

Jani Cerar, Jani Dugonik, Simon Gliha, Jani Lemajič, Matej Mlaker, Dejan Volk

MOTOMAN.NET

Predelava JRobSim v okolje .NET

Poročilo

Maribor, januar 2011

**VSEBINA**

[1 UVOD 1](#_Toc316516246)

[2 OPIS PROBLEMA 2](#_Toc316516247)

[3 Moduli aplikacije 3](#_Toc316516248)

[3.1 Model robota 3](#_Toc316516249)

[3.2 Rezkar 4](#_Toc316516250)

[3.3 »JEZIK« ROBOTA 6](#_Toc316516251)

[3.4 ZUNANJE KOORDINATE 7](#_Toc316516252)

[4 VIRI, LITERATURA 8](#_Toc316516253)

**UPORABLJENE KRATICE**

.NET – Microsoft .NET Framework

VS – Visual studio

C# - programski jezik C(Sharp)

LL1 – **L**eft to right parser, **l**eftmost derivation

3D – tridimenzionalni prostor

.JBI – končnica programa jobeditor32, ki se uporablja v krmilniku robota Motoman

# UVOD

Za projekt pri predmetu Robotizacija smo se odločili za prenos simulacije robota JrobSim v razvojno okolje .NET.

Aplikacija JRobSim se uporablja kot pripomoček pri simuliranju, gradnji in opazovanju gibanja različnih tipov robotov. Namenjena je tako študentom, profesorjem kot tudi ostalim uporabnikom, ki imajo interesne sfere v robotizaciji.

JRobSim je napisan v programskem jeziku Java s pomočjo razvojnega okolja NetBeans. Razvoj orodja je potekal že več let, naša skupina pa se je odločila projekt implementirati še v razvojnem okolju Visual Studio zaradi vsesplošne razširjenosti programskega jezika C#.

# OPIS PROBLEMA

Implementacija simulatorja v .NET je potekala v naslednjih korakih:

* Implementacija zunanje kinematike, ki je bila rešena že v projektu JRobSim, v jezik C#,
* Implementacija LL1 gramatike, ki predstavlja »jezik« robota,
* Implementacija vizualnega modela robota Motoman MH6 s pomočjo knjižnice OpenTK (OpenGL),
* Implementacija podprograma za rezkanje, skupaj z izhodno funkcijo

# Moduli aplikacije

V naslednjih poglavjih bomo opisali posamezne dele simulatorja in postopke implementacije.

## Model robota

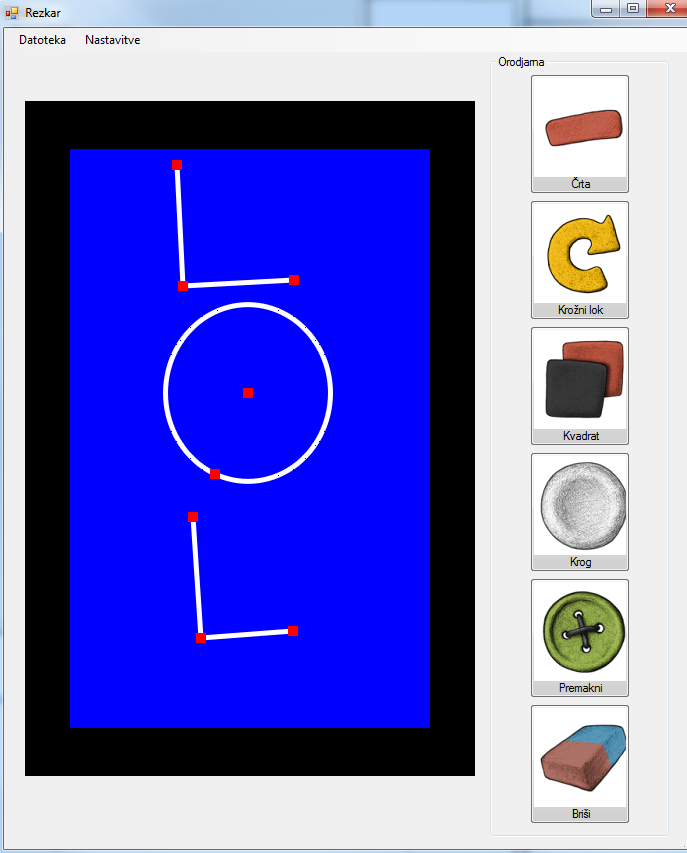
Model robota smo izdelali v LightWave programskem paketu za 3D modeliranje, ki ga je izdelalo podjetje NewTek, in se uporablja za rederiranje 3D animacij ali statičnih slik. Model smo nato izvozili v format .LWO, ki smo ga nato s pomočjo LightWave knjižnice ClassLibrary.dll dodali kot referenco v naš VS projekt.



Slika 1: Model robota v LightWave

## Rezkar

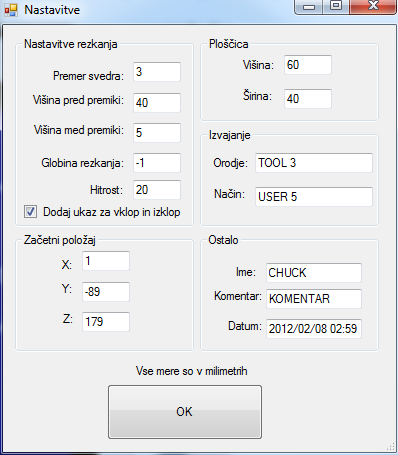
Rezkar je bil izdelan v programu Visual Studio 2010, poženemo ga znotraj glavne aplikacije. Vmesnik je prikazan na sliki 2.



Slika 2: Glavno okno rezkarja

Na levi strani okna se nahaja Orodjarna, kjer imamo osnovne gradnike za risanje likov. Poleg osnovnih črt lahko rišemo še krožne loke, kvadrate in kroge, vsak element pa lahko po potrebi zbrišemo ali premaknemo.

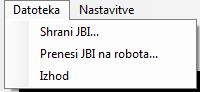
Pod menijem Nastavitve (slika 3) lahko spreminjamo parametre Rezkarja, kot npr. Premer svedra ali pa začetni položaj svedra, nastavimo pa lahko tudi mere same ploščice, na katero bomo izrezljali motiv.



Slika 3: Nastavitve rezkarja

Ko želimo naše delo shraniti, to lahko storimo v meniju Datoteka, kjer imamo na voljo dve opciji (slika 4):

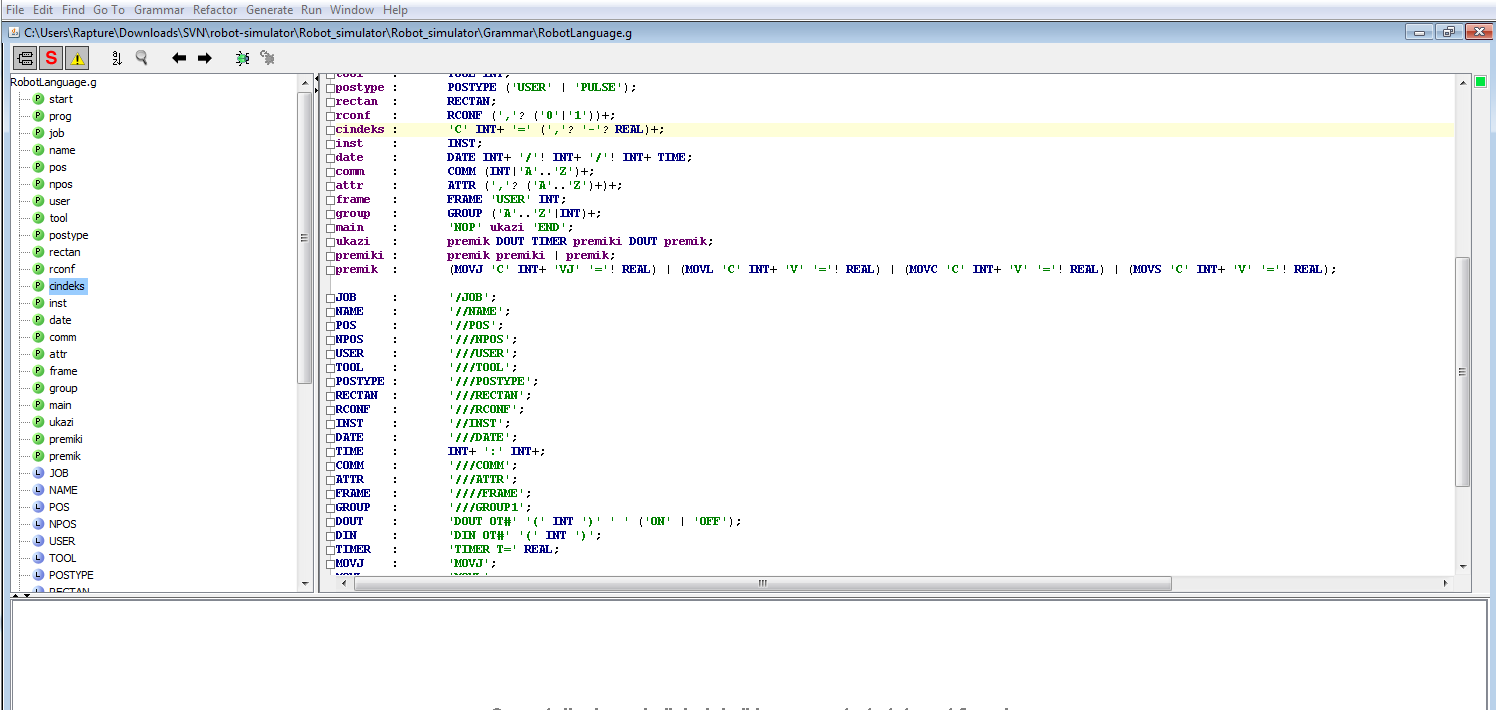
* Shrani .JBI, kjer shranimo našo sliko v datoteko .JBL, ki vsebuje avtomatsko ustvarjene ukaze v programskem jeziku INFORM II (Yaskawa).
* Prenesi JBI na robota, kjer našo sliko prenesemo v simulator, kjer se nato izvede simulacija rezkanja.



Slika 4: Nabor opcij za izvoz ustvarjenih slik

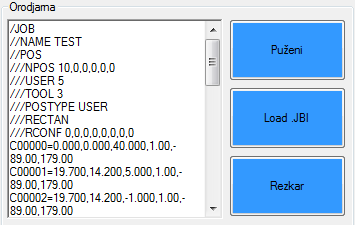
## »JEZIK« ROBOTA

Jezik robota tvori leksikalna analiza, ki pretvori nabor znakov v sekvenco t.i. »token-ov«, semantična analiza, ki poskrbi za dodajanje semantičnih informacij v semantična drevesa (»parse trees«) in zgradi tabelo znakov, s katero nato preverjamo tipe ali vežemo (»bind«) objekte. Na koncu imamo še ovrednotenje ukazov, s katerim interpretiramo ukaze in obveščamo o napakah v ukazih.



Slika 5: ANTLRWorks

Tvorba gramatike je potekala s pomočjo jezika ANTLR v programskem paketu ANTLR Works (slika 5), s katerim smo nato ustvarili semantični (RobotLanguageParser.cs) in leksikalni analizator (RobotLanguageLexer.cs) za jezik C#.



Slika 6: Naložena datoteka .JBL

Nato smo ta dva razreda vključili v skupni projekt in dodali referenco na knjižnico Antlr3.Runtime.dll, kjer nato kličemo oba analizatorja, ko uvozimo JBI datoteko (slika 5) oz. v Rezkarju uporabimo ukaz »Prenesi JBI na robota…«.

## ZUNANJE KOORDINATE

Implementacija inverzne kinematike ni uspela zaradi pomanjkanja časa. Prvotni načrt je predvideval uporabo že obstoječe funkcije, ki se nahaja v JRobSim vendar se je izkazalo, da je izvorna koda zelo nepregledna ter, da jo pesti pomanjkanje kvalitetne dokumentacije.

Implementirali smo približek inverzne kinematike vendar smo zaradi pomanjkanja časa opustili razvoj le-te ter se raje osredotočili na implementacijo ostalih delov simulatorja.

# VIRI, LITERATURA

Yaskawa. (brez datuma). NX100 INFORM MANUAL. *Motoman Instructions*.