Andreas Rejbrand

Algoritmsimulator 2.0

– Projektbeskrivning

Reservationer

- Detta dokument innehåller mycket data som sammanställts från olika källor och under högst begränsad tid, utan någon korrekturläsning. Dokumentet kan innehålla åtskilliga tryckfel.
- Detta dokument beskriver en tidig plan för en kommande programvara. Innehållet i detta dokument kan fort bli inaktuellt. Dokumentet gör endast anspråk på att i stora drag beskriva ideologin bakom en framtida mjukvara.

Innehållsförteckning

Introduktionsdel

Introduktion

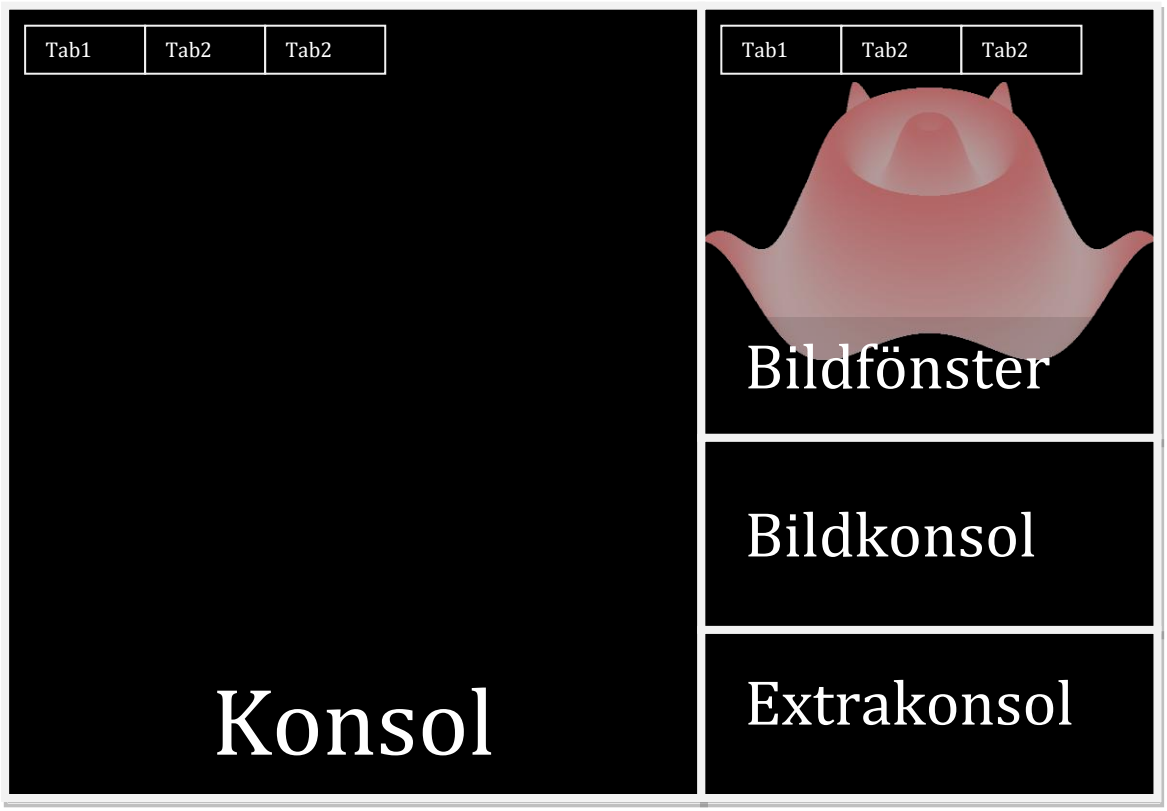
AlgoSim 2.0 är en framtida numerisk matematikprogramvara, väsentligen i samma stil som Matlab och besläktade applikationer, fast något mindre omfattande, och något nytänkande. Tanken är att programmet helt skall vara kommandostyrt och att funktionerna skall vara så allmänna och kraftfulla att i stort sett allt ”tänkbart” skall kunna utföras i programmet; inga begränsningar skall finnas i funktionernas utformning. AlgoSim 2.0 kommer att tolka ett eget programspråk direkt i kommandoprompten och tillåta enkel konstruktion av grafiska gränssnitt med vanliga och specialanpassade GUI-kontroller. Stor vikt läggs också vid grafritning och annan visualisering av funktioner och data. Vidare skall skönheten i matematiken betonas, och programmets gränssnitt är tänkt att spegla denna skönhet (till skillnad från många liknande programvaror).

AlgoSim 2.0 kommer från grunden att vara en helt ny programvara, även om namnet antyder att det finns en version 1.0 av en programvara med samma namn – och det gör det. Under gymnasiet utvecklade Andreas Rejbrand under en vecka den nämnda produkten, som kan betraktas på www.algosim.se.

I detta dokument kommer den huvudsakliga designen och de huvudsakliga funktionerna i AlgoSim 2.0 att beskrivas.

Gränssnitt

I stort kommer AlgoSim 2.0:s gränssnitt att se ut som följer:



De huvudsakliga operationerna kommer att ske i den flikbaserade konsolen (kommandoprompten), och bildfönster (plan och rum) visas i det flikbaserade området för bildfönster. Bildkonsolen är tänkt att användas för att kontrollera det aktuella bildfönstret, så att konsolens historik inte behöver tyngas ner. Dessutom finns en högerklicksmeny i bildkonsolen, med alla kommandon för att hantera bildfönstret. Klick på kommando i menyn, infogar kommandot i konsolen, varefter bara [Retur] behöver infogas för att exekvera kommandot. Extrakonsolen är en allmän extrakonsol, som också visar detaljerade utmatningar. Märk emellertid att samtliga konsoler i grund har precis samma funktionalitet, naturligtvis (skönhetsargument). Varje område kan maximeras, så att det fyller hela skärmen. Gränssnittet kan också enkelt programmeras om enligt användarens egen smak.

3D-behandling (kamerarotation m.m.) i bildfönster sköts företrädelsevis med tangentbordskommandon och mus, men textkommandon existerar också (för demonstrationsskript, t.ex.).

Datatyper

AlgoSim 2.0 kommer att arbeta med olika datatyper. Nedan listas samtliga datatyper samt ett kort urval av de grundläggande operatorer och funktioner som kan användas med dem.

- Reella tal, element i $\mathbb{R}$
    - Operatorer: addition (+), subtraktion (−), multiplikation (·), division (/), potens (^), fakultet (!), belopp (||), procent, multiplikation med 0.01 (%), promille, multiplikation med 0.001 (‰), grader till standardvinkelenhet (°), radianer till standardvinkelenhet (rad), siffra i tal från LSD (_)
    - Logiska operatorer: lika med (=), mindre än (<), mindre än eller lika med (≤), större än (>), större än eller lika med (≥), inte lika med, skilt från (≠), approximativt lika med (≈), delar heltal (|)
    - Funktioner: sin, cos, tan, cot, arcsin, arccos, arctan, arccot, sinh, cosh, tanh, coth, arsinh, arcosh, artanh, arcoth, exp, ln, lg, log, abs, C, P, ceil, floor, round, frac, int, rescale, mod, fact, fibonacci, prime, IsPrime, NextPrime, PrevPrime, factor, IsSquare, rationalize, FindExactExpression, RandomReal, RandomInt, RandomSign
- Vektorer i $\mathbb{R}^n$
    - Operatorer: addition (+), subtraktion (−), multiplikation med skalär *eller* skalärmultiplikation (·), kryssmultiplikation (×), belopp, norm (||), komponent (_)
    - Logiska operatorer: lika med (=), inte lika med, skilt från (≠), är parallell med (∥), är ortogonal mot (⊥)
    - Funktioner: dim, abs, ZeroVector, norm, max, min, mean, med, q1, q3, ForEach, FillVect, GCD, LCM, sort, ReverseOrder, sum, product, LeastSquare, SwapCoord, IndexOf, CoordLogicToScreen, CoordScreenToLogic, CreateBasis, BasisTransformationMatrix, TransformCoordinates, LinearTransformationMatrix
- Matriser (med element av godtycklig datatyp, även olika typer i samma matris) – speciellt representeras bitmappsbilder och ljud med matriser
    - Operatorer: addition (+), subtraktion (−), multiplikation med skalär *eller* matrismultiplikation (·), potens (^), elementär multiplikation (⊙), invers (^-1), transponat (*), division (/), determinant (||), element (_)
    - Logiska operatorer: lika med (=), inte lika med, skilt från (≠)
    - Funktioner: format, transpose, inverse, cols, rows, det, ZeroMatrix, IdentityMatrix, FillMatrix, FillDiagonal, FillMatRow, FillMatCol, ForEach, SortCol, SortRow, IndexOf, MatrixReplace, EditMatrixAsBitmap, BlendMatrix, ScaleMatrix, GetSubmatrix, MatrixAutoSize, Pixellate, InvertColors, EdgeDetect, blur, MotionBlur, GerRLayer, GetGLayer, GetBLayer, MergeRGBLayers, GetHLayer, GetSLayer, GetVLayer, MergeHSVLayers, colorize, brightness, contrast, BitmapMatrixToColorSet, PaintMatrixRows, PaintMatrixCols, PaintMatrixCell, PaintMatricRow, PaintMatrixCol, PaintMatrixRule, sum, product, RowMultiply, ColMultiply, RowSwap, ColSwap, RowOp, ColOp, LeastSquare, DifferenceRow, DifferenceCol, GCD, LCM, max, min, mean, med, q1, q3
- Mängder (med element av godtycklig datatyp, även olika typer i samma matris)

- Operatorer: union (∪), snitt (∩), mängddifferens (\), kartesisk produkt (×), komplement (C), direkt summa (⊕), antal element (||), element (_)
- Logiska operatorer: tillhör/har som medlem (∈/∋), tillhör inte/har inte som medlem (∉/∌), är delmängd av/innehåller (⊆/⊇), är äkta delmängd av/innehåller som äkta delmängd (⊊/⊋)
- Funktioner: NumOfElements, DeleteElement, ForEach, random, CreateSet, subset, ShortestPath, NeaestNeighbour, CenterOfMass, GCD, LCM, sum, product, max, min, mean, med, q1, q3, LeastSquare, ∃, ∀, PlaneToSpace, SpaceToPlane, CorrectPlane, CorrectSpace

- Text
  - Operatorer: tillägg (+), repetition (×), längd, antal tecken (||), tecken från vänster (_)
  - Logiska operatorer: lika med (=), inte lika med, skilt från (≠)
  - Funktioner: length, trim, TrimRight, TrimLeft, ReverseOrder, DelWhite, UpperCase, LowerCase, ChrEncode, ChrDecode, ChrDescription, substring, StrLeft, StrRight, StrPos, StrFindFirst, StrFindAll, StrReplaceFirst, StrReplaceAll, StrDel, StrDelLeft, StrDelRight, RandomStr, FindAnagrams, FindAllAnagrams, IsPalindrome, FindAllPalindromes, IsHeteropalindrome, FindAllHeteropalindromes, ForEachChar, ForEachWord, StrSplitSet, StrSplitMatrix, StrChrSet, StrChrMatrix, StrWordSet, StrWordMatrix

- Funktioner
  - Operatorer: godtyckliga operatorer vars operander kan vara av funktionsresultatens typer, t.ex. f + g och f·g om f och g är reellvärda funktioner, sammansättning (∘)
  - Logiska operatorer: lika med, exakt samma funktion (=), inte lika med (≠)
  - Funktioner: FuncName, NumOfArguments, diff, CreateDiffGraph, int, ∫, ∬, ∭, IntRiemann, IntIllustrate, CreateIntGraph, table, solve, AnalyzeFunction, contineous, taylor, maclaurin, CreateGradientVectorField, CreateDivergenceGraph, CreateDivergenceColorPlane, CreateDivergenceLevelPlane, CreateDivergenceColorSpace, CreateDivergenceLevelSpace, CreateRotationVectorField, CreateRotationColorSpace, ∇, ∇·, ∇×

- Logiska värden
  - Operatorer: och (∧), icke och, NAND (⊼), eller (∨), icke eller, NOR (⊽), exklusivt eller (⊻), icke (¬), implikation åt höger (⇒), implikation åt vänster (⇐), ekvivalens (⇔)
  - Logiska operatorer: är lika med (=), d.v.s. samma funktion som ekvivalens (⇔), inte lika med (≠)
  - Funktioner: CreateTruthTable

- Komplexa tal *(endast begränsat stöd)*
  - Operatorer: addition (+), subtraktion (−), multiplikation med reellt tal *eller* multiplikation mellan komplexa tal (·), division (/), potens (^), belopp (||), komplexkonjugat (*)
  - Logiska operatorer: är lika med (=), är inte lika med, är skilt från (≠)
  - Funktioner: Re, Im, abs, arg, exp

- Tvådimensionellt bildfönster *(se separat avsnitt)*
- Tredimensionellt bilfönster *(se separat avsnitt)*
- Progress (lyckades, lyckades inte)
- Kontroll (knapp, textfält, textetikett, skjutreglage, tvådimensionellt skjutreglage, vinkelangivare, kryssruta, radioknapp, textyta, progressmätare, rullväljare, listruta, färgväljare, nyansväljare, flyttbar punkt i bildfönster)
- Gränssnitt (med kontroller) *(se separat avsnitt)*
- Referens (pekare till variabel – skickas som funktionsargument när själva variabeln snarare än dess värde skall bearbetas)
- Strukturtyp *(se separat avsnitt)*
- Struktur *(se separat avsnitt)*

För alla variabler existerar dessutom följande allmänna operatorer och funktioner:

- Operatorer: referens (@), tilldelning (≔)
- Logiska operatorer: *saknas*
- Funktioner: delete, rename, declare, CreationTime, ModificationTime, type, description, MetaData

Metadata

Varje variabel innehåller metadata som bland annat berättar när variabeln skapades och när den senast ändrades. Även en beskrivning av variabeln kan lagras, vilket speciellt är praktiskt för matematiska och fysikaliska konstanter; t.ex. är beskrivningen för $\hbar$ lika med ”Placks konstant dividerad med 2π”. Andra exempel är m_p, ”Protonens vilomassa” och c, ”Ljusets fart i vakuum”. För att söka efter en variabel vars beskrivning innehåller en sträng <str>, används kommandot **find**(str).

Om ett fotografi importeras till en BMP-matris, kommer matrisens metadata att innehålla all EXIF-information lagrad i bildfilen. För att allmänt erhålla metadata med namn <mtnm>, använd kommandot **MetaData**(mtnm). T.ex. kan MetaData(”exponeringstid”) ge värdet 0.001.

En kolonnmatris som beskriver en ljudvåg lagrar samplingsfrekvensen, bitdjupet och den relativa ljudstyrkan som metadata.

Numerisk räkning

Reella tal i $\mathbb{R}$

Numerisk räkning med reella tal i $\mathbb{R}$ fungerar på enklast möjliga sätt. Tal skrivs in på normal form, t.ex. −7.98, och operatorer och funktioner fungerar på förväntat sätt. Till exempel är

```
(sin(0.5))^2 + (cos(0.5)^2
```

ett giltigt uttryck, som kommer att evalueras till 1. Eftersom programmet är kommandoradsbaserat, sparas beräkningshistoriken. AlgoSim 2.0 utför varje beräkning i en egen tråd, så om en beräkning tar väldigt lång tid (3D-rendering, sökning efter anagram etc.), kan den pågå i bakgrunden under den tid som krävs, medan andra beräkningar utförs som vanligt. Information om hur långt varje tråd kommit ges i extrakonsolen.

Det finns ingen begränsning i hur många nivåer av parenteser som kan användas, eller hur långt ett uttryck får vara. Stora och små tal kan skrivas på grundpotensform med operatorn ”E”, som definieras på vanligt sätt:

$$a\mathrm{E}n \equiv a \cdot 10^n, n \in \mathbb{Z}$$

Ett reellt tal *kan* i AlgoSim ges det symboliska värdet ±∞; de allra flesta funktioner kan emellertid *inte* användas med dessa ”värden” som argument, men för vissa är det praktiskt (se t.ex. **AnimateImages** på sidan 25). För de enkla räknesätten kan ±∞ inte heller användas, om inte AS:SIMPLIFIEDINFINITY är satt till TRUE. I sådana fall gäller reglerna i tabellen nedan. Inom vissa områden, såsom geometrisk optik, kan det vara praktiskt.

$\pm\infty + a = \pm\infty,$ $a \in \mathbb{R}$	$a \cdot (\pm\infty)$ $= \operatorname{sgn} a \cdot (\pm\infty),$ $a \in \mathbb{R}, \quad a \neq 0$	$0 \cdot (\pm\infty) = 0$	$\frac{\pm\infty}{a} = \operatorname{sgn} a \cdot (\pm\infty),$ $a \in \mathbb{R}$	$\frac{a}{\pm\infty} = 0,$ $a \in \mathbb{R}$
$\frac{a}{0} = \operatorname{sgn} a \cdot \infty,$ $a \in \mathbb{R}, \quad a \neq 0$	$\infty + \infty = \infty$	$-\infty - \infty = -\infty$	$\infty \cdot \infty = \infty$	$(-\infty) \cdot \infty = -\infty$

Inte ens om AS:SIMPLIFIEDINFINITY är TRUE har uttrycken $[\infty/\infty]$, $[\infty - \infty]$ och $[0/0]$ någon mening.

Texter

Texter deklareras genom att de anges inom dubbla citationstecken (”). Om citationstecknet behövs i texten, koden \q användas. Tecknet ”\” infogas med ”\\” osv. Exempel:

```
greeting ≔ ”Welcome to AlgoSim, the \qbest\q software ever made!”
```

Tecken nummer *k* i en sträng hämtas med operatorn _. T.ex. är ”Test”_2 = ”e”.

Enkla formateringskommandon kan anges i texter, enligt MediaWikis välbekanta syntax.

```
”Detta är ett ’’väldigt’’ bra program, ’’’eller hur’’’?”
```

ger

Detta är ett *väldigt* bra program, **eller hur**?

Funktioner

Funktioner från ett objekt till ett annat (specialfall: från $\mathbb{R}^{\boldsymbol{n}}$ till $\mathbb{R}^{\boldsymbol{m}}$) definieras med funktionsskaparen ↦. ↦ är en binär operator som tar in två textsträngar, en med en kommaseparerad lista av de oberoende variablerna (deras namn) och en med ett uttryck för den beroende variabeln. Exempel:

```
f ≔ "x" ↦ "x^2 + 1"

g ≔ "u, v" ↦ "(u, v, u·v)"

h ≔ "mat1, mat2" ↦ "|mat1* · mat2|"

greeting ≔ "name" ↦ "\qWelcome to AlgoSim, dear \q + name + \q!\q"
```

Vektorer i $\mathbb{R}^n$

Vektorer i $\mathbb{R}^n$ har formatet $(x_1, x_2, x_3, \dots)$ där $x_k \in \mathbb{R}$. AlgoSim 2.0 förstår också notationen

$$(x_1, x_2, \dots) = \underline{\mathbf{e}} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} = \underline{\mathbf{e}} A \tag{1}$$

där standardbasen i $\mathbb{R}^n$, $\underline{\mathbf{e}}$, är en radmatris som definieras enligt

$$\underline{\mathbf{e}} = (\mathbf{e}_1\ \mathbf{e}_2\ \dots)$$

där basvektorerna $\mathbf{e}_k = (0, \dots, 1, \dots, 0)$ med ettan som komponent nummer $k \in [1, n]$. Detta gör sambandet (1) trivialt sant för alla vektorer i $\mathbb{R}^n$. Kolonnmatrisen A kallas *koordinatmatris*.

Den k:te komponenten i vektorn **v** erhålles med operatorn _ som **v**_k. Ofta behöver man lägga till en ny komponent sist i en vektor. Detta åstadkommes enkelt genom användande av det symboliska indexet $k = -1$, som alltså syftar till komponenten med index n i den n-dimensinella vektorn, som har komponenterna $0, 1, \dots, n-1$.

Matriser

Matriser är tvådimensionella uppsättningar av godtyckliga objekt (med stöd av AlgoSim). En flerdimensionell matris kan följaktligen erhållas om en tvådimensionell matris fylls med matriser. Kommandot **CreateMatrix** öppnar matriseditorn, och en ny matris fylls i, och returneras. På detta sätt kan man enkelt skapa en ny matris:

```
MyMatrix ≔ CreateMatrix
```

Kommandot **EditMatrix**(@MyMatrix) kan sedan användas för att redigera matrisen i samma editor. En m×n-matris kan också skapas direkt i kommandoraden, med syntaxen **CreateMatrix**(m, n, a_{11}, a_{12}, …, a_{1n}, …, a_{m1}, a_{m2}, …, a_{mn}):

```
MyMatrix ≔ CreateMatrix(2, 2, 1, 0, 0, 1)
```

En matris sådan att det existerar en funktion $(i, j) \mapsto f(i, j) = a_{i,j}$ kan också skapas automatiskt:

```
MyMatrix ≔ CreateMatrix(100, 100, f)
```

Vi kan på detta sätt enkelt erhålla en multiplikationstabell:

```
MyMatrix ≔ CreateMatrix(12, 12, ”i,j” ↦ ”i·j”)
```

Vi kan också skapa en tabell med slumpvisa heltal mellan 0 och 100:

```
MyMatrix ≔ CreateMatrix(100, 20, ”i,j” ↦ ”RandomInt(0, 100)”)
```

Funktionen i det tredje argumentet till **CreateMatrix** kan också bytas ut mot ett objekt (som då kommer att fylla matrisen).

Till exempel kan vi skapa en 100×100-matris fylld med nollor eller texten ”Test”:

```
MyMatrix ≔ CreateMatrix(100, 100, 0)

MyMatrix ≔ CreateMatrix(100, 100, ”Test”)
```

Funktionerna **rmat** och **cmat** används på liknande sätt, fast förenklat i den meningen att de bara skapar rad- respektive kolonnmatriser. Element (i, j) i en matris M erhålles med operatorn _, enligt M_(i,j).

Med hjälp av matrisformalismen kan (entydigt lösbara) linjära ekvationssystem av godtycklig dimension mycket enkelt lösas; om systemet skrives som en matrisekvation

$$AX = B$$

där A är koefficientmatrisen, X är en kolonnmatris med de sökta variablerna och B är högerledsmatrisen så är ju lösningen

$$X = A^{-1} \cdot B.$$

Denna beräkning kan givetvis skrivas in som den står ovan, men AlgoSim förstås också den ekvivalenta notationen

$$X = B/A$$

där matrisdivisionen $B/A \equiv A^{-1} \cdot B$.

Mängder

Mängder deklareras på det mest naturliga sättet:

```
MySet ≔ {1, 2, ”Example”}
```

Intervall skapas också på det mest naturliga sättet: intervallets vänstra gräns a och högra gräns b anges innanför [(]) respektive] ([) om intervallet är slutet (öppet) vid respektive gräns. Eftersom ingen dator kan hålla oändligt många tal i minnet samtidigt, så blir mängderna diskreta, och avståndet mellan två närliggande tal i mängden kallas för intervallets *upplösning*. Som standard är denna i en sådan storleksordning att diskretiseringen i allmänhet inte märks vid numerisk och grafisk analys. Eventuellt kan ett *tredje* tal infogas mellan [(]) och] ([); detta sista tal anger explicit upplösningen.

```
MyInterval ≔ [-10, 10]

MyExtraFineInterval ≔ [-10, 10, 0.00001]
```

En plan rektangel kan erhållas som den kartesiska produkten mellan två intervall, och på samma sätt kan rätblock av godtycklig dimension erhållas. T.ex. är enhetskuben med medelpunkt i origo och ytor parallella med koordinatplanen

```
UnitCube ≔ [-1/2, 1/2]×[-1/2, 1/2]×[-1/2, 1/2]
```

Vi kan också skapa enhetssfären med funktionen **CreateSet**, som tar in en text med ett logiskt villkor med koordinater, och returnerar mängden av alla punkter vars koordinater uppfyller villkoret, inom ett bestämt testintervall på de tre koordinaterna (om vi befinner oss i rummet):

```
UnitSphere ≔ CreateSet(”x^2 + y^2 + z^2 = 1”, -1, 1, -1, 1,-1,
1)
```

Detta går i allmänhet bra, men det är *inte* det mest effektiva sättet, av anledningen som diskuteras nedan på sidan 17. Ett bättre sätt är att använda polära koordinater (mängden blir då ett plan, men vid plottning med sfäriska koordinater får vi ändå en sfär), eller parametrisera enhetssfären. Vi kan också använda det inbyggda kommandot **CreateSphere**. Som märks finns det många sätt att utföra en given operation på, och detta tillåter varje algoritm att skrivas på det sätt som känns mest naturligt.

Som standard är alla mängder i AlgoSim ”äkta” mängder, d.v.s. mängder som innehåller ett ändligt antal element, och inget annat. Detta bygger hela programmets funktionalitet på, bland annat grafritning och annan visualisering. Till exempel består mängden [0, 1] av alla tal mellan noll och ett *med en bestämd upplösning*. Emellertid orsakar detta vissa problem. Säg till exempel att vi skapar mängden *D* med alla punkter i planet som uppfyller $x^2 + y^2 \leq 1$, d.v.s. den fyllda, slutna enhetscirkeln. Då skall uttrycket $(x, y) \in D$ vara sant för alla punkter (x, y) i enhetscirkeln, men i praktiken kommer det bara att vara sant för de punkter (x, y) som råkar sammanfalla med något av de ändligt många elementen i *D*.

För att råda bot på detta problem, kommer en extra mängd information, i form av en textsträng, en s.k. *generator*, att skapas med varje mängd. Generatorn är ett uttryck som beskriver villkoret för att ett element skall tillhöra mängden. Till exempel, när mängden *D* först skapas, kommer – förutom att den fylls med de ändligt antal elementen – den att få generatorsträngen ”$x^2 + y^2 \leq 1$”. Denna är helt irrelevant i de flesta situationer, t.ex. vid grafritning och annan visualisering, men gör så att uttryck i stil med $(x, y) \in D$ alltid blir så sanna, som de förväntas bli (till och med om man bortser från begränsningar i numeriska mjukvaror). Generatorn kommer sedan att finnas kvar så länge som möjligt; om exempelvis unionen $E = D \cup ([-2, 2] \times [-0.6, 0.6])$ bildas, så kommer generatorn för E att bli ”$(x^2 + y^2 \leq 1) \vee ((x \geq -2 \wedge x \leq 2) \wedge (y \geq -0.6 \wedge y \leq 0.6))$”. Observera emellertid att AlgoSim har stöd för godtyckliga mängder, varav de flesta inte alls använder generatorsträngar. Till exempel skapas förstås ingen generatorsträng när mängden {”McDonald's”, ”Max”, ”Burger King”} skapas, i syfte att med slumpelementfunktionen **random** bestämma kvällens restaurang.

Vidare, vid skapande av öppna intervall, t.ex.]0, 2], kommer extra element, logaritmiskt närmare och närmare den öppna randpunkten, att läggas till mängden. Om upplösningen är 0.01 kommer t.ex. elementen 0.01, 0.02, 0.03, …, 1,98, 1.99, 2.00 förstås att tillhöra mängden, men även elementen 0.001, 0.0001, 0.00001, …, vilket t.ex. gör det grafiska studiet av gränsvärden enklare (precis som om programvaran varit symbolhanterande).

Som standard tolkar AlgoSim konstanten ∅ ≡ { } som den tomma mängden. Även svar visas som denna; till exempel kommer beräkningen {1,2,3} ∩ {4,5} att ge svaret ∅, så länge användaren inte omdefinierar konstanten ∅.

Logiska värden

Logiska värden är TRUE och FALSE. TRUE och FALSE är följaktligen reserverade namn, så inga variabler kan ges dessa namn. Om användaren försöker skriva ett numeriskt värde till en logisk variabel, kommer en nolla att tolkas som FALSE, och en etta som TRUE. Andra tal kommer att resultera i väntat felmeddelande.

Bildfönster

Två- och tredimensionella bildfönster är två datatyper som används för att skapa bildfönster som representerar planet respektive rummet och används för diagram och grafritning. I dessa variabler lagras alla egenskaper för bildfönstret, såsom vilken region som syns på skärmen, hur kameran är orienterad, vilka punkt- och linjefärger som används, vilka koordinatsystem som används och *om* och i sådana fall även *hur* koordinatkurvor och –ytor samt eventuell gradering av axlar ritas upp.

Progressvärde

Objekt av datatypen ”progressvärde” returneras av vissa funktioner, för att meddela att funktionen utfördes framgångsrikt eller att den misslyckades. Om ett progressvärde anges där ett logiskt värde förväntas, kommer mappningen TRUE = DONE och FALSE = FAIL att användas.

Kontroll

Objekt av datatypen ”kontroll” är GUI-komponenter. Följande komponenter kommer att finnas med:

knapp	textfält	textetikett
skjutreglage	2D-skjutreglage (vektorväljare)	vinkelangivare
kryssruta	radioknapp	textyta
progressmätare	rullväljare	listruta
färgväljare	nyansväljare	flyttbar punkt i bildfönster
ljudspelare	klocka	

Gränssnitt

Ett objekt av typen ”gränssnitt” är en panel med kontroller, d.v.s. ett GUI-gränssnitt. Såväl kontroller som gränssnitt hanteras symboliskt direkt i kommandoraden med funktioner avsedda för ändamålet (gås ej igenom i detta avsnitt). Gränssnitt kan följaktligen skapas automatiskt.

Referens

En referens är en pekare till en variabel, och används främst i funktionsargument där själva variabeln, snarare än dess värde, skall behandlas. En referens till en variabel A skrives @A.

Strukturtyp

En strukturtyp innehåller information om vilka objekt som ingår i en viss typ av struktur (den typ som strukturtypen definierar), d.v.s. ett objekt som innehåller andra objekt.

En strukturtyp deklareras med **NewStructType** och ges medlemmar med **StructAddMember**, **StructDeleteMember**, **StructChangeMemberNames** och **StructChangeMemberDefaultValue**. En visuell strukturtypseditor nås med **VisualStructTypeEdit**.

Struktur

En struktur skapas med kommandot **CreateStructure**(*StructType*), där *StructType* är den strukturtyp som skall användas. För att nå ett enskilt element i en struktur, går man in i strukturen med operatorn ”.”. Strukturer används med fördel i endimensionella matriser vid programmering. Utskriften (i kommandoraden) av en struktur är samtliga medlemmar listade intill varandra.

```
Fiona ≔ CreateStructure(djurinfo);

Fiona.art ≔ ”råtta”

Fiona.kön ≔ ”hona”

Fiona.ålder ≔ 1.3
```

Visualisering

Stor vikt har lagts vid grafisk visualisering av funktioner och data. I princip alla tänkbara möjligheter skall kunna utföras.

Exempel på vad som enkelt kan utföras

- visualisering av funktioner $x \mapsto f(x)$ av en variabel, mellan koordinater i planet, som ofärgad eller systematiskt färgad kurva, även av funktioner som ändras med tiden eller med kontrollstyrd parameter
- visualisering av parametriserade kurvor $t \mapsto (f(t), g(t))$ i tvådimensionellt koordinatsystem samt $t \mapsto \big(f(t), g(t), h(t)\big)$ i tredimensionellt koordinatsystem, ofärgade eller systematiskt färgade, även av funktioner som ändras med tiden eller med kontrollstyrd parameter
- visualisering av funktioner $x \mapsto f(x)$ av en variabel med färgad eller nivåmarkerad tallinje, även av funktioner som ändras med tiden eller med kontrollstyrd parameter
- visualisering av funktioner $(x, y) \mapsto f(x, y)$ av två variabler, mellan koordinater i rummet, som ofärgad eller systematiskt färgad yta, även av funktioner som ändras med tiden eller med kontrollstyrd parameter
- visualisering av parametriserade ytor $(u, v) \mapsto \big(f(u, v), g(u, v), h(u, v)\big)$ i tredimensionellt koordinatsystem, ofärgade eller systematiskt färgade, även av funktioner som ändras med tiden eller med kontrollstyrd parameter
- visualisering av funktioner $(x, y) \mapsto f(x, y)$ av två variabler med färgat plan eller med nivåkurvor i planet, även av funktioner som ändras med tiden eller med kontrollstyrd parameter
- visualisering av funktioner $(x, y, z) \mapsto f(x, y, z)$ av tre variabler med färgat rum eller med nivåytor i rummet, även av funktioner som ändras med tiden eller med kontrollstyrd parameter
- uppritning av godtyckliga mängder i planet och rummet, även av funktioner som ändras med tiden eller med kontrollstyrd parameter
- uppritning av punktmängder i plan och rum som punkter och som vektorer
- uppritning av vektorfält, gradientfält, divergensfält, rotationsfält m.m.
- uppritning av bitmappsbilder och vektorgrafiska bilder

Tillvägagångssätt

Det huvudsakliga sättet att skapa en graf på är att parametrisera den.

1. *Skapa en definitionsmängd.* En definitionsmängd skapas, kanske med någon av följande metoder:
   - $[a\,, b]$ ger mängden av alla tal mellan a och b, d.v.s. ett linjestycke. Den kartesiska produkten av två linjestycken blir en rektangel, och på samma sätt kan ett rätblock av godtycklig dimension erhållas.
   - **CreateSet** – tar in ett logiskt uttryck som innehåller koordinater i något koordinatsystem i planet eller rummet och ger mängden av alla punkter som uppfyller villkoret

- **CreateNet** – fungerar som **CreateSet** fast ger ett rutnät i området
- **CreateSphere**, **CreateBall**, **CreateLine**, **CreatePlane**, **CreatePolotype**

2. *Skapa en värdemängd*. Funktionen **DomainImage** tar en funktion och en definitionsmängd och ger värdemängden, d.v.s. definitionsmängden efter att funktionen verkat på varje element. **DomainImage** använder vi emellertid inte här, utan vi använder **DomainImageEx**, som också tar en funktion och en värdemängd, men returnerar *grafen*, t.ex., i det tvådimensionella fallet, mängden av alla punkter $(x, f(x))$.

3. *Rita grafen*. Funktionen **DrawPointSet** tar grafen och ritar upp den i bildfönstret.

Bara med dessa allmänna funktioner kan vi rita funktiongrafer (kurvor, t.ex. av $x \mapsto \sin x$) i planet och ytor (t.ex. av $(x, y) \mapsto \sin\sqrt{x^2 + y^2}$) rummet, *i alla koordinatsystem*. Funktionen **DrawPointSet** tar nämligen ett frivilligt argument, som anger koordinatsystemet $(x, y), (x, y, z), (r, \phi), (r, \phi, z), (r, \theta, \phi)$ etc. Vi kan också rita parameterkurvor, t.ex. av $t \mapsto (t, t^2, \sin t)$, -ytor, t.ex. av $(u, v) \mapsto (u, v^2, u \cos v)$, och -kroppar; med hjälp av **DomainImage** och en parametriseringsfunktion erhåller vi ju direkt en mängd som kan skickas till **DrawPointSet**.

Grafer som ändras med tiden erhålles med extra koordinat t (som AlgoSim känner igen), och grafer som ändras med kontroller erhålles med extra koordinater AS:Control1, AS:Control2, …, AS:ControlN.

Vidare, om vi byter ut **CreateSet** mot **CreateNet**, så kan vi rita nätytor och nätkroppar, bestående av kurvor längs vilka endast en parameter ändras.

Färgade linjer, plan och rum kan erhållas genom att (del-) linjen, (del-) planet respektive (del-) rummet först skapas (det är ju en vanlig mängd), varefter funktionen **ColorizeSet**, med en given funktion, lägger till en extra koordinat, som anger färgen, till varje vektor (punkt) i mängden. Sedan ritas den färgade punktmängden upp med **DrawPointSetColor**, som fungerar precis som **DrawPointSet**, fast den förväntar sig en färgkoordinat till varje punkt. Nivåkurvor och –ytor skapas med **ColorizeSetLevels**, som tar en mängd och ger delmängden som innehåller nivåkurvorna och –ytorna. Denna mängd kan sedan ritas upp som vanligt, med **DrawPointSet**.

Bildfunktioner, d.v.s. funktioner definierade i ett område och som ger ifrån sig en färg för varje punkt, kan visualiseras med funktionen **ColorFunctionSet**, som tar in en bildfunktion och en definitionsmängd, och returnerar definitionsmängden med en extra färgkoordinat, given av bildfunktionen. Denna mängd ritas förstås med **DrawPointSetColor**.

Vektorfält i rummet respektive planet är mängder (x, y, z, v_x, v_y, v_z) respektive (x, y, v_x, v_y) som för varje punkt i planet eller rummet associerar en vektor. Ett vektorfält skapas av funktionen **CreateVectorField**, som tar in ett definitionsområde och en funktion (vektorfältets funktion, i rummet av formen $(x, y, z) \mapsto \big(f(x), g(y), h(z)\big)$), och returnerar en av mängderna ovan. Vektorfält ritas med **PlotVectorField**, en analog till **DrawPointSet**.

AlgoSim 2.0 är inte begränsat till att enbart arbeta med koordinatsystemen $(x, y), (x, y, z), (r, \phi), (r, \phi, z), (r, \theta, \phi)$; i själva verket är dessa inte ens hårdkodade, utan varje koordinatsystem (förutom de kartesiska) är definierat som en struktur av två transformationsekvationer samt en text, som är namnet på systemet. Det sfäriska koordinatsystemet definieras t.ex. med strukturen

$$\begin{cases} \text{Transform1} = (r,\theta,\phi) \mapsto \underline{\mathbf{e}} \begin{pmatrix} r\sin\theta\cos\phi \\ r\sin\theta\sin\phi \\ r\cos\theta \end{pmatrix} \\ \text{Transform2} = (x,y,z) \mapsto \underline{\mathbf{e}} \begin{pmatrix} \sqrt{x^2+y^2+z^2} \\ \arctan2\left(\sqrt{x^2+y^2}, z\right) \\ \arctan2(y,x) \end{pmatrix} \\ \text{Name} = \text{"r}\theta\phi\text{"} \end{cases}$$

Inga begränsningar!

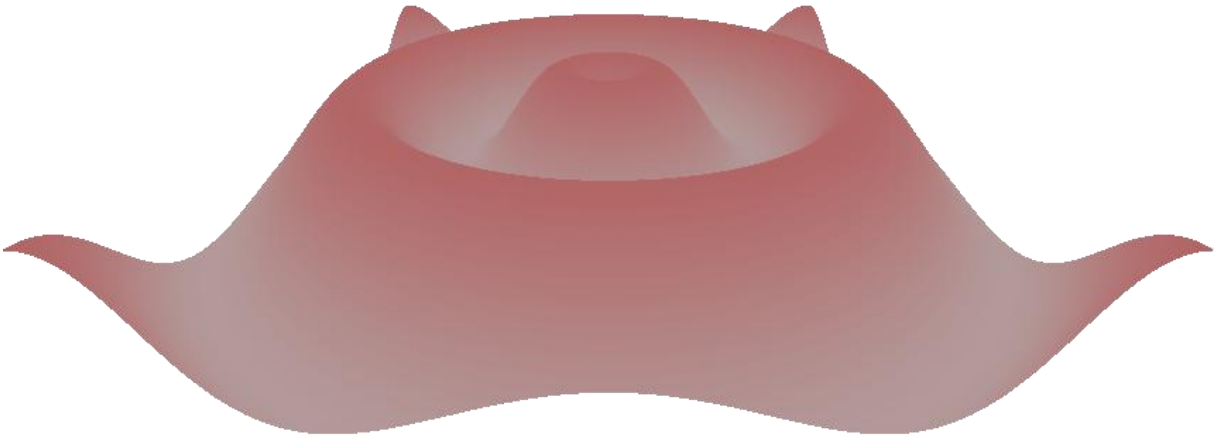
Notera att enkelheten hos den standardmetod för visualisering som angivits ovan rimligen kan ifrågasättas; man skulle ju kunna skapa en mängd åt **DrawPointSet** direkt med funktionen **CreateSet**. Problemet med detta ligger i naturen hos en numerisk programvara.

Antag att vi vill rita upp en enkel, plan och tidsoberoende sinuskurva. Om vi enbart använder **CreateSet** så ser kommandot ut som följer:

```
DrawPointSet(CreateSet("y=sin(x)", -10, 10, -1, 1))
```

CreateSet delar in området $(x \in [-10{,}10]) \wedge (y \in [-1{,}1])$ i många små delar, och undersöker i varje del om det där gäller att $y = \sin x$. I sådana fall läggs punkten (x, y) till i den resulterande mängden. Problemet är att planet ju egentligen består av *oändligt* många punkter, så en avrundning sker. Om funktionen **CreateSet** vore helt naiv, skulle följaktligen i princip inga punkter alls returneras, ty sannolikheten att just de punkter som uppfyller ekvationen $y = \sin x$ väljs ut för test, är väldigt liten (annat än i speciella punkter, såsom i origo). Nu är emellertid **CreateSet** smartare än så; den tillåter små avvikelser, så att resultatet hade blivit tämligen acceptabelt trots allt, men hela problemet försvinner om vi använder den rekommenderade metoden beskriven ovan. Då skapar vi ju först en definitionsmängd, med många närliggande punkter på *x*-axeln, och *för varje sådan punkt*, beräknar vi sinus och plottar punkten. Vi slipper alltså avrunda.

```
DrawPointSet(DomainImageEx(sin, [-10 10]))
```



Figur 1 Funktionen $(x, y) \mapsto \sin\left(\sqrt{x^2+y^2}\right)$ ritad med mycket tidig experimentell version av den numeriska 3D-renderingsalgoritm som kommer att implementeras i AlgoSim 2.0.

Bildfönstret

Inställningar för ett två- eller tredimensionellt bildfönster sparas i respektive datatyp. För varje variabel av en sådan datatyp i minnet, visas en flik i bildfönsterområdet i AlgoSim.

Ett nytt tvådimensionellt bildfönster skapas med **Create2DSpace**. Inställningar kan bland annat göras med följande kommandon:

SetIntervals	SetGridDistance	SetGridVisibility
SetAxisVisibility	SetAxisMarkerDistance	SetAxisMarkerVisibility
SetAxisNumeralMarkerDistance	SetAxisNumeralMarkerVisibility	SetAxisColors
ShowCoordinateSystem	ActiveCoordinateSystem	Duplicate2DSpace
ShowBasis	HideBasis	

SetIntervals anger de intervall av koordinater som syns på skärmen. **SetGridDistance** och **SetGridVisibility** styr rutnätet av koordinatkurvorna, medan **SetAxisVisibility**, **SetAxisMarkerDistance**, **SetAxisMarkerVisibility**, **SetAxisNumeralMarkerDistance, SetAxisNumeralMarkerVisibility** samt **SetAxisColors** styr koordinataxlarna. **ShowCoordinateSystem** väljer vilka koordinatsystem som skall visas (kartesiska, sfäriska, …), och **ActiveCoordinateSystem** väljer vilket koordinatsystem som de tidigare nämnda inställningarna påverkar. **Duplicate2DSpace** skapar en ny 2DSPACE-variabel identisk med den aktuella. **ShowBasis** och **HideBasis** visar respektive döljer basvektorerna i det koordinatsystem som ges av argumentet.

För att förenkla användningen av AlgoSim existerar också effektivare kommandon i stället för **SetIntervals**. Dessa är **xmin**, **xmax**, **ymin**, **ymax**, **zmin** och **xmax** som sätter minsta eller största värdet på respektive koordinat. **xlim**, **ylim** och **zlim** sätter båda gränserna samtidigt för respektive koordinat, såsom xlim(−10, 10). **SetAxisVisibility** kan ersättas med **axes** som anropas med en textsträng bestående av beteckningarna för de axlar som skall visas; t.ex. kommer axes(”xyz”) att visa *x*-, *y*- och *z*-axlarna, men dölja alla andra. **grid** fungerar likadant, fast sätter visningen av rutnätet (koordinatkurvorna); t.ex. kommer grid(”θϕ”) att visa koordinatkurvor för θ- och ϕ-koordinaterna, men dölja alla andra koordinatkurvor. **griddist** anger avståndet mellan koordinatkurvorna för en viss koordinat, t.ex. grid(”x”, 2) eller grid(”ϕ”, π/4).

Etiketter på axlar behöver inte bygga på enheten 1, utan kan bygga på vilken konstant som helst, t.ex. π. **SetAxisUnit** styr detta. SetAxisUnit(@π) kommer att använda bråk multiplicerade med symbolen π, där π har sitt vanliga värde, som etiketter på axlarna. Avstånden mellan etiketterna väljs automatiskt till något lämpligt värde. Man kan ange speciella inställningar för olika axlar; SetAxisUnit(”x”, @π) verkar t.ex. bara på *x*-axeln. Avståndet mellan etiketterna kan naturligtvis anges med **SetAxisNumeralMarkerDistance**, men även t.ex. SetAxisUnit(”x”, @π, 1/4) fungerar; märk att vi med **SetAxisNumeralMarkerDistance** måste ange (1/4)π som argument i stället för 1/4 som räcker för SexAxisUnit. Slutligen kan vi också välja att förvisso använda π som enhet, men använda en annan symbol än just den som programmet använder för att spara värdet av π i, d.v.s. ”π”, som etikett. SetAxisUnit(”x”, π, ”pi-enheter”, 1/4) åstadkommer detta. Märk att vi nu, när vi bara behöver ange *värdet av* π som argument – och inte hela variabeln med dess namn – så skriver vi ”π” i stället för ”@π” som andra argument.

SetAxisLabel lägger till en etikett under eller vid sidan av en axel. Denna kan t.ex. innehålla namnet på den storhet som axeln svarar mot. T.ex. är SetAxisLabel(”Temperatur/K”) ett tänkbart anrop. Etiketten som är en vektorgrafisk matris (se Vektorgrafik nedan) sparas under AS:DRAW:<spnm>:AXES:<axnm> där <spnm> är namnet på den aktuella ritytan (det aktuella bildfönstret) och <axnm> är namnet på den aktuella axeln.

Grafer som ändras med tiden kan hanteras med följande kommandon:

SetDynamicPlaySpeed	pause	resume	restart

SetDynamicPlaySpeed anger uppspelningshastigheten, **pause**/**resume** pauserar/fortsätter uppspelningen och **restart** börjar om vid första bildrutan. Motsvarande kommandon finns för tredimensionella bildfönster.

Till varje bildfönster hör en uppsättning variabler som inleds med AS:<NAMN>, där <NAMN> är bildfönstrets variabelnamn. Dessa lagrar variabelbaserade inställningar för fönstret. När ett bildfönster kopieras eller raderas, följer dessa med. Nedan listas några av dessa variabler (utan prefix):

DRAW:COLOR	DRAW:DOTSIZE	DRAW:DOTROUND
DRAW:MATRIXDOT	DRAW:FUNCTIONDOT	FONT:NAME
FONT:SIZE	FONT:BOLD	FONT:ITALIC
FONT:UNDERLINE	FONT:STRIKETHROUGH	FONT:ROTATION
FONT:XSCALE		

DRAW:COLOR och DRAW:DOTSIZE anger färg respektive storlek på punkter på skärmen. DRAW:DOTROUND anger huruvida punkter är runda eller kvadrater. Om DRAW:MATRIXDOT är sann, kommer matrisen AS:DOTMATRIX att användas som bitmappsbild för punkten, istället för att den blir en fylld cirkel eller kvadrat. Om DRAW:FUNCTIONDOT är sann, kommer funktionen AS:DOTFUNCTION att anropas vid varje punktritning. Funktionen ges samtliga koordinater för den aktuella punkten, och förväntas returnera en bitmapp. FONT-serien anger typsnitt för textuppritning på skärmen. För att förenkla hanteringen av dessa inställningar finns följande genvägar: **font**(FontName), **font**(TextSize) **font**(FontName, FontSize) samt **fontrot**(RotationAngle). **fontstyle**("bi") gör texten fet (**b**old) och kursiv (**i**talic), men inte understruken (**u**nderlined) eller genomstruken (**s**trikethrough). **fontscale**(xcscale) anger x-skalningen. **color**(colour) anger färgen på allt som ritas upp – allt från monokromatiska mängdplottar, monokromatiska matrisplottar och monokromatiska bitmappsbilder till monokromatiska vektorgrafiska bilder och texter.

Övriga funktioner som anger inställningar och utgör kommandon för bildfönster listas nedan:

SetVectorOrigin	DrawVector	DrawPointSet
DrawPointSetColor	DrawVectorSet	DrawMatrix
DrawMatrixLined	PlotData2D	PlotData3D
PlotVectorField	DrawText	GetStringWidth
GetStringHeight	DrawPlaneMatrix	DrawPlaneMatrixColor

SetVectorOrigin anger den punkt där vektorpilar börjar vid uppritning av vektorer med **DrawVector**. **DrawVectorSet** ritar alla vektorer i en mängd. **DrawMatrix** ritar alla punkter (vektorer) i en matris, och **DrawMatrixLined** ritar dessa med linjer mellan dem. **DrawPlaneMatrix** ritar upp en matris i planet där varje punkt färgas enligt matriselementets storlek, medan **DrawPlaneMatrixColor** färgar punkten enligt matriselementet som är en färgkod.

Funktionerna **DrawPointSet**, **DrawVectorSet**, **DrawMatrix**, **PlotVectorField**, **PlotData2D**, **PlotData3D**, **DrawPlaneMatrix** och **DrawPlaneMatrixColor** kan förkortas **plot**. **plot** är således en funktion som beroende på argumenten skickar vidare exekveringen till lämplig specifik funktion.

Det finns också några funktioner som enbart ritar på skärmen, utan att ritningen baseras på argumentdata på samma tydliga sätt. Dessa är:

DrawPoint	DrawLine	DrawPolyLine

DrawPolygon	DrawRectangle	DrawCircle
DrawCircleArc	DrawPicture	DrawAngle
DrawRoundedRectangle	Measure2D	SaveScreenToFile
DrawEllipse		

Measure2D mäter vinklar och avstånd på skärmen, och **SaveScreenToFile** sparar den aktuella skärmen som en bitmappsfil eller vektorgrafisk illustration.

DrawCtrl	MoveCtrl	DelCtrl	CtrlUseGrid
CtrlAlignToGrid			

Kontroller kan läggas direkt i bildfönster (i stället för i konsolen eller i gränssnitt). Funktionerna **DrawCtrl**, **MoveCtrl**, **DelCtrl, CtrlUseGrid** och **CtrlAlignToGrid** styr detta.

Vektorgrafik

Det är inte effektivt att rita figurer bestående av räta linjer och plan med hjälp av de vanliga metoderna, eftersom rätlinjigheten möjliggör mycket snabbare rendering än vad som är möjligt med punkt-för-punkt-rendering. I själva verket räcker det ju med att rita en rät linje mellan start- och slutpunkt, utan att några punkter däremellan behandlas. Av denna anledning har AlgoSim 2.0 också stöd för vektorgrafiska illustrationer. En sådan lagras som en matris med information om de ingående primitiverna.

Andra fördelar med vektorgrafik är givetvis mycket små filstorlekar, möjlighet till obegränsad skalning och enkel redigering i efterhand.

Datatyp: Matris med ett ritobjekt per rad. Första raden innehåller allmän information. Kolonner:

dim (2, 3)	ver (1.0)	
”line”	pos	style
”polyline”	pos	style
”irregular shape”	pos	style
”vector”	pos	style
”tetragon”	pos	style
”rectangle”	pos	style
”square”	pos	style
”rounded rectangle”	pos	style
”triangle”	pos	style
”polygon”	pos	style
”circle”	pos	style
”ellipse”	pos	style
”cuboid”	pos	style
”sphere”	pos	style
”cylinder”	pos	style
”text”	pos	style
⋮		

DrawVectorGraphics ritar upp en vektorgrafisk bild i det aktuella bilfönstret. För mer information om pos- och style-attributen, se avsnittet ”Ritfunktioner” nedan.

AlgoSim kommer att kunna importera och exportera enkla SVG-filer.

Ritfunktioner

AlgoSim fungerar som ett tekniskt illustrationsprogram. Varje graffönster är en rityta. Följande kommandon kan användas för att rita på en sådan.

- **line**(r1, r2, style[, name])
- **polyline**(r1, r2, …, style[, name]) // Tekniskt består funktionen bara av två (tre) argument; första argumentet är *en matris* med koordinaterna r1, r2, ….
- **vector**(r1, r2, style[, name])
- **circle**(r1, r2, style[, name])
- **ellipse**(r1, r2, r3, style[, name])
- **tetragon**(r1, r2, r3, r4, style[, name])
- **rectangle**(r1, r2, style[, name])
- **square**(r1, r2, style[, name])
- **polygon**(r1, r2, …, style[, name]) // matris med koordinater
- **IrregularShape**(r1, r2, …, style[, name]) // matris med koordinater
- **RoundedRectangle**(r1, r2, r3, style[, name])
- **cuboid**(r1, r2, r3, r4, style[, name])
- **sphere**(r1, r2, style[, name])
- **cylinder**(r1, r2, r3, style[, name])
- **text**(r1, text, style[, name])
- **pen**(style[, name])
- **brush**(style[, name])

Argumenten <r<n>> är vektorer eller skalärer och anger position och storlek (ibland även form), och argumenten <style> är textsträngar som anger objektets stil. En cirkel kan t.ex. ha stilen ”border:dotted red 3 px; face:striped yellow 50 % transparent” medan en text kan ha stilen ”’Times New Roman’ 90° red bold italic shadow:2 px black 50 % transparent”. Om <r<n>>-argumenten inte anges så används musen för att ange position och storlek. Då används Ctrl för att flytta längs rutnätet (**SetGridSize**(x, y)) och Alt för att fästa längs andra objekt (på ett *smart* sätt). Alla objekt kan flyttas och storleksändras i efterhand och redigeras (style) genom att man dubbelklickar på dem. Internt lagras objekt som (enrads) vektorgrafiska variabler med prefixet AS:DRAW:<spnm>: där <spnm> är namnet på den aktuella ritytan (det aktuella bildfönstret).

PolyPlaneSurf

Funktionen **PolyPlaneSurf**(function, t1, t2, u1, u2) eller **ppsurf**(…) eller bara **surf**(…) används för att rita funktionsytor, inte genom att plotta punkter som tillhör ytorna (mängderna), utan genom att approximera funktionsytan med flera trianglar. Funktionen ger ifrån sig en vektorgrafisk bild bestående av sådana rektanglar; funktionen <function> är en parameterbeskrivning av ytan. Fördelen med detta förfarande är att uppritning och kamerarotation går betydligt mycket snabbare med en yta angiven på det här sättet. (Man skulle givetvis kunna tänka sig en endimensionell motsvarighet ”**PolyLineCurve”**, men på grund av de få beräkningar som krävs för att rita upp en enkel kurva är det knappast nödvändigt.)

Skärmdumpar. Filmsekvenser

Det är möjligt att spara det aktuella utseendet i ett bildfönster som en BMP-matris. Använd kommandot **BitmapFromView**([WinID]) där [WinID] är bildfönstrets ID (ordningsnummer i flikarna); som standard används det aktiva bildfönstret.

En filmsekvens (ungefär som en AVI-film utan ljud) representeras i AlgoSim av en 1×*n*-matris av bilder (d.v.s. BMP-matriser, 2D-matriser där varje punkt är en pixel med ett färgvärde). För att spela upp en sådan i ett eget fönster, använd kommandot **AnimateImages**(DataMatrix[, FrameInterval[, RepCount]]). <DataMatrix> är filmsekvensen, <FrameInterval> är antalet millisekunder varje bild visas (som standard 100) och <RepCount> är antalet repetitioner av sekvensen. <RepCount> kan vara ∞. Om <DataMatrix> är en 2×*n*-matris (notera 2:an!) kommer elementet $a_{(2,\,i)}$ att tolkas som antalet millisekunder bilden *i* skall visas (ignorerar <FrameInterval> för bilden).

Pixelgrafisk editor

Kommandot **EditMatrixAsBitmap** redigerar en BMP-matris i en pixelgrafisk editor (i all väsentlighet ett vanligt bildfönster omgivet av verktygsfält för pixelgrafisk redigering). Editorn har fullt stöd för RGBA-data. Ritverktygen (se Ritfunktioner) kan användas, och ritverktyg och olika lager kan sammanfogas på olika sätt (även enligt regler som användaren själv programmerar) – se bilden på sidan 69. Dessutom finns mängder av filter (effekter) som kan appliceras på bilden, eller en del av den. Flera av dessa effekter listas under Kommandoreferens på sidan 47. AlgoSim är följaktligen en riktig bildbehandlare!

Uppspelning av ljud

AlgoSim kommer med flera funktioner för att producera ljud från matematiskt material. Nedan listas några enkla funktioner av den här typen. Ett ljudobjekt är en *n*×1-matris med samplade värden för vågfunktionen, samt med metadata som anger bitdjup, samplingsfrekvens och relativ ljudstyrka.

Funktioner för att skapa ljud

EmptySound ger ett tomt ljudobjekt.

SineSound(freq, dur, amp[, SampleFreq]) skapar ett ljudobjekt med den rena sinustonen med frekvensen <freq> och uppspelningstiden <dur> med den relativa ljudstyrkan <amp>. Om inte samplingsfrekvensen <SampleFreq> anges, används den minsta möjliga samplingsfrekvensen. Om användaren explicit anger en samplingsfrekvens vars Nyquistfrekvens understiger (eller är nära att understiga) tonens frekvens <freq>, visas en varning.

SineSounds(freqs: set, dur, amp[, SampleFreq]) skapar ett ljudobjekt med de rena sinustonerna med frekvenserna i <freks> överlagrade.

SineSoundsSequence(freks: matrix) spelar upp en *n*×3-matris där varje rad har formen (frek, dur, amp), en rad i taget. Om någon kolonn saknas, används standardvärden.

sound(data: matrix, speed, amp) ger ett ljudobjekt av matrisen <data> (som gärna kan vara en värdemängd till en funktion) med farten <speed> värden per sekund och relativa ljudstyrkan <amp>.

FunctionSound(function, min, max, speed, amp) ger ett ljudobjekt av den reellvärda funktionen <function> där argumentet går från <min> till <max>. <speed> anger hur många heltalssteg den oberoende variabeln skall gå per sekund; standardvärdet är 1.

CombineSounds(s1, s2) kontrollerar att de två ljuden <s1> och <s2> har samma samplingsfrekvens och superponerar dem sedan de fått samma bitdjup.

AppendSound(ref: s1, s2) lägger till <s2> vid slutet av <s1>, sedan samplingsfrekvens och bitdjup kontrollerats (och eventuellt korrigerats).

InsertSound(ref: s1, s2, index) infogar ljudet <s2> i <s1> vid tidsindex <index>.

ChangeSoundAmp(ref: s1, ampd) ändrar den relativa ljudstyrkan i <s1> värdet <ampd>.

MultiChannelSound(s1, s2, …) ger ett flerkanaligt ljudobjekt (en *n*×*m*-matris) bestående av ljuden s1, s2, …. Speciellt om antalet argument är två (2) erhålles ett stereoljud. Om flerkanaliga ljud kan spelas upp med flera högtalare eller ej, beror givetvis på det lokala systemets konfiguration.

Funktioner för att spela upp ljud

För att spela upp ett ljud, ange ljudvariabelns namn. Det som returneras i konsolen är då en ljudspelare laddad med det aktuella ljudet. För att spela upp en ljudvariabel utan spelare, använd kommandot **PlaySound**(sound) eller **PlaySoundWait**(sound), om programmet skall vänta med att exekvera nästa kodrad tills ljudet är färdiguppspelat.

SaveSoundToFile sparar ljudvariabel till fil, och **LoadSoundFromFile** laddar ljudvariabel från fil. AlgoSim kommer att ha stöd för PCM-kodat WAV-ljud (RIFF-fil).

MIDI-funktionalitet

AlgoSim stödjer MIDI-formatet, vilket innebär att AlgoSim kan spela toner från musikinstrument, och till och med spela upp hela symfonier. Omkring 100 musikinstrument kan användas. För att spela upp en ton, använd kommandot **PlayNote**([InstrumentID,]tone[, intensity]). <InstrumentID> är ett heltal mellan 1 och 104 som anger vilket instrument som skall anges. <tone> är ett heltalsvärde i intervallet $[0, 127]$ och anger vilken ton som skall spelas. <intensity> är den relativa ljudstyrkan på tonen. Om <InstrumentID> utelämnas används det senast spelade instrumentet (vid första gången: instrument #1, piano), och om <intensity> utelämnas används standardljudstyrkan 1. För att slippa memorera instrumentens ID-nummer kan <InstrumentID> också anges som en sträng med instrumentets namn, såsom ”Acoustic grand piano” eller ”Church organ”. Vad instrumenten heter kan användaren själv ändra; i själva verket mappas instrumentnamnen till rätt ID-nummer via matrisen AS:MIDI:INSTRUMENTS, som från början har följande utseende.

1	Acoustic grand piano
2	Bright acoustic piano
3	Electric grand piano
4	Honky-tonk piano
5	Rhodes piano
6	Chorused piano
7	Harpsichord
8	Clavinet
9	Celesta
10	Glockenspiel
11	Music box
12	Vibraphone
13	Marimba
14	Xylophone
15	Tubular bells
16	Dulcimer
17	Hammond organ
18	Percussive organ
19	Rock organ
20	Church organ
21	Reed organ
22	Accordion
23	Harmonica
24	Tango accordion
25	Acoustic guitar (nylon)
26	Acoustic guitar (steel)
27	Electric guitar (jazz)
28	Electric guitar (clean)
29	Electric guitar (muted)
30	Overdriven guitar
31	Distortion guitar
32	Guitar harmonics
33	Acoustic bass
34	Electric bass (finger)
35	Electric bass (pick)
36	Fretless bass
37	Slap bass 1
38	Slap bass 2
39	Synth bass 1
40	Synth bass 2
41	Violin
42	Viola

43	Cello
44	Contrabass
45	Tremolo strings
46	Pizzicato strings
47	Orchestral harp
48	Timpani
49	String ensemble 1
50	String ensemble 2
51	Synth. strings 1
52	Choir Aahs
53	Synth. strings 2
54	Voice Oohs
55	Synth voice
56	Orchestra hit
57	Trumpet
58	Trombone
59	Tuba
60	Muted trumpet
61	French horn
62	Brass section
63	Synth. brass 1
64	Soprano sax
65	Alto sax
66	Synth. brass 2
67	Tenor sax
68	Baritone sax
69	Oboe
70	English horn
71	Bassoon
72	Clarinet
73	Piccolo
74	Flute
75	Recorder
76	Pan flute
77	Bottle blow
78	Shakuhachi
79	Whistle
80	Ocarina
81	Lead 1 (square)
82	Lead 2 (sawtooth)
83	Lead 3 (calliope lead)
84	Lead 4 (chiff lead)
85	Pad 1 (new age)
86	Lead 5 (charang)
87	Pad 2 (warm)
88	Lead 6 (voice)
89	Pad 3 (polysynth)
90	Lead 7 (fifths)
91	Pad 4 (choir)
92	Lead 8 (brass + lead)
93	Pad 5 (bowed)
94	Pad 6 (metallic)
95	Pad 7 (halo)
96	Pad 8 (sweep)
97	Guitar fret noise
98	Breath noise
99	Seashore
100	Bird tweet
101	Telephone ring

102	Helicopter
103	Applause
104	Gunshot

Även <tone> kan anges som en sträng med tonens namn, såsom ”D”, ”F♯” eller ”C3”. Mappning sker via matrisen AS:MIDI:TONES.

Instrumenten är av två typer. Vissa, såsom ”Acoustic grand piano”, är av den karaktären att ett anrop till **PlayNote** spelar tonen och låter den klinga av i sin egen takt, precis vad som händer om man på ett piano slår an en tangent. Andra instrument, emellertid, såsom ”Church organ”, är sådana att en tangent i verkligheten kan hållas intryckt hur längs som helst, och tonen upphör inte förrän tangenten släpps. Även i MIDI-systemet finns denna distinktion; för att ”stänga av” toner används kommandot **NoteOff**([InstrumentID][Tone]).

För att ändra instrument utan att spela en ton kan kommandot **SetInstrument**(InstrumentID) användas.

För att på låg nivå skicka ett eget kort MIDI-meddelande till ljudkortet, använd kommandot **MIDIMessage**(msg). (För referens, se Win32 API:s **midiOutShortMsg**.)

I AlgoSim kan data för MIDI-musik sparas i en kolonnmatris med element i $\mathbb{R}^4$ (och eventuellt texter). Varje $\mathbb{R}^4$-element i matrisen är en ton som spelas upp. Varje sådan ton har formatet

(interval, InstrumentID, tone, intensity)

där <interval> är antalet millisekunder mellan föregående och aktuell ton, <InstrumentID> är det nya instrumentet (0 betyder ”samma som föregående”), <tone> är tonen och <intensity> är den relativa ljudstyrkan (normalt 1). Om något element i matrisen är en sträng som inleds med AlgoSims bokmärkestecken ❦ så anger detta ett bokmärke; om en text ”bmk” förekommer i matrisen utan bokmärkessymbolen, så kommer AlgoSim att flytta uppspelningen till tonen direkt efter elementet ”❦bmk”.

För att spela upp en MIDI-sekvens i AlgoSims format, använd kommandot **PlayMIDIData**(DataMatrix).

Kommandot **MIDIEditor** visar en editor för MIDI-sekvenser, i vilken godtyckligt många instrument kan spelas in (eller anges manuellt) i olika kanaler. AlgoSim är följaktligen en musikstudio!

Inspelning av ljud

För att (från mikrofon eller annan ljudkälla) spela in ett nytt (wave-) ljud, använd kommandot **BeginSoundRecording**([duration]) för att påbörja inspelningen, och avbryt den med **EndSoundRecording**. Om <duration> anges avbryts inspelningen automatiskt efter <duration> sekunder. **SoundRecording** öppnar en kontrollpanel lik den gamla *Ljudinspelaren* i Windows 95-XP. (För mig obegripligt varför Microsoft *försämrade* den till Vista…)

Programkommandon. Flödeskontroll

AlgoSim 2.0 har en inbyggd tolk för det egna programspråket. Programkommandon (främst för s.k. flödeskontroll) infogas direkt i kommandoraden, eller i skript (som består av kommandoradsdata).

Nedan följer ett program som exemplifierar bruket av samtliga kommandon för flödeskontroll.

Två kommandon kan infogas på en rad med semikolon: <CMD1>; <CMD2>. Semikolon (;) fungerar precis som en radbrytare. På samma tema återfinnes symbolen ↲, som låter en rad fortsätta på nästa, precis som om radbrytningen saknades.

```
BEGIN PROGRAM <NAME>
          // Kommentar, resten av raden

          BEGIN COMMENT
                    Kommentar på flera rader
          END COMMENT

          GetArguments(”r, ϕ, ω”) // Lägger argumenten i r, ϕ
resp. ω, vilka  är LOKALA VARIABLER, som bara kan nås från pro-
grammet. Om en lokal variabel sammanfaller med en global, kan
den globala nås med kommandot GlobalVariable(VName: string): ob-
ject

          A ≔ 1; B ≔ 2

          C = 1 + 2 + 5 + 3 + 3 + 4 + 6 + 1 + 2 + 5 + 2 + 4 + ↲
          1 + 5 + 4 + 3

          IF <LOGIC>
          END IF

          <LOGIC> :> <COMMAND> // fungerar som IF .. END IF
          AS:ANSWER = AS:YES :> n = 0

          n = 0 <: AS:ANSWER = AS:YES

          UNLESS <LOGIC>
          END UNLESS

          CASE <OBJECT>
                    <OBJECT> DO <COMMAND>
                    <OBJECT> DO <COMMAND>
                    …
                    ELSE <COMMAND>
          END CASE

          CASE <OBJECT> IN
                    <SET> DO <COMMAND>
                    <SET> DO <COMMAND>
                    …
                    ELSE <COMMAND>
```

```
                END CASE

                FOR <n> START <a> STOP <b> STEP <s>
                        BREAK
                        NEXT
                END FOR

                FOREACH <VECTOR>, <SET>, <MATRIX> INDEXING <i>[, <j>]
                        <VECTOR>_<i>
                        <SET>_<i>
                        <MATRIX>_<(i,j)>
                        BREAK
                        NEXT
                END FOREACH

                WHILE <LOGIC>
                        BREAK
                        NEXT
                END WHILE

                INPUT <VARTYPE>:<VARNAME> WRITING <TEXT>
                WRITE <TEXT>
                WRITELN <TEXT>

        RETURN <OBJECT>
        // Om inget anges som returvärde, returneras PROGRESS:DONE; kom-
        mandot avslutar aktuellt program omedelbart.

        EXIT // avbryter programmet

        END PROGRAM
```

if-funktionen

if(LOGIC, OBJ1, OBJ2) returnerar OBJ1 om LOGIC är sant; annars returneras OBJ2. Funktionen **if** är följaktligen ett vid flera tillfällen effektivare alternativ till flödessatsen **IF**.

choice-funktionen

choice(”alfa”, ”beta”, ”gamma”, …) ritar upp en tabell i stil med

```
1. alfa    4. …
2. beta    5. …
3. gamma   6. …
```

varefter t.ex. inmatningen 1, ”alfa” eller klickande på alternativet ”alfa” i tabellen sätter AS:ANSWER = 1, AS:ANSWER2 = ”alfa” och fortsätter sedan programmet.

Bokmärken

En rad som inleds med symbolen ❦ är ett bokmärke, och texten som kommer efter symbolen på raden är bokmärkets namn. I program kan kommandot **goto** <bmnm> användas för att flytta exekveringspunkten till raden direkt under bokmärket vid namn <bmnm>. Om det inte finns något bokmärke med exakta namnet <bmnm>, så kommer en lista (likadan som i choice-funktionen) att låta användaren välja mellan de bokmärken som åtminstone innehåller <bmnm> som en delsträng. Om <bmnm> är en tom sträng, visas samtliga bokmärken i listan. **goto** *n* går till raden *n* i koden.

Dialogkonsoler

Även om den mesta interaktionen med användaren sker genom den vanliga konsolen, så finns också möjligheten att använda dialogrutor, men dessa är helt vanliga konsoler som visas i egna, små fönster.

DialogInfo(msg, modal) visar meddelandet <msg> i en dialogkonsol.

DialogChoice(msg, set of options, modal) visar en lista med val m h a choice-funktionen i en dialogkonsol.

DialogInput(msg) ber användaren om inmatning i en dialogkonsol.

Om det logiska värdet <modal> sätts till TRUE, kommer AlgoSim:s huvudfönster att bli obrukbart innan dialogkonsolen stängts. Kommandot **unmodal** (som kan anges i dialogkonsolen, förstås) ogör emellertid denna låsning.

Enhetskonvertering

AlgoSim har ett omfattande stöd för enhetskonvertering. Givetvis är enhetsdatabasen inte hårdkodad, utan består av vanliga AlgoSim-matriser som användaren själv kan redigera. Alla enhetsdatamatriser har namn som inleds med prefixet AS:UNITS:. En (del av en) typisk matris har följande utseende:

AS:UNITS:LENGTH

1	”metre”	”m”
0.1	”decimetre”	”dm”
0.01	”centimetre”	”cm”
0.001	”millimetre”	”mm”
10^{-6}	”micrometre”	”µm”
10^{-9}	”nanometre”	”nm”
10^{-12}	”picometre”	”pm”
10^{-15}	”femtometre”	”fm”
10^{-18}	”attometre”	”am”
10^{-21}	”zeptometre”	”zm”
10^{-24}	”yoctometre”	”ym”
10	”decametre”	”dam”
100	”hectometre”	”hm”
1000	”kilometre”	”km”
10^{6}	”megametre”	”Mm”
10^{9}	”gigametre”	”Gm”
10^{12}	”terametre”	”Tm”
10^{15}	”petametre”	”Pm”
10^{18}	”exametre”	”Em”
10^{21}	”zettametre”	”Zm”
10^{24}	”yottametre”	”Ym”
10^{10}	”Ångström”	”Å”
$9.46073047258 \cdot 10^{15}$	”light-year”	”ly”
149597870691	”astronomical unit”	”AU”
$3.08567758131 \cdot 10^{16}$	”parsec”	”pc”
0.0254	”inch/inches”	”in”
0.30480	”foot/feet”	”ft”

För att konvertera ett tal <a> från enheten <u1> till enheten <u2>, använd kommandot **UnitConvert**(a, u1, u2). <u1> och <u2> anges antingen med enhetsnamn eller enhetsbeteckning. Enhetskonvertering kan enklare utföras med kommandot **convert**, som läser in nästa rad i prompten. Denna rad kan ha ett format i stil med ”50 parsecs in astronomical units”, eller bara ”pc to AU”; om inget mätetal anges används senaste reella talet i konsolen. Denna funktionalitet efterliknar naturligt språk, och funktionen returnerar det konverterade mätetalet, och skriver dessutom ut en sträng i stil med ”50 parsecs = 10 313 240.3 astronomical units”.

AlgoSim kommer att innehålla åtskilliga enhetsdatamatriser för olika storheter.

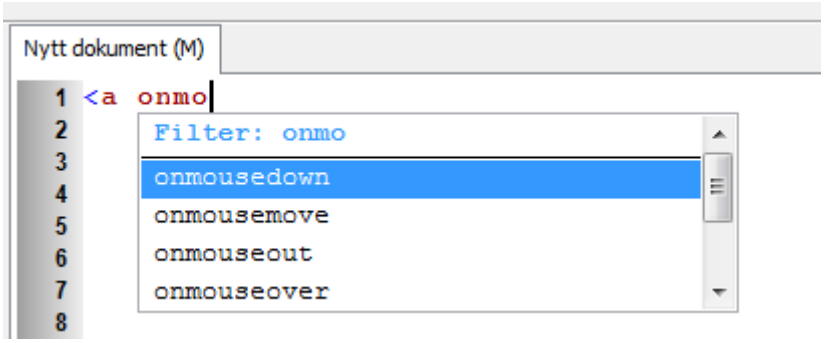
Inmatning

Följande tabell anger tangentbordskommandon för inmatning av specialtecken.

Kommando	1:a gången	2:a gången	3:e gången	4:e gången	5:e gången	
Ctrl+Alt+U	∪	∩				
Ctrl+Alt+I	∫	∬	∭	∮	∯	∰
Ctrl+Alt+E	∃	∈	∉	∋	∌	■
Ctrl+Alt+A	∧	⊼	∀			
Ctrl+Alt+O	∨	⊽	⊥			
Ctrl+Alt+X	⊻					
Ctrl+Alt+P	∥	∦	∂	¶		
Ctrl+Alt+R	ℝ	▭	▯			
=	=	≔	≠	≡		
*	·	×	*	∘	⊙	
<	<	≤	≪			
>	>	≥	≫			
Ctrl+Alt+S	⊂	⊊	⊄	⊃	⊋	⊅
Ctrl+Alt+F	∘					
Ctrl+Alt+W	∼					
Ctrl+Alt+Q	ℚ	⇒	⇐	⇔		
Ctrl+Alt+N	ℕ	¬	∇			
Ctrl+Alt+C	ℂ	∁	◯	♔	♕	♖
		♗	♘	♙		
Ctrl+Alt+D	\	∇				
Ctrl+Alt+H	*h*	ℏ				
Ctrl+Alt+L	↲	ℒ	ℓ	…	⋯	⋮
		⋰	⋱			
+	+	⊕				
-	−	⊖				
CTRL+ALT+T	∆	◿	∠	∡	⦍	
CTRL+ALT+M	∅					
CTRL+ALT+Z	ℤ					
CTRL+ALT+Y	⊙	☿	♀	♁	♂	♃
		♄	♅	♆	♇	
CTRL+ALT+J	❶	❷	❸	❹	❺	❻
		❼	❽	❾	❿	
CTRL+ALT+B	❦	⌬	∵	∴		
CTRL+ALT+K	♠	♣	♥	♦		
CTRL+ALT+G	Växlar till/från grekiskt tangentbordsläge, då a blir till α, b blir till β, c blir till γ osv.					
CTRL+ALT+V	←	↑	→	↓	↖	↗
		↘	↙	↔	↕	↶
		↷	↱	↲	↳	↴
		⇄	↹	⇅	⇋	⇤
		⇥	⇜	⇝	↩	↪
		↫	↬	↭		
: följt av =	≔					
Ctrl+U	Infogar Unicode-symbolen med koden intill markören.					
Ctrl+K	Visar den aktuella symboltabellen (denna), samt tillåter redigering av den.					

I kommandoraden används tangentbordspilarna upp och ned för att på aktuell rad infoga tidigare kommandon. För att verkligen flytta markören från inmatningsraden (den nedersta raden) används kommandona Ctrl+Uppil samt Ctrl+Nedpil. I kommandoraden markeras också tillhörande parenteser på ett tydligt sätt, och variabler/konstanter av olika datatyper formateras på lämpligt sätt; t.ex. formateras vektorer med fetstil i stället för kursiv stil.

För att enkelt kunna infoga rätt kommando bland de hundratals tillgängliga, kan *ListMenuHelper* användas. Detta är en listruta som visar möjliga kommandon, och som öppnas med knappen för kontextualmenyn på tangentbordet. Inmatning av de första tecknen i kommandots namn utesluter alla objekt i listan vilka inte inleds med dessa tecken. Principen för en *ListMenuHelper* illustreras i skärmbilden från min (XHTML-) webbeditor Easy Coder nedan.



Figur 2 ListMenuHelper i Easy Coder 7.0

Editorn har stöd för upphöjd och nedsänkt text. En upphöjd text <txt> tolkas som ^(<txt>) och en nedsänkt text <txt> tolkas som _(<txt>). Till exempel kan man i editorn skriva

$$3 \cdot x^3 + x^2 - 2 \cdot x + 1$$

och

$$A_{1,2}$$

(där A är en matris).

Inställningar

Inställningar är vanliga variabler, med prefixet AS. Nedan följer några av dessa.

AS:DIG	REAL	Antalet siffror hos reella tal som skrivs ut, som mest [12].
AS:SCI	LOGIC	Grundpotensform [FALSE]
AS:ENG	LOGIC	Använd prefixen … G, M, k, m, µ, n … [FALSE]
AS:ANGLE	REAL	Kvoten mellan aktuell vinkelenhet och vinkelenheten radianer [1]
AS:DIFFH	REAL	Det lilla tal *h* som används vid numerisk beräkning av derivator [10^{-6}]
AS:INTN	REAL	Det stora tal *N* som är antalet områden vars area beräknas vid numerisk beräkning av integraler [500]
AS:ITER	REAL	Det lagom stora tal som anger antalet iterationer vid numerisk ekvationslösning med Newton-Raphsons metod [50]
AS:TALKATIVE	REAL	Anger hur pass pratsamt AlgoSim skall vara. 4: Kallprat, tips, varningar och fel 3: Tips, varningar och fel 2: Varningar och fel 1: Fel 0: *(inget)* [3]
AS:STRINGMODE	LOGIC	Anger huruvida strängar skall formateras enligt den basala MediaWiki-syntaxen. [TRUE]
AS:MAXIMIZETOFULLSCREEN	LOGIC	Anger huruvida maximeraknappen går till helskärmsläge. [FALSE]
AS:CLIPBOARDCOLLECT	LOGIC	Anger huruvida urklipp samlas in under en egen flik i konsolen. [FALSE]

Operatortabell

De operatorer som används i AlgoSim är inte hårdkodade. Användaren kan ta bort, förändra och lägga till operatorer. Samtliga operatorers definitioner finns lagrade i matrisen AS:OPTABLE med formatet nedan.

Symbol	”left”/”right”/”binary”/”middle”	Funktionsnamn

Den andra kolonnen anger om operatorn är binär eller unär, och i det senare fallet var operanden står i förhållande till operatorn (till vänster, till höger eller mellan operatorns två tecken). Som standard ser matrisen ut som följer:

”+”	”binary”	”OP:PLUS”
”−”	”binary”	”OP:MINUS”
”·”	”binary”	”OP:DOT”
”/”	”binary”	”OP:DIV”
”^”	”binary”	”OP:POWER”
”×”	”binary”	”OP:CROSS”
”!”	”left”	”OP:FACT”
”||”	”middle”	”OP:ABS”
”%”	”left”	”OP:PERCENT”
”‰”	”left”	”OP:PERMILLE”
”°”	”left”	”OP:DEG”
”rad”	”left”	”OP:RAD”
” ¯”	”binary”	”OP:INDEX”
”=”	”binary”	”OP:EQUALS”
”<”	”binary”	”OP:LESSTHAN”
”≤”	”binary”	”OP:LESSEQUAL”
”>”	”binary”	”OP:GREATER”
”≥”	”binary”	”OP:GREATEREQUAL”
”≠”	”binary”	”OP:NOTEQUAL”
”≈”	”binary”	”OP:APPROX”
”|”	”binary”	”OP:DIVIDES”
” ∥”	”binary”	”OP:PARALLEL”
”∦”	”binary”	”OP:NOTPARALLEL”
”⊥”	”binary”	”OP:ORTHOGONAL”
”⊙”	”binary”	”OP:CIRCDOT”
”*”	”left”	”OP:ASTERIX”
”∪”	”binary”	”OP:UNION”
”∩”	”binary”	”OP:INTERSECTION”
”\”	”binary”	”OP:SETMINUS”
”∁”	”right”	”OP:COMPLEMENT”
”⊕”	”binary”	”OP:CIRCPLUS”
”∈”	”binary”	”OP:IN”
”∋”	”binary”	”OP:NI”
”∉”	”binary”	”OP:NOTIN”
”∌”	”binary”	”OP:NOTNI”
”⊆”	”binary”	”OP:SUBSET”
”⊊”	”binary”	”OP:PROPERSUBSET”
”⊇”	”binary”	”OP:SUPERSET”
”⊋”	”binary”	”OP:PROPERSUPERSET”
”∘”	”binary”	”OP:COMPOSITE”
”∧”	”binary”	”OP:AND”
”∨”	”binary”	”OP:OR”
”⊼”	”binary”	”OP:NAND”
”⊽”	”binary”	”OP:NOR”
”⊻”	”binary”	”OP:XOR”

”¬”	”right”	”OP:NOT”		
”⇒”	”binary”	”OP:IMPLIES”		
”⇐”	”binary”	”OP:IMPLIESLEFT”		
”⇔”	”binary	”OP:EQUIVALENT”		
”↦”	”binary”	”OP:MAPSTO”		
”@”	”right”	”OP:ADDRESS”	”right”	
”≔”	”binary”	”OP:ASSIGN”	”left”	”rtl”

Notera att en operator bara är en genväg till en funktion. En operator kan som mest ta två argument (operander), som då står på varsin sida om operatorn. Uttrycket ”1+1” är alltså en förkortning för ”OP:PLUS(1,1)”, och ”1+1=2” är en förkortning för ”OP:EQUALS(OP:PLUS(1,1),2)”. Operatorernas prioritet anges genom ordningen på raderna i matrisen – höst prioritet har den översta raden. Matrisen ovan är emellertid för överskådlighetens skull sorterad efter kategori i stället för prioritet.

Den fjärde kolonnen i matrisen är, som synes, oanvänd *förutom* för operatorerna @ och ≔. Denna kolonn anger om någon operand måste tolkas *rått*, vilket innebär att den ursprungliga textsträngen med texten vid sidan om operatorn skickas till funktionen i stället för det värde som erhålls när AlgoSim bearbetbar operanden. I fallet @ är det nödvändigt med en rå tolkning; t.ex., om variabeln a har värdet 5, så kommer annars uttrycket @a vara likvärdigt med @5 ur funktionen OP:ADDRESS:s synvinkel. Detsamma gäller för tilldelningsoperatorn ≔, som måste veta namnet på variabeln, snarare än dess aktuella värde. Namnet är ju textsträngen, och inte det tidigare värdet på variabeln (eller ett felmeddelande om det är första gången variabelnamnet används). Utan möjlighet för rå tolkning, skulle variabelnamn behöva anges som strängar, inom citationstecken.

Operatorn ≔ ger ifrån sig värdet av den högra operanden, varför multipeltilldelningar i stil med $a \coloneqq b \coloneqq c \coloneqq 0$ är möjliga. Detta är naturligtvis endast möjligt om uttrycket bearbetas från höger till vänster – i motsats till alla andra operatorer – vilket anges av värdet ”rtl” i den femte kolonnen för operatorn ≔.

Symbolhantering

AlgoSim kommer att ha ett grundläggande stöd för symbolhantering. Uttryck som hanteras lagras som strängar. Funktionerna **SymbDiff**, **SymbInt**, **simplify**, **expand**, **factor** och **SymbSubst** är centrala. För att använda ett uttryck lagrat som en sträng, kan funktionsskaparen ↦ eller funktionen **eval** användas. Nedan visas några exempel.

```
SymbDiff(”x·sin(x^2)”, ”x”)
            ”x·cos(x^2)·2·x + sin(x^2)”
simplify(ans)
            ”2·x^2·cos(x^2) + sin(x^2)”
y ≔ ”x” ↦ ans
            x ↦ 2·x^2·cos(x^2) + sin(x^2)
y(0)
            0
SymbInt(”2·x^2·cos(x^2) + sin(x^2)”, ”x”)
            ”x·sin(x^2) + C”
SymbSubst(ans, ”C”, -1)
            ”x·sin(x^2) - 1”
SymbSubst(ans, ”x”, π)
            ”π·sin(π^2) - 1”
eval(ans)
            -2,3518311
expand(”(a+b)^4”)
            ”a^4 + 4·a^3·b + 6·a^2·b^2 + 4·a·b^3 + b^4”
factor(”x^3 - 3·x2 - 4”)
            ”(x + 1)(x - 2)^2”
SymbTaylor(”e^x”, ”x”, 5)
            ”1 + x + (1/2)·x^2 + (1/6)·x^3 + (1/24)·x^4 + (1/120)·x^5”
PartFractions(”(x-19)/(x^2-3·x-10)”)
            ”3/(x+2) - 2/(x-5)”
```

Kommandogränssnitt

I följande avsnitt diskuteras AlgoSims inbyggda funktioner för att kommunicera med operativsystemet och vissa interna system.

Editorsgränssnitt

Följande kommandon styr konsolen i AlgoSim samt Windows urklippsminne.

CopyTextToClipboard	GetTextFromClipboard	GetClipboardTextLength
GetClipboardTextSize	ClipboardContainsText	CopyBitmapToClipboard
GetBitmapFromClipboard	GetClipboardBitmapDimensions	GetClipboardBitmapSize
ClipboardContainsBitmap	GetTabList	NewTabBehind
NewTabFocus	CloseTab	CloseAllButCurrentTab
CloseAllTabs	GetTabName	GetTabIndex
SetTabName	SetTabName	TabExists
GetCaretPos	SetCaretPos	GetCurrentLine
SetCurrentLine	GetSelStart	SetSelStart
GetSelLength	SetSelLength	GetSelText
SetSelText	CopySelectedTextToClipboard	CutSelectedTextToClipboard
PastTextFromClipboard	SelectEntireLine	SelectAllText
SelectText	SelectLine	ClearAllText
ClearSelection	GetLineLength	GetFileLength
GetCharacter	GetLine	GetText
SetText	AppendText	AppendLines
SetInterfaceTab	RestoreInterfaceTab	RedrawCurrentTab

SetInterfaceTab gör så att kommandona ovan som arbetar i den aktuella fliken, börjar arbeta i denna flik, oavsett vilken som visas för användaren. **RestoreInterfaceTab** återställer till normalt läge igen.

Kommandona samt variablerna nedan anger editorskomponentens (textinmatningskontrollens) funktion.

SetEditorOverwriteMode	AS:EDITOR:OVERWRITE	SetEditorFontName
AS:EDITOR:FONTNAME	SetEditorFontSize	AS:EDITOR:FONTSIZE

Menygränssnitt

Följande kommandon tillåter skapande av egna (anpassade) menyer och flytande verktygsfält i AlgoSim 2.0. Rootmenyn heter ”ROOT”.

CreateMenu	CreateMenuItem	DeleteMenu
DeleteMenuItem	SetMenuFormat	MergeMenus
ChangeMenuItemIndex	GetAllMenus	GetAllMenuItems
CreateToolbar	CreateToolbarButton	DeleteToolbar
DeleteToolbarButton	SetToolbarFormat	ShowToolbar
HideToolbar	ToggleToolbar	MoveToolbar
MoveToolbarD	GetToolbarPos	DockToolbar
UndockToolbar	MergeToolbars	ChangeToolbarButtonIndex
GetAllToolbars	GetAllToolbarButtons	

Gränssnitt till filsystemet

Följande kommandon hanterar kataloger och filer i Windows filsystem. AlgoSim fungerar som en filhanterare!

ChangeDir (cd)	DirUp	ListDirs
ListFiles	ListDirsAndFiles (ls, dir)	RenameFile/RenameFiles
DeleteFile/DeleteFiles	MoveFile/MoveFiles	CopyFile/CopyFiles
RenameDir/RenameDirs	DeleteDir/DeleteDirs	MoveDir/MoveDirs
CopyDir/CopyDirs	ShellOpenFile	GetFileURL
GetFileSize	GetFileCreated	GetFileModified
GetFileAttributes	ExtractFileExt	GetFilesPattern
GetDirsPattern	FileExists	DirExists
DownloadFile	WebAvailable	

Gränssnitt till FTP

Följande kommandon hanterar FTP (File Transfer Protocol). AlgoSim fungerar som en FTP-klient!

FTPOpen	FTPClose	FTPChangeDir (ftpcd)
FTPSetASCII	FTPGetFile (ftpget)	FTPSendFile (ftpsend)
FTPRenameFile	FTPRemoveFile	FTPRemoveDir
FTPHelp	FTPQuit	FTPListDirsAndFiles (ftpls, ftpdir)
FTPGetObject	FTPSendObject	

Gränssnitt till Windows systemregister

Följande kommandon ger tillgång till Windows systemregister. AlgoSim fungerar som en registereditor!

REGOpenKey	REGCreateKey	REGDeleteKey
REGRenameKey	REGWriteString	REGReadString
REGWriteReal	REGReadReal	REGWriteBool
REGReadBool	REGDeleteValue	REGClearKey
REGKeyExists	REGValueExists	

Säkerhetslås

Av säkerhetsskäl är alla funktioner som rör systemregistret och som rör filer som inte ligger i AS-personkatalogen låsta från början.

SecurityUnlock låser upp programmet, och **SecurityLock** låser systemet igen. **SetSecurityPassword** ändrar lösenordet (om föregående lösenord angivits riktigt) medan **BlankSecurityPassword** tar bort lösenordet (om föregående lösenord angivits riktigt).

Editorer och visare

AlgoSim:s konsol, upptäcker man, är egentligen en högst avancerad texteditor. Gräver man ännu mer lär man upptäcka att den till och med är en editor för formaterade textdokument, d.v.s. textdokument som kan formateras (olika typsnitt, teckenstorlekar, -färger och -effekter i samma dokument, och även bilder och andra objekt kan infogas). I själva verket bygger AlgoSim på två fundamentala komponenter: *panelkomponenten* och *konsolen*. (Se bilden under Gränssnitt på sidan 7.)

Panelkomponenten

Panelkomponenten är den komponent som återfinnes i en eller flera instanser i varje fönster, så att fönstren är helt täckta av dessa i olika konfigurationer. Panelkomponentens enda syfte är att kunna hålla en eller flera konsoler under olika flikar. Endast en konsol visas åt gången i en panelkomponent.

Konsolen

Titta på bilden under Gränssnitt på sidan 7, på vilken det tydligt framgår att AlgoSim i standardutförandet kommer att ha fem paneler synliga. Att ”konsolen”, ”bildkonsolen” och ”extrakonsolen” är konsoler bör inte komma som någon överraskning. Att bildfönstren är konsoler, emellertid, kan få några att höja på ögonbrynen. Bildfönstren är konsoler (”textdokument”) som bara visar ett grafobjekt. Fördelen med denna design är givetvis att samma allmänna funktioner som kan utföras i vanliga konsoler (”textdokument”) också kan utföras på graffönstren (t.ex. utskrift). I allmänhet är det en elegant lösning att använda så generaliserade system som möjligt.

De övriga konsolerna som visas på bilden är i *read only*-läge i den omfattningen att endast den sista raden kan redigeras; det ligger ju i kommandoradsdesignens natur att man inte har något behov av att ändra tidigare kommandon eller resultat.

Ett slående argument för att konsolen bör vara en fullfjädrad editor för formaterade textdokument är att den skall kunna visa matriser, graffönster, bilder, ljudspelare och andra UI-kontroller som utmatningar. Detta vore inte möjligt i en rent textbaserad konsol.

Följande objekt kan infogas i en konsol (texteditor):

- Matris (tabell)
- Graffönster
- Kontroll
- Gränssnitt (uppsättning kontroller)
- Bild

Editorer

AlgoSim kommer med en avancerad och mycket effektiv texteditor för redigering av oformaterade texter (samt, om man vill, även formaterade sådana). Denna editor kan användas till att redigera program (skript) och strängar, men även till att redigera vanliga textdokument. I funktionalitet och effektivitet kommer denna att slå Windows Anteckningar (Notepad) med mycket stor marginal. Denna texteditor är givetvis den vanliga konsolen (fast i en texteditormiljö).

En tabelleditor för redigering av matriser existerar. Inmatning av uttryck i matriscell gör att cellen får det värde som returneras av uttrycket (precis som om det hade exekverats i konsolen). Matriseditorn är förstås en vanlig konsol fylld med en tabell (matris) i ett speciellt gränssnitt.

Varje graffönster fungerar som illustrationsprogram, som nämnt under Ritfunktioner på sidan 24. En bitmappseditor (d.v.s. en matriseditor men där matrisen tolkas som en bitmapp) existerar också; denna är fullt kapabel att utföra alla grundläggande operationer för bildredigering direkt i dess grafiska användargränssnitt.

Visare

AlgoSim kommer också med en bildvisare, som fokuserar på enkelhet och tekniska detaljer. Denna kan användas för att analysera bitmappsmatriser i AlgoSim, men också för att öppna och visa bilder i datorns bildbibliotek. All EXIF-data visas vid sidan av bilderna. Visaren är strikt *read only*, främst av datasäkerhetsskäl (så att man inte av misstag modifierar bilder), men har en funktion som kan importera bilder (t.ex. från datorns bildbibliotek) till bitmappsmatriser i AlgoSim. Bildvisaren kan förstora bilder både med och utan interpolation.

En liknande ”visare” för ljud finns med; denna skall också kunna spela in ljud från datorns mikrofon (eller annan ljudenhet).

Jag utvecklar för närvarande bibliotek för att kunna importera/exportera bilder av formaten BMP, XBM, PNG och JPG, samt ljud med formatet WAV (RIFF, PCM). Vektorgrafik kan användas med SVG och ”vektorljud” med MIDI (SMF) eller RMI (SMF i RIFF-container).

Hexadecimal editor

AlgoSim kommer också med en hexadecimal editor för att läsa och redigera filer på bitnivå. Vidare finns funktioner för att i programkod läsa och skriva byten till filer, varför användare av AlgoSim själva kan programmera funktioner för att läsa och skriva till ”okända” filtyper.

Medföljande bibliotek

Med AlgoSim kommer ett bibliotek med matematiska och fysikaliska konstanter, tabeller (matriser) och funktioner att följa med. Användaren får välja vilka delar hon själv vill implementera. Dessa bibliotek är helt vanliga AlgoSim-objekt, och kan redigeras helt fritt. T.ex. följer periodiska systemets data med, och med ett program skrivet i AlgoSims eget språk, renderas själva tabellen (periodiska systemet), med data från biblioteket. Många funktioner i AlgoSim är just skrivna i AlgoSims eget språk; fördelen med detta är att slutanvändaren själv kan programmera om dem.

Exempelprogram

För att visa möjligheterna med AlgoSim, kommer flera imponerande exempelprogram att bifogas, och, om användaren tillåter det, starta automatiskt med demoprogrammet som startar första gången AlgoSim körs. Nedan följer några exempel på dylika program, som mycket enkelt kan skapas med AlgoSim 2.0:

- Många plana och tredimensionella grafer kommer att visas. Dels visas hur enkelt dessa mängder skapas, och sedan visas vilken imponerande uppsättning av operationer som kan utföras på dem.
- Integraler, derivator och grafer av sådana studeras, och det faktum att funktionen **FindExactExpression** får AlgoSim att verka symbolhanterande illustreras.
- Vågsimulator, som visar två tidsberoende vågor och (den superponerade) summavågen. Fenomen som svävning och stående vågor illustreras enkelt.
- Simulator för geometrisk optik.
- Simulator för en fjäder och matematisk pendel, och kopplingen till den differentialekvation som ger upphov till sinusuttrycket.
- Simulering som visar varför experiment med inskjutna α-partiklar mot atomer visar att atomen snarare ser ut som ett planetsystem än en ”gröt”.
- Simulator för dopplereffekt.
- Simulator för enkel Newtonsk fysik. N partiklar med färger c_i, massor m_i, laddningar q_i, initialpositioner $\mathbf{r}_i$ och initialhastigheter $\dot{\mathbf{r}}_i$ ritas upp i planet eller rummet, i vilket ett godtyckligt kraftfält (t.ex. ett potentialfält) existerar oberoende av partiklarna. Dessutom kan partiklarna sinsemellan växelverka med varandra via såväl avståndskrafter som kontaktkrafter. På detta sätt kan gaser, partiklar som studsar i gravitationsfält, planeter kring en stjärna, formationen av ett solsystem, elektroner i magnetfält och mycket annat simuleras, vilket i många sammanhang kan vara närmast oumbärligt.
- Pedagogiskt program som illustrerar bruk av linjära avbildningar i planet och rummet.
- Pedagogiskt program som illustrerar skalär- och vektorfält och plottar grafer för gradient, divergens och rotation
- Program som flyger över den (tredimensionella) energidalen.
- Program som låter användaren flytta N stycken partiklar med samma eller olika massa och som i realtid använder funktionen **CenterOfMass** för att markera masscentrum av systemet.

Referensdel

Kommandoreferens

I denna del beskrivs många (men inte alla) kommandon som ingår i en standardinstallation.

Funktion	Argument	Returvärde	Beskrivning
sin	$\mathbb{R}$	$\mathbb{R}$	Ger sinus av vinkeln.
cos	$\mathbb{R}$	$\mathbb{R}$	Ger cosinus av vinkeln.
tan	$\mathbb{R}$	$\mathbb{R}$	Ger tangens av vinkeln.
cot	$\mathbb{R}$	$\mathbb{R}$	Get cotangens av vinkeln.
arcsin	$\mathbb{R}$	$\mathbb{R}$	Arcussinus
arccos	$\mathbb{R}$	$\mathbb{R}$	Arcuscosinus
arctan	$\mathbb{R}$	$\mathbb{R}$	Arcustangens
arctan2	$y \in \mathbb{R}$, $x \in \mathbb{R}$	$\mathbb{R}$	Ger den polära koordinaten ϕ för punkten (x, y). I första kvadranten lika med $\arctan(y/x)$.
arccot	$\mathbb{R}$	$\mathbb{R}$	Arcuscotangens
sinh	$\mathbb{R}$	$\mathbb{R}$	Sinus hyperbolicus
cosh	$\mathbb{R}$	$\mathbb{R}$	Cosinus hyperbolicus
tanh	$\mathbb{R}$	$\mathbb{R}$	Tangens hyperbolicus
coth	$\mathbb{R}$	$\mathbb{R}$	Cotangens hyperbolicus
arsinh	$\mathbb{R}$	$\mathbb{R}$	Invers till sinus hyperbolicus
arcosh	$\mathbb{R}$	$\mathbb{R}$	Invers till cosinus hyperbolicus
artanh	$\mathbb{R}$	$\mathbb{R}$	Invers till tangens hyperbolicus
arcoth	$\mathbb{R}$	$\mathbb{R}$	Invers till cotangens hyperbolicus
sinc	$\mathbb{R}$	$\mathbb{R}$	$\sin x \,/x$ samt 1 i origo
exp	$\mathbb{R}$	$\mathbb{R}$	Ger e upphöjt till argumentet.
ln	$\mathbb{R}$	$\mathbb{R}$	Ger den naturliga logaritmen av argumentet.
lg	$\mathbb{R}$	$\mathbb{R}$	Ger 10-logaritmen av argumentet.
log	$a \in \mathbb{R}$, $x \in \mathbb{R}$	$\mathbb{R}$	Ger a-logaritmen av argumentet x.
abs	$\mathbb{R}$	$\mathbb{R}$	Ger absolutbeloppet av argumentet.
sgn, sign, signum	$\mathbb{R}$	$\mathbb{R}$	Ger tecknet hos argumentet.
C	$a \in \mathbb{R}$, $b \in \mathbb{R}$	$\mathbb{R}$	Ger antalet kombinationer.
P	$a \in \mathbb{R}$, $b \in \mathbb{R}$	$\mathbb{R}$	Ger antalet permutationer.
ceil	$\mathbb{R}$	$\mathbb{R}$	Avrundar argumentet uppåt till närmaste heltal.
floor	$\mathbb{R}$	$\mathbb{R}$	Avrundar argumentet nedåt till närmaste heltal.
round	$\mathbb{R}$	$\mathbb{R}$	Avrundar argumentet till närmaste heltal.
frac	$\mathbb{R}$	$\mathbb{R}$	Ger decimaldelen av argumentet.
int	$\mathbb{R}$	$\mathbb{R}$	Ger heltalsdelen av argumentet.
rescale	$x \in \mathbb{R}$, $a \in \mathbb{R}$, $b \in \mathbb{R}$, $A \in \mathbb{R}$, $B \in \mathbb{R}$	$\mathbb{R}$	Skalar om argumentet x från skalan $[a, b]$ till skalan $[A, B]$.
mod	$p \in \mathbb{R}$, $q \in \mathbb{R}$	$\mathbb{R}$	Ger resten vid heltalsdivisionen p/q.
fact	$\mathbb{R}$	$\mathbb{R}$	Ger fakultet av argumentet.
IsPrime	$\mathbb{R}$	LOGIC	Undersöker om argumentet är ett primtal.
prime	$n \in \mathbb{R}$	$\mathbb{R}$	Ger det n:te primtalet.
NextPrime	$\mathbb{R}$	$\mathbb{R}$	Ger primtalet som kommer närmast efter argumentet.
PrevPrime	$\mathbb{R}$	$\mathbb{R}$	Ger primtalet som kommer närmast före argumentet.
IsFibonacci	$\mathbb{R}$	LOGIC	Undersöker om argumentet är ett Fibonaccital.

Fibonacci	$n \in \mathbb{R}$	$\mathbb{R}$	Ger det n:te Fibonaccitalet.
NextFibonacci	$\mathbb{R}$	$\mathbb{R}$	Ger Fibonaccitalet som kommer närmast efter argumentet.
PrevFibonacci	$\mathbb{R}$	$\mathbb{R}$	Ger Fibonaccitalet som kommer närmast före argumentet.
rationalize	$\mathbb{R}$	$\mathbb{R}^2$	Ger en rationell approximation p/q av argumentet, som en vektor (p, q).
FindExactExpression	$\mathbb{R}$	TEXT	Försöker finna ett exakt uttryck som approximerar argumentet (t.ex. $\sqrt{3}/2$).
wave	*type*: TEXT, $A \in \mathbb{R}$, $k \in \mathbb{R}$, $\omega \in \mathbb{R}$, $\delta \in \mathbb{R}$, *time*: LOGIC	FUNCTION	Ger en funktion som är en våg av typen *type* (”sine”, ”square”, ”triangle”, ”sawtooth” m.m.) med amplituden A, initialfasen δ, vågtalet k och vinkelfrekvensen ω (om den är tidsberoende, d.v.s. rörlig).
dim	$\mathbb{R}^n$	$\mathbb{R}$	Ger dimensionen (antalet komponenter) hos vektorn.
abs	$\mathbb{R}^n$	$\mathbb{R}$	Ger vektorns absolutbelopp (norm).
ZeroVector	$\mathbb{R}$	$\mathbb{R}^n$	Ger nollvektorn av argumentets dimension.
norm	$\mathbb{R}^n$	$\mathbb{R}^n$	Ger argumentvektorn efter normering.
norm	REF: $\mathbb{R}^n$	$\mathbb{R}^n$	Normerar argumentvektorn.
max	$\mathbb{R}^n$	$\mathbb{R}$	Ger det största värdet i vektorn.
min	$\mathbb{R}^n$	$\mathbb{R}$	Ger det minsta värdet i vektorn.
mean	$\mathbb{R}^n$	$\mathbb{R}$	Ger det aritmetiska medelvärdet av komponenterna i vektorn.
med	$\mathbb{R}^n$	$\mathbb{R}$	Ger medianvärdet av komponenterna i vektorn.
q1	$\mathbb{R}^n$	$\mathbb{R}$	Ger första kvartilen av komponenterna i vektorn.
q3	$\mathbb{R}^n$	$\mathbb{R}$	Get tredje kvartilen av komponenterna i vektorn.
ForEach	$\mathbb{R}^n$, FUNC	$\mathbb{R}^n$	Ger vektorn sedan funktionen verkat på varje komponent i vektorn.
ForEach	REF: $\mathbb{R}^n$, FUNC	$\mathbb{R}^n$	Låter funktionen verka på varje komponent i vektorn.
FillVect	REF: $\mathbb{R}^n$, $\mathbb{R}$	$\mathbb{R}^n$	Fyller vektorn med det reella talet, så att varje komponent i vektorn sätts lika med talet.
GCD	$\mathbb{R}^n$	$\mathbb{R}$	Ger största gemensamma delaren av komponenterna i vektorn.
LCM	$\mathbb{R}^n$	$\mathbb{R}$	Ger minsta gemensamma nämnaren av komponenterna i vektorn.
sort	$\mathbb{R}^n$	$\mathbb{R}^n$	Ger vektorn där komponenterna sorterats i storleksordning.
sort	REF: $\mathbb{R}^n$	$\mathbb{R}^n$	Sorterar komponenterna i vektorn i storleksordning.
ReverseOrder	$\mathbb{R}^n$	$\mathbb{R}^n$	Ger vektorn med komponenterna i motsatt ordningsföljd.
ReverseOrder	REF: $\mathbb{R}^n$	$\mathbb{R}^n$	Vänder ordningsföljden på komponenterna i vektorn.
sum	$\mathbb{R}^n$	$\mathbb{R}$	Ger summan av samtliga komponenter i vektorn.
product	$\mathbb{R}^n$	$\mathbb{R}$	Ger produkten av samtliga komponenter i vektorn.
LeastSquare	$\mathbb{R}^n$, $\mathbb{R}^n$	$\mathbb{R}^2$	Väljer ut parametrarna k och m så att den räta linjen $y = kx + m$ bäst

			passar till den plott som erhålles till grafen (x, y) där x-värdena kommer från den första vektorn och y-värdena kommer från den andra vektorn.
SwapCoord	$\mathbb{R}^n$, $a \in \mathbb{R}$, $b \in \mathbb{R}$	$\mathbb{R}^n$	Ger argumentvektorn där komponent a och b bytt plats med varandra.
SwapCoord	REF: $\mathbb{R}^n$, $a \in \mathbb{R}$, $b \in \mathbb{R}$	$\mathbb{R}^n$	Byter plats på komponent a och b i vektorn.
InsertComponent	REF: $\mathbb{R}^n$, *index* $\in \mathbb{R}$, *NewComponent* $\in \mathbb{R}$	$\mathbb{R}^n$	Infogar komponenten *NewComponent* på index *index* i vektorn (så att index förskjuts med skillnaden +1 efter detta index).
IndexOf	$\mathbb{R}^n$, $\mathbb{R}$	$\mathbb{R}$	Ger komponentindexet för den första förekomsten av det reella talet som komponent i vektorn.
IndexOfVector	$\mathbb{R}^n$, $\mathbb{R}$	$\mathbb{R}$^m	Ger vektorn med samtliga komponentindex för de komponenter i vektorn vilka är lika med det reella talet.
CoordLogic2DToScreen	$\mathbb{R}^2$	$\mathbb{R}^2$	Omvandlar argumentkoordinaten, given i det aktuella bildfönstrets koordinatsystem, till skärmens pixelsystem.
CoordScreenToLogic2D	$\mathbb{R}^2$	$\mathbb{R}^2$	Omvandlar argumentkoordinaten, given i skärmens pixelsystem, till det aktuella bildfönstrets koordinatsystem.
CoordLogic3DProjectOnScreen	$\mathbb{R}^3$	$\mathbb{R}^2$	Ger koordinaten på skärmens pixelsystem för argumentvektorn given i det aktuella bildfönstrets tredimensionella koordinatsystem.
CoordScreenToLogic3D	$\mathbb{R}^2$, $\mathbb{R}$	$\mathbb{R}^3$	Ger koordinaten i det aktuella bildfönstrets tredimensionella koordinatsystem som svarar mot argumentkoordinaten given i skärmens pixelsystem och på djupet givet av det reella talet.
CreateBasis	$\mathbb{R}^n$_1, $\mathbb{R}^n$_2, …, $\mathbb{R}^n$_n	M	Ger basmatrisen för basvektorerna.
BasisTransformationMatrix	M, M	M	Ger transformationsmatrisen mellan de två baserna, angivna med deras basmatriser.
TransformCoordinates	M, $\mathbb{R}^n$, M	$\mathbb{R}^n$	Ger vektorn given i basen som svarar mot den första basmatrisen i basen given av den andra basmatrisen.
LinearTransformationMatrix	*A*: M, *e*: M, *f*: M	M	Ger avbildningsmatrisen *A*, given i basen med basmatrisen *e*, uttryckt i basmatrisen *f*.
length	TEXT	$\mathbb{R}$	Ger längden (antalet tecken) i argumentsträngen.
trim	TEXT	TEXT	Returnerar textsträngen med alla inledande och avslutande vittecken (blanksteg, tabulatortecken etc.) borttagna.
trim	REF: TEXT	TEXT	Tar bort alla inledande och avslutande vittecken (blanksteg, tabulatortecken etc.) från strängen.

TrimRight	TEXT	TEXT	Returnerar textsträngen med alla avslutande vittecken (blanksteg, tabulatortecken etc.) borttagna.
TrimRight	REF: TEXT	TEXT	Tar bort alla avslutande vittecken (blanksteg, tabulatortecken etc.) från strängen.
TrimLeft	TEXT	TEXT	Returnerar textsträngen med alla inledande vittecken (blanksteg, tabulatortecken etc.) borttagna.
TrimLeft	REF: TEXT	TEXT	Tar bort alla inledande vittecken (blanksteg, tabulatortecken etc.) från strängen.
ReverseOrder	TEXT	TEXT	Ger textsträngen baklänges, så att t.ex. första tecknet hamnar sist och tvärtom.
ReverseOrder	REF: TEXT	TEXT	Inverterar textsträngen, så att t.ex. första tecknet hamnar sist och tvärtom.
DelWhite	TEXT	TEXT	Ger strängen med samtliga vittecken (blanksteg, tabulatortecken etc.) borttagna.
DelWhite	REF: TEXT	TEXT	Tar bort samtliga vittecken (blanksteg, tabulatortecken etc.) från textsträngen.
UpperCase	TEXT	TEXT	Ger textsträngen med samtliga bokstäver som versaler.
UpperCase	REF: TEXT	TEXT	Gör om samtliga gemener i textsträngen till motsvarande versaler.
LowerCase	TEXT	TEXT	Ger textsträngen med samtliga bokstäver som gemener.
LowerCase	REF: TEXT	TEXT	Gör om samtliga versaler i textsträngen till motsvarande gemener.
ChrEncode	TEXT (CHR)	$\mathbb{R}$	Ger den decimala teckenkoden för tecknet.
ChrEncode	TEXT	$\mathbb{R}^n$	Ger vektorn med samtliga decimala teckenkoder för tecknen i strängen, i rätt ordning.
ChrDecode	$\mathbb{R}$	TEXT (CHR)	Ger tecknet som svarar mot den decimala teckenkoden.
ChrDecode	$\mathbb{R}^n$	TEXT	Ger textsträngen som svarar mot vektorn av teckenkoderna.
ChrDescription	TEXT (CHR)	TEXT	Ger beskrivningen av tecknet.
ChrDescription	TEXT	M: TEXT	Ger kolonn matrisen med beskrivningar av samtliga tecken i strängen, i rätt ordning.
substring	TEXT, $a \in \mathbb{R}$, $b \in \mathbb{R}$	TEXT	Ger delsträngen som börjar på teckenpositionen a och är b tecken lång, av argumentet.
substring	REF: TEXT, $a \in \mathbb{R}$, $b \in \mathbb{R}$	TEXT	Byter ut strängen mot delsträngen som börjar på teckenpositionen a och är b tecken lång.
StrLeft	TEXT, $a \in \mathbb{R}$	TEXT	Ger de a första tecknen i strängen.
StrLeft	REF: TEXT, $a \in \mathbb{R}$	TEXT	Tar bort allt förutom de a första tecknen i strängen.
StrRight	TEXT, $a \in \mathbb{R}$	TEXT	Ger de a sista tecknen i strängen.
StrRight	REF: TEXT, $a \in \mathbb{R}$	TEXT	Tar bort allt förutom de a sista tecknen i strängen.
StrPos	TEXT, *text*: TEXT, $a \in \mathbb{R}$	$\mathbb{R}$	Ger teckenpositionen för den första förekomsten av deltexten *text* till

			höger om teckenpositionen a i argumentet. -1 returneras om texten inte återfinnes.
StrReplaceFirst	TEXT, t_1: TEXT, t_2, $a \in \mathbb{R}$	TEXT	Ger argumentet sedan den första förekomsten efter teckenpositionen a av texten t_1 i argumentet ersatts mot t_2.
StrReplaceFirst	REF: TEXT, t_1: TEXT, t_2, $a \in \mathbb{R}$	TEXT	Byter ut den första förekomsten efter teckenpositionen a av texten t_1 i argumentet mot t_2.
StrReplaceFirst	TEXT, t_1: TEXT, t_2, $a \in \mathbb{R}$	TEXT	Ger argumentet sedan samtliga förekomster efter teckenpositionen a av texten t_1 i argumentet ersatts mot t_2.
StrReplaceFirst	REF: TEXT, t_1: TEXT, t_2, $a \in \mathbb{R}$	TEXT	Byter ut samtliga förekomster efter teckenpositionen a av texten t_1 i argumentet mot t_2.
StrDel	TEXT, $a \in \mathbb{R}$, $b \in \mathbb{R}$	TEXT	Ger argumentsträngen sedan texten från position a och b tecken framåt raderats.
StrDel	REF: TEXT, $a \in \mathbb{R}$, $b \in \mathbb{R}$	TEXT	Tar bort texten från position a och b tecken framåt.
StrDelLeft	TEXT, $a \in \mathbb{R}$	TEXT	Ger argumentsträngen sedan de a sista tecknen raderats.
StrDelLeft	REF: TEXT, $a \in \mathbb{R}$	TEXT	Tar bort de a sista tecknen ur strängen.
StrDelRight	TEXT, $a \in \mathbb{R}$	TEXT	Ger argumentsträngen sedan de a första tecknen raderats.
StrDelRight	REF: TEXT, $a \in \mathbb{R}$	TEXT	Tar bort de a första tecknen ur strängen.
RandomStr	$n \in \mathbb{R}$[, S: SET]	TEXT	Ger en n tecken lång sträng med slumpvis utvalda tecken ur mängden S, default: AS:ALPHABET.
AreAnagrams	TEXT, TEXT	LOGIC	Undersöker huruvida de två texterna är anagram av varandra.
FindAnagrams	TEXT	SET: TEXT	Ger mängden av alla funna anagram till argumentet.
FindAllAnagrams		SET: SET: TEXT	Ger mängden av alla mängder av anagram som funnits ur ordlistan. Tar mycket lång tid att slutföra, men kan avbrytas när som helst.
IsPalindrome	TEXT	LOGIC	Undersöker huruvida argumentet är ett palindrom.
FindAllPalindromes		SET: TEXT	Ger mängden av alla palindrom som funnits ur ordlistan.
IsHeteropalindrome	TEXT	LOGIC	Undersöker huruvida argumentet är ett heteropalindrom, d.v.s. ett ord som blir ett annat ord vid invertering.
FindAllHeteropalindromes		SET: TEXT	Ger mängden av alla heteropalindrom.
ForEachChar	TEXT, FUNCTION	TEXT	Ger strängen som erhålles då funktionen utförs på samtliga tecken i argumentsträngen.
ForEachChar	REF: TEXT, FUNCTION	TEXT	Utför funktionen på samtliga tecken i argumentsträngen.
ForEachWord	TEXT, FUNCTION	TEXT	Ger strängen som erhålles då funktionen utförs på samtliga ord i argumentsträngen.

ForEachWord	REF: TEXT, FUNCTION	TEXT	Utför funktionen på samtliga ord i argumentsträngen.
StrSplitSet	TEXT, *sep*: TEXT	SET: TEXT	Ger mängden av alla delsträngar som uppkommer då argumentet delas upp vid varje förekomst av separatortexten *sep* (tas bort).
StrSplitMat	TEXT, *sep*: TEXT	MATRIX: TEXT	Ger matrisen av alla delsträngar som uppkommer då argumentet delas upp vid varje förekomst av separatortexten *sep* (tas bort).
StrChrSet	TEXT	SET:TEXT (CHR)	Ger mängden av alla tecken som återfinns i argumenttexten.
StrChrMat	TEXT	MAT: TEXT (CHR)	Ger matrisen av alla tecken som återfinns i argumenttexten.
StrWordSet	TEXT	SET:TEXT	Ger mängden av alla ord som återfinns i argumenttexten.
StrWordMat	TEXT	MAT: TEXT	Ger matrisen av alla ord som återfinns i argumenttexten.
FindWord	TEXT	SET: TEXT	Finner ett ord med bokstäver på bestämda platser. T.ex. ger FindWord(”l_pf_t”) svaret {”lampfot”}.
DefineWord	TEXT[, *lang*: TEXT, *class*: TEXT]	SET: TEXT	Ger mängden av alla ordboksdefinitioner av ordet (i språket *lang* och av ordklassen *class*).
WordIsKnown	TEXT[, *lang*: TEXT, *class*: TEXT]	LOGIC	Anger huruvida det (i ordlistan) finns ett ord (i språket *lang* och av ordklassen *class*) lika med första strängen.
NumOfElements	SET	$\mathbb{R}$	Ger antalet element i mängden.
DeleteElement	REF: SET, OBJECT	SET	Tar bort objektet ur mängden. (Icke-referensfunktion behövs ej tack vare operatorn \.)
ForEach	SET, FUNCTION	SET	Ger mängden som erhålles då funktionen utförs på varje element.
ForEach	REF: SET, FUNCTION	SET	Utför funktionen på varje element i mängden.
random	SET	OBJECT	Väljer slumpmässigt ut ett element ur mängden. Bra för att välja restaurang.
CreateSet	TEXT (LOGIC med koordinater)[, 4 eller 6×$\mathbb{R}$, *koordsys*: TEXT]	SET: $\mathbb{R}^n$	Ger mängden av alla punkter som uppfyller det logiska villkoret i strängen i koordinatsystemet *koordsys*, default: ”xy” eller ”xyz”, inom gränserna angivna som de fyra eller sex talen.
CreateNet	TEXT (LOGIC med koordinater)[, 4 eller 6×$\mathbb{R}$, 4 eller 6×*dist*∈$\mathbb{R}$ *koordsys*: TEXT]	SET: $\mathbb{R}^n$	Ger ett nät ur mängden av alla punkter som uppfyller det logiska villkoret i strängen i koordinatsystemet *koordsys*, default: ”xy” eller ”xyz”, inom gränserna angivna som de fyra eller sex talen. Nätavstånd *dist* för respektive koordinat.
CreateSphere	$n \in \mathbb{R}$, $r \in \mathbb{R}$, $p \in \mathbb{R}^n$	SET: $\mathbb{R}^n$	Ger en sfär av dimensionen n med radie r och medelpunkt i p.
CreateBall	$n \in \mathbb{R}$, $r \in \mathbb{R}$, $p \in \mathbb{R}^n$[, LOGIC]	SET: $\mathbb{R}^n$	Ger en (öppen eller, default, sluten) boll av dimensionen n med radie r och medelpunkt i p.
CreateLine	$n \in \mathbb{R}$, $v \in \mathbb{R}^n$[, $p \in \mathbb{R}^n$], l	SET: $\mathbb{R}^n$	Ger ett linjestycke i rummet av dimension n med riktningsvektorn v

			genom punkten p (default: origo) och med längden l.
CreatePlane	$n \in \mathbb{R}$, $v_1 \in \mathbb{R}^n$, $v_2 \in \mathbb{R}^n$[, $p \in \mathbb{R}^n$], $l \in \mathbb{R}^n$	SET: $\mathbb{R}^n$	Ger i ett n-dimensionellt rum planet som spänns upp av vektorerna v_1 och v_2 och innehåller p (default: origo), med utsträckningar i vektorn l i varje koordinat.
CreatePolotope	*type*: TEXT[, $r \in \mathbb{R}$, $p \in \mathbb{R}^n$]	SET: $\mathbb{R}^n$	Ger polytopen med Schläli-symbolen *type* med radie r och medelpunkt i p (default: origo).
CreateVectPolytope	*type*: TEXT[, $r \in \mathbb{R}$, $p \in \mathbb{R}^n$]	MATRIX: VectGraph	Ger en AlgoSim-matrisrepresentation av en vektorgrafisk polytop.
CreateShadow	SET: $\mathbb{R}^n$, *direction* $\in \mathbb{R}^n$, *plane*: TEXT (LOGIC med koord.)	SET: $\mathbb{R}^n$	Ger mängden med skuggan av argumentmängden i riktningen *direction* mot planet *plane*.
CreateShadowUsingSpotlight	SET: $\mathbb{R}^n$, *position* $\in \mathbb{R}^n$, *plane*: TEXT (LOGIC med koord.)	SET: $\mathbb{R}^n$	Ger mängden med skuggan av argumentmängden från den punktformiga ljuskällan i punkten *position* mot planet *plane*.
CreateReflection	SET: $\mathbb{R}^n$, $p \in \mathbb{R}^n$, *plane*: TEXT (LOGIC med koord.)	SET: $\mathbb{R}^n$	Ger reflektionen av mängden mot planet *plane* till observatören i punkten p.
DomainImage	FUNCTION, Domain: SET: $\mathbb{R}^n$	SET: $\mathbb{R}^n$	Ger bilden av definitionsmängden.
DomainImageEx	FUNCTION, Domain: SET: $\mathbb{R}^n$	SET: $\mathbb{R}^n$	Ger bilden av definitionsmängden med extra första koordinater för de oberoende variablerna (parametrarna).
subset	SET, TEXT (LOGIC med ”ELEMENT”)	SET	Ger den delmängd av argumentmängden, vilken delmängd har element som uppfyller villkoret. Fungerar bättre än ∩-operatorn för mängder utan generatorer.
ShortestWay	SET: $\mathbb{R}^n$	MATRIX: $\mathbb{R}^n$	Ger matrisen med samtliga punkter i mängden, ordnad så att en resa genom punkterna i matrisordningen är den kortast möjliga.
NearestNeighbour	SET: $\mathbb{R}^n$, $\mathbb{R}^n$	$\mathbb{R}^n$	Ger det element i mängden som är närmast vektorn.
CenterOfMass	SET: $\mathbb{R}^n$	$\mathbb{R}^n$	Ger masscentrum, d.v.s. tyngdpunkten i ett homogent gravitationsfält av de identiska partiklarna i mängdens punkter.
GeneratorString	REF: SET: $\mathbb{R}^n$	TEXT	Visar mängdens generatorsträng och frågar användaren om hon vill ändra denna.
GetGeneratorString	REF: SET: $\mathbb{R}^n$	TEXT	Visar mängdens generatorsträng.
SetGeneratorString	REF: SET: $\mathbb{R}^n$, TEXT	TEXT	Ändrar mängdens generatorsträng.
ClearGeneratorString	REF: SET: $\mathbb{R}^n$[, LOGIC]	PROGRESS	Tar bort mängdens generatorsträng (utan att fråga).
GCD	SET: $\mathbb{R}$	$\mathbb{R}$	Ger den största gemensamma delaren till talen i mängden.
LCM	SET: $\mathbb{R}$	$\mathbb{R}$	Ger den minsta gemensamma nämnaren till talen i mängden.
sum	SET: $\mathbb{R}$	$\mathbb{R}$	Ger summan av talen i mängden.
product	SET: $\mathbb{R}$	$\mathbb{R}$	Ger produkten av talen i mängden.
max	SET: $\mathbb{R}$	$\mathbb{R}$	Ger det största värdet i mängden.
min	SET: $\mathbb{R}$	$\mathbb{R}$	Ger det minsta värdet i mängden.

mean	SET: $\mathbb{R}$	$\mathbb{R}$	Ger det aritmetiska medelvärdet av talen i mängden.
med	SET: $\mathbb{R}$	$\mathbb{R}$	Ger medianvärdet av talen i mängden.
q1	SET: $\mathbb{R}$	$\mathbb{R}$	Ger den första kvartilen.
q3	SET: $\mathbb{R}$	$\mathbb{R}$	Ger den tredje kvartilen.
LeastSquare	SET: $\mathbb{R}^2$	$\mathbb{R}^2$	Ger parametrarna k och m för den räta linje $y = kx + m$ som bäst ansluter till mängden av punkter (x, y).
∃	SET, TEXT (LOGIC med ”ELEMENT”)	LOGIC	Undersöker huruvida det existerar ett element i mängden sådant att det uppfyller villkoret.
∀	SET, TEXT (LOGIC med ”ELEMENT”)	LOGIC	Undersöker huruvida samtliga element i mängden uppfyller villkoret.
PlaneToSpace	SET: $\mathbb{R}^2$, $z \in \mathbb{R}$, *ext*: LOGIC, $\Delta z \in \mathbb{R}$	SET: $\mathbb{R}^3$	Tar en plan mängd och ger en mängd i rummet, där den plana mängden återfinnes på den tredje koordinaten lika med z (default: 0). Om *ext* så sker ”prismautdragning” Δz enheter i z-led.
SpaceToPlane	SET: $\mathbb{R}^3$, TEXT (LOGIC med koord.), *tolerance* $\in \mathbb{R}$	SET: $\mathbb{R}^2$	Tar en mängd i rummet och snittar den med planet (med den givna ekvationen), med toleransen *tolerance*, och returnerar den erhållna plana mängden.
CorrectPlane	SET: $\mathbb{R}^2$ *blandat med* $\mathbb{R}^3$, *tolerance* $\in \mathbb{R}$	SET: $\mathbb{R}^2$	Ger den plana mängden som erhålles då rymdpunkter tas bort och ersättes med motsvarande plana punkter (den sista koordinaten borttages) om denna är tillräckligt (max avst. *tolerance*) nära 0.
CorrectPlane	REF: SET: $\mathbb{R}^2$ *blandat med* $\mathbb{R}^3$, *tolerance* $\in \mathbb{R}$	SET: $\mathbb{R}^2$	Tar bort rymdpunkter och ersätter dessa med motsvarande plana punkter (den sista koordinaten borttages) om denna är tillräckligt (max avst. *tolerance*) nära 0.
CorrectSpace	SET: $\mathbb{R}^2$ *blandat med* $\mathbb{R}^3$, $z \in \mathbb{R}$	SET: $\mathbb{R}^3$	Ger mängden i rummet med samtliga punkter från argumentmängden, där punkter utan z-koordinat ges koordinaten z.
CorrectSpace	REF: SET: $\mathbb{R}^2$ *blandat med* $\mathbb{R}^3$, $z \in \mathbb{R}$	SET: $\mathbb{R}^3$	Lägger till z-koordinaten z för punkter som saknar sådan koordinat.
format	MATRIX	$\mathbb{R}^2$	Ger formatet (typen) av matrisen.
transpose	MATRIX	MATRIX	Ger transponatet av matrisen.
transpose	REF: MATRIX	MATRIX	Transponerar matrisen.
inverse	MATRIX	MATRIX	Ger inversen till matrisen.
inverse	REF: MATRIX	MATRIX	Inverterar matrisen.
cols	MATRIX	$\mathbb{R}$	Ger antalet kolonner.
rows	MATRIX	$\mathbb{R}$	Ger antalet rader.
det	MATRIX	$\mathbb{R}$	Ger matrisens determinant.
ZeroMatrix	$m \in \mathbb{R}$[, $n \in \mathbb{R}$]	MATRIX	Ger m×n-nollmatrisen (n def m).
IdentityMatrix	$n \in \mathbb{R}$	MATRIX	Ger enhetsmatrisen med n rader.
CreateTruthTable	TEXT (LOGIC)	MATRIX	Ger en sanningstabell över det sammansatta logiska uttrycket.
CreateMatrix			Skapar en matris med matriseditorn.
CreateMatrix	$m \in \mathbb{R}$, $n \in \mathbb{R}$, a_{11}, a_{12}, …, a_{1n}, …, a_{m1}, a_{m2}, …, a_{mn}:	MATRIX	Skapar en m×n-matris med de valda objekten.

	OBJECT(S)		
CreateMatrix	$m \in \mathbb{R}$, $n \in \mathbb{R}$, OBJECT	MATRIX	Skapar en $m \times n$-matris fylld med OBJECT.
CreateMatrix	$m \in \mathbb{R}$, $n \in \mathbb{R}$, FUNCTION (av i, j)	MATRIX	Skapar en $m \times n$-matris med element som ges ur funktionen $(i, j) \mapsto a_{i,j}$.
EditMatrix	REF: MATRIX	MATRIX	Redigerar matrisen i editorn.
FillMatrix	REF: MATRIX, FUNCTION (av m, n)	MATRIX	Fyller matrisen med element givna av funktionen $(m, n) \mapsto a_{m,n}$.
FillDiagonal	REF: MATRIX, FUNCTION (av n)	MATRIX	Fyller matrisens huvuddiagonal med element givna ur funktionen $n \mapsto a_{n,n}$.
FillMatRow	REF: MATRIX, $m \in \mathbb{R}$, FUNCTION (av n)	MATRIX	Fyller matrisens rad m med element givna av funktionen $n \mapsto a_{m,n}$.
FillMatCol	REF: MATRIX, $n \in \mathbb{R}$, FUNCTION (av m)	MATRIX	Fyller matrisens kolonn n med element givna av funktionen $m \mapsto a_{m,n}$.
ForEach	MATRIX, FUNCTION	MATRIX	Ger resultatmatrisen sedan funktionen utförts på varje element i argumentmatrisen.
ForEach	REF: MATRIX, FUNCTION	MATRIX	Utför funktionen på varje element i matrisen.
SortCol	MATRIX, $n \in \mathbb{R}$, LOGIC	MATRIX	Ger den matris som erhålls då argumentmatrisens kolonn n sorterats (i omvänd ordning).
SortCol	REF: MATRIX, $n \in \mathbb{R}$, LOGIC	MATRIX	Sorterar kolonn n (i omvänd ordning).
SortRow	MATRIX, $m \in \mathbb{R}$, LOGIC	MATRIX	Ger den matris som erhålls då argumentmatrisens rad m sorterats (i omvänd ordning).
SortRow	REF: MATRIX, $m \in \mathbb{R}$, LOGIC	MATRIX	Sorterar rad m (i omvänd ordning).
IndexOf	MATRIX, OBJECT	$\mathbb{R}^2$	Ger positionen för den första förekomsten av objektet i matrisen.
MatrixReplace	MATRIX, OBJECT, OBJECT	MATRIX	Ger matrisen som erhålls då argumentmatrisens samtliga förekomster av det första objektet ersätts med det andra objektet.
MatrixReplace	REF: MATRIX, OBJECT, OBJECT	MATRIX	Byter ut samtliga förekomster av det första objektet mot det andra objektet i matrisen.
EditMatrixAsBitmap	REF: MATRIX	MATRIX	Redigerar matrisen (som en bitmap) i en pixelgrafisk editor.
BlendMatrix	MATRIX, MATRIX[, *pos* $\in \mathbb{R}^2$]	MATRIX	Ger den bild som erhålles då de två bilderna överlagras enligt regeln som anges av AS:BLENDMODE (pos def origo).
RotateMatrix	MATRIX[, $\mathbb{R}$]	MATRIX	Ger bilden roterad den angivna vinkeln (def $\pi/2$ rad).
RotateMatrix	REF: MATRIX[, $\mathbb{R}$]	MATRIX	Roterar bilden den angivna vinkeln (def $\pi/2$ rad).
ScaleMatrix	MATRIX, $x \in \mathbb{R}$ [, $y \in \mathbb{R}$]	MATRIX	Ger matrisen skalad en faktor x i x-led och en faktor y i y-led (y def x).
ScaleMatrix	REF: MATRIX, $x \in \mathbb{R}$ [, $y \in \mathbb{R}$]	MATRIX	Skalar matrisen en faktor x i x-led och en faktor y i y-led (y def x).
GetSubmatrix	MATRIX, $p \in \mathbb{R}^2$, $q \in \mathbb{R}^2$	MATRIX	Ger den del av bilden som börjar i p och slutar i q.
MatrixAutoSize	MATRIX, *color*	MATRIX	Ger bilden utan enfärgade kanter runt. Kantområden med den givna färgen tas bort.
MatrixAutoSize	REF: MATRIX, *color*	MATRIX	Tar bort enfärgade kanter runt bil-

			den. Färgen *color* tas bort.
pixellate	MATRIX, $x \in \mathbb{R}$[, $y \in \mathbb{R}$]	MATRIX	Ger bilden pixellerad med pixelbredden x och -höjden y (y def x).
pixellate	REF: MATRIX, $x \in \mathbb{R}$[, $y \in \mathbb{R}$]	MATRIX	Pixellerar bilden med pixelbredden x och -höjden y (y def x).
InvertColors	MATRIX	MA TRIX	Ger bilden med färgerna inverterade.
InvertColors	REF: MATRIX	MATRIX	Inverterar färgerna.
InvertAlpha	MATRIX	MA TRIX	Ger bilden med alphavärdena inverterade.
InvertAlpha	REF: MATRIX	MATRIX	Inverterar alphavärdena.
EdgeDetect	MATRIX	MATRIX	Ger bilden med kanterna markerade.
EdgeDetect	REF: MATRIX	MATRIX	Markerar kanterna i bilden.
blur	MATRIX[, $\mathbb{R}$]	MATRIX	Ger bilden med minskad skärpa.
blur	REF: MATRIX[, $\mathbb{R}$]	MATRIX	Minskar skärpan i bilden.
MotionBlur	MATRIX[, $\mathbb{R}$[,*angle* $\in \mathbb{R}$]]	MATRIX	Ger bilden med rörelseoskärpa i den valda vinkelriktningen *angle*.
MotionBlur	REF: MATRIX[, $\mathbb{R}$[,*angle* $\in \mathbb{R}$]]	MATRIX	Ger bilden rörelseoskärpa i den valda vinkelriktningen *angle*.
GetRLayer	MATRIX[, $\mathbb{R}$, $\mathbb{R}$]	MATRIX	Ger bildens röd-kanal (övr. kanaler).
GetGLayer	MATRIX[, $\mathbb{R}$, $\mathbb{R}$]	MATRIX	Ger bildens grön-kanal.
GetBLayer	MATRIX[, $\mathbb{R}$, $\mathbb{R}$]	MATRIX	Ger bildens blå-kanal.
MergeRGBLayers	3×MATRIX	MATRIX	Sammanfogar R-, G- och B-kanaler.
SetRLayer	MATRIX, $\mathbb{R}$	MATRIX	Ger bilden sedan alla punkter tvingats till det valda R-värdet.
SetRLayer	REF: MATRIX, $\mathbb{R}$	MATRIX	Tvingar alla punkter till det valda R-värdet.
SetGLayer	MATRIX, $\mathbb{R}$	MATRIX	Ger bilden sedan alla punkter tvingats till det valda G-värdet.
SetGLayer	REF: MATRIX, $\mathbb{R}$	MATRIX	Tvingar alla punkter till det valda G-värdet.
SetBLayer	MATRIX, $\mathbb{R}$	MATRIX	Ger bilden sedan alla punkter tvingats till det valda B-värdet.
SetBLayer	REF: MATRIX, $\mathbb{R}$	MATRIX	Tvingar alla punkter till det valda B-värdet.
GetHLayer	MATRIX[, $\mathbb{R}$, $\mathbb{R}$]	MATRIX	Ger bildens nyans-kanal (övr. kanaler).
GetSLayer	MATRIX[, $\mathbb{R}$, $\mathbb{R}$]	MATRIX	Ger bildens mättnads-kanal.
GetVLayer	MATRIX[, $\mathbb{R}$, $\mathbb{R}$]	MATRIX	Ger bildens ljusstyrke-kanal.
MergeHSVLayers	3×MATRIX	MATRIX	Sammanfogar H-, S- och V-kanaler.
SetHLayer	MATRIX, $\mathbb{R}$	MATRIX	Ger bilden sedan alla punkter tvingats till det valda H-värdet.
SetHLayer	REF: MATRIX, $\mathbb{R}$	MATRIX	Tvingar alla punkter till det valda H-värdet.
SetSLayer	MATRIX, $\mathbb{R}$	MATRIX	Ger bilden sedan alla punkter tvingats till det valda S-värdet.
SetSLayer	REF: MATRIX, $\mathbb{R}$	MATRIX	Tvingar alla punkter till det valda S-värdet.
SetVLayer	MATRIX, $\mathbb{R}$	MATRIX	Ger bilden sedan alla punkter tvingats till det valda V-värdet.
SetVLayer	REF: MATRIX, $\mathbb{R}$	MATRIX	Tvingar alla punkter till det valda V-värdet.
colorize	MATRIX, *color*	MATRIX	Ger bilden färglagd endast med den valda nyansen.
colorize	REF: MATRIX, *color*	MATRIX	Färglägger bilden med den valda nyansen.
brightness	MATRIX, $\mathbb{R}$	MATRIX	Ger bilden med ändrad ljusstyrka.
brightness	REF: MATRIX, $\mathbb{R}$	MATRIX	Ändrar bildens ljusstyrka.

contrast	MATRIX, $\mathbb{R}$	MATRIX	Ger bilden med ändrad kontrast.
contrast	REF: MATRIX, $\mathbb{R}$	MATRIX	Ändrar bildens kontrast.
BitmapMatrixToColor-Set	MATRIX	SET: $\mathbb{R}^3$	Ger en mängd för uppritning med **DrawPointSetColor**.
PaintMatrixRows	REF: MATRIX, *color1*, *color2*[, *color3*]	MATRIX	Ändrar matrisens formatering, så att alla udda rader får färgen *color1* och alla jämna rader får färgen *color2*. Första raden får färgen *color3* def *color1*.
PaintMatrixCols	REF: MATRIX, *color1*, *color2*[, *color3*]	MATRIX	Ändrar matrisens formatering, så att alla udda kolonner får färgen *color1* och alla jämna kolonner får färgen *color2*. Första kolonnen får färgen *color3* def *color1*.
PaintMatrixCell	REF: MATRIX, $\mathbb{R}^2$, *color*	MATRIX	Den valda cellen i matrisen får färgen *color*.
PaintMatrixRow	REF: MATRIX, $\mathbb{R}$, *color*	MATRIX	Färgar den valda raden.
PaintMatrixCol	REF: MATRIX, $\mathbb{R}$, *color*	MATRIX	Färgar den valda kolonnen.
PaintMatrixRule	REF: MATRIX, FUNC-TION: LOGIC (av *m, n, object*), *color*	MATRIX	Färgar alla celler som uppfyller villkoret med en viss färg. Använd t.ex. för att markera alla primtal i en tabell med tal!
PaintMatrixRuleEx	REF: MATRIX, FUNC-TION: *color* (av *m, n, object*)	MATRIX	Färgar varje cell enligt funktionsvärdet för cellen.
sum	MATRIX	$\mathbb{R}$	Ger summan av samtliga tal i matrisen.
product	MATRIX	$\mathbb{R}$	Ger produkten av samtliga tal i matrisen.
RowMultiply	MATRIX, $m \in \mathbb{R}$, $c \in \mathbb{R}$	MATRIX	Ger matrisen sedan raden m multiplicerats med faktorn c.
RowMultiply	REF: MATRIX, $m \in \mathbb{R}$, $c \in \mathbb{R}$	MATRIX	Multiplicerar raden m med faktorn c.
ColMultiply	MATRIX, $n \in \mathbb{R}$, $c \in \mathbb{R}$	MATRIX	Ger matrisen sedan kolonnen n multiplicerats med faktorn c.
ColMultiply	REF: MATRIX, $n \in \mathbb{R}$, $c \in \mathbb{R}$	MATRIX	Multiplicerar kolonnen n med faktorn c.
RowSwap	MATRIX, $\mathbb{R}$, $\mathbb{R}$	MATRIX	Ger matrisen sedan de två raderna bytit plats med varandra.
RowSwap	REF: MATRIX, $\mathbb{R}$, $\mathbb{R}$	MATRIX	Byter plats på de två raderna.
ColSwap	MATRIX, $\mathbb{R}$, $\mathbb{R}$	MATRIX	Ger matrisen sedan de två kolonnerna bytit plats med varandra.
ColSwap	REF: MATRIX, $\mathbb{R}$, $\mathbb{R}$	MATRIX	Byter plats på de två kolonnerna.
RowOp	MATRIX, $a \in \mathbb{R}$, $b \in \mathbb{R}$, $c \in \mathbb{R}$	MATRIX	Adderar rad a med produkten av rad b och faktorn c och sparar resultatet i den resulterande matrisen.
RowOp	REF: MATRIX, $a \in \mathbb{R}$, $b \in \mathbb{R}$, $c \in \mathbb{R}$	MATRIX	Adderar rad a med produkten av rad b och faktorn c.
ColOp	MATRIX, $a \in \mathbb{R}$, $b \in \mathbb{R}$, $c \in \mathbb{R}$	MATRIX	Adderar kolonn a med produkten av kolonn b och faktorn c och sparar resultatet i den resulterande matrisen.
ColOp	REF: MATRIX, $a \in \mathbb{R}$, $b \in \mathbb{R}$, $c \in \mathbb{R}$	MATRIX	Adderar kolonn a med produkten av kolonn b och faktorn c.
LeastSquare	MATRIX ($2\times n$ eller $n\times 2$)	$\mathbb{R}^2$	Ger parametrarna k och m för den räta linje $y = kx + m$ som bäst ansluter till mängden av punkter (x, y) i matrisen.
RowDifference	MATRIX	MATRIX	Ger den matris som innehåller diffe-

			renserna mellan varje horisontellt par av tal i matrisen.
ColDifference	MATRIX	MATRIX	Ger den matris som innehåller differenserna mellan varje vertikalt par av tal i matrisen.
GCD	MATRIX: ℝ	ℝ	Ger den största gemensamma delaren till talen i matrisen.
LCM	MATRIX: ℝ	ℝ	Ger den minsta gemensamma nämnaren till talen i matrisen.
max	MATRIX: ℝ	ℝ	Ger det största värdet i matrisen.
min	MATRIX: ℝ	ℝ	Ger det minsta värdet i matrisen.
mean	MATRIX: ℝ	ℝ	Ger det aritmetiska medelvärdet av talen i matrisen.
med	MATRIX: ℝ	ℝ	Ger medianvärdet av talen i matrisen.
q1	MATRIX: ℝ	ℝ	Ger den första kvartilen.
q3	MATRIX: ℝ	ℝ	Ger den tredje kvartilen.
objection			Insistera att föregående beräkning har utförts felaktigt av AlgoSim. Först kontrollerar programmet om användaren måhända inmatat fel kommando, sedan kan cachen kontrolleras, kommandot utföras igen, och så småningom kan en mer eller mindre automatisk felrapport skapas.
BaseNToDec	TEXT, ℝ	ℝ	Ger den decimala representationen av talet i strängen angivet i den givna basen.
DecToBaseN	ℝ, ℝ	TEXT	Ger strängen med det i det decimala talsystemet angivna talet, i den angivna basen.
BaseNToM	TEXT, ℝ, ℝ	TEXT	Ger det i den första basen i en sträng givna talet som en sträng givet i det andra systemet.
DecToBin	ℝ	TEXT	Bas 10 till bas 2.
BinToDec	TEXT	ℝ	Bas 2 till bas 10.
DecToOct	ℝ	TEXT	Bas 10 till bas 8.
OctToDec	TEXT	ℝ	Bas 8 till bas 10.
DecToHex	ℝ	TEXT	Bas 10 till bas 16.
HexToDec	TEXT	ℝ	Bas 16 till bas 10.
DecToRoman	ℝ	TEXT	Bas 10 till romerska siffror.
RomanToDec	TEXT	ℝ	Romerska siffror till bas 10.
DecToUnary	ℝ	TEXT	Bas 10 till bas ”unärt talsystem”.
UnaryToDec	TEXT	ℝ	”Unärt talsystem” till bas 10.
translate	$\mathbb{R}^n$, $\mathbb{R}^n$	$\mathbb{R}^n$	Utför en translation av vektorn.
Rotate2D	$\mathbb{R}^2$[, $\mathbb{R}^2$], ℝ	$\mathbb{R}^2$	Utför en tvådimensionell rotation av vektorn. (Använd gärna med ForEach i punktmängd.)
Rotate3D	$\mathbb{R}^3$[, $\mathbb{R}^3$], $\mathbb{R}^3$. ℝ	$\mathbb{R}^3$	Utför en tredimensionell rotation av vektorn. (Använd gärna med ForEach i punktmängd.)
Mirror2D	$\mathbb{R}^2$[, $\mathbb{R}^2$], $\mathbb{R}^2$	$\mathbb{R}^2$	Utför en tvådimensionell spegling av vektorn. (Använd gärna med ForEach i punktmängd.)
Mirror3D	$\mathbb{R}^3$[, $\mathbb{R}^3$], $\mathbb{R}^3$, $\mathbb{R}^3$	$\mathbb{R}^3$	Utför en tredimensionell spegling av vektorn. (Använd gärna med ForEach i punktmängd.)
RandomReal	$[a \in \mathbb{R}, b \in \mathbb{R}]$	ℝ	Ger ett slumpmässigt valt reellt tal i

			intervallet $[a, b]$ (def [0,1]).
RandomInt	$[a \in \mathbb{R}, b \in \mathbb{R}]$	$\mathbb{R}$	Ger ett slumpmässigt valt heltal i intervallet $[a, b]$ (def $[0, 10]$).
RandomSign		$\mathbb{R}$	Ger slumpmässigt ett tal ur mängden $\{-1,1\}$.
UsePrefix	$\mathbb{R}$, $\mathbb{R}^n$, SET: $\mathbb{R}^n$ *eller* MATRIX: $\mathbb{R}^n$	$\mathbb{R}$, $\mathbb{R}^n$, SET: $\mathbb{R}^n$ *eller* MATRIX: $\mathbb{R}^n$	Visar samtliga tal med det mest lämpliga prefixet.
UsePrefixLow	$\mathbb{R}$, $\mathbb{R}^n$, SET: $\mathbb{R}^n$ *eller* MATRIX: $\mathbb{R}^n$	$\mathbb{R}$, $\mathbb{R}^n$, SET: $\mathbb{R}^n$ *eller* MATRIX: $\mathbb{R}^n$	Visar samtliga tal med det mest lämpliga prefixet, så att talet blir stort.
UsePrefixHigh	$\mathbb{R}$, $\mathbb{R}^n$, SET: $\mathbb{R}^n$ *eller* MATRIX: $\mathbb{R}^n$	$\mathbb{R}$, $\mathbb{R}^n$, SET: $\mathbb{R}^n$ *eller* MATRIX: $\mathbb{R}^n$	Visar samtliga tal med det mest lämpliga prefixet, så att talet blir litet.
UseAllPrefixes	$\mathbb{R}$	MATRIX: TEXT	Ger en matris med argumentet skrivet med samtliga prefix, i ordning.
CreateLatinSquare	$n \in \mathbb{R}$; SET: OBJECT	MATRIX	Skapar en latinsk kvadrat av storleken n med elementen ur mängden.
IsLatinSquare	MATRIX	LOGIC	Undersöker huruvida matrisen när en latinsk kvadrat.
CreateEulerSquare	$n \in \mathbb{R}$; 2×SET: OBJECT	MATRIX	Skapar en Euler-kvadrat av storleken n med elementen ur mängderna.
IsEulerSquare	MATRIX	LOGIC	Undersöker huruvida matrisen när en Euler-kvadrat.
CreateMagicSquare	$n \in \mathbb{R}$; *sum* $\in \mathbb{R}$, *normal*: LOGIC	MATRIX	Skapar en magisk kvadrat av storleken n och summan *sum*.
IsMagicSquare	MATRIX	LOGIC	Undersöker huruvida matrisen när en magisk kvadrat.
RadToDeg	$\mathbb{R}$	$\mathbb{R}$	Gör om radianer till grader.
DegToRad	$\mathbb{R}$	$\mathbb{R}$	Gör om grader till radianer.
CoordTransform	$\mathbb{R}^n$, *from*: TEXT, *to*: TEXT	$\mathbb{R}^n$	Gör om koordinaten från ett koordinatsystem till ett annat, t.ex. från sfäriska till kartesiska koordinater.
delete	REF: OBJECT	PROGRESS	Tar bort variabeln.
rename	REF: OBJECT, TEXT	PROGRESS	Byter namn på variabeln.
declare	REF: OBJECT	TEXT: OBJECT	Ger en textsträng som kan användas för att deklarera variabeln.
CreationTime	REF: OBJECT	TEXT	Visar när variabeln skapades.
ModificationTime	REF: OBJECT	TEXT	Visar när variabeln senast ändrades.
type	REF: OBJECT	TEXT	Visar vilken datatyp variabeln har.
describe	REF: OBJECT	TEXT	Ber AlgoSim beskriva variabeln så utförligt som möjligt.
VectToMatRow	$\mathbb{R}^n$	MATRIX	Skapar en radmatris med vektorns koordinater.
VectToMatCol	$\mathbb{R}^n$	MATRIX	Skapar en kolonnmatris med vektorns koordinater.
MatRowToVect	MATRIX, $\mathbb{R}$	$\mathbb{R}^n$	Lägger den valda raden i matrisen i en vektor.
MatColToVect	MATRIX, $\mathbb{R}$	$\mathbb{R}^n$	Lägger den valda kolonnen i matrisen i en vektor.
VectToSet	$\mathbb{R}^n$	SET: $\mathbb{R}$	Lägger vektorns komponenter i en mängd.
SetToVect	SET: $\mathbb{R}$	$\mathbb{R}^n$	Skapar en vektor med element från mängden.
MatRowToSet	MATRIX, $\mathbb{R}$	SET: OBJECT	Skapar en mängd med elementen i den valda raden i matrisen.
MatColToSet	MATRIX, $\mathbb{R}$	SET: OBJECT	Skapar en mängd med elementen i

			den valda kolonnen i matrisen.
SetToMatRow	SET	MATRIX	Skapar en radmatris med element från mängden.
SetToMatCol	SET	MATRIX	Skapar en kolonnmatris med element från mängden.
MatToSet	MATRIX	SET	Lägger alla element i matrisen i en mängd.
TextToReal	TEXT	ℝ	Tolkar strängen som ett reellt tal.
RealToText	ℝ	TEXT	Skapar en sträng med talet i decimal form.
RealToWords	ℝ	TEXT	Skapar en sträng med talet i decimal form, med hjälp av ord istället för siffror. (”Tvåtusenfemhundrasjuttioåtta”, exv.)
WordsToReal	TEXT	ℝ	Tolkar texten som ett reellt tal.
SetToStr	SET[, *konj*: TEXT]	TEXT	Skapar en sträng som räknar upp samtliga element i mängden, avgränsade med komma. Mellan de sista elementen infogas konjunktionen *konj* def ”,”.
type	OBJECT	TEXT	Returnerar typen av objektet (”real number”, ”vector”, ”matrix” osv.).
IsReal	OBJECT	LOGIC	Returnerar TRUE om OBJECT är ett reellt tal; annars returneras FALSE.
IsVector, IsMatrix, IsSet osv.	OBJECT	LOGIC	Som ovan.
ExpectReal	OBJECT	LOGIC	Returnerar TRUE om OBJECT är ett reellt tal; annars visas ett felmeddelande och programmet avslutas.
ExpectVector, ExpectMatrix, ExpectSet osv.	OBJECT	LOGIC	Som ovan.
CurrentTime		ℝ	Ger tidsindex för den aktuella millisekunden.
TimeIndex	TEXT	ℝ	Omvandlar strängen med tids- och datuminformation till ett reellt tidsindex.
TimeString	ℝ	TEXT	Omvandlar det reella tidsindexet till en sträng med tids- och datuminformation.
FormatTime	ℝ, *format*: TEXT	TEXT	Skapar en textsträng med ett speciellt format från det givna tidsindexet.
SecondsBetween	ℝ, ℝ	ℝ	Ger antalet sekunder mellan de två tiderna.
MinutesBetween	ℝ, ℝ	ℝ	Ger antalet minuter mellan de två tiderna.
HoursBetween	ℝ, ℝ	ℝ	Ger antalet timmar mellan de två tiderna.
DaysBetween	ℝ, ℝ	ℝ	Ger antalet dygn mellan de två tiderna.
WeeksBetween	ℝ, ℝ	ℝ	Ger antalet veckor mellan de två tiderna.
MonthsBetween	ℝ, ℝ	ℝ	Ger antalet månader mellan de två tiderna.
YearsBetween	ℝ, ℝ	ℝ	Ger antalet år mellan de två tiderna.
YearOf	ℝ	ℝ	Ger året för den aktuella tiden.
MonthOf	ℝ	ℝ	Ger månaden för den aktuella tiden.
WeekOf	ℝ	ℝ	Ger veckan för den aktuella tiden.
DayOf	ℝ	ℝ	Ger dagen på året för den aktuella

			tiden.
DateOf	ℝ	ℝ	Ger datumet för den aktuella tiden.
WeekdayOf	ℝ	ℝ	Ger veckodagen för den aktuella tiden.
HourOf	ℝ	ℝ	Ger timmen för den aktuella tiden.
MinuteOf	ℝ	ℝ	Ger minuten för den aktuella tiden.
SecondOf	ℝ	ℝ	Ger sekunden för den aktuella tiden.
MillisecondOf	ℝ	ℝ	Ger millisekunden för den aktuella tiden.
DateAndTime	ℝ	TEXT	Ger en sträng med datum och tid för den aktuella tiden.
TimeOfDayName	ℝ	TEXT	Ger den aktuella tiden på dygnet (t.ex. ”kväll”) som en sträng vid den aktuella tiden.
GetTimeZone		ℝ	Ger den aktuella tidzonen.
GetTimeZoneName		ℝ	Ger namnet på den aktuella tidzonen.
NumOfArguments	FUNCTION	ℝ	Ger antalet argument till funktionen.
diff	FUNCTION, ℝ[, ℝ]	ℝ	Beräknar derivatan av funktionen i den valda punkten. Den valfria parametern skriver över AS:DIFFH.
diff	TEXT, TEXT, ℝ[, ℝ]	ℝ	Beräknar derivatan av funktionen med avseende på den givna variabeln i den givna punkten. Den valfria parametern skriver över AS:DIFFH.
CreateDiffGraph	FUNCTION, ℝ, ℝ[, ℝ]	SET: $\mathbb{R}^2$	Skapar en graf av derivatan till funktionen, mellan de två punkterna. Den valfria parametern skriver över AS:DIFFH.
CreateDiffGraph	TEXT, TEXT, ℝ, ℝ[, ℝ]	SET: $\mathbb{R}^2$	Skapar en graf av derivatan till den givna funktionen av den givna variabeln, mellan de två punkterna. Den valfria parametern skriver över AS:DIFFH.
∫, int	F, ℝ, ℝ[, ℝ]	ℝ	Beräknar den bestämda integralen av funktionen mellan de två punkterna. Den sista parametern skriver över AS:INTN.
∫, int	TEXT, TEXT, ℝ, ℝ[, ℝ]	ℝ	Beräknar den bestämda integralen av den givna funktionen av den givna variabeln mellan de två punkterna. Den sista parametern skriver över AS:INTN.
IntRiemann	F, ℝ, ℝ[, ℝ]	ℝ	Beräknar summan av areorna av de rektanglar som lagts under grafen till funktionen mellan de två punkterna. Den frivilliga parametern skriver över AS:INTN, och anger följaktligen antalet rektanglar.
IntRiemann	TEXT, TEXT, ℝ, ℝ[, ℝ]	ℝ	Beräknar summan av areorna av de rektanglar som lagts under grafen till den givna funktionen av den givna variabeln mellan de två punkterna. Den frivilliga parametern skriver över AS:INTN, och anger följaktligen antalet rektanglar.
IntIllustrate	F, ℝ, ℝ[, ℝ]	SET: $\mathbb{R}^2$	Skapar grafen av de riemannrek-

			tanglar som används för att approximera arean under grafen, d.v.s. integralen. *(Se IntRiemann)*
IntIllustrate	TEXT, TEXT, ℝ, ℝ[, ℝ]	SET: $\mathbb{R}^2$	Skapar grafen av de riemannrektanglar som används för att approximera arean under grafen, d.v.s. integralen. *(Se IntRiemann)*
int	SET: $\mathbb{R}^2$	ℝ	Drar linjer mellan punkterna i planet, och beräknar arean av den polygon som uppstår.
int	SET: $\mathbb{R}^3$	ℝ	Lägger plan över punkterna i rummet och beräknar volymen av den kropp som uppstår.
CreateIntGraph	F, ℝ, ℝ[, ℝ]	SET: $\mathbb{R}^2$	Skapar en graf av integralen av funktionen från startvärdet till värdet på abskissan. *(Se int)*
CreateIntGraph	TEXT, TEXT, ℝ, ℝ[, ℝ]	SET: $\mathbb{R}^2$	Skapar en graf av integralen av den givna funktionen av den givna variabeln från startvärdet till värdet på abskissan. *(Se int)*
table	F, ℝ, ℝ[, ℝ]	MATRIX	Skapar en värdetabell av funktionen mellan de två punkterna. Den frivilliga parametern anger avståndet i den oberoende variabeln mellan två följande poster i tabellen (def 1).
table	TEXT, TEXT, ℝ, ℝ[, ℝ]	MATRIX	Skapar en värdetabell av den givna funktionen av den givna variabeln mellan de två punkterna. Den frivilliga parametern anger avståndet i den oberoende variabeln mellan två följande poster i tabellen (def 1).
solve	*eq*: TEXT[, *start* ∈ ℝ [, *itr* ∈ ℝ]]	ℝ	Löser ekvationen *eq* med Newton-Raphsons metod med initialvärde *start* (def 0) och antalet iterationer *itr* (def AS:NITER).
AnalyzeFunction	FUNCTION, ℝ, ℝ	T	Undersöker om funktionen har några stationära punkter mellan de två punkterna, och presenterar rapporten i en sträng.
AnalyzeFunction	TEXT, TEXT, ℝ, ℝ	T	Undersöker om den givna funktionen av den givna variabeln har några stationära punkter mellan de två punkterna, och presenterar rapporten i en sträng.
Contineous	FUNCTION, ℝ	L	Undersöker om funktionen är kontinuerlig i den givna punkten eller ej.
Contineous	TEXT, TEXT, ℝ	L	Undersöker om den givna funktionen av den givna variabeln är kontinuerlig i den givna punkten eller ej.
lim	FUNCTION, ℝ	L	Beräknar gränsvärdet av funktionen i den givna punkten.
liml	FUNCTION, ℝ	L	Beräknar vänstergränsvärdet av funktionen i den givna punkten.
limr	FUNCTION, ℝ	L	Beräknar högergränsvärdet av funktionen i den givna punkten.
lim	TEXT, TEXT, ℝ	L	Beräknar gränsvärdet av den givna funktionen av den givna variabeln i

			den givna punkten.
liml	TEXT, TEXT, ℝ	L	Beräknar vänstergränsvärdet av den givna funktionen av den givna variabeln i den givna punkten.
limr	TEXT, TEXT, ℝ	L	Beräknar högergränsvärdet av den givna funktionen av den givna variabeln i den givna punkten.
taylor	FUNCTION, ℝ[, ℝ]	FUNCTION	Skapar Taylorpolynomet av funktionen runt den givna punkten. Den frivilliga parametern anger ordningen (def 10).
taylor	TEXT, TEXT, ℝ[, ℝ]	FUNCTION	Skapar Taylorpolynomet av den givna funktionen av den givna variabeln runt den givna punkten. Den frivilliga parametern anger ordningen (def 10).
maclaurin	FUNCTION[, ℝ]	FUNCTION	Skapar Maclaurinpolynomet av funktionen. Den frivilliga parametern anger ordningen (def 10).
maclaurin	TEXT, TEXT[, ℝ]	FUNCTION	Skapar Maclaurinpolynomet av den givna funktionen av den givna variabeln. Den frivilliga parametern anger ordningen (def 10).
table	FUNCTION, ℝ, ℝ[, ℝ], ℝ, ℝ[, ℝ]	MATRIX	Skapar en värdetabell av funktionen av två variabler, där variablerna går mellan respektive ändpunkter. De frivilliga argumenten anger steget mellan varje post i tabellen hos respektive oberoende variabel (def 1).
table	TEXT, TEXT, TEXT, ℝ, ℝ[, ℝ], ℝ, ℝ[, ℝ]	MATRIX	Skapar en värdetabell av den givna funktionen av de givna två variablerna, där variablerna går mellan respektive ändpunkter. De frivilliga argumenten anger steget mellan varje post i tabellen hos respektive oberoende variabel (def 1).
FunctionZero, fzero	FUNCTION[, ℝ, ℝ]	SET: ℝ(^*n*)	Ger mängden av alla nollställen till funktionen.
FunctionZero, fzero	TEXT, TEXT[, ℝ, ℝ]	SET: ℝ(^*n*)	Ger mängden av alla nollställen till funktionen av den givna variabeln.
FunctionMax, fmax	FUNCTION[, ℝ, ℝ]	SET: ℝ(^*n*)	Ger mängden av alla maximipunkter till funktionen.
FunctionMax, fmax	TEXT, TEXT[, ℝ, ℝ]	SET: ℝ(^*n*)	Ger mängden av alla maximipunkter till funktionen av den givna variabeln.
FunctionMin, fmin	FUNCTION[, ℝ, ℝ]	SET: ℝ(^*n*)	Ger mängden av alla minimipunkter till funktionen.
FunctionMin, fmin	TEXT, TEXT[, ℝ, ℝ]	SET: ℝ(^*n*)	Ger mängden av alla minimipunkter till funktionen av den givna variabeln.
FunctionStat, fstat	FUNCTION[, ℝ, ℝ]	SET: ℝ(^*n*)	Ger mängden av alla stationära punkter till funktionen.
FunctionStat, fstat	TEXT, TEXT[, ℝ, ℝ]	SET: ℝ(^*n*)	Ger mängden av alla stationära punkter till funktionen av den givna variabeln.
CreateGradientVector-Field	FUNCTION, ℝ, ℝ, ℝ, ℝ[, *dist*: ℝ][, *coord*: TEXT]	SET: $\mathbb{R}^4$	I planet: skapar gradientens vektorfält av funktionen av två variabler mellan de givna gränserna. *dist*

			anger avståndet mellan vektorerna och *coord* anger koordinatsystemet (t.ex. kartesiskt eller sfäriskt). **PlotVectorField**
CreateGradientVectorField	TEXT, TEXT, TEXT, ℝ, ℝ, ℝ, ℝ[, *dist*: ℝ][, *coord*: TEXT]	SET: $\mathbb{R}^4$	I planet: skapar gradientens vektorfält av funktionen av två givna variabler mellan de givna gränserna. *dist* anger avståndet mellan vektorerna och *coord* anger koordinatsystemet (t.ex. kartesiskt eller sfäriskt). **PlotVectorField**
CreateDivergenceGraph	FUNCTION, ℝ, ℝ, ℝ, ℝ[, *coord*: TEXT]	SET: $\mathbb{R}^3$	Skapar en graf i rummet av divergensen av det tvådimensionella vektorfältet i det givna rektangulära området. **DrawPointSet**
CreateDivergenceGraph	TEXT, TEXT, TEXT, ℝ, ℝ, ℝ, ℝ[, *coord*: TEXT]	SET: $\mathbb{R}^3$	Skapar en graf i rummet av divergensen av det tvådimensionella vektorfältet i det givna rektangulära området. **DrawPointSet**
CreateDivergenceColorPlane	FUNCTION, ℝ, ℝ, ℝ, ℝ[, *coord*: TEXT]	SET: $\mathbb{R}^3$	Skapar en färgad plan yta av divergensen av det tvådimensionella vektorfältet i det givna rektangulära området. **DrawPointSetColor**
CreateDivergenceColorPlane	TEXT, TEXT, TEXT, ℝ, ℝ, ℝ, ℝ[, *coord*: TEXT]	SET: $\mathbb{R}^3$	Skapar en färgad plan yta av divergensen av det tvådimensionella vektorfältet i det givna rektangulära området. **DrawPointSetColor**
CreateDivergenceLevelPlane	FUNCTION, ℝ, ℝ, ℝ, ℝ[, *coord*: TEXT]	SET: $\mathbb{R}^2$	Skapar nivåkurvor av divergensen av det tvådimensionella vektorfältet i det givna rektangulära området. **DrawPointSet**
CreateDivergenceLevelPlane	TEXT, TEXT, TEXT, ℝ, ℝ, ℝ, ℝ[, *coord*: TEXT]	SET: $\mathbb{R}^2$	Skapar nivåkurvor av divergensen av det tvådimensionella vektorfältet i det givna rektangulära området. **DrawPointSet**
∬, int	FUNCTION, ℝ, ℝ, ℝ, ℝ[, ℝ]	ℝ	Beräknar integralen av funktionen av två variabler, i det rektangulära definitionsområdet som anges. Den frivilliga parametern skriver över AS:INTN.
∬, int	TEXT, TEXT, TEXT, ℝ, ℝ, ℝ, ℝ[, ℝ]	ℝ	Beräknar integralen av den givna funktionen av de två givna variablerna, i det rektangulära definitionsområdet som anges. Den frivilliga parametern skriver över AS:INTN.
∭, int	FUNCTION, ℝ, ℝ, ℝ, ℝ, ℝ, ℝ[, ℝ]	ℝ	Beräknar integralen av funktionen av tre variabler, i det rätblocksformade definitionsområdet som anges. Den frivilliga parametern skriver över AS:INTN.
∭, int	TEXT, TEXT, TEXT, TEXT, ℝ, ℝ, ℝ, ℝ, ℝ, ℝ[, ℝ]	ℝ	Beräknar integralen av den givna funktionen av de tre givna variablerna, i det rätblocksformade definitionsområdet som anges. Den frivilliga parametern skriver över AS:INTN.
CreateGradientVectorField	TEXT, TEXT, TEXT, TEXT, ℝ, ℝ, ℝ, ℝ, ℝ, ℝ[,	SET: $\mathbb{R}^6$	I rummet: skapar gradientens vektorfält av funktionen definierad i

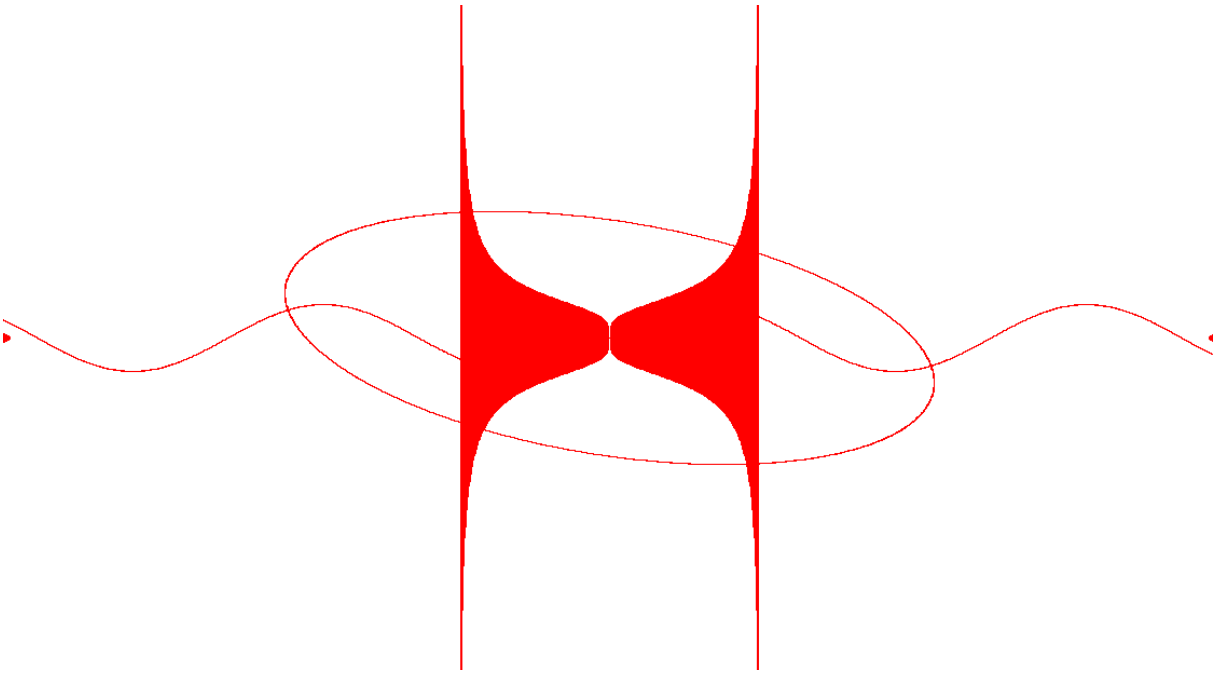
	dist ∈ ℝ][, *coord*: TEXT]		rummet i det givna rätblocksformade området. *dist* anger avståndet mellan vektorerna och *coord* anger koordinatsystemet (t.ex. kartesiskt eller sfäriskt). **DrawVectorField**
CreateDivergenceColorSpace	FUNCTION, ℝ, ℝ, ℝ, ℝ, ℝ, ℝ[, *coord*: TEXT]	SET: $\mathbb{R}^4$	Skapar ett färgat område i rummet svarande mot divergensen av det givna tredimensionella vektorfältet i det givna rätblocksformade området. *coord* anger koordinatsystemet. **DrawPointSetColor**
CreateDivergenceColorSpace	TEXT, TEXT, TEXT, TEXT, ℝ, ℝ, ℝ, ℝ, ℝ, ℝ[, *coord*: TEXT]	SET: $\mathbb{R}^4$	Skapar ett färgat område i rummet svarande mot divergensen av det givna tredimensionella vektorfältet i det givna rätblocksformade området. *coord* anger koordinatsystemet. **DrawPointSetColor**
CreateDivergenceLevelSpace	FUNCTION, ℝ, ℝ, ℝ, ℝ, ℝ, ℝ[, *coord*: TEXT]	SET: $\mathbb{R}^3$	Skapar nivåytor i rummet svarande mot divergensen av det givna tredimensionella vektorfältet i det givna rätblocksformade området. *coord* anger koordinatsystemet. **DrawPointSet**
CreateDivergenceLevelSpace	TEXT, TEXT, TEXT, TEXT, ℝ, ℝ, ℝ, ℝ, ℝ, ℝ[, *coord*: TEXT]	SET: $\mathbb{R}^3$	Skapar nivåytor i rummet svarande mot divergensen av det givna tredimensionella vektorfältet i det givna rätblocksformade området. *coord* anger koordinatsystemet. **DrawPointSet**
CreateRotationVectorField	FUNCTION, ℝ, ℝ, ℝ, ℝ, ℝ, ℝ[, *dist* ∈ ℝ][, *coord*: TEXT]	SET: $\mathbb{R}^6$	Skapar rotationens vektorfält av det givna vektorfältet definierat i det rätblock som anges. *dist* är avståndet mellan vektorerna och *coord* anger koordinatsystemet (t.ex. sfäriskt eller kartesiskt). **PlotVectorField**
CreateRotationVectorField	TEXT, TEXT, TEXT, TEXT, ℝ, ℝ, ℝ, ℝ, ℝ, ℝ[, *dist* ∈ ℝ][, *coord*: TEXT]	SET: $\mathbb{R}^6$	Skapar rotationens vektorfält av det givna vektorfältet definierat i det rätblock som anges. *dist* är avståndet mellan vektorerna och *coord* anger koordinatsystemet (t.ex. sfäriskt eller kartesiskt). **PlotVectorField**
∇	FUNCTION, $\mathbb{R}^n$[, *coord*: TEXT]	$\mathbb{R}^n$	Beräknar numeriskt gradienten av funktionen i den givna punkten. *coord* anger koordinatsystemet.
∇·	FUNCTION, $\mathbb{R}^2$ *eller* $\mathbb{R}^3$[, *coord*: TEXT]	ℝ	Beräknar numeriskt divergensen av vektorfältet i den givna punkten. *coord* anger koordinatsystemet.
∇×	FUNCTION, $\mathbb{R}^2$ *eller* $\mathbb{R}^3$[, *coord*: TEXT]	$\mathbb{R}^2$ *eller* $\mathbb{R}^3$	Beräknar numeriskt rotationen av vektorfältet i den givna punkten. *coord* anger koordinatsystemet.
ConfirmIdentity	*ident*: TEXT, *var*: TEXT, *a* ∈ ℝ, *b* ∈ ℝ	L	Undersöker om uttrycket *ident* verkar vara sant för samtliga värden på variabeln *var* i intervallet *var* ∈ [*a*, *b*]. Skulle t.ex. bekräfta att $\sin 2x = 2 \sin x \cos x$.
PathIntegral	*field*: FUNCTION; *path*: FUNCTION; *a* ∈ ℝ, *b* ∈ ℝ	ℝ	Beräknar värdet av kurvintegralen längs kurvan *path* i vektorfältet *field*

			i parameterintevallet $[a, b]$.
FlowIntegral	*field*: FUNCTION; *path*: FUNCTION; $a \in \mathbb{R}$, $b \in \mathbb{R}$	$\mathbb{R}$	Beräknar flödet genom kurvan *path* i det tvådimensionella vektorfältet *field* i parameterintervallet $[a, b]$.
FlowIntegral	*field*: FUNCTION; *surf*: FUNCTION; $a \in \mathbb{R}$, $b \in \mathbb{R}$, $c \in \mathbb{R}$, $d \in \mathbb{R}$	$\mathbb{R}$	Beräknar flödet genom ytan *surf* i det tredimensionella vektorfältet *field* i parameterintervallen $[a, b]$ och $[c, d]$.
CreateDiffEqField2D		SET: $\mathbb{R}^4$	Skapar differentialekvationens vektorfält i planet.
CreateDiffEqField2D		SET: $\mathbb{R}^4$	Skapar differentialekvationens vektorfält i planet.
CreateDiffEqField3D		SET: $\mathbb{R}^6$	Skapar differentialekvationens vektorfält i rummet.
CreateDiffEqField3D		SET: $\mathbb{R}^6$	Skapar differentialekvationens vektorfält i rummet.
CreateDiffEqSolutionPlot2D		SET: $\mathbb{R}^2$	Skapar en lösningsgraf i planet till differentialekvationen.
CreateDiffEqSolutionPlot2D		SET: $\mathbb{R}^2$	Skapar en lösningsgraf i planet till differentialekvationen.
CreateDiffEqSolutionPlot3D		SET: $\mathbb{R}^4$	Skapar en lösningsgraf i rummet till differentialekvationen.
CreateDiffEqSolutionPlot3D		SET: $\mathbb{R}^4$	Skapar en lösningsgraf i rummet till differentialekvationen.
SaveVectorToFile	$\mathbb{R}^n$, *filename:* TEXT	P	Sparar vektorn till en fil.
LoadVectorFromFile	*filename:* TEXT	$\mathbb{R}^n$	Läser vektor från fil.
SaveMatrixToFile	MATRIX, *filename:* TEXT	P	Sparar matrisen till en fil.
LoadMatrixFromFile	*filename:* TEXT	MATRIX	Läser matris från fil.
SaveSetToFile	SET, *filename:* TEXT	P	Sparar mängden till en fil.
LoadSetFromFile	*filename:* TEXT	SET	Läser mängd från fil.
SaveRealToFile	$\mathbb{R}$, *filename:* TEXT	P	Sparar reellt tal till en fil.
LoadRealFromFile	*filename:* TEXT	$\mathbb{R}$	Läser reellt tal från fil
SaveLogicToFile	LOGIC, *filename:* TEXT	P	Sparar logiskt värde till en fil.
LoadLogicFromFile	*filename:* TEXT	LOGIC	Läser logiskt värde från fil.
SaveFunctionToFile	FUNCTION, *filename:* TEXT	P	Sparar funktion till en fil.
LoadFunctionFromFile	*filename:* TEXT	FUNCTION	Läser funktion från fil.
SaveTextToFile	TEXT, *filename:* TEXT	P	Sparar text till en fil.
LoadTextFromFile	*filename:* TEXT	TEXT	Läser text från fil.
SaveProgramToFile	*progname:* TEXT, *filename:* TEXT	P	Sparar program till en fil.
LoadProgramFromFile	*filename:* TEXT	P	Laddar program från fil.
Save2DSpaceToFile	2DSPACE, *filename:* TEXT	P	Sparar 2DSPACE till en fil.
Load2DSPaceFromFile	*filename:* TEXT	2DSPACE	Läser 2DSPACE från fil.
Save3DSpaceToFile	3DSPACE, *filename:* TEXT	P	Sparar 3DSPACE till en fil.
Load3DSpaceFromFile	*filename:* TEXT	3DSPACE	Läser 3DSPACE från fil.
Save2DSpaceAsImage	2DSPACE, *filename:* TEXT	P	Sparar 2DSPACE som bitmapp.
Save3DSpaceAsImage	3DSPACE, *filename:* TEXT	P	Sparar 3DSPACE som bitmapp.
SaveInterfactToFile	INTERFACE, *filename:* TEXT	P	Sparar gränssnitt som en fil.
LoadInterfaceFromFile	*filename:* TEXT	INTERFACE	Läser gränssnitt från fil.
SaveStructTypeToFile	STRUCTTYPE, *filename:* TEXT	P	Sparar strukturtyp till en fil.
LoadStructTypeFrom-	*filename:* TEXT	STRUCTTYPE	Läser strukturtyp från fil.

File			
SaveStructToFile	STRUCT, *filename:* TEXT	P	Sparar struktur till fil
LoadStructFromFile	*filename:* TEXT	STRUCT	Läser struktur från fil.
GetColor	TEXT	$\mathbb{R}$	Hämtar färg från AS:COLORTABLE
GetColorName	$\mathbb{R}$	TEXT	Hämtar färgens namn från AS:COLORTABLE
ColorEncode	$\mathbb{R}^3$ *eller* $\mathbb{R}^4$	$\mathbb{R}$	Kodar RGB(A)-vektorn till ett reellt tal.
ColorDecode	$\mathbb{R}$	$\mathbb{R}^3$ *eller* $\mathbb{R}^4$	Avkodar färgvärdet till en RGB(A)-vektor.
RGBToHSV	$\mathbb{R}^3$ *eller* $\mathbb{R}^4$	$\mathbb{R}^3$ *eller* $\mathbb{R}^4$	Från RGB(A) till HSV(A).
HSVToRGB	$\mathbb{R}^3$ *eller* $\mathbb{R}^4$	$\mathbb{R}^3$ *eller* $\mathbb{R}^4$	Från HSV(A) till RGB(A).
CreateColor-GraphHSV2D	*res*: 3× $\mathbb{R}$	SET: $\mathbb{R}^3$	Ger färggraf. **DrawPointSetColor**
CreateColor-GraphHSV3D	*res*: 3× $\mathbb{R}$	SET: $\mathbb{R}^4$	Ger färggraf. **DrawPointSetColor**
CreateColor-GraphRGB3D	*res*: 3× $\mathbb{R}$	SET: $\mathbb{R}^4$	Ger färggraf. **DrawPointSetColor**
GetColorFromDesktop	[$\mathbb{R}^2$]	$\mathbb{R}$	Hämtar färg från skrivbordet via musen, eller genom att koordinaten för bildpunkten på skärmen anges.
CreateColor	$\mathbb{R}$	$\mathbb{R}$	Öppnar en avancerad färgvalsdialog, och returnerar vald färg.
EditColor	REF: $\mathbb{R}$	$\mathbb{R}$	Redigerar färgen i avancerad färgvalsdialog.
RandomColor		$\mathbb{R}$	Väljer slumpvis ut en färg.
CreatePieChart	$\mathbb{R}^n$[,$\mathbb{R}^n$] *eller* MATRIX	VectGraph	**DrawVectorGraphics**
CreateBoxPlot	$\mathbb{R}^n$[,$\mathbb{R}^n$] *eller* MATRIX	VectGraph	**DrawVectorGraphics**
CreateBarChart	$\mathbb{R}^n$[,$\mathbb{R}^n$] *eller* MATRIX	VectGraph	**DrawVectorGraphics**
CreateHistogram	$\mathbb{R}^n$[,$\mathbb{R}^n$] *eller* MATRIX	VectGraph	**DrawVectorGraphics**
CreatePercentageBar	$\mathbb{R}$ *eller* SET: $\mathbb{R}$	VectGraph	**DrawVectorGraphics**
CreatePercentagePie-Chart	$\mathbb{R}$ *eller* SET: $\mathbb{R}$	VectGraph	**DrawVectorGraphics**
clear			Rensar konsolskärmen.
restart			Startar om AlgoSim.
terminate			Avbryter AlgoSim.
terminate	*ThreadID* ∈ $\mathbb{R}$		Avbryter tråden *ThreadID*.
status	*ThreadID* ∈ $\mathbb{R}$		Visar information om tråden *ThreadID*.
status			Visar information om programmets processor- och minnesanvändning.
NewTab			Skapar en ny konsolflik.
CloseTab			Stänger den aktuella konsolfliken.
wait	$\mathbb{R}$		Väntar en viss tid.
WaitTimer	$\mathbb{R}$		Väntar en viss tid och visar en nedräkning.
WaitOptional	$\mathbb{R}$		Väntar en viss tid, eller tills användaren trycker på någon tangent.
WaitTimerOptional	$\mathbb{R}$		Väntar en viss tid, eller tills användaren trycker på någon tangent, och visar en nedräkning.
about			Visar Om-data.
help	*cmd*: TEXT		Visar hjälp om kommandot *cmd*.
demo			Kör AlgoSims inbyggda demoprogram.
eval	TEXT		Exekverar kommandoradskommandot som anges. Kan användas för att exekvera strängar som an-

		vändaren ger eller för att automatiskt skriva kommandon.
ExecuteScript	*filename*: TEXT	Exekverar skriptet som anges med sitt filnamn.
NewTimer	*name*: TEXT, *time*: ℝ, *command*: TEXT	Startar en ny timer med namnet *name*. Utför kommandot *command* efter tiden *time*.
AddTimeToTimer	*name*: TEXT, *time*: ℝ	Lägger till extra tid *time* till timern *name*.
PauseTimer	*name*: TEXT	Pauserar timern *name*.
DeleteTimer	*name*: TEXT	Tar bort timern *name*.
DeleteAllTimers		Tar bort samtliga timrar.
ListTimers		Listar samtliga aktiva timrar.
NewAlarm	*name*: TEXT, *time*: ℝ[,*command*: TEXT]	Lägger till ett alarm vid namnet *name*. Vid tidpunkten *time* utförs kommandot *command* (def: meddelande).
DeleteAlarm	*name*: TEXT	Tar bort namnet *name*.
DeleteAllAlarms		Tar bort samtliga alarm.
ListAlarms		Listar samtliga aktiva alarm.
NewCounter	*name*: TEXT	Skapar en ny räknare med namnet *name*.
RestartCounter	*name*: TEXT	Startar om räknaren *name*.
PauseCounter	*name*: TEXT	Pauserar räknaren *name*.
AddTimeToCounter	*name*: TEXT, *time*: ℝ	Lägger till tiden *time* till räknaren *name*.
DeleteCounter	*name*: TEXT	Tar bort räknaren *name*.
DeleteAllCounters		Tar bort samtliga räknare.
ListCounters		Listar samtliga aktiva räknare.
NewCountdown	*name*: TEXT, *time*: ℝ[, *command*: TEXT]	Skapar en ny nedräknare med namn *name*. Efter tiden *time* utförs kommandot *command* (def: meddelandepip).
RestartCountdown	*name*: TEXT	Startar om nedräknaren *name*.
PauseCountdown	*name*: TEXT	Pauserar nedräknaren *name*.
AddTimeToCountdown	*name*: TEXT, *time*: ℝ	Lägger till tiden *time* till nedräknaren *name*.
DeleteCountdown	*name*: TEXT	Tar bort nedräknaren *name*.
DeleteAllCountdowns		Tar bort alla nedräknare.
ListCountdowns		Listar samtliga aktiva nedräknare.
ListAllTimekeepers		Listar samtliga tidshanterare.
DeleteAllTimekeepers		Tar bort samtliga tidshanterare.
WindowsShutdown		Stänger av Windows.
WindowsReboot		Startar om Windows.
WindowsLogoff		Loggar ut ur Windows-kontot.
WindowsSleep		Försätter datorn i viloläge.
WindowsLock		Låser datorn.
ScreenSaver		Aktiverar skärmsläckaren.

Märk att AlgoSim kan användas för det mesta: som äggklocka, kalender, för att stänga av datorn vid en viss tid… Endast fantasin sätter gränserna.



Figur 3 Experiment med egenutvecklade algoritmer för 2D-rendering. Bilden ovan har skapats både med naivt CreateSet-approach och parametrisering.

Figur 4 Experiment med olika matematiska funktioner för sammanfogning av två matriser (bitmappsbilder, lager). Här visas funktionen *distance*.

http://www.rejbrand.se/algosim