LATVIJAS UNIVERSITĀTE
FIZIKAS UN MATEMĀTIKAS FAKULTĀTE
DATORIKAS NODAĻA

DARBĪBU PLĀNU VIZUALIZĀCIJA

KURSA DARBS

Autors: **Kirils Solovjovs**
Stud. apl. ks05020
Darba vadītājs: prof. Dr.dat. Guntis Bārzdiņš

RĪGA 2008

ANOTĀCIJA

Kursa darbā “Darbību plānu vizualizācija” tiek pārbaudīta hipotēze: valodas tekstu ir iespējams “saprast” arī nezinot precīzu tā domēnu (apakškopu no visa valodas domēna). Parasti rakstītajā tekstā iekļautā informācija ir diezgan nepilnīga, tamdēļ šo informāciju var precīzi interpretēt tikai jau iepriekš zinot domēnu, par ko ir runa dotajā tekstā. Uzstādītās hipotēzes pārbaudei tiek izveidota datorprogramma, kas spēj automatizēti vizualizēt darbību plānus.

Kursa darbā ir iekļauta eksistējošu alternatīvu datorprogrammas analīze, jaunas datorprogrammas izstrāde, jaunu formātu izstrāde, darba rezultātu apraksts un to analīze.

Atslēgvārdi: plānošana, vizualizācija, PDDL, SemTi-Kamols, CarSim.

ABSTRACT

In the dissertation “Visualisation of action plans” the following hypothesis is verified: written text can be “understood” even if the exact domain (subset of the whole language domain) of the text is not known. Usually the information included in the written text is quite poor, therefore this information can be precisely interpreted only by knowing the domain being discussed in the specific text beforehand. A computer program that can visualize action plans in an automatic manner is created to check the hypothesis.

Dissertation includes analysis of pre-existing alternative computer programs, engineering of a new program and new formats, description and analysis of the work results.

Keywords: planning, visualisation, PDDL, SemTi-Kamols, CarSim.

АННОТАЦИЯ

В диссертации «Визуализация планов действия» подтверждается гипотеза: написанный текст можно «понять» даже если не известен точный домен (подмножество домена всего языка) текста. Обычно информация, включенная в письменный текст, довольно неполная, поэтому эта информация может быть точно интерпретирована только, заранее зная домен, обсуждаемой в определенном тексте. Чтобы проверить гипотезу, создана компьютерная программа, которая может автоматически визуализировать планы действия.

Диссертация включает анализ заранее существующих альтернативных компьютерных программ, разработку новой программы и новых форматов, описания и анализа результатов работы.

Ключевые слова: планирование, визуализация, PDDL, SemTi-Kamols, CarSim.

SATURS

APZĪMĒJUMU SARAKSTS

plānošanas domēns – teksts speciālā sintaksē, kas cita starpā ietver arī predikātu un darbību sarakstu

plānošanas problēma – teksts speciālā sintaksē, kas konkrēta plānošanas domēna ietvaros apraksta sākotnējo stāvokli un vēlamo gala stāvokli

darbību plāns – teksts, kurā aprakstīts plānošanas problēmas risinājums (secīgu un/vai paralēlu darbību saraksts)

plānošana – process, kura rezultātā no plānošanas problēmas tiek iegūts darbību plāns

gājiens – viens darbību plāna solis, kurā var izpildīties vairākas paralēlas darbības

predikātu izskata apraksts – teksts, kurā aprakstīts kā tiks attēlots kurš predikāts

objektu sākotnējo pozīciju saraksts – teksts, kas apraksta, kādas koordinātēs vizualizācijas plaknē sākotnēji jāatrodas objektiem

PDDL (Planning Domain Definition Language) – plānošanas domēnu definīcijas valoda

IPC (International Planning Competition) – starptautiskās plānošanas sacensības

XML (Extensible Markup Language) – vispārīga valoda, ar kuras palīdzību iespējams aprakstīt patvaļīgas ontoloģijas.

SWF (Shockwave Flash) – nerediģējama vektorgrafikas animācija, kas spēj mijiedarboties ar lietotāju

GIF (Graphical Interface Format) – rastrgrafikas animācija

RGB (Red, Green, Blue) – formāts, kurā krāsu izsaka ar koeficientiem pie trim summējošas krāsu sistēmas pamatkrāsām

Faila saturs vai programmas kods darbā tiks apzīmēts šādi. Fails *test.php:*

```
<?php
 echo 'Hello world!';
?>
```

IEVADS

Latvijas Universitātes Matemātikas un informātikas institūts izstrādā projektu *SemTi-Kamols*, kura mērķis ir "padarīt tiešsaistē pieejamo decentralizēto un lielākoties nestrukturēto [tekstuālo] informāciju saprotamu ne tikai cilvēkiem, bet arī automatizētām datorprogrammām (aģentiem), tādējādi paverot ceļu masveidīgai informatīvo procesu automatizācijai visdažādākajās tautsaimniecības nozarēs un sabiedrībā kopumā."(5) Tas nozīmētu iespēju datorprogrammām "saprast" tekstu un līdz ar to arī veikt ar to darbības, ko šobrīd spēj veikt tikai cilvēks. Viena no šādām darbībām ir teksta vizualizācija jeb teksta (pasakas, vēsturiskas liecības u.c.) pārveidošana animācijā. Ja būtu iespējams salīdzinoši īsā laikā automatizēti izveidot patvaļīga teksta animāciju, tas atvieglotu tekstā esošās informācijas uztveršanu.

1998. gadā Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld un David Wilkins radīja PDDL(3) – valodu, ar kuras palīdzību iespējams aprakstīt plānošanas problēmas. Šī valoda bija paredzēta IPC sacensībām, kas norisinās reizi divos gados.

IPC ir sacensības, kurās dalībnieki sacenšas par labāko plānošanas datorprogrammu (t.i. tādu datorprogrammu, kas konkrētā plānošanas domēnā atrisina plānošanas problēmu), taču šajā kursa darbā plānošanai kā tādai netiks pievērsta pārāk liela uzmanība. Tā vietā autors koncentrēsies uz to, kā zinot plānošanas domēnu, problēmu **un** darbību plānu, kā arī predikātu izskata aprakstu, to automatizēti pārveidot par animāciju, jo šobrīd projekta *SemTi-Kamols* ietvaros tiek izstrādāts modulis, kas tekstu spēs pārvērst plānošanas problēmā un darbību plānā.

Radot datorprogrammu, kas spēs pārvērst plānošanas domēnu, problēmu, darbību plānu un predikātu izskata aprakstu par animāciju ir būtiski atcerēties, ka datorprogrammai ir jābūt ļoti elastīgai, tajā jāizvairās no konstrukcijām, kas ierobežotu tās pielietošanu kādam plānošanas domēnam.

Parasti rakstītajā tekstā iekļautā informācija ir diezgan nepilnīga – tā neietver domēnu. Tamdēļ šo informāciju var precīzi interpretēt un uztvert tikai jau iepriekš zinot domēnu, par ko ir runa dotajā tekstā. Piemēram, teikumā "Meitenes iegāja zālē un spēlēja spēli ar bumbu", nezinot domēnu, vārdu "zāle" var uztvert gan kā "sporta zāle", gan "gara zaļa zāle", bet vārdu "bumba", gan kā "sporta buma", gan kā "spridzeklis"(6), līdz ar to izprast šī teikuma jēgu ir gandrīz neiespējami.

Ja tekstu ir iespējams vizualizēt animācijā, tātad tekstu ir bijis iespējams “iztēloties”, bet ja tekstu ir bijis iespējams **korekti** “iztēloties”, tātad teksts ir “saprasts”. Tātad, ja **jebkuru** plānošanas problēmu būtu iespējams automatizēti vizualizēt, tad varētu secināt, ka sekojoša hipotēze ir patiesa: valodas tekstu ir iespējams “saprast” arī nezinot **precīzu** tā domēnu (apakškopu no visa latviešu valodas domēna).

Šī darba mērķi ir:

- radīt konceptuālu iestrādni datorprogrammai, kas spēj pārvērst plānošanas domēnu, problēmu, darbību plānu un predikātu izskata aprakstu par animāciju;
- pārbaudīt augstākminēto hipotēzi.

Šajā darbā tiek izmantotas sekojošas metodes mērķu sasniegšanai:

- PDDL izpēte;
- eksistējošu alternatīvu risinājumu meklēšana un izpēte;
- datorprogrammas saskarnes plānošana;
- predikātu izskata apraksta formāta izstrāde;
- datorprogrammas moduļu izstrāde un testēšana;
- datorprogrammas darbināšana uz dažādiem plānošanas domēniem.

Kursa darbs ir sadalīts nodaļās. Darba sākumā tiek izpētīts uzdevums, PDDL sintakse, kā arī apskatītas eksistējošas datorprogrammas alternatīvas. Tad tiek plānota datorprogrammas izstrāde, tajā skaitā arī predikātu izskata apraksta formāts. Tam seko ieskats datorprogrammas būtiskāko funkciju realizācijā un uzdevumos, bet darbs noslēdzas ar rezultātiem un secinājumiem.

1. UZDEVUMA IZPĒTE

Šajā sadaļā ir pētīti darba mērķa sasniegšanai nepieciešamie priekšnosacījumi un standarti, kā arī ir iztirzāts darba mērķis.

1.1. PDDL izpēte

PDDL – valodu, ar kuras palīdzību iespējams aprakstīt plānošanas problēmas – 1998. gadā radīja plānošanas sacensību komiteja. Pirmā publiski pieejamā PDDL versija bija 1.2.(7) Kopš tā laika ir iznākuši daudzi PDDL specifikācijas atjauninājumi un pēdējais versijas numurs ir 3.1.

Neskatoties uz to, darbā tiks izmantots PDDL 1.2, jo šis standarts ir pietiekoši plašs konceptuālās datorprogrammas iestrādnes vajadzībām, turklāt, ja nepieciešams, vienmēr būs iespējams iestrādāt papildu funkcionalitāti, ko apraksta vēlākas PDDL versijas.

Ar PDDL palīdzību var definēt gan plānošanas domēnu, gan plānošanas problēmu.

1.1.1. Sintakse

PDDL valodas sintakse ir samērā vienkārša. Īsumā plānošanas domēns tiek aprakstīts ar predikātu un darbību sarakstu.

Predikātu saraksts sākas ar atslēgas vārdu "*:predicates*" un noslēdzas ar attiecīgu aizverošo iekavu. Predikātu sarakstā katrs predikāts tiek rakstīts savās iekavās, kur tiek norādīts predikāta nosaukums, aiz kura tiek norādīti tā parametri, atdalīti ar atstarpēm, piemēram, "*(at ?obj ?loc)*". Šajā gadījumā predikāts ir "*at*", kas var pieņemt vērtību patiess vai aplams, atkarībā no saviem parametriem. Predikāta parametri ir objekti. Parametri vienmēr sākas ar jautājuma zīmi.

Predikātu saraksta piemērs:

```
(:predicates  (OBJ ?obj)
              (TRUCK ?truck)
              (LOCATION ?loc)
              (AIRPLANE ?airplane)
              (CITY ?city)
              (AIRPORT ?airport)
              (at ?obj ?loc)
              (in ?obj ?obj)
              (in-city ?obj ?city))
```

Darbību sarakstu veido neviena, viena vai vairākas darbības. Katras darbības apraksts sākas ar atverošo iekavu un atslēgas vārdu "*:action*" un noslēdzas ar attiecīgu aizverošo iekavu. Aiz šī atslēgas vārda seko darbības nosaukums un tad, kāds cits atslēgas vārds. Šajā darbā interesēsimies par atslēgas vārdiem "*:parameters*" un "*:effect*". Iekavās aiz atslēgas vārda "*:parameters*" seko parametru saraksts, piemēram, "*(?obj ?truck ?loc)*", bet aiz "*:effect*" seko predikātu stāvokļa izmaiņa, piemēram, "*(and (not (at ?obj ?loc)) (in ?obj ?truck)))*".

Darbību saraksta piemērs:

```
(:action LOAD-TRUCK
  :parameters
   (?obj
    ?truck
    ?loc)
  :precondition
   (and (OBJ ?obj) (TRUCK ?truck) (LOCATION ?loc)
   (at ?truck ?loc) (at ?obj ?loc))
  :effect
   (and (not (at ?obj ?loc)) (in ?obj ?truck)))

(:action UNLOAD-TRUCK
  :parameters
   (?obj
    ?truck
    ?loc)
  :precondition
   (and (OBJ ?obj) (TRUCK ?truck) (LOCATION ?loc)
        (at ?truck ?loc) (in ?obj ?truck))
  :effect
   (and (not (in ?obj ?truck)) (at ?obj ?loc)))
```

Savukārt plānošanas problēma īsumā tiek aprakstīta ar atsauci uz plānošanas domēnu, objektu sarakstu, predikātu sākuma stāvokli un nepieciešamo predikātu beigu stāvokli. No dotajiem datiem interesēsimies tikai par objektu sarakstu un sākuma stāvokli.

PDDL valoda pieļauj komentārus, kas sākas ar simbolu "#" vai ";" un turpinās līdz rindiņas beigām.

Ar pilnu PDDL sintaksi iespējams iepazīties PDDL 1.2 standartā(3).

1.1.2. Plānošanas programmatūra

Jau pirmās plānošanas sacensības guva lielu atsaucību un to rezultātā radās dažāda programmatūra, kas spēj apstrādāt PDDL. Pat vēl pirms pirmā IPC, kas notika 1998. gadā, jau eksistēja plānošanas programmas. Izstrādājot kursa darbu, darbību plānu radīšanai tiks izmantota plānotāja *blackbox* 42. versija, kā arī plānošanas domēnu un problēmu piemēri, kas tam nāk komplektā(8).

Darbību plāna formāts ir sekojošs. Katrā rindiņā definēta viena darbība. Rindiņa sākas ar skaitli – gājiena numuru –, tad seko darbības vārds, kam seko parametri. Pa vidu tam var būt vai nebūt saliktas iekavas. Iespējas, ka vairākas darbības izpildās vienā gājienā.

Plānotājs *blackbox* izvēlēts dēļ daudzajiem dažādajiem piemēriem, kas tam nāk līdzi. Paša plānotāja ātrdarbība vai plānošanas kvalitāte šim darbam nav būtiska. Katrs piemēra komplekts sastāv no plānošanas domēna un plānošanas problēmas. Izmantojot plānotāju *blackbox*, no piemēra komplekta tiek iegūts arī darbību plāns.

Mūsdienās plānošanas programmatūra ir kļuvusi spējīga risināt plašāku problēmu loku. Zinātnieki ir atskārtuši līdzību starp plānošanas un grafika sastādīšanas problēmām, tāpēc tagad lielākā daļa plānotāju spēj risināt abu veidu problēmas. Tomēr šī darba kontekstā tas nemaina mērķi.

1.1.3. Plānošanas domēnu piemēri

Apskatot plānotāja *blackbox* piemērus, redzam tādus plānošanas domēnus kā "*fridge-typed*", "*tire-world*", "*logistics-strips*" u.c. Pēdējais no šiem plānošanas domēniem redzams pielikumā nr. 1.1 "Plānošanas domēna piemērs: domain.pddl". Lai droši pārliecinātos par kursa darba hipotēzes patiesumu, izmantosim vēl vienu papildus plānošanas domēnu "*freecell*", kas apraksta kāršu kavu un kāršu spēli *freecell*.

1.2. Alternatīvas teksta vizualizācijas programmas

Lai pārliecinātos, ka hipotēze jau nav pierādīta, kā arī lai apskatītu līdzīgu jau eksistējošo datorprogrammu iespējas, tika pētīti eksistējoši risinājumi.

1.2.1. Datorprogramma *CarSim*

Izdevās sameklēt vienu kaut nedaudz līdzīgu programmu, kas realizē teksta pārveidi par saturīgu attēlu vai animāciju. Tā ir Zviedrijas valdības izmantotā datorprogramma *CarSim*(9), kas ir spējīga pārveidot rakstītu zviedru valodas tekstu par XML veidni, ko savukārt tā var pārveidot par 3-dimensiju animāciju, kurā gan pazūd praktiski visas nianses.

Varētu šķist, ka hipotēze ir pierādīta, taču *CarSim* tiek galā tikai ar autoavāriju aprakstiem(4). Līdz ar to neizpildās nosacījums, ka datorprogrammai ir jāspēj automatizēti "saprast" tekstu jebkurā valodas domēna apakškopā.

Jāpiebilst, ka arī šādas datorprogrammas viennozīmīgi ir noderīgas, piemēram, darbam ar lielu tekstuālās informācijas apjomu, kur lasītājam nav visās detaļās jāizpēta teksta semantiskā jēga, bet gan tikai jārodas priekšstatam par tā saturu. Vēl šādas datorprogrammas varētu būt lietderīgas, lai ļautu cilvēkiem, kas tādu vai citu iemeslu dēļ nespēj lasīt, iepazīties ar tekstu vizuāli. Taču šādas datorprogrammas nav šī darba mērķis, tamdēļ tālāk netiks apskatītas.

1.3. Nepieciešamais rezultāts

Ir noskaidrots, ka hipotēze pagaidām nav pierādīta, tāpēc būs jācenšas izveidot datorprogrammu, kas spēj darboties visā valodas domēnā (vai vismaz stipri plašākā valodas domēnā nekā *CarSim*). Lai maksimāli tuvotos hipotēzes pierādīšanai, datorprogrammas konceptuālajai iestrādnei ir jābūt ļoti elastīgai un tajā jāizvairās no konstrukcijām, kas ierobežotu tās pielietošanu kādam plānošanas domēnam. Darba autors uzskata, ka uzskatāmības labad šai konceptuālajai datorprogrammai vajadzētu automatizēti radīt 2-dimensiju animāciju.

1.3.1. Ierobežojumi

Kursa darba gaitā radītajai datorprogrammai, vajadzētu spēt strādāt ar vismaz 100 darbībām, un līdz ar to arī ar vismaz 100 objektiem un 100 gājieniem. Plānošanas domēnā definētajām darbībām ir jābūt determinētām, jo rakstītas valodas "uztverei" arī tādai jābūt, citādi nevar runāt par teksta viennozīmīgu "sapratni".

Iestrādne var neatbalstīt universālkvantorus un nosacītos efektus, jo, lai arī tie ir definētas PDDL 1.2, viņi kursa darba mērķa kontekstā nerada papildus iespējas vai apgrūtinājumus. Tos iespējams priekšapstrādāt ar samērā vienkāršu lineāru algoritmu.

2. RISINĀJUMS

Šajā sadaļā ir aprakstīta datorprogrammas izveides gaita un pati datorprogramma, kā arī hipotēzes pārbaudes veikšana.

2.1. Sagatavošanās

2.1.1. Līdzekļu izvēle

Programmu būtu iespējams izstrādāt jebkādā izstrādes vidē, taču tika apsvērti divi varianti: *C++* un *PHP*. No dotajām datorprogrammu izstrādes valodām tika izvēlēts *PHP*, jo tā ir elastīgāka savās konstrukcijās, kā arī tai ir plaša pieejamo funkciju bāze, kas nozīmē, ka izstrādāt programmu ir ērtāk. Ne mazāk svarīgs arguments bija tas, ka *PHP* valodā izstrādātu programmu būs viegli apvienot ar projekta *SemTi-Kamols* esošo programmas bāzi.

Tika izvēlēta tieši *PHP* 5. versija(1), nevis 4. versija, jo tai ir uzlabots objektorientētās programmēšanas atbalsts, kas šādas sarežģītības konceptuālas iestrādnes veidošanā var palīdzēt.

Kā animācijas moduli tika izvēlēts *Ming* – bibliotēku, kas spēj veidot SWF animācijas. *Ming* ir veidots valodā *C*, taču pieejams kā modulis daudzām programmēšanas valodām: *C++*, *PHP*, *Python*, *Ruby* un *Perl*(10). Alternatīvi autors apdomāja izmantot GIF, taču šim formātam ir sliktāks atbalsts izvēlētajā programmēšanas vidē, kā arī tā nespēj mijiedarboties ar lietotāju. SWF savukārt to spēj (tas varētu būt noderīgi papildinot konceptuālo datorprogrammas iestrādni ar jaunu funkcionalitāti lietotāja ērtībai), taču SWF vairāk nekā GIF noslogo lietotāja datora procesoru.

2.1.2. Programmas pamatdaļas

Programmu tika nolemts izstrādāt tādā veidā, ka to viegli adaptēt vajadzīgajai situācijai, tamdēļ programmas pamatfails tika veidots maksimāli pielāgojams. Pamatfails nodrošina programmas saskarni – tas ielasa plānošanas domēna failu, plānošanas problēmas failu, darbību plānu failu, kā arī predikātu izskata apraksta failu un izsauc programmas galvenās daļas. Pamatfails redzams pielikumā nr. 2.1: “Programmas pamatfails”.

Programma sastāv no divām galvenajām daļām – PDDL parsētāja un SWF ģeneratora. Šos failus iespējams apskatīt attiecīgi pielikumā nr. 2.2: “PDDL parsētājs: pddl-parser.php" un pielikumā nr. 2.3: “SWF ģenerators: flash-generator.php”.

PDDL parsētājs apstrādā plānošanas domēnu, plānošanas problēmu, darbību plānu un predikātu izskata aprakstu, uzģenerējot konkrētu "pasauli" – ievaddatu strukturētu reprezentāciju datora atmiņā. (Skat. pielikumu nr. 3: "PDDL parsētāja izveidotais atmiņas struktūras fragments")

SWF ģenerators savukārt, apstrādā "pasauli", izveidojot no tās animāciju. Tas var ņemt palīgā objektu sākotnējo pozīciju sarakstu, taču tas nav obligāti. Ja šāds saraksts nav definēts, ģenerators izvieto objektus patvaļīgi.

SWF ģenerators izmanto palīgfailu, lai varētu pārveidot krāsas no angļu valodas uz RGB(11). Palīgfails redzams pielikumā nr. 2.4: "Palīgfails: colornames.php".

Programmējot PDDL parsētāju un SWF ģeneratoru tika izmantota literatūra(2,12).

Datorprogrammas konceptuālā iestrādne veidota tā, ka kļūdas gadījumā tā izmet izņēmuma situāciju, ko ir iespējams noķert izsaucošajā programmā. Ne vienmēr tas beigsies ar pozitīvu rezultātu, taču situācijas, kas nav kritiskas, šādi būs iespējams apstrādāt, vienlaikus nepalaižot tās garām pilnīgi nepamanītas.

2.2. Predikātu izskata apraksta formāts

Predikātu izskata aprakstu veido viens vai vairāki predikātu apraksti. Katrs predikāta izskata apraksts sākas ar atverošo iekavu un atslēgas vārdu ":predicate" un noslēdzas ar attiecīgu aizverošo iekavu. Aiz šī atslēgas vārda seko predikāta nosaukums un tad, kāds cits no atslēgas vārdiem "*:movement*", "*:style*" vai "*:logic*". Pirms predikāta nosaukuma var atrasties izsaukuma zīme (neatdalot to ar atdalošajiem simboliem). Tas nozīmes, ka apraksts attiecas uz gadījumu, kad predikāts kļūst aplams. Citādi apraksts attieksies uz gadījumu, kad predikāts kļūst patiess.

Aiz katra no atslēgas vārdiem iekavās seko viens no darbības parametriem – tas, uz ko attieksies turpmākās īpašības. Ja atslēgas vārda nav vai tas ir "*world*", īpašības attieksies uz pašu animācijas vidi.

Pēc tam seko atverošā iekava, tad īpašības nosaukumus un konkrētās īpašības vērtība iekavās. Tad seko aizverošā iekava. Piemēram, ":logic(?x)(:attach(?y))". Ar predikātu izskata apraksta formāta paraugu iespējams iepazīties pielikumā nr. 1.4: "Predikātu izskata apraksta piemērs: predicates".

Atslēgas vārds "*:movement*" norāda, ka tiks aprakstīta objekta kustība. Tas var saturēt īpašību "*:location*", kas norāda, uz kura objekta atrašanās vietu pārvietot doto objektu. Vai arī tas var saturēt īpašību "*:style*", kurai iespējams trīs vērtības "*smooth*", "*before*" un "*after*". Šī īpašība norāda, kādā veidā animēt doto objektu:

- vērtība "*smooth*" animēs objektu ar lēzenu pāreju no viena stāvokļa vai pozīcijas otrā;
- vērtība "*before*" animēs objektu lēcienveidīgi *pirms* tiks animēti objekti ar vērtībām "*smooth*" un "*after*";
- vērtība "*after*" animēs objektu lēcienveidīgi *pēc tam*, kad objekti ar vērtībām "*smooth*" un "*before*" jau būs animēti.

Atslēgas vārds "*:style*" norāda, ka tiks aprakstīts objekta izskats. Tas var saturēt īpašību "*:resize-to-fit*". Šai īpašībai nav vērtības un tā norāda, ka objektu ir jāpārzīmē citā izmērā tā, lai tajā vizuāli satilpst pārējie objekti, kas tajā atrodas. Šis atslēgas vārds var saturēt arī šādas īpašības:

- "*visible*", kas norāda objekta redzamību; pieļaujamās vērtības ir "*yes*", "*1*", "*true*" u.tml. (redzams) vai "*no*", "*false*", "*0*" u.tml. (slēpts);
- "*shape*", kas norāda objekta ieņemamo formu; šobrīd tiek atbalstītas vērtības "*circle*", "*triangle*", "*rectangle*", "*square*", kas attiecīgi apzīmē apli, trijstūri, taisnstūri un kvadrātu;
- "*opacity*", kas norāda objekta caurspīdīgumu; vērtības var būt skaitlis no 0 (pilnīgi caurspīdīgs) līdz 255 (pilnīgi necaurspīdīgs);
- "*size*", kas norāda objekta izmēru pikseļos vienā dimensijā;
- "*color*", kas norāda objekta krāsu angļu valodā; pieļaujamās vērtības redzamas pielikumā nr. 2.4: "Palīgfails: colornames.php";
- "*border*", kas norāda objekta apmales krāsu angļu valodā.

Atslēgas vārds "*:logic*" norāda, ka tiks aprakstīts objekta loģiskais modelis. Tas var saturēt īpašību "*:attach*" vai "*:detach*", kas attiecīgi nozīmē pievienot un atvienot. Šīs īpašības vērtība ir cits parametrs, kuru vajag piesaistīt vai atsaistīt dotajam. Piesaistot objektu, tas turpmāk sekos savam vecākam, kamēr tiks atsaistīts. Mēģinot piesaistīt objektu, kurš jau ir piesaistīts, tiks izsaukta kļūdas situācija.

Šis atslēgas vārds var saturēt arī īpašību "*:resize-to-fit*", kas tiek definēta un darbojas identiski tās darbībai atslēgas vārdā "*:style*".

Ar laiku kursa darbā izstrādātas datorprogrammas konceptuālo iestrādni iespējams vienkārši papildināt ar jaunām īpašībām.

Predikātu izskata apraksta piemērs:

```
(:predicate in-city
  :parameters (?x ?y)
  :movement (?x) (
    :location (?y)
    :style (smooth)
  )
  :logic (?x) (
    :attach (?y)
  )
  :visual (?y) (
    :resize-to-fit
  )
))


(:predicate !in-city
  :parameters (?x ?y)
  :logic (?x) (
    :detach (?y)
  )
  :visual (?y) (
    :resize-to-fit
  )
))


(:predicate OBJ
  :parameters (?x)
  :visual (?x) (
    :shape (circle)
    :color (red)
    :visible (yes)

))
```

2.3. Objektu sākotnējo pozīciju saraksta formāts

Objektu sākotnējo pozīciju sarakstu veido neviens, viens vai vairāki apraksti, kas katrs novietots savā faila rindā. Viens apraksts sastāv no objekta nosaukuma, atstarpes, šī objekta sākotnējās x-koordinātes, atstarpes un šī objekta sākotnējās y-koordinātes.

Animācijas sākumā visi objekti, kuru nosaukumi atradīsies objektu sākotnējo pozīciju sarakstā, tiks novietoti attiecīgās pozīcijās. Ja šajā sarakstā atradīsies objekts ar nosaukumu "*#world*", tad parametri tiks izmantoti, lai uzstādītu animācijas laukuma izmērus.

Objektu sākotnējo pozīciju saraksta piemērs:

```
#world 1000 700
bos 200 170
la 200 520
pgh 700 300
```

2.4. Hipotēzes pārbaude

Lai pārbaudītu hipotēzi – vai valodas tekstu ir iespējams "saprast" arī nezinot **precīzu** tā domēnu (apakškopu no visa latviešu valodas domēna) –, tika veiktas šādas pārbaudes, izmantojot izveidoto datorprogrammas konceptuālo iestrādni:

- viena plānošanas domēna – "*logistics-strips*" – ietvaros ar *blackbox* palīdzību tika atrisinātas dažādas plānošanas problēmas, kā rezultāta tika iegūts darbību plāns; pēcāk ar izstrādātās datorprogrammas palīdzību tika vizualizētas šīs plānošanas problēmas kopā ar attiecīgajiem darbību plāniem;
- definējot predikātu izskata aprakstu un izvēloties citu plānošanas domēnu – "*freecell*" –, tika vizualizēta konkrēta plānošanas problēma ar darbību plānu.

Ja, veicot šīs pārbaudes, vizualizācija izrādījās korekta, tad valodas tekstu ir iespējams "saprast" arī nezinot **precīzu** tā domēnu.

3. REZULTĀTI

Kursa darba gaitā autors iepazinās ar PDDL standartu un apguva šo valodu. Darba rezultātā ir tikusi izanalizēta programma *CarSim* un noskaidrots, ka šī programma, lai arī ir noderīga, jo ļauj cilvēkiem, kas nespēj vai nevēlas lasīt tekstu, iepazīties ar šo informāciju vizuāli, tomēr realizē vizualizāciju vienā konkrētā domēnā, tamdēļ nav uzstādītās hipotēzes pierādījums.

Darba gaitā tikusi izveidota datorprogrammas konceptuāla iestrādne funkcionējošas datorprogrammas līmenī, kas spēj vizualizēt plānošanas domēnu, plānošanas problēmu, darbību plānu un predikātu izskata aprakstu, pārvēršot tos par animāciju.

Programma realizēta valodā *PHP* 5, animācijas izvadam izmantojot *Ming* moduli, kas nodrošina 2-dimensiju SWF animācijas izveidi. Realizācija veikta modulāri, veidojot klases un grupējot saistītas funkcijas vienā failā, bet nesaistītas – citā. Realizācijas gaitā netika uzstādīti nekādi ierobežojumi uz darbību, objektu vai gājienu skaitu.

Izstrādātā datorprogramma pagaidām neatbalsta universālkvantorus un nosacītos efektus, jo tie kursa darba mērķa kontekstā nerada papildus iespējas var apgrūtinājumus.

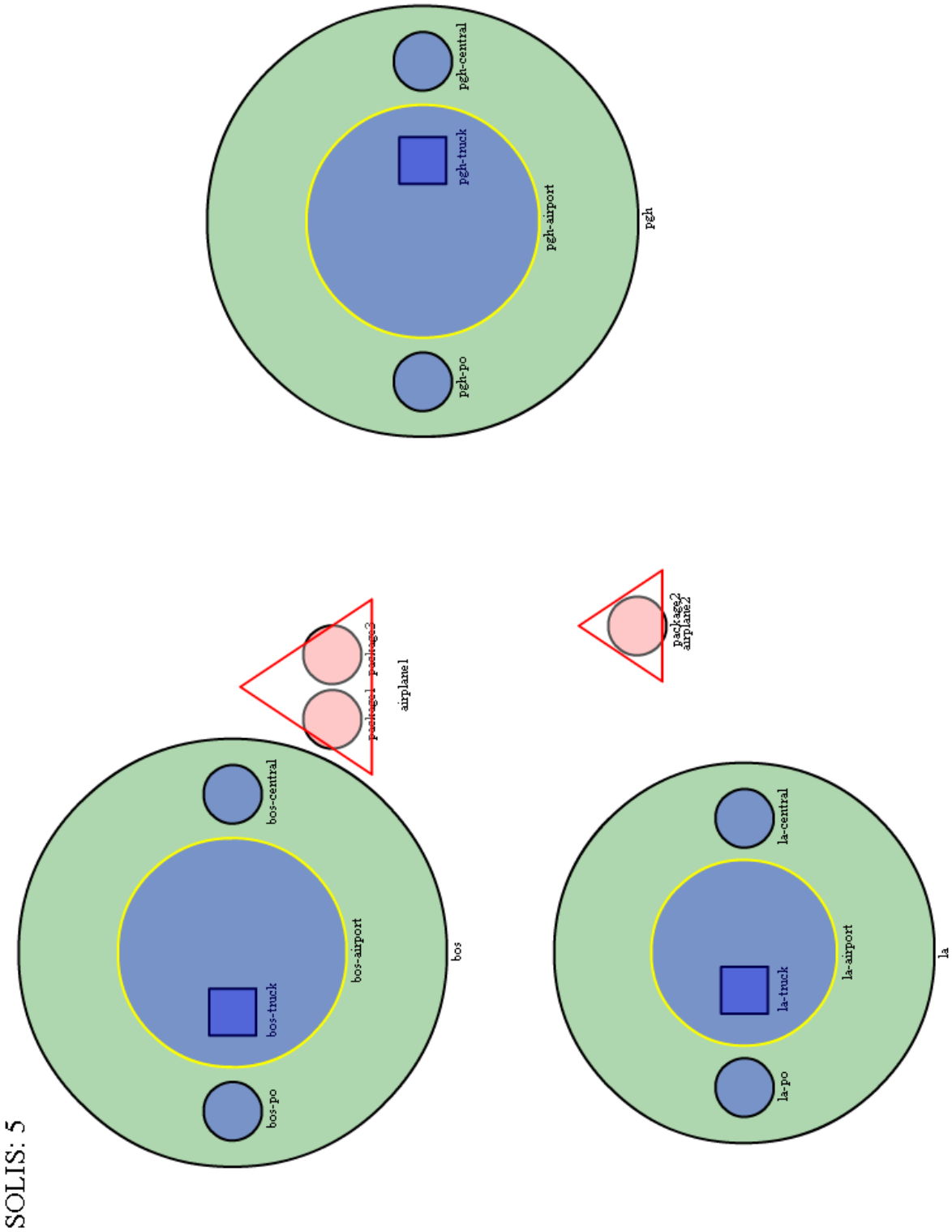
Darba nobeigumā datorprogrammas konceptuāla iestrādne tika testēta, izmantojot kursa darba pielikumā esošo plānošanas domēnu, uzdevumu, darbību plānu, predikātu izskata aprakstu un objektu sākotnējo pozīciju sarakstu (attiecīgi pielikums nr. 1.1: “Plānošanas domēna piemērs: domain.pddl”, pielikums nr. 1.2: “Plānošanas uzdevuma piemērs: problem.pddl”, pielikums nr. 1.3: “Darbību plāna piemērs: solution.plan”, pielikums nr. 1.4: “Predikātu izskata apraksta piemērs: predicates” un pielikums nr. 1.5: “Objektu sākotnējo pozīciju saraksta piemērs: positions”).

Konkrētā darbību plāna 4. soļa beigas – paciņas ir tikko salādētas lidmašīnās – redzamas 3.1. att.

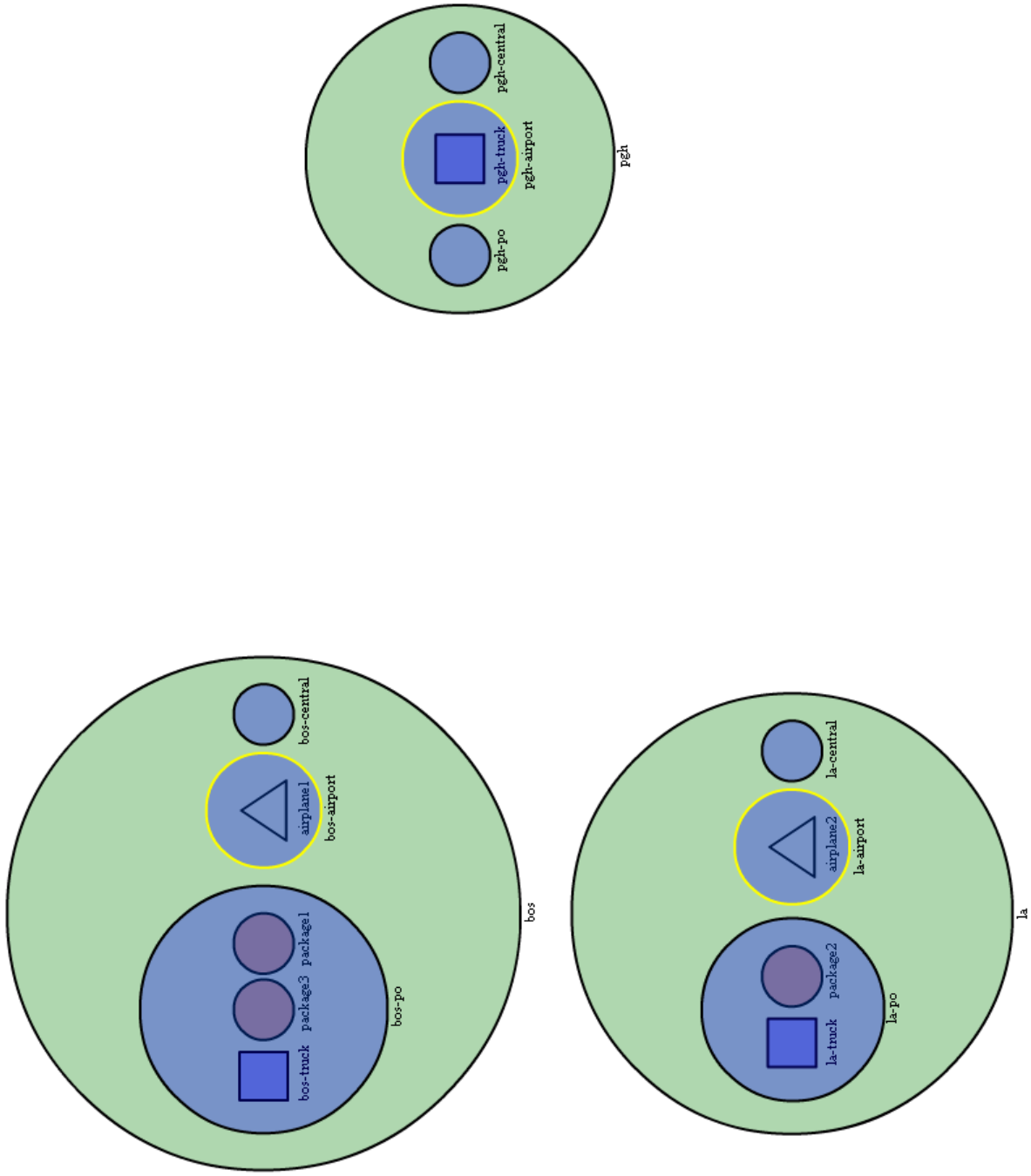
5. soļa vidus – lidmašīnas, pārvadājot paciņas, kas tām piesaistītas ar “*:attach*” īpašību, dodas katra uz savu lidostu – redzams 3.2. att.

3.3. att. redzams stāvoklis pēc visu soļu animēšanas; vizualizācijas ir beigusies un visi objekti atrodas savās gala pozīcijās un koordinātās.

3.1. att. **Vizualizācijas 4. solis: objekti "*package*" salādēti objektos "*airplane*"**



3.2. att. **Vizualizācijas 5. solis: objekti "*airplane*" ar tajos esošajiem bērniem "lido" uz objektiem "*airport*"**



3.3. att. **Vizualizācijas beigas: visi objekti ir nokļuvuši savās gala koordinātēs**

4. SECINĀJUMI

Noslēdzot darbu, var secināt, ka tika radīta konceptuāla iestrādne datorprogrammai, kas spēj pārvērst plānošanas domēnu, problēmu, darbību plānu un predikātu izskata aprakstu par animāciju. Šī datorprogramma ģenerē 2-dimensiju SWF animāciju un atšķiras no *CarSim* ar to, ka spēj darboties vairākos dažados domēnos. Izveidotās datorprogrammas maksimāli apstrādājamo darbību, objektu un gājienu daudzums ir ierobežots vienīgi ar datora, kas ģenerē animāciju, atmiņas apjomu un datora, kas attēlo animāciju, jaudu.

Datorprogramma ir diezgan elastīga, jo tā nav piesaistīta vienam konkrētam plānošanas domēnam. Šī programma spēj veiksmīgi funkcionēt dažādos diezgan atšķirīgos problēmu domēnos, kā piemēram "*freecell*" un "*logistics-strips*".

Līdz ar to var secināt, ka šī programma spēj automatizēti vizualizēt jebkuru plānošanas problēmu neatkarīgi no domēna. Tā kā programma spēj vizualizēt plānošanas problēmas, nezinot precīzu domēnu, tad programma spēj šo tekstu "iztēloties", kas savukārt noved pie secinājuma, ka programma "saprot" šo tekstu.

No augstākminētā var secināt, ka darbā uzstādītā hipotēze – valodas tekstu ir iespējams "saprast" arī nezinot **precīzu** tā domēnu (apakškopu no visa latviešu valodas domēna) – ir pierādīta.

PATEICĪBAS

Vēlos izteikt pateicību cilvēkiem bez kuriem šī darba tapšanas process būtu sāpīgs un grūts.

- ✔ Savai mātei Larisai Solovjovai, kas ir tērējusi daudz naudas, mani skolojot, un ir mani garīgi un morāli atbalstījusi kursa darba izstrādes un visu līdzšinējo studiju laikā.
- ✔ Savai Universitātei, bet īpaši savam darba vadītājam Guntim Bārzdiņam, par neizmērojam pretimnākšanu un neskaitāmajām atbildēm uz maniem neskaitāmajiem jautājumiem.
- ✔ Saviem draugiem par to, ka piecieta manu aizņemtību, tomēr nepameta grūtā brīdi.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. Grāmatas

1. **Holzner, S.** The Complete Reference. PHP. New York: McGraw-Hill, 2007. 590 p.

2. **Wilson, T., Greant Z., Merral G., Michlitsch B.** *PHP Functions essential reference.* California: Sams, 2001. 768 p.

2. Standarti

3. PDDL — The Planning Domain Definition Language. Version 1.2.

3. Zinātniski pētnieciskie darbi

4. **Åkerberg, O., Svensson, H., Schulz B., Nugues P.** *CarSim: An Automatic 3D Text-to-Scene Conversion System Applied to Road Accident Reports.* Lund:Lund University, LTH, 2003. 4 p.

4. Elektroniskie informācijas avoti

5. *Semantiskais tīmeklis - Par projektu* [tiešsaiste] – [atsauce 25.05.2008.] Pieejams: http://www.semti-kamols.lv/?sadala=185

6. *Video "Avārijas Brigāde – Latvijas dzelzceļš: bumba"* [tiešsaiste] – [atsauce 15.06.2008.] Pieejams: http://www.youtube.com/watch?v=NebPYc3okoA

7. *PDDL papers* [tiešsaiste] – [atsauce 29.05.2008.] Pieejams: http://ipc.informatik.uni-freiburg.de/PddlResources

8. *blackbox, a SAT technology planning system* [tiešsaiste] – [atsauce 23.05.2008.] Pieejams: http://www.cs.rochester.edu/u/kautz/satplan/blackbox/index.html

9. *Development of a Text-to-Scene Converter for Vehicle Accident Reports* [tiešsaiste] – [atsauce 23.05.2008.] Pieejams: http://nlp.cs.lth.se/carsim.shtml

10. *Libming Wiki* [tiešsaiste] – [atsauce 27.05.2008.] Pieejams: http://www.libming.org/

11. *HTML Color Names* [tiešsaiste] – [atsauce 14.06.2008.] Pieejams: http://www.w3schools.com/HTML/html_colornames.asp

12. *Ming the Manual* [tiešsaiste] – [atsauce 27.05.2008.] Pieejams: http://ming.sourceforge.net/docs

PATSTĀVĪBAS APLIECINĀJUMA FORMA

Ar savu parakstu apliecinu, ka šodien iesniegto kursa darbu es esmu veicis pašrocīgi un esmu izmantojis tikai tajā norādītos palīglīdzekļus.

Rīgā, 2008. gada __. jūnijā.

Paraksts:

PIELIKUMU SARAKSTS

Kursa darbam ir pievienoti sekojoši pielikumi:

- Pielikums nr. 1.1: “Plānošanas domēna piemērs: domain.pddl”
  → Pielikums satur plānošanas domēna failu, kas izmantots, ģenerējot kursa darbā esošos attēlus, darbību plānu un atmiņas struktūru.
- Pielikums nr. 1.2: “Plānošanas uzdevuma piemērs: problem.pddl”
  → Pielikums satur plānošanas uzdevuma failu, kas izmantots, ģenerējot kursa darbā esošos attēlus, darbību plānu un atmiņas struktūru.
- Pielikums nr. 1.3: “Darbību plāna piemērs: solution.plan”
  → Pielikums satur darbību plāna failu, kas ģenerēts no pielikumos esošā plānošanas domēna un plānošanas uzdevuma.
- Pielikums nr. 1.4: “Predikātu izskata apraksta piemērs: predicates”
  → Pielikums satur predikātu izskata apraksta failu, kas rakstīts ar roku, atbilstoši pielikumā esošajam plānošanas domēnam.
- Pielikums nr. 1.5: “Objektu sākotnējo pozīciju saraksta piemērs: positions”
  → Pielikums satur objektu sākotnējo pozīciju saraksta failu, kas veidots ar roku, atbilstoši pielikumā esošajam plānošanas uzdevumam; šis fails izmantots, ģenerējot kursa darbā esošos attēlus.
- Pielikums nr. 2.1: “Programmas pamatfails”
  → Pielikums satur kursa darbā izstrādāto programmu (pamatfailu).
- Pielikums nr. 2.2: “PDDL parsētājs: pddl-parser.php"
  → Pielikums satur PDDL parsētāju (programmas sastāvdaļu).
- Pielikums nr. 2.3: “SWF ģenerators: flash-generator.php”
  → Pielikums satur SWF ģeneratoru (programmas sastāvdaļu).
- Pielikums nr. 2.4: “Palīgfails: colornames.php”
  → Pielikums satur palīgfailu, ar kura palīdzību var pārveidot krāsas no angļu valodas uz RGB.
- Pielikums nr. 3: “PDDL parsētāja izveidotais atmiņas struktūras fragments”
  → Pielikums satur nelielu fragmentu no atmiņas struktūras, kas izveidojas, kad tiek programmā tiek ielādēti pielikumos esošie plānošanas faili.

PIELIKUMS NR. 1.1: "PLĀNOŠANAS DOMĒNA PIEMĒRS: DOMAIN.PDDL"

```
;; logistics domain
;;
;; logistics-strips: pure strips, no types, no equality
;;    Compatible with problem instances for the domain01.pddl
;;    from the PDDL 1.3 distribution.
;;    Fixes a few missing parameter predicates and
;;    has unload truck and plane as separate operators
;;    (as in original formulation).
;;    12 June 1998, Bart Selman & Henry Kautz

(define (domain logistics-strips)
  (:requirements :strips)
  (:predicates       (OBJ ?obj)
              (TRUCK ?truck)
                     (LOCATION ?loc)
              (AIRPLANE ?airplane)
                (CITY ?city)
                (AIRPORT ?airport)
              (at ?obj ?loc)
              (in ?obj ?obj)
              (in-city ?obj ?city))

(:action LOAD-TRUCK
  :parameters
   (?obj
    ?truck
    ?loc)
  :precondition
   (and (OBJ ?obj) (TRUCK ?truck) (LOCATION ?loc)
   (at ?truck ?loc) (at ?obj ?loc))
  :effect
   (and (not (at ?obj ?loc)) (in ?obj ?truck)))

(:action LOAD-AIRPLANE
  :parameters
   (?obj
    ?airplane
    ?loc)
  :precondition
   (and (OBJ ?obj) (AIRPLANE ?airplane) (LOCATION ?loc)
   (at ?obj ?loc) (at ?airplane ?loc))
  :effect
   (and (not (at ?obj ?loc)) (in ?obj ?airplane)))

(:action UNLOAD-TRUCK
  :parameters
   (?obj
    ?truck
    ?loc)
  :precondition
   (and (OBJ ?obj) (TRUCK ?truck) (LOCATION ?loc)
        (at ?truck ?loc) (in ?obj ?truck))
  :effect
   (and (not (in ?obj ?truck)) (at ?obj ?loc)))

(:action UNLOAD-AIRPLANE
  :parameters
   (?obj
    ?airplane
    ?loc)
```

```
  :precondition
   (and (OBJ ?obj) (AIRPLANE ?airplane) (LOCATION ?loc)
        (in ?obj ?airplane) (at ?airplane ?loc))
  :effect
   (and (not (in ?obj ?airplane)) (at ?obj ?loc)))

(:action DRIVE-TRUCK
  :parameters
   (?truck
    ?loc-from
    ?loc-to
    ?city)
  :precondition
   (and (TRUCK ?truck) (LOCATION ?loc-from) (LOCATION ?loc-to) (CITY ?city)
   (at ?truck ?loc-from)
   (in-city ?loc-from ?city)
   (in-city ?loc-to ?city))
  :effect
   (and (not (at ?truck ?loc-from)) (at ?truck ?loc-to)))

(:action FLY-AIRPLANE
  :parameters
   (?airplane
    ?loc-from
    ?loc-to)
  :precondition
   (and (AIRPLANE ?airplane) (AIRPORT ?loc-from) (AIRPORT ?loc-to)
       (at ?airplane ?loc-from))
  :effect
   (and (not (at ?airplane ?loc-from)) (at ?airplane ?loc-to)))
)
```

PIELIKUMS NR. 1.2: "PLĀNOŠANAS UZDEVUMA PIEMĒRS: PROBLEM.PDDL"

```
;; original name logistics.easy
;; (:length (:parallel 9))
;; optimal
;;

(define (problem log001)
    (:domain logistics-strips)
    (:objects
        package1
        package2
        package3

        airplane1
        airplane2

        pgh
        bos
        la

        pgh-truck
        bos-truck
        la-truck

        pgh-po
        bos-po
        la-po

        pgh-central
        bos-central
        la-central

        pgh-airport
        bos-airport
        la-airport
    )
    (:init
        (OBJ package1)
        (OBJ package2)
        (OBJ package3)

        (AIRPLANE airplane1)
        (AIRPLANE airplane2)

        (CITY pgh)
        (CITY bos)
        (CITY la)

        (TRUCK pgh-truck)
        (TRUCK bos-truck)
        (TRUCK la-truck)

        (LOCATION bos-po)
        (LOCATION la-po)
        (LOCATION pgh-po)

        (LOCATION bos-central)
        (LOCATION la-central)
        (LOCATION pgh-central)

        (AIRPORT bos-airport)
```

```
        (LOCATION bos-airport)
        (AIRPORT pgh-airport)
        (LOCATION pgh-airport)
        (AIRPORT la-airport)
        (LOCATION la-airport)

        (in-city pgh-po pgh)
        (in-city pgh-airport pgh)
        (in-city pgh-central pgh)

        (in-city bos-po bos)
        (in-city bos-airport bos)
        (in-city bos-central bos)

        (in-city la-po la)
        (in-city la-airport la)
        (in-city la-central la)

        (at package1 pgh-po)
        (at package2 pgh-po)
        (at package3 pgh-po)

        (at airplane1 pgh-airport)
        (at airplane2 pgh-airport)

        (at bos-truck bos-po)
        (at pgh-truck pgh-po)
        (at la-truck la-po)

    )
    (:goal (and
        (at package1 bos-po)
        (at package2 la-po)
        (at package3 bos-po)
    ))
)
```

PIELIKUMS NR. 1.3: “DARBĪBU PLĀNA PIEMĒRS: SOLUTION.PLAN”

```
1 (load-truck package2 pgh-truck pgh-po)
1 (drive-truck bos-truck bos-po bos-airport bos)
1 (load-truck package3 pgh-truck pgh-po)
1 (drive-truck la-truck la-po la-airport la)
1 (load-truck package1 pgh-truck pgh-po)
2 (drive-truck pgh-truck pgh-po pgh-airport pgh)
3 (unload-truck package3 pgh-truck pgh-airport)
3 (unload-truck package2 pgh-truck pgh-airport)
3 (unload-truck package1 pgh-truck pgh-airport)
4 (load-airplane package1 airplane1 pgh-airport)
4 (load-airplane package2 airplane2 pgh-airport)
4 (load-airplane package3 airplane1 pgh-airport)
5 (fly-airplane airplane2 pgh-airport la-airport)
5 (fly-airplane airplane1 pgh-airport bos-airport)
6 (unload-airplane package1 airplane1 bos-airport)
6 (unload-airplane package2 airplane2 la-airport)
6 (unload-airplane package3 airplane1 bos-airport)
7 (load-truck package2 la-truck la-airport)
7 (load-truck package1 bos-truck bos-airport)
7 (load-truck package3 bos-truck bos-airport)
8 (drive-truck bos-truck bos-airport bos-po bos)
8 (drive-truck la-truck la-airport la-po la)
9 (unload-truck package3 bos-truck bos-po)
9 (unload-truck package2 la-truck la-po)
9 (unload-truck package1 bos-truck bos-po)
```

PIELIKUMS NR. 1.4: “PREDIKĀTU IZSKATA APRAKSTA PIEMĒRS: PREDICATES”

```
(:predicate in-city
  :parameters (?x ?y)
  :movement (?x) (
    :location (?y)
    :style (smooth)
  )
  :logic (?x) (
    :attach (?y)
  )
  :visual (?y) (
    :resize-to-fit
  )
))

(:predicate !in-city
  :parameters (?x ?y)
  :logic (?x) (
    :detach (?y)
  )
  :visual (?y) (
    :resize-to-fit
  )
))

(:predicate OBJ
  :parameters (?x)
  :visual (?x) (
    :shape (circle)
    :color (red)
    :visible (yes)

))

(:predicate TRUCK
  :parameters (?x)
  :visual (?x) (
    :shape (square)
    :color (blue)
    :visible (yes)
))

(:predicate LOCATION
  :parameters (?x)
  :visual (?x) (
    :shape (circle))
    :color (blue)
    :visible (yes)
))

(:predicate AIRPLANE
  :parameters (?x)
  :visual (?x) (
    :shape (triangle)
    :color (white)
    :visible (yes)
))

(:predicate CITY
  :parameters (?x)
  :visual (?x) (
```

```
    :shape (circle)
    :color (green)
    :visible (yes)
))

(:predicate AIRPORT
  :parameters (?x)
  :visual (?x) (
    :shape (circle)
    :color (yellow)
    :visible (yes)
    :border (yellow)
))

(:predicate at
  :parameters (?x ?y)
  :movement (?x) (
    :location (?y)
    :style (smooth)
  )
  :visual (?y) (
    :resize-to-fit
  )
  :logic (?x) (
   :attach (?y)
  )
))

(:predicate !at
  :parameters (?x ?y)
  :visual (?y) (
    :resize-to-fit
  )
  :logic (?x) (
   :detach (?y)
  )
))

(:predicate in
  :parameters (?x ?y)
  :movement (?x) (
    :location (?y)
    :style (smooth)
  )
  :logic (?x) (
    :attach (?y)
  )
  :visual (?y) (
    :resize-to-fit
    :border (red)
  )
))

(:predicate !in
  :parameters (?x ?y)
  :logic (?x) (
    :detach (?y)
  )
  :visual (?y) (
    :border (black)
  )
))
```

PIELIKUMS NR. 1.5: “OBJEKTU SĀKOTNĒJO POZĪCIJU SARAKSTA PIEMĒRS: POSITIONS”

```
#world 1000 700
bos 200 170
la 200 520
pgh 700 300
```

PIELIKUMS NR. 2.1: “PROGRAMMAS PAMATFAILS”

```
<?php

define('DEBUG',1);

if(DEBUG){
 error_reporting(E_ALL);
 ini_set('display_errors', '1');
 $debugfile=fopen('debug.log','w');
}

function debug($text){
 if(!DEBUG)
  return;
 fputs($GLOBALS['debugfile'],$text."\n");
  fflush($GLOBALS['debugfile']); //gadiijumam, ja programma peekshnji apstaajas
}

require 'pddl-parser.php';
require 'flash-generator.php';

$testworld=new PDDL;

$testworld->loaddomain(file_get_contents('domain.pddl'),
file_get_contents('predicates'));
$testworld->loadtask(file_get_contents('problem.pddl'),
file_get_contents('solution.plan'));

$testanim=new animation;
$testanim->create($testworld,file_get_contents('positions'));
$testanim->create($testworld);
$testanim->display();

if(DEBUG)
 fclose($debugfile);
?>
```

PIELIKUMS NR. 2.2: “PDDL PARSĒTĀJS: PDDL-PARSER.PHP"

```
<?php

define('WORLDNUMBER',-1); // jaabuut negatiivam, lai nepaarklaatos ar lietotaaja
objektu ID

class predicate {
 var $id;
 var $name;
 var $paramcount;
 var $params=array();  //masiivs ar parametru nosaukumiem
 var $negmodels=array(); //masiivs ar model tipa elementiem, ko izpildiit nolieguma
gadiijumaa
 var $posmodels=array(); // --"-- apstiprinaajuma gadiijumaa
 // indeksi noraada, kuram predikaata parametram izpildiit modeli
 // ja indekss ir WORLDNUMBER, tad modelis jaaizpilda uz #world
}

class pobject {
 var $name;
 var $id;
 var $type;
}

class action {
 var $id;
 var $name;
 var $triggers=array(); // masiivs ar command tipa elementiem
 var $params=array(); //masiivs ar parametru nosaukumiem
}

class command{
 var $predicate;  // predikaata nosaukums, ko izpildiit
 var $params=array(); //masiivs ar skatiljiem - kurus (saakot no 0.)  darbiibas
parametrus izmantot kaa predikaata parametrus
 var $sign; // vai predikaats tiek izpildiits noliegumaa vai apstiprinaajumaa
}

class task{
 var $action; // darbiibas nosaukums
 var $params=array(); //masiivs ar objektu nosaukumiem, kas ir shiis konkreetaas
darbiibas parametri
}

$MODELTYPES=array('visual','movement','logic');
class model{ // visi masiivi saturees ipashiibu nosaukumus kaa indeksus un iipashiibas
kaa iipashiibu veertiibas
 var $visual=array();
 var $movement=array();
 var $logic=array();
}

class PDDL {
 var $predicates,$actions,$objects,$initcommands;
 var $commands=array();

 function normalize($text){ //paarveidot PDDL tekstu par vienmeeriigu tekstu

  $ignore=0; $prev='';$norm='';
 /* komentaaru un rindsimbolu novaakshana */
  for($x=0;$x<strlen($text);$x++){
   $current=$text[$x];
```

```
  if(($current==';') || ($current=='#')) // novaacam komentaarus
   $ignore=1;
  if (($current==chr(10)) || ($current==chr(13))){ // novaacam rindsimbolus
   $ignore=0;
  }

  if($ignore)
   continue;

/* whitespace normalizeeshana */
  $current=strtr($current,chr(10).chr(13)."\t",'   ');
  if (($prev==' ') && ($current==' '))
   continue;

  $prev=$current;
  $norm.=$current;
 }
 return strtolower($norm);
}

function vectorizetext($text){ //paarveerst PDDL atrisinaajuma tekstu par masiivu
 $ignore=2;$norm='';

/* komentaaru novaakshana */
 for($x=0;$x<strlen($text);$x++){
  $current=$text[$x];

  if(is_numeric($current)) // nocirst saakumu pirms skaitlja
   $ignore=0;

  if(($current==';') || ($current=='#'))
   $ignore=max($ignore,1);
  if (($current==chr(10)) || ($current==chr(13)))
   if($ignore==1)
    $ignore=0;


  if($ignore)
   continue;
  $norm.=$current;
 }

/* newline normalizeeshana */
  $norm=strtr(trim($norm),chr(13).chr(10).'():','##   ');

  $norm=preg_replace('/##+/','#',$norm);
  $norm=preg_replace('/\s\s+/',' ',$norm);
  $norm=str_replace(' #','#',trim($norm));
  $norm=str_replace('# ','#',$norm);

  return explode('#',strtolower($norm)); // par masiivu
}

function applymodels($modeltext){ // pielietot modeleeshanas definiicijas uz
ielaadeetato pasauli
 $preds=explode(':predicate',$modeltext);
 for($i=1;$i<sizeof($preds);$i++){ // 0. nav apraksts
  list(,$name)=explode(' ',$preds[$i]);

  list($name)=explode(':',$name);
  if($name[0]=='!'){ // ja nosaukums saakas ar "!", tad tas ir domaats noliegums
   $name=substr($name,1);
   $modelnamespace=&$this->predicates[$name]->negmodels;
  } else
   $modelnamespace=&$this->predicates[$name]->posmodels;
```

```
   if(is_null($modelnamespace))
    throw new Exception('modelling defined for unknown predicate: '.$name);

   ereg(':parameters ?\(( ?\?[a-zA-Z0-9_-]+)*\)',$preds[$i],$found);
   @list(,$prm)=explode('(',$found[0]);
   $prm=explode(' ',substr($prm,0,-1));

   if((sizeof($prm)==1) && ($prm[0]=='')) //ja nav parametru, tad nav :)
    $prm=array();

   for($ip=0;$ip<sizeof($prm);$ip++)
    $modelnamespace[$ip]=new model;

   if(strpos($preds[$i],':parameters')!==false)
    list(,$mg)=explode(':parameters',$preds[$i]);
   else
    $mg=$preds[$i];

   $mg=trim(preg_replace('/\s\s+/',' ',strtr($mg,'()','  '))); //novaacam iekavas un
dubultaas atstarpes
   $mg=explode(':',$mg);

   array_shift($mg); // 0. nav deriigs
   unset($currentprop);
   foreach ($mg as $line){

    @list($word,$param)=explode(" ",trim($line));
    $status=0;
    if (in_array($word,$GLOBALS['MODELTYPES'])){
     $status=-1;
     foreach ( $prm as $n=>$u){
      if($u==$param){
       $currentprop=&$modelnamespace[$n]->$word;
       $status=1;
       break;
      }
     }
     if($status==1)
      continue;
     if((($param=='world')||($param=='')) &&($word=='visual')){
       $modelnamespace[WORLDNUMBER]=new model;
       $currentprop=&$modelnamespace[WORLDNUMBER]->visual;
       continue;
     }
    }

    if($status==-1)
     throw new Exception('unknown parameter '.$param.' for model scope '.$word);
    if(is_null($currentprop))
     throw new Exception('expression outside model scope: '.$line);

    if($param[0]=='?') //parametrs ir dinamisks
    {
     $atb=array_search($param,$prm);
     if($atb===false)
      throw new Exception('unknown parameter '.$param.' for effect of predicate '.
$name);
     else
       $currentprop[$word]=$atb;
    } else
     $currentprop[$word]=$param; //accept parameter
   }
  }
 }
```

```
 function loadpredicates($pddl){ // ielaadeet predikaatu objektus
  ereg('\( ?:predicates ?( ?\( ?([a-z0-9_-]+)( (\?[a-z0-9_-]+))* ?\))+ ?\)',$pddl,
$found);
  $found=str_replace(') (',')(',substr($found[0],1,-1));
  $found=substr($found,strpos($found,'(')+1,-1);

  $found=explode(')(',$found);

  $predicates=array();

  foreach ($found as $pred) {
   $predexp=explode(' ',$pred);
   $pname=$predexp[0];
   $predicates[$pname]=new predicate;
   $predicates[$pname]->name=$pname;
   $predicates[$pname]->id=sizeof($predicates)-1;
   array_shift($predexp); // nonjemam nost predikaata nosaukumu
   $predicates[$pname]->paramcount=sizeof($predexp);
   $predicates[$pname]->params=$predexp;
  }

  $this->predicates=$predicates;
 }

 function loadactions($pddl){ // ielaadeet darbiibas

  $actionl=explode(':action',$pddl);
  $actions=array();

  for($i=1;$i<sizeof($actionl);$i++){ // 0. nav action
   list(,$name)=explode(' ',$actionl[$i]);

   list($name)=explode(':',$name);
   ereg(':parameters ?\(( ?\?[ a-z0-9_-]+)*\)',$actionl[$i],$found);
   list(,$prm)=explode('(',$found[0]);
   $prm=explode(' ',substr($prm,0,-1));

   $goodprm=array();
   foreach ($prm as $smp){
    if ($smp[0]=='?')
     $goodprm[]=$smp;
   }
   $prm=$goodprm;
   list(,$effect,)=explode(':effect',$actionl[$i]);
   $effect=trim(preg_replace('/\s\s+/',' ',strtr($effect,'()','  '))); // novaacam
iekavas un dubultaas atstarpes
   $effect=explode(' ',$effect);

   $sign=1;$left=0;
   $preds=array();$predcnt=-1;

   foreach ( $effect as $word){
    if ($word=='and') // ignore. default behaviour
     continue;
    if ($word=='not'){ // "nee".  uzstaadam ziimi uz preteejo.
     $sign=-$sign;
     continue;
    }
    $ok=0;
    foreach ( $this->predicates as $p){
     if($p->name==$word){
      $left=$p->paramcount;
      $preds[++$predcnt]=new command;
      $preds[$predcnt]->predicate=$p->name;
      $preds[$predcnt]->sign=$sign;
      $sign=1;
```

```
     $ok=1;
     break;
    }
   }
   if($ok)
    continue;

   if($left>0){
    foreach ( $prm as $n=>$u){
     if($u==$word){
      $preds[$predcnt]->params[]=$n;
      $ok=1;
      break;
     }
    }
    $left--;
   }
   if($ok)
    continue;

   // ELSE: ielasiits kas cits
   throw new Exception('unexpected effect of action '.$name.': '.$word);
  }

  $actions[$name]=new action;
  $actions[$name]->name=$name;
  $actions[$name]->id=sizeof($actions)-1;
  $actions[$name]->params=$prm;
  $actions[$name]->triggers=$preds;
 }
 $this->actions=$actions;
}

function loadobjects($pddl){

 $objects=array();
 ereg('\( ?:objects ([^()]+)?\)',$pddl,$found);
 $found=str_replace(' - ','#',trim($found[1]));
 $found=explode(' ',$found);

 foreach ($found as $obj){
  @list($name,$type)=explode('#',$obj);
  $objects[$name]=new pobject;
  $objects[$name]->name=$name;
  $objects[$name]->type=$type;
  $objects[$name]->id=sizeof($objects)-1;
 }

 $this->objects=$objects;
}

function loadinitstatus($pddl){ // ielaadeet saakotneejo staavokli
 ereg('\( ?:init([^:]+)\) *\(:goal',$pddl,$found);
 $found=trim(str_replace(') (',')(',trim($found[1])));
 $found=substr($found,1,-1);
 $found=explode(')(',$found);

 $initpred=array();
 $predcnt=-1;

 foreach ($found as $pred) {
  $pred=str_replace('(',' (',$pred);
  $pred=trim(str_replace(array('(',')'),'',$pred)); // novaakt iekavas
  $pred=preg_replace('/\s\s+/',' ',$pred); //novaakt dubultaas atstarpes
  $predexp=explode(' ',$pred);
```

```
   if($predexp[0]=='not'){ // noliegums.
    array_shift($predexp); //nometam "not"
    $sign=-1; // pierakstam faktu
   }
    else
    $sign=1;

   if(!isset($this->predicates[$predexp[0]]))
    throw new Exception('initial state contains unknown predicate: '.$predexp[0]);

   $initpred[++$predcnt]=new command;
   $initpred[$predcnt]->predicate=$this->predicates[$predexp[0]]->name;
   $initpred[$predcnt]->sign=$sign;

   for($i=1;$i<sizeof($predexp);$i++){ // 0. ir predikaats, nevis objekts
    if(!isset($this->objects[$predexp[$i]]))
     throw new Exception('initial state contains unknown parameter: '.$predexp[$i]);
    $initpred[$predcnt]->params[]=$this->objects[$predexp[$i]]->name;
   }
  }
  $this->initcommands=$initpred;

 }

 function loadstates($solvevector){ // ielaadeet risinaajuma gaitu
  foreach ($solvevector as $line){
   $items=explode(' ',$line);
   if(!isset($items[1]))
    continue;
   $shrt=&$this->commands[$items[0]];

   $en=sizeof($shrt);
   $shrt[$en]=new task;

   if(!isset($this->actions[$items[1]]))
    throw
 new Exception('state '.$items[0].' contains unknown action: '.$items[1]);

   $shrt[$en]->action=$this->actions[$items[1]]->name;
   for($i=2;$i<sizeof($items);$i++){ // 0. ir kaartas nr un 1. ir predikaats
    if(!isset($this->objects[$items[$i]]))
     throw new Exception('state '.$items[0].' contains unknown parameter: '.
$items[$i]);
    $shrt[$en]->params[]=$this->objects[$items[$i]]->name;
   }
  }
 }

 function loaddomain($domain,$modelling){ //ielaadeet domeenu un domeena modeleeshanu
  $normaldomain=$this->normalize($domain);
  $this->loadpredicates($normaldomain); // ielaadeet predikaatus
  $this->loadactions($normaldomain); // ielaadeet darbiibas
  $this->applymodels($this->normalize($modelling)); //pielietot modeleeshanas
definiicijas
 }

 function loadtask($problem,$solution){ //ielaadeet  probleemu un taas atrisinaajumu
  $normalproblem=$this->normalize($problem);
  $this->loadobjects($normalproblem); // ielaadeet objektus
  $this->loadinitstatus($normalproblem); // ielaadeet saakotneejo pasaules staavokli
  $this->loadstates($this->vectorizetext($solution)); //ielaadeet starpstaavokljus
 }

} // PDDL class
?>
```

PIELIKUMS NR. 2.3: “SWF ĢENERATORS: FLASH-GENERATOR.PHP”

```
<?php

require 'colornames.php';

/* dazhas konstantas veertiibas */
define('_X',0);
define('_Y',1);
define('_BOTH',2);
define('CHILDMARGIN',32); //horozontaalaa beernu apmale (ja shis ir paarskaitlis, tad
ir smukaak)
define('CHILDSPACERATIO',1.1); //vieta, kas rezerveeta vienam beernam = beerna izmeers
(1.0) + malas liidz kaiminju malaam
define('WORLD','#world'); // pasaules apziimeejums. jaasatur specsimbols ; vai #, lai
nepaarklaatos ar lietotaaja objektiem
if(!defined('WORLDNUMBER')) // shai konstantei jaanaak no PDDL parsera
 throw new Exception('What is the number of the WORLD? It is not defined.');


/* dazhas nokluseetaas veertiibas */
define('MAXX',1000); //FLASH matricas izmeers
define('MAXY',700);
define('BACKGROUNDCOLOR','lightgray'); //fona krasa
define('EPOCHFRAMES',70); //cik kadrus atveeleet vienam solim. (jo vairaak, jo labaaka
kvalitaate)
define('FLASHRATE',30); //cik kadrus atteelot vienaa sekundee (jo vairaak, jo lielaaka
slodzee buus uz skatiitaaja datora procesoru)
define('DEFOPACITY',0x50); //caurspiidiigums
define('DEFSIZE',40); //izmeers
define('DEFSHAPE','circle'); //forma
define('DEFCOLOR',0xffffff); //kraasa
define('DEFBORDER',0); //apmales kraasa
define('DEFMOVESTYLE','smooth'); // paarvietoshanaas veids

class aniobject{
 var $id;
 var $name;
 var $title;
 var $visible=false;
 var $border=DEFBORDER;
 var $color=DEFCOLOR;
 var $opacity=DEFOPACITY;
 var $size=DEFSIZE;
 var $shape=DEFSHAPE;
 var $movestyle=DEFMOVESTYLE;

 var $x;
 var $y;
 var $lastvx; // objekta peedeejaa "vizuaalaa" poziicija
 var $lastvy;
 var $lastsize;
 var $devxaschild=0; //novirze no vecaaka centra (kad objekts ir beerns)
 var $devyaschild=0;
 var $resizetofit=0;
 var $fitfactor=1;
 var $lfitfactor;


 var $sprite,$visual,$label,$visuallabel; //SWF objekti

 var $childs=array();
```

```
 var $parent; // atpakaljvirziena saite uz vecaaku


 function getrecdev($n=_BOTH){ //rekursiivi ieguut pilno novirzi
  if(isset($this->parent))
   $pardev=$this->parent->getrecdev();
  else
   $pardev=array(0,0);
  $rez=array($pardev[_X]+$this->devxaschild,$pardev[_Y]+$this->devyaschild);

  switch($n){
   case _BOTH: return $rez;
   case _X: return $rez[_X];
   case _Y: return $rez[_Y];
   default:throw new Exception('calling getrecdev() with parameter '.$n.' is not
allowed');
  }
 }

 function updatevxy(){ //atjaunojam objekta veesturiskaas veertiibas
  list($this->lastvx,$this->lastvy)=$this->getvisualxy();
  $this->lastsize=$this->size;
  $this->lfitfactor=$this->fitfactor;
 }

 function getvisualsize(){ //ieguustam objekta vizuaalo izmeeru
  return $this->size*$this->fitfactor;
 }

 function needsresizerec(){ //vai sho objektu vajag paarmeerogot (ja vecaaks tiek
paarmeerogots, tad arii vajag)
  $needs=$this->resizetofit;
  if(isset($this->parent))
   $needs|=$this->parent->needsresizerec();
  return $needs;
 }

 function getvisualxy($n=_BOTH){ // noskaidrot vizuaalo objekta atrashanaas vietu
  $rez1=$this->getxy();
  $rez2=$this->getrecdev();
  $rez=array($rez1[_X]+$rez2[_X],$rez1[_Y]+$rez2[_Y]);
  switch($n){
   case _BOTH: return $rez;
   case _X: return $rez[_X];
   case _Y: return $rez[_Y];
   default:throw new Exception('calling getvisualxy() with parameter '.$n.' is not
allowed');
  }
 }

 function getxy($n=_BOTH){ // noskaidrot logjisko objekta atrashanaas vietu
  $rez=array($this->x,$this->y);
  switch($n){
   case _BOTH: return $rez;
   case _X: return $rez[_X];
   case _Y: return $rez[_Y];
   default:throw new Exception('calling getxy() with parameter '.$n.' is not allowed');
  }
 }

 function setxy($x,$y){ // uzstaadiit logjisko objekta atrashanaas vietu
  $this->x=$x;
  $this->y=$y;
  if(!isset($this->lastvx))
   $this->updatevxy();
```

```
  foreach ($this->childs as $ch) // njemam liidzi beernus
   $ch->setxy($x,$y);
 }
} // aniobject class


class animation{
 var $movie; // filma: SWFMovie
 var $objects=array(); // aniobject tipa masiivs
 var $MAXX=MAXX;
 var $MAXY=MAXY;

 function animation(){ //kontruktors
  Ming_setScale(20.0);               //In the beginning God created the heavens and the
earth.
  ming_useswfversion(6);             //Now the earth was formless and empty, darkness
was over the surface of the deep, and the Spirit of God was hovering over the waters.
  $this->movie=new SWFMovie();       //And God said, "Let there be light," and there
was light.
  $this->movie->setRate(FLASHRATE);  //God saw that the light was good, and He
separated the light from the darkness.
  /*God called the light "day," and the darkness he called "night." And there was
evening, and there was morning—the first day. */

  $this->objects[WORLD]=new aniobject; //automaatiski izveidojam pasaules objektu
  $this->objects[WORLD]->title=WORLD;
  $this->objects[WORLD]->color=getcolor(BACKGROUNDCOLOR);  // nokluseetaa fona kraasa
 }

 function loadobjects($world,$positions){ // ielaadeejam objektus [poziicijaas]
  foreach($world->objects as $obj){
   $this->objects[$obj->name]=new aniobject;
   $this->objects[$obj->name]->title=$obj->name; // origjinaalais objekta nosaukums
tiek noglabaats sheit
   $this->objects[$obj->name]->name='obj'.($obj->id); //ieksheejie flasha nosaukumi
nedriikst satureet speciaalus simbolus, tamdeelj nosaucam objektus vienkaarshi
   $this->objects[$obj->name]->id=$obj->id;

   if(isset($positions[$obj->name])) // ja shim objektam eksistee pozicioneejums
    $this->objects[$obj->name]->setxy($positions[$obj->name]->x,$positions[$obj->name]-
>y); //uzstaadam to
   else
    $this->randomcoords($this->objects[$obj->name]); // izveelamies pseidopatvaljiigi
  }

  if(isset($positions[WORLD])){ // ja pasaulei ir noteiktas poziicijas, tad tas ir
FLASH matricas izmeers
   $this->MAXX=$positions[WORLD]->x;
   $this->MAXY=$positions[WORLD]->y;
  }
  $this->movie->setDimension($this->MAXX,$this->MAXY); //jebkuraa gadiijumaa,
jaauzstaada dimensijas
 }


 function positioninit($world){ //nostaadiit visus objektus izejas staavoklii
  foreach($world->initcommands as $command){
   $cmd=$world->predicates[$command->predicate]; // konkreetais predikaats
   if($command->sign>0) //pielietosim pozitiivo vai negatiivo modeli?
    $do=&$cmd->posmodels;
   else
    $do=&$cmd->negmodels;

   foreach ($do as $n=>$partaction)
    $this->apply($this->objects[($n==WORLDNUMBER)?WORLD:($command->params[$n])],
$partaction,$command); // pielietot izmainjas objektam
```

```
 }
}

function apply($to,$what,$cmd){ //izmainju pielietoshana.
/*
  $to - objekts, kam pietlietot izmainjas
 $what - izmainju kopums
 $cmd - komanda (lai no taas vajadziibas gadiijumaa vareetu "izvilkt" parametrus
*/
  if(isset($what->visual['shape']))
   $to->shape=$what->visual['shape'];

  if(isset($what->visual['opacity']))
   $to->opacity=$what->visual['opacity']+0;

  if(isset($what->visual['size']))
   $to->size=$what->visual['size']+0;

  if(isset($what->visual['color']))
   $to->color=getcolor($what->visual['color']);

  if(isset($what->visual['border']))
   $to->border=getcolor($what->visual['border']);

  if(isset($what->visual['visible']))
   $to->visible=((($what->visual['visible'])>0) || (($what->visual['visible']
[0])=='y') || (($what->visual['visible'][0])=='t'))?true:false;

  if(array_key_exists('resize-to-fit',$what->logic)||array_key_exists('resize-to-fit',
$what->visual))
   $to->resizetofit=1;

  if(isset($what->logic['attach']))
   $this->addchild($this->objects[$cmd->params[$what->logic['attach']]],$to);

  if(isset($what->logic['detach']))
   $this->delchild($this->objects[$cmd->params[$what->logic['detach']]],$to);

  if(isset($what->movement['style']))
   $to->movestyle=$what->movement['style'];

  if(isset($what->movement['location']))
   $to->setxy($this->objects[$cmd->params[$what->movement['location']]]->x,$this-
>objects[$cmd->params[$what->movement['location']]]->y);

}

function addchild($parent,$child){ //piesaistiit beernu
 if($parent===$child)
  throw new Exception('object {$parent->title} can not be added as a child to
itself');
 debug("adding {$child->title} to {$parent->title}");
 $parent->childs[$child->name]=&$child;
 $child->parent=&$parent;
}


function delchild($parent,$child){ //atsaistiit beernu
 debug("removing {$child->title} from {$parent->title}");
 $child->devxaschild=0;
 $child->devyaschild=0;
 unset($child->parent);
 unset($parent->childs[$child->name]);
}
```

```
function randomcoords($obj){ //uzstaadiit pseidopatvaljiigas koordinaates
 $obj->setxy(rand(0,$this->MAXX),rand(0,$this->MAXY));
}

/* FIGUURU izveides funkcijas */
function createShape($obj,$shape,$size){ // kaada figuura
 $r=false;
 switch($shape){
  case 'circle':$r=$this->createCircle($obj,$size);break;
  case 'rectangle':$r=$this->createRectangle($obj,$size);break;
  case 'square':$r=$this->createSquare($obj,$size);break;
  case 'triangle':$r=$this->createTriangle($obj,$size);break;
 default:throw new Exception('shape '.$shape.' not implemented');
 }
 return $r;
}

function createCircle($obj,$diam){ //aplis
 $obj->drawCircle(round($diam/2)); // obligaati jaanoapaljo! (matemaatiski!)
 return $obj;
}

function createSquare($obj,$diam){ // kvadraats
 $obj->movePen(0,-$diam*0.4);

$obj->drawLine($diam*0.4,0);
 $obj->drawLine(0,$diam*0.8);
 $obj->drawLine(-$diam*0.8,0);
 $obj->drawLine(0,-$diam*0.8);
 $obj->drawLine($diam*0.4,0);
 return $obj;
}

function createRectangle($obj,$diam){ //taisnstuuris
 $obj->movePen(0,-$diam*0.2);
 $obj->drawLine($diam*0.4,0);
 $obj->drawLine(0,$diam*0.4);
 $obj->drawLine(-$diam*0.8,0);
 $obj->drawLine(0,-$diam*0.4);
 $obj->drawLine($diam*0.4,0);
 return $obj;
}

function createTriangle($obj,$diam){ //trijstuuris
 $obj->movePen(0,-$diam*0.75/2);
 $obj->drawLine($diam/2,$diam*0.75);
 $obj->drawLine(-$diam, 0);
 $obj->drawLine($diam/2,-$diam*0.75);
 return $obj;
}

function createText($text,$w,$h){ //teksts (UZMANIIBU! funkcija izveido objektu PATI)
 $t = new SWFTextField(SWFTEXTFIELD_NOEDIT|SWFTEXTFIELD_NOSELECT);
 $t->setFont(new SWFFont("_serif"));
 $t->addString($text);
 $t->setHeight($h);
 $t->setBounds($w,$h*1.2);
 return $t;
}


/* KRAASU uzstaadiishanas funkcijas */
function setCcolor($obj,$col,$op){ //uzstaadiit ieksheejo kraasu prieksh ziimeeshana
PA pulkstenjraadiitaaju
 $obj->setLeftFill(($col/0x10000)%0x100,($col/0x100)%0x100,$col%0x100,$op);
}
```

```
 function setLcolor($obj,$col,$op){ // uzstaadiit liinijas kraasu
  $obj->setLine(2,($col/0x10000)%0x100,($col/0x100)%0x100,$col%0x100,$op);
 }

 function setBcolor($col){ // uzstaadiit animaacijas fona kraasu
  $this->movie->setBackground(($col/0x10000)%0x100,($col/0x100)%0x100,$col%0x100);
 }


 function generatesprites(){ //animaacijas objektu gjenereeshana
  $this->setBcolor($this->objects[WORLD]->color);

  foreach ($this->objects as $o){
   if($o->visible){ // ja objekts redzams
   $p = new SWFMorph(); //jauna paareja
   $shape=$p->getShape1(); // no
   //$shape = new SWFShape();
   $this->setCcolor($shape,$o->color,$o->opacity);
   $this->setLcolor($shape,$o->border,0xff);
   $this->createShape($shape,$o->shape,$o->lastsize*$o->lfitfactor);

   $shape = $p->getShape2(); //uz
   //$shape = new SWFShape();
   $this->setCcolor($shape,$o->color,$o->opacity);
   $this->setLcolor($shape,$o->border,0xff);
   $this->createShape($shape,$o->shape,$o->getvisualsize());

   $o->sprite=$p; // noglabaajam paareju objektaa
   $o->label=$this->createText($o->title,DEFSIZE*1.2,DEFSIZE/4); //izveidojam objekta
parakstu

   } else { //ja objekts nav redzams
    debug("hidden {$o->title}");
    unset($o->sprite);
    unset($o->label);
   }
  }
 }


 function draw(){ //uzziimeet vienu gaajienu

  foreach ($this->objects as $o){ //katram objektam
   if(is_object($o->visual)){ // ja var, novaakt veco objektu un parakstu
    $o->visual->remove();
    $o->visuallabel->remove();
   }
   if(!isset($o->sprite)) // ja nav uzgjenereets animaacijas objekts, tad njemam
naakamo
    continue;

   $o->visual = $this->movie->add($o->sprite); //pievienojam un uzstaadam vaardu
   $o->visual->setName($o->name);
   $o->visuallabel = $this->movie->add($o->label); //to pashu ar parakstu
   $o->visuallabel->setName($o->name.'label');
  }

  for($miniframe=0;$miniframe<=EPOCHFRAMES;$miniframe++){ // animeejaem paareju
   foreach ($this->objects as $o){
    if(!isset($o->sprite))
     continue;

    switch($o->movestyle){ // atkariibaa no paarvietoshanaas stila defineejam lokaalo
paarejas staavokli
     case 'before':$miniframeforobject=EPOCHFRAMES;break;
```

```
     case 'after':$miniframeforobject=($miniframe==EPOCHFRAMES)?EPOCHFRAMES:0;break;
     case 'smooth':$miniframeforobject=$miniframe;break;
     default:throw new Exception('movement style '.($o->movestyle).' not implemented');
    }

    $curx=$o->x+$o->getrecdev(_X);
    $cury=$o->y+$o->getrecdev(_Y);

    $o->visual->setRatio($miniframeforobject/EPOCHFRAMES); //paareja

    /* objekta atrashanaas vieta */
    $ex=$o->lastvx+($curx-$o->lastvx)/EPOCHFRAMES*$miniframeforobject;
    $ey=$o->lastvy+($cury-$o->lastvy)/EPOCHFRAMES*$miniframeforobject;

    /* objekta paraksta atrashanaas vieta */
    $lex=$ex-strlen($o->title)*DEFSIZE/22; //nocentreet tekstu
    $ley=$ey+($o->size/2)*($o->lfitfactor+($o->fitfactor-$o->lfitfactor)/EPOCHFRAMES*
$miniframeforobject);

    $o->visual->moveTo($ex,$ey);
    $o->visuallabel->moveTo($lex,$ley);
   }
   $this->movie->nextFrame(); // naakamais kadrs
  }
 }


 function animate($n){ //animeet soli N

  $label=$this->movie->add($this->createText("SOLIS: $n",150,20));
  $label->moveTo(10,10);   //uzstaadam solja nr

  $this->childlogic(); //izpildam beernu apstraadi
  $this->generatesprites(); //uzgjenereejam animaacijas objektus
  $this->draw();  //uzziimeejam tos
  $label->remove(); //novaacam solja nr

  foreach($this->objects as $o){ // noglabaajam jauno vizuaalo atrashanaas vietu
   $o->updatevxy();
   $o->movestyle=DEFMOVESTYLE;
  }
  $this->debugchilds(); //paarliecinaamies, ka beerniem nekas nekaish ;)
 }


 function process($world){ //apstraadaajam pasauli pa solim
  foreach($world->commands as $n=>$stage){ // katrs solis
   debug("STAGE $n");
   foreach($stage as $tst){ //katra solja  darbiibas
    $act=$world->actions[$tst->action]; // konkreetaa darbiiba

    foreach($act->triggers as $command){ // pa predikaatiem
     $transcommand=clone $command;

     foreach ($command->params as $ix=>$dun)
      $transcommand->params[$ix]=$tst->params[$dun]; // notransleejam komandas
parametrus
     $cmd=$world->predicates[$transcommand->predicate]; /// un ieguustam komandu

     if($transcommand->sign>0)
      $do=$cmd->posmodels;
     else
      $do=$cmd->negmodels;

     foreach ($do as $m=>$partaction)
       $this->apply($this->objects[($n==WORLDNUMBER)?WORLD:($transcommand-
```

```
>params[$m])],$partaction,$transcommand); // pielietot izmainjas objektam
    }
   }
  $this->animate($n); //noanimeejam doto soli
  }
 }

 function debugchilds(){ //paarliecinamies, ka kaadam beernam nav vairaaki vecaaki.
 /*
 taa vietaa, lai addchild() funcijaa automaatiski nomestu iepriekshseejo vecaaku, es
izveelos briidinaat lietotaaju,
par probleemu, kuru citaadi vinjsh nevareetu tik viegli atrast
 */
  $tmp=array();
  foreach ($this->objects as $o){
   foreach ($o->childs as $c){
    if(isset($tmp[$c->title]))
     throw new Exception("WARNING child {$c->title} was in {$tmp[$c->title]}, now found
in {$o->title}! multiple parents not supported.");
    $tmp[$c->title]=$o->title;
   }
  }
 }

 function resizerecursively($o){ //izmainiit objekta izmeerus taa, lai tajaa salien
visi beerni
  if(isset($o->childs)){
   foreach($o->childs as $c)
    $this->resizerecursively($c);
  }

  if(!$o->needsresizerec())
   return;

  $sums=CHILDMARGIN;
  foreach ($o->childs as $c){
   $sums+=$c->getvisualsize()*CHILDSPACERATIO;
  }
  if($sums==CHILDMARGIN)
   $o->fitfactor=1;
  else
   $o->fitfactor=$sums/($o->size);
 }


 function childlogic(){
  foreach($this->objects as $o) //izmainam izmeerus visiem beernu vecaakiem
   $this->resizerecursively($o);


  foreach($this->objects as $o){ //vizuaali paarvietojam beernus

   $mysize=$o->getvisualsize();
   $childcount=sizeof($o->childs);
   $nlocx=-$mysize/2+CHILDMARGIN/2;
   foreach ($o->childs as $c){
    debug("{$o->title} has child {$c->title}");
    $childsize=$c->getvisualsize();
    $c->devxaschild=$nlocx+$childsize/2*CHILDSPACERATIO;

    $c->devyaschild=0;
    if($c->parent->shape=='triangle') //trijstuurii objektus novietojam nedaudz zemaak
     $c->devyaschild+=$c->parent->getvisualsize()*0.15;

    $nlocx+=$childsize*CHILDSPACERATIO;
   }
```

```
  }
  foreach($this->objects as $o) //uzstaadam, ka vairs nekas nav jaapaarmeerogo
   $this->resizetofit=0;
 }


 function stop(){ //aptureet filmu
  $this->movie->add(new SWFAction("stop();"));
  $this->movie->nextFrame();
 }

 function create($world,$positions=NULL){ //izveidot filmu njemot veeraa pasauli [un
objektu poziicijas]
  $normalpositions=$this->parsepositions($positions);
  $this->loadobjects($world,$normalpositions); //ielaadeejam objektus
  $this->positioninit($world); //uzstaadam saakumstaavokli
  $this->animate('INIT'); // animeejam saakumstaavokli
  $this->process($world); //animeejam paareejo pasauli
  $this->stop(); //apturam filmu. ja filmu neaptur notiek briinumi :D
 }

 function display(){ //paraadam filmu uz ekraana
  header('Content-Type: application/x-shockwave-flash');
  $this->movie->output();
 }

 function parsepositions($text){ // apstraadaajam poziiciju failu formaatu
  if(is_null($text))
   return NULL;

  $rez=array();
  $norm=preg_replace('/\r\n|\r/', "\n", strtolower(trim($text)));
  $lines=explode("\n",$norm);
  foreach ($lines as $line){
   $line=trim($line);
   if($line[0]==';') //komentaars
    continue;
   @list($name,$x,$y)=explode(' ',$line);
   if(!is_numeric($x)||!is_numeric($y))
    continue;
   $rez[$name]->x=$x;
   $rez[$name]->y=$y;
  }
  return $rez; //atgriezham masiivu ar poziicijaam
 }
} // animation class
?>
```

PIELIKUMS NR. 2.4: “PALĪGFAILS: COLORNAMES.PHP”

```
<?php
function getcolor($name){

 $colornames=array(
 'aliceblue'=>0xf0f8ff,'antiquewhite'=>0xfaebd7,'aqua'=>0x00ffff,'aquamarine'=>0x7fffd4
,'azure'=>0xf0ffff,
 'beige'=>0xf5f5dc,'bisque'=>0xffe4c4,'black'=>0x000000,'blanchedalmond'=>0xffebcd,'blu
e'=>0x0000ff,
 'blueviolet'=>0x8a2be2,'brown'=>0xa52a2a,'burlywood'=>0xdeb887,'cadetblue'=>0x5fea,'ch
artreuse'=>0x7fff00,
 'chocolate'=>0xd2691e,'coral'=>0xff7f50,'cornflowerblue'=>0x6495ed,'cornsilk'=>0xfff8d
c,'crimson'=>0xdc143c,
 'cyan'=>0x0ffff,'darkblue'=>0x00008b,'darkcyan'=>0x008b8b,'darkgoldenrod'=>0xb8860b,'d
arkgray'=>0xa9a9a9,
 'darkgrey'=>0xa9a9a9,'darkgreen'=>0x006400,'darkkhaki'=>0xbdb76b,'darkmagenta'=>0x8b00
8b,'darkolivegreen'=>0x556b2f,
 'darkorange'=>0xff8c00,'darkorchid'=>0x9932cc,'darkred'=>0x8b0000,'darksalmon'=>0xe996
7a,'darkseagreen'=>0x8fbc8f,
 'darkslateblue'=>0x483d8b,'darkslategray'=>0x2f4f4f,'darkslategrey'=>0x2f4f4f,'darktur
quoise'=>0x000ced1,'darkviolet'=>0x9400d3,
 'deeppink'=>0xff1493,'deepskyblue'=>0x00bfff,'dimgray'=>0x696969,'dimgrey'=>0x696969,'
dodgerblue'=>0x1e90ff,
 'firebrick'=>0xb22222,'floralwhite'=>0xfffaf0,'forestgreen'=>0x228b22,'fuchsia'=>0xff0
0ff,'gainsboro'=>0xdcdcdc,
 'ghostwhite'=>0xf8f8ff,'gold'=>0xffd700,'goldenrod'=>0xdaa520,'gray'=>0x808080,'grey'=
>0x808080,
 'green'=>0x008000,'greenyellow'=>0xadff2f,'honeydew'=>0xf0fff0,'hotpink'=>0xff69b,'ind
ianred'=>0xcd5c5c,
 'indigo'=>0x4b0082,'ivory'=>0xfffff0,'khaki'=>0xf0e68c,'lavender'=>0xe6e6fa,'lavenderb
lush'=>0xfff0f5,
 'lawngreen'=>0xcfc00,'lemonchiffon'=>0xfffacd,'lightblue'=>0xadd8e6,'lightcoral'=>0xf0
8080,'lightcyan'=>0xe0ffff,
 'lightgoldenrodyellow'=>0xfafad2,'lightgray'=>0xd3d3d3,'lightgrey'=>0xd3d3d3,'lightgre
en'=>0x90ee90,'lightpink'=>0xffb6c1,
 'lightsalmon'=>0xffa07a,'lightseagreen'=>0x20b2aa,'lightskyblue'=>0x8cefa,'lightslateg
ray'=>0x778899,'lightslategrey'=>0x778899,
 'lightsteelblue'=>0xb0c4de,'lightyellow'=>0xffffe0,'lime'=>0x00ff00,'limegreen'=>0x3cd
32,'linen'=>0xfaf0e6,
 'magenta'=>0xff00ff,'maroon'=>0x800000,'mediumaquamarine'=>0x66cdaa,'mediumblue'=>0x00
00cd,'mediumorchid'=>0xba55d3,
 'mediumpurple'=>0x9370d8,'mediumseagreen'=>0x3cb371,'mediumslateblue'=>0x7b68ee,'mediu
mspringgreen'=>0x00fa9a,'mediumturquoise'=>0x48d1cc,
 'mediumvioletred'=>0xc71585,'midnightblue'=>0x191970,'mintcream'=>0xf5fffa,'mistyrose'
=>0xffe4e1,'moccasin'=>0xffe4b5,
 'navajowhite'=>0xffdead,'navy'=>0x000080,'oldlace'=>0xfdf5e6,'olive'=>0x808000,'olived
rab'=>0x6b8e23,
 'orange'=>0xffa500,'orangered'=>0xff4500,'orchid'=>0xda70d6,'palegoldenrod'=>0xeee8aa,
'palegreen'=>0x98fb98,
 'paleturquoise'=>0xafeeee,'palevioletred'=>0xd87093,'papayawhip'=>0xffefd5,'peachpuff'
=>0xffdab9,'peru'=>0xcd853f,
 'pink'=>0xffc0cb,'plum'=>0xdda0dd,'powderblue'=>0xb0e0e6,'purple'=>0x800080,'red'=>0xf
f0000,
 'rosybrown'=>0xbc8f8f,'royalblue'=>0x4169e1,'saddlebrown'=>0x8b4513,'salmon'=>0xfa8072
,'sandybrown'=>0xf4a460,
 'seagreen'=>0x2e8b57,'seashell'=>0xfff5ee,'sienna'=>0xa0522d,'silver'=>0xc0c0c0,'skybl
ue'=>0x87ceeb,
 'slateblue'=>0x6a5acd,'slategray'=>0x708090,'slategrey'=>0x708090,'snow'=>0xfffafa,'sp
ringgreen'=>0x00ff7f,
 'steelblue'=>0x4682b4,'tan'=>0xd2b48c,'teal'=>0x008080,'thistle'=>0xd8bfd8,'tomato'=>0
xff6347,
 'turquoise'=>0x40e0d0,'viole'=>0xee82ee,'wheat'=>0xf5deb3,'white'=>0xffffff,'whitesmok
```

```
e'=>0xf5f5f5,
 'yellow'=>0xffff00,'yellowgreen'=>0x9acd32);
 //147 kraasas
 //avots: http://www.w3schools.com/HTML/html_colornames.asp

 if(isset($colornames[$name]))
  return $colornames[$name];
 else
  if(isset($name))
   if(is_numeric($name))
    return $name;
   else
    throw new Exception('color '.($name).' is undefined');

}
?>
```

PIELIKUMS NR. 3: "PDDL PARSĒTĀJA IZVEIDOTAIS ATMIŅAS STRUKTŪRAS FRAGMENTS"

```
object(PDDL)#1 (5) {
  ["predicates"]=>
  array(9) {
    ["obj"]=>
    object(predicate)#2 (6) {
      ["id"]=>
      int(0)
      ["name"]=>
      string(3) "obj"
      ["paramcount"]=>
      int(1)
      ["params"]=>
      array(1) {
        [0]=>
        string(4) "?obj"
      }
      ["negmodels"]=>
      array(0) {
      }
      ["posmodels"]=>
      array(1) {
        [0]=>
        object(model)#33 (3) {
          ["visual"]=>
          array(3) {
            ["shape"]=>
            string(6) "circle"
            ["color"]=>
            string(3) "red"
            ["visible"]=>
            string(3) "yes"
          }
          ["movement"]=>
          array(0) {
          }
          ["logic"]=>
          array(0) {
          }
        }
      }
    }
  ...
    ["in-city"]=>
    object(predicate)#10 (6) {
      ["id"]=>
      int(8)
      ["name"]=>
      string(7) "in-city"
      ["paramcount"]=>
      int(2)
      ["params"]=>
      array(2) {
        [0]=>
        string(4) "?obj"
        [1]=>
        string(5) "?city"
      }
      ["negmodels"]=>
      array(2) {
        [0]=>
```

```
      object(model)#31 (3) {
        ["visual"]=>
        array(0) {
        }
        ["movement"]=>
        array(0) {
        }
        ["logic"]=>
        array(1) {
          ["detach"]=>
          int(1)
        }
      }
      [1]=>
      object(model)#32 (3) {
        ["visual"]=>
        array(1) {
          ["resize-to-fit"]=>
          NULL
        }
        ["movement"]=>
        array(0) {
        }
        ["logic"]=>
        array(0) {
        }
      }
    }
    ["posmodels"]=>
    array(2) {
      [0]=>
      object(model)#29 (3) {
        ["visual"]=>
        array(0) {
        }
        ["movement"]=>
        array(2) {
          ["location"]=>
          int(1)
          ["style"]=>
          string(6) "smooth"
        }
        ["logic"]=>
        array(1) {
          ["attach"]=>
          int(1)
        }
      }
      [1]=>
      object(model)#30 (3) {
        ["visual"]=>
        array(1) {
          ["resize-to-fit"]=>
          NULL
        }
        ["movement"]=>
        array(0) {
        }
        ["logic"]=>
        array(0) {
        }
      }
    }
  }
}
["actions"]=>
```

```
array(6) {
  ["load-truck"]=>
  object(action)#13 (4) {
    ["id"]=>
    int(0)
    ["name"]=>
    string(10) "load-truck"
    ["triggers"]=>
    array(2) {
      [0]=>
      object(command)#11 (3) {
        ["predicate"]=>
        string(2) "at"
        ["params"]=>
        array(2) {
          [0]=>
          int(0)
          [1]=>
          int(2)
        }
        ["sign"]=>
        int(-1)
      }
      [1]=>
      object(command)#12 (3) {
        ["predicate"]=>
        string(2) "in"
        ["params"]=>
        array(2) {
          [0]=>
          int(0)
          [1]=>
          int(1)
        }
        ["sign"]=>
        int(1)
      }
    }
    ["params"]=>
    array(3) {
      [0]=>
      string(4) "?obj"
      [1]=>
      string(6) "?truck"
      [2]=>
      string(4) "?loc"
    }
  }
  ...
}
["objects"]=>
array(20) {
  ["package1"]=>
  object(pobject)#47 (3) {
    ["name"]=>
    string(8) "package1"
    ["id"]=>
    int(0)
    ["type"]=>
    NULL
  }
 ...
}
["initcommands"]=>
array(40) {
  ...
```

```
    [39]=>
    object(command)#106 (3) {
      ["predicate"]=>
      string(2) "at"
      ["params"]=>
      array(2) {
        [0]=>
        string(8) "la-truck"
        [1]=>
        string(5) "la-po"
      }
      ["sign"]=>
      int(1)
    }
  }
  ["commands"]=>
  array(9) {
    [1]=>
    array(5) {
      ...
      [4]=>
      object(task)#111 (2) {
        ["action"]=>
        string(10) "load-truck"
        ["params"]=>
        array(3) {
          [0]=>
          string(8) "package1"
          [1]=>
          string(9) "pgh-truck"
          [2]=>
          string(6) "pgh-po"
        }
      }
    }
    ...
    [9]=>
    array(3) {
      [0]=>
      object(task)#129 (2) {
        ["action"]=>
        string(12) "unload-truck"
        ["params"]=>
        array(3) {
          [0]=>
          string(8) "package3"
          [1]=>
          string(9) "bos-truck"
          [2]=>
          string(6) "bos-po"
        }
      }
      ...
    }
  }
}
```

REĢISTRĀCIJAS LAPA

Kursa darbs “Darbību plānu vizualizācija” izstrādāts LU Fizikas un matemātikas fakultātes Datorikas nodaļā.

Autors: Kirils Solovjovs (st. apl. nr. ks05020)

Rekomendēju darbu aizstāvēšanai.

Darba vadītājs: profesors Dr.dat. Guntis Bārzdiņš

Darbs iesniegts Datorikas nodaļā 2008. gada __. jūnijā.

Sekretāre:

Darbs aizstāvēts datorzinātņu kursa pārbaudījumu komisijas sēdē

2008. g. _____ ar vērtējumu _____

Kursa pārbaudījumu komisija: