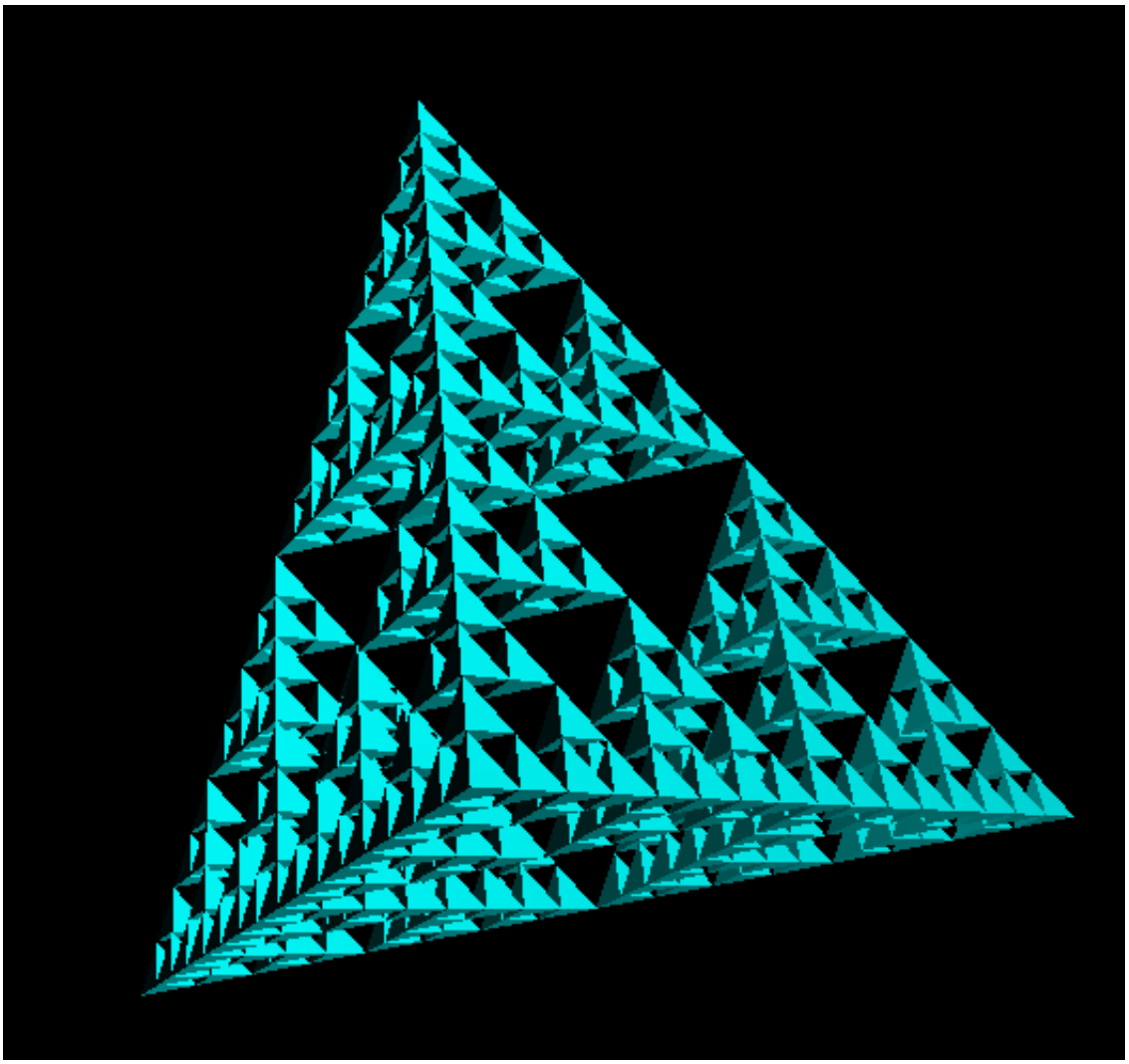
XLogo: referenca gvidlibro

Loïc Le Coq

4–a de marto, 2009



http://xlogo.tuxfamily.org

Enkonduko

Logo estas programlingvo disvolvata en la jaroj 60 de Seymour Papert. Li apogis sin sur originala teorio pri la lernado, nomata konstruismo, kies koncepton oni povas resumi per la esprimo «*lerni-per-fari*».

La lingvo Logo ebligas vaste disvolvi iujn matematikajn kaj logikajn kapablojn; ĝi estas bonega lingvo por ekstudi la programadon kaj lerni la bazojn kiel la buklojn, la provojn, la procedurojn... La uzulo povas movi objekton nomatan «testudo» sur la ekrano per komandoj tiel simplaj kiel `antaŭen`, `malantaŭen`, `dekstren` kaj aliaj. Post ĉiu movo, la testudo lasas ŝpuron malantaŭ si kaj tiel oni povas krei desegnojn. La fakto povi ordoni en lingvo preskaŭ kutima faciligas multe la lernadon. Ankaŭ pli altnivela uzado eblas; oni povas manipuli objektojn tiajn kiel listojn, vortojn aŭ eĉ dosierojn.

Logo estas lingvo interpretata, tio estas, la komandoj skribitaj de la uzulo estas tuj rulotaj de la komputilo. Oni rimarkas rekte de la rulado de la programo, la erarojn faritajn; tio favoras la lernadon.

XLogo estas do interpretilo por lingvo Logo. La adreso de la ĉefa loko de la programo estas:

`http://xlogo.tuxfamily.org/`

Vi povos deŝuti la programon kaj la dokumentaron. Galerio de kelkaj ekzemploj ebligas pli bone ekkoni la kapablojn de la programo.

XLogo subtenas nun 10 lingvojn (angla, araba, astura, esperanto, germana, hispana, franca, galega, greka kaj portugala) kaj estas verkita en Java. Tiu programlingvo havas la avantaĝon esti plurplatforma, tio estas, ke la programo XLogo ruliĝos sendepende de la mastruma sistemo instalita. Ĉu vi estas en GNU/Linukso, en Vindozo aŭ eĉ en Makintoŝo, ne estas problemo; la malgranda testudo sin oferas al vi!

XLogo estas sub permesilo GPL:

Ĝi estas do libera programo; tio garantias al la uzulo:

1. la liberecon ruli la programon, por ia ajn celo;
2. la liberecon studi la funkciadon de programo kaj adapti ĝin al siaj bezonoj; tio postulas alireblon al la fontokodojn;
3. la liberecon disdoni kopiojn;
4. la liberecon plibonigi la programon kaj publikigi la modifojn por ke la tuta komunumo profitu.

Strukturo de la gvidlibro:

Tiu gvidlibro ebligos vin ekkoni XLogon.

- La unua parto estas dediĉita al priskribo de la interfaco kaj de la diversaj menuoj.
- Poste, kelkaj ĉapitroj al vi prezentas la unuajn bazajn instrukciojn de XLogo. La malfacileco de la enĉenado de la nocioj estas gradita. Ekzercoj aplikaj estas proponataj je la fino de ĉapitro; iliaj korektigoj estas en kromĉapitro.
- Finfine, kelkajn specialajn temojn oni traktas por la altnivelaj uzuloj.
- En kromĉapitro, vi trovos la priskribon de ĉiuj primitivoj, kaj la diversajn elektaĵojn por ekruli XLogon.

Ĉi tiu gvidlibro haveblas en diversaj formatoj:

- PDF: http://downloads.tuxfamily.org/xlogo/downloads-eo/manual-eo.pdf
- HTML ZIPITA: http://downloads.tuxfamily.org/xlogo/downloads-eo/manual-html-eo.zip
- LaTeX 2$_\varepsilon$: Fontokodo de la gvidlibro: http://downloads.tuxfamily.org/xlogo/downloads-eo/manual-src-eo.zip
- JavaHelp: Per la menuo Helpo-Gvido dumrule de XLogo

Enhavo

Ĉapitro 1

Instalado de XLogo

- Unue, vi bezonas instali rulmedion JAVA en via komputilo. Iru al tiu paĝo:

  `http://java.sun.com/javase/downloads/index.jsp`

  Deŝutu la JRE (Java Runtime Environment) korespondantan al via mastruma sistemo (Vindozo, GNU/Linukso...); poste instalu ĝin.

- Due, necesas deŝuti la dosieron `xlogo.jar` estanta ĉe la adreso:

  `http://downloads.tuxfamily.org/xlogo/common/xlogo.jar`

  Se ne, pli simple, iru al la loko de XLogo, ĉe la adreso `http://xlogo.tuxfamily.org`; poste elektu la lingvon kaj la menuon deŝuti.

1.1 Agordado de XLogo

1.1.1 Medio GNU/Linukso

En Ubuntu 8.04:

1. Por instali JAVA:
   - Sistemo -> Administri -> Administrilo de pakoj Synaptic
   - Instali la pakon `sun-java6-jre`
2. Por malfermi la dosieron `xlogo.jar` per duobla klako:
   - Dekstreklaku sur `xlogo.jar`, Atributoj
   - Tabo «Malfermi per»: Elektu Sun Java Runtime
3. Asociigu la dosiertipon `lgo` al XLogo:
   - Dekstreklaku sur `xlogo.jar`, Atributoj
   - Tabo «Malferm per»:
   - Butono «Aldoni»
   - En «Uzi komandon personigitan:», tajpu:

     `java -jar vojo_al_xlogo.jar`

Rimarku: XLogo estas enhavita en la distribuo OpenSuse.

1.1.2 Medio Windows

Komence, se vi duobleklakas sur la ikono de XLogo, la programo devas starti. Se ĝi okazas, iru al la sekva paragrafo. Se ne, la kaŭzo estas ke alia programo okupiĝas pri la dosierojn de tipo «`jar`» (ofte, malkunpremaj programoj, kiel WinZip kaj aliaj).

Jen kiel asociigi la programon «`java`» al la dosieroj de tipo «`jar`». (Kelkaj vojoj povas esti malsamaj, laŭ ke vi posedas Vindozo 98, 2000, XP...)

1. Starto –> Parametroj —> Elekto de dosieroj...
2. Klaku poste sur la tabo «Dosiertipoj» (la 3ª).
3. Serĉu en la listo la elektaĵojn rilatajn al dosieroj JAR (Dosieroj JAR, Ruldosieroj JAR, Arĥivo JAR...)
4. Elektu tiun dosiertipon kaj klaku sur «Modifi...»
5. Nova fenestro aperas, tiam elektu «Modifi... »
6. Elekt tiam «Traserĉi...»
7. Necesas indiki la vojon al `javaw.exe`, ekzemple

   ```
   c:\Program Files\java\j2re1.4.1\bin\javaw.exe
   ```

8. Tion farinte, aperas en la kampo Aplikaĵo uzata por efektivigi la agon:

   ```
   c:\Program Files\java\j2re1.4.1\bin\javaw.exe
   ```

   Necesas tiam aldoni ĉe la fino:

   ```
   "c:\Program Files\java\j2re1.4.1\bin\javaw.exe" -jar "%1" %*
   ```

   (Rimarku ke necesas spaceto je ĉiu flanko de -jar)
9. Poste, nur fermu ĉiun fenestron kaj poste duobleklaku sur la ikono de XLogo.

Se tio ne ĉiam funkcias, estas dua eblo: Vi malfermu konsolon MSDOS (Starto –> Programoj -–> Komandoj MSDOS aŭ Starto –> Programoj –> Iloj –> Invito MSDOS); poste tajpu la ordonon jenan:

```
java -jar la_vojo_kie_troviĝas_la_dosiero
```

Por ekzemplo: `java -jar c:\xlogo\xlogo.jar`

Se tio enuas vin, sisteme devi tajpi tiun ordonon, tajpu tion en teksta dosiero kaj konservu ĝin ekzemple sub la nomo `xlogo.bat`. Nur restas duobleklaki sur `xlogo.bat` por startigi XLogon.

Asociigi la dosierojn de tipo `lgo` kun XLogo

Mi ne atingis agordi tion en Vista. (Sed mi ne tre serĉis... Rimarku, amatoroj! Dankon pro komuniki al mi la solvon.)

Principe, la dosieroj de tipo `.lgo` ne estas rekonataj de via komputilo; kiam vi duobleklakas ilin, dialogskatolo aperas por demandi al vi kiun aplikaĵon oni uzu por malfermi la dosieron.

- Indiku «Alia»; poste indiku la vojon al la programo `javaw.exe`

  Ĝenerale, `c:\Program Files\java\j2re1.4.1\bin\javaw.exe`

- Doni nomon por nomi la dosierojn je tipo `lgo`.
  Por ekzemplo: Dosieroj Logo

- Starto -> Parametroj -> Agordaĵoj de la dosieroj
- Tabo «Dosiertipoj »
- Serĉu en la listo la dosierojn `lgo`
- Elektu tiun dosiertipojn; poste klaku sur «Modifi»
- Nova fenestro aperas; refoje «Modifi»
- En la kampo «Aplikaĵo uzata por efektivigi la agon»,

```
"c:\Program Files\java\j2re1.4.1\bin\javaw.exe" -jar xlogo.jar "%1" %*
```

- Fermu la fenestrojn.

1.2 Ĝisdatigoj

`http://xlogo.tuxfamily.org/rss.xml`

Por ĝisdatigi XLogo, sufiĉas anstataŭigi la dosieron `xlogo.jar` per ĝia nova versio. Se vi deziras esti avertata de la apero de ĉiu nova versio, aŭ de ĉiu plibonigo, eblas aboni al la RSS-fadeno de XLogo. La adreso de la RSS-fadeno estas:

`http://xlogo.tuxfamily.org/rss.xml`

Ekzistas pluraj softvoj ebligantaj sekvi la fadenojn RSS; se vi ne konas tiun teĥnikon, la plej simpla estas uzi Mozilla Thunderbird:

- Menu' Redakti - Parametroj de la kontoj
- Buton' «Aldoni konton»
- «Novaĵoj RSS kaj blogoj»
- Nomo de la konto: «Fadenoj RSS» por ekzemplo
- Butonoj «Sekva» kaj «Fini»
- En la fenestro «Parametroj de la kontoj», elektu tiam «Fadenoj RSS» en la menu' maldekstra; poste klaku sur la butono «Administri la abonoj».
- Buton' «Aldoni»
  - URL de la fadeno: `http://xlogo.tuxfamily.org/rss.xml`
  - Aktivigu la skatolon «Afiŝi la resumon de l' artikolo anstataŭ deŝuti la retpaĝon»

Jen, per la butono «Sendi-Ricevi», vi ricevos la novaĵojn de XLogo sammaniere kiel vi ricevas viajn retpoŝtaĵoin.

1.3 Malinstalado

Por malinstali XLogo, sufiĉas forigi la dosieron `xlogo.jar` kaj la startan dosieron `.xlogo` (ĝi estas lokita en via uzula dosierujo, tio estas, `/home/via_konto` por la gnulinuksistoj aŭ `c:\windows\.xlogo` por la vindozistoj.

Ĉapitro 2

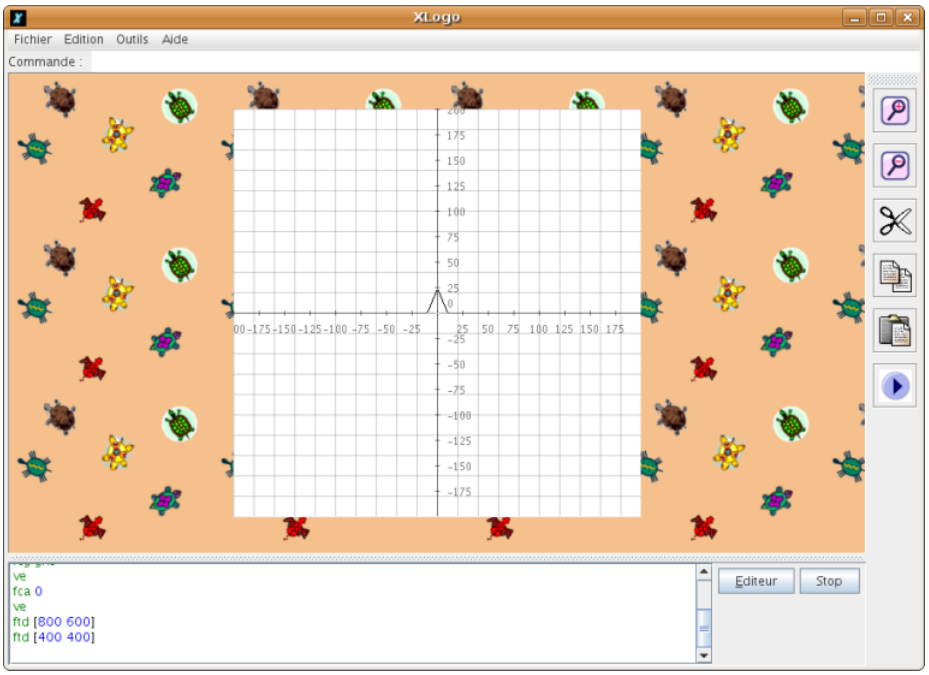
Prezentado de l' interfaco:

2.1 Je la unua ekrulado

Je la unua fojo kiam vi ekrulas Xlogon (aŭ se vi forigis la dosieron `.xlogo` — rigardu sekcion 1.3), dialogfenestro aperas por ebligi vin elekti la lingvon uzotan.

Tiu elekto ne estas definitiva, kompreneble; ĝin oni povas korekti tuj per helpo de la dialogfenestro Preferoj (rigardu sekcion 3.3).

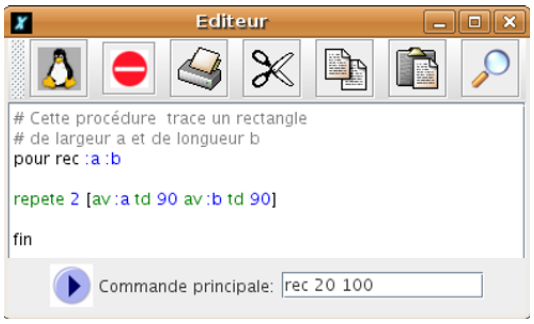
2.2 Ĉefa fenestro



- Supre, la tradiciaj menuoj **Dosiero**, **Redakti**, **Iloj** kaj **Helpo**
- Apude sube, **la komandlinio** ebliganta skribi la logo-instrukciojn.
- Meze, **la areo por desegni**.

- Dekstre de la desegnareo, **ilbreto** vin ebligas realigi diversajn agojn:
  - Zomi antaŭen/posten.
  - Diversaj redaktaj kapabloj (fortondi/kopii/alglui, tio estas, forigi/enpoŝigi/elpoŝigi).
  - La butono «Legi» ebligas ruli la ĉefan komandon difinitan en la redaktilo.
- Malsupre, **la areo «historia »** kiu memoras ĉiun lastajn komandojn tajpitajn kaj la respondojn rilatajn. Por reskribi rapide instrukcion jam tajpitan, estas du solvoj: ĉu klaki sur la malnova instrukcio en la historia, ĉu klaki plurfoje sur la sago supra ĝis la instrukcio dezirata aperos. La du sagoj supra kaj malsupra efektive ebligas moviĝi tra la tuta historio de la antaŭe tajpitaj komandoj (tre utile).
- Dekstre de l' historio, du butonoj: **HALTI** kaj **REDAKTILO** .
  - La butono HALTI haltas ĉiun ruladon kurantan.
  - La butono REDAKTILO ebligas malfermi la proceduran redaktilon.

2.3 La proceduran redaktilon



Por malfermi la redaktilon, tri ebloj:

- Tajpi `ed` en la komandlinio. La redaktilo malfermiĝos tiam kun ĉiuj proceduroj jam difinitaj. Se vi nur deziras redakti kelkajn procedurojn, tajpu tiam:
  `ed [proceduro_1 proceduro_2 ...]`
- Klaku sur la butono Redaktilo de la ĉefa fenestro.
- Uzu la klavkombinon Alt+E

Jen la diversaj butonoj kiujn vi trovos en la redaktilo:


Konservi la modifojn de la enhavo de la redaktilo, poste ĉi tiun fermi. Ja sur tiu butono oni klaku ĉiufoje ke oni volas konservi la tajpitajn procedurojn. Se vi preferas, vi povas uzi la klavkombinon ALT+Q.


Eliru la redaktilon konservante neniun modifon faritan en tiu. Oni ankaŭ povas uzi la klavkombinon ALT+C.

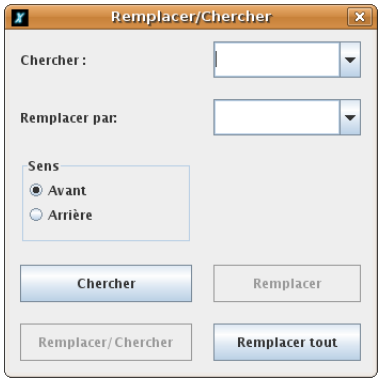

Presi la enhavon de la redaktilo.


Kopii la elektitan tekston en la poŝon.


Meti la elektitan tekston en la poŝon.

Kopii la elektitan tekston de la poŝo.

Malfermu dialogfenestron ebligantan serĉi aŭ anstataŭigi tekston en la redaktilo.



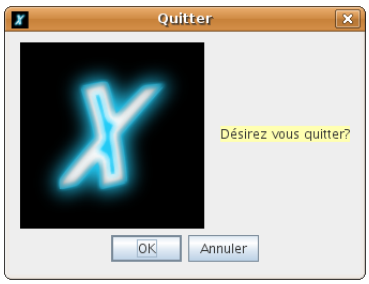
 Malsuprege de la redaktilo, teksta kampo ebligas difini ĉefan komandon. Ĉi tiu reprezentas la ĝeneralan komandon ebligantan ruli programon. Ĝi estas atingebla per la butono «legado» de la ilbreto en la ĉefa fenestro. Kiam oni konservas la enhavon de la redaktilo en dosieron kun formato `.lgo`, ankaŭ tiu komando estas konservata

GRAVE:

- Neniel utilas klaki la krucon supre dekstre por fermi la fenestron! Nur la du unuaj butonoj permesas eliri el la redaktilo.
- Por forigi unu aŭ plurajn procedurojn nedezirataj, uzu la primitivon `efp, effaceprocedure` aŭ klaku en la menubreto **Iloj - Gestionnaire de procédures**.

2.4 Quitter

Por eliri el XLogo, en la menubreto **Dosiero - For**, aŭ klaku la ferman krucon de la fenestro. Dialogfenestro por konfirmi aperas je tiu momento.

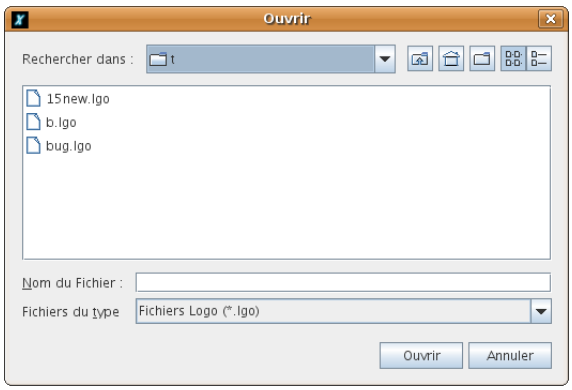


Ĉapitro 3

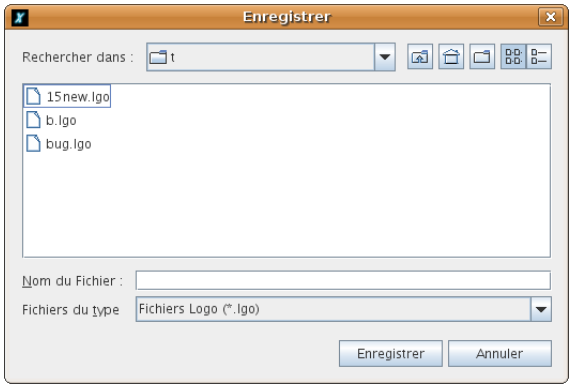
Elektebloj de la menuoj:

3.1 Menu' «Dosiero»

- **Dosiero→Nova**: detruas ĉiujn procedurojn kaj variabloj difinitajn por krei tiele novan laborspacon.
- **Dosiero→Malfermi**: malfermas logo-dosieron antaŭe konservitan.



- **Dosiero→Konservi kiel...**: konservas la kurantajn procedurojn sub elektitan nomon.



- **Dosiero→Konservi**: konservas la proceduroj en la dosieron nun uzatan.
- **Dosiero→Kapti la bildon→Konservi la bildon kiel...**: ebligas konservi la bildon kun formato jpg aŭ png. Se vi deziras elekti nur parton de l' bildo, eblas difini elektan ortangulon per gliti la muson sur la desegna areo.
- **Dosiero→Kapti la bildon→Presi la bildon**: ebligas presi la bildon. Kiel la antaŭa, vi povas elekti ĝustan areon presotan.

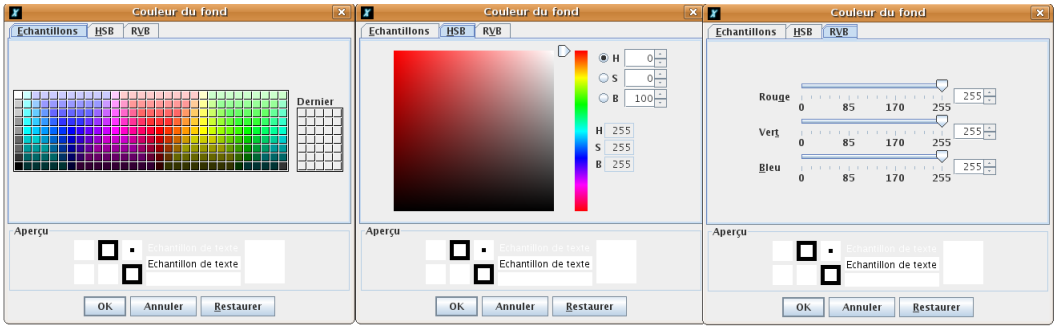
- **Dosiero→Kapti la bildon→Kopii la bildon en la poŝon**: Ebligas sendi la bildon en la poŝan sistemon. Kiel por presi kaj konservi, vi povas ankaŭ elekti nur areon de la bildo. Ĝi funkcias bone en Vindozo, ne en Linukso, ne provita en Makintoŝo.
- **Dosiero→Tekstareo→Konservi je formato RTF**: Ebligas konservi la historian areon je formato RTF (konservas la kolorojn kaj formatadon de la signoj).
- **Dosiero→Eliri**: Eliri el la programo XLOGO.

3.2 Menu' «Redakti»

- **Redakti→Kopii**: Kopias la elektitan tekston en la poŝon.
- **Redakti→Fortranĉi**: Movas la elektitan tekston en la poŝon.
- **Redakti→Alglui**: Elpoŝigas la tekston en la komandlinion.
- **Redakti→Elekti ĉion**: Elektas la tutan tekston de la komandareo.

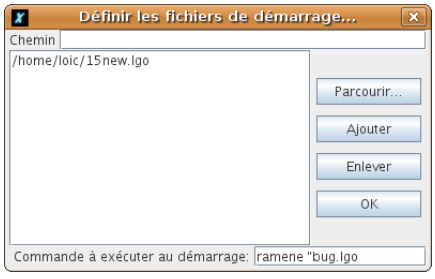
3.3 Menu' «Iloj»

- **Iloj→Elekti krajonan koloron**: Ebligas elekti la koloron per kiu skribas la testudo, helpe de koloraro.

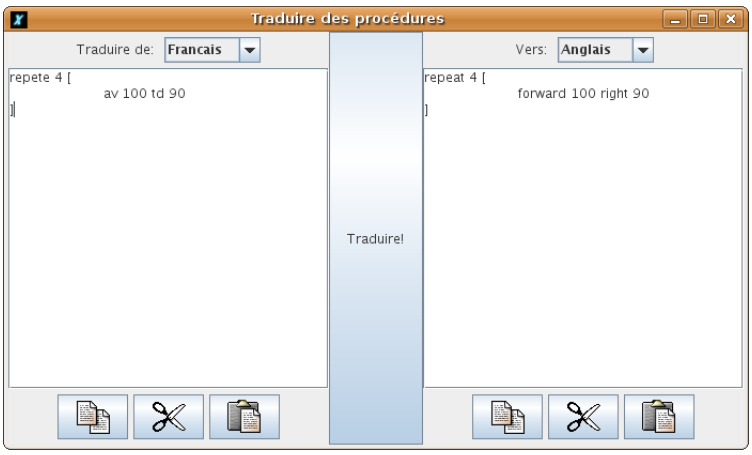


Havebla ankaŭ per la primitivo `fcc` (rigardu kromaĵon A.1.2).

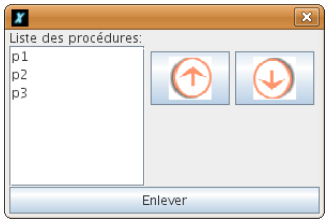
- **Iloj→Elekti fonkoloron**: Same pri la ekranfono. Havebla per la primitivo `fcfg` (rigardu kromaĵon A.1.2).
- **Iloj→Difini startodosierojn**: ebligas difini vojojn al dosieroj kun formato *.lgo nomataj «startecaj». Ĉiuj proceduroj en tiuj dosieroj estiĝos «kvazaŭ-primitivoj» de la lingvo XLogo. Ili ne estas redakteblaj nek modifeblaj de l' uzulo. Vi povas ankaŭ difini personigitajn primitivojn. Vi povas ankaŭ doni al ĝi komandon (en logo) rulatan dum starto de XLogo. Vi povas ankaŭ ruli programon koncipitan de vi, ekde la malfermo de XLogo.



- **Iloj→Traduki procedurojn**: Malfermas dialogfenestron ebligantan traduki komandojn XLogo en la lingvon deziratan. (Tre utila speciale kiam oni prenas en interreto Logo-fontokodon en la angla, por ilin esperantigi.)



- **Iloj→Procedura administrilo**: Malfermas dialogfenestron kiu ebligas forigi procedurojn. Ĝi permesas ankaŭ ŝanĝi la aperordon de la proceduroj en la redaktilo.



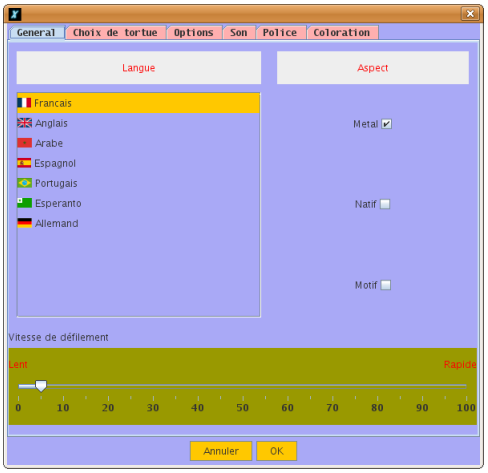
- **Iloj→Preferoj**: Malfermu dialogfenestron en kiu vi povas agordi plurajn aferojn:

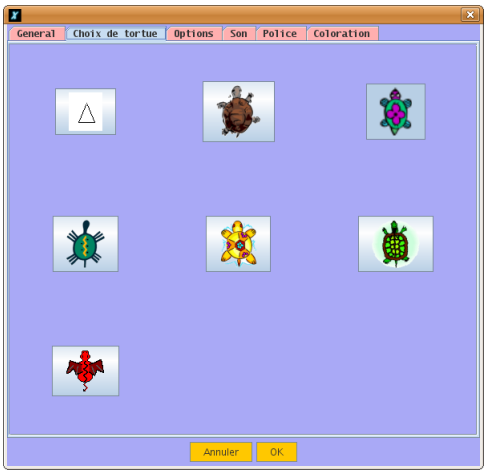
  – **Ĝenerala langeto:**

    → **Lingvo**: Ĝi ebligas elekti inter la franca, la angla, la hispana, la portugala, l' araba, la germana kaj esperanto. Atentu, ĉar la primitivoj ŝanĝiĝas de lingvo al alia.

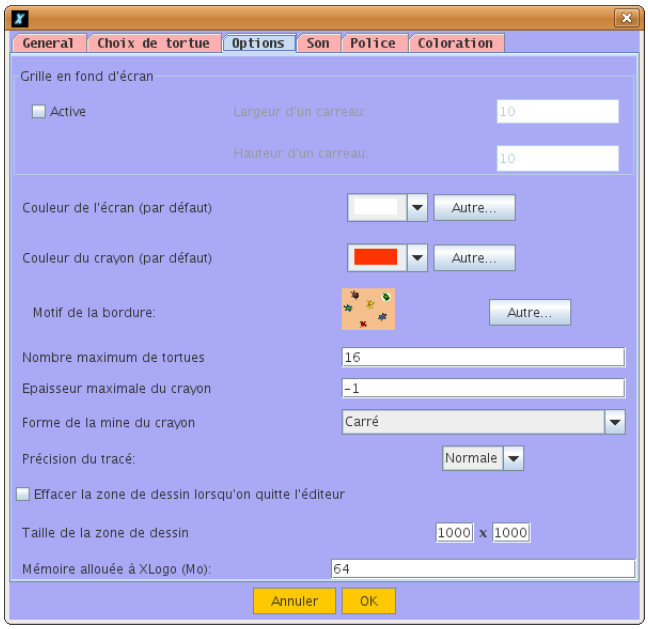
    → **Aspekto:** Ebligas difini la «look»on de la fenestro XLogo. Ĉu stilo nativa, ĉu stilo Java (metala), ĉu stilo Motif.

    → **Elekti la skribrapidon:** Se vi deziras vidi ĉiun translokiĝon de la testudo, vi povas malrapidigi ĝin helpe de la glitbutono celita por tio.





– **Langeto Elekti testudon**: Vi povas elekti vian preferatan testudon.



– **Langeto Elektoj**: Oni povas agordi plurajn aferojn.

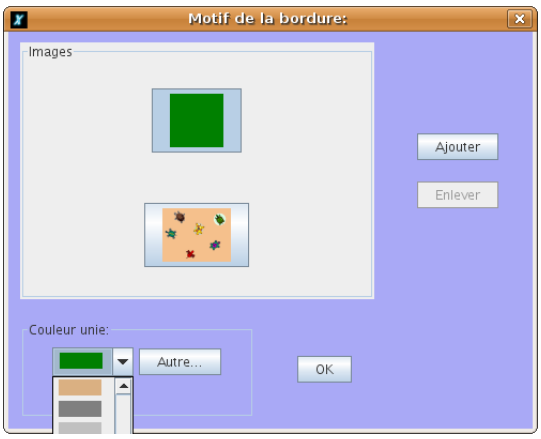
→ **Dratreto:** Vi povas elekti ĉu desegni dratreton sur l' ekranfono. Vi povas elekti la larĝon kaj la alton de kvadrato de la dratreto, kaj ankaŭ ĝian koloron.

→ **Aksoj:** Vi povas elekti ĉu desegni la vertikalan akson kajaŭ la horizontalan akson sur l' ekranfono. Vi povas difini la distancon inter du gradumoj kaj ankaŭ la koloron de ĉiu akso.

→ **Koloro de ekranfono**: Eblo difini aprioran koloron de ekranfono.

→ **Koloro de krajono**: Eblo difini aprioran koloron krajonan.

→ **Bordera motivo**: Eblo difini preciza motivon por la marĝeno enkadriganta la desegnejon (ĉu per bildo, ĉu per nura koloro)



→ **Diko de krajono**: Oni povas indiki liman amplekson por la dikeco de krajono. Si oni ne volas uzi tiun limigon, metu la nombron −1 en la areon tekstan rilatan.

→ **Formo de krajono**: Tuj, oni povas elekti la formon de la testuda krajono. Por ĝin rimarki, elektu krajondikon pli grandan ol 1.

→ **Maksimuma nombro de testudoj**: Oni povas ŝanĝi la maksimuman nombron de testudoj en plurtestuda moduso (apriore 16).

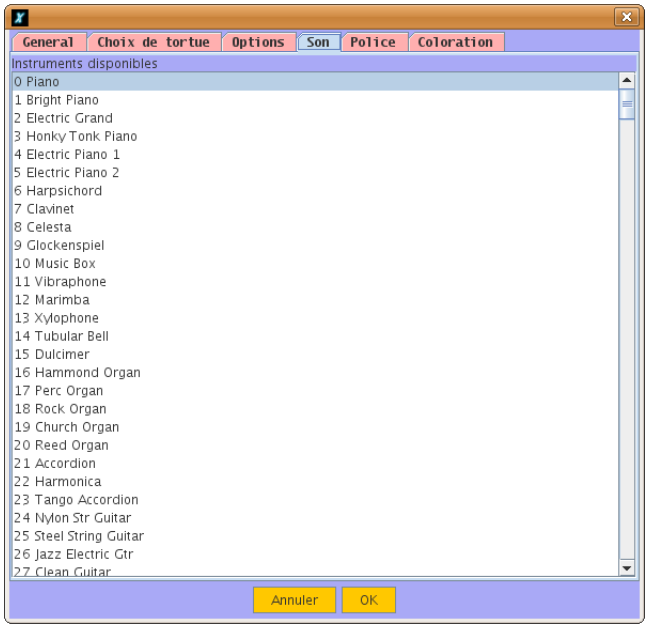
→ **Desegna precizeco**: Vi povas elekti la desegnan kvaliton. Je alta kvalito, vi ne havos la efikon de linipikselado. Male, atentu ke, ju pli da kvalito, des malpli da rulrapideco.

→ **Purigado je eliro el redaktilo**: Oni povas elekti ĉu aŭtomate purigi la desegnejon kiam oni eliras el la redaktilo.

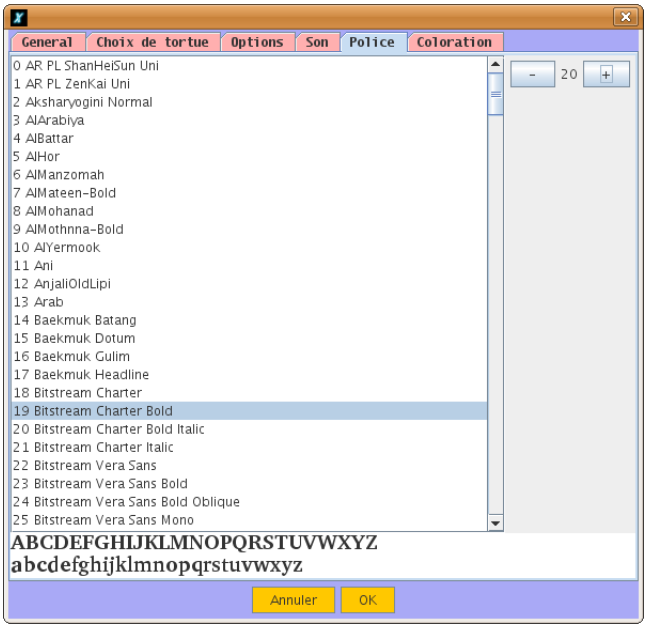
→ **Amplekso de la desegnejo**: Vi povas elekti propran amplekson por la desegnejo. Apriore XLogo ruliĝas kun areo de 1000 pikseloj mul 1000 pikseloj. Atentu: Kiam vi pligrandigas la bildon, eble necesas pligrandigi la kvanton de memoro atribuita al XLogo. Erarmesaĝo vin avertos pri tio.

→ **Memoro atribuita al XLogo**: Vi povas tial ankaŭ ŝanĝi la valoron rilatan al la memora spaco atribuita al XLogo. Apriore, tiu valoro estas 64 MiB. Eble vi devos pligrandigi ĝin se vi deziros labori sur desegnejo pli granda. Kiam oni modifas tiun parametron, la ŝanĝo nur efikas post la restarto de XLogo. Atentu, ne pligrandigu multe sen kaŭzo tiun valoron; ĝi povas multe malrapidigi vian sistemon.

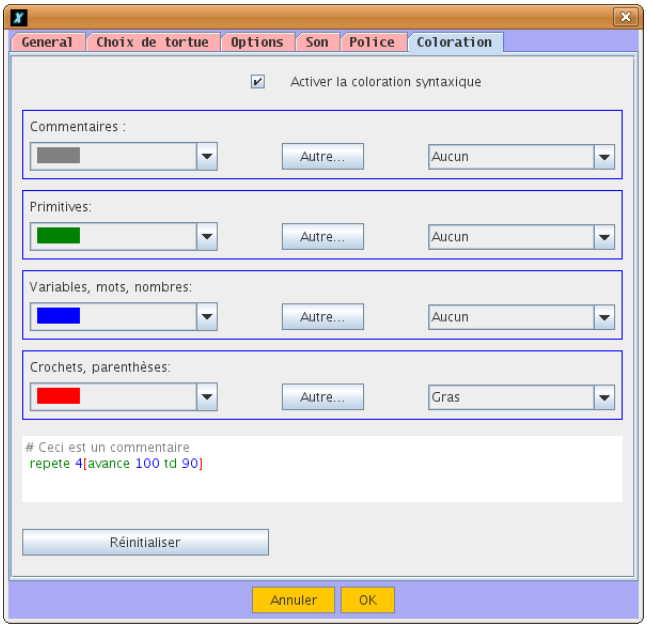
→ **Nombro de pordo TCP:** Ebligas elekti iun valoron por la pordo uzata por la retkomunikadoj. Rigardu p. 121.



– **Langeto Sono**: vi trovos la liston de instrumentoj kiujn povas ŝajnigi via sonkarto per l' interfaco MIDI. Vi povas elekti instrumenton klakante ĝian nomon. (Vi povas elekti instrumenton ankaŭ per la primitivo `instrumenton_provizu numero`.) Se la listo de instrumentoj ne aperas, rigardu la Oftajn Demandojn fine de l' gvidlibro pri tiu afero.



– **Langeto Tiparo**: En la kvina langeto, vi povas elekti la tiparon de la grafika interfaco kaj ĝian amplekson. Atentu ke tio ne influas la tiparon uzatan de la primitivoj `skribu` et `etikedu`.

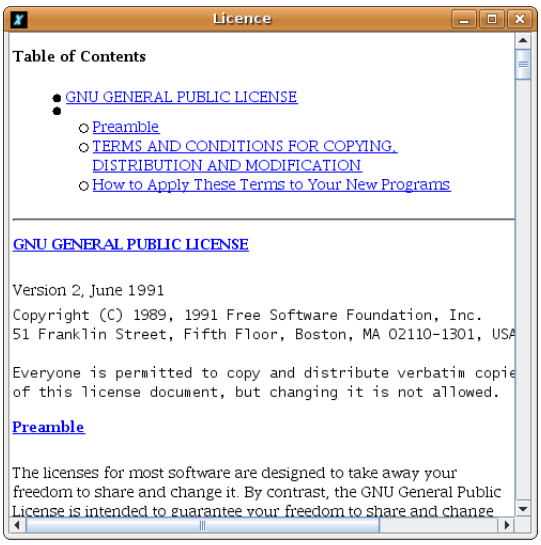


– **Langeto Sintaksa kolorigo**: Eblo (mal)aktivigi la sintaksan kolorigon kaj difini proprajn kolorojn.

3.4 Menu' «Helpo»

- **Menu' Helpo→Reta lernolibro**: Aliras la referencan lernolibron de XLogo, nur se ekzistas interreta konekto.

- **Menu' Helpo→Permesilo**: Aliras la permesilon GPL sub kiu oni distribuas la programon.

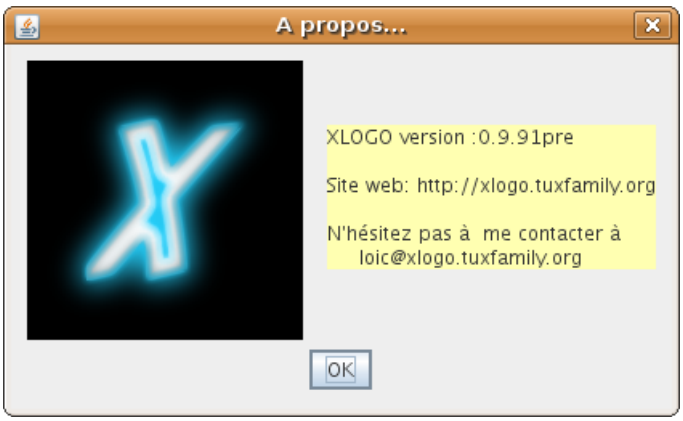


- **Menu' Helpo→Esperanta traduko**: Aliras esperantigitan permesilon GPL. Tiu tradukaĵo havas nenian valoron oficialan, nur la angla originalo.

- **Menu' Helpo→Traduki XLogo-n**: Malfermas dialogfenestron ebligantan konsulti / modifi / kompletigi la aron de tradukaĵoj de XLogo (mesaĝoj kaj primitivoj).

Eblas ankaŭ krei tradukaĵojn por nova lingvo. Je ĉiu okazo oni sendu la dosieron generitan al `loic@xlogo.tuxfamily.org`.

- **Menu' Helpo→Rilate...**: Klasika ... kaj `http://xlogo.tuxfamily.org` por viaj ĝisdatigoj!! `o:)`


A propos...
XLOGO version :0.9.91pre
Site web: http://xlogo.tuxfamily.org
N'hésitez pas à me contacter à
loic@xlogo.tuxfamily.org
OK

Ĉapitro 4

Konvencioj adoptitaj en XLOGO

Jen prezentado de iuj aferoj pri la programlingvo LOGO mem kaj de aliaj pri XLOGO specife.

4.1 Komandoj kaj interpretado

Programlingvo LOGO konsistas el internaj komandoj: tiajn komandojn oni nomas **primitivoj**. Ĉiu primitivo atendas iun nombron de parametroj nomataj **argumentoj**. Por ekzemplo, la primitivo `ev` kiu ebligas viŝi l' ekranon prenas nul argumenton, dum la primitivo `sum` atendas du argumentojn: `sum 2 3` skribos 5 redone.

Estas tri specoj de argumentoj en LOGO:

- **La nombroj:** Iuj primitivoj atendas nombrojn kiel argumenton. Ekzemple `antaŭen 100`
- **La vortoj:** Ĉiuj vortoj komenciĝas per ". Ekzemplo de primitivo kapabla labori pri vortoj estas la primitivo `skribu`.

  ```
  skribu "saluton
  ```

  Tiu komando kaŭzas l' aperon de la vorto `saluton` en la teksta areo.

  Rimarku ke se vi forgesas la ", l' interpretilo respondos per erarmesaĝo. Efektive, `skribu` atendas argumenton, sed por l' interpretilo `saluton` signifas nenion, ĉar ĝi estas nek nombro nek vorto nek listo nek jam difinita proceduro.
- **La listoj:** Ilin oni difinas inter rektaj krampoj.

Rimarku: La nombroj estas traktataj jen kiel nombraj valoroj, jen kiel vortoj. Ekzemple: `skribu unuan 12` redonas 1. Iuj primitivoj akceptas ĝeneralan formon, tio estas, ili povas ricevi nedifinitan nombron de argumentoj. Jen la listo de tiuj primitivoj:

skribu	sumon	produton	aŭ
kaj	liston	frazon	vorton

Por sciigi l' interpretilon ke oni uzos ilin sub ilia ĝeneralan formon, oni tajpu la komandon inter rondaj krampoj; jen kelkaj ekzemploj:

```
skribu (sumon 1 2 3 4 5)
15

(list [a b] 1 [c d])
Kiel uzi [[a b] 1 [c d]]?

se (kaj 1=1 2=2 8=5+3) [an 100 dn 90]
```

4.2 Proceduroj

Krom tiuj primitivoj, vi povas difini viajn proprajn komandojn. Oni nomas ilin *proceduroj*. La procedurojn oni komencas difini per helpo de la vorto `por` kaj oni finas difini per la vorto `fino`. Oni uzas la proceduran redaktilon internan je XLOGO por tajpi ilin. Jen malgrandan ekzemplon:

```
por kvadrato
ripetu 4 [antaŭen 100 dekstren 90]
fino
```

Ankaŭ tiaj proceduroj rajtas akcepti argumentojn. Por tio, oni uzas variablojn. Variablo estas vorto al kiu oni povas rilatigi valoron. Jen tre simpla ekzemplo:

```
por tuto :a :b
skribu sum :a :b
fino

tuto 2 3 -----> 5
```

4.3 La speciala signo «\»

La signo «\» (maloblikva streko) ebligas krei vortojn enhavantajn spacojn aŭ enhavantajn linisalton. «\n» enmetas linisalton kaj «\␣» enmetas spacon en vorton. Ekzemple:

```
skribu "xlogo\ xlogo
xlogo xlogo
skribu "xlogo\nxlogo
xlogo
xlogo
```

Tial por skribi signon «\» oni tajpu ĝin duoble: «\\».

Same, la signoj «( ) [ ] # » estas limiloj de la lingvo Logo kiuj ne povas esti uzataj en vortoj. Oni povos enmeti ilin per aldoni signon «\» antaŭe.

Ĉiu signo «\» sola estos ignorita. Tio tre gravas specife por administri dosierojn.

Por establi la aktualan dosierujon je `C:\Miaj dokumentoj`, necesos tajpi:

```
dosierujon_provizu "c:\\Miaj\ dokumentoj
```

Rimarku l' uzadon de «\␣» por indiki la spacon inter «Miaj» kaj «dokumentoj». Se aliflanke, vi ne metas la duoblan maloblikvan strekon, la vojo difinitas estos tiam `c:Miaj dokumentoj` kaj la interpretilo skribos erarmesaĝon.

4.4 Reguloj pri uskleco

XLogo ne diferencas uskle pri la nomoj de proceduroj kaj primitivoj. Tial, pri la proceduro `kvadrato` difinita antaŭe, ĉu vi tajpus `KVADRATO`, ĉu `KvaDRato`, l' interpretilo de komandoj ĝuste interpretos kaj rulos `kvadrato`. Male, XLogo diferencas en listoj kaj vortoj:

```
skribu "Saluton ----> "Saluton (oni konservas la majusklan S)
```

4.5 Operatoroj kaj sintakso

Estas du manieroj skribi kelkajn komandojn. Ekzemple, por adicii 4 kaj 7, estas du ebloj:

- jen oni uzas la primitivon `sumon` kiu atendas du argumentojn: oni skribas `sumon 4 7`
- jen oni uzas l' operatoron +: oni skribas `4+7`.

La du havas saman efikon. Jen la listo de rilatoj inter operatoroj kaj primitivoj:

`sumon`	`subtrahon`	`produton`	`dividon`
+	-	*	/
`aŭ`	`kaj`	`egala?`	
|	&	=	

Ekzistas ankaŭ du operatoroj de numeraj provoj rilataj al neniu primitivo:

- Operatoro «malpli granda aŭ egala» `<=`
- Operatoro «pli granda aŭ egala» `>=`

Atentu: Neniu spaco inter la signoj `>` kaj `=`!

Rimarku: La du operatoroj | et & estas specifaj operatoroj de XLOGO. Ili ne ekzistas en la tradiciaj versioj de LOGO. Jen kelkaj ekzemploj de uzo:

```
s 3+4=7-1 ----> vera
s 3=4 | 7>=49/7 ----> vera
s 3=4 & 7=49/7 ----> malvera
```

Ĉapitro 5

Malkovri la bazajn primitivojn

Nivelo: komencanto

Por movi la testudon sur la ekrano, oni uzas antaŭdifinitajn komandojn nomatajn «primitivoj». En tiu ĉi ĉapitro, ni malkovros kelkajn bazajn primitivojn ebligantajn gvidi la testudon.

5.1 Novaj primitivoj uzotaj:

1. `an nombro` `an 50`
   Antaŭenigi la testudon je la nombro de testudaj paŝoj indikitaj.

2. `man nombro` `man 100`
   Malantaŭenigi la testudon je la nombro de testudaj paŝoj indikitaj.

3. `dn nombre` `dn 90`
   La testudon turni dekstren je l' angulo indikita.

4. `mdn nombre` `mdn 45`
   La testudon turni maldekstren je l' angulo indikita.

5. `ev` `ev`
   Viŝi l' ekranon kaj remeti la testudon centren de l' ekrano.

6. `tdm` `tdm`
   La testudo esti videbla sur l' ekrano.

7. `tdk` `tdk`
   Testudon kaŝi. Eble ebligas grafiki pli rapide.

8. `l` `l`
   Levi la krajonon. La testudo ne lasas ŝpuron post si kiam ĝi moviĝas.

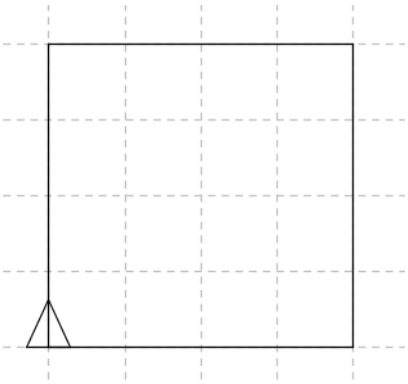
9. `ml` `ml`
   Mallevi la krajonon. La testudo skribas kiam ĝi moviĝas.

10. `ripetu nombro listo` `ripetu 5 [an 50 dn 45]`
    Ripeti l' instrukciojn enhavatajn en la listo je la nombro de fojoj indikita.

5.2 Desegni regulan plurlateron

Ĉi tie, ni lernos desegni kvadraton, egallateran trilateron, regulan kvinlateron, ktp.

5.2.1 La kvadrato



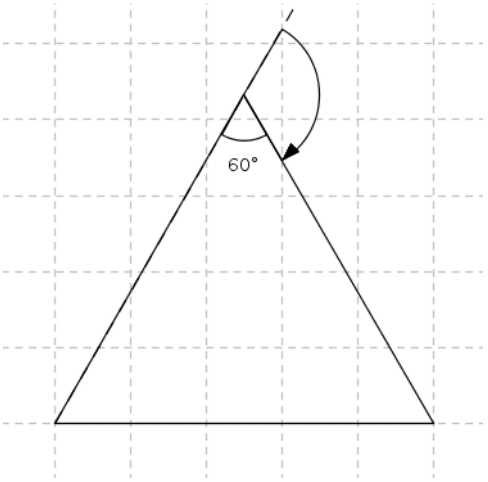
Unu reta ĉelo reprezentas 50 testudajn paŝojn. Por desegni la apudan kvadraton, oni do tajpu:

```
an 200 dn 90 an 200 dn 90 an 200 dn 90 an 200 dn 90
```

Oni rimarku ke oni ripetas 4 fojojn la saman instrukcion, do jen sintakso pli rapida:

```
ripetu 4 [an 200 dn 90]
```

5.2.2 La egallatera trilatero



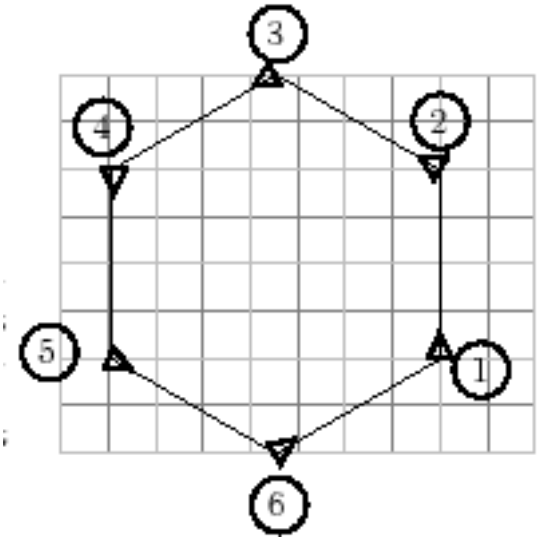
Ĉi tie, ĉelo reprezentas 30 testudpaŝojn. Ni vidos kiel desegni tiun egallateran trilateron kun lateroj de 150 testudpaŝoj. La instrukcio similos ion tian:

```
ripetu 3 [an 150 dn ...]
```

Restas kalkuli la bonan angulon. En egallatera trilatero, l' anguloj havas ĉiuj 60 gradojn. Ĉar la testudo turniĝas ekster la trilatero, l' angulo validas $180 - 60 = 120$ gradojn. L' instrukcio estas do:

```
ripetu 3 [an 150 dn 120]
```

5.2.3 La seslatero



Ĉi tie, ĉelo reprezentas 20 testudpaŝojn.

```
ripetu 6 [an 80 dn ...]
```

Oni rimarku ke dum ĝia moviĝo, la testudo efektivigas kompletan turniĝon (ĝi ekiras adresita al supro, fine ĝi revenas en tiun saman pozicion). Tiu turniĝo je 360 gradoj efektiviĝos post 6 etaĝoj. Tial, je ĉiu fojo, ĝi turniĝas je $360/6 = 60°$.

L' instrukcio estu do: `ripetu 6 [an 80 dn 60]`

5.2.4 Desegni regulan plurlateron ĝenerale

Efektive, ripetante la malgrandan pensadon antaŭan, oni rimarku ke, por desegni plurlateron je n lateroj, l' angulon oni kalkulu per divido de 360 per n. Por ekzemplo:

- Por grafiki regulan kvinlateron je latero 100:

  ```
  ripetu 5 [an 100 dn 72]    (360:5=72)
  ```

- Por grafiki regulan naŭlateron je latero 20:

  ```
  ripetu 9 [an 20 dn 40]    (360:9=40)
  ```

- Por grafiki regulan ee... 360-lateron je latero 2 (ĝi ja similas cirklon!):

  ```
  ripetu 360 [an 2 dn 1]
  ```

- Por grafiki seplateron je latero 120:

  ```
  ripetu 7 [an 120 dn 360/7]
  ```

5.3 Registri proceduron

Por ne devi retajpi ĉiufoje l' instrukciojn por desegni kvadraton, trilateron... oni povas difini personajn komandojn nomatajn «proceduroj». Proceduro komenciĝas per la ĉefvorto `por` kaj finiĝas per la ĉefvorto `fino`. Oni malfermu la redaktilon kaj tajpu ekzemple

```
por kvadrato
ripetu 4 [an 100 dn 90]
fino
```

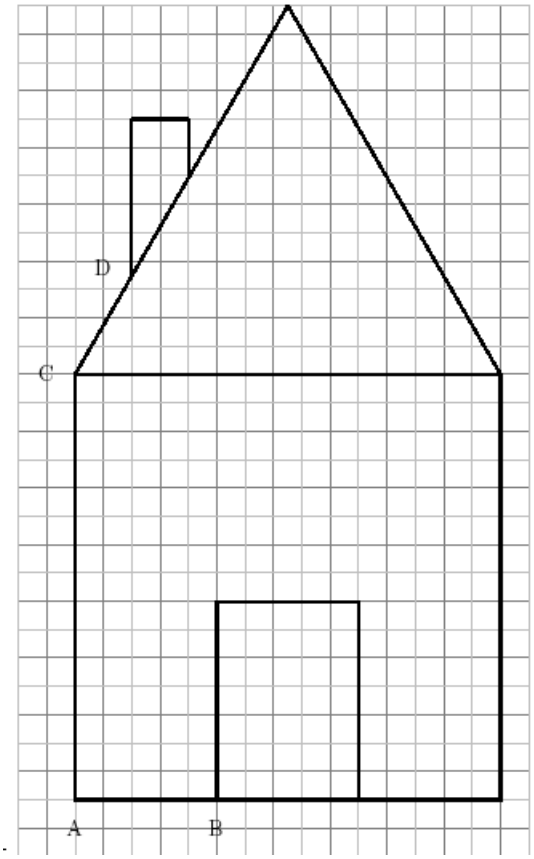
Poste oni fermu l' redaktilon registrante l' modifojn per klaki la butonon testudo. Nun je ĉiu fojo kiam oni tajpas `kvadrato`, kvadrato aperas sur l' ekrano!

5.4 Ekzerco...

Malgranda reta ĉelo reprezentas 10 testudpaŝojn.

Provu realigi la grafikon malsupran per difini ok procedurojn:

- Proceduron «`kvadrato`» kiu grafikos la bazan kvadraton de la domo.
- Proceduron «`tri`» kiu grafikos l' egallateran trilateron kiu reprezentos la tegmenton doman.
- Proceduron «`pordo`» kiu grafikos l' ortangulon reprezentantan la pordon.
- Proceduron «`kam`» kiu grafikos la kamentubon.
- Proceduron «`mov1`» kiu ebligos la testudon moviĝi de pozicio A al pozicio B.
- Proceduron «`mov2`» kiu ebligos la testudon moviĝi de pozicio B al pozicio C.
- Proceduron «`mov3`» kiu ebligos la testudon moviĝi de pozicio C al pozicio D. (Atentu: eble necesos levi la testudan krajonon...)
- Proceduron «`domo`» kiu ebligos grafiki la domon tutan helpe de ĉiuj aliaj proceduroj.



Ĉapitro 6

Uzi koordinatojn

Nivelo: komencanto

6.1 Prezentado

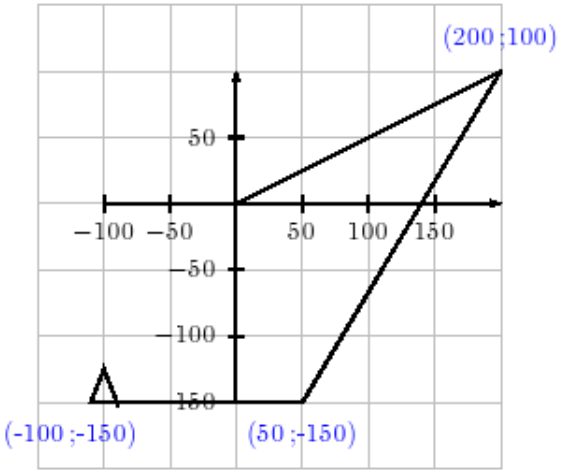
En tiu ĉi ĉapitro, ni malkovros la primitivon `situon_provizu`. La desegnareo havas dratreton kies origino estas lokita je la centro de la ekrano. Oni povas atingi ĉiun punkton de la desegnejo per helpo de ĝiaj koordinatoj.

```
sitp listo                    sitp [100 -250]
```

Movas la testudon al la punkto kies koordinatojn difinas la listo.

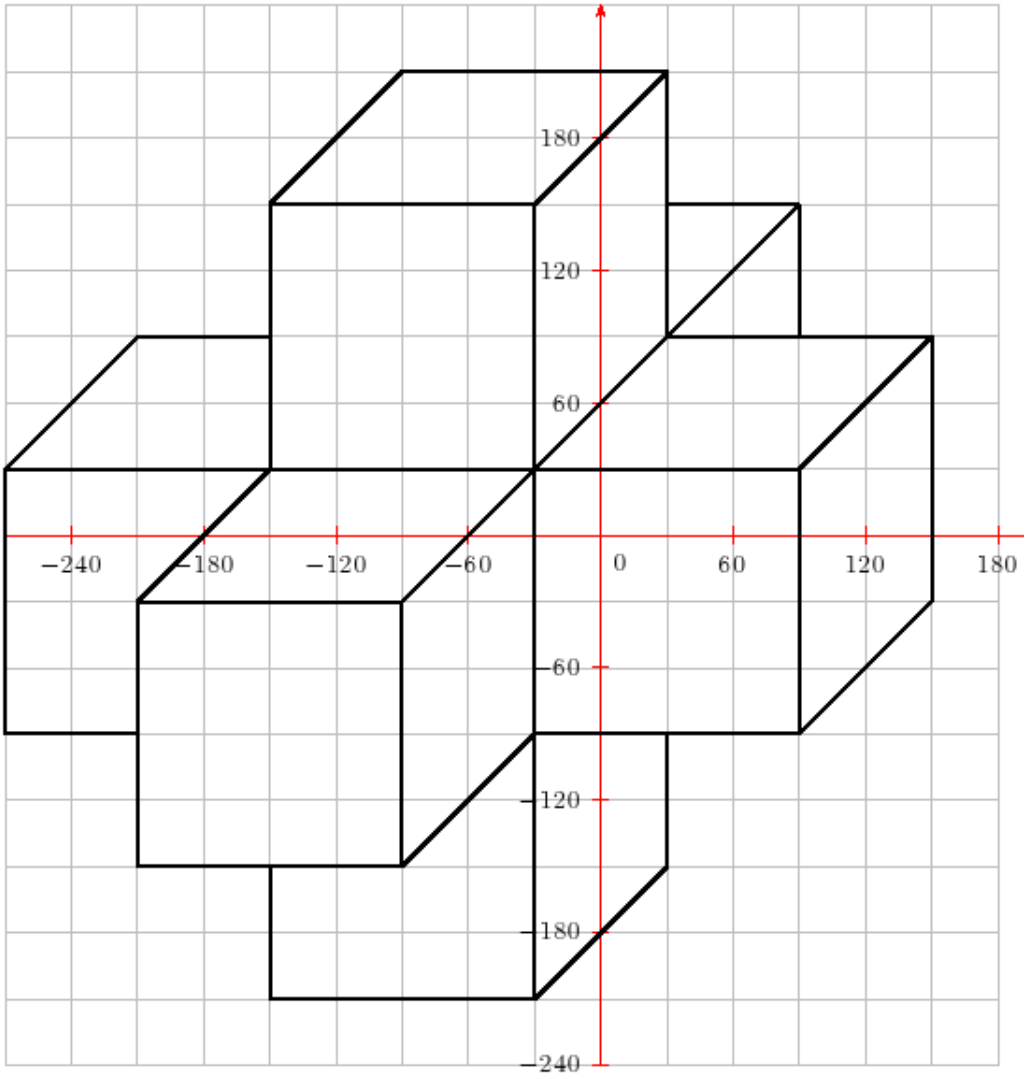
Malgranda uzekzemplo:

```
ev sitp [200 100] sitp [50 -150] sitp [-100 -150]
```



6.2 Ekzerco:

Realigu tiun figuron nur uzante la primitivojn: `sitp`, `ev`, `l`, `ml`.



Ĉapitro 7

La variabloj

Nivelo: komencanto

Kelkafoje, oni deziras grafiki figuron je malsamaj skaloj. Por ekzemplo, se oni dezirus desegni kvadraton de latero 100, kvadraton de latero 200 kaj kvadraton de latero 50, oni difinus tri malsamajn procedurojn rilatajn al ĉiu kvadrato.

```
por kvadrato1
ripetu 4 [an 100 dn 90]
fino
por kvadrato2
ripetu 4 [an 200 dn 90]
fino
por kvadrato3
ripetu 4 [an 50 dn 90]
fino
```

Oni rimarkas tuj ke estus pli simple, difini solan proceduron al kiu oni dirus la ĝustan longon de la latero desegnota. Ekzemple, `kvadrato 200` grafikus la kvadraton de latero 200, `kvadrato 100` grafikus la kvadraton je latero 100, ktp. Ja tion ebligos la variabloj.

7.1 Uzekzemploj

Por grafiki kvadraton je latero 100, oni uzu:

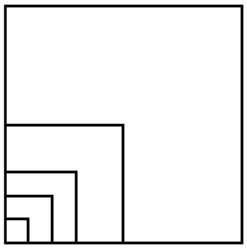
```
por kvadrato
ripetu 4 [an 100 dn 90]
fino
```

Ni modifos tiun proceduron por ke ĝi ricevu parametron (oni diras egale «argument») indikantan la longon grafikotan. Variabla nomo ĉiam estas antaŭata de la signo «:». Kiam oni volas indiki ke la proceduron `kvadrato` dependas je la variablo `:l`, oni aldonu `:l` ĉe la fin' de la lini' de la difino.

Tiel, oni antaŭeniros ne plu 100 testudpaŝojn, sed `:l` testudpaŝojn. La proceduro estiĝu:

```
por kvadrato :l
ripetu 4 [an :l dn 90]
fino
```

Tiel, tajpante: `kvadrato 100 kvadrato 50 kvadrato 30 kvadrato 20 kvadrato 10`



7.2 Grafiki ortangulon je longo kaj larĝo difinitaj

Oni difinos ĉi tie proceduron nomatan `ort` kiu dependu je du variabloj reprezentantaj la du dimensiojn de ortangulo. `ort 200 100` grafikos ortangulon je alto 200 kaj larĝo 100.

```
por ort :lo :la
ripetu 2 [an :lo dn 90 an :la dn 90]
fino
```

Faru provojn:

```
ort 200 100 ort 100 300 ort 50 150 ort 1 20 ort 100 2
```

Kompreneble, se vi donas nur unu argumenton al la proceduro `ort`, l' interpretilo signalos per erarmesaĝo ke la proceduro atendas alian argumenton.

7.3 Grafiki formon je malsamaj ampleksoj

Ni jam vidis kiel grafiki kvadraton, ortangulon je malsamaj ampleksoj. Ni reprenos l' ekzemplon de la domo de p. 30 kaj vidos kiel modifi la kodon por grafiki la domon je la dezirata skalo.

La celo estas pasigi argumenton al proceduro `domo` por ke laŭ la parametro, la domo estu pli aŭ malpli granda. Ni deziras ke `domo 1` grafiku la domon je reala amplekso.

`domo 0.5` grafikos domon je skalo 0.5.

`domo 2` grafikos domon je dimensioj duoblaj, ktp.

La koncepto proporcieco estas kompreneble subkaŝita. En reala grando, la proceduro `kvadrato` estis jena:

```
por kvadrato
ripetu 4 [an 150 dn 90]
fino
```

Ĉiuj originalaj diminsioj de la domo estas multiplikitaj per la skalo. La proceduro `kvadrato` estiĝas:

```
por kvadrato :l
ripetu 4 [an 150*:l dn 90]
fino
```

Do kiam oni tajpos `kvadrato 2`, la kvadrato havos lateron longan je $150 \times 2 = 300$. La proporciojn oni respektos! Efektive, oni rimarku ke necesos repreni ĉiujn procedurojn kaj ŝanĝi la longojn je movo laŭ la jena maniero:

`an 70` fariĝos `an 70*:l`

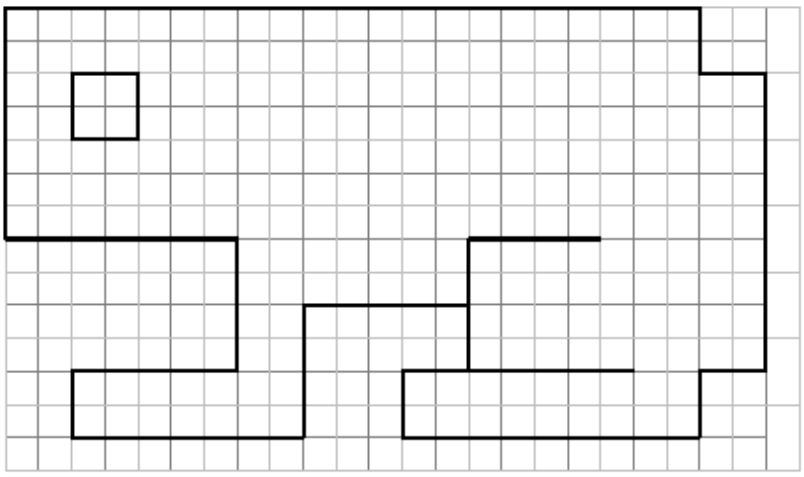
`an 45` fariĝos `an 45*:l`

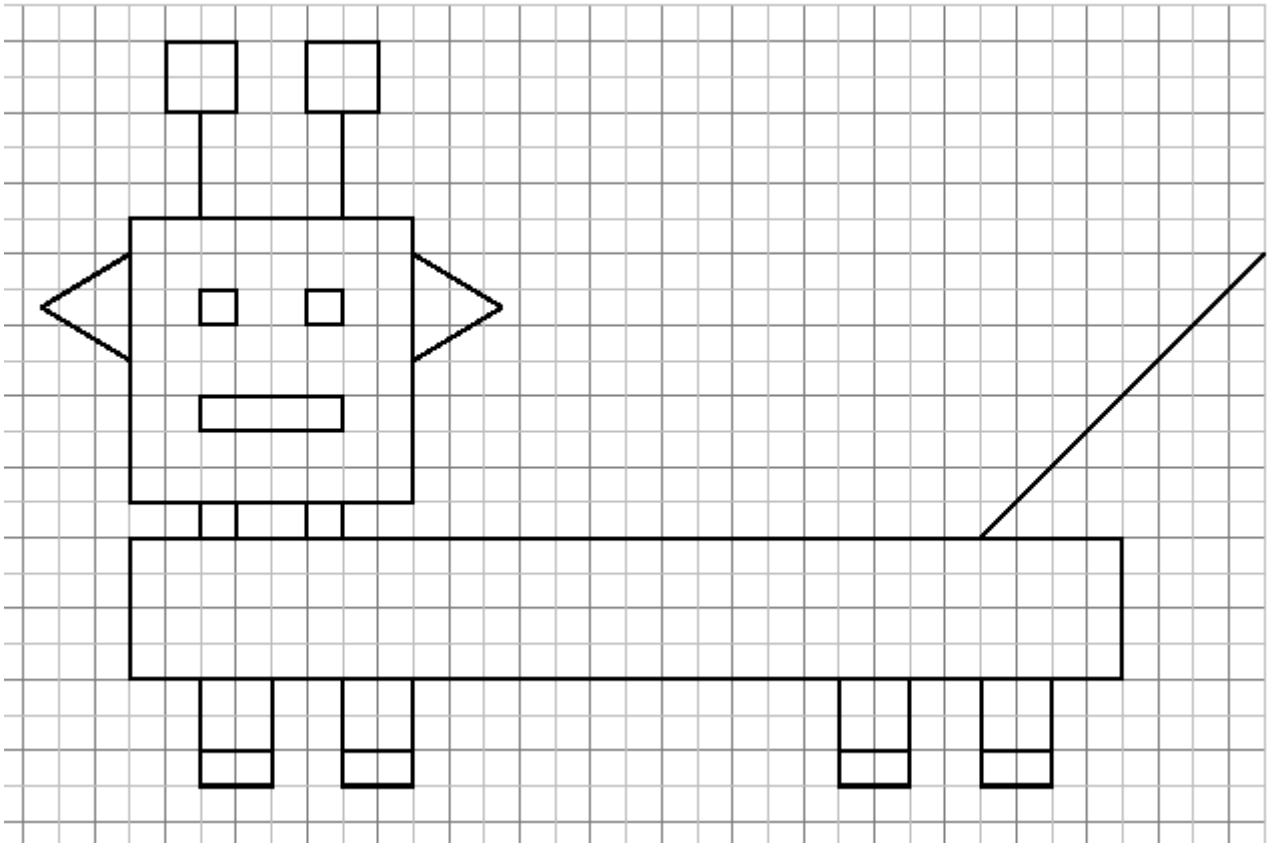
ktp.

```
por kvadrato :l
ripetu 4 [an 150*:l dn 90]
fino

por tri :l
ripetu 3[an 150*:l dn 120]
fino

por pordo :l
ripetu 2 [an 70*:l dn 90 an 50*:l dn 90]
fino

por kam :l
an 55*:l dn 90 an 20*:l dn 90 an 20*:l
fino

por mov1 :l
dn 90 an 50*:l mdn 90
fino

por mov2 :l
mdn 90 an 50*:l dn 90 an 150*:l dn 30
fino

por mov3 :l
l dn 60 an 20*:l mdn 90 an 35*:l ml
fino

por dom :l
kvadrato :l mov1 :l pordo :l mov2 :l tri :l mov3 :l kam :l
fino
```

7.4 Ekzerco:

Realigu la desegnojn jenajn per variabloj tiel ke oni povas obteni ilin je diversaj ampleksoj.





Ĉapitro 8

La rekursiveco

Nivelo: meza

La lingvo Logo uzas tre ofte teĥnikon programadan nomatan rekursiveco. En ĉi tiu ĉapitro, ni malkovros tuj tiun koncepton per simplaj ekzemploj por poste profundiĝi per ĉefe la desegnado de fraktalo nomata la neĝero de Van Koch. Por komenci, jen malgranda klarigo:

Proceduro estas rukursivo se ĝi vokas sin mem.

8.1 En desegnejo

8.1.1 Unua ekzemplo

```
por ekz1
dn 1
ekz1
fino
```

Tiu proceduro estas rekursiva ĉar la proceduro `ekz1` estas vokata je la lasta linio. Dum la rulado, oni konstatas ke la testudo ne ĉesas turniĝi. Por haltigi la programon, oni nepre premu la butonon STOP.

8.1.2 Dua ekzemplo

Antaŭ ĉio, jen tri novaj primitivoj:

- `atendu nombro` `atendu 60`
  Haltigu la programon dum tiom da 60^{onoj} de sekundo kiel indikite.
  Ekzemple, `atendu 120` haltigos la programon dum du sekundoj.

- `gum,gumskrapu` gumskrapu
  Kiam la testudo moviĝas, ĝi forviŝas anstataŭ skribi post si.

- `desegne` desegne
  Metu la testudon en la moduson de klasika desegno: la testudo skribas post si dum moviĝi.

```
por ekz2
an 200 gum atendu 60
man 200 desegne dn 6
ekz2
fino
```

Nur restas ruli tiun programon. Je ĉiu sekundo la sama motivo rekomenciĝas kaj la programo ŝajnigas sekundhorloĝon!

8.2 En tekstejo

8.2.1 Unua ekzemplo

La primitivo `skribu, s` ebligas skribi tekston en la tekstareon. Ĝi atendas kiel argumenton jen liston, jen vorton. Ekz.: `s "saluton s [Mi skribas kion mi volas]`. (Ne forgesu la citilon " kiam oni volas skribi nur vorton.)

```
por ekz3 :n
skribu :n
ekz3 :n+1
fino
```

Rulu la komandon `ekz3 0`, poste haltigu per butono STOP. Faru la ŝanĝojn necesajn en tiu programo por ke la numeroj aperu duope.

Nun mi volas skribi ĉiun nombron pli grandan ol 100 kiu estas en la multipliktabelo de la kvino. Sufiĉas do modifi la programon jene:

```
por ekz3 :n
skribu :n
ekz3 :n+5
fino
```

kaj ruli: `ekz3 100`

8.2.2 Realigi eliran provon

Tajpu la jenajn komandojn:

```
se 2+1=3 [skribu [tio estas vera]]
se 2+1=4 [skribu [tio estas vera]] [skribu [la kalkulo estas malvera]]
se 2+5=7 [s "vera] [s "malvera]
```

Se vi ankoraŭ ne komprenas la sintakson de la primitivo `se`, adresiĝu al la referenca gvidlibro XLogo.

```
por ekz3 :n
se :n=100 [haltu]
skribu :n
ekz3 :n+1
fino
```

Rulu la komandon `ekz3 0`

Faru la ŝanĝojn necesajn en tiu programo por aperigi la nombrojn kuŝantaj inter 55 kaj 350 kiu estas en la multipliktabelo de la dek-unuo.

8.3 Ekzemplo de fraktalo: la neĝero de Koch

Danke al la rekursiveco, tre facilas generi en Logo objektojn nomatajn en matematiko fraktaloj.

Jen la unuaj stadioj ebligantaj krei la malglatan linion de Van Koch.

En ĉiu stadio:

1. Ĉiu segmento estu partigita en tri egalajn partojn.

2. Oni grafiku egallateran trilateron sur la dua segmento.

3. Oni forigu tiun duan segmenton.

Rimarkenda: Konsiduru la duan stadion; konstatu ke tiun linion formas kvar motivoj rilataj al l' antaŭa stadio kaj kies amplekso estas triono. Tiel evidentiĝas la rekursiva naturo de la fraktalo.

Nomu $L_{n,\ell}$ la motivon longa je ℓ, grafikita en la stadio n. Por grafiki tiun motivon jen la procedo:

1. Desegnu $L_{n-1,\ell/3}$

2. Turnu maldekstren je 60 gradoj

3. Desegnu $L_{n-1,\ell/3}$

4. Turnu dekstren je 120 gradoj.

5. Desegnu $L_{n-1,\ell/3}$

6. Turnu maldekstren je 60 gradoj

7. Desegnu $L_{n-1,\ell/3}$

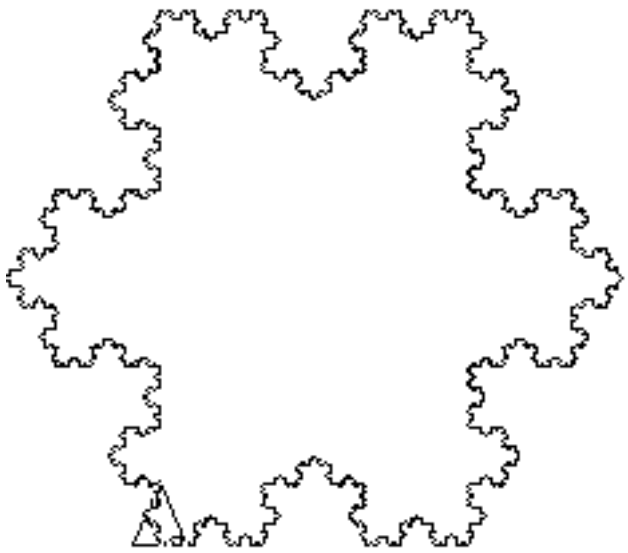
En Logo, tio fariĝas tutsimple:

```
# :l longo de la motivo
# :p stadio
por linio :l :p
se :p=0 [an :l]
  [linio :l/3 :p-1 dn 60 linio :l/3 :p-1 dn 120 linio :l/3 :p-1 dn 60 linio :l/3 :p-1]
fino
```

Se oni desegnas egallateran trilateron konsistanta el tri tiaj linioj, oni akiras mirindan neĝeron de Van Koch

```
# :l longo de la latero
por neĝero :l :p
ripetu 3 [linio :l :p dn 120]
fino
```

Poste rulu: `flocon 200 6`



8.4 Rekursiveco pri vortoj

Priserĉu la liston de primitivoj je p. 85 por kompreni la rolon de la primitivoj `vort`, `lastan`, kaj `senlastan`.

Jen rekursiva proceduro kiu ebligas renversi l' ordon de la literoj de vorto.

```
por renversuv :v
se malplena? :v [sendu "]
sendu vorton lastan :v renversuv senlastan :v
fino

skribu renversuv "abcĉde
edĉcba
```

Oni diras ke vort' estas palindromo se oni povas legi ĝin je ambaŭ direktoj (ekzemploj: ama, radar', onano...).

```
# testu ĉu la vorto :v estas palindromo
por palindromo :m
se  :m = renversuv :m [sendu vera] [sendu malvera]
fino
```

Kaj finfine jen mojosa programeto (dankon Olivier SC):

```
por palin :n
se palindromo :n [skribu :n haltu]
skribu (list :n "PLUS renversuv :n "EGALAS sumon :n renversuv :n)
palin :n + renversuv :n
fino

palin 78
78 PLUS 87 EGALAS 165
165 PLUS 561 EGALAS 726
726 PLUS 627 EGALAS 1353
1353 PLUS 3531 EGALAS 4884
4884
```

8.5 Kalkuli faktorialon

Oni difinas faktorialon de 5, indikite 5! jene:

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

Ĝenerale, por n strikte pozitiva, rimarku ke: $n! = n \times (n-1)!$. Tiu rilato klarigas la rekursivan naturon de jena programo:

```
por fak :n
se :n=0 [snd 1] [snd :n * fak :n-1]
fino

s fak 5
120
s fak 6
720
```

8.6 Proksimumo de π

Oni povas akiri proksimumon de la nombro π per la formulo:

$$\pi \approx 2^k \sqrt{2 - \sqrt{2 + \sqrt{2 + \ldots \sqrt{2 + \sqrt{2}}}}}$$

kie k estas la nombro de kvadrataj radikoj. Ju pli granda estas k des pli tiu esprimo proksimiĝas al nombro π.

La formulo konsistas el la esprimo $2 + \sqrt{2 + \ldots \sqrt{2 + \sqrt{2}}}$ kiu estas klare rekursiva, de kie la programo jena:

```
# k estas la nombro de radikoj
por aprokspi :k
tajpu "Proksimume:\  s (potencon 2 :k) * radikon (2 - radikon (kalk :k-2))
s "-------------------------
tajpu "Pi:\  s pi
fino

por kalk :p
se :p=0 [snd 2] [snd 2 + racine kalk :p-1]
fino

aprokspi 10
Proksimume: 3.141591421568446
-------------------------
Pi: 3.141592653589793
```

Oni akiris la 5 unuajn decimalojn! Se oni deziras pli, necesos forigi kelkajn kalkulerarojn pro ne precize kalkuli la koncernitajn kvadrataj radikojn. Por tio ni pligrandigos la nombron de decimaloj per la primitivo `decimalojn_provizu`.

```
decimalojn_provizu 100
aprokspi 100
Proksimume: 3.14159265358979323846264338327950288419733930696701609758076843138804 68...
-------------------------
Pi: 3.14159265358979323846264338327950288419716939937510582097494459230781640 6....
```

Kaj nun oni akiras 39 decimalojn...

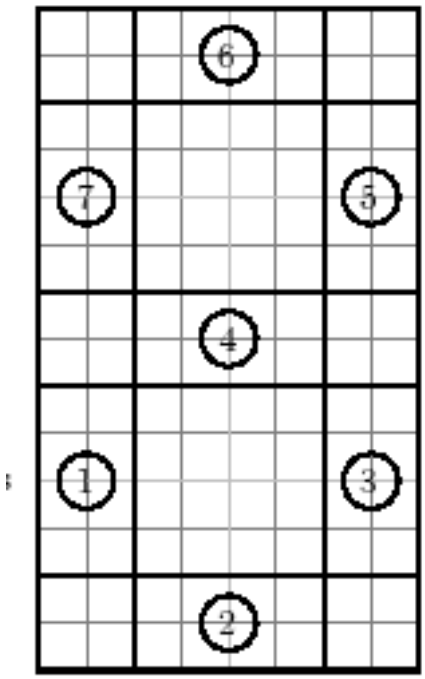
Ĉapitro 9

Krei movadon

Nivelo: meza

Ĉi tiu ĉapitro proponas du aferojn tre malsamajn kies celo estas krei movadon per XLogo.

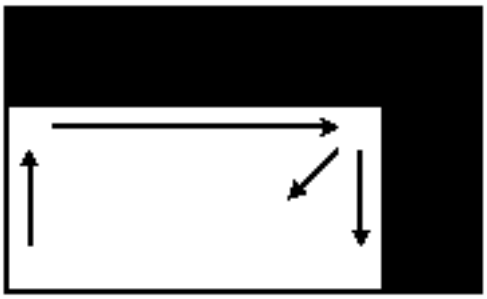
9.1 La ciferoj de la kalkulilo



Ĉi tiu tasko baziĝas sur la fakto ke ĉiun kalkulilan nombron oni povas akiri per la jena ŝablono:

- Por ekzemplo, por desegni «4», oni ŝaltu l' ortangulojn 3, 4, 5, 7.
- Por desegni «8», oni ŝaltu l' ortangulojn 1, 2, 3, 4, 5, 6, 7.
- Por desegni «3», oni ŝaltu l' ortangulojn 2, 3, 4, 5, 6.

9.1.1 Plenigi ortangulon



Se oni deziras ekzemple grafiki plenan ortangulon grandan je 100 mul 200, unua ideo povas esti desegni la ortangulon je 100 mul 200, poste desegni ortangulon je 99 mul 199, poste ortangulon je 98 mul 198... ĝis la ortangul' estos tute plena.

Ni komencu per difini ortangulon je langoj kaj larĝo dependaj je du variabloj.

```
por ort :lo :la
ripetu 2 [an :lo dn 90 an :la dn 90]
fino
```

Por plenigi nian grandan ortangulon, oni rulu:

`ort 100 200 ort 99 199 ort 98 198 .....  ort 1 101`

Difinu tiam proceduron ortangulo dediĉita grafiki tiun plenan ortangulon.

```
por ortangulo :lo :la
ort :lo :la
ortangulo :lo-1 :la-1
fino
```

Oni provu `ortangulo 100 200` kaj oni rimarku ke estas problemo: la proceduro ne haltas kiam la ortangulo estas plena; ĝi plu grafikas ortangulojn! Oni do aldonu provon ebligantan detekti ĉu la longo aŭ la larĝo egalas 0. Tiam, oni petas la programon halti per la komando `haltu`.

```
por ortangulo :lo :la
se aŭ :lo=0 :la=0 [haltu]
ort :lo :la
ortangulo :lo-1 :la-1
fino
```

Rimarko: anstataŭ uzi la primitivon `aŭ`, oni povas uzi la simbolon «|»; oni skribus:

```
se :lo=0 | :la=0 [haltu]
```

9.1.2 La programo

Ni bezonas la plenan ortangulon antaŭan:

```
por ort :lo :la
se :lo=0 |:la=0 [haltu]
ripetu 2 [an :lo dn 90 an :la dn 90]
ort :lo-1 :la-1
fino
```

Ni supozas ke la testudo ekiras de la malsupra maldekstra angulo. Ni difinos proceduron nomatan `cifero` akceptantan 7 argumentojn `:a`, `:b`, `:c`, `:d`, `:e`, `:f`, `:g`. Kiam `:a` valoras 1, oni desegnu la ortangulon 1. Se `:a` valoras 0, oni ne desegnu ĝin. Jen la principo.

Jen la proceduro:

```
por cifero :a :b :c :d :e :f :g
# Oni desegnu la ortangulon 1
se :a=1 [ort 160 40]
# Oni desegnu la ortangulon 2
se :b=1 [ort 40 160]
l dn 90 an 120 mdn 90 ml
# Oni desegnu la ortangulon 3
se :c=1 [ort 160 40]
l an 120 ml
# Oni desegnu la ortangulon 5
se :e=1 [ort 160 40]
# Oni desegnu la ortangulon 4
mdn 90 l man 40 ml
se :d=1 [ort 160 40]
# Oni desegnu la ortangulon 6
dn 90 l an 120 mdn 90 ml
se :f=1 [ort 160 40]
# Oni desegnu la ortangulon 7
l an 120 mdn 90 man 40 ml
se :g=1 [ort 160 40]
fino
```

9.1.3 Krei malgradan animadon

Ĉi tie ni simulos retronombradon aperigante sekvence la ciferojn de 9 ĝis 0 je ordo malkreska.

```
por retronombro
ev tdk cifero 0 1 1 1 1 1 1 atendu 60
ev tdk cifero 1 1 1 1 1 1 1 atendu 60
ev tdk cifero 0 0 1 0 1 1 0 atendu 60
ev tdk cifero 1 1 1 1 0 1 1 atendu 60
ev tdk cifero 0 1 1 1 0 1 1 atendu 60
ev tdk cifero 0 0 1 1 1 0 1 atendu 60
ev tdk cifero 0 1 1 1 1 1 0 atendu 60
ev tdk cifero 1 1 0 1 1 1 0 atendu 60
ev tdk cifero 0 0 1 0 1 0 0 atendu 60
ev tdk cifero 1 1 1 0 1 1 1 atendu 60
fino
```

Jen malgranda problemo: estas palpebruma efiko malagraba dum krei ĉiun ciferon. Por fluemigi tion oni uzos la primitivojn `movado`, `neplu_movigu` kaj `novigu`.

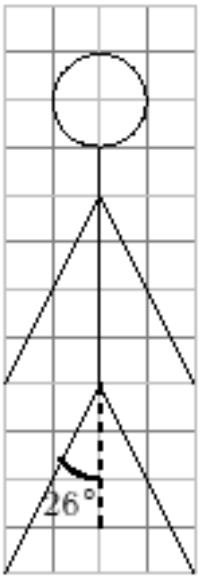
- `movado` ebligas ŝalti la moduson «movado». La testudo ne desegnos plu sur l' ekrano, sed en bufro, tio estas, en memoro. Ĝi aldonos la bildon nur kiam oni petos per la primitivo `novigu`.
- `neplu_movigu` ebligas malŝalti tiun moduson kaj reveni en la klasikan moduson.

Jen la modifita programo:

```
por retronombro
# Pasu en moduson movado
movado
ev tdk cifero 0 1 1 1 1 1 1 novigu atendu 60
ev tdk cifero 1 1 1 1 1 1 1 novigu atendu 60
ev tdk cifero 0 0 1 0 1 1 0 novigu atendu 60
ev tdk cifero 1 1 1 1 0 1 1 novigu atendu 60
```

```
ev tdk cifero 0 1 1 1 0 1 1 novigu atendu 60
ev tdk cifero 0 0 1 1 1 0 1 novigu atendu 60
ev tdk cifero 0 1 1 1 1 1 0 novigu atendu 60
ev tdk cifero 1 1 0 1 1 1 0 novigu atendu 60
ev tdk cifero 0 0 1 0 1 0 0 novigu atendu 60
ev tdk cifero 1 1 1 0 1 1 1 novigu atendu 60
# Revenu en moduson klasikan
neplu_movigu
fino
```

9.2 Animado: la hometo kiu kreskas



Antaŭ ĉio, ni difinu proceduron `hometo` kiu grafikas la hometon apudan je elektita amplekso.

```
por hometo :c
mdn 154 an 44*:c man 44*:c
mdn  52 an 44*:c man 44*:c
mdn 154 an 40*:c
mdn 154 an 44*:c man :c*44
mdn  52 an 44*:c man :c*44
mdn 154 an 10*:c
mdn  90 ripetu 180 [an :c/2 dn 2] dn 90
fino
```

Nun ni kreos animadon ŝajnigantan ke la hometon kreskas po malmulte. Por tio, ni grafikos `hometo 0.1`, poste `hometo 0.2`, `hometo 0.3`... ĝis `hometo 5`. Inter ĉiu grafikado, oni forviŝos l' ekranon. Jen la du proceduroj:

```
por hometo :c
mdn 154 an 44*:c man 44*:c
mdn  52 an 44*:c man 44*:c
mdn 154 an 40*:c
mdn 154 an 44*:c man :c*44
mdn  52 an 44*:c man :c*44
mdn 154 an 10*:c
mdn  90 ripetu 180 [an :c/2 dn 2] dn 90
```

```
se :c=5 [haltu]
ev tdk hometon :c+0.1
fino

por komenci
ev tdk
hometo 0
fino
```

Finfine, por fluemigi la tuton, oni helpu sin per la moduson movado kaj la primitivo `novigu`.

```
por hometo :c
mdn 154 an 44*:c man 44*:c
mdn  52 an 44*:c man 44*:c
mdn 154 an 40*:c
mdn 154 an 44*:c man :c*44
mdn  52 an 44*:c man :c*44
mdn 154 an 10*:c
mdn  90 ripetu 180 [an :c/2 dn 2] dn 90
novigu
se :c=5 [haltu]
ev tdk hometo :c+0.1
fino

por komenci
tdk movado
hometo 0
neplu_movigu
fino
```

Rimarku: Tie, la proceduro `hometo` estas rekurziva; oni pli simple povus uzi la primitivon `ripetupor` por variigi `:c` de 0.1 ĝis 5. Jen la programo tiel:

```
por hometo :c
ev tdk mdn 154 an 44*:c man 44*:c
mdn  52 an 44*:c man 44*:c
mdn 154 an 40*:c
mdn 154 an 44*:c man :c*44
mdn  52 an 44*:c man :c*44
mdn 154 an 10*:c
mdn  90 ripetu 180 [an :c/2 dn 2] dn 90
novigu
fino

por komenci
tdk movado
ripetupor [c 0 5 0.1] [hometo :c]
neplu_movigu
fino
```

Ĉapitro 10

Interaktiva programado

Nivelo: komencanto

10.1 Komuniki kun l' uzulo

Ni realigos malgrandan programon kiu demandas de l' uzulo ŝlian nomon, baptonomon kaj aĝon. Je l' fin' de la demandaro, la programo respondos per memorigilo jene:

```
Via familinomo estas:........
Via baptonomo estas: .......
Via aĝo estas: .......
Vi estas (mal)plenkreskulo
```

Por tio, ni uzos la jenajn primitivojn:

- `legu`: `legu [Kiom estas via aĝo? ] "a`

  Aperigas dialogfenestron kun titolo kiel la listo argumento (tie, «Kiom estas via aĝo?»). La respondo donita de l' uzulo estas memorita kiel vorto aŭ listo (se l' uzul' tajpas plurajn vortojn) en la variablo `:a`.

- `provizu, p`: `provizu "a 30`

  Donas la valoron 30 al la variablo `:a`

- `frazon, fr`: `frazon [30 k] "a`

  Aldonas valoron en liston. Se tiu valoro estas listo, kunigas la du listojn.

  ```
  frazon [30 k] "a ---> [30 k a]
  frazon [1 2 3] 4 ---> [1 2 3 4]
  frazon [1 2 3] [4 5 6] ---> [1 2 3 4 5 6]
  ```

Jen la kodo:

```
por demandaro
legu [Kiom aĝas vi?] "aĝo
legu [Kio estas via familinomo?] "famnomo
legu [Kio estas via baptonmo?] "bapnomo
skribu frazon [Via familinomo estas: ] :famnom
skribu frazon [Via baptonomo: ] :bapnomo
skribu frazon [Via aĝo estas: ] :aĝo
se aŭ :aĝo>18 :aĝo=18 [skribu [Vi estas plenkreskulo]] [skribu [Vi estas malplenkreskulo]]
fino
```

10.2 Programi malgrandan ludon

La celo de ĉi tiu sekcio estas krei la jenan ludon:

La programo elektas hazardan nombron inter 0 kaj 32 kaj memoras ĝin. Dialogfenestro aperas kaj demandas l' uzulon enigi nombron. Se la proponita nomo estas egala al la memorita nomo, ĝi skribas «venkis»en la tekstejo. En mala okazo, la programo indikas ĉu la nombro memorita estas pli malgranda aŭ granda ol la nombro proponita de l' uzulo; poste ĝi reaperigas la dialogfenestron. La programo haltos kiam l' uzulo trovas la memoritan nombron.

Vi bezonos uzi la jenan primitivon:

`hazardon, hzd`: `hazardon 8`

`hazardon 20` donas nombron hazarde elektitan inter 0 kaj 19.

Jen kelkaj reguloj respektendaj por realigi tiun ludon:

- La nombro memorita de l' komputilo estas memorata en variablo nomata `nombro`.
- La dialogfenestro havos por titolo; «Proponu nombron:».
- La nombro proponita de l' uzulo estos registrita en variablo nomata `provo`.
- La proceduro kiu ebligas ruli la ludon nomiĝos `ludo`.

Kelkaj eblaj plibonigoj:

- Skribi la nombro de provoj.
- La nombro serĉota estu inter 0 kaj 2000.
- Konstati ĉu tio enigita de l' uzulo estas vere nombro. Por tio, uzu la primitivon `nombra?`.

  Exemples: `nombra? 8` estas vera.
  `nombra? [5 6 7]` estas malvera ([5 6 7] estas listo sed ne nombro).
  `nombra? "abcde` estas malvera ("abcde estas vorto sed ne nombro).

Ĉapitro 11

Temo: Sumi du kubojn

Nivelo: Meza

Kiam oni ĵetas du kubojn kaj kalkulas la tuton de poentoj de ambaŭ kuboj, oni akiras entjeron inter 2 kaj 12. Ĉi tie, ni vidos la distribuon de la diversaj rezultoj kaj ĝin reprezentos per malgranda grafiko.

11.1 Simuli ĵeti kubon

Por simuli ĵeton de kubo, ni uzos la primitivon `hazardon`. Jen kiel procedi.

`hazardon 6` ⟶ redonas entjeron hazarde elektitan el 0, 1, 2, 3, 4, 5.

Tial, `(hazardon 6) + 1` redonas entjeron elektitan el 1, 2, 3, 4, 5, 6. Rimarku ja la parentezojn; alie l' interpretilo Logo komprenus `hazardon 7`. Por ŝpari l' parentezojn, oni povas tajpi `1 + hazardon 6`.

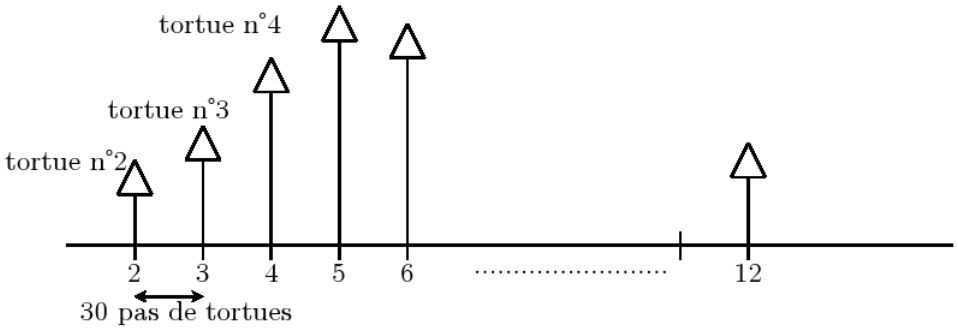
Oni difinu tiel la proceduron `ĵetu` kiu simulas ĵeti ludkubon.

```
por ĵetu
  sendu 1 + hazardon 6
fino
```

11.2 La programo

Ni uzos la moduson plur-testudan. Por tiel havi plurajn testudojn sur l' ekran', oni uzu la primitivon `testudon_provizu` sekvitan de la numero de la testudo kiun oni volas elekti.

Bona skemo valoras pli ol mil klarigoj...



Ĉiu testudo numerata de 2 ĝis 12 antaŭeniros unu testudpaŝon kiam la sumo de la du kuboj ĵetitaj estas egala a ĝia numero. Por ekzemplo, se la kubo sumiĝas 8, la testudo 8a antaŭeniru unu paŝon. Inter ĉiuj du testudoj estu 30 testudpaŝoj horizontale.

Oni lokos la testudojn per koordinatoj.

- La testudo n°2 estu lokita en $(-150; 0)$

- La testudo n°3 estu lokita en $(-120; 0)$
- La testudo n°4 estu lokita en $(-90; 0)$
- La testudo n°5 estu lokita en $(-60; 0)$

$\vdots$

```
 testudon_provizu 2 sitp [-150 0]
 testudon_provizu 3 sitp [-120 0]
 testudon_provizu 4 sitp  [-90 0]
 testudon_provizu 5 sitp  [-60 0]
 testudon_provizu 6 sitp  [-30 0]
.....
```

Pli bone ol tajpi 11 fojojn preskaŭ la saman komandlinion, oni aŭtomatigu tion uzante la primitivon `ripetupor`. Per tiu primitivo, oni povas havigi al variablo sekvencon de valoroj prenitaj en intervalo laŭ samaj spacoj. Ĉi tie, oni volas ke la variablo `:i` prenu sinsekve la valorojn 2, 3, 4... 12. Oni tajpu:

```
ripetu por [i 2 12] [ listo de rulotaj instrukcioj ]
```

Por loki la testudojn, oni kreu do la proceduron `pretigu`

```
por pretigu
  ev tdk
  ripetupor [i 2 12]
      [# Loku la testudon
       testudon_provizu :i sitp liston -150+(:i-2)*30 0
       # Skribu la numeron de la testudo apude sube
       l man 15 etikedu :i an 15 ml]
fino
```

Bone komprenu la formulon `-150+(:i-2)*30`. Oni ekiras de -150; poste por ĉiu nova testudo oni aldonas 30. Provu per la diversaj valoroj de `:i` se vi ne estas konvinkita.

Finfine jen la programo:

```
por ĵeti
 sendu 1 + hazardon 6
fino

por pretiu
  ev tdk
  ripetupor [i 2 12]
      [# Loku la testudon
       testudon_provizu :i sitp liston -150+(:i-2)*30 0
       # Skribu la numeron de la testudo apude sube
       l man 15 etikedu :i an 15 ml]
fino

por startu
pretigu
# Realigu 1000 provoj
ripetu 1000
  [provizu "sumo ĵetu+ĵetu
   testudon_provizu :sumo an 1]
# Skribu la frekvencojn de la ĵetado
ripetupor [i 2 12]
 [testudon_provizu :i
  # L' ordinato de l' testudo reprezentas la nombron de ĵetoj
```

```
  loke_provizu "efika lastan sit
  l an 10 mdn 90 an 10 dn 90 ml etikedu :efika/1000*100]
fino
```

Jen ĝeneraligo de tiu programo. Oni demandos al l' uzulo la nombron de deziratajn kubojn kaj la nombron de ĵetojn farotajn.

```
por ĵetu
lokp "sumo 0
ripetu :kuboj
  [lokp "sumo :sumo + 1 + hazardon 6]
sendu :sumo
fino

por pretigu
  ev tdk testudkiomon_provizu :maks+1
  ripetupor fr list "i :min :maks
     [# Loku la testudon
      testudon_provizu :i sitp list (:min-:maks)/2*30+(:i-:min)*30 0
      # Skribu la numeron de la testudo apude sube
      l man 15 etikedu :i an 15 ml]
fino

por startu
leg [Nombro de kuboj:] "kuboj
se ne nombra? :kuboj [s [La nombro enigita ne estas valida!] haltu]
provizu "min :kuboj
provizu "maks 6*:kuboj
leg [Nombro de ĵetoj realigotaj] "ĵetoj
se ne nombra? :ĵetoj [s [La nombro enigita ne estas valida!] haltu]
pretigu
# Realigu 1000 provoj
ripetu :ĵetoj
  [testudon_provizu ĵetu an 1]
# Ŝkribu la frekvencojn de la ĵetoj
ripetupor fr list "i :min :maks
  [testudon_provizu :i
   # L' ordinato de l' testudo reprezentas la nombron de ĵetoj
   lokp "efika lastan sit
   # Oni proksimumu je 0.1
   l an 10 mdn 90 an 10 dn 90 ml etikedu (entjeran :efika/:ĵetoj*1000) / 10]
fino
```

Ĉapitro 12

Temo: Proksimumi probablike al π

Nivelo: Alta

AVERTO: Necesas kelkaj nocioj pri matematiko por bone kompreni ĉi tiun ĉapitron.

12.1 Nocio de pgkd (plej granda komuna dividanto)

Donitaj du entjeroj, ilia pgkd estas la plej granda el la dividantoj de ambaŭ.

- Por ekzemplo, 42 kaj 28 havas kiel pgkd 14 (ĝi dividas samtempe al 28 kaj al 42, kaj ĝi estas la plej granda el la nombroj tiaj).
- 25 kaj 55 havas kiel pgkd 5.
- 42 kaj 23 havas kiel pgkd 1.

Kiam du nombroj havas 1 kiel pgkd, oni nomas ilin primoj inter si. Do por l' antaŭa ekzemplo, 42 kaj 23 estas primoj inter si. Tio signifas ke ili havas neniun komunan dividanton krom 1 (kompreneble, ĝi dividas ĉiun entjeron!).

12.2 Algoritmo de Eŭklido

Por kalkuli la pgkd de du nombroj, oni povas uzi metodon nomatan algoritmo de Eŭklido (oni ne pruvos ĉi tie la validecon de tiu algoritmo). Jen la principo:

Donitaj du pozitivalaj entjeroj a kaj b, oni komence provu ĉu b estas nul. Se jes, tiam la PGKD egalas a. Se ne, oni kalkulu r, la resto de la divido de a per b. Anstataŭigu a per b, poste b per r, kaj rekomencu la procedon.

Ni kalkulu, ekzemple, la pgkd de 2160 kaj 888 per tiu algoritmo; jen la stadioj:

a	b	r
2160	888	384
888	384	120
384	120	24
120	24	0
24	0	

La pgkd de 2160 kaj 888 estas do 24. Estas neniu pli granda entjero kiu dividas tiujn du nombrojn. (Efektive $2160 = 24 \times 90$ kaj $888 = 24 \times 37$).

La pgkd estas efektive la lasta ne nula resto.

12.3 Kalkuli pgkd en LOGO

Malgranda rekursiva algortimo ebligas kalkuli la pgkd de du nombroj `:a` kaj `:b`:

```
por pgkd :a :b
se (rest :a :b) = 0 [sendu :b] [sendu pgcd :b rest :a :b]
fino

skribu pgkd 2160 888
24
```

Rimarku: Oni nepre metu parentezojn ĉirkaŭ `rest :a :b`; se ne, l' interpretilo provos evalui `:b = 0`. Por ŝpari la parentezojn, skribu: `se 0 = rest :a :b`

12.4 Kalkuli proksimumon de π

Efektive, konata rezulto de entjerteorio asertas ke la probablo ke du entjeroj hazarde elektitaj estas primoj inter si estas $6/\pi^2 \approx 0,6079$. Por provi konstati tiun rezulton, jen tio kion ni faros:

- Prenu du nombrojn hazarde inter 0 kaj 1 000 000.
- Kalkulu ilian pgkd.
- Se ĝi egalas 1, aldonu 1 al variablo nombrilo.
- Ripetu tion 1000 fojojn.
- La frekvencon de la paroj de nombroj primoj inter si oni akiru dividante la nombrilon per 1000 (la nombro de ripetoj).

```
por test
# Komencu la variablon nombrilo je 0
provizu "nombrilo 0
ripetu 1000
  [se (pgkd hazardon 1000000 hazardon 1000000) = 1 [provizu "nombrilo :nombrilo + 1]]
skribu [frekvenco:]
skribu :nombrilo / 1000
fino
```

Rimarko: Kiel antaŭe, oni devas meti la parentezojn ĉirkaŭ `pgkd hazardon 1000000 hazardon 1000000`; se ne, l' interpretilo provos evalui 1 000 000 = 1. Por ne skribi parentezojn, skribu tiel: `se 1 = pgkd hazardon 1000000 hazardon 1000000`.

Rulu la programon `test`.

```
test
0.609
test
0.626
test
0.597
```

Oni akiras valorojn proksimaj de la teoria valoro 0.6097. Rimarkindas ja ke tiu frekvenco estas valoro proksima al $\frac{6}{\pi^2}$.

Se mi indikas per f la trovitan frekvencon, oni do havas: $f \approx \frac{6}{\pi^2}$.

Do $\pi^2 \approx \frac{6}{f}$ kaj do $\pi \approx \sqrt{\frac{6}{f}}$.

Mi aldonu tiun proksimumigon en mia programo; mi transformu la finon de la proceduro `test`:

```
por test
# Komencu la variablon nombrilo je 0
provizu "nombrilo 0
ripetu 1000
  [se 1 = pgkd hazardon 1000000 hazardon 1000000 [provizu "nombrilo :nombrilo + 1]]
# Kalkulu la frekvencon
provizu "f :nombrilo/1000
# Skribu la valoron proksimuman al pi
skribu frazon [proksimumigo de pi:] radikon (6/:f)
fino
test
proksimumigo de pi: 3.164916190172819
test
proksimumigo de pi: 3.1675613357997525
test
proksimumigo de pi: 3.1008683647302115
```

Nu, mi modifu mian programon tiel ke kiam mi rulos ĝin, mi indiku la nombron de provoj deziratan. Mi intencas provi per 10 000 provoj; jen tio kion mi akiras en miaj tri unuaj ruladoj:

```
por test :provoj
# Komencu la variablon nombrilo je 0
provizu "nombrilo 0
ripetu :provoj
  [se 1 = pgkd hazardon 1000000 hazardon 1000000 [provizu "nombrilo :nombrilo + 1]]
# Kalkulu la frekvencon
provizu "f :nombrilo/:provoj
# Skribu la valoron proksimuman al pi
skribu frazon [proksimumigo de pi:] radiko (6/:f)
fino

test 10000
proksimumigo de pi: 3.1300987144363774
test 10000
proksimumigo de pi: 3.1517891481565017
test 10000
proksimumigo de pi: 3.1416626832299914
```

Ne malbone, ĉu?

12.5 Ni kompliku iom pli: π kiu generas π.....

Kio estas hazarda entjero? Ĉu hazarde prenita entjero inter 1 kaj 1 000 000 estas vere reprezentiva de iu ajn hazarda entjero? Oni rimarkas tre rapide ke nia modelado nur proksimiĝas de la ideala modelo. En ordo, ja pri la maniero generi la hazardan nombron ke ni realigos kelkajn ŝanĝojn... Ne ne uzos plu la primitivon `hazardon` sed uzos la sekvencon de la decimaloj de π. Mi klarigu: la decimaloj de π de ĉiam intrigis la matematikistojn pro ilia manko de reguleco; la ciferoj de 0 ĝis 9 ŝajnas aperi laŭ kvantoj preskaŭ egalaj kaj laŭ hazarda maniero. Ni vidos tuj kiel generi hazaradan nombron per decimaloj de π. Antaŭ ĉio, necesos kolekti la unuajn decimalojn de π (ekzeple unu milionon).

- Ekzistas malgrandaj programoj kiuj faras tion tre bone. Mi konsilas PiFast por Vindozo kaj ScnhellPi por Linukso.
- Pluku tiun dosieron de la retpaĝaro de XLogo:

http://downloads.tuxfamily.org/xlogo/common/millionpi.txt

Por krei niajn hazardajn nombrojn, ni prenu pakojn de 8 ciferojn el la sekvenco de decimaloj de π. Por klarigo, la dosiero komenciĝas tiel:

3.1415926 53589793 23846264 338327950288419716939 ktp

Unua nombro Dua nombro Tria nombro

Mi forigu la «.» de 3.14... kiu ĝenos kiam oni grupigos la decimalojn. Ĉio en ordo, ni kreu novan proceduron nomatan `hazardpi` kaj modifu malmulte la proceduron `test`:

```
por pgkd :a :b
se (rest :a :b) = 0 [sendu :b] [sendu pgkd :b rest :a :b]
fino

por test :provoj
# Malfermu flukson indikatan de la cifero 1 al la dosiero millionpi.txt
# (ĉi tie, supozate ke ĝi estas en la kuranta dosierujo;
# se ne, uzu dosieron_provizu kaj absolutan vojon)
flukson_malfermu 1 "millionpi.txt
# Provizu al la variablo linio la unuan linion de la dosiero millionpi.txt
provizu "linio unuan flukslinion_legu 1
# Komencu la variablon nombrilo je 0
provizu "nombrilo 0
ripetu :provoj
  [se 1 = pgkd hazardpi 7 hazardpi 7 [provizu "nombrilo :nombrilo + 1]]
# Kalkulu la frekvencon
provizu "f :nombrilo / :provoj
# Skribu la valoron proksimuman al pi
skribu frazon [proksimumigo de pi:] radiko (6/:f)
flukson_fermu 1
fino

por hazardpi :n
lokp "nombre "
ripetu :n
  [# Se estas plu neniu signo sur la linio
   se 0 = kmpt :linio [provizu "linio unuan flukslinion_legu 1]
   # Provizu la variablon signo per la valoro de la unua signo de la linio
   provizu "signo unuan :linio
   # poste oni forigu tiun unuan signon de la linio
   provizu "linio senunuan :linio
   provizu "nombro vorton :nombro :signo]
sendu :nombro
fino

test 10
proksimumigo de pi: 3.4641016151377544
test 100
proksimumigo de pi: 3.1108550841912757
test 1000
proksimumigo de pi: 3.081180112566604
test 10000
proksimumigo de pi: 3.1403714651066386
test 70000
```

```
proksimumigo de pi: 3.1361767950325627
```

Oni trovas do proksimumigon de la nombro π per ĝiaj propraj decimaloj!

Ankoraŭ eblas plibonigi tiun programon indikante ekzemple la tempon uzitan por la kalkulo. Aldonu en unua linio de la proceduro test:

```
provizu "komenco tempon
```

Aldonu ĝuste antaŭ `flukson_fermu 1`:

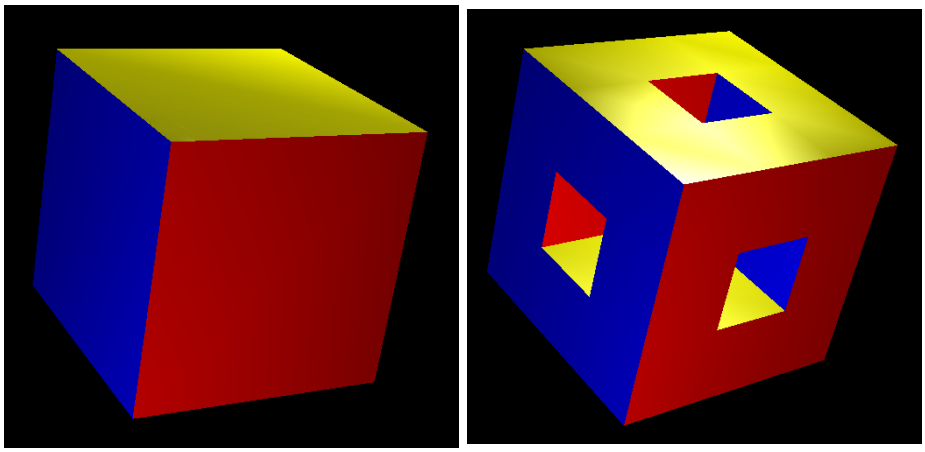
```
skribu frazon [Tempo uzita:  ]  tempon - :komenco
```

Ĉapitro 13

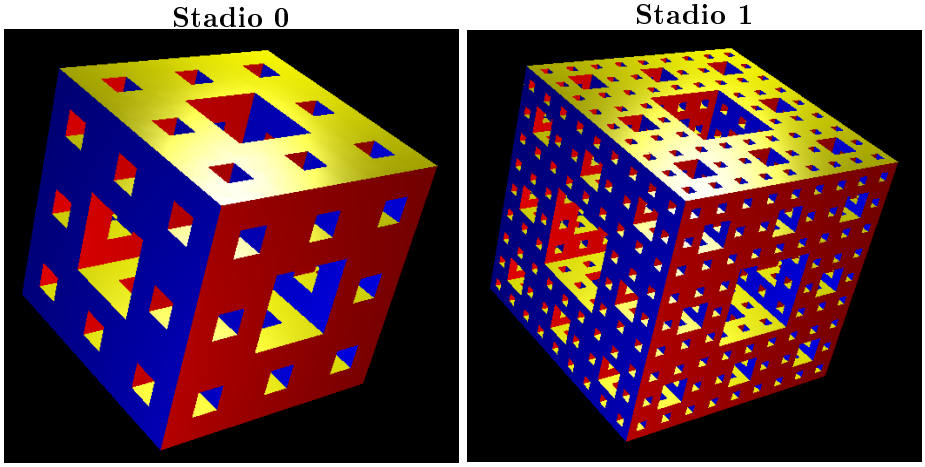
Temo: Spongo de Menger

Nivelo: Alta

En ĉi tiu ĉapitro, ni konstruos solidan fraktalon nomatan spongo de Menger. Jen la unuaj iteracioj por konstrui tiun solidon:



Stadio 0 **Stadio 1**



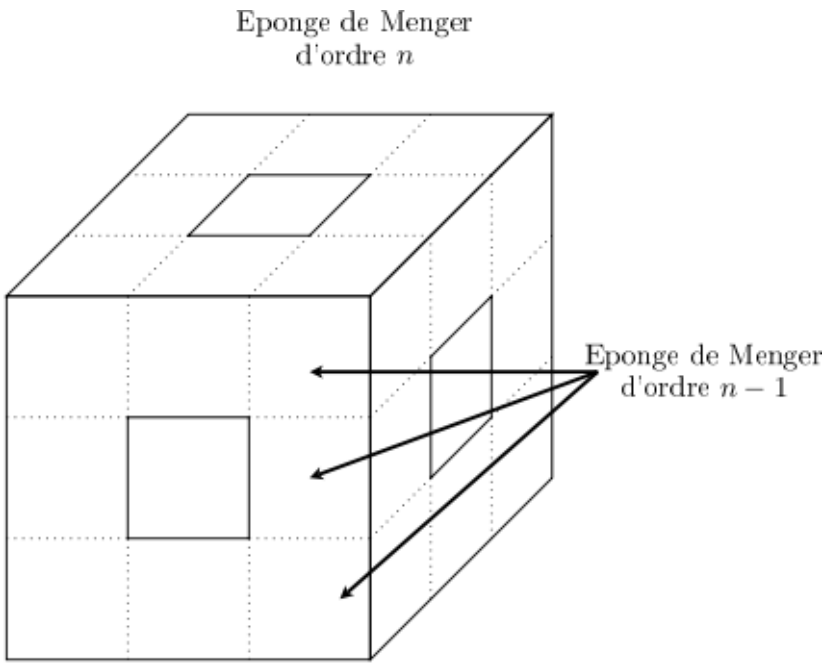
Stadio 2 **Stadio 3**

La ĉapitro konsistas el du partoj:

- Antaŭ ĉio, ni montros kiel krei tiun solidon facile per uzo de rekursiveco.
- Poste, oni provos plibonigi la grafikadon por grafiki spongon de Menger de ordo 4.

13.1 Uzante rekursivecon

Konsideru spongon de Menger de ordo n kies latero mezuras L.



La skemo montras bone ke tiu spongo konsistas efektive el 20 spongoj de Menger de ordo $n-1$ havantaj ĉiuj lateron je $L/3$. La rekursiva strukturo de la spongo evidentiĝas tiel.

La programo:

```
# Ĉefa komando: spongo 3
por kubo :l
se :nombrilo = 10000 [tridimensie_vidigu]
# Koloroj de la flankaj facoj
lokp "koloroj [flavan violruĝan verdbluan bluan]
# flankaj facoj
ripetu 4 [skolp ekzek eron kmpt :koloroj kvadrato :l dn 90 an  :l mdn 90 dkn 90]
# Sube
skolp ruĝan malsupren 90 kvadrato :l supren 90
av :l msn 90 skolp verdan kvadrato :l sn 90 man :l
fino

por kvadrato :c
provizu "nombrilo :nombrilo + 1
por_edro
ripetu 4 [an :c dn 90]
fino_edro
fino

# Spongo de Menger
# p: profundeco de rekursiveco
# l: longeco de la granda kubo
por menger :l :p
se :p=0 [kubo :l]
 [lokp "p :p-1
  lokp "l :l/3
  # antaŭa faco
  ripetu 3 [menger :l :p an :l] man 3*:l
  dn 90 an :l mdn 90
  menger :l :p an 2*:l menger :l :p man 2*:l
  dn 90 an :l mdn 90
```

```
  ripetu 3 [menger :l :p av :l] man 3*:l
  # dekstra flanko
  malsupren 90 an :l supren 90
  menger :l :p an 2*:l menger :l :p re 2*:l
  malsupren 90 an :l supren 90
  ripetu 3 [menger :l :p an :l] man 3*:l
  mdn 90 an :l dn 90
  menger :l :p an 2*:l menger :l :p man 2*:l
  mdn 90 an :l dn 90
  ripetu 3 [menger :l :p an :l] man 3*:l
  malsupren 90 man :l supren 90
  menger :l :p an 2*:l menger :l :p man 2*:l
  malsupren 90 man :l supren 90]
fino

por spongo :p
ev tdk provizu "nombrilo 0 perspektive fkolp 0 menger 800 :p
tajpu [Nombro de kvadratoj: ] s :nombrilo
tridimensie_vidigu
fino
```

Tiu programo konsistas el kvar proceduroj:

- `kvadrato :c`
  Tiu proceduro grafikas kvadraton kun lateroj longaj je `:c`. Krome, tiun kvadraton registras la modelilo 3D. La variablo `nombrilo` responsas pri nombri la nombron de kvadratojn desegnitajn.

- `kubo :l`
  Tiu proceduro grafikas kubon kun lateroj longaj je `:l`. Kompreneble, ĝi uzas la proceduron `kvadrato`.

- `menger :l :p`
  Tiu proceduro estas la ĉefaĵo de l' programo; ĝi desegnas motivon de Menger de ordo p kaj do la latero mezuras l. Tiun motivon oni kreas laŭ rekursiva maniero tute nature sekvante la antaŭan skemon.

- `spongo :p`
  Tiu proceduro grafikas spongon de Menger de ordo p kaj kun latero 800 kaj aldonas ĝin al modelilo 3D.

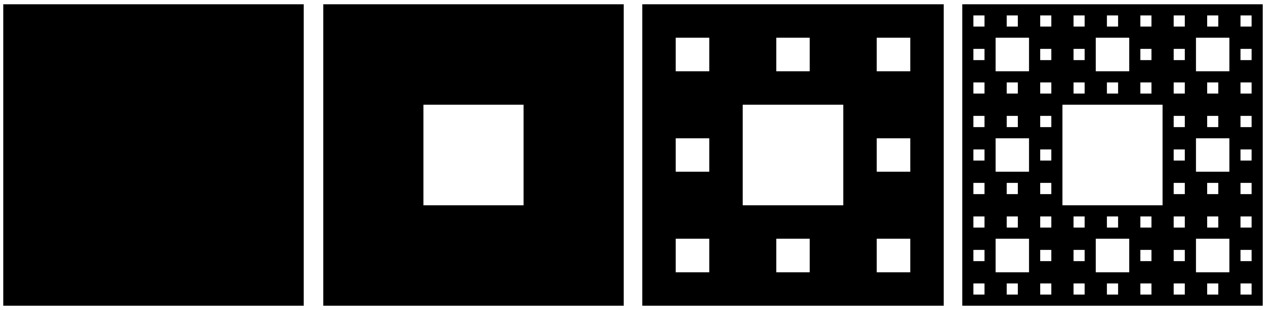
13.2 Dua pritrakto: solida objekto de 4-a ordo

La antaŭa programo havas kiel ĉefan avantaĝon ekspluati la nature rekursivan strukturon de la fraktala solido. Rimarku ke tiu sama metodo povas esti uzata ankaŭ por generi aliajn fraktalajn solidojn aŭ, pli simple, aliajn fraktalajn kurbojn. Ĉiuokaze, la tuja konsekvenco de la rekursiva pritrakto estas mallonga fontokodo kaj simple komprenebla. Bedaŭrinde, rimarku ke spongo je 3-a ordo postulas jam 48 000 kvadratojn. Necesas tiam estigi la memoron dediĉitan al XLogo je 256 MB en la fenestro pri preferoj por ke la programo povu ruliĝi tute.

Se oni deziras grafiki Menger-an spongon je 4-a ordo, baldaŭ oni estos barita de forĉerpado de memoro. En ĉi tiu parto ni vidos programon bazitan sur tute malsama algoritmo; ĝi ebligos krei spongon de Menger je ordo 0, 1, 2, 3 aŭ 4.

13.2.1 La tapiŝo de Sierpinski

La spongo de Menger estas efektive ĝeneraligon en 3 dimensioj de ebena figuno nomata tapiŝo de Sierpinski. Jen la unuaj iteracioj de tiu figuro:



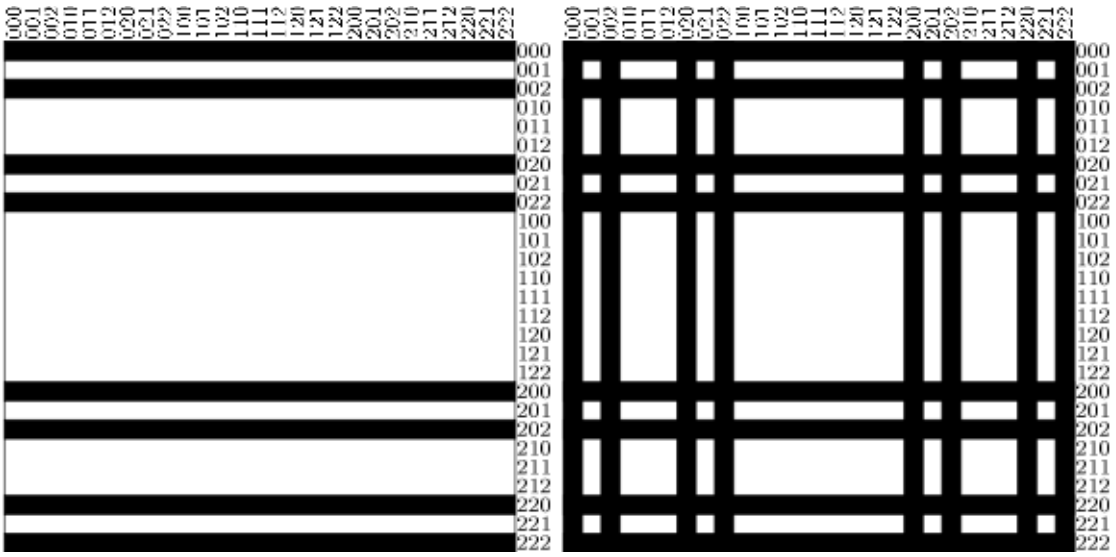
Stadio 0 **Stadio 1** **Stadio 2** **Stadio 3**

La motivo estanta sur ĉiu faco de spongo de Menger je ordo p-a estas tapiŝo de Sierpinski je ordo p-a.

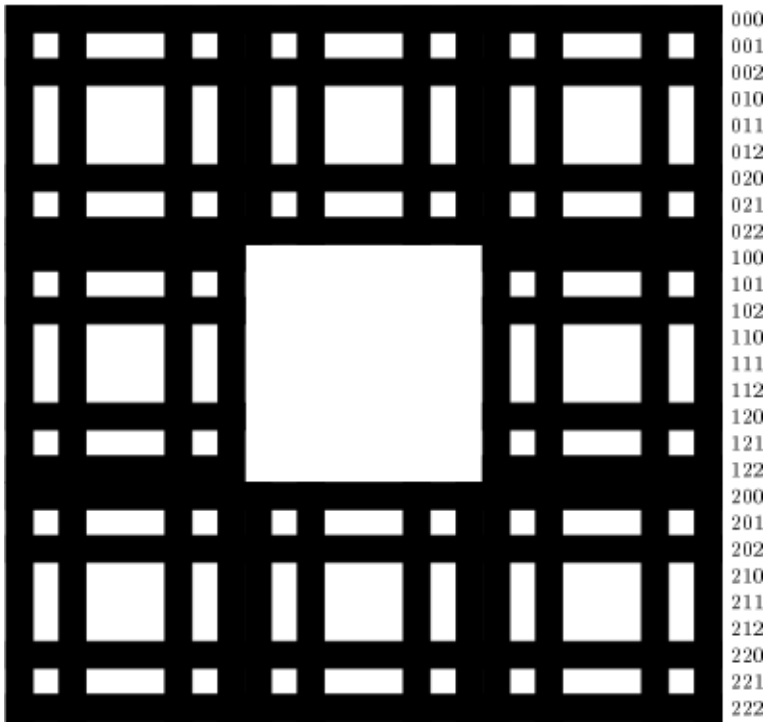
13.2.2 Grafiki tapiŝon de Sierpinski je ordo p-a

La celo estas atingi malpligrandigi la nombron de postulitaj kvarlateroj por desegni tapiŝon de Sierpinski. La jena ekzemplo klarigas la la procedon uzitan por krei tapiŝon de Sierpinski je ordo 3-a. Ĉi tie, la komenca kvadrato konsistas do el $3^3 = 27$ horizontaloj kaj 27 vertikaloj. Oni skribas je bazo 3 la numeron de ĉiu horizontalo kaj ĉiu vertikalo.

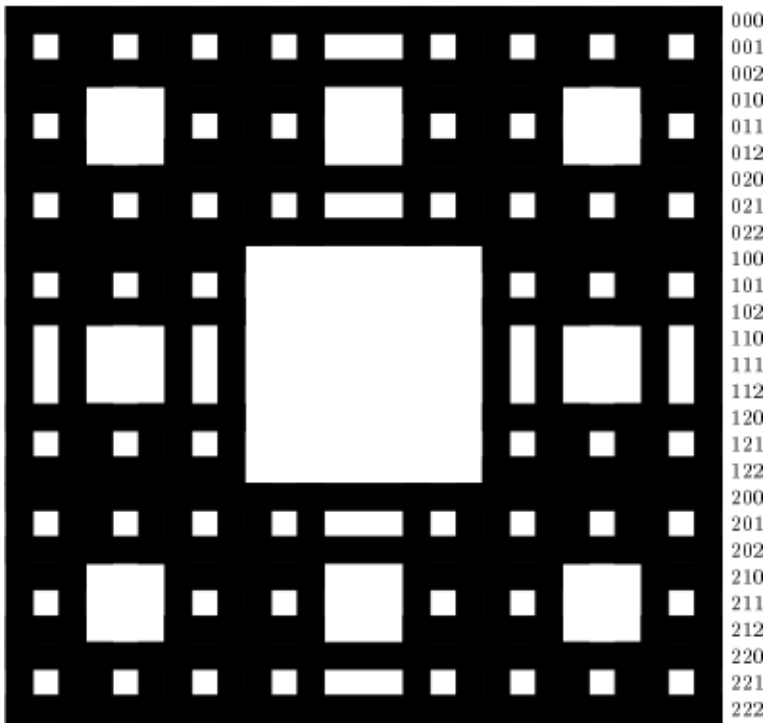
- **Unua stadio:** Por ĉiu horizontalo kies numero konsistas el neniu 1, grafiku horizontalon de 27 ĉeloj. Pro simetrio, efektivigu la saman operacion vertikale.



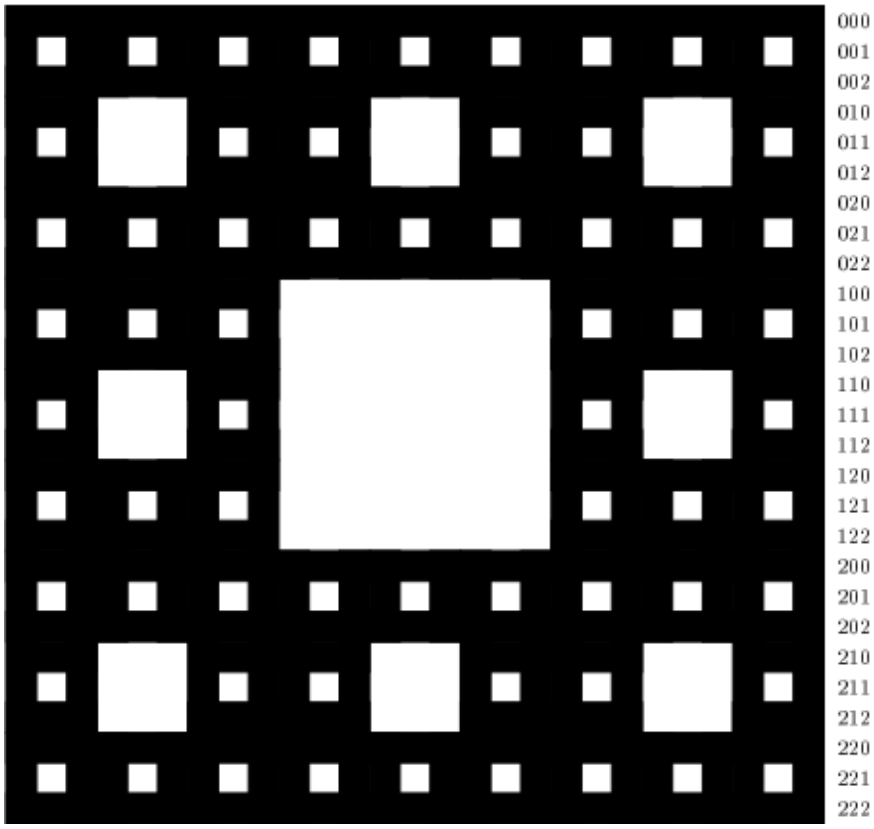
- **Dua stadio:** Nun interesiĝu pri la horizontaloj kies numero konsistas el ununura 1 en l' unua loko. Grafiku sinsekve alterne ortangulojn longaj je 9 ĉeloj. Faru por la vertikaloj simetrie.



- **Tria stadio:** Nun interesiĝu pri la horizontaloj kies numero konsistas el nur unu 1 en la dua loko. Grafiku sinsekve alterne ortangulojn laŭ la skemo [3 3 6 3 6 3 3] (3 ĉeloj plenaj, 3 malplenaj, 6 plenaj, ktp...). Simetrie faru por la vertikaloj.



- **Lasta stadio:** Interesiĝu pri horizontaloj kies numero konsistas el du 1 lokitaj en l' unuaj lokoj. Grafiku sinsekve alterne ortangulojn laŭ la skemo [3 3 3 9 3 3 3]. Operaciu same por la vertikaloj.



Tiam finiĝas la konstruado de la tapiŝo de Sierpinski je ordo 3-a. Por krei tiun tapiŝon necesis uzi entute: $16 + 16 + 32 + 16 = 80$ ortangulojn.

13.2.3 Malsamaj skemoj de vertikaloj eblaj

Por resumi la antaŭan konstruadon, jen la malsamaj tipoj de skemaj de vertikaloj laŭ ilia numero. (La simbolo * indikas la ciferon 0 aŭ la ciferon 2.

Numero de la tipo	Skemo aplikenda
***	27
1**	9 9 9
1	3 3 6 3 6 3 3
11*	3 3 3 9 3 3 3

Sur la sama principo, por krei tapiŝon je ordo 4-a, oni uzu kvadraton kun $3^4 = 81$ ĉeloj. La numeroj de horizontaloj kaj vertikaloj havos do 4 ciferojn en ilia prezentado je bazo 3. Por ĉiu tipo de numero, jen la skemo aplikenda (la simbolo * indikas la ciferon 0 aŭ la ciferon 2):

Numero de tipo	Skemo aplikenda
****	81
1***	27 27 27
*1**	9 9 18 9 18 9 9
**1*	3 3 6 3 6 3 6 3 6 3 6 3 6 3 6 3 3
11	3 3 3 9 3 3 6 3 3 9 3 3 6 3 3 9 3 3 3
1*1*	3 3 6 3 6 3 3 27 3 3 6 3 6 3 3
11**	9 9 9 27 9 9 9
111*	3 3 3 9 3 3 3 27 3 3 3 9 3 3 3

496 kvarlateroj estas do necesaj por grafiki tapiŝon de Sierpinski je ordo 4.

Finfine, jen la konstruskemoj por solidoj je ordo 2:

Numero de tipo	Skem' aplikenda
**	9
1*	3 3 3

13.2.4 La programo

```
# Grafikas tapi^son de Sierpinski je ordo :p kaj je amplekso :amplekso
por tapiŝo :amplekso :p
provizu "unuo :amplekso / (potencon 3 :p)
se :p=0 [ort :amplekso :amplekso haltu]
se :p=1 [ripetu 4 [ort :amplekso :unuo an :amplekso dn 90] haltu]
ripetupor (list "x 1 potencon 3 :p)
  [lokp "cantorx cantor :x :p []
  # Ne grafiku la erojn havantajn unu 1 en la lasta loko
  se  ne (1 = lastan :cantorx)
    [lokp "nom valorigu senlastan :cantorx "
     grafiku_vertikalon :x econ_sendu "map :nom]]
fino

# Donas la nombron x je bazo 3
# p profundeca indekso 3^p
# :list listo malplena ĉe l' komenco

por cantor :x :p :list
se :p=0 [sendu :list]
lokp "a potencon 3 :p-1
se :x <= :a
  [sendu cantor :x :p-1 frazon :list 0]
  [se :x <= 2*:a [sendu cantor  :x-:a :p-1 frazon :list 1]
   sendu cantor :x - 2*:a :p-1 frazon :list 0]
fino

# Grafiku la x-an vertikalon laŭ la konstruskemo difinita en la listo
por grafiku_vertikalon :x :list
  l  dn 90 an (:x-1)*:unuo mdn 90 ml des :list
  l mdn 90 an (:x-1)*:unuo mdn 90 an :x*:unuo dn 90 ml des :list
l mdn 90 man :x*:unuo ml
fino

# Grafiku ortangulon laŭ donitaj dimensioj
# Ĝin registras la 3d-vidilo
por ort :lo :la
provizu "nombrilo :nombrilo + 1
por_edro
ripetu 2 [an :lo dn 90 an :la dn 90]
fino_edro
fino

# Pretigu la malsamajn eblajn vertikalojn por la tapiŝoj je ordo 1 al 4
por pretmap
econ_provizu "map 111 [3 3 3 9 3 3 3 27 3 3 3 9 3 3 3]
econ_provizu "map 110 [9 9 9 27 9 9 9]
econ_provizu "map 101 [3 3 6 3 6 3 3 27 3 3 6 3 6 3 3]
econ_provizu "map 011 [3 3 3 9 3 3 6 3 3 9 3 3 6 3 3 9 3 3 3]
```

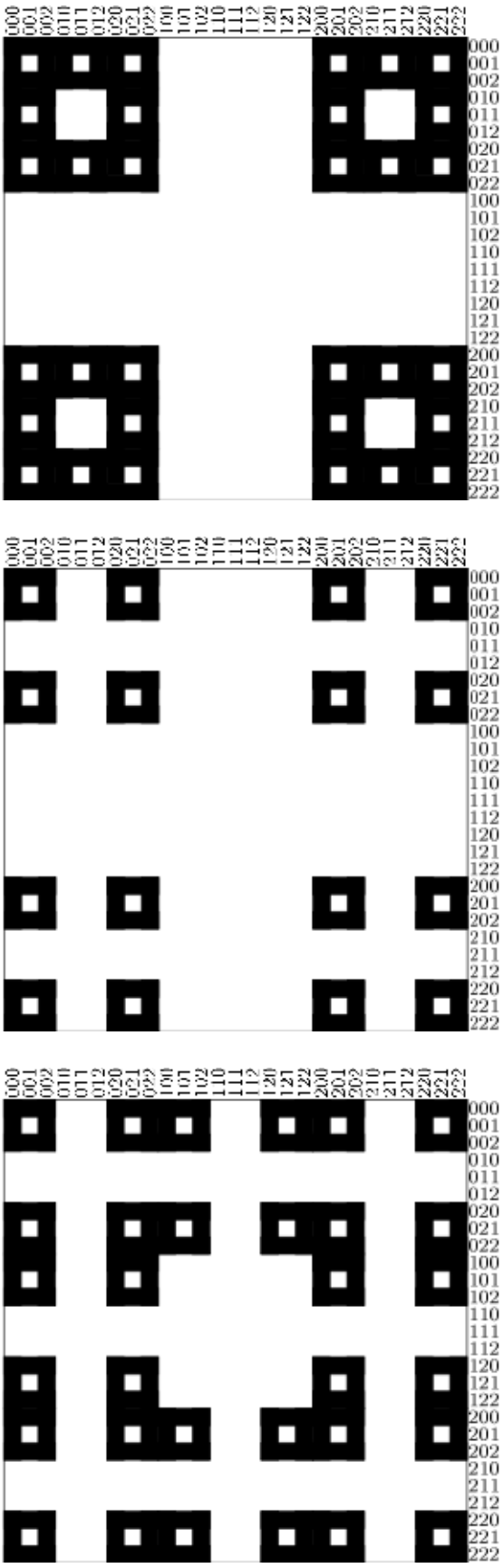
```
econ_provizu "map 000 [81]
econ_provizu "map 100 [27 27 27]
econ_provizu "map 010 [9 9 18 9 18 9 9]
econ_provizu "map 001 [3 3 6 3 6 3 6 3 6 3 6 3 6 3 6 3 3]
econ_provizu "map  01 [3 3 6 3 6 3 3]
econ_provizu "map  00 [27]
econ_provizu "map  10 [9 9 9]
econ_provizu "map  11 [3 3 3 9 3 3 3]
econ_provizu "map   1 [3 3 3]
econ_provizu "map   0 [9]
fino

# Se la prezento estas [1 0 1] --> sendu 101
por valorigu :list :vort
  se malplena? :list [sendu :vort]
  [lokp "vort vort :vort unuan :list
   sendu valorigu senunuan :list :vort]
fino

# Desegnu la ortangulojn de ĉiu vertikalo alterne
por des :list
lokp "sumo 0
ripetupor (list "i 1 kmpt :list)
   [lokp "ero eron :i :list
    lokp "sumo :ero + :sumo
    se para? :i [l an :ero*:unuo ml] [ort :ero*:unuo :unuo an :ero*:unuo]]
l man :sumo * :unuo ml
fino

# Testu ĉu nombro estas para
por para? :i
sendu 0 = reston :i 2
fino

por siertapiŝo :p
ev perspektive tdk pretmap
provizu "nombrilo 0
tapiŝo 810 :p
tajpu "Nombro\ de\ kvarlateroj:\  s :nombrilo
vue3d
fin
```

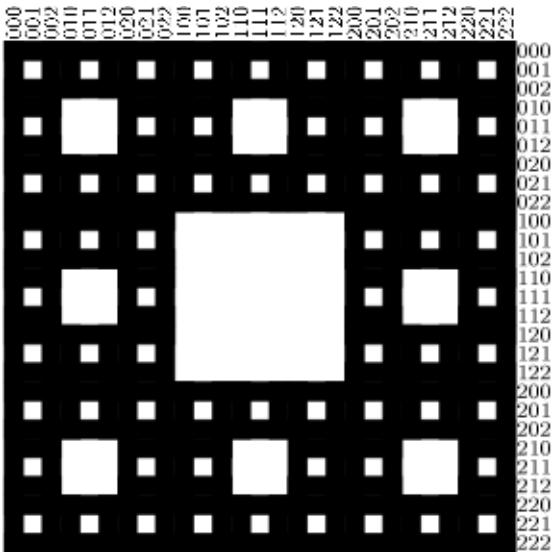
`siertapiŝo 3` desegnas tapiŝon de Sierpinski je ordo 3 kaj latero 810. Jen, ni pretas pasi al la spongo de Menger!

13.2.5 La spongo de Menger je ordo 4

La spongo de Menger havas plurajn simetriecajn atributojn. Por generi ĝin ni grafikos la diversajn sekciojn laŭ la ebeno (xOy), poste portos tiujn figurojn laŭ (yOz) kaj (xOz). Por bone klarigi tion kio okazas, ni restu sur l' ekzemplo de la spongo je ordo 3:

Kiam oni tranĉas la spongon laŭ vertikala ebeno, oni povas akiri kvar malsamajn motivojn:


000 001 002 010 011 012 020 021 022 100 101 102 110 111 112 120 121 122 200 201 202 210 211 212 220 221 222



Por grafiki spongon je ordo 3, ni trairos la nombrojn de 1 ĝis 27, tio estas, de 001 ĝis 222 je bazo 3. Por ĉiu numero, oni aplikos la taŭgan sekcion kiun oni portos laŭ la 3 direktoj (Ox), (Oy) kaj (Oz).

La kodo

La jena programo permesas grafiki la solidojn de Menger je ordoj 0, 1, 2, 3, 4. La nombro de proceduroj estas grava, do mi klarigos tuj.

```
# Grafiki tapiŝon de Sierpinski je ordo :p kaj je amplekso :amplekso
por  tapiŝo :amplekso :p
provizu "unuo :amplekso / (potencon 3 :p)
se :p=0 [ort :amplekso :amplekso haltu]
se :p=1 [ripetu 4 [ort :amplekso :unuo an :amplekso dn 90] haltu]
ripetupor (list "x 1 potencon 3 :p)
  [lokp "cantorx cantor :x :p []
   # Ne grafiku erojn havantajn unu 1 en la lasta loko
   se  ne (1 = lastan :cantorx)
     [lokp "nom valorigu senlastan :cantorx "
      grafikuvertikalon :x econ_sendu "map :nom]]
fino

# Sendu la prezenton je bazo 3 de la nombro x
# p profundeca indekso 3^p
# :list listo malplena ĉe l' komenco

por cantor :x :p :list
se :p=0 [sendu :list]
lokp "a potencon 3 :p-1
se :x <= :a
  [sendu cantor :x :p-1 frazon :list 0]
  [se :x <= 2*:a [sendu cantor :x-:a :p-1 frazon :list 1]
   sendu cantor :x-2*:a :p-1 frazon :list 2]
fino

# Grafiku la numeron x laŭ la konstruskemo difinita en la listo
por grafikuvertikalon :x :list
  l  dn 90  an (:x-1)*:unuo mdn 90 ml des :list
  l mdn 90  an (:x-1)*:unuo  dn 90 an :x*:unuo dn 90 ml des :list
```

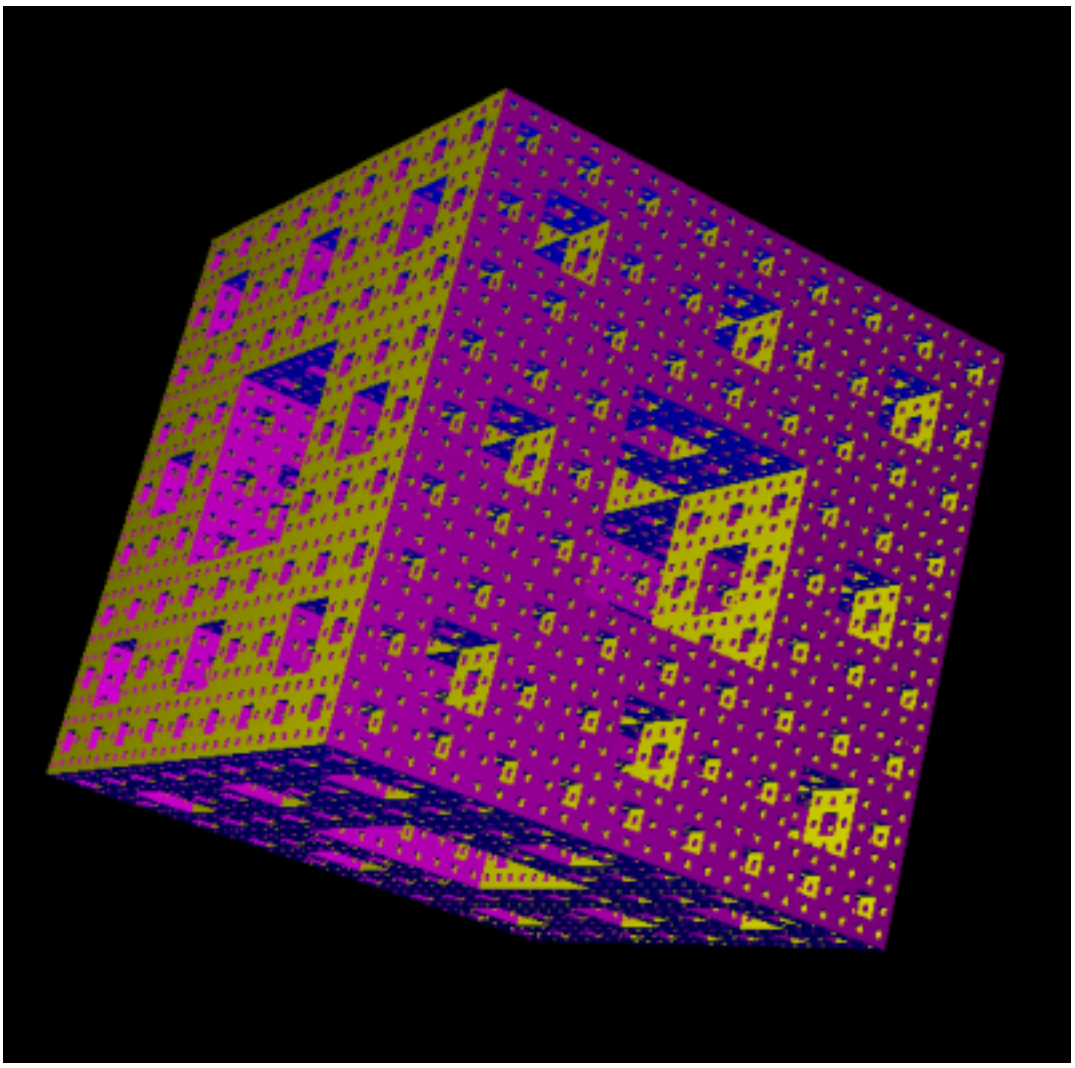
```
  l mdn 90 man :x*:unuo ml
fino

# Grafiku ortangulon laŭ donitaj dimensiojn
# La plurlatero estas registrita de la 3d-vidigilo
por ort :lo :la
  provizu "nombrilo :nombrilo+1
  por_edro
    ripetu 2 [an :lo dn 90 an :la dn 90]
  fino_edro
fino

# Komencu la malsamajn vertikalojn eblajn por la tapiŝojn je ordo 1 ĝis 4
por pretmap
econ_sendu "map 111 [3 3 3 9 3 3 3 27 3 3 3 9 3 3 3]
econ_sendu "map 110 [9 9 9 27 9 9 9]
econ_sendu "map 101 [3 3 6 3 6 3 3 27 3 3 6 3 6 3 3]
econ_sendu "map 011 [3 3 3 9 3 3 6 3 3 9 3 3 6 3 3 9 3 3 3]
econ_sendu "map 000 [81]
econ_sendu "map 100 [27 27 27]
econ_sendu "map 010 [9 9 18 9 18 9 9]
econ_sendu "map 001 [3 3 6 3 6 3 6 3 6 3 6 3 6 3 6 3 6 3 3]
econ_sendu "map  01 [3 3 6 3 6 3 3]
econ_sendu "map  00 [27]
econ_sendu "map  10 [9 9 9]
econ_sendu "map  11 [3 3 3 9 3 3 3]
econ_sendu "map   1 [3 3 3]
econ_sendu "map   0 [9]
fino

# Se la prezento estas [1 0 1] --> sendu 101
# Se la prezento estas [1 0 2] --> sendu 100
# La eroj de la listo estas kunmetataj en vorton.
# Krome, la 2 estas anstataŭataj de nuloj
por valorigu :list :vort
  se malplena? :list [sendu :vort]
    [lokp "unua unuan :list
     se :unua=2 [lokp "unua 0]
     lokp "vort vort :vort :unua
     sendu valorigu senunuan :list :vort]
fino

# Desegnu la ortangulojn de ĉiu vertikalo alterne
por des :list
lokp "sumo 0
ripetupor (liston "i 1 kmpt :list)
   [lokp "ero eron :i :list
    lokp "sumo :ero+:sumo
    se para? :i [l an :ero*:unuo ml] [ort :ort*:unuo :unuo an :ero*:unuo]]
l man :sumo * :unuo ml
fino

# Testu ĉu nombro estas para
```

```
por para? :i
  sendu 0 = resto :i 2
fino

por siertapiŝo :p
  ev perspektive tdk pretmap
  provizu "nombrilo 0
  tapiŝo 810 :p
  tajpu "Nombro\ de\ plurlateroj:\  s :nombrilo
  tridimensie_vidigu
fino

# Forigas la lastan 1 en la listo :list
por forigulastanunu :list
  ripetupor (list "i kmpt :list 1 minusigan 1)
    [lokp "ero eron :i :list
     se :ero=1 [lokp "list anstataŭigu :list :i 0 haltu] [se :ero=2 [haltu]]]
  sendu :list
fino

# Spongo de Menger je amplekso donita kaj je profundeco :p

por menger :amplekso :p
provizu "unuo :amplekso / (potencon 3 :p)
ripetupor (list "z 1 potencon 3 :p)
  [lokp "cantorz cantor :z :p []
   lokp "last lastan :cantorz
   lokp "cantorz senlantan :cantorz
   se :last=0 [lokp "ordo valorigu forigulastanunu :cantorz "] [lokp "ordo valorigu :cantorz "]
   lokp "ordo vort "trançi :ordo
   graf3tapiŝon :amplekso :ordo :z
   l supren 90 an :unuo malsupren 90 ml]
graf3tapiŝon :amplekso :ordo (potencon 3 :p) + 1
fino

# Grafiku la tapiŝojn de Sierpinski je ordo :p
# laŭ ĉiu akso (Ox), (Oy) et (Oz)
# je la alto :z
pour draw3carpet :size :order :z
  l originen
  supren 90 an (:z-1)*:unuo malsupren 90 ml
  skolp bluan ekzek :ordo :amplekso
  l originen
  mdfn 90 an (:z-1)*:unuo malsupren 90 ml
  skolp flavan ekzek :ordo :amplekso
  l originen
  supren 90 an :amplekso dn 90 an (:z-1)*:unuo malsupren 90 ml
  skolp violruĝan ekzek :ordo :amplekso
fino

# Ĉefa proceduro
# Grafiku spongon de Menger je profundeco :p
por spongo :p
```

```
  ev perspektive tdk
  lokp "tempo tempon
  pretmap
  provizu "nombrilo 0
  se :p=0 [kubo 405] [menger 405 :p]
  # Skribu la tempon kaj la nombron de plurlateroj necesaj por konstrui
  tajpu "Nombro\ de\ plurlateroj:\  s :nombrilo
  tajpu "Tempo\ uzita:\  s tempon - :tempo
  tridimensie_vidigu
fino

# Sekcio por la Menger je ordo 2

por tranĉi1 :amplekso
  ripetu 4 [tapiŝu :amplekso/3 1 l an :amplekso dn 90 ml]
fino

por tranĉi0 :amplekso
  tapiŝo :amplekso 2
fino

# Sekcio por la Menger je ordo 3

por tranĉi10 :amplekso
  ripetu 4 [tapiŝo :amplekso/3 2 l an :amplekso dn 90 ml]
fino

por tranĉi01 :amplekso
  ripetu 4 [ripetu 2 [tranĉi1 :amplekso/3 l an :amplekso/3 ml] an :amplekso/3 dn 90]
fino

por tranĉi11 :amplekso
  ripetu 4 [tranĉi1 :amplekso/3 l an :amplekso dn 90 ml]
fino

por tranĉi00 :amplekso
  tapiŝo :amplekso 3
fino

# Sekcio por la Menger je ordo 4

por tranĉi000 :amplekso
  tapiŝo :amplekso 4
fino

por tranĉi100 :amplekso
  ripetu 4 [tapiŝo :amplekso/3 3 l an :amplekso dn 90 ml]
fino

por tranĉi010 :amplekso
  ripetu 4 [ripetu 2 [tranĉi10 :amplekso/3 l an :amplekso/3 ml] an :amplekso/3 dn 90]
fino
```

```
por tranĉi001 :amplekso
  ripetu 4 [ripetu 2 [tranĉi01 :amplekso/3 l an :amplekso/3 ml] an :amplekso/3 dn 90]
fino

por tranĉi110 :amplekso
  ripetu 4 [tranĉi10 :amplekso/3 l an :amplekso ml dn 90]
fino

por tranĉi111 :amplekso
  ripetu 4 [tranĉi11 :amplekso/3 l an :amplekso dn 90 ml]
fino

por tranĉi101 :amplekso
  ripetu 4 [tranĉi01 :amplekso/3 l an :amplekso dn 90 ml]
fino

por tranĉi011 :amplekso
  ripetu 4 [ripetu 2 [tranĉi11 :amplekso/3 l an :amplekso/3 ml] an :amplekso/3 dn 90]
fino

por tranĉi :amplekso
  tapiŝo :amplekso 1
fino

por kubo :amplekso
  ripetu 2
    [skolp bluan  ort :amplekso :amplekso l an :amplekso malsupren 90 ml
     skolp flavan ort :amplekso :amplekso l an :amplekso malsupren 90 ml]
  skolp violruĝan
  l mdfn 90 mdn 90 an :amplekso dn 90 ml ort :amplekso :amplekso
  l dn 90 an :amplekso mdn 90 dfn 90 dn 90 an :amplekso mdn 90 dfn 90 ml ort :amplekso :amplekso
  mdfn 90 mdn 90 an :amplekso dn 90
fino

por kuboj
  ev perspektive tdk
  lokp "tempo tempon
  pretmap
  provizu "nombrilo 0
  ripetu 4 [se komputu = 1 [kubo 405] [menger 405 komputu-1] l an 1000 dn 90 ml]
  # Montru la tempon uzitan kaj la nombron de plurlateroj necesaj por konstrui
  tajpu "Nombro\ de\ plurlateroj:\  s :nombrilo
  tajpu "Tempo\ uzita:\  s tempon - :tempo
  tridimensie_vidigu
fino
```

Nun, establu la memoron rezervitan por XLogo je 640 MiB: `spongo 4`



Ĉapitro 14

Temo: Sistemo de Lindenmayer

Nivelo: Alta

Por ĉi tiu parto, mi indiku kelkajn referaĵojn:

- de la retpaĝo Vikipedio pri la L-sistemoj: `http://eo.wikipedia.org/wiki/L-Sistemo`.
- de la libro «The Algorithmic Beauty of Plants» verkita de Przemyslaw Prusinkiewicz kaj Aristid Lindenmayer.

Temos pri la nocio de sistemo de Lindenmayer aŭ L-sistemo inventita en 1968 de la biologo hungaro Aristid Lindenmayer. L-Sistemo estas aro da reguloj kaj simboloj kiuj modelas procezon de kreskado de vivuloj kiel plantoj aŭ ĉeloj. La ĉefa koncepto de la L-sistemoj estas la nocio de reskribado. Reskribado estas teĥniko por konstrui malsimplajn objektojn per anstataŭigi partojn de komenca simpla objekto uzante regulojn de reskribado.

Por fari tion, la ĉeloj estas modelitaj per simboloj. Je ĉiu generacio, la ĉeloj dispartiĝas, tio estas, simbolon anstataŭas unu aŭ pluraj aliaj simboloj formantaj vorton.

14.1 Formala difino

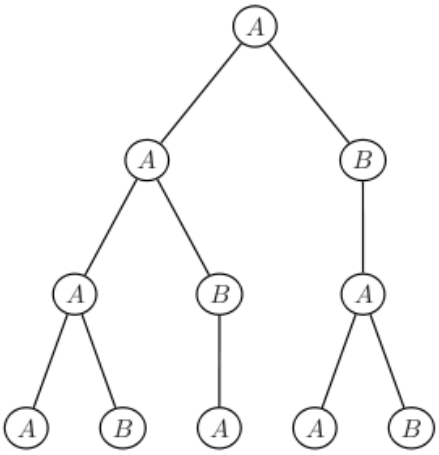
L-sistemo estas formala gramatiko konsistanta el:

1. Unu alfabeto V: l' aro de la variabloj de la L-Sistemo. $V*$ estas la aro de la "vortoj" kiujn oni povas konstrui per la simboloj de V, kaj $V+$ l' aro de la vortoj enhavantaj almenaŭ unu simbolon.
2. Unu aro de valoroj konstantaj S. Kelkaj el tiuj simboloj estas komunaj al ĉiuj L-Sistemoj (ĉefe kiam uzi la testudon).
3. Unu komenca aksiomo ω elektita inter $V+$, tio estas la komenca stato.
4. Unu ekzemplo de reguloj, indikata P, pri reproduktado de la simboloj de V.

L-sistemo estas do indikata $\{V, S, \omega, P\}$.
Konsideru la jenan L-sistemon:

- Alfabeto: $V = \{A, B\}$
- Konstantoj: $S = \{\emptyset\}$
- Komenca aksiomo: $\omega = A$
- Reguloj: $\boxed{\begin{array}{l} A \to AB \\ B \to A \end{array}}$

La du reguloj donitaj estas la reguloj de reskribado de la sistemo. Je ĉiu stadio, A estas anstataŭita de la sinsekvo AB, kaj B estas anstataŭita de A. Jen la unuaj iteracioj de tiu sistemo de Lindemayer:



- Iteracio 1: A
- Iteracio 2: AB
- Iteracio 3: ABA
- Iteracio 4: $ABAAB$

Bone, bone... sed konkrete? Daŭrigu legi!

14.2 Interpretado de la testudo

Tiu unua ekzemplo ebligis vin kompreni la nocion de sistemo de Lindenmayer, eble sen ekkonscii kiel ni uzos tion konkrete kun la testudo.

Ja tie tio estiĝas interesa: Ĉiu vorto tiel konstruita nur havas propran signifon. Oni tiam kroĉos al ĉiu litero de la sinsekvo, komandon rulotan de la testudo, por tiel generi desegnojn 2D aŭ 3D.

14.2.1 Oftaj simboloj

- F : Moviĝi je unu unueca paŝo ($\in V$)
- $+$: Turniĝi maldekstren je angulo α ($\in S$).
- $-$: Turniĝi dekstren je angulo α ($\in S$).
- & : Pivoti al malsupro je angulo α ($\in S$).
- ^: Pivoti al supro je angulo α ($\in S$).
- \: Ruliĝi maldekstren je angulo α ($\in S$).
- /: Ruliĝi dekstren je angulo α ($\in S$).
- |: Efektivigi duonturniĝon. En XLogo: `dn 180`

Ni prenu por ekzemplo $\alpha = 90$ kaj unuecan moviĝon je 10 testudpaŝojn; jen:

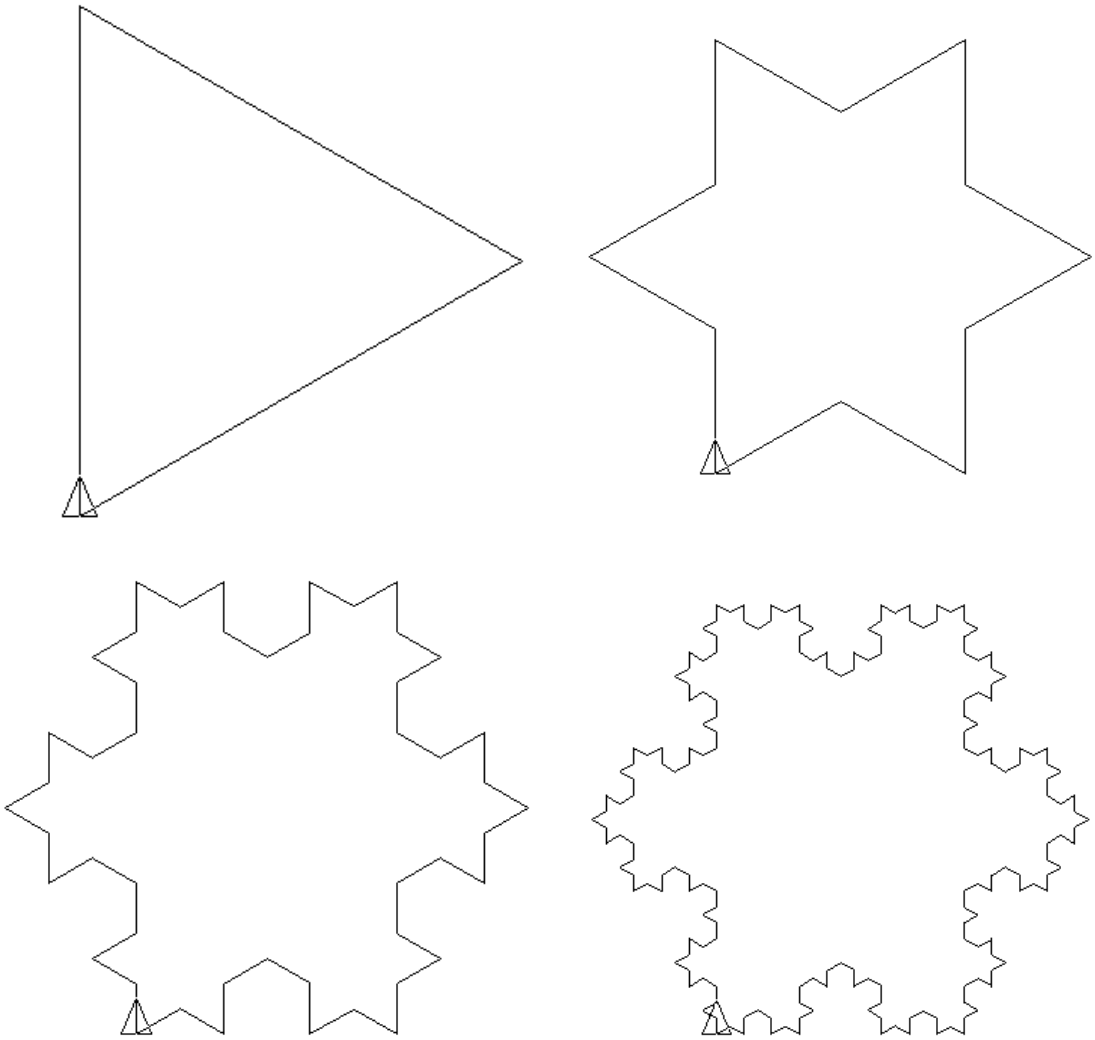
Simbolo	F	$+$	$-$	&	^	\	/	|
Komando XLogo	`an 10`	`mdn 90`	`dn 90`	`malsupren 90`	`supren 90`	`mdfn 90`	`dfn 90`	`dn 180`

14.2.2 Neĝero de Koch

Konsideru la L-sistemon:

- Komenca stato: $F--F--F--$
- Produkta regulo: $F \to F+F--F+F$
- Angulo $\alpha = 60°$, la unuecan paŝon oni dividu per 3 je ĉiu iteracio.

Unuaj iteracioj:



Programo en Logo:

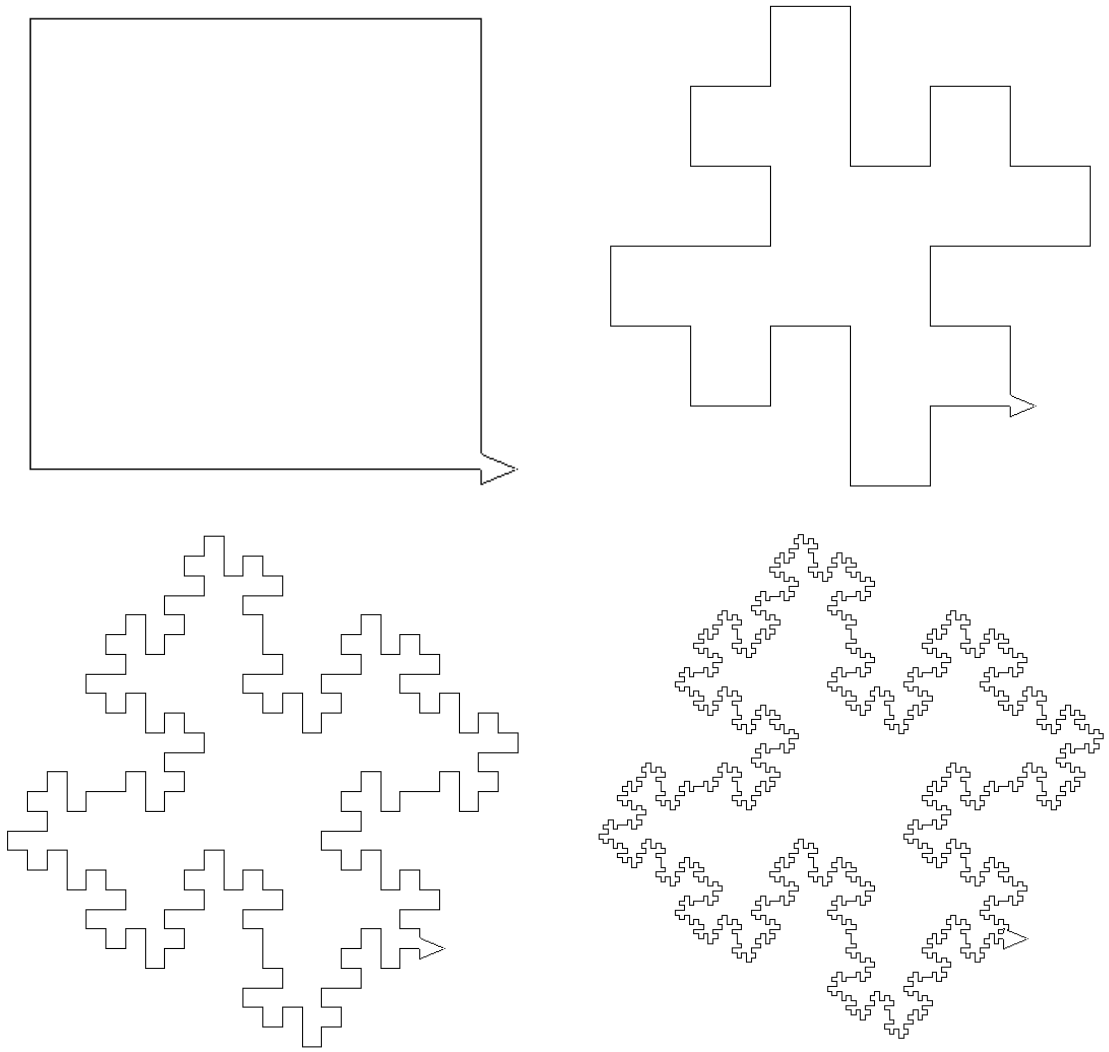
```
por neĝero :p
provizu "unuo 300 / potencon 3 :p-1
ripetu 3 [F :p-1 td 120]
fino

por F :p
se :p=0 [an :unuo haltu]
F :p-1 mdn 60 F :p-1 dn 120 F :p-1 md 60
F :p-1
fino
```

14.2.3 Kurbo de Koch je ordo 2

Ni interesiĝu pri la jena L-sistemo:

- Komenca stato: $F - F - F - F$
- Produkta regulo: $F \to F - F + F + FF - F - F + F$

Jen la unuaj reprezentoj uzante $\alpha = 90$ kaj alĝustigante la unuecan paŝon tiel ke la figuro havu ĉiam la saman amplekson:



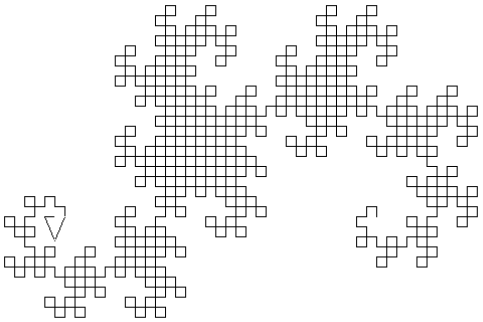
Tre facilas do krei la programon Logo ebligantan generi tiujn desegnojn:

```
# p indikas l' iteracion
por koch :p
  # Je ĉiu iteracio, la unueca distanco dividatas per 4
  # Ĉi tie, la fina figuro havos amplekson 600x600 maksimume
  provizu "unuo 300 / potencon 4 :p-1
  ripetu 3 [F :p-1 tg 90] F :p-1
fino

# La ĉeno reskribada
por F :p
  se :p=0 [an :unuo haltu]
  F :p-1 mdn 90 F :p-1 dn 90 F :p-1 dn 90
  F :p-1 F :p-1 mdn 90 F :p-1 mdn 90 F :p-1 dn 90 F :p-1
fino
```

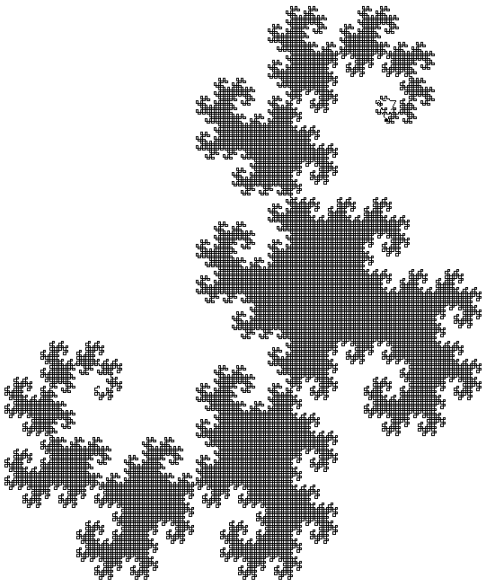
14.2.4 Kurbo de l' dragono

- Komenca stato: F
- Produkta regulo: $\boxed{\begin{array}{l} A \to A + B+ \\ B \to -A - B \end{array}}$

```
por a :p
se :p=0 [an :unuo haltu]
a :p-1 mdn 90 b :p-1 mdn 90
fino

por b :p
se :p=0 [an :unuo haltu]
dn 90 a :p-1 dn 90 b :p-1
fino

por dragono :p
provizu "unuo 300 / 8 / :p
a :p
fino
```



dragono 10



dragono 15

14.2.5 Kurbo de Hilbert en 3D

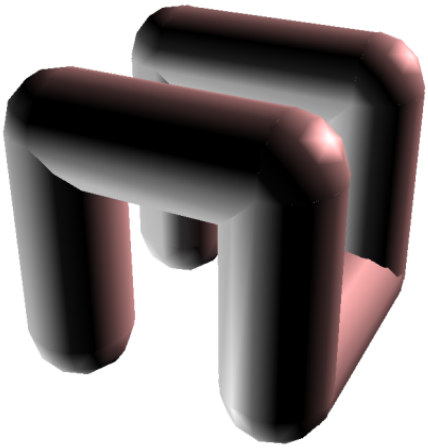
La sekva ekzemplo estas la kurbo de Hilbert en la spaco; ĝi estas kurbo kun la atributo plenigi tute kubon kiam oni pligrandigas la nombron de iteracioj .

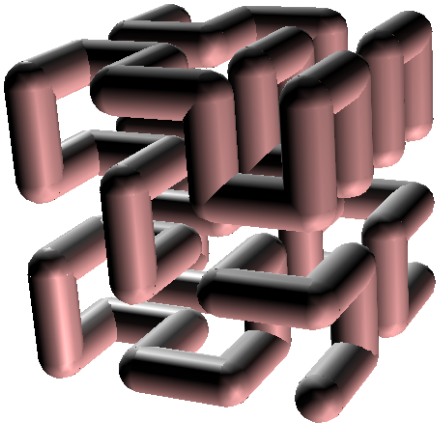
Jen la rilata sistemo:

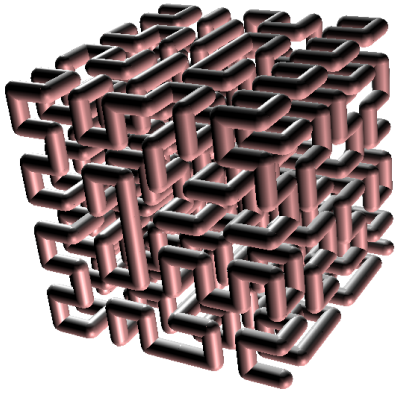
- Komenca stato: A
- Angulo $\alpha = 90°$, dividu la unuecan longon per du je ĉiu iteracio.
- Produkta regulo: $\boxed{\begin{array}{l} A \to B - F + CFC + F - D\&F\hat{}D - F + \&\&CFC + F + B// \\ B \to A\&F\hat{}CFB\hat{}F\hat{}D\hat{}\hat{} - F - D\hat{}|F\hat{}B|FC\hat{}F\hat{}A// \\ C \to |D\hat{}|F\hat{}B - F + C\hat{}F\hat{}A\&\&FA\&F\hat{}C + F + B\hat{}F\hat{}D// \\ D \to |CFB - F + B|FA\&F\hat{}A\&\&FB - F + B|FC// \end{array}}$

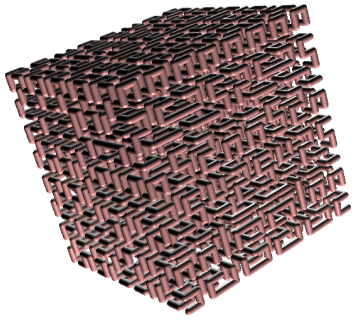
```
por hilbert :p
ev perspektive
provizu "unuo 400 / potencon 2 :p
linia_difino sdikp :unuo/2
a :p
linia_difinhalto
tridimensie_vidigu
fino

por a :p
se :p=0 [haltu]
b :p-1 dn 90 an :unuo mdn 90 c :p-1 an :unuo c :p-1
mdn 90 an :unuo dn 90 d :p-1 malsupren 90 an :unuo supren 90 d :p-1
dn 90 an :unuo mdn 90 malsupren 180 c :p-1 an :unuo c :p-1
mdn 90 an :unuo mdn 90 b :p-1 dfn 180
fino

por b :p
se :p=0 [haltu]
a :p-1 malsupren 90 an :unuo supren 90 c :p-1 an :unuo b :p-1 supren 90
an :unuo supren 90 d :p-1 supren 180 dn 90 an :unuo dn 90 d :p-1 supren 90
dn 180 an :unuo supren 90 b :p-1 dn 180 an :unuo c :p-1 supren 90 an :unuo
supren 90 a :p-1 dfn 180
fino

por c :p
se :p=0 [haltu]
dn 180 d :p-1 supren 90 dn 180 an :unuo supren 90 b :p-1 dn 90 an :unuo mdn 90
c :p-1 supren 90 an :unuo supren 90 a :p-1 malsupren 180 an :unuo a :p-1 malsupren 90
an :unuo supren 90 c :p-1 mdn 90 an :unuo mdn 90 b :p-1 supren 90 an :unuo supren 90
d :p-1 dfn 180
fino

por d :p
se :p=0 [haltu]
dn 180 c :p-1 an :unuo b :p-1 dn 90 an :unuo mdn 90 b :p-1 dn 180
an :unuo a :p-1 malsupren 90 an :unuo supren 90 a :p-1 malsupren 180 an :unuo
b :p-1 dn 90 an :unuo mdn 90 b :p-1 dn 180 an :unuo c :p-1 dfn 180
fino
```

Jen l' unuaj iteracioj:









Apendico A

Listo de la primitivoj

Kiel dirite antaŭe, oni kontrolas la testudon per internaj komandoj nomataj prakomandoj aŭ «primitivoj». Jen klasado de tiuj primitivoj:

A.1 Movi la testudon, administri la krajonon kaj la kolorojn

A.1.1 Movi

Primitivoj por movi la testudon.

antaŭen, an, antauen, antawen, antauxen *n*
Movas la testudon antaŭen je *n* paŝoj laŭ la nuna direkto.

malantaŭen, man, malantauen, malantawen, malantauxen *n*
Movas la testudon malantaŭen je *n* paŝoj laŭ la nuna direkto.

dekstren, dn *n*
Turnas la testudon je *n* gradoj dekstren rilate al la nuna direkto.

maldekstren, mdn *n*
Turnas la testudon je *n* gradoj maldekstren rilate al la nuna direkto.

rondon_desegnu, rond *R*
Desegnas cirklon kun radiuso R ĉirkaŭ la testudo.

arkon_desegnu, ark *R angulo1 angulo2*
Grafiki cirklan arkon kun radiuso R ĉirkaŭ la testudo, de la angulo 1 ĝis la angulo 2. (Angulo 0 estas alsupre kaj kreskas horloĝe.)

originen, o
Remetas la testudon en ĝian komencan situon, tio estas, en la punkton kun koordinatoj [0 0] kaj rigardanta supren.

situon_provizu, sitp *listo*
Movas la testudon en punkton kun koordinatoj indikitaj de la listo de du nombroj (absciso kaj ordinato).

x_provizu, xp *x*
Movas horizontale la testudon ĝis la punkto kun absciso x.

y_provizu, yp *y*
Movas vertikale la testudon ĝis la punkto kun ordinato x.

xy_provizu,xyp *x y*
Same kiel sitp[x y]

direkton_provizu, dirp *n*
Direktas la testudon al la angulo indikita. 0 signifas vertikale alsupre; turni kiel indikiloj de horloĝo.

etikedu, etik *vortolisto*
Desegnas la vorton aŭ la liston indikitan, tie kie troviĝas la testudo kaj laŭ ĝia inklino. Ekzemple: `etikedu [Saluton al vi]` skribos la frazon «Saluton al vi» tie kie estas lokita la testudo kaj respektante la direkton de ĝi.

punkton_montru, punkt *listo*
Ŝaltas la punkton difinitan de la koordinatoj de la listo (je la koloro de krajon').

A.1.2 Atributoj de la testudo

La primitivoj prezentotaj ĉi tie ebligas modifi l' atributojn de la testudo. Por ekzemplo, ĉu necesas ke la testudo estu videbla sur l' ekrano? Je kiu koloro ĝi skribu kiam ĝi moviĝos?

testudon_montru, tdm
Videbligu la testudon sur l' ekrano.

testudon_kaŝu, tdk
Malvidebligu la testudon sur l' ekrano.

ekranon_viŝu, ev
Forviŝu la desegnejon kaj remetu la testudon en ĝian komencan situon.

purigu, pur
Forviŝu la desegnejon sed lasu la testudon sur la sama loko.

pradifine, pradif
Forviŝu la desegnareon kaj valorigu laŭ la apriorajn valorojn kelkajn parametrojn:

- krajonkoloro: nigra
- ekrankoloro: blanka
- moduso movada: malŝaltita
- tiparo por la grafikejo kaj la historiejo: Dialog 12 punktoj
- krajonformo: kvadrata
- desegna kvalito: normala
- maksimuma nombro de testudoj: 16
- moduso kontrolo: malŝaltita
- ekranamplekso: 1000x1000

mallevu, ml
La testudo skribu kiam ĝi moviĝas.

levu, l
La testudo ne skribu plu kiam ĝi moviĝas.

gumskrapu, gum
La testudo forviŝas ĉiun skribaĵon trovitan.

strekon_inversu, si
Mallevu la krajonon kaj metu la testudon en moduson inversi.

desegne, dsg
Mallevu la krajonon kaj metu ĝin en moduson de klasika desegno.

skribkoloron_provizu, skolp *entjer-listo [r v b]*
Establu la koloron de la krajono. Rigardu p. 89.

fonkoloron_provizu, fkolp *entjer-listo [r v b]*
Establu la koloron de la ekranfono. Rigardu p. 89.

situon, sit
Redonu la nunan situon de la testudo. Ekz.: `sit` redonus [10 -100]

direkton, dir
Redonas la direkton de la testudo (rigardu `fixecap`).

aldirektu, diral *listo*
La listo enhavu du nombrojn prezentantajn koordinatojn. Ĝi redonas la direkton kiu necesas doni al la testudo por iri ale al la punkto difinita de la koordinatoj de la listo.

distancon, dist *listo*
La listo enhavu du nombrojn prezentantajn koordinatojn. Ĝi redonas la nombron de paŝoj inter la nuna situ kaj la punkto difinita de la koordinatoj de la listo.

skribkoloron, sk
Redonu la nunan koloron de la krajono. Tiu koloro estas indikita per listo [r v b] kie r estas la konsistaĵo ruĝa, b la blua kaj v la verda.

fonkoloron, fkol
Redonu la nunan koloron de la ekranfono. Tiu koloro estas indikita per listo [r v b].

volve, vlv
Agordu la ekranliman moduson, tiel ke se la testudo eliras el la desegnejo, ĝi reaperas sur la mala flanko!

fenestre, fen
Agordu la ekranliman moduson, tiel ke la testudo estas libera por eliri en la desegnejo. Kompreneble, ĝi ne skribas ekster la desegnejo.

ferme, f
Agordu la ekranliman moduson, tiel ke la testudo restu en la desegnejo. Se ĝi devus eliri, erarmesaĝo avertos kaj donos la maksimuman paŝnombron antaŭ ol eliri.

perspektive
Agordu la desegnejon, tiel ke la testudo povu direktiĝi en la spaco. (Rigardu la sekcion A.2 dediĉita al tiu moduso.) Por eliri el tiu moduso, uzu la primitivon `fenestre`, `volve` aŭ `ferme`.

koloron, kol *listo*
Redonu la koloron de la pikselo (rastrumero) kun koordinatoj de la listo. Tiu koloro prezentiĝas kiel listo [r v b].

skribdikon_provizu, sdikp *nombro*
Establu la dikecon de la krajonpinto laŭ pikseloj. Apriore ĝi valoras 1.

skribdikon, sdik
Redonu la dikecon de la krajonpinto laŭ pikseloj.

sformp, skribformon_provizu *0-1*
Establu la formon de la krajonpinto.

- 0→kvadrata.

- 1→ronda.

sform, skribformon
Redonu la formon de la krajonpinto. 0→kvadrata. 1→ronda.

dsgcp, desegnecon_provizu *0-1-2*
Establu desegnan kvaliton.

- 0→normala.
- 1→alta.
- 2→malalta.

dsgc, desegnecon
Redonas la kvaliton de desegnado.

- 0→normala.
- 1→alta.
- 2→malalta.

dsgamplp, desegnamplekson_provizu *listo*
Establu l' amplekson de la desegnejo.

dsgampl, desegnamplekson
Redonu la amplekson de la desegnejo.

formon_provizu, formp *entjero*
Vi povas elekti l' aspekton de la testudo uzata, ĉu per Elekto - Preferoj - Elektu testudon, ĉu per tiu primitivo. La nombro devas esti entjero inter 0 kaj 6 (0 indikas la trilateran formon).

formon, form
Redonas numeron kiu reprezentas la nunan bildon de la testudo.

tiparon_provizu, tipp *entjero*
Kiam oni skribas tekston sur l' ekrano per la primitivo `etikedu`, eblas modifi la amplekson de la tiparo uzata, per tiu primitivo. Apriore, l' amplekso estas 12.

tiparon, tip
Redonas la amplekson de la tiparo nun uzata kiam oni skribas per la primitivo `etiquette`.

tiparnomon_provizu, tipnp *entjero*
Establu la tiparon uzata por skribi sur l' ekrano per la primitivo `etikedu`. La numero indikanta la tiparon uzotan estas trovebla en Menuo - Elektoj - Preferoj - Langeto Tiparo.

tiparnomon, tipn
Redonu liston konsistanta el du eroj. La unua estal la numero de la tiparo uzata por skribi per la primitivo `etikedu`. La dua estas listo enhavanta la nomon de tiu sama tiparo.

ekranon_disigu *nombro*
Establu la proporcio inter la grafikejo kaj la historiejo. La nombro estu inter 0 kaj 1. Kiam ĝi valoras 1 la desegnejo okupas ĉiom; kiam 0, la historiejo okupas ĉiom.

ekranon_disigon
Redonas la proporcion nunan inter la desegnejo kaj la historiejo.

dratretu *a b*
a kaj b estas entjeroj. Ĝi desegnas dratreton kies ĉiu ĉelo grandas a mul b.

neplu_dratretu
Forigas la dratreton.

dratretkoloron_provizu *koloro*
Ebligas elekti la koloron de la dratreto. Ekzemple: `dratretkoloron_provizu ruĝan`

dratreta_koloron
Redonas la nunan koloron de la dratreto.

dratreta?
Testu ĉu la dratreto desegnitas, kaj redonu vera aŭ malvera laŭ la okazo.

aksigu *n*
Grafiku la du aksojn. La indikiloj estas disigitaj de *n* testudpaŝoj.

x_aksigu *n*
Grafiku la horizontalan akson. L' indikilojn disigas *n* testudpaŝoj.

y_aksigu *n*
Grafiku l' vetikalan akson. L' indikilojn disigas *n* testudpaŝoj.

aksojn_viŝu
Forigu la aksojn.

akskoloron_provizu *koloro*
Establu la koloron de la aksoj. Ekzemple: `akskoloron_provizu [120 5 100]`

akskoloron
Redonas la nunan koloron de la aksoj.

x_aksa?
Testu ĉu la horizontala akso estas grafikita. Redonu vera aŭ malvera laŭ l' okazo.

y_aksa?
Testu ĉu la vertikala akso estas grafikita. Redonu vera aŭ malvera laŭ l' okazo.

zomu *n*
Zomu al la desegnejo. La argumento `n` reprezentas la skalon rilate al l' amplekso de la bildo establita en la preferejo.

fenestramplekson, fenampl
Redonu liston formita de la koordinatoj de la angulo supra maldekstra de la desegnejo kaj de la angulo malsupra dekstra.

avertu, avrt *listo*
Skribu informa mesaĝon en dialogfenestron; programrulado haltas ĝis jesa musklako.

etikedlongon, etikl *vortolisto*
Redonu la longon necesan por skribi la vorton aŭ la liston sur la desegnareon uzante la elektitan tiparon. Tiun longon oni esprimu per testudpaŝoj.

A.1.3 Iom pri l' koloroj

En XLogo la kolorojn oni difinas per tri nombroj inter 0 kaj 255. Tiu koda sistemo nomiĝas «RVB» (ruĝa, verda, blua). Ĉiu nombro respondas al la respektiva intenseco de la ruĝo, la verdo kaj la bluo por la konsiderata koloro. Ĉar tio ne estas tre intuicia, XLogo proponas ankaŭ 16 antaŭdifinitaj koloroj uzeblaj ĉu per numero, ĉu per primitivo.

Numero	Primitivoj	[R V B]	Koloro
0	`nigra`	[0 0 0]	
1	`ruĝan`	[255 0 0]	
2	`verdan`	[0 255 0]	
3	`flavan`	[255 255 0]	
4	`bluan`	[0 0 255]	
5	`violruĝan`	[255 0 255]	
6	`verdbluan`	[0 255 255]	
7	`blankan`	[255 255 255]	
8	`grizan`	[128 128 128]	
9	`hele_grizan`	[192 192 192]	
10	`malhele_ruĝan`	[128 0 0]	
11	`malhele_verdan`	[0 128 0]	
12	`malhele_bluan`	[0 0 128]	
13	`oranĝkoloran`	[255 128 0]	
14	`rozan`	[255 175 175]	
15	`violan`	[128 0 255]	
16	`maronan`	[153 102 0]	

```
# Jenaj tri komandoj efikas same
skolp oranĝkoloran
skolp 13
skolp [255 200 0]
```

A.1.4 La moduson movado (animado)

Ekzistas tri primitivoj: `movado`, `neplu_movigu` kaj `novigu`; kiuj ebligas ruli komandojn sen ke la testudo montru la rezultojn.

movado

Ŝaltu moduson movado. La testudo ne plu desegnu sur l' ekrano, sed nur en memoro. Por ĝisdatigi la desegnon sur l' ekrano, uzu la primitivon `rafraichis`. Tre utila por krei animadon aŭ por desegni pli rapide.

neplu_movigu

Tio haltigas la moduson movado: oni revenas en moduson klasikan, do oni tuj vidas la movojn de la testudo sur l' ekrano.

novigu

En moduso movado, novigu l' ekranon: la bildo sur la desegnejo estu ĝisdatigita.

Por indiki la moduson movado, ikono prezentanta fotilon aperas en la historiejo. Se vi klakos la fotilon, l' animado haltos; do tio egalas uzi la primitivon `neplu_movigu`.

A.1.5 Skribi tekston en la historiejo

Tiu tabelo grupas la primitivojn rilatajn al la historiejo. Ĉiu primitivo rilata al la amplekso kaj la koloro de la tiparo uzata, nur validas por la rezulto de la primitivo `skribu`.

tv, tekston_viŝu

Purigas la areon enhavantan la historion de la komandoj kaj komentarioj.

s, skribu *arg1*

Skribu l' argumenton *arg1* en la historiejon.

```
skribu "abcd --------> abcd
s [1 2 3 4] ----> 1 2 3 4
s 4 ------------> 4
```

tajpu *arg1*

Same kiel la primitivo `skribu` sed ĝi ne saltas linion.

teksttiparon_provizu, ttipp *n*

Difinu l' amplekson de la tiparo en la historiejo. Valida nur por la primitivo `skribu`.

teksttiparon, ttip

Redonas la amplekson de la tiparo uzata de la primitivo `skribu`.

tekstkoloron_provizu, tkolp *koloro*

Difinu la koloron de la tiparo en la historiejo. Valida nur por la primitivo `skribu`. Rigardu p. 89.

tekstkoloron, tkol

Redonas la koloron de la tiparo uzata de la primitivo `skribu` en la historiejo.

teksttiparnomon_provizu, ttipnp *n*

Establu la tiparon uzatan por skribi en la historiejo per la primitivo `skribu`. La numero de la tiparo estas serĉebla en Menuo→Elektu→Preferoj→Langeto Tiparo.

teksttiparnomon, ttipn

Redonas liston konsistanta el du eroj. La unua estas la numero reprezentanta la tiparon uzatan por skribi sur l' ekrano per la primitivo `skribu`. La dua estas listo enhavanta la nomon de tiu sama tiparo.

stilon_provizu, stip *arg1*

Establu la stilon de la tiparo uzata de la primitivo `skribu`. La stiloj elekteblaj estas `simple`, `dike`, `kursive`, `forstreke`, `indice`, `eksponente`, `substreke`. Se vi deziros uzi plurajn samtempe, indiku ilin en listo.

Jen kelkaj ekzemploj por formati la tekston per la primitivo `skribu`:

```
stilon_provizu [dike substreke] skribu "saluton
bonjour
stip "forstreke tajpu [teksto forstrekita] stip "kursive tajpu "\ x stip "eksponente skribu
2
```

~~teksto forstrekita~~ x^2

sti, stilon

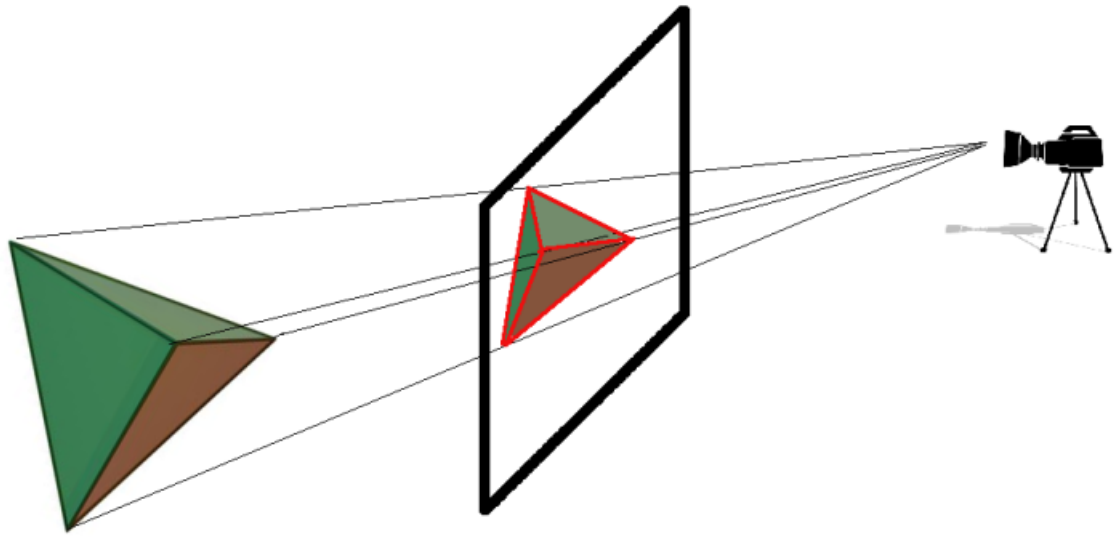
Redonu liston konsistantan el la malsamaj setiloj nun uzataj de la primitivo `skribu`.

A.2 La testudo en la spaco

De la versio 0.9.92, la testudo povas eliri el la ebeno por moviĝi en la spaco. Por tio oni uzu la primitivon `perspektive`. Bonvenon en la mondon de la 3D-perspektivo!

A.2.1 La perspektiva teĥniko

Por prezenti la spacon tridimensian en ebeno nur dudimensia, oni uzu projektan perspektivon. Kamerao rigardas la 3D-scenon kaj ĝia vidado estas projektata sur intermezan ebenon. Jen skemo ilustranta tiun teĥnikon.



Kelkaj primitivoj ebligas loki la kameraon laŭ via volo, kun la projekta ekrano ĝuste je duona distanco.

A.2.2 Kompreni la movojn en la spaco

Sur la ebeno, testudan direkton difinas nur l' angulo rilate al la vertikalo. En la spaco, la direkton donas 3 angulaj valoroj:

- La flankklino: La testuda klino laŭ la akso (Oy)
- La frontklino: Laŭ la akso (Ox)
- Le direkto: Laŭ l' akso (Oz)

Efektive, por moviĝi en la spaco, la testudo kondutas kiel aviadilo. Jen malgranda skemo ebliganta prezenti tiujn magnitudojn:

La flankklino **La frontklino** **La direkto**

Tio povas ŝajni komplikita komence, sed rimarku ke multaj aferoj rilatas al kutimaj movoj sur l' ebeno.

an, antaŭen, man, malantaŭen *n*

Sama konduto kiel sur l' ebeno.

dn, dekstren, mdn, maldekstren *n*

Same kondutas kiel sur l' ebeno.

dfn, dekstraflanken *n*

La testudo pivotas dekstren laŭ ĝia longeca akso je *n* gradoj.

mdfn, maldekstraflanken *n*

La testudo pivotas madekstren laŭ ĝia longeca akso je *n* gradoj.

supren *n*

La testudo pivotas supren laŭ ĝia larĝeca akso je *n* gradoj.

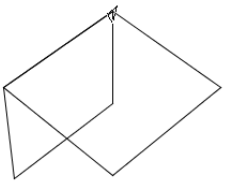
malsupren *n*

La testudo pivotas supren laŭ ĝia larĝeca akso je *n* gradoj.
Sur l' ebeno por grafiki kvadraton je latero 200:

```
ripetu 4 [an 200 dn 90]
```

Tiuj komandoj restas validaj en la spaco, kaj la kvadrato estas desegnita perspektive. Se oni turnus «malsupren» la testudon je 90 gradoj oni povus grafiki tiam novan kvadraton.

```
ev
ripetu 4 [an 200 dn 90]
malsupren 90
ripetu 4 [an 200 dn 90]
```



Restas trejni sin por lerni ĉiun eblan direkton!
Necesas ĉiuokaze kompreni ke la tri turnaj primitivoj estas reciproke ligitaj. Por ekzemplo, provu la jenan sinsekvon:

```
ev
mdfn 90 supren 90 dfn 90
```

La movo farita estus egala al fari `maldekstren 90` (provu simuli la testudon per via mano, ekzemple...).

A.2.3 Listo de aliaj primitivoj

Ĉiuj sekvaj primitivoj valoras en la spaco kaj sur l' ebeno. La sola diferenco estas la naturo de la argumentoj atendataj aŭ la naturo de la respondoj. Por ekzemplo, la primitivo `sitp` ou `situon_provizu` atendas ĉiam

liston kiel argumenton, sed nun necesas ke tiu listo enhavu tri nombrojn $(x; y; z)$ reprezentantajn la spacajn koordinatojn de la dezirata punkto. Jen resumo de tiuj komandoj:

Primitivoj validaj kaj sur l' ebeno kaj en la spaco

`rond, rondon_desegnu`	`ark, arkon_provizu`	`o, originen`	`diral, aldirektu`
`dist, distancon`	`sitp, situon_provizu`	`xp, x_provizu`	`yp, y_provizu`
`dirp, direkton_provizu`	`etikedu`	`etikedlongon, etikl`	`punkt, punkton_montru`
`sit, situon`	`dir, direkton`		

Primitivoj validaj nur en moduso 3D

xyzp, xyz_provizu *x y z*

Tiu primitivo movas la testudon al la punkto kun koordinatoj indikitaj. Ĝi atendas tri argumentojn; tiu primitivo similas al `sitp` krom ke la koordinatojn oni ne indikas en listo.

Ekzemplo, `xyzp -100 200 50`: movu la testudon al punkto je koordinatoj $x = -100; y = 200; z = 50$.

zp, z_provizu *z*

Tiu primitivo movas la testudon al punkto kies koordinato z egalas l' argumenton indikitan. Ĝi atendas do nombron kiel argumenton; tiu primitivo estas komparebla al `xp` kaj `yp`.

orientadon_provizu *listo*

Loku la testudon laŭ la dezirata klino. Tiu primitivo atendas liston enhavantan tri nombrojn, respektive la flankklinon, la frontklinon kaj la direkton.

Ekzemple, `orientadon_provizu [100 0 58]`: la testudo iĝos flankklina je 100 gradoj, frontklina je 0 gradoj kaj direkta je 58 gradoj.

orientadon

Redonas l' orientadon de la testudo kiel liston enhavantan respektive la flankklinon, la frontklinon kaj la direkton. Atentu l' ordon de tiuj nombroj; ekzemple, se l' orientado estas `[100 20 90]`, tio signifas ke por atingi la saman orientadon, de la komenca situo (ekzemple post viŝi l' ekranon), necesos tajpi:

```
dekstraflanken 100 supren 20 dekstren 90
```

Se vi permutus l' ordon de tiuj instrukcioj, vi ne akirus l' orientadon deziratan!

flankklinon_provizu *n*

Pivotigu la testudon laŭ ĝia longeca akso tiel ke ĝi prenu l' frankklinon indikitan.

flankklinon

Redonas la nunan valoron de la flankklina angula.

frontklinon_provizu *n*

Pivotigu la testudon laŭ ĝian larĝeca akso tiel ke ĝi prenu la frontklinan angulon indikitan.

frontklinon

Redonas la nunan valoron de la frontklinan angulon.

A.2.4 La 3D-modelilo

XLogo havas modelilon 3D kiu ebligas montri vian tridimensian grafikaĵon en sceno kun lumoj kaj mallumoj. Tiu modulo uzas la bibliotekon JAVA3D kiu instalitu se vi volas profiti tiun kapablon.

Jen kiel uzi la modelilon:

Dum oni desegnas, post ĉiu grupo de komandoj, indiku al modelilo la geometriajn formojn kiujn ĝi konservu por estonta grafikado. Eblas registri plurlaterojn (surfacojn), liniojn, punktojn kaj eĉ tekstojn. Por tio, oni havas la jenajn primitivoj:

por_edro

Ĉiun sekvan movon oni registros por krei plurlateron.

fino_edro

La aro de la verticoj tra kiuj pasis la testudo post la voko de `por_edro` faras kvarlateron kies koloron establas l' aro de verticoj. Tiu primitivo finas krei la plurlateron.

linia_difino

Ĉiun sekvan movon registru por krei sinsekvon de segmentoj.

linia_difinhalto

La aro de verticoj tra kiuj la testudo pasis post voki `linia_difino` materialigas plursegmentan linion. La primitivo finas difini la linion.

punkta_difino

Ĉiun sekvan movon registru por krei aron de punktoj.

punkta_difinhalto

Finu registri la aron de punktoj tra kiuj la testudo pasis post voki `punkta_difino`

teksta_difino

Ĉiam kiam l' uzulo afiŝos tekston per la primitivo `etikedu`, ĝin oni registros por esti uzota de la modelido 3D.

teksta_difinhaltu

Finu registri la tekstojn afiŝatajn.

tridimensie_vidigu

Ekrulu la modelilon 3D; ĉiun objekton antaŭe registritan oni afiŝu sur l' ekranon.

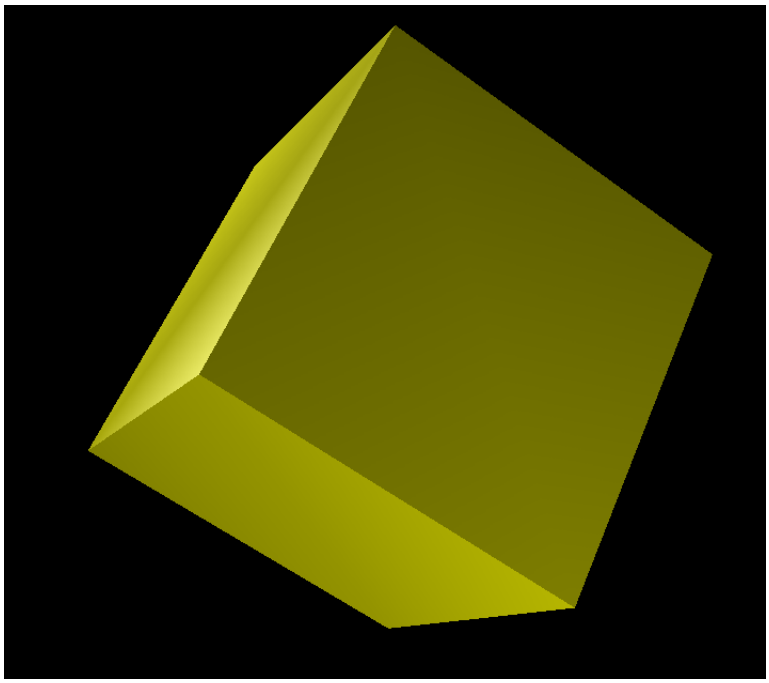
A.2.5 Krei kubon

Ĉiu faco estas kvadrato kun latero je 400 testudpaŝojn. Jen la programo;

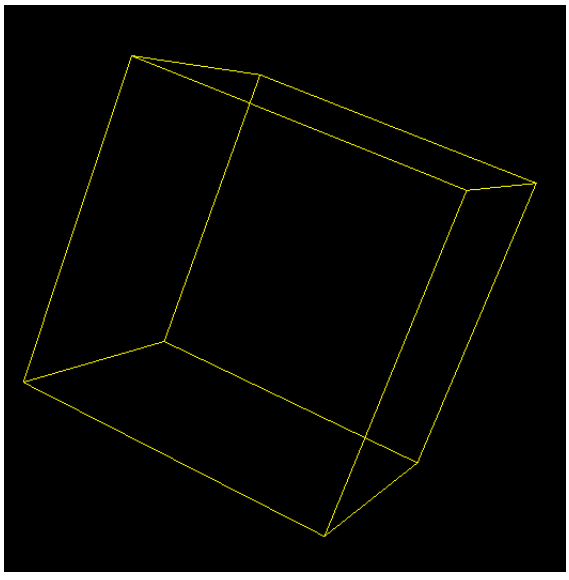
```
por kvadrato
# registru la verticojn de l' kvadrato
por_edro ripetu 4 [an 400 dn 90] fino_edro
fino

por simplaKubo
# flava kubo
ev perspektive skolp flavan
# flankaj facoj
```

```
ripetu 4 [kvadrato l dn 90 an 400 mdn 90 dfn 90 ml]
# malsupra faco
malsupren 90 kvadrato supren 90
# supra faco
an 400 malsupren 90 kvadrato
# vidigo
tridimensie_vidigu
fino
```

Rulu la komandon `simplaKubo`:



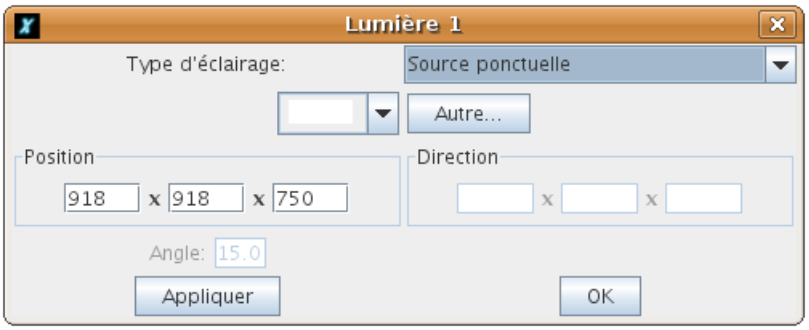
Post anstataŭigi en la proceduro `kvadrato`, `por_edro` per `linia_difinu` kaj `fino_edro` per `linia_difinhaltu`:



Se oni uzus `punkta_difino` kaj `punkta_difinhaltu` anstataŭ `linia_difinu` kaj `linia_difinhaltu`, oni havus sur l' ekrano nur la 8 verticojn de la kubo. Tiujn du primitivojn oni uzu por vidigi punktonubojn en la spaco.

A.2.6 Administri la lumojn

Por klarigi viajn 3D-scenojn vi povas uzi kvar lumojn. Apriore, la scenon klarigas du lumoj el tipo punkta. Klaku sur unu el la 4 ampuloj en la 3D-modelilo; la jena dialogfenestro aperos tiam:



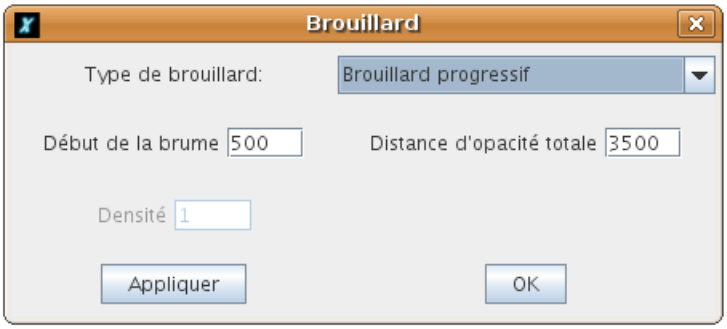
Eblas pluraj elektoj de lumoj:

- Media lumo: unuforma lumo; necesas nur indiki ĝian koloron.
- Unudirekta fonto: lumo klariganta laŭ unu nura konstanta direkto; ĝi egalas la okazon de punkta fonto lokita tre malproksime, ekzemple la suno.
- Punkta fonto: lumo kies loko oni konas; komparebla al ampulo, lumturo...
- Lummakulo: punkta fonto lumanta nur en konuso, kies angulon oni indiku.

  Plej bone oni simple provu ilin por kompreni ilian funkciadon!

Nebuleca efiko

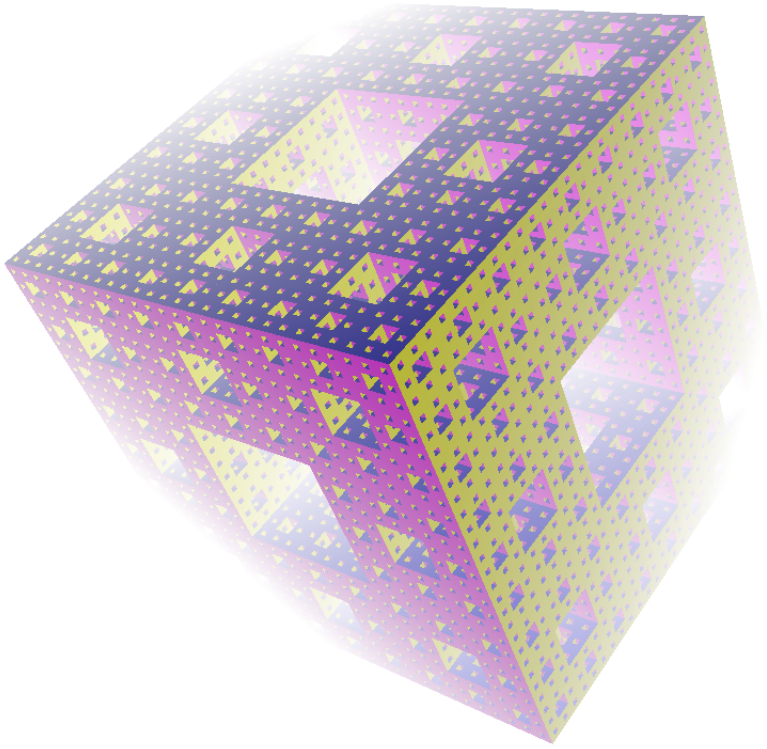
Vi povas aldoni efikon kvazaŭ nebulan al via 3D-sceno. Klaku la butonon nuboforman en la 3D-modelilon; jen la dialogfenestro kiu aperas:



Du elekteblaj nebultipoj:

- Grada nebulo: nebulo kies opakeco fariĝas ĉiam pli granda. Vi indiku du parametrojn:
  - La distanco kie komenciĝas la nebulo.
  - La distanco kie la opakeco de la nebulo estas tuta.
- Densa nebulo: nebulo konstanta ĉie en la sceno. Indiku nur la densecon de la nebulo.

Ekzemplo kun grada nebulo:



A.3 Aritmetikaj kaj logikaj operaciojn

sumon x y

Adiciu la du nombrojn x kaj y, poste redonu la rezulton.
Ekzemple: `sumon` 40 60 redonas 100

subtrahon x y

Redonu la subtrahon $x - y$.
Ekzemple: `subtrahon` 100 20 redonas 80

mns, minusigan x

Redonu $-x$.
Ekzemple: `mns` 5 redonas -5. Legu la rimarkon post la sekvaj primitivoj.

prod, produton x y

Redonu la produton de x kaj y (x mul y).

div, dividon x y

Redonu la kvocienton de x per y (x div y).
`div 15 6` redonas 2.5

kvoc, kvociento x y

Redonu la entjeran kvocienton de x per y
`kvoc 15 6` redonas 2

rest, reston x y

Redonas la reston de la divido de x per y.

entjeran x

Redonu la plej proksiman entjeron de la nombro x.
`entjeran 6.4` redonas 6
`entjeran 6.8` redonas 7

entjeran_parton x

Redonu la plej proksiman entjeron de la nombro x ale al nulo.
`entjeran_parton 6.8` redonas 6

potencon x n

Redonu x je la n-a potenco (x pot n).
`potencon 3 2` redonas 9

rdk, radikon n

Redonu la kvadratan radikon de n

log x

Redonu la logaritmon de x.

eksp x

Redonu la eksponencialon de x.

log10 x

Redonu la dekuman logaritmon de x.

sinuson, sin x

Redonu la sinuson de la nombro x (x en gradoj).

kosinuson, kos x

Redonu la kosinuos de la nombro x (x en gradoj).

tangenton, tan x

Redonu la tangenton de la nombro x (x en gradoj).

arkokosinuson, akos x

Redonu la malkosinuson, la angulon kies kosinuso valoras x (angulo en gradoj).

arkosinuso, asin x

Redonu la malsinuson, la angulon kies sinuso valoras x (angulo en gradoj).

arkotangenton, atan x

Redonu la maltangenton, la angulon kies tangento estas x (angulo en gradoj).

pi

Redonu la nombron π (3.141592653589793).

hazardon, hzd n

Redonu hazardan entjeron inter 0 kaj $n-1$.

absolute, abs *x*

Redonu l' absolutan valoron (distancon al nulo) de la nombro donita.

decimalojn_provizu *n*

Establu la nombron de decimaloj uzataj por la kalkuloj.
Efektive ĝi regulas la precizecon de la kalkuloj. Jen komentarioj:

- Apriore, la kalkuloj uzas 16 decimaloj.
- Se *n* estas negativa, la aprioran afiŝan moduson oni elektu.
- Se *n* estas nulo, uzu la entjeron plej proksiman por afiŝi.

Tiu primitivo estas tre utila por fari kalkulojn bezonantajn multajn decimalojn. Rigardu l' ekzemplon pri la nombro π je la p. 41.

decimalojn

Redonu la nombron de decimaloj uzataj por la kalkuloj. Apriore, tiu valoro estas -1. *RIMARKO :*

Atentu la primitivojn bezonantajn du parametrojn!

Ekzemple: `xyp` a b Se b estas negativa
Por ekzemplo, `xyp` 200 -10

L' interpretilo Logo faros l' operacion $200-10$. Ĝi do konsideros ke estas nur unu parametro (190) sed necesas al ĝi du, tial erarmesaĝo. Por ne havi tiajn problemojn, uzu la primitivon «`mns` ». `xyp 200 mns 10` tiel, nul problemo plu!

Alia eblo estas uzi krampojn: `xyp 200 (-10)`

Jen listo de logikaj operatoroj:

aŭ *b1 b2*

Redonu `vera` se `b1` aŭ `b2` estas vera; se ne, redonu `malvera`.

kaj *b1 b2*

Redonu `vera` se `b1` kaj `b2` estas veraj ambaŭ; se ne, redonu `malvera`.

ne *b*

Redonu la neaĵon de la bulea `b`.

- Se `b` estas vera, redonu `malvera`.
- Se `a` estas malvera, redonu `vera`.

A.4 Operacioj al listoj kaj vortoj

vort, vorton *vor1 vor2*

Konkatenu (kunigu en novan vorton) la du vortojn *vor1* kaj *vor2*.
Ekzemple: `s vort "a 1` redonas `a1`

list, liston *arg1 arg2*

Redonu liston konsistantan el *arg1* kaj *arg2*. Ekzemple:
`list 3 6` redonas [3 6].
`list` "unu "listo redonas [unu listo]

frazon, fr *arg1 arg2*

Redonu liston konsistantan el *arg1* kaj *arg2*. Se *arg1* aŭ *arg2* estas listo, tiam ĉiu konsistiganto el *arg1* aŭ *arg2* fariĝu ero de la kreota listo (tio estas, forigu la krampojn).

Ekzemple:
`fr [4 3] "saluton` redonas [4 3 saluton]
`fr` [kiel vi] "fartas redonas [kiel vi fartas]

kununuan, unk *arg1 listo2*

Enmetu *arg1* en l' unuan lokon de la listo.
Ekzemple: `uk` "hola [2] redonas [hola 2]

kunlastan, lastk *arg1 listo2*

Enmetu *arg1* en la lastan lokon de la listo.
Ekzemple: `lk` 5 [7 9 5] redonas [7 9 5 5]

inversan, inv *lista*

Inversigu l' ordon de la eroj de la listo.
`inv` [1 2 3] redonas [3 2 1]

elekton, elkt *arg1*

- Se *arg1* estas vorto, redonas unu literon el *arg1* hazarde prenitan.
- Se *arg1* estas listo, redonas unu eron el *arg1* hazarde prenitan.

forigu, for *arg1 listo*

Forigu la eron *arg1* de la listo se ĝi aperas ene.
Ekzemple: `for` 2 [1 2 3 4 2 6] redonas [1 3 4 6]

eron, er *n arg2*

- Se *arg2* estas vorto, redonu la *n*-an literon el la vorto (1 indikas la unuan literon).
- Se *arg2* estas listo, redonu la *n*-an eron el la listo.

senlastan, ls *arg*

- Se *arg* estas listo, redonu la tutan liston krom la lasta ero.
- Se *arg* estas vorto, redonu la vorton krom ĝia lasta litero.

senunuan, us *arg*

- Se *arg* estas listo, redonu la tutan liston krom la unua ero.
- Se *arg* estas vorto, redonu la vorton sen ĝia unua litero.

lastan, last *arg*

- Se *arg* estas listo, redonu la lastan eron el la listo.
- Se *arg* estas vorto, redonu la lastan literon el la vorto.

unuan, un *arg*

- Se *arg* estas listo, redonu la unuan listeron.
- Se *arg* estas vorto, redonu l' unuan literon el la vorto.

anstataŭigu *listo1 n arg*

En `listo1`, anstataŭu la `n`-an eron la vorto aŭ la listo donita.
`anstataŭigu [a b c] 2 8 --> [a 8 c]`

almetu *listo1 n arg*

En `listo1`, enmetu en `n`-an lokon la vorton aŭ liston donitan.
`almetu [a b c] 2 8 --> [a 8 b c]`

nombru *arg*

- Se *arg* estas listo, redonu la nombron de eroj en *arg*.
- Se *arg* estas vorto, redonu la nombron de literoj en *arg*.

unikode *vor1*

Redonu l' unikodan valoron de la signo «*vor1*».
`s unikode "A` redonas 65

literige, lit *n*

Redonu la signon (literon) kies unikoda valoro estas *n*.
`s lit 65` redonas `"A`

A.5 Buleaj

Primitivo estas bulea se ĝi redonas vorton `"vera` aŭ vorton `"malvera`. Tiuj primitivoj finiĝas per demandosigno.

vera

Redonu `"vera`.

malvera, mvera

Redonu `"malvera`.

vort?, vorta? *arg1*

Redonu `"vera` se *arg* estas vorto, `"malvera` se ne.

nb?, nombra? *arg1*

Redonu `"vera` se *arg1* estas nombro, `"malvera` se ne.

entjera? *arg1*

Redonu `"vera` se *arg1* estas entjero, `"malvera` se ne

list?, lista? *arg1*

Redonu `"vera` se *arg1* estas listo, `"malvera` se ne.

mpl?, malplena? *arg1*

Redonu `"vera` se *arg1* estas malplena listo aŭ malplena vorto; `"malvera` se ne.

eg?, egal? *arg1 arg2*

Redonu `"vera` se *arg1* kaj *arg2* estas egalaj; `"malvera` se ne.

ĉu_antaŭas? *vor1 vor2*

Redonu `"vera` se *vor1* estas antaŭ *vor2* laŭ alfabeta ordo; `"malvera` se ne.

membra?, mbr? *arg1 arg2*

- Se *arg2* estas listo, respondu ĉu *arg1* estas ero el *arg2*.
- Se *arg2* estas vorto, respondu ĉu *arg1* estas signo el *arg2*.

membron, mbr *arg1 arg2*

- Si *arg2* estas listo, serĉu `arg1` en tiu listo; du eblaj okazoj:
  - Se *arg1* estas en *arg2*, redonu la subliston generitan ekde la unua apero de *arg1* en *arg2*.
  - Se *arg1* ne estas en *arg2*, redonas la vorton `"malvera`.
- Se *arg2* estas vorto, serĉu la signon *arg1* en *arg2*; du eblaj okazoj:
  - Se *arg1* estas en *arg2* redonu la finon de la vorto, ekde *arg1*.
  - Se ne, redonu la vorton `"malvera`.

`mbr "o "coucou` redonas `oucou`
`mbr 3 [1 2 3 4]` redonas `[3 4]`

mallevata?, ml?

Redonu la vorton `"vera` se la krajono estas mallevita; `"malvera` se ne.

videbla?

Redonu la vorton `"vera` se la testudo estas videbla; `"malvera` se ne.

primitiva?, prim? *vor1*

Redonu `"vera` se la vorto estas primitivo de XLogo; `"malvera` se ne.

programera?, prog? *vor1*

Redonu `"vera` se la vorto estas proceduro difinita de l' uzulo; `"malvera` se ne.

var?, variabla? *vor1*

Konstatu ĉu *vor1* estas variablo. Redonu `"vera` aŭ `"malveraa` laŭ la okazo.

A.6 Efektivigu teston per la primitivo se

Kiel en ĉiu programlingvo, Logo ebligas konstati ĉu donita kondiĉo estas vera aŭ malvera, por ruli la rilatan kodpecon.

La primitivo **se** ebligas tion.

se *testo listo1 listo2*

- Se *testo* estas vera la komandojn en *listo1* rulu.
- Se *testo* estas malvera la komandojn en *listo2* rulu.

Oni povas ne meti la duan liston de instrukciojn.
Ekzemploj de uzo:

- `se 1+2>=3 [skribu "vera] [skribu "malvera]`
- `se (unuan "XLOGO)="Y [an 100 dn 90] [s [XLOGO komenciĝas per X!]]`
- `se (3*4)=6+6 [s 12]`

Rimarko: Kiam la rezulto de la unua esprimo estas `malvera`, la primitivo `se` serĉas duan liston, tio estas esprimon komenciĝantan per malferma krampo. En kelkaj tre specialaj okazoj, ĝi ne povas plenumi tiun kondiĉon, kaj tiam necesas uzi la primitivon `se_sene` . Ekzemple:

```
# Provizu du listojn al la variabloj a kaj b
provizu "a [skribu vera]
provizu "b [skribu malvera]

# unue testu per primitivo "se" --> la duan liston oni ne povas evalui
se 1=2 :a :b
Kiel uzi [skribu malvera]?

# due testu per primitivo "se_sene" --> efiko dezirita
se_sene 1=2 :a :b
malvera
```

A.7 La laborspaco

La laborspaco konsistas el ĉiu objekto difinita de l' uzulo. Tio enhavas:

- La proceduroj.
- La variabloj.
- La listoj de atributoj.

A.7.1 La proceduroj

Prezentado

Proceduroj estas iaj «prograoj». Per voki ĝiajn nomojn, oni rulas l' instrukciojn enhavatajn en la korpo de la proceduro. Por difini proceduron oni uzu la ŝlosilvorton **pour**.

```
por nomo_de_la_proceduro :v1 :v2 :v3 .... [:v4 ....] [:v5 ....]
  Korpo de la proceduro
fino
```

- nomo_de_la_proceduro estas la nomo donita al la proceduro.
- :v1 :v2 :v3 reprezentas la variablojn uzatajn en tiu proceduro (lokaj variabloj).
- [:v4 ...], [:v5 ...] estas nenecesaj variabloj, kiujn oni povas aldoni al la proceduro. (Rigardu klarigon postan.)
- Korpo de la proceduro reprezentas l' instrukciojn rulotajn kiam voki tiun proceduron.

Ekz:

```
por kvadrato :c
  ripetu 4 [an :c dn 90]
fino
```

La proceduro nomiĝas `kvadrato` kaj havas parametron nomiĝantan c. `kvadrato` 100 produktos do kvadraton je latero 100. (Rigardu l' ekzemplojn de proceduroj ĉe la fin' de la libro.)

Post la versio 0.7c eblas aldoni komentariojn en la kodo per antaŭigi la signon # al ili.

```
por kvadrato :c
# ĉi tiu proceduro ebligas grafiki kvadraton je latero :c
ripetu 4 [an :c dn 90] # praktika, ĉu ne?
fino
```

Malnepraj variabloj

Nun estas eble en XLogo uzi «apriorajn»valorojn por argumentoj. Konsideru la jenan proceduron:

```
por poli :n [:l 10]
ripetu :n [an :l dn 360/:n]
fino

# Tio grafikas poligonon (plurlateron) kies
# 20 lateroj mezuras 10 testudpaŝojn
poli 20
```

Dum l' interpretado, la variablon `:l` anstataŭas ĝia apriora valoro, tio estas 10. Se oni deziras ŝanĝi tiun valoron, voku la proceduron `poli` inter krampoj por indiki al interpretilo ke oni uzos malneprajn (fakultativajn, nenecesajn) parametrojn.

```
# Tio grafikas regulan plurlateron kies
# 20 lateroj mezuras nun 5 testudpaŝojn
(poli 20 5)
# Tio grafikas kvadraton kies lateroj
# mezuras 100 testudpaŝojn
(poli 4 100)
```

La primitivo 'programon_kontrolu'

Por kontroli la ruladon de programo eblas skribigi al ĝi la procedurojn rulatajn. Tiu moduso ebligas ankaŭ afiŝi ĉu la proceduroj redonas valorojn per la primitivo `sendu`.

programon_kontrolu

Ŝaltas la kontrolan moduson.

programon_kontrolhaltu

Malŝaltas la kontrolan moduson.

Jen malgranda ekzemplo per la faktorialo (rigardu p. 40).

```
program_kontrolu skribu fak 4
fak 4
  fak 3
    fak 2
      fak 1
      fak sendas 1
    fak sendas 2
  fak sendas 6
fak sendas 24
24
```

A.7.2 La variabloj

Ekzistas du specoj de variabloj:

- La mallokaj variabloj: ili ĉiam haveblas ĉie ajn en la programo.
- La lokaj variabloj: ili nur haveblas en la proceduro kie oni difinas ilin.

En tiu versio de Logo, la lokaj variabloj ne estas uzeblaj en la sub-proceduroj. Dum eliri el la proceduro, la lokaj variabloj malaperas.

provizu *vor1 arg2*

- Se la loka variablo *vor1* ekzistas, provizu al ĝi la valoron *arg2*.
- Se ne, kreu la mallokan variablon *vor1* kaj provizu al ĝi la valoron *arg2*.

Ekzemplo: `provizu` "a 100 provizas 100 al la variablo `a`.

lokvark, lokan_varianton_kreu *arg1*

- Se *arg1* estas vorto, kreu la lokan variablon nomatan *arg1*.
- Se *arg1* estas listo, el ĉiu ero kreu lokan variablon.

Por provizi al ĝi valoron, rigardu `lokp`.

loke_provizu, lokp *vor1 arg2*

Kreu novan lokan variablon *vor1* kaj provizu ĝin per la valoro *arg2*.

dif, difinu *vor1 listo1*

Difinu novan proceduron nomatan *vor1*.
`listo1` enhavas plurajn listojn:

- La unua listo enhavas la variablojn de la proceduro, inkluzive de la malnepraj variabloj.
- Ĉiu sekva listo prezentas linion de la proceduro.

```
dif "plurlatero [[nl longeco] [ripetu :nl [an :longeco dn 360/:nl]]]
```

Tio difinas proceduron nomatan `plurlatero` kun du variabloj (`:nl` kaj `:longeco`). Ĝi ebligas grafiki regulan plurlateron kies nombron de lateroj, kaj la longon de ĉiu latero, oni povas elekti.

difinon *vor1*

Donu ĉiun informon pri la proceduro nomata `vor1`. Ĝi donas liston enhavantan plurajn listojn.

- La unua el tiuj listoj enhavas la variablojn de la proceduro `vor1`.
- La sekvaj listoj prezentas ĉiun linion de la proceduro.

Tiu primitivo estas kompreneble asociita al la primitivo `difinu`.

enhv, enhavon *vor1*

Donu la valoron de la variablo *vor1*.
`enhv "a` kaj `:a` estas du samefikaj notacioj.

progcit, programerojn_citu

Donu liston enhavantan ĉiun proceduron nun difinitan.

varlist, variantliston

Donu liston enhavantan ĉiun variablon nun difinitan.

ecan_liston

Donu liston enhavantan la ecolistojn aktuale difinitajn.

enhavon

Redonu liston konsistantan el tri aliaj listoj. La unua enhavas ĉiun difinitan proceduron, la dua ĉiun variablon kaj la tria ĉiun eco-liston.

programeraro

Redonu liston enhavantan ĉiun konatan primitivon.

nomon_viŝu, nv *arg1*

Forviŝu la proceduron nomatan *arg1*, aŭ ĉiun proceduron enhavitan en la listo *arg1*.

varianton_viŝu, varv *arg1*

Forviŝu la variablon *arg1* aŭ ĉiun variablon enhavitan en la listo *arg1*.

ecan_liston_viŝu *arg1*

Forviŝu la ecoliston nomatan *arg1* aŭ ĉiun ecoliston enhavitan en la listo *arg1*.

njv, nomojn_viŝu

Forviŝu ĉiun variablon, ecoliston kaj proceduron difinitan en la laborspaco.

adiaŭ, adiau, adiaw, adiaux

Eliru XLogo.

ekzekutu, ekzek *listo1*

Rulu la instrukciliston enhavitan en *listo1*.

startigu *listo1*

Ebligas ruli sistemkomandon disde XLogo. *listo1* devas enhavi plurajn listojn enhavantajn ĉiun vorton konsistigantan la komandon. Jen ekzemploj:
`startigu [[gedit] [/home/xlogo/dosiero.txt]]`
Rulu l' aplikon `gedit` kaj malfermu la dosieron `/home/xlogo/fichier.txt` (GNU/Linux)
`startigu [[notepad] [C: /dosiero.txt]]`
Rulu l' aplikon `notepad` kaj malfermu la dosieron nomatan `C: /dosiero.txt` (ReactOS)

Tiu iom stranga sintakso ebligas ja uzi spacojn en la dosiervojoj.

A.7.3 La ecolistoj

Ekde la versio 0.9.92, XLogo subtenas ecolistojn. Ĉiu ecolisto havas ĉefan nomon kaj konsistas el aro de paroj Ŝlosilo-Valoro.

Por ekzemplo, konsideu ecoliston nomatan «aŭto». Ĝi povas enhavi ekzemple la ŝlosilon «koloro» asociitan al la valoro «ruĝa», aŭ ankaŭ la ŝlosilon «speco » asociitan al la valoro «kabrioleto».

Por manipuli tiajn listojn, oni havas la jenanj priitivojn:

econ_provizu *nomo ŝlosilo valoro*

Konsideru la ecoliston *nomo* (se ĝi ne ekzistas, kreu ĝin). *valoro* kiu estos havebla per la vorto *ŝlosilo*.

econ_sendu *nomo ŝlosilo*

Pluku en la ecolisto `nomo` la valoron asociita al la dezirata ŝlosilo. Se la ecolisto ne ekzistas aŭ se la ŝlosilo ne ekzistas, donu malplenan liston.

econ_viŝu *nomo ŝlosilo*

En la ecolisto `nom`, forviŝu l' valoron asociitan al la elektita ŝlosilo.

ecajn_listojn *nomo*

Donu la aron de paroj ŝlosilo-valoro enhavita en la ecolisto `nomo`. Repensu pri l' ekzemplo de la listo «voiture».

```
# Plenigi la liston
 econ_provizu "aŭto "koloro "ruĝa
 econ_provizu "aŭto "speco "kabrioleto
 econ_provizu "aŭto "firmao "Citroën

# Konsulti iun valoron
 skribu econ_sendu "aŭto "koloro
 ruĝa

# Konsulti ^ciun eron
 skribu ecan_liston "aŭto
 koloro ruĝa speco kabrioleto firmao Citroën
```

A.8 Administri dosierojn

bildon_ŝargu, bild *vor1*

Ŝargu la bildodosieron *vor1*. Ĝia supra maldekstra angulo estos lokita tie kie estas la testudo. La formatoj subtenataj estas PNG (png) kaj JPEG (jpg).

La vojo indikita estu relativa rilate al la nuna dosierujo. Ekz.: `bild "testudo.jpg]`

katalago, ktlg

Listu la enhavon de la apriora dosierujo. (Responda al la komando `ls` por Linukso aŭ `dir` por FreeDOS).

dosierujon_provizu, regp *vor1*

Faru ke la nuna dosierujo estu tiu indikata de la vojo `vor1`.

celu_dosieron, cd *vor1*

Ĝi ebligas elekti la nunan dosierujon. La voj' estu relativa rilate al la ankoraŭa nuna dosierujo. Oni povas uzi la notacion «..» por celi la patran dosierujon.

dosierujon, dos

Donu la nunan dosierujon. Apriore, ĝia valor' estas la uzula hejma dosierujo, tio estas `/home/via_login` por la gnulinuksuloj, `C:\WINDOWS` por aliaj.

konservu, ksrv *vor1 listo2*

Ekzemplo pli bone klarigas tion:

`konservu "provo.lgo [proc1 proc2 proc3]` konservas en la dosieron `provo.lgo` de la nuna dosierujo la procedurojn `proc1`, `proc2` kaj `proc3`. Se la finaĵo .lgo forestas, ĝin aldonas oni apriore. La vorto indikas relativan vojon rilate al la nuna dosierujo. Tiu komando ne funkcias per absoluta vojo.

paĝon_registru *vor1*

`paĝon_registru "provo.lgo` konservas en la dosieron `provo.lgo` de la nuna dosierujo ĉiujn procedurojn nun difinitajn. Se la finaĵo `.lgo` forestas, oni aldonas ĝin apriore. La vorto indikas relativan vojon rilate al la nuna dosierujo. Tiu komando ne funkcias per absoluta vojo.

eldonu *arg1*

Malfermu en la redaktilo ĉiun proceduron kies nomo estas indikita en la listo *arg1* aŭ la vorto *arg1*.

ĉion_eldonu

Malfermu en la redaktilo ĉiun proceduron nun difinitajn.

ramenu *vor1*

Malfermu kaj interpretu la dosieron *vor1*. Por ekzemplo, por forviŝi ĉiun proceduron difinitan kaj ŝargi la dosieron `provo.lgo`, skribu: `nv progcit ramenu "provo.lgo`. La vorto indikas relativan vojon rilate al la nuna dosierujo. Tiu komando ne funkcias per absoluta vojo.

flukson_malfermu, flumf *id dosiero*

Kiam oni volas legi el aŭ skribi al dosiero, necesas antaŭe malfermi fluon al tiu dosiero. L' argumento *dosiero* estu la nom' de la koncerna dosiero. Oni uzu vorton por indiki la nomon de la dosier' en la nuna dosierujo. L' argumento `id` estas la numero kiun oni donu al tiu fluo por povi identigi ĝin.

flulist, fluksliston

Donu la liston de la malfermitaj fluoj kun iliaj identigiloj.

flulinleg, flukslinion_legu *id*

Malfermu la fluon kies identigilo estas la numero *id*, poste legu linion en tiu dosiero.

flulitleg, fluksliteron_legu *id*

Malfermu la fluon kies identigila numero estas tiu pasigita kiel argumento, poste legu signon (literon) en tiu dosiero. Tiu primitivo redonas nombron reprezentas la valoron de la signo (simile al `litleg`).

flulins, flukslinion_skribu *id listo2*

Skribu la tekstan linion enhavatan en la listo je la komenco de la dosiero indikita de la identigilo *id*. Atentu, la skribado ne estas efektiva ĝis oni fermos la fluon per la primitivo `fluf`.

flulinald, flukslinion_aldonu *id listo2*

Skribu la tekstan linion enhavatan en la listo ĉe la finon de la dosiero indikita de l' identigilo *id*. Atentu, la skribado ne estas efektiva ĝis oni fermos la fluon per la primitivon `fluf`.

fluf, flukson_fermu *id*

Fermu la fluon kies identigila numero estas tiu pasigita en argumento.

flufin?, fluksfine? *id*

Redonu `"vera` se oni alvenis al la dosierfino. Redonu `"malvera` se ne.

Jen ekzemplo uzi primitivojn ebligantajn legi kaj skribi en dosiero. Ĝi estas prezentota por arĥitekturo Vindoza. Alispecaj uzuloj adaptu l' ekzemplon.

La celo estas krei la dosieron `c:\ekzemplo` enhavantan la tri liniojn:
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789

```
# Malfermu fluon al dezirata dosiero.  Al ĝi rilatigu la numeron 2
dosierujon_provizu "c:\\
fluml 2 "ekzemplo
# Skribu la deziratajn liniojn
flulins 2 [ABCDEFGHIJKLMNOPQRSTUVWXYZ]
flulins 2 [abcdefghijklmnopqrstuvwxyz]
flulins 2 [0123456789]
# Fermu la fluon por fini skribi
fluf 2
```

Nun oni povas konstati ĉu la skribado funkciis:

```
# Malfermu fluon al la dosiero legota.  Tiu fluo rilatos al la numero 0
flumf 0 "c:\\ekzemplo
# Legu la liniojn de la dosiero sinsekve
s flulinleg 0
s flulinleg 0
s flulinleg 0
# Fermu la fluon
fluf 0
```

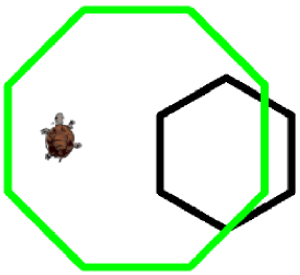
Se oni deziras nun aldoni la linion «Grandioze!»:

```
dosierujon_provizu "C:\\
flumf 1 "ekzemplo
flulinald 1 [Grandioze!]
fluf 1
```

A.9 Plenigi per koloro

Ekzistas du primitivoj ebligantaj kolori formon: La primitivo `plenigu` kaj la primitivo `kovru`. Oni pripensu tiujn primitivojn rilataj kun la funkcio "farboskatolo" aŭ "farbositelo" uzata en multaj bildoredaktaj softvoj. Oni plenigas je koloro, oni povas atingi la limojn de la desegnejo. Estas du reguloj observendaj por ĝuste uzi tiujn primitivojn:

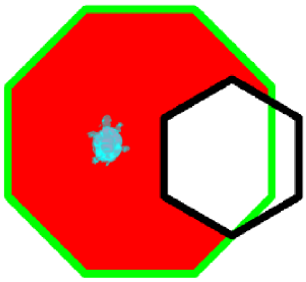
1. La krajono devas esti mallevita (`ml`).
2. La testudo ne estu sur pikselo (rastrumero) je la sama koloro je kiu oni volas plenigi la formon (se oni volas kolori je ruĝa, oni ne loku sin mem sur ruĝaĵo...).



Figuro A.1: Komenca situacio

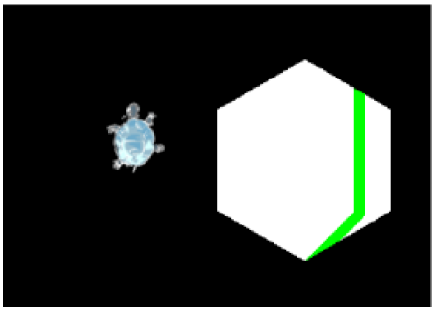
Rigardu ekzemplon por klarigi la diferencon inter `plenigu` kaj `kovru`:

La pikselo sub la testudo estas nun je blanka koloro. La primitivo `plenigu` farbos ĉiun najbaran blankan pikselon je la nuna krajona koloro. Se ekzemple oni tajpas: `skolp 1 plenigu`



Figuro A.2: Per primitivo `plenigu`

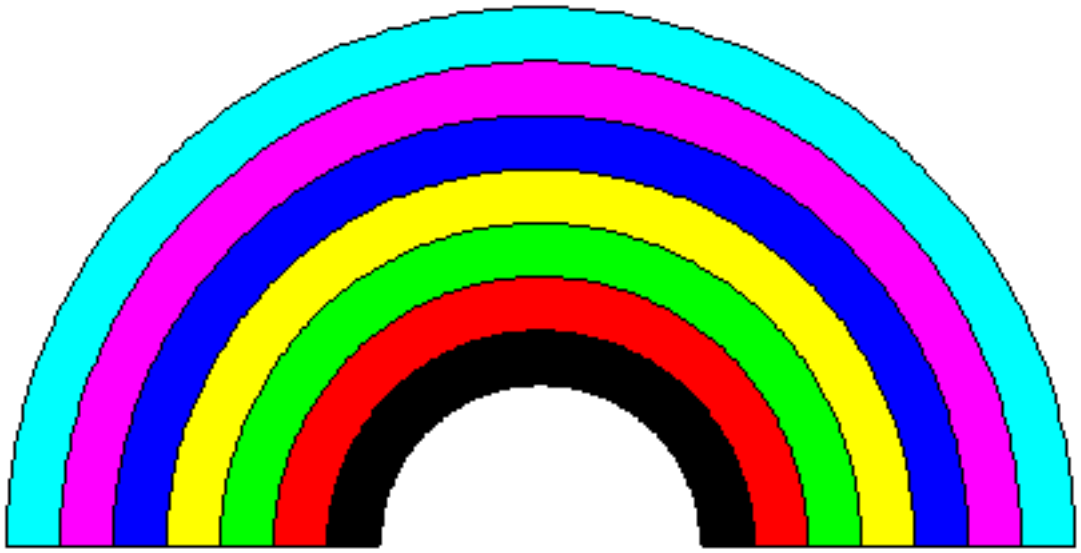
Revenu al la unua okazo. Se la krajona koloro estas nigra, la primitivo `kovru` farbas ĉiun pikselon najbaran ĝis trovi la nunan koloron (ĉi-okaze nigra).



Figuro A.3: Per primitivo `kovru`, tajpante: `skolp 0 kovru`

Jen bela ekzemplo uzi primitivon `plenigu`:

```
por duonci :c
# grafiku duoncirklon je diametro :c
ripetu 180 [an :c * tan 0.5 dn 1]
an :c * tan 0.5
dn 90 an :c
fino

por ĉielarko :c
se :c<100 [haltu]
```



Figuro A.4: LOGOarko

```
duonci :c td 180 an 20 mdn 90
ĉielarko :c-40
fino

por mov
l dn 90 an 20 mdn 90 ml
fino

por komenci
tdk ĉielarko 400 gum mdn 90 an 20 man 120 dsg l dn 90 an 20 ml
skolp 0 plenigu mov
skolp 1 plenigu mov
skolp 2 plenigu mov
skolp 3 plenigu mov
skolp 4 plenigu mov
skolp 5 plenigu mov
skolp 6 plenigu mov
fino
```

A.10 Instrukcioj por saltoj

XLogo havas tri instrukciojn por salto: `haltu`, `ĉion_haltu` kaj `sendu`.

haltu

`haltu` povas havi du efikojn: Se ĝi estas en buklo `ripetu` aŭ `dum`, tiam oni eliras el la buklo. Se ĝi estas en proceduro, oni eliras tuj el la proceduro.

ĉion_haltu

`ĉion_haltu` haltigas definitive la ruladon de ĉiu kuranta proceduro.

sendu, snd *arg1*

`sendu` ebligas eliri el proceduro redonante la valoron *arg1*.

A.11 La plurtestuda moduso

Eblas direkti sur l' ekrano plurajn testudojn samtempe. Apriore, kiam oni ekrulas XLogo, nur unu testudo estas aktiva. Ĝia identiga numero estas 0. Por "krei" novan testudon sur l' ekrano, uzu la primitivon `tdp` aŭ `testudon_provizu` sekvata de la numero de la testudo dezirata. Por malebligi tohuvabohuon sur la desegnejo, la nova testudo estos kreita je l' origino, tio estas, je koordenatoj $(0; 0)$ kaj ĝi estos nevidebla, tio estas ke necesos uzi la komandon `tdm` por aperigi ĝin. Sekve tiu nova testudo obeas l' ordonojn klasikajn ĝis oni ŝanĝos la testudon per l' ordono `testudon_provizu`. La maksimuma nombro de testudoj haveblaj estas agordebla en Agordaj iloj - Preferoj - Langeto elektebloj.

Jen la listo de primitivoj koncernantaj la plurtestudan moduson:

testudon_provizu, tdp *n*

Faru ke la aktiva testudo estu tiu kun numero *n*. Apriore, la unua aktiva testudo post ekruli XLogo havas la numeron 0.

testudon, td

Donu la numeron de la nun uzata testudo.

testudojn, tdj

Donu liston konsistantan el ĉiu numero de la testudoj nun uzataj.

testudon_buĉu, tdb *n*

Forigu de l' ekrano la testudon kun numero *n*.

tdkiomp, testudkiomon_provizu *n*

Agordu la maksimuman nombron de testudo sur l' ekrano en plurtestuda moduso.

tdkiom, testudkiomon

Donu la maksimuman nombron de testudoj sur l' ekrano en plurtestuda moduso.

A.12 Ludi muzikon

sekvencon, sek *listo1*

Metu en memoron la muzikan sinsekvon lokita en la listo. Por lerni pri muzikaj sinsekvoj, rigardu l' instrukciojn post la tabelo.

muziku

Ludu la sinsekvon nun metitan en memoron.

instrumenton, instr

Donu la numeron koncernan al l' instrumento nun elektita.

instrumenton_provizu, instrp *n*

Elekti l' instrumenton kun numero *n*. Vi povas rigardi la liston de instrumentoj haveblaj en menu' Agordaj iloj - Preferoj - Langeto sono (se ne estas problemoj detekti l' interfacon MIDI).

sekvencindekson, sekvind

Donu kie estas lokita la kursoro en la nuna sinsekvo.

sekvencindekson_provizu, sekvindp *n*

Movu la kursoron al la n-an takton en la muzika sinsekvo nun en memoro.

sekvencon_viŝu, sekvv

Forviŝu la sinsekvon nun en memoro.

Por ludi muzikon, necesas antaŭe meti la deziratan komponaĵon en ĉi tie tiel nomatan muzikan sinsekvon. Kreu la sinsekvon per la komando `sek` aŭ `sekvencon`. Jen kelkaj reguloj observendaj por taŭge skribi muzikan sinsekvon:

`do re mi fa sol la si`: indikas la kutimajn notojn de la unua oktavo (C D E F G A B).

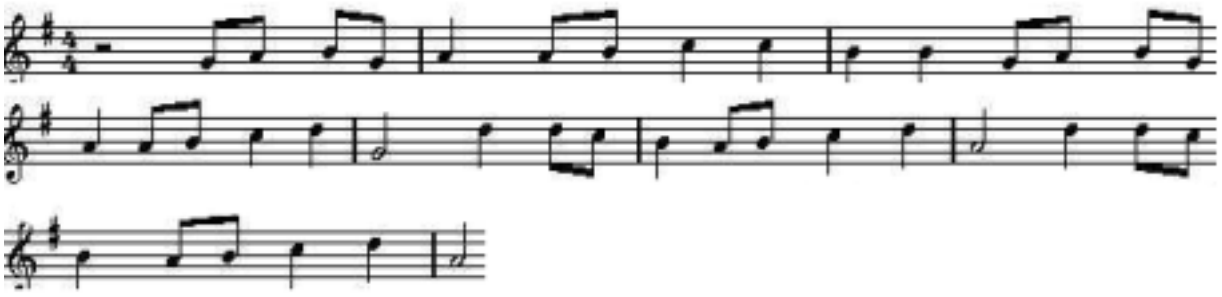
Por fari diesan D (re-tono), tajpu `re +`

Por fari bemolan D, tajpu `re -`

Se oni volas ŝanĝi la oktavon, uzu la simbolon ":" sekvatan de "+" aŭ "−". Por ekzemplo, post tajpi `:++`, ĉiu noto ludota estos plialtigita je du oktavoj (estas du "+").

La notojn oni ludos apriore dum daŭro de unu kvaronnoton. Se oni volas ŝanĝi la daŭron de kelkaj notoj, tion indiku per nombro indikantan la deziratan daŭron. Por tajpi kelkajn okonnotojn (1/2 kvaronnoto), tajpu `sek [0.5 sol la si]`.

Bona ekzemplo valoras pli ol mil klarigoj:



```
por tabako
# Metu en memoron la partituron
sekvenco [0.5 sol la si sol 1 la 0.5 la si 1 :+ do do :- si si 0.5 sol la si sol
          1 la 0.5 la si 1 :+ do re 2 :- sol ]
sekvenco [:+ 1 re 0.5 re do 1 :- si 0.5 la si 1 :+ do re 2 :- la ]
sekvenco [:+ 1 re 0.5 re do 1 :- si 0.5 la si 1 :+ do re 2 :- la ]
sekvenco [0.5 sol la si sol 1 la 0.5 la si 1 :+ do do :- si si 0.5 sol la si sol
          1 la 0.5 la si 1 :+ do re 2 :- sol ]
fino
```

Por ruli la muzikon, nur restas tajpi: `tabak muziku`.

Jen nun interesa aplikado de la primitivo `sekvindp`. Tajpu la komandojn:

```
sekvv       # Forviŝu la sinsekvon nun en memoro
tabako      # Reŝargu la antaŭan muzikon
sekvindp 2  # Remetu la kursoron je la nivelo de la unua nigra "la" de la dua mezuro
tabako      # Reŝargu la saman sinsekvon sed prokrastita je du taktoj
muziku      # Grandioza kanono!
```

Vi povas ankaŭ ŝanĝi l' instrumenton, jen per la komando `instrp`, jen en la menu' Agordaj iloj – Preferoj – Langeto Sono. Vi trovos la liston de ĉiu havebla instrumento kun ĝia numero (eble en la angla, sed tio ebligas idei; ĉe mi, 411 haveblaj instrumentoj!).

A.13 Bukloj

XLOGO havas kvin primitivojn ebligantajn efektivigi buklojn: `ripetu`, `ripetupor` kaj `dum`, `por_ĉiu`, `ĉiam_ripetu`.

A.13.1 Buklo kun `ripetu`

ripetu *n listo_de_instrukcioj*

n estas entjero kaj `listo_de_instrukcioj` estas listo enhavanta instrukciojn rulotajn. L' interpretilo LOGO efektivigos je *n* fojoj la komandojn enhavatajn en la listo: tio ŝparas reskribi *n* fojojn la saman instrukcioj!

Ekz:

```
ripetu   4 [antaŭen 100 maldekstren 90]    # Kvadrato kun latero 100
ripetu   6 [antaŭen 100 maldekstren 60]    # Seslatero kun latero 100
ripetu 360 [antaŭen   2 maldekstren  1]    # Ee... 360-latero kun latero 2
                                           # Resume, preskaŭ cirklo!
```

nombrilon

En buklo `repete`, estas difinita interna variablo `nombrilon`. Tiu enhavas la numero de l' iteracio kuranta (la unua iteracio havas numeron 1).

```
ripetu 3 [s nombrilon]
1
2
3
```

A.13.2 Buklo kun `ripetupor`

`ripetupor` ludas la rolon de la bukloj `for` en aliaj programlingvoj.

ripetupor *listo1 listo2*

Tiu buklo konsistas el doni al variablon kelkajn valorojn en iu intervalo laŭ iu kreskokvanto.

listo1 enhavas tri parametrojn: la nomon de la variablo, la komencan limon, la finan limon. Oni povas aldoni kvaran argumenton nenepran indikantan la kreskokvanton (la paŝon laŭ kiu la variablo marŝas); se ĝi forestas, apriore valoras 1. Jen kelkaj uzadaj ekzemploj:

```
ripetupor [i 1 4] [s :i*2]
2
4
6
8

# Nun oni variigas i inter 7 kaj 2 malkreskante je 1.5 je ĉiu fojo
# Rimarku la negativan kreskokvanton
# Oni skribas post i ĝian kvadraton

ripetupor [i 7 2 -1.5] [s listo :i potencon :i 2]
7 49
5.5 30.25
4 16
2.5 6.25
```

A.13.3 Buklo kun `dum`

dum *listo_testota listo_de_instrukcioj*

listo_testota estas listo enhavanta instrukciojn redonantajn bulean.

listo_de_instrukcioj estas listo enhavanta rulotajn instrukciojn. L' interpretilo LOGO rulos refoje *listo_do_instrukcioj* dum *listo_testota* redonos "vera".

Ekz:

```
dum ["vera] [dn 1]                    # Testudo turnu sin

# Ekzemplo por skribi renversitan alfabeton

provizu "listo "abcĉdefgĝhĥijĵklmnoprsŝtuŭvz
dum [ne malplena? :listo] [s lastan :listo provizu "listo senlastan :listo]
```

A.13.4 Buklo kun `por_ĉiu`

por_ĉiu *nomon_variablan listo_aŭ_vorto komando*

Tiu primitivo ebligas priskribi ĉiun eron el listo aŭ ĉiun signon el vorto, poste rulas je ĉiu fojo la enhavon de la komandolisto.

```
por_ĉiu "i "XLOGO [skribu :i]
X
L
O
G
O
por_ĉiu "i [a b c] [skribu :i]
a
b
c
```

A.13.5 Buklo kun `ĉiam_ripetu`

ĉiam_ripetu *instrukcilisto*

Ripetu sen fino instrukciliston.

```
ĉiam_ripetu [an 1 dn 1]
```

Atentu: uzu tiun primitivon prudente pro la senfina buklo!

A.14 Interkapti la uzulajn agojn

XLOGO povas interagi kun la uzulo dum la rulado de programo, per klavaro kaj muso.

A.14.1 Interago kun la klavaro

Oni povas do ricevi tekston de l' uzulo dum ruli la programon, per 3 primitivoj: `klave?`, `litleg` kaj `leg`.

klave?

Donu "vera" aŭ "malvera" laŭ ĉu oni premis klavon aŭ ne post la komenco de ruli la programon.

litleg

- Se `klave?` estas malvera, haltigu la programon ĝis l' uzulo premos klavon.

A $\Longrightarrow$ 65	B $\Longrightarrow$ 66	C $\Longrightarrow$ 67	ktp...	Z $\Longrightarrow$ 90
$\leftarrow \Longrightarrow -37$ aŭ -226 (NumKla)	$\uparrow \Longrightarrow -38$ aŭ -224	$\rightarrow \Longrightarrow -39$ aŭ -227	$\downarrow \Longrightarrow -40$ aŭ -225	
Esk $\Longrightarrow$ 27	F1 $\Longrightarrow -112$	F2 $\Longrightarrow -113$	...	F12 $\Longrightarrow -123$
Uskl $\Longrightarrow -16$	Spaco $\Longrightarrow$ 32	Stir $\Longrightarrow -17$	Enig $\Longrightarrow$ 10	

Tabelo A.2: Kelkaj valoroj de klavoj

- Se `klave?` estas vera, donu la valoron koncernan al la klavo laste premita.

Se vi havas dubon pri la vorto redonata de klavo, sufiĉas tajpi: `s litleg`. L' interpretilo tiam atendos ke vi premos klavon, poste donos la rilatan valoron.

leg, legu *listo1 vor2*

Afiŝu dialogfenestron kies titro estu *listo1*. L' uzulo povas tiam enigi respondon en tekstokampo; la respondon oni konservos kiel vorton aŭ liston (se l' uzulo tajpos plurajn vortojn) en la variablon *vor2* kiam ŝli validigos aŭ klakos la butonon Akceptu.

A.14.2 Kelkaj ekzemploj uzi

```
pour juna
 leg [Kiom vi aĝas?] "aĝo
 provizu "aĝo :aĝo
 se :aĝo<18 [s [Vi estas malplenkreskulo]]
 se aŭ :aĝo=18 :age>18 [s [Vi estas plenkreskulo]]
 se :aĝo>99 [s [Mi respektu!!]]
fino

por ralio
    se klave?
        [provizu "sig litleg
         si :sig=-37 [mdn 90]
         si :sig=-39 [dn  90]
         si :sig=-38 [an  10]
         si :sig=-40 [man 10]
         si :sig=27  [haltu]]
    ralio
fino
# Kontrolu la testudon per la klavaro, haltigu per Esk
```

A.14.3 Interkapti iujn musajn eventojn

Por tio, oni havas tri primitivojn: `musleg`, `muse?` kaj `mussit`.

musleg, muson_legu

Haltigu la programon ĝis musa evento okazas. Estas museventoj: movi la muson aŭ klaki iun butonon ĝian. Okazinte l' evento, `musleg` donas nombron ebligantan karakterizi l' eventon. Jen la diversaj kodoj asociitaj al la diversaj eventoj:

- 0 $\rightarrow$ oni movis la muson.
- 1 $\rightarrow$ oni klakis la butonon 1 de la muso.
- 2 $\rightarrow$ oni klakis la butonon 2 de la muso.

La butonoj estas numeritaj de maldekstre al dekstre (principe...).

mussit, mussituon

Donu liston enhavantan la koordinatojn de la muso dum la lasta interkaptita evento.

muse?

Donu "vera" aŭ "malvera" laŭ ĉu oni agis aŭ ne per la muso post la komenco ruli la programon.

A.14.4 Kelkaj uzekzemploj

En tiu unua proceduro, la testudo sekvas la muson kiam ĝi moviĝas sur la desegnejo.

```
por ekzemplo1
# Se oni movas la muson, loku sin al la novan situon
se musleg=0 [sitp mussit]
ekzemplo1
fino
```

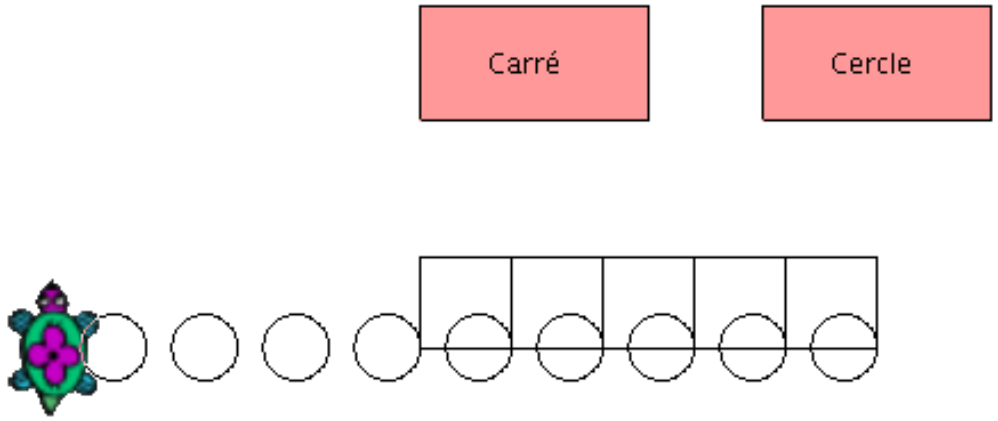
En tiu dua proceduro, estas la sama principo krom ke necesas klaki la maldekstran butonon musan por movi la testudon sur la desegnejo.

```
por ekzemplo2
se musleg=1 [sitp mussit]
ekzemplo2
fino
```

En tiu tria ekzemplo, ni kreos du butonojn. Tiu maldekstra ebligos grafiki kvadraton je 40 mul 40 al la dekstro; tiu dekstra, malgrandan cirklon al la maldekstro. Finfine, se oni klakos la trian butonon de la muso, la programo haltos.

```
por butono
# kreu ortangulan butonon je 50 mul 100 farbita je salmakoloro
ripetu 2 [an 50 dn 90 an 100 dn 90]
dn 45 l an 10 ml skolp [255 153 153]
plenigu mal 10 mdn 45 ml skolp 0
fino

por lanĉu
ev butono l skolp [150 0] ml butono
l skolp [ 30   20] ml etikedu "Kvadrato
l skolp [180   20] ml etikedu "Cirklo
l skolp [  0 -100] ml
muso
fino

por muso
# Konservu la rezulton de musleg en la variablon ev
provizu "ev musleg
# Konservu la unuan koordinaton de la muso en la variablon x
provizu "x eron 1 mussit
# Konservu la duan koordinaton de la muso en la variablon y
provizu "y eron 2 mussit
# Se oni klakus la maldekstran butonon
se :ev=1 & :x>0 & :x<100 & :y>0 & :y<50 [kvadrato]
# Se oni klakus la dekstran butonon
```

```
se :x>150 & :x<250 & :y>0 & :y<50
         [si :ev=1 [cirklo]
          si :ev=3 [haltu]]
muso
fino

por cirklo
ripetu 90 [an 1 mdn 4] mdn 90 l  an 40 dn 90 ml
fino

por kvadrato
ripetu 4 [an 40 dn 90] dn 90 an 40 mdn 90
fino
```



A.14.5 Uzi grafikajn konsistaĵojn

XLogo ebligas ankaŭ aldoni kelkajn grafikajn konsistaĵojn (butonon, malvolveblan menuon...) al la desegnejo. Ĉar tiaj konsistaĵoj rilatas al grafikaj uzulaj interfacoj, ĉiu primitivo por tiu afero komenciĝas per la prefikso «GUI».

Krei konsistaĵon

Por manipuli tiajn grafikajn objektojn, antaŭ ĉio necesas krei ilin, aldoni al ili iujn atributojn, kaj poste afiŝi ilin.

- Por krei butonon:

  gui_butonon *vor1 vor2*

  Tiu komando kreas butonon kies identiga nomo estas *vor1* kaj sur kiu estas skribita *mot2*.

  Ekzemplo: `gui_butonon "b "Klaki`

- Por krei malvolveblan menuon:

  gui_menuon *vor1 listo2*

  Tiu komando kreas menuon kies nomo estas *vor1* enhavantan l' erojn de la listo *listo2*

  Ekzemplo: `gui_menuon "m [ero1 ero2 ero3]`

Atribui atributojn al konsistaĵo

gui_koordinatojn *vor1 listo2*

Ebligas loki la grafikan elementon sur la deziratan situon en la desegnejo. Ekzemple, por loki la antaŭan butonon al punkto kun koordinatoj (20; 100), skribu:

```
gui_koordinatojn "b [20 100]
```

Se la situo de la konsistaĵo ne estas indikita, la konsistaĵo estos lokita apriore je la supra maldekstra angulo de la desegnejo.

gui_forigu *vor1*

Forviŝu grafikan elementon. Ekzemple, por forigi la antaŭan butonon:

```
gui_forigu "b
```

gui_agadon *vor1 listo2*

Difinu agadon realigendan kiam l' uzulo interagos kun la grafika elemento konsiderita.

```
# La testudo anta^uen iru 100 paŝojn se oni klakos butonon "b
gui_agadon "b [an 100]

# Por la malvolvebla menuo, ^ciu ero havas sian propran agon
gui_agadon "m [[skrbiu "ero1]  [skribu "ero2] [skribu "ero3]]
```

gui_desegnu *vor1*

Afiŝu la grafikan konsistaĵn sur la desegnejon. Ekzemple, por montri la butonon:

```
gui_desegnu "b
```

A.15 Administri la tempon

XLogo havas plurajn primitivojn ebligantajn koni la horon, la daton aŭ ankaŭ administri nombradojn (utilaj por ripetu taskon laŭ fiksitaj intervaloj).

atnd, atendu *n*

Haltu la programon kaj do la testudon dum n 60^{onoj} de sekundo.

tmpko, tempokomencon *n*

Komencu nombri n sekundojn. Oni povas scii ĉu la nombrado estas finita per la primitivo `tmpfi`.

tmpfi, tempofine?

Donu `"vera` se neniu nombrado estas aktiva. Donu `"malvera` se la nambrado ne estas finita.

daton

Redonu liston konsistantan el tri entjeroj prezentantaj la daton. La unuo indikas la tagon. La dua la monaton. La tria la jaron. $\Longrightarrow$ [tago monato jaro]

horon

Donu liston kun tri entjeroj prozentantaj la horon. La unua prezentas la horojn, la dua la minutojn kaj la lasta la sekundojn. $\Longrightarrow$ [horo minuto sekundo]

tmp, tempon

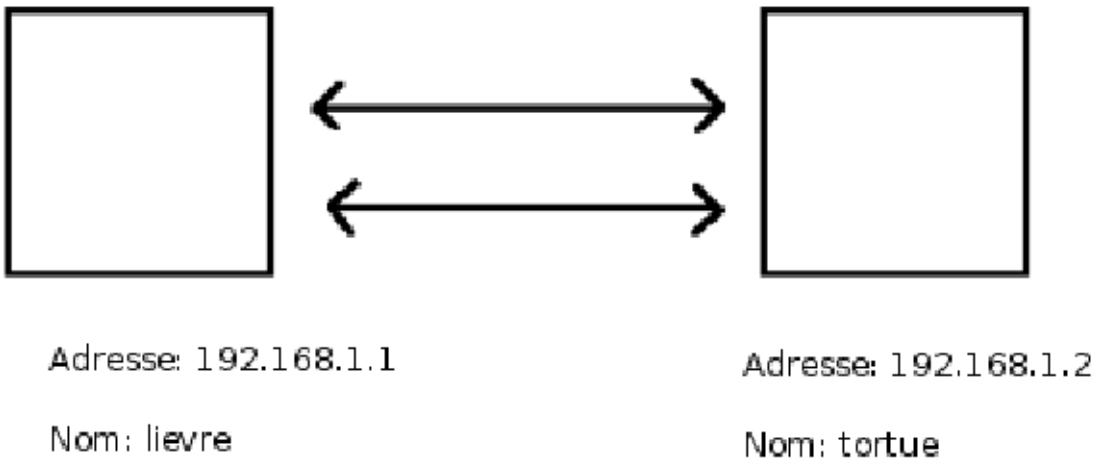
Donu la tempon pasintan de post la starto de XLogo. Tiu tempo estas esprimata en sekundoj.
Jen malgranda proceduro ekzemplo:

```
por horloĝo
# afiŝu la horon en formo cifera
# (ĝisdatigu l' afiŝadon je ĉiu 5 sekundoj)
se tmpfi
  [ev
   tiparon\_provizu 75
   tdk
   provizu "hor horon
   provizu "h unuan :hor
   provizu "m er 2 :hor
   # afiŝi je du ciferoj la minutojn (oni aldonas la 0)
   se :m-10 < 0 [p "m vort 0 :m]
   p "s lastan :hor
   # afiŝi je du ciferoj la sekundojn
   se :s-10 < 0 [p "s vort 0 :s]
   etikedu vort vort vort vort :h ": :m ": :s
   tmpko 5]
horloĝo
fino
```

A.16 Uzi la reton kun XLogo

A.16.1 La reto: kiel ĝi funkcias?

Antaŭ ĉio, en ĉi tiu enkonduko, necesas klarigi kelkajn konceptojn por bone kompreni la uzadon de la primitivoj.



Figuro A.5: Nocio de reto

Du komputiloj povas komuniki tra la reto se ili havas retkarton (ethernet) aŭ similan rimedon. Al ĉiu komputilo oni donas personan adreson: *ĝia adreso IP*. Tiu adreso IP konsistas el 4 entjeroj inter 0 kaj 255,

disigitaj de punktoj. Ekzemple, l' adreso IP de la unua komputilo en la antaŭa skemo estas 192.168.1.1.

Ĉar ne facilas memori tiajn adresojn, eblas ankaŭ rilatigi al ĉiu adreso IP nomon pli kutiman pli facile memoreblan. Sur la antaŭa skemo, oni povas adresi sin al la dekstra komputilo jen vokante ĝin per ĝia IP-adreso 192.168.1.2, jen vokante ĝin per ĝia nomo `tortue`.

Mi ne parolos pli pri la signifojn de tiuj nombroj. Mi aldonos nur ion bonan por scii: la loka komputilo sur kiu oni laboras havas ĉiam specifan IP-adreson, 127.0.0.1 (krom eble alia aŭ aliaj IP-adresojn); ĝi havas specifan nomon, ofte `localhost` (krom eble alia aŭ aliaj nomoj).

A.16.2 Porretaj primitivoj

XLogo havas 4 primitivojn ebligantajn komuniki per reto: `tcp_aŭskultu`, `ekzekucutcp`, `diskutilotcp` kaj `sendutcp`. Por la sekvaj ekzemploj konsideru ĉiam la okazo de la du komputiloj de la antaŭa skemo.

tcp_aŭskultu, tcp_auskultu, tcp_awskultu, tcp_auxskultu

Ĝi estas la bazo de ĉiu retkomunikado. Ĝi ekspektas neniun argumenton. Ĝi ebligas ke komputilo rulanta ĝin aŭskultu ordonojn donitajn de aliaj komputiloj en la sama reto.

ekzekucutcp *vor1 listo2*

Tiu primitivo ebligas ruli instrukciojn sur iu komputilo en la reto.

vor1 indikas la IP-adreson aŭ la nomon de la vokata komputilo, *listo2* enhavas la rulotajn instrukciojn.

Ekzemple: Mi estas sur la komputilo `lievre`, mi deziras grafiki kvadraton kun latero 100 sur l' alia komputilo. Tial, necesas ke sur la komputilo `tortue` mi rulu la ordonon `tcp_aŭskultu`; poste, sur la komputilo `lievre`, mi rulu:

```
ekzekucutcp "192.168.1.2 [ripetu 4 [an 100 dn 90]]
```

ou

```
exekucutcp "tortue [ripetu 4 [an 100 dn 90]]
```

diskutilotcp *vor1 listo2*

Ĝi ebligas dialogi inter du komputiloj de la reto, afiŝante fenestron ebligantan la interparolon.

vor1 indikas la IP-adreson aŭ la nomon de la vokita komputilo, *listo2* enhavas la frazon afiŝotan.

Ekzemple: `lievre` volas diskuti kun `tortue`.

`tortue` rulu `tcp_aŭskultu` por meti sin en atendon de peto far komputiloj en la reto. `lievre` rulu tiam: `diskutilotcp "192.168.1.2 [saluton]`.

Du fenestroj ebligantajn la dialogon malfermiĝas tiam sur ĉiu komputilo.

sendutcp *vor1 listo2*

Sendu datumojn al komputilo de la reto, poste donu la respondon de la alia komputilo.

vor1 indikas la IP-adreson aŭ la nomon de la komputilo vokata, *listo2* enhavas la datumojn sendotajn. Se la komunikado fariĝos kun alia komputilo kie XLogo ruliĝas, tiu komputilo respondos OK post fini l' operacion. Eblas ankaŭ dialogi kun roboto havanta retan interfacon, sed la respondo povos esti malsama tiam.

Ekzemple:

`tortue` volas sendi al `lievre` la sinsekvon "3.14159 preskaŭ la nombro pi".

`lievre` rulu `tcp_aŭskultu` por atendi peton far komputiloj de la reto. `tortue` rulu tiam: `skribu sendutcp "lievr preskaŭ la nombro pi]`.

Jen konsileto: Ekrulu du fojojn XLogo sur la sama komputilo.

- En la unua fenestro, rulu `tcp_aŭskultu`.
- En la dua, rulu `ekzekucutcp "127.0.0.1 [an 100 dn 90]`

Vi tiel movis la testudon sur l' alian fenestron!

Apendico B

Ekruli XLogo en komandlinio

Jen la sintakso de la komando tajpenda por ekruli XLogo:

```
java -jar xlogo.jar [-a] [-lang eo] [-memory 64][dosiero1.lgo dosiero2.lgo ...]
```

Jen detaloj de la diversaj elektebloj:

- Elekteblo `-lang`: ĝi ebligas indiki homan lingvon por XLogo. Tiu parametro superregas tiun enhavatan en la persona agorda dosiero nomata `.xlogo`. La lingvojn oni povas elekti laŭ tiu tabelo:

Franca	Angla	Hispana	Germana	Araba	Portugala	Esperanto	Galega	Greka
fr	en	es	de	ar	pt	eo	gl	el

- Elekteblo `-a`: ĝi ebligas ruli ekde la malfermo de XLogo la ĉefan komandon enhavatan en la dosierojn ŝargitajn je la starto.
- Elekteblo `-memory`: ĝi ebligas establi la memoron rezervita por la virtuala maŝino Java.
- dosiero1.lgo, dosiero2.lgo ...: tiuj dosieroj kun finaĵo `.lgo` estas ŝargataj je la starto de XLogo. Tiuj dosieroj povas esti lokaj aŭ foraj, tio estas, ilia adreso povas indiki vojon en la loka hierarĥia arbo de dosierujoj aŭ interretan adreson.
- Elekteblo `tcp_port`: ĝi ebligas elekti pordan numeron por la reta komunikado. Apriora pordo estas 1948. Rigardu p. 121.

Jen ekzemploj de komandoj:

- `java -jar xlogo.jar -lang es prog.lgo`: La dosieroj `xlogo.jar` kaj `prog.lgo` estas en la nuna dosierujo. Tiu komando ekrulas XLogo agordita en la hispana kaj ŝargas tuj poste la dosieron `prog.lgo` (kiu do devas esti redaktita en la hispana...).
- `java -jar xlogo.jar -a -lang en http://xlogo.tuxfamily.org/prog.lgo`: Tiu komando rulas XLogo agordita en la angla kaj ŝargas la dosieron nomatan `http://xlogo.tuxfamily.org/prog.lgo`. Por fini, la ĉefa (startiga) komando difinita en tiu dosiero estas rulota je la starto.

Apendico C

Ekruli XLogo disde la reto

Vi havas retpaĝon sur kiu vi parolas pri XLogo. Eĉ pli bone: vi deziras havigi iujn programojn kiujn vi verkis. Anstataŭ simple distribui la dosierojn `.lgo`, estus pli agrable por l' uzulo povi ruli XLogo enrete por provi rekti tiujn ekzemplojn. Jen la sekvenda proceduro:

La enretan ruleblon de XLogo certigas la teĥnologio JAVA WEB START. Efektive, sufiĉas meti en vian retpaĝon ligilon al dosiero kun finaĵo `.jnlp`; tio certigas la ruladon de XLogo.

Krei dosieron kun ligilo `jnlp`

Jen ekzemplo de tia dosiero. Tiu dosiero estas efektive tiu uzata en la sekcio «exemples» de la franca retpaĝo. Ĝi ebligas ŝargi la programon grafikantan la ludkubon en la sekcio pri 3D. La grandaj linioj por klarigi aperas poste.

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.5+" codebase="http://downloads.tuxfamily.org/xlogo/common/webstart">
<information>
  <title>XLogo</title>
  <vendor>xlogo.tuxfamily.org</vendor>
  <homepage href="http://xlogo.tuxfamily.org"/>
  <description>Logo Programming Language</description>
  <offline-allowed/>
</information>

<security>
 <all-permissions/>
</security>

<resources>
  <j2se version="1.4+"/>
  <jar href="xlogo.jar"/>
</resources>

<application-desc main-class="Lanceur">
  <argument>-lang</argument>
  <argument>fr</argument>
  <argument>-a</argument>
  <argument>http://xlogo.tuxfamily.org/fr/html/examples-fr/3d/de.lgo</argument>
</application-desc>
</jnlp>
```

Tiu dosiero estas skribita observante la formaton XML. La grava parto estas je la fino, ĉefe tiuj 4 linioj:

```
  <argument>-lang</argument>
```

```
<argument>fr</argument>
<argument>-a</argument>
<argument>http://xlogo.tuxfamily.org/fr/html/examples-fr/3d/de.lgo</argument>
```

Ja tie oni indikas la ekrulajn parametrojn.

- La du unuaj linioj devigas uzi la francan lingvon.
- La lasta linio indikas la adreson de la ŝargota dosiero.
- La tria linio indikas ke la startiga komando de tiu dosiero estas rulota je la starto de XLogo.

Lasta konsileto: Se vi deziras ne troŝargĝi la servilon de Tuxfamily, vi povas meti la dosieron `xlogo.jar` sur vian servilon. Por ligi la dosieron `.jnlp` al tiu dosiero, sufiĉus ŝanĝi l' adreson en la dua linio, post `codebase=`.

Apendico D

Korektaĵoj de l' ekzercoj

D.1 Ĉapitro 5

```
por kvadrato
  ripetu 4 [an 150 dn 90]
fino

por tri
  ripetu 3 [an 150 dn 120]
fino

por pordo
  ripetu 2 [an 70 dn 90 an 50 dn 90]
fino

por kam
  an 55 dn 90 an 20 dn 90 an 20
fino

por mov1
  dn 90 an 50 mdn 90
fino

por mov2
  mdn 90 an 50 dn 90 an 150 dn 30
fino

por mov3
  l dn 60 an 20 mdn 90 an 35 ml
fino

por domo
  kvadrato mov1 pordo mov2 tri mov3 kam
fino
```

D.2 Ĉapitro 6

```
por superkubo
ev l sitp [-30 150] ml sitp [-150 150] sitp [-90 210] sitp [30 210] sitp [-30 150]
sitp [-30 -210] sitp [30 -150] sitp [30 -90] sitp [-30 -90] sitp [90 -90] sitp [90 30]
sitp [-270 30] sitp [-270 -90] sitp [-210 -90] sitp [-210 -30] sitp [-90 -30] sitp [-90 -150]
```

```
sitp [-210 -150] sitp [-210 -30] sitp [-150 30] sitp [-30 30] sitp [-90 -30] sitp [90 150]
sitp [30 150] sitp [30 210] sitp [30 90] sitp [90 90] sitp [90 150] sitp [90 90] sitp [150 90]
sitp [150 -30] sitp [90 -90] sitp [90 30] sitp [150 90] l sitp [-150 30] ml sitp [-150 150]
sitp [-150 90] sitp [-210 90] sitp [-270 30] l  sitp [-90 -150] ml sitp [-30 -90]
l sitp [-150 -150] ml sitp [-150 -210] sitp [-30 -210]
fino
```

D.3 Ĉapitro 7

D.3.1 La roboto

```
por ort :lo :la
# grafiku ortangulon je longo :lo kaj je larĝo :la
ripetu 2 [an :lo dn 90 an :la dn 90]
fino

por kvadrato :c
# grafiku kvadraton je latero :c
ripetu 4 [an :c dn 90]
fino

por tri :c
# grafiku trilateron egallateran je latero :c
ripetu 3[an :c dn 120]
fino

por piedo :c
ort 2*:c 3*:c kvadrato 2*:c
fino

por anteno :c
an 3*:c mdn 90 an :c dn 90 kvadrato 2*:c
l man 3 *:c dn 90 an :c mdn 90 ml
fino

por roboto :c
ev tdk
# La korpo
ort 4*:c 28*:c
# La piedoj
dn 90 an 2*:c piedo :c an 4* :c piedo :c an 14*:c piedo :c an 4*:c piedo :c
# La vosto
l mdn 90 an 4* :c ml dn 45 an 11*:c man 11 * :c mdn 135
# la kolo kaj la kapo
an 18 *:c kvadrato :c an 3*:c kvadrato :c dn 90 an :c mdn 90 an 2*:c dn 90 kvadrato 8*:c
# Oreloj
an 4*:c mdn 60 tri 3*:c l dn 150 an 8 *:c mdn 90 ml tri 3*:c
# La antenoj
an 4*:c mdn 90 an 2*:c dn 90 anteno :c mdn 90 an 4*:c dn 90 anteno :c
# la okuloj
l man 3*:c ml kvadrato :c dn 90 l an 3*:c ml mdn 90 kvadrato :c
# La buŝo
l man 3*:c mdn 90 an 3*:c dn 90 ml ort :c 4*:c
fino
```

D.3.2 La rano

```
por rano :c
ev tdk
an 2 *:c dn 90 an 5*:c mdn 90 an 4*:c mdn 90 an 7 *:c dn 90 an 7*:c dn 90
an 21 *:c dn 90 an 2*:c mdn 90 an 2*:c dn 90 an 9*:c dn 90 an 2*:c mdn 90
an 2*:c dn 90 an 9*:c dn 90 an 2*:c dn 90 an 7*:c man 5*:c mdn 90 an 4*:c
dn 90 an 4*:c man 4*:c mdn 90 man 2*:c mdn 90 an 5*:c mdn 90 an 4*:c dn 90 an 7*:c
dn 90 l an 9*:c ml ripetu 4[an 2*:c dn 90]
fino
```

D.4 Ĉapitro 10

```
por ludo
# Oni pretigas la nombron serĉotan kaj la nombro de provoj
provizu "nombro hazardon 32
provizu "nombrilo 0
buklo
fino

por buklo
leg [proponu nombron] "provo
se nombra? :provo
  [# Se la enigita valoro ja estas nombro
   se :nombro=:provo
      [s fr fr [vi gajnis post ] :nombrilo+1 [provo(j)]]
      [se :provo>:nombro
            [s [Malpli granda]]
            [s [Pli granda]]
       provizu "nombrilo :nombrilo+1
       buklo]]
  [skribu [Vi devas enigi validan nombron!] buklo]
fino
```

Apendico E

Oftaj demandoj – Konsiloj

E.1 Se mi forviŝas proceduron en la redaktilo, ĝi reaperas ĉiam!

Kiam oni eliras el la redaktilo, tiu limigas sin konservi aŭ ĝisdatigi la enhavon de la redaktilo. La sola rimedo forviŝi proceduron en XLogo estas uzi la primitivon `nomon_viŝu` aŭ `nv`.

Ekzemple: `nv "toto` ⟶ forviŝu la proceduron `toto`.

E.2 Mi uzas la esperantan version sed mi ne povas skribi la ĉapelitajn signojn!

Dum vi tajpas en la komandlinio aŭ la redaktilo, se vi premas la dekstran musbutonon, aperos ekmenuon. En tiu menu, aperas la tradiciaj redaktagoj (kopiu/enpoŝigu, fortranĉu, algluu/elpoŝigu) kaj la ĉapelitaj signoj de l' Esperanto, kiam tiun lingvon oni elektis.

E.3 En la langeto sono de la dialogfenestro Agordaj iloj, neniu instumento haveblas.

Kalkafoje, la listo de MIDI-instrumentoj ne aperas en `Agordaj iloj / Sono` kaj oni ne povas uzi ĉiel la funkciojn sonajn de XLogo. Adresu vin al:

`http://java.sun.com/products/java-media/sound/soundbanks.html`

Deŝutu unu el la sonbenkoj (soundbank) proponitaj (minimal, midsize aŭ deluxe), poste maldensigu ĝin en `C:\Program Files\Java\jre1.6.0_05\lib\audio\`.

- La dosiero `jre1.6.0_05` respondas al via versio de la instalita JRE.
- Se la dosiero `audio` ne ekzistas, necesos krei ĝin.
- Necesos alinomi la maldensigitan dosieron en: `soundbank.gm`

Poste rerulu XLogo kaj iru do rigardi en `Agordaj iloj / Elektebloj / Sono`

E.4 Kiel faru por tajpi rapide komandon jam uzitan?

- Unua metodo: per la muso, klaku en la historiejo sur la dezirata linio; ĝi reaperos tuj en la komandlinio.
- Dua metodo: per la klavaro, la sagoj supren kaj malsupren ebligas navigi en la listo de la laste tajpitaj komandoj.

E.5 Kiel oni povas helpi vin?

- Raportante pri cimoj (eraroj) konstatitaj. Estus eĉ pli bone, se vi kapablus sisteme aperigi konstatitan problemon.
- Viaj sugestoj por la plibonigo, estas ĉiam bonvenaj.
- Helpante pri tradukoj.
- Malgranda kuraĝigo ĉiam bonfaras!

Dankado

- Mi danku antaŭ ĉio la aktivaj tradukantoj de XLogo.
  - Angla: Guy Walker
  - Hispana: Marcelo Duschkin, Alvaro Valdes Menendez
  - Araba: El Houcine Jarad
  - Portugala: Alexandre Soares
  - Germana: Michael Malien
  - Esperanto: Michel Gaillard
  - Galega: Justo freire
  - Greka: Anastasios Drakopoulos
- Mi danku ankaŭ speciale Eitan Gurari pro la pacienco, kaj por la programo LaTeX `tex4ht` kiu ebligas eksporti la gvidlibrojn al diversaj formatoj.

  `www.cse.ohio-state.edu/~gurari/TeX4ht`
- Pluraj liberaj projektoj ebligantaj ekzisti XLogo:
  - Java3D: `https://java3d.dev.java.net/`
  - JavaHelp: `http://java.sun.com/javase/technologies/desktop/javahelp/`
  - Eclipse: `www.eclipse.org/`
- Finfine, GRANDAN dankon al Tuxfamily pro la kvalito de la provizita gastigado kaj ilia engaĝiĝo por libera programado!

  `http://www.tuxfamily.org`

Indekso