GÖMÜLÜ SİSTEMLER PROJE ÖDEVİ RAPORU

Okul:

Bolu İzzet Baysal Üniversitesi

Proje Üyeleri:

Bekir ORUK

Cevahir ATICI

Hediye Zeynep ÖZDER

Projenin Adı:

LCD ekranda oyun

Projenin Amacı:

LCD ekranında butona basarak engel üstünden zıplattığımız bir oyun yapmak.

Projede Kullanılan Malzemeler:

- Arduino Uno
- Arduino Ana Kablosu
- 10k Potansiyometre
- Buton
- Led
- 220 Volt Direnç
- Breadboard
- Jumper Kablo
- 16*2 Led Ekran

Proje Yapımında İzlenen Adımlar:

Breadboard' a Lcd ekran, 10k Potansiyometre ve buton yerleştirildi. Daha sonra Jumper kablolar ile projenin elemanları birbirine doğru şekilde bağlandı. Son olarak gömülü sistemin çalışması için hazırlanan kod Arduino Uno kartına ana kablo yardımıyla kopyalandı.

Projede Kullanılan Kodun Açıklamalı Hali:

```
#include <LiquidCrystal.h>

#define PIN_BUTTON 2

#define PIN_AUTOPLAY 1

#define PIN_READWRITE 10

#define PIN_CONTRAST 12
```

Yukarıdaki kod, LiquidCrystal kütüphanesini içe aktarır ve bazı sabitlerin tanımlanmasını sağlar. Bu sabitler, belirli pin numaralarına atanır ve bu pinlerin ne işe yaradığını belirtir.

```
#define SPRITE_RUN1 1

#define SPRITE_RUN2 2

#define SPRITE_JUMP 3

#define SPRITE_JUMP_UPPER '.'

#define SPRITE_JUMP_LOWER 4

#define SPRITE_TERRAIN_EMPTY ' '

#define SPRITE_TERRAIN_SOLID 5

#define SPRITE_TERRAIN_SOLID_RIGHT 6

#define SPRITE_TERRAIN_SOLID_LEFT 7
```

Yukarıdaki kod, çeşitli sabitlerin tanımlandığı bir koddur. Bu sabitler, bir oyun veya animasyon senaryosunda kullanılan farklı grafik öğelerini temsil etmek için kullanılabilir.

```
#define HERO_HORIZONTAL_POSITION 1
```

Yukarıdaki kod, bir sabitin tanımlandığı bir koddur ve bu sabit, kahramanın ekran üzerindeki yatay konumunu temsil eder.

HERO_HORIZONTAL_POSITION, kahraman karakterin ekran üzerindeki yatay pozisyonunu belirtir.

```
#define TERRAIN_WIDTH 16

#define TERRAIN_EMPTY 0
```

```
#define TERRAIN_LOWER_BLOCK 1

#define TERRAIN_UPPER_BLOCK 2

#define HERO_POSITION_OFF 0   //kahramanın görünmediği kısmı belirtir.

#define HERO_POSITION_RUN_LOWER_1 1  // kahramanın alt satırda 1. Pozisyonda koşmasını belirtir.

#define HERO_POSITION_RUN_LOWER_2 2 // kahramanın alt satırda 2. Pozisyonda koşmasını belirtir.

#define HERO_POSITION_JUMP_1 3       // zıplama başlatır

#define HERO_POSITION_JUMP_2 4       // zıplamada yarısında olduğunu belirtir

#define HERO_POSITION_JUMP_3 5       // zıplamanın en üstte olduğunu belirtir

#define HERO_POSITION_JUMP_4 6       // zıplamanın en üstte olduğunu belirtir

#define HERO_POSITION_JUMP_5 7       // zıplamanın en üstte olduğunu belirtir

#define HERO_POSITION_JUMP_6 8       // zıplamanın en üstte olduğunu belirtir

#define HERO_POSITION_JUMP_7 9       // zıplamanın aşağı doğru yarı yolda olduğunu belirtir

#define HERO_POSITION_JUMP_8 10      // zıplamanın indiğini belirtir


#define HERO_POSITION_RUN_UPPER_1 11 // kahramanın üst satırda 1. Pozisyonda koşmasını belirtir.

#define HERO_POSITION_RUN_UPPER_2 12 // kahramanın üst satırda 2. Pozisyonda koşmasını belirtir.

LiquidCrystal lcd(11, 9, 6, 5, 4, 3);

static char terrainUpper[TERRAIN_WIDTH + 1];

static char terrainLower[TERRAIN_WIDTH + 1];

static bool buttonPushed = false;
```

Bu değişkenler, bir oyun veya uygulama senaryosunda LCD ekran kontrolü, zemin karakterleri ve düğme durumu gibi bilgileri tutmak veya kontrol etmek için kullanılabilir. Bu sayede LCD ekran üzerinde metin ve grafikleri görüntülemek, zeminin karakterlerini belirlemek ve düğme durumunu izlemek gibi işlemler gerçekleştirilebilir.

```
void initializeGraphics(){

  static byte graphics[] = {

    // 1. Koşma pozisyonu

    B01100,
```

```
B01100,
B00000,
B01110,
B11100,
B01100,
B11010,
B10011,
```

// 2. Koşma pozisyonu

```
B01100,
B01100,
B00000,
B01100,
B01100,
B01100,
B01100,
B01110,
```

// Zıplama

```
B01100,
B01100,
B00000,
B11110,
B01101,
B11111,
B10000,
B00000,
```

// Aşağı inme

```
B11110,
B01101,
B11111,
B10000,
```

```
B00000,
B00000,
B00000,
B00000,
```

// Yere inme

```
B11111,
B11111,
B11111,
B11111,
B11111,
B11111,
B11111,
B11111,
```

// Sağa inme

```
B00011,
B00011,
B00011,
B00011,
B00011,
B00011,
B00011,
B00011,
```

// Sola inme

```
B11000,
B11000,
B11000,
B11000,
B11000,
B11000,
B11000,
```

```
    B11000,
  };
int i;
  for (i = 0; i < 7; ++i) {
            lcd.createChar(i + 1, &graphics[i * 8]);
  }
  for (i = 0; i < TERRAIN_WIDTH; ++i) {
    terrainUpper[i] = SPRITE_TERRAIN_EMPTY;
    terrainLower[i] = SPRITE_TERRAIN_EMPTY;
  }
}
```

initializeGraphics fonksiyonu, grafiklerin oluşturulması ve zemin karakter dizilerinin başlangıç değerlerinin atanması gibi ön hazırlıkları gerçekleştirir. Bu hazırlıklar, oyun veya uygulamanın grafiksel öğelerini oluşturmak veya sıfırlamak için kullanılabilir.

```
void advanceTerrain(char* terrain, byte newTerrain){
  for (int i = 0; i < TERRAIN_WIDTH; ++i) {
    char current = terrain[i];
    char next = (i == TERRAIN_WIDTH-1) ? newTerrain : terrain[i+1];
    switch (current){
      case SPRITE_TERRAIN_EMPTY:
        terrain[i] = (next == SPRITE_TERRAIN_SOLID) ?
SPRITE_TERRAIN_SOLID_RIGHT : SPRITE_TERRAIN_EMPTY;
        break;
      case SPRITE_TERRAIN_SOLID:
        terrain[i] = (next == SPRITE_TERRAIN_EMPTY) ?
SPRITE_TERRAIN_SOLID_LEFT : SPRITE_TERRAIN_SOLID;
        break;
      case SPRITE_TERRAIN_SOLID_RIGHT:
        terrain[i] = SPRITE_TERRAIN_SOLID;
        break;
      case SPRITE_TERRAIN_SOLID_LEFT:
        terrain[i] = SPRITE_TERRAIN_EMPTY;
```

```
      break;
    }
  }
}
```

advanceTerrain fonksiyonu, terrain adında bir karakter dizisi (string) ve newTerrain adında bir bayt değeri alır. Fonksiyonun görevi, terrain dizisini güncellemek ve zemin karakterlerini ilerletmektir. İlerleyen bir platform oyununda, karakteri düzenli bir şekilde ilerletmek veya değiştirmek için kullanılır. Bu sayede oyunun dinamik ortamı simüle edilir.

```
bool drawHero(byte position, char* terrainUpper, char* terrainLower, unsigned int score) {
  bool collide = false;
  char upperSave = terrainUpper[HERO_HORIZONTAL_POSITION];
  char lowerSave = terrainLower[HERO_HORIZONTAL_POSITION];
  byte upper, lower;
  switch (position) {
    case HERO_POSITION_OFF:
      upper = lower = SPRITE_TERRAIN_EMPTY;
      break;
    case HERO_POSITION_RUN_LOWER_1:
      upper = SPRITE_TERRAIN_EMPTY;
      lower = SPRITE_RUN1;
      break;
    case HERO_POSITION_RUN_LOWER_2:
      upper = SPRITE_TERRAIN_EMPTY;
      lower = SPRITE_RUN2;
      break;
    case HERO_POSITION_JUMP_1:
    case HERO_POSITION_JUMP_8:
      upper = SPRITE_TERRAIN_EMPTY;
      lower = SPRITE_JUMP;
      break;
```

```
    case HERO_POSITION_JUMP_2:
    case HERO_POSITION_JUMP_7:
      upper = SPRITE_JUMP_UPPER;
      lower = SPRITE_JUMP_LOWER;
      break;
    case HERO_POSITION_JUMP_3:
    case HERO_POSITION_JUMP_4:
    case HERO_POSITION_JUMP_5:
    case HERO_POSITION_JUMP_6:
      upper = SPRITE_JUMP;
      lower = SPRITE_TERRAIN_EMPTY;
      break;
    case HERO_POSITION_RUN_UPPER_1:
      upper = SPRITE_RUN1;
      lower = SPRITE_TERRAIN_EMPTY;
      break;
    case HERO_POSITION_RUN_UPPER_2:
      upper = SPRITE_RUN2;
      lower = SPRITE_TERRAIN_EMPTY;
      break;
  }
```

drawHero fonksiyonu, kahraman karakterin pozisyonuna ve zemin karakter dizilerine (terrainUpper ve terrainLower) göre karakterin görüntüsünü çizer. Ayrıca, score adında bir unsigned int değişken alır ve çarpışma durumunu (collide) döndürür.

Bu fonksiyon, oyun veya uygulamada kahraman karakterin görüntüsünün çizilmesi ve güncellenmesi için kullanılır. Kahraman karakterin farklı pozisyonlarda ve hareketlerde nasıl görüneceği belirlenir ve ilgili grafikler kullanılarak ekrana çizilir. Fonksiyon aynı zamanda çarpışma durumunu kontrol eder ve çarpışma olduğunda collide değişkenini true olarak döndürür.

```
if (upper != ' ') {
    terrainUpper[HERO_HORIZONTAL_POSITION] = upper;
    collide = (upperSave == SPRITE_TERRAIN_EMPTY) ? false : true;
```

```
  }
  if (lower != ' ') {
    terrainLower[HERO_HORIZONTAL_POSITION] = lower;
    collide |= (lowerSave == SPRITE_TERRAIN_EMPTY) ? false : true;
  }
```

Bu bölümde, kahraman karakterin üst ve alt bölümlerinin zemin karakterleriyle etkileşimi kontrol edilir. Eğer kahraman karakterin üst veya alt bölümü zemin karakteriyle temas ediyorsa, ilgili zemin karakteri güncellenir ve çarpışma durumu belirlenir. Çarpışma durumu, collide değişkenine atanır ve sonuç olarak drawHero fonksiyonu tarafından döndürülür. Bu sayede çarpışma durumu oyun veya uygulama senaryosunda kullanılabilir ve ilgili işlemler gerçekleştirilebilir.

```
  byte digits = (score > 9999) ? 5 : (score > 999) ? 4 : (score > 99) ? 3 : (score > 9) ? 2 : 1;
```

Yukarıdaki satırda yer alan ifade, score değişkeninin değerine bağlı olarak digits adında bir byte değişkenin değerini belirler. Bu ifade, puanın basamak sayısını hesaplamak için kullanılır.

```
  terrainUpper[TERRAIN_WIDTH] = '\0';
  terrainLower[TERRAIN_WIDTH] = '\0';
  char temp = terrainUpper[16-digits];
  terrainUpper[16-digits] = '\0';
  lcd.setCursor(0,0);
  lcd.print(terrainUpper);
  terrainUpper[16-digits] = temp;
  lcd.setCursor(0,1);
  lcd.print(terrainLower);

  lcd.setCursor(16 - digits,0);
  lcd.print(score);

  terrainUpper[HERO_HORIZONTAL_POSITION] = upperSave;
  terrainLower[HERO_HORIZONTAL_POSITION] = lowerSave;
  return collide;
}
```

Yukarıdaki kod, drawHero fonksiyonunun son kısmını oluşturur.

Bu bölümde, LCD ekrana zemin ve kahraman karakterlerinin güncellenmiş hallerini, puanı ve çarpışma durumunu görüntülemek için gerekli işlemler gerçekleştirilir. LCD ekranın uygun konumlarına karakterler ve puan yerleştirilir ve sonuç olarak çarpışma durumu döndürülür.

```
void buttonPush() {
  buttonPushed = true;
}
```

buttonPush fonksiyonu, bir düğme veya butonun basılması durumunda çağrılır. Fonksiyonun görevi, buttonPushed değişkeninin değerini true yapmaktır.

Bu tür bir fonksiyon, bir düğmenin veya butonun durumunu izlemek ve bu durumu programın ilerleyişini etkilemek için kullanılabilir. Örneğin, düğme basıldığında belirli bir aksiyonun gerçekleşmesini tetikleyebilir veya belirli bir koşulu kontrol etmek için kullanılabilir. buttonPushed değişkeni, diğer kısımlarda bu durumun kontrol edilmesi için kullanılabilir.

```
void setup(){
  pinMode(PIN_READWRITE, OUTPUT);
  digitalWrite(PIN_READWRITE, LOW);
  pinMode(PIN_CONTRAST, OUTPUT);
  digitalWrite(PIN_CONTRAST, LOW);
  pinMode(PIN_BUTTON, INPUT);
  digitalWrite(PIN_BUTTON, HIGH);
  pinMode(PIN_AUTOPLAY, OUTPUT);
  digitalWrite(PIN_AUTOPLAY, HIGH);
  pinMode(A0,OUTPUT);

  attachInterrupt(0/*PIN_BUTTON*/, buttonPush, FALLING);

  initializeGraphics();

  lcd.begin(16, 2);
}
```

setup fonksiyonu, Arduino başlangıcında bir kez çalıştırılır ve gerekli pin konfigürasyonlarını yapar, kesmeleri ayarlar, grafikleri başlatır ve LCD ekranı başlatır. Bu işlev, programın diğer

kısımlarında kullanılan donanımların başlangıç durumlarını ayarlamak ve gereken ayarları yapmak için kullanılır.

```
void loop(){
  static byte heroPos = HERO_POSITION_RUN_LOWER_1;
  static byte newTerrainType = TERRAIN_EMPTY;
  static byte newTerrainDuration = 1;
  static bool playing = false;
  static bool blink = false;
  static unsigned int distance = 0;


  if (!playing) {
    drawHero((blink) ? HERO_POSITION_OFF : heroPos, terrainUpper, terrainLower,
distance >> 3);
    if (blink) {
      lcd.setCursor(0,0);
      lcd.print("Press Start");
    }
    delay(250);
    blink = !blink;
    digitalWrite(A0,blink);
    if (buttonPushed) {
      initializeGraphics();
      heroPos = HERO_POSITION_RUN_LOWER_1;
      playing = true;
      buttonPushed = false;
      distance = 0;
    }
    return;
  }
```

Bu loop fonksiyonu, Arduino'nun sürekli olarak çalıştığı ve oyunun durumunu ve ilerlemesini takip ettiği yerdir. Oyunun başlaması, oyunun oynanması veya oyunun yeniden başlatılması gibi durumlar bu fonksiyon içinde kontrol edilir ve ilgili işlemler yapılır.

```
  advanceTerrain(terrainLower, newTerrainType == TERRAIN_LOWER_BLOCK ?
SPRITE_TERRAIN_SOLID : SPRITE_TERRAIN_EMPTY);
  advanceTerrain(terrainUpper, newTerrainType == TERRAIN_UPPER_BLOCK ?
SPRITE_TERRAIN_SOLID : SPRITE_TERRAIN_EMPTY);
    if (--newTerrainDuration == 0) {
    if (newTerrainType == TERRAIN_EMPTY) {
      newTerrainType = (random(3) == 0) ? TERRAIN_UPPER_BLOCK :
TERRAIN_LOWER_BLOCK;
      newTerrainDuration = 2 + random(10);
    } else {
      newTerrainType = TERRAIN_EMPTY;
      newTerrainDuration = 10 + random(10);
    }
  }
  if (buttonPushed) {
    if (heroPos <= HERO_POSITION_RUN_LOWER_2) heroPos =
HERO_POSITION_JUMP_1;
    buttonPushed = false;
  }

  if (drawHero(heroPos, terrainUpper, terrainLower, distance >> 3)) {
    playing = false; // The hero collided with something. Too bad.
  } else {
    if (heroPos == HERO_POSITION_RUN_LOWER_2 || heroPos ==
HERO_POSITION_JUMP_8) {
      heroPos = HERO_POSITION_RUN_LOWER_1;
    } else if ((heroPos >= HERO_POSITION_JUMP_3 && heroPos <=
HERO_POSITION_JUMP_5) && terrainLower[HERO_HORIZONTAL_POSITION] !=
SPRITE_TERRAIN_EMPTY) {
      heroPos = HERO_POSITION_RUN_UPPER_1;
    } else if (heroPos >= HERO_POSITION_RUN_UPPER_1 &&
terrainLower[HERO_HORIZONTAL_POSITION] == SPRITE_TERRAIN_EMPTY) {
```

```
    heroPos = HERO_POSITION_JUMP_5;
   } else if (heroPos == HERO_POSITION_RUN_UPPER_2) {
    heroPos = HERO_POSITION_RUN_UPPER_1;
   } else {
    ++heroPos;
   }
   ++distance;


   digitalWrite(PIN_AUTOPLAY, terrainLower[HERO_HORIZONTAL_POSITION + 2] ==
SPRITE_TERRAIN_EMPTY ? HIGH : LOW);
  }
  delay(100);
}
```

Yukarıdaki loop fonksiyonu içerisinde yer alan kodlar, oyunun ilerlemesi ve olayların kontrol edilmesiyle ilgilidir. advanceTerrain fonksiyonu çağrılarak, terrainLower ve terrainUpper dizileri üzerinde yeni bir zemin oluşturulur. Eğer newTerrainType TERRAIN_LOWER_BLOCK ise, zemin solid (dolu) olacak şekilde güncellenir. Aksi halde, zemin boş olacak şekilde güncellenir. Belirli bir bekleme süresi (delay) kullanılarak oyun döngüsü kontrol edilir ve tekrar çalıştırılır.