STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Simulátor Yunimin – virtuální svět pro simulaci robotů

Bedřich Said

Brno 2010

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

obor SOČ: 18. Informatika

Simulátor Yunimin – virtuální svět pro simulaci robotů

Yunimin simulator – a virtual world for simulating robots

Autor:	Bedřich Said
Škola:	Gymnázium Brno tř. Kpt. Jaroše 14 658 70 Brno 2
Konzultant:	Martin Vejnár

Brno 2010

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně, použil jsem pouze podklady citované v práci a postup při zpracování práce je v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V ……. dne …… podpis: ……

Bedřich Said

Poděkování

Děkuji svému konzultantovi Martinu Vejnárovi za odborné vyjádření k této práci, za jeho podnětné příspěvky týkající se kontroly a kritiky zdrojových kódů, za rady a návrhy, které mi aktuálně průběžně poskytoval během mé tvůrčí činnosti. Srdečně mu děkuji za všechny chvíle, které mi věnoval svým obětavým a rychlým přístupem v dané problematice.

Bedřich Said

Anotace

Cílem projektu je sestavit aplikaci typu client-server-viewer, která je schopna simulovat předem naprogramované roboty. Idealizovat jim podmínky, ve kterých se vyskytují, a dále sledovat a přesně měřit jejich pohyb.

Jedná se o aplikaci, která má primárně pomáhat odstraňovat chyby v programech určených pro roboty. Vzhledem k tomu, že reálný svět si nemůžeme nijak idealizovat, nemůžeme v něm předem nastavit některé podmínky a sledovat, jak se v takových situacích robot zachová. Právě tyto problémy stály u zrodu tohoto projektu. Aplikace má za úkol simulovat chování robotů v idealizovaném světě a přesně měřit hodnoty, které v reálném světě jen těžko zjišťujeme.

Program bude zvládat simulování více robotů současně, a to s možností vzájemného ovlivňování robotů mezi sebou, včetně jejich komunikace. Celá simulace může běžet na více počítačích a lze ji sledovat a řídit přes internet s účastí neomezeného počtu zájemců. Dalším cílem je stálý chod hlavního vlákna na serveru, ke kterému se připojují jednotliví účastníci simulace a vkládají své roboty do virtuálního světa z libovolného místa pomocí internetu.

Klíčová slova: software, simulátor, robot, aplikace, server, client

Anotation

The project aims to compile the application client-server-viewer, which is able to simulate the pre-programmed robots. The application can idealize the conditions, monitor and accurately measure their movement.

It is an application that is primarily intended to help debug programs for robots. Because we cannot idealize the real world, we cannot pre-set certain conditions and see the behavior of robots. These problems were at the root of this project. The application is responsible for simulating behavior of robots in the idealized world and the application will measure the exact values which we can measure only with difficulty in the real world.

The program will manage to simulate multiple robots with the possibility of interaction among robots, including their communications, too. Whole simulation can run on multiple computers and can monitored and managed via the Internet with the participation of an unlimited number of viewers. Another aim is to continue the main thread running on the server, which connects individual participants to the simulation and puts their robots in a virtual world from any location via the Internet.

Key words: software, simulator, robot, application, server, client

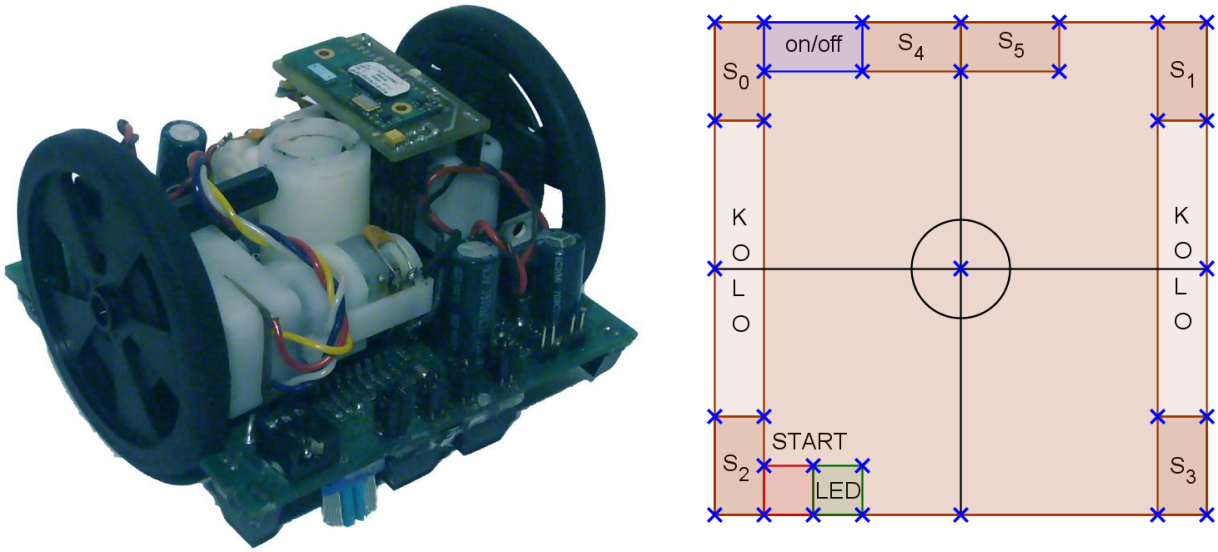
Obsah

Obrazová dokumentace

1. Charakteristika simulovaných robotů

Simulace je navržena pro dvoukolé roboty s nezávislým řízením jednotlivých kol. Střed spojnice těchto kol je také střed robota.

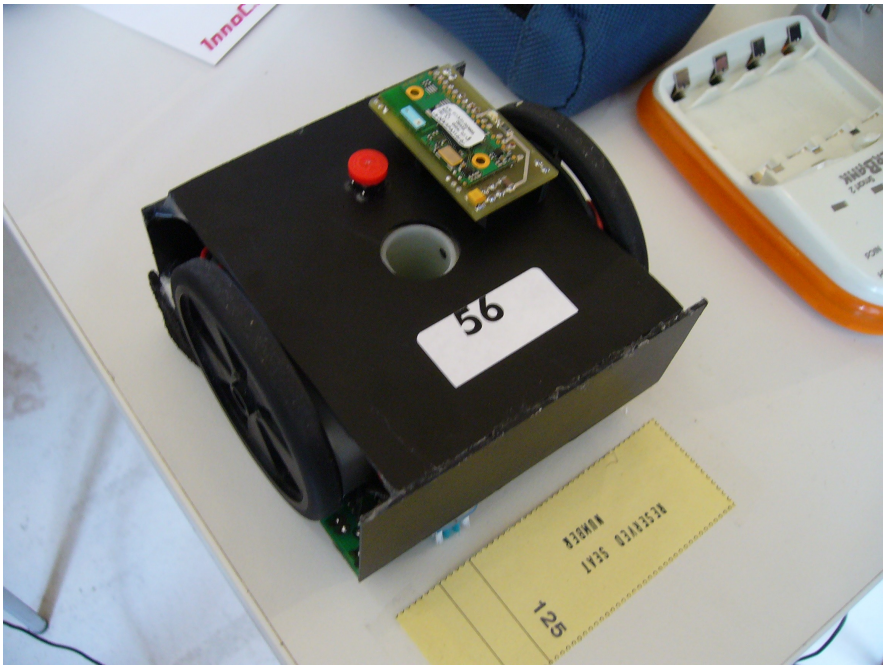
Robot je vybaven šesti infračervenými sensory snímající odrazivost podkladu. Čtyři z těchto sensorů jsou umístěny v rozích robota a zbývající dva ve středu přední části robota. V dalších úpravách aplikace se předpokládá nastavitelná poloha sensorů vůči robotovi. Dále nalezneme jednu programově řízenou LED diodu a jedno programovatelné tlačítko. Nechybí ani tlačítka na zapnutí či vypnutí celého robota. V úvaze je možnost nastavení více tlačítek či LED diod. Simulace libovolných robotů nebo nastavování údajů, jako způsob pohybu a podobně, není plánována. Závěrem se předpokládá simulace výše uvedených dvoukolových robotů s volitelným nastavením dalších údajů charakterizujících daného robota.



Celkový vzhled simulovaných robotů a plánek rozvržení jejich součástí

Podrobný popis programování výše uvedeného hardwaru lze najít na webových stránkách http://technika.junior.cz/.

1.1 Fotografie ze soutěží s roboty, kteří budou simulováni:



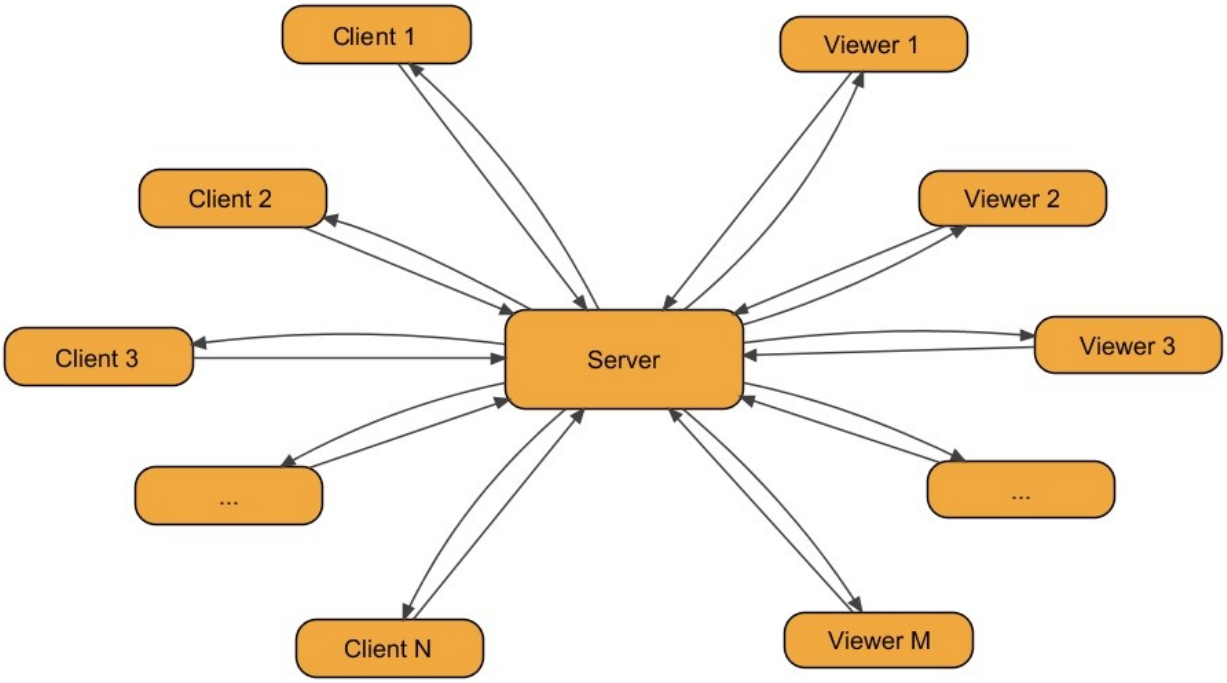
Fotografie ze souěže Robot Challenge ve Vídni. Zde je robot ve vypnutém stavu.

Fotografie pořízená na soutěži IST Robot v Bratislavě. Robot je zachycen při sledování černé čáry.

2. Architektura aplikace

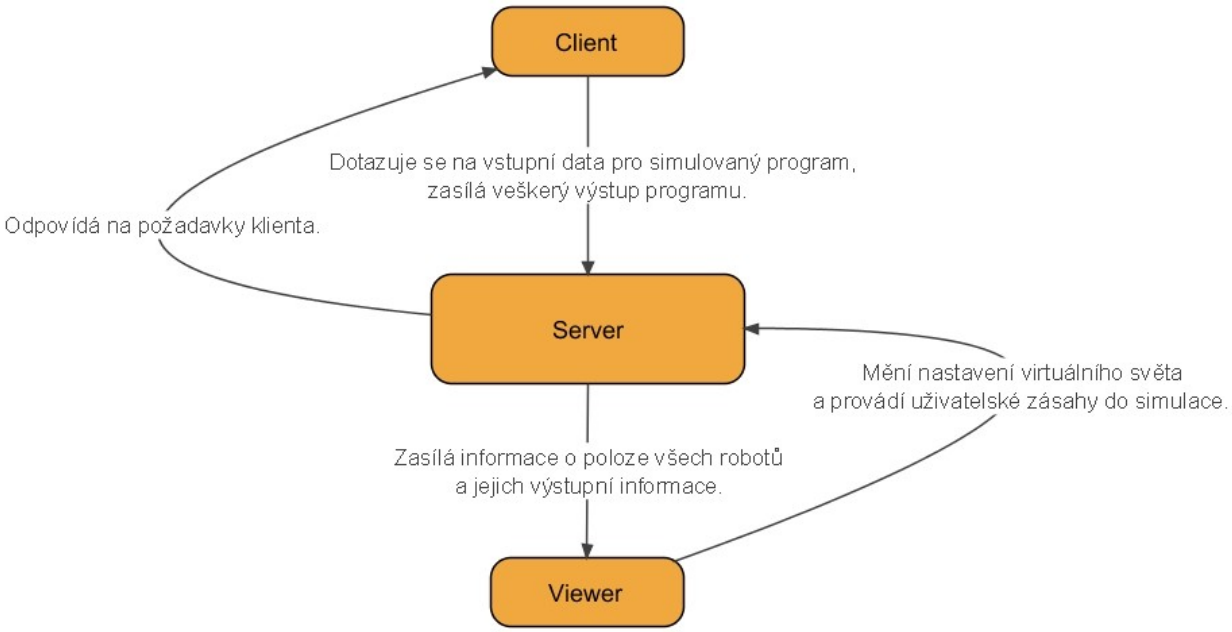
Celá aplikace se skládá ze serveru, ke kterému se připojují clienti a viewery. Client je konsolová aplikace, která simuluje program nahraný do robota. Viewer je uživatelská aplikace, která uživateli zobrazuje průběh simulace a dává mu možnost zasahovat do scény.

Server pracuje jako simulační prostředí, ke kterému se může připojit libovolné množství robotů, kteří simulují svůj chod v tomto prostředí. Dále se k serveru může připojit libovolný počet viewerů, kteří od serveru získávají veškeré informace o dění ve virtuálním světě a nastavují veškeré ručně prováděné operace. Uživatel má tak možnost ovládat celou simulaci pouze prostřednictvím vieweru.



Způsob komunikace mezi jednotlivými částmi aplikace

Lze zároveň používat různé verze clientů a viewerů. Aplikace je navržena tak, aby jednotlivé části aplikace musely splňovat pouze kritéria komunikačního protokolu. Je možné vytvořit více různých clientů a viewerů pracujících na různých platformách. Díky tomuto navržení se celá aplikace stává multiplatformní.



Specifikace komunikace clientů a viewerů se serverem

3. Popis a funkce aplikace

Celá aplikace ještě není zprovozněna, není tedy možné napsat celkový uživatelský manuál. V této kapitole jsou popsány funkce jednotlivých částí celé aplikace, tj. clienta, serveru a vieweru.

3.1 Server:

Server primárně přijímá spojení od clientů a viewerů a vyřizuje jejich požadavky. Vlastní centrální uložiště všech informací o všech robotech zapojených do simulace. Všechny změny zasílá připojeným viewerům. Díky tomuto uložišti je server schopen odpovídat na dotazy klientů, kteří potřebují vstupní data k odvedení korektní simulace.

3.1.1 Výpočet polohy a dalších dat ovlivňovaných činností robotů:

Výpočetní jednotka serveru počítá údaje dané vnějším prostředím. Ze získaných výkonů motorů počítá aktuální rychlost kol a aktuální polohu robota v prostoru. Obnovuje informace o natočení každého kola, a tím udržuje aktuální hodnoty enkodérů. Na základě těchto údajů dále počítá snímané hodnoty jednotlivých sensorů v závislosti na poloze robota a podkladu, který má server k dispozici jako bitmapu. Udržuje aktuální informace o stavu LED diody a programovatelného tlačítka. Přijímá datový proud ze sériových linek jednotlivých robotů a podle nastavení tato data může přeposílat dalším robotům. Jedná se o simulaci vzájemné komunikace mezi roboty pomocí sériové linky. Výpočetní jednotka má také na starosti chod stopek a na základě toho řídí centrální čas simulace.

Ve zdrojových kódech je tato výpočetní jednotka reprezentována vláknem, které cyklicky volá funkci, která postupně všechny tyto úlohy obstarává.

3.1.2 Aktuální informace o robotech - „robotProperties“:

Tato datová struktura je vytvořena pro každého robota zvlášť a obsahuje všechny informace potřebné k simulaci.

Obsahuje polohu každého robota v prostoru včetně natočení vůči systému souřadnic, výkon levého i pravého motoru, rychlost a hodnotu enkodérů obou kol, stav LED diody a programovatelného tlačítka, hodnotu levého i pravého serva (Serva je možné programově ovládat, avšak do simulace zatím zapojena nejsou.), dále obsahuje informaci o blokaci serv, aktuální naměřené hodnoty všech sensorů, data zaslaná robotem pomocí sériové linky a data, která čekají na zaslání do robota.

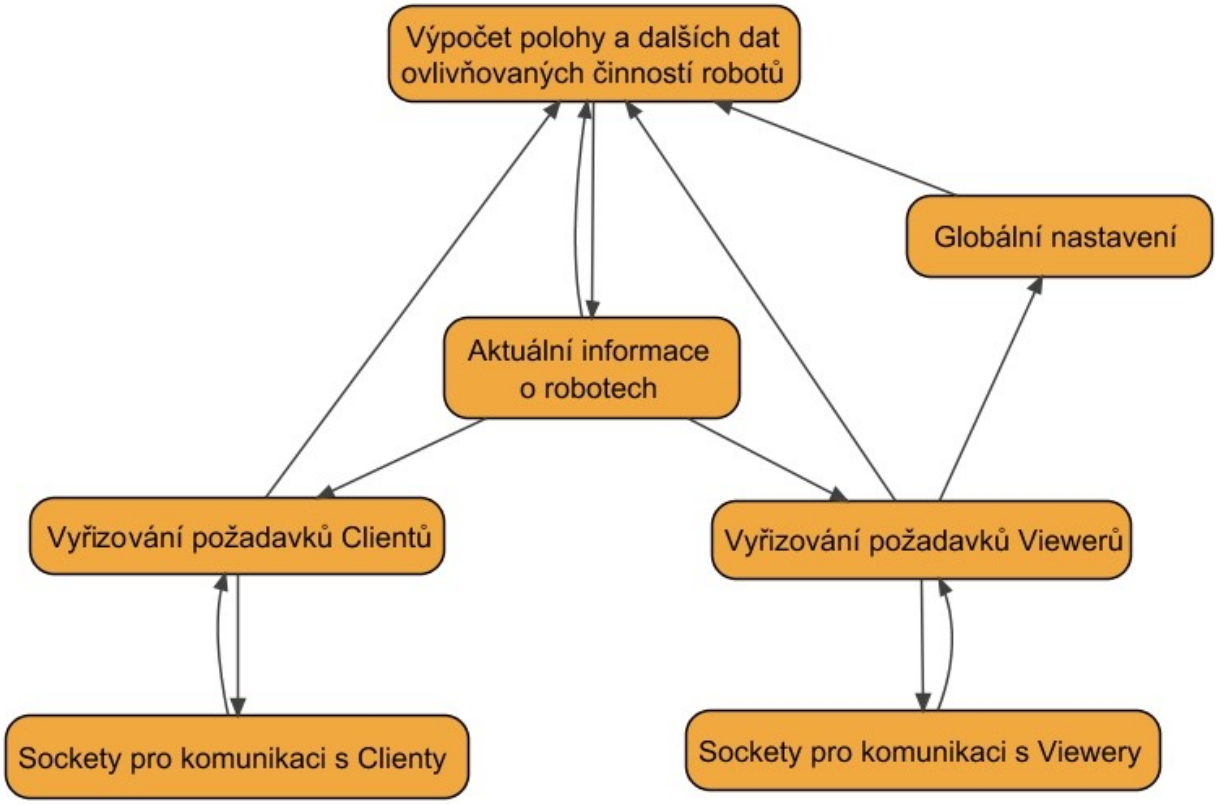
3.1.3 Globální nastavení:

Uchovává informace o virtuálním světě a nastavení provedená uživatelem pomocí vieweru.

Obsahuje bitmapu, po které se roboti pohybují, a informace o případném naklonění terénu. Vyřizuje uživatelské požadavky na stisk programovatelného tlačítka, či celkové vypnutí, respektive zapnutí robota. Stará se též o ruční přemístění robotů a o všechny uživatelem provedené zásahy do simulace.

3.1.4 Vyřizování požadavků clientů a viewerů:

Tato vlákna naslouchají na daných portech a vyřizují veškerou komunikaci serveru s ostatními částmi aplikace. Příchozí informace zapisují do centrální paměti, dávají podněty výpočetní jednotce a odpovídají na požadavky clientů. Pro každého připojeného clienta nebo viewera server vytvoří nové vlákno, na kterém probíhá samotná komunikace. Vytvořené vlákno je ukončeno v okamžiku odpojení clienta nebo viewera od serveru.



Architektura a funkce serveru

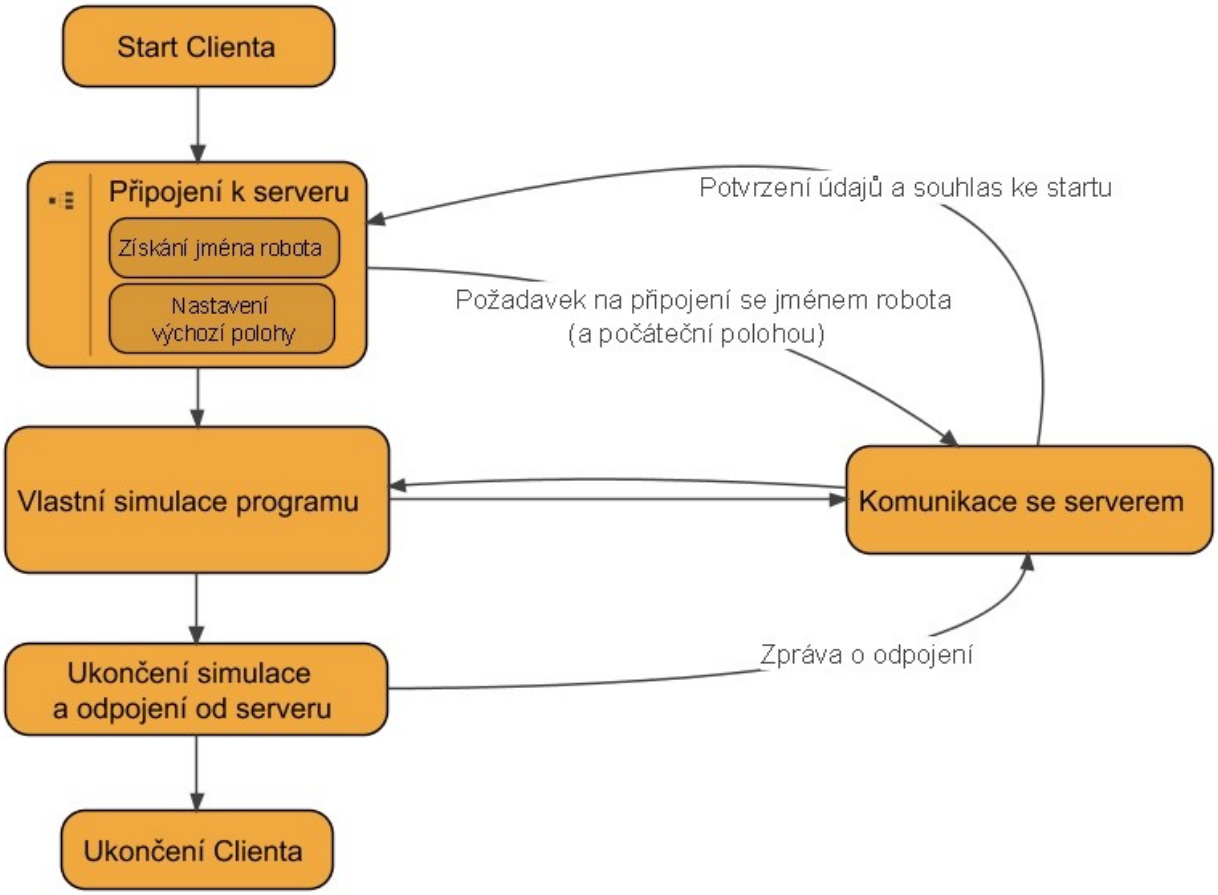
3.2 Client:

Každý client běží pouze po dobu účasti konkrétního robota v simulaci. Odstartování a připojení clienta k serveru znamená odstartování simulace programu v konkrétním robotovi. Po připojení k serveru se client snaží identifikovat pod nastavitelným jménem. Pokud není do simulace zapojen stejnojmenný robot, je tomuto clientovi zaslán souhlas ke startu programu. V opačném případě se robot pokouší připojit pod dalšími různými jmény, dokud není některé z nabízených jmen

akceptováno serverem. Před startem programu zasílá client informace o počáteční poloze robota, které je možné nastavit i pomocí vieweru. Viewer má dále možnost s robotem provést libovolný ruční pohyb proti jeho vlastní vůli. Viewer tedy má možnost upravit všechny údaje o robotovi kromě dat řízených programem a jména připojeného clienta. Client nemůže měnit své jméno v průběhu simulace z důvodu jednoznačné identifikace robotů v simulovaném prostředí.

Po přijetí souhlasu ke startu od serveru se rozběhne vlastní simulace programu se zasíláním požadavků o vstupní data. Server na tyto požadavky odpovídá tak, že před zasláním dalšího dotazu client vždy čeká na odpověď na původní dotaz. Takto proběhne celý program až po jeho ukončení. Pokud je simulovaný program zacyklen, provádí se až do ručního odpojení robota pomocí vieweru nebo do ukončení celé simulace.

Pokud clientský program skončí, client zašle na závěr zprávu o jeho ukončení, počká na potvrzení přijetí této zprávy serverem, a poté se ukončí.



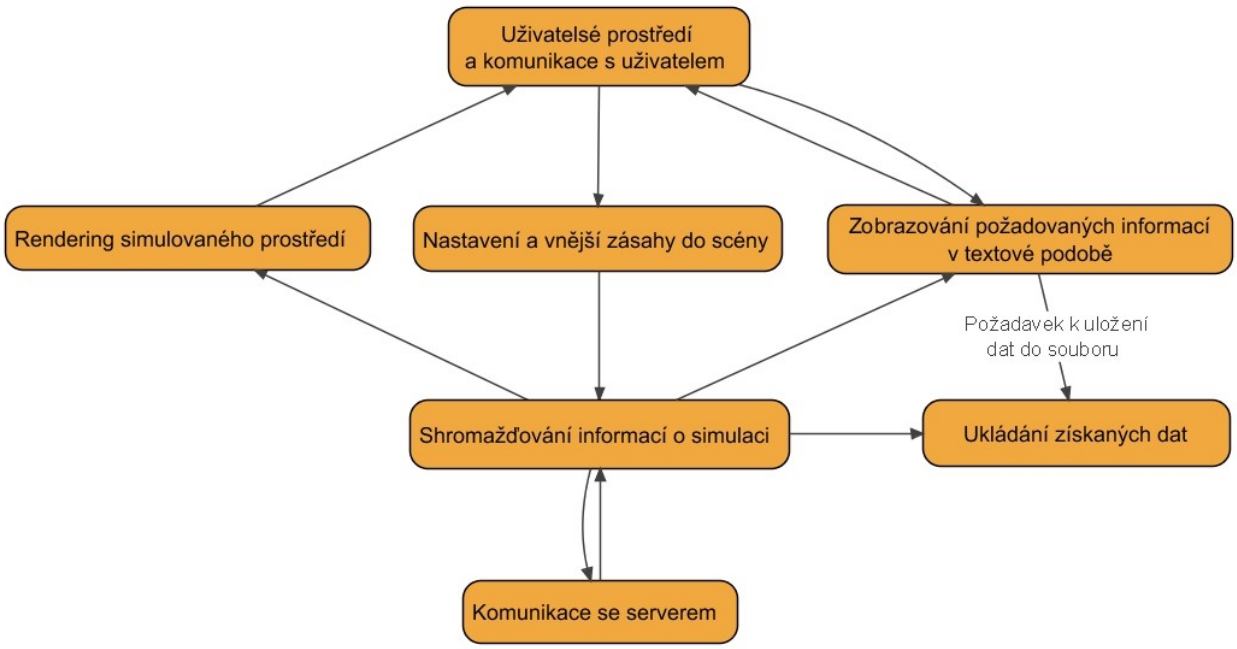
Postup chování clienta od jeho startu až po ukončení

3.3 Viewer:

Viewer zajišťuje kompletní spojení mezi uživatelem a celým systémem. Díky této výhodě je možné sledovat a řídit běžící simulaci přes síť bez ručního zásahu do počítačů, které tuto simulaci počítají. Je také možné pomocí více viewerů z různých míst připojovat další roboty a clientské programy spouštět na vzdálených počítačích. Je tedy možné provést simulaci libovolného množství tobotů bez závislosti na poloze zapojených počítačů.

Viewer přijímá od serveru všechna data o probíhající simulaci. Jeho úkolem je tato data zobrazovat uživateli a případně je ukládat na uživatelem zvolená uložiště. Poloha robotů je nyní reprezentována na 2D ploše dvojrozměrnými útvary. V plánu je i 3D rendering celé scény pomocí OpenGL. Výhoda ve 2D renderingu je v přesném zobrazování všech údajů bez transformace. Lze tak některé věci odhalit mnohem rychleji než vyhledáváním v jednotlivých vizualizovaných částech výstupu s pouhým náhledem do celé scény.

Uživatel má možnost pomocí vieweru přidávat do simulace nové roboty, měnit polohu spuštěných robotů, zastavovat nebo spouštět programy v zúčastněných robotech, provádět ruční zásahy do scény typu stisknutí programovatelného tlačítka a podobně. Kromě zásahu do činnosti jednotlivých robotů má uživatel možnost ukončit celou simulaci nebo odstartovat novou, měnit podkladovou bitmapu a celkově měnit nastavení scény. Celou aplikaci lze tedy ovládat pouze za použití viewerů, které vyřídí všechny uživatelské požadavky po síti.



Architektura a funkce vieweru

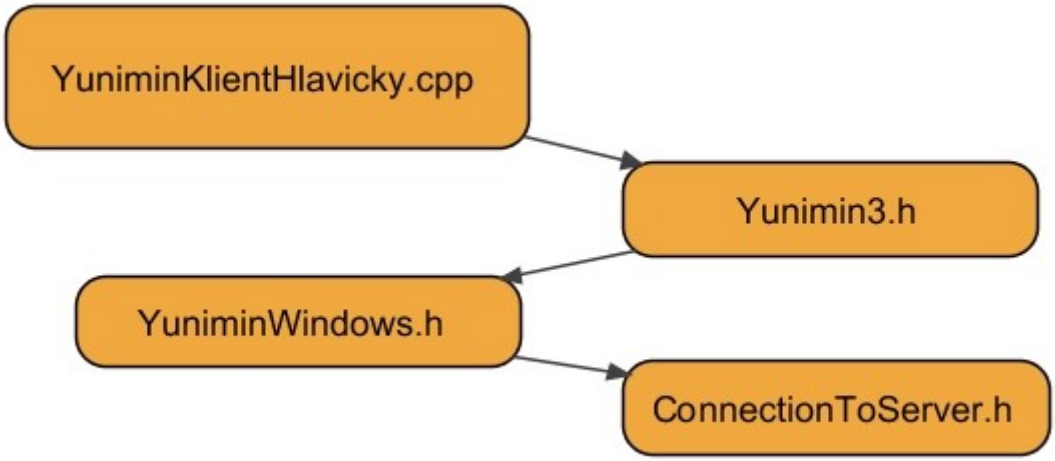
4. Podrobná specifikace aplikace

4.1 Vytvoření clienta s programem, který chceme simulovat:

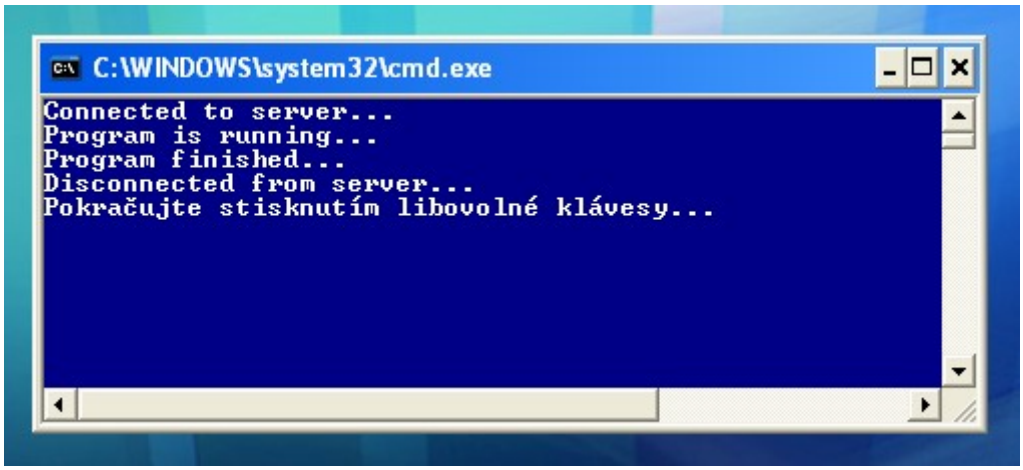
Zdrojové kódy clienta jsou navrženy tak, aby uživatel mohl použít standartní zdrojový kód, který se kompiluje pro reálné roboty. Includuje se stejnojmenná hlavička „Yunimin3.h“, která zajišťuje veškeré potřebné zdrojové kódy ke kompilaci pro simulaci. Podrobnější popis hlavičky „Yunimin3.h“ je uveden v kapitole „Zdrojové kódy clienta podle souborů“. Zdrojové kódy se skompilují na zvolené platformě a výstupem je připravený client, kterého je možné spustit jako samostatnou konsolovou aplikaci. Podrobný popis zdrojového kódu pro robota je uveden v kapitole „Zdrojové kódy clienta podle souborů“ v popisu souboru „YuniminKlientHlavičky.cpp“.

4.2 Globální struktura clienta:

Zdrojové kódy clienta se skládají z hlavního souboru „YuniminKlientHlavicky.cpp“ a hlavičkových souborů „Yunimin3.h“, „yuniminWindows.h“ a „connectionToServer.h“. K hlavnímu souboru, který je měněn uživatelem, se připojí pouze jedna hlavička, a to „Yunimin3.h“, která si obstarává veškeré další zdrojové kódy nutné k simulaci. Uživatel již nezasahuje do žádného z hlavičkových souborů.



Odkazování hlavičkových souborů u clienta



Client pracuje jako console application, uživatele může informovat i pomocí standartního výstupu

4.3 Zdrojové kódy clienta podle souborů:

4.3.1 YuniminKlientHlavicky.cpp:

Tento soubor je určen ke kompletnímu nahrazení uživatelskými soubory. V tomto souboru se nachází samotný program pro robota, který bude simulován. Skládá se z importu hlavičky „Yunimin3.h“:

```
#include "Yunimin3.h"
```

Dále pokračuje samotný program pro robota, jehož hlavní funkce se musí jmenovat „*run*“.

```
void run()
{
	waitForStartButton();
	setMotorPower(60,60);
	stopwatch stopky;
	while(stopky.getTime() < 10000000)
	{
	}
	setMotorPower(0,0);
}
```

Výše je uvedená ukázka programu, který po svém spuštění počká na stisk programovatelného tlačítka a potom nastaví oběma kolům rychlost 60 vpřed. Touto rychlostí jede po dobu deseti sekund, potom se zastaví a program skončí.

Data, která client potřebuje k navázaní spojení se serverem (jméno robota, adresa serveru a připojovací port), se clientu předají pomocí vstupu z příkazové řádky.

4.3.2 Yunimin3.h:

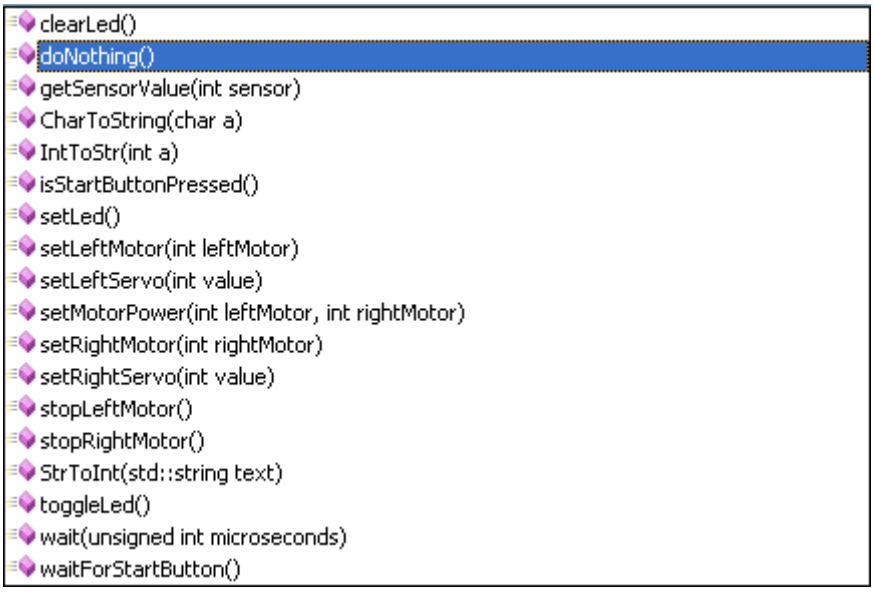
Tato hlavička nahrazuje stejnojmennou hlavičku používanou ke kompilaci programů pro reálné roboty. Je to jediná hlavička, kterou je nutné do programu implementovat, stejně jako je tomu i u reálného spuštění. Tato hlavička obsahuje všechny základní funkce včetně počáteční funkce main, které jsou používané u reálných robotů. Funkce jsou upraveny tak, aby client komunikoval se serverem. Tj. všechny požadavky na vstupní data jsou dotazovány na server pomocí funkce *sendMessageToServer* místo přímého čtení vstupu, který je k dispozici pouze u reálných robotů. Veškerý výstup je stejným způsobem odesílán na server.

```
std::string sendMessageToServer(std::string const & type,
                                 std::string const & value)
{
      (...)
}
```

Funkce *sendMessageToServer* pošle zadaný požadavek na server a vrátí jeho odpověď. Jako vstup přebírá dva parametry typu string. První je typ žádosti a druhý obsah žádosti. O této funkci je více zmíněno v kapitole „Komunikační protokol a způsob komunikace mezi klientem a serverem“.

4.3.3 yuniminWindows.h:

Tato hlavička je oproti svému mírně zavádějícímu názvu multiplatformní. Obsahuje deklarace všech funkcí, které používá robot pro svou práci. Většina z těchto funkcí pouze odesílá nebo přijímá data ze serveru. Činnost všech funkcí je zřejmá z jejich definice. Stopky pracují i intervalu 1 mikrosekunda. Funkce, které komunikují se serverem používají protokol, který je popsán v kapitole „Komunikační protokol a způsob komunikace mezi klientem a serverem“. Funkce sendMessageToServer přebírá jako první parametr typ požadavku na server a jako druhý parametr obsah požadavku. Vrací odpověď serveru a na tuto odpověď funkce čeká. Funkce doNothing má zabraňovat cyklení při čekání ve smyčce. Zatím je ale nadeklarována s prázdným tělem.

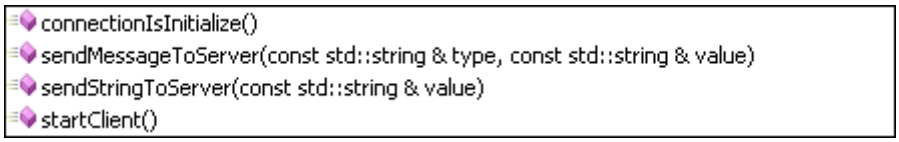


Seznam funkcí definovaných v hlavičkovém souboru yuniminWindows.h

Dokumentace k výše uvedeným funkcím je totožná s dokumentací k funkcím pro reálné roboty. Tato dokumentace je uvedena na stránkách: http://technika.junior.cz/.

4.3.4 connectionToServer.h:

Hlavička connectionToServer.h zajišťuje komunikaci clienta se serverem. Obsahuje všechny funkce, které tuto komunikaci zajišťují. Je napsána pro operační systém Microsoft Windows.



Seznam funkcí definovaných v hlavičkovém souboru connectionToServer.h

Po spuštění clienta je zavolána funkce startClient, která zajistí navázání spojení se serverem. Funkce sendStringToServer pošle požadavek ve stringovém formátu na server a čeká na jeho odpověď. Funkce sendMessageToServer vnitřně volá funkci sendStringToServer, které

předformátuje vstupní informace na formát používaného protokolu. Pro uzavření spojení se použije funkce sendMessageToServer, která dostane jako typ požadavku „close“ a nakonec se spojení uzavře funkcí closesocket.

4.4 Specifikace připojení a údaje o síti:

Ke komunikaci je používán protokol TCP/IP.

Veškeré údaje a požadavky na síť jsou nastavitelné. Defaultně je nastaveno naslouchání na připojování clientů na portu 1111. Klient se defaultně připojuje k serveru „localhost“ na port 1111 se jménem „Yunimin“. Komunikace viewera je očekávána defaultně na portu 2222. Viewer se též připojuje k serveru „localhost“, pokud uživatel nenastaví jinak. Celá aplikace se přizpůsobí téměř libovolně rychlé síti. Rychlost sítě mírně ovlivňuje rychlost simulace.

4.5 Komunikační protokol a způsob komunikace mezi klientem a serverem:

Komunikace mezi clientem a serverem probíhá v textovém režimu. Po spuštění naváže client spojení se serverem. Server vytvoří pro toto připojení nové vlákno a socket, na kterém potom očekává dotazy. Dotaz na server se zasílá jako string, který je ukončen standartní ukončovací nulou '\0'. První řádek dotazu obsahuje typ dotazu, v níže uvedené tabulce jsou uvedeny podporované typy dotazů. Tabulka nemusí být kompletní.

Název dotazu	Popis dotazu
setLeftMotor	Nastaví výkon levého motoru. Druhý řádek obsahuje nastavovanou rychlost. Očekávána je celočíselná hodnota v intervalu (-128,128).
setRightMotor	Nastaví výkon pravého motoru. Druhý řádek obsahuje nastavovanou rychlost. Očekávána je celočíselná hodnota v intervalu (-128,128).
stopLeftMotor	Zabrzdí nebo uvolní levý motor/servo. Po zabrždění motoru se uvolní servo. Na druhém řádku je očekáván znak '0' nebo '1'. Při přijetí '0' se zabrzdí motor a uvolní servo, v opačvém případě se zablokuje servo a uvolní motor.
stopRightMotor	Zabrzdí nebo uvolní pravý motor/servo. Po zabrždění motoru se uvolní servo. Na druhém řádku je očekáván znak '0' nebo '1'. Při přijetí '0' se zabrzdí motor a uvolní servo, v opačvém případě se zablokuje servo a uvolní motor.
getSensorValue	Zeptá se serveru na hodnotu sensorů. Na druhém řádku je číslem reprezentován dotazovaný sensor. Očekávána jsou celá čísla v intervalu <0,5>. Na hodnoty vyšší než 5 server vrátí nulu. Server odpoví celým číslem v intervalu (-512,512), kde -511 charakterizuje bílou barvu a 511 charakterizuje černou barvu.
isStartButtonPressed	Zeptá se serveru, jestli je stisknuté programovatelné tlačítko. Nezasílají se žádné doplňující údaje, protože tlačítko je zatím pouze jedno. Server vrátí '1' v případě stisknutí tlačítka a '0' v případě, že tlačítko není stisknuté.
setLeftServo	Nastaví pozici levého serva. Na druhém řádku je očekávána celočíselná hodnota v intervalu (-128,128). Hodnota '0' nastaví servo doprostřed. Server tuto zprávu potvrdí libovolným znakem nebo textem.
setRightServo	Nastaví pozici pravého serva. Na druhém řádku je očekávána celočíselná hodnota v intervalu (-128,128). Hodnota '0' nastaví servo doprostřed. Server tuto zprávu potvrdí libovolným znakem nebo textem.
led	Rozsvítí nebo zhasne LED diodu. Na druhém řádku je očekáván znak '0' nebo '1'. Při přijetí '0' se dioda zhasne a při přijetí '1' se dioda rozsvítí bez ohledu na její dřívější stav. Pokud dioda zaujímá stejný stav jako je uveden v požadavku, není zaznamenána žádná změna.

Název dotazu	Popis dotazu
encoder	Dotazuje se serveru na hodnotu levého nebo pravého enkodéru. Pokud je na druhém řádku „left“, server vrátí hodnotu levého enkodéru. Pokud je na druhém řádku „right“, server vrátí hodnotu pravého enkodéru. Hodnoty enkodérů jsou reprezentovány 32bitovými integery.
rs232	Pošle serveru text, který by robot poslal po sériové lince. Text je zaslán na všech řádcích kromě prvního. Server čeká na ukončovací nulu '\0'. Přijetí textu server potvrdí libovolným znakem nebo textem.
rs232get	Dotáže se serveru na znak, který čeká ve frontě na sériové lince jako první. Požadavek neobsahuje žádné další údaje. Server vrátí první znak ve frontě. Pokud je fronta prázdná, server vrátí prázdný string.
close	Uzavře spojení se serverem. Sever tento požadavek potvrdí libovolným znakem nebo textem.
connect	Naváže spojení se serverem. Jako parametr se zadává jméno připojovaného robota. Server tento požadavek buť potvrdí znakem '1', nebo vyvrátí znakem '0'. Při vyvrácení se spojení navazuje znovu s jiným jménem robota.

4.6 Komunikační protokol a způsob komunikace mezi viewerem a serverem:

Po připojení vieweru k serveru se na serveru vytvoří nové vlákno zajišťující komunikaci. Server opakovaně zasílá vieweru veškeré údaje o celé simulaci. V každém paketu se pošle název robota, který bude specifikován, a dále pokračují na každém řádku jedna informace v pořadí, v jakém jsou proměnné ve struktuře robotProperties. Zpráva je ukončena standartní ukončovací nulou '\0'. Struktuře robotProperties je věnována kapitola „Uložiště dat 'robotProperties'“. Viewer tato data potvrzuje libovolným znakem nebo textem.

Viewer může poslat na server požadavek s novým nastavením nebo se může dotázat o aktuální globální nastavení. Textové nastavení se zasílá po stejném protokolu jako komunikace mezi clientem a serverem. Pokud se mění podkladová bitmapa, je nejprve poslán požadavek s textem „changeBitmap“ nebo „changeTerrain“ nebo „changeBarrier“, server tento požadavek potvrdí a

očekává zaslání souboru s novou bitmapou ve formátu portable network graphic „*.png“. Popis jednotlivých požadavků je popsán v kapitole „Globální nastavení“.

4.7 Uložiště dat „robotProperties“:

Deklarace celé struktury včetně komentářů je uvedena níže. Tuto strukturu není třeba dále komentovat.

```
class robotProperties
{
public:
	double positionX;		//poloha robota v prostoru
	double positionY;
	double rotation;
	int leftMotorPower;		//vykon jednotlivych motoru
	int rightMotorPower;
	bool led;			//stav LED diody
	bool startButton;		//stav programovatelneho tlacitka
	int leftEncoderValue;		//hodnoty enkoderu
	int rightEncoderValue;
	int leftServoValue;		//poloha jednotlivych serv
	int rightServoValue;
	bool lockLeftServo;		//uzamceni/uvolneni serva
	bool lockRightServo;
	int sensorValues[8];		//posledni namerene hodnoty jednotlivych sensoru
	queue<char> serialLine; //data cekajici na zaslani po seriove lince
	std::string serialSent; //data odeslana robotem po seriove lince
};
```

4.8 Popis jednotlivých vláken serveru:

4.8.1 Výpočetní jednotka:

Výpočetní jednotka je hlavní vlákno serveru, které neustále cyklicky volá jednu funkci, která běží od začátku až do konce celé simulace. S každým zavoláním této funkce se celá simulace

posune o jednotkový čas.

Cyklicky volaná funkce *computeSimulation* přebírá z globální paměti výkon motorů, aktuální pozici a hodnoty enkodérů. Z výkonu motorů vypočítává rychlost kol s pohlédnutím na setrvačnost kol. Všechny tyto údaje jsou nastavitelné. Výkon motoru je brán jako lineární funkce k tažné síle motorů. V dalších verzích simulátoru se předpokládá změna na reálnou funkci s nastavitelným silovým ekvivalentem k výkonu. Obsah této funkce je pouze matematicko-fyzikální záležitost, která zde není dopodrobna popsána z důvodu neustálého dynamického vývinu této části aplikace.

4.8.2 Komunikace:

Pro komunikaci s každým připojeným clientem nebo viewerem server vytvoří nové vlákno, které naslouchá na vytvořeném socketu. Každé takové vlákno má přístup do globální paměti, včetně zapisovacího práva.

V případě clientského vlákna se globální paměť přepisuje aktuálním výstupem simulovaných programů. Více v kapitole „Komunikační protokol a způsob komunikace mezi klientem a serverem“.

V případě připojení vieweru nově vytvořené vlákno periodicky zasílá kompletní obsah globální paměti a vyřizuje požadavky zaslané viewerem. Více v kapitole „Komunikační protokol a způsob komunikace mezi viewerem a serverem“.

4.8.3 Centrální uložiště:

Centrální uložiště je pole struktur, ve kterém je pro každého robota zapojeného do simulace přidána nová struktura. Tato struktura je podrobně popsána v kapitole „Uložiště dat 'robotProperties'“. Úlohou této globální paměti je uchovávat veškeré informace o každém robotovi, který je zapojený do simulace.

Vzhledem k tomu, že k těmto strukturám mají přístup všechna vlákna na serveru, má tato paměť vytvořenou kritickou sekci, kterou si každé vlákno, které tuto paměť žádá, přivlastní po dobu komunikace s touto pamětí. Jednotlivá vlákna jsou ve Windows synchronizována pomocí těchto funkcí:

```
CRITICAL_SECTION GlobalCriticalSection;
//deklarace kriticke sekce
EnterCriticalSection(&GlobalCriticalSection);
//vlakno si touto funkci privlastni kritickou sekci,
//pokud je kriticka sekce obsazena, funkce pocka na jeji uvolneni
LeaveCriticalSection(&GlobalCriticalSection);
//uvolneni kriticke sekce
```

4.8.4 Globální nastavení:

Globální nastavení obsahuje bitmapu, která je texturou podkladu, po kterém jezdí simulovaní roboti. Tito roboti získávají díky této textuře vstupní data pro jejich sensory. Dále je v plánu zařadit terénní bitmapu, která umožní simulovat jízdu robotů do kopce či z kopce nebo vytvořit mosty.

Název požadavku	Popis požadavku
changeBitmap	Mění podkladovou bitmapu. Z této bitmapy se čtou hodnoty sensorů v závislosti na aktuální poloze robota v prostoru.
changeTerrain	Mění nastavení kopců a polohu mostů. Tato funkce je zatím ve vývoji.
changeBarrier	Mění polohu všech statických překážek na trase. Překážky jsou reprezentovány jako hranoly.

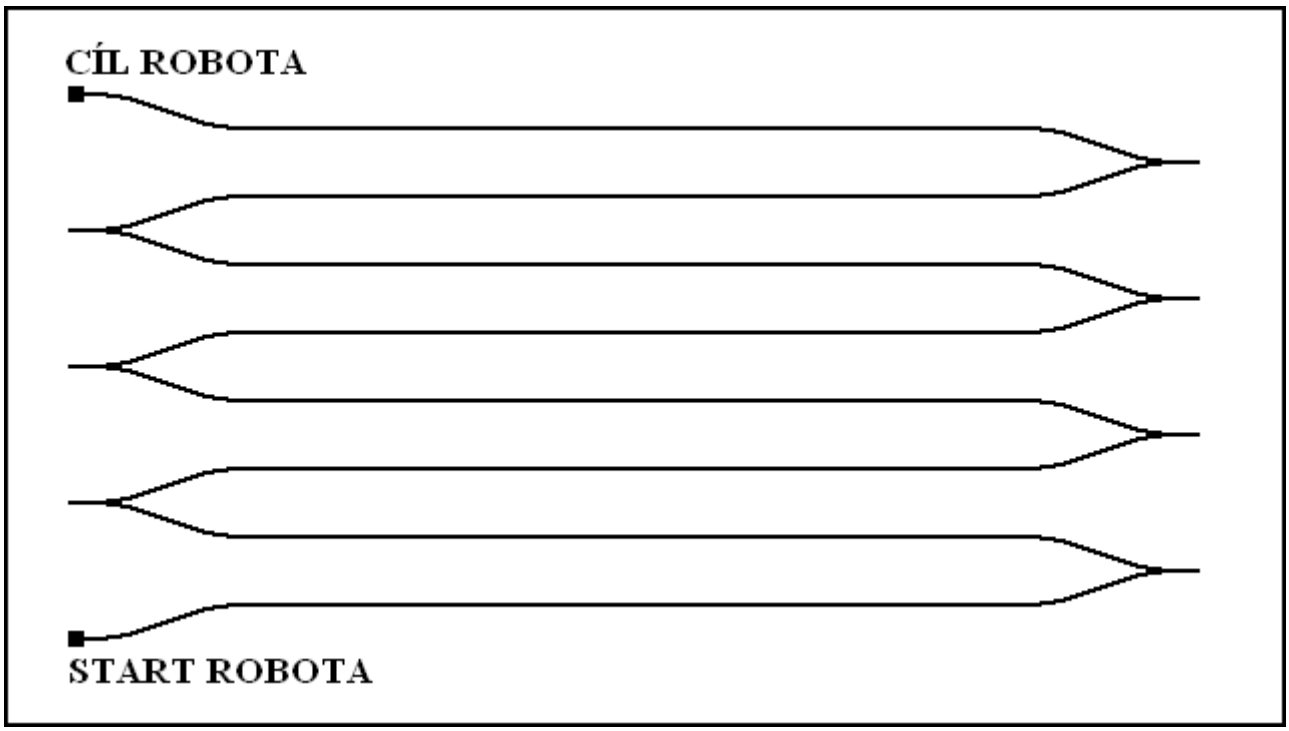
Ve všech třech případech server po zaslání tohoto požadavku očekává soubor s novou bitmapou ve formátu portable network graphic „*.png“.

Tato paměť dále obsahuje souřadnice a natočení výchozího bodu umísťování nových robotů, pokud nejsou tito roboti umísťováni ručně pomocí vieweru. Obsahuje polohy sensorů vůči robotovi, v plánu je nastavitelná poloha těchto sensorů. V této paměti také běží centrální stopky a ukládají se zde informace o zastavení, pozastavení, či spuštění celé simulace.

4.9 Komunikace s reálnými roboty:

Existuje funkce pro robota, která při každém zavolání pošle všechny údaje, které robot zná do počítače pomocí sériové linky. Cyklickým voláním této funkce jsme schopni detailně zrekonstruovat celý pohyb robota včetně všech prováděných činností. Pokud známe počáteční polohu robota v prostoru včetně natočení, jsme schpni na základě údajů z enkodérů zrekonstruovat trajektorii celého pohybu robota.

Díky tomuto faktu byla například objevena zásadní chyba aplikace, která měla zaručovat jízdu robota rovně. Robot se bohužel po čase od své rovné jízdy mírně odchyloval do strany, avšak zůstával stále natočený správným směrem. Chybou aplikace byla neúplná synchronizace rychlostí mezi jednotlivými koly a robot tak měnil rychlost s mírně esovitou trajektorií. Rovně jel rovně pouze v případě jízdy konstantní rychlostí.



Znázornění pohybu robota se špatně synchronizovanými koly

Viewer může visualizovat i data zaslaná reálnými roboty. Na funkci, která převede hodnoty enkodérů na popis trajektorie, se zatím pracuje. Zapojení reálných robotů do simulace je obtížné, protože se předpokládá vzájemné ovlivňování při simulaci. Pokud by reálný robot zapojený do simulace například narazil do virtuálního robota, nedošlo by k žádné srážce. Proto se tato kombinace při simulaci zatím nepředpokládá. Avšak je možné realizovat komunikaci mezi reálnými a simulovanými roboty, kde se každý robot pohybuje ve vlastním prostoru. Tato komunikace není zatím implementována, ale předpokládá se, že reálný robot bude moci komunikovat s virtuálními roboty zapojenými do simulace.

4.10 Ovládání aplikace pomocí vieweru:

Viewer je schopen celou simulaci spustit, pozastavit, znovu obnovit její chod nebo ji úplně ukončit. Při pozastavení lze měnit podkladovou bitmapu, bitmapu se stoupáním či klesáním a bitmapu s překážkami v podobě hranolů. Je také možné měnit nastavení polohy sensorů vůči robotovi, které je zatím konstantně nastaveno. Uživatel může do simulace přidávat nové roboty

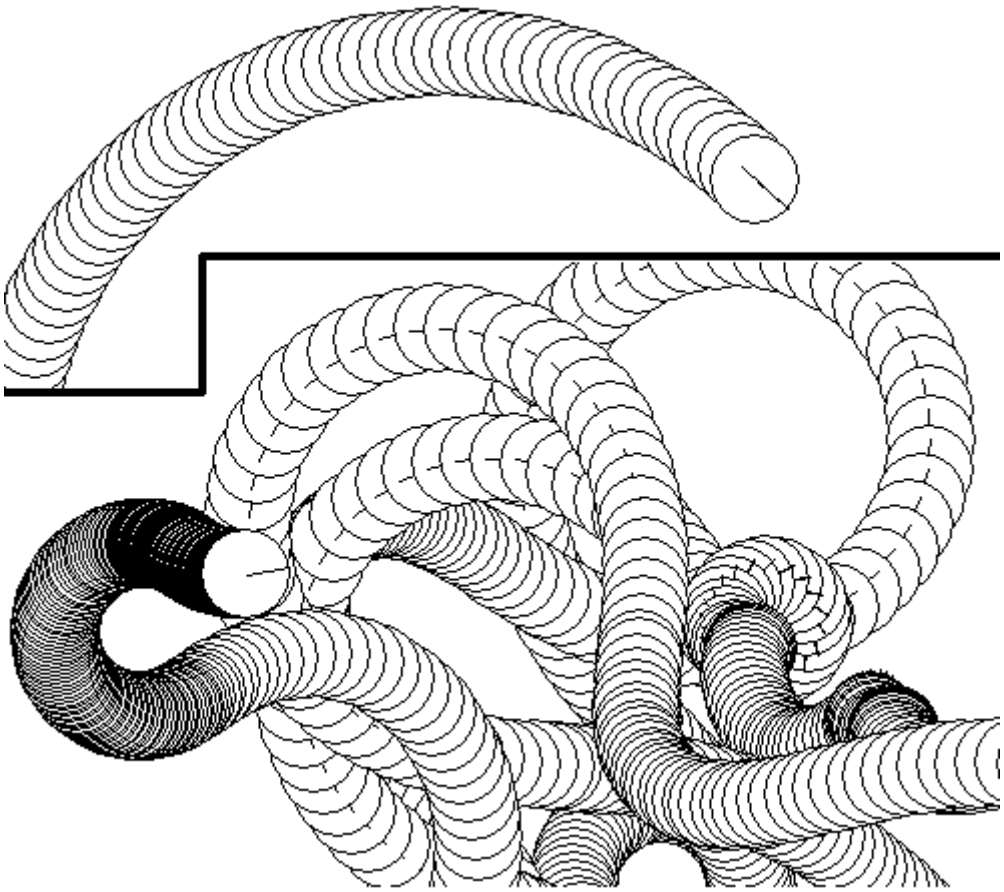
nebo běžící roboty odebrat. Lze stisknout programovatelné tlačítko na každém robotovi. Komunikace s reálnými roboty probíhá též pomocí vieweru. Nastavuje také komunikaci mezi jednotlivými simulovanými roboty a posílá serveru sériový vstup od uživatele.

Díky těmto nástrojům je možné celou simulaci řídit pouze pomocí vieweru.

4.11 Uživatelské prostředí vieweru:

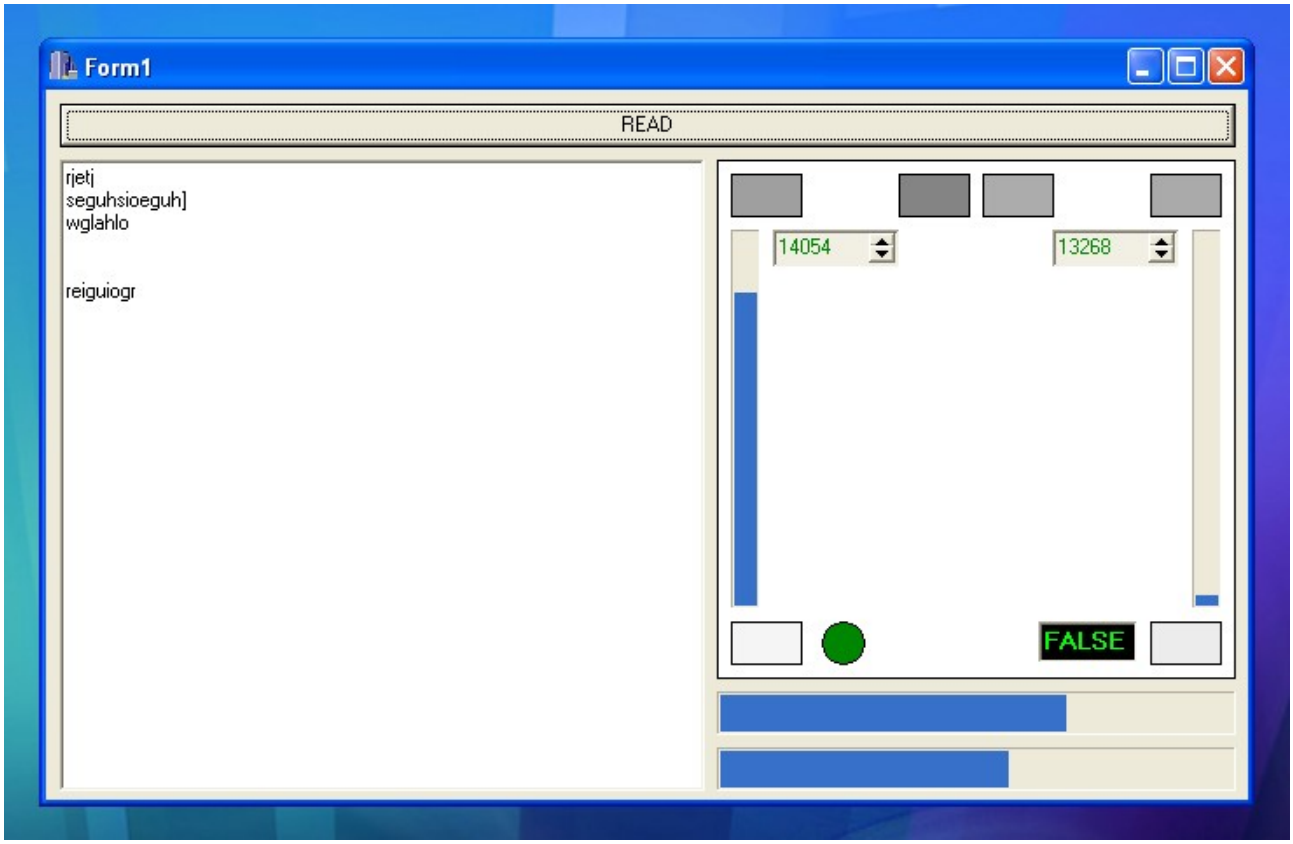
Viewer vizualizuje uživateli celý průběh simulace. Data jsou zatím zobrazována buť textově nebo pomocí 2D renderingu polohy robota v závislosti na čase. Dále zobrazuje veškeré aktuální informace o jednotlivých robotech zapojených do simulace.

Vizualizace pohybu robota je kreslena buť jako zobrazování pouze aktuální polohy robota nebo je zobrazována stopa v místech, kudy robot projel. Ve zjednodušené podobě je zobrazován pouze kruh se směrnicí pohybu. Na detailním zobrazování se zatím pracuje.



Zjednodušený náhled do simulace pohybu jednoho robota zanechávajícího stopu

Pro vizualizaci detailních informací o jednotlivých robotech je použito rozhraní GUI, které průběžně zobrazuje informace o zvoleném robotovi.



Zobrazení jednotlivých hodnot vstupu a výstupu konkrétního robota

Na výše uvedeném obrázku je zobrazena zjednodušená visualizace aktuálních dat vstupu a výstupu konkrétního robota. V levém poli je zobrazován výstup sériové linky RS232, v pravém dolním rohu je znázorněno celkové napětí na bateriích včetně aktuálně odebíraného proudu. V pravé horní části je znázorněn robot s visualizovanými hodnotami sensorů, enkodérů, výkonu motorů, signalizace LED diody a pozice programovatelného tlačítka.

Na kvalitní vizualizaci všech dat včetně polohy robotů a renderingu celé scény se zatím pracuje. Vizualizace je prováděna z textového vstupu, který obsahuje číselné údaje o všech výše zmiňovaných hodnotách a text sériové linky RS232. Tento vstup lze získat i čtením všech informací reálného robota a posíláním těchto údajů do počítače. Textový vstup pro výše ukázanou visualizaci a více informací o textovém formátu je uvedeno v kapitole „Data, která zpracovává viewer“.

4.12 Data, která zpracovává viewer:

Viewer na svém vstupu obdrží textová data, která potom visualizuje. Příklad textového vstupu pro jednoho robota bez údajů o poloze v prostoru je uveden níže. Tento příklad byl zaslán reálným robotem pro testování vieweru, který se ještě nedokázal spojit se serverem. Po sériové lince je v tomto příkladu zaslána náhodná kombinace znaků.

```
~ -328 -160 -409 -178 -72 -96 207 25 8418 7854 0 22 72 0
~ -319 -202 -411 -212 -76 -91 206 26 8419 7856 0 23 73 0
~ -306 -207 -409 -269 -72 -99 205 28 8419 7859 0 24 74 0
~ -291 -195 -407 -281 -60 -110 206 27 8419 7861 0 25 75 0
~ -267 -157 -404 -231 -37 -111 206 27 8419 7864 0 26 76 0
~ -244 -99 -408 -164 -14 -96 206 27 8420 7866 0 27 77 0
~ -233 -43 -413 -160 -6 -59 205 27 8420 7869 0 28 78 0
~ -230 30 -417 -190 -8 -18 205 29 8420 7872 0 29 79 0
~ -238 67 -425 -231 -19 8 205 28 8421 7874 0 30 80 0
seguhsioeguh]
wglahlo


reiguiogr
~ -249 93 -432 -289 -31 10 205 28 8421 7877 0 31 81 0
~ -255 140 -436 -315 -35 -18 204 28 8422 7880 0 32 82 0
~ -261 189 -436 -290 -33 -60 204 28 8422 7883 0 33 83 0
~ -274 175 -429 -224 -16 -93 204 28 8423 7886 0 34 84 0
~ -293 146 -417 -175 -3 -120 203 30 8423 7889 0 35 85 0
~ -311 88 -410 -191 -5 -125 204 31 8424 7892 0 36 86 0
```

Ukázka formátu dat zasílaných reálným robotem

Kromě těchto dat týkajících se jednoho robota viewer obdržuje data týkající se polohy a natočení robota v prostoru. Reálný robot není schopen tyto informace získat. Může se řídit pouze hodnotami enkodérů. Viewer v každém cyklu obdrží soubor těchto dat v pořadí, v jakém jsou tato data uložena ve struktuře „robotProperties“.

Popis formátu požadavků, které viewer zasílá na server je popsán v kapitole „Komunikační protokol a způsob komunikace mezi viewerem a serverem“.

4.13 Ukládání výsledných dat:

Přímo podporované formáty: txt, csv, xml.

Export renderované scény do formátu: png (případně s možností animace)

Všechna data, která získáme ze simulace, lze reprezentovat v textové podobě. Z důvodu kompaktibility jsou všechna data ukládána textově. Výstup simulace v podobě detailních dat o všech robotech lze uložit jako prostý textový dokument pobobný ukázce v kapitole „Data, která zpracovává viewer“. Uživatel si může zvolit mezi uložením těchto dat jako prostý textový dokument '*.txt', jako formát '*.csv' s možností exportu do aplikace Microsoft Excel, OpenOffice Calc a aplikací, které umí tento formát zpracovat. Sériová linka RS232 je v tomto formátu reprezentována samostatným sloupcem. Dále je nabízená možnost ukládání výstupu jako formát '*.xml', kde je pro každý krok vložen nový tag, jehož podtagy obsahují jednotlivá exportovaná data. Díky formátu xml lze tato data zobrazovat ve formátu html a díky kompaktibilitě s aplikacemi Microsoft Excel a OpenOffice Calc lze tato data konvertovat do množství dalších formátů.

5. Závěr

Hlavní prioritou uvedené aplikace je pomáhat odstraňovat softwarové chyby ve zdrojových kódech programů pro roboty. Její možnosti však mají mnohem širší využití. Lze ji použít například jako nástroj pro simulaci scény s velkým množstvím robotů, které bychom v reálu jen těžko dali dohromady. Nynější verze aplikace pracuje s roboty, kteří jsou navrženi pro přesný pohyb po laboratorní podložce, ze které mohou číst odrazivost jejího povrchu prostřednictvím svých infračervených sensorů. Aplikace je navržena tak, aby nebyla omezována operačními systémy, na kterých ji lze spustit, a stala se tak multiplatformní. Věřím, že zmíněná mnohostrannost této aplikace najde své uplatnění v reálném životě.