IOWA STATE UNIVERSITY

AEROSPACE ENGINEERING DEPARTMENT

Computational Techniques for Aerospace Design

AERE 361

Spring 2021

Final Project Report
AirBuds

Team Member Names :
Sitarski, Dylan
Nasers, Ryan
Ackert, John
Shramek, Cade
Romans, Dillon

Contents

1 ABSTRACT

This report will cover the creation, design, and execution of our project Light Painter. We will start by discussing our team's goals for the game, how we ended up making it, and how we made it interesting and satisfying. We wanted to see if we could create a game that would work on a simple CPX board, while being interesting, fun, and challenging. In the end, it should be able to be considered a classically "good" game based on simple assessments of mechanics and possibilities. To accomplish this, we needed to research what has been found to be important principles of good games. Using the proven principles, we were determined to achieve our goal. We introduced each principle through a mechanic of its own which was introduced specifically to achieve a response and lasting impact in the player. Our group believes that we have successfully achieved our goals, and even that the game could have been taken further with real potential.

2 INTRODUCTION

Our project was created by our team consisting of Cade Shramek, Dillon Romans, Dylan Sitarski, John Ackert, and Ryan Nasers. The project came to mind during a team meeting where Dylan mentioned attempting to create a handheld game the touched on all five pillars of game design. After the group agreed we began brainstorming ideas for what sort of game we could use the CPX board to create. A light based game eventually was chosen where the player would interact with the LED's on the CPX board.

Light Painter is the game that was designed to be played on the CPX board. We wanted the game to be small and simple to play, yet rewarding and entertaining.[2] The goal of creating this game was to keep it simple, but complex enough to be played without much explanation, and learned through player exploration. We began with a rule set and expanded upon that as newer or more interesting ideas came through. We began work on creating the game and now we will discuss its progress throughout the semester.

3 FEATURES

Light Painter is our hand held game, based on revolving colors that tests the players reactions and quick thinking. This game is controlled by using the 2 buttons located on the face of CPX board, one controls the direction your cursor, and the other changes the color of the LED beneath your cursor. Connected to the CPX board is a LED strip that shows the players progress by the amount of colors that are lit up, the more lights your have the more progress you have made.

4 PROBLEM STATEMENT

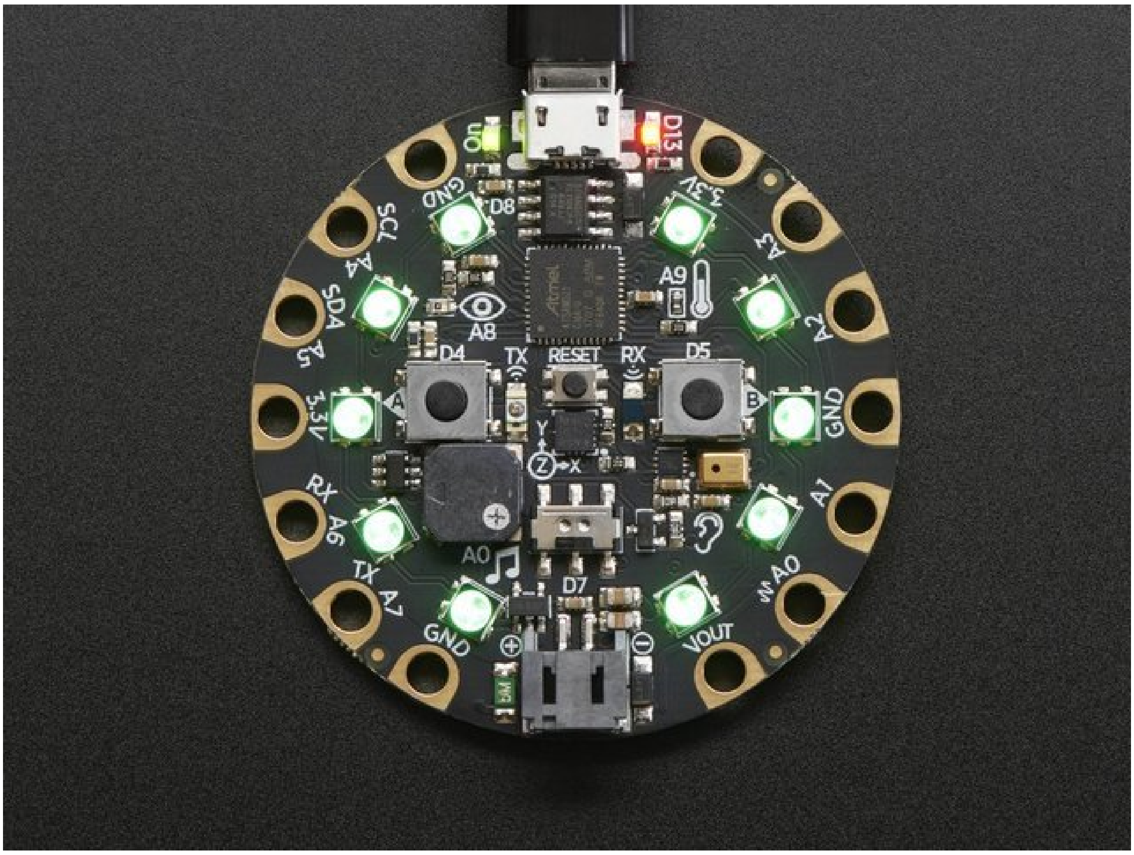
For our problem statement, we wanted to create a game that was simple enough for the Circuit Playground Express, but still satisfy the design principles of a good game.[3] We wanted players to experience a sense of freedom and curiosity, as well as a lasting impact. The game needed to have some sort of progression that would eventually lead to a reward. Our group desired to have competition that would allow for skill mastery and replay-ability. Most importantly, our main objective was for the player to have a high retention rate and to have fun!

5 PROBLEM SOLUTION

The solution we created to make this game meet all of our goals was to use a system of colors on the LEDs and player controlled changes. These changes of the colors and their locations directly affect the game and players will begin to learn what each color does. The player starts with a single color and as the pointer orbits the LEDs he can change the color of specific LEDs. Starting with only one color forces the player to change LEDs until there is a change in the game. These changes compound and force the player to learn what each color does while attempting to unlock the next color.

Understanding progression is also important. This is where the neopixel LED strip comes in. The LED strip lights up with each color the player currently has. When the conditions are met to unlock the next color, the LED strip will light up with the next color.

Understanding the goals for the game came down to some research as well to make sure our ideas worked. Ensuring that our ideas met some criteria of game design we referenced a guide to game design[1].



6 STATUS

At this point of Light Painters development, we have a fully working playable game that follows a set of rules that a player must follow in order to progress. The project can be considered an overall success, with working features that are simple but engaging and that seem to hold the player's attention well. The game closely follows the principles of good game design, including features of lasting impact, reward, progress, creative freedom and curiosity, and more. We had some trouble adding a JSON file to track time to completion, but people can time themselves anyways to add the elements of competition and mastery.

6.1 Lessons Learned

This project has given us more experience in a wide variety of areas. Firstly, we have not only gained experience with coding in C but also have learned how to use C to communicate with devices like the Circuit Playground Express. This has given us insight into how devices in real-world application communicate with each other and are programmed. Secondly, we now have more broad experience with working on a team to see a project through from brainstorming to final result. This gives us an advantage going into future classes and careers because we will be equipped to work better with a greater variety of teams. We have learned about managing project schedules, organizing meetings, and being team leaders and members. Finally, the project proposal, presentation, and this final report have helped us learn about technical communication and improve our skills in conveying our ideas and the importance of them.

7 RESULTS

We believe that the results of our project are positive and that our goals have been essentially met. When asked to play and review the game, we had quite positive reactions. Many people mentioned that the game had quite a bit to explore and learn. It wasn't just over in 1 or 2 minutes (especially if you don't know the rules). The rules were dynamic, and that for how simple the controls are the game is still not too easy.

Our GitHub repository can be found at:

`https://github.com/AerE361/AirBuds/tree/main`

8 FUTURE WORK

Changes and improvements that could be made for this game include making the game more competitive and adding something like a leader board. A leader board could track stats such as completion of the game in least moves or least amount of time. Other changes could be made including adding more features and colors. Since some colors add changes to the game play, if we were to add more we could expand the different features within the game. The game also has plenty of room to grow with the platforms it is made for. For instance, a LED array could make the game 2 dimensional and creates more possibilities for game play. If we would continue to develop the program, we want to bring it in the direction of the whole is more than the sum of its parts. Something that's similar to John Conway's game of life.

9 CONCLUSION

This report has covered the final form of our project. From the first ideas to the current features, our game has evolved in a way that we feel we can be proud of.

Creating a game that is truly fun and engaging but simple enough for the CPX is not a rudimentary task, but we believe that our final project is a satisfying solution. It was done using simple mechanics based on widely accepted principles, giving us a ”scientifically” fun game. Finally, we are all grateful for the experience we have gathered from this project and from working together!

References

[1] Carles Homs. A short guide to game design.

[2] Dustin Tyler. 3 primary game design principles to keep in mind when making games. June 2021. Online; posted 25-June-2021.

[3] Jonas Tyroller. Can we make this button fun to press?

A SOURCE CODE

Source Code

```
#include <Adafruit_NeoPixel.h>


#include <Adafruit_CircuitPlayground.h>
#include <Adafruit_Circuit_Playground.h>
#include <Arduino.h>
#include <math.h>

//If we knew how to get the SPI flash working we would use these.
//From what we found online there were basically nothing that would let us use the flash.  SPI flash library didn't work due to variant.h not having a QSPI/SPI flash defined in it.
//Very unsure how to fix that, but we're quite interested.
//#include <ArduinoJson.h>
//#include <SPI.h>
//#include <SdFat.h>
//#include <Adafruit_SPIFlash.h>

#define PIN 6
#define NUMPIXELS 30

//Defining all Colors so that code is easier to read.
#define BLACK 0
#define WHITE 1
#define YELLOW 2
#define GREEN 3
#define BLUE 4
#define RED 5
#define PURPLE 6
#define ORANGE 7
#define CYAN 8
#define PINK 9

//Color Setting Functions for easy reading.
void setColor(int pos,int color) {
  switch(color)  {
    case BLACK:
      CircuitPlayground.setPixelColor(pos,0,0,0);
      break;
    case WHITE:
      CircuitPlayground.setPixelColor(pos,255,255,255);
      break;
    case YELLOW:
      CircuitPlayground.setPixelColor(pos,255,255,0);
      break;
    case GREEN:
      CircuitPlayground.setPixelColor(pos,0,255,0);
      break;
    case BLUE:
      CircuitPlayground.setPixelColor(pos,0,0,255);
      break;
    case RED:
      CircuitPlayground.setPixelColor(pos,255,0,0);
      break;
    case PURPLE:
      CircuitPlayground.setPixelColor(pos,255,0,255);
      break;
    case ORANGE:
      CircuitPlayground.setPixelColor(pos,255,128,0);
      break;
    case CYAN:
      CircuitPlayground.setPixelColor(pos,0,255,255);
      break;
    case PINK:
      CircuitPlayground.setPixelColor(pos,244,0,180);
      break;
  }
}

//Variables for color burning logic
int greenBurned = 0;
int cyanBurned = 0;

//Variables for timing and direction
int location = 0;
```

```
bool dir = true;
int period = 500;

// Number of colors fire will burn in 2 seconds
int spreadRate = 1;
int currentColor = 1;
bool colorChanged = false;

//More timing variables
unsigned long time_now = 0;
unsigned long lastPress = 0;
unsigned long timerForCheck = 0;
unsigned long fireTimer = 0;
unsigned long lastTeleport = 0;
unsigned long lastPinkDirChange = 0;
unsigned long timeSpread = 0;


//Arrays for tracking what is happening
int ColorAPos[10] = {BLACK,BLACK,BLACK,BLACK,BLACK,BLACK,BLACK,BLACK,BLACK,BLACK};
int ColorPlaced[10] = {0,0,0,0,0,0,0,0,0,0};
bool UnlockedColors[10] = {true,true,true,false,false,false,false,false,false,false};
//Update the Circle Colors after an event
void updateCircle() {
  for(int i = 0; i<10; i++) {
    if (i != location) {
      setColor(i, ColorAPos[i]);
    }
  }
}


//The function that checks each time the board is changed to see if the player has unlocked new colors
void checkforcolorunlock() {
  for(int i = 0; i<10 ;i++) {
    if (!UnlockedColors[3]) {
      if( ColorAPos[(i-1 + 10)%10] == YELLOW && ColorAPos[i] == YELLOW && ColorAPos[(i+1)%10] == YELLOW) {
        UnlockedColors[3] = true;
        ColorAPos[(i-1 + 10)%10] = BLACK;
        ColorAPos[i] = GREEN;
        ColorAPos[(i+1)%10] = BLACK;
        Serial.println("GREEN␣UNLOCKED");
      }
    }
    if (!UnlockedColors[4]) {
      if ((ColorAPos[(i-1 + 10)%10] == GREEN && ColorAPos[i] == YELLOW) || (ColorAPos[(i-1 + 10)%10] == YELLOW && ColorAPos[i] == GREEN)) {
        UnlockedColors[4] = true;
        ColorAPos[(i-1 + 10)%10] = BLUE;
        ColorAPos[i] = BLUE;
        Serial.println("BLUE␣UNLOCKED");
      }
    }
    if (!UnlockedColors[5]) {
      if( ColorAPos[(i-1 + 10)%10] == GREEN && ColorAPos[i] == GREEN && ColorAPos[(i+1)%10] == GREEN) {
        UnlockedColors[5] = true;
        ColorAPos[i] = RED;
        Serial.println("RED␣UNLOCKED");
      }
    }
    if (!UnlockedColors[6]) {
      if( ColorAPos[(i-1 + 10)%10] == RED && ColorAPos[i] == BLUE && ColorAPos[(i+1)%10] == RED) {
        UnlockedColors[6] = true;
        ColorAPos[i] = PURPLE;
        Serial.println("PURPLE␣UNLOCKED");
      }
    }
    if (!UnlockedColors[8]) {
      if((ColorAPos[(i-1 + 10)%10] == BLUE && ColorAPos[i] == ORANGE) || (ColorAPos[(i+1)%10] == BLUE && ColorAPos[i] == ORANGE)) {
        UnlockedColors[8] = true;
        ColorAPos[(i-1 + 10)%10] = CYAN;
        ColorAPos[i] = CYAN;
        Serial.println("CYAN␣UNLOCKED");
      }
    }
  }
  updateCircle();
  updateStrip();
}

//Applys and effects to the board
void eventChecker() {

  //Teleporter
  if (ColorAPos[location] == PURPLE && millis() >= lastTeleport + 550) {
    if (dir) {
      for (int i = location+1; i<location+9; i++) {
        if (ColorAPos[i%10] == PURPLE) {
          location = i%10;
          setColor(location, WHITE);
          lastTeleport = millis();
          break;
        }
      }
    }
    else {
      for (int i = location-1; i>location-9; i--) {
        if (ColorAPos[(i+10)%10] == PURPLE) {
          location = (i+10)%10;
          setColor(location, WHITE);
          lastTeleport = millis();
          break;
        }
      }
    }
  }
  //Pink Direction Changer
  if (ColorAPos[location] == PINK && millis() >= lastPinkDirChange + 550) {
    dir = !dir;
    lastPinkDirChange = millis();
  }

  //Spreaders
  if (millis() >= timeSpread + 2/spreadRate*1000) {
    bool blueCheck = false, redCheck = false, cyanCheck = false;
    timeSpread += 2/spreadRate*1000;
    for (int i = 0; i<10; i++) {
    //Fire dowser
    if (ColorAPos[i] == BLUE && !blueCheck) {
      if (ColorAPos[(i-1+10)%10] == RED) {
        ColorAPos[(i-1+10)%10] = BLUE;
        blueCheck = true;
      }
      else if (ColorAPos[(i+1)%10] == RED) {
        ColorAPos[(i+1)%10] = BLUE;
        blueCheck = true;
      }
    }

    //Fire Spreader
    if (ColorAPos[i] == RED && !redCheck) {
      if (ColorAPos[(i+1)%10] == GREEN) {
        ColorAPos[(i+1)%10] = RED;
        greenBurned++;
        if (greenBurned > 3 && !UnlockedColors[7]) {
          ColorAPos[(i+1)%10] = ORANGE;
          UnlockedColors[7] = true;
        }
        redCheck = true;
      }
      else if (ColorAPos[(i-1+10)%10] == GREEN) {
        ColorAPos[(i-1+10)%10] = RED;
        greenBurned++;
        if (greenBurned > 3 && !UnlockedColors[7]) {
          ColorAPos[(i-1+10)%10] = ORANGE;
          UnlockedColors[7] = true;
        }
        redCheck = true;
      }
      else if (ColorAPos[(i-1+10)%10] == CYAN) {
        ColorAPos[(i-1+10)%10] = RED;
        cyanBurned++;
        if (cyanBurned > 3 && !UnlockedColors[9]) {
          ColorAPos[(i-1+10)%10] = PINK;
          UnlockedColors[9] = true;
        }
        redCheck = true;
      }
      else if (ColorAPos[(i+1)%10] == CYAN) {
        ColorAPos[(i+1)%10] = RED;
        cyanBurned++;
        if (cyanBurned > 3 && !UnlockedColors[9]) {
          ColorAPos[(i+1)%10] = PINK;
          UnlockedColors[9] = true;
        }
        redCheck = true;
      }
    }

    //Cyan Spreader
    if (ColorAPos[i] == CYAN && !cyanCheck) {
      if (ColorAPos[(i+1)%10] == BLACK || ColorAPos[(i+1)%10] == BLACK) {
        ColorAPos[(i+1)%10] = CYAN;
        cyanCheck = true;
      }
      else if (ColorAPos[(i-1+10)%10] == BLACK || ColorAPos[(i-1+10)%10] == BLACK) {
        ColorAPos[(i-1+10)%10] = CYAN;
        cyanCheck = true;
      }
    }
    }
    updateCircle();
    updateStrip();
  }
}
```

```


Adafruit_NeoPixel strip(NUMPIXELS, PIN, NEO_GRB +
    NEO_KHZ800);

void updateStrip(void) {
  for (int i = 0; i<10; i++) {
    if (UnlockedColors[i]) {
      stripSetColor(i,i);
    }
  }
}

void stripSetColor(int pos,int color) {
  switch(color)  {
    case BLACK:
      strip.setPixelColor(pos,0,0,0);
      break;
    case WHITE:
      strip.setPixelColor(pos,255,255,255);
      break;
    case YELLOW:
      strip.setPixelColor(pos,255,255,0);
      break;
    case GREEN:
      strip.setPixelColor(pos,0,255,0);
      break;
    case BLUE:
      strip.setPixelColor(pos,0,0,255);
      break;
    case RED:
      strip.setPixelColor(pos,255,0,0);
      break;
    case PURPLE:
      strip.setPixelColor(pos,255,0,255);
      break;
    case ORANGE:
      strip.setPixelColor(pos,255,128,0);
      break;
    case CYAN:
      strip.setPixelColor(pos,0,255,255);
      break;
    case PINK:
      strip.setPixelColor(pos,244,0,180);
      break;
  }
  strip.show();
}

//File myFile;

void setup() {
  Serial.begin(115200);
  CircuitPlayground.begin();
  CircuitPlayground.setBrightness(10);
  Serial.print("Initializing␣Flash");
  strip.begin();
  strip.show();
  strip.setBrightness(50);

}
bool lastState = CircuitPlayground.slideSwitch();

void loop() {
  //Color Changer / Timer
  if(millis() >= time_now + period){
    if (colorChanged != false) {
      setColor(location,currentColor);
      ColorAPos[location] = currentColor;
    }
    if (dir) {
      location++;
    }
    else {
      location--;
    }
    location = location%10;
    if (location<0) {
      location = 9;
    }

    if (ColorAPos[location] == GREEN) {
      time_now += period*2;
    }
    else if (ColorAPos[location] == ORANGE) {
      time_now += period*.5;
    }
    else {
      time_now += period;
    }

    setColor(location,WHITE);
    colorChanged = false;
    currentColor = 1;
    for (int i = 0; i<10; i++) {
      Serial.print(ColorAPos[i]);
    }
    Serial.println(" ");

  }

  if(millis() >= lastPress+300) {
    if(CircuitPlayground.rightButton()) {
      lastPress = millis();
      dir = !dir;
    }

    if(CircuitPlayground.leftButton()) {
      lastPress = millis();
      time_now = millis();
      currentColor++;
      while(UnlockedColors[currentColor] != true) {
        currentColor++;
        currentColor = currentColor%10;
      }
      setColor(location,currentColor);
      colorChanged = true;
    }

    if(CircuitPlayground.slideSwitch() == !lastState) {
      for (int i = 0; i <10; i++) {
        ColorAPos[i] = 0;
      }
      CircuitPlayground.clearPixels();
      lastState = !lastState;
    }
  }


  checkforcolorunlock();
  eventChecker();
}
```