CAB202 Assignment 2: Falling Faces

Assignment 2 will be marked during your assigned tutorial session in Week 13 (19 to 23 October) using files submitted to AMS
Marks: 40 (40% of your final mark)

For this assignment, you will be writing your own version of a game called Falling Faces.

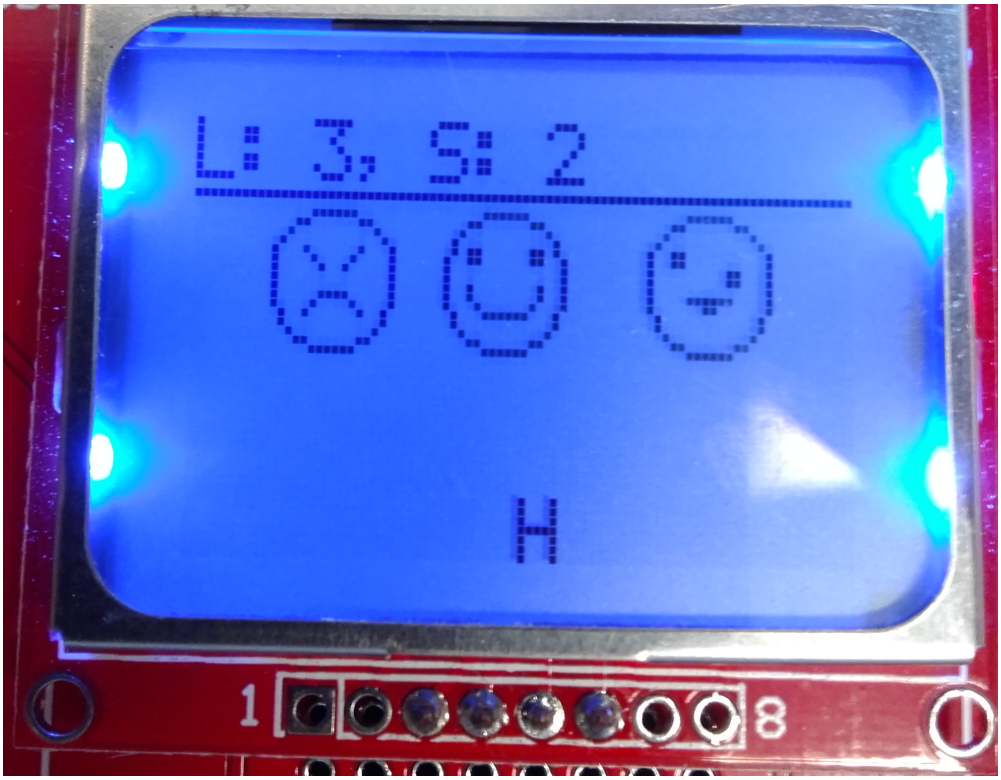


Figure 1. Game initial concept

Requirements

Your task is to implement a game in which several faces will appear on the LCD display and fall from the top of the screen. The player controls a sprite at the bottom of the display, moving it left and right to with the goal being to maximise the score before losing all lives. Points are gained, or lives lost, when the player's sprite collides with the faces, as described in detail below. You will use the skills and techniques learned in the first half of the semester, together with microcontroller programming techniques covered since week 7. A partial implementation of the game is demonstrated in the week 10 lecture.

WARNING: QUT takes plagiarism very seriously. If your assignment has material which has been copied from other class members, previous assignments, or code from any other source you will be penalised severely. If you cannot explain exactly what your code is doing, it is clearly not your own work, and you should not be submitting it as such!

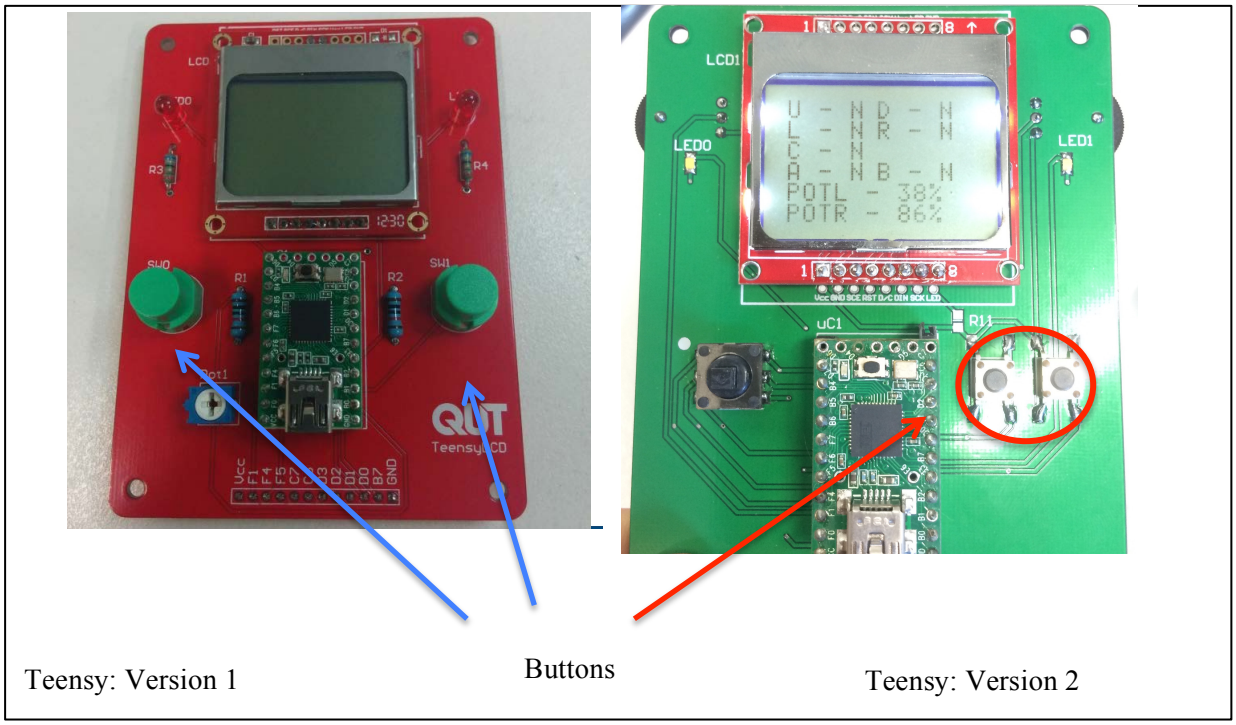
Game Overview

The basic aim of the game is for the player to use buttons to control the teensy and move a small sprite around the screen until the player achieves 20 points. The sprite may be a simple character, such as the letter 'H', or something more imaginative. Faces appear at the top of the screen and move vertically downwards, however they can appear initially at random horizontal positions. The user has no control over where the faces appear initially. The player starts with 3 lives. There are three (3) possible faces: Smiley, Mad, and Angry. When the player's avatar hits Smiley two (2) points are gained; when the player hits Mad the falling speed of face motion is increased; and finally when the player hits Angry, one (1) life is lost. The game ends when the player has no remaining lives, or when a score of 20 is achieved. A higher level of difficulty is possible where movement of the hero are controlled by an external command or character sent from a laptop or PC.

Level 1: Basic functionality [20 marks total]

All versions require the following basic functionality (you will not be eligible for any more than 20 marks unless everything in this section is completed):

- The game must show your name and student number when teensy is powered on.
- Following, a brief menu should appear giving the user the option to navigate through levels using the right button, i.e Level 1, Level 2, or level 3
- Next, the game should start after pressing the left button.
- A status display must appear at the top of the screen, showing the current score and number of remaining lives.
- The area below the status display is the legal play area. All sprites and characters that do not form part of the status display must be confined to this area.
- Three faces (one of each) should appear at the top of the legal play area at random horizontal (x) positions, subject to additional constraints appearing below.
- Faces should move down the screen at a constant speed. The initial speed of each face is *Slow* – see below.
- Score and lives must be shown at the top. See example in Figure 1.



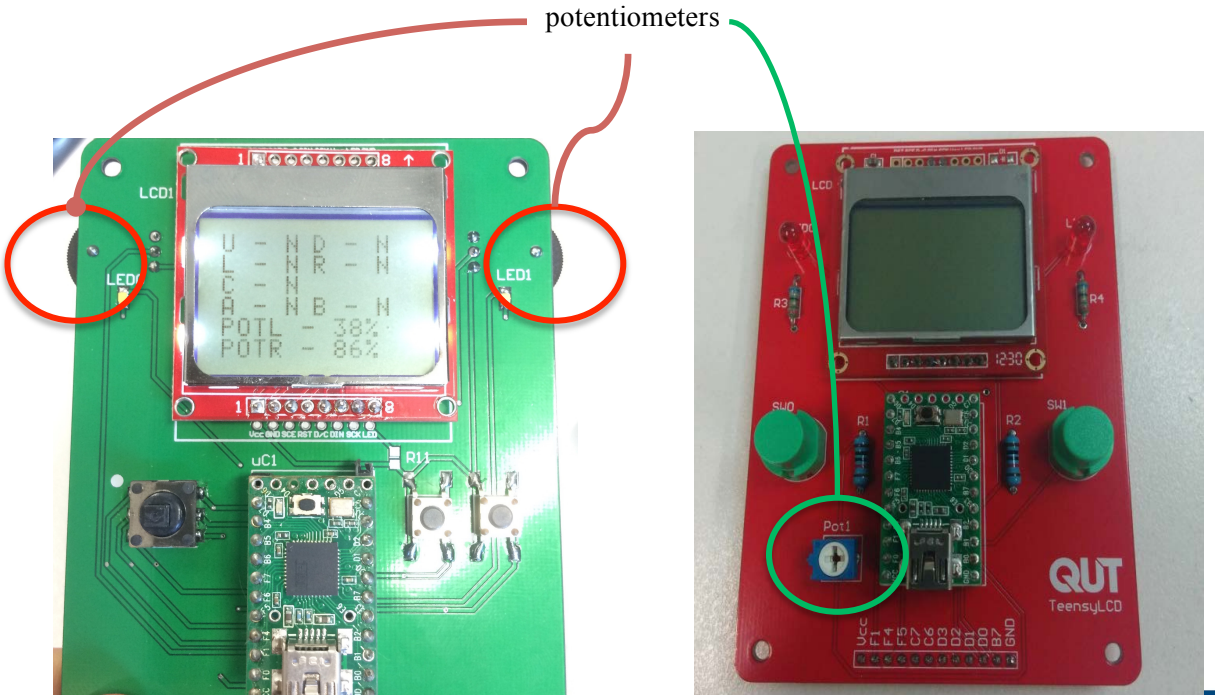
Teensy: Version 1 Buttons Teensy: Version 2

- Faces should appear whole at the top of the screen, below the status display line.
- Faces must fit inside the play area at all times. Faces must not be cut by any of edges of the screen and they must not overlap the display area at the top of the screen.
- No more than three (3) faces should be in the screen at any time.
- Faces must not overlap. There must be a horizontal separation of at least 5 pixels between faces. You can treat faces as square boxes when performing collision detection.
- The Hero should stay at the same vertical position at the bottom of the screen, but move horizontally.
- The Hero's horizontal motion should be controlled by the buttons. Left button will move the Hero towards the left of the screen, right button will move the Hero towards the right of the screen.
- The player has 3 lives at the start of play.
- When player hits Smiley two (2) points are gained, when Mad is hit the speed of motion of all faces increases, and when Angry is hit a life is lost and speed remains the same.
- Face speed has three (3) qualitative levels, *Slow*, *Medium* and *Fast*. There is no rigid specification for these speeds in terms of pixels per second, however they must obey the following rules:
    - *Slow* is a constant speed which is greater than 0 pixels per second.
    - *Medium* is a constant speed which is easily and indisputably seen to be greater than *Slow*.
    - *Fast* is a constant speed which is easily and indisputably greater than *Medium*.
    - *Slow* is fast enough to allow the game to be played in a reasonable time frame.
    - *Fast* is moderate enough to give the player a chance to win.
- The game ends when player has no more lives or 20 points are reached.

Level 2: Motion using potentiometers [7 marks total]

In this mode the motion of the hero is now controlled by the potentiometers (see figures below). Teensy version 1 and 2 have both potentiometer that can be used to move the Hero on the screen:

- Teensy version 2: You can use the right or left wheel (must choose only one) to control horizontal motion.
- Teensy version 1: You can use the potentiometer to control horizontal motion. Clockwise direction should move the Hero to the right, anticlockwise should move the Hero to the left.



Teensy: Version 2

Teensy: Version 1

Level 3: Motion using serial port [13 marks total]

In this mode, a more complicated version of the game is played with the hero controlled by the keyboard on a computer connected via serial port. An external program (like hyperterminal or putty) is used to read keyboard input and send it to the Teensy. The level should do the following:

- A program (hyperterminal, putty, or similar) running on a computer is used to send key presses to the teensy using the USB port.
- The Hero is now moved left, right, up, and down by pressing the 'a', 'd', 'w', and 's' keys on the computer, respectively. The movement distance per key press is unspecified, but must be a value that makes your game easily playable.
- The faces spawn at a random position within the legal play area. No part of the face can be outside the legal play area, and the spawn must be at least 5 pixels away from the hero and any other face. If there is no legal spawn available, no spawn should happen until one is possible.
- On spawning, each face is given a random initial direction to travel in.
- Faces must bounce off the boundaries of the legal play area. The bouncing behaviour should match the figure below:

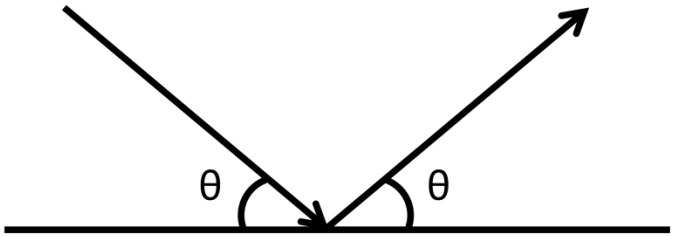


Figure 2 Angle of incidence = Angle of reflection

- When faces collide with each other, they should **both** perform the same bouncing behaviour as above (i.e. **both** bounce off at the reflected angle).
- Motion speed of the faces follows same behaviour described in level1.

Marking

The assignment will be out of 40 marks and will be worth 40% of your total grade in this subject. The Criterion Referenced Assessment document (released separately) describes the apportionment of marks. The following should be noted about marking:

- **If your code does not compile, you will get 0 marks for the entire assignment.**
- If the game crashes or locks up during testing, you will receive marks for what has been tested up that point. No more of your assignment will be marked. We will not debug your code to make it compile or run.
- Your game must be easily playable. If timings, settings, or controls are set in a manner that makes it difficult to play (e.g. not using the key inputs specified, ridiculously fast movement, etc.), you **will receive 0 for the assignment**.
- The assignment will be marked during your scheduled tutorial in week 13. To receive a mark for the assignment, you are required to:
    - attend the tutorial in person,
    - demonstrate your program to a tutor,
    - explain details of your implementation, and
    - hand in your Teensy .
    - If you fail to do any of these things you will receive a mark of 0 for the assignment.

Submission

Your assignment is due on Sunday October 18th, 2015 at 11:59pm. Submission will be online through the AMS used in the tutorials. You must submit through the submission page (available at http://bio.mquter.qut.edu.au/CAB202/Exercise?TopicNumber=6&ProblemNumber=2), and follow all of the instructions.

When submitting to the AMS, it is your responsibility to make sure that your assignment compiled correctly. The AMS will compile your code and return any errors that occur. As mentioned above, if you do not submit a compiled assignment, you will receive 0 marks. You will have unlimited submissions of the assignment, so there are no excuses for not resolving any compilation issues.