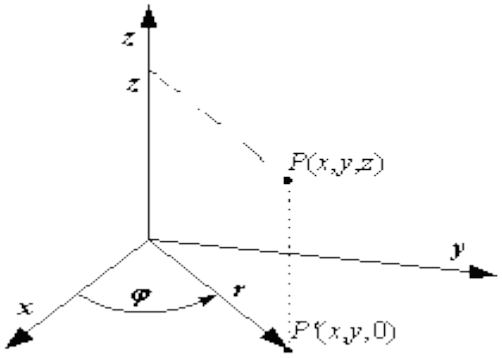
Programowanie Obiektowe i Grafika
Sprawozdanie z projektu

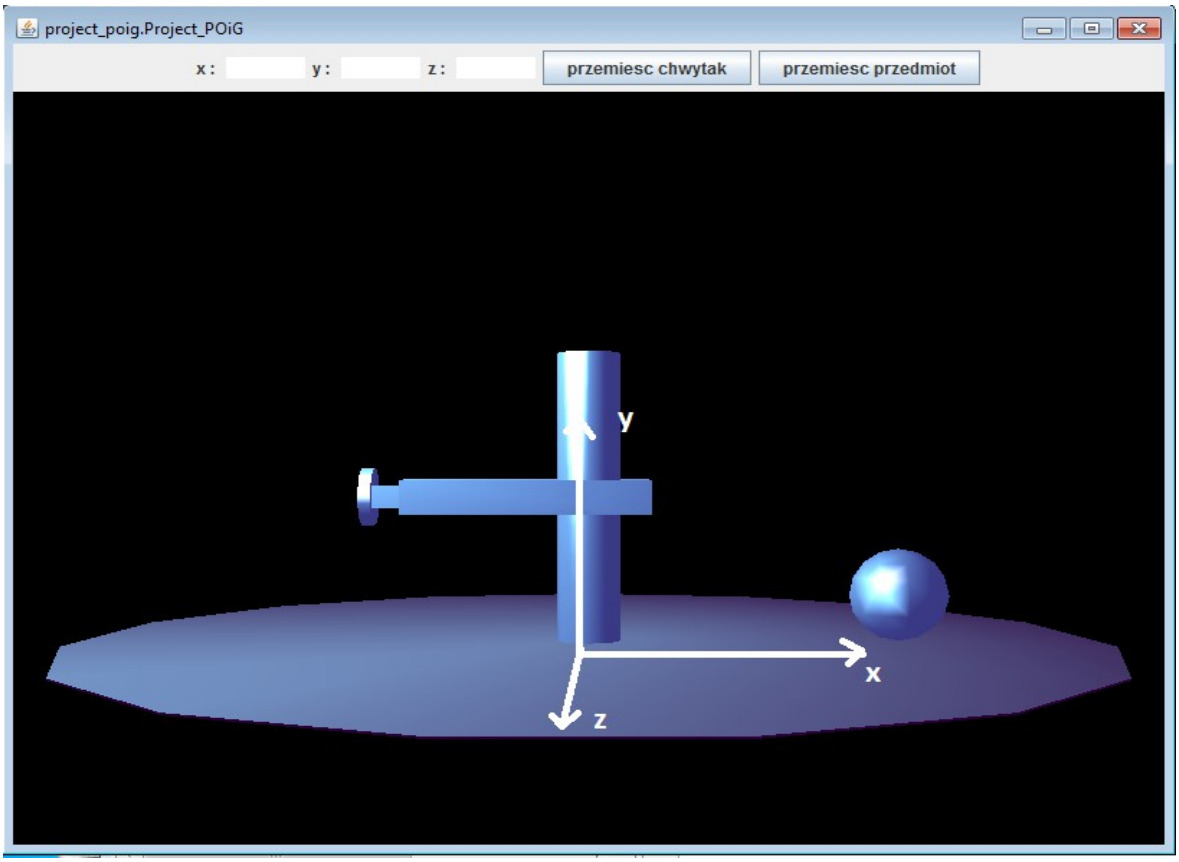
Wykonawcy :
Szymon Broda 171652 AiR1
Piotr Pietruszka 171842 AiR1

Robot typu „Cylindrical arm”

Celem naszego projektu jest animowana wizualizacja ramienia robota typu „cylindrical arm” (zadanie nr 3). Robot posiada trzy elementy zapewniające nam trzy stopnie swobody – rotor, które może obracać się wokół pionowej osi, ramię mogące poruszać się w górę i w dół oraz ramię wysuwane które pozwala na zwiększenie zasięgu. Takie działanie robota pozwala nam dotrzeć w dowolne miejsce w przestrzeni cylindrycznej.

Położenie chwytaka można opisać za pomocą walcowego układu współrzędnych, gdzie obrót wokół osi pionowej odpowiada zmianie współrzędnej φ, przemieszczenie wzdłuż tej samej osi – zmianie współrzędnej 'z', a zwiększenie promienia – zmianie współrzędnej 'r'.



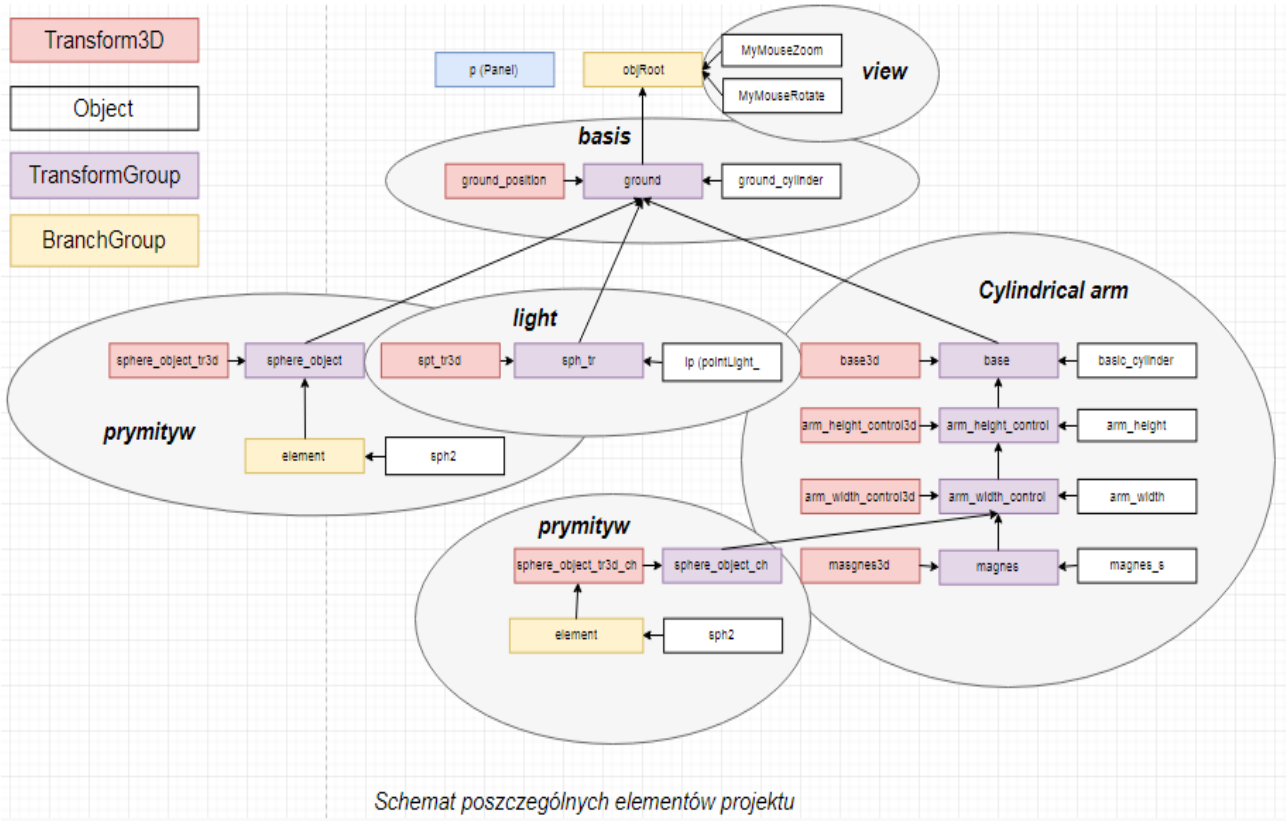


Ogólne informacje na temat działania programu

Zaimplementowana przez nas konstrukcja robota posiada trzy sterowane przez użytkownika elementy- rotor,który może obracać się wokół pionowej osi (przyciski **'z'**, **'x'**),ramię poruszające się w górę i w dół (przyciski **'w', 's'**). oraz ramię wysuwane, zwiększające zasięg robota (przyciski **'a','d'**). Na końcu ramienia umieszczony jest chwytak działający na zasadzie elektromagnesu. Po naciśnięciu przycisku **'p'** magnes zostaje uruchomiony i jeśli prymityw znajduje się w zakresie działania jest on przyciągany do chwytaka. W celu dostosowania odległości oraz kąta widzenia obserwatora możemy posługiwać się myszką, natomiast aby osiągnąć powrót kamery do pozycji początkowej odpowiedzialny jest przycisk **'r'.** Program posiada również funkcję numerycznego wprowadzania położenia prymitywu **('przemieść przedmiot')** i położenia robota **('przemieść robot')** oraz funkcję nagrywania (przycisk **'k')** i odtwarzania (przycisk **'l')** wcześniej zapamiętanej trajektorii. Funkcje te zostaną omówione dokładniej w dalszej części sprawozdania.

Struktura i hierarchia programu

W naszym programie zastosowaliśmy kaskadowe połączenie obiektów TransformGroup, tak aby kolejne elementy poczynając od podstawy zawierały niezbędne transformacje służące do uzyskania przez chwytak wymaganej pozycji. Wszystkie obiekty typu TransformGroup zostały wyróżnione kolorem fioletowym, natomiast ich transformacje – kolorem czerwonym. Kolor biały odpowiada za poszczególne elementy klasy Object. W schemacie zostały również zaznaczone poszczególne grupy elementów odpowiadające za daną część projektu. Chmura *'view'* odpowiada za przesunięcie oraz rotację obserwatora, *'basis'* – za podstawę na której wykonywana jest akcja programu, *'light'* odpowiedzialne jest za oświetlenie, *'Cylindrical arm'* odpowiada schematowi działania robota, natomiast chmura *'prymityw'* prezentuje zasady operacji na przedmiocie. BranchGroup '*element* ' jest w przypadku podnoszenia i opuszczania obiektu dodawany i usuwany z TransformGroup'a *sphere_object,* opisującego jego położenie na ziemi i *sphere_object_ch* odnoszącego się do jego pozycji przy chwytaku.



Schemat poszczególnych elementów projektu

Nagrywanie i odtwarzanie ruchów robota

Robot posiada możliwość nagrywania oraz odtwarzania wcześniej wykonanej trajektorii. Aby aktywować tą funkcję należy nacisnąć przycisk **'k'** z klawiatury, ponowne jego naciśnięcie zakończy nagrywanie trasy. Naciśnięcie przycisku **'l'** spowoduje powrót robota do pozycji sprzed rozpoczęcia nagrywania, a następnie odtworzenie zapisanej trajektorii krok po kroku. Podczas wykonywania rekonstrukcji trasy chwytak aktywowany jest w tych samych momentach co podczas nagrywania, więc jeśli prymityw znajdzie się w zasięgu magnesu, w czasie gdy magnes jest aktywowany, przedmiot zostanie podniesiony. Następnie kolejne polecenia będą wykonywane zgodnie z nagraną trajektorią. W trakcie odtwarzania trasy robot i otoczenie stają się niewrażliwe na wszelkie komendy, dopiero po zakończeniu wykonywania tej operacji, możemy kontynuować sterowanie konstrukcją.

Numeryczne wprowadzanie położenia robota i sfery

W programie znajdują się pola tekstowe służące do wprowadzania położenia chwytaka robota i elementu interaktywnego we współrzędnych kartezjańskich (oś X skierowana w prawo, oś Y pionowo w górę, oś Z od ekranu). Jeśli podane wartości są spoza zakresu pracy robota, przemieści się on możliwie najbliżej (stosunek X i Z wyznaczający kąt obrotu zostanie zachowany, ramię zostanie maksymalnie (lub minimalnie) wysunięte, a w zależności od współrzędnej Y, ramię osiągnie maksymalną lub minimalną wysokość). W przypadku elementu współrzędna Y jest pomijana (zawsze będzie znajdował się na wysokości ziemi), a w razie przekroczenia granic położenia (zbyt duża lub zbyt mała odległość od środka układu) nie zostanie on przemieszczony.

Wybrane fragmenty kodu

1) Podnoszenie elementu:

Konieczne jest znalezienie współrzędnych chwytaka i kuli w układzie współrzędnych związanych z ziemią, w celu uzyskania ich względnej odległości. W przypadku kuli mamy jedną transformację, informacje o której zapisujemy do macierzy 4x4 m4__sphere. Dla chwytaka musimy wziąć pod uwagę 4 transformacje (base, arm_height_control, arm_width_control, magnes), które przemnożone przez siebie w odpowiedniej kolejności zapisujemy do macierzy 4x4 m4_chwytak (zawiera ona informacje o wypadkowej translacji i orientacji chwytaka). Trzy pierwsze wiersze z ostatniej kolumny macierzy 4x4 opisują translację względem początku układu, czyli położenie środka elementu (chwytaka). Można więc łatwo obliczyć ich względną odległość i ograniczyć zasięg działania magnesu.

```
  if(!picked_up)
  {
    //polozenie sfery
    sphere_object.getTransform(tr5);
    Matrix4f m4_sphere = new Matrix4f();
    tr5.get(m4_sphere);

    //polozenie chwytaka
    base.getTransform(tr1);
    arm_height_control.getTransform(tr2);
    arm_width_control.getTransform(tr3);
    magnes.getTransform(tr4);

    sphere_object_tr3d.setIdentity();
    sphere_object_tr3d.mul(tr1);
    sphere_object_tr3d.mul(tr2);
    sphere_object_tr3d.mul(tr3);
    sphere_object_tr3d.mul(tr4);

    Matrix4f m4_chwytak = new Matrix4f();
    sphere_object_tr3d.get(m4_chwytak);

    if((Math.pow(m4_sphere.m03-m4_chwytak.m03 ,2) + Math.pow(m4_sphere.m13-m4_chwytak.m13, 2)
          +Math.pow(m4_sphere.m23-m4_chwytak.m23, 2)  <  1.5))
    {
      sphere_object.removeChild(element);
      sphere_object_ch.addChild(element);
      picked_up=!picked_up;
    }
  }
```

2) Numeryczne współrzędne chwytaka:

Podane numerycznie współrzędne kartezjańskie są zapisywane do zmiennych x_desired, y_desired, z_desired. Poruszanie się robota jest wykonywane iteracyjnie z wykorzystaniem funkcji zapewniającej obrót, zmianę wysokości i wydłużenie w sterowaniu ręcznym, aż do osiągnięcia pożądanej wartości z odpowiednią dokładnością. Najpierw zmieniany jest kąt obrotu robota. Wartość pożądana jest wyznaczona przy użyciu funkcji atan2, a jej argumentami są współrzędne x_desired i z_desired. Kąt obecny również jest wyznaczany za pomocą funkcji atan2, gdzie argumentami są sinus i cosinus aktualnego kąta obrotu (uzyskane jako odpowiednie elementy macierzy translacji i obrotu 4x4 'matrix'). Wykorzystanie funkcji arcsin lub arccos nie byłoby wystarczające, bo kąt należy do przedziału (-pi, pi).
Podobnie zmieniane są wysokość i wysunięcie robota. Tam wartościami pożądanymi są odpowiednio: y_desired i promień (sqr(x_desired^2 + z_desired^2)). Są również wprowadzone ograniczenia zapobiegające przekroczeniu granicznych położeń.

```
x_desired = Double.parseDouble(wsp_x.getText());
y_desired = Double.parseDouble(wsp_y.getText()) - 1.8;
z_desired = Double.parseDouble(wsp_z.getText());
//obrot
base3d.get(matrix);
for(int i = 0; i < 100 && (Math.atan2(matrix.m02, matrix.m00) > Math.atan2(z_desired, -x_desired) + Math.PI/32 ||
      Math.atan2(matrix.m02, matrix.m00) <  Math.atan2(z_desired, -x_desired) - Math.PI/32); i++)
{
   move(-1);
   Thread.sleep(70);
   base3d.get(matrix);
}
//wysokosc
arm_height_control3d.get(matrix);
for(int i = 0; i < 100 && ((matrix.m13 > y_desired +0.06 && matrix.m13 > -1.7)
      ||(matrix.m13 < y_desired -0.06&& matrix.m13 < 2.2 )); i++)
{
   if(matrix.m13 < y_desired)
      move(2);
   else
      move(-2);
   Thread.sleep(70);
   arm_height_control3d.get(matrix);
}
//promien
arm_width_control3d.get(matrix);
for(int i = 0; i < 100 && (( matrix.m13+2. > Math.sqrt(Math.pow(x_desired,2)+Math.pow(z_desired,2)) +0.06 && matrix.m13+2. > 3.2 )
      ||( matrix.m13+2. < Math.sqrt(Math.pow(x_desired,2)+Math.pow(z_desired,2)) -0.06 && matrix.m13+2. < 4.9)); i++)
{
   if(matrix.m13+2. < Math.sqrt(Math.pow(x_desired,2)+Math.pow(z_desired,2)))
      move(3);
   else
      move(-3);
   Thread.sleep(70);
   arm_width_control3d.get(matrix);
```