UNIVERZITET U KRAGUJEVCU
Fakultet inženjerskih nauka
Smer:**Računarska tehnika i softversko inžinjerstvo,**
Predmet: **Računarska grafika**

Seminarski rad

Shark Jaws

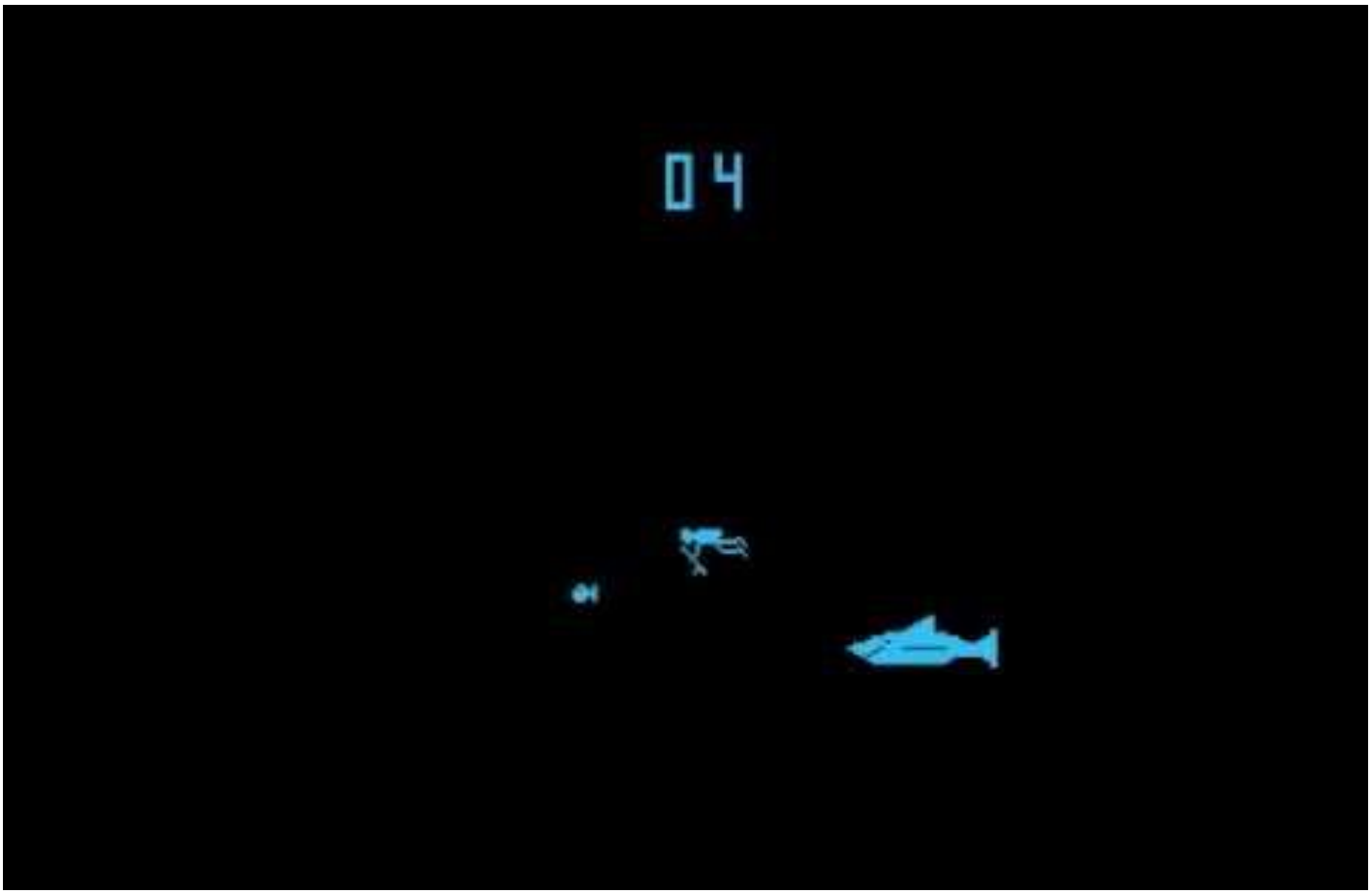
Studenti:
Mladen Cvetković 649/2019

Mentori:
red.prof.dr Nenad Filipović
Dr Tijana Geroski

Sadržaj:

1. Postavka zadataka i opis problema

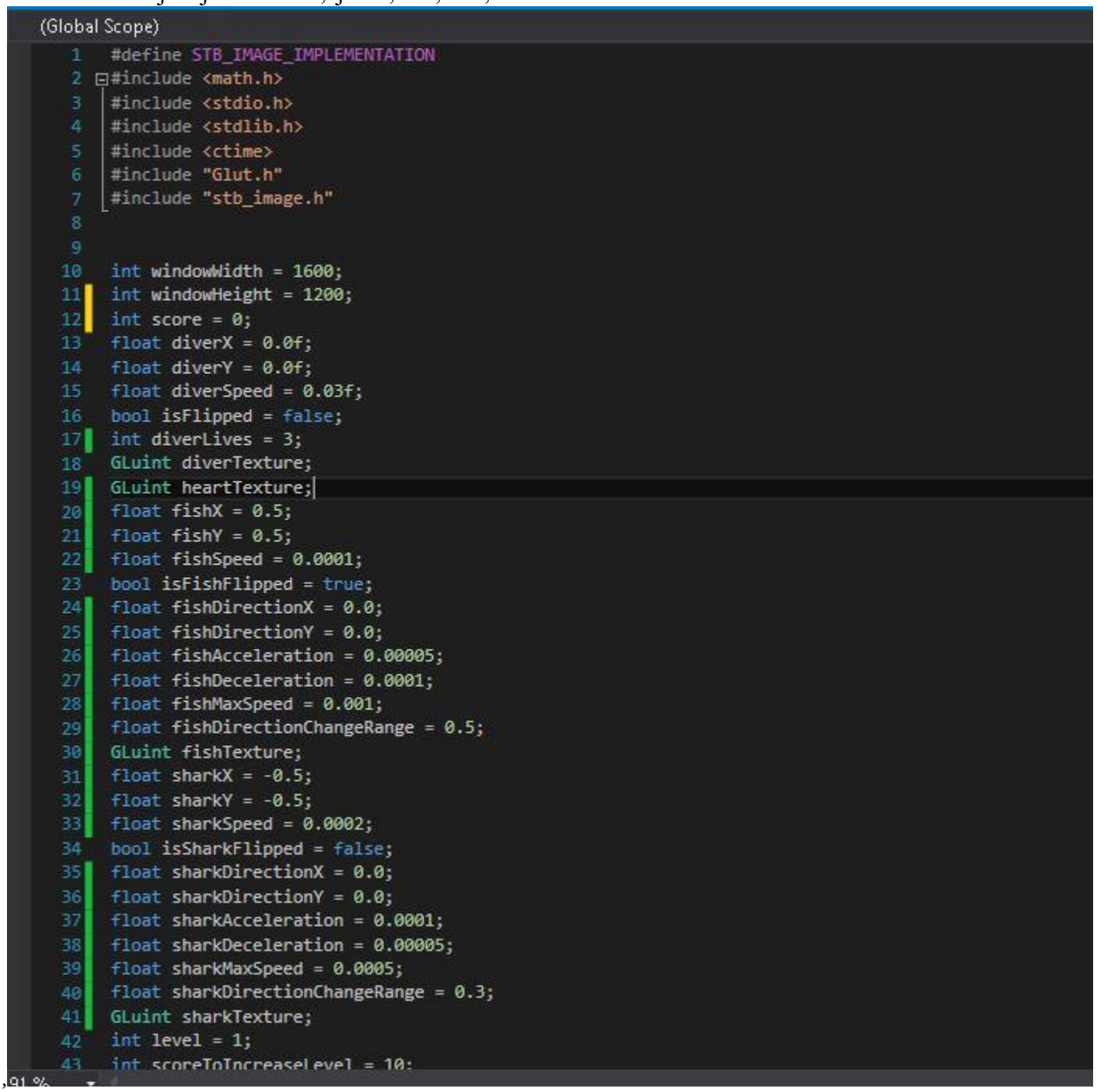
Shark jaws je video igra nastala 1975.godine i napravljena je da bude za jednog igrača. Cilj igre je da igrac koji kontrolise ronioca sakupi sto veci broj poena(score) tako sto ce pokustati da uhvati ribu koja se krece po polju izbegavajući ajkulu koja pokušava da ga pojede. Poeni se postižu tako što ronilac pređe preko ribe.



Slika 1. Prikaz izgleda originalne igrice Shark Jaws

2. Opis koda

Na početku programa definišemo biblioteke koje su nam neophodne za rad. “Glut” biblioteka sadrži definicije grafickih primitive, kao i skup funkcija za kreiranje i upravljanje prozorima, rukovanje korisničkim unosom(događaji miša i tastature) i izvođenje osnovnih OpenGL operacija.″stb_image″ biblioteka je jednostavna I lagana biblioteka koja pruža funkcionalnost za učitavanje slika iz datoteka. Na slici su takodje I sve globalne promenljive koje definišu karakteristike svakog entiteta koji će da se nalazi na ekranu uključujući ronioca,ajkulu,ribu,skor,level kao I teksturu srca.



Slika 2. Prikaz biblioteka i globalnih promenjivih

```
46
47  // Function to load an image file as a texture
48 GLuint loadTexture(const char* filename) {
49     GLuint textureID;
50     glGenTextures(1, &textureID);
51
52     int width, height, channels;
53     stbi_set_flip_vertically_on_load(true);
54     unsigned char* image = stbi_load(filename, &width, &height, &channels, 0);
55
56     if (image) {
57         GLenum format = (channels == 4) ? GL_RGBA : GL_RGB;
58
59         glBindTexture(GL_TEXTURE_2D, textureID);
60         glTexImage2D(GL_TEXTURE_2D, 0, format, width, height, 0, format, GL_UNSIGNED_BYTE, image);
61         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
62         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
63
64         stbi_image_free(image);
65     }
66     else {
67         printf("Failed to load texture: %s\n", filename);
68         stbi_image_free(image);
69         return 0;
70     }
71
72     return textureID;
73 }
74
```

Slika 3. Prikaz funkcije za učitavanje slika

Na slici iznad prikazana je funkcija ″**loadTexture**″ koja se koristi za učitavanje datoteka u vidu slika(jpg,png,itd) i kreiranje OpenGL tekstura od njih.

```
74
75 void initialize() {
76     srand(static_cast<unsigned>(time(0)));
77     // Omogucava da se transparentne slike lepo loaduju
78     glClearColor(0.0, 0.0, 0.0, 1.0);
79     glEnable(GL_BLEND);
80     glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
81     glEnable(GL_ALPHA_TEST);
82     glAlphaFunc(GL_GREATER, 0.1f);
83 }
84
85
```

Slika 4. Funkcija initialize

Zatim imamo funkciju ″**initialize**″ koja nam podešava početno stanje OpenGL okruženja I učitavanja tekstura. Konkretno u njoj smo koristili random number generator kako bi smo dobili random seed pri svakom pokretanju programa što će nam kasnije trebati za kretnu ajkule I ribe, dok smo ostale glutove funkcije koristili da bi omogucili transparentnost pozadine slikama.

Na sledećoj slici prikazana nam je funckija ″**drawDiver**″ koja služi za iscrtavanje I prikaz objekta ronioca na ekranu. Ukratko, funkcija postavlja neophodna OpenGL stanja, primenjuje transformacije na pozicioniranje I skaliranje objekta ronioca, definiše četvorougao za ronioca, binduje teksturu(sliku) na njega i prikazuje ga na ekranu.

```
88  void drawDiver() {
89      glEnable(GL_TEXTURE_2D);
90      glBindTexture(GL_TEXTURE_2D, diverTexture);
91
92      glPushMatrix();
93      glTranslatef(diverX, diverY, 0.0);
94
95      if (isFlipped) {
96          glScalef(-1.0, 1.0, 1.0);
97      }
98
99      float scale = 0.25;
100     glScalef(scale, scale, 1.0);
101
102     glBegin(GL_QUADS);
103     glTexCoord2f(0.0, 0.0); glVertex2f(-0.5, -0.5);
104     glTexCoord2f(1.0, 0.0); glVertex2f(0.5, -0.5);
105     glTexCoord2f(1.0, 1.0); glVertex2f(0.5, 0.5);
106     glTexCoord2f(0.0, 1.0); glVertex2f(-0.5, 0.5);
107     glEnd();
108
109     glPopMatrix();
110
111     glDisable(GL_TEXTURE_2D);
112 }
```

Slika 5. Funkcija drawDiver

Slika 6. Izgled
modela ronioca

Kao na prethodnoj slici funkcija ¨**drawFish**¨ ima istu funkcionalnost samo što je ovoga puta u pitanju objekat ribe.

```
114 void drawFish() {
115     glEnable(GL_TEXTURE_2D);
116     glBindTexture(GL_TEXTURE_2D, fishTexture);
117 
118     glPushMatrix();
119     glTranslatef(fishX, fishY, 0.0);
120 
121     if (isFishFlipped) {
122         glScalef(-1.0, 1.0, 1.0);
123     }
124 
125     float scale = 0.15;
126     glScalef(scale, scale, 1.0);
127 
128 
129     glBegin(GL_QUADS);
130     glTexCoord2f(0.0, 0.0); glVertex2f(-0.5, -0.5);
131     glTexCoord2f(1.0, 0.0); glVertex2f(0.5, -0.5);
132     glTexCoord2f(1.0, 1.0); glVertex2f(0.5, 0.5);
133     glTexCoord2f(0.0, 1.0); glVertex2f(-0.5, 0.5);
134     glEnd();
135 
136     glPopMatrix();
137 
138     glDisable(GL_TEXTURE_2D);
139 }
140 
```

Slika 7. Funcija drawFish

Slika 8. Izgled modela ribe

Kao i prethodne dve funkcije, funkcija ″**drawShark**″ nam iscrtava objekat za ajkulu.

```
142 void drawShark() {
143     glEnable(GL_TEXTURE_2D);
144     glBindTexture(GL_TEXTURE_2D, sharkTexture);
145 
146     glPushMatrix();
147     glTranslatef(sharkX, sharkY, 0.0);
148 
149     if (isSharkFlipped) {
150         glScalef(-1.0, 1.0, 1.0);
151     }
152 
153     float scale = 0.5;
154     glScalef(scale, scale, 1.0);
155 
156     glBegin(GL_QUADS);
157     glTexCoord2f(0.0, 0.0); glVertex2f(-0.5, -0.5);
158     glTexCoord2f(1.0, 0.0); glVertex2f(0.5, -0.5);
159     glTexCoord2f(1.0, 1.0); glVertex2f(0.5, 0.5);
160     glTexCoord2f(0.0, 1.0); glVertex2f(-0.5, 0.5);
161     glEnd();
162 
163     glPopMatrix();
164 
165     glDisable(GL_TEXTURE_2D);
166 }
```

Slika 9. Funkcija drawShark

Slika 10. Izgled
modela ajkule

Funkcija ˝**displayScore**˝ služi za prikazivanje informacija o skoru I levelu na kome se igrač nalazi. Ukratko, postavili smo boju teksta na belu, formatira rezultate o skoru I levelu u niz, pozicionira ga u gornji levi ugao, prolazi kroz svaki znak u nizu I dodeljuje mu odredjeni font.

```
167 void displayScore() {
168     glPushAttrib(GL_CURRENT_BIT);
169
170     glColor3f(1.0, 1.0, 1.0);
171     char scoreText[100];
172     sprintf_s(scoreText, "Score: %d  Level: %d", score, level);
173
174
175     glRasterPos2f(-0.95, 0.9);
176     for (int i = 0; scoreText[i] != '\0'; i++) {
177         glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, scoreText[i]);
178     }
179     glPopAttrib();
180 }
```

Slika 11. Funkcija displayScore

Sledeća funkcija ¨**drawHearts**¨ služi za iscrtavanje teksture srca koje će nam reprezentovati broj života koje poseduje ronilac. Kao I kod ostalih funkcija za iscrtavanje objekta I ovde vršimo neophodno pozicioniranje,skaliranje I u for petlji pravimo četvorouglove za 3 srca pošto će toliko života imati ronilac.

```
184 void drawHearts() {
185     glEnable(GL_TEXTURE_2D);
186     glBindTexture(GL_TEXTURE_2D, heartTexture);
187
188     glPushMatrix();
189     glTranslatef(-0.05, 0.9, 0.0);
190
191     float scale = 0.1;
192     glScalef(scale, scale, 1.0);
193
194     for (int i = 0; i < diverLives; i++) {
195         glBegin(GL_QUADS);
196         glTexCoord2f(0.0, 0.0); glVertex2f(-0.5, -0.5);
197         glTexCoord2f(1.0, 0.0); glVertex2f(0.5, -0.5);
198         glTexCoord2f(1.0, 1.0); glVertex2f(0.5, 0.5);
199         glTexCoord2f(0.0, 1.0); glVertex2f(-0.5, 0.5);
200         glEnd();
201
202         glTranslatef(1.0, 0.0, 0.0);
203     }
204
205     glPopMatrix();
206
207     glDisable(GL_TEXTURE_2D);
208 }
209
```

Slika 12. Funkcija drawHearts

Slika 13. Izgled modela srca

Funkcija ″**renderScene**″ služi za prikaz svih objekata na ekranu uključujući ribu, ajkulu, ronioca, rezultat, srca kao i game over teksturu kada ronilac ostane bez svih života.

```
212 void renderScene() {
213     glClear(GL_COLOR_BUFFER_BIT);
214
215
216     drawFish();
217     drawShark();
218     drawDiver();
219     displayScore();
220     drawHearts();
221
222
223
224     if (gameOver) {
225         glClear(GL_COLOR_BUFFER_BIT);
226         glEnable(GL_TEXTURE_2D);
227         glBindTexture(GL_TEXTURE_2D, gameOverTexture);
228
229         glBegin(GL_QUADS);
230         glTexCoord2f(0.0, 0.0); glVertex2f(-0.5, -0.5);
231         glTexCoord2f(0.0, 1.0); glVertex2f(-0.5, 0.5);
232         glTexCoord2f(1.0, 1.0); glVertex2f(0.5, 0.5);
233         glTexCoord2f(1.0, 0.0); glVertex2f(0.5, -0.5);
234         glEnd();
235
236
237     }
238
239
240     glutSwapBuffers();
241
242
243 }
```

Slika 14. Funkcija renderScene

Funkcija ″**reshapeWindow**″ je callback funkcija koja se pokreće svaki put kada dodje do promene oblik ili veličina prozora, radi prilagođavanje prozora, I postavlja ortografsku projekcijsku matricu koja definiše dokle će nam ići koordinate po x,y osi u levo,desno,dno,vrh.

Funkcija ″**getRandomDirection**″ služi da generiše random float vrednost unutar opsega (-range,+range) korišćenjem rand() funkcije. Koristi se za dobijanje random pravca kod pomeranja objekata.

```
246 void reshapeWindow(int width, int height) {
247     glViewport(0, 0, width, height);
248     glMatrixMode(GL_PROJECTION);
249     glLoadIdentity();
250     gluOrtho2D(-1, 1, -1, 1);
251     glMatrixMode(GL_MODELVIEW);
252 }
253
254 float getRandomDirection(float range) {
255     return ((static_cast<float>(rand()) / static_cast<float>(RAND_MAX)) * 2.0 - 1.0) * range;
256 }
257
258
259
```

Slika 15. Funkcija reshapeWindow I getRandomDirection

Funkcija “update” je odgovorna za ažuriranje položaja i kretnju ribe, ajkule, ronioca i rukovanje kolizije izmedju samih objekata. Na slici ispod prikazan je deo ″update″ funkcije koji služi za ažuriranje pozicije ribe po ekranu.

```
259 void update() {
260
261
262         float fishSpeedX = fishSpeed * fishDirectionX;
263         float fishSpeedY = fishSpeed * fishDirectionY;
264         float fishSpeedMagnitude = sqrt(fishSpeedX * fishSpeedX + fishSpeedY * fishSpeedY);
265
266
267         if (fishSpeedMagnitude < fishMaxSpeed) {
268             fishSpeedMagnitude += fishAcceleration;
269         }
270
271
272         if (fabs(fishDirectionX) < 0.001 && fabs(fishDirectionY) < 0.001) {
273             fishSpeedMagnitude -= fishDeceleration;
274             if (fishSpeedMagnitude < 0.0) {
275                 fishSpeedMagnitude = 0.0;
276             }
277         }
278
279
280         fishSpeedX = (fishSpeedX / fishSpeedMagnitude) * fishSpeed;
281         fishSpeedY = (fishSpeedY / fishSpeedMagnitude) * fishSpeed;
282
283
284         fishX += fishSpeedX;
285         fishY += fishSpeedY;
286
287
288         if (fishX > 1.0 || fishX < -1.0 || fishY > 1.0 || fishY < -1.0) {
289             fishDirectionX = -fishDirectionX;
290             fishDirectionY = -fishDirectionY;
291             isFishFlipped = !isFishFlipped;
292             fishDirectionX += getRandomDirection(fishDirectionChangeRange);
293             fishDirectionY += getRandomDirection(fishDirectionChangeRange);
294         }
```

Slika 16. Funkcija Update – deo za kretanje ribe.

Deo funkcije ″update″ koji služi za generisanje kretanja ajkulinog objekta po ekranu.

```
297         float sharkSpeedX = sharkSpeed * sharkDirectionX;
298         float sharkSpeedY = sharkSpeed * sharkDirectionY;
299         float sharkSpeedMagnitude = sqrt(sharkSpeedX * sharkSpeedX + sharkSpeedY * sharkSpeedY);
300
301
302         if (sharkSpeedMagnitude < sharkMaxSpeed) {
303             sharkSpeedMagnitude += sharkAcceleration;
304         }
305
306
307         if (fabs(sharkDirectionX) < 0.001 && fabs(sharkDirectionY) < 0.001) {
308             sharkSpeedMagnitude -= sharkDeceleration;
309             if (sharkSpeedMagnitude < 0.0) {
310                 sharkSpeedMagnitude = 0.0;
311             }
312         }
313
314
315         sharkSpeedX = (sharkSpeedX / sharkSpeedMagnitude) * sharkSpeed;
316         sharkSpeedY = (sharkSpeedY / sharkSpeedMagnitude) * sharkSpeed;
317
318
319         sharkX += sharkSpeedX;
320         sharkY += sharkSpeedY;
321
322         // gen ajkulin direction nakon udarca u zid
323         if (sharkX > 1.0 || sharkX < -1.0 || sharkY > 1.0 || sharkY < -1.0) {
324             sharkDirectionX = -sharkDirectionX;
325             sharkDirectionY = -sharkDirectionY;
326             isSharkFlipped = !isSharkFlipped;
327             sharkDirectionX += getRandomDirection(sharkDirectionChangeRange);
328             sharkDirectionY += getRandomDirection(sharkDirectionChangeRange);
329         }
```

Slika 17. Funkcija Update – deo za kretanje ajkule

Deo ″update″ funkcije koji upoređuje poziciju ronioca sa pozicijom ribe i ajkule i ukoliko se desi kolizija izmedju ronioca i ribe skor se povećava za 1, generiše se nova random pozicija i putanja za ribu, i proveravamo ukoliko skor dostigne zadati trešhold da uvećamo level i podignemo težinu tako što ćemo uvećati brzinu ajkuli a smanjiti roniocu. Ukoliko se desi kolizija između ronioca i ajkule roniocu oduzimamo 1 život, i resetujemo pozicije roniocu i ribi. Ukoliko ronilac izgubi sve živote postavićemo fleg ″gameOver″ da bude true tako ćemo da trigerujemo game over screen.

```
330         // kolizija izmedju ribe i ronioca
331         if (fabs(fishX - diverX) < 0.1 && fabs(fishY - diverY) < 0.1) {
332             score += 1;
333
334
335
336             fishX = getRandomDirection(1.0);
337             fishY = getRandomDirection(1.0);
338             fishDirectionX = getRandomDirection(fishDirectionChangeRange);
339             fishDirectionY = getRandomDirection(fishDirectionChangeRange);
340
341             //Proverava da li je dostignut odredjen skor i onda povecava level
342             if (score % scoreToIncreaseLevel == 0) {
343                 level++;
344                 sharkSpeed += 0.0002;
345                 diverSpeed -= 0.0005;
346             }
347         }
348
349
350         // kolizija ajkula - ronilac
351         if (fabs(sharkX - diverX) < 0.2 && fabs(sharkY - diverY) < 0.2) {
352
353             diverLives--;
354
355
356
357             diverX = getRandomDirection(1.0);
358             diverY = getRandomDirection(1.0);
359             isFlipped = false;
360
361
362             fishX = 0.5f;
363             fishY = 0.5f;
364             fishDirectionX = getRandomDirection(fishDirectionChangeRange);
365             fishDirectionY = getRandomDirection(fishDirectionChangeRange);
366         }
367         if (diverLives <= 0) {
368             gameOver = true;
369         }
370
371     glutPostRedisplay();
372
```

Slika 18. Funkcija Update – deo za koliziju

Funkcija ″**keyboardInput**″ je callback funkcija i služi za upravljanje događajima unosa sa tastature. U našem slučaju na pritisak tastera ″W″ ronilac će se kretati verikalno po Y osi, na pritisak tastera ″A″ ronilac će se kretati negativno po X osi, na pritisak tastera ″S″ će se kretati negativno po Y osi, na pritisak tastera ″D″ će se kreatit vertikalno po X osi. Na pritisak tastera ″R″ proverićemo da li je fleg ″gameOver″ pozitivan I ukoliko jeste resetovaćemo ga na negativan i takodje ćemo restetovati sve nephodne komponente na početne vrednosti kako bi se igra mogla ponoviti. Na pritisak ″escape″ izlazimo iz programa.

```
376 void keyboardInput(unsigned char key, int x, int y) {
377     switch (key) {
378     case 'w':
379     case 'W':
380         diverY += diverSpeed;
381         break;
382     case 'a':
383     case 'A':
384         diverX -= diverSpeed;
385         isFlipped = false;
386         break;
387     case 's':
388     case 'S':
389         diverY -= diverSpeed;
390         break;
391     case 'd':
392     case 'D':
393         diverX += diverSpeed;
394         isFlipped = true;
395         break;
396     case 'r':
397     case 'R':
398         if (gameOver) {
399             gameOver = false;
400             score = 0;
401             diverLives = 3;
402             level = 1;
403             fishX = 0.5;
404             fishY = 0.5;
405             fishDirectionX = getRandomDirection(fishDirectionChangeRange);
406             fishDirectionY = getRandomDirection(fishDirectionChangeRange);
407             sharkX = -0.5;
408             sharkY = -0.5;
409             sharkDirectionX = getRandomDirection(sharkDirectionChangeRange);
410             sharkDirectionY = getRandomDirection(sharkDirectionChangeRange);
411             sharkSpeed = 0.0002;
412             glutPostRedisplay();
413         }
414         break;
415     case 27: exit(0);
416         break;
417         
418     }
419 
420 
421     glutPostRedisplay();
422 }
```

Slika 19. Funkcija keyboardInput

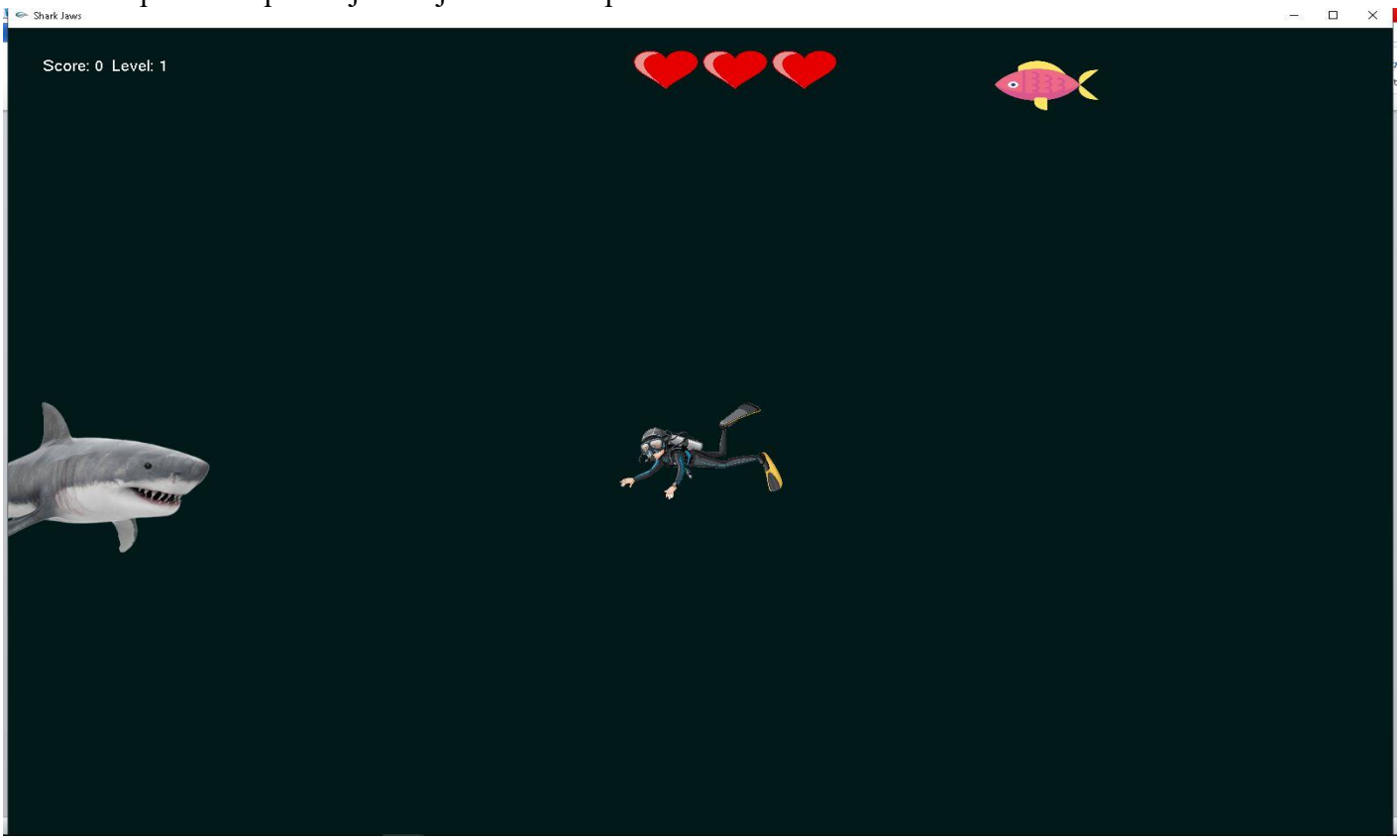
Funkcija **main** implementira sve prethodno navedene funkcije i vezuje ih za njihove funkcionalnosti i inicijalizuje grafička i tehnička podešavanja.

```
421 int main(int argc, char** argv) {
422     glutInit(&argc, argv);
423     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
424     glutInitWindowSize(windowWidth, windowHeight);
425     glutCreateWindow("Shark Jaws");
426
427
428     diverTexture = loadTexture("C:/Users/Mladen/Downloads/scuba2.png");
429     fishTexture = loadTexture("C:/Users/Mladen/Downloads/fish2.png");
430     sharkTexture = loadTexture("C:/Users/Mladen/Downloads/shark2.png");
431     heartTexture = loadTexture("C:/Users/Mladen/Downloads/heart.png");
432     gameOverTexture = loadTexture("C:/Users/Mladen/Downloads/gameover.png");
433
434
435
436     if (diverTexture == 0 || fishTexture == 0 || sharkTexture == 0 || heartTexture == 0 || gameOverTexture == 0) {
437         return 1;
438     }
439
440     // random seed za fish i shark movement
441     srand(static_cast<unsigned int>(time(NULL)));
442     fishDirectionX = static_cast<float>(rand() % 3 - 1);
443     fishDirectionY = static_cast<float>(rand() % 3 - 1);
444     sharkDirectionX = static_cast<float>(rand() % 3 - 1);
445     sharkDirectionY = static_cast<float>(rand() % 3 - 1);
446
447     glutDisplayFunc(renderScene);
448     glutReshapeFunc(reshapeWindow);
449     glutIdleFunc(update);
450     glutKeyboardFunc(keyboardInput);
451     initialize();
452     glClearColor(0, 0.1, 0.1, 1.0);
453
454     glutMainLoop();
455
456     return 0;
457 }
```

Slika 20. Funkcija main

3. Izgled aplikacije i rad

Kada se pokrene aplikacija dobijamo sledeći prozor:



Slika 21. Izgleda aplikacije pri pokretanju

Korisnik kontroliše ronioca preko W,A,S,D dugmića na tastaturi i njegov cilj je da prelazi preko ribe kako bi sakupio što veći skor. Kada predje skor 10, level će se povećati i ajkula će biti brža dok će ronilac biti sporiji. Ukoliko ga uhvati ajkula ronilac će izgubiti jedan život. Kada izgubi sva tri života korisnik će videti sledeći prozor:

Slika 22. Izgleda aplikacije kada korisnik izgubi

Literatura

[1] http://moodle.fink.rs , kurs Računarska grafika

[2] https://github.com/nothings/stb 07.06.2023.
[3] https://stackoverflow.com/ 07.6.2023.
[4] Internet