Struktura danych

```
class Przeszkody
{

public:

    virtual bool czy_kolizja(std::shared_ptr <Przeszkody>
&p)=0;
    virtual std::string get_nazwa() const=0;
    virtual void zapis()=0;
    virtual Vector<3> wez_srodek() =0 ;
    virtual double promien()=0;
    virtual double wez_promien()=0;
};
```

Bryła Geometryczna

-NazwaPliku_BrylaWzorcowa std::string
-NazwaPliku_NazwaFinalna std::string
-katOX Bryły, katOY Bryly, katOZ bryly

-move(Vector<3> V) – translacja o zadany wektor
-obrot(kat) obrot bryly o zadany kat,
-obrotP obrot dla prostopadloscianu wzgledem jego srodka
ObrotW obort dla graniastoslupa wzgledem jego srodka
ObrotW1 obrot graniastoslupa wzgledem srodka prostopadloscianu
Zapis() zapis do pliku;

```
    double promien(){return sqrt(pow(srodek[0]-pkt1[0][0],2) +
pow(srodek[1]-pkt1[0][1],2)+pow(srodek[2]-pkt1[0][2],2));}

    Vector<3> wez_srodek() ;

    Vector<3> get_srodek() const ;

    bool czy_kolizja(std::shared_ptr <Przeszkody> &p);

    // bool czy_kolizja(std::shared_ptr <Przeszkody> p) const;
     double wez_promien(){return bpromien;}

    void set_srodek(Vector<3> srodek);
```

```
class PPlaskowyz : public BrylaGeometryczna
{

double ppromien;
public:
    PPlaskowyz(){};

    PPlaskowyz(Vector<3> pkt, double h, double w,
double d, string nazwa_pliku,
string nazwa_pliku_do_zapisu);

      Vector<3> wez_srodek();
      double wez_promien(){return ppromien;}
};
```

```
class Ostroslup : public BrylaGeometryczna
{
    protected:
    int nrprzeszkody;
    double opromien;
    // Vector<3> srodek;
    public:
    Ostroslup(Vector<3> pkt,double dlugosc ,
double szerokosc, double wysokosc,
string nazwa_pliku, string nazwa_pliku_do_zapisu);
    // ~Ostroslup();
        Vector<3> wez_srodek() ;
        double wez_promien(){return opromien;}

};
```

Graniastoslup6

Konstruktor Graniastoslupa

Operator <<

```
class Gran : public BrylaGeometryczna
{
    protected:
    int nrprzeszkody;
    double gpromien;
    public:
    Gran(Vector<3> pkt,double dlugosc ,double szerokosc,
 double wysokosc, string nazwa_pliku,
 string nazwa_pliku_do_zapisu);
    // ~Ostroslup();
        Vector<3> wez_srodek();
        double wez_promien(){return gpromien;}

};
```

Prostopadloscian

Konstruktor Prostopadloscian

Vector<3> wez_srodek();
double wez_promien()

Operator <<

Scena

```
double katOZ; // obrot do okola osi Z
    double katOY; // kat wznoszenia i opadania
    double katOX; // kat rotacji np. wirnikow

    vector<Vector<3>> scr1;
    Vector<3> Przesuniecie;
    Vector<3> srodek;
    string nazwa_pliku;
    string nazwa_pliku_do_zapisu;
    int k=0;// ilosc punktów na prostej
```

Konstruktor sceny
Operacja poruszania scena
Zapis () zapis do pliku

```
Vector<3> &operator [](int i){return scr1[i];}
    Vector<3> operator [](int i)const {return scr1[i];}
```

Dron

Vector bazowy dla 1 drona;
Vector bazowy dla 2 drona;
Nr drona 1; Nr drona 2;
Prostopadloscian *korpus;
Graniastoslup *wirniki[4];
Prostopadloscian *korpus1;
Graniastoslup *wirniki1[4];

Konstruktor dronów;
Ruch operacja poruszania drone,;
Obrot operacja obrotu calego drona;
AnimacjaLotuDrona zlozona z poszczegolnych skladowych ruch i obrot o odpowiednie wspolzedne;
Zwiad2 – dron wykonuje zwiad o zadany promien;
Wspolrzedne – podaje aktuwalne wspolzedne wybranego drona
bool czy_kolizja(shared_ptr <Przeszkody> p);

```
    Vector<3> get_srodek() ;
```