Математичка гимназија

МАТУРСКИ РАД

- из програмирања -

Визуелна симулација кретања кугли у билијару

Ученик:
Владимир Мисовић IVц

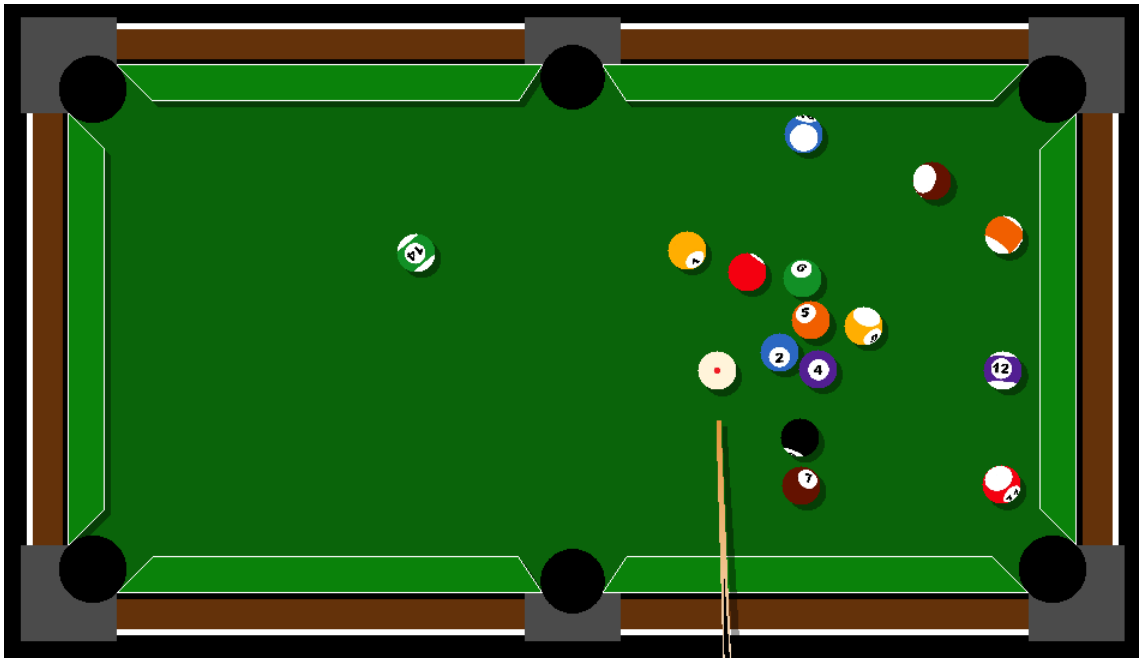
Ментор:
Филип Хаџић

Београд, април 2022.

Садржај

1

Увод



Овај програм за циљ има да уверљиво прикаже кретање, одбијање и ротирање кугли на билијарском столу. Изазов је био да уз помоћ једноставне дводимензионе графичке библиотеке прикажем тродимензионе објекте. Пробао сам да пронађем баланс између уверљивог изгледа ротације и кретања са једне стране и брзине извршавања програма у реалном времену.

Приликом израде пројекта користио сам алате за контролу верзија и аутоматизацију. Програм, код, као и сва упутства можете пронаћи на гитхаб страници: https://github.com/vmisovic/bilijar/

2

Како користити програм

Када су кугле заустављене:

- Померањем миша намештате штап
- Точкићем миша бирате јачину ударца
- Десни клик фиксира штап
- Леви клик удара белу куглу

У колико се бела кугла убаци у рупу враћање на сто се врши помоћу миша. Када поставите миш на жељено место притисните десни клик.

Пречице на тастатури:

Q	или ESC излази из програма
P	паузира игру
U	враћа позицију пре последњег ударца
L	укључује/искључује помоћне линије
R	ротира кугле ка играчу
S	зауставља кугле које су у покрету
A	враћа кугле у почетну позицију
N	циљање наопачке (намешта путању/намешта штап)
T	укључује/искључује једноставно цртање
O	укључује/искључује осетљивост штапа за прецизно бирање јачине ударца

3

Физика

3.1 Кретање кугли

У симулацији кугле се крећу равномерно успорено. Кугле ротирају без клизања и проклизавања.

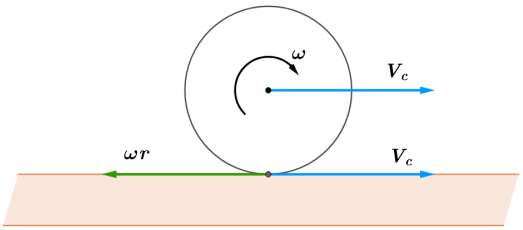
Почетна брзина кугле се задаје ударцом штапа. На куглу делује само сила трења. По другом Њутновом закону важи

$$ma = F_{tr} = N\mu = mg\mu$$

одакле се добија да је $a = g\mu$. Убрзање односно успорење је дефинисано формулом $a = \frac{\Delta v}{\Delta t}$. Одатле добијамо промену брзине за један фрејм $\Delta v = g\mu\Delta t$. Гравитационо убрзање је $g = 9,81 m/s$, а за коефицијент трења узето $\mu = 0.6$ и Δt период који протекне између два померања куги у симулацији.

Како би симулација при истим углом и задатиом брзином на различитим рачунарима давала исти резултат, фиксирано је $\Delta t = 1/120 s$. Постоји уграђена функција у СФМЛ која враћа период између два фрејма, али се приликом већих интервала, проузрокованих изненадним напором прцесора или слабијим хардвером, могу догодити грешке у симулацији због мале прецизности.

За котрљање кугле без проклизавања важи да се тачка на кугли која додирује подлогу не креће.



Веза између брзине зентра масе и угаоне брзине је $v = \omega r$.

3.2 Судари

Укупна енергија система пре и после судара је константна. У реалним системима долази до губитка енергије који се испољава у виду топлоте. У овој симулацији ти губици су занемарени. Механича енергија је збир кинетичке и потенцијалне енергије. Пошто се билијар игра на равном столу потенцијалне енергије кугли су једнаке и не мењају се.

$$\sum_{i=0}^{n} E_{ki} = \sum_{i=0}^{n} E'_{ki}$$

$$\sum_{i=0}^{n} \frac{m_i v_i^2}{2}$$

$$E_{kA} + E_{kB} = \frac{m_A v_A^2}{2} + \frac{m_B v_B^2}{2}$$

Како би смо решили ову једначину неопходно је да брзине раставимо на паралелну и нормалну компоненту у односу на раван судара. Компоненте брзина кугли паралелне са равни судара не мењају.

$$V_{y1\prime} = V_{y1}$$

$$V_{y2\prime} = V_{y2}$$

Док се нове брзине кугли нормалне компоненте налазе по формули:

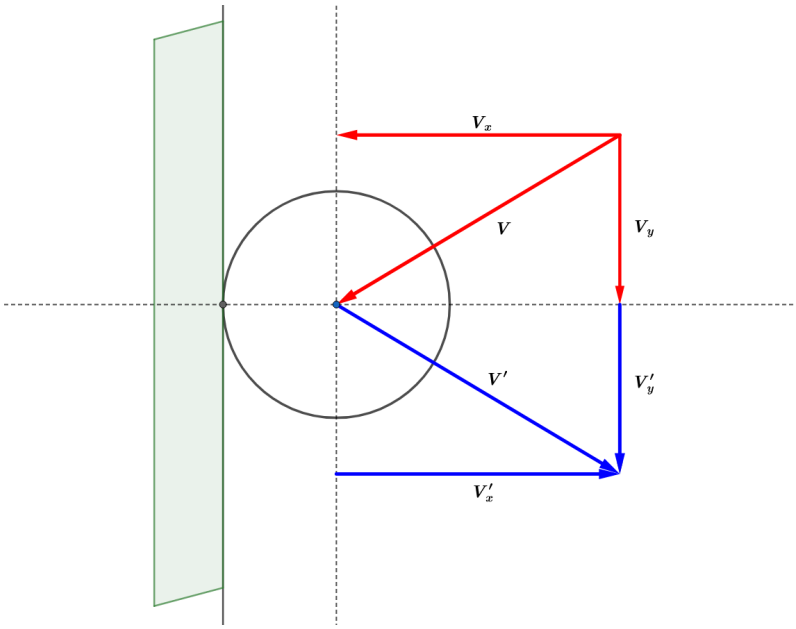
$$V_{x1\prime} = \frac{m_1 - m_2}{m_1 + m_2} V_{x1} + \frac{2m_2}{m_1 + m_2} V_{x2}$$

$$V_{x2\prime} = \frac{2m_1}{m_1 + m_2} V_{x1} + \frac{m_2 - m_1}{m_1 + m_2} V_{x2}$$

Из предходних формула можемо добити и формулу судара кугле о теме или ивицу. За зид узимамо да има много већу масу од кугле ($m_2 >> m_1$) и да се не помера ($V_2 = 0$).

$$V_{y1}\prime = V_{y1}$$

$$V_{x1}\prime = -V_{x1}$$



4

Геометрија

4.1 Детектовање судара

4.1.1 Судар две кугле

Детектовање судара кугли се врши тако што се пронађе раздаљина центара кугли и упореди са збиром њихових полупречника. У програму чувају се вектори позиција кугли. Разлика тих вектора представља вектор који спаја центре тих кугли. Уколико је дужина тог вектора мања од збира нихових полупречника кугле су се сударile.

4.1.2 Судар кугле о теме

Слично као и код судара две кугле дужина вектора разлике позиција темена и кугле се упоређује са дужином полупречника.

4.1.3 Судар кугле о ивицу

Ивица је одређена теменима. За одређивање раздаљине кугле и ивице користио сам формулу за раздаљину тачке (x_0, y_0) од праве $Ax + By + C = 0$.

$$d = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$$

Такође угао између темена и центра кугле мора бити мањи од $90°$.
Проналажење косинуса угла између вектора:
Нека је $\vec{p_1} = (x_1, y_1)$ и $\vec{p_2} = (x_2, y_2)$. Формуле за скаларни производ вектора су

$$\vec{p_1} \cdot \vec{p_2} = |\vec{p_1}||\vec{p_2}| cos\angle(\vec{p_1}, \vec{p_2})$$

$$\vec{p_1} \cdot \vec{p_2} = x_1x_2 + y_1y_2$$

Одакле добијамо да је

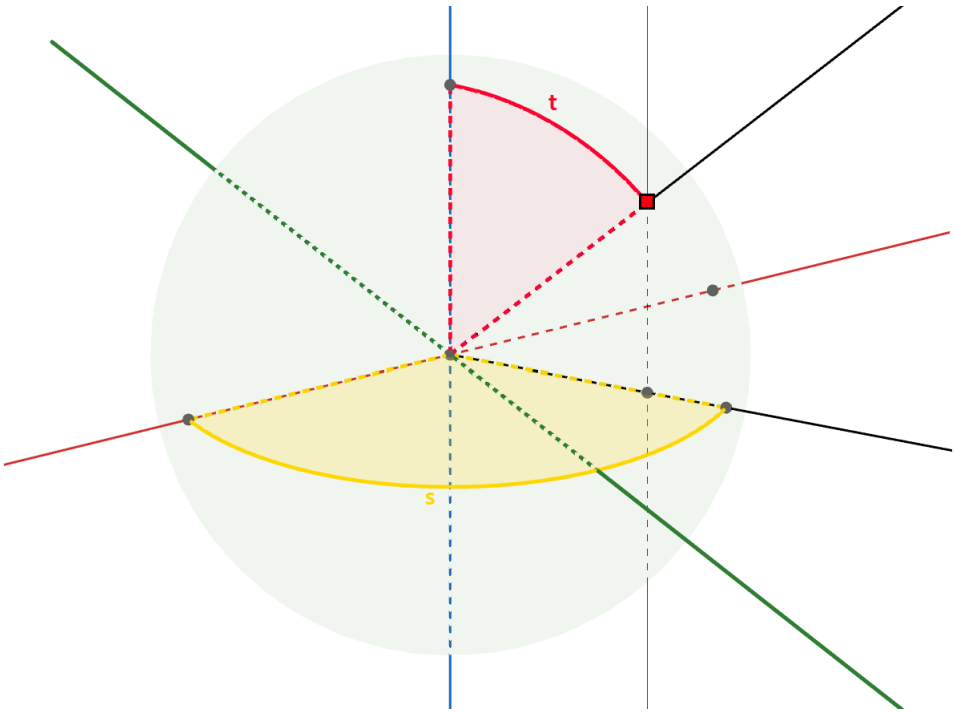
$$cos\angle(\vec{p_q}, \vec{p_2}) = \frac{x_1x_2 + y_1y_2}{\sqrt{x_1^2 + y_1^2} + \sqrt{x_2^2 + y_2^2}}$$

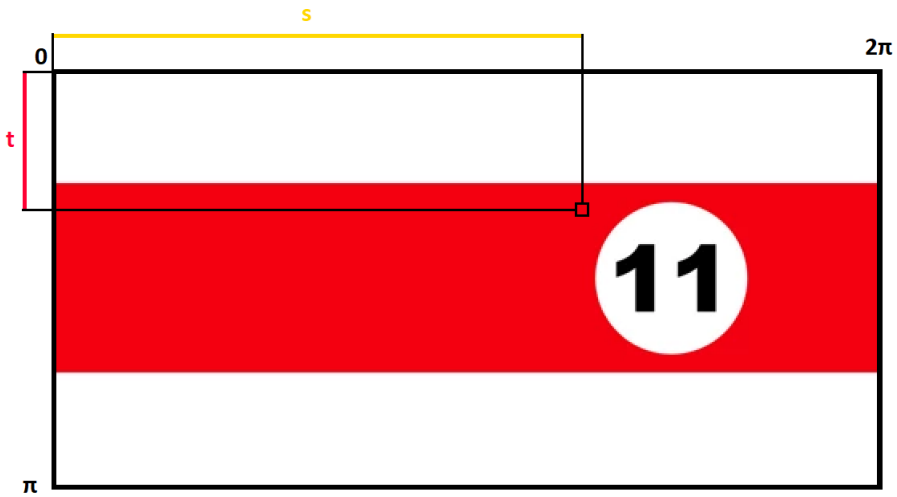
4.2 Реализације судара

За сваку куглу се прво проналази угао φ између вектора брзине и вектора који спаја центре маса кугли. Потом се та брзина разлаже на нормалну и паралелну компоненту.
Добијамо: $V_{x1} = V_1 cos\varphi$, $V_{y1} = V_1 sin\varphi$, $V_{x2} = V_2 cos\varphi$, $V_{y2} = V_2 sin\varphi$.
Брзине по y остају исте, а за x користимо формулу из физике раније објашњену.

4.3 Пројекција лопте на раван

Текстуре кугли за програм се чувају као фотографије чији је однос страна 2 према 1. Дужа страна је на x оси, а краћа је на y оси. Узмимо свферни кординатни систем као са слике. Тада за тачку на кугли са сфверним кординатама s и t одговара тачка на текстури са декартовим кординатама s и t.





Пребацивање из сферног координантног система у тродимензионални координантни систем са истим центром се врши помоћу:

$$x = r \cdot sin(t) \cdot cos(s)$$

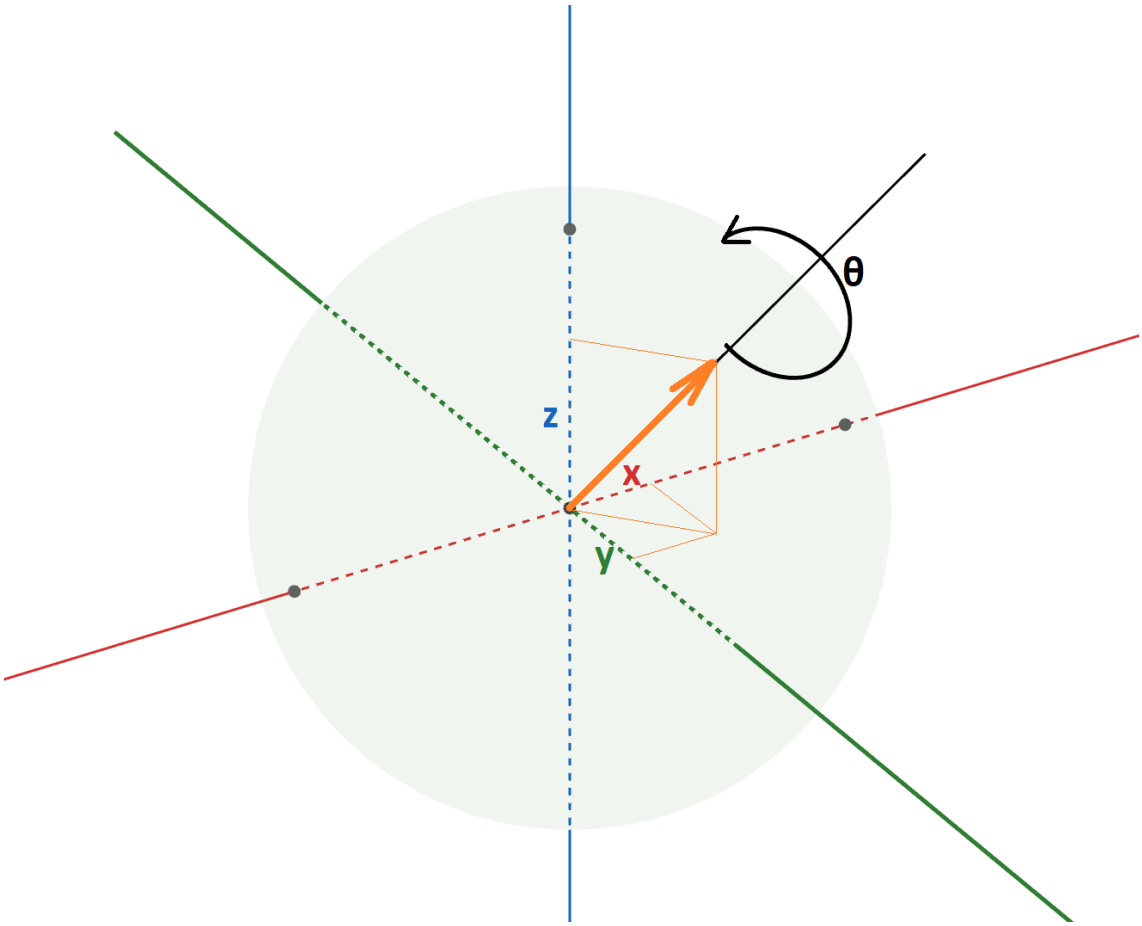
$$y = r \cdot sin(t) \cdot cos(s)$$

$$z = r \cdot cos(s)$$

4.4 Матрице ротације

Матрице ротације су математичка репрезентација ротирања тродимензионог простора. Матрица ротација M је димензије $3 * 3$ и те матрице са позицијом тачке (матрица облика $3*1$) у тродимензионом систему добијамо нове кординате (такође матрица облика $3 * 1$).

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = M \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$



Матрица која представља ротацију тела око задате осе и неки угао је:

$$\Delta M = \begin{bmatrix} c + x^2(1-c) & xy(1-c) - za & xz(1-c) + ys \\ yx(1-c) + zs & c + y^2(1-c) & yz(1-c) - xs \\ zx(1-c) - ys & zy(1-c) + xs & c + z^2(1-c) \end{bmatrix}$$

Вредности x, y и z су компоненте вектора дужине 1 који има правац и смер исти као и оса ротације, а c је косинус угла за који ротирамо, а s је синус угла за који ротирамо.

Почетна мартица рорације неротиране кугле је $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. При сваком померању кугле прво се израчунава ΔM па се множењем са старом добија нова матрица ротације.

$$M_1 = \Delta M \cdot M_0$$

5

Библиотека СФМЛ

5.1 Шта је СФМЛ

СФМЛ *(Simple and Fast Multimedia Library)* је једноставна и брза библиотека за мултимедију. Програми писани у СФМЛ-у се могу компајлирати на свим познатим оперативним системима. Сачињена је од пет делова: систем, прозор, графика, аудио и мрежа. Мој програм користи само графичку библиотеку. Ова библиотека је подржана на многим програмским језицима, за овај пројекат користио сам бибиотеку за C++.

5.2 Класе и структуре СФМЛ-а

1. sf::Window
   Класа прозора за исцртавање. Неке од дефинисаних метода су:
   bool isOpen() Да ли је прозор отворен. Користи се у главној петљи.
   bool pollEvent(Event &event) Смешта у event сигнал који је корисник унео.
   void close() Затвара прозор.

2. sf::Mouse
   Класа која чува стање миша. Метода која враћа позицију миша: Vector2i getPosition()

3. sf::Keyboard
   Класа која чува стања свих дугмића на тастатури. Метода за проверу да ли је неко дугме притиснуто: bool isKeyPressed(Key key).

4. sf::Vector2<typename T>
   Дводимензионални вектор типа T. Чува x и y вредности типа T. Дефинисане су операције сабирања и одузимања вектора истог типа, као и операција множења вектора са целобројним и децималним вредностима.

5. sf::Vector3<typename T>
   Тродимензионални вектор типа T. Чува x, y и z вредности типа T. Исте операције су дефинисане.

6. sf::Color
   Је класа која се прослеђује објектима за цртање. СФМЛ има дефинисане основне боје, а за било коју другу жељену боју користи се конструктор са три (r, g, b) или четири (r, g, b, alfa) параметра.

7. sf::RectangleShape
   Класа која чува атрибуте правоугаоника као што су позиција, величина правоугаоника и оквира, боја. У програму се помоћу ове класе исцртава сто и оквир.

8. sf::CircleShape
   Класа која чува атрибуте Круга као што су позиција, полупречник круга и дебљина оквира, боја. sf::CircleShape сам користио за исцртанвање рупа.

9. sf::VertexArray
   Класа за исцртавање низа пиксела. За сваки пиксел се подешава позиција и боја. Исцртавање објекта се врши методом draw(sf::VertexArray) из класе sf::Window. Ову класу сам користио за исцртавање кугли, зато што се изглед кугли из перспективе корисника мења током кретања.

Све класе и структуре можете пронаћи у документацији СФМЛ-а.

5.3 Пример једноставног СФМЛ програма

```
#include <SFML/Graphics.hpp>

int main()
{
    // Inicijalizacija przora za crtanje
    sf::RenderWindow window(sf::VideoMode(200, 200), "SFML works!");
    // Inicijalizacija objekta koji ce se iscrtavati
    sf::CircleShape shape(100.f);
    shape.setFillColor(sf::Color::Green);

    // Glavna petlja
    while (window.isOpen())
    {
        // Provera da li je korisnik poslao signal programu
        sf::Event event;
        while (window.pollEvent(event))
        {
            // Korisnk kliknuo X da zatvori prozor
            if (event.type == sf::Event::Closed)
                window.close();
        }

        window.clear();// Brisanje prethodnog frejma
        window.draw(shape);// Iscrtavanje objekta
        window.display();// Crtanje novog frejma
    }
    return 0;
}
```

6

Програмирање

6.1 Струтура кода програма

Главни фајл је Source.cpp и у њему се налази main() као и главна петља за сртање. Хедер фајлови написани за програм су ivica.h, kugla.h и matrica.h. У њима се декларисане класе "ивица" и "кугла", као и структура "матрица". Методе су дефинисане у посебним фајловима: ivica.cpp, kugla.cpp и matrica.cpp.

6.2 Source.cpp

У Source.cpp дефинишу се низ тачака које су типа sf::Vector2f и низови објеката типа кугли и ивица. Потм се функцијама за иницијализацију додељују предефинисане вредности позиција ивица, као о рупа.
Такође су у овом фајлу дефинисана функција за исцртавање стола, функција за проверу да ли је неко место валидно као и функција за исцртавање помоћне линије правца приликом циљања.

Главна петља програма изгледа слично као и пример једноставног СФМЛ програма из претходног поглавља. Прво се прварава да ли је притиснуто неко од дугмића на тастатури (Погледајте поглавље "Како користити програм" за све прчице на тастатури). Потом се у колико су кугле заустављене проверава позиција миса којим се помера се штап. Такође се проверава стање точкића миша, пошто се њиме бира јачина ударца. Када корисник притисне леви клик, позива се метода класе кугла за додељивање брзине за белу куглу.

Када се заврши провера улазно-излазних уређаја врши се пролаз низ кугли и освежавају се њихове позиције и брзине. Након тога проверава се да ли је

дошло до судара између кугли, или са ивицама стола. Прво се брише претходни фрејм. Потом се врши цртање стола, ивица и кугли. На крају сваког пролаза кроз главну петљу приказује се слика на екран.

6.3 Класа Ивица

Једноставна класа која описује ивицу. Методе за исцртавање се позивају из Source.cpp, док се методе за правац и раздаљину од неке тачке користе за проверу судара из класе кугла.

```
#include <SFML/Graphics.hpp>
#include <cmath>

extern sf::Vector2f pozicija_stola, dimenzije_stola, senka_vektor;
extern sf::Color boja_senke;

class ivica
{
public:
   sf::Vector2f tacka1, tacka2, pravac;
   float k, n;
   sf::VertexArray linija, senka;
   sf::RenderWindow* prozor;

   ivica()
   {
      prozor = NULL;
      linija = sf::VertexArray(sf::Lines, 2);
      senka = sf::VertexArray(sf::Quads, 4);
   }
   vo..id povezi_grafiku(sf::RenderWindow* prozor1) { this->prozor =
       prozor1; }
   void podesi(sf::Vector2f t1,sf::Vector2f t2);
   float razdaljina_od(sf::Vector2f A);
   sf::Vector2f getPravac() { return pravac; }
   void crtaj();
   void crtaj_senku();
};
```

6.4 Структура Матрица

Ова структура је коришћена због уредности кода унутар класе кугла. Дефинисан је оператор множења над структуром матрица, па се поједностављује код писан унутар класе кугла.

```
#define VELICINA_MATRICE 10

extern float** memorija_nova;
float** inicijalizuj_memoriju();

struct matrica
{
   int kolona,red;
   float** mat;

   matrica();
   void dodeli(int,int,float**);
   matrica operator * (const matrica druga);
   void operator = (const matrica druga);
};
```

Како би се смањио број непотребних алокација и дислокација у меморији (изазваним великим бројем позивања из класе кугла), када се ради са матрицама ротације коришћена је иста меморија.

6.5 Класа Кугла

```
class kugla
{
   //promenljive
   sf::Vector2f brzina;//vektor brzine sa vrednostima x i y
   sf::Vector2f pozicija;//kordinate centra kugle x i y
   sf::Vector2f prethodna_pozicija;//kordinate centra kugle x i y
   matrica mat_rotacije, mat_drotacije, xyz;
   bool bio_sudar;//1 u koliko je prethodni frejm bio sudar, da se ne bi
      ponovno pozivale sudar_... funkcije
   bool u_igri;//1 u koliko je kugla na stolu tj. u igri je, u koliko je
      upala u rupu 0
   bool bila_u_igri;//1 u koliko je prethodni udarac kogla bila u igri
   bool oznacena;//1 kako bi se iscrtavalo precrtan znak
```

```
   bool animacija;

   //konstante
   int red_br;
   float masa, poluprecnik, trenje;

   //grafika
   sf::Image slika;//cuva teksturu kugle
   sf::CircleShape krug, kruzic, senka;
   sf::RenderWindow* prozor;
   sf::VertexArray pointmap;

public:
   //metode
   ...
```

6.5.1 Метода Освежи

Помера куглу, и додељује кугли нову брзину. Рачуна матрицу ротације, за угао који је ротирала.

```
void kugla::osvezi()
{
   float usporenje=0.9;
   if (u_igri && !animacija)
   {
      sf::Vector2f nova_brzina = brzina * (1.f - 9.81f * trenje/60);
      pozicija += (brzina+nova_brzina)/(2.f*120.f);
      brzina = nova_brzina * (0.f + (intenzitet(brzina) > 5.f));

      if(intenzitet(brzina)!=0.f)
      {
         sf::Vector2f osa=rotiraj(brzina, PI/2.f);
         osa/=intenzitet(osa);
         float ugao,ux=osa.x,uy=osa.y,uz=0;
         ugao=intenzitet((brzina+nova_brzina)/(2.f*120.f))/poluprecnik;

         //Pravljenje matrice, za koliko se kugle okrenula
         float c=cosf(ugao),s=sinf(ugao);
```

```
        mat_drotacije.mat[0][0]=c+ux*ux*(1.f-c);
        mat_drotacije.mat[0][1]=ux*uy*(1.f-c)-uz*s;
        mat_drotacije.mat[0][2]=ux*uz*(1.f-c)+uy*s;
        mat_drotacije.mat[1][0]=uy*ux*(1.f-c)+uz*s;
        mat_drotacije.mat[1][1]=c+uy*uy*(1.f-c);
        mat_drotacije.mat[1][2]=uy*uz*(1.f-c)-ux*s;
        mat_drotacije.mat[2][0]=uz*ux*(1.f-c)-uy*s;
        mat_drotacije.mat[2][1]=uz*uy*(1.f-c)+ux*s;
        mat_drotacije.mat[2][2]=c+uz*uz*(1.f-c);

        //Nova matrica se dobija mnozenjem pomeraja sa starom rotacijom
        mat_rotacije=mat_rotacije*mat_drotacije;
     }
   }
}
```

6.5.2 Метода Цртај

Пролази корз све пикселе из квадрата дужине странице једнаке пречнику кугле, чји је центар такође центар масе кугле. За све пикселе који су унутар круга прво се проналази z координата, па се применом матрице рорације добијају нове координате које одговарају позицији те тачке на неротираној кугли. Како бисмо порнашли боју те тачке пребацујемо у сферне координате. Маповањем добијених вредности на текстуру добијамо боју тражене тачке. Процес се понавља за следећи пиксел.

```
void kugla::crtaj()//iscrtavanje
{
   //deklarisanje i podesavanje niza piksela (pozicija i boja)
   int kx,ky,rd_br,x,y,r=poluprecnik;
   float s=0,t,d,x1=1,y1=1,z1=1,z;
   //Za svaki piskel u kvadratu stranice 2r
   for (x = -r; x < r; x++)
      for (y = -r; y < r; y++)
      {
         //Proveri da li je u krogu poluprecnika r
         d = (float)sqrt(x * x + y * y);
         if (poluprecnik >= d)
         {
            rd_br = (int)(x + poluprecnik + (y + poluprecnik) * poluprecnik
```

```
                * 2.f);
            //Podesi piksel sa koordinatama
            pointmap[rd_br].position = sf::Vector2f((float)x, (float)y) +
                pozicija + pozicija_stola;

            //Odredi z koordinatu (Znamo da je x^2+y^2+z^2=r^2 i z>0)
            z=sqrtf(poluprecnik*poluprecnik-x*x-y*y);

            //Koordinate na rotiranoj kugli
            xyz.mat[0][0]=x;
            xyz.mat[1][0]=y;
            xyz.mat[2][0]=z;

            //Mnozenjem dobijamo koordinate na nerotiranoj kugli
            xyz=mat_rotacije*xyz;

            x1=xyz.mat[0][0];
            y1=xyz.mat[1][0];
            z1=xyz.mat[2][0];

            //Prebacivanje iz koordinatnog sistema u sferni
            t=acosf(z1/poluprecnik);

            if(x1>0) s=atanf(y1/x1);
            if(x1<0) s=atanf(y1/x1)+PI;
            if(x1==0) s=PI/2.f;
            s=PI*2.f-s;

            //Pronaci koordinate te tacke na teksturi
            kx = (int)((s / 2.f / PI) * slika.getSize().x) %
                slika.getSize().x;
            ky = (int)((t /PI) * slika.getSize().y) % slika.getSize().y;

            //Podesi boju piksela
            pointmap[rd_br].color = slika.getPixel(kx,ky);
        }
    }
  //iscrtavanje na prozor/ekran
  prozor->draw(pointmap);
}
```

7

Додатни алати и платформе

7.1 Мејкфајл

Компајловање овог програма на линуксу се врши помоћу команде:

```
g++ -std=c++14 -g -O2 -Wall -o program.out *.cpp -I SFML-2.5.1\include -L
   SFML-2.5.1\lib -lsfml-graphics -lsfml-window -lsfml-system
```

Да не би памтили ову дугачку команду и убрзали компајловање програма приликом малих измена у коду користио сам алат Мејк *(Make)*. Мејк је алатка за аутоматизацију којом се прави, компајлује или покреће програм. Правила су дефинисана у фајлу са називим "Makefile".

```
CXX=g++
MKDIR=mkdir -p
RM=rm -rf
GDB=gdb

CXXFLAGS=-std=c++14 -O2 -Wall
LDFLAGS=-lsfml-graphics -lsfml-window -lsfml-system

BUILD_DIR=build
OBJECTS=Source.o kugla.o ivica.o matrica.o
OBJECT_FILES=$(patsubst %,$(BUILD_DIR)/%,$(OBJECTS))

BINARY=bilijar.out

.PHONY: all run debug clean
```

```
all: compile

$(BUILD_DIR)/%.o: %.cpp
   $(CXX) $(CXXFLAGS) -MMD -MP -c $< -o $@

$(BINARY): $(OBJECT_FILES)
   $(CXX) $(CXXFLAGS) $(OBJECT_FILES) -o $@ $(LDFLAGS)

compile:
   $(MKDIR) $(BUILD_DIR)
   $(MAKE) $(BINARY)

run: compile
   ./$(BINARY)

debug: $(BINARY)
   $(GDB) $(BINARY)

clean:
   $(RM) $(BUILD_DIR) $(BINARY)
```

7.2 Гит

Гит је алат за контролу верзија софтвера, који сам такође користио, како би пратио измене. Гит омогућава лако враћање одрђене сачуване верзије као и организован начин рада више људи на истом пројекту.

7.3 Гитхаб

Гитхаб је платформа која омогућава хостовање гит репозиторијума. Поред тога гитхаб омогућава свакоме да пошаље предлоге или измене, које власник репозиторијума може прихватити или одбити. Гитхаб је највећа платформа на којој се производе пројекти отвореног кода.

Овај пројекат, као и све новије верзије пројекта, можете пронаћи на овој гитхаб страници.

7.4 Аутоматизација

Како би остао у конкуренцији, гитхаб је омогућио својим корисницима да аутоматизују неке радње како би убрзали или отклонили грешке у коду. Када се догоди нека од предефинисаних акција на репозиторијуму на гитхабовом серверу се покреће машина на којој се извршавају задате команде.

Пример аутоматизације на мом пројекту је да се свака означена (тагована) верзија која у анзиву има "v*" (* је било који број) прво компајлује, па ако нема грешака компајловани фајлови се запакују и поставе на гитхаб страницу са које се могу преузети.

```
name: Release
on:
  push:
    tags:
      - "v*"
jobs:
  buildwin:
    name: BuildWin
    runs-on: "ubuntu-latest"
    steps:
      - name: GitCheckout
        uses: actions/checkout@v2
        with:
          ref: master
      - name: Instalacija
        run: sudo apt install g++-mingw-w64-x86-64
      - name: MingwVerzija
        run: x86_64-w64-mingw32-g++ -v
      - name: DownloadSFML
        run: |
          wget https://www.sfml-dev.org/files/
              /SFML-2.5.1-windows-gcc-7.3.0-mingw-64-bit.zip
          unzip SFML-2.5.1-windows-gcc-7.3.0-mingw-64-bit.zip
          mv SFML-2.5.1 sfml
      - name: Compile
        run: |
          x86_64-w64-mingw32-g++ -DSFML_STATIC -std=c++14 -O2 -o
              bilijar.exe src/*.cpp -I sfml/include -L sfml/lib
              -lsfml-graphics-s -lsfml-window-s -lsfml-system-s -lopengl32
              -lwinmm -lgdi32 -static-libstdc++ -static-libgcc
      - name: Archive
```

```
      run: |
        mv bilijar.exe src
        cd src
        tar -czf bilijar-wind0ws-x86_64.tar.gz bilijar.exe resources
        zip -r bilijar-wind0ws-x86_64.zip bilijar.exe resources
    - name: Release
      uses: softprops/action-gh-release@v1
      if: startsWith(github.ref, 'refs/tags/')
      with:
        files: |
          src/bilijar-wind0ws-x86_64.tar.gz
          src/bilijar-wind0ws-x86_64.zip
buildlinux:
  name: BuildLinux
  runs-on: "ubuntu-latest"
  steps:
    - name: GitCheckout
      uses: actions/checkout@v2
      with:
        ref: master
    - name: Instalacija
      run: sudo apt install libsfml-dev
    - name: GppVerzija
      run: g++ -v
    - name: Compile
      run: |
        cd src
        make
    - name: Archive
      run: |
        cd src
        tar -czf bilijar-linux-x86_64.tar.gz bilijar.out resources
        zip -r bilijar-linux-x86_64.zip bilijar.out resources
    - name: Release
      uses: softprops/action-gh-release@v1
      if: startsWith(github.ref, 'refs/tags/')
      with:
        files: |
          src/bilijar-linux-x86_64.tar.gz
          src/bilijar-linux-x86_64.zip
```

8

Закључак

Узевши у обзир величину кода програма врло је вероватно да су се догодиле неке грешке. Али пошто је овај пројекат на гитхабу, све грешке се могу исправити, тако да тамо увек можете пронаћи најновију верзију. Уз аутоматизацију овај процес је убрзан, а код проверен.

Саветујем свакоме ко прави програме да користи гит и да свој код поставе на платформама као што су гихаб и гилаб. Ови алати отварају могућност рада више људи и тиме повећавају квалитет кода.

Искористите могућности које вам ово време нуди.

Литература

[1] Наташа Чалуковић *Физика за први разред Математичке Гимназије*

[2] Милан Митровић, Срђан Огњановић, Михаило Вељковић, Љубинка Петковић и Ненад Лазаревић *Геометрија за за први разред Математичке Гимназије*

[3] https://www.sfml-dev.org/

[4] https://sr.wikipedia.org/wiki/

[5] https://github.com/actions