Wizualizacja ramienia robota typu “articulated arm”

Opis struktury programu projektu zadania:

Zrealizować animowaną wizualizację ramienia robota typu “articulated arm”.

Projekt został wykonany w środowisku JavaFX. Do realizacji projektu wykorzystano podstawowe funkcje JavaFX.

Projekt składa się z 2 części: wizualizacji 3D oraz samego kodu pozwalającego na realizację wizualizacji 3D.

Podstawowymi obiektami wizualizacji 3D są:

- **ziemia** - funkcja dająca w odpowiedzi **Cylinder -** *prepare Podstawa();*
- **podstawa robota** - funkcje dające w odpowiedzi **Cylinder** - *prepareSecondPodstawa() i preparePodstawaRobota();*
- **1 oś swobody robota** - funkcje dające w odpowiedzi **Cylinder-** *preparePierwszeRamieRobota1(), preparePierwszeRamieRobota2(), preparePierwszeRamieRobota3();*
- **2 oś swobody robota** - funkcja dająca w odpowiedzi **Box-** *prepareDrugieRamieRobota1();*
- **3 oś swobody robota** - funkcje dające w odpowiedzi **Box-** *prepareTrzecieRamieRobota1(), prepareTrzecieRamieRobota_magnes();*
- **duża płyta PG** - funkcja *PGBox()* dająca w odpowiedzi **Box**, funkcja *ImageView1()* - obraz 2D (cały herb PG umieszczony na *płyty PG*), funkcja *PGMagnes()* - magnes do którego jest przyciągany herb PG);
- **słup z przewodnikiem** - funkcje *Zasady()*, *ZasadySlup()* dające w odpowiedzi **Box** oraz funkcja *Przewodnik()* - obraz 2D z zasadami wizualizacji robota.
- **obiekt ruchomy (herb PG)** - funkcje dające w odpowiedzi **Box -** *Herb()*, *HerbPodstawka();*
- **światła** - *prepareLightSource1(), prepareLightSource2(), prepareLightSource3()* dające w odpowiedzi źródła światła

Do wygodnego interfejsu wykorzystano funkcje:

- *initMouswControl()* do przemieszczenia horyzontu w 3D (z ograniczeniem horyzontu oraz możliwością ZOOM);

[3-31] Importowano potrzebne biblioteki do realizowania projektu.
[37 - 102] Na początku głównej funkcji Main() umieszczono różne zmienne, które są stosowane w kodzie.

[108-112] Po utworzeniu *camery* i *sceny* [163-166] następuje zainicjalizowanie SmartGroups, które odpowiadają za przemieszczenie ruchomych i nieruchomych części robota (obiekty zamieszczone do smart grupy *calkiem_nieruchoma* i *nieruchoma*, jak sama

nazwa wskazuje, nie będą się przemieszczać ze względu na brak translacji tych smart grup. Podczas gdy obiekty *pierwszegoRamienia, drugiegoRamienia i trzeciegoRamienia* będą się obracać, ponieważ zajdzie translacja (*Translate()* ich smart grup).

[128-151] Obowiązkowo następuje przepisanie jednej smart grupy do drugiej (dodanie do smart grupy dzieci).

[115-117] Funkcje *rotate_pierwsze_ramie(), rotate_drugie_ramie(), rotate_trzecie_ramie()* są do obrotu poszczególnych osi swobody. W liniach [156-161] jest dodawane do tych funkcji dzieci (określono w taki sposób, jakie części robota powinny się ruszać w poszczególnych smart grupach).

[146-154] Utworzenie poszczególnych grup i dodanie do nich dzieci.

Sterowanie ramieniem robota przechodzi za pomocą klawiszy klawiatury. Do wyboru klawiszy wykorzystano funkcję switch [183-307]. W tym case:

- **A** - obrót 1 osi robota w lewą stronę;
- **D** - obrót 1 osi robota w prawą stronę;
- **W** - kierowanie 2 osi robota w górę;
- **S** - kierowanie 2 osi robota w dół;
- **Q** - kierowanie 3 osi robota w górę;
- E - kierowanie 3 osi robota w dół;

- **G** (grab) - wziąć obiekt (tutaj płytkę *Herb()*);
- **R** (release) - puścić obiekt (tutaj płytkę *Herb()*);
- **C** (cofnąć) - cofnąć się do pozycji początkowej ((0,0,0) osi robota);
- **L** (learning) - wykonać ruch, którego robot się nauczył;
- **Z** (zero) - zerowanie całej symulacji (reset).

Ruchy robota są zadawane przyciskami **A, D, W, S, Q, E** poprzez funkcje *obrot_w_lewa_strone_1_ramie(), obrot_w_prawa_strone_1_ramie, kierowanie_w_gore_2_ramie, kierowanie_w_dol_2_ramie, kierowanie_w_gore_3_ramie, kierowanie_w_dol_3_ramie()*. Każda z tych 6 funkcji ma identyczny sposób działania i różni się jedynie warunkami działania.

Tak na przykład obrót lewostronny 1 osi robota odbędzie się (*counter1* to zmienna mówiąca o aktualnym kącie 1 osi robota) wówczas, gdy spełniony będą warunki *if()* oraz gdy nie będzie występowała kolizja robota z płytą PG (gdy *kolizja() == false*). W momencie obrotu do macierzy dwuwymiarowej są zapisywane aktualne położenia kąta 1 osi robota, żeby później na podstawie tego odtworzyć przebieg pracy robota (jest to tak zwany tryb wykonywania robota).

Jak już wcześniej było wspomniane, przed wykonaniem ruchu osi robota jest sprawdzane, czy nie zaszła kolizja. Funkcja *kolizja()* ma odpowiedź boolowską i sprawdza na podstawie położenia płyty PG czy następnie wykonany ruch przez robota nie będzie ruchem kolizyjnym.

Wzięcie obiektu (przycisk **G**) następuje wtedy, gdy magnes chwytaka robota jest położony wystarczająco blisko herbu, czyli gdy są spełnione warunki *if()* w funkcji *funkcja_grab().*

Funkcja *GRAB_HERB()* znajdująca się w *funkcja_grab()* jest kluczową funkcją wzięcia obiektu. Działa ona w taki sposób, że jeżeli magnes chwytaka robota jest wystarczająco blisko herbu PG, to HERB jest zdejmowany ze smart grupy nieruchoma (*nieruchoma.getChildren().remove(HERB)*) i dodawany do smart grupy *trzecieRamie* (*trzecieRamie.getChildren().add(HERB)*), w międzyczasie zmienia się translacja grupy *HERB*, do której jest dodany tylko herb PG. Równolegle z tym jest zapisywane aktualne położenie chwytaka robota - sczytywane dane z *uczenie[0][a],uczenie[1][b],uczenie[2][c]* i przypisywano je do *spra,sprb,sprc.*

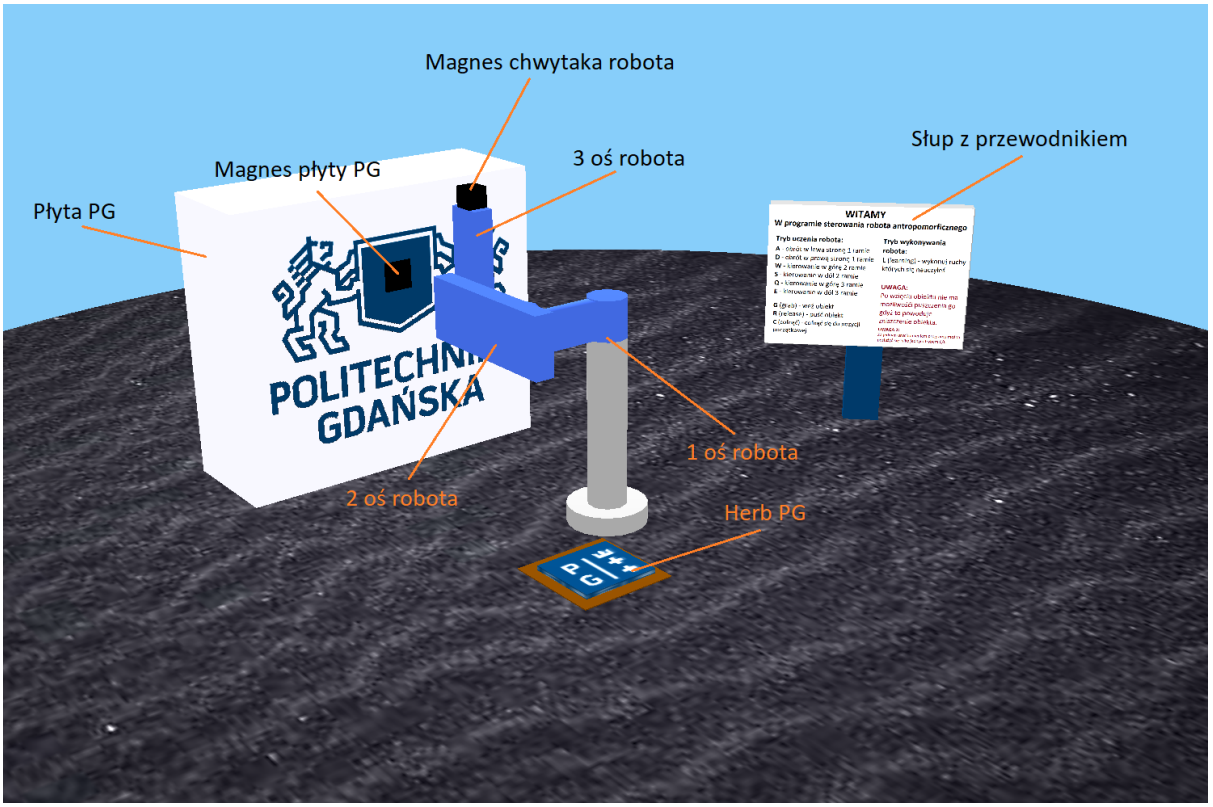
Puszczenie obiektu (przycisk **R**) nie jest możliwe po przymagnetowaniu herbu do chwytaka robota (po użyciu przycisku **G**) i możliwe jest tylko wtedy, gdy ramię robota jest umieszczono maksymalnie blisko magnesu płyty PG (gdy spełnione są warunki *if()* funkcji *funkcja_release()*).

Funkcja *RELEASE_HERB()* znajdująca się w *funkcja_release()* działa w sposób bliźniaczy do funkcji *GRAB_HERB()*. Czyli jeżeli ramię robota znajduje się wystarczająco blisko magnesu płyty PG, to *HERB* jest zdejmowany ze smart grupy *trzecieRamie* i z powrotem dodany do smart grupy nieruchoma (jednocześnie zmienia się translacja grupy *HERB* i jest zapisywane aktualne położenie chwytaka robota - sczytywane dane z *a,b,c* i przypisywano je do *rpra,rprb,rprc*).

W danej wersji projektu za jednym uruchomieniem programu można posłużyć się tylko jednym klawiszem z **C** i **L**. Jest tak z powodu tego, że obydwie funkcje *LEARING_timer()* i *COFANIE_timer()* zawierają wewnątrz siebie osobne timery i pracując razem wywierają na siebie wpływ. Wynikiem tego może być błędna praca programu. Z tego powodu powstały zabezpieczenia, uniemożliwiające wykorzystanie **L** i **C** podczas jednego uruchomienia programu.

Aby cofnąć się do pozycji początkowej osi robota (klawisz **C**), trzeba wykryć najkrótszą drogę od aktualnego położenia robota do punktu początkowego (0,0,0) ramienia robota. Tę funkcję pełnią warunki *if()* wewnątrz funkcji *COFANIE_timer().*

Za odtworzenie przez operatora zadanego zakresu ruchów (klawisz **L**), odpowiada funkcja *LEARNING_timer()*. Działa ona na podstawie warunków funkcji *if()*. Ruchy będą otworzone z pozycji zerowej ramienia robota (0,0,0) do pozycji, w której pozostawiono robot przed naciśnięciem klawiszy **L**. W trakcie odtworzenia tych ruchów robota sprawdza się, w jakiej chwili zostały wciśnięte przyciski **G** i/lub **R** (jeżeli w ogóle zostały wciśnięte), żeby w odpowiedniej chwili odtworzyć wzięcie obiektu oraz puszczenie obiektu.



Rys.1: Wizualizacja programu.