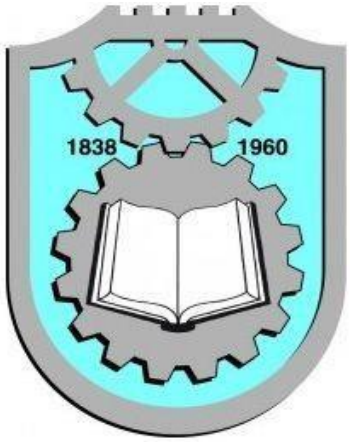
UNIVERZITET U KRAGUJEVCU
FAKULTET INŽENJERSKIH NAUKA



Računarska grafika

Dokumentacija za projektni zadatak

Arkadna igra GORF

Student:
Nikola Džajević 615/2017

Profesori:
Nenad Filipović
Tijana Šušteršič

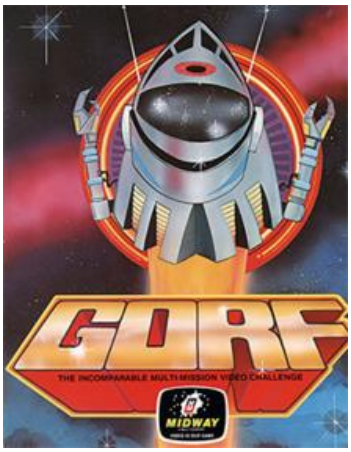
Kragujevac, 2020/21.

Sadržaj

1. Uvod

Tema projektnog zadatka je kreiranje aplikacije igrice *Gorf* koristeći *OpenGL*. Gorf je arkadna igra koju je 1981. izdao Midvai Mfg., čije je ime oglašeno kao skraćenica za "Galactic Orbiting Robot Force". Gorf je fiksna pucačina u kojoj igrači preuzimaju kontrolu nad neimenovanim svemirskim brodom iz Međuzvezdanih svemirskih snaga, sposobnim da se slobodno kreću u svim pravcima oko donje trećine ekrana, u pokušaju da spreče Gorfijsko carstvo da osvoji Zemlju. Igra se sastoji od pet različitih misija; svaka misija predstavlja svoj poseban stil igre, ali centralni cilj svake je uništenje svih neprijatelja. Uspešno obavljenih svih pet misija povećava rang igrača, koji predstavlja trenutni nivo težine igre, i vraća se na prvu misiju. Igra se nastavlja sve dok igrač ne izgubi čitav život. [1]

Gorf je prvobitno trebao biti povezan sa Star Trek-om: The Motion Picture, ali nakon što su pročitali scenario filma, dizajneri igara su shvatili da koncept neće raditi kao video igra; međutim, brod igrača i dalje podseća na Starship Enterprise. Osnovna hardverska platforma za Gorf omogućila je arkadnim operaterima da lako zamene šablon, CPU i RAM ploče sa drugim sličnim igrama, kao što je Vizard of Vor, budući da su samo logika igre i ROM ploče specifične za svaku igru.



Gorf je objavljen u Severnoj Americi u februaru 1981. , i postao jedan od prvih naslova koji su koristili sintetizovani govor (opremljen čipom za govor Votrak), kao i jedna od prvih igara sa više ekrana. Arkadna verzija je dosegla vrh arkadnih top lista američkih Play Metera u septembru 1981. Verzija igre Atari 2600 dobila je Sertifikat o zaslugama u kategoriji "Najbolja video igra" na 4. godišnjoj dodeli nagrada Arkie, dobila je nagradu "Najbolji audio-vizuelni efekti za računarske igre" 1984. Arkija sledeće godine Na 5. Arkiju sudije su istakle da su Atari verzije nadmašile i verzije igre ColecoVision i Commodore 64, i predložile su da se „raznovrsna akcija“ igre „stalno vraća igračima“.

Što se tiče verzije VIC-20, Electronic Games je napisao da je "ovaj živopisni šareni unos neophodan ... jedna od najboljih igara dostupnih za VIC-20" i izjavio je da verzija VIC-20 "i dalje ima moj glas za najbolje iz gomile ... Grafika je odlična".

Gorf je uključen u knjigu 1001 Video igre iz 2010. koju morate igrati pre nego što umrete.

2. Realizacija

Aplikacija je kreirana pomoću *Code Blocks-a* kao i programskog jezika *C++* korišćenjem *OpenGL* biblioteke u operativnom sisitemu *Linux*. U ovom delu biće detaljno opisani delovi koda kao i prikaz izvornog koda.

Odmah na početku je potrebno definisati sve potrebno biblioteke i dodatke koji se koriste tokom izrade aplikacije.

```
1   #include <math.h>
2   #include <stdio.h>
3   #include <stdlib.h>
4   #include <GL/glut.h>
5   #include <vector>
6   #include <time.h>
7   #include <string>
```

Pored standardnih biblioteka, tu se nalazi i biblioteka "*glut.h*". Glut odnosno *OpenGL Utility Toolkit* je biblioteka koja pruža usluge za korišćenje *OpenGL* programa koji uglavnom izvršava sistemski nivo ulazno-izlaznih poziva sa operativnim sistemom. Među tim funkcijama se nalaze manipulacije prozora programa, kontrola prozora kao i nadgledanje ulaznih parametara sa tastature i miša. Sve GLUT funkcije počinju sa prefiksom *glut*. [2]

Potom je potrebno definisati promenljive i strukture koje se koriste unutar programa.

```
9   #define PI 3.14159
10
11  // Definisanje globalne promenljive
12  GLfloat hunter_x = 0;
13  // Definisanje tipa tacka
14  typedef struct Point
15  {
16      float x, y;
17      Point(float _x, float _y) : x(_x), y(_y) {}
18  } Point;
19  // Definisanje linije odbrane
20  struct defence
21  {
22      GLfloat g, x, y;
23      defence() {}
24      defence(GLfloat _x, GLfloat _y, GLfloat _g) : x(_x), y(_y), g(_g) {}
25  };
26  // Definisanje neprijatelja (Invejdera)
27  struct enemy
28  {
29      GLfloat x, y;
30      int type;
31      enemy() {}
32      enemy(GLfloat _x, GLfloat _y, int _type) : x(_x), y(_y), type(_type) {}
33  };
34  // Definisanje rakete
35  struct Missle
36  {
37      float x, y;
38      bool moving;
39  };
```

Svaka od promenljivih ima svoju ulogu i namenu tokom izvršavanja samog programa – aplikacije.

Strukture koje se definišu pomažu uz kasnijoj inicijalizaciji figura, to jest, olakšavaju crtanje grafičkih primitiva. Definisana je struktura Point koja predstavlja tačku u prostoru, yatim struktura Defence koja predstavlja koordinate i boju grafičkih primitiva na ekranu, kao i struktura Enemy koja definiše poziciju napadača na ekranu. Struktura Missle definiše koordinate i promenjivu pomeraja za raketu koju ispaljuje igrač.

```
41   bool startInitializationFlag = false; // fleg za pocetak
42   bool gameOver = false; // fleg za kraj igre
43   bool flash = false; // fleg za fles ekrana
44   bool forward = true;
45   int score = 0;
46   int lives = 3;
47   // Definisanje vektora
48   std::vector<struct defence> block;
49   std::vector<struct enemy> invaders;
50   std::vector<struct Missle> Missle;
51   std::vector<struct enemy> hit;
52   std::vector<Point> explosion;
```

Definisanje flegova, početnih vrednosti i vektora. Kao početne vrednosti, igrač počinje igru sa rezultatom 0 koji se kasnije pogađanjem neprijatelja povećava, početni broj života je 3. Flegovi za inicijalizaciju, kraj igre, fles ekrana pri pogotku, i pomeranja predstavljaju jako bitnu stavku u kasnijoj realizaciji aplikacije.

```
53   // Definisanje pocetnih pozicija odbrane, invejdera i lovca
54   void startInitialization()
55   {
56       struct defence temp;
57       struct Missle temp2;
58       srand(time(NULL));
59       for (int i = 0, k = -24; i < 24; i++, k += 2)
60       {
61           temp = defence(k, 2, 1.0f);
62
63           block.push_back(temp);
64           temp.y = 2;
65           block.push_back(temp);
66           temp.x = k + 1;
67           block.push_back(temp);
68       }
69       for (int i = 0; i < 5; i++)
70       {
71           for (float j = 0, k = -13; j < 10; j++, k += 3)
72           {
73
74               invaders.push_back(enemy(k, 25.5 - i * 2.5, i + 1));
75           }
76       }
77       for (int i = 0; i < 2; i++)
78       {
79
80           temp2.moving = false;
81           temp2.y = -1;
82           temp2.x = hunter_x - 1 + 2 * i;
83           Missle.push_back(temp2);
84       }
85   }
```

Početna inicijalizacija definisana je funkcijom startInitialization, čiji je zadatak da formira početnu scenu, to jest, da pozicionira invejdere, lovca i liniju odbrane. Linija obbrane jeste linija formirana od zbijenih kvadrata, da bi se kvadrati uništili potrebno je da invejder ili lovac svojim raketama pogode blok. Svaki prvi blok ima snagu od 4 pogotka, a svaki drugi od 8 pogodaka, što znači, da svaki je svaki blok potrebno uništiti kako bi rakete došle od lovca do invejdera i obrnuto. [3]

```
86  // Pomeranje invejdera
87  void moveInvader(bool *forward)
88  {
89      bool overflow = false; //definisanje flega za overflow
90      float step = 0.01;
91
92      for (int i = 0; i < invaders.size(); i++)
93      {
94          if (invaders[i].y < 2)
95          {
96              //ako su napadaci preblizu, igra se zavrsava
97              gameOver = true;
98              return;
99          }
100         if (invaders[i].x + step > 18 && *forward == true)
101         {
102             overflow = true;
103             break;
104         }
105         if (invaders[i].x - step < -20 && *forward == false)
106         {
107             overflow = true;
108             break;
109         }
110     }
111     if (overflow)
112     {
113         *forward = !*forward;
114         for (int i = 0; i < invaders.size(); i++)
115         {
116             invaders[i].y -= 0.5;
117         }
118     }
119     step = (*forward == true) ? step : -step;
120
121     for (int i = 0; i < invaders.size(); i++)
122     {
123         invaders[i].x += step;
124     }
125 }
```

Funkcija moveInvader definiše pomeranje invejdera, kao i računanje da li su invejderi na optimalnom rastojanju od lovca i linije odbrane. Invejderi se pomeraju koracima, koraci su definisani promenljivom step, koja iznosi 0.01.

```
126  // Definisanje ispisivanja teksta na ekranu
127  void drawText(const char *text, int length, float x, float y, float size, float r, float g, float b)
128  {
129      glPushMatrix();
130      glColor3f(r, g, b);
131      glTranslatef(x, y, 0.0f);
132      glScalef(size, size, 1.0f);
133
134      for (int i = 0; i < length; i++)
135      {
136          glutStrokeCharacter(GLUT_STROKE_ROMAN, (int)text[i]);
137      }
138      glPopMatrix();
139  }
```

Funkcija drawText kao parametre zahteva tekst, dužinu teksta, koordinate teksta, kao i vrednosti r, g i b koje služe za bojenje unetog teksta. Funkcija se kasnije u kodu poziva radi lakšeg ispisivanja teksta na ekranu.

```
140  // Pomeranje neprijateljskih raketa
141  void moveHit()
142  {
143      for (int i = 0; i < hit.size(); i++)
144      {
145          hit[i].y -= 0.1; // brzina raketa invejdera
146      }
147  }
148  // Pomeranje raketa lovca
149  void moveMissle()
150  {
151      for (int i = 0; i < Missle.size(); i++)
152      {
153          if (Missle[i].moving == true)
154          {
155              Missle[i].y += 0.5; // brzina rakete lovca
156          }
157      }
158  }
```

Funkcija moveHit definiše pomeranje rakete invejdera, kao i njenu brzinu. U aplikaciji brzine kretanja raketa invejdera i lovca nisu iste. Sa druge strane, funkcija moveMissle definiše kretanje i brzinu kretanja rakete lovca prema invejderu.

```
159  // Crtanje lovca
160  void Hunter()
161  {
162      glPushMatrix();
163      glTranslatef(hunter_x, 0, 0);
164      glColor3f(0.5f, 0.6f, 0.86f);
165      glLineWidth(2);
166      glBegin(GL_TRIANGLES);
167      glVertex2f(-1.0f, -1.0f);
168      glVertex2f(1.0f, -1.0f);
169      glVertex2f(0.0f, 0.0f);
170      glVertex2f(0.0f, 0.5f);
171      glVertex2f(0.5f, -0.1f);
172      glVertex2f(-0.5f, -0.1f);
173      glVertex2f(0.0f, 1.0f);
174      glVertex2f(0.2f, 0.5f);
175      glVertex2f(-0.2f, 0.5f);
176      glVertex2f(0.0f, -1.4f);
177      glVertex2f(0.2f, -1.0f);
178      glVertex2f(-0.2f, -1.0f);
179      glColor3f(1.0f, 0.0f, 0.0f);
180      glVertex2f(0.1f, -1.5f);
181      glVertex2f(-0.1f, -1.5f);
182      glColor3f(1.0f, 1.0f, 0.0f);
183      glVertex2f(0.0f, -1.9f);
184      glEnd();
185
186      glColor3f(0.5f, 0.6f, 0.86f);
187      glBegin(GL_QUADS);
188      glVertex2f(0.2f, -0.2f);
189      glVertex2f(0.2f, 0.5f);
190      glVertex2f(-0.2f, 0.5f);
191      glVertex2f(-0.2f, -0.2f);
192      glEnd();
193      glPopMatrix();
194  }
```

Funkcija Hunter kao rezultat vraća grafički prikaz lovca, lovac se sastoji od skupa grafičkih primitiva koji zajedno daju konačni izgled lovca. Lovac se sastoji iz vrha, sredine i krila, kao i plamena na kraju repa.

```
195  // Crtanje prvog invejdera
196  void invader_1()
197  {
198      glColor3f(0.6f, 0.1f, 0.1f);
199      glBegin(GL_QUADS);
200      glVertex2f(0, 0);
201      glVertex2f(0, 2);
202      glVertex2f(2, 2);
203      glVertex2f(2, 0);
204      glEnd();
205
206      glColor3f(0.0f, 0.5f, 1.0f);
207      glBegin(GL_TRIANGLES);
208      glVertex2f(0, 1.2);
209      glVertex2f(0, 2);
210      glVertex2f(1, 2);
211      glVertex2f(2, 1.2);
212      glVertex2f(2, 2);
213      glVertex2f(1, 2);
214      glVertex2f(0, 1);
215      glVertex2f(0, 0);
216      glVertex2f(0.8, 1);
217      glVertex2f(2, 1);
218      glVertex2f(2, 0);
219      glVertex2f(1.2, 1);
220      glVertex2f(0, 0);
221      glVertex2f(2, 0);
222      glVertex2f(1, 0.9);
223      glEnd();
224      // praznina za oci
225      glColor3f(0.0f, 0.5f, 1.0f);
226      glBegin(GL_QUADS);
227      glVertex2f(0.5, 1.3);
228      glVertex2f(0.5, 1.5);
229      glVertex2f(0.7, 1.5);
230      glVertex2f(0.7, 1.3);
231      glVertex2f(1.3, 1.3);
232      glVertex2f(1.3, 1.5);
233      glVertex2f(1.5, 1.5);
234      glVertex2f(1.5, 1.3);
235      glEnd();
236  }
```

Crtanje invejdera se vrši preko zasebnih funkcija za svaki. Za crtanje prvog invejdera se koristi funkcija invader_1. Funkcija crta invejdera kao skup grafičkih primitiva spojenih u jedan konačan izgled invejdera 1. Invejder se sastoji od tela i dva pipka, crvene je boje i ima dva pravougaonikja koji predstavljaju oči invejdera.

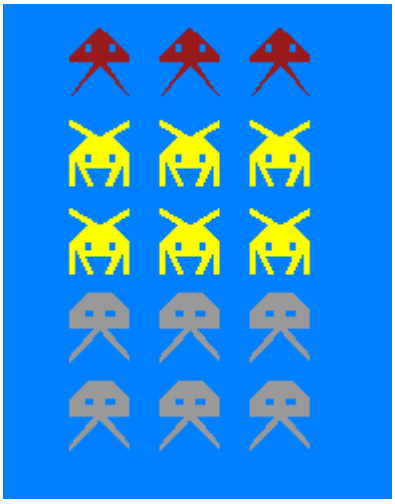
```
237  // Crtanje drugog invejdera
238  void invader_2()
239  {
240      glColor3f(1.0f, 1.0f, 0.0f);
241      glBegin(GL_QUADS);
242      glVertex2f(0, 0);
243      glVertex2f(0, 2);
244      glVertex2f(2, 2);
245      glVertex2f(2, 0);
246      glEnd();
247
248      glColor3f(0.0f, 0.5f, 1.0f);
249      glBegin(GL_TRIANGLES);
250      glVertex2f(0 - 0.2, 2);
251      glVertex2f(2.2, 2);
252      glVertex2f(1, 1.4);
253      glVertex2f(0, 0.9);
254      glVertex2f(0, 1.7);
255      glVertex2f(0.6, 1.4);
256      glVertex2f(2, 0.9);
257      glVertex2f(2, 1.7);
258      glVertex2f(1.4, 1.4);
259      glVertex2f(0.2, 0);
260      glVertex2f(0.2, 0.7);
261      glVertex2f(0.6, 0);
262      glVertex2f(1.8, 0);
263      glVertex2f(1.8, 0.7);
264      glVertex2f(1.4, 0);
265      glVertex2f(0.5, 0.5);
266      glVertex2f(1.4, 0.5);
267      glVertex2f(1, 0 - 0.3);
268      glEnd();
269      // praznina za oci
270      glBegin(GL_QUADS);
271      glVertex2f(0.5, 0.7);
272      glVertex2f(0.5, 0.9);
273      glVertex2f(0.7, 0.9);
274      glVertex2f(0.7, 0.7);
275      glVertex2f(1.3, 0.7);
276      glVertex2f(1.3, 0.9);
277      glVertex2f(1.5, 0.9);
278      glVertex2f(1.5, 0.7);
279      glEnd();
280  }
```

Crtanje drugog invejdera se vrši preko zasebne funkcije. Za crtanje drugog invejdera se koristi funkcija invader_2. Funkcija crta invejdera kao skup grafičkih primitiva spojenih u jedan konačan izgled invejdera 2. Invejder se sastoji od tela i četiri pipka, kao i dve antene, žute je boje i ima dva pravougaonikja koji predstavljaju oči.

Za crtanje trećeg invejdera se koristi funkcija invader_3. Funkcija spajanjem grafičkih primitiva daje konačnu sliku invejdera. Invejder 3 se sastoji od tela i dva pipka, kao i dva pravougaonika koji predstavljaju oči i sive je boje.

```
281  // Crtanjetreceg invejdera
282  void invader_3()
283  {
284      glColor3f(0.6f, 0.6f, 0.6f);
285      glBegin(GL_QUADS);
286      glVertex2f(0, 0);
287      glVertex2f(0, 2);
288      glVertex2f(2, 2);
289      glVertex2f(2, 0);
290      glEnd();
291
292      glColor3f(0.0f, 0.5f, 1.0f);
293      glBegin(GL_TRIANGLES);
294      glVertex2f(0, 0);
295      glVertex2f(2, 0);
296      glVertex2f(1, 0.9);
297      glVertex2f(0, 1.5);
298      glVertex2f(0, 2);
299      glVertex2f(0.6, 2);
300      glVertex2f(2, 1.5);
301      glVertex2f(2, 2);
302      glVertex2f(1.4, 2);
303      glVertex2f(0, 1);
304      glVertex2f(0, 0.2);
305      glVertex2f(0.8, 1);
306      glVertex2f(2, 1);
307      glVertex2f(2, 0.2);
308      glVertex2f(1.2, 1);
309      glEnd();
310      // praznina za oci
311      glBegin(GL_QUADS);
312      glVertex2f(0.5, 1.3);
313      glVertex2f(0.5, 1.5);
314      glVertex2f(0.8, 1.5);
315      glVertex2f(0.8, 1.3);
316      glVertex2f(1.2, 1.3);
317      glVertex2f(1.2, 1.5);
318      glVertex2f(1.5, 1.5);
319      glVertex2f(1.5, 1.3);
320      glEnd();
321  }
```

Invejderi su crtani po ugledu na originalne invejdere iz igrice Gorf.
U prilogu je izgled sve tri vrste invejdera.



Kako bi program lakše birao invejdere koji će biti prikazani na ekranu, napsiana je funkcija za biranje invejdera. Funkcija za biranje invejdera formira matricu invejdera, tako što su definisani prethodno broj redova invejdera i broj kolona, na osnovu tog broja, funkcija određuje koliko kojih invejdera će biti nacrtano na početnom ekranu.

```
322  // Pozivanje invejdera koji ce biti iscrtani na ekranu
323  void Invaders(struct enemy inv)
324  {
325      glPushMatrix();
326      glTranslatef(inv.x, inv.y, 0);
327      if (inv.type >= 4)
328      {
329          invader_3();
330      }
331      else if (inv.type >= 2)
332      {
333          invader_2();
334      }
335      else
336      {
337          invader_1();
338      }
339      glPopMatrix();
340  }
```

Funkcija za crtanje blokova linije odbrane formira pravougaoni blok, predefinisane zelene boje i predefinisanim koordinatama položaja postavlja blok.

```
341  // Crtanje linije odbrane
342  void drawBlock(struct defence block)
343  {
344      glPushMatrix();
345      glTranslatef(block.x, block.y, 0);
346      glColor3f(0.0f, block.g, 0.0f);
347      glBegin(GL_QUADS);
348      glVertex2f(0, 0);
349      glVertex2f(0, 1);
350      glVertex2f(1, 1);
351      glVertex2f(1, 0);
352      glEnd();
353      glPopMatrix();
354  }
```

Funkcija drawExplosion prima koordinate x i y i služi da simulira eksploziju pri udaru rakete u cilj. Kao simulaciju eksplozije koristi se OpenGL funkcija za crtanje TRIANGLE_FAN, svaki put kada raketa dodirne lovca, invejdera ili liniju odbrane, poziva se funkcija za simulaciju eksplozije.

```
355  // Funkcija za crtanje eksplozije i njen efekat
356  void drawExplosion(GLfloat x, GLfloat y)
357  {
358      glPushMatrix();
359      glTranslatef(x, y, 0);
360      int konstanta = 20;
361      GLfloat dvaPi = 2.0f * PI;
362      glLineWidth(1);
363      glColor3f(1.0f, 0.0f, 0.0f);
364      glBegin(GL_TRIANGLE_FAN);
365      glVertex2f(0, 0);
366      glColor3f(1.0f, 1.0f, 0.0f);
367      for (int i = 0; i <= konstanta; i++)
368      {
369          glVertex2f((cos(i * dvaPi / konstanta)), sin(i * dvaPi / konstanta));
370      }
371      glEnd();
372      glPopMatrix();
373  }
374  void flashScreen()
375  {
376      glClearColor(0.6, 0, 0, 1);
377      glClear(GL_COLOR_BUFFER_BIT);
378      glClearColor(0, 0.5, 1, 1);
379  }
```

Funkcije za crtanje raketa invejdera i lovca kao parametre koriste koordinate x i y i pomoću grafičkih primitiva formiraju izgled raketa lovca i invejdera. Boja rakete invejdera je crvena, a lovca bela. Funkcija drawHit definiše raketu invejdera, a funkcija drawMissle definiše raketu lovca.

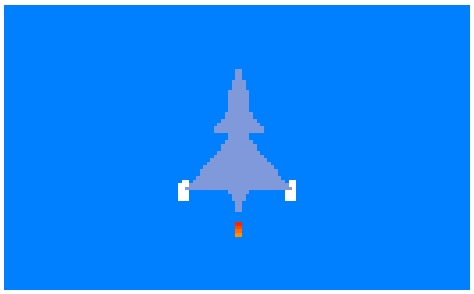
```
380  // Crtanje rakete invejdera
381  void drawHit(GLfloat x, GLfloat y)
382  {
383      glPushMatrix();
384      glTranslatef(x, y, 0);
385      glColor3f(1.0f, 0.0f, 0.1f);
386      glLineWidth(2);
387      glBegin(GL_QUADS);
388      glVertex2f(-0.1f, -0.2f);
389      glVertex2f(0.1f, -0.2f);
390      glVertex2f(0.1f, 0.2f);
391      glVertex2f(-0.1f, 0.2f);
392      glEnd();
393      glPopMatrix();
394  }
395  // Crtanje rakete lovca
396  void drawMissle(float x, float y)
397  {
398      glPushMatrix();
399      glTranslatef(x, y, 0);
400      glColor3f(1.0f, 1.0f, 1.0f);
401      glLineWidth(2);
402      glBegin(GL_POLYGON);
403      glVertex2f(-0.1f, -0.2f);
404      glVertex2f(-0.1f, 0.1f);
405      glVertex2f(0.0f, 0.2f);
406      glVertex2f(0.1f, 0.1f);
407      glVertex2f(0.1f, -0.2f);
408      glEnd();
409      glPopMatrix();
410  }
```

Bitna funkcija je funkcija za resetovanje raketa, to jest, kada raketa lovca pogodi invejdera, liniju odbrane ili izađe iz opsega prozora, tada se aktivira funkcija za resetovanje i postavlja rakete u početni položaj do sledećeg poziva funcije za ispaljivanje raketa.

```
411  // Funkcija za resetovanje raketa kada raketa pogodi cilj
412  void resetMissle(int m)
413  {
414      Missle[m].y = -1;
415      Missle[m].moving = false;
416      Missle[m].x = hunter_x - 1 + 2 * m;
417  }
418  // Funkcija za pogodak invejdera, kada je pogodjen, invejder nestaje
419  void hitInvader(int invader)
420  {
421      hit.push_back(invaders[invader]);
422  }
```

Funkcija resetMissle postavlja rakete u početni položaj. Predefinisani položaji raketa su sa leve i desne strane lovca, lovac poseduje dve rakete i dog god je raketa ispaljena, to jest, dok ne pogodi cilj ili ne nestane iz scene, mesto na krilima lovca će biti prazno, kada se uslov izvrši, sa strana lovca se ponovo crtaju rakete koje sledećom komandom bivaju ispaljene, proces se ponavlja do kraja igre.

Dok funkcija hitInvader definiše događaj kada raketa lovca pogodi invejdera, to jest, pri svakom pogotku invejdera, taj invejder se briše sa scene. Na slici u prilogu prikazan je izgled lovca kao i početne pozicije raketa.



Funkcija moveAll pokreće scenu, njena funkcija je da pokrene scenu, tačnije da pozove funkcije moveInvader, moveHit, moveMissle, Colider i glut funkcije za pokretanje scene.

```
510  void moveAll(int move)
511  {
512      moveInvader(&forward);
513      moveHit();
514      moveMissle();
515      glutPostRedisplay();
516      Collider(0);
517      glutTimerFunc(move, moveAll, move);
518  }
```

```
519  // Funkcija za ispisivanje vrednosti rezultata i broja zivota preostalih igracu
520  void results(void)
521  {
522      // Prelazi na model koordinatnog sistema
523      glMatrixMode(GL_MODELVIEW);
524      // Inicijalizuje trenutnu matricu transformacije
525      glLoadIdentity();
526      // cisti prozor za pregled sa bojom
527      glClear(GL_COLOR_BUFFER_BIT);
528      // ispisivanje rezultata na ekranu
529      std::string pts = ("Rezultat ");
530      pts += std::to_string(score);
531      drawText(pts.c_str(), pts.length(), -19.5, 29.5, 0.008, 1, 1, 1);
532      // ispisivanje broja zivota na ekranu
533      pts = ("Zivoti ");
534      pts += std::to_string(lives);
535      drawText(pts.c_str(), pts.length(), -19.5, 28, 0.008, 0.7, 0.7, 1);
536
537      for (int i = 0; i < block.size(); i++)
538      {
539          drawBlock(block[i]);
540      }
541      for (int i = 0; i < invaders.size(); i++)
542      {
543          Invaders(invaders[i]);
544      }
545      for (int i = 0; i < hit.size(); i++)
546      {
547          drawHit(hit[i].x, hit[i].y);
548      }
549      for (int i = 0; i < Missle.size(); i++)
550      {
551          drawMissle(Missle[i].x, Missle[i].y);
552      }
553      if (flash)
554      {
555          flashScreen(); // funkcija za fles ekrana pri pogotku lovca
556          flash = false;
557      }
558      for (int i = 0; i < explosion.size(); i++)
559      {
560          drawExplosion(explosion[i].x, explosion[i].y);
561      }
562      explosion.clear();
```

Funkcija result služi za računanje rezultata koji je korisnik napravio, kao i broj života lovca, svaki put kada invejder pogodi lovca, broj zivota se smanjuje, kada taj broj postane veći od 3 tada se aktivira fleg za kraj igre i na ekranu se ispisuje poruka da je igra gotova. Poeni koje igrač ostvari se prikazuju u gornjem levom uglu, kao i broj preostalih života. Svaki pogodak linije odbrane ne nosi poene, već samo kada raketa pogodi invejdera. Prva linija invejdera nosi po 10 poena, druga linija po 30 poena, dok poslednja linija invejdera nosi 50 poena. Ostvareni poeni se sabiraju i prikazuju na ekranu, na kraju igre korisnik dobija ukupan broj ostvarenik poena do trenutka kada se igra završava.

Na kraju funkcije result se proverava fleg gaveOver, koji služi za prekidanje igre, kada se fleg podigne, ispisuje se poruka na ekranu da je igra završena i definisana je funkcija koja proverava da li je broj invejdera jednak 0, račnije, ukoliko su svi invejderi uništeni, igrač je pobedio. Kada igrač pobedi, na ekranu se ispisuje poruka pobede i konačan maksimalan broj poena koje igrač može da sakupi je 1000.

```
563         //ako je "pauzirano" ispisuje poruku za kraj igre
564         //ako nema više neprijatelja ispisuje poruku za pobedu
565         if (gameOver)
566             drawText("Kraj igre!", 10, -6, 15, 0.025, 1, 0, 0);
567
568         if (invaders.size() == 0)
569         {
570             drawText("Pobeda!", 8, -6, 15, 0.025, 1, 1, 1);
571         }
572         Hunter();
573         glutSwapBuffers();
574     }
```

Funkcija windowSizeChange kao parametre dobija vrednosti visine i širine ekrana i na osnovu toga srazmerno pravi glavni prozor, ukoliko igrač poveća prozor, cela scena će biti srazmerno uvećana.

```
575 // Funkcija za promenu velicine prozora
576 void windowSizeChange(GLsizei w, GLsizei h)
577 {
578     GLsizei width, height;
579     if (h == 0)
580         h = 1;
581     // Ažurirajte promenljive
582     width = w;
583     height = h;
584
585     // Određuje dimenzije prikaza
586     glViewport(0, 0, width, height);
587
588     // Inicijalizuje koordinatni sistem
589     glMatrixMode(GL_PROJECTION);
590     glLoadIdentity();
591     // Uspostavlja prozor za izbor (levo, desno, dole,
592     // gore) držeći proporciju sa prozorom za pregled
593     if (width <= height)
594     {
595         gluOrtho2D(-20.0f, 20.0f, -3.0f * height / width, 31.0f * height / width);
596     }
597     else
598     {
599         gluOrtho2D(-20.0f * width / height, 20.0f * width / height, -3.0f, 31.0f);
600     }
601 }
```

```
602  // Funkcija za definisanje specijalnih karaktera unetih sa tastature
603  void specialKeys(int key, int x, int y)
604  {
605      // printf("%f\n", hunter_x);
606      // Pomeranje lovca
607      if (key == GLUT_KEY_LEFT && !gameOver)
608      {
609          hunter_x -= 0.5f;
610          if (hunter_x < -19)
611              hunter_x = -19;
612      }
613      if (key == GLUT_KEY_RIGHT && !gameOver)
614      {
615          hunter_x += 0.5f;
616          if (hunter_x > 19)
617              hunter_x = 19;
618      }
619      if (key == GLUT_KEY_UP && Missle[0].moving != true)
620      {
621          Missle[0].moving = true;
622      }
623      if (key == GLUT_KEY_UP && Missle[1].moving != true)
624      {
625          Missle[1].moving = true;
626      }
627      //glutTimerFunc(10, moveMissle, 20);
628      for (int i = 0; i < Missle.size(); i++)
629      {
630          if (Missle[i].moving == false)
631          {
632              Missle[i].x = hunter_x - 1 + 2 * i;
633          }
634      }
635      glutPostRedisplay();
636  }
```

Funkcija specialKeys definiše tastere sa tastature koji se koriste pri izvršavanju aplikacije. Za pomeranje lovca koriste se strelice levo i desno sa tastature koje su definisane u funkciji. Svaki put se proverava da li je fleg za kraj igre podignut i ako nije funkcije se izvršavaju. Za ispaljivanje raketa definisana je strelica na gore, svaki put kada igrač pritisne strelicu na gore, lovac ispaljuje jednu od raketa, kao što je prethodno rečeno, jedna ista raketa može biti ispaljena po jednom, to jes, tek kada pogodi cilj ili izađe iz definisanog ekrana, lovcu se dodeljuje nova.

```
637  // Funkcija za definisanje karaktera sa tastature odgovornih za kljucne dogadjaje
638  void keyboardKeys(unsigned char key, int x, int y)
639  {
640      switch (key)
641      {
642      case 27:
643          exit(0);
644
645      case 'p':
646          gameOver = !gameOver;
647          break;
648      }
649  }
650  // Funkcija odgovorna za inicijalizaciju parametara i promenljivih
651  void Initialization(void)
652  {
653      // boja pozadine
654      glClearColor(0.0f, 0.5f, 1.0f, 1.0f);
655      gluOrtho2D(-20, 20, -3, 31);
656  }
```

Funkcija keyboardKey definiše ostale korišćene tastere sa tastature. U funkciji su definisani tasteri ''P'' i ''Esc''. Pritiskom na taster ''P'' igrač gubi mogućnost da pomera lovca, podiže se fleg za kraj igre i ispisuje se na ekranu poruka o kraju igre, ponovnim pritiskom na taster, igra se nastavlja, dok pritiskom na taster ''Esc'' igra se završava i prozor se gasi.

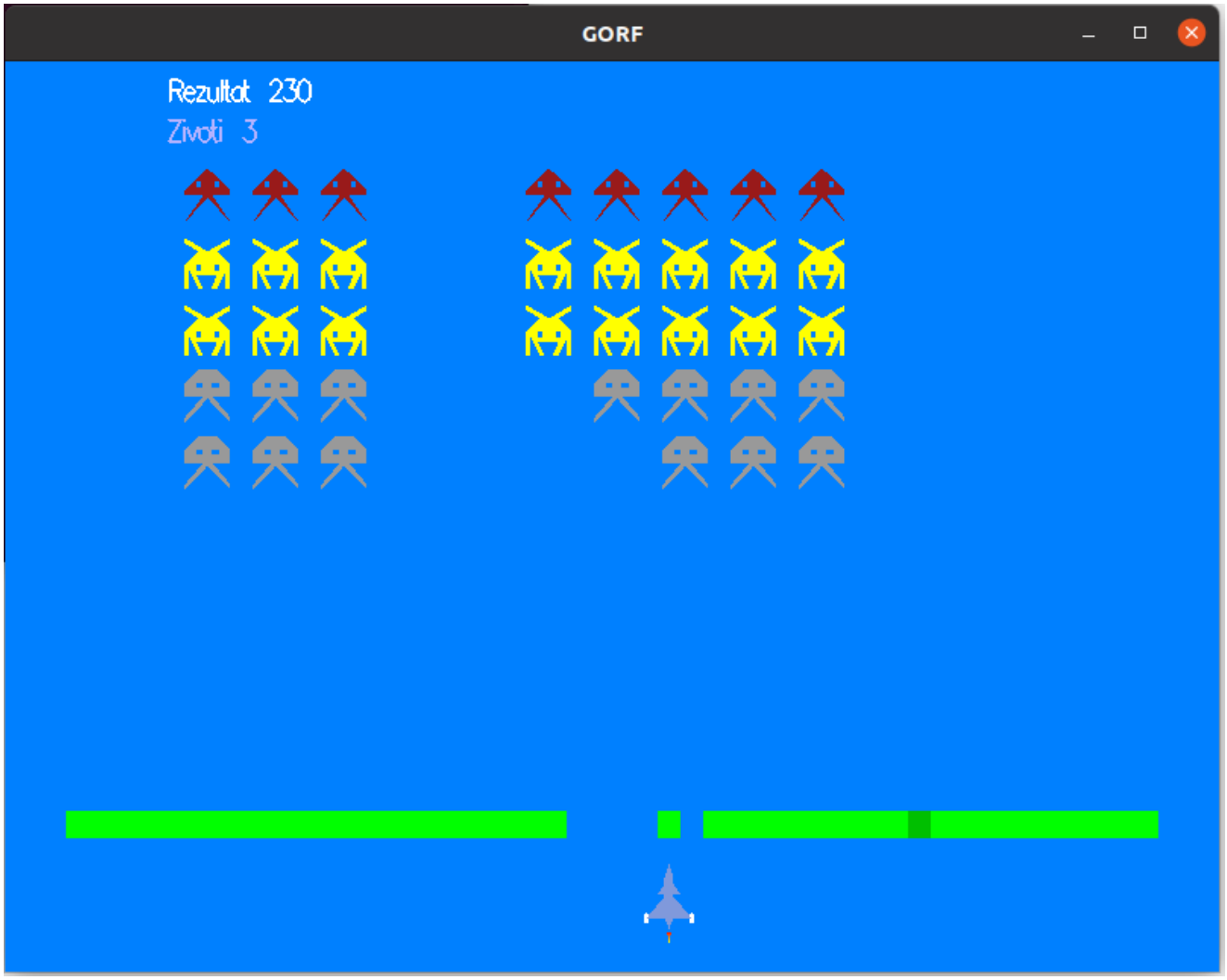
Takođe je definisana funkcija za inicijalizaciju, pod nazivom, Initialization. Svrha funkcije je da inicijalizuje pozadinu ekrana i glut parametar inicijalizacije dvodimenzionalne matrice za formiranje scene.

```
657  // Main funkcija
658  int main(int argc, char *argv[])
659  {
660      glutInit(&argc, argv); // inicijalizacija gluta
661      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
662      glEnable(GL_LINE_SMOOTH);
663      glEnable(GL_BLEND);
664      glutInitWindowPosition(512, 192);
665      glutInitWindowSize(800, 600); // velicinja pocetnog prozora
666      glutCreateWindow("GORF clone"); // ime prozora
667
668      if (!startInitializationFlag)
669      {
670          startInitialization();
671          startInitializationFlag = true;
672      }
673      glutTimerFunc(0, moveAll, 5);
674      // inicijalizacija invejdera, pocinju kretanje
675      glutTimerFunc(0, chooseInvader, 100);
676      // Poziv funkcije za ispisivanje vrednosti
677      glutDisplayFunc(results);
678      glMatrixMode(GL_PROJECTION);
679      // Poziv funkcije za promenu veličine prikaza
680      // Poziv funkcije za rukovanje posebnim ključevima
681      glutSpecialFunc(specialKeys);
682      // Poziv funkcije za rukovanje ASCII ključevima
683      glutKeyboardFunc(keyboardKeys);
684      Initialization();
685      glutReshapeFunc(windowSizeChange);
686      glutMainLoop();
687
688      return 0;
689  }
690
```

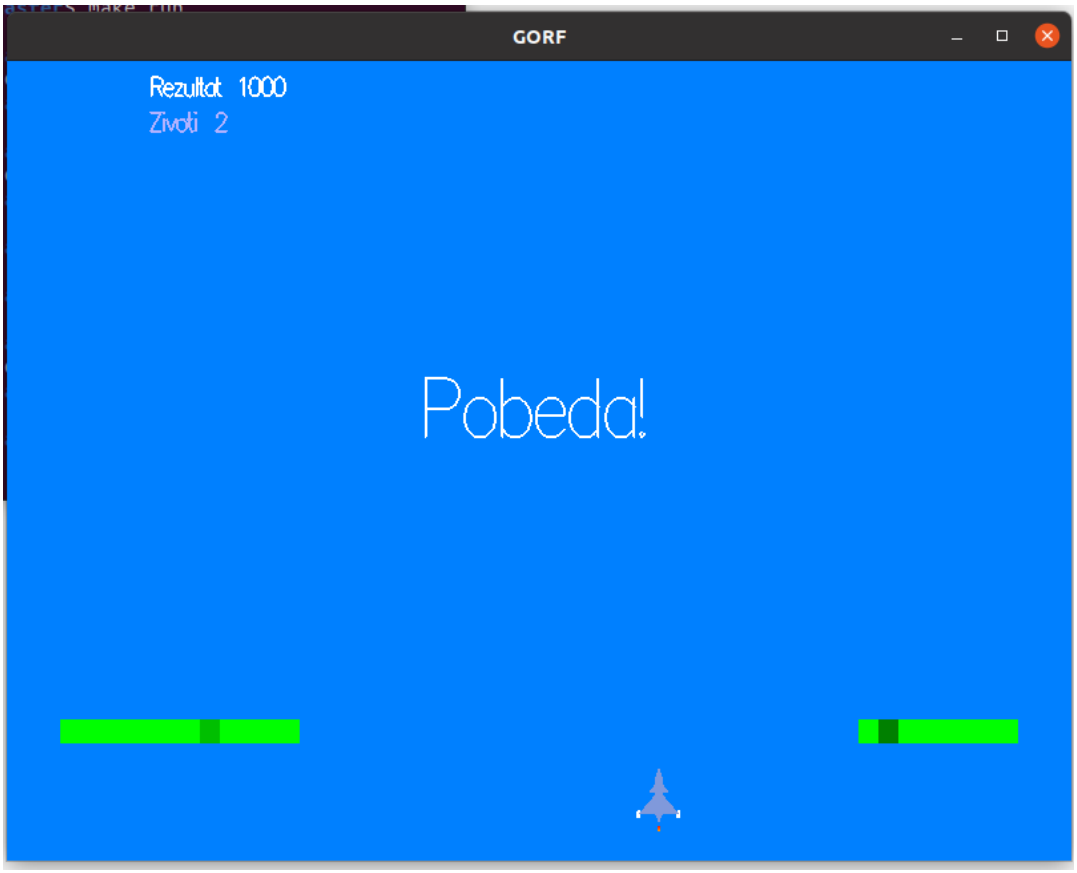
Funkcija main objedinjuje sve definisane metode i funkcije i spaja ih u konačnu aplikaciju.
Na početku se definiše mod displeja, pozicija prozora, kao i veličina prozora i ime prozora.
Proveravanjem flega za inicijalizaciju se pokreće inicijalizacija scene i pokretanje iscrtavanja primitiva, invejdera, lovca i linije odbrane, definišu se tasteri sa tastature za pokretanje lovca, funkcije za biranje invejdera, pomeranje svih primitiva i pokretanje funkcije za promenu veličine prozora.

2.1 Prikaz aplikacije tokom korišćenja

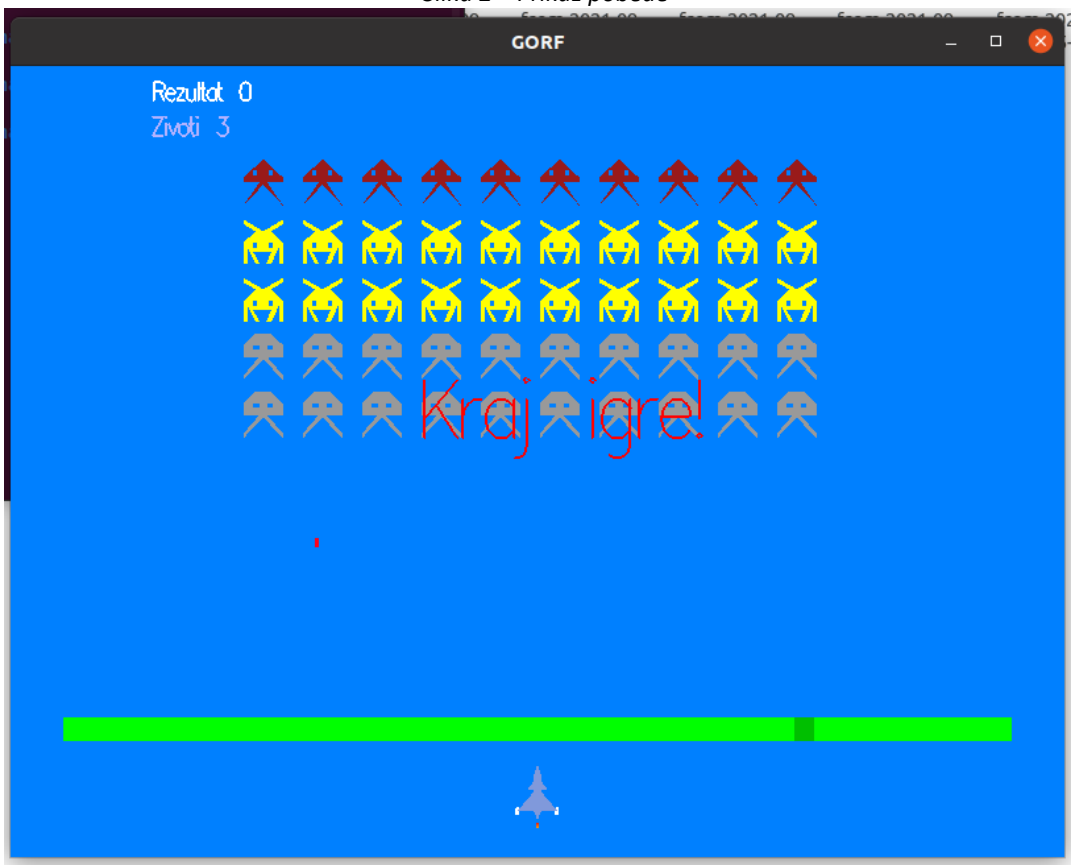
U prilogu su date slike koje prikazuju izgled kreirane aplikacije tokom njenog korišćenja.



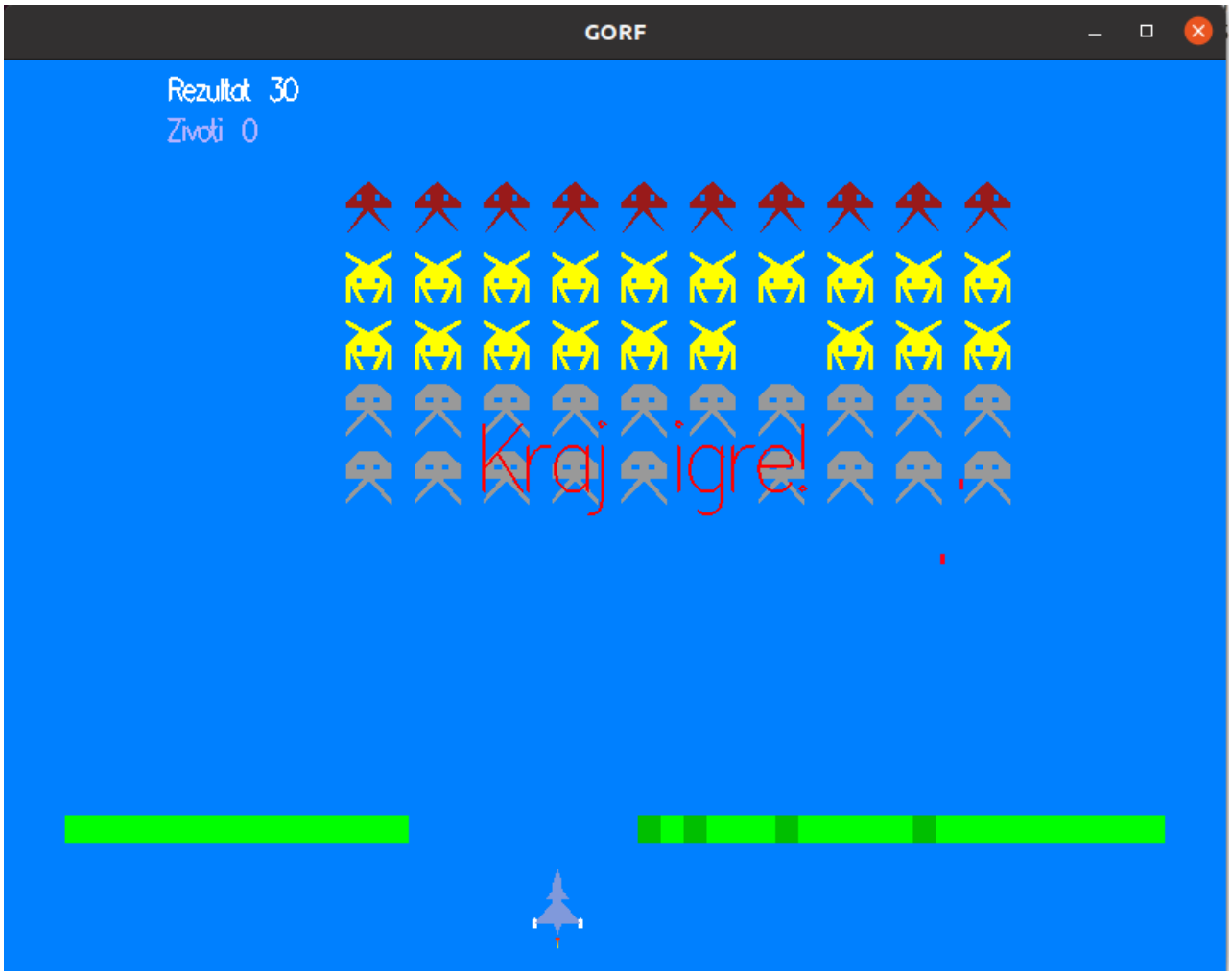
Slika 1 – Početak igrice



Slika 2 – Prikaz pobede



Slika 3 – Prikaz pritiska tastera ''P'' tokom igre



Slika 4 – Kraj igre, igrač nema više života

Literatura

Web:

[1] Wikipedia. Preuzeto: Jul 15, 2021, od LINK - *https://en.wikipedia.org/wiki/Gorf*

[2] Wikipedia. Preuzeto: Avgust 16, 2021, od LINK - *https://en.wikipedia.org/wiki/OpenGL_Utility_Toolkit*

[3] Microsoft. Preuzeto: Avgust 22, 2021, od LINK - *https://docs.microsoft.com/en-us/windows/desktop/opengl/opengl*