CS50 Final Project Design Document

Project title: Chimpanzee Counting
Student name: Anna Currey
Student ID: 80936273
TF name: Alex Liu
Semester: Spring 2014

Overview

Chimpanzee Counting is a web-based game that is implemented using HTML, CSS, and JavaScript.

HTML and CSS

The HTML and CSS in this game is mostly straightforward. There is only one HTML file, index.html, and one CSS file, styles.css.

index.html

For the most part, this file should not need much explanation. The most important thing to note here is that I decided to have all of the elements on the page from the beginning, including the **Play** button, messages, and the cards, even though they are not always shown. I then used JavaScript to show or hide these as appropriate (and to modify their HTML). Note also that I only used one message and one button, and later changed their text and colors using JavaScript. I did this because the messages and buttons have consistent styling and behavior, and there are never multiple buttons or messages showing up at once.

styles.css

This document is for the most part straightforward; however, I want to note that I chose not to make the game area responsive because I did not think it would work well on a smaller screen size. Below I will briefly describe the IDs and classes.

- *#top*: the part of the page with the game title and instructions.
- *#header*: the game title.
- *#instruct-text*: the instructions.
- *#timer*: the space where the timer (or the "GO!" text) appears.
- *#middle*: the game board.
- *.button*: buttons on the game; currently this consists of only the **Play** button, but I kept it as a class in case I wanted to add more buttons with the same style when more enhancements are added to the game.
- *#message*: the message telling the user if they won, lost, or passed the round.
- *#footer*: the bottom of the page, with my name and the page's purpose.

- *.card*: the cards on the game board; contains the general card styling.
- *#card0* - *#card5*: each individual card on the board; contains the card's position.

JavaScript

The game consists of two JavaScript files: cards.js and game.js. The game.js file modifies the HTML, while the cards.js file keeps track of the cards used in the game. The cards are hidden and the **Play** button is shown. Once the button is clicked, the game starts. This consists of a memorization period where the cards are shown; then they are flipped over so that their values are hidden. The game advances based on the cards that the user clicks.

cards.js

The cards.js file contains an array of objects, with each object representing the one of the cards on the board. For each card, this file keeps track of:

- The card number (*card_num*):
  - This is a convenient way of keeping track of the cards, as each one is given a unique number (0-5).
  - Currently, this is not actually used anywhere in the code; however, I added it because I felt it might be useful for future enhancements.
- The value on the card (*card_val*):
  - This stores the value that is randomly assigned to the card (which dictates the order in which it should be revealed).
  - In the cards.js file, all cards are given a *card_val* of false, indicating that they do not yet have any value. The value is assigned once the game starts.
- The card's ID (*card_id*):
  - This is the ID given to that particular card in the HTML.
  - This is used to modify the HTML for that card in order to show and hide the value on the card (giving the effect of unflipped and flipped cards).
  - The card ID is also used to indicate which card was clicked (to make sure that card is the one that reacts).
- Whether or not the card is flipped (*flipped*):
  - If *flipped* is false, then the value is not shown on the card.
  - This is used to make sure that cards respond correctly when they are clicked (based on whether or not a value is already shown). If a value is already shown on the card, then clicking the card does nothing.

game.js

The game.js file is the file in which the game is actually implemented. In this section, I will walk through each of the components and functions in this file.

Constants

At the top of the file are constants that are used in the game. They are for the most part self-explanatory; below, I will explain some that might be less clear.

- *INTERVAL*: the number of milliseconds to count down by (1000 milliseconds = 1 second, so the counter will show “3-2-1”).
- *MAX_SECONDS*: the number of seconds for the first round, plus one (one is added so that the last round is not zero seconds; the rounds are counted from 1 instead of 0 to make more user-friendly labels).
- *CURRENT*: the value on the most recent flipped card; this is initially set to -100 because the first flipped card should always be correct for a round. When a card is flipped, its value is checked against *CURRENT*; if the new value is less than *CURRENT*, then the player loses.

Main function
Once the window loads, these steps are executed to start the game. The cards are hidden and the **Play** button is shown, and once the **Play** button is clicked a series of functions are called. These functions are each described below.

assign_val
The *assign_val* function assigns a random value between -99 and 99 to each of the cards in the game. It is called at the beginning of each round so that each round has different cards. This function only stores the values in the cards; it does not display the values. It is implemented by looping through the cards and choosing a random number for each by using the *Math.random()* function.

correct
The *correct* function reacts to a correctly clicked card. There are three cases in this function.

- If the card is not the last card in the round, the function does nothing. It tests whether it is the last card in the round using the *FLIPPED* variable, which keeps track of how many cards have been flipped in the current round.
- If the card is the last card in the round, but the round is not the last round in the game, the function shows a message that the round has been completed as well as a button to move on to the next round while hiding the cards on the board and the timer. It accomplishes this using the jQuery functions *html*, *show*, and *hide*. The *css* function is also used to ensure that the message is shown in an encouraging color. Finally, the round number is incremented (this is what is used to tell if we are on the last round) and the current value and the number of cards flipped are reset.
- If the card is the last card in the last round, the behavior is similar to that of the previous bullet point, although the message and the button text are different and the round number is reset to 1.

hide

This function hides the value on each of the cards by modifying the card's HTML so that it is blank (""). It also updates the value of *flipped* for that card to false (since no value is shown). This function is called once the memorization time ends.

lose
The *lose* function consists of steps taken when a user loses. Note that this function is called *lose* because any wrong answer automatically entails losing the game; the *correct* function is not called *win* because a correct answer does not necessarily imply winning. When the user loses, we hide the cards and the timer and show a message and a **Play again** button. This is accomplished using the jQuery *hide*, *html*, and *show* functions on existing HTML elements. The jQuery *css* function is also used to update the message color (so that the message is not shown in a happy color). Finally, the values for global variables (discussed above) are reset so that the game can be played again.

play
The *play* function handles the reaction to the user clicking a card. It uses the jQuery event handler *click* and passes the ID of the clicked card into the *show* function (described below).

populate
This function populates the board with the cards, which are hidden before the game starts. It is called as soon as the **Play** button is clicked. The function loops through all of the cards and updates their HTML so that their values are shown. It also marks all of the cards as flipped, and shows all of the cards using the *show* jQuery function (calling the cards by class "card").

show
This function takes as input the ID of a card and then shows the value on that card. To find out which card was clicked, it has to loop through all of the cards and see if the ID matches. If the card's value is already shown, it does nothing. If the card's value is not shown, it shows the value using the jQuery html function, updates its *flipped* value to true, and updates the number of flipped cards. Finally, it checks if the card is correct by comparing the value on the card to the value on the previous card. If the value on this card is greater than or equal to the value on the previous card, it is correct and the *correct* function is called; if not, then it is incorrect and the user loses the game, so the *lose* function is called.

timer
This function shows the countdown timer during the memorization period. This is the function that is passed into the *setInterval* function at the beginning of the game, and it runs for the correct amount of seconds for the given round. It counts down until the number of seconds reaches zero and shows the number of seconds as it is counting down. Once the number of seconds is zero, the interval is cleared and the timer is replaced with the word "GO!" in green.