Univerza *v Ljubljani*

Univerza v Ljubljani

Fakulteta za Računalništvo in Informatiko

Tržaška 25, Ljubljana

Amorfno procesiranje

[GPL & Proto]

Poročilo pri predmetu Optične in nanotehnologije

MENTOR: Miha Mraz

AVTORJI: Jani Černe, 63040020

Jan Vesel, 63050127

Marko Živec, 63030187

Jure Janša, 63030118

Jernej Svetek, 63030054

Kazalo

1. Uvod

Kaj je amorfno procesiranje?

Kolonija celic lahko sodeluje v obliki večceličnih organizmov pod vodstvom genetskega programa, ki si ga delijo člani kolonije. Eden izmed takih primerov sodelovanja so čebele in gradnja panja. Prav tako tudi skupine ljudi lahko s skupnimi močmi zgradijo mesta. Ti primeri sprožijo temeljna vprašanja za organizacijo računalniških sistemih:

- Kako pridobiti usklajeno obnašanje iz sodelovanja velikega števila nezanesljivih delov, ki so med seboj povezani na neznane, nepravilne, in časovno različne načine?
- Katere metode uporabiti za poučevanje miriadnih („myriads“) programirljivih subjektov, da sodelujejo za doseganje posebnih ciljev?

S tema dvema vprašanjema (seveda to nista edina) se ukvarja področje amorfnega procesiranja.

Koristi amorfnega procesiranja so tako rekoč brezmejne. Te tematike bi se v naši seminarski nalogi zgolj dotaknili, vendar bi ponudili kasnejšim generacijam možnost nadaljnjega raziskovanja.

Cilji seminarske naloge.

Za cilje seminarske naloge smo si zastavili:

- poiskati ustrezne programe in literaturo, ki jo je moč uporabiti v raziskovalne namene
- preučiti najdeno programsko opremo in s pomočjo le-te rešiti v naprej določene enostavne probleme
- opisati delovanje ter napisati kratka in jedrnata navodila za prihodnje generacije, ki se bi bolj podrobno poglobile v delovanje amorfnega procesiranja

2. Izvedba

2.1. GPL (Growing Point Language)

Na začetku seminarske naloge smo začeli s preučevanjem GPL jezika. „Growing point language" je opisni programski jezik s katerim lahko kontroliramo oziroma programiramo kako se posamezni subjekti v mediju obnašajo. To je možno doseči preko naravnega pojava, ki ga imenujemo „tropizem". Medij je sestavljen iz poljubno razporejenih enakih točk (subjektov). Le-te imajo svoje lastnosti in atribute na katere vplivamo preko feromonov.

GPL pozna dva primitivna podatkovna tipa:

- material
- feromon

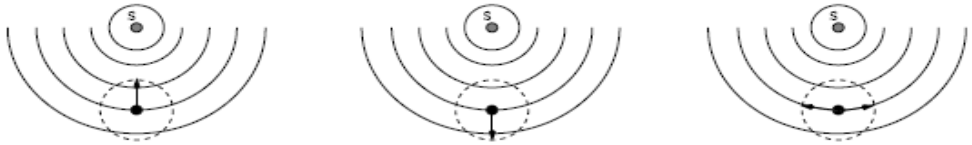
Material se uporablja kot pokazatelj, kje se je rastoča točka nahajala do tega trenutka. Ukaz color določa barvo materiala, ki je v teh točkah. Prav tako lahko rastoče točke zaznavajo materiale, zato lahko odločitve o premiku točk temeljijo na že odloženih materialih. Z njihovo pomočjo tudi na koncu preverjamo pravilnost vzorca, ki smo ga želeli izrisati.

Feromoni nam pomagajo pri usmerjanju premikanja točke. Proizvajamo jih z ukazom secrete, ki mu določimo obseg, do koder ima vpliv. Širi se v vse smeri od izvora in z oddaljenostjo enakomerno pada moč signal. Natančna oblika porazdelitve namerno ni točno določena, zaradi česar je potrebna previdnost pri pisanju programov.

Definicija rastoče točke je sestavljena iz dveh delov, in sicer atributov in navodil. Atributi opisujejo material, ki ga »odlaga«, širina proizvedene krivulje in pravilo za določanje naslednje lokacije rastoče točke. Širino določamo z ukazom »size«. Ta atribut določa širino v eno smer, torej dejansko polovico širine, ki se izriše. V vse točke, ki se nahajajo v oddaljenosti manjši od tega atributa, se odloži material, ki je določen v definiciji točke. Če ni eksplicitno določena, je vrednost tega atributa 0. Atribut »tropizem« določa feromone, na katere je rastoča točka občutljiva, kakor tudi način, kako se na njih odziva. Tega določamo s petimi različnimi ukazi:

- ortho+
- ortho-
- plagio+
- plagio-
- dia

Spodnja slika prikazuje smer gibanja rastoče točke za vsakega izmed naštetih ukazov:



ortho+ ortho- dia

plagio+ plagio-

Atribut avoid določa feromone, katerih se rastoča točka izogiba. To pomeni, da ne bo nikoli napredovala v smeri, kjer je prisoten določen feromon.

```
(define-growing-point (A-to-B-segment)
  (material A-material)
  (size 0)
  (tropism (ortho+ B-pheromone))
  (actions
   (when ((sensing? B-material)
          (terminate))
         (default
           (propagate)))))
```

V zgornjem primeru je ime rastoče točke A-to-B-segment. Ta točka odlaga material A-material, ima širino nič (to pomeni, da bo širina sledi enaka širini točke) in raste smeri povečevanja signala B-pheromone (to določa ukaz ortho+).

Drugi del, torej navodila oz. instrukcije, vsebuje zaporedje ukazov, ki se izvedejo v vsaki izmed lokacij, ki jih rastoča točka obišče. Ti ukazi določajo, ali se bo točka premaknila, oddala feromone in ali se bodo inicializirale druge rastoče točke. V našem primeru se bo točka ustavila, če bo na lokaciji, kjer se nahaja, prisoten b-material. V nasprotnem primeru se poskuša premaknit do neke točke v neposredni bližini trenutne lokacije. Čeprav v tem delu z ukazi določamo, ali se neka točka premika, način njenega gibanja določamo v deklaraciji tropizma (v našem primeru ukaz (tropism(ortho+ B-pheromone))).

Opisan primer je del programa za risanje direktne povezave med točkama A in B. Vendar pa moramo za delovanje definirati tudi točko B:

```
(define-growing-point (B-point)
  (material B-material)
  (size 0)
  (for-each-step
   (secrete 10 B-pheromone)))
```

Ime te točke je B-point, odlaga B-material in ima širino 0. Ta točka nima definiranega tropizma, kar pomeni, da je statična. V drugem delu definicije se nahaja ukaz secrete, ki povzroča, da točka proizvaja signal B-pheromone, ki seže do deset enot od njene lokacije v vse smeri. Moč signala postopoma enakomerno slabi. Zaradi tega bo vsaka točka A-to-B-segment, ki se nahaja največ deset enot vstran od točke B-point, začela "rasti" proti tej točki.

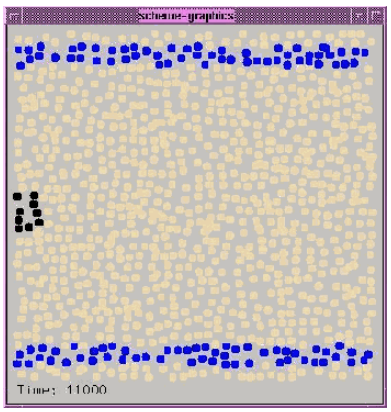
Ukazi, ki jih lahko uporabljamo pri določanju akcij rastoče točke:

- **start-gp** Ustvari novo točko na trenutni lokaciji.
- **propagate** Poišče novo pozicijo, kamor se bo točka prestavila.
- **terminate** Zaključi dinamiko točke.
- **secrete** <oddaljenost> <ime_feromon> Proizvede toliko feromona da je pri enakomerno padajoči moči signala doseg enak argumentu oddaljenost.
- **when** Pogojni stavek za vejanje izvajanja akcij.

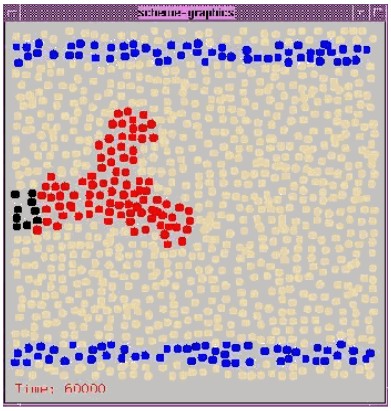
Pri amorfnem procesiranju dobro oblikovanega GPL programa lahko pričakujemo enega izmed dveh možnih rezultatov. Bodisi se bo procesiranje zaključilo z željenim ciljem ali pa se bo rastoča točka nekje zataknila in ne bo mogla dokončati svoje poti. Pogosti razlog za to je da točka nima nobenih sosednjih vrednosti, ki bi ustrazale kriteriju rasti. Seveda se tako hitro pojavi vprašanje kako naj GPL prepozna, da se je točka zataknila in kako se naj temu primeru izogne.

S to tematiko se je v svoji doktorski disertaciji „*Botanical Computing: A Developmental Approach to Generating Interconnect Topologies on an Amorphous Computer*“ ukvarjal Dr. Daniel Coore.

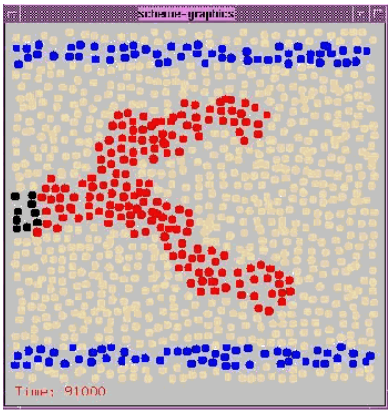
Primer: Veriga CMOS inverterjev - popolnoma lokalni nadzor natančnosti topologije



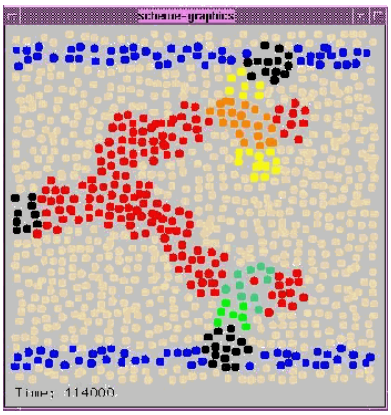
Slika 1: Dve modri skupini točk (Vdd in Vcc liniji) vsebujejo feromone proti katerima se bo rastoča točka začela širiti. Črna skupina točk prestavlja izvor ("poly") rastoče točke.



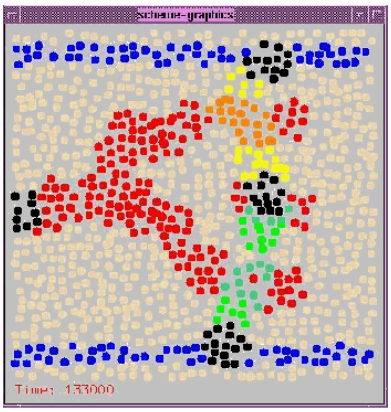
Slika 2: Rastoča točka se začne širiti glede na feromone modrih točk.



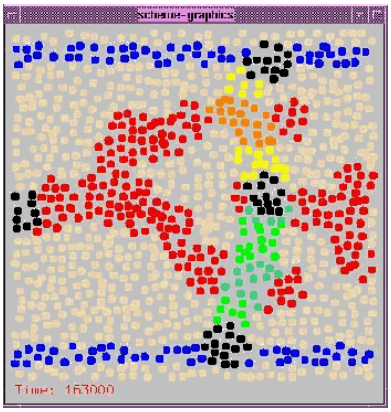
Slika 3: Rastoča točka se ustavi ko pride dovolj blizu modrim točkam.



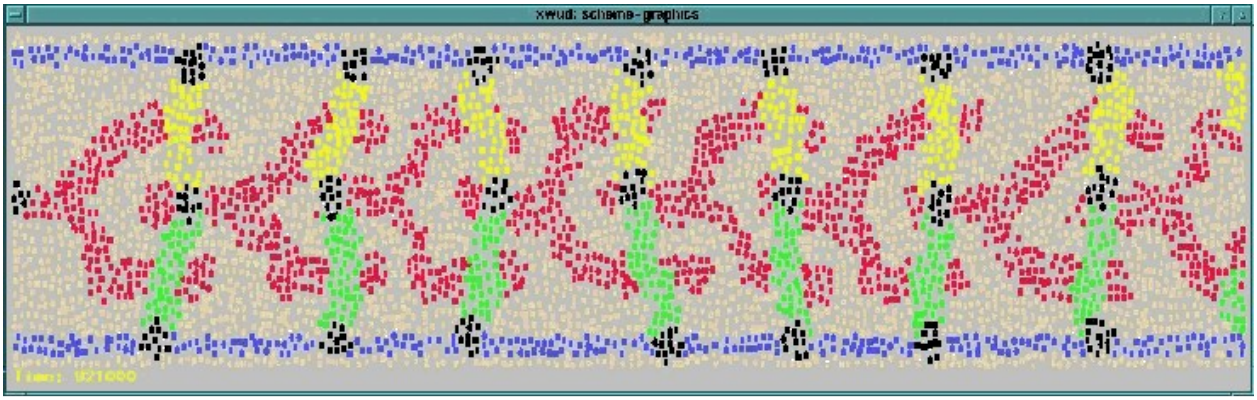
Slika 4: Ko se rastoča točka poveže z modrimi točkami najprej porodi P in N tranzistorja nato pa umre.



Slika 5: Difuzija točk se poveže in ustvari nov izvor rastoče točke.



Slika 6: Proces se ponovi.



Slika 7: Po nekaj ponavljanih dobimo verigo elementov (inverterji).

Na internetu smo poiskali doktorsko disertacijo Dr. Coore-a in njegove kontaktne podatke. Še isto minuto smo se spravili sestavljati email v katerem smo ga prosili za kakršne koli informacije glede GPL programa, ki ga je razvil. Njegov program GPL naj bi sicer delal naj bi imel nekaj hroščev v programski kodi. V nadaljevanju smo ga prosili, če nam ga lahko pošlje skupaj z dokumentacijo ter da bomo kakršna koli opažanja nepravilnega delovanja poročali nazaj. Žal nam v času izdelave seminarske naloge programa ni poslal.

Kolikor smo zasledili, se program ni posodabljal že vsaj od leta 2002, saj nam je dr. Daniel Coore v sporočilu napisal, da je od takrat program nezdružljiv z programom MIT Scheme in da zato prihaja do težav. Napake sta sicer popravljala študent s Harvarda Seth Frey in Cooreov študent Orrett Gayle, vendar dr. Coore ni mogel zagotoviti, da program deluje tako kot bi moral. Po tem sklepamo, da se je sam razvoj programa ustavil in da se samo poskuša odpraviti težave z združljivostjo.
Dela, v katerih je podrobneje razložen in analiziran GPL:

- Daniel N. Coore , Gerald J. Sussman , Harold Abelson, Botanical computing: a developmental approach to generating interconnect topologies on an amorphous computer, 1999
- Ellie D'Hondt, Exploring the Amorphous Computing Paradigm, 2000

Na internetu smo našli program MIT Proto, ki se nam je zdel zanimiv in uporaben za naše raziskovanje.

2.2. Proto

Program je poleti 2008 na MIT razvil Jacob Beal (jake.beal@mit.edu). Na naša vprašanja se je hitro odzval in nam pomagal pri reševanju težav. O samem programu in poteku njegove izdelave je na spletni strani zelo malo podatkov. Vsa dokumentacija je bila napisana 8. julija 2008, zadnja verzija programa pa je bila objavljena 29.8.2008 pod imenom "Release 0, 8-29-08".

Proto je orodje, ki omogoča pisanje kompleksnih programov prostorskih računalnikov. Prostorski računalnik je zbirka naprav porazdeljenih za zapolnitev prostora. V tem prostoru je komunikacija med napravami močno odvisna od njihovih geometrijskih razdalj (To so lahko senzorska omrežja, robotski roji, celice med morfogenezo, FPGAji, ad-hoc brezžični sistemi, biofilmi in porazdeljeni sistemi nadzora). Proto je jezik razvit za načrtovanje prostorskih računalnikov z uporabo neprekinjenega prostora abstrakcije. Namesto opisovanja obnašanja posamezne naprave, programer vidi prostor napolnjen z napravami kot amorfni medij-- področje zveznega prostora z računalniško napravo na vsaki točki-- in opisuje obnašanje področij. Ti programi so avtomatsko pretvorjeni v lokalne akcije, ki so izvršene le približno z dejanskim omrežjem naprav. Ko program "uboga" abstrakcijo, akcije zanesljivo proizvajajo aproksimacijo želenega sestavljenega obnašanja.

Inštalacija MIT Proto:

a) Linux

Izvorno kodo programa smo dobili na spletni strani, vendar nam je namestitev povzročala nemalo sivih las. Jacob Beal se je hitro odzval na naš email in nam pomagal v grobem razložil, da sistem temelji na OpenGL. Za Ubuntu OS je zato potrebno namestiti knjižnico „freeglut3-dev", ki podpira OpenGL. Glavni razlog za nedelovanje programa Proto pa je bil popravek, ki se je namestil na Ubuntu sistem in spremenil knjižnico „freeglut3-dev". Program smo brez večjih težav nato namestili na OpenSUSE distribuciji Linuxa, kjer smo ga tudi testirali.

1. Linux OS s podporo OpenGL (openSUSE)
2. Inštalacija OpenGL knjižnic (odvisno od distribucije)
3. > cd src
4. > make

b) Windows

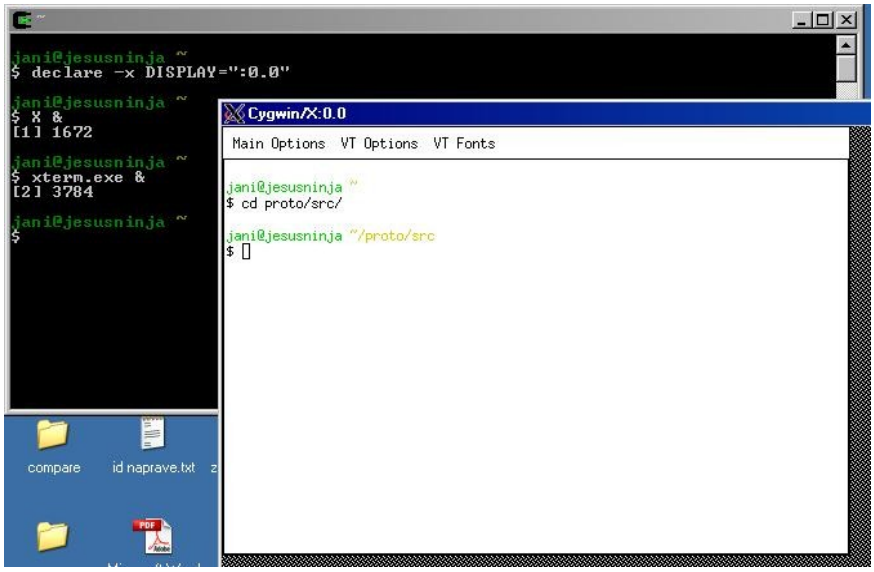
Operacijski sistem Windows sam po sebi ni razvijalsko okolje in ker Proto uporablja določene knjižnice, ga moramo zaganjati preko Cygwina. Po uspešni namestitvi Cygwina, v konzoli vpišemo npr. naš testni program(./proto ..). Ker Cygwin privzeto nima vseh potrebnih paketov, nas na to opozori. Vse kar nas čaka je njihova namestitev. Nove pakete namestimo tako, da ponovno poženemo namestitev Cygwina. Prikazalo se nam bo spet okno z obstoječimi paketi kjer dodajamo nove. Ko namestimo vse potrebno, program zaženemo po tem zaporedju:

- wget http://groups.csail.mit.edu/stpg/dist/proto-release0.tgz
- tar xf proto-release0.tgz
- declare -x DISPLAY=":0.0" ()
- X & //poženemo X server (grafični strežnik)
- xterm.exe & //poženemo konzolo ki jo bomo uporabljali

Odpre se nam konzola, ki jo uporabljamo za zagon programa.
Od tu naprej naj bi načeloma uporabljali program povsem enako kot v okolju Linux.

Problem: Ker se nisemo spuščali v delovanje programa v Windows okolju in smo zadevo poganjali le kot zanimivost, namestitve nismo dokončali. Ustavili smo se pri ukazu "make", ki požene Compile skripto.

1. Windows XP z OpenGL gonilniki
2. Cygwin s knjižnicami
3. …



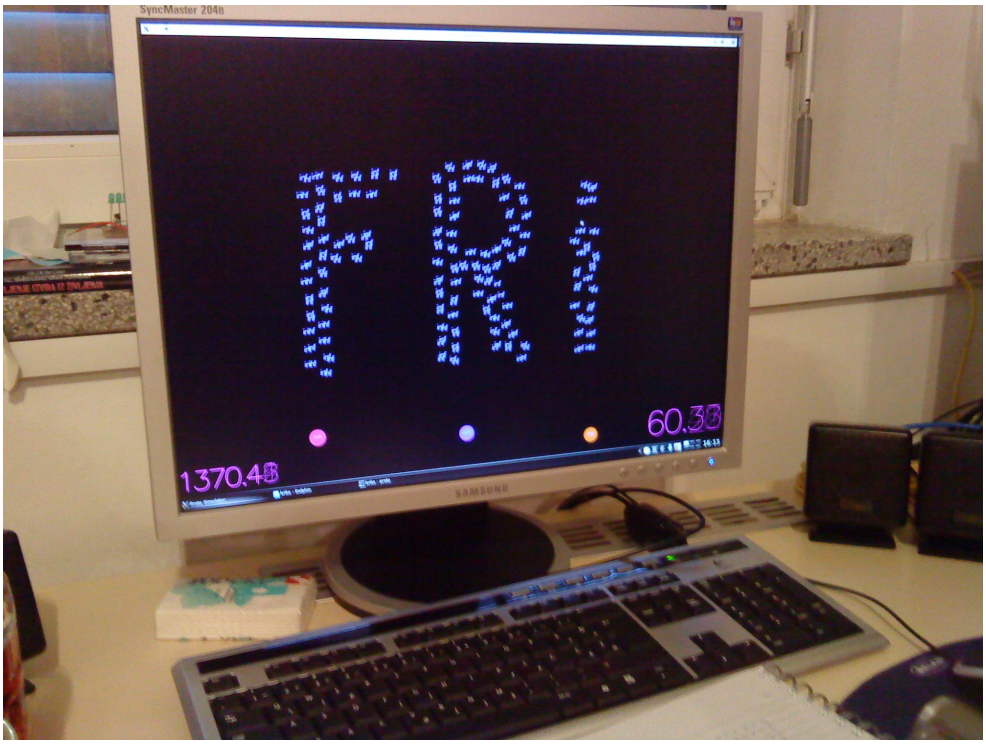
Slika 8: Namestitev programa Proto v Windows okolju

Primer: Širjenje dima

./proto -n 1000 -r 10 -l -c -T -v "(red (gradient (once (< (rnd 0 1) 0.01))))"

Na kratko opišimo ukaz : **- n 1000** pomeni uporabi 1000 naprav in **-r 10** poveži jih na razdalji 10 metrov. Naslednji parameter **-c** pokaže povezave med napravami. Ko uporabimo c parameter se nam število v desnem spodnjem kotu poveča, prikazuje število slik na sekundo. V levem spodaj pa imamo čas pretečen v sekundah, ki ga prižgemo s **-T**. **rnd (0 1)** pomeni da vse naprave izberejo naključno vrednot med 0 in 1. Primerjava **(< (rnd 0 1) 0.01)** izbere gorišča, to so tiste naprave, ki imajo naključno število manjše od 0,01 (v tem primeru dobimo približno 10 naprav). Funkcija **once** nam prižge naprave in si jih zapomni. Te parameter potem podamo funkciji **gradient**, katera poišče najbližjo razdaljo med vsako napravo do ostalih naprav, ki so v njenem dometu (razdalja -r). Vsi parametri so potem podani funkciji **red**, katera nam prikaže rdeče pike na zaslonu. Številke v modri barvi nam povejo višino teh rdečih pik. Te modre številke se prikažejo zaradi parametra **-v**. Z tipko **n** jih lahko prikažemo ali skrijemo. Rdeče pike lahko skrivamo z tipko **L**, prikažejo pa se zaradi parametra **-l**.

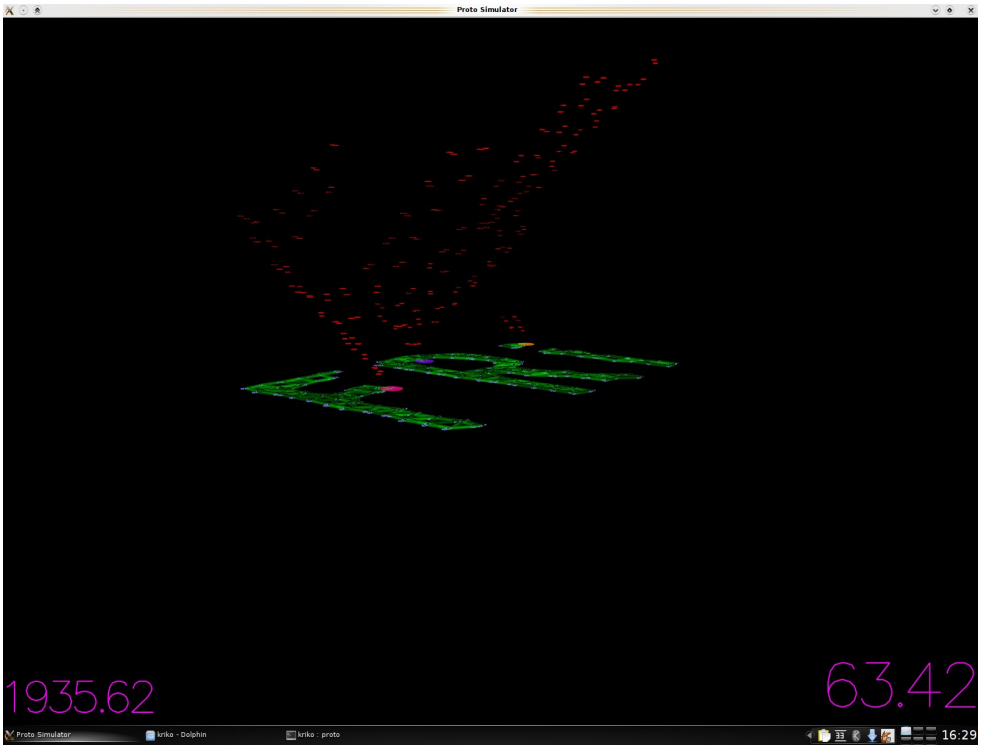
3.Rezultati



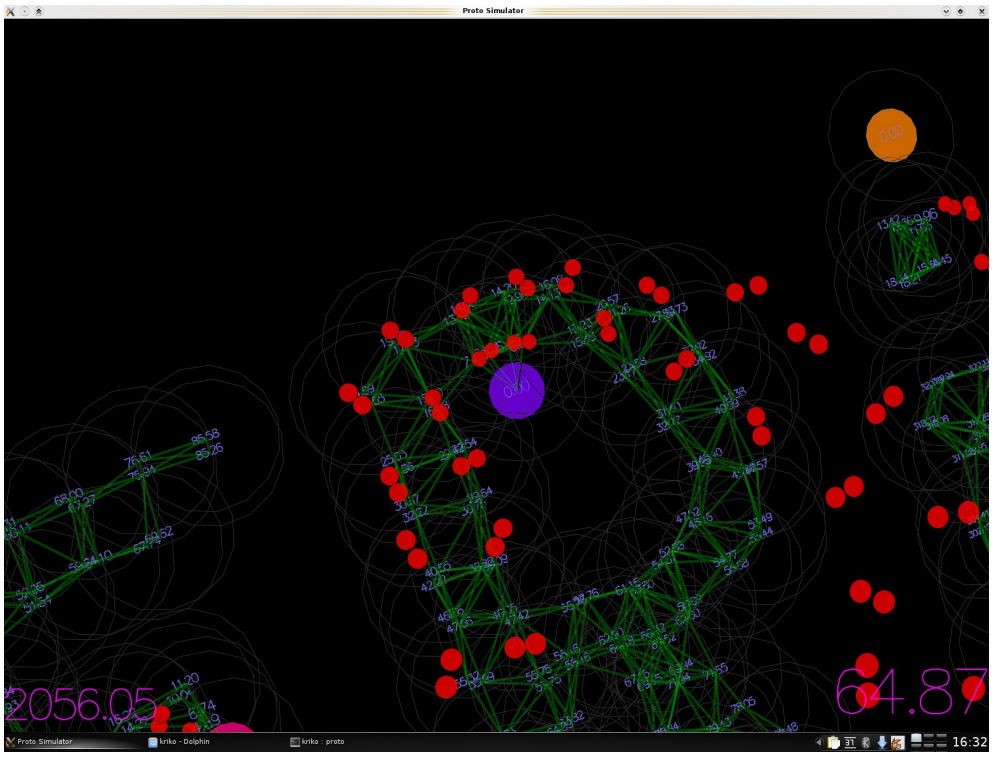
Slika 9: Primer ročne postavitve naprav (žal je orodje dokaj "zoprno" za uporabo, saj ne ponuja možnosti shranjevanja primera, ki se ga obdeluje).

Slika 10: Porazdelitev naprav s povezavami

Slika 11: Modre številke nam predstavljajo višino (oddaljenost) rdečih točk. Desna številka nam predstavlja sekunde koliko časa se program izvaja. Leva številka pa predstavlja koliko slik se prikazuje na sekundo.



Slika 12: Prikaz rezultata iz profila, kjer se vidi oddaljenost od gorišč.



Slika 13: Bolj natančen pogled naprav in njihovih povezav. Povezane so samo tiste naprave, ki so medsebojno dovolj blizu druga drugi.

4.Zaključek

Čeprav smo se pri seminarski nalogi zelo trudili, nam zaradi tehnično-človeškega faktorja žal ni uspelo doseči zastavljenih ciljev. Kljub temu smo vztrajali in z Proto programom začeli novo poglavje v našem raziskovanju. Proto je kljub pomanjkljivostim zelo uporaben program na področju prostorskega procesiranja. Upamo, da smo zbrali dovolj gradiva za raziskovanje prihodnji generaciji študentov, ki se bo ukvarjala z amorfnim procesiranjem. Žal nam zaradi pomanjkanja časa ni uspelo razviti lastnega algoritma za program Proto smo pa uspešno testirali enega izmed primerov. Čeprav Proto ni najbolje oblikovan in uporabniku prijazen program vidimo v njemu velik potencial.

5. Literatura

- MIT Amorfno procesiranje …
  http://groups.csail.mit.edu/mac/projects/amorphous/#research

- Dr. Coore (doktorsko delo)
  http://groups.csail.mit.edu/mac/projects/amorphous/papers/coore-phdthesis.ps

- Wikipedia Amorfno procesiranje
  http://en.wikipedia.org/wiki/Amorphous_computing

- Jonathan Bachrach and Jacob Beal....
  http://groups.csail.mit.edu/stpg/proto.html