Федеральное агентство связи

Сибирский Государственный Университет Телекоммуникаций и Информатики

СибГУТИ

Кафедра ПМ иК

КУРСОВАЯ РАБОТА

по дисциплине «Объектно-ориентированное программирование»

Тема: «Солнечная система»

Выполнили: студентки 2 курса группы ИП-916

Адова А.С.

Александрова А.С.

Проверил: доцент кафедры

ПМиК
Ситняковская Е.И

Новосибирск, 2020

Содержание

1. Постановка задачи.

Задача состояла в написании модели Солнечной системы. Планеты должны двигаться вокруг солнца с определенной скоростью, также предусматривается наличие пояса астероидов. Код проекта должен быть удобочитаемым. Должны быть соблюдены принципы SOLID. Проект должен быть написан с помощью ООП. Программа пишется на языке программирования c++. Для реализации графики используется graphics.h.

2. Технологии ООП

1. В данной задаче используется перегрузка конструкторов
2. Используются конструкторы
3. Списки инициализации
4. Абстрактный класс (subject)
5. Вложенный класс
6. Использование объектов в качестве аргументов

3. Программная реализация

Абстрактный класс – subject. Содержит два виртуальных метода. Происходит перегрузка конструктора.

```
class subject{
    protected:
        int color;
        float x, y, speed;
        int xcenter;
        int ycenter;
    public:
        subject(){
            xcenter=0;
            ycenter=0;
        }
        subject(int d,int ch){
            xcenter=d/2;
            ycenter=ch/2;
     }
        virtual void otris()=0;
        virtual void anime()=0;
};
```

Класс star. Отвечает за объект Солнце. Методы наследуются от абстрактного класса.

```
class star:public subject{
    private:
        char *name;
        int a, b;
        int j, i;
        float R;
    public:
        star(int rada, int radb, float skor, int cvet, float raz, char *nazv, int d, int ch):subject(d,ch), i(1), j(1){
            a=rada;
            b=radb;
            speed=skor;
            color=cvet;
            R=raz;
            name=nazv;
            x=xcenter+a;
            y=ycenter+sqrt(pow(b,2)*((pow(a,2)-pow((x-xcenter),2))/pow(a,2)));
        }
        star(float koor1, float koor2, int cvet, float raz, char *nazv, int d, int ch):subject(d, ch), a(0), b(0){
            x=xcenter-koor1;
            y=ycenter-koor2;
            color=cvet;
            R=raz;
            name=nazv;
            speed=0;
        }
```

Переопределяется метод otris (который отвечает за прорисовку объекта)

```
 void otris() override{
     int i=0;
     setcolor(color);
     circle(x, y, R);
     setfillstyle(1, color);
     floodfill(x,y,color);
     setcolor(0);
     setbkcolor(color);
     outtextxy(x-13, y-7, name);
     setbkcolor(0);
     setcolor(color);
     luchiki();
 }
```

Переопределяется метод anime (который отвечает за изменение координат объекта)

```
void anime() override{
    if(j==1){
        if((x>xcenter-a) && (x-speed>xcenter-a)){
            i=-1;
        }
        else{
            j=0;
            i=1;
        }
    }
    if(j==0){
        if((x<xcenter+a) && (x+speed<xcenter+a)){
            i=1;
        }
        else{
            i=-1;
            j=1;
        }
    }
    if(speed>0){
        x=x+(speed*i);
        y=pow(b,2)*((pow(a,2)-pow((x-xcenter),2))/pow(a,2));
        y=sqrt(y);
        y=(y*i)+ycenter;
    }
}
```

Прорисовываются лучики у Солнца.

```
void luchiki(){
    moveto(x+R+5,y);
    lineto(x+R+40,y);
    moveto(x-R-5,y);
    lineto(x-R-40,y);

    moveto(x,y+R+5);
    lineto(x,y+R+40);
    moveto(x,y-R-5);
    lineto(x,y-R-40);

    moveto(x+R-2.5,y+R-2.5);
    lineto(x+R+30,y+R+30);
    moveto(x-R+2.5,y-R+2.5);
    lineto(x-R-30,y-R-30);

    moveto(x+R-2.5,y-R+2.5);
    lineto(x+R+30,y-R-30);
    moveto(x-R+2.5,y+R-2.5);
    lineto(x-R-30,y+R+30);

    moveto(x+R+5,y-R+15);
    lineto(x+R+25,y-R+5);
    moveto(x-R-5,y-R+15);
    lineto(x-R-25,y-R+5);
    moveto(x-R-5,y+R-15);
    lineto(x-R-25,y+R-5);
    moveto(x+R+5,y+R-15);
    lineto(x+R+25,y+R-5);

    moveto(x+R-15,y-R-5);
    lineto(x+R-7.5,y-R-20);
    moveto(x-R+15,y-R-5);
    lineto(x-R+7.5,y-R-20);

    moveto(x-R+15,y+R+5);
    lineto(x-R+7.5,y+R+20);
    moveto(x+R-15,y+R+5);
    lineto(x+R-7.5,y+R+20);
}
```

Класс planet. Наследник абстрактного класса. Осуществляется логика планеты. //belt-булевая переменная для реализации колец Сатурна.

```
class planet:public subject{
    private:
        int a, b, kol, *dal;
        char *name;
        int i, j;
        float R;
        bool belt;
    public:
        planet(int rada, int radb, float skor, int cvet, float raz, char *nazv, int d, int ch, bool poyas):subject(d,ch),
            int k;                                                                                          i(1), j(1){
            float koor1, koor2, s;
            a=rada;
            b=radb;
            speed=skor;
            color=cvet;
            R=raz;
            name=nazv;
            belt=poyas;
            x=xcenter+a;
            y=ycenter+sqrt(pow(b,2)*((pow(a,2)-pow((x-xcenter),2))/pow(a,2)));
            if(belt){
                kol=rand()%(6-1+1)+1;
                dal=new int [kol];
                for(k=0;k<kol;k++){
                    dal[k]=rand()%(15-3+1)+3;
                }
            }
        }
```

Переопределяющиеся методы класса

```
void otris() override{
    int g;
    setcolor(color);
    circle(x, y, R);
    setfillstyle(1, color);
    floodfill(x,y,color);
    if(belt){
        setcolor(8);
        for(g=0;g<kol;g++){
            circle(x,y,R+dal[g]);
        }
        setcolor(15);
        setbkcolor(0);
        outtextxy(x-12, y-R-33, name);
    }
    else{
        setcolor(15);
        setbkcolor(0);
        outtextxy(x-12, y-R-17, name);
    }
}

void anime() override{
    if(j==1){
        if((x>=xcenter-a) && (x-speed>=xcenter-a)){
            i=-1;
        }
        else{
            j=0;
            i=1;
        }
    }
    if(j==0){
        if((x<=xcenter+a) && (x+speed<=xcenter+a)){
            i=1;
        }
        else{
            i=-1;
            j=1;
        }
    }
    if(speed>0){
        x=x+(speed*i);
        y=pow(b,2)*((pow(a,2)-pow((x-xcenter),2))/pow(a,2));
        y=sqrt(y);
        y=(y*i)+ycenter;
    }
}
```

Класс kam. Отвечает за логику пояса астероидов.

```
class kam:public subject{
    private:
        int a, b;
        float R;
        int i, j;
        int koltpoint;
        class tpoint{
```

Вложенный класс tpoint. Отвечает за логику астероида.

```
class tpoint{
    private:
        float x2, y2;
        int i, j;
        int a2, b2;
    public:
        tpoint(){
        }
        tpoint(float x, float y, int z, int f, int a, int b): i(z), j(f){
            x2=x;
            y2=y;
            a2=a;
            b2=b;
        }
        void otrist(float R, int color){
            setcolor(color);
            circle(x2, y2, R);
        }
```

```
void animet(int xcenter, int ycenter, float speed){
    if(j==1){
        if((x2>=xcenter-a2) && (x2-speed>=xcenter-a2)){
            i=-1;
        }
        else{
            j=0;
            i=1;
        }
    }
    if(j==0){
        if((x2<=xcenter+a2) && (x2+speed<=xcenter+a2)){
            i=1;
        }
        else{
            i=-1;
            j=1;
        }
    }
    if(speed>0){
        x2=x2+(speed*i);
        y2=pow(b2,2)*((pow(a2,2)-pow((x2-xcenter),2))/pow(a2,2));
        y2=sqrt(y2);
        y2=(y2*i)+ycenter;
    }
}
```

Создается массив из объектов класса tpoint.

```
    tpoint *chastica;
public:
    kam(int rada, int radb, float skor, int d, int ch, int kol1, int r1, int r2):subject(d,ch),i(1),j(1), R(1){
        int s, g;
        a=rada;
        b=radb;
        srand(time(NULL));
        speed=skor;
        color=8;
        koltpoint=kol1;
        chastica = new tpoint [koltpoint];
        x=xcenter+a;
        for(s=0;s<koltpoint;s++){
            g=rand()%(r1-(r2)+1)+(r2);
            a=g+rada;
            b=g+radb;
            if(j==1){
                if((x>xcenter-a) && (x-speed>xcenter-a)){
                    i=-1;
                }
                else{
                    j=0;
                    i=1;
                }
            }
```

```
    if(j==0){
        if((x<xcenter+a) && (x+speed<xcenter+a)){
            i=1;
        }
        else{
            i=-1;
            j=1;
        }
    }
    if(speed>0){
        x=x+(5*i);
        y=pow(b,2)*((pow(a,2)-pow((x-xcenter),2))/pow(a,2));
        y=sqrt(y);
        y=(y*i)+ycenter;
        chastica[s]=tpoint(x,y, i, j, a, b);
    }
}
```

Класс model. Создает модель Солнечной системы. Использовует объекты класса в качестве аргументов.

```
class model{
    public:
        system();
        void show(subject *object){
            object->otris();
        }
        void dvij(subject *object){
            object->anime();
        }
};
```

main. Создаются объекты классов.

```
int main(){
    int d=1280, ch=864, page=0, i;
    model solar;
    char *s1="Sun";
    star Sun(15, 0, 14, 25,s1, d, ch);
    s1="Mercury";
    planet Mer(50, 30, 6.5, 7, 2, s1, d, ch, false);
    s1="Venus";
    planet Ven(75, 45, 5, 12, 4, s1, d, ch, false);
    s1="Earth";
    planet Ear(100, 60, 3.5, 2, 4, s1, d, ch, false);
    s1="Mars";
    planet Mars(137, 82, 2.5, 4, 3, s1, d, ch, false);
    s1="Jupiter";
    planet Jup(312, 187, 0.5, 6, 10, s1, d, ch, false);
    s1="Saturn";
    planet Sat(412, 287, 0.15, 13, 9, s1, d, ch, true);
    s1="Uran";
    planet Uran(512, 387, 0.1, 9, 7, s1, d, ch, false);
    s1="Neptune";
    planet Nep(612, 487, 0.1, 3, 7, s1, d, ch, false);
    kam asteroid1(212, 137, 1, d, ch, 300, 20, -20);
```

Реализуется логика Солнечной системы.

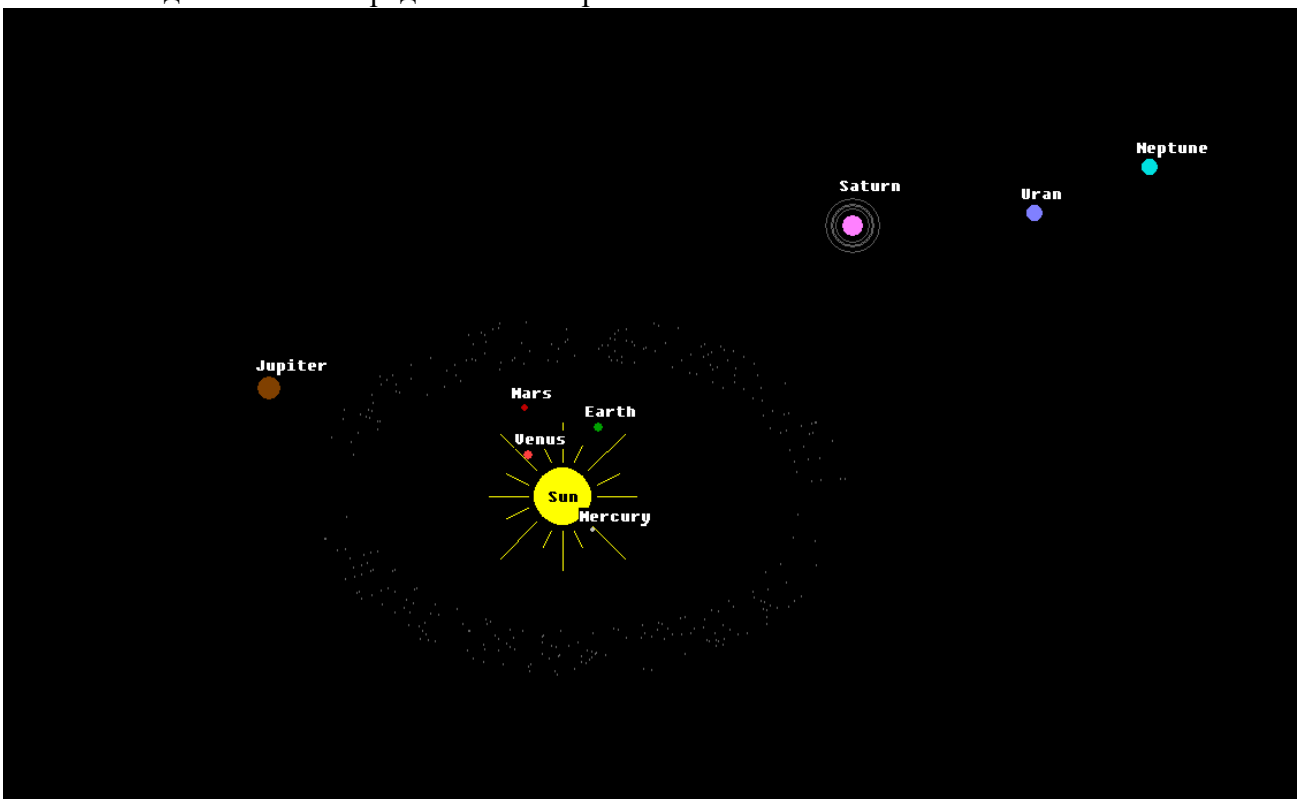
```
initwindow(d,ch);
while(!kbhit()){
    setactivepage(page);
    setvisualpage(1-page);
    cleardevice();
    solar.show(&Sun);
    solar.show(&Mer);
    solar.show(&Ven);
    solar.show(&Ear);
    solar.show(&Mars);
    solar.show(&Jup);
    solar.show(&Sat);
    solar.show(&Uran);
    solar.show(&Nep);
    solar.show(&asteroid1);

    solar.dvij(&Mer);
    solar.dvij(&Ven);
    solar.dvij(&Ear);
    solar.dvij(&Mars);
    solar.dvij(&Jup);
    solar.dvij(&Sat);
    solar.dvij(&Uran);
    solar.dvij(&Nep);
    solar.dvij(&asteroid1);
    page=1-page;
    delay(10);
}

    closegraph();
    system("PAUSE");
    return 0;
}
```

4. Результаты работы

Планеты солнечной системы вращаются вокруг солнца по эллиптичной орбите.

Показан пояс астероидов между Марсом и Юпитером.

Планеты двигаются с определенной скоростью.



5. Заключение

В итоге данная курсовая рабоота включает в себя множество различных технологий, пройденных за этот курс. Демонстрирует понимание и усвоение материала. Отвечает всем требованиям. Использует принципы SOLID и другие принципы хорошего кода.