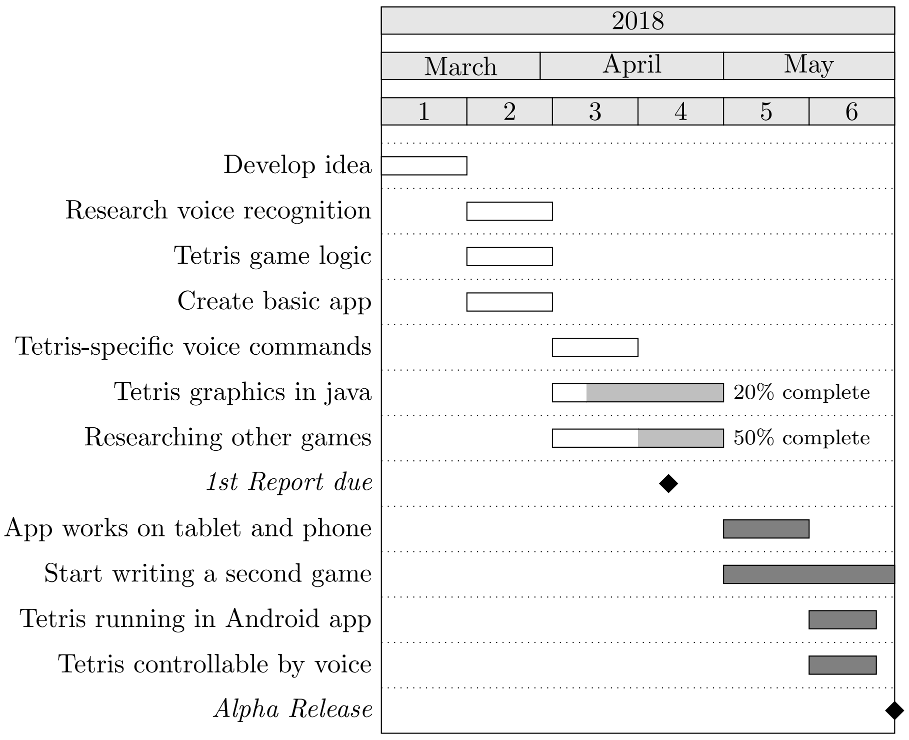

Figure 1: Semester One timeline. We have divided our first semester efforts into 6 two-week sprints, beginning March 6th and ending with Assignment Two deadline on Monday 28th May.

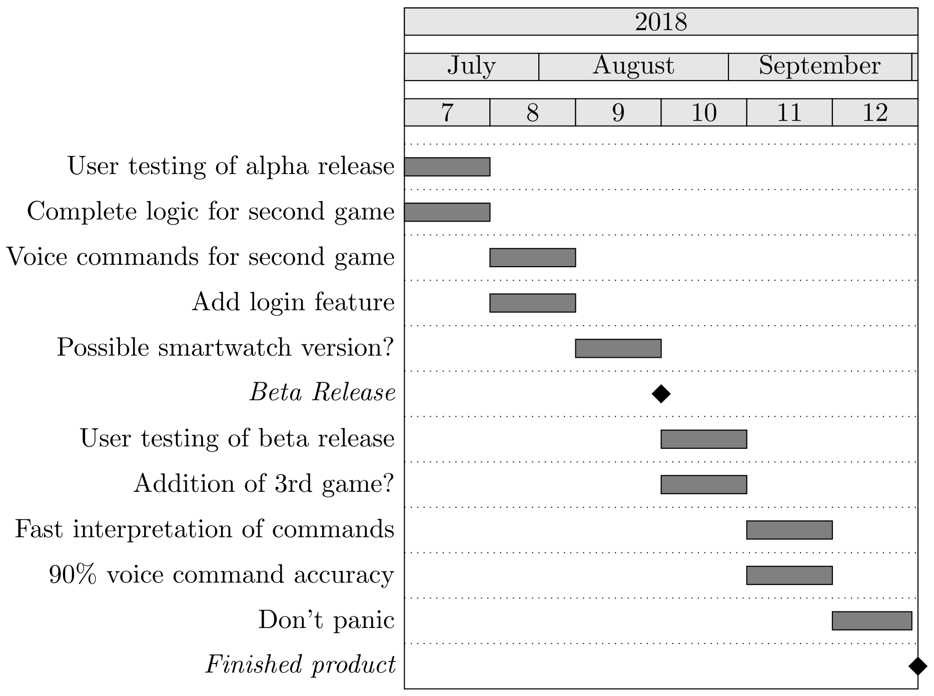

Figure 2: Rough outline of our Semester Two timeline, composed of 6 two-week sprints beginning Monday 9th July, with an estimated due date for the final product of Monday 1st October.