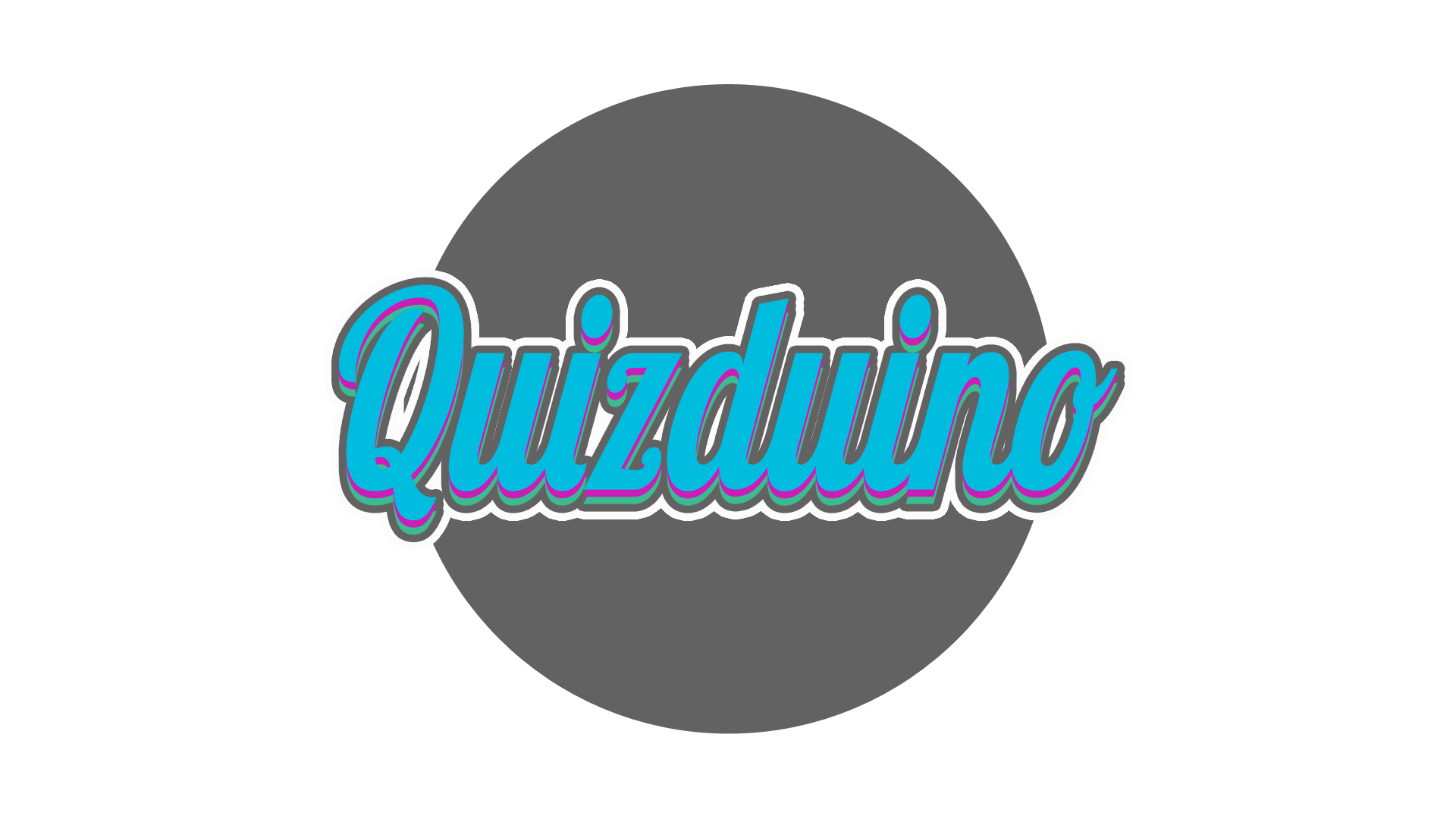
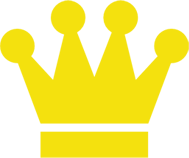
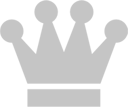
The design process involved researching similar games and applications and testing various colour schemes. The final design that was settled upon by the group features a simplistic and stylish design, utilising the use of neutral yet bright colours, which leap from the screen. This design was intended to grab the users attention and entice them to play the game.



For the logo we went for a simple design with a circular grey background with the word Quizduino in the centre. The text colour is aqua blue with grey and light pink edges to make the outline stand out more.



This is the main background where every interaction will be performed. The colour scheme was specially chosen to be neutral so as not to interfere visually when playing the game.



These symbols are used to represent the leaderboards. The symbols will be assigned to the top 3 players and the rest of the players will be displayed below without a symbol in a list.

The question ‘state’ shall be displayed visually by the use of colour. The images below and their description explain how this is achieved.



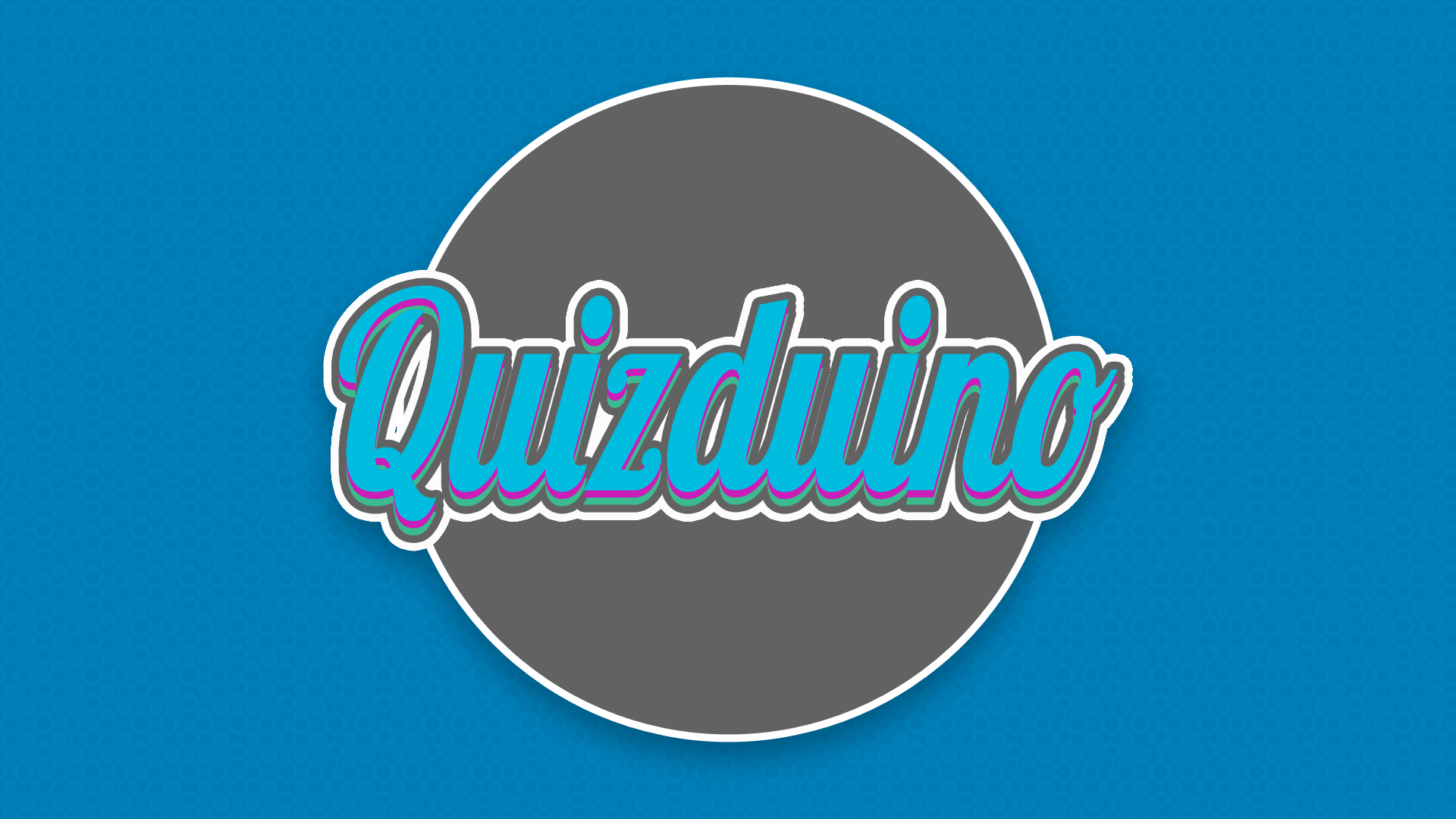
When question is initially displayed i.e. not yet answered, the colour shall remain blue.



When the question has been answered correctly, the colour shall change to green.



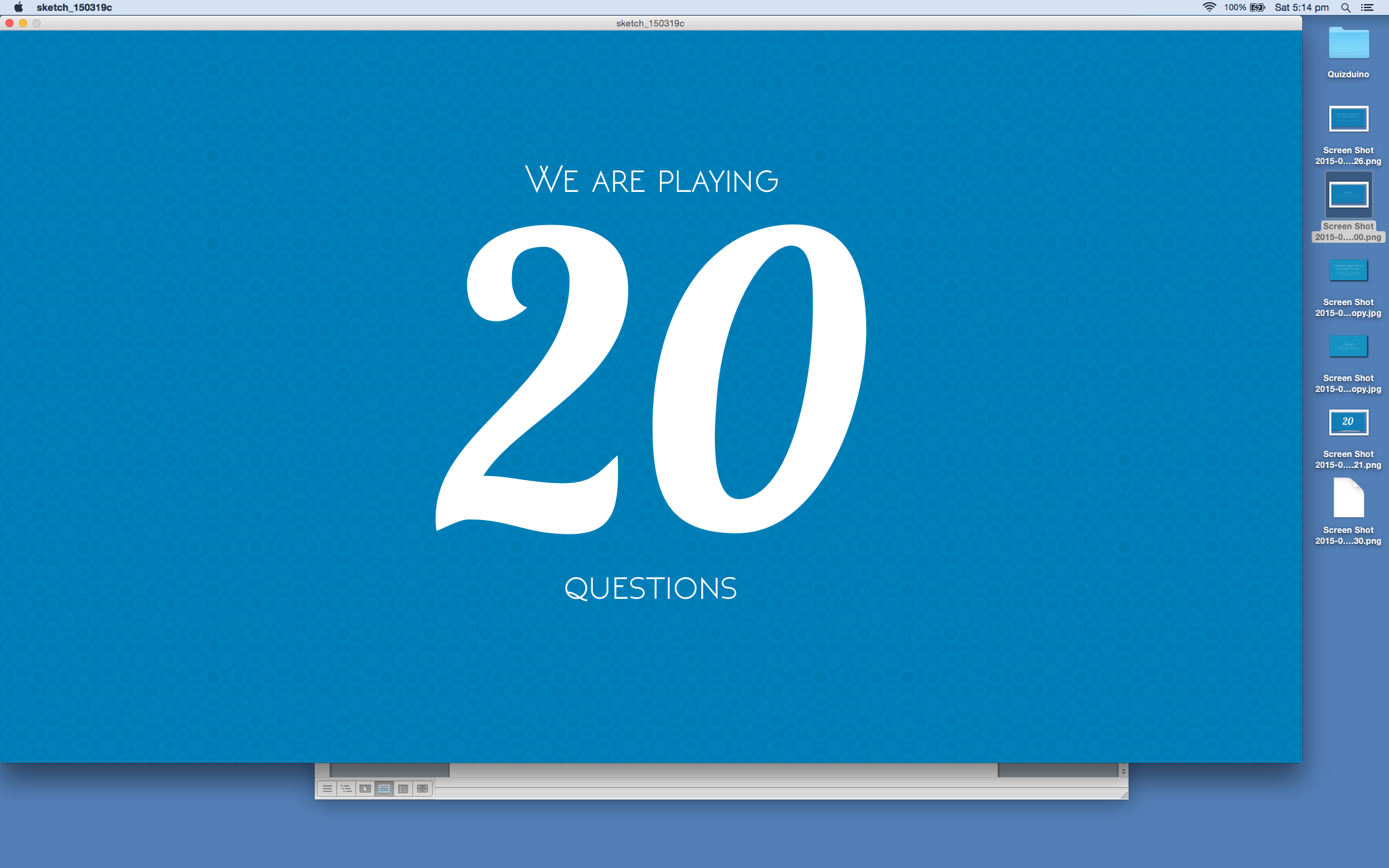
An incorrect answer to the question shall result in the colour changing to red.



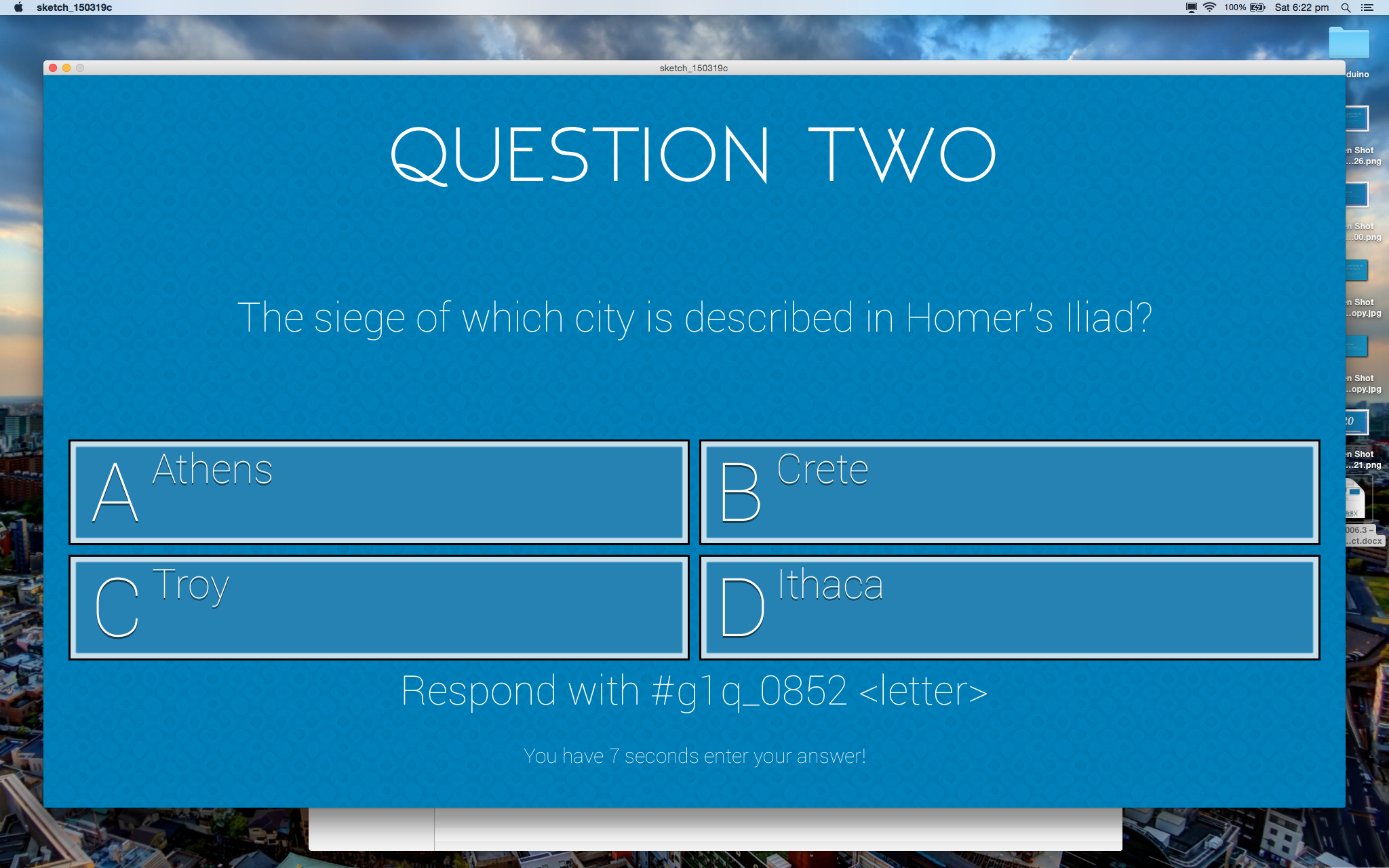
After the logo screen disappears, you need to tweet how many questions you want to play by using the hashtag ‘#g1q\_’ followed by a unique identifier like ‘2944’ and then ‘ 5’ which indicates the number of questions you’d like to play e.g. #g1q\_2944 5. The first four numbers are generated randomly for each tweet to be received. If more than one players are playing, it will collect all the tweets’ question numbers and display an average.



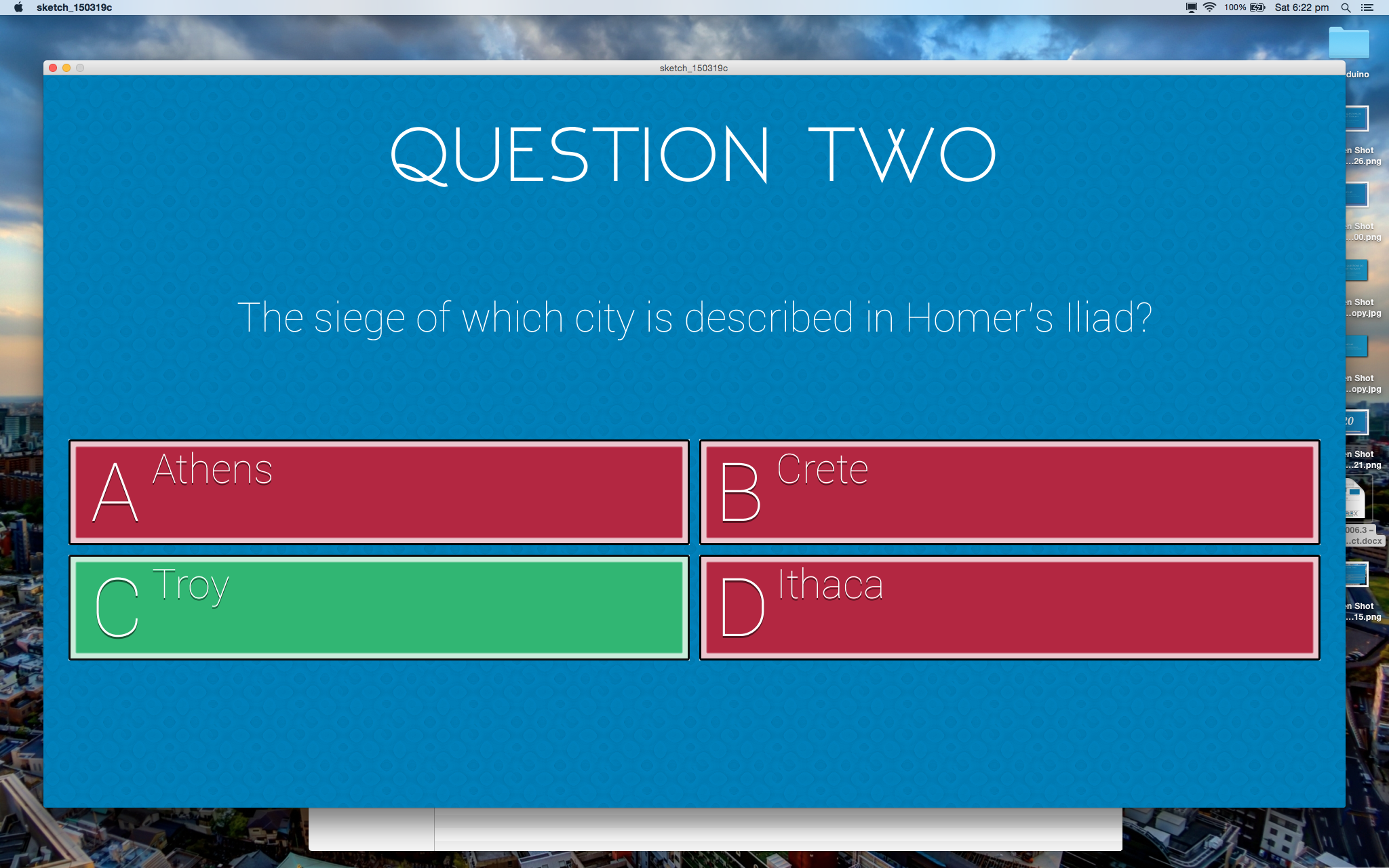
When the players have finished tweeting their question numbers a second screen will appear telling them that the time is up and that the tweets from now on will no longer count.



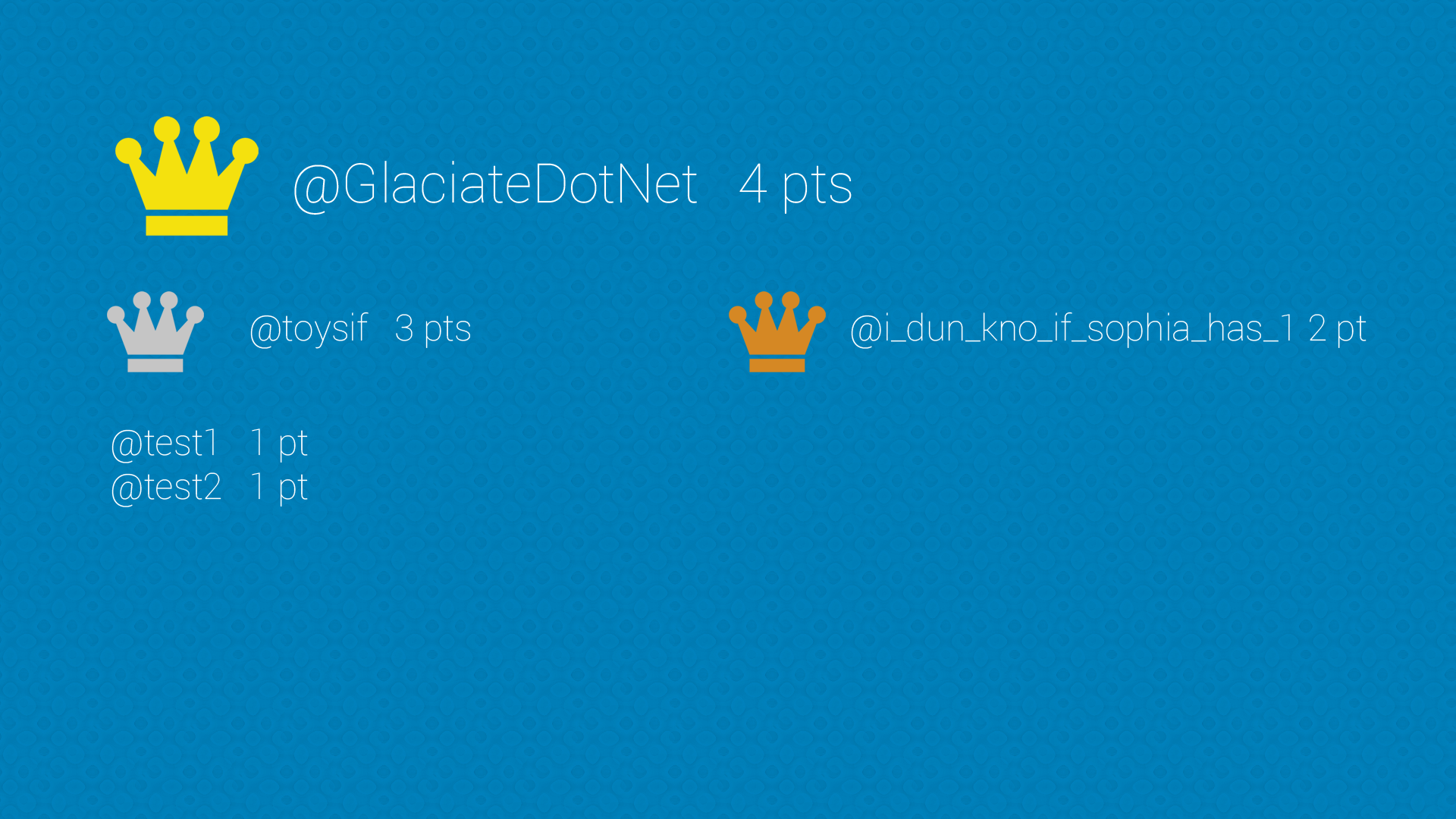
After all the players have submitted their tweets, this screen should appear showing how many questions will be played.



A question appears: the user must include another unique code in their tweet followed by their answer. They have about 30 seconds to tweet an answer.  
(The position of the correct answer is randomised for each question)



Answers appear: after the tweet has been collected the correct and incorrect answers are indicated with green and red colours respectively



The top 3 players with the most correct answers will appear on the leader board with the correct crown colour beside their name, while the other players will appear in a list.

Evaluation

The general program structure consists of loading the questions and answers for the quiz, the creation of the user interface, random display of the questions/answers, collection of user input/tweets (question number and question answers), score tracking.

While coding this project, “moving” through screens, states and methods proved to be quite challenging. Because of numerous methods being involved in the program, it was a bit messy getting everything to work in order. An example of this could be collecting the desired question number by the user.

In order to tackle this -and generally the more complex problems, various information printing was involved in the testing part. Another approach to this were multiple and various input tests.

What proved to be a bit more difficult than expected was “reading” the users’ tweets in general. In connection to the previous challenge, this was hard to get to work properly. The methods wouldn’t process in sequence and we wouldn’t get the expected results by reading a tweet (e.g. the answers to the questions wouldn’t be read properly). Also through testing, we realised that a tweet sent less than 20 seconds ago wouldn’t be read, which at first caused more problems than expected.

What seemed to be easier than expected was creating the user interface. We guess that the reason for this was because processing is a very object oriented work space. Pictures, music and fonts were easy to load into the program, shapes were easy to create and colours easy to change.