# Final Project

ECE 241   
Jarod DeWeese & Mark Mavrovich Spring 2018   
Lab Section 01B & 07B

**Objective**: To create an 2 LCD games with an Arduino and external LCD screen that implemented the requirements. Both were scrolling games, where asteroids would approach the player. At the end of the games, they were supposed to print the correct results.

**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](#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.

The paradigm that we used was a state machine, using two kinds of enums, one to represent the game we were playing, and another to represent the game state (playing, end of game, new game, etc). Another important thing about the software of this project was writing lots of descriptive helper methods, so we just called those in the main program, which allowed us to think about the flow of the program at a more abstract level.

The testing we did was just to make sure the functionality worked, that the short and long presses did what they were supposed to, and that we accurately kept track of lives and encoder position.

If the current state was `Playing`

    If we are over 30 seconds and are game2, we set the state to be  EndOfGame and calculate the delta.

    If there was a long press, we switch the game we are playing

    If there was a short press, we make a new game

    We then clear the LCD and perform operations on the array to update it.

    If there was a collision, we handle that depending on which game we are playing.

If the current state was `EndOfGame`

    We print the end Data , reset `INTERVAL`

    If there was a button press, we change the state to playing and make a new game.

If we are in some other state, we make a new game, and set the current state to Playing.

You will find pictures in [AppendixB](#AppendixB)

# 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 beginning of each game e

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 time interval if playing to speed up game

        {

            INTERVAL -= 10;

        }

        bState = ButtonTest();

        if(curState == Playing)

        {

            if((millis() - startTime)/1000 >= 30 && gameState == Game2)*//*if we are 30 seconds into game2

            {*//*we should end the game

                curState = EndOfGame;

                delta = millis() - startTime;*//*calculate how long the game was played for

            }

            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 current game

            }

            LcdDriver.clear();*//*clear screen

            shiftArrayRight(contents);*//*move array right

            insertRandomChars(contents);*//*put the new asteroids in the leftmost side

            if(hasCollision(contents, curRow))*//*if the cursor is at the same place an asteroid

            {

                if(gameState == Game1)*//*if game 1

                {

                    if(--lives <= 0)*//*if the decremented lives is less than zero

                    {

                        curState = EndOfGame;

                        delta = (millis() - startTime);*//*calculate how long the game lasted

                    }

                }

                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 collecetd

                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;*//*sets to space

        }

    }

    for(int i =0; i<numRows;i++)

    {

        for(int j =0; j<numCols;j++)

        {

            contents[i][j] = (random(0,100) < 20)? spriteChar:emptyChar; *//*has a 20% chance of inserting a \* at [i][j]

        }

        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 2 and 3 respectively

    {

        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);

}

# Appendix B

This is the screen that is displayed at the end of the first game, it shows the number of seconds that the player played the game.

This is the screen that is displayed at the end of the second game, it shows the number that the player was able to collect in the 30 second time span.

This is the game in action, the cursor is on the right side, and the asteroids are moving across the screen, with the score/lives at the top left.