Gömülü sistem tabanlı oyun geliştirme

Sezer Yildirim 210202001@kocaeli.edu.tr

Mart 2023

1 Özet

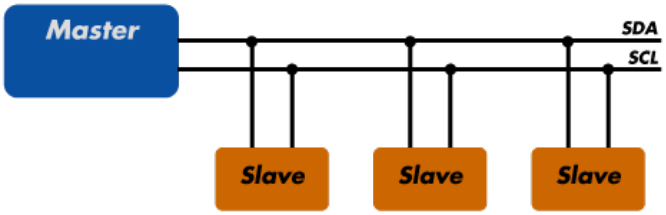
Mikroişlemciye sahip bir kart ve Simülasyon ortamları ile konsept bir oyun tasarlanmıştır. Mikroişlemci olması sebebiyle bellek yönetiminin yapıldığı pointerların sıkça kullanıldığı; donanım tabanlı bit kaydırma, piksel özellikleri, devre temelleri, Mux, demux bilgileri kullanılmıştır. Oyun ise bir uzay gemisi, rastgele düşen engellerden kaçarak ilerlemeli ve yeterli mühimmata sahip olursa objeleri yok etmelidir. Oyun, OLED ekran kullanarak görsel olarak sunulacak ve 8x16 matris üzerinde gerçekleştirilmiştir. Arduino IDE geliştirme ortamı ve Proteus entegre şekilde kullanılmıştır. Bir oyun yapısı temeli oyun motoru olmadan geliştirildiğinden oyun motoru temeli anlaşılmıştır.

2 Giriş

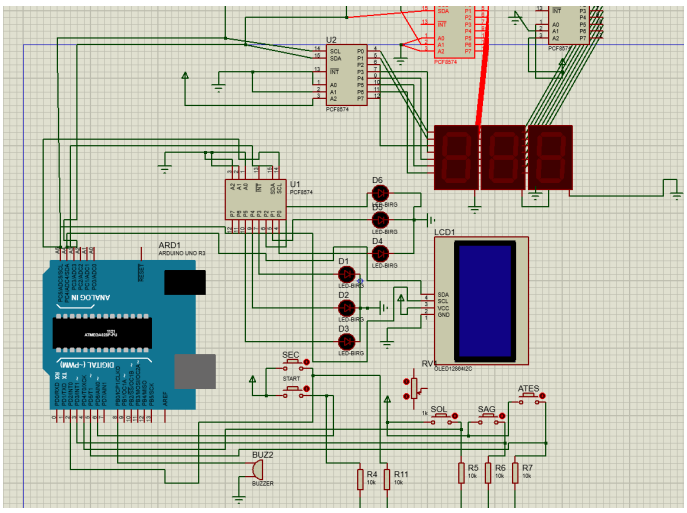
Arduino dili C++ dile benzer bir yapıya sahiptir. Arduino'da OLED ekran görselleştirmeleri için u8g2 kütüphanesi kullanılmıştır. u8g2 kütüphanesi, monokrom OLED ve LCD ekranlar için Arduino ve diğer mikrodenetleyicilerde kullanılan bir grafik kütüphanesidir. Kütüphanede OLED ekran olarak 128x64'lük ve SSD1306 kontörölöre sahip bir ekran türü seçilmiştir. U8g2 kütüphanesinin draw, font vb. yapıları kullanılmıştır. Ekranın yönlendirilmesi, kontrast ayarı ve diğer özellikleri gibi ekran ayarları da belirlenen sabitler ile kontrol edilmiştir.

İsterlerin gerçekleştirilebilmesi için sensör, buton, çoklayıcı, buzzer, LED, 7 segment display kullanımı gerekmektedir. Bu elemanların hepsinin arduinonun mevcut pinlerine sığdırılması mümkün değildir. Bu nedenle I2C çoklayıcı olarak PFC 8574 kullanılmıştır.

Bu yapı sayesinde SDA, SCL üzerinden bir çoklama yapılarak tüm ekipmanların eklenmesi sağlanmıştır.



Çok pin tutan 7-segment displayler ve ledler bu çoklayıcılar üzerinden bağlanarak devreye tüm elemanların eklenmesi sağlanmıştır.



Devrenin proje için istenilen ekipmanlar ile birlikte görünümü yukarıdadır. Birden fazla çoklayıcı bulunduğundan bu çoklayıcıların adresleme ayarları ve bağlantı yapıları yapılmıştır.

Sol tarafta görülen ledler can ve silah hakkını göstermektedir. Oyun başlamadan önce oyun ekranından seçim yapmak ve başlatmak için sol tarafta görülen pinler kullanılır. Oyun sırasında ateş etme ve sağ sol hareketi için ise OLED ekranın aşağısında görünen butonlar ve potansiyometre vardır. Oyuncunun engellere çarpması durumunda buzzer çalarak can hakkı kaybettiği anlaşılır. Çarpışmadan sonra 3 saniye boyunca can kaybetmez ve oyun akışı devam eder.

Oyun zorluğu oyun başında seçilerek bu oyun moduna uygun oyun akışı gerçekleşir.

3 Program Mimarisi

Program akışında setup işlemleri, step işlemleri ve draw işlemleri olarak 3 ayrılan bir akış vardır. Ancak temelde oyunun her aşamasını ifade eden ve Step içerisindeki ilerleme ile gidişi sağlayan switch case yapısına dayalı bir akış vardır. Bu stateler:

define ST_STATE_PREPARE 0
define ST$_STATE_IPREPARE1$
$define ST_STATE\_GAME2$
$define ST_STATE\_END3$
$define ST_STATE\_IEND4$
$uint8\_t st_state = ST\_STATE\_PREPARE;$

Oyun ilk olarak prepare modunda başlar ve akışa göre devam eder. Temelde oyunda iki struct vardır. Bir obje yapısı ve bu obje yapısının tipine göre özelliklere sahip olan bir obje tipi yapısıdır. Her obje bu iki yapıya uygun olarak şekillenir.

Nesnelerin ateş edebilir ya da edemez, yok edilebilir ya da edilemez gibi isterlere uygun olarak şekillenmesi gerekmektedir. Tüm olayların gerçekleşebilmesi için bu olayları karşılayan tipler, bu tipleri işleyebilen fonksiyonlar ve bu fonksiyonların gerçekleştiği akışlar gerekmektedir. Bu nedenle program 4 e ayrılır. Nesneye ait özelliklerin belirlenmesi, nesnenin somut hale getirilmesi, akışın başlaması ve eylemlerin gerçekleşmesi olarak ilerler.

Diğer projelerden farklı olarak temel programlamaya daha yakın olması sebebiyle derslerde teorik olarak görülen ya da yeni öğrenilen birkaç terimden bahsedilecektir.

"Maske" (mask) kelimesi, genellikle belirli bir işlemin yapılması için kullanılan özel bir değer kümesini ifade eder. Bu değer kümesi, genellikle ikili (binary) formdadır ve belirli bir işlevi yerine getiren bitleri içerir. Örneğin, bit işlemlerinde kullanılan bir maske, belirli bir biti etkinleştirmek veya devre dışı bırakmak için kullanılabilir. Özellikle bir nesnenin vurulması ya da bir eylemin gerçekleşmesi için koşulların sağlanması, maske işlemi ile gerçekleşir. Örneğin, bir nesnenin "missile_mask" özelliği nesnenin bir füze olup olmadığını belirleyen bir maske olabilir.

define ST_PKSEL 4 yapısı pozisyon bildiren değişkenlerin uygun piksel yapısına çevrilmesi ile gerçekleşir. Burada bit kaydırma işlemi gerçekleşir. Kod, sağdaki bit öteleme operatörü (>>) kullanarak o'nun y koordinatını ST$_F P'yekaydırırvesonucust_px_ydeğiskenineatar.Benzerse$

Bu kod, sabit noktalı aritmetiği kullanır. Sabit noktalı aritmetik, değişkenlerin tam sayı olarak saklanması ve hesaplamaların kayan noktalı sayılar yerine tam sayılarla yapılmasıdır. ST_PKSEL 4, kaydırma işlemi için kullanılan sabit noktalı sayıdır. ST_PKSEL 4, kaydırma işlemi sonrasında koordinatların kesirli kısımlarını temsil eden bit sayısını belirtir. Bu sayı ne kadar büyük olursa, kaydırma işlemi sonrası sayısal hassasiyet de o kadar artar.

Örneğin, eğer ST_PKSEL 4 16 ise, o'nun y koordinatı 16 bit sağa kaydırılır, bu da y koordinatının 2^16 (65536) ile bölündüğü anlamına

gelir. Bu, y koordinatının 16 bitlik kesirli kısmının kaybedilmesine neden olur. Sonuç olarak, o'nun y koordinatı, tamsayı olarak st_px_y değişkeninde saklanır.

Oyundaki şekillerin bir bitmap yapısıyla çizilmesi gerekmektedir. Bitmap (Bit Eşlemi Haritası), bir görüntü veya grafik nesnesinin piksel tabanlı bir gösterimini sağlayan bir veri yapısıdır. Piksel tabanlı bir görüntü, küçük noktalar veya piksellerin bir matrisi olarak temsil edilir. Örnek olarak bir meteorun bitmap yapısı görülmektedir.

const uint8_t st_bitmap_meteor[] =

/* 00111000 */ 0x038,
/* 01111100 */ 0x07c,
/* 10111110 */ 0x0be,
/* 11111111 */ 0x0ff,
/* 11111111 */ 0x0ff,
/* 10111110 */ 0x0be,
/* 01111100 */ 0x07c,
/* 00111000 */ 0x038,
;

loop döngüsündeki çekirdek kısım şekildeki gibidir. Loop döngüsünün içerisindeki diğer değişkenler; 7 segment display, ledler, buzzer, sayaç gibi diğer elemanların çalışmasını sağlayacak kodları bulundurur.

$st_Setup(u8g2.getU8g2());$
$for(;;)$
$st_Step(y, 0, digitalRead(pin_fire));$
$u8g2.firstPage();$

$if(digitalRead(pin_sec))$
$sec_cnt++;$
$sec_cnt = sec_cntbreak;$

do
$st_Draw(0, sec_cnt, digitalRead(pin_start));$
while(u8g2.nextPage());
case $ST_DESTROY_PLAYER$:
if (lastCollisionTime == 0 —— millis() - lastCollisionTime ¿ collisionDelay)

// Zamanlayıcı sıfır veya belirtilen süreden daha uzun bir süre geçtiyse
$st_can_hak--;$
$lastCollisionTime = millis();$
$digitalWrite(buzzerPin, HIGH);$
$buzzerStartTime = millis();$

if(st_can_hak == 0)$st_Disappear(objnr); st_State = ST_STATE_END; o- > tmp =$

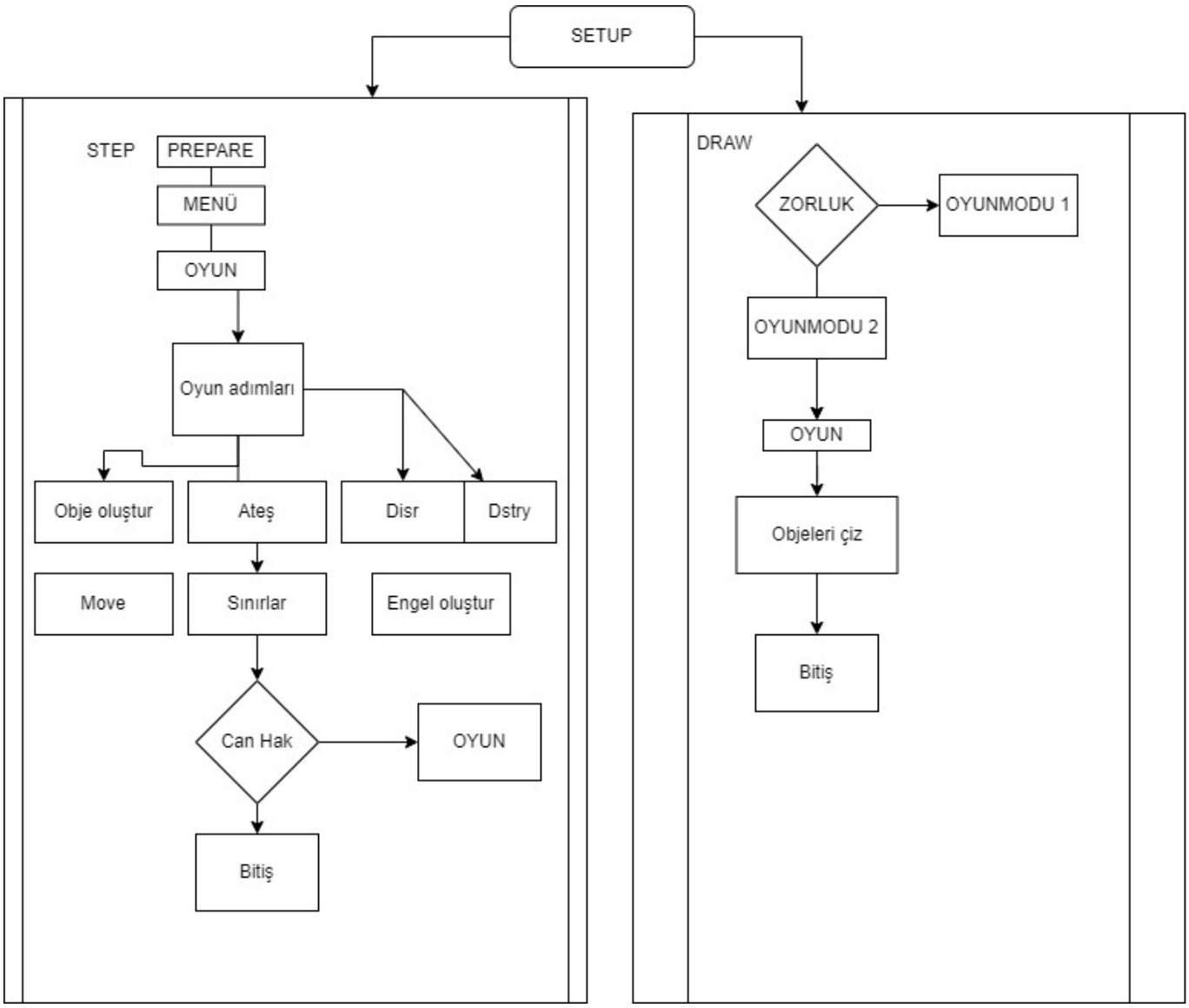
$break;$

Oyundaki can hakkının bitmesi ile oyun sonlanır. Yukarıdaki kodda bir eylemin gerçekleşmesi içerisinden 3 farklı özelliğin gerçekleşmesi görülmüştür. Bu nedenle kodun anlaşılmasını sağlayacak önemli bir özelliktir.

4 Referanslar

1) github.com (05.04.2023)
2)https://cplusplus.com/(12.04.2023)
3)https://www.arduino.cc/en/software(10.04.2023)
4)https://www.labcenter.com/(08.04.2023)
5)https://www.w3schools.com/c/$c_pointers.php$(07.04.2023)

5 Akış


SETUP
STEP
PREPARE
MENÜ
OYUN
Oyun adımları
Obje oluştur
Ateş
Disr
Dstry
Move
Sınırlar
Engel oluştur
Can Hak
OYUN
Bitiş
DRAW
ZORLUK
OYUNMODU 1
OYUNMODU 2
OYUN
Objeleri çiz
Bitiş