Univerzitet u Sarajevu
Elektrotehnički fakultet Sarajevo
Odsjek za računarstvo i informatiku

Dokumentacija implementacije – Snake Game

Ugradbeni sistemi

Ime i prezime: **Din Švraka 18857**
Ismar Višća 18912
Grupa: **četiri (4)**
Datum: **10.06.2022.**

Najbolji način da vas provedemo kroz naš implementacijski postupak i strukturu našeg projekta jeste da krenemo od najjednostavnijeg dijela, a to je dio koda namijenjen za izgled ekrana.

```
888     char start[20];
889     snprintf(start, 200, "START");
890     BSP_LCD_SetTextColor(LCD_COLOR_WHITE);
891     BSP_LCD_SetBackColor(LCD_COLOR_BLUE);
892     BSP_LCD_SetFont(&Font20);
893     BSP_LCD_DisplayStringAt(70, 82, (uint8_t *)start, LEFT_MODE);
894     char snake[20];
895     snprintf(snake, 200, "SNAKE GAME");
896     BSP_LCD_SetTextColor(LCD_COLOR_BLACK);
897     BSP_LCD_SetBackColor(LCD_COLOR_GREEN);
898     BSP_LCD_SetFont(&Font20);
899     BSP_LCD_DisplayStringAt(40, 60, (uint8_t *)snake, LEFT_MODE);
900
```

Ovo je samo mali dio izgleda početnog ekrana. Korištene su osnovne mbed funkcije za ispisivanje na ekran.

Veoma bitan dio našeg koda jesu funkcije namijenjene tasterima koje nam omogućuju da odaberemo neku opciju ili ono najbitnije, a to je kretanje zmije na ekranu.

Primjer jedne od tih funkcija jeste ako je funkcija namijenjena za taster “lijevo” u njoj će se varijabla “li”, koja predstavlja da li se zmija kreće lijevo, postaviti na true, a ostale na false.

```
42  void funLijevo() {
43    if (jeLiPocelo) {
44      li = true;
45      pocetak = false;
46      g = false;
47      d = false;
48      de = false;
49    }
50  }
```

Ista logika je korištena i u funkcijama za ostale tastere.

Prelazimo na dio koda namijenjen za poziciju miša na ekranu. Pozicija miša na ekranu je randomizirana, a to smo implementirali tako što smo napravili dva niza sa 27 pozicija (od 0 do 26) koji sadrže sve moguće x i y koordinate pojavljivanja miša na ekranu. Randomiziranu poziciju smo određivali tako što funkcijom rand() dobijemo randomiziran broj i onda uzmemo ostatak dijeljenja tog broja sa brojem 26 i to nam određuje poziciju u nizu od 0 do

26. Na početku i svaki put kada zmija pojede jednog miša, na ekranu se pojavi novi miš na randomiziranoj poziciji. Miš je predstavljen preko bijelog kruga.

```
976     int x[] = {18,  26,  34,  42,  50,  58,  66,  74,  82,
977                90,  98,  106, 114, 122, 130, 138, 146, 154,
978                162, 170, 178, 186, 194, 202, 210, 218, 226};
979     int y[] = {230, 222, 214, 206, 198, 190, 182, 174, 166,
980                158, 150, 142, 134, 126, 118, 110, 102, 94,
981                86,  78,  70,  62,  54,  46,  38,  30,  22};
982     int randomx, randomy;
983     randomx = rand() % 26;
984     randomy = rand() % 26;
985     BSP_LCD_SetTextColor(LCD_COLOR_WHITE);
986     BSP_LCD_FillCircle(x[randomx], y[randomy], 4);
```

Sada imamo implementaciju kretanja zmije na ekranu i implementaciju repa zmije. Trenutni smjer kretanja zmije zavisi od toga koji je taster posljednji pritisnut. Zmiju "prati" crni krug koji briše posljednji dio zmije. Pod dijelom zmije podrazumijevamo glavu zmije ukoliko nema rep (crveni krug), a ukoliko ima rep onda posljednji zeleni krug. Što se tiče same implementacije repa, to smo uradili tako što dužina repa zavisi od broja pojedenih miševa, odnosno broja bodova u određenom trenutku. Pomoću dva vektora smo pamtili posljednje pozicije glave zmije. Jedan vektor pamti x koordinate, a drugi y koordinate. Pomoću ovih vektora smo znali na kojim pozicijama da nacrtamo rep na ekranu. Ukoliko je, naprimjer, broj bodova 5 onda će glavu zmije pratiti 5 zelenih krugova.

U nastavku prvo imamo dio koda koji se izvršava ako se u tom trenutku zmija kreće ulijevo.

```
//Kretanje zmije lijevo
if (li) {
  BSP_LCD_SetTextColor(LCD_COLOR_BLACK);
  BSP_LCD_FillCircle(x1, y1, 4);
  BSP_LCD_SetTextColor(LCD_COLOR_RED);
  if (x2 == 18)
    x1 = 226;
  x2 = x1 - 8;
  y2 = y1;
  BSP_LCD_FillCircle(x2, y2, 4);
  x1 = x2;
  y1 = y2;
  vX.push_back(x2);
  vY.push_back(y2);
}
```

Ista logika je korištena i za druge smjerove.

Sada imamo dio koda koji se izvršava ukoliko zmija ima rep, odnosno broj bodova je jedan ili više.

```
//Rep zmije
if (vX.size() > 1) {
  int temp1, temp2, temp5;
  if (vX.size() - 1 - points < 0) {
    temp5 = 0;
  } else {
    temp5 = vX.size() - 1 - points;
  }
  for (int i = vX.size() - 2; i >= temp5; i--) {
    if (i == -1) {
      break;
    }
    BSP_LCD_SetTextColor(LCD_COLOR_GREEN);
    BSP_LCD_FillCircle(vX.at(i), vY.at(i), 4);
    temp1 = vX.at(i);
    temp2 = vY.at(i);

    //Zmija je ujela samu sebe
    if (x2 == vX.at(i) && y2 == vY.at(i)) {}
  }
  BSP_LCD_SetTextColor(LCD_COLOR_BLACK);
  BSP_LCD_FillCircle(temp1, temp2, 4);
}

//Promjena brzine kretanja zmije na levelu 2
if (temp == 2 && points == 11) {
  T = 0.25;
}
```

Prethodni dio koda nam može poslužiti da vidimo kako smo implementirali trenutak kada zmija ujede samu sebe gdje varijable x2 i y2 predstavljaju narednu poziciju glave i u petlji se porede sa svim pozicijama gdje je rep.

Sada ćemo proći i kroz najbitnije funkcije u projektu.

- level1Complete() – funkcija se poziva kada se završi level 1, postavlja izgled ekrana i omogućuje izbor za prelazak na level 2 ili restart
- level2Complete() – funkcija se poziva kada se završi level 2, postavlja izgled ekrana i omogućuje izbor za prelazak na level 3 ili restart
- level3Complete() – funkcija se poziva kada se završi level 3,
  postavlja izgled ekrana i omogućuje izbor pokretanja nove igre
- fail() - funkcija se poziva kada zmija ujede samu sebe, postavlja izgled ekrana i omogućuje izbor pokretanja nove igre

Sada kada smo prošli detaljno kroz implementaciju našeg projekta, kada povežemo sve što smo naveli dobijamo u potpunosti funkcionalnu aplikaciju “**SnakeGame**”.