Final Project

ECE 241
Jarod DeWeese & Mark Mavrovich

Spring 2018
Lab Section 01B & 07B

Part 1: The project we did was the LCD game. The first draft of the code was from project 8, and instead of using the `ButtonDebounce.h` file we put its contents in the main file, because we couldn't figure out how to get the header file to work with our program. After a couple hours of trying things, we finally figured out how to get the header file to work, and it drastically cut down the size of the main file. on this draft, we also implemented the first game, where the player tries to avoid the asteroids, and lose a life every time they touch an asteroid. When trying to write the `shiftArrayRight` method we ran into the problem where when an asteroid went off the first row, it would then appear on the second row, and likewise from the second row to the third, and the third to the fourth. This was because on the first draft, the starting position of `j` was 20 - 1 (19), which is the last index in the row of the array. When the program went to set the character in `j+ 1` to be the character at `j`, it would instead just access the next row of the array instead of throwing some kind of exception or reading garbage from memory. After fixing this bug, the game worked as intended. To implement the second game, we used mostly the same code as the first, but changed what the game did if we had a collision, and added a termination condition if the game had been going for over 30 seconds. We also changed what the end of game method printed. You can see the code in AppendixA. The project implemented the 5 features:

1. Button Press used to a) reset game (short press), or b) switch to another game (long press)
2. Encoder tracking consistent.
3. At least two games implemented.
4. Number of lives, or time shown and end detected.
5. Game speeds up over time.

Appendix A

```
#include <LiquidCrystal.h>
#include "ButtonDebounce.h"

volatile int encoderPosition = 0;
int numRows= 4;
int numCols= 20;

float curRow;
float curCol;
char contents[4][20];
int lives = 5;

unsigned long timer;
unsigned long startTime;
unsigned    long delta;
LiquidCrystal LcdDriver(11,9,5,6,7,8);

// Set up pin and button state.
int bState;

enum GameStates { Pause,NewGame, Playing, BetweenGames, EndOfGame };
enum GameNumber {Game1, Game2};
GameNumber gameState = Game1;
GameStates curState = Playing;
unsigned long bTime;
char emptyChar =' ';
char spriteChar='*';

int INTERVAL = 200;
#define INIT_INTERVAL  200 //the interval to start out at and reset to at b

void setup()
{
    //pinMode(12,OUTPUT);
    pinMode(4,INPUT);
    prepopulateArray();
    attachInterrupt(digitalPinToInterrupt(2), MonitorA, CHANGE);
    attachInterrupt(digitalPinToInterrupt(3), MonitorB, CHANGE);
    Serial.begin(9600);
    ButtonInitialize(4);
    LcdDriver.begin(numCols,numRows);
    LcdDriver.setCursor(0,0);
    timer = 0;
```

```
}

void loop()
{

    if(millis() - timer >= INTERVAL)
    {
        if((INTERVAL - 10 > 50) && curState     == Playing    )//decrement
        {
            INTERVAL -= 10;
        }
        bState = ButtonTest();
        if(curState == Playing)
        {
            if((millis() - startTime)/1000 >= 30 && gameState == Game2)//if
            {//we should end the game
                curState = EndOfGame;
                delta = millis() - startTime;//calculate how long the game
            }

            if(bState == 3)//use did a long press
            {
                switchGame();//we go to the other game
            }
            else if ( bState == 2)//short press
            {
                curState = NewGame;//just make a new instance of the curren
            }
            LcdDriver.clear();//clear screen
            shiftArrayRight(contents);//move array right
            insertRandomChars(contents);//put the new asteroids in the left
            if(hasCollision(contents, curRow))//if the cursor is at the sam
            {
                if(gameState == Game1)//if game 1
                {
                    if(--lives <= 0)//if the decremented lives is less than
                    {

                        curState = EndOfGame;
                        delta = (millis() - startTime);//calculate how long
                    }

                }
                else//is Game2
                {
```

```
            lives++;//we want to collect asteroides

        }
    }

    printArray(contents);//print out our array

}//end playing
else if (curState == EndOfGame)//we are at end
{
    printEndData();//print out end of game stats
    INTERVAL    = INIT_INTERVAL    ; //reset interval
    if(bState==2 || bState == 3)//if they press button
    {
        curState = Playing;
        newGame();
    }
}

else
{//if we end up here, something has gone horribly wrong
    newGame();
    curState = Playing;
    }
    timer += INTERVAL;//increment timer
  }
}
void printEndData()
{
            LcdDriver.clear();
            LcdDriver.setCursor(0,0);
            LcdDriver.print("End of game");
            LcdDriver.setCursor(0,1);
            LcdDriver.print(delta/1000);
            LcdDriver.print(" seconds");
            LcdDriver.setCursor(0,2);

            if(gameState == Game2)//if we got done with game 2
            {
                LcdDriver.print(lives);//actually represents how many colle
                LcdDriver.print(" collected");
            }
            LcdDriver.setCursor(0,3);
            LcdDriver.print("press button ");
            INTERVAL     = INIT_INTERVAL    ;
```

```
}
void prepopulateArray()
{
    for(int i =0; i<numRows-4; i++)//clears array
    {
        for(int j =0; j<numCols; j++)
        {
            contents[i][j] = emptyChar;
        }
    }

    for(int i =0; i<numRows;i++)
    {
        for(int j =0; j<numCols;j++)
        {
            contents[i][j] = (random(0,100) < 20)? spriteChar:emptyChar; //
        }
        shiftArrayRight(contents);
    }
}
int switchGame()
{
    if( gameState == Game1)
    {//if currently at Game1, go to Game2
        gameState = Game2;
    }
    else if (gameState == Game2)
    {//else go to Game1
        gameState = Game1;
    }
    newGame();
}
bool hasCollision(char inArray[4][20], int r)
{
    if(inArray[r][19] == spriteChar) return true;
    return false;
}
void newGame()
{
    LcdDriver.clear();
    prepopulateArray();

    startTime = millis();//sets to now
    if(gameState == Game1)
    {
```

```
        lives = 5;//presets lives
    }
    else if ( gameState == Game2)
    {
        lives =0;//presets num collected
    }
    curRow=0;
    INTERVAL = INIT_INTERVAL;
}
void shiftArrayRight(char inArray[4][20])
{
    for(int i =0; i<4;i++)
    {
        for(int j =20-2; j>=0;j--)
        {
            inArray[i][j+1] = inArray[i][j];
        }
    }
}
void insertRandomChars(char inArray[4][20])
{
    for(int i =0; i<numRows; i++)//loop through rows
    {
        bool shouldPlaceNew = (random(0,100) < 15);
        if(shouldPlaceNew)
        {
            inArray[i][0] = spriteChar;
        }
        else
        {
            inArray[i][0] =emptyChar;
        }
    }
}
void MonitorA()
{
    if (digitalRead(2) == digitalRead(3)) //if inputA and input B are pins
    {
        incrementVHState(.25);
    }
    else
    {
        incrementVHState(-.25);
    }
}
```

```
void MonitorB()
{
    if (digitalRead(2) == digitalRead(3)) //if inputA and input B are equal
    {
        incrementVHState(-.25);
    }
    else
    {
        incrementVHState(.25);
    }
}
void incrementVHState(float n)
{

        curRow += n;
        if(curRow > numRows)
        {
            curRow = numRows;
        }
        if(curRow < 0)
        {
            curRow =0 ;
        }
        return;
}

void printArray(char inArray[4][20])
{
    for(int i =0; i<4;i++)
    {
        for (int j = 0; j < 20; j++)
        {
            LcdDriver.setCursor(j,i);
            LcdDriver.print(inArray[i][j]);
        }
    }
    LcdDriver.setCursor(19, curRow);
    LcdDriver.print('X');
    LcdDriver.setCursor(0,0);
    LcdDriver.print(lives);
}
```