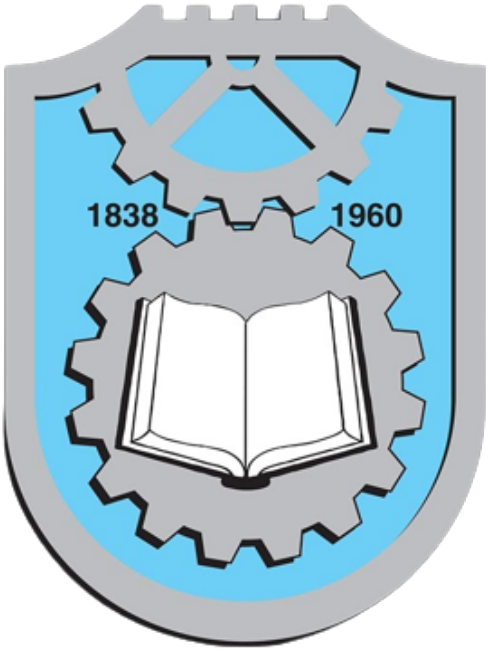
Универзитет у Крагујевцу
Факултет инжењерских наука



Рачунарска графика

Семинарски рад

OpenGL - Rebound (1974)

Професор:
Проф. др Ненад Филиповић

Студент:
Каришић Ђорђе 657/2019

__.__. 2023.

Fakultet inženjerskih nauka
Univerzitet u Kragujevcu

Računarska tehnika i softversko inženjerstvo
IV godina
VIII semestar

Predmet: Računarska grafika

SEMINARSKI RAD IZ PREDMETA RAČUNARSKA GRAFIKA

1. Napraviti aplikaciju igrice Rebound koristeći OpenGL. Opis igrice nalazi se na linku https://en.wikipedia.org/wiki/Rebound_(video_game)
   Omogućiti različite opcije (skupljanje poena, skupljanje/gubljenje života, bar dva nivoa itd.). Preporučuje se razvojno okruženje programskog jezika Visual C++ 6.0 ili Visual Studio (verzije sve do 2015).

Ime i prezime studenta, broj indeksa:
1. _____

Seminarski rad je potrebno poslati na tijanas@kg.ac.rs (izvorni kod programa, izvršna verzija, pomoćne datoteke itd.) i predati u pisanom obliku.

Pisani deo seminarskog rada povezan u spiralu treba da sadrži sledeće delove:
- naslovnu strana sa jasno napisanim osnovnim podacima
- postavku zadatka
- opis delova programa sa ilustracijama i samim izvornim kodom
- spisak korišćene literature

Odbrana seminarskog rada je deo usmenog ispita koji se brani u zakazanom terminu. Kandidat je u obavezi da praktičnom demonstracijom rada programa detaljno objasni rad pojedinih delova programa.

Kragujevac,
29. mart, 2023. godine

PREDMETNI NASTAVNIK
dr Nenad Filipović, red. prof.

Садржај

1 Увод

Rebound је видео игра за два играча, направљена од стране компаније Atari 1974. године. Игра прати дефинисани сет правила:

1. Механика игре се заснива на контроли рекета који представља једног играча, који за циљ има да лопту удари тако да она пређе на противничку страну и представи задатак другом играчу.

2. Рекет играча је подељен на 5 делова од којих сваки одбија лопту под другим углом. Почевши од првог дела који одбија лопту под оштрим углом, угао се на наредним деловима повећава како би играч имао више могућности о избору тактичких приступа ударцу лопте.

3. Играче дели мрежа која, уколико погођена, кажњава играча који је задњи ударио лопту тако што поен додељује противнику.

4. Уколико лопта испадне са терена са леве или десне стране, такође се кажњава играч који је задњи ударио лопту.

5. Уколико лопта погоди земљу тј. крајњу доњу страну, поен се додељује играчу који је задњи ударио лопту ако је она у доњу страну ударила са противничке стране терена, у супротном (са играчеве стране терена) се противнику додељује поен.

6. Након поентирања, лопта се насумично са центра пребацује на страну играча који је постигао претходни поен.

7. Игра се завршава када један од играча достигне 11 или 15 (11 или 16 у овој верзији) поена.

Оригинална верзија игре приказана на слици 1, има пар недостатака о којима ће се продискутовати у оквиру наредних секција.

Слика 1: Оригинална верзија игре

Рад је раздвојен на део који описује игру и на део који описује логику иза игре.

2 Игра

У оквиру ове секције продискутоваће се дизајн корисничког интерфејса, корисничка ограничења и могућности. Игра је раздвојена у две сцене, главни мени и свет у којем се игра дешава. У главном менију или корисничком интерфејсу корисник има могућности утицања на саму игру, коју може играти у сцени света.

2.1 Главни мени

Један од недостатака игре Rebound је могућност корисника да утиче на свет игре, било то променом стила света у којем се игра извршава или отежавањем/олакшавањем саме механике игре променом апстрактне представе играча тј. рекета или препрека попут мреже. Потребно је пружити корисницима интерфејс који даје могућност истима да утичу на претходно дефинисане карактеристике игре на једноставан начин. Сликом 2 приказан је кориснички интерфејс који игра улогу дефинисану у оквиру ове секције.

Слика 2: Главни мени игре

Након што је игра или апликација покренута и интерфејс исте видљив, покреће се музика инспирисана ретро мотиву игре. Музика је амбијентална и прати потпуно искуство корисника кроз игру, тј. кроз све сцене игре. Опције које су доступне кориснику су при промени одмах видљиве на екрану, и дате су листом:

- **Почетак игре**
  Притиском **Enter** типке тастатуре корисник покреће игру са опцијама које је поставио у оквиру менија.

- **Излаз из игре**
  Притиском **Escape** типке на тастатури интефејс нестаје и апликација престаје са радом.

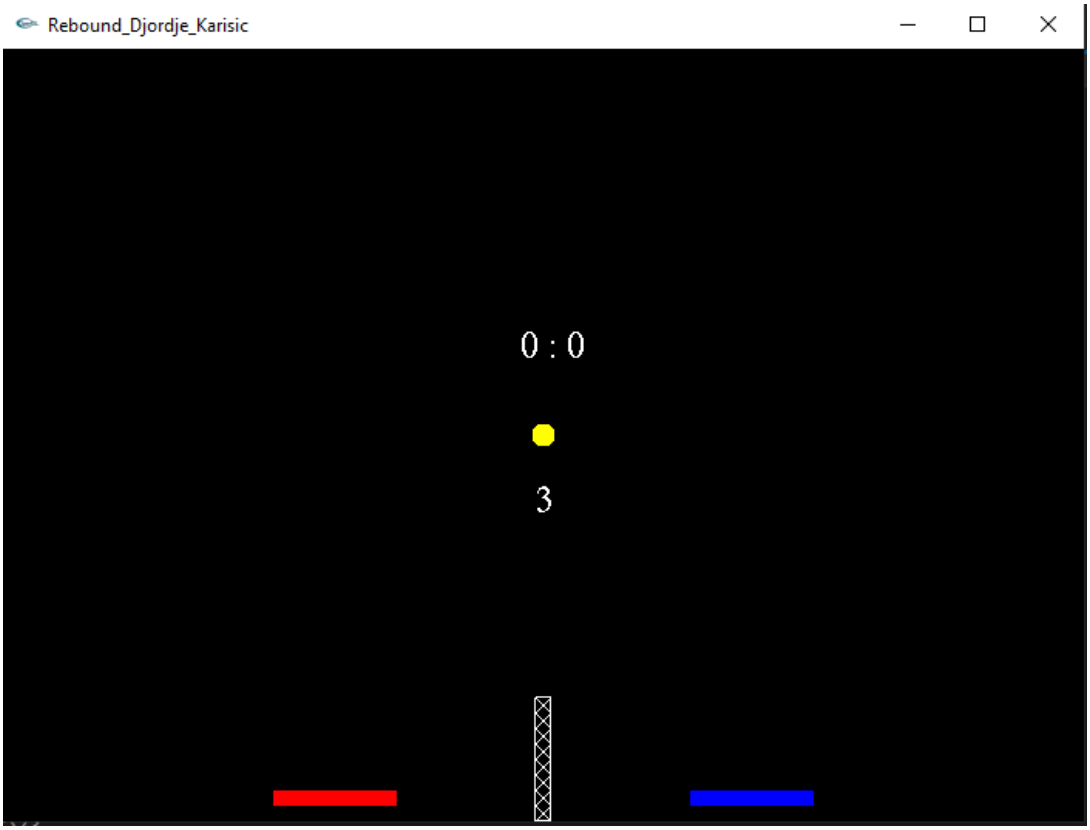
- **Промена ширине рекета играча**
  Left и **Right** стрелицом тастатуре се одређује ширина рекета играча. Левом се смањује, десном повећава ширина. Постављена је минимална и максимална граница ширине рекета како би се игра понашала жељено и притом поставила јасан опсег тежине игре.

- **Висина мреже**
  Притиском **Up** и **Down** стрелица тастатуре играч може управљати висином мреже.

- **Услов одређивања победника на основу броја поена**
  Типка **P** на тастатури врши промену услова победе тј. максималног броја поена једног играча у партији игре. Могуће вредности су 11 и 16.

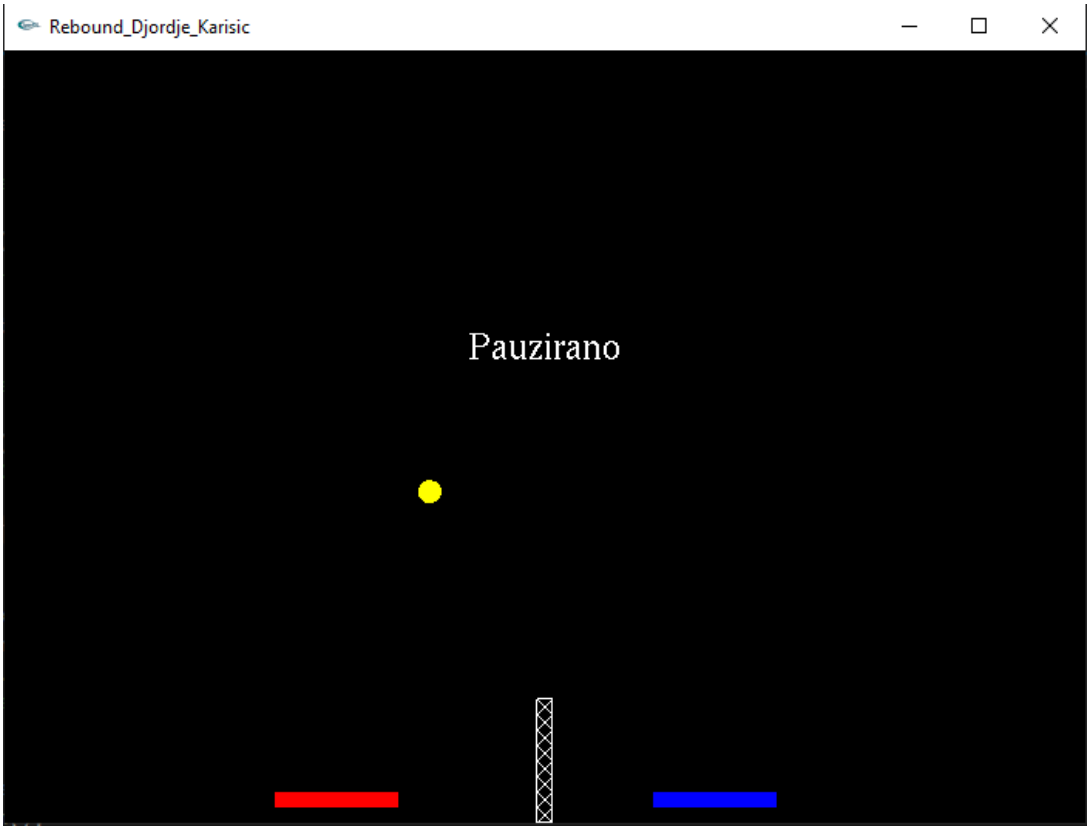
2.2 Свет игре

Приликом покретања игре појављује се нова сцена на прозору. Ова сцена представља дводимензионалну раван на чијој X оси се шетају рекети играча, сваки на својој половини у границама прозора.
На почетку и након сваког постигнутог поена почиње одбројавање до убацивања лопте у терен које траје три секунде. Лопта се на почетку убацује тако да плави играч има први прилику да удара, док се након поентирања лопта убацује у страну играча који је постигао поен, тако што што се прослеђује вектором брзине чији је интензитет X компоненте насумично одређен тако да лопта не иде истом брзином за свако убацивање.
Рекети за циљ имају прослеђивање лопте на другу страну терена преко мреже која представља главну препреку. Висина мреже зависи од избора корисника, и кажњава играча који неуспешно пребаци лопту преко ње тако што противнику додељује поен. Уколико играч проследи лопту превише далеко и она изађе ван терена, додељује се поен противнику. Ако не стигне да пребаци лопту на противничку страну док је на његовој страни у границама терена и она падне на земљу, противнику се додељује поен. Ако играч лопту удари више од четири пута за редом, противнику се додељује поен. Када један од играча дође до предефинисаног услова победе по броју поена, игра се прекида и проглашава се победник.
Сликама 3, 4, 5 и 6 приказан је свет игре и стања у којима може бити.

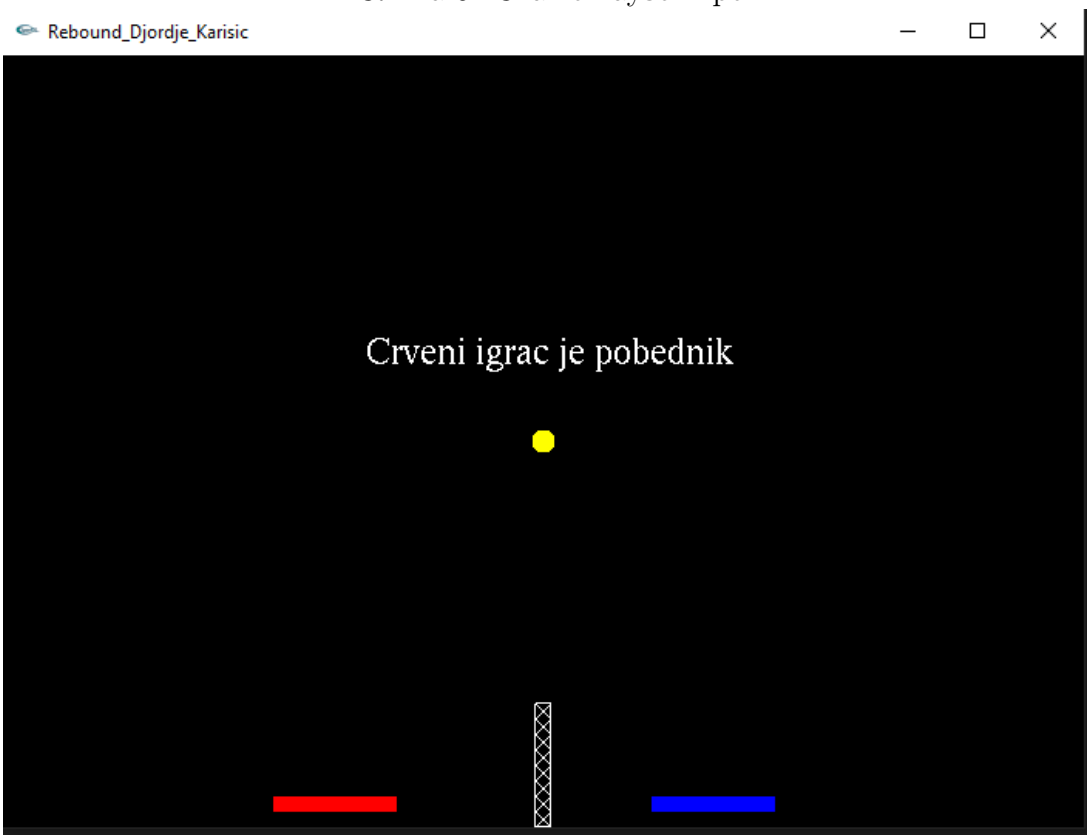


Слика 3: Почетно стање, одбројавање до почетка

Слика 4: Постигнут поен



Слика 5: Стање паузе игре



Слика 6: Испуњен услов победе, коначно стање једне партије

2.3 Механика игре

Одбијање лопте од рекета представља главну механику ове игре. Како би се обезбедио ефикасан алгоритам и систем одбијања лопте потребно је размотрити предлог решења из оригиналне игре. Предлог налаже да је рекет потребно поделити на пет делова, од којих сваки део одбија лоптицу под неким углом. Пример решења на основу овог предлога би био да се рекет подели на више једнаких делова, потом дефинисати најмањи и највећи могући угао под којим се може лоптица одбити, и инкрементовати са неким константним кораком **C** угао одбијања како редни број подеока расте. Посматраће се плави играч као пример за израчунавање одбијања углова.
Нека је рекет подељен на N подеока тако да редни бројеви подеока расту ако се рекет посматра од ивице своје стране терена ка мрежи, потребно је узети редни број неког подеока рекета i, који треба да одбије лопту под углом β, где је минимални угао ψ, а максимални $\phi = 2\psi$. На основу предлога могуће је направити формулу:

$$\beta = \psi + \frac{i(\frac{\phi}{\psi} - 1)\psi}{N} \equiv \beta = \psi + \frac{i(\phi - \psi)}{N}, \phi = 2\psi \Rightarrow \beta = \psi + \frac{i\psi}{N} \tag{1}$$

Подеок који је ближи мрежи ће лопту ударити под већим углом, а подеок који је даљи под мањим углом. Овај приступ омогућава избегавање проблема који би се десио да је систем одбијања супротан, наиме, уколико лопта пада у непосредној близини мреже играч неће моћи да је пребаци због оштрог угла одбијања.
Овај приступ има своје мане, као што су потреба за дељењем рекета и неприродан принцип одбијања, као и уколико је N мали, рекет ће имати мали број подеока и самим тим лош систем одбијања. Новонастали приступ решавању проблема вуче корен из обрасца 1, и дефинисан је:

$$\beta = \psi + (\phi - \psi)(x_1 - x_L), \phi = 2\psi \Rightarrow \beta = \psi(x_1 - x_L + 1) \tag{2}$$

Где су:

x_1	X координата почетне тачке рекета (Тачка најближа ивици терена, максимална вредност рекета на X оси)
x_L	X координата тренутне позиције лопте

Овај приступ не захтева поделу рекета и природно одбија лопту од рекета. Такође, и овде може доћи до проблема уколико је разлика између X координата најдаље тачке на рекету и лопте већа од 1, јер тада максимални угао неће бити ϕ или 2ψ, већ већи.

$$\beta = \psi + (\phi - \psi)\frac{x_1 - x_L}{x_1 - x_2}, \phi = 2\psi \Rightarrow \beta = \psi(\frac{x_1 - x_L}{x_1 - x_2} + 1) \tag{3}$$

$x_1 - x_2$	Дужина рекета, разлика X координата најдаље и најближе тачке.

Једначина 3 представља коначну формулу за рачунање угла одбијања.
Како је црвени играч готово плави играч под ефектом огледала, потребно је формуле поставити у складу са тим.

$$\beta = \psi + (\phi - \psi)\frac{x_L - x_1}{x_2 - x_1}, \phi = 2\psi \Rightarrow \beta = \psi(\frac{x_L - x_1}{x_2 - x_1} + 1) \tag{4}$$

x_1	Минимална вредност црвеног рекета на X оси.
x_2	Максимална вредност црвеног рекета на X оси.

3 Анализа кода и логике

Ова апликација или игра зависи од следећих екстерних библиотека:

- **irrKlang**
  Ова библиотека омогућава апликацији да емитује звук. Методама класе која је дефинисана у оквиру библиотеке се пушта позадинска музика и звучни ефекти. На овом линку је могуће преузети библиотеку.

- **Glut**
  Библиотека за исцртавање по прозору. Омогућава креирање потребног интерфејса игре.

Апликација се састоји из главног програма који обрађује тренутно стање игре и очитава наредно и услужне класе Player која дефинише играча.

3.1 Класа Player

Класа Player представља играча, тј. његов рекет, као и број поена и услов победе.

```
1
2  class Player {
3  private:
4          float x1, y1,x2,y2;
5          float l;
6          float h;
7          int pts;
8          bool hasWon;
9          bool lr;
```

Сви атрибути класе Player су приватни. Опис атрибута дат је наредном табелом.

x_1, y_1	Координате почетне тачке рекета
x_2, y_2	Координате крајње тачке рекета
l	Дужина рекета
h	Висина рекета
pts	Број поена играча
$hasWon$	Логичка променљива - да ли је играч победио
lr	Логичка променљива - са које стране се налази играч

```
public:
	Player(float x,float y, float len,float hgt,bool leftyrighty) {
		x1 = x;
		y1 = y;
		l = len;
		h = hgt;
		pts = 0;
		hasWon = false;
		lr = leftyrighty; // mirrorovan
		if (lr) {
			x2 = x1 - l;
			y2 = y1 + h;
		}
		else {
			x2 = x1 + l;
			y2 = y1 + h;
		}
	}
```

Све методе класе су јавне, као и конструктор. Конструктор на основу координата почетне тачке, дужине и висине рекета и на којој се страни рекет налази креира објекат са прослеђеним параметрима које пакује у одређене атрибуте и рачуна остале. У зависности од стране на којој се рекет налази другачије се тумаче координате почетка и краја рекета.

```
	void setLength(float lt) {
		l = lt;
		if (lr) {
			x2 = x1 - l;
			y2 = y1 + h;
		}
		else {
			x2 = x1 + l;
			y2 = y1 + h;
		}
	}
```

Метода setLength подешава дужину рекета на основу прослеђеног параметра *lt* и поновно рачуна координате његове крајње тачке у зависности од стране на којој се налази. Код плавог играча крајња тачка је ближа мрежи, док је код црвеног обрнуто.

```
bool Side() {
        return lr;
    }
    void updateCoords() {
        if (lr) {
            x2 = x1 - l;
            y2 = y1 + h;
        }
        else {
            x2 = x1 + l;
            y2 = y1 + h;
        }
    }
```

Метода Side враћа логичку променљиву која носи информацију о на којој страни се налази рекет.
Метода updateCoords освежава координате крајње тачке рекета тако да буду у складу са новим координатама почетне тачке.

```
float getX1() {
        return x1;
    }
    float getY1() {
        return y1;
    }
    float getX2() {
        return x2;
    }
    float getY2() {
        return y2;
    }
```

Овај сет метода враћа жељене координате жељене тачке рекета.

```
void setY(float y) {
        y1 = y;
        y2 = y1 + h;
        updateCoords();
    }
```

Метода setY очитава нову вредност у оквиру атрибута y_1 и рачуна остале атрибуте у складу са променом.

```
1       void setX(float x) {
2           if (lr) {
3               if (x >= (3.5+l) && x <= 7.0)
4                   x1 = x;
5           }
6           else {
7               if (x >= 0.0 && x <= (3.5 - l))
8                   x1 = x;
9           }
10          updateCoords();
11      }
```

Функија setX налик претходној функцији очитава нову вредност у атрибут x_1, са изузетком да се пре очитавања проверава вредност прослеђеног аргумента, и уколико је у складу са условима усваја као нова. На крају методе освежавају се све координате.

```
1   void setPt(int p) {
2           pts = p;
3       }
4       void incrementPoints() {
5           pts += 1;
6       }
7       void winCondition(bool gametype) {
8           hasWon = false;
9           if (!gametype && pts == 11)
10              hasWon = true;
11          else if (gametype && pts == 15) {
12              hasWon = true;
13          }
14      }
```

Метода setPt прослеђени аргумент учитава као тренутни број поена играча.
Метода incrementPoints инкрементује тренутни број поена за један.
Метода winCondition проверава да ли је испуњен услов за победу корисника на основу типа игре (до 11 или 16 поена) и броја поена играча.

3.2 Главни мени - имплементација

Сцена главног менија се састоји из графичке репрезентације и обрађивања догађаја приликом притиска типке тастатуре.

```
void menuKeyOperations(void) {
	if(keyStates[13]){
		play = true;
		init = false;
		pts1 = 0;
		pts2 = 0;
		p1.setLength(l);
		p2.setLength(l);
		xBall = 3 * Xmax / 4;
		yBall = Ymax / 2;
		yFactor = -mp;
		xFactor = 0.0000;
		glutDisplayFunc(drawScene);
	}
	if (keyStates['p']) {
		wincond = (wincond==11) ? 16 : 11;
		keyStates['p'] = false;
	}
	if (keyStates[27]) {
		glutDestroyWindow(glutGetWindow());
		keyStates[27] = false;
		exit(0);
	}}
void menuSpecialKeyOperations() {
	if (keySpecialStates[GLUT_KEY_DOWN]){
		if(netHeight>0.4)
			netHeight -= 0.0005;
	}
	if (keySpecialStates[GLUT_KEY_UP]){
		if(netHeight<1.2)
			netHeight += 0.0005;
	}
	if (keySpecialStates[GLUT_KEY_LEFT]){
		if (l > 0.4)
			l -= 0.0005;
	}
	if (keySpecialStates[GLUT_KEY_RIGHT]){
		if (l < 1.0)
			l += 0.0005;
	}}
```

Код изнад представља обрађивање догађаја приликом притиска тастера типки тастатуре. Стања типки тастатуре се чувају у бафери који представља низ логичнких вредности које су мапиране на индексе тако да сваки индекс представља једну типку. Притиском неке типке стање бафера за индекс типке се пребацује на True, и сигнализује се програму да је активна та типка.
Табелом приказане су фунцкије које се обављају након притиска на типку.

Enter (13)	Покреће се игра са **default** вредностима лопте и **подешеним** атрибутима играча.
P	Подешава се тип игре, до 11 или 16 поена.
Escape (27)	Прекид игре.
Arrow Key Down	Умањити мрежу по висини уколико није достигла минималну висину.
Arrow Key Up	Увећати мрежу по висини уколико није достигла максималну висину.
Arrow Key Left	Смањити рекет играча по ширини уколико није достигао минималну ширину.
Arrow Key Right	Увећати рекет играча по ширини уколико није достигао маскималну ширину.

```
1  void drawMenu(void) {
2      if (!menuMusic) {
3          SoundEngine2->setSoundVolume(0.1);
4          SoundEngine2->play2D("audio/MF-W-90.XM", true);
5          menuMusic = true;
6      }
7      xj = Xmax / 4;
8      yj = 0.1;
9      x11 = 3 * Xmax / 4;
10     y11 = 0.1;
11     p1.setX(xj);
12     p2.setX(x11);
13     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
14     glMatrixMode(GL_MODELVIEW);
15     glLoadIdentity();
16     if (!init)
17         Initialization();
18     menuKeyOperations();
19     menuSpecialKeyOperations();
20     drawEnter();
21     Writer(3.5,3.7,"Pocetak igre/Izlaz iz igre",0.6);
22     drawMainArrows();
23     drawSideArrows();
```

```
24          drawP();
25          drawEsc();
26          drawPlayer(p1);
27          drawPlayer(p2);
28          glColor3f(1.0, 1.0, 1.0);
29          Writer(3.5, 2.9, "Sirina igraca", 0.6);
30          Writer(3.5, 2.1, "Visina mreze", 0.6);
31          Writer(3.5, 1.3, "Prvi do ", 0.6);
32          glColor3f(1.0, 1.0, 0.0);
33          Writer(3.5, 1.3, std::to_string(wincond),-0.15);
34          glColor3f(1.0, 1.0, 1.0);
35          drawNet();
36          colorShift += 0.0003;
37          if (colorShift >= 1.0)
38                  colorShift = 0.5;
39          glColor3f(colorShift, 0.2, 0.2);
40          glBegin(GL_LINE_LOOP);
41          glVertex2f(4.0, 4.6);
42          glVertex2f(4.0, 5.0);
43          glVertex2f(2.9, 5.0);
44          glVertex2f(2.9, 4.6);
45          glEnd();
46          Writer(3.0, 4.8, "Rebound", 0.0);
47          glColor3f(1.0, 0.2, 0.2);
48          Writer(5.5, 4.8, "Djordje Karisic", 0.0);
49          glColor3f(1.0, 1.0, 1.0);
50          glutPostRedisplay();
51          glEnd();
52          // Flush the pipeline, swap the buffers
53          glFlush();
54          glutSwapBuffers();
55  }
```

Функција drawMenu исцртава главни мени, и помоћне функције. непосредно након позива функције укључује се музика са смањеним звуком, потом подешавају default вредности лопте и корисника како би се, након тога исцртали. Врши се иницијализација која између осталог укључује чишћење бафера, укључују се ослушкивачи догађаја и исцртавају елементи менија. Елементи менија укључују исцртане типке и текст који објашњава њихову функцију, функција Writer која низ карактера претвара у Bitmap како би се очитали на екрану на одређеној позицији и функције исцртавања облика (играчи, мрежа...). Наслов игре има ефекат промене боје фонта које су у опсегу нијанси црвене боје. Играчи у реалном времену могу подешавати параметре игре и одмах видети како ће изгледати у току игре. Када играч притисне типку Enter долази до промене сцене.

3.3 Свет игре - имплементација

Свет игре у оквиру имплементације садржи исцртавање облика играча, мреже, лопте, провере њихових стања, утврђивање ограничења, звучни ефекти и сл.

```
void keyOperations(void) {
	if (keyStates['a']) {
		if(run)
			p1.setX(p1.getX1() - factor);
	}
	if (keyStates['d']) {
		if (run)
			p1.setX(p1.getX1() + factor);
	}
	if (keyStates['r']){
		xBall = 3 * Xmax / 4;
		yBall = Ymax/2;
		yFactor = -mp;
		xFactor = 0.0000;
		pts1 = 0;
		pts2 = 0;
		reset = true;
		resetTerrain();
	}
	if (keyStates[' ']) {
		run = !run;
		keyStates[' '] = false;
	}
	if (keyStates[27]) {
		pts1 = 0;
		pts2 = 0;
		filter = true;
		SoundEngine2->stopAllSounds();
		menuMusic = false;
		keyStates[27] = false;
	}
	if (keyStates['h']) {
		pts1 = 10;}}
void keySpecialOperations(void) {
	if (keySpecialStates[GLUT_KEY_LEFT])
		if (run)
			p2.setX(p2.getX1() - factor);
	if (keySpecialStates[GLUT_KEY_RIGHT])
		if (run)
			p2.setX(p2.getX1() + factor);}
```

Код изнад представља ослушкиваче догађаја притиска типки тастатуре. Табелом испод дат је распоред и значај сваке типке.

a/A	Црвени играч се помера на лево.
r/R	Потпуни ресет игре. Поени се враћају на 0 и лопта на default позицији.
Space	Пауза игре. Зауставља се било каква кретња лопте или играча. Притиском на исто дугме се наставља игра.
Escape (27)	Повратак на главни мени. Потпуна тренутна сцена се гаси и покреће се главни мени.
h/H	Хак како би се једном од играча повећао број поена, корисно ради тестирања.
Arrow Key Left	Плави играч се помера на лево.
Arrow Key Right	Плави играч се помера на десно.

Функција која исцртава свет игре зависи од више функција, које су испод наведене.

```
1  void drawNet() {
2          glBegin(GL_LINE_LOOP);
3          glColor3f(1.0,1.0,1.0);
4          glVertex2f(Xmax/2-0.05, 0.0);
5          glVertex2f(Xmax / 2 + 0.05, 0.0);
6          glVertex2f(Xmax / 2 + 0.05, netHeight);
7          glVertex2f(Xmax / 2 - 0.05, netHeight);
8          glEnd();
9          glBegin(GL_LINE_STRIP);
10         glColor3f(1.0,1.0,1.0);
11         glVertex2f(Xmax / 2 - 0.05,netHeight);
12         float temp = netHeight;
13         float tempX = Xmax / 2 + 0.05;
14         while (temp > 0.0) {
15                 glVertex2f(tempX, temp - 0.1);
16                 temp = temp - 0.1;
17                 if (tempX > Xmax / 2)
18                         tempX = Xmax / 2 - 0.05;
19                 else
20                         tempX = Xmax / 2 + 0.05;
21         }
22         glEnd();
23         glBegin(GL_LINE_STRIP);
24         glColor3f(1.0,1.0,1.0);
25         glVertex2f(Xmax / 2 + 0.05, netHeight);
26         temp = netHeight;
```

```
27          tempX = Xmax / 2 - 0.05;
28          while (temp > 0.0) {
29                  glVertex2f(tempX, temp - 0.1);
30                  temp = temp - 0.1;
31                  if (tempX > Xmax / 2)
32                          tempX = Xmax / 2 - 0.05;
33                  else
34                          tempX = Xmax / 2 + 0.05;
35          }
36          glEnd();
37  }
38  void drawBall() {
39          glBegin(GL_TRIANGLE_FAN);
40          glColor3f(1.0, 1.0, 0.0);
41          float dPI = 2.0 * 3.1415926535;
42          int n = 30;
43          for (int i = 0; i < n; i++) {
44                  glVertex2f(
45                          (xBall + (r * cos(i * dPI / n))), (yBall + (r * sin(i * dPI / n))));
46          }
47          glEnd();
48  }
49  void drawPlayer(Player p) {
50          glBegin(GL_QUADS);
51          glColor3f(1.0, 0.0, 0.0);
52          if (p.Side())
53                  glColor3f(0.0, 0.0, 1.0);
54          p.setLength(l);
55          glVertex2f(p.getX1(), p.getY1());
56          glVertex2f(p.getX2(), p.getY1());
57          glVertex2f(p.getX2(), p.getY2());
58          glVertex2f(p.getX1(), p.getY2());
59          glEnd();
60  }
```

- **drawNet()**
  Функција drawNet исцртава мрежу без аргумената, користи глобалну промењиву на коју корисник може утицати из главног менија и која указује на висину мреже.

- **drawBall()**
  Функција drawBall исцртава лопту, не прима аргументе јер су димензије и боја лопте константни.

- **drawPlayer(Player p)**
  Функција drawPlayer исцртава играча тако што као аргумент прима објекат играча, учитава његове атрибуте и на основу њих исцртава рекет играча.

Како би се осигурало се игра понаша у складу са правилима које треба да прати, потребно је направити функције које проверавају стање игре, и уколико дође до услова за поентирање или колизију (услови након којих се мења стање) да се из датог стања игра проследи на предвиђено наредно стање (нпр. уколико дође до поена, одброји три секунде и убаци лопту играчу који је постигао поен).

```
void paddleCollision(Player p) {
        if (yBall < p.getY2()) {
                if (p.Side()) {
                        if (xBall > p.getX2() && xBall < p.getX1()) {
                                SoundEngine->play2D("audio/solid.wav");
                                float ang = 45 + 45 * (p.getX1()-xBall);
                                yFactor = sin(DegToRadians(ang))*mp*6;
                                xFactor = -cos(DegToRadians(ang)) * mp*6;
                                if (who) {
                                        howmuch = 0;
                                        who = false;
                                }
                                howmuch += 1;
                                if (howmuch == 4) {
                                        resetTerrain();
                                        pts1 += 1;
                                        pt1 = true;
                                }
                        }
                }
                else {
                        if (xBall > p.getX1() && xBall < p.getX2()) {
                                SoundEngine->play2D("audio/solid.wav");
                                float ang = 45 + 45 * (xBall- p.getX1());
                                yFactor = sin(DegToRadians(ang)) * mp * 6;
                                xFactor = cos(DegToRadians(ang)) * mp * 6;
                                if (!who) {
                                        howmuch = 0;
                                        who = true;
                                }
                                howmuch += 1;
                                if (howmuch == 4) {
                                        resetTerrain();
                                        pts2 += 1;
                                        pt2 = true;
                                }
                        }
                }
                }
}
```

```
41  void netCollision() {
42      if (((xBall >= Xmax / 2 - 0.05) && (xBall <= Xmax / 2 + 0.05))
43      && ((yBall >= 0.0) && (yBall <= netHeight))) {
44          SoundEngine->play2D("audio/powerup.wav");
45          if (who) {
46              pts2 += 1;
47              pt2 = true;
48          }
49          else {
50              pts1 += 1;
51              pt1 = true;
52          }
53          isNet = true;
54          resetTerrain();
55      }
56  }
57  void groundCollision() {
58      if (yBall < 0.0) {
59          SoundEngine->play2D("audio/powerup.wav");
60          if (who) {
61              if (xBall <= Xmax / 2) {
62                  pts2 += 1;
63                  pt2 = true;
64              }
65              else {
66                  pts1 += 1;
67                  pt1 = true;
68              }
69          }
70          else {
71              if (xBall >= Xmax / 2) {
72                  pts1 += 1;
73                  pt1 = true;
74              }
75              else {
76                  pts2 += 1;
77                  pt2 = true;
78              }
79          }
80          resetTerrain();
81      }
82      if (xBall < 0.0 || xBall>7.0) {
83          if (who) {
84              pts1 += 1;
85              pt1 = true;
86          }
```

```
87              else {
88                  pts2 += 1;
89                  pt2 = true;
90              }
91              resetTerrain();
92          }
93  }
94
```

- **paddleCollision(Player p)**
  Функција paddleCollision на основу аргумента који представља објекат играча проверава да ли је прослеђени играч урадио лопту. Уколико је услов испуњен (лопта додирнула координату крајње тачке Y осе и налази се између почетне и крајње тачке на X оси), лопта се адекватно одбија од рекета. Броји се колико пута је лопта ударила у један рекет за редом како би се испунио услов казне након 4 узастопна ударца. При ударцу покреће се звучни ефекат који симулира звук ударца рекета у лопту.

- **netCollision()**
  Функција netCollision проверава тренутну позицију лопте и да ли је погодила мрежу. Уколико јесте, тражи се играч који је последњи ударио лопту и кажњава се тако што се његовом противнику додељује поен. Након колизије лопте са мрежом и доделе поена, терен се ресетује.

- **groundCollision()**
  Функција groundCollision проверава да ли је лопта ударила у земљу тако што проверава Y координату тренутне позиције лопте и да ли је изашла из границе терена. Поени се додељују у складу са правилима. Након додељивања поена терен се ресетује.

```
1  void Writer(float x,float y,std::string what,float offset) {
2      std::string sh = what;
3      char const* fin = sh.c_str();
4      sprintf_s(buf, fin);
5      renderBitmap(x-offset,y, GLUT_BITMAP_TIMES_ROMAN_24, buf);
6  }
```

- **Writer(float x,float y, string what, float offset)**
  Функција Writer налази тачку на екрану са координатама примљених аргумената позиције x и y, и на њу исписује ниску из аргумента what са померајем од offset аргумента. Прослеђену ниску претвара у низ карактера који учитава у бафер, из којег чита тај низ и претвара у bitmap који се исцртава. Параметар функције offset је јако битан јер је потребно померити X координату почетка исписивања ниске у лево у зависности од њене дужине.

Главна функције за исцртавање света игре користи све претходно дефинисане функције.

```
void drawScene(void){
	glutSetKeyRepeat(GLUT_KEY_REPEAT_OFF);
	if (!init)
		Initialization();
	keyOperations();
	keySpecialOperations();
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	drawPlayer(p1);
	drawPlayer(p2);
	drawNet();
	drawBall();
	if (run) {
		xBall += xFactor;
		yBall += yFactor;
		yFactor -= 9.8 * dt * mp;
		paddleCollision(p1);
		paddleCollision(p2);
		groundCollision();
		netCollision();
	}
	glColor3f(1.0, 1.0, 1.0);
	char buf[100] = { 0 };
	std::string res = "Pauzirano";
	float offset = 0.35;
	std::string sh;
	if (hero && !villain && run) {
		char buf2[100] = { 0 };
		if (reset) {
			sh = "Restartovana partija";
		}
		else {
			sh = "Poen ";
			sh += (pt1) ? "za crvenog igraca!" : "za plavog igraca!";
		}
		Writer(2.5, 2.0, sh, 0.0);
		pt1 = false;
		pt2 = false;
	}
	else if (villain && run) {
		std::string sh;
		glColor3f(1.0, 1.0, 1.0);
		sec -= 1;
			sh = std::to_string(sec);
```

```
46                  Writer(3.45, 2.0, sh, 0.0);
47                  if (!sec) {
48                      sec = 4;
49                      hero = false;
50                  }
51                  Sleep(1000);
52          }
53          if (pts1 == wincond || pts2 == wincond) {
54              //SoundEngine->play2D("audio/explosion.wav");
55              offset = 1.0;
56              res = (pts1 >pts2) ? "Crveni igrac je pobednik" : "Plavi igrac je pobednik";
57              run = false;
58              menuMusic = false;
59          }
60          else if (run) {
61              res = std::to_string(pts1) + " : " + std::to_string(pts2);
62              offset = 0.0;
63          }
64          Writer(3.35,3.0,res,offset);
65          glutPostRedisplay();
66          glEnd();
67          glFlush();
68          glutSwapBuffers();
69          if (hero)
70              villain = true;
71          else
72              villain = false;
73          reset = false;
74          if (filter) {
75              glutDisplayFunc(drawMenu);
76              filter = false;
77          }}
```

- **drawMenu()**
  Функција drawMenu исцртава потпун свет игре - симулира кретање лопте и рекета, проверава колизије, ослушкује типке тастатуре и приказује информације о стању игре играчима. Користи се занимљива комбинација променљивих *Hero u Villain* (херој и негативац) који се међусобно подстичу тако што херој подстиче постојање негативца док негативац негира постојање хероја и таквом комбинацијом омогућавају смењивање порука и тајмера који се јављају када се постигне поен или рестартује игра. Херој се укључује приликом сваког ресетовања терена, што укључује негативца али искључиво на **крају** једне итерације ове функције. Негативац позива тајмер док херој приказује догађај до којег је дошло. Када негативац заврши са исписивањем тајмера (готово одбројавање), негира хероја и на крају итерације то негирање хероја води до негације негативца.

Остале услужне функције дате су кодом испод.

```
float DegToRadians(float ang) {
        return ang * (3.14159265358979 / 180);
}
void Initialization() {
        for (int i = 0; i < 256; i++) {
                keyStates[i] = false;
                keySpecialStates[i] = false;
        }
        init = true;
        pts1 = 0;
        pts2 = 0;
}
void resetTerrain() {
        float xT = xBall;
        float yT = yBall;
        xBall = Xmax / 2;
        yBall = Ymax / 2;
        yFactor = -mp;
        int a = rand() % 2;
        if (pt1) {
                        xFactor = -mp * (1 + (rand() % 5));
        }
        else {
                if(pt2) //visak linija
                        xFactor = mp * (1 + (rand() % 5));
        }
        isNet = false;
        hero = true;}
```

- **DegToRadians(float ang)**
  Функција DegToRadians претвара прослеђени угао у радијане.
- **Initialization()**
  Функција Initialization врши иницијализацију стања типки тастатуре и поена играча.
- **resetTerrain()**
  Функција resetTerrain врши потпуни рестарт терена, враћа лопту на средину и насумично рачуна X компоненту вектора брзине (5 могућих вредности) и одређује смер лопте на основу играча који је постигао последњи поен.

3.4 Резултат

На основу могућности које игра пружа, могуће је играти игру на 1,920,000 различитих комбинација терена, и одиграти чак 1,351,680,000 различитих партија игре.