Grafika komputerowa

Sprawozdanie z laboratorium 3

Temat: Przekształcenia geometryczne

Student: Bartosz Baniak
Nr albumu: 80512
Nr w grupie: 1

Grupa: WCY21IJ2S1
Daty laboratoriów: 27.11.2023 r.
Prowadzący: dr inż. Marek Salamon

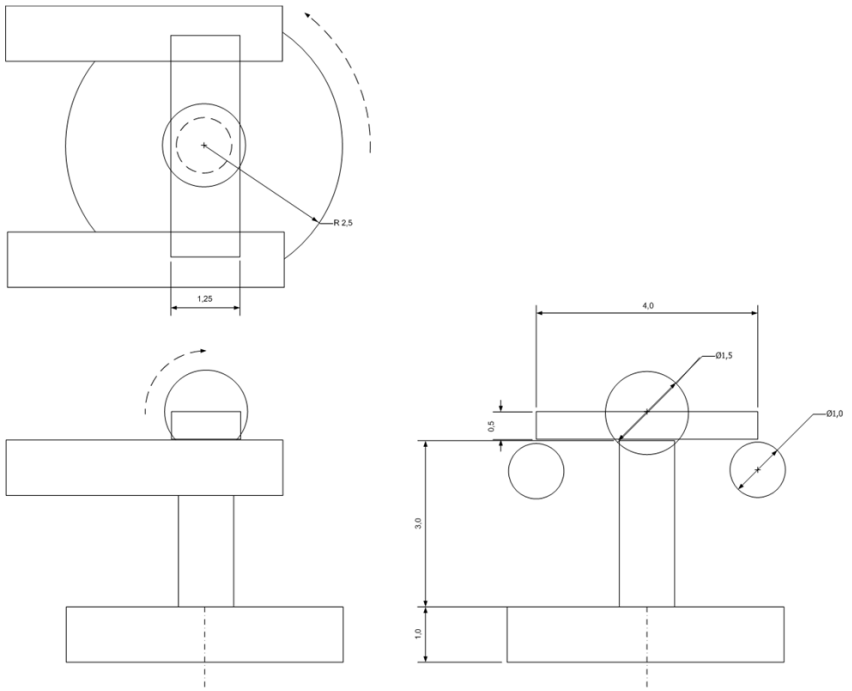
1. Treść zadania

1. Wykorzystując projekt „**robot**” napisać fragment programu:
   a) umożliwiający zmianę odległości obserwatora od obiektu w zakresie (*odlmin*, *odlmax*); 0.5 pkt.
   b) umożliwiający zmianę orientacji obserwatora (bez ograniczeń) względem osi *0X, 0Y i 0Z*. 0.5 pkt.
2. Przesunąć obiekt z projektu „**robot**” w położenie (0.0, 0.0, -100.0) i wykorzystując funkcję ***glutSolidCube*** oraz funkcje działające na stosie macierzy modelowania napisać program odpowiedzialny za utworzenie obiektu, którego rzuty przedstawiono na rysunkach w Zad.1. 1 pkt.
3. Wykorzystując funkcję ***glutWireSphere*** oraz funkcje działające na stosie macierzy modelowania napisać program odpowiedzialny za dynamiczne tworzenie sceny przedstawiającej układ słoneczny zgodnie z parametrami przedstawionymi w Zad.2 poruszający się wokół obiektu z pkt.2, po orbicie o promieniu 30 z prędkością kątową 0.25 stopni/klatkę w kierunku CW. Zaznaczyć orbity poruszających się obiektów. Bieguny generowanych sfer powinny leżeć w osi pionowej. 2 pkt.
4. Scenę uzupełnić o:
   a) planetę o promieniu 6 poruszająca się po orbicie o środku w (0,0,0) i promieniu 40 z prędkością kątową 0.15 stopnia/klatkę w kierunku CCW w płaszczyźnie nachylonej do osi OX pod kątem 30 stopni
   b) planetę o promieniu 10 poruszająca się po orbicie o środku w (0,0,0) i promieniu 60 z prędkością kątową 0.5 stopnia/klatkę w kierunku CW w płaszczyźnie nachylonej do osi OX pod kątem -45 stopni;
   Zaznaczyć orbity poruszających się obiektów. Bieguny sfer powinny leżeć w osi pionowej orbity.
   a) 1 pkt., b) 1pkt.
5. Przesunąć utworzoną scenę w położenie (0.0, 50.0, 0.0) i napisać program przedstawiający obiekt zbudowany z prymitywów przestrzennych udostępnianych przez biblioteki ***GLU*** i ***GLUT***. /zadanie indywidualne oceniane na podstawie sprawozdania/ 2 pkt.
   Ocena końcowa: 4 pkt. DST, 5 pkt. DST+, 6 pkt. DB, 7 pkt. DB+, 8 pkt. BDB

Zadanie indywidualne:

Napisać program przedstawiający obiekt zbudowany z prymitywów przestrzennych udostępnianych przez biblioteki *GLU* i *GLUT*. Użytkownik za pomocą klawiatury powinien mieć możliwość wprowadzania zmian następujących parametrów:

1. Kąta obrotu wyrzutni w płaszczyźnie poziomej,
2. Kąta podniesienia wyrzutni w płaszczyźnie pionowej,



W programie uwzględnić możliwość interakcyjnej zmiany położenia obserwatora poprzez podanie następujących parametrów:

1. Odległości obserwatora od obiektu,
2. Kąta obrotu wokół obiektu w zakresie [0, 360].

UWAGA: Obserwator jest zawsze zwrócony przodem w kierunku obiektu.

2. Sposób rozwiązania

Wykorzystywane transformacje i funkcje:
glTranslatef() – funkcja przesuwa obiekt wzdłuż osi X, Y i Z w trójwymiarowej przestrzeni. Przyjmuje trzy argumenty: wartości przesunięcia dla każdej osi;
glRotatef() –funkcja obraca obiekt wokół określonej osi o określony kąt. Przyjmuje cztery argumenty: kąt obrotu oraz trzy wartości określające osie;
glScalef() – funkcja skaluje obiekt wzdłuż osi X, Y i Z w trójwymiarowej przestrzeni. Przyjmuje trzy argumenty: wartości skali dla każdej osi;
glutSolidCube() – funkcja służąca do generowania prostopadłościanu w trójwymiarowej przestrzeni;
gluCylinder() – funkcja rysuje walcowy kształt w trójwymiarowej przestrzeni. Tworzy ona cylindryczny obiekt o zadanej wysokości, promieniach górnym i dolnym oraz ilości segmentów uzytych do zdefiniowania jego kształtu;
gluDisk() – funkcja rysuje płaski dysk w trójwymiarowej przestrzeni. Umożliwia tworzenie płaskich okrągłych kształtów o określonym promieniu i ilości segmentów użytych do ich definiowania;
glutWireCube() – funkcja rysująca sześcian w przestrzeni trójwymiarowej jako siatkę linii, czyli obrys sześcianu
glutWireSphere() – funkcja rysująca sferę w przestrzeni trójwymiarowej jako siatkę linii, czyli obrys sfery. Tworzy ona sferę o zadanym promieniu.

1) Aby umożliwić zmianę odległości obserwatora od obiektu wykorzystałem glTranslatef znajdujące się w funkcji void WyswietlObraz(); - w tym celu podmieniłem wartość -40 wartością, która poprzez wykorzystywanie przycisków z klawiatury (‘w’ oraz ‘s’) odpowiedzialna będzie za zmianę odległości (odlObserwatora).
   W celu zmiany orientacji obserwatora bez ograniczeń dla osi X i Y usunąłem warunek ograniczający obroty wokół tych osi. Adekwatnie postąpiłem dla osi Z, dodając na początku odpowiednie klawisze odpowiedzialne za obrót względem tej osi.
2) W celu przesunięcia robota wykorzystałem funkcję glTranslatef na początku funkcji void RysujRamieRobota();
   Aby utworzyć obiekt, którego rzuty zostały przedstawione w Zad.1. wykorzystałem funkcje glutSolidCube() oraz odpowiednie funkcje działające na stosie macierzy modelowania. Wszystkie funkcje odpowiedzialne za utworzenie obiektu napisałem w funkcji void RysujWieze();
3) W funkcji void RysujUklad() napisałem odpowiednie funkcje i transformacje, które rysują układ planetarny w trójwymiarowej przestrzeni zgodnie z parametrami podanymi w Zad.2 poruszający się wokół obiektu z pkt.2. Wykorzystałem przemieszczanie i obracanie obiektów w przestrzeni. Najpierw przesuwam układ współrzędnych, a następnie rysuję orbity słońca i planet. Kolejno dodaję planety i ich księżyce, obracając każdy z nich wokół innych obiektów lub układu. Za każdym razem, gdy wywołuję tę funkcję, planeta krąży wokół słońca, a księżyc wokół planety. Dodatkowo, aktualizuję kąty obrotu, co zapewnia płynną animację ruchu planet.
4) W funkcji void RysujPlanety() napisałem odpowiednie funkcje, które rysują dwie planety w przestrzeni trójwymiarowej. Wykorzystałem przemieszczanie, obracanie i skalowanie, by stworzyć efekt przestrzeni trójwymiarowej. Pierwsza planeta jest manipulowana, przesuwana i obracana wokół swojej osi. Kolejno, wykorzystując analogiczne operacje, tworzę drugą planetę, podlegającą przesunięciom, obrotowi i skalowaniu. Obydwie planety rotują wokół punktu orbitowania, co symuluje ich ruch

po orbicie. Na koniec, wartości kątów obrotu są aktualizowane, co nadaje płynność ich ruchowi wzdłuż orbit.

5) W celu przesunięcia całej sceny zastosowałem w każdej funkcji odpowiednie przesunięcie każdego elementu za pomocą funkcji glTranslatef().
Następnie tworzę funkcję RysujWyrzutnie(), która zawiera sekwencje operacji transformacyjnych i rysowania obiektów graficznych, które tworzą wyrzutnię. W celu wytworzenia wyrzutni wykorzystuję różne transformacje takie jak glRotatef(), glTranslatef(), glScalef() oraz funkcje takie jak gluCylinder(), gluDisk(), glutWireCube() i glutWireSphere().
Pierwszym krokiem było utworzenie podstawy wyrzutni za pomocą funkcji gluCylinder() oraz gluDisk().
Drugim krokiem było utworzenie słupa wyrzutni za pomocą funkcji gluCylinder() oraz odpowiednie jego przemieszczenie za pomocą transformacji glTranslatef() i glRotatef().
Trzecim krokiem było narysowanie ramienia, na którym będą znajdowały się armaty wyrzutni. Samo ramię znajduje się na wcześniej utworzonym słupie.
Czwartym krokiem było narysowanie obu armat znajdujących się pod krańcowymi punktami ramienia.
Piątym krokiem było narysowanie głowy armaty, która znajduje się na ramieniu w jego centralnym punkcie.
W celu utworzenia animacji ruchu dla armaty w pionie i poziomie zastosowałem transformacje glRotate() w odpowiednich miejscach w funkcji. Aby armata mogła się poruszać w poziomie wykorzystałem wyżej wymienioną funkcję na początku funkcji, zaraz po przygotowaniu stosu macierzy modelowania. Funkcji glRotate() użyłem również przed rysowaniem jej górnej części (ramienia, głowy i armat), aby mogła się poruszać ona w pionie. Funkcja ta poprzedzona była funkcją glTranslatef() w celu wyznaczenia osi, wokół której będzie się ona poruszać w pionie.

3. Wyniki

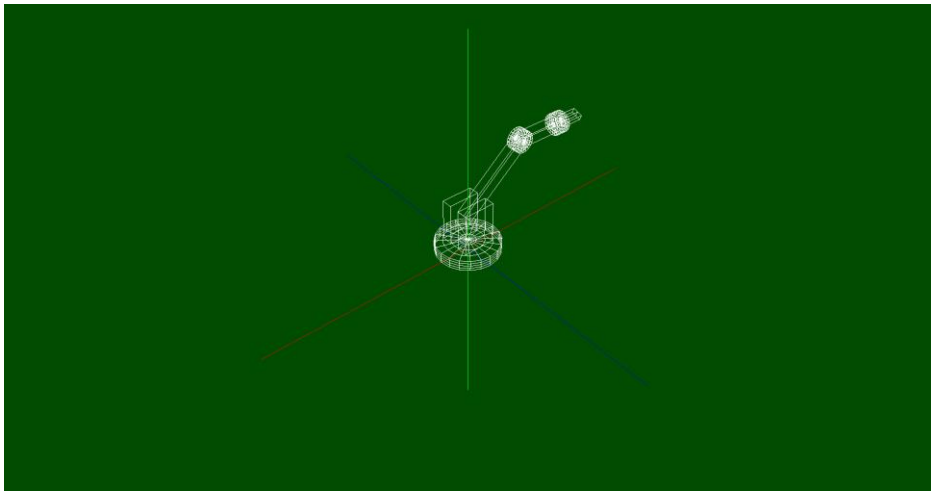
1) KOD:

```
// Wyznaczenie polozenia obserwatora (przeksztalcenie uladu wspolrzednych
// sceny do ukladu wspolrzednych obserwatora).
glTranslatef(0, 0, odlObserwatora);
glRotatef(rotObsX, 1, 0, 0);
glRotatef(rotObsY, 0, 1, 0);
glRotatef(rotObsZ, 0, 0, 1);
```

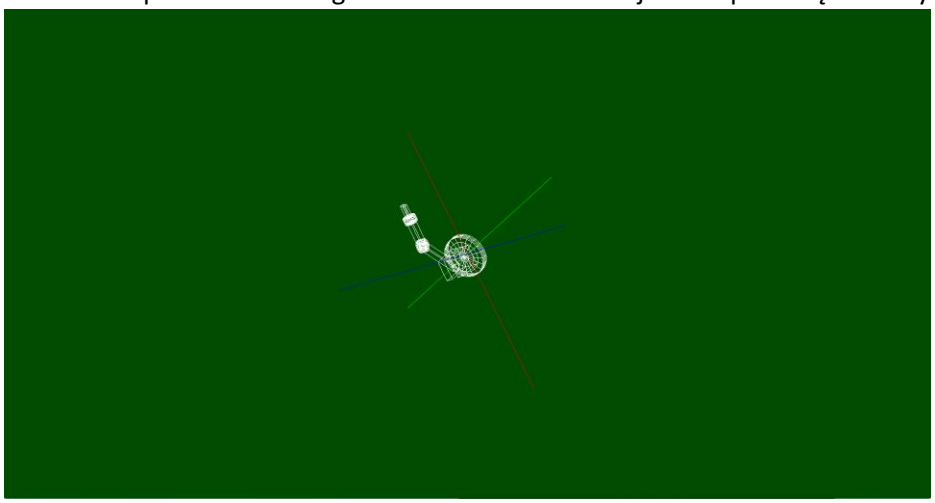
```
////////////////////////////////////////////////////////////////////////////////
// Funkcja klawiszy specjalnych
void ObslugaKlawiszySpecjalnych(int klawisz, int x, int y)
{
    switch (klawisz)
    {
    case GLUT_KEY_UP:
        rotObsX = rotObsX + 1.0;
        break;

    case GLUT_KEY_DOWN:
        rotObsX = rotObsX - 1.0;
        break;

    case GLUT_KEY_LEFT:
        rotObsY = rotObsY - 1.0;
        break;

    case GLUT_KEY_RIGHT:
        rotObsY = rotObsY + 1.0;
        break;
    }
}
```

```
case 'a':
    rotObsZ = rotObsZ - 1.0;
    break;

case 'd':
    rotObsZ = rotObsZ + 1.0;
    break;
```

DZIAŁANIE:
Początkowe ustawienie po włączeniu programu:



Ustawienie po zmianie odległości obserwatora i rotacji osi za pomocą klawiszy:



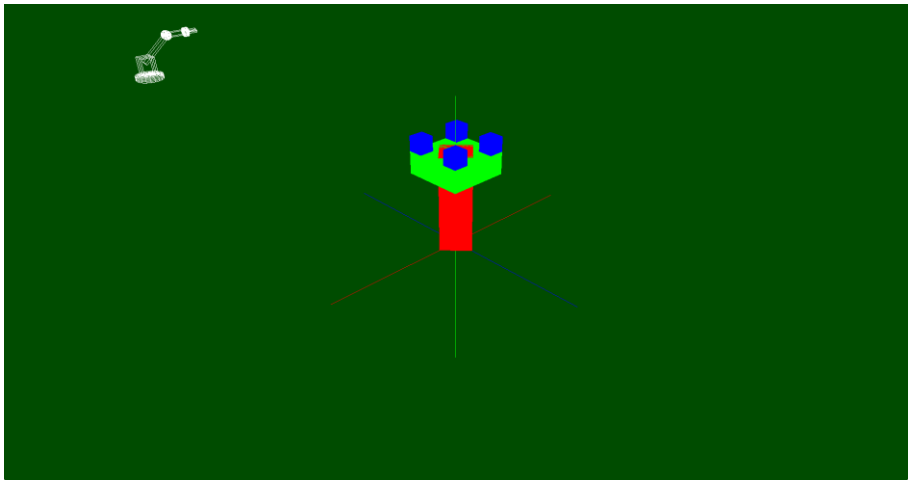
2) KOD:

```
glTranslatef(0.0, 0.0, -100.0); //Przesunięcie robota
```

```
void RysujWieze() {

    glPushMatrix();
    //glTranslatef(0.0, 50.0, 0.0);//Przesunięcie wieży

    //Rysowanie zielonego
    glPushMatrix();

    // Przesunięcie prostopadłościanu względem pierwszego prostopadłościanu
    glTranslatef(0.0, 11.5, 0.0);

    // Ustawienie koloru na zielony
    glColor3f(0.0, 1.0, 0.0);

    // Skalowanie prostopadłościanu na wymiary 8x3x8
    glScalef(8.0, 3.0, 8.0);

    // Rysowanie prostopadłościanu za pomocą glutSolidCube
    glutSolidCube(1.0);
    glPopMatrix();


    //Rysowanie czerwonego
    glPushMatrix();

    glTranslatef(0.0, 6.5, 0.0);
    glColor3f(1.0, 0.0, 0.0);
    glRotatef(45.0, 0.0, 1.0, 0.0);
    // Skalowanie prostopadłościanu
    glScalef(4.24, 13.0, 4.24);

    // Rysowanie prostopadłościanu za pomocą glutSolidCube
    glutSolidCube(1.0);

    glPopMatrix();
```

```
    //Rysowanie Niebieskich wież
    glPushMatrix();

    glTranslatef(3.0, 14.0, -3.0);
    glColor3f(0.0, 0.0, 1.0);
    glScalef(2.0, 2.0, 2.0);
    glutSolidCube(1.0);

    glTranslatef(-3.0, 0.0, 0.0);
    glColor3f(0.0, 0.0, 1.0);
    glutSolidCube(1.0);

    glTranslatef(0.0, 0.0, 3.0);
    glColor3f(0.0, 0.0, 1.0);
    glutSolidCube(1.0);

    glTranslatef(3.0, 0.0, 0.0);
    glColor3f(0.0, 0.0, 1.0);
    glutSolidCube(1.0);

    glPopMatrix();

    glPopMatrix();

    glColor3f(1.0, 1.0, 1.0);
}
```

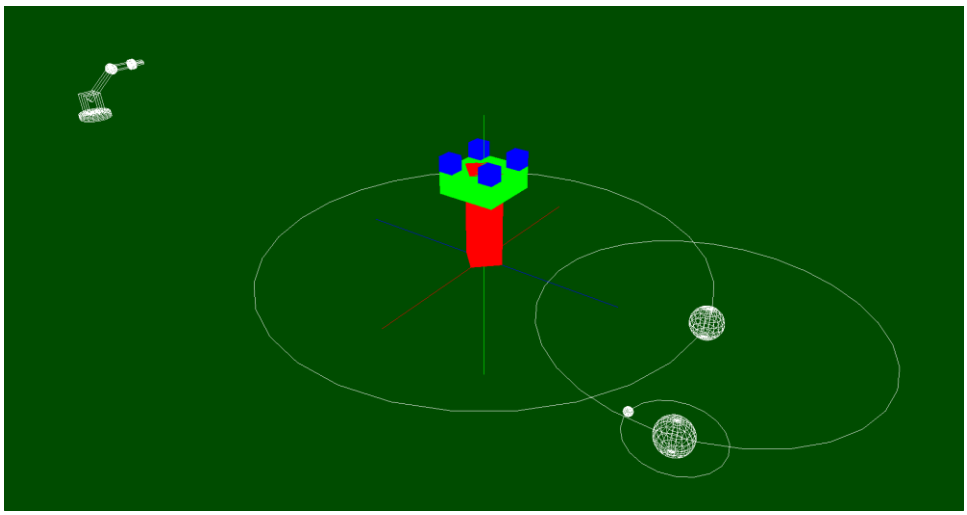
DZIAŁANIE:



3) KOD:

```
void RysujUklad() {

    glPushMatrix();
    //glTranslatef(0.0, 50.0, 0.0); //Przesunięcie wieży

    glColor3f(1.0, 1.0, 1.0);

    glRotatef(90, 1, 0, 0);
    gluDisk(orbitaSlonca, 30.0, 30.0, 30, 4);
    glRotatef(-90, 1, 0, 0);

    glRotatef(katSloncaWzglObiektu, 0.0, 1.0, 0.0);
    glTranslatef(30.0, 0.0, 0.0);
    glRotatef(90, 1, 0, 0);
    glutWireSphere(2.0, 20, 10);


    gluDisk(orbitaPlanety, 20.0, 20.0, 30, 4);

    glPushMatrix();

    glRotatef(90, 1, 0, 0);
    glRotatef(katPlanetyWzglSlonca, 0.0, -1.0, 0.0);
    glTranslatef(20.0, 0.0, 0.0);
    glRotatef(katPlanety, 0, -1, 0);
    glRotatef(90, 1, 0, 0);
    glutWireSphere(2.0, 20, 10);


    gluDisk(orbitaKsiezyca, 5.0, 5.0, 20, 4);
    glRotatef(-90, 1, 0, 0);

    glPushMatrix();
    glRotatef(katKsiezycaWzglPlanety, 0.0, -1.0, 0.0);
    glTranslatef(5.0, 0.0, 0.0);
    glRotatef(90, 1, 0, 0);
    glutWireSphere(0.5, 20, 10);
    glPopMatrix();

    glPopMatrix();

    glPopMatrix();


    glRotatef(-90, 1, 0, 0);

    katSloncaWzglObiektu = katSloncaWzglObiektu - 0.25;
    katPlanetyWzglSlonca = katPlanetyWzglSlonca - 0.25;
    katPlanety++;
    katKsiezycaWzglPlanety = katKsiezycaWzglPlanety - 0.25;
}
```

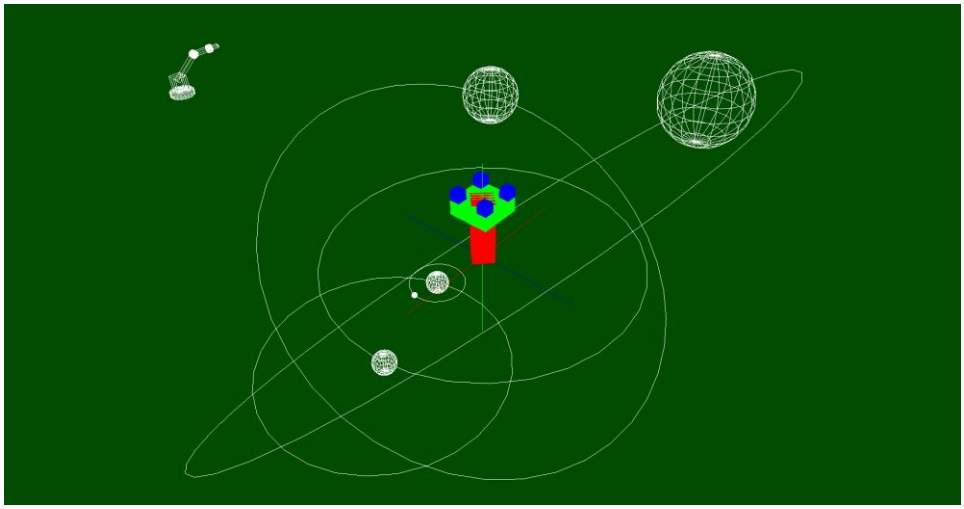
DZIAŁANIE:



4) KOD:

```
void RysujPlanety() {
    glPushMatrix();
    //glTranslatef(0.0, 0.0, 50.0); //Przesunięcie planety 1
    glRotatef(30.0, 1, 0, 0);
    gluDisk(orbitaPlanety, 40.0, 40.0, 40, 4);
    glRotatef(-30.0, 1, 0, 0);
    glRotatef(-60.0, 1, 0, 0);
    glTranslatef((40.0 * cos(katPlanetyA)), 0.0, (40.0 * sin(-katPlanetyA)));
    glRotatef(60.0, 1, 0, 0);
    glutWireSphere(6.0, 20, 10);
    glPopMatrix();

    glPushMatrix();
    //glTranslatef(0.0, 0.0, 50.0); //Przesunięcie planety 2
    glRotatef(-45.0, 1, 0, 0);
    gluDisk(orbitaPlanety, 60.0, 60.0, 40, 4);
    glRotatef(45.0, 1, 0, 0);
    glRotatef(45.0, 1, 0, 0);
    glTranslatef((60.0 * cos(katPlanetyB)), 0.0, (60.0 * sin(-katPlanetyB)));
    glRotatef(-45.0, 1, 0, 0);
    glutWireSphere(10.0, 20, 10);
    glPopMatrix();



    katPlanetyA = katPlanetyA + 0.00261799;
    katPlanetyB = katPlanetyB + 0.00872665;
}
```

DZIAŁANIE:



5) KOD:

```
void RysujWieze() {

    glPushMatrix();
    glTranslatef(0.0, 50.0, 0.0);//Przesunięcie wieży
```

```
void RysujUklad() {

    glPushMatrix();
    glTranslatef(0.0, 50.0, 0.0); //Przesunięcie układu
```

```
void RysujPlanety() {
    glPushMatrix();
    glTranslatef(0.0, 0.0, 50.0); //Przesunięcie planety 1
```

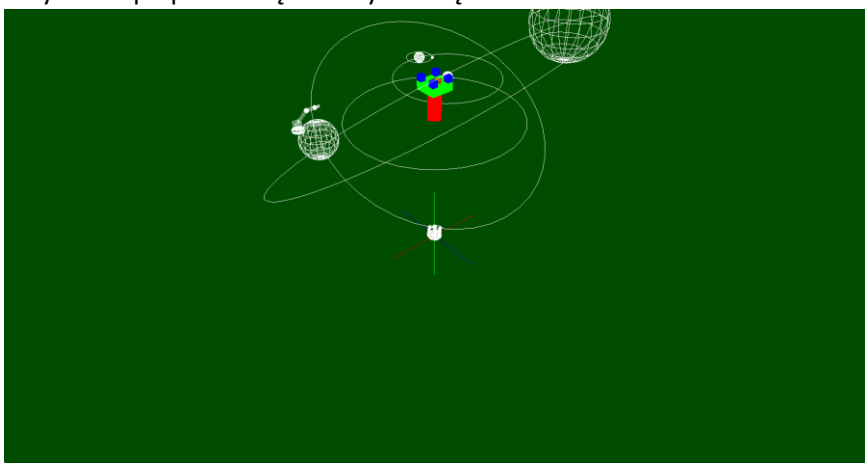
```
    glPushMatrix();
    glTranslatef(0.0, 0.0, 50.0); //Przesunięcie planety 2
```

```
void RysujWyrzutnie() {

    glPushMatrix(); //cała macierz wyrzutni

    glRotatef(90, 1, 0, 0);
    glRotatef(-45.0, 0, 1, 0);
    glRotatef(podstawaWyrzutni, 0, 1, 0);

    //Rysowanie dolnej podstawy wyrzutni
    glPushMatrix();
    glRotatef(-90.0, 1, 0, 0);
    //sciany boczne
    gluCylinder(podstawaSciany, 2.5, 2.5, 1.0, 20, 4);

    //gorna podstawa
    gluDisk(podstawaDyskG, 0.0, 2.5, 20, 4);

    //dolna podstawa
    glTranslatef(0.0, 0.0, 1.0);
    gluDisk(podstawaDyskG, 0.0, 2.5, 20, 4);
    glPopMatrix();

    //Rysowanie słupa wyrzutni bez podstaw
    glPushMatrix();
    glTranslatef(0.0, 1.0, 0.0);
    glRotatef(-90, 1, 0, 0);
    gluCylinder(podstawaSciany, 0.5, 0.5, 3.0, 20, 4);
    glTranslatef(0.0, 4.0, 0.0);
    glPopMatrix();

    glTranslatef(0.0, 4.5, 0.0);
    glRotatef(armatyWyrzutni, 1, 0, 0);
```

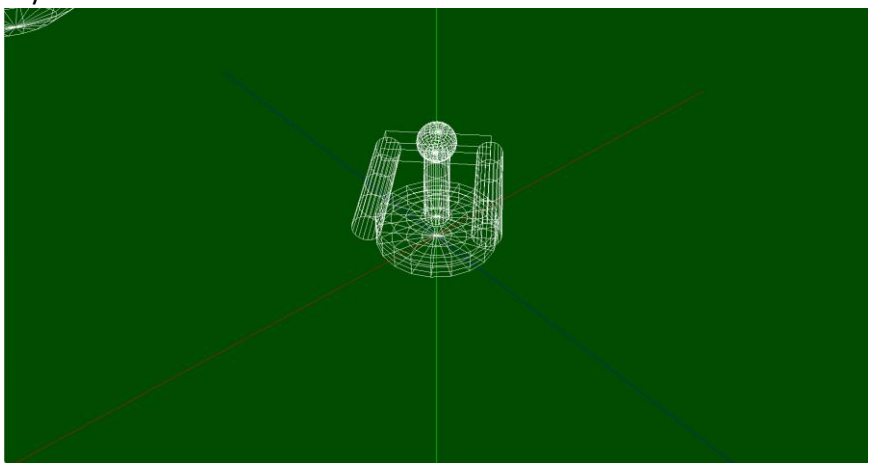
```
    //Rysowanie ramienia
    glPushMatrix();
    glTranslatef(0.0, -0.25, 0.0);
    glScalef(4.0, 0.5, 1.25);
    glutWireCube(1);
    glPopMatrix();

    //Rysowanie armaty 1
    glPushMatrix();
    glTranslatef(-2.0, -1.0, -1.0);
    gluCylinder(podstawaSciany, 0.5, 0.5, 5.0, 20, 4);
    glPopMatrix();

    //Rysowanie armaty 2
    glPushMatrix();
    glTranslatef(2.0, -1.0, -1.0);
    gluCylinder(podstawaSciany, 0.5, 0.5, 5.0, 20, 4);
    glPopMatrix();

    //Rysowanie głowy
    glPushMatrix();
    glutWireSphere(0.75, 20, 20);
    glPopMatrix();


    glPopMatrix(); //Koniec macierzy wyrzutni

}
```

DZIAŁANIE:

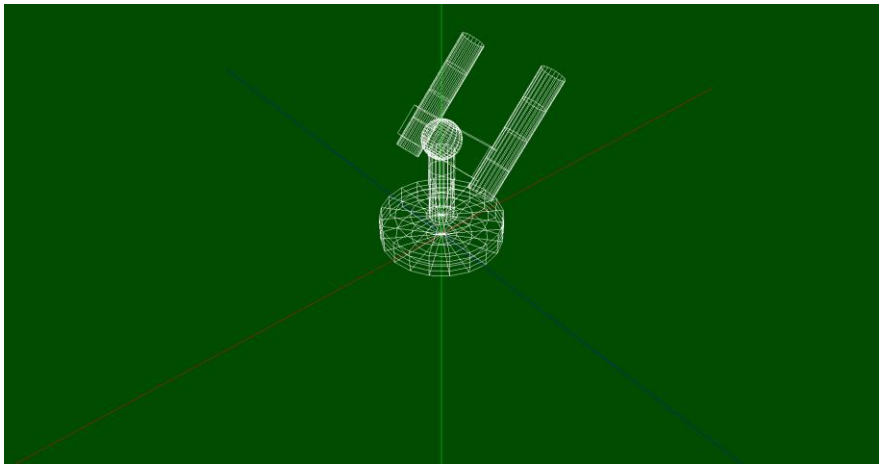
Cały układ po przesunięciu z wyrzutnią w centrum:



Wyrzutnia:



Ruch poziomy i pionowy wyrzutni:



4. Wnioski

Podczas pracy nad tym kodem w środowisku OpenGL eksplorowałem techniki renderowania grafiki trójwymiarowej. Poznałem zasady transformacji obiektów, takie jak translacja, rotacja i skalowanie, co pozwoliło mi manipulować różnymi elementami sceny. Dodatkowo, zdobyłem wiedzę na temat zarządzania stanem sceny, reakcji na zdarzenia użytkownika oraz animowania obiektów w czasie rzeczywistym. To pozwoliło mi lepiej zrozumieć proces tworzenia interaktywnych aplikacji 3D oraz umiejętność projektowania i manipulowania obiektami w trójwymiarowej przestrzeni.