Programátorsky manuál pre Twin program

Úvod

Dokument slúži pre nadväznú prácu s kódom aplikácie Twin. Aplikácia zabezpečuje simuláciu a zrkadlenie robotického pracoviska. Bola tvorená ako bakalársky projekt pre organizáciu VÚEZ, ktorá sa venuje výskumu a vývoju. V tejto dokumentácií už nie je obsiahnuté ako s programom pracovať, prípadne ako upravovať súbory, ktoré sú potrebné na jeho spustenie. Toto všetko nájdete v užívateľskej dokumentácií "*Používateľský manuál pre Twin program*".

Keďže víziou tohto programu je jeho nadstavba vo firme VÚEZ v rôznych formách, tento dokument vysvetľuje fungovanie programu jednotlivých tried a dôležitých funkcií. Hlavný dôraz je kladený na výpočet kinematiky a zobrazovania robotov. Program je tvorený v C++ s pomocou knižníc VTK a PCL.

Obsah

Inštalácia PCL a VTK

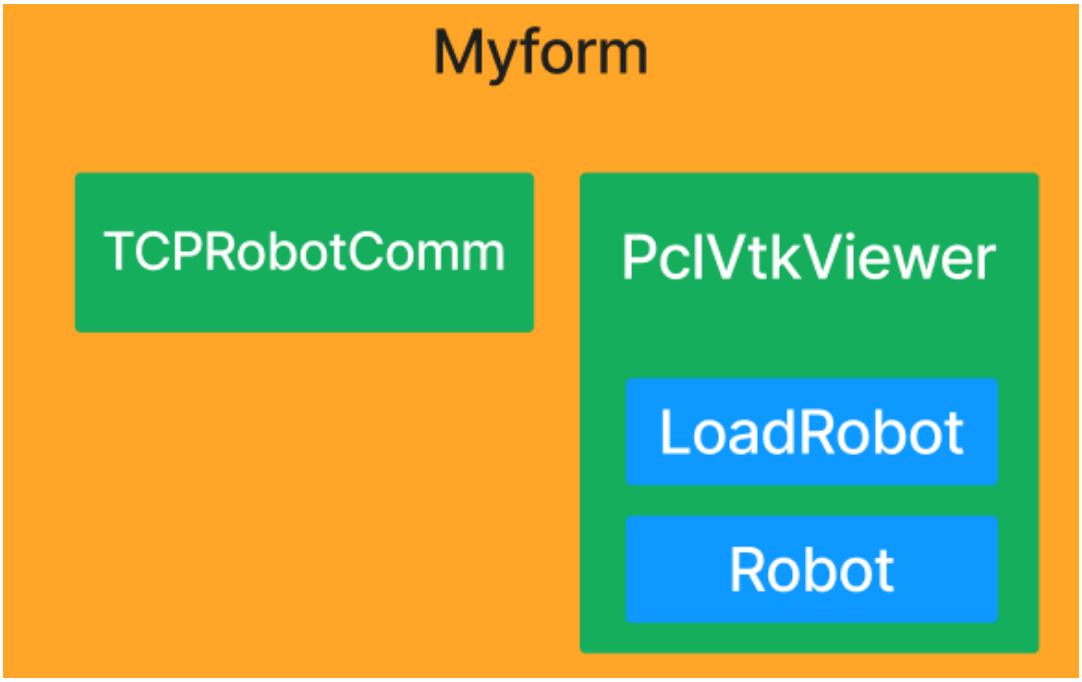
Na spustenie a editáciu programu je potrebné správne nainštalovať a nalinkovať knižnice PCL a VTK, aplikácia funguje na verziách pre PCL 1.12.0 aj 1.12.1 a pre VTK na 9.0 a 9.1. Aktuálna aplikácie je spustiteľná pre verziu PCL 1.12.1. Pre zmenu verzie PCL knižnice na verziu 1.12.0 by bolo nutné prelinkovať a nastaviť jej používanie v programe.

Z hľadiska operačného systému je možné použiť Windows 10 aj 11.

Kompletný návod pre inštaláciu knižníc nájdete tu:

Prehľad tried v programe

Názov	Charakteristika
Myform	Main trieda, obsahuje všetky ovládacie a zobrazovacie prvky GUI, na ktoré reaguje
PclVtkViewer	Vytvára okno v ktorom vizualizuje robotov. Stará sa o vytvorenie a manažovanie robotov.
Robot	Obsahuje dátovú štruktúru spojenú s robotom. Ukladá jeho parametre a vlastnosti, taktiež sprístupňuje jeho úpravy.
LoadRobot	Načítava všetky potrebné informácie z XML do triedy Robot a vykresľuje okolie robotov.
TCPRobotComm	Vytvára server a manažuje jeho stavy, taktiež z prichádzajúcej komunikácie získava a posiela údaje.



Trieda Robot

Ako je spomenuté v tabuľke slúži na uskladnenie a sprístupnenie dát. Či už na prepis alebo čítanie. Preto je tvorená zväčša metódami get a set. Nachádza sa tu však aj metóda, ktorá zostaví robota po načítaní dát z XML. Kľúčové v tejto triede je štruktúra **Part,** ktorá má v sebe všetky vlastnosti jedného ramena.

Premenné so slovenským komentárom

```
	struct Part {
		const char* name;  //cesta k 3D modelu ramena po konvertovaní zo stringu
		string link;         //cesta k 3D modelu
		string color;        //farba modelu
		double xOffset;      //Odchýlka v osi X od predchádzajúcej súčiastky
		double yOffset;      //Odchýlka v osi Y od predchádzajúcej súčiastky
		double zOffset;      //Odchýlka v osi Z od predchádzajúcej súčiastky
		double angle=0;      //Aktuálne natočenie ramena v stupňoch
		double angleOffset=0; //Odchýlka ramena v stupňoch
		vector<double> rotation{0,0,0}; //Vektor otáčania ramena v osiach {X,Y,Z}
		vtkColor3d actorColor;
		vtkNew<vtkActor> actor;
		vtkNew<vtkPolyDataMapper> mapper;
		vtkNew<vtkPLYReader> reader;
		vtkNew<vtkTransform> translation;
		vtkNew<vtkNamedColors> colors;
	};
	int parts;
	Part* field;
```

Konštruktor

Argument konštruktora je celé číslo (**int**), ktoré definuje počet ramien robota. Podľa tohto počtu sa potom generuje počet štruktúr Part.

Metódy

Ako bolo spomenuté zväčša sa jedná o get a set metódy, ktoré netreba bližšie opisovať. V metóde setup() sa však nachádza základ vizualizácie 3D modelu v podobe actora – činiteľa s ktorým sa v tejto knižnici ďalej ľahko pracuje.

```
void Robot::setup() {

	for (int i = 0; i < parts; i++) {
		field[i].actorColor = field[i].colors->GetColor3d(field[i].color);
		field[i].name = field[i].link.c_str();
		field[i].reader->SetFileName(field[i].name);
		field[i].reader->Update();
		field[i].mapper->SetInputConnection(field[i].reader->GetOutputPort());
		field[i].actor->SetMapper(field[i].mapper);
		field[i].actor->GetProperty()-
>SetDiffuseColor(field[i].actorColor.GetData());*/
	}

}
```

Odhliadnuc od nastavenia farby je najprv PLY objekt daný do vtkPLYReader, ktorý umožňuje načítať PLY (Polygon File Format) súbory do VTK a používať tieto dáta pre vizualizáciu, analýzu alebo spracovanie. Ďalej ho teda používame na spracovanie a teda používame vtkPolyDataMapper, ktorý tento model zmení na body, čiary a polygóny. Tie sú následne pripravené na vloženie do actora – činiteľa.

Trieda LoadRobot

Má za úlohu vpísať všetky potrebné dáta pre stavbu robota z XML súboru do triedy Robot. Taktiež sa stará o vykreslenie prostredia robotov.

Metódy

ParseXML(Robot *robot, string link_XML) – metóda, ktorá má ako vstupné argumenty cestu na umiestnenie XML, ktoré má čítať a objekt Robot do ktorého má vpisovať tieto údaje. Prechádza potom XML, ktoré má mať presnú štruktúru (pomenované v užívateľskom manuáli) a ukladá ich do objektu Robot.

LoadTable() - metóda, ktorá podobne ako setup() v triede Robot načítava cez actor 3D modely. Využíva však vtkOBJReader na demonštráciu vizualizovania viacerých formátov 3D modelov.

```
vtkNew<vtkPolyDataMapper> obMapper2;
vtkNew<vtkOBJReader> obReader2;
vtkColor3d actorColor2;
vtkNew<vtkNamedColors> colors2;
actorColor2 = colors2->GetColor3d("Gray");
obReader2->SetFileName("robots_ply/enviroment/pracovny_stol.obj");
obReader2->Update();
obMapper2->SetInputConnection(obReader2->GetOutputPort());
table->SetMapper(obMapper2);
table->GetProperty()->SetDiffuseColor(actorColor2.GetData());
```

SetPosTablePosition(int i, double x, double y) – metóda na posúvanie stolov a skladu obrobkov počas behu programu.

Trieda TCPRobotComm

Zabezpečenie TCP komunikácie s pracoviskom. Pracovisko posiela špeciálny typ správy ktorý treba rozdeliť na jednotlivé údaje a posielať robotovi v jemu známej forme. Správa vyzerá nasledovne:

<ActPos X="879.987122" Y="-48.003094" Z="1630.830200" A="-177.769058" B="17.104849" C="179.977203" A1="3.541394" A2="-117.052872" A3="106.828979" A4="-1.729984" A5="83.206032" A6="5.723405"></ActPos>

Kde parametre X,Y,Z,A,B,C – nepotrebujeme hovoria o koncovej polohe efektora.

Parametre A1-A6 - vkladáme do poľa, ktorým následne naplníme robota.

Konštruktor

Pri vytváraní servera potrebuje inštancia dostať ako argument port.

Metódy

RunServer() – Spúšťa server a čaká na pripojenie klienta. Ak je klient pripojený Číta dáta pomocou metódy ParseXML().

ParseXML() – funguje rovnako ako ParseXML() v triede LoadRobot. Jej zapisovacia funckia XML_GetRotation je však prispôsobená na čítanie tohto typu správy. Prečítané správy ukladá do poľa ActPos

GetActPos() – Vracia aktuálnu hodnotu poľa. Podobne ako pri zapisovaní aj pri vracaní hodnoty sa používa mux. Jedná sa o bool premennú ktorá zabezpečuje striedanie zapisovania aj vracania hodnoty. Sú tak vytvorené 2 polia, ActPos a ActPos2. Takto fungujúci algoritmus zabezpečí, že program nečíta z premennej do ktorej práve zapisuje.

StopServer() – zastavenie servera

GetStatus() - Zistenie aktuálneho stavu servera. Stavy pre server sú:

0 – odpojený

1 – bežiaci mód

2 – čaká na pripojenie klienta

3 – čaká na zápis dát

Trieda PclVtkViewer

Trieda vytvára okno v ktorom sa vizualizuje robot. Taktiež sa v nej nachádza hlavná logika vykresľovania ramien a ich vzájomná kinematika, ktorá tvoria robota a jeho pohyb.

Trieda zahŕňa triedy LoadRobot a Robot.

Konštruktor

Bez argumentov, no má dôležitú úlohu, keďže inicializuje okno, prostredie a robotov. Po vytvorení inštancií robot, ich aj naplní pomocou rozdeľovania XML. To cez inštanciu triedy LoadRobot a funkcie ParseXML, spomenutej vyššie.

```
load->ParseXML(man1, "robot_xml/man1.xml");
man1->setup();
```

Pracovisko má v nulových polohách trochu iné hodnoty ako 0 stupňov v každom kĺbe. Ak by to tak bolo nastala by kolízia, preto sú počiatočne uhly posunuté podľa reality pracoviska. Posun sa deje nazačiatku a pri resetovaní polôh robotov.

```
man1->setAngle(2, man1->getAngle(2) - 90);
man1->setAngle(3, man1->getAngle(3) + 90);
man2->setAngle(2, man2->getAngle(2) - 15);
```

Metódy

loadRobot – najdôležitejšia funkcia programu. Berie inštanciu Robot a zostaví robota na náklade jeho vlastností a aktuálnych posunov.

```
void PclVtkViewer::loadRobot(Robot* robot) {
    for (int i = 0; i < robot->getParts(); i++) {
        if (i == 0) {
            robot->getTransform(i)->DeepCopy(translation);
        }
        else {
            robot->getTransform(i)->DeepCopy(robot->getTransform(i-1));
        }
        robot->getTransform(i)->Translate(robot->getXOffset(i), robot->getYOffset(i), robot->getZOffset(i));
        robot->getTransform(i)->RotateWXYZ((robot->getAngle(i) - robot->getAngleOffset(i)), robot->getRotation(i, 0) , robot->getRotation(i, 1), robot->getRotation(i, 2));
        robot->getActor(i)->SetOrigin(0, 0, 0);
        robot->getActor(i)->SetPosition(0, 0, 0);
        robot->getActor(i)->SetUserTransform(robot->getTransform(i));
        viewer->getRenderWindow()->GetRenderers()->GetFirstRenderer()->AddActor(robot->getActor(i));

    }
}
```

Argumentom metódy je teda inštancia robota. Následne cyklus berie každé rameno a dáva ho do závislosti k predchádzajúcemu ramenu.

changePosition() – volá funkciu loadRobot pre všetky tri roboty pri akejkoľvek zmene pohybu.

resetActual a resetAll – funkcie na resetovanie jedného alebo všetkých robotov.

Poznámka: inštancia currentRobot má v sebe vždy referenciu zvoleného robota v programe. Takto sa programovo ľahko upravuje iba daný robot.

Trieda Myform

Je najrozsiahlejšou triedou. Myform.h je automatiky generovaná z MyForm.g [Design]. Väčšina funkcií a prvkov sa týka zobrazovania a reakcií na ovládacie prvky v okne. Tieto funkcie sú intuitívne a majú vlastné dokumentácie preto sa im táto dokumentácia nevenuje. Dôležité v tejto triede je **beh servera**, **práca s kamerou** a **základná interakcia s inštanciou triedy PclVtkViewer.**

Premenné

Vytvárajú sa inštancie triedy TCPRovotComm, pre každý server jedna:

```
TCPRobotComm^ server1;
TCPRobotComm^ server2;
TCPRobotComm^ server3;
```

Konštruktor

Vytvára **3 vlákna** ktoré naštartujú server. Sú spúšťané hneď od začiatku a bežia počas celého behu programu. Takisto sa spúšťajú časovače, ktoré čítajú údaje z TCP komunikácie každých 40ms.

Metódy

`threadServer1_DoWork()` – Vlákno, ktoré zabezpečuje spustenie servera, sú 3 totožné funkcie:

```
this->server1 = gcnew TCPRobotComm(59021);
demoviewer->readyServers += 1;
server1->RunServer();
```

Inicializácia inštancie a spustenie servera. Argument konštruktora je číslo portu.

setCamera(double x, double y, double z, double fx, double fy, double fz) - funkcia nastaví kameru na pozície x,y,z a nastaví taktiež bod na ktorý sa má kamera sústrediť na pozície fx,fy,fz. Nakoniec zmenu renderuje.

```
void Pclviewerdemo::MyForm::setCamera(double x, double y, double z, double fx, double fy, double fz)
{
    demoviewer->viewer->setCameraPosition(x, y, z, 0, 1, 0, 0);
    demoviewer->viewer->getRenderWindow()->GetRenderers()->GetFirstRenderer()->GetActiveCamera()->SetFocalPoint(fx, fy, fz);
    demoviewer->viewer->getRenderWindow()->Render();
}
```

V programe sa v triede Myform.cpp na riadku 261 nachádza zakomentovaný kód, ktorý vypisuje aktuálnu polohu kamery pre budúce použitia a nastavenia vlastných kamier.

setSizeTrackbar() – nastavuje aktuálne rozsahy sliderov v GUI, je volaná pri zmene výberu aktuálneho robota. Možné použiť pri pridávaní nových robotov.

cloud() – zakomentovaná metóda, slúži na vloženie pointcloudu do programu.

timer_TicCam – časovač, ktorý sa spúšťa každých 100ms. Ak je zapnuté sledovanie kamery nastavuje pozíciu kamery podľa poslednej transformácie efketroa a posúva kameru zároveň s efektorom.

timer_Tick - časovač, ktorý sa spúšťa každých 40ms, kontroluje ktorý robot je pripojený a od pripojených pomocou metódy fillRobot() napĺňa uhly robotov. Taktiež odčíta uhly z robotov a premieta ich do textových štítkov a sliderov.