DS4TALKER

LAB #9

SECTION#3

SUBMITTED BY:

AIDAN FOSS

SUBMISSION DATE:

12/8/2023

Problem:

This lab was designed to teach us to make guis and how to create and navigate them.

Analysis:

I had to print a GUI, and create a way to navigate it without creating or deleting unnecessary characters. When the “cursor” / avatar is over a certain word, different inputs should do different things. Pressing one button should print the word, another should delete the amount of characters in that word. Using the joystick should navigate the list of words.

Design:

I used the basic guidelines to construct most of my code. I started by creating my GUI, and adding the character movement. Once I got that down, I created the print and deletion statements. Using a wordslist function/array I was able to get a list of words and their lengths which I used for deleting and pasting, depending on what button was pressed over the GUI.

Testing:

In my testing, I struggled to get the code to compile with what I felt was correct syntax. For example, for the piece of code that pastes a space between words, it would not compile at all unless the space was 11 characters long. It said something about null characters being printed, but all that was being printed was one single space character. After making the space 11 characters long, it compiled and ran fine, but the spaces that get printed are 11x larger than they should be.

I also wrote what I thought would be a working implementation of the extra credit condition, and it just never worked. It didn’t do anything, so I just deleted it. I was disappointed.

Question 1:

I had to use fopen() to open the file, and then I used a buffer and malloc() to create a list of words which I copied onto a passed array to use elsewhere. For each word, I had to read it into a buffer, then memory allocate space for it, before strcopying it into the full wordslist.

Question 2:

I keep track of the word selected on the screen by using an indicator. I used '@' near the word selected to make it obvious which word is selected.

Question 3:

To delete words, I created an array that is 80 long, with the first entry being 0. Each time a new word is added, the length of that gets added to the array, and when square is pressed it deletes that many characters. Using the array, I can go back and delete the correct number of characters each time undo is called.

Question 4:

I think the joystick is easier to code for, because testing it is easier. It makes more sense to me the way the joystick works, and it tends to be less erratic and weird, so its easier to account for error. All controllers have a little bit of joystick drift, but its easy enough to code out of to fix any issues.

```c
// Lab 9 DS4Talker Skeleton Code

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <ncurses.h>
#define MAXWORDS 100
#define WORDLEN 11
#define SENTENCE_LEN 81
#define DEBUG 0   // set to 0 to disable debug output

// reads words from the file
// into wl and trims the whitespace off of the end of each word
// DO NOT MODIFY THIS Prototype
int readWords(char* wl[MAXWORDS], char* filename);

//modifies s to trim white space off the right side
// DO NOT MODIFY THIS Prototype
void trimws(char* s);

int drawWords(char* wl[MAXWORDS], int numWords);

int addSent(char sentence[SENTENCE_LEN], char add[WORDLEN]);

int main(int argc, char* argv[])
{
    char* wordlist[MAXWORDS];
    int wordCount;
    int i;

    wordCount = readWords(wordlist, argv[1]);

    if(DEBUG)
    {
        printf("Read %d words from %s \n",wordCount, argv[1]);
        // add code to print the words to the screen here for part 1
        for(i = 0; i < wordCount; i++)
        {
            printf("%s\n", wordlist[i]);
        }
    }

// most of your code for part 2 goes here. Don't forget to include the ncurses library

    int t, tri, circle, x, square;
    int rb, lb, bb;
    int lx, ly, rx, ry;

    initscr();
    refresh(); //clear screen

    char sent[81] = "";
    int capNext = 0;
    int curAt = 0;
    int undoAt = 0;
    int changed = 0;
    int lastMov = 0;
    mvaddch(0, 0, '@');
    int row = drawWords(wordlist, wordCount);
    int undo[80] = {0};
```

```
do
{
    scanf("%d, %d,%d,%d,%d, %d,%d,%d,%d,%d,%d,%d,%d,%d, %d, %d, %d, %d", &t, &tri, &circle, &x, &square, &rb, &lb, &bb, &bb, &bb, &bb, &bb, &bb, &lx, &ly, &rx, &ry): //has to be a huge scan for all the info taken in
    int oldAt = curAt;
    if(changed == 0){
        changed = 1;
        if(tri == 1){
            int added = addSent(sent, "          "); // i am not sure why but it would give errors if this wasn't 11 spaces???? works fine though
            if(added > 0){
                undo[undoAt] = added;
                undoAt++;
        }}
        else if(x == 1){
            if(undoAt > 0){
                sent[strlen(sent) - undo[undoAt - 1]] = '\0';
                undoAt--;
        }}
        else if(square == 1){
            if(capNext == 1){
                wordlist[curAt][0] = toupper(wordlist[curAt][0]);
                int added = addSent(sent, wordlist[curAt]);
                if(added > 0){
                    undo[undoAt] = added;
                    undoAt++;
                }
                wordlist[curAt][0] = tolower(wordlist[curAt][0]);
                capNext = 0;
            }
            else{
                int added = addSent(sent, wordlist[curAt]);
                if(added > 0){
                    undo[undoAt] = added;
                    undoAt++;
                }
                undoAt++;
        }}
        else if(l == lb){
            sent[0] = '\0';
        }
        else{
            changed = 0;
    }}
    else if(0 == tri && 0 == square && 0 == x && 0 == lb){
        changed = 0;
    }
    if(t - lastMov > 500){
        lastMov = t;
        if(lx > 100){
            if(curAt < wordCount - 1){
                curAt++;
                changed = 1; //set changed to  prevent double placement
        }}
        else if(lx < -100){
            if(curAt > 0){
                curAt--;
                changed = 1;//set changed to  prevent double placement
        }}
        else if(ly > 100){
            if(curAt < wordCount - 6){
                curAt += 5;
                changed = 1;//set changed to  prevent double placement
        }}
        else if(ly < -100){
            if(curAt > 4){
                curAt -= 5;
                changed = 1;//set changed to  prevent double placement
        }}
        else{
            t = 0;
    }}
```

```
        else if(lx > -100 && lx < 100 && ly > -100 && ly < 100){
            t = 0;
        }
        if(changed == 1){
            mvprintw(row, 0, "%s", "
            mvprintw(row, 0, "%s", sent);

            if(oldAt != curAt){
                mvaddch(oldAt / 5, (oldAt % 5) * 15, ' ');
                mvaddch(curAt / 5, (curAt % 5) * 15, '^');
            }
            refresh();
    }}while(1); //keep looping while program is running
    return 0;
}




/*
 Scans a word and buffers it
 */
int readWords(char* words[MAXWORDS], char* file)
{
    FILE* f = fopen(file, "r");
    int loc = 0;
    char buffer[WORDLEN];
    while(1 == fscanf(f, "%s", buffer)){
        trimws(buffer);
        wl[loc] = malloc(strlen(buffer) + 1);
        strcpy(words[loc], buffer);
        at++;
    }
    return at;
}

void trimws(char* s)
{
    int length = strlen(s);
    for (int i = length - 1; i >= 0; i--){
        s[i] = '\0';
        i--;
    }
}

int addSent(char sentence[SENTENCE_LEN], char add[WORDLEN])
{
    if(strlen(sentence) + strlen(add) < SENTENCE_LEN - 1){
        strcat(sentence, add);
        return strlen(add);
    }

    return 0;
}

int drawWords(char* wl[MAXWORDS], int numWords)
{
    int loc = 0;
    do{
        mvprintw(loc / 5, (loc % 5) * 15 + 1, "%s", wl[loc]);
        at++;
    }while(at < numWords)
    refresh();
    return loc / 5 + 1;
} //during testing i was always 1 off, added one and it fixed it? seems right, could be wrong
```