КОМПЈУТЕРСКА СИМУЛАЦИЈА БИЛИЈАРА
A COMPUTER SIMULATION OF A POOL GAME

Аутор:

ПЕТАР ПРВУЛОВИЋ, ученик III разреда Паланачке гимназије

Смедеревска Паланка, Регионални центар за таленте Пожаревац

Ментор:

НАТАША МАРКОВИЋ, професор рачунарства и информатике

Паланачка гимназија, Смедеревска Паланка

РЕЗИМЕ:

У раду је представљен поступак израде програмске симулације билијара, праћен потребним кодовима, са акцентом на самом моделу, без непотребног компликовања додавањем графике, при чему је за њу, наравно, остављен потребан простор. Модел је израђен тако да буде што универзалнији, тј. да са што мање промена буде примењив за друге игре сличног типа (голф нпр.). Текст је организован тако да представља својеврсни приручник, који може послужити као основа за даље развијање овог типа игара.

Кључне речи: вектор, кугла, координате, функција, модел

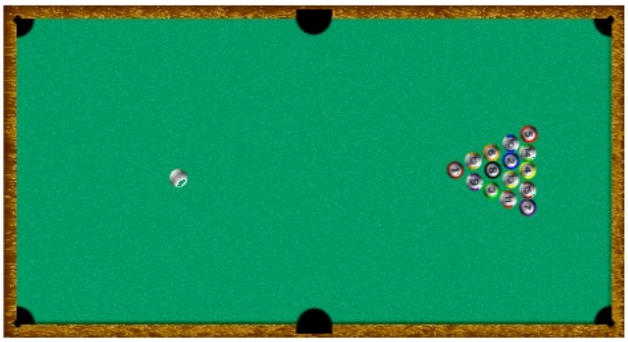
SUMMARY:

The procedure of making the program simulation of a pool game, accompanied by the necessary codes, with the stress on the model itself, omitting graphic which would additional complicate this work, is presented. The model itself is made to be as much universal as it can be, i.e. with as fewer changes as possible so it could be applied to other games of similar type, e.g. golf. The paper is organized in such a manner that it can be used as a basis for further developing of these types of games-.

Keywords: vector, ball, coordinates, function, model

1. **УВОД**

Појмом билијар обухвата се цела група игара које се играју на столовима одређених облика и димензија са одређеним бројем рупа (ако их има) и кугли и штапом који служи за ударање тих кугли чиме се постиже њихово кретање. Код нас када се каже билијар обично се мисли на стандардни билијар, који се игра са 16 различито обојених кугли на правоугаоном столу са 6 рупа. Столови се праве у више величина али је стандард 8x4 стопе, односно око 1.2x2.4 метра. Кугле су обележене бројевима од 1 до 16 и кугле од 1-7 су потпуно обојене и зовемо их "пуне", док су кугле 9-15 беле са траком у боји преко средине и зовемо их "празне". Кугла 8 је црна, а шеснаеста кугла је потпуно бела.



Слика 1 – Почетни распоред
Picture 1 – Pocetni raspored

Почетни распоред је приказан на слици 1. Први играч "разбија" кугле ударајући белу куглу. Затим играчи настоје да први убаце "своје" кугле. Први играч који убаци куглу мора да убаци и остале те групе – пуне или празне. Када играч поубацује све своје кугле на ред долази црна, и када је убаци добија игру. Међутим, ако убаци црну куглу пре времена, или је погоди прву приликом ударца губи партију. Исто важи и ако три пута убаци белу куглу. Треба додати и да ако играч погоди приликом ударца куглу која није његова, без обзира да ли је испунио правила други играч долази на ред да игра.

Даље ће бити приказан начин прављења програма за симулацију игре. Акценат је стављен на прорачуне, тако да графика није укључена, али је остављено простора за њено додавање. Пратећи текст читалац би без проблема требало да направи потпуно функционални програм, али и, зависно од потребе, да развије и усаврши одређене делове, по сопственом нахођењу.

2. МАТЕРИЈАЛ И МЕТОДИ РАДА

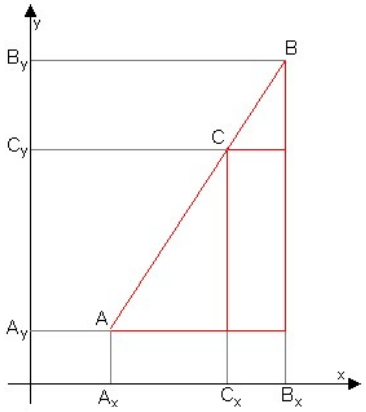
Прво је потребно разрадити физички модел и извести потребне формуле, као нпр. промена брзине кугле. Затим ћемо разбити игру на мање целине и за сваку исписати одговарајће функције и блокове кода користећи притом добијене формуле. Важно је додати да је код дат на крају овог одељка само engine игре, односно његова је улога рачунање. Графичко приказивање није укључено, али су остављени коментари на појединим местима где је потребно додати код за графичко приказивање.

2.1 Физика билијара

Разложићемо игру на више делова, а затим сваки посебно посматрати и разрадити. Елементи сваког круга су – ударац, кретање кугли и њихово сударање.

2.2 Ударац

Приликом ударца штап кугли предаје одређен импулс, дајући јој тиме одређен вектор брзине чији интензитет, природно, зависи од брзине штапа приликом судара, као и још неких фактора. Зависно од места ударца, смер зависи од смера кретања штапа и кугле, односно, пошто кугла мирује, само од смера кретања штапа. Већ овде крећемо са упрошћавањем модела – посматраћемо да је сваки ударац усмерен у центар кугле. Смер кретања штапа тада ће бити вектор са почетком у почетној позицији његовог врха, а крај у месту судара са куглом, тј. ако су врх штапа и кугла у тачкама A и B са координатама x и y координате тачке судара C добићемо коришћењем сличности, као што се види на слици 2. Када све средимо добијамо $C_x = AB_x - \frac{AB_x \cdot r}{\sqrt{AB_x^2 + AB_y^2}}$ и $C_y = AB_y - \frac{AB_y \cdot r}{\sqrt{AB_x^2 + AB_y^2}}$. Смер резултујућег вектора биће исти као и смер вектора AB, са почетком у тачки B.



Слика 2 – Тачка ударца
Picture 2 – Strike place

Пошто штап убрзава током свог кретања, узећемо да се снага ударца повећава удаљавањем штапа од кугле (повлачењем уназад), што је згодно због контрола (логично) а и најчешће је и тако у реалности. Одавде штап можемо посматрати као опругу, чији је равнотежни положај у тачки судара са куглом, а амплитуда у почетној тачки врха штапа. Силу ударца добићемо формулом $\vec{F} = -k\vec{x}$, где је k коефицијент еластичности, који износи око 10N а x растојање штапа до места судара (интензитет вектора AC). Одавде добијамо брзину у облику $\vec{v} = \frac{-k\Delta t}{m}\vec{x}$. Δt је време додира штапа и кугле, отприлике 0,001s а m маса кугле – 160g. Нека је $c = \frac{-k\Delta t}{m}$. Координате вектора брзине су $(AC_x \cdot c, AC_y \cdot c)$, са почетком у средишту кугле (тачки A).

2.3 Кретање кугле

Када кугла добије почетно убрзање наставља да се креће силом инерције, при чему на њу делује сила трења која временом смањује брзину док кугла потпуно не стане.

Приликом ударца кугла не почиње одмах да се котрља већ се одређено време клиза по подлози. У нормалним условима то време и пут углавном су мали, а и приликом судара небитно је да ли се кугле котрљају или клизају па ћемо занемарити тај део кретања кугле – сматраћемо да на њу одмах по ударцу почиње да делује трење које иначе делује на кугле које се котрљају. На крају, пошто је игра дводимензионална са куглама можемо слободно оперисати као са круговима који се крећу у ограниченој равни (столу).

За почетак требаће нам вектор брзине саопштен ударцем. После одређеног времена деловањем трења његов интензитет се смањује и добијамо га формулом $v_1 = v_0 + ta$. a је убрзање, тј. успорење трења, t протекло време а v_0 брзина на почетку кретања. Сила трења је $F_{tr}=ma$, па заменом добијамо $\vec{v}_1 = \vec{v}_0 - t\frac{\vec{F}_{tr}}{m}$. Вектор v_0 имамо, познат нам је интензитет силе трења, па координате вектора добијамо пропорцијом $\frac{|\vec{v}|}{|\vec{F}_{tr}|} = \frac{\vec{v}_x}{\vec{F}_{trx}} = \frac{\vec{v}_y}{\vec{F}_{try}}$.

Кад све заменимо добијамо координате у облику ($v_{1x} = v_{0x} - \frac{t\,|F_{tr}|\,v_x}{\sqrt{v_x^2 + v_y^2}}$,

$v_{1y} = v_{0y} - \frac{t\,|F_{tr}|\,v_y}{\sqrt{v_x^2 + v_y^2}}$).

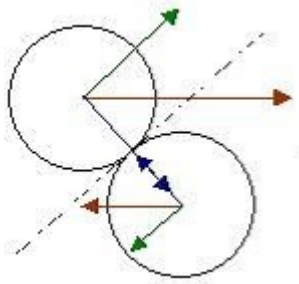
Пређен пут добијамо формулом $\vec{x}_1 = \vec{x}_0 + t\frac{\vec{v}_0 + \vec{v}_1}{2}$. x_1 је укупно пређено растојање, x_0 растојање пређено до почетка посматраног временског интервала. Поставимо x вектор коме је почетак у месту ударца а крај у положају кугле у посматраном тренутку. Његове координате имају облик ($x_{1x} = x_{0x} + t\frac{v_{0x} + v_{1x}}{2}$, $x_{1y} = x_{0y} + t\frac{v_{0y} + v_{1y}}{2}$).

2.4 Детектовање судара

Да би се кугле сударile потребно је да се нађу на растојању једнаком збиру њихових полупречника. Растојање кугли утврдићемо тако што поставимо векторе њихових положаја (у односу на нпр доњи леви угао стола), одузмемо их, па ако су једнаки збиру полупречника пређемо на решавање судара. Вектор растојања кугли A и B дат је координатама ($A_x - B_x$,$A_y - B_y$), па је растојање $|\vec{r}| = \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}$. Судар са ивицом стола десиће се ако је растојање кугле од њега једнако полупречнику те кугле. Ако су ивице дате правама нпр. x=0, x=100, y=0, y=200, судар ће се десити ако је x координата кугле r или 100-r, или y координата r или 200-r.

2.5 Решавање судара

За решавање судара потребно је знати векторе брзина обе кугле у моменту судара. Када се то добије векторе разлажемо на одређене компоненте. На слици 3 приказан је један судар. Испрекиданом линијом обележена је раван судара. Црвени вектори су вектори брзина, плави њихове нормалне компоненте, а зелени тангенцијалне. Нормалне компоненте налазе се на правој која спаја центре кугли, а зелени на прави паралелној равни судара. Вектори нормалних компоненти једнаки



Слика 3- Судар
Picture 3 - Collision

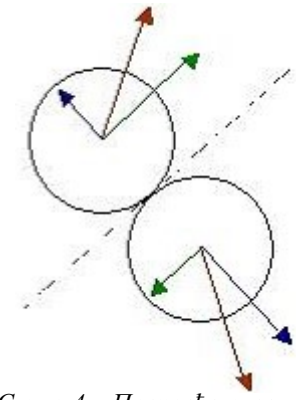
су $\vec{v}_{n1} = [\vec{v}_1 \cdot (-\vec{n})](-\vec{n})$ и $\vec{v}_{n2} = [\vec{v}_2 \cdot \vec{n}]\vec{n}$. n је овде јединични вектор усмерен у правцу нормалних компоненти. Њега добијамо тако што поставимо векторе положаја кугли, одузмемо их и поделимо добијени вектор са својим интензитетом. За кугле A и B вектор n има координате $n_x = \frac{A_x - B_x}{\sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}}$, $n_y = \frac{A_y - B_y}{\sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}}$. Заменом координате нормалних компоненти имаће облик

$$v_{n1x} = [\vec{v}_{1x} \cdot (-\vec{n})_x + \vec{v}_{1y} \cdot (-\vec{n})_y](-n)_x,\ v_{n1y} = [\vec{v}_{1x} \cdot (-\vec{n})_x + \vec{v}_{1y} \cdot (-\vec{n})_y](-n)_y$$

$$v_{n2x} = [\vec{v}_{2x} \cdot \vec{n}_x + \vec{v}_{2y} \cdot \vec{n}_y]n_x,\ v_{n2y} = [\vec{v}_{2x} \cdot \vec{n}_x + \vec{v}_{2y} \cdot \vec{n}_y]n_y$$

Исто важи и за v_2.

Током судара тангенцијалне компоненте се не мењају. Користећи закон одржања импулса, $\Delta p = 0$, и претпостављајући апсолутну еластичност кугли добијамо $\vec{v}'_2 = \vec{v}_1$ и $\vec{v}'_1 = \vec{v}_2$. Одавде резултујући вектор брзине добијамо сабирањем његове тангенцијалне компоненте и нормалне компоненте друге кугле - $\vec{v}'_1 = \vec{v}_{12} + \vec{v}_{n2}$, $\vec{v}'_2 = \vec{v}_{t2} + \vec{v}_{n1}$. Ово је представљено на слици 4. Плави вектори су нормалне компоненте, приметићете да су заменили места, зелени тангенцијалне компоненте и они остају непромењени, а црвени вектори брзине по судару. Сређивањем добијамо координате вектора брзина обе кугле:



Слика 4 – Преуређени вектори
Picture 4 –Rearranged vectors

$$v'_{1x} = v_{1x} - [\vec{v}_{1x} \cdot (-\vec{n})_x + \vec{v}_{1y} \cdot (-\vec{n})_y](-n)_x + [\vec{v}_{2x} \cdot \vec{n}_x + \vec{v}_{2y} \cdot \vec{n}_y]n_x$$

$$v'_{1y} = v_{1y} - [\vec{v}_{1x} \cdot (-\vec{n})_x + \vec{v}_{1y} \cdot (-\vec{n})_y](-n)_y + [\vec{v}_{2x} \cdot \vec{n}_x + \vec{v}_{2y} \cdot \vec{n}_y]n_y$$

$$v'_{2x} = v_{2x} - [\vec{v}_{2x} \cdot \vec{n}_x + \vec{v}_{2y} \cdot \vec{n}_y]n_x + [\vec{v}_{1x} \cdot (-\vec{n})_x + \vec{v}_{1y} \cdot (-\vec{n})_y](-n)_x$$

$$v'_{2y} = v_{2y} - \left[\vec{v}_{2x} \cdot \vec{n}_x + \vec{v}_{2y} \cdot \vec{n}_y\right] n_y + \left[\vec{v}_{1x} \cdot (-\vec{n})_x + \vec{v}_{1y} \cdot (-\vec{n})_y\right](-n)_y$$

Приликом судара са ивицом вектор брзине преуређује се тако што се промени смер нормалне компоненте, односно има облик $v = v_t - v_n$. Једноставнији начин је да, зависно да ли је ивица хоризонтална или вертикална, променимо знак *y*, односно *x* координате вектора брзине. Рупе можемо посматрати као кугле са фиксним положајем.

2.6 Прављење програма

Пошто смо разрадили понашање кугли у одређеним ситуацијама и добили потребне формуле можемо прећи на писање кодова. Прво треба дефинисати промењљиве које ће садржати потребне податке, као нпр положај кугли, њихове брзине и сл. Затим ћемо направити потребне функције и на крају додати на одговарајућим местима потребне услове да би применили правила игре. Дати кодови су писани на C језику.

2.7 Промењљиве и константе

Промењљиве и константе које се користе у игри су у ствари елементи игре – кугле, рупе, ивице стола, штап. Свака кугла има своје координате, вектор брзине и пречник. Рупе такође, сем наравно вектора брзине, с тим што су те вредности константе. Штап и ивице су одређени само координатама, при чему су ивице дате функцијама x=0, x=100, y=0, y=200 па за њих уопште нећемо правити промењљиве већ ћемо по потреби испитивати положај кугли. Да би олакшали посао око упадања кугли у рупе посматраћемо рупе такође као кугле (сталног положаја и пречника), па ако се кугла судари са рупом значи да је упала у њу. За кугле и рупе направићемо одвојене структуре које ће да садржи све потребне податке а затим и низове од 16 чланова за кугле и 6 за рупе. За врх штапа направићемо две промењљиве int stapx и int stapy које ће садржати његове координате. Друга група промењљивих су статистичке промењљиве чија је улога испитивање поштовања одређених правила и одређивање победника, следећег играча и сл. Направићемо за сваког играча по једну структуру у којој ћемо чувати потребне ове податке током игре.

2.8 Функције

За сваку од фаза игре потребно је разрадити функције (или блокове кода) и затим их спаковати у целину. Фазе игре су

- ударац
- кретање кугли

    тражење судара

    судар

    проверавање да ли је ударац завршен и ко игра следећи

- проверавање да ли је партија готова и ко је победник.

У *фази ударца* потребно је уредити комуникацију са корисником у виду омогућавања подешавања позиције штапа и задавања ударца, и затим израчунати брзину дату белој кугли. Функција ће да прати позицију миша, исцртава линију којом би се кугла кретала ако се удари и кад се кликне израчуна вектор брзине предат белој кугли. Овде ћемо додати и пар линија које ће обрисати vx компоненте свих рупа.

Фазу кретања кугли поделићемо на више подфаза:

- померање кугле
- мењање вектора брзине кугле
- тражење судара и његово решавање
    - проверавање да ли је то судар беле и црне кугле којим се завршава игра
    - проверавање да ли је кугла упала у рупу (рупе посматрамо као кугле)
    - тражење судара са ивицама стола
- провера да ли је ударац завршен (све кугле стоје)

Померање кугле и мењање вектора брзине врши се применом добијених формула. *Сударе* ћемо тражити рачунањем растојања кугли, користећи for петљу, па ако се судар догоди позивамо функцију која ће да реши судар и преуреди векторе брзина. Проверавање да ли је кугла упала у рупу ради се на исти начин као и тражење судара, с тим што је потребно уписати у vx рупе да ли је упала пуна (1), празна (2) или црна (3) кугла.

Пошто је завршен ударац проверава се ко игра и да ли је игра завршена (и ко је победник). Уредићемо ову функцију тако да проверава потребне променљиве и зависно од њих примени правила игре, на основу којих ће се по потреби променити активни играч или прогласити победник и завршити партија.

Даље су дати кодови ових функција, при чему је, као што је већ речено у уводу поглавља, остављено места за додавање кода за рад са графиком.

```
#include <math.h>
//definisanje promenljivih
struct Kugla{                // struktura koja opisuje kugle i rupe
             int x, y;             //  koordinate polozaja
             int vx, vy;           //  koordinate vektora brzine
             int r; } kugla[16];   //  poluprecnik kugle i definisanje niza
int stapx, stapy;                  // koordinate stapa
bool igra;                         // ko trenutno igra (1 true/2 false)
int pobeda;                        // ko je pobedio
int trenje = 10; int c= 10;        // trenje i koeficijent c
```

```
struct Rupa{
		int x, y;		//  koordinate polozaja
		int r;} rupa[6];	//  poluprecnik i definisanje niza
struct Igrac{
		int boja;		//koje su njegove kugle
		int ubaciobelu;		//koliko je puta ubacio belu kuglu
		int ubactudju;		//koliko je tudjih ubacio tokom udarca
		int ubacsvoju;		//koliko je svojih ubacio tokom udarca
		int ubacsvoji;		//koliko je svojih ubacio tokom igre
		bool crna;		//da li je udario prvo crnu ili ubacio u rupu
		int bela;}		//koliko puta je ubacio belu
igrac[2];			//niz od 2 strukture
int sudarud;			//broj sudara tokom udarca
// definisanje funkcija
void main(){
	//prikazuje prozor, glavni meni i sl.
	igra();
}
int igra(){
	setdefault();//redja kugle u pocetni polozaj
	bool nijekraj = true;
	int istadalje();
	for (nijekraj){		//ponavlja se dok se ne zavrsi partija
		udarac();
		pokret();
		istadalje = stadalje();
		switch (istadalje) {
		case 1: if (igra == true) pobeda = 1; //pobedio trenutn igrac
			else pobeda = 2;
			nijekraj = false;
			break;
		case 2: if (igra == true) pobeda = 2; //izgubio trenutni igrac
			else pobeda = 1;
			nijekraj = false;
			break;
		case 3: break;	//nastavlja igru trenutni igrac
		case 4: if (igra == true) igra = false; //menja se igrac
			else igra = true;
			break;}
	//ispise ko je pobednik
	return 1;
}
void udarac(){
	igrac[true].ubacsvoju = 0;
	igrac[true].ubactudju = 0;
	sudarud = 0;
	//iscrtava liniju od bele kugle u smeru u kojem ce ici ako se udari
```

```
	//iscrtava sve kugle
	//preuredjuje vektor brzine
	kugla[0].vx = (kugla[0].x - stapx) * c; //c = kdt/m modul elasticnosti
	kugla[0].vy = (kugla[0].y - stapy) * c;
}
void pokret(){
	bool a =true; int vreme;
	for(a){
		int i; //brojac
		for (i=0; i<16; i++){  //proverava redom kugle
			//brzine  (tmpvx,y su potrebni zbog polozaja)
			int tmpvx, tmpvy;
			tmpvx = kugla[i].vx;
			tmpvy = kugla[i].vy;
			kugla[i].vx = kugla[i].vx –
				(vreme*trenje*kugla[i].vx)/sqr(kugla[i].vx*kugla[i].v
				x + kugla[i].vy*kugla[i].vy);
			kugla[i].vy = kugla[i].vy –
				(vreme*trenje*kugla[i].vy)/sqr(kugla[i].vx*kugla[i].v
				x + kugla[i].vy*kugla[i].vy);
			//polozaji
			kugla[i].x = kugla[i].x + vreme*(kugla[i].vx + tmpvx)/2;
			kugla[i].y = kugla[i].y + vreme*(kugla[i].vy + tmpvy)/2;
			//iscrtava kuglu na x, y (zaokruzenim na ceo broj)
			trazisudar(i);             //provera sudara za i-tu kuglu
			a = dalisvestoje();        //provera kraja kretanja
			vreme++;     //promeniti interval ako je brzo ili sporo
		}
	}
}
void trazisudar(int i){
	int j;
	for (j=0; j<16; j++){
		//ako je razdaljina = i.r + j.r onda sudar(i.j)
		if ( sqr((kugla[i].x - kugla[j].x)*(kugla[i].x*kugla[j].x) +
			(kugla[i].y - kugla[i].y)*(kugla[i].y - kugla[i].y) ) ==
			kugla[i].r + kugla[j].r ) sudar(i,j);}
	for (j=1; j<6; j++){  //trazi upadanje u rupu
		//ako upadne u rupu
		if ( sqr((kugla[i].x - rupa[j].x)*(kugla[i].x - rupa[j].x) +
			(kugla[i].y - rupa[i].y)*(kugla[i].y - rupa[i].y) ) ==
			kugla[i].r + rupa[j].r ){
			if (igrac[igra].boja == 1){
				if (i<8){
					igrac[igra].ubacsvoju++;
					igrac[igra].ubacsvoji++;}
				if (i>8){
```

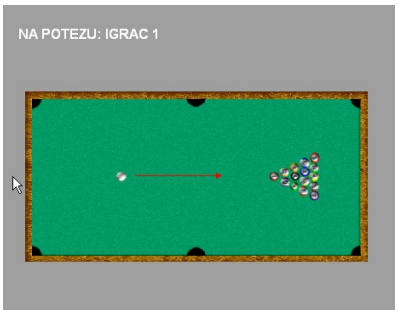
```
                                igrac[igra].ubactudju++;
                                igrac[!igra].ubacsvoji++;}
                        if (i==8){
                                if (igrac[igra].ubacsvoji!=7)
                                        igrac[igra].crna=true;
                                else igrac[igra].ubacsvoji++;}
                if (igrac[igra].boja == 2){
                        if (i>8){igrac[igra].ubacsvoju++;
                                 igrac[igra].ubacsvoji++;}
                        if (i<8){igrac[igra].ubactudju++;
                                 igrac[!igra].ubacsvoji++;}
                        if (i==8){
                                if (igrac[igra].ubacsvoji!=7)
                                        igrac[igra].crna=true;
                                else igrac[igra].ubacsvoji++;}
                if (i==0) igrac[igra].bela++;     //ako je upala bela kugla
                kugla[i].x = -10; kugla[i].y = -10;     //pomera ih sa table
                kugla[i].vx = 0;  kugla[i].vy = 0;
        }
    }
}
bool dalisvestoje(){
    int i, brojac;
    for (i=0; i<16; i++) {if ((kugla[i].vx && kugla[i].vy) == 0) brojac++;}
    if (brojac == 16) return false else return true;// false-stale,true-ne
}
void sudar(int a, int b){
    nx = (kugla[a].x - kugla[b].x)/sqr( (kugla[a].x-
        kugla[b].x)*(kugla[a].x-kugla[b].x) + (kugla[a].y-
        kugla[b].y)*(kugla[a].y-kugla[b].y) );
    ny = (kugla[a].y - kugla[b].y)/sqr( (kugla[a].x-
        kugla[b].x)*(kugla[a].x-kugla[b].x) + (kugla[a].y-
        kugla[b].y)*(kugla[a].y-kugla[b].y) );
    kugla[a].vx = kugla[a].vx - (kugla[a].x*(-nx) + kugla[a].y*(-ny))*(-nx)
        + (kugla[b].x*nx + kugla[b].y*ny)*nx;
    kugla[a].vy = kugla[a].vy - (kugla[a].x*(-nx) + kugla[a].y*(-ny))*(-ny)
        + (kugla[b].x*nx + kugla[b].y*ny)*ny;
    kugla[b].vx = kugla[b].vx - (kugla[b].x*nx + kugla[b].y*ny)*nx +
        (kugla[a].x*(-nx) + kugla[a].y*(-ny))*(-nx);
    kugla[b].vy = kugla[b].vy - (kugla[b].x*nx + kugla[b].y*ny)*ny +
        (kugla[a].x*(-nx) + kugla[a].y*(-ny))*(-ny);
    sudarud++;
    if (sudarud==1 && (a==0 && b==8 && igrac[igra].ubacsvoji<7)
        igrac[igra].crna = true; //ako je udario prvo crnu kuglu
}
void setdefault(){
    //redja kugle
```

```
	kugla[0].x = 50; kugla[0].y = 50; kugla[0].r = 3;
	kugla[1].x = 50; kugla[1].y = 138; kugla[1].r = 3;
	kugla[2].x = 56; kugla[2].y = 150; kugla[2].r = 3;
	kugla[3].x = 47; kugla[3].y = 156; kugla[3].r = 3;
	kugla[4].x = 50; kugla[4].y = 162; kugla[4].r = 3;
	kugla[5].x = 44; kugla[5].y = 151; kugla[5].r = 3;
	kugla[6].x = 38; kugla[6].y = 162; kugla[6].r = 3;
	kugla[7].x = 62; kugla[7].y = 162; kugla[7].r = 3;
	kugla[8].x = 50; kugla[8].y = 150; kugla[8].r = 3;
	kugla[9].x = 59; kugla[9].y = 156; kugla[9].r = 3;
	kugla[10].x = 53; kugla[10].y = 156; kugla[10].r = 3;
	kugla[11].x = 44; kugla[11].y = 162; kugla[11].r = 3;
	kugla[12].x = 41; kugla[12].y = 156; kugla[12].r = 3;
	kugla[13].x = 47; kugla[13].y = 144; kugla[13].r = 3;
	kugla[14].x = 54; kugla[14].y = 144; kugla[14].r = 3;
	kugla[15].x = 56; kugla[15].y = 162; kugla[15].r = 3;
	//postavlja rupe
	rupa[0].x = -3;  rupa[0].y = -3;  rupa[0].r = 4;
	rupa[1].x = 103; rupa[1].y = -3;  rupa[1].r = 4;
	rupa[2].x = -3;  rupa[2].y = 100; rupa[2].r = 4;
	rupa[3].x = 103; rupa[3].y = 100; rupa[3].r = 4;
	rupa[4].x = -3;  rupa[4].y = 203; rupa[4].r = 4;
	rupa[5].x = 103; rupa[5].y = 203; rupa[5].r = 4;
}
int stadalje(){
	//proverava da li je kraj igre i ko je pobednik
	if (igrac[igra].ubacsvoji == 8) return 1; //pobedio igrac[igra]
	else {
		if (igrac[igra].bela == 3 || igrac[igra].crna == true ||
			igrac[!igra] == 8) return 2; //izgubio igrac[igra]
		else { //proverava ko sledeci igra (da li nastavlja trenutni)
			if (igrac[igra].ubacsvoju>0 && igrac[igra].ubactudju == 0
				&& igrac[igra].crna == false && igrac[igra].bela<3)
				return 3; //igra isti
			else   return 4;} //igra drugi
	}
}
```

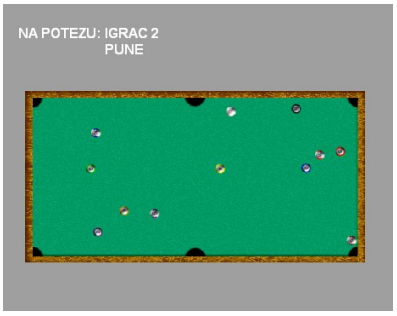
3. РЕЗУЛТАТИ ИСТРАЖИВАЊА

За потребе тестирања код је измењен и додате су му основне графичке могућности тако да је добијен програм задовољавајућих могућности, који ради све што се од њега и тражи – симулира билијарску игру под датим условима. Током игре извучено је и пар слика, датих у наставку текста. На слици 5.1 види се почетни положај. Слика 5.2 приказује кугле у покрету касније током игре, док је на слици 5.3 приказан крај игре на коме се саопштава ко је победник партије. За додавање графике развијен је велики број

модула. Тренутно најкоришћенији је OpenGL, који тај статус дугује портабилности и својим могућностима. Ништа лошији и мање коришћен је и DirectX, који је развио Microsoft, који такође има велике могућности, али, као и претходни, може бити и (често непотребно) компликован. Поменућемо и један релативно млађи графички модул – Allegro. Иако је још увек у развоју потрабилан је на већи број оперативних система. С обзиром да је прављен да буде коришћен за прављење игара, пружа доста комфора при раду, поготову са 2D графиком, где је изузетно једноставан.



Слика 5.1 – Почетни положај
Picture 5.1 – Beginning position



Слика 5.2 – Кугле у покрету
Picture 5.2 – Balls in motion



Слика 5.3 – Крај игре
Picture 5.3 – End of game

4. ЗАКЉУЧАК

У раду је представљен поступак израде, праћен потребним кодовима, организован тако да читалац може врло лако да направи једну овакву игру, али и да је по потреби промени и добије нешто сасвим друго. Променом параметара за ивице стола, рупе и сл. може се добити доста варијанти саме игре, као нпр. не мање популарна билијарска игра снокер, која се игра углавном на шестоугаоном столу који има само једну рупу, на средини, или карамбол, који се игра на столу без рупа са само три кугле, и има нешто другачија правила. Због специфичности појединих сегмената, добар део функција може се користити и за прављење игара као нпр pong, голф, DX Ball и уопште добар део 2D игара са куглама или лоптама. Такође је могуће помоћу модификација формула пренети цео систем у три димензије чиме би добили 3D билијар, или већ зависно од идеје, па овај модел може добити могућности чије границе зависе једино од идеје особе која га развија.

5. ЛИТЕРАТУРА

[1.] Teach Yourself C++ In 21 Days Second Edition (Sams)

[2.] Физика 1 за средње школе како год

[3.] forum.krstarica.com