Osnove mikroprocesorske elektronike

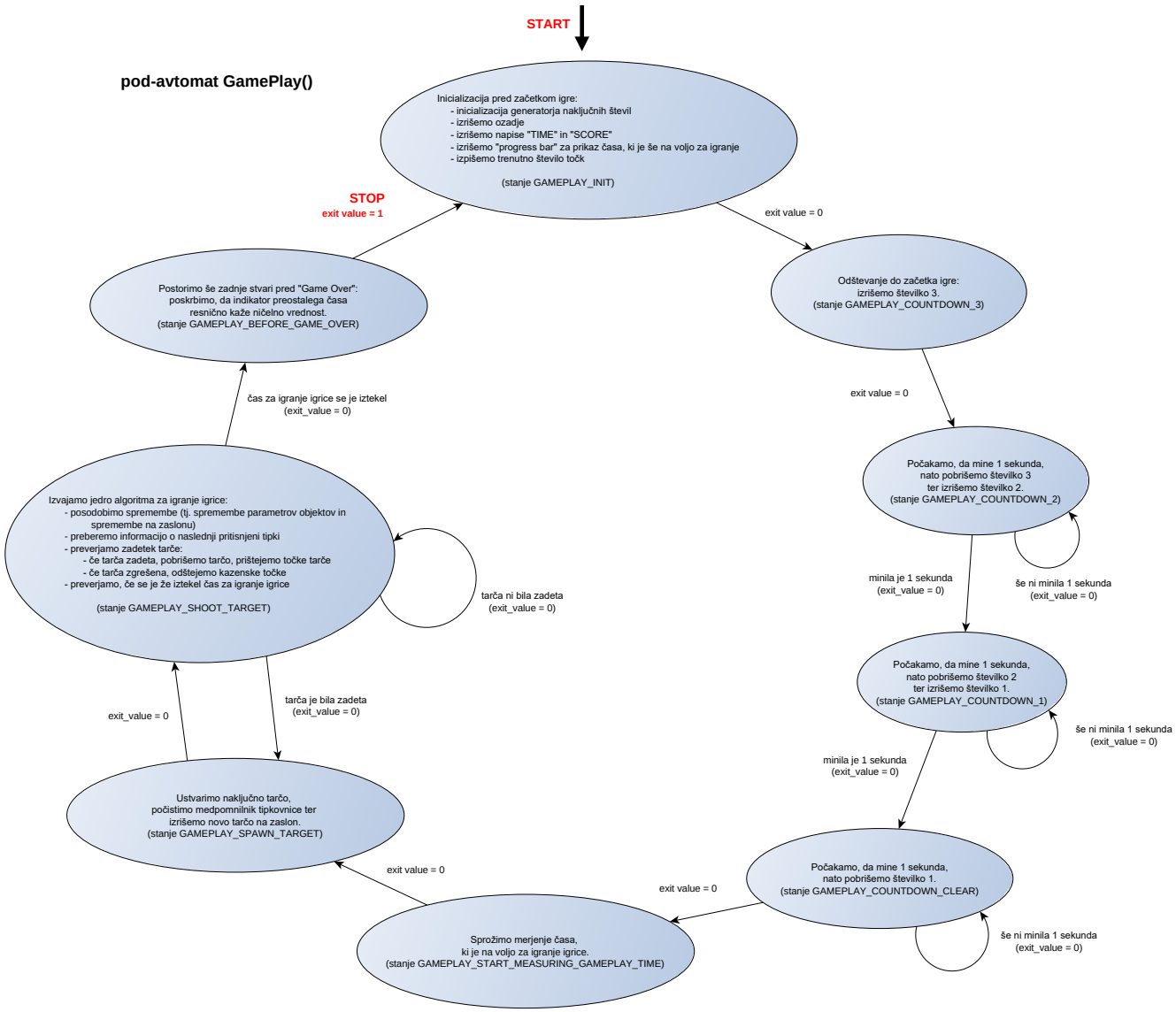
Vaja 14: Igrica - drugič

Pri tej vaji boste *dokončali implementacijo aplikacije* – igrice. Dopolniti bo potrebno še zadnji, ključni del algoritma igrice, kjer se implementira "obnašanje in pravila igranja igrice" ter hkrati poskrbi za izris vsega "dogajanja v igrici" na grafični vmesnik.

Naloge vaje

1. ***S pomočjo naštevnega tipa definirajte stanja pod-avtomata*** `GamePlay()` ***ter poimenujte ta naštevni tip*** `GAMEPLAY_states_t`.

   Pri tem si pomagajte s spodnjim diagramom prehajanja stanj.



Namig: kadar naštevni tip vsebuje mnogo elementov, lahko *uporabite prelom vrstice* in vsak element pišete v svojo vrstico. Vsebina naštevnega tipa tako postane bolj *pregledna*. Poglejte primer spodaj.

```
typedef enum  { ELEMENT_1,
                ELEMENT_2,
                ELEMENT_3,
                ELEMENT_4,
                ELEMENT_5,
                ELEMENT_6,
                ELEMENT_7,
                ELEMENT_8,
                ELEMENT_9
} ENUMERATED_ELEMENTS_t;
```

2. ***Preden se lotite programiranja algoritma igre, poskrbite še za inicializacijo nekaterih ključnih parametrov igre:***

   a) določite nastavitve igre:

      i. minimalna absolutna hitrost vzdolž koordinatne osi = 1,
      ii. maksimalna absolutna hitrost vzdolž koordinatne osi = 4,
      iii. čas, ki je na voljo za igranje igrice = 20 sekund,
      iv. kazenske točke, ki se odštejejo ob zgrešenem strelu = 50.

      Nastavitve igre so shranjene v podatkovni strukturi – objektu `settings`.

   b) Nastavite začetno vrednost doseženih točk igralca na nič.

      Vrednost doseženih točk igralca hranimo v objektu `game_status`.

   c) Nastavite število točk, ki jih prinese sestrelitev posameznega arduinota.

      Arduinoti naj bodo zaenkrat točkovani enakovredno in naj prinesejo 100 točk ob sestrelitvi.

      Vrednost točk, ki jih prinese sestrelitev posameznega arduinota hranimo v objektih tipa `arduino_object_t`, ki pa so shranjeni v tabeli `arduino[]` in služijo kot predloga ("template") za generiranje naključne tarče.

Ne pozabite, da začetne vrednosti parametrov objektov določamo znotraj `OBJ_init_` funkcij, ki so kot nekakšne "*konstruktor funkcije*" *objektov*, saj pomagajo "*ustvariti* objekte".

3. *Dopolnite implementacijo funkcije* `GamePlay()`.

Funkcija `GamePlay()` je sestavljena iz treh delov:

a) *definicij* (npr. spremenljivke, makro parametri ipd.);

b) pomožne *ugnezdene funkcije* (angl. nested function) `GamePlay_UpdateChanges()`, s katero bomo poskrbeli, da se objekti, ki jih algoritem igre spreminja, *posodobijo*: tako njihovi parametri kot njihov prikaz na zaslonu.

Pri implementaciji te pomožne funkcije vas bodo vodili komentarji v predlogi.

c) *avtomata stanj*, ki dejansko implementira algoritem za igranje igrice.

Pri implementaciji avtomata stanj se zgledujte po že izvedenih implementacijah avtomatov za opravila `Intro()` in `GameOver()`.

Pri implementaciji algoritma uporabljajte *funkcionalnost sistemskih modulov* (npr. TIMUT modula, KBD modula ipd.) ter *funkcionalnost aplikacijskih modulov*.

Pri uporabi funkcij aplikacijskih modulov si pomagajte z že znanim seznamom aplikacijskih funkcij:

Funkcije modula `GFX`

`GFX_draw_gfx_object()`	funkcija izriše grafični objekt brez upoštevanja plasti in transparentnega dela grafičnega objekta
`GFX_draw_one_gfx_object_on_background()`	funkcija izriše en sam grafični objekt na ozadje, upoštevajoč transparentni del grafičnega objekta
`GFX_draw_two_gfx_objects_on_background()`	funkcija izriše dva grafična objekta na ozadje, upoštevajoč plast (angl. layer) na katerem se nahaja grafični objekt
`GFX_clear_gfx_object_on_background()`	funkcija pobriše en sam grafični objekt na ozadju
`GFX_update_moving_gfx_object_location()`	funkcija posodobi lokacijo premikajočega grafičnega objekta, upoštevajoč hitrost objekta
`GFX_display_text_object()`	funkcija izriše tekstovni objekt (npr. tekst, ki vsebuje število doseženih točk)
`GFX_display_progress_bar()`	funkcija izriše "progress bar" indikator

Funkcije modula `OBJ`

`OBJ_spawn_target()`	funkcija ustvari naključno tarčo: naključno izbere vrsto tarče, njeno pozicijo in hitrost
`OBJ_crosshair_set_center_location_with_joystick()`	funkcija nastavi lokacijo namerka upoštevajoč pozicijo "joysticka"
`OBJ_crosshair_update_distance_to_target()`	funkcija izračuna razdaljo med namerkom in tarčo in shrani to informacijo med parametre namerka
`OBJ_is_crosshair_on_target()`	funkcija preveri, če se namerek nahaja znotraj tarče; funkcija vrne vrednost 1, če namerek v tarči, sicer pa vrednost 0
`OBJ_set_timeout_bar_value()`	funkcija nastavi vrednost "progress bar" indikatorja; vhodni argument je vrednost podana v procentih
`OBJ_set_score_text_value()`	funkcija nastavi vrednost besedila za izpis točk; vhodni argument je vrednost točk

Funkcije modula `MATH`

`MATH_init_random_generator()`	funkcija inicializira generator naključnih števil (tj. nastavi "seme" (angl. seed) za generacijo naključnih števil)

Funkcije modula `Game`

`GamePlay_Update_Aiming_LED_Indicator()`	funkcija posodobi stanje LED indikatorja točnostni namerka: bližje kot je namerek tarči, več LEDic sveti
`GamePlay_UpdateChanges()`	funkcija poskrbi, da se objekti, ki jih algoritem igre spreminja, posodobijo: tako njihovi parametri kot njihov prikaz na zaslonu

4. ***Stestirajte implementacijo igrice tako, da preverite, ali je obnašanje igre res tako, kot smo si ga zamislili:***

   ko po uvodnem "splash screen" zaslonu igralec pritisne katerokoli tipko, se nato izriše prostor (slika), kjer bo igralec streljal tarče. V spodnjem statusnem delu zaslona lahko igralec vidi indikator preostalega časa in število točk, ki jih je dosegel. Na zaslonu se odšteva čas do začetka igranja: 3 – 2 – 1, nato se igra prične. Na zaslonu se pojavljajo naključne tarče, ki se odbijajo od roba prostora. Igralec jih mora sestreliti tako, da z "joystickom" postavi namerek na tarčo ter s pritiskom na katerokoli tipko ustreli. Če je bil strel uspešen, tarča izgine in se pojavi nova. Če je strel zgrešil, se igralcu odšteje točke. Zgrešene strele se torej kaznuje. V spodnjem statusnem delu zaslona se sproti izrisuje čas, ki je še ne voljo za igranje ter število doseženih točk. Ko mine 20 sekund, se igra zaključi in pridemo do "game over" zaslona.

5. ***Čestitke ob uspešni implementaciji! ☺***

   Za konec le še nekaj zaključnih misli, ki jih lahko odnesete z zadnjih vaj:

   a) aplikacijo je smiselno razdeliti (vsaj) na tri sestavne dele:
      i. *podatkovne strukture* aplikacije,
      ii. *algoritem* aplikacije,
      iii. *izris na grafični* vmesnik.

   b) če si smiselno in *sistematično pripravite "osnovne gradnike aplikacije"*, lahko nato s temi gradniki razmeroma enostavno *sestavite kompleksnejšo aplikacijo* (gradniki so tako podatkovne strukture kot funkcije, ki izvajajo operacije nad temi podatkovnimi strukturami)

   c) avtomat stanj je močno orodje, ki vam lahko pride prav vedno, kadar imate opravka pri *realizaciji procesov*, kjer je *postopek poteka procesa točno določen in znan*

   d) ključna navodila za uporabo avtomatov najdete na vseh pomembnejših napravah v izobraževalnih ustanovah: