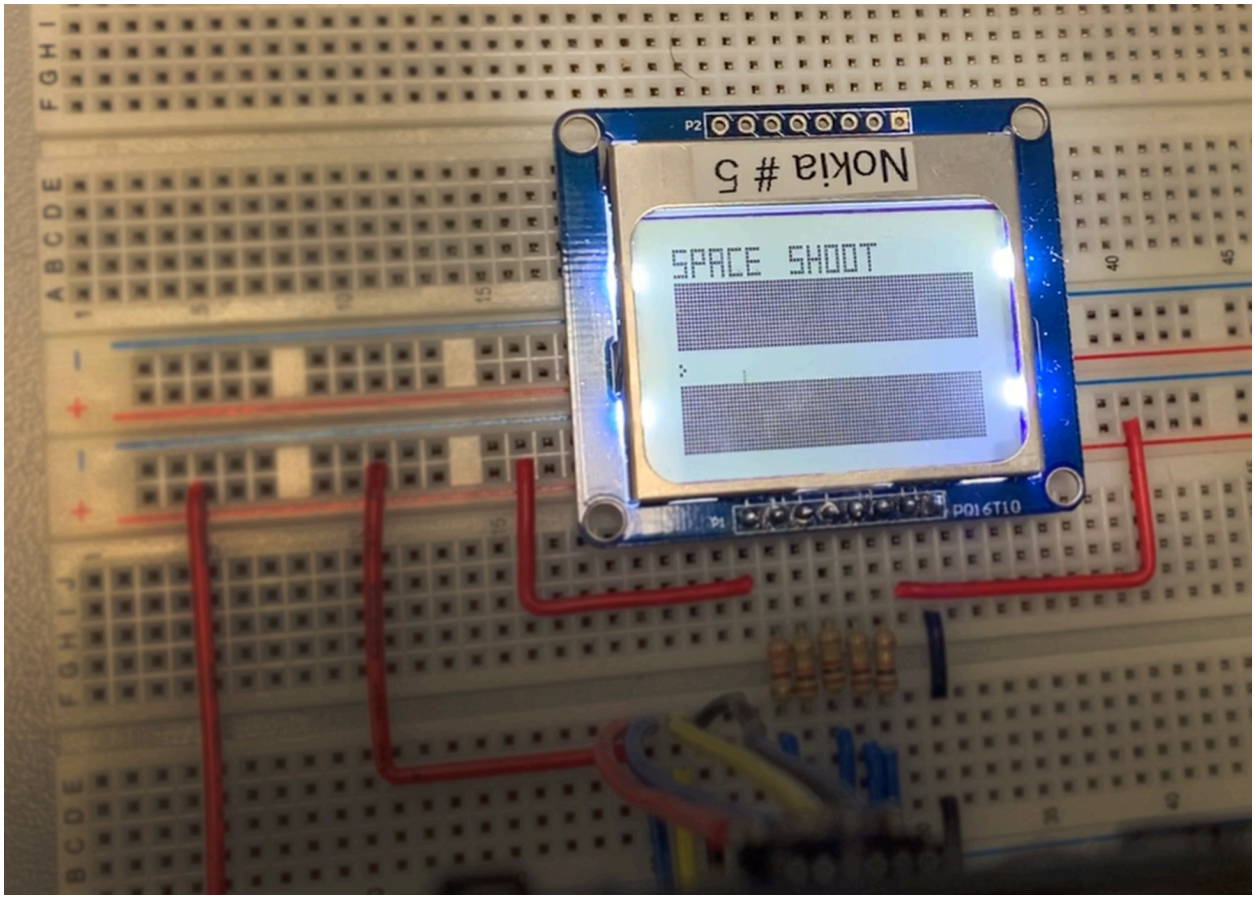
Cover Page

Space Shooter
David Estrin
Jan 2024



Detailed Description of Project

## Purpose of Project

I built this interactive space shooter game in order to learn how to code programs on platforms that don't offer too many supports, like libraries, and to rival the Wii (it has too much market share!). I set out to utilize what I used in class (ultrasonic sensors, user inputs, loops) and learn how to code on LCD displays in order to challenge myself. With my ambitious goal, I even learned more about functions, random values, counting systems and about bytes in the code as I challenged myself to not only get the lcd display to work, but to also create a motion controlled space shooter style game.

## Game Setup

To start my game, the user has to upload my code to the arduino uno then open up the serial monitor. Once prompted, enter "Start" to start the game.
To play, simply position the controller in front of the ultrasonic sensor (towards the eye looking cylinders). Observe how the distance between the controller and the sensor changes the position of your spaceship on the lcd display. As you move farther away from the sensor, your spaceship will move in the upward direction & as you move it closer your spaceship will move in the downward direction.

## Gameplay Mechanics

The goal of the game is to survive for as long as you can and get the highest score. To do this you must navigate the spaceship away from the obstacles that fly towards your spaceship. Everytime you successfully pass an obstacle, the game will display your current level in the game. Through levels 1-10 players learn basic controls and as players advance obstacles become faster and user controls evolve. Obstacles differ across the different levels and their location is randomized with obstacles becoming more challenging as the game progresses. As the levels progress, players also face changing controls where the spaceship's position on the board changes relative to the controller. The in-game position of the spaceship mirrors the real-life movement between the controller and ultrasonic sensor. On harder levels, the ratio of movement changes forcing players to relearn controls.

## End-Game Details

The game ends when you make contact with the obstacles. The game will display that the game has ended on the serial monitor as well as the final score achieved by the player marking the end of the gaming experience.
To restart the game, simply press the reset button on the Arduino Uno.

Project Breakdown

## Serial Inputs in Space Shooter

In my game, the serial monitor is used to start the game using serial inputs and display critical information to the user. To get my serial inputs working, I started by initializing the serial monitor with the “Serial.begin(9600)” function. This set the communication speed to be 9600 bits per second which set the Arduino to the appropriate port. The next step was to check when a user input, which was done by using the “available()” function to check if there was any input done by the user. This would loop with the void loop() until a player would enter something in the serial monitor which would then be checked using the “readString()” function. Using the string, I used the “trim()” and “toUpperCase()” functions that I learned in class (during the challenges) to modify the input to fix capitalization and spacing errors in my program. My next step was to check if the value was what I expected, in this case “START” to start my program. Once that was true, I continued onto my game and used my serial monitor with the “println()” function to display further information like the game status and the users score.

## Components in the Project

During this project I used multiple components including the Nokia 5110 LCD display, ultrasonic sensor and Arduino Uno to create an motion controlled interactive gaming experience for my user. To start, I used the Nokia 5110 LCD display to display the game's visuals to the player. When starting the project, I had the choice between the Waveshare 1.5 inch OLED display and an LCD display. Given the limited information available for the OLED display as well as the time constraints, I opted to go for the LCD display. The LCD display worked with 8 pins, LED, RST, CS, D/C, DIN, CLK, VCC & GND, that were connected to different parts of the Arduino Uno.

- I connected the LED pin to the positive (5V) on the Arduino Uno to control the backlight,
- the VCC pin to the positive (5V) on the Arduino Uno to provide power to the LCD display,
- the GND pin to the ground connection on the Arduino Uno (GND),
- the RST pin to digital pin 6 on the Arduino board to reset the LCD display,
- the CS pin to digital pin 5 on the Arduino board to enable or disable the communication,
- the D/C pin to digital pin 4 on the Arduino board to recognize data & commands sent to the display,
- the DIN pin to digital pin 3 on the Arduino board to transmit data to the display and
- the CLK pin to digital pin 2 on the Arduino board to synchronize the timing between the data sent from the Arduino Uno.

I also used an ultrasonic sensor that worked with 4 pins, VCC, Trig, Echo & GND, that were also connected to the Arduino Uno. It worked by sending pulses through the trigger (Trig) pin and measuring the time it is sent between the time it returns back through the Echo pin. The ultrasonic sensor’s Trig pin is connected to the Arduino Uno’s digital pin 10 and the Echo pin is connected to the Arduino Uno’s digital pin 9. Similar to the Nokia 5110 LCD display, the VCC pin connects to the positive (5V) to the Arduino Uno & the GND pin connects to the Arduino Uno’s ground.
The Arduino Uno has a lot of pins and components making it work; however, I mainly used the positive (5V), GND & Digital pins during this project. These allowed me to send and receive signals from the sensor and lcd display.

## Key Code Segments and Project Functionality

One essential component to my project was getting the Nokia 5110 LCD display to work. I spent the first few days trying to download the Adafruit libraries to no success; however, determined to get it working I consulted my teacher and began learning how to project images & animate them on the lcd display. This is what I learned:

A key component to get any image on the screen is to initialize it. This is done by a LCD_initilise() function that sets up the lcd display to receive input properly.
The lcd display also needs to know what to display, which is done by another key component and which is stored in an array known as grap[]. This array stores 0x0(insert hexadecimal value) followed by a hexadecimal value starting from 1 until FF. As the value increases, the 1x8 pixelated section displays the number in binary. This array stores each of the pixels and the current hexadecimal value that is being displayed.
To display this, it is necessary to create a LCD_write() function to send these hexadecimal values to the lcd display which is then looped for every pixel for every frame of the code.

Now that the user has an interface to use, they need to be able to control a spaceship with the ultrasonic sensor.
Now it is critical to add some difficulty to the game in the form of obstacles. I did this by creating the function drawObs() to keep track of the level and update the hexadecimal values for certain pixels each frame to animate the objects on the lcd display. To add collisions, I associated a value to their positions using the obsFrame and obsIndex and checked for specific collisions with ssPosition using this collision() function.

## New Key Concepts Learned in the Project

Throughout this project I am very proud to have applied my learning throughout this course and built up on it to apply it to a real project. Whether it was from binary/hexadecimal numbers to if statements, everything I learned in class played a key role in helping me to focus on learning new concepts while researching and doing the project.

- I learned about the differences between OLED and LCD displays and how to implement them into my project.
- The most challenging thing I found was figuring out how to get the Nokia 5110 LCD display to work but as I researched, tested and consulted I was able to successfully get it to work.
- There was so much to learn from working with collisions and animating the display without libraries.
- I learned some out of the box methods to code without any set libraries like associating values with collisions & effectively working with pixel-by-pixel coding.
- I also learned how to use the random() & randomSeed() inbuilt functions to create randomized sets of random values. Not only did I learn about inbuilt functions, but I also learned how to create my own functions with parameters and how to use them effectively to make my code more efficient.

Before this project I couldn't fathom the idea of creating games without libraries and game engines; however, after completing this project I now look at the technology around me differently. Now when I look at cellphones and the old nintendo games I can visualize how some of the parts and backend works making me value its technology much more than I would have done previously.

Final Build