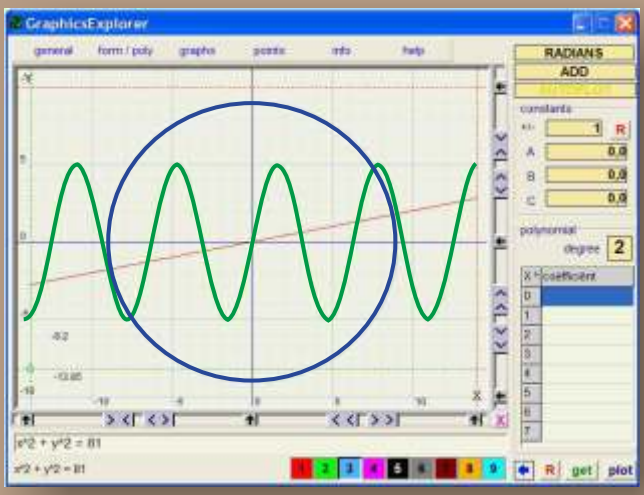
DAVID DIRKSE

PREVIEW




```
procedure ;
var
begin
 for i := 1 to 9
do
  begin
```

BLAISE PASCAL MAGAZINE
www.blaisepascal.eu

COMPUTER MATH & GAMES IN PASCAL

Foreword

At young age I wondered how on earth machines could make calculations because in my view this required intelligence.
The first step to see the light was a study in electrical engineering. The second step was joining CDC, an American computer company, and installing number crunching mainframes at scientific institutes for the next 25 years.

Some years before that period passed, it became clear that CDC was passed by its competitors and contracts were likely to end.
What to do?
Besides installing computers and fixing hardware problems I had attended over a hundred courses and had been an instructor in about 30 occasions.
Also I had witnessed several presentations by scientists who had used "my" machines. In most cases I did not understand the math and since my question how computers make calculations was sufficiently answered I decided to enter a new field and become a math teacher.
Back to college for four years of evening classes. Just in time.

Meanwhile new quests showed up. How to draw the graphics of a mathematical function on a computer screen?
How does a computer calculate the best move in a game?
How can a computer solve a puzzle?
With CDC I programmed machine code, sometimes assembler language, which was adequate to diagnose and repair hardware failures.
At a computer dump store I bought Delphi-3 for self-study. I also joined the Dutch Delphi community, which later published so many of my projects.

The future once predicted did not come true. In the early days of computers futurologists envisioned the year 2000 with mankind passing time singing and dancing while robotized factories were taking care of our material needs.
Another prediction was that complex computers would only be part-time operational. Such extremely complex machines were destined to be stuck in an undefined state most of the time.
In the seventies a Dutch scientific institute declared that academic research in micro-electronics was useless.

The present was not foreseen. The supercomputer of the eighties that occupied the space of a tennis court, fits now on my desk. It was a bargain at the local supermarket. The internet allows for worldwide sharing of knowledge and information.
Everybody is its own publisher.

The year 2000 long past and retired for six years. I don't sing and dance.
I program and balance this sedentary work by walks in the nearby dunes and along the beach. Once a year I take a break for a walking tour abroad, preferably the UK.
For the cultural side of my life I attend baroque concerts.

About calculations and machines I changed my mind. At second thought calculations have little to do with intelligence. Leave the arithmetic to machines, which are much quicker than humans in this respect. Intelligence is the ability to learn and also to understand and to discover and to design structures.

Manufacturers of beachwear recommend their products as: “made with only you in mind”.
Some projects in this book indeed were made with my former students in mind. But in most cases it was solely my personal interest, the challenge to solve a problem.
I realize that I reinvented the wheel in many cases. As a hobbyist I am permitted to do so.

Plane geometry is nice for the study of mathematics because it combines analytic reasoning with algebraic calculations. From secondary school I remember my struggles with this subject. Maybe for this reason the book contains many pages of geometry.

A glance at history emphasizes our great world. For the first time at least part of mankind lives a free and prosperous life with a broad education system and vast cultural life from football stadiums to concert halls. We own this to science, technology and abundant cheap energy.

Our society embraces new products rather quickly. However there is far less interest in the underlying technology. But designing technology is a creative process just as finger painting of course.
For myself I look in amazement to our magical world and I count my blessings.
Also I am sure there must be numerous other people with interest in technology and who eagerly want to participate in the preservation of our society and shaping the future.
Despite some sleepless nights when problems refused to rest, I have written this book with great pleasure which I hope to share with my readers.

David E. Dirkse
August 2, 2015

the Netherlands

Developmental Editor: Detlef Overbeek
Production Editor: Detlef Overbeek
Technical Editor: Peter Bijlsma
Proofreaders: Peter Bijlsma, Rik Smit,
Cover Designer: Detlef Overbeek

Pro Pascal Foundation -Stichting Ondersteuning Programeertaal Pascal
Edelstenenbaan 21 3402 XA IJsselstein Netherlands

This edition is registered by the Nederlandse Koninklijke Bibliotheek

Office@blaisepascal.eu **http://www.blaisepascalmagazine.eu**
ISBN: 97 89 49 09 68 106

CONTENTS

Delphi programming

Ict - algorithms

Introduction
Here you see a 3 dimensional tic-tac-toe game. One in progress one done. It is a two-player version, written for Windows computers.

Winner is the one who is the first to place three adjacent O or X characters horizontal, vertical or diagonal.
See figure 1

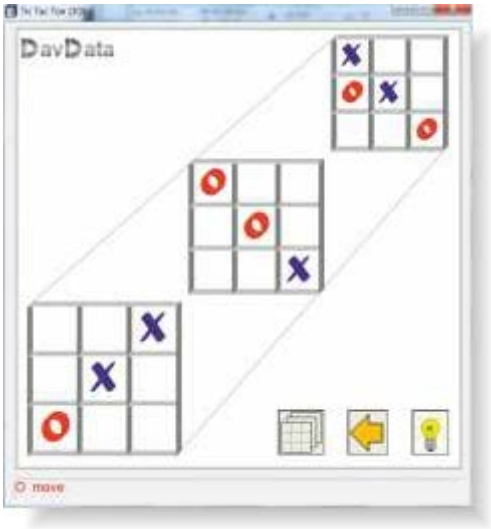


Figure 1a: 3D Tic-Tac-Toe in progress

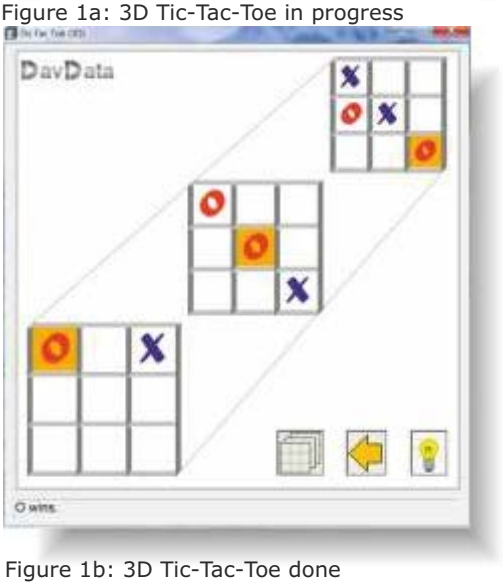


Figure 1b: 3D Tic-Tac-Toe done

A single player may try to find the best strategy.

There are three buttons:
- cube: new game
- arrow: take move back
- lamp: analyse board state

Analysis displays the result of a move in each field.
W3 means : winning in 3 moves.
L5 means : losing in 5 moves.

You can download the accompanying files:
ttt-3d.exe code

The complete project and code is available at chapter: 47

CHAPTER 2 - GAMES

Introduction
Match5 is a board game.
It may be played in two modes: 1. player against computer, 2. player against player.
The computer is self-learning. In case of a loss, the board state is archived to avoid the same error a second time. Each time the player wins, it becomes a bit harder to win again.
Actually, the player competes with him/her self.

The game

The board counts 9 x 9 fields. A field may hold a red or a blue ball. The computer plays with **red**, the player with **blue**. Alternately they place a ball of their color in an empty field. Winner is the first to place a horizontal, vertical or diagonal row of five balls of his color. Look at the picture below, where the computer plays red and the player blue.
It is clear that the computer still has zero knowledge and is beaten easily by blue.

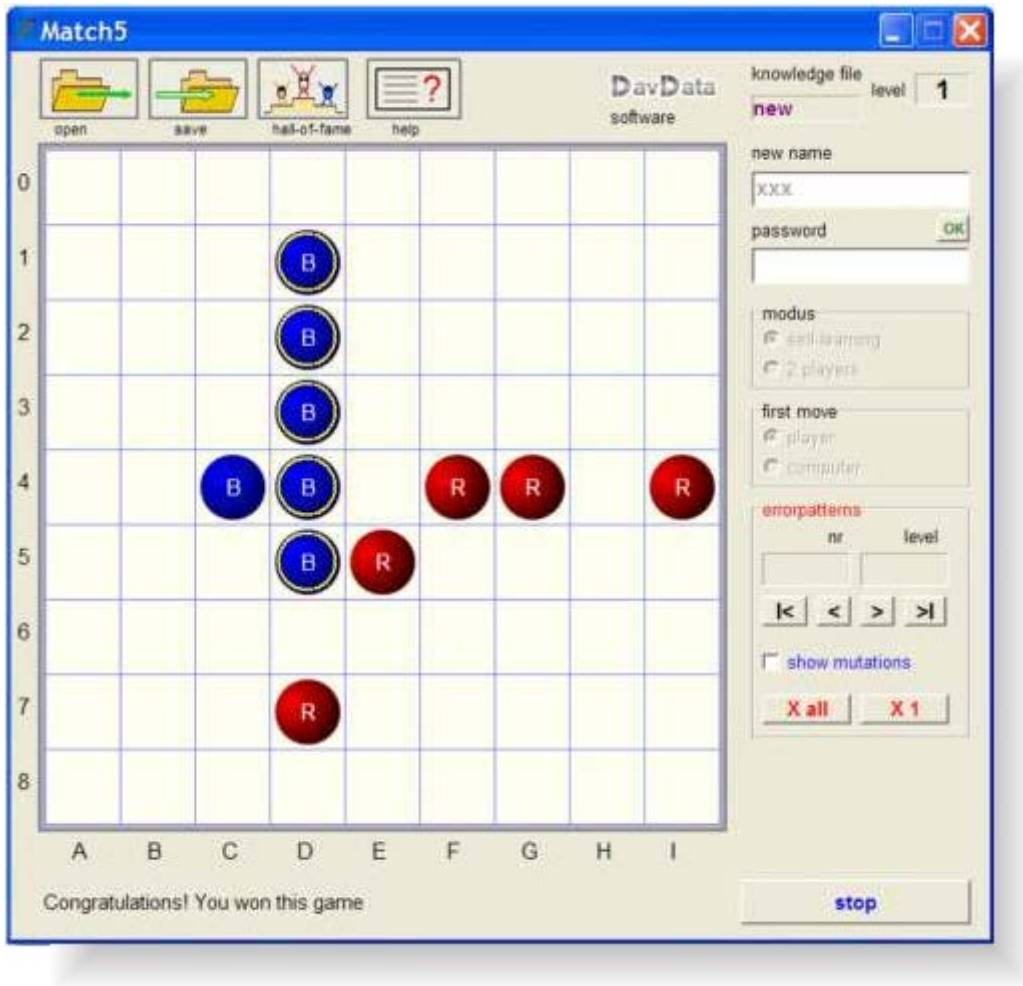


Figure 1: Match5

Introduction
Colorstacks is a logistic puzzle with 7 levels. The game consists of columns with colored blocks. The lowest block depicts the color of the column. This block is fixed.
The other blocks may move, one or two at the time, to another column. Rule is that only blocks of the same color may rest on each other. In the starting position a column only contains blocks of "foreign" colors. In the final, solved, position each column only contains blocks of its own color.

Options
- search for solutions
- move blocks using the mouse
- take moves back
- restore game to starting position
- replay earlier solutions
- have the computer search for solutions
- save a solution on disc
- open a solution from disc
- print a solution

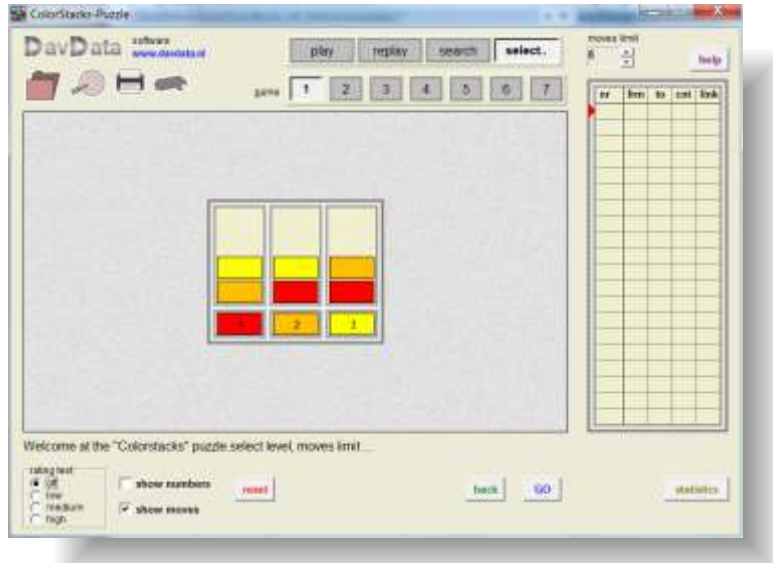


Figure 1: The program in progress

Information
The program has in-line help information. It is written for Windows in the Delphi (7) programming language. There is no installation procedure: simply copy **colorstacks.exe** to a directory of choice.

The search algorithm
I do not know of an analytical approach of this puzzle. But I feel that there must be one. The computer uses the "brute force" (*systematically try all move sequences*) method to find solutions.

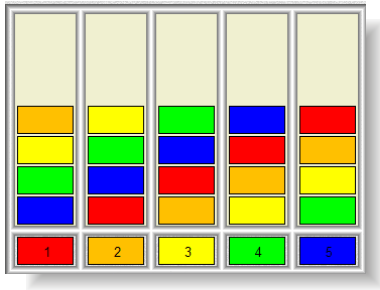


Figure 2: starting position of level 3

To limit the search time however, a number of tricks are applied to avoid redundant work. Solutions of levels 1..6 are found by the computer in (milli) seconds. For level 7 however I feel that the shortest solution has not yet been found.

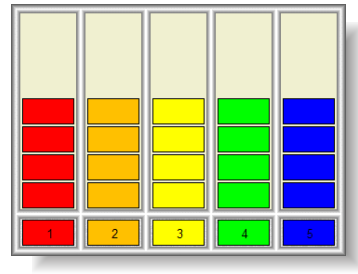


Figure 3: solution of level 3

You can download the accompanying files: colorstacks.exe

The complete project and code is available for download colorstacks.zip

Introduction
"SHIFT" is a very difficult puzzle, unsolvable at first glance. Not invented by myself, but found on the web where is was called "the most difficult puzzle in the world". To solve the puzzle using mouse or keyboard, I wrote this Delphi-7 program. Also the program can search for solutions in case the user gets tired trying. Below are pictured two (half size) screenshots. The left is the original- and the right is a possible solved board state.

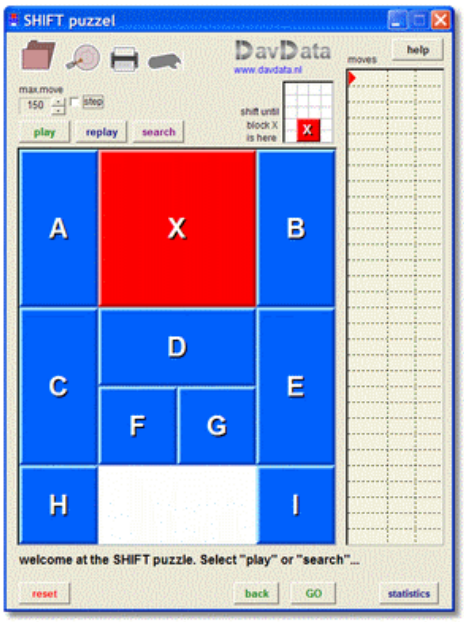


Figure 1: Original board state

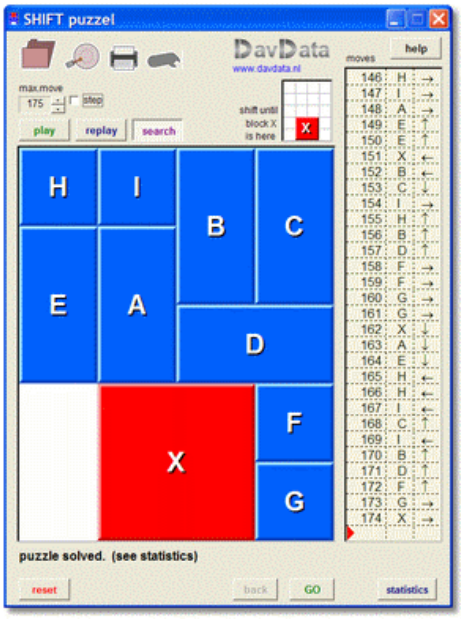


Figure 2: A possible solved board state

The puzzle consists of a board having 4 x 5 fields. Blocks of size 1x1, 2x1, 1x2, or 2x2 are shifted until the red 2x2 block occupies the the fields at the bottom center. The position of the other blue blocks is not important.

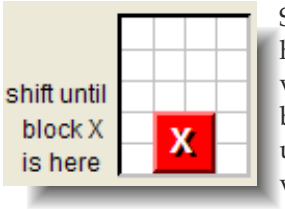


Figure 3

Some websites have an on-line version of SHIFT, but the primitive user interface is very frustrating.

This SHIFT program has the following options:

- find solutions yourself (*use mouse or keyboard to move. Moves can be taken back*)
- replay previous moves
- have the computer search for solutions
- save moves on disc
- open moves from disk
- print moves including picture of initial- and final board state
- In-Line help information

I unveil that a solution counts over 100 - but less than 200 moves (*the solution in the screenshot is not he shortest one*).

You can download the accompanying files: shift.exe
For the code see chapter 68

Introduction
"Jumping Goats" is a 3 - level puzzle for all ages. Below is a reduced picture of the start for level - 3:

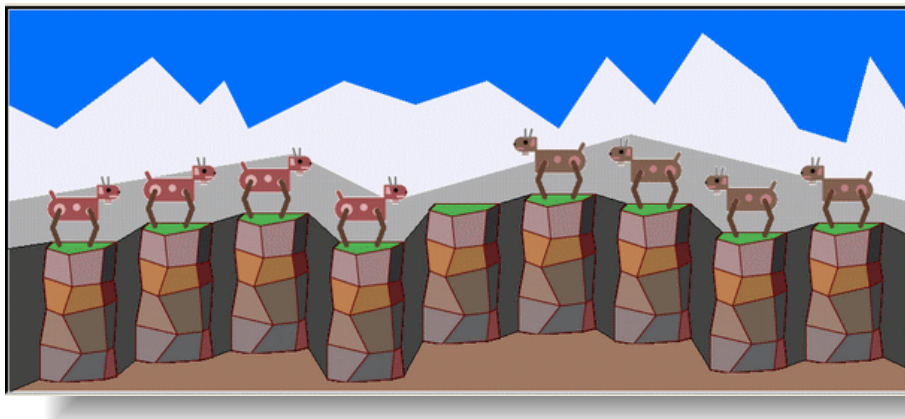


Figure 1: Gameplay start

Rules

- a goat may jump forward one or two places to an open spot
- a puzzle is solved if the goats have changed place.

And below is a solved game:

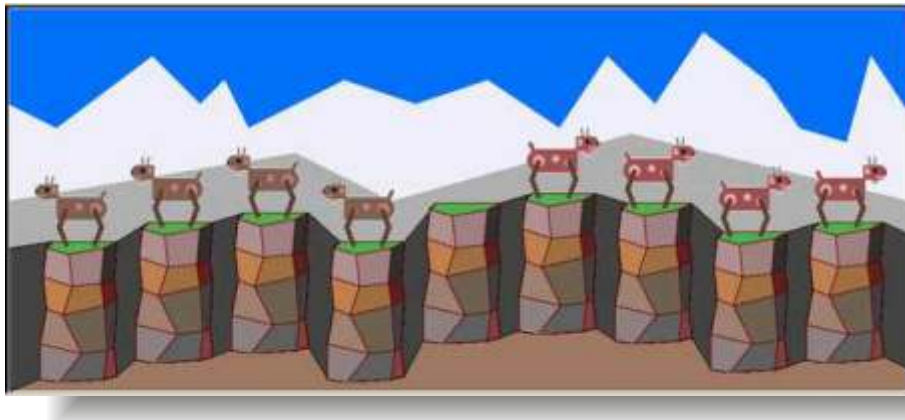


Figure 2: Gameplay finished

The game may be played using mouse or keyboard.

Mouse

- click on goat that has to jump
- click on right bottom button to restart or progress to next level.
- click on button several times to go back to a previous level

You can download the accompanying files: goats.exe goats.zip

INTRODUCTION
This connect4 game has 8 levels and 8 strategies.
In "analyse" mode the program per move calculates the result:
win or lose in .. moves.
Also there is a **Hall-of-Fame.**

The Game

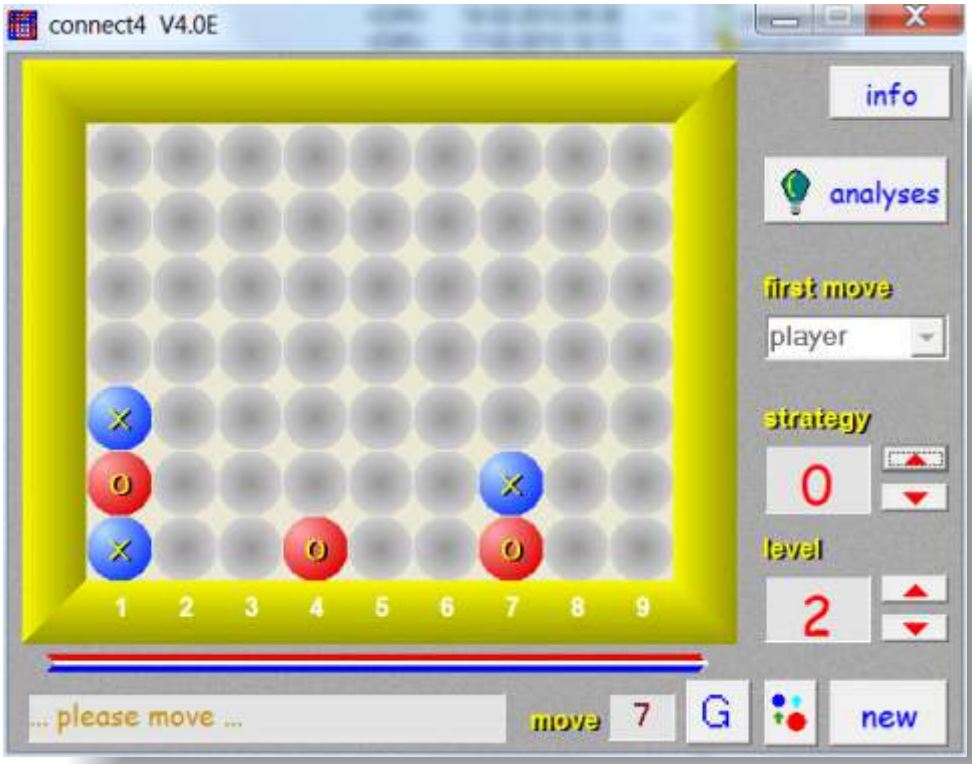


Figure 1: A new gameplay

My version of CONNECT4 is single player, you play against the computer. The board has 9 columns and 7 rows, resulting in 63 fields.
Computer and player alternately place a ball of their color in a field. The computer always plays with red, the player with blue. The columns are filled bottom-up, so the balls are actually dropped down a column. The winner is the first to achieve a horizontal,vertical or diagonal line of 4 balls of his color.

You can download the accompanying files: connect4.exe

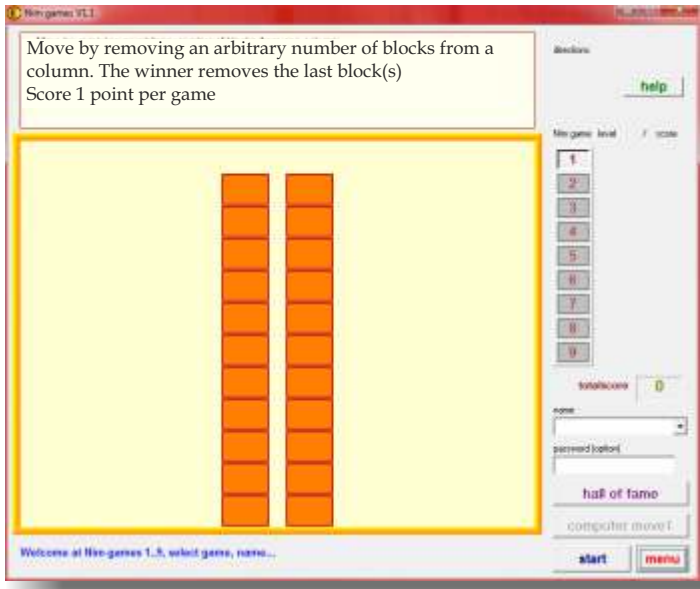
For the code see chapter 83 page page 313

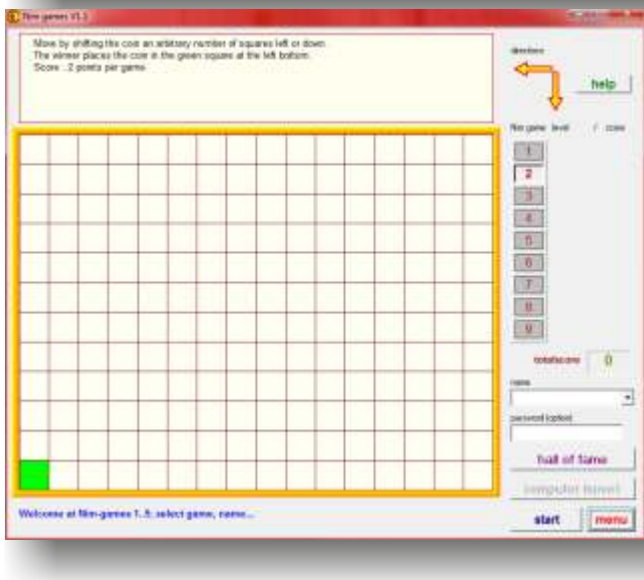
Introduction
Nim games are a category of board games that may look very different, but the math behind is the same.
The following characteristics are shared

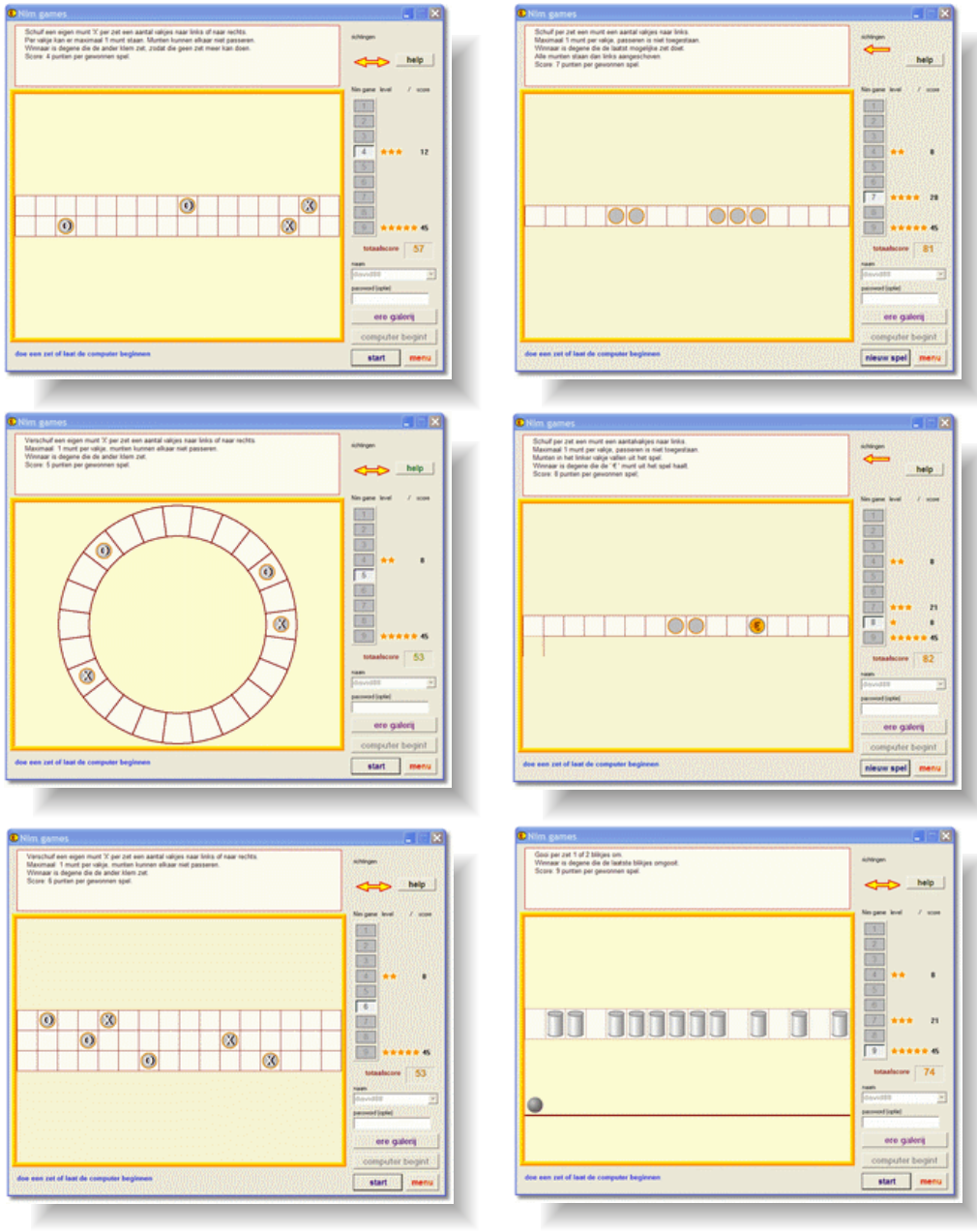
- finite number of moves
- always ends with winner and loser
- winner is the player that first reaches the final board state
- each board state is either good or bad

About these Nim games
This program offers 9 Nim games: from easy (1) to difficult (9).
They are all single player versions, but the player can defeat the computer in every game.

When the mathematics are understood, a board state may be analysed as "good" or "bad". A "bad" board state has a connection to a "good" board state.
A "good" board state only has connections to "bad" board states.
The goal is to unveil the similarity between the games and to find a winning strategy.
Below are pictures of the various Nim - games (*reduced images*) .







Installation

Nim9 is written in the Delphi programming language and for all Windows versions. Nim9 ships as a single .exe file. There is no installation procedure. Simply copy Nim9 to a directory of choice. The program has In-Line help.

Introduction
This article explains the math behind Nim games.

COMMON CHARACTERISTICS:
- there are two players
- there are a finite number of moves
- each game ends with a winning player
- there is no luck or chance : a game state is either a winning or a losing one

Also may be noted:
- a game position is "good" or "bad". The final position is "good" of course.
- from a "bad" position there exists a move to a "good" position.
- from a "good" position no move exists to a next "good" position.

The name Nim was introduced by the Canadian mathematician Charles L. Bouton, which analysed the versions 1..8 of the Nim games in the year 1902. The analysis of game number 9 was realized much later, in 1937, by the German mathematician Roland Sprague and some years later, independently, by P.M.Grundy.

A QUICK SURVEY
Please look at **Nim game 2.**
Goal is to shift the coin to the green field at the left bottom.
A move may shift the coin left or down by any number of fields (but not both).
The final state (0,0) is "good".
As a consequence, any position after a backward move originating in field (0,0) is "wrong".
The column above (0,0) and the row right of (0,0) may by all marked wrong (red dot).
The nearest "good" position is (1,1), marked green. Above and right of (1,1) fields must be marked "wrong". And so on, see figure 1. right.

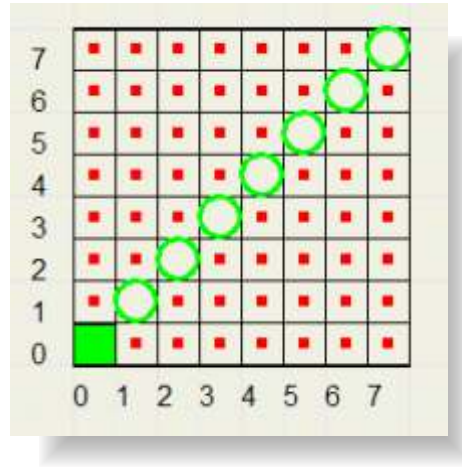


Figure 1.

The winner is the first to place the coin on a green field. If the starting position is green, then have the computer do the first move. Figure 2 shows the similarity between games 1 and 2.

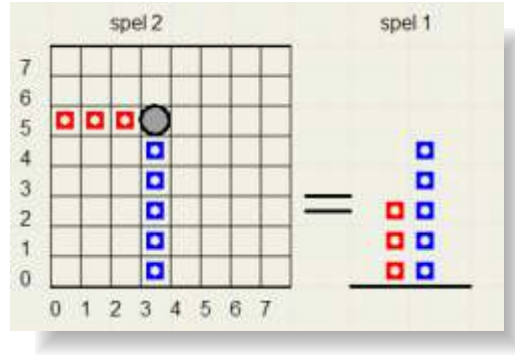


Figure 2.

Note: "spel" is Dutch for "game".
In game 1 we remove squares from a chosen pile, in game 1 we "remove" a number of horizontal or vertical fields. Game 1 is won by the player who makes the columns equal.

Nim game 3
Here, the coin may move as well diagonally in the left-down direction. This makes the up-right diagonal fields starting at a good field all wrong.
See figure 3.

Introduction

Peg Solitaire is a single player puzzle.

The board consists of holes (33) and pegs (initially 32). The center position only is open. The final, solved, state has left 1 peg in the center position when 31 moves have striked the other pegs.
A move takes a peg over its neigbour (horizontally or vertically) to an empty hole. The peg that was jumped over is removed from the game. See the picture below for a reduced image of an initial- and a solved game:

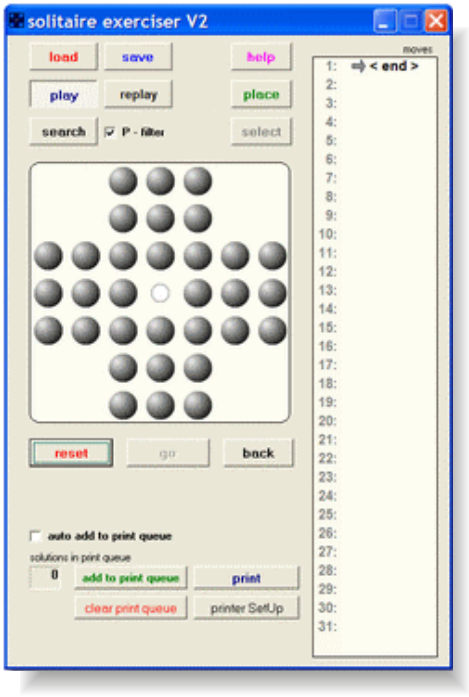


Figure 1: Initial board state

Installation
Platform : all windows versions.
There is no installation procedure:
simply copy the program to a directory of choice.
The windows registry is not changed.

You can download the accompanying files: solitaire.exe

For the code see chapter 67 page 225

Program Options

play search for solutions
replay replay previous moves or solution
search have computer search for solutions
place place balls at board to create starting position for search
select select 1 of 12 preset games, from easy to difficult
save save board / solution to disk
reload reload game from disk
add to print queue
add a solution to the print queue
print print previously stored solutions
in line help
P-filter
permutation filter removes similar solutions : same moves in different sequence

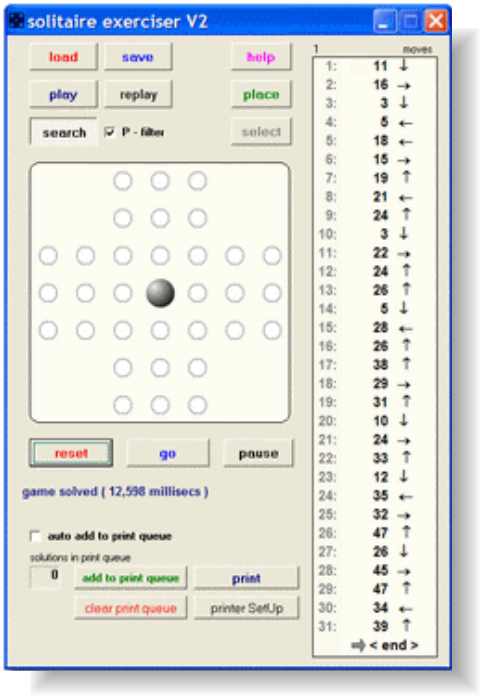
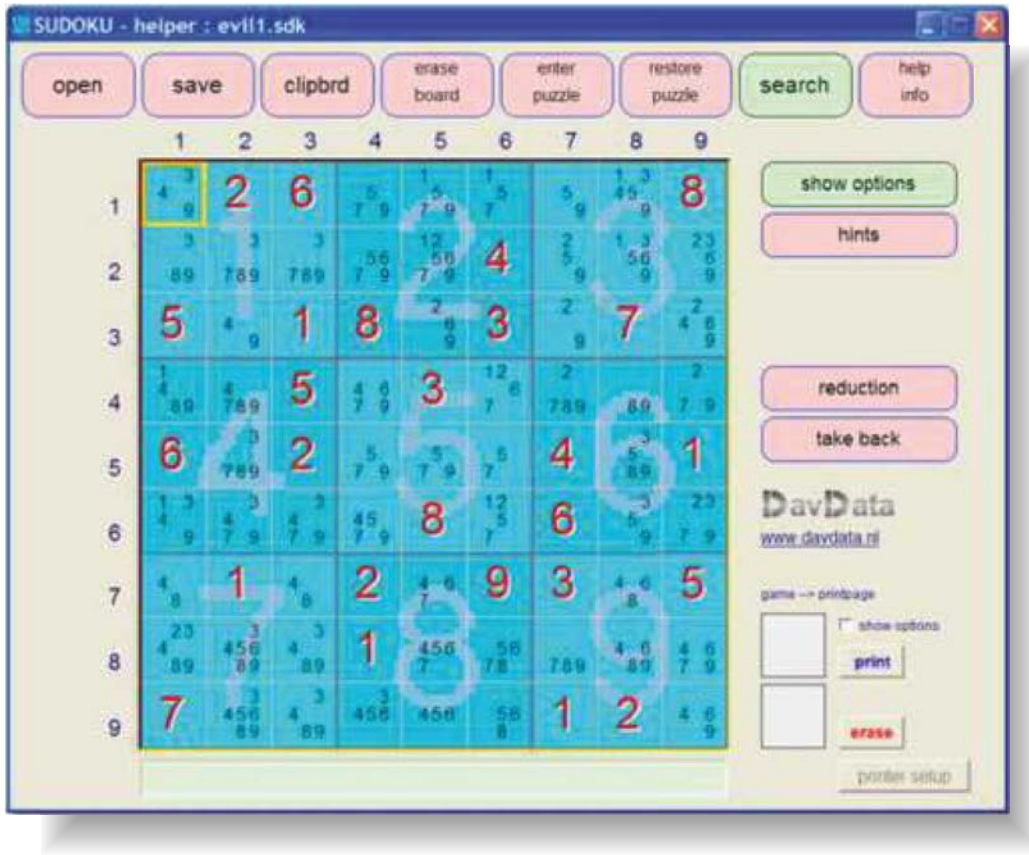


Figure 2: Final solution

CHAPTER 9 GAME

SUDOKU HELPER

Introduction
Sudoku is a very popular number puzzle. This Sudoku Helper - Solver program assists in the solution of Sudoku puzzles from easy to very difficult. Below is a reduced image of the SUDUKO - helper / solver.



New in version 3: Printing of puzzles, 1 or 2 per page. With/without digit options.

INSTALLATION
Click on the download icon (lightning) at the top op this page to download the Sudoku Helper - Solver.

Sudoku Helper - Solver is freeware and may be distributed without restrictions.
It ships as a single (.exe) file.
There is no installation procedure. Just copy it to a map of your choice.
The Windows Registry is not changed.

PROGRAM INFORMATION

programmer	David Dirkse
size	490kB (1 .exe file)
system	Windows 95 +
screen resolution	minimal 1024*768
color depth	minimal 16 bits
documentation	in - line help
programming language	**Delphi 7**
version 3	aug. 2011

CHAPTER 10 GENERAL

Introduction
For several reasons, western countries want to decrease their dependency on oil. Alternatives for oil are:
- windmills
- solar cells
- thermal solar heat collectors / solar towers
- methanol production from agricultural waste
- nuclear energy
- hydroxen (*liquid or gas*)

In the case of windmills, solar collectors or solar panels a problem arises:
sunshine and wind are variable to a high degree. During favourable conditions, excess energy must be stored to survive periods of shortage.

This article focuses on some possibilities to buffer energy. I promise interesting comparisons, especially by adding the energy-densities of different solid fuels and uranium used in nuclear power plants. Not accounted for are (*maintenance*) costs and efficiency of the systems.

Motivation for this small study was an article stating that the world's total need of electricity can be met by a solar collector system occupying the area of France in the Sahara desert.
In such a thermal solar power plant sunlight is projected on tubes containing water. Water is heated to steam which drives turbines and generators.
Electricity is transported by wire to the consuming countries.
Every city or houshold however should buffer an amount of energy for at least a few days to survive cloudy days, malfunctions or terrorist attacks.

The following storage systems are considered:

ENERGY STORAGE SYSTEMS

- electrical / chemical : batteries
- mechanical : potential energy
- mechnical : kinetic energy
- pneumatical : pressured air

Objective is to store an amount of energy sufficient for one household for one day..

Energy in general

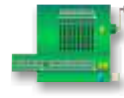
Energy is what causes change or movement:
- electrical : moving electrons
- mechanical : moving matter
- thermal : oscillating atoms
- chemical : binding energy of atoms

Energy is never lost.
One type of energy may be transformed into other types:
- chemical to electrical: battery
- electrical to mechanical: motor
- mechanical to electrical: generator
- chemical to thermal: combustion
- thermal to chemical: growth of plants

The unit of **force** is **Newton:**
1 Newton accellaretes a mass of 1 kg by 1 m/s² (in case of no friction).
A mass of 1kg on earth experiences a gravitational force of 9.8 Newton.

The unit of **energy** is **Joule:**
In the case of mechanical energy:
If a force of 1 Newton moves an object over 1 meter, the required amount of energy is 1 Joule.
This energy is lost as (*friction*) heat.

Energy E = F.d (Joule) if a force F moves an object over a distance d.
Energy per second is called **Power**.
If a constant power is used during s seconds, is the total energy:
E = P.s
P = E / s.



The unit of **power** is **Watt:**

1 watt = 1 Joule / second.

Because 1 Watt is very little power, the power of electric stoves or motors is indicated in kiloWatt (kW). But 1 kW during 1 second is still a small amount of energy. Electric companies therefore count in **kWh, kiloWatt * hours**. so,

`1 kWh = 1000*3600 = 3.6MJ`...*{3.6 mega-Joule}*

The average use of a household in the Netherlands is : **10kWh = 36MJ** per day.

Liquid fuels

One liter gasoline produces **`30MJ`** when combusted, so a household would need a tank of **`1.2 liters`** daily.

The energy-density of methanol is only little less: 2 liter is needed for **`36MJ`** of energy. Methanol may be produced from agricultural waste. It may be used in fuel cells to generate electricity.

Liquid hydroxen has an energy-density of **`8.4MJ / liter.`**

This seems interesting but problems are big: it can only exist near absolute zero, about 270 degrees Celsius below zero.

Batteries

The best type of rechargable battery is Li-ion, having an energy-density of **`0.1kWh/kg = 360kJ/kg.`**

So, a battery of 100 kg is needed to survive a cloudy day without a breeze.

Gas

Vapoured hydroxen under pressure of 200 bar has an energy-density of **`1.9MJ / liter`**, this is 16 times less than gasoline. So for one day, a tank of 19 liters is needed.

Compression needs much energy.

Natural gas consists for 90% of methane. The energy-density is **`35MJ/m³.`**

Potential energie

This is the principle of the cuckoo-clock : a lifted weight.

In our home we build a shaft in which a large weight can move up and down.

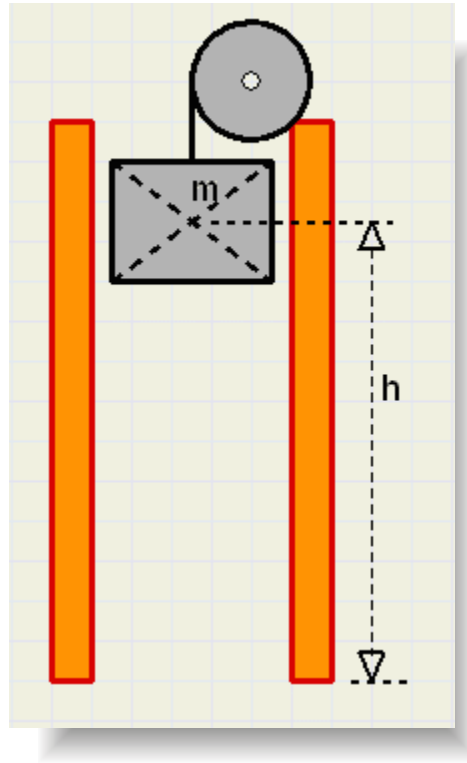


Figure 1: Potential energie

When a mass m (kg) is lifted h (meter), the required energy is

`E = m.g.h,`

where **`g = 9.8 m/s²`**, the accelleration of gravity on earth.

In case of a concrete block of 1m³ lifted 10 meter and a density of the concrete of **`2500kg/m³`** the total energy amounts:

`E = 2500 * 9 * 8 * 10 = 245kJ.`

So 147 blocks are needed to supply the energy for one day.

A 1kW vacuum cleaner can be operated for 245 seconds or little over 4 minutes before the block reaches the ground.
A nice thing is that the position of the block immediately shows the possibilities for the rest of the day:
in case of downward movement one is discouraged to switch on the flat-iron.

Kinetic energy

This is an application of the flywheel. Flywheels are used to bridge a short power loss. In this time, a diesel generator can be started.

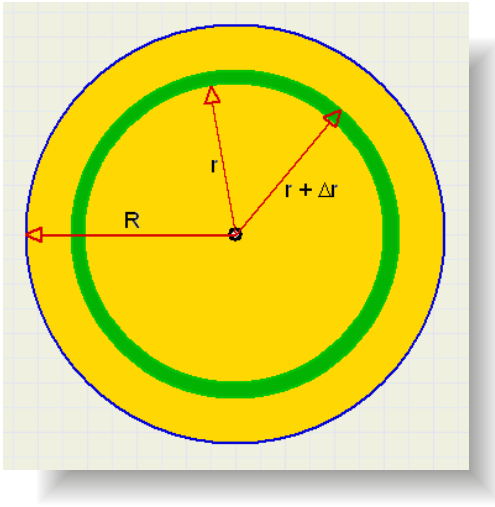


Figure 2.

We calculate the energy of a flywheel and start with the formula for the kinetic energy of a mass m moving at v (m/s):

$$E = \tfrac{1}{2} m v^2$$

Problem : the speed depends on the distance from the center.
We consider a small concentric circle with radius r and width Δr.(*green in figure above*). Because the width is very small, every point has the same speed.
The material has a density of ρ (kg/m^3).
The thickness of the wheel is d (*meter*).

The mass of the layer at distance r from the center is :

$$\Delta m = 2\pi r.d.\Delta r.\rho$$

At n revolutions per second the speed is, at distance r from the center :

$$v = 2\pi r.n$$

combining, the kinetic energy of the layer is :

$$\Delta E = 4\pi^3 d\rho n^2 r^3 \Delta r$$

so

$$E = 4\pi^3 d\rho n^2 \int r^3 dr$$

$$E = \pi^3 d\rho n^2 R^4 \quad \text{....integrated over 0...R}$$

However, the mass of the complete wheel is

$$M = \pi R^2 \rho d$$

which simplifies the formula

$$E = Mn^2\pi^2R^2$$
$$E = M(\pi Rn)^2$$

The speed at the perimeter is

$$v = 2\pi Rn$$

so

$$E = \tfrac{1}{4} Mv^2$$

The density of concrete still **2500 kg/m³**, we install a concrete flywheel with radius 1 (m) and thickness **0.25m.**
The number of revolutions is 25 / s.
The amount of stored energy is in this case:
E=0.25*3.14*0.25*2500(2*3.14*25)²
=12 MJ
Not bad!

We can operate the vacuum cleaner for over 3 hours. Unfortunately a lot of noise must be taken for granted: devices like this simulate a thunderstorm.

Pressured air

Recently the press reported of cars being moved by compressed air.
Time to investigate the amount of energy in a cylinder holding compressed air.

We consider a very long cylinder where a piston compresses the air as it it forced inside.

Newspaper article
„...The new -Gemini- windfarm, largest in the Netherlands, will count 150 windmills of 4MWatt capacity each.
The project costs 3 billion euro's and will provide 785,000 households with electrical energy."

LIE 1
The Gemini windfarm will provide exactly zero energy to housholds on calm days. Modern societies need electrical energy around the clock, so standby power stations are necessary.
Less dependency on fossile fuels is out of the question.

LIE 2
In energy calculations on windmills, the so called "production factor" is used.
This is the average energy production divided by the peak energy.
For windfarms at sea, a realistic figure is 33%.
It means, that for every day of sufficient wind there are two days without wind.

A (*Dutch*) household needs about 10kWh electrical energy per day.

The Gemini windfarm average daily production is
`0.33 * 150 * 4MWatt * 24 = 4.752GWh.`
So, energy is supplied to
`(4.752 * 109) : (10 * 103) = 475,200`
households.

LIE 3
The number before is rather optimistic.
First we recall, that calculation of averages must imply addition. Without addition, no average can be calculated.
The energy of a windmill must be added, which means: stored for later use.
In this storage- and unloading process, energy is lost.

REALISM
Let's observe a reliable energy system.
Pumped storage is the cheapest and most efficient storage system.
However, for the Netherlands and other flat and overcrowded countries this is no option. Gas storage is (*but research is necessary, there are no working mass storage systems yet*). Electrical energy from the windfarm is used for the electrolyse of water (H_2O) into hydroxen (H_2) and oxygen. Hydroxen is nasty to handle, therefore using carbondioxide (CO_2) we produce methane gas (CH_4).

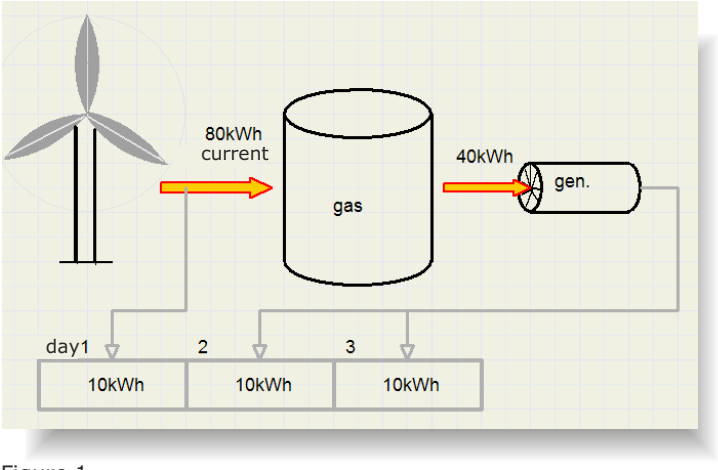


Figure 1.

The efficiency of this process of current to methane is about 50%. Burning CH_4 to power generators for electricity production again has an efficiency of 50%. Total efficiency of current --> gas --> current is 25%.

No code available

Introduction
In a triangle with rectangle sides of length a en b and hypothenuse c the equation : $a^2 + b^2 = c^2$ is true (*fig.1*).

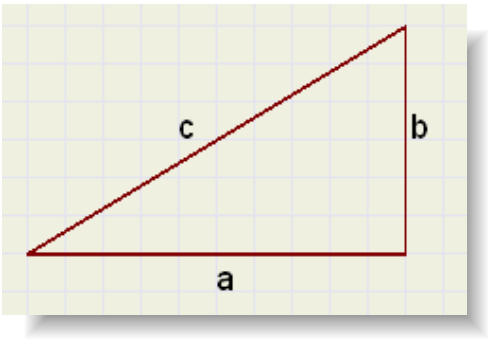


Figure 1: $a^2 + b^2 = c^2$

This equation is known as the Pythagoras theorem. See figure 2.

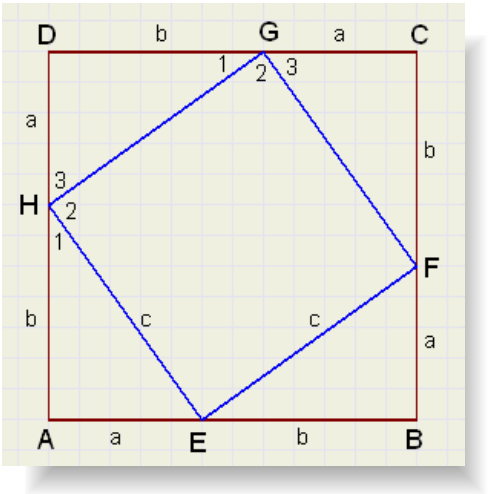


Figure 2: the Pythagoras theorem

This theorem is the basis of many other theorems and calculations.
This article supplies four proofs.

Proof -1-
We start with square ABCD.
On the sides, points E,F,G,H are placed such that AE = BF = CG = DH.
We call this distance a.
A side has length a + b.

THE PYTHAGORAS THEOREM

The proof has two parts:

1. Proof that EFGH is a square.

The construction implies that triangles AEH,BFE,CGF and DHG are congruent.
So, in any case, EFGH is a rhombus.
The length of a side we call c. Angles (1) and (3) are 90 degrees together, so angle (2) is 90 degrees.
EFGH is a square.

2. The equation:

Let's write an area as [ ]
Look at the areas and notice that:

$$[ABCD] = 4 * [AEH] + [EFGH]$$
$$(a + b)^2 = 2ab + c^2$$
$$a^2 + 2ab + b^2 = 2ab + c^2$$
$$a^2 + b^2 = c^2$$

Proof -2-
Now we start with two unequal squares: ABCD with a side of a BEFG with a side of b. Point H is on AB such, that AH = b. Which implies that HE = a (*fig.3*).

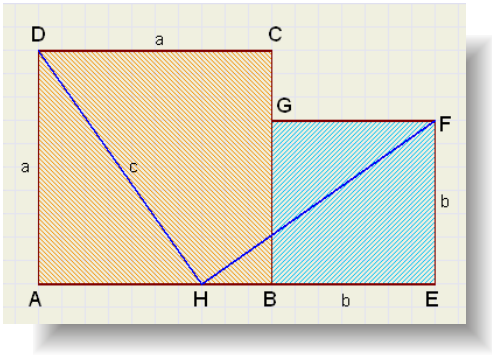


Figure 3

We use the scissors and cut triangles ADH and HEF placing them on sides DC and GF. Similar to proof -1- we see, that HFID is a square. The coloring shows:

$$a^2 + b^2 = c^2$$

See figure 4:

Introduction

Picture below shows two circles with centers M and N. The smaller circle touches the bigger one inside and has half the diameter. The smaller circle rolls inside the bigger one without friction. Center N moves over the green circle.

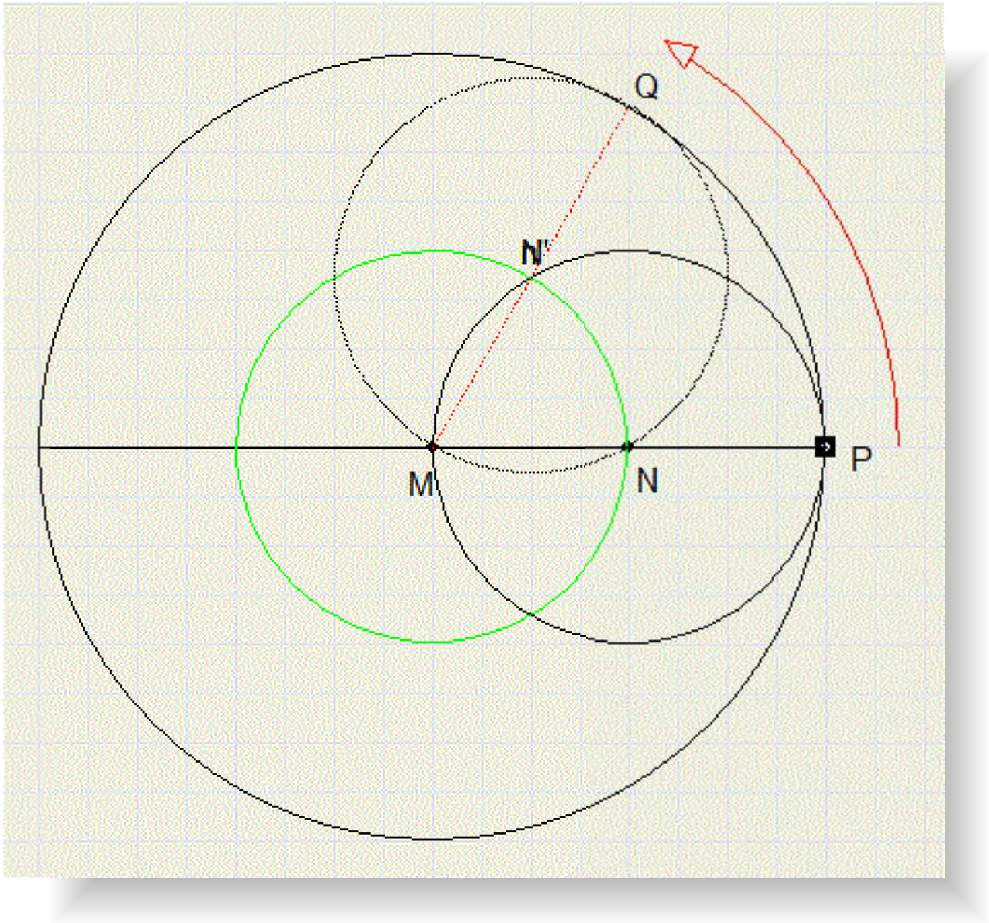


Figure 1.

After rotation, N is moved to position N'. The circles now touch in Q.

Question:

which way does P travel if it is fixed to the small circle? In the picture above line MN is rotated to MN', we call this angle v. This implies an arc on the bigger circle of $v\pi/180$ degrees, if the radius is 1.

The smaller circle has half the radius, so its rotation will be double the angle: 2v degrees. Rotation is clockwise.

Introduction
While waiting for train- or bus connections, travellers pass time by reading a book. In most cases, these books have little value and there is no need to keep them in good shape. Also, the book will not be finished during one period of waiting. Therefore, the page to continue reading has to be remembered. Three ways are available to accomplish that:

1. Pages read are torn from the book and discarded. This method is recommended for walkers.
2. A bookmark is inserted after the page that was last read. Disadvantage: it may me be lost.
3. The page last read is folded such, that a small triangle sticks out.

In this article we choose the third option. The problem is how to make the area of GHI maximum

This problem carries the danger of getting stuck in endless calculations. A good strategy is of major importance.

We assume that the Pythagoras lemma is needed as well as the theory of similarity of triangles. In figure 1 we extend edge BC and fold line EG to make triangles. See figure 2 next page.
For convenience we define:

```
AB = a
BC = b
AF = x
BE = EF = p
TC = h
CG = GH = v
```

Markers denote equal angles.
(*calculations left to the reader*)
Final goal is a formula that expresses the area of triangle GHI in x,a and b.

The sign ~ means similarity. The area of triangle GHI is written as [GHI]

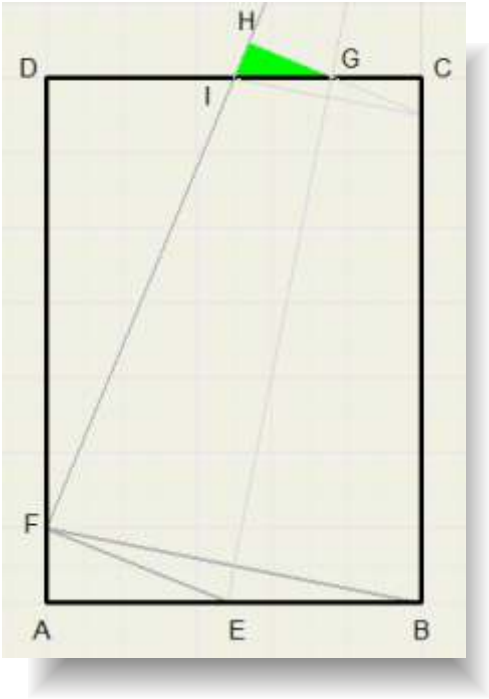


Figure 1: EG is folding line. B folds over F , C becomes H.
BE=EF , CG=GH

The strategy
p may be calculated from width a and x.
ΔTBE ~ ΔBAF, so: h may be calculated
ΔTCG ~ ΔTBE, so: v may be calculated

ΔGHI ~ ΔEAF
Define f = v : AE, then
[GHI] = f²[EAF], because the ratio of areas of similar triangles is the squared ratio of similar edges.

Calculation of p
Pythagoras lemma in ΔEAF

$$x^2 + (a-p)^2 = p^2$$
$$x^2 + a^2 - 2ap + p^2 = p^2$$
$$2ap = x^2 + a^2$$

$$p = \frac{x^2 + a^2}{2a}$$

Introduction
This article describes the calculation of the turning radius of a car or bicycle.
This radius depends on two things:
- the wheelbase w , which is the distance between the front- and the rear wheel
- the angle of the front wheel

We suppose that only the front wheel is able to turn. See figure 1 and 2 below:

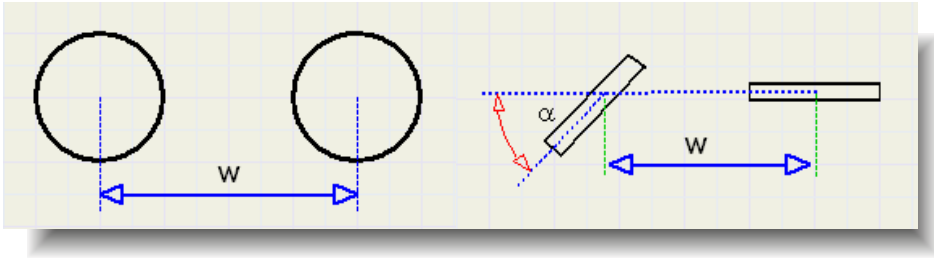


Figure 1: Side view

Figure 2: Top view

Calculation
The front and rear wheel follow a circle with the same center.
At all times, the direction is perpendicular to the radius.
See figure 3.

The radius of the front wheel is R, the rear wheel r.
From the figure 3 we conclude:

$$\sin\alpha = \frac{w}{R}$$

so

$$R = \frac{w}{\sin\alpha}$$

also

$$\tan\alpha = \frac{w}{r}$$

so

$$r = \frac{w}{\tan\alpha}$$

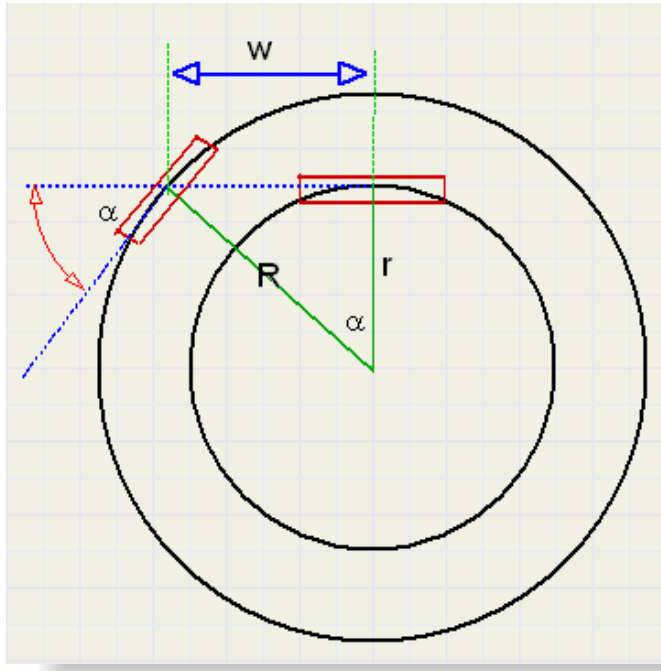


Figure 3: Turning radius for front- and rear wheel

No code available

CHAPTER 17 - MATH

Introduction
Boolean algebra is a type of algebra that is used in the design of (*digital*) logic circuitry, computer programs such as search engines and in general in analytic reasoning. It is an arithmetic interpretation of Proposition Logic and is also similar to Set theory. Boolean algebra was designed by the British mathematician George Boole (1815 - 1864).

A program (Logic 10) that handles Boolean algebra formulas is described in the section Programming Truth Table Reduction Chapter 49, page 152

Variables and operators
A boolean variable may have the value "true" or "false". For the purpose of arithmetic operations,
"**true**" is represented by "**1**" and
"**false**" is represented by "**0**".
In this article, a boolean variable is denoted by a single character like A, B or C.
A variable stands for a proposition such as "*start switch pressed*" , "*temperature within range*", "*counter value* = 6" , "*John is 10 years old*" ....etc.

NOTE, that a proposition is either true(1) or false(0):
Outside, the sun is shining or not.
A switch, or a door, is open or closed.

Boolean Algebra knows 3 basic operations which are **AND, OR** and **NOT**

Using these operators, Boolean variables may be combined into formulas for more complex propositions.

AND

The operator for **AND** is a single dot ".".

```
A B A.B
0 0  0
1 0  0
0 1  0
1 1  1
```

A.B equals 1 if both A and B are 1.
In an AND family, both the parents must agree to grant the request of a child.
All possible AND operations between 2 variables are listed:

```
0 . 0 = 0
0 . 1 = 0
1 . 0 = 0
1 . 1 = 1
```

The AND operation is also called "logical product". In electro-mechanics, an AND may be formed by switches in series: current can flow only if both switches are closed.
In electronics, a circuit that performs an AND function is called an AND gate.

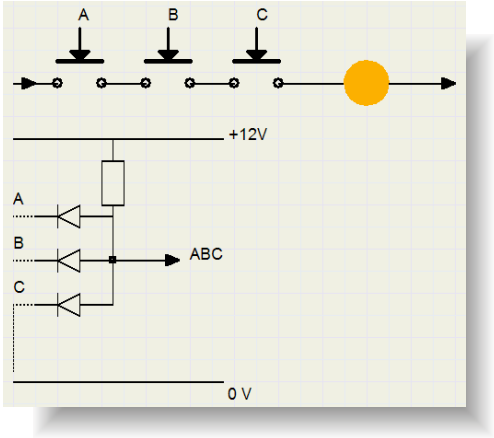


Figure 1:

In figure 1 the lamp is on when all switches A,B,C are pressed (true = 1).
In diode logic, the output voltage is high (1) when all A,B,C inputs are high.
The diodes isolate variables A,B,C.

Introduction
This article descibes some basic geometric constructions using only pencil, compasses and ruler. The ruler enables the drawing of straigth lines, the compasses are for drawing arcs and for the duplication of equal distances.

NOTE:
The scale of the ruler will not be not used.

CONTENTS

NOTE: Symbol ∟ means „angle".
Symbol Δ means „triangle".

Perpendicular bisector
This is the the most important line in plane geometry. It holds all the points that have equal distance to two other (*given*) points.
Given are points A and B, see figure 1. Construct a line perpendicular to AB that intersects AB in the center:

1. draw arcs of same radius with centers A en B. Arcs intersect in P
2. again draw arcs of same (*other*) radius with centers A en B. Arcs now intersect in Q
3. line PQ is the perpendicular bisector of AB

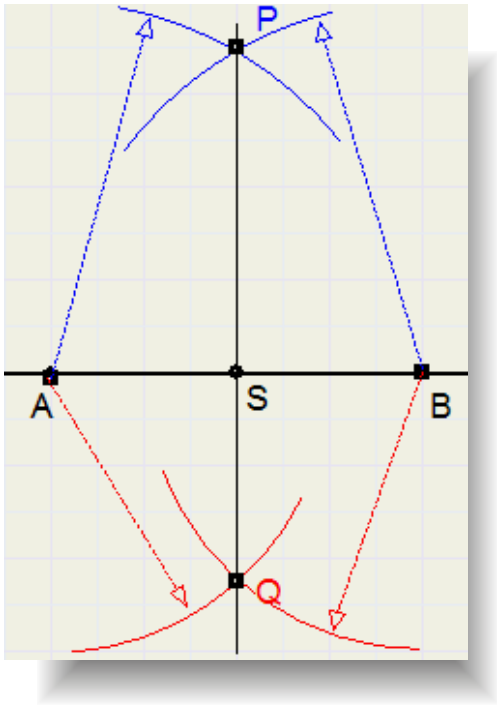


Figure 1: Line PQ is the perpendicular bisector of AB

NOTE:
S is the ***intersection*** of PQ and AB
AS = BS
AP = BP
AQ = BQ
∟PAB = ∟PBA

No code available

Introduction
This article describes the construction of a regular pentagon.
Also explained is why this construction is correct.

The construction
Please look at figure 1.
ABCDE is the pentagon.
The construction involves the following steps:

1. choose the length of line CD, the base of the pentagon
2. construct center M of CD
3. construct line perpendicular to CD and through D
4. draw N, so DN = DM
5. extend line CN
6. draw circle with center N and radius DN, P is intersection with extended line from 5.
7. extend perpendicular bisector of CD
8. draw circle with center C and radius CP, A is intersection with bisector of CD
9. draw circles with radius CD and centers A, C and D. Points B and E are other angles of pentagon.

Why is this correct?
Look at figure 2.

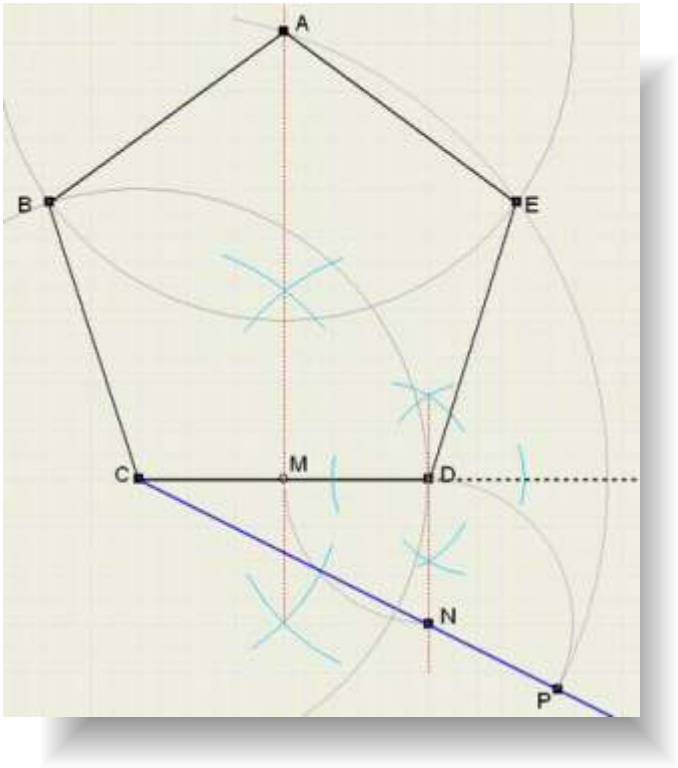


Figure 1: The construction of a regular pentagon

CHAPTER 20 - MATH

THE RANK OF A FRACTION

Theory
It is possible to assign a unique sequential number (*called rank*) to a fraction. This is the trick: if t / n is a non reducible fraction, then 2 other fractions my be derived, called the left- and the right child. See figure 1

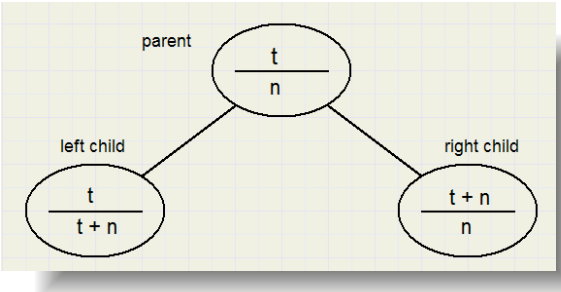


Figure 1:

Starting with 1/1 at the top, all possible fractions may be covered in this way. Important is that:

- no fraction is omitted
- no fraction will appear more than once
- fractions are irreducible (*have no common factors*)

Above rules are easy to prove, using contradiction. Also, the process is very similar to the Euclidian algorithm to calculate the GCD of two numbers. Figure 2 below shows the top part. Each fraction is assigned a bit code.

Starting with code 1 at the top, the left child adds a "0" to the code of the parent, the right child adds a "1" . To start with rank 0 for the top fraction, the rank of a fraction is simply its code - 1. In the figure 2 the bit code is listed in red, the rank in blue.

The Program

A program has been written for the calculation of the rank of a fraction and vice versa. Edit fields for the nominator, denominator and the rank allow for data entry. When one field is changed, the other are adjusted. See figure 3.

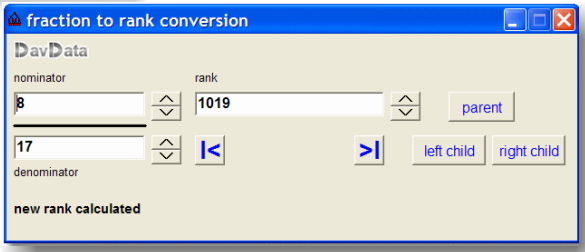


Figure 3: The Frank Program

Speedbuttons allow for sequential increment or decrement of each field. Also it is possible to navigate through the tree with the parent and child buttons.

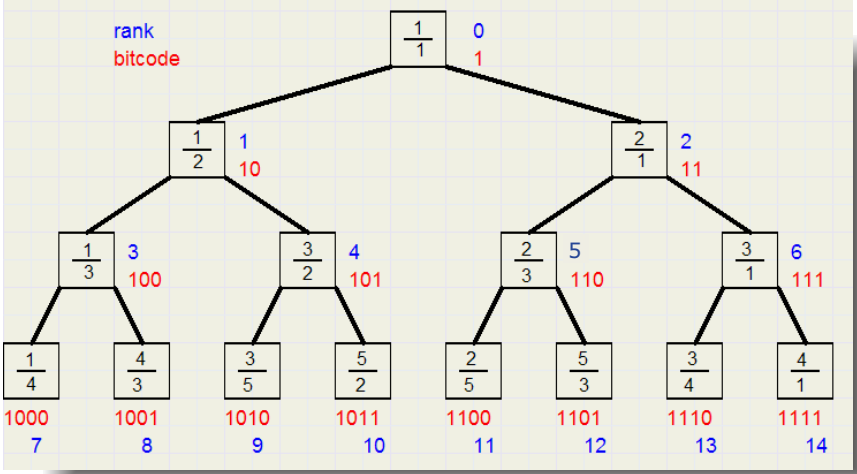
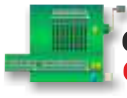


Figure 2: Each fraction is assigned a bit code.

You can download the accompanying file

frank.zip
frank.exe



Introduction
Below, you find a nice geometry puzzle.

Description

- a circle with radius r_1 and inside
- two circles with radius r_2 and r_3 which contact circle 1 at points A and B
- the centers of circles 1, 2 and 3.
- the intersection point P of circles 2 and 3 (*See figure 1*)

Prove the following theorem:

If P is on line AB then
$r_1 = r_2 + r_3$
must be true

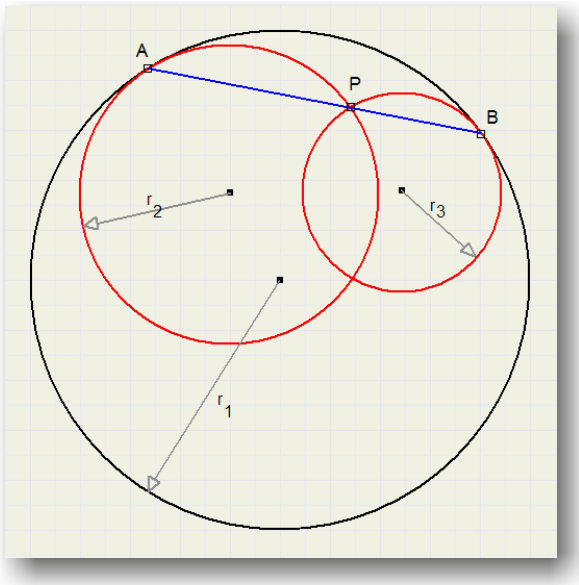


Figure 1: The theorem

Solution 1 (*Bruce*)
The problem is to prove that line APB is straight if $r_1 = r_2 + r_3$
For APB to be a straight line, we need to show that the sum of angles
APD + DPE + EPB = 180 degrees

See figure 2.

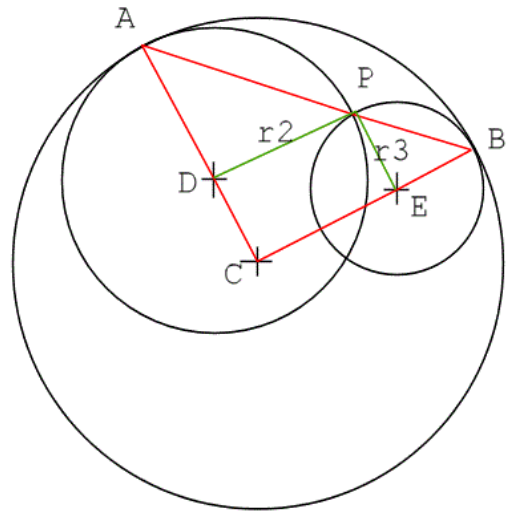


Figure 2: Bruce's solution

Notice that CD + AD = r_1
also CE + EB = r_1
But, our condition is $r_1 = r_2 + r_3$

So, this means that DPEC is a parallelogram.

So, CD = r_3
and CD = r_2

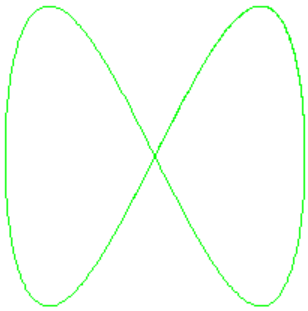
Finally, we use the parallelogram to relate angles
DAP = EPB and ADP = DPE

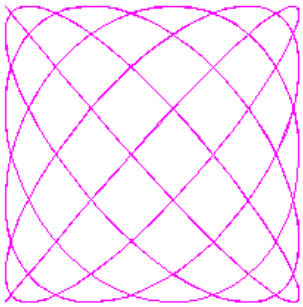
Since angles of triangle ADP sum to 180 degrees, then so do the angles on the line APB.

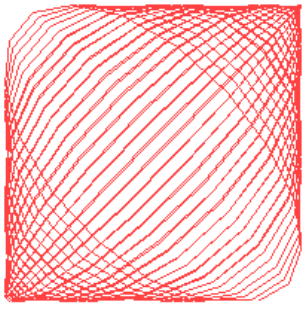
No code available

Theory.
Lissajous was a French mathematician, who lived 1822 to 1880. He is famous for his research on waves and oscillations.
A particular kind of graphs are called "Lissajous curves", 3 examples are pictured right.

Lissajous curves are made by so called "parametric functions".







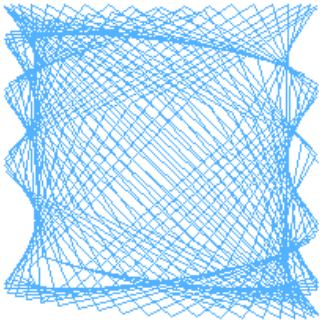
"Common" functions, like y = 5sin(x) produce a - single- y value for a value of x.
So, it is not possible to make graphs of spirals or even a circle, which needs
2 y- values for each value of x.

Parametric functions overcome this limitation by the following trick:
instead of y = f(x)
we write: y = g(v) and x = h(v)
So x and y are both functions of a new variable v.
(*Graphics-Explorer uses v, often this variable is called t*)

An example:
If y = 5sin(v), x = 5cos(v) and v has domain 0..6.28 (2*π radians), the plotted curve is a circle with centre (0,0) and a radius of 5.

In general:
a Lissajous curve is the graph of a parametric function with both x and y being trigonometric functions of v.

Molested Lissajous curves.
That's how we may call the picture right.
Lissajous curves, like common functions, will be smooth, without sharp angles.
The blue picture right is the result of the steps of v being too large.



CHAPTER 23 - MATH

LINEAR REGRESSION

Introduction
In this article I present a nice formula for the regression line though a set of points in a plane.
Given are points
(xi, yi)...where i = 1, 2, ..., n
Asked the line y = ax + b which has the smallest deviation with these

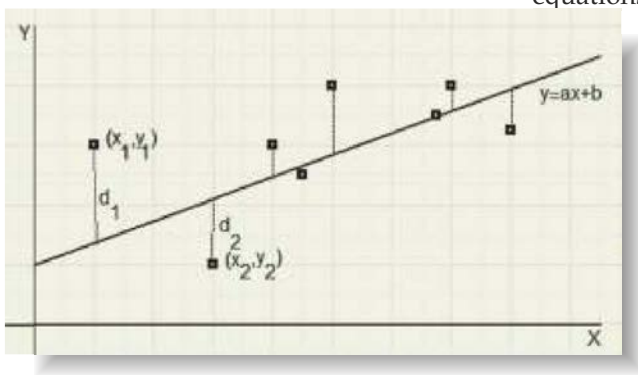


Figure 1: The regression line y = ax + b

A commom measure for the deviation is the sum of squares of the differences:
$d_1^2 + d_2^2 + \ldots + d_n^2$
in case of n points.
For point i we have :
$d_i = y_i - (ax_i + b)$

Before continuing, first some definitions and rules:

Definition

sum:
$\Sigma x_i = x_1 + x_2 + \ldots + x_n$
average:
$\bar{x} = \frac{\Sigma x_i}{n}$

Arithmetic rules

$\Sigma(x_i + y_i) = \Sigma x_i + \Sigma y_i$
if c is a constant:
$\Sigma cx_i = c\Sigma x_i$
and $\Sigma c = nc$
from the average we conclude:
$\Sigma x_i = n\bar{x}$
Application of the rules:
$\Sigma \bar{x}\,\bar{y} = n\bar{x}\,\bar{y} = \bar{y}\Sigma x_i$

The formulas for a and b of the regression line y = ax + b

Function f(a,b) of the sum of the squared deviations of points 1..n is:
$f(a,b) = \Sigma(y_i - (ax_i + b))^2$

f(a,b) is first differentiated to a, then to b.

differentiation to a:
$f'_a(a,b) = 2\Sigma(y_i - (ax_i + b)) \cdot (-x_i)$
differentiation to b:
$f'_b(a,b) = 2\Sigma(y_i - (ax_i + b)) \cdot (-1)$
For the best fit, both derivatives must be zero. This yields the following system of equations:

$\Sigma(x_i y_i - ax_i^2 - bx_i) = 0$.....(1)
$\Sigma(y_i - ax_i - b) = 0$.....(2)
from ...(.2) we see
$\Sigma y_i - a\Sigma x_i - bn = 0$
$\frac{\Sigma y_i}{n} - \frac{a\Sigma x_i}{n} - b = 0$
$b = \bar{y} - a\bar{x}$ .....(3)
substitute result for b at .....1)
$\Sigma(x_i y_i - ax_i^2 - (\bar{y} - a\bar{x})\ x_i) = 0$
$\Sigma(x_i y_i - ax_i^2 - \bar{y}x_i + a\bar{x}x_i) = 0$
$\Sigma\ x_i y_i - a\Sigma x_i^2 - \bar{y}\Sigma x_i + a\bar{x}\Sigma x_i = 0$
$\Sigma\ x_i y_i - a(\Sigma x_i^2 - \bar{x}\Sigma x_i) - \bar{y}\Sigma x_i = 0$
$a(\Sigma x_i^2 - \bar{x}\Sigma x_i) = \Sigma x_i y_i - \bar{y}\Sigma x_i$
$a = \frac{\Sigma x_i y_i - \bar{y}\Sigma x_i}{(\Sigma x_i^2 - \bar{x}\Sigma x_i)}$

Formally, we have found the formulas for a and b. Above value of a can be substituted at .....3) to find b.
However, with some manipulation the formula may be converted into a more elegant form. We separately attack numerator and denominator.

1. The numerator

$\Sigma x_i y_i - \bar{y}\Sigma x_i =$
$\Sigma(x_i y_i - \bar{y}x_i - \bar{x}\bar{y} + \bar{x}\bar{y}) =$
$\Sigma(x_i y_i - \bar{y}x_i - \bar{x}y_i + \bar{x}\bar{y}) =$
$\Sigma(x_i - \bar{x}) \cdot (y_i - \bar{y})$

2. The denominator

$\Sigma x_i^2 - \bar{x}\Sigma x_i =$
$\Sigma(x_i^2 - \bar{x}x_i) =$
$\Sigma(x_i^2 - 2\bar{x}x_i + \bar{x}x_i) =$
$\Sigma(x_i^2 - 2\bar{x}x_i + (\bar{x})^2) =$
$\Sigma(x_i - \bar{x})^2$

Summarizing:

$a = \frac{\Sigma(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\Sigma(x_i - \bar{x})^2}$ $\quad b = \bar{y} - a\bar{x}$

NOTE:
Refer to the article about the **Least Squares method** (next page) for an article about the best polynomial through a set of points. It is a nice application of linear algebra.

No code available

LINEAR REGRESSION

THE LEAST SQUARES METHOD

The relation between two numbers may be defined by
- a table
- a coordinates system
- an equation

The Least Squares method
Given are points (x_1,y_1) , (x_2,y_2)...(x_n , y_n)
Requested: a polynomial degree m:
$y = c_0 + c_1x^0 + c_2x^2 + ... + c_mx^m$
through these points having the minimal deviation.

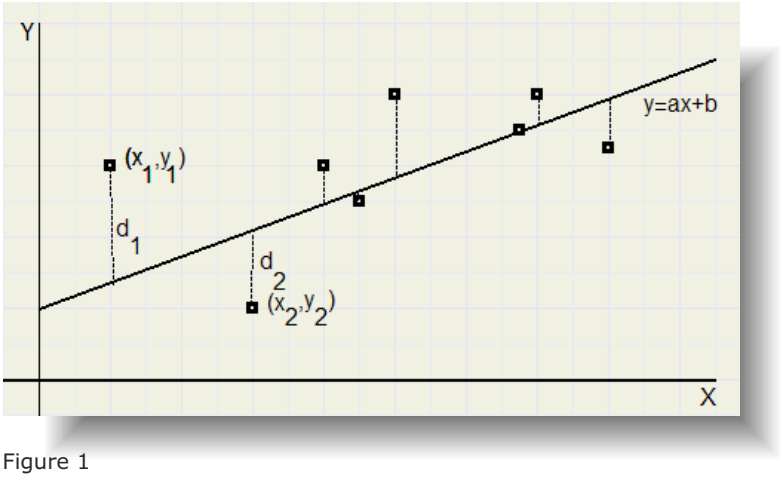


Figure 1

Each (x,y) pair we may paint as a point in a coordinate system. Then we may search for the best fitting polynomial.

NOTE: A polynomial degree n is:
$y = c_0 + c_1x + c_2x^2 + ... + c_nx^n$

Please look at figure 1 above:

Painted are points (x1,y1).....and asked is the best fitting polynomial degree 1 through these points.
"Best fitting" means, that the sum of the squared differences for each point is minimal. These differences are painted as dotted lines in figure 1.
The applied "Least Squares" method to find the best fitting polynomial is a nice application of linear algebra.

My equation grapher Graphics-Explorer (See chapter 36, page 111) uses this method, the degree may be 0 to 7.

If all points are exactly on the polynomial, so m+1 = n, we have:

$y_1 = c_0 + c_1x_1 + c_2x_1^2 + ... + c_mx_1^m$
$y_2 = c_0 + c_1x_2 + c_2x_2^2 + ... + c_mx_2^m$
.....
.....
$y_n = c_0 + c_1x_n + c_2x_n^2 + ... + c_mx_n^m$

written in matrix form:

$$\begin{pmatrix} y_1 \\ y_2 \\ ... \\ ... \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1^1 & x_1^2 & ..... & x_1^m \\ 1 & x_2^1 & x_2^2 & ..... & x_2^m \\ ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... \\ 1 & x_n^1 & x_n^2 & ..... & x_n^m \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ ... \\ ... \\ c_m \end{pmatrix}$$

or shorter: y = M . c

If the points are not exactly on the polynomial, there will be a difference vector: y - M . c

CHAPTER 25 - MATH

GEOMETRIC PROOFS OF TRIGONOMETRIC IDENTITIES

Introduction
Below, some proofs are presented of trigonometric identities. The proofs do not use any trigonometric formula or rule, what makes them quite special.

Identity Nr. 1
arctan (1/2) + arctan (1/3) = arctan (1)
Proof A.
See figure 1, pictured are 4 squares.

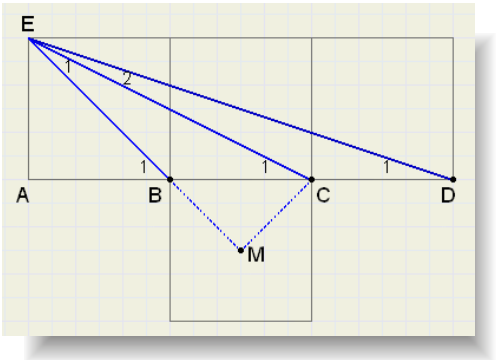


Figure 1.

Since: $\angle C1 = \arctan(1/2)$
$\angle D1 = \arctan(1/3)$
$\angle B1 = \arctan(1)$
We must prove: $\angle C1 + \angle D1 = \angle B1$ *(1)*

$\Delta EMC \sim \Delta DAC$ because:
1. $\angle A = \angle M = 90^\circ$
2. ME = 3.MC
3. AD = 3.AE

so: $\angle E_1 = \angle D_1$, which changes *(1)*
into: $\angle C1 + \angle D1 = \angle B1$

This is a basic geometric theorem, as:
$180^\circ - \angle B_1 = 180^\circ - (\angle C_1 + \angle E_1)$

This concludes proof A.

Proof B.
See figure 2, note that EF = 5, because of theorem of Pythagoras. ΔADG is the rotation of ΔABE over $90^\circ = \angle GAE$.
Now observe polygon AEFG.

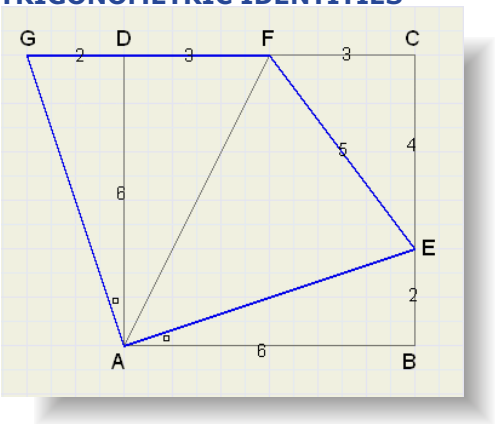


Figure 2.

FG = FE and AG = AE
so: ΔAFG and ΔAFE are congruent.
so: $\angle GAF = \angle FAE = 45^\circ$
$\angle GAD = \angle DAF = 45^\circ$
arctan (1/2) + arctan (1/3) = arctan (1)

wich concludes proof B.

Identity Nr. 2

$\sin(20^\circ) + \sin(40^\circ) = \sin(80^\circ)$

Proof.
See figure 3:

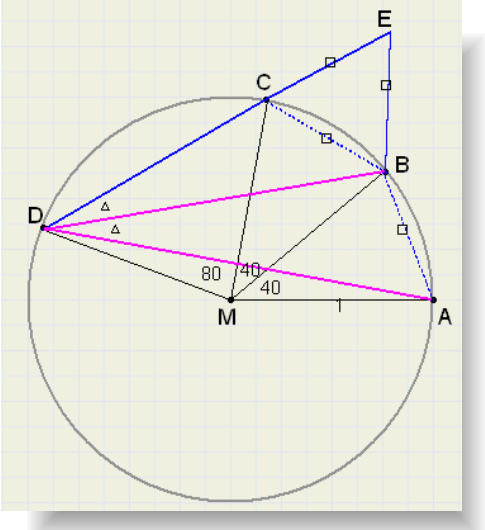


Figure 3.

Introduction
This article describes some formulas in plane geometry.
They are nice applications of algebra.
List of contents:

- the area of a triangle
- the projection formula
- Stewart's formula
- the length of the median
- the bisector formula
- the radius of the circumscribing circle
- the radius of the inscribed circle
- the radius of the escribed circle
- kissing circles
- intersecting circles

The area of a triangle

In cases where the length of the sides of a triangle are known, but not the height, a formula for the area of the triangle without this height would be of help.

Please look at figure 1:
The base, opposite angle A, has length a.
BD = x, so DC = a - x. AD = h.
Side b is opposite angle B, side c is opposite angle C.

There we go:
application of the theorem of Pythagoras in triangles ABD and ADC

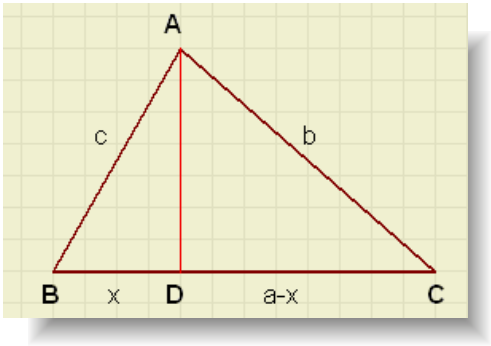


Figure 1

results in two equations:

$c^2 = h^2 + x^2$
$b^2 = h^2 + (a-x)^2$

or: $h^2 = c^2 - x^2$
$h^2 = b^2 - (a-x)^2$
so: $c^2 - x^2 = b^2 - (a-x)^2$
and
$c^2 - x^2 = b^2 - (a^2 - 2ax - x^2)$
$c^2 - x^2 = b^2 - a^2 + 2ax - x^2$
$c^2 = b^2 - a^2 + 2ax$
$2ax = a^2 - b^2 + c^2$

$$x = \frac{a^2 - b^2 + c^2}{2a}$$

This value of x, substituted in
$h^2 = c^2 - x^2 = (c - x)(c + x)$:

$$h^2 = \left(c - \frac{a^2 - b^2 + c^2}{2a}\right)\left(c + \frac{a^2 - b^2 + c^2}{2a}\right)$$

$$h^2 = \frac{2ac - a^2 + b^2 - c^2}{2a} \cdot \frac{2ac + a^2 - b^2 + c^2}{2a}$$

$$h^2 = \frac{-(a^2 - 2ac + c^2 - b^2)}{2a} \cdot \frac{a^2 + 2ac + c^2 - b^2}{2a}$$

$$h^2 = \frac{-((a-c)^2 - b^2)}{2a} \cdot \frac{(a+c)^2 - b^2}{2a}$$

$$h^2 = \frac{b^2 - (a-c)^2}{2a} \cdot \frac{(a+c)^2 - b^2}{2a}$$

$$h^2 = \frac{(b - a + c)(b + a - c)}{2a} \cdot \frac{(a + c - b)(a + c + b)}{2a}$$

$$h^2 = \frac{(-a + b + c)(a + b - c)(a - b + c)(a + b + c)}{4a^2}$$

This formula can be simplified with a trick.
If s is half of the circumference, then

$(a + b + c) = 2s$
$(-a + b + c) = 2s - 2a$
$(a + b - c) = 2s - 2c$
$(a - b + c) = 2s - 2b$

$$h^2 = \frac{2s\,(2s - 2a)(2s - 2b)(2s - 2c)}{4a^2}$$

$$h^2 = \frac{4\,s\,(s-a)(s-b)(s-c)}{a^2}$$

$$h = \frac{2}{a}\sqrt{s\,(s-a)(s-b)(s-c)}$$

CHAPTER 27 - MATH

CALCULATE PI(π) WITH PENCIL AND PAPER

Introduction

In formulas about circles and spheres we find the constant Π.
When the radius is R, then :

circumference circle	$= 2\pi R^2$
area circle	$= \pi R^2$
area sphere	$= 4\pi R^2$
volume sphere	$= (4/3)\pi R^3$

π is about equal to 3.141592654.....but no number exists that is exactly π. Therefore, in formulas we rather use π instead of say 3.14... so we can substitute later the number of digits to to achieve the accuracy we want. π can only be approximated: more computing yields a higher accuracy. This article describes one of many ways to calculate π, by using a regular polygon to approximate the circumference of an arc.

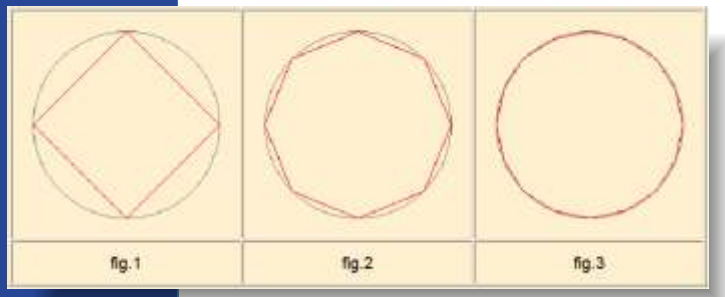
The greek mathematician Archimedes used this method at 250 bC. Using a regular 96 polygon, he found that π was a number between $\frac{22}{7}$ and $\frac{223}{71}$

In 1585, Metius calculated π in 6 digits. Vieta used a regular 393216 polygon and in 1579 found 9 digits of π. Adriaen van Roomen , in 1579 , calculated 16 digits and Ludolf van Ceulen, continuing this work, approximated (1621) π with 35 digits, using a regular 265 polygon. Thanks to fast computers and power series, today over a million digits of π are known. This knowledge has no practical application. Lambert proved in 1761 that π is an unmeasurable number. Lindemann found in 1882 , that π also is transcendental, meaning that π cannot be

This implied, that no method can exist to construct a line (*by compass and ruler*) having the same length as the circumference of a circle.

The Method

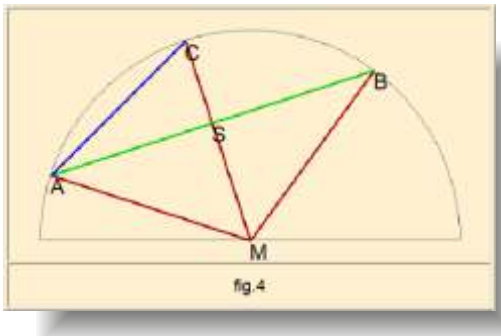
In fig.1 the circumference is approximated by a square, in fig.2 by a regular octagon and in fig.3 by a regular 16 - polygon.



Starting with a circle having a radius of 1, half the circumference is exactly π. The more angles, the better the approximation. Before we start the real work, some initial considerations.

The "half-chord" formula

A chord is a line with both ends on a circle.



In fig.4 , AB and AC are chords. MA is not. Starting with (*the length of*) AB, we calculate the length of chord AC. **NOTE**, that MA = MB = MC = 1. MC is perpendicular to AB, AS = SB, because of symmetry.
Half the circumference is exactly π.

CHAPTER 28 - MATH

Problem
Given are 3 lines with lengths a, b and c. Which conditions enable the construction of a triangle having a,b and c as sides?
Refer to figure 1. below.
The triangle has c as base. Angle C is the intersection of the circles with radii a and b, and centres A and B.

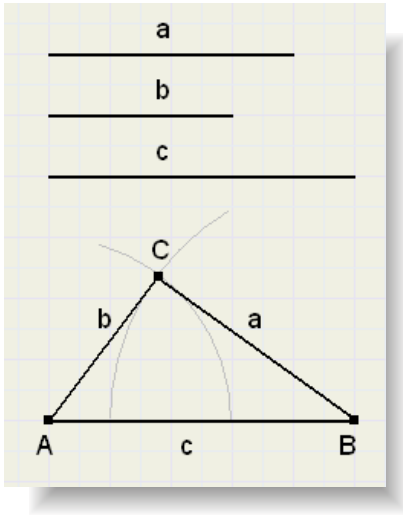


Figure 1

Now look at figure 2.

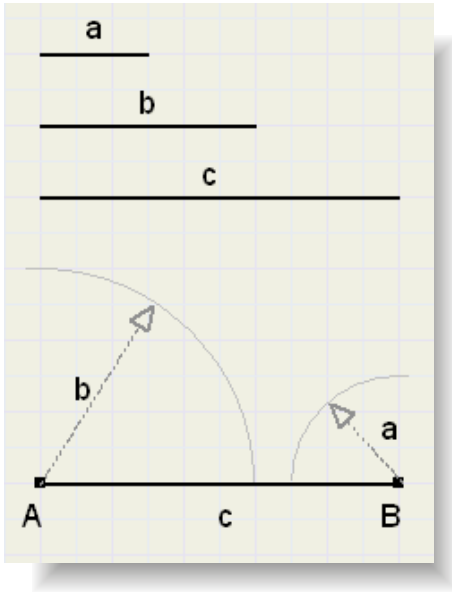


Figure 2

In fig.1, the construction of a triangle was possible. In fig.2 it is not. The reason is clear: in fig. 2 , a en b together are smaller than c. The condition therefore can be written as the inequality:

$c < a + b$ Of course, also must be valid

$a < b + c$ and $b < a + c$

The following is funny:
If we call s half the perimeter of the triangle, then:

$a + b + c = 2s$

starting with the inequality

$a < b + c$

we may write (*left and right : add a*):

$2a < a + b + c$

so: $2a < 2s$ so: $a < s$

This should be true for b and c as well, so we state:

EVERY SIDE OF A TRIANGLE MUST BE SMALLER THAN HALF OF THE PERIMETER

Another nice problem
Given is a equilateral triangle containing an arbitrary point P.
From P, we add lines

PA = a,
PB = b and
PC = c.

Prove, that a, b and c always may be sides of a triangle.
See figure 3.

CHAPTER 29 - MATH

Introduction
In this article the ancient art of calculating a square root, using pencil and paper, is rediscovered.
The reader should know the addition and multiplication tables by heart.
After the examples an explanation follows why this method is correct.

Notation: the square root of 100 is written as **SQRT(100)**.

Examples

1. We calculate SQRT(4096)

step 1:
Split the number (*right to left*) in groups of 2 digits, result 40 | 96

step 2:
Find the digit which square is closest (*below or equal*) to 40, this is the 6 , because 6 * 6 = 36.

step 3:
Subtract 36 from 40 and shift next 2 digits in place.

```
SQRT( 40  96) = 6
       36
      ---
        4  96
```

The temparary answer is 6.
The remainder is **4 96**.

Step 4:
Multiply temporary answer by 2, so 2 * 6 = 12.

Step 5:
Write 12 as 12? * ?

step 6:
Find digit ? for value closest (*below*) or equal to remainder.
This digit is 4 , because 124 * 4 = 496

step 7:
Subtract and add digit 4:

```
SQRT( 40  96) = 64
       36
      ---
        4  96
        4  96
      ---------
              0
```

SQRT(4086) = 64, because 64 *64 = 4096

2. Calculating SQRT(1522756)

- Split number in groups of 2 digits, add 0 when number of digits is odd:
  01 | 52 | 27 | 56
- Find first square = 1

```
SQRT( 01  52  27  56) = 1
        1
       ---
        0  52
```

- Multiply by 2: 2 * 1 = 2
- Write 2? * ?
- ? = 2
- 22 * 2 = 44

```
SQRT( 01  52  27  56) = 12
        1
       ---
        0  52
           44
           ---
           8  27
```

- Multiply temporary answer by 2: 2 * 12 = 24
- Write 24? * ?
- ? = 3, because 243 * 3 = 729

```
SQRT( 01  52  27  56) = 123
        1
       ---
        0  52
           44
           ---
           8  27
           7  29
           -------
                98  56
```

Introduction
This article describes how my program "Euclid" solves x,y in the equation Ax + By = C, for integers A,B,C,x,y. These type of equations arise from puzzles like this:

> A collector of vintage cars leaves his 4 children the following will:
>
> **John**: 40% of the cars is for you. Of the remaining 60%, grant 5 of your choice to a museum.
> **Helen**: 40% of the remaining cars is for you. After your choice, grant 3 of the remaining cars to a museum.
> **Emily**: 40% of the remaining cars is for you. After your choice, grant 1 car to a museum.
> **Roderick**: the remaining cars are yours.

Question: how many cars should this collection count minimally, to make such a distribution possible?
Say, the collection counts x vehicles, then:

0.6(0.6(0.6x - 5) - 3) - 1

should be a positive integer.
If Roderick receives y cars, then:

0.216x - 4.6 = y
216x - 1000y = 4600

So, positive integers x and y must be found, that satisfy the above equation.

The core of solving such equations is the use of the Euclidean Algorithm.
So, first this algorithm is discussed.
After that, we will see how the algorithm is applied to obtain the desired results.

The Euclidean Algorithm
A somewhat free interpretation of the Euclidean algorithm is:

gcd(A,B) = gcd(A - B,B)

where gcd means "greatest common divisor"
In words:

instead of calculating gcd(56,35),
we may calculate gcd(21,35)

gcd(56,35) = gcd(21,35) = gcd(35,21) = gcd(14,21) = gcd(21,14) = gcd(7,14) = gcd(14,7) = gcd(7,7) = 7.

A proof can be constructed of 2 sub-proofs:

1. if d is a factor of A and B, then
   d is a factor of A - B
   {no common factor is lost}
2. if d is a factor of only A or B,
   then d cannot be a factor of A - B
   {no new common factor is introduced}

proof 1.
assume:

A = ad and B = bd then:
A - B = ad - bd = d(a - b) and
(A - B)/d = a - b which is an integer.
So (A - B) must have a factor d.

proof 2.
given:

A does not have a factor d.
B has a factor d and B = bd
we must proof:
gcd(A-B,B) = d is impossible

assume:

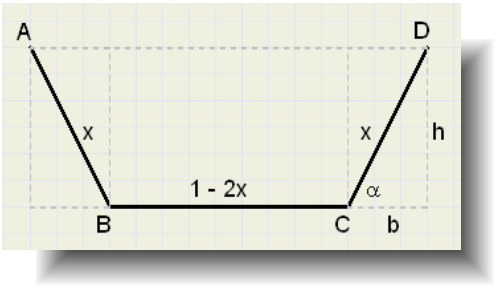
A - B has a factor d, then
A - B = A - bd = kd, where k is some integer. Dividing by d :
A/d - b = k so: A/d = k + b

b and k are integers, A/d is a fraction,
so this is impossible.
A - B can never have a factor d.

This concludes the proof.

Introduction
From the Netherlands, land reclaimed from sea, with its canals, locks, dikes, ditches and windmills, a search for the Ultimate Gutter Dimensions..

See section below:



In this trapezium the total length AB + BC + CD = 1.
Variables are angle α and length x.
Question: for which values of α and x, will area ABCD have a maximum?
Two solutions are presented:
1. graphical (*using Graphics-Explorer*)
2. analytical (*application of calculus*)

A formula for the area ABCD
See picture above.
The area A is the addition of a rectangle (1-2x)h and two triangles, together bh, so:

```
A = (1 - 2x)h + bh
```

NOTE:

```
h = x.sin(α), b = x.cos(α)
```

so:

```
A=x².sin(α).cos(α)+(1-x).x.sin(α)
A=x².sin(α).cos(α)+x.sin(α)-
                        2x².sin(α)
```

The Graphical solution
Graphics-Explorer (See chapter 36 page 111) uses variables x en y.
Also, constants (a,b,c) may be used in formulas. These constants are changeable by mouseclick, graphics adjust to new values. It's obvious to use x for a side, y for the area and a for the angle α.

Luctor et Emergo
(Pump or Drown)

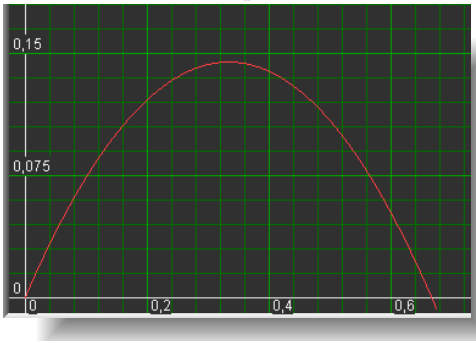
Type the formula:

```
y= x^2*sin(a)*cos(a)+ x*sin(a)-
                      2x^2*sin(a)
```

and change the following settings of Graphics-Explorer:

- angles: degrees instead of radians
- coordinates (0,0) left-bottom
- +/- value for constants increment: 5 (degrees)
- zoom-center at (0,0)
- x- scale (1) at right side (x < 1)
- y- scale (0.2) at top
- "Autoplot" (*to make graphics adjust on constants change*)
- "replace" (*to enable cleanup of old graphics*)

Plot the formula and change value of a. Observe maximum value of y and read values of x en a, see picture below:



CHAPTER 32 PASCAL PROGRAMMING

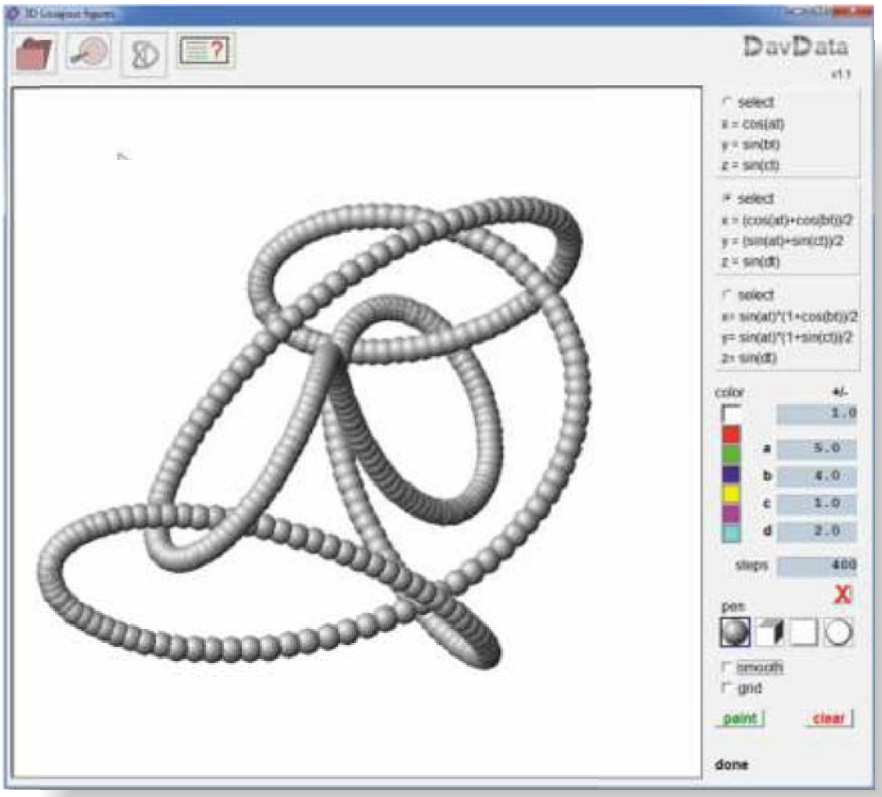
COMPUTER ART 3D LISSAJOUS GRAPHICS

Introduction

In this chapter we will produce computer art by means of 3 dimensional Lissajous figures. Lissajous (1822..1880) was a French mathematician, famous for his research on waves. The lissajous3d program allows the plotting of 3 dimensional Lissajous graphics. Below is a reduced picture of this (Windows) program at work:

Selections are made by mouseclicks. For the constants a,b,c,d :
- a left mouseclick adds the (+/-) value
- a right click subtracts the (+/-) value.

The (+/-) value itself may be changed also by left- or right mouseclicks.



PROGRAM FEATURES

- choice of 3 sets of Lissajous formulas
- choice of 4 pen styles:
  - sphere
  - cube
  - square
  - circle
- choice out of 7 colors
- single dots or connected (smooth) lines
- step count from 100..1000
- save settings to disc (*.l3d extension attached)
- load settings from disc
- save picture (*.bmp)

Interim variable t counts from 0 to the selected stepcount.
Stepcount is selectable between 100 and 1000 in steps of 50.

The menu and buttons are self explanatory.

CHAPTER 33 - FREEWARE

Introduction
Logic10 is a program that:
1. Generates Truth Tables from formulas in Boolean Algebra
2. Reduces Truth Tables by applying the rules of Boolean Algebra

Contents

LOGIC10: TRUTH TABLE REDUCER DELPHI

Logic10 is a great help in the study of Boolean Algebra, the design of digital electronic circuitry and proposition logic in general.
This is Logic10 version 2.0 (figure 1)
It supersedes all previous versions.

Features
Logic10 features are:
- **input:** (Proposition Logic) formula or Truth Table (CNF format)
- **output:** Truth Table in **CNF** or **DNF** format
- **save and reload** of all input and output data
- **printing** of input data and output results
- **selectable:** information of the formula translation process
- **selectable:** step by step information of the reduction process
- **In-Line help information**

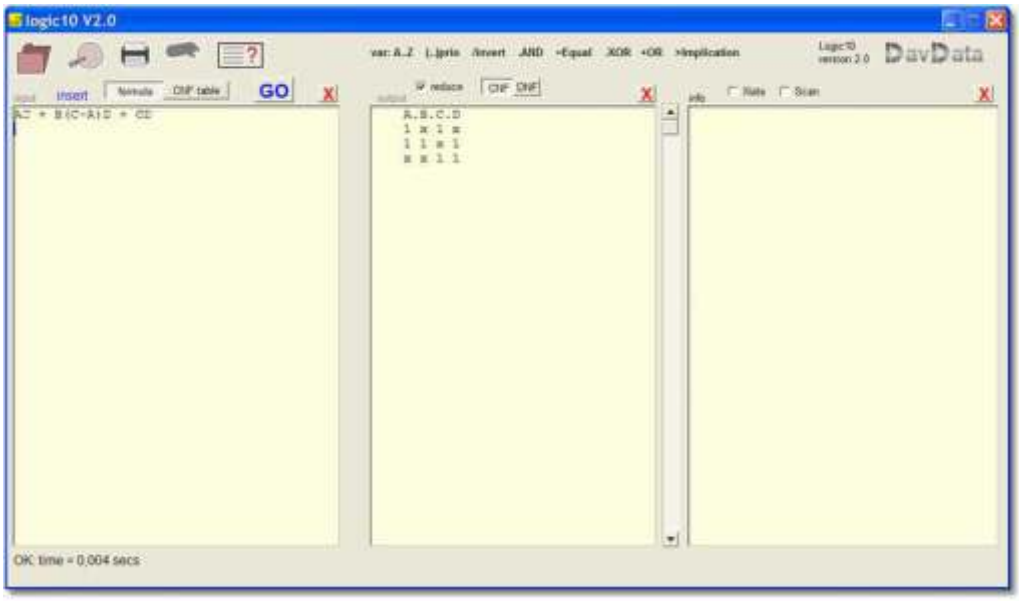


Figure 1: logic10 at work

CHAPTER 34 - FREEWARE

A PRIME NUMBER GENERATOR

INTRODUCTION
Prime numbers are numbers that are only divisible by 1 or by themself. The first 10 prime numbers are:
2, 3, 5, 7, 11, 13, 17, 19, 23, 29.
A number can be factorized only in one unique way, so prime numbers may be considered the building blocks of numbers. Prime numbers play an important role in mathematics, specially when coding secret messages as is the case in electronic banking. This article describes how the program "prime200" calculates prime numbers.

THEORY
The number investigated to be prime should be divided by all smaller primes. The quotient may not be an integer, because this means that the number is a multiple of this other prime. We also could say: the remainder of the integer division may not be zero. In the Delphi programming language, the operator mod calculates the remainder of an integer divide.

The fomula `P mod d` yields the remainder of the division of P by d.
If this result equals zero, P is not a prime number. We research the number 229 to be prime. The sequential divisors may be `d = 2, 3, 4, 5.....228`
and when all remainders are unequal to zero we know that P is prime. However, the job may be accomplished with less work. Prime numbers bigger than 2 cannot be even, because these numbers are multiples of 2. So starting from 3 we may increase P and its divisors by 2. Divisor d then has the values : `3, 5, 7, 9, .....` But looking more closely at numbers above 5 we observe:
`...7, 11, 13, 17, 19, 23, 29, 31,...`
an increment by 2 never happens twice in a row.

A PRIME NUMBER GENERATOR
Reason is that prime numbers above 6 have the form 6K+1 or 6K+5, where K = 1,2,3,4..... Please look:

- 6K may be divided by 6
- 6K + 2 may be divided by 2
- 6K + 3 may be divided by 3
- 6K + 4 may be divided by 2

so the list if divisors becomes (*for prime numbers above 6 and with the form 6K+1 or 6K+5*):
`..5, 7, 11, 13, 17, 19, 23, 25, 29..`
alternating addition of 2 or 4.

But another accelleration is possible. Computers distinguish between real numbers and integers. The operator for division of real numbers is '**/**'.
For integer divides, the (*Delphi language*) operator is **`div`**. These division neglects the digits right of the decimal point. We continue investigation of number 229:

1) .....`229 div 13 = 17`.....
   and the next divisor is 17
2) .....`229 div 17 = 13`

In line 1) the quotient is bigger than the divisor (*17 > 13*).
In line 2) the quotient is smaller than the divisor.
This means that it is meaningless to increase the divisor again and keep dividing 229, because if 229 had a (*prime*) factor we had met the factor already.

NOTE, that if `P = a.b` where a and b are prime, then

```
P mod a = 0
P mod b = 0
P div a = b
P div b = a.....this division is
                superfluous if a < b
```

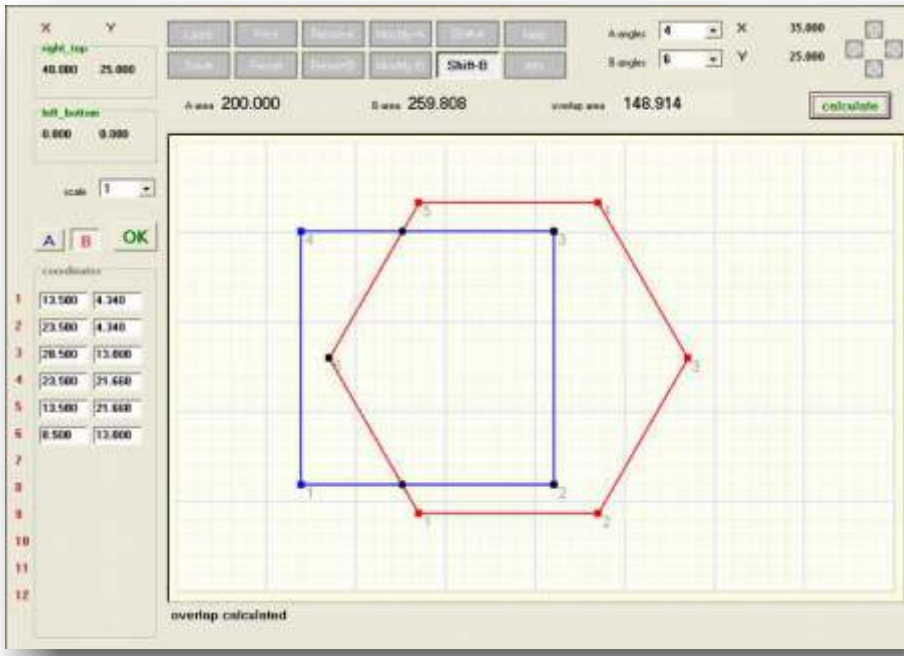
CHAPTER 35 - FREEWARE

POLYGON OVERLAP CALCULATOR

Introduction
The Polygon-Overlap-Calculator program calculates the area of two (*convex*) polygons and their overlap. Convex means: any angle of the polygon is less than 180 degrees. The number of angles may vary from 3 to 12. Below is a reduced image of the program at work:

- In-Line help
- no installation procedure, simply copy **"overlap.exe"** to a map of choice.

Freeware
The Polygon-Overlap-Calculator is freeware and may be used, copied and distributed without restriction.



The options of the Polygon-Overlap-Calculator are:

- number of angles selectable from 3 to 12
- calculate area of polygons and also their overlap
- shape of polygon changeable by mouse or by keyboard
- polygons can be shifted (*by mouse*) in the coordinate system
- changeable scale and position of coordinate system
- images can be copied to clipboard or file
- save coordinates of polygons, together with settings, to file
- open files to load previously saved polygons and settings
- print polygons, together with coordinates information

Application
The Polygon-Overlap-Calculator may, besides calculation of areas, be used to match geographic images such as maps and aerial photographs with earth coordinates.

The Polygon-Overlap-Calculator **overlap_e.exe** is written in Delphi-7. Size is about 330kB.
d7_overlap.zip: the source listing of the overlap unit with all data formats and procedures, source listing explanation and a description of the algorithm.

You can download the accompanying file overlap.exe overlap.zip

Introduction
Graphics Explorer is a versatile and powerful program to plot, print or investigate graphs, equations and functions. Besides the graphing of different type of functions, Graphics Explorer also is able to find the best fitting polynomial or exponential function given a set of points.

Specifications

- coordinate system 600 * 450 pixels
- crosshairs with (x,y) indication
- 9 different colors and formulas
- scrolling and zooming with mouseclick, x/y independent
- add or delete dots with mouseclick or by table editing
- recognizes 4 types of equations:

Equations may contain constants (a,b,c) that can be changed by a mouseclick.
Graphs adjust instantly which illustrates the role of constants as in
`y = a*sin(b*x + c),`
where a,b,c are amplitude, frequency and shift of phase.

	type	example	description
1	y = f(x)	y = 3cos(5x)	function
2	x = f(y)	x = (y-1)^2	inverse
3	y = f(v) ; x = g(v)	y = 5sin(2v) ; x = 7cos(3v)	parametric
4	f(x) = g(x)	(x^2+y^2)(y^2+x(x+10))=40x*y^2	implicit

table 1: 4 types of equations

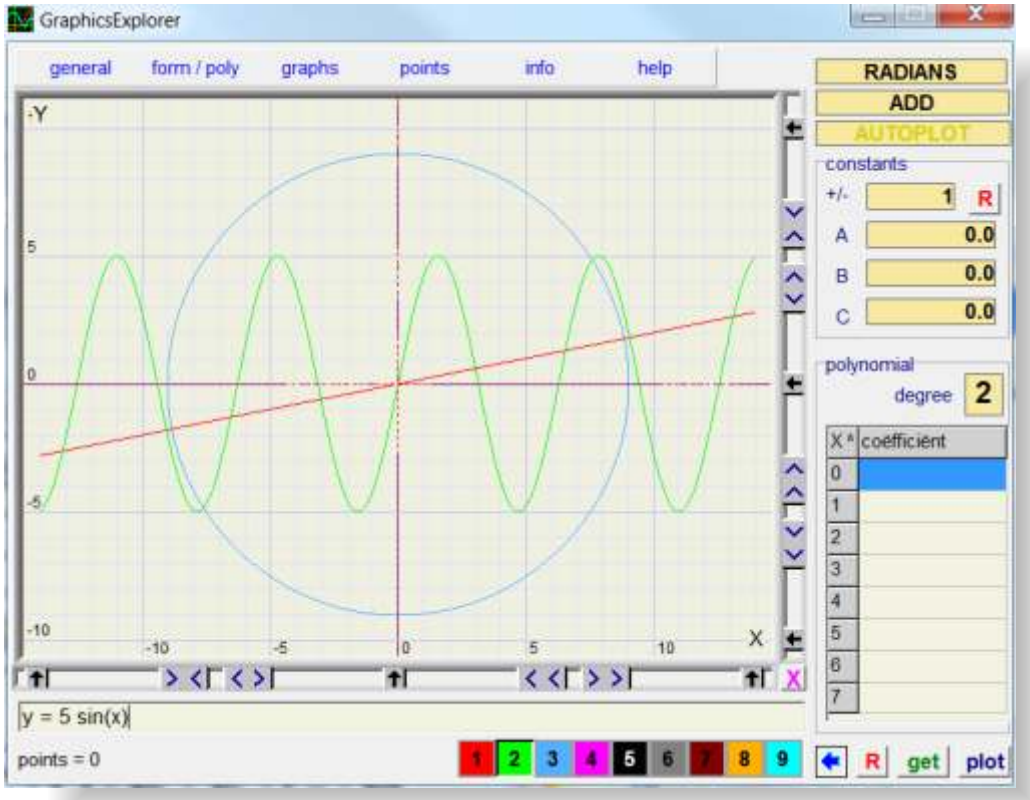


Figure 1: GraphicsExplorer 3 plotted:
red: y = 0.2x **green** y = 5sin(x) **Blue** x^2 + y^2 = 81

The program **Euclid.exe** at work:

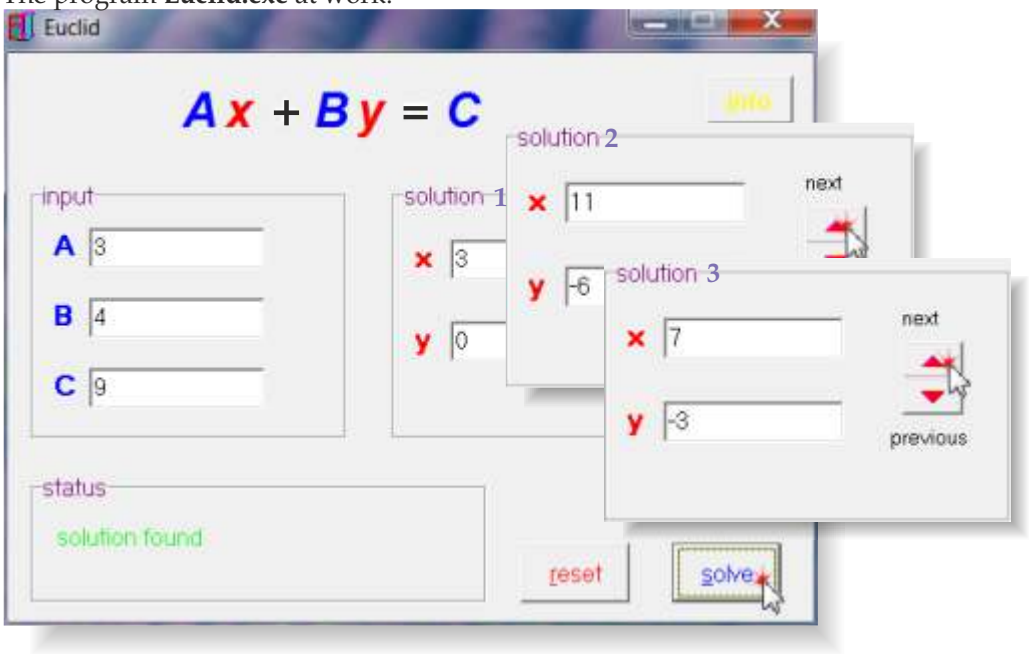


Figure 1: Euclid with a=3, b=4, c=9 and three posible solutions

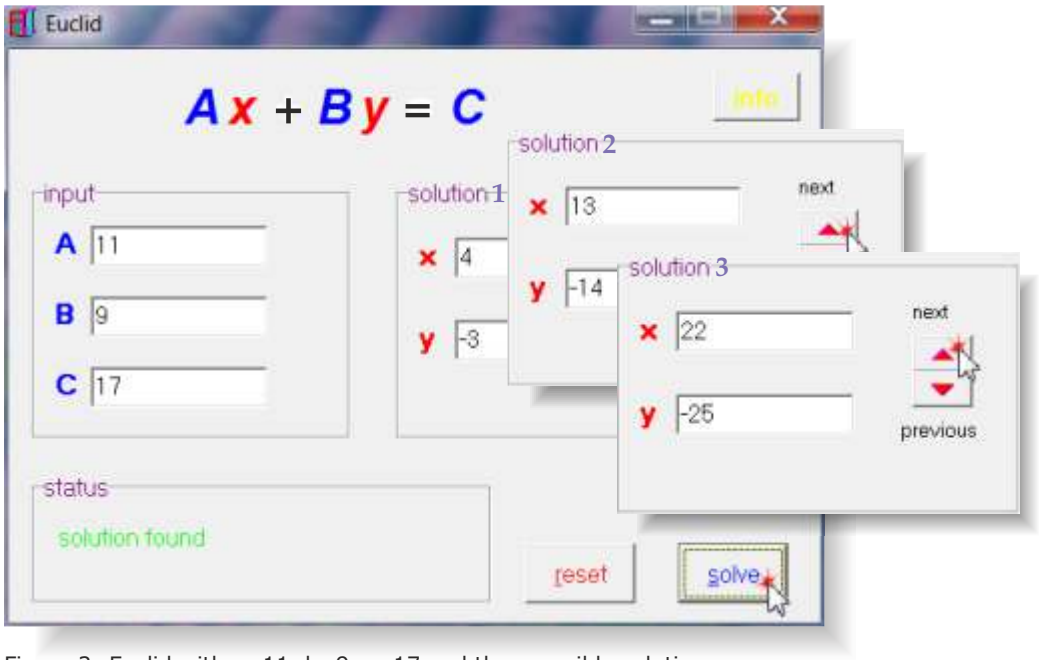


Figure 2: Euclid with a=11, b=9, c=17 and three posible solutions

You can download the accompanying files: euclid.exe
The complete project and code is available for download euclid.zip

CHAPTER 38 - FREEWARE

DISPLAY CHARACTER CODES

Introduction
This freeware program displays the character codes of a particular font. This is helpful when the code is needed of a character that is not represented by the keyboard. Below is a snapshot of the program at work.

Installation
The program **chrcode.exe** runs on all windows platforms.
There is no installation procedure.
Simply copy the **.exe** file to a directory of choice.

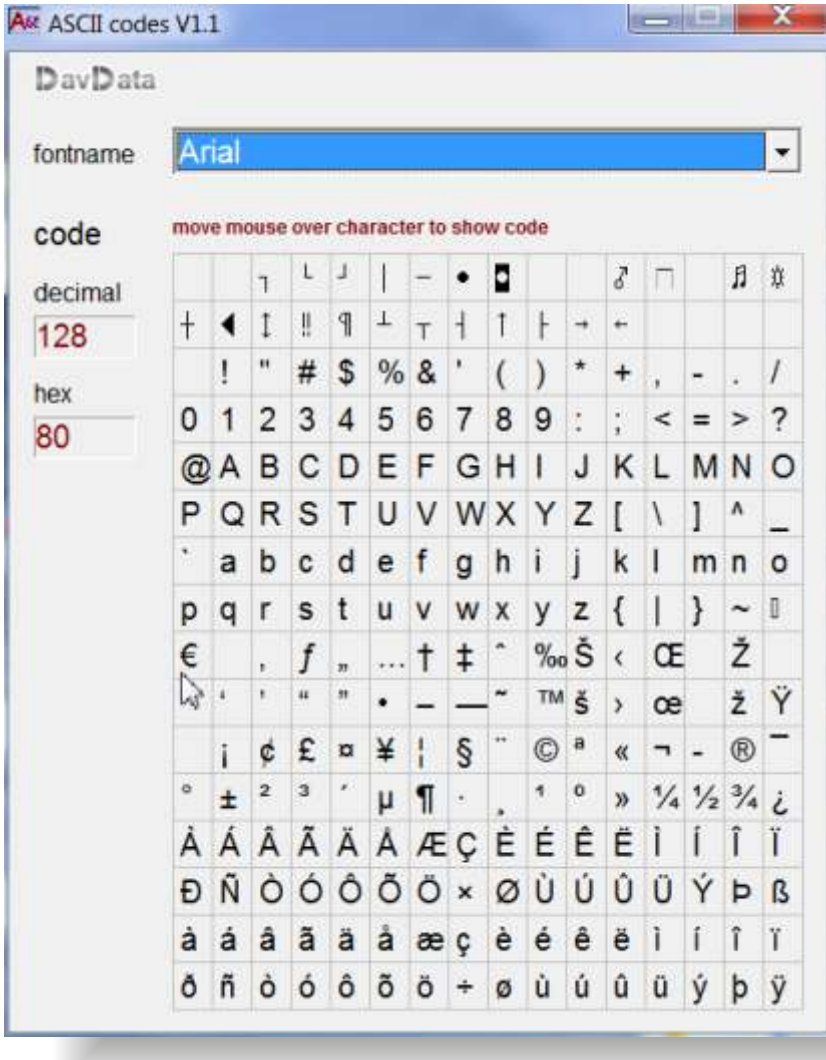


Figure 1: The program chrcode at work

You can download the accompanying files: chrcode.exe

The complete project and code is available for download chrcode.zip

Introduction
This program factorizes numbers and calculates the GCD

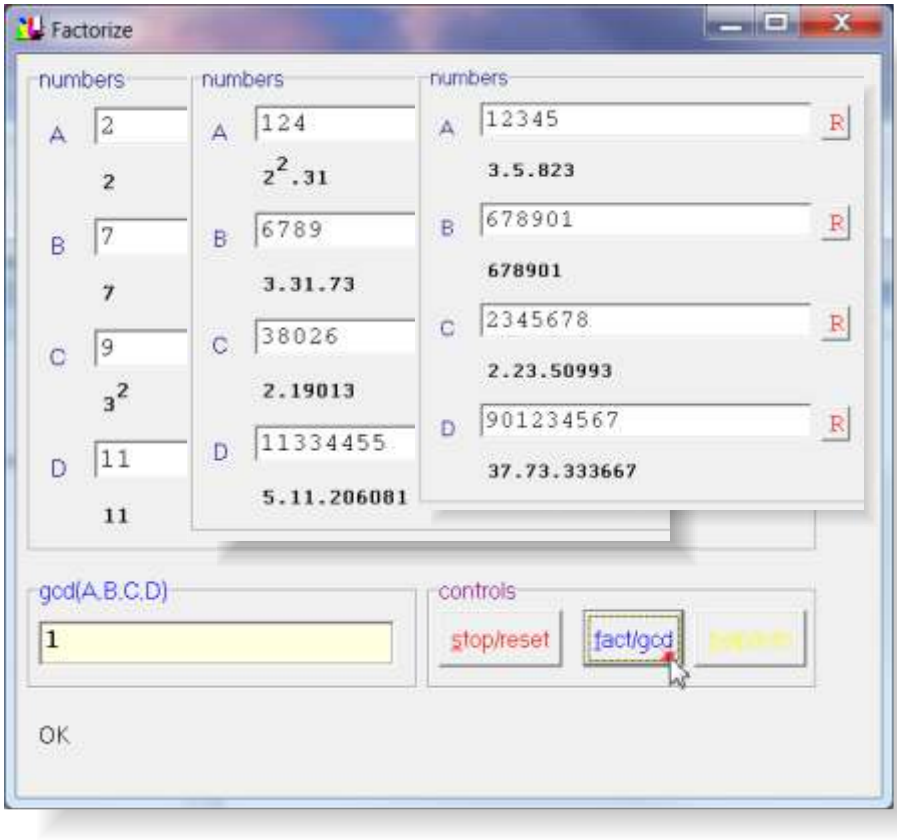


Figure 1: Factors.exe with three examples (two extra inside)

You can download the accompanying files: factors.exe
The complete project and code is available for download factors.zip

Introduction

Fonttest
This program shows how Windows renders a character on the screen. Individual pixels are enlarged 10 times, to show the details.
Below is a picture of the program

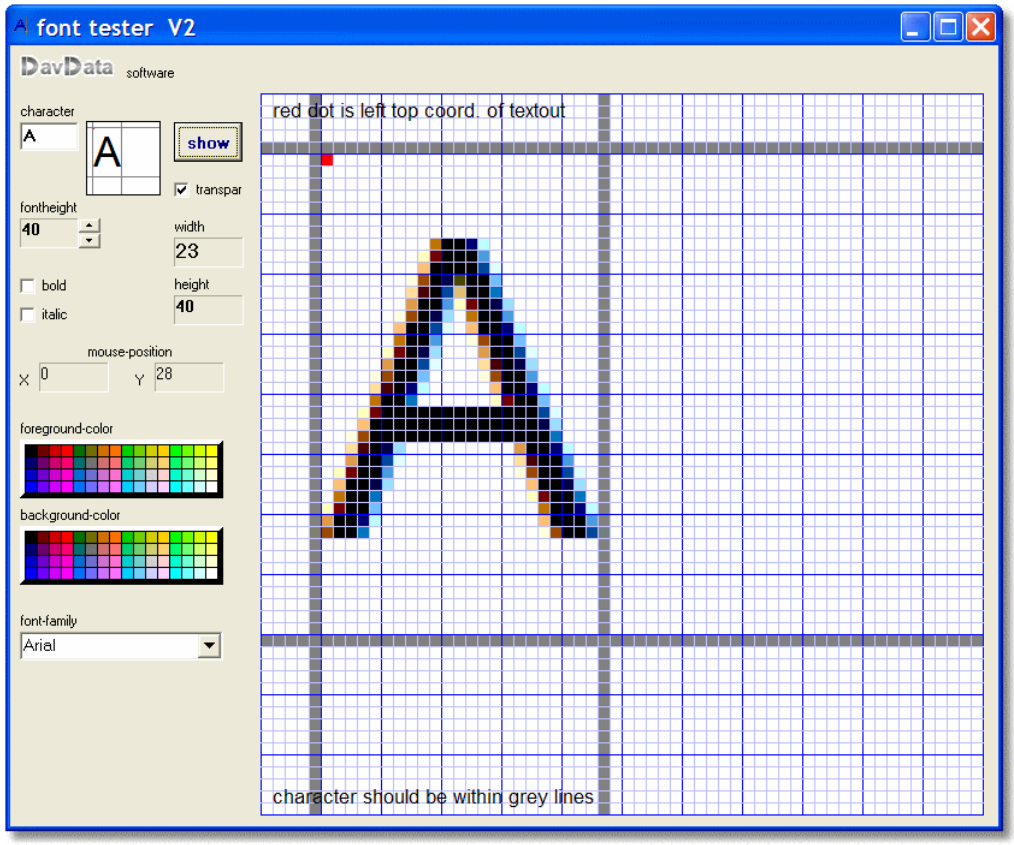


Figure 1

The red dot (*see figure 1*) has the (x,y) coordinates of the statement ....

```
canvas.textout(x,y,.....)
```

The transparent checkbox is equivalent to the statement ....

```
canvas.brush.style := bsClear
```

Interesting (*or shocking*) may be the next picture (*see figure 2*), where italic is selected with transparent off.
The boundaries of the character as obtained with the statements

```
...canvas.textheight("f")
...canvas.textwidth("f")
```

are violated severily.

You can download the accompanying files: fonttest.exe/keystroke.exe
The complete project and code is available for download fonttest.zip

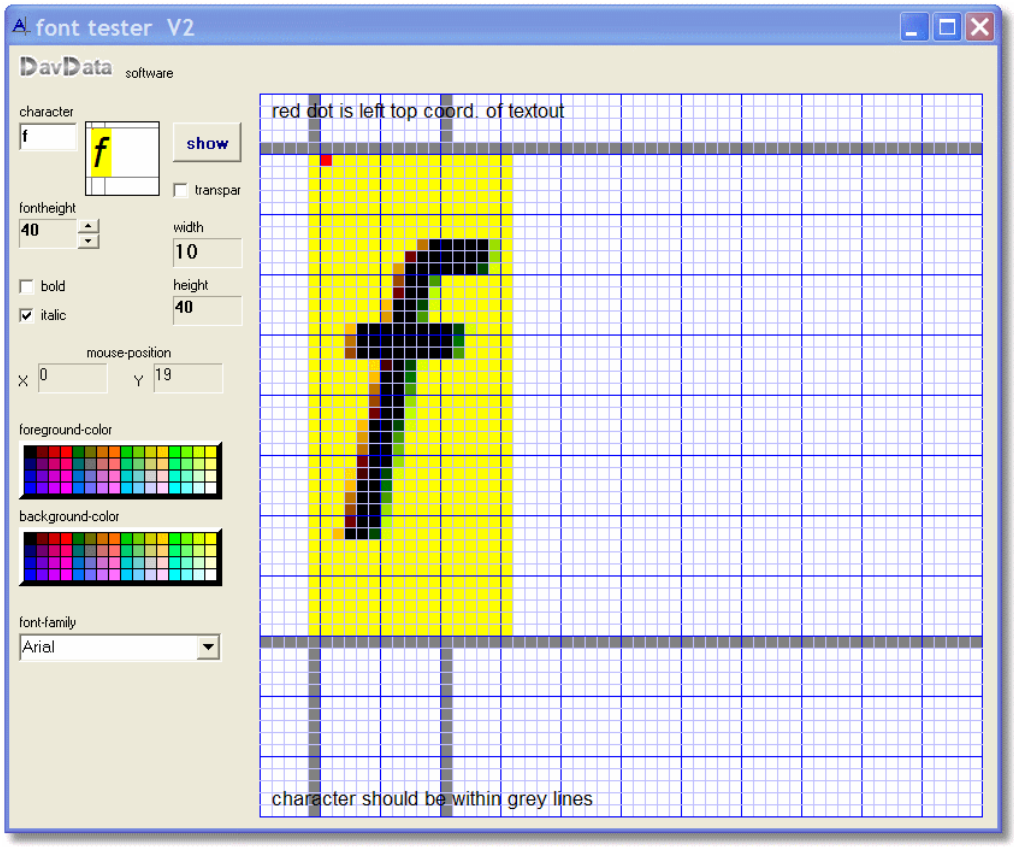


Figure 2

Keystroke

This program shows the code generated when a combination of keys is pressed. The keydown event generates a (16 bit) word when a key is pressed. The keyup event generates a word when a key is released. The keypress event generates an 8 bit character code when a key is pressed. Below is a picture of the program at work with the „?“ key.

You can download the accompanying files keystroke.exe The complete project and code is available for download keystroke.zip

Introduction
This freeware Delphi program solves systems of linear equations from 2 * 2 to 9 * 9 rows and columns. It uses the "Gauss-Jordan" elimination.
To illustrate this method, select STEP mode to watch the process step by step.
Below is a snapshot of the program:

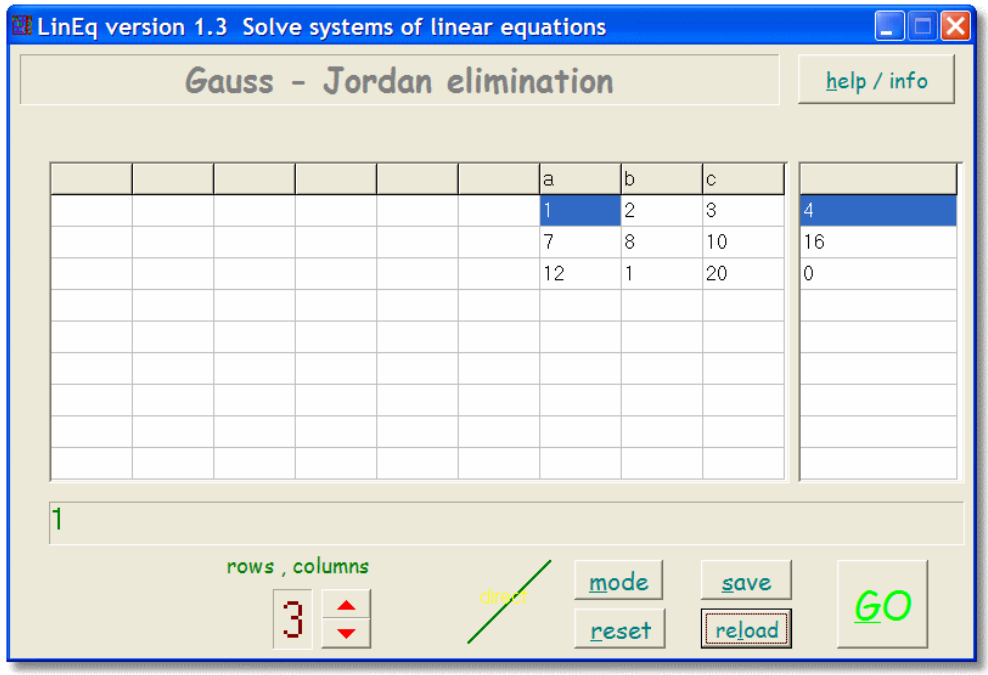


Figure 1

Installation
The program runs on all Windows versions.

Features
LinEq solves systems of linear equations from 2 equations and 2 unknowns to 9 equations and 9 unkwowns.

MODE
Click mode button to switch STEP mode on and off. In step mode, the program halts after each step that sweeps the columns.

SAVE
Click Save button to save data.

LOAD
Click Load button to reload previously saved data.

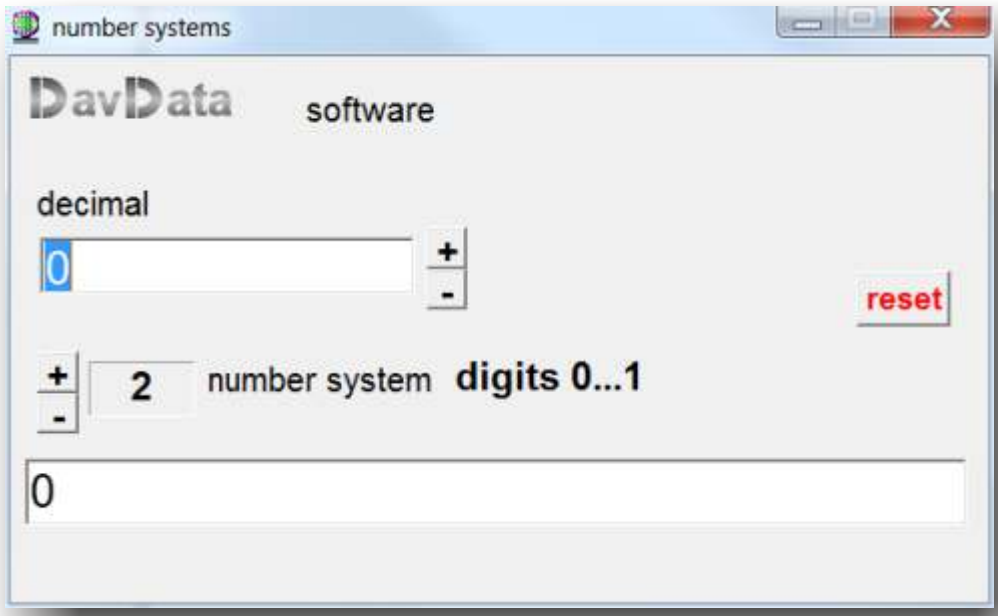
HELP
Click Help-Info button to show in-line help information.

THE PROJECT
LinEq was written many years ago in the Delphi-3 programming language.
It was one of my first efforts.

You can download the accompanying files: lineq.exe
The complete project and code is available for download lineq.zip

Introduction

This program converts between number systems 2..16

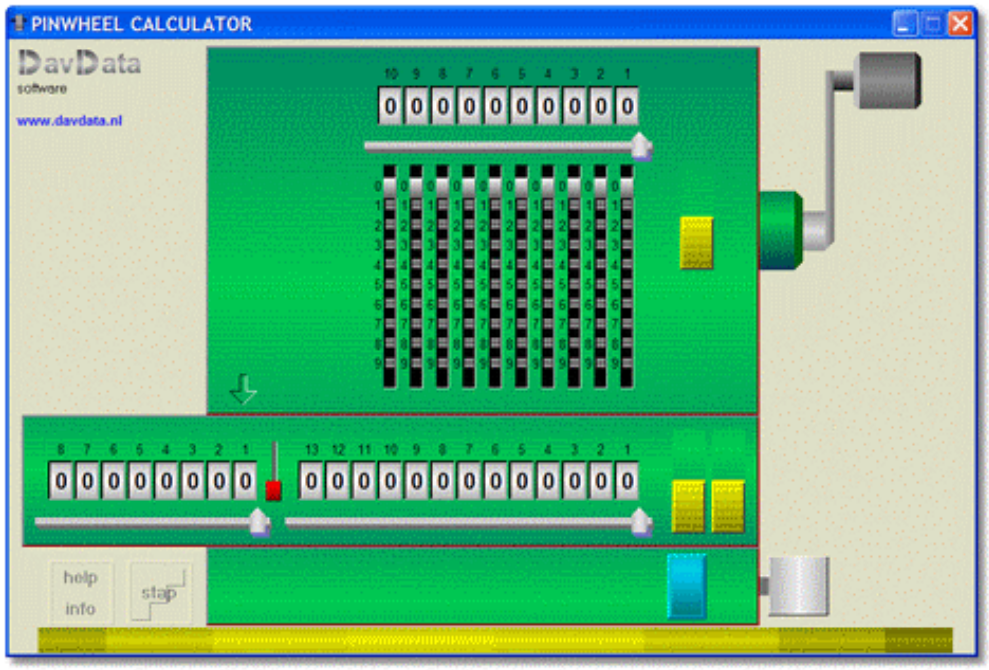


Numbers converts between decimal and binary(2) to hexadecimal (16) number systems.
The program is self - explanatory.

You can download the accompanying files: numbers.exe

The complete project and code is available for download numbers.zip

Introduction
Pinwheel is a program which simulates an old fashioned mechanical calculator. Below you see a half size image:



This type of calculators was invented around 1850 and they were in use until about 1970. Electronic calculators already existed at that time, but they were very expensive.

The mechanical calculator is build around a rotating drum with pins that can shift in or out. As the drum turns, an outward pin advances a gearwheel one step.
A gearwheel position shows a digit.
Pinwheel calculators may be found at antique shops, but they are becoming expensive and in most cases they are defective.

If you search the web with phrases like "mechanical", "calculator", "pinwheel" you will discover private collections or museums showing beautiful machines.

A very good one is www.calculi.nl with a big collection of vintage calculator machines.
Using a Pinwheel calculator you can add, subtract, multiply or divide numbers with or without decimal points.

My simulator features:
- control by mouse or keyboard
- In-Line help information and examples
- optional "step by step" operation, showing carry or borrow propagation

Pinwheel is written for Windows.
Programming language is Delphi-7.
There is no installation procedure.

You can download the accompanying files: pinwheel.exe
The complete project and code is available for download pinwheel.zip

RANKS

Introduction
Ranks is a program that assigns a sequential number (rank) to a combination, permutation or partition. Given a rank, the combination permutation or partition may be generated. Given a combination permutation or partition, the rank may be calculated.

Installation

Ranks ships as a single .exe file.
It contains In-Line help information.
The system is Windows 95 and up.
Minimum screen resolution is 600*800.

There is no installation procedure, just copy to a directory of choice.
The Windows Registry is not changed.
The size is 268kB.

Combinations

A combination is a selection of elements from a set, where each element may be chosen once and the sequence of selection is unimportant.
In this program (ranks) the elements are always the natural numbers 1,2,3,4,.....
The maximum number of elements is 50.
The ranking of combinations may be useful in the analysis of lotto games or any case where combinations have to be generated in a systematical way.

The number of possible combinations of k elements from a set of n is written as

```
C(n,k) = n!/(k!*(n-k)!)
```

Next, all combinations of 2 elements from a set of 4 are listed together with the rank of that combination.

rank	combination	rank	combination
0	1-2	3	2-3
1	1-3	4	2-4
2	1-4	5	3-4

There are 6 possible combinations, the ranks are 0 to 5.

Permutations

A permutation is a sequence of elements.
Elements are always named 1,2,3,4,....
The maximum number of elements is 12.
n elements have n! permutations, where

```
n! = n.(n-1).(n-2) ...(3).(2).(1)
```

Ranking a permutation is useful when sequences have to be generated in a systematical way, as is the case in logigram puzzle solving. Below are listed all permutations of a set of 4 elements:

rank	comb.	rank	comb.	rank	comb.
0	1-2-3-4	8	2-3-1-4	16	3-4-1-2
1	1-2-4-3	9	2-3-4-1	17	3-4-2-1
2	1-3-2-4	10	2-4-1-3	18	4-1-2-3
3	1-3-4-2	11	2-4-3-1	19	4-1-3-2
4	1-4-2-3	12	3-1-2-4	20	4-2-1-3
5	1-4-3-2	13	3-1-4-2	21	4-2-3-1
6	2-1-3-4	14	3-2-1-4	22	4-3-1-2
7	2-1-4-3	15	3-2-4-1	23	4-3-2-1

So, there are 24 permutations, ranked 0 to 23.

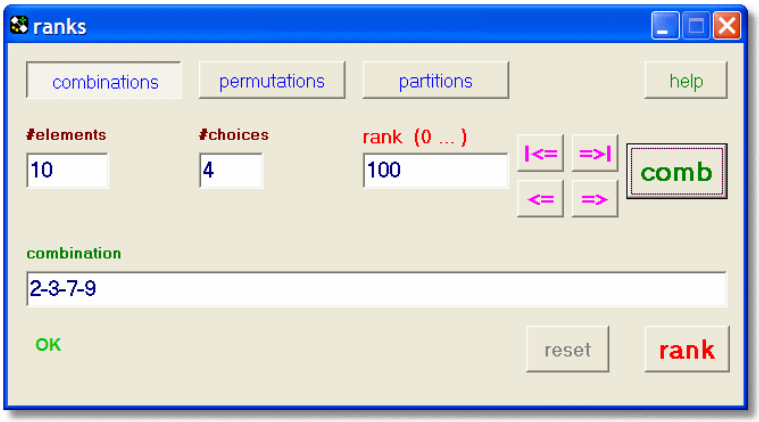


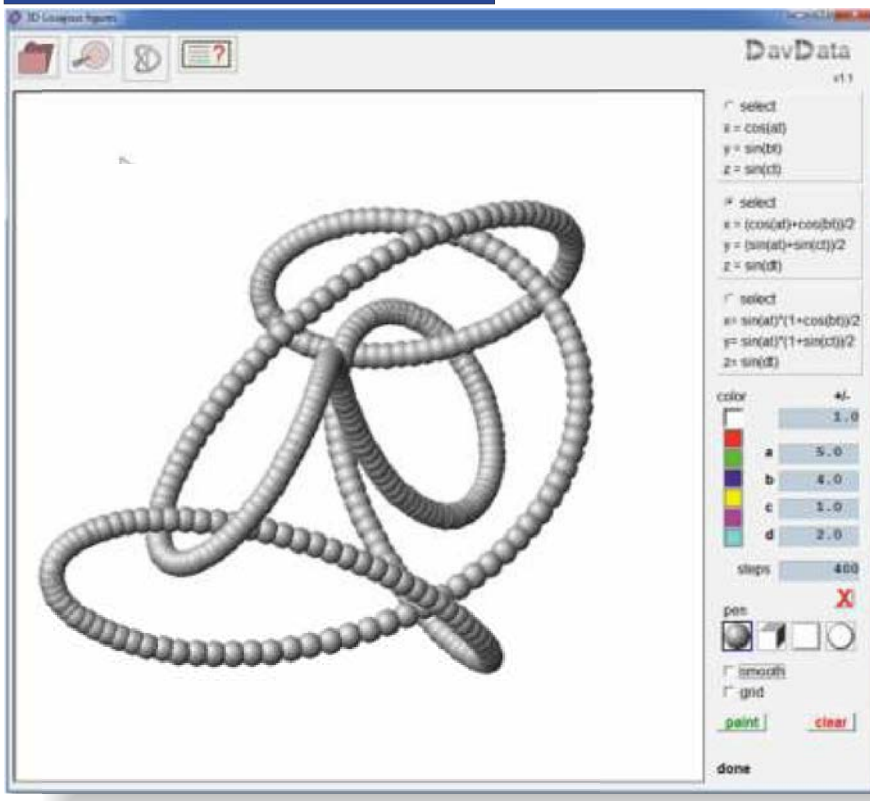
Figure 1: Rank at work; 4 elements of 10: rank 100

CHAPTER 46 PASCAL PROGRAMMING

COMPUTER ART 3D LISSAJOUS GRAPHICS

Introduction

In this chapter we will produce computer art by means of 3 dimensional Lissajous figures. Lissajous (1822..1880) was a French mathematician, famous for his research on waves. The lissajous3d program allows the plotting of 3 dimensional Lissajous graphics. Below is a reduced picture of this (Windows) program at work: Below is a reduced picture of the program at work:



Selections are made by mouseclicks. For the constants a,b,c,d :
- a left mouseclick adds the (+/-) value
- a right click subtracts the (+/-) value.

The (+/-) value itself may be changed also by left- or right mouseclicks.

PROGRAM FEATURES

- choice of 3 sets of Lissajous formulas
- choice of 4 pen styles:
  - sphere
  - cube
  - square
  - circle
- choice out of 7 colors
- single dots or connected (smooth) lines
- step count from 100..1000
- save settings to disc (*.l3d extension attached)
- load settings from disc
- save picture (*.bmp)

Interim variable t counts from 0 to the selected stepcount.
Stepcount is selectable between 100 and 1000 in steps of 50.

The menu and buttons are self-explanatory.

Introduction
Here you see a 3 dimensional tic-tac-toe game. One in progress one done. It is a two-player version, written for Windows computers. Winner is the one who is the first to places three adjacent O or X characters horizontal, vertical or diagonal. See fig 1b A single player may try to find the best strategy.

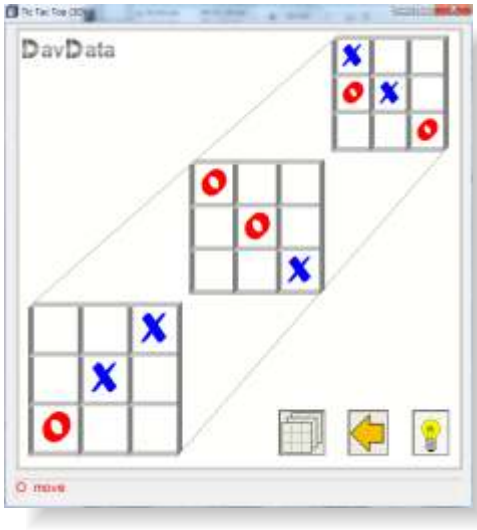


Figure 1a: 3D Tic-Tac-Toe in progress

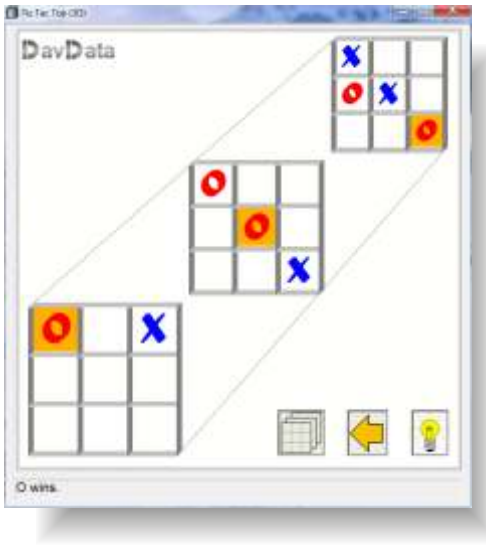


Figure 1b: 3D Tic-Tac- Toe done

There are three buttons:
- cube : new game
- arrow : take move back
- lamp : analyse board state

Analysis displays the result of a move in each field.
W3 means : winning in 3 moves.
L5 means : losing in 5 moves.

This article describes a Delphi programming project of a 3 dimensional tic-tac-toe game.
It is a two player version, however, for a single player it may be interesting to find a winning strategy. A game state may be analysed: per field a calculation is made that predicts the result of a move in that field (*e.g. winning in 5 moves, losing in 7 moves*).

This game is simple, as is the program. However, in the case of more complex games having menus to choose from, single and dual player options or several game levels, the same structure may be used. Also the analysis procedure is applicable to a wide variety of games. Therefore this project is a general blueprint for board games.

The Delphi(7) project has one form and two units.
Form1 holds the game and three images used as buttons.
Unit1 handles events, paints the game and contains the procedures to control the game.
Unit2 has the data for the game, moves and the procedures for game analysis.

CHAPTER 48 PASCAL PROGRAMMING

BITMAP ROTATION DELPHI

Introduction
This article describes a Delphi project for bitmap rotation. There are 3 units:
- unit1: exerciser to test the rotation procedures
- rotation_unit: procedures for bitmap rotation
- clock_unit : time measurement procedures

Exerciser

The form has buttons for loading and saving bitmaps.
Also 3 modes of rotation are selectable
- coarse: fast but less accurate
- medium: somewhat slower but destination bitmap is fully covered
- fine: slow, but with soft edges

Bitmaps are displayed in paintbox1. Moving the mousepointer over the paintbox with leftmousebutton pressed, causes the picture to rotate in the selected mode. Below is an example of medium mode rotation

Figure 1: Medium mode rotation

Rotation takes place between a source and a destination bitmap. In coarse mode, the source bitmap is scanned pixel by pixel and projected on the destination bitmap. Therefore, not every pixel of the destination bitmap may be covered.
In medium mode, the pixels of the destination bitmap are scanned and their value is loaded from the source bitmap. This insures that all pixels of the destination bitmap are covered.
In fine mode, the scanning is the same as in medium mode, but each pixel is divided in 9 equal parts. Parts may cover different pixels in the source map, the colors are averaged for the final color of the destination pixel.

Programming

The programmer has to create both the source and the destination bitmap. Before a rotation may take place, the rotation_unit has to be informed about the names of the bitmaps.
This is done by a call to

```
procedure
    setmaps(sourcemap,destination map)
```

Setmaps sets the pixelformat of both bitmaps to 32 bit.
Also, the dimensions of the destination bitmap are adjusted to accomodate all possible rotations of the source map.

```
procedure coarserotate(deg : word)
procedure mediumrotate(deg: word)
procedure finerotate(deg: word)
```

may be called. Deg is the rotation angle in degrees from 0..360. Rotation is clockwise, so, for a left rotation of 90 degrees, 270 degrees must be specified. Do not forget to call the setmaps procedure after loading an image from a file into the source map.
This insures the proper 32 bit format and dimensions of the destination map.

CHAPTER 49 PASCAL PROGRAMMING

Introduction
Logic10 is a Windows program for applications of Boolean Algebra.
Its purpose is:
- generate Truth Tables from formulas
- reduce Truth Tables to the simplest form.

This article describes truth table reduction by Logic10 version 2.0

Boolean Algebra

Like any algebra, Boolean Algebra has variables and operators.

Variables

Are denoted by a single capital letter: A, B, C.....
A variable can have the value true (1) or false (0).

Operators

. for AND
+ for OR
/ for inverse

NOTE: Normally, the inverse oparator is a horizontal bar on top of the character however, this is hard to edit.
" / " has the highest priority, then " . " , then " + "
NOTE: Between variables, the " . " operator may be omitted, so AB = A.B

Formulas

Variables and operators together make formulas. The most convenient format is CNF (*conjunctive normal form*).
This is the format
ABC + DEFG + H + IJ
ABC, DEFG, H , IJ are called terms.

CNF is a list of terms that are OR'ed.

The CNF formula AB + /AC is true
if A=1 and B=1
.....OR.....
if A=0 and C=1

PROGRAMMING TRUTH TABLE REDUCTION

Data Structures

In Logic10, a maximum of 15 variables is allowed. These characters are stored in alphabetic order in the `Variable Table`.

In a formula, a variable may be in the *true* or in the *false* state.
In:

AB/CD	A,B,D are in true state
C	is in false (*negated*) state.

Terms are coded in a 16 bit word.
Each bit (0..14) corresponds with a variable in the Variable Table. A bit is 0 for the variable in the negated state, a bit is 1 for the variable in the true state.
So, the term is true when its bits are the same as the variable values.
Below is the term: A/BCD/E/FGH

A	B	C	D	E	F	G	H
1	0	1	1	0	0	1	1

Figure 1

In formula: AB/C + ADCE + D/E
we notice variables A,B,C,D,E.
Each term holds only some variables, not all. So per variable a bit is needed to indicate the presence. Another 16 bit word holds bits that enable (1) or disable (0) the corresponding variable.

Figure 2 shows the term /BCEF/G where the Variable Table is ABCDEFGH.

CHAPTER 50 PASCAL PROGRAMMING

ABSOLUTE FUNCTION DELPHI PROGRAM

Introduction
The Delphi "system" unit contains a function called abs(x) where x may be real or integer. This function returns the absolute value of x, so

```
abs( 15) = 15
abs(-15) = 15
```

Note, that the abs( ) function combines actually two different functions:

```
if x < 0 then abs(x) = -x
if x > 0 then abs(x) = x
```

This article focuses on some surprising cases where use of the abs function is time saving and convenient.

1. Distances

While moving the mouse over the screen, (x,y) coordinates are generated.
Say, successive mouseclicks save these coordinates in (x1,y1) and (x2,y2).
We want variables dx and dy to hold the horizontal and vertical distances between points 1 and 2.

This code does the job

```
dx := abs(x1 - x2)
dy := abs(y1 - y2)
```

Note, that there is no difference between the results of abs(x1 -x2) and abs(x2 -x1).

To avoid negative distances, if statements could be used, but this approach is slower. Modern processors execute instructions in pipelines that read codes ahead. If statements disrupt this proces, the pipeline has to be rebuild when instruction flow alters.

2. Lower boundery

Say we use a variable x in our program and x is not allowed to be negative.
This means, that a negative value of x must be rounded to zero.
An if statement may do the job, but an abs( ) function can do the job faster.

To understand how this works we first have a look at the graph of the function y=abs(x).

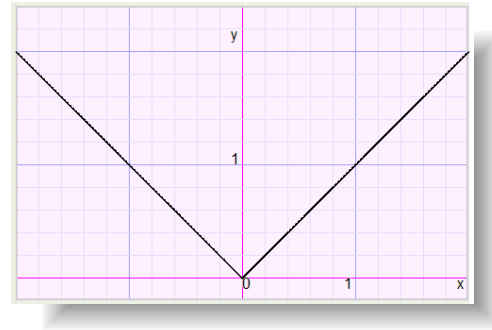


Figure 1: Function y=abs(x)

In the above graph the scale is 0.2.
In position (0,0) the graph has a sharp bend. This is a unique property of the abs function. Other functions like polynomials, sin, cos, tan, log have smooth graphs, no bends.

All following results are obtained from the above graph.
The bend may not be positioned at (0,0).
Therefore we first introduce a few trics to manipulate graphs in general:
shifting, scaling, reflection.

If in a general function y = f(x) , the x is replaced by (x-1) than the graph will shift 1 scale to the right.
This is obvious: say point P(p,q) is on the graph y = f(x) , which means that q = f(p).
However if x is replaced by x-1 then not p = x, but p = x -1 is the case so: x = p + 1.
p is changed to p + 1, indicating a shift of 1 scale to the right.
In a similar way we conclude that replacing y by (y-1) shifts the graph 1 scale up in vertical direction.
Replacing x by (x+1) shifts the graph 1 scale left, replacing y by (y+1) shifts the graph 1 scale down.

Introduction
This article is about the programming of tree graphs.
It includes creation, modifications and also undo operations. In mathematics a graph is a picture of dots (*also called node*), which may be interconnected by lines. Each dot (*node*) represents a situation or object.
An interconnecting line indicates a dependency between the dots or a transition. Graph theory is about the properties of graphs. Graphs are applied in a variety of fields such as

- road maps and floor plans
- electrical networks
- molecules
- industrial planning
- combinations
- information technology :
- file systems
- computer games
- text editing

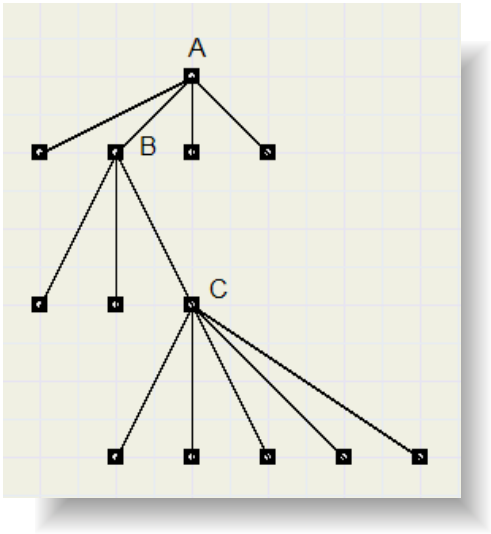


Figure 1: Example of a tree graph

The Delphi-7 project has three units:

1 : unit1
- paintboxes to visualize the tree and show edit actions
- buttons for creation and modification of the tree

2 : data unit with custom defined data
3 : tree unit with all data structures, procedures and functions.

All code to modify trees and undo are contained in the tree unit. Form1, Unit1 allow testing of the tree procedures.
The data unit is only there to isolate the custom data from the tree properties.

Trees
A tree is just a type of graph. Common properties of tree graphs are

- there are no isolated nodes (*each node is connected to at least one other node*)
- if 1 interconnecting line is removed, 2 isolated trees will result
- the route from node A to B is the same as from B to A, there are no loops

This tree could represent a file system where map A contains four files, including map B with three files including map C containing five files.

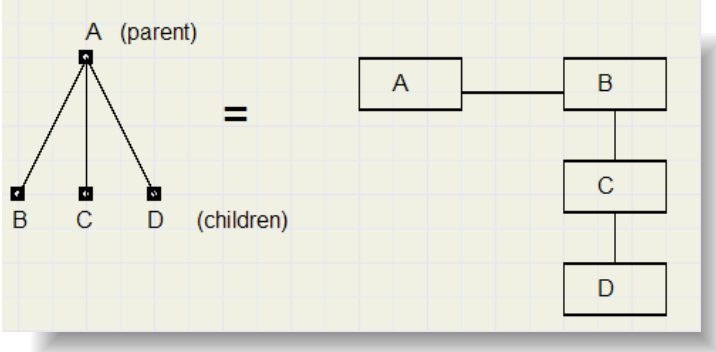


Figure 2: A graph is pictured with elements A,B,C,D

CHAPTER 52 PASCAL PROGRAMMING

Introduction
This article is about drawing in the Delphi programming language.

In part - 1, the very basic principles are covered : single pixels.
The described methods may be used as the basis for more complicated geometry.
And, by having full control of the drawing process, the programmer may add enhancements that Delphi does not offer, such as:

- dash-dot lines of greater width then 1 pixel
- lines that change color on the way
- canvasses with several levels
- use of a clipping rectangle
- slow motion drawing for educational purposes

Part-2 covers drawing dots and lines.

Part-3 describes a method for smooth, flicker free, drawing.

Part-4 describes a way to paint circles and ellipses in the Delphi programming language.

Part-1
In Delphi, several ways exist to draw lines or fill shapes.
This article focuses on the comparison between these methods.

Ultimately, all drawing amounts to setting a single pixel on a bitmap canvas.
Therefore, single pixels are written and the time required is measured.
Each drawing method is used in two type of actions
1. drawing a line
2. filling a rectangle with a color

About the program
On the Form, several BitButtons are placed.
Pressing a button triggers a specific action and the time needed per pixel is indicated in the Statictext.

All drawing is done in a 100 * 100 bitmap, named bm.
The pixelformat is 32 bit.
A paintbox is added to show the contents of the bitmap after drawing.

The time is obtained from the system clock, which counts milliseconds, by the GetTickCount function.
For accurate measurement, the operation has to be repeated many times.
Procedure ShowTime(n) presents the elapsed time after writing n pixels.

CHAPTER 53 PASCAL PROGRAMMING

DELPHI DRAWING, PART 2 DELPHI

INTRODUCTION PART 2
Part-1 of this chapter summarized the basic ways of modifying pixels. Main conclusion was that for fast drawing it is best to write directly to memory, minimizing the use of the pixels[..,..] or scanline[..] properties. This requires calculation of the pixel address.
Goal of this little research is to add some extra features in the TBitmap class.
This new class is called the Xbitmap class.

The new features are:
- clipping rectangle
- 4 drawing levels
- multipixel wide dash-dot lines, 20 styles
- improved stretchdraw method
- improved floodfill, with (*optional*) enlarged fill patterns for printing
- optional arrows at begin/end of lines and arcs
- calculation of rectangle that was modified

Clipping Rectangle

By default, the XCliprect rectangle is set to the full size of the bitmap. Pixels outside this rectangle cannot by modified. When lines, circles, polygons ... are drawn, painting outside the XCliprect is suppressed.

Drawing Levels

Bits 0,1 of a pixel (*so blue bits 0,1*) make the drawing level. Highest level is `0 (00)`, lowest is `3 (11)`. Reason is the white `($ffffff)` background of the printer canvas. Typical use of the levels may be:

level 0 - (00) :
foreground color for text, lines
level 1 - (01) :
grids
level 2 - (10) :
user supplied background
(*brush when filling rectangles...*)
level 3 - (11) :
system supplied backgound

Rule : a pixel cannot be overwritten by a lower level pixel.
The effect is a 4 layer bitmap.

Modified Rectangle

The `XModRect` rectangle contains the modified pixels. Outside `XModRect` no pixels were changed in the most recent operation. `XModRect` assists in dynamic drawing where modifications in the bitmap must be copied to a paintbox.

XBitmap Properties

Xpenwidth
(1..32) the diameter of the pen
XPenLevel, XBrushLevel (0..3)
XPenColor, XBrushColor
(*DWord with r.g.b. color value*)
XlineStyle
(0..19) dash - dot pattern selection
Xarrowcode
(0..15) type of arrow + bit8 = 1
for begin arrow, bit7 = 1 for end arrow
Xfillstyle
(0..15) selects fill pattern for floodfill
XfillSize
8*8 (normal) or 24*24 (*printer*)
Xusebrush true if brush is in use

NOTE:
above properties are added to the Bitmap, not the canvas, so this is valid:

```
myXbitmap.Xpenwidth := 10;
```

CHAPTER 54 PASCAL PROGRAMMING

Introduction
This article describes a method for smooth, flicker free, drawing.
Please refer to figure 11 below:

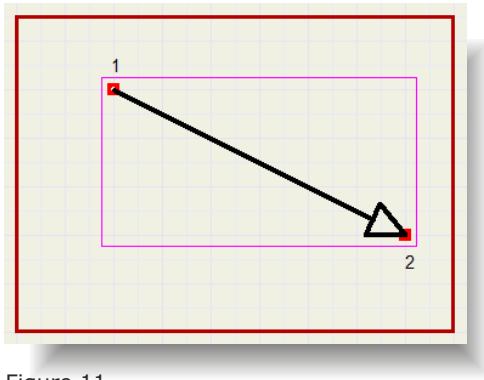


Figure 11

On a paintbox a line with arrow is drawn. This is done by moving the mouse-pointer over the paintbox, pressing down (*and holding*) the left mousebutton at point 1, moving the pointer to point 2, where the mouse-button is released.

General Description
The affected rectangle of the paintbox is marked by a purple line.
Outside this rectangle, no changes took place. While the mousebutton is down and the mouse is moving, a new line must painted after each mouse-move.

So, at each mouse-move, we have to

- erase the old line
- paint the new line

Erasing may be done by repainting the effected rectangle. But by painting directly in the paintbox, erasing and repainting causes flickering, which is very unpleasant to the eyes.

A better way is to paint in a Bitmap and copy only the changes to the paintbox. While painting in this bitmap, during the paint process old images have to be erased before the new image, representing the new position, can be painted. So, we add another bitmap, which holds the unchanged image and thus can be used to restore parts of the bitmap we use for painting.
Restoring is conveniently done by

```
procedure
bitmap2.canvas.copyrect
(pRect,bitmap1.canvas,pRect)
```

where pRect is the affected area that needs restoring.

See figure 12.

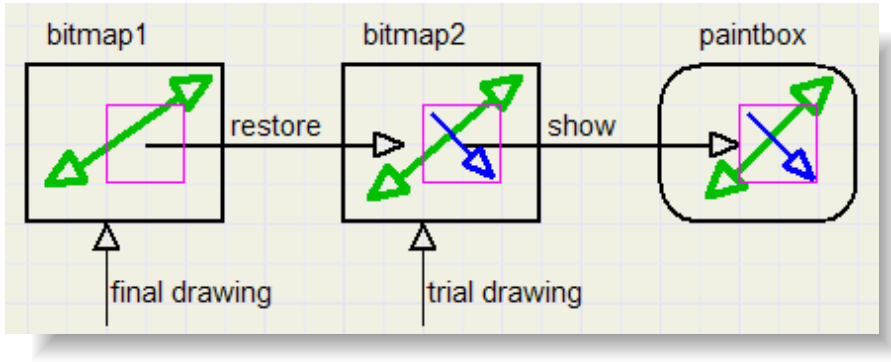


Figure 12

Introduction
This article describes a way to paint circles and ellipses in the Delphi programming language.
Please look at the picture below:

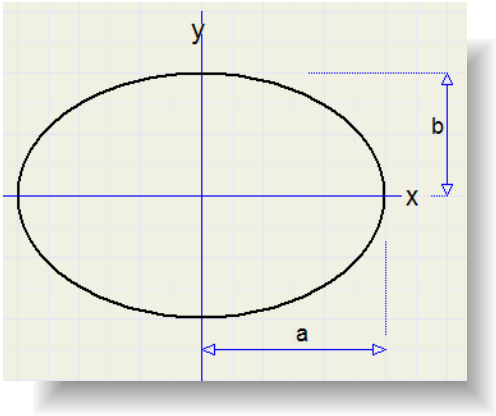


Figure 13:

The general equation of an ellipse is:

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1$$

In case of a circle, where a = b = r (*radius*), the equation becomes
$x^2 + y^2 = r^2$

For the convienience of calculations we assume the center of the ellipse at (0,0).

The ellipse is painted point by point. To avoid unpainted "holes" between pixels, the slope of the tangent must be observed.
If the tangent is < 1 (and > -1) , a horizontal oriented line , the x coordinate steps by 1 and the corresponding y is calculated.
However, if the tangent is > 1 (or < -1) , a vertical oriented line, then y must be stepped by 1 and x is calculated.
So we first calculate the point on the ellipse where the tangent = 1 (or -1):

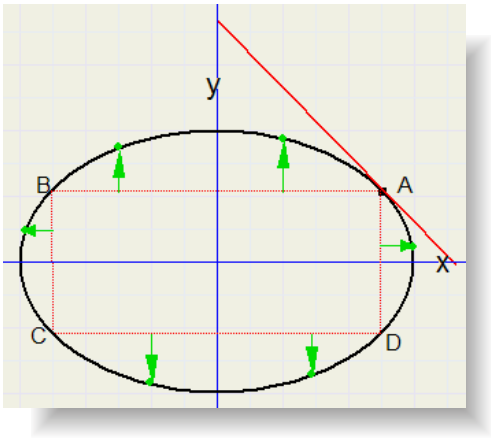


Figure 14:

The derivative of the equation is: .....

$$\frac{2x}{a^2} + \frac{2yy'}{b^2} = 0$$

If y' = -1 we get

$$\frac{x}{a^2} - \frac{y}{b^2} = 0 \ldots y = \frac{b^2 x}{a^2}$$

Substitution of y in the original equation of the ellipse:

$$\frac{x^2}{a^2} + \frac{b^2 x^2}{a^4} = 1 \quad ==>$$

$$a^2 x^2 + b^2 x^2 = a^4 \quad ==>$$

$$x = \frac{a^2}{\sqrt{a^2 + b^2}} \quad ==>$$

and similar

$$y = \frac{b^2}{\sqrt{a^2 + b^2}}$$

CHAPTER 56 PASCAL PROGRAMMING

Introduction
Sometimes is it desirable to save text in an unreadable form, as is the case with passwords. This article describes a simple way to encrypt a character string.

CHARACTER STRING ENCRYPTION DELPHI

The Method
A character string is regarded as groups of 4 characters. So, characters 1..4 make group 0, characters 5..9 make group 1, and so forth. Each group of 4 characters is packed into a 32 bit integer (`DWORD`).

Then the bits of this DWORD are interchanged. Next the DWORD is regarded as groups of 6 bits. Groups 0..4 are 6 bits wide, group 5 has 2 bits left.

A 6 bits group is an index into a preset string of 64 characters, so each 6 bit value yields a character. These characters together make the encrypted string.

The advantage of this method is , that a character has no fixed translation, because it depends on the neighbouring characters in the string.

Note, that 4 characters (*each 8 bits in size*) translate to 6 characters.

The figure below shows the data flows.

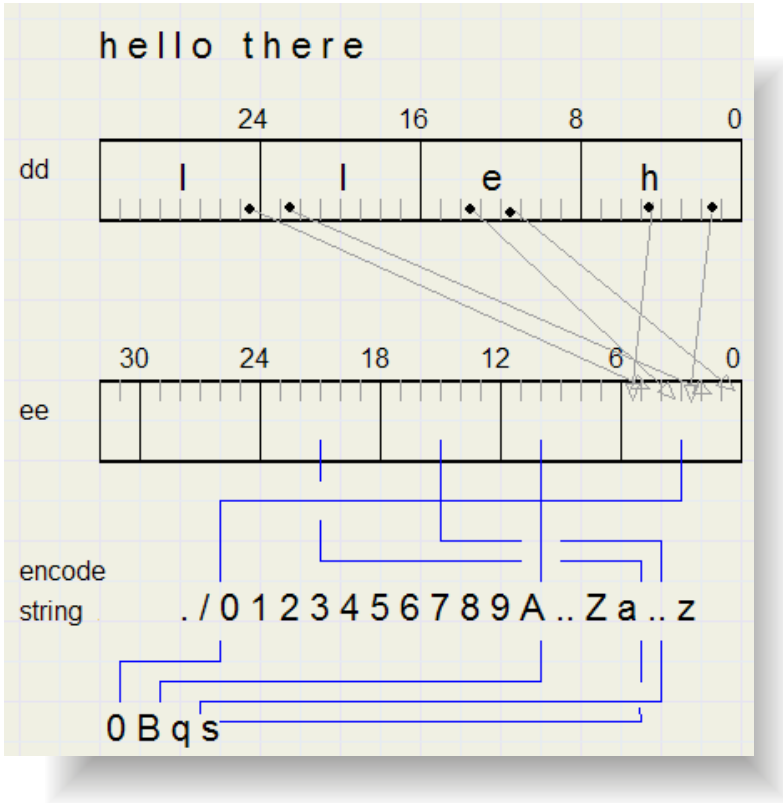


Figure 1:

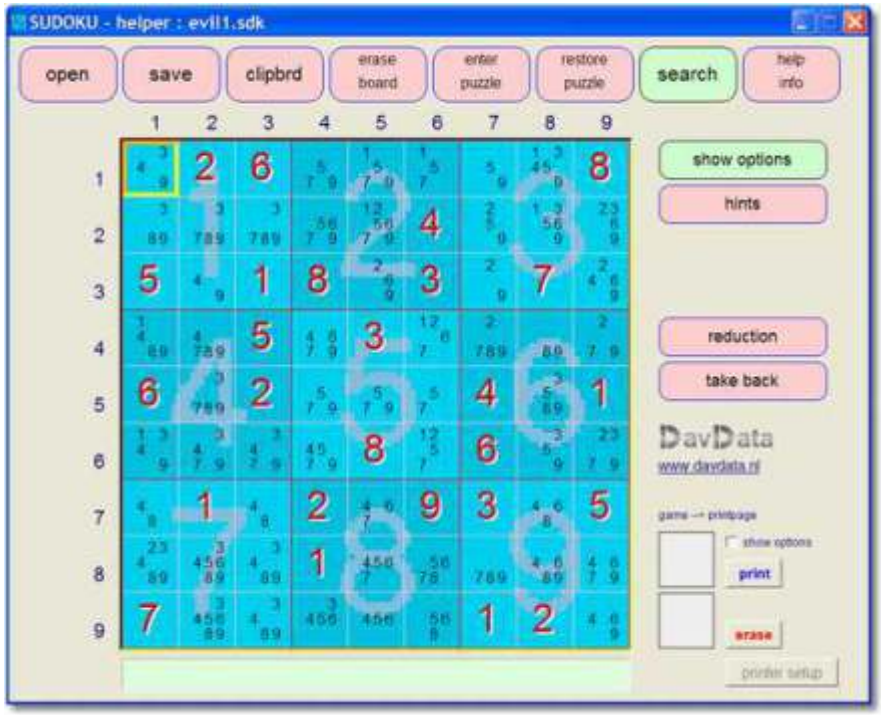
CHAPTER 57 PASCAL PROGRAMMING

SUDOKU HELPER - SOLVER

Introduction
Sudoku is a very popular number puzzle. This Sudoku Helper - Solver program assists in the solution of Sudoku puzzles from easy to very difficult.

Below is a reduced image of the SUDUKO - helper / solver.

- in-line help information
- while searching for solutions:
  - take numbers back
  - color difference between original numbers and numbers added
  - reject wrong numbers already present in the row, column or group
  - restore original puzzle



Options

standard:

- input of numbers for a new puzzle
- erase the board
- save puzzles on disc (*.sdk extension*)
- open puzzles from disc (*.sdk extension*)
- print puzzles, including options, maximum 2 per page
- copy puzzle to clipboard
- paste puzzle from clipboard

selectable:

- while searching for solutions:
- show options in open fields
- warn if erroneous board state occurs (*zero-option field, missing options*)
- indicate single choice fields
- automatic moves on single choice fields
- reduction of options per field, - by analysis

The **Sudoku Helper - Solver** program is freeware and may be distributed without restrictions. It ships as a single (*.exe*) file. There is no installation procedure. Just copy it to a directory of your choice. The Windows Registry is not changed.

CHAPTER 58 PASCAL PROGRAMMING

AN APPLICATION OF "ABS" FUNCTIONS

Introduction

We have to paint a rectangle 45 degrees rotated. In this article I describe a neat way to do this. Let us give the top pixel the coordinates (sx,sy). See figure 2 below. The sides of the rectangle have lengths of h and v. Say, we are painting the line indicated yellow. We will accomplish this task by painting a horizontal line from x1 to x2 at y pixels from the top of the bitmap.

Figure 1 below shows a bitmap with rectangle. The pixels of this rectangle have to be set to the color black.

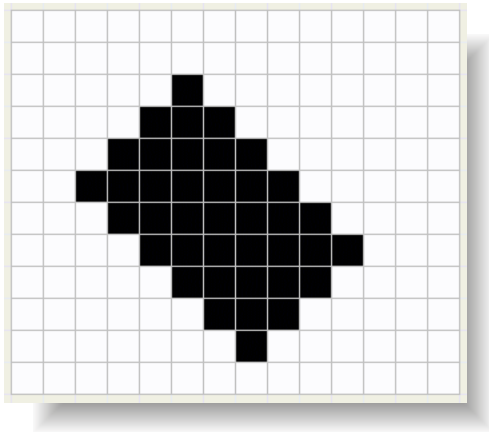


Figure 1

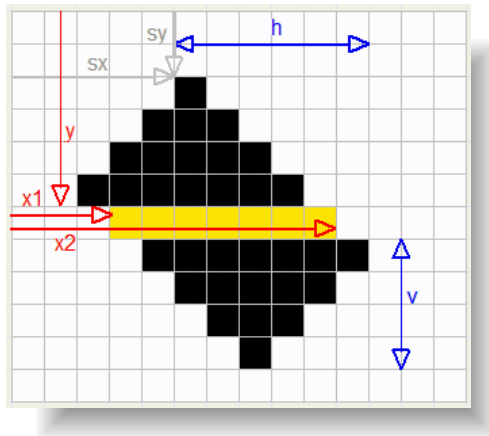


Figure 2

So, the program will look like this:

```
var   j : word;
      h,v : word;
      sx,sy : word;
      bm : Tbitmap;
begin
.....
//creation of bitmap
//setting rectangle dimensions h and v
//setting position (sx,sy)
...
 with bm.canvas do
  begin
   pen.color := 0;
   pen.width := 1;
   for j := 0 to h + v - 2
         //from top to bottom of rectangle
    begin
     y := sy + j;
     x1 := ???;
     x2 := ???;
     moveto(x1,y);
     lineto(x2,y)
    end;//for j
  end;//with bm
....
end;
```

Let's take a look at x1 and its relation to j. To simplify the situation we assume that the top pixel has coordinates (0,0).

j	0	1	2	3	4	5	6	7	8
x1	0	-1	-2	-3	-2	-1	0	1	2

Figure 3 shows the graphics

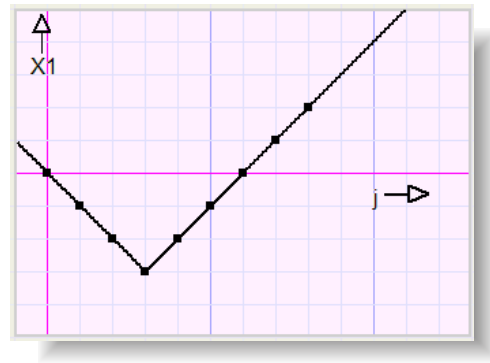


Figure 3

CHAPTER 59 PASCAL PROGRAMMING

Introduction
Often it occurs that instructions have to be repeated as long as a number of conditions are satisfied. Look at the following flowchart, where proc1,2,3 are statements and procedures and quest 1,2,3 are questions to decide where to continue the program based on different conditions.

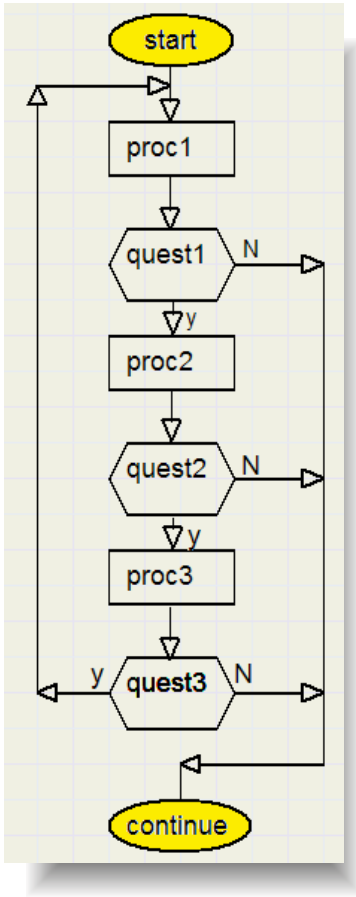


Figure 1
These loops are simple to program in the old-fashioned way: labels and goto statements.
In a structured way however complications may arise, see flowchart figure 2.

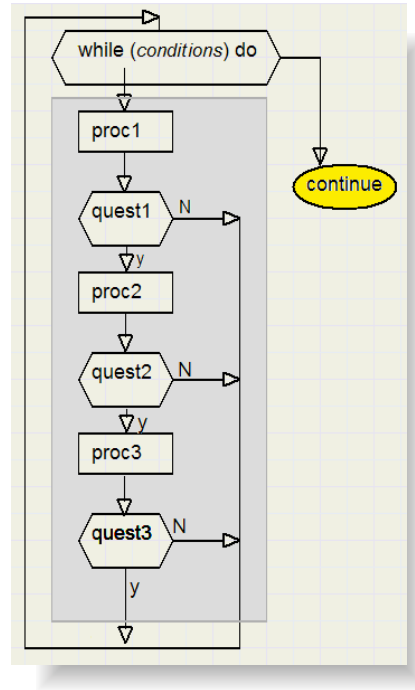


Figure 2:
The loop is repeated as long as the conditions in the while statement are satisfied.
Problem is, that when returning to the while statement after a loop, the path taken through quest1..quest3 is not obvious.
This results in complex condition checking.

CHAPTER 60 PASCAL PROGRAMMING

Introduction
This chaptor gives some suggestions how to control your (board) game.
Illustration below shows the layout of such a game, connect4 or whatever.

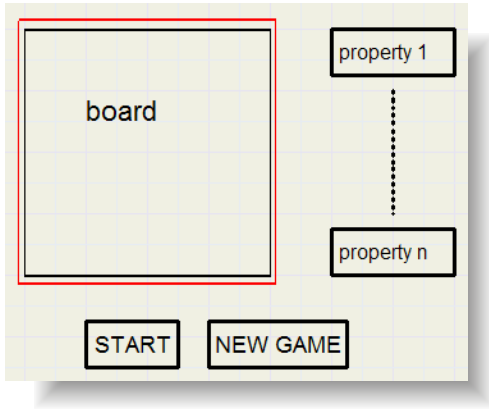


Figure 1: Layout of a (board) game

There is a board, where pieces are placed by the players. A game is started by pressing the "start" button. A game is ended by pressing the "new game" button. Properties as level or strategy are selectable by the "property" buttons.

First of all, procedures have to be written to paint or erase the board.
These are low level paint procedures.

Next, data structures must be designed, such as:

- a two dimensional array representing the board state
- a one dimensional array holding the list of moves
- additional arrays of constants that control game behaviour

NOTE: the low level paint routines must not use the arrays, which are higher in level. Reason is, that a change of the data structures during the design phase does not need a redesign of the low level paint routines.

CONTROLLING YOUR GAME

Next, the high-level procedures must be designed. They take care of the moves by altering the arrays and calling the low level paint routines.

The high level procedures are called by events from mouse or keyboard.
Problem here is, that actions originating from such events depend on the situation of the game.
So, together with the data structures, the game states must be defined:

```
type TGameState =
(gsInitializing, gsMenu,
gsPlayerMove, gsComputermove,
gsPlayerWin, gsComputerwin, gsEnd);
var GameState : TGameState;
```

gsInitializing	during startup of game. Initialize tables such as clearing the moves list.
gsMenu	allow for property selections between games.
gsPlayerMove	allow mouse or keyboard events to define player move.
gsComputerMove	block events, while computer calculates/ displays its move.
gsPlayerWin	game end by winning of player.
gsComputerWin	game end by winning of computer.
gsEnd	game end, board full, no winner.

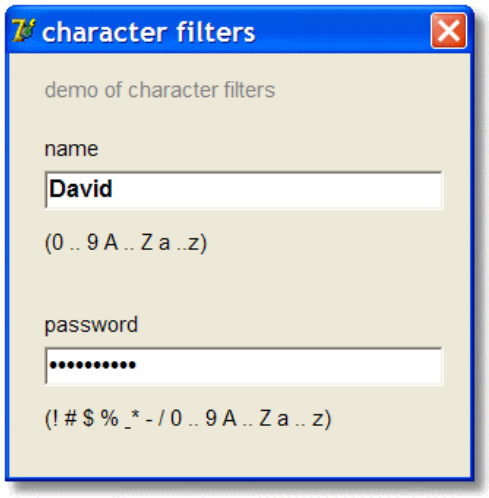
Besides the mouse and keyboard events, it is convenient to consider the control buttons as well as events.

```
type TGameEvent =
(geInitDone, geStart, geNewGame,
gePropertyChange, gePlayermoveDone,
geComputermovedone);
```

CHAPTER 61 PASCAL PROGRAMMING

FILTERING CHARACTERS

Introduction
This chapter describes how to program character filters, to allow only a selected set of characters to enter an Edit component.
Below is a form with two Edit boxes, one (nameEdit) for entering a name, a second (pwEdit) for entering passwords



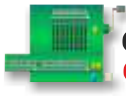
Below the Edit components, the allowed characters are listed. For the pwEdit, the property passwordChar is set to #149 during design time. So each character entered here will be displayed as a big dot. Both Edit components are placed on form1. To be able to inspect a typed character, the keyPreview property of the form must be set true.

A typed character generates a `TForm1.FormKeyPress(Sender: TObject; var Key: Char)` event.
If the character key is not part of a predefined set, key is set to #0, character code 0, which is ignored.

Make sure the character #8 (backspace) is also allowed, else a backspace in the Edit box is not possisble.
This is the complete Delphi-7 unit.

```
unit Unit1;
interface

uses
  Windows, Messages, SysUtils,
  Variants, Classes, Graphics,
  Controls, Forms,Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    nameEdit: TEdit;
    pwEdit: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    procedure FormKeyPress(Sender:
              TObject; var Key: Char);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var  Form1: TForm1;

implementation

{$R *.dfm}

 procedure namefilter(var ch : char);
 begin
  if not (ch in
      ['0'..'9','A'..'Z','a'..'z',#8])
   then ch := #0;
 end;

 procedure pwfilter(var ch : char);
 begin
  if not (ch in ['!','#'..'&','*','-
 ','/','0'..'9','A'..'Z','a'..'z',#8])
   then ch := #0;
 end;

procedure
TForm1.FormKeyPress(Sender:TObject;
  var Key: Char);
 begin
  if activecontrol = nameEdit then
   namefilter(key);
  if activecontrol = pwEdit then
   pwfilter(key);
 end;
end.
```

You can download the accompanying files: filter.exe
The complete project and code is available for download filter.zip



CHAPTER 62 PASCAL PROGRAMMING

Introduction
This chapter presents a simple function that searches for specific data in an array.
As an example an array of strings, representing names, is chosen.
A while ... do ... loop is used.

WHILE DO LOOP TO SEARCH AARRY

In this case, make sure that in Project Options/compiler "complete boolean evaluation" is set false.
So, if (i <= namecount) is false, (names[i] <> s) comparison will not take place.

Data

```
const maxname = 1000;
 var names : array[1..maxname] of string;
     namecount : word; // number of names in array
```

Function findname compares string s against the entries in array names.
If s is found, then the index of s is returned.
If no match is found, 0 is returned.

If no match is found, i will be incremented until it is greater than namecount.
A false result is returned.

The function

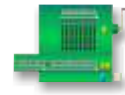
```
function FindName(s : string) : word;
var i : word;
    hit : boolean;
begin
  hit := false;
  i := 0;
  while (hit = false) and (i < namecount) do
    begin
     inc(i);
     hit := names[i] = s;
    end;
  result := i;
end;
```

The while..do.. loop repeats as long as no compare is found and the top of the array is not reached.
A convenient way of programming is also having the result of the function indicate if the search was successful.
The index is returned as a var parameter.

No code available

Alternative search function

```
function FindName(var w : word; s : string) : boolean;
 var i : word;
 begin
   i := 1;
   while (i <= namecount) and (names[i] <> s) do inc(i);
   result := i <= namecount;
   if result then w := i;
 end;
```

CHAPTER 63 PASCAL PROGRAMMING

Introduction
In some cases, as in educational software, it is necessary to execute a program step-by-step.
A convenient way to advance each step is by hitting the SPACE bar.
But also it must be possible to terminate the process and return to an input mode or whatever. A good key for this is ESCAPE.

Implementation of both demands requires two boolean variables.

1. `var stopflag : boolean;`

`stopflag` is normally true.
However, hitting the space bar sets stopflag false, which causes the program to continue.

2. `var runflag : boolean;`

`runflag` is normally true.
Program execution may proceed in this case.

Hitting ESCAPE clears runflag and the execution of our process stops.
The program itself does not stop, but only waits for new commands after skipping procedures.
A good way to halt program execution is `application.processmessages;`
This calls Windows to handle events.

A procedure for the program stop may be

```
procedure programstep;
begin
  stopflag := true;
  while stopflag do
   application.processmessages;
end;
```

STEP BY STEP PROGRAM EXECUTION

Say we have processes **1..4 to** step through.

```
The program now will look like:
procedure DoJob;
begin
  runflag := true;
  process1;
  programstep;
  if runflag then process2;
  programstep;
  if runflag then process3;
  programstep;
  if runflag then process4;
  programstep;
end;
```

runflag may also depend on process results such as error codes. In the case of errors, the processes that follow will be skipped if runflag is set false.

A problem is closing the form while stopped. So, the Close event must set runflag and stopflag to false, otherwise the program will not close until finished.

Normally a button press will start the job. It is convenient to use this button as well to continue the program when stopped. The final program (*see below*) takes care of that. In a project, events from keyboard or mouse have a meaning that depends on the state of the program.
Editing input data, processing data, waiting for a command.....

Therefore a variable with name such as programstate is needed. In this simple case, programstate may be psIdle or psExecute, to differentiate between executing and not executing the processes 1..4.

CHAPTER 64 PASCAL PROGRAMMING

Introduction
This chapter decribes the reading and writing of serial bit streams from/to a buffer. This problem was part of a project for image compression...

HANDLING SERIAL BIT STREAMS

Operations
Bit strings of varying length are presented to be stored in a buffer.
Also, bit streams of varying length are to be extracted from a buffer. The buffer is an array of dwords (*cardinal*).
The bit packages are first assembled into a 32 bit register, called ACCU.
When the accu is full, it is copied to the buffer and a pointer is updated to address the next entry in the buffer.
The data flow is illustrated in figure 1.

Acount counts the number of bits present in the Accu. When accu reaches a count of 32 (*or more*) it has to be saved into the buffer.
In the figure 1, the accu is shown twice for clarity.

Next page is the procedure for the storing of bit streams. The buffer is named "director" , its index is dirPTR.
A dword containing n bits must be stored ($n < 32$)

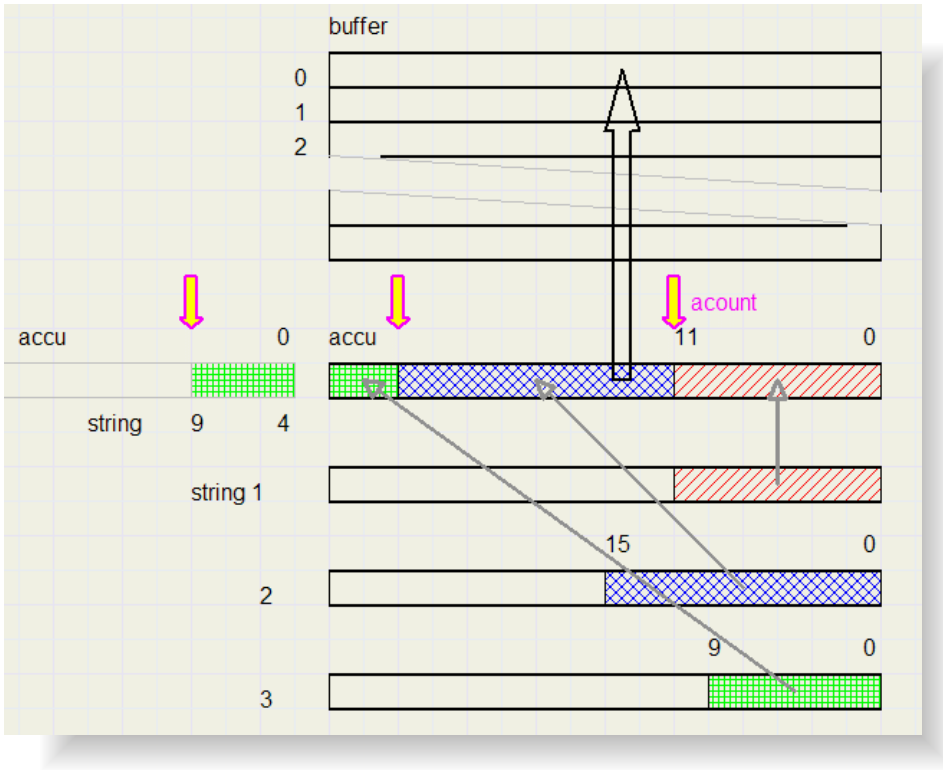


Figure 1

CHAPTER 65 PASCAL PROGRAMMING

Introduction
Sometimes it is convenient to have a simple color selector to be placed on the canvas of a dialog form. The component presented here is of moderate size but still capable of selecting 1 of 256000 colors, because it is constructed of separate slides for red, green and blue, each with 64 positions. Below is a picture of the (colormixer) component in action

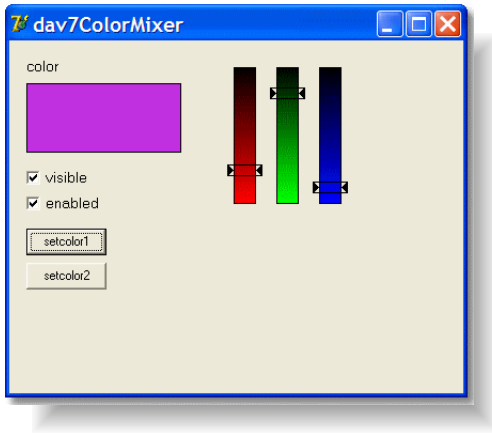


Figure 1

The colormixer class is placed in a separate unit. Form1 and Unit1 make an exerciser to test the component. The mixer only consists of the three vertical color bars. The other components are part of the exerciser.

The only property added is "color".
By writing a new value to the color property, the slides may be preset.
Read the color property to get the selected color. The color is always in the Windows `bb-gg-rr` format. Each time a slide changes position an event is generated that presents the new color to the application.
Each slide has a position of 0 to 63.
Before packing the slide positions into the 32 bits color value, 2 bits of value "1" are placed behind it to make the slideposition 8 bits in length. So, the minimal color value is `$00030303` and the highest is `$00ffffff`

The colormixer component has a fixed size of 120 * 140 pixels.
Do not set the width or height.

Here you see 4 colormixers in action (*picture 50% reduced*) in a dialogform to select frame properties.

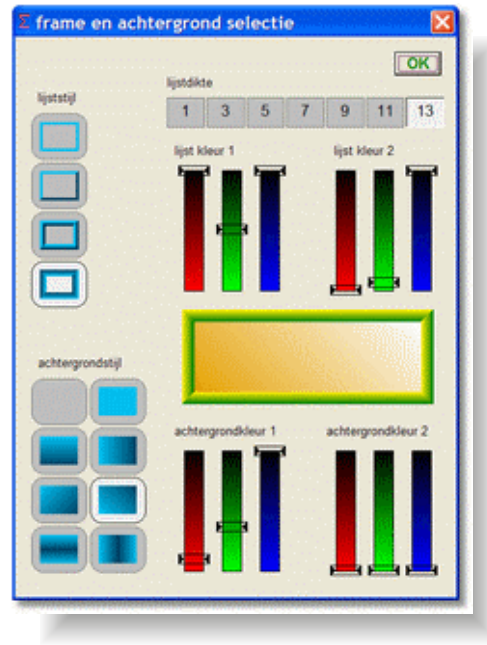


Figure 2:

You can download the accompanying files:
colormixer.exe
The complete project and code is available for download
colormixer.zip

CHAPTER 66 PASCAL PROGRAMMING

INTRODUCTION

On many occasions in programs a color has to be selected to fill a shape or draw a line. A convenient way for selection is a specialized pop-up form, a so called dialog form. After the selection the form closes. This article describes the programming of such a dialog form. Below is a real size picture:

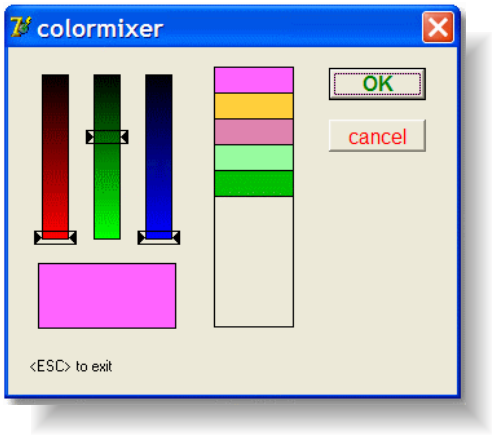


Figure 1

Color selection takes place by mixing the colors red, green and blue.

New however is that the last ten selections are saved. A quick selection of previously used colors is possible by simply clicking on the color.

Components

- Colormixer 3 slides for color mixing, my own component
- Showbox paintbox to show the mixer color
- Historybox paintbox with the last 10 stored selections

Constants and Variables

```
const maxhistory = 10;

var colorhistory :
      array[1..maxhistory+1] of dword;
    hcount : byte;
```

Colors have the pf32bit format.
Old selections are stored in array `colorhistory`.
`hcount` is the number of stored colors.

Procedures

A new color is stored in colorhistory[1]. Old colors in the colorhistory are shifted one place down.

Several cases have to be considered:

1. A first color is added to the `history.hcount` was 0 and is increased to 1.

2. Already some colors are in the history. All colors are shifted one place down, new color in `colorhistory[ 1 ]`. `hcount` is increased by 1.

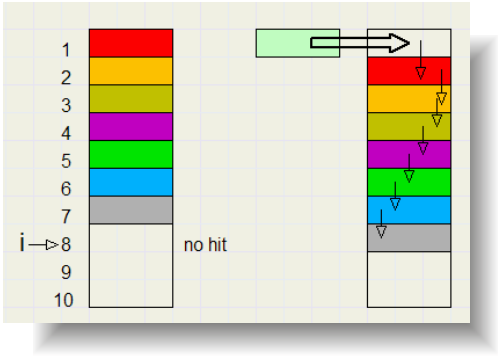


Figure 2

Introduction

Peg Solitaire is a single player puzzle.

The board consists of holes (33) and pegs (initially 32). The center position only is open. The final, solved, state has left 1 peg in the center position when 31 moves have striked the other pegs.
A move takes a peg over its neighbour (horizontally or vertically) to an empty hole. The peg that was jumped over is removed from the game. See the picture below for a reduced image of an initial- and a solved game:

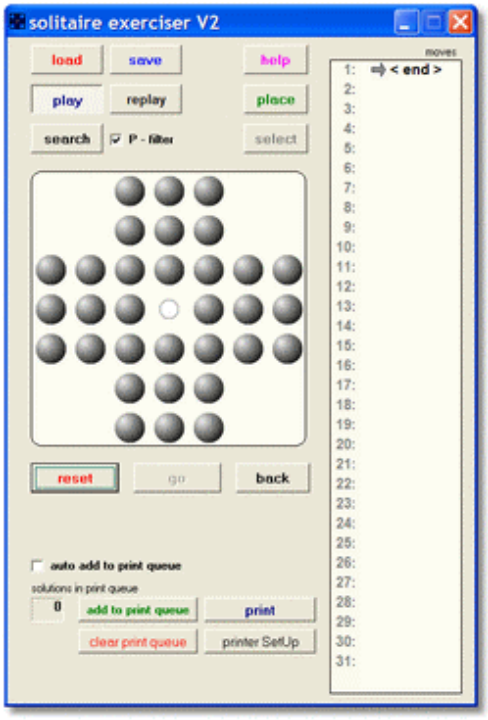


Figure 1: Initial board state

Figure 1 shows the original Solitaire board state and the final state is shown in Figure 2.
However, I have added several other starting positions, from easy to difficult. The final state is always the same: 1 ball left in the center.

Program Options

play	search for solutions
replay	replay previous moves or solution
search	have computer search for solutions
place	place balls at board to create starting position for search
select	select 1 of 12 preset games, from easy to difficult
save	save board / solution to disk
reload	reload game from disk
add to print-queue	add a solution to the print queue
print	print previously stored solutions
in line help	-
P - filter	permutation filter removes similar solutions : same moves in different sequence

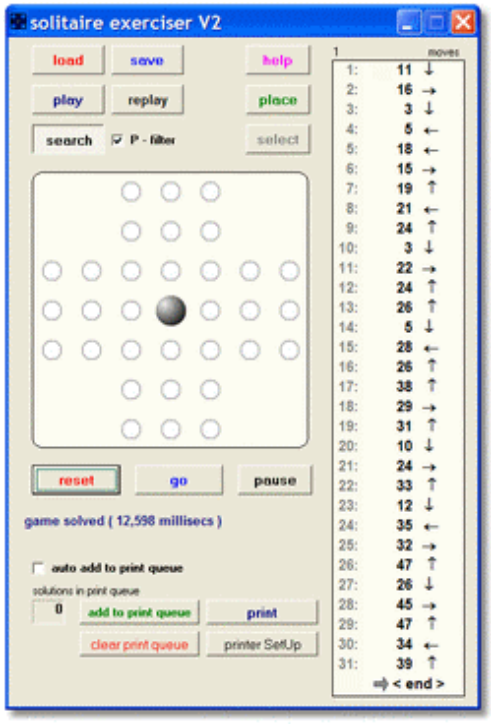


Figure 2: Final solution

Introduction
The SHIFT puzzle is sometimes called the most difficult puzzle in the world. Below are reduced pictures of the SHIFT puzzle in the initial- and in the final, solved, state.
The puzzle consists of a board with 10 blocks called A..I and X. The goal is to shift the blocks (which may not overlap) until the red X-block is in the center bottom position. At first glance, this exercise looks impossible.

How to write a program that solves this puzzle?

The Brute Force method
In this approach simply all possible move sequences are tried until the solution is met. (*by accident we may say*) Programming is relatively simple, but for more difficult puzzles the processing time will be very long : hours or even days. The process time may be shortened by avoiding superfluous moves.
The brute force approach is a good general starting point to solve puzzles but ways to speed up the process are puzzle dependent. In this case of the SHIFT puzzle we expect many moves until the solution is met.
Also we note that the number of possible moves is limited as are the different board states (*the way the blocks occupy the board*).
The first measure to limit processing time is establishing a maximal allowed number of moves.
If the last move is tried and there is no solution, this last move is taken back and the next sequential move is tried.

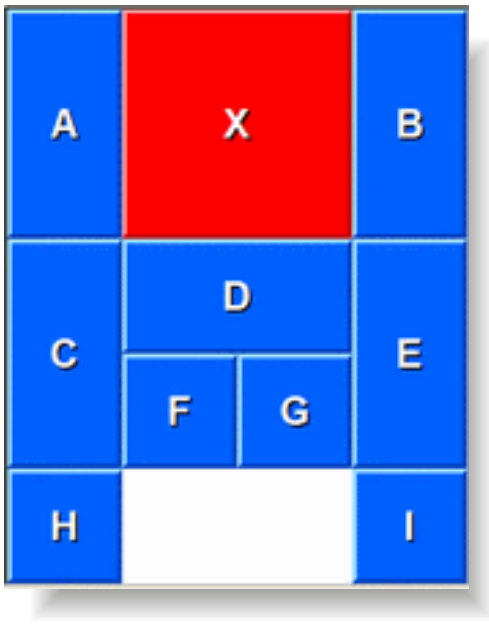


Figure 4: Initial state

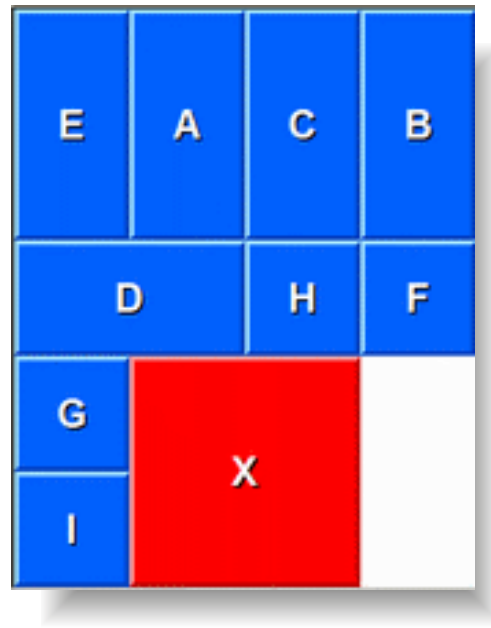


Figure 5: Solved state

Because no pieces (*blocks*) are removed by moves, the danger of repeating board states exists. This situation is avoided by storing the board states in a buffer.
If a move produces a board state that is already present in the buffer, the move is skipped. This is the second measure to limit processing time.

CHAPTER 69 PASCAL PROGRAMMING

FREEHAND DRAWING DELPHI

Introduction
This article describes a Delphi-7 project for freehand drawing (Figure 1).

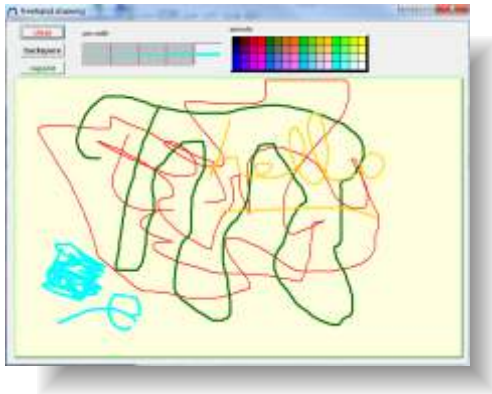


Figure 1: Freehand drawing

The drawing is generated by mouse-movements over a paintbox, connecting the (x,y) coordinates by lines.

A `ColorPicker` component is added to the form as well as an `ArrayButton`, to allow selection of pen-color and pen-width.

Drawing is not too difficult, but we want to store the drawing as well to be repainted later. The program allows for storage of 9 drawings. To verify this storage, bitbuttons "clear" and "repaint" are added. "Clear" erases the paintbox and "repaint" draws all recorded drawings again. The "backspace" button removes only the last added drawing.

Coding a drawing
I choose to code a drawing by means of a step-by-step route description. (*figure 2*)

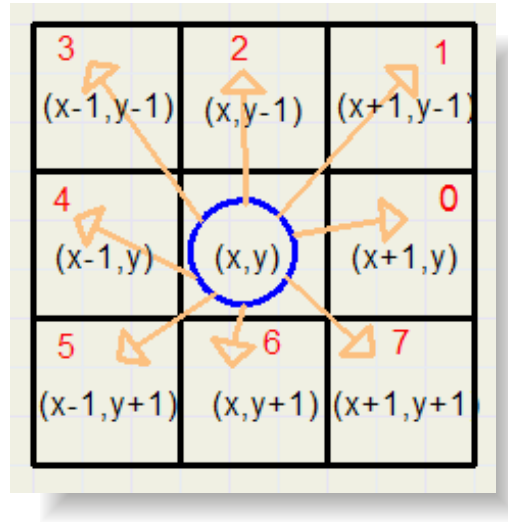


Figure 2: A step-by-step route description

Pictured are 3 * 3 pixels. From center (x,y) a code (0..7) points to the next pixel.
A freehand drawing can be defined by a list of these codes, given the coördinates of the starting point.

The code requires 3 bits for each step.
This is far from ideal, because multiples of 3 bits do not fit bytes or words.
A 4 bit code would be better.
The extra 4th bit provides an extra possibility, see figure 3:

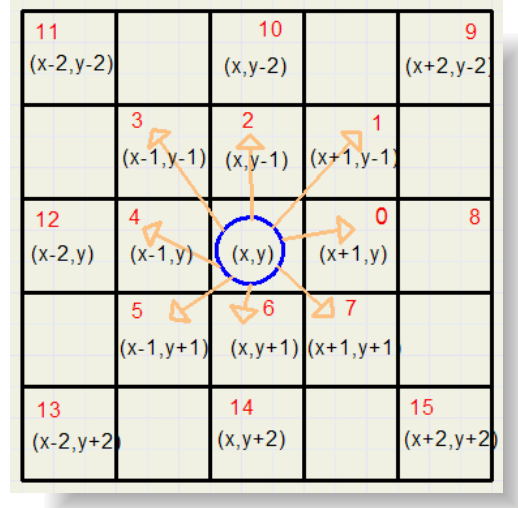


Figure 3: Extra 4th bit

Introduction
The Xfont project allows the creation of your own font. Xfont project was started to build a math-font containing characters of special operators as of set theory together with the characters of the Greek alphabet. To avoid redundant work, existing images may be used. These character images may be altered, new images added and new character codes may be assigned.

Which properties this font should have? We want the same font for the screen as well as the printer. Normal height of a screen character is 16 pixels. For a printout, having a higher resolution, a 3 fold magnification is required, so normal fontheight here is 48 pixels. Another requirement is a printout of double the normal height without any loss of quality. So, the image should be defined minimally as a 96 pixel height image.

The next question is how to encode the character images. The most simple way is a bitmap. The best quality is achieved by using separate bitmap images per size and per style. However, an enormous amount of storage space is required because each bitmap also should have its 3 fold magnified counterpart for printouts. Handling such an amount of bitmaps is not very practical. Another way of encoding is by means of vectors. (*a vector is simply a sequence of numbers which describe an object*). Characters are decomposed in short lines and ellipse arcs in this case. Advantage is sharp character images, even if enlarged where enlarged bitmaps cause 'steps' in diagonal lines. The drawing of vectored fonts is more difficult. Different font files are needed per style such as italic or bold. The 'arial' font is organised in this way.

In case of the Xfont a blend is chosen of the above methods. This method may be called a "vectorised bitmap". A character is first pictured in a big , monocolor, 96 * 96 bitmap. By reduction or magnification the desired fontheight is achieved.
Also, in case of reduction, the character is displayed with soft edges. With help of some math, the character may be pictured in various styles. Only one small font file is needed and the drawing process is relatively fast.

Character Layout
Figure 1. below shows the layout of the 96 * 96 bitmap.

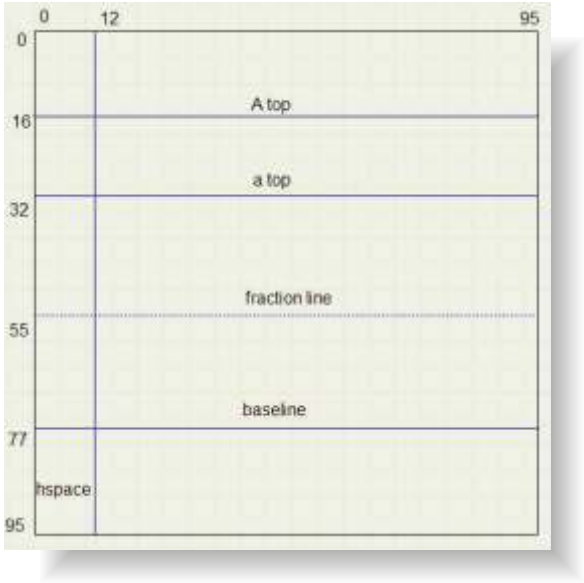


Figure 1: The layout of the 96 * 96 bitmap

The 'hspace' distance of 12 pixels takes care of the separation of characters in a line. Also the textcursor may be parked here.
Characters rest on the baseline. Room below the baseline and above A top provides separation between lines.
The ' - ' sign is positioned on the fraction line, the ' = ' is centered around it.

CHAPTER 71 PASCAL PROGRAMMING

THE XFONT PROJECT 2

Introduction
This article describes the XFont class, an implementation of the previously described XFont project.
To recall: this project allows home brew scalable fonts allowing for a variety of styles. Coding is very effective. A complete font needs less than 30 kBytes, including the styles. Drawing of characters is fast: typical over 10000 a second when the character height is 20 or less.

To implement the XFont in applications, a class is build around it. Also an exerciser program is designed to test the properties and methods of this new class.

This article also describes the exerciser. The XFont writes characters in a XBitmap. This is a TBitmap component with some extra's. The pixelformat is always 32 bits. But with only small modifications a "normal" TBitmap may be used.

XFont class options

- scalable characters, vertical pixel size from 10 to 96. Above 96 a loss of quality becomes evident.
- selectable font styles
  - bold
  - italic

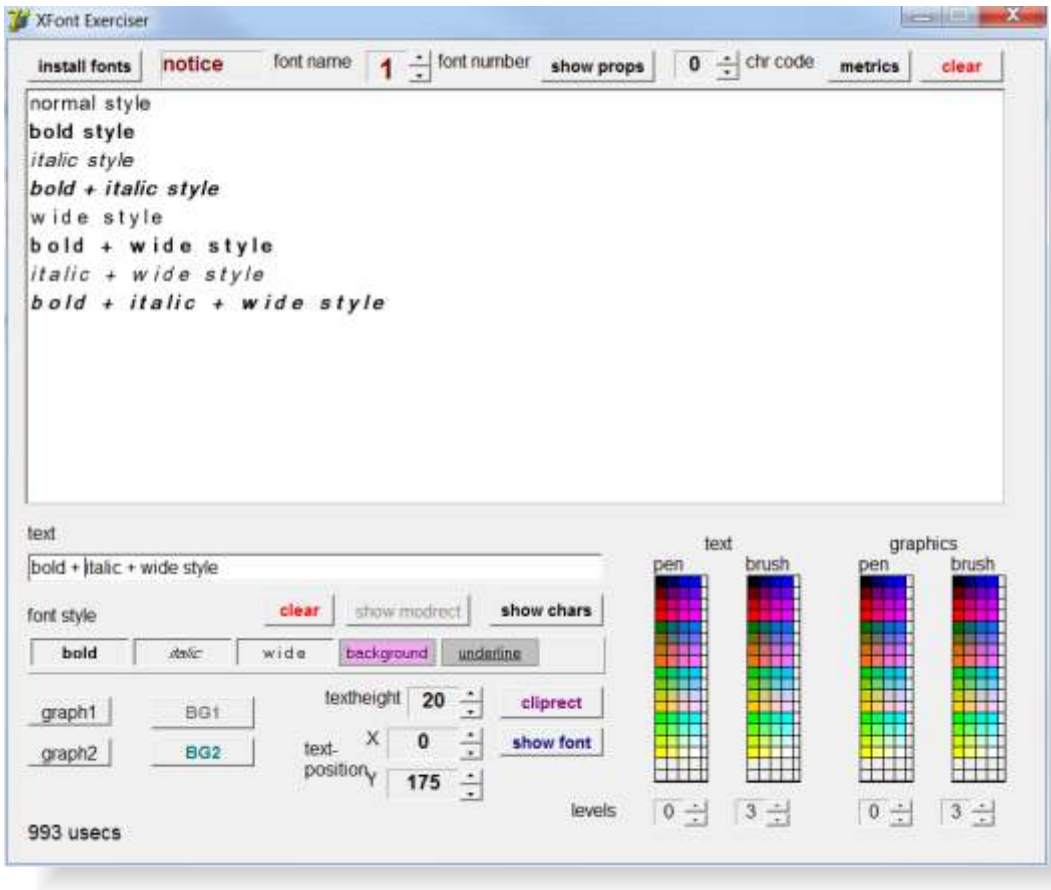


Figure 1: The XFont Exerciser program.

Introduction
The Xbitmap class adds following drawing options to the Bitmap class:

1. dash-dot lines with a penwidth of 1 to 32 pixels
2. lines with arrows at begin,end or both (*16 arrow types selectable*)
3. 4 levels (0..3) of drawing
4. xdot[x,y] method for fast drawing of dots in selected penwidth
5. improved stretchdraw method
6. clipping rectangle
7. fast direct access to individual pixels with the xpixel[ ] method
8. floodfill styles of 8 x 8 or 24 x 24 pixels, 16 styles selectable
9. modification rectangle: the area that changed during the last operation

CREATION AND PIXELFORMAT.
To create a xbitmap:

```
var myxbitmap : TXbitmap;
.........
begin
 myxbitmap := TXbitmap.create;
 with myxbitmap do
  begin
   width := 600;
   height := 400;
  end;
.....
```

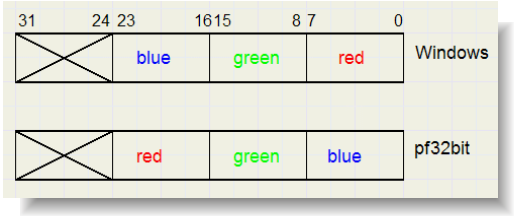


Figure 1: Windows vs pf32bit color format

Do not use the pixelformat property. Xbitmap only operates in the pf32bit mode and this is included in the setting of width and height. The extra properties and methods are added to the Bitmap class, not to the canvas.

In the pf32bit pixelmode, the position of the red and blue fields in a 32 bit word are swapped compared to the Windows color format. (see figure 1).

To trade the red and blue fields in a 32 bit word use:

```
function swapcolor(c: DWORD):
                              DWORD;
```

where c is the color.
For the `xpencolor` and `xbrushcolor` properties, the Windows format is used and the RB colorfields are traded internally. For the `xpixel[,]` method, the color is in pf32bit format for speed purposes. Internally, Xbitmap modifies pixels using pointers to the pixel location in memory. Therefore it needs the pointer to row 0, column 0 and the increment value of a pointer to the next row. These values are obtained by method xadjust when setting width or height, using the scanline[ ] property twice.

Loading the Xbitmap from memory:

```
with myxbitmap do
begin
  loadfromfile(filename);
  xadjust;   //restores the pf32bit property
  ...
end;
```

After a loadfromfile, xadjust must be called to insure that the pixelformat is pf32bit and to recalculate the pointer to (0,0) and the row increment value.

NOTE:
Pixels in a (x)bitmap are located in the computer memory and are therefore not visible. To make a (x)bitmap visible it has to be copied to a screen canvas (*such as a paintbox*).

CHAPTER 73 PASCAL PROGRAMMING

MICROSECONDS COUNTER

Introduction
For time measurements, the Delphi programmer may use a milliseconds counter. This counter variable is of type cardinal and delivers the expired time in milliseconds since Windows was activated. (*So, it overflows after 49 days*).
However, the accuracy is not very good, because the counter is not updated every millisecond but much less.
The following little program shows the behaviour of the timer:

```
procedure TForm1.Button1Click(Sender:
                              TObject);
var t : cardinal;
begin
 t := gettickcount;
 while gettickcount = t do;

 label1.Caption :=
            inttostr(gettickcount-t);
end;
```

In this case, label1 shows mostly a count of 16 and sometimes a count of 15 which indicates that the timer value is incremented only once per 15,xxx milliseconds. Also, most processes complete in less than a millisecond so, for accurate measurement, they have to be repeated hundreds of times.
The Pentium processor however has a 64 bit counter that is updated every clock cycle.

The RDTSC instruction (*code $0F,$31*) copies bits 0..31 to register EAX and bits 32..63 to register EDX.
This allows for accurate measurements of short execution times.

Problem however is, that the CPU clock speed is system dependable.
To calculate elapsed time in microseconds, first the clock frequency of the processor must be calculated.

This is accomplished by sampling the CPU's 64 bit clock register, wait 500 milliseconds using the milliseconds timer, then sampling the CPU clock counter again. The difference of the CPU clock values divided by the elapsed time in milliseconds * 10^{-3} makes the clock speed in MHz (megahertz).Once the clock speed is set, proces time is calculated by

- sampling the CPU clock
- execute the process
- sample the CPU clock again
- calculate the difference of the clocks
- convert this difference to the "double" floating point format
- divide this difference by the previously set CPU clock frequency

To implement such time measurements, two cases must be considered

1. the Delphi version supports 64 bit integers
2. the Delphi version does not support 64 bit integers

Support of 64 bit integers
See code listing on the next page.

Introduction
The Delphi programming environment already contains a dialog form for color selection. In some cases however, a simple component, not a pop-up form, for the selection of colors is needed.

This article describes such a simple component, which may also be integrated in complex, home built, dialog forms. The component source code is listed and the complete project may be downloaded.
How the component looks

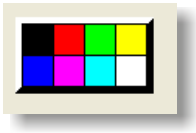


Figure 1: 8 color mode, horizontal, square = 20*20 pixels

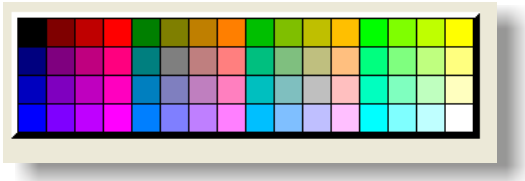


Figure 2: 64 color mode, horizontal, square = 20*20 pixels

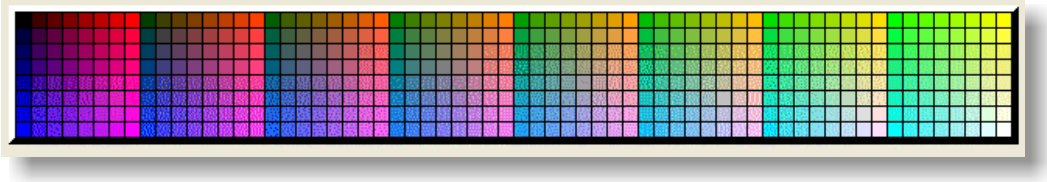


Figure 3: 512 color mode, horizontal, square = 10*10 pixels

Ancestor Class
The ancestor class is the `TGraphicControl` component, which is also the ancestor of the `Tpaintbox` component. Basically, the `davColorBox` is a modified paintbox.

Component Properties

- **direction**
  - **`cbHor`** ...horizontal orientation
  - **`cbVert`** ...vertical orientation
- **colorDepth**
  - **`cb8`** 1 bit per color, 8 selectable colors
  - **`cb64`** 2 bits per color, 64 selectable colors
  - **`cb512`** 3 bits per color, 512 selectable colors
- **border**
  - **`0...10`** number of pixels of edges
- **borderlight**
  - **`32 bit integer`** color of top and left edges
- **borderdark**
  - **`32 bit integer`** color of bottom and right edges
- **csquare**
  - byte, value 5 .. 40 edge length of each colored square in rectangle

CHAPTER 75 PASCAL PROGRAMMING

Introduction
Arrays are an efficient way to organize large amounts of similar data.
A graphic-user-interface uses many buttons with similar characteristics to start processes or select modes of operation.
So, why not organize buttons as arrays?
This article describes a Delphi Array-Button Component, which arranges buttons in a rectangle of columns and rows.

ARRAYBUTTON COMPONENT DELPHI

Characteristics

The component can have a maximum of 48 buttons `[0..47]` which may be organized as a single row, single column or any combination of rows and columns. All buttons have the same width and height.

Properties:

btnHeight the height of each button in pixels
btnwidth the width of each button in pixels
border the border around the total panel in pixels
btnEdge the width of the line around each button, 1 or 2
btnSpacing the spacing in pixels between the buttons
btnShape bs3D(*sharp corners*) or bsFlat(*rounded corners*)

Button Behaviour

A button can have the status:

stFlat not activated, in rest
stDown pressed by mousebutton or by program command
stHI mousepointer moving over button
stHidden button not shown

In addition, a button can operate in the following modes:

omMom `stDown` when pressed down by mousebutton, return to stFlat when mousebutton is released.
omPress `stDown` when pressed by mousebutton.
omToggle `stDown` when pressed down by mousebutton, released to stFlat by 2nd mousebutton press

Each button is assigned to a group, 0..15. In a group only one button can have the status stDown. So, pressing a button releases any stDown button with the same group number.

Responsibilities

The component takes care of:

- painting edges and background colors
- generating events when the status of a button changes

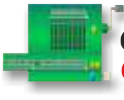
The application programmer should:

- paint the foreground image or caption according to button status

Colors

The property "color" is the color of the border and spacing between the buttons. Five colors in an `array[bcInactBG..bcLO]` of LongInt are used to indicate the status of a button.

bcInactBG
background color of inactive (*flat*) button
bcActiveBG
background color of active (*pressed*) button
bcFlat Edge color of flat button
bcHI Edge color when mousepointer over button
bcLO Edge color of pressed down button



CHAPTER 76 PASCAL PROGRAMMING

DRAWING SPHERES DELPHI

Introduction
This article explains how to draw 3D spheres in the Delphi programming language. An exerciser program is provided and also the complete Delphi project may be downloaded.

Surprisingly, no difficult math is required: only the Pythagoras lemma and linear functions are needed. Some controls, small arrows, are added to adjust light position and colors.
The article also describes how these arrows are programmed and how the sliding is accomplished. Color selection is done by 7 speedbuttons, which are placed in an array and are created at runtime. Figure.1 below is a reduced image of the 3D spheres generator program.

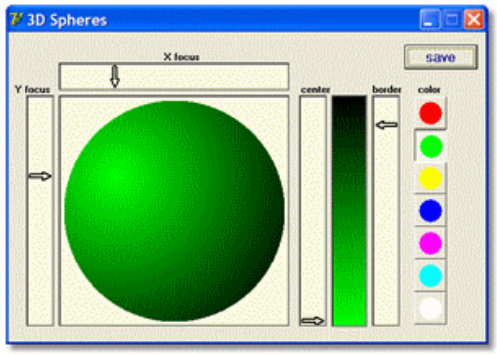


Figure 1: The 3D spheres generator program

The colors
The colors on the sphere change from the point of highest intensity to the borders. This suggests the 3 dimensional effect. It is unnatural to change the colors from, say, blue to red. Not the color itself changes but the intensity does. Therefore, at first, we only define the intensity, which ranges from 0 (dark) to 255 (bright).

Next , a 3 bit colorcode defines the colors that participate. colorcode bit -0- enables red, bit -1- enables green and bit -2- enables blue. At design time, the color code is stored in the tag property of the corresponding speed button (*figure 2*).

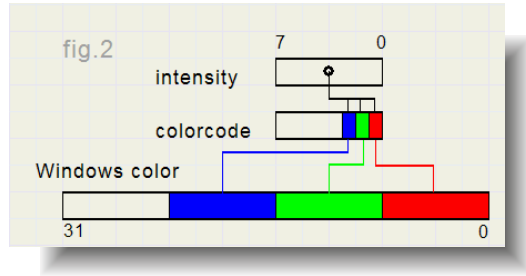


Figure 2: Color code is stored in the tag property

Drawing the sphere
This is done in bitmap Smap. (*S - Sphere*)

All pixels of the bitmap are addressed left to right, top to bottom.
For each pixel (x,y) a check is made to find its position: inside or outside the circle (*Figure 3*).

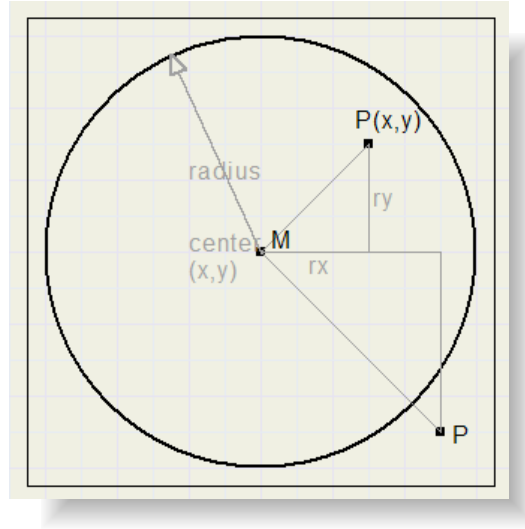


Figure 3:pixel (x,y) check to find its position

You can download the accompanying files: spheres.exe
The complete project and code is available for download spheres.zip

Introcdution
On the web I found the "leap-frog puzzle" , an educational game for kids.

Figure 1: Leap-frog puzzle

Two groups of 3 frogs each sit opposite each other, one empty space between them. The challenge is to have the frogs trade positions by selecting smart jumps.

This article describes a computerprogram that searches for (and finds) solutions. For a more general approach, the number of frogs may vary from 2 * 1 to 2 * 10.

I leave it to the reader to draw beautiful jumping frogs. Instead I focus on the programming and some math behind. So, the "left" type frogs I call simply X and the "right" frogs are named Y.

This is the starting board state:
X X X - Y Y Y

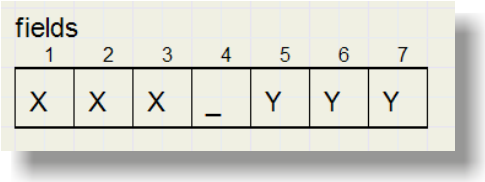


Figure 2: Situation at game start

The rules for jumping are:

- a X frog only jumps to the right
- a Y frog only jumps to the left
- a frog jumps to the next open field or
- over the next frog to the next open field

So a jump is either 1 or 2 fields.
The puzzle is solved when the board state is Y Y Y - X X X

Coding the game

A position on the board I call a "field".
A field may be empty, occupied by aX- or by a Y-frog.

```
type Tfrog = (none, frogX, frogY);
 var board : array[1..7] of Tfrog;
//holds the board state
```

Note: array[1..7] is for 2 * 3 frogs. For k frogs : array[1..2k+1]

For each field a maximum of one jump is possible.
However, it is more convenient to define a jump by the origin- as well as the destination field.

```
type Tleap = record
   org , dest : byte;
//start and end field of frog leap
                end;

var leaps : array[1..20] of Tleap;
//all leaps
    leapNr  : byte = 0;
//last leap
```

Introduction
In this article we explore two IEEE floating point formats implemented by Intel. The program floatformats accepts floating point numbers and shows the internal bit representation.
Below is a picture of the program:

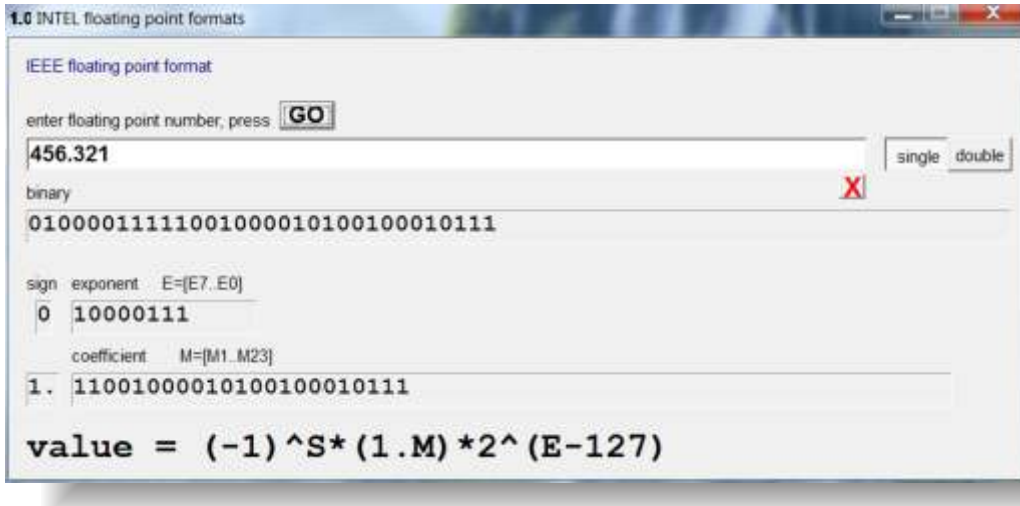


Figure 1: Floating point number 456.321

Floating point: the scientific notation

Floating point numbers are numbers that, other than integers, have a decimal point. Examples of floating point numbers are 0.123, 12.09, -431.987

In the "scientific notation", numbers are normalized to one digit left of the decimal point. A power of 10 is then postfixed to maintain the correct value.
So 0.123 becomes $1.23 * 10^{-1}$
12.09 becomes $1.209 * 10^{1}$
-4321.987 becomes $-4.321987 * 10^{3}$

In binary, a number in scientific notation will always have the format
$1.xxxxxxx * 2^{yyyy}$
where x...x is called the mantissa, y...y is called the exponent. The IEEE standard saves floating point numbers in the normalized scientific notation.

In the IEEE floating point format, the 1 left of the decimal point is not stored, but automatically inserted by the processor.
There are two formats:

32 bit (short, Delphi name: "single")
64 bit (long, Delphi name: "double")

32 bit

Figure 2: 32 bit

The decimal point (.) is placed left of M1.
Bit M ^{n}represents the value 2^{-n}
The M bits together are called "mantissa".

Bit 31 is the sign bit of the mantissa.
If "1" the mantissa is negative, if "0" the mantissa is positive.

CHAPTER 79 ICT

Introduction
This chapter describes my Delphi project spic for the compression of pictures.
The images are stored in bitmaps. From the pixels of this bitmap, spic generates a file of type .pic which is much smaller in byte count then the .bmp file holding the original bitmap. Well known formats for compressed images are : .gif , .jpg and .png. Later in this article I include a comparison of them with the .pic format.

AN ALGORITHM FOR IMAGE COMPRESSION

The algorithm

A bitmap with randomly colored pixels will be hard to compress. Compression is possible only if the picture has some regularity: repeating patterns, areas with less colors. I tested a lot of methods and so, in an organic way, the algorithm evolved that works best for picures like the naval battle.
Below is a reduced picture of spic at work:

Figure 1: The naval battle painting was reduced to 16.07%

In general, two type of images may be distinguished: photographs and geometric figures. The first category has fluent colors and no sharp boundaries, the second have lines and rectangles filled with a single color. My spic project is aimed at photographs and paintings. For geometric pictures other approaches would yield better results however, the `.pic` files are far from bad.

Surprisingly enough, only a limited number of simple commands remained.

Colors

A bitmap in true color format has 24 bit colors, which are stored in the pf32 bit format, as seen in figure 2.

CHAPTER 80 ICT

EXPONENTIAL CURVE FITTING

Introduction
This article describes the exponential curve fitting method implemented in Graphics-Explorer,
my equations grapher program.
New is an exerciser program allowing step by step observation of the curve fitting process. The curve fitter calculates the best fitting exponential function given a set of points.

This function is

$$y = a.b^x + c$$

where a,b,c are called the parameters.

Contents

Exerciser description
Figure 1 is a reduced picture of the exerciser at work.

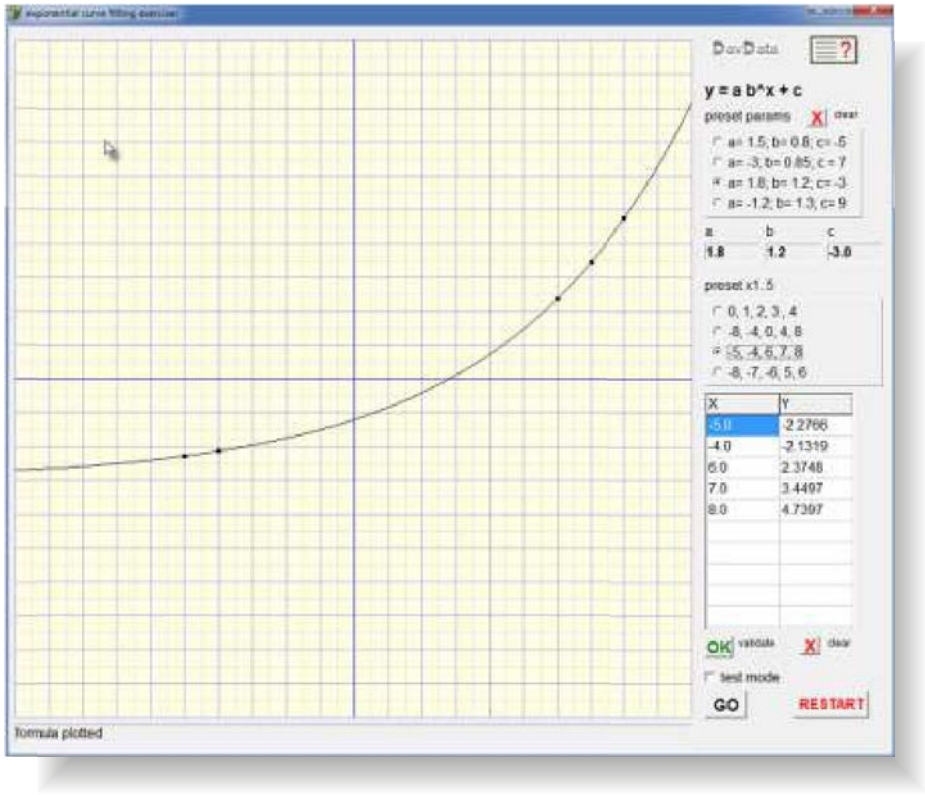


Figure 1: The Exerciser at work with:

a,b,c	calculated parameters. (*black: valid; red: invalid*)
preset params	click radio button to preset parameters
preset x1..x5	click radio button to preset points
x, y	list of points, maximum is 10
[OK] validate	click to check/organize points prior to calculation
[GO]	start calculation of parameters a,b,c
[ ] testmode	mark to watch step by step calculation

Introduction
This article describes how a polygon is dissected into triangles. Triangulation is necessary if the polygon has to be colored or when the area has to be calculated. The painting or calculations may then be performed on the individual triangles, instead of the complete and sometimes complicated shape of the polygon.
Polygons may appear in different shapes.

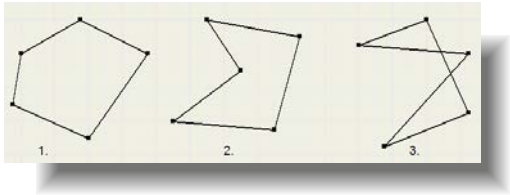


Figure 1: Where
1. is a convex polygon, each angle is less than 180 degrees.
2. has an "inside" angle, bigger than 180 degrees.
3. has intersecting edges.

In this application type 3. is illegal. The program will raise an error message.

Types 1. and 2. , whatever complicated, can be broken down into triangles. These individual triangles then may be colored or the areas may be summed.

A polygon is made up of sequential points interconnected by lines. There is an area inside- and an area outside the polygon. When decomposing the polygon into triangles, the biggest problem is to classify an angle as "inside" or "outside".
In case 1. before, each angle is "outside" and 3 successive points always are angles of a triangle which is inside and part of the polygon. Triangulation may also be accomplished by drawing lines from one point to all others.
In the case of polygon 2. however, the "inside" angle must be recognized to avoid coloring.

To distinguish "outer"- and "inner" angles I use some basic vector geometry. The details will be introduced later in this chapter

Outer- and Inner angles
A line (*edge*) has a starting and ending point. In mathematics it is called a vector, because more than one number is required to describe it. See figure 2. (*2 numbers, in 2D geometry*).
The next figure shows line l with vector AB. An arbitrary point on line l may be characterized by a single number, I call this factor f. Point A, the starting point, corresponds with f = 0. Point B, the ending point, corresponds with f = 1. Right of B (*the forward extension of AB)* has points that correspond to f > 1.
Left of A (*the backward extension of AB*) has points that correspond to f < 0..

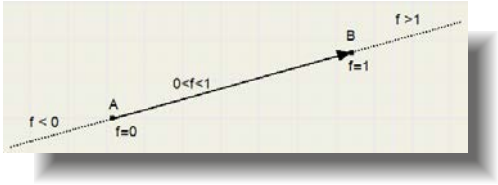


Figure 2: A line has a starting and ending point

In case of two vectors (*red and blue, see figure 3 on next page*) there is an intersecting point S. Since S is on both the red and the blue vector, we may calculate the f values (*f1 for red, f2 for blue*) for S. f1 and f2 give an indication of the relative position of the vectors.
Red and blue may be parallel or coincide. In that case there is no intersection.
A flag fvalid is set false in this case.
Using this vector geometry we are able to distinguish between inner- and outer angles in case of simple polygons. Look at the figure 3: we examine angle B of figure 4 (next page).

Introduction
In geometry, a vector is a line from one point to another in a plane.
This article describes how to obtain the direction (*in degrees*) of a vector.
see figure 1 below

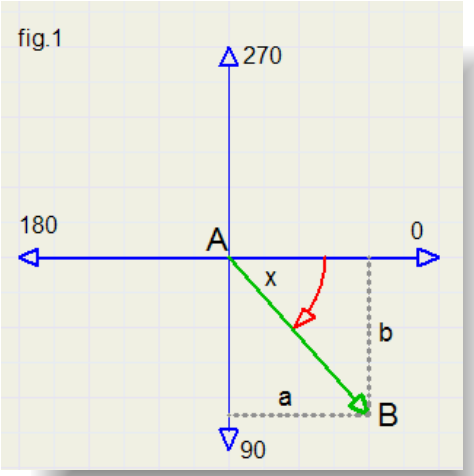


Figure 1

Directions are
0 right
90 down
180 left
270 up
All measured in degrees.

We notice vector AB = (a,b) and the tangent equals tan(x) = b/a

NOTE:
Positive y direction is down for coordinates on the screen.

Our objective is to calculate x for any vector.

Tangent
The tan function supplies the `b/a` ratio for a given value of angle x.....`tan(x) = b/a`. The inverse tangent function arctan supplies the angle x for a given value of `b/a`.....`x = arctan(b/a)`
However, x is in radians.
Knowing that 2*Π radians = 360 degrees, we simply multiply by 180/p to convert radians to degrees.
Figure 2 shows the arctan function.
We actually plot (*Graphics-Explorer*) the equation `y = atan(x)` where x is ratio b/a and degrees are selected instead of radians.
The atan function result ranges from -90.....+90 degrees, which is not what we want. Also, for straight up or down vectors where `a = 0`, an exception (*error*) is generated because of division by zero .....{ *atan(b/0)* }

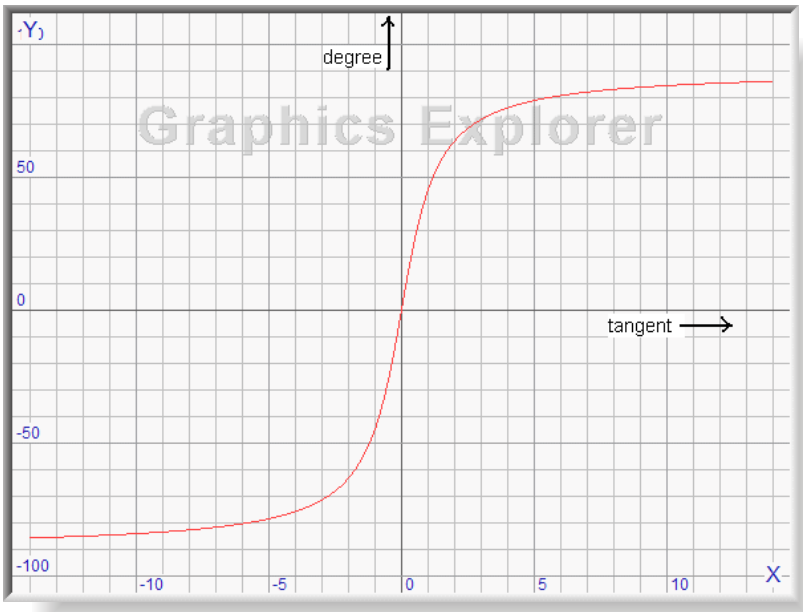


Figure 2

Introduction
This document describes how my computer program CONNECT4 (*version 4.0*) calculates the best move.

My work on CONNECT4 started in 2001. The first ideas for an algorithm came up while working on solutions of Peg-Solitaire.

After the first working example I kept wondering how to speed up the search process to look further ahead within acceptable time and how to make the computer a more interesting player. These additions to the basic algorithm will be described as well.

In the current version 4.0 the search process has two distinct steps:
1. A qualitative analysis of the board, looking just one move ahead. This analysis results in a recommended move for
2. A quantitative (*brute force*) approach, which tries to find a better move than the one recommended.

Contents

The result of step 1 is influenced by the selected Strategy.
In step 2, the search depth is `2*level+1`.

The Game

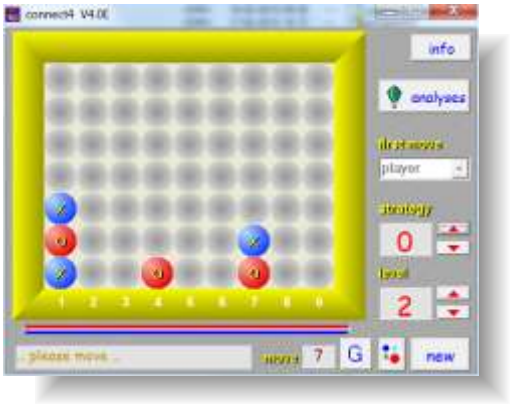


Figure 2: A new gameplay

My version of CONNECT4 is single player, you play against the computer.
The board has 9 columns and 7 rows, resulting in 63 fields.
Computer and player alternately place a ball of their color in a field. The computer always plays with red, the player with blue. The columns are filled bottom-up, so the balls are actually dropped down a column. The winner is the first to achieve a horizontal,vertical or diagonal line of 4 balls of his color.

Coding the board and moves
The game has 63 fields, each of which can be can be:

```
- empty  (code 0)
- red    (code 1)
- blue   (code 2)
```

`Array BORD[1..9,1..7]` holds the boardstate
(*BORD is Dutch for board as you guessed).*

`BORD[2,3] :=` `1` places a red ball in column 2, row 3.
`BORD[4,7] :=` `0` removes the ball of column 4, row 7.

CHAPTER 84 ICT

Introduction
Information can be stored in an analog or digital form. The big advantage of digital storage is the possibility to manipulate the data using mathematical functions. This article handles a simple algorithm to resize (enlarge or reduce) computer pictures. Data formats and mathematics are discussed in detail. The Delphi-implementation and full source-code listing are also included.

Data formats

A computer picture (*or screen*) is made up of dots which are called pixels. A colored pixel is a mix of three colors: red, green and blue. See figure 1.
Using realistic colors, the intensity of each color is a number ranging 0..255. 0 means zero intensity, 255 (*or hexadecimal ff*) indicates the highest intensity.

This is what a (*very enlarged*) pixel looks like:

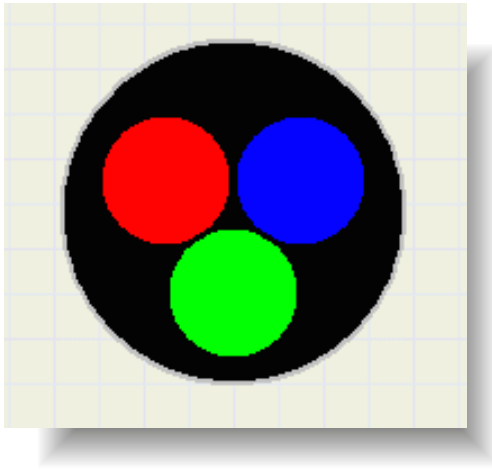


Figure 1: A very much enlarged pixel

maximum intensity of all three colors produces white.

BITMAP RESIZE ALGORITHM

Internally, Windows stores the data of each pixel in a 32 bit word:

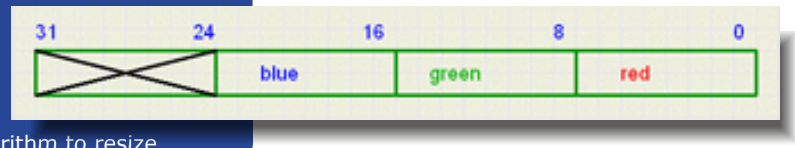


Figure 2:Pixel data stored in a 32 bit word

The red information is stored in bits 0..7, green in 8..15, blue in 16..23. Bits 24..31 are not used. A picture is a 2-dimensional table of pixels.
In Delphi, this table may be part of a `Bitmap` component. This component offers the choice of several data formats and has procedures to manipulate the pixels (*drawing lines, filling areas, copying ...*). Bitmaps may be copied to a `paintbox` to be displayed on the screen. Pixeldata in the bitmap is addressed by using the [x,y] coordinates.
Image below shows a bitmap with some coordinates. A pixel is pictured as a square.

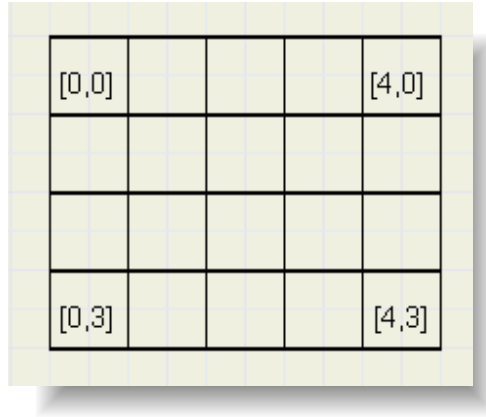


Figure 3

Commonly used pixelformats are: 16 bits, 24 bits or 32 bits.

CHAPTER 85 ICT

A NON RECURSIVE FLOODFILL ALGORITHM

Introduction

This is a new floodfill article, including a much faster exerciser program. Floodfill is a method to fill a random shape with a pattern. This floodfill method is implemented in my TXBitmap class. TXBitmap is a Bitmap with several new features such as

- 4 drawing levels
- a cliprect
- 16 floodfill patterns

Using drawing levels, a lower level pen cannot overwrite a higher level pixel. The highest level is 0, the lowest is 3.

Floodfill boundary is formed by pixels having level 0 or 1 (the 2 highest levels). Pattern drawing is at level 2. The level of a pixel is stored in blue bits 0,1.
The cliprect is a rectangle on the screen. Painting outside the cliprect is not possible.
The floodfill algorithm here presented is non recursive. This is an advantage because, while filling large shapes, stack boundaries cannot be violated.

Below is a reduced picture of the floodfill exerciser at work.

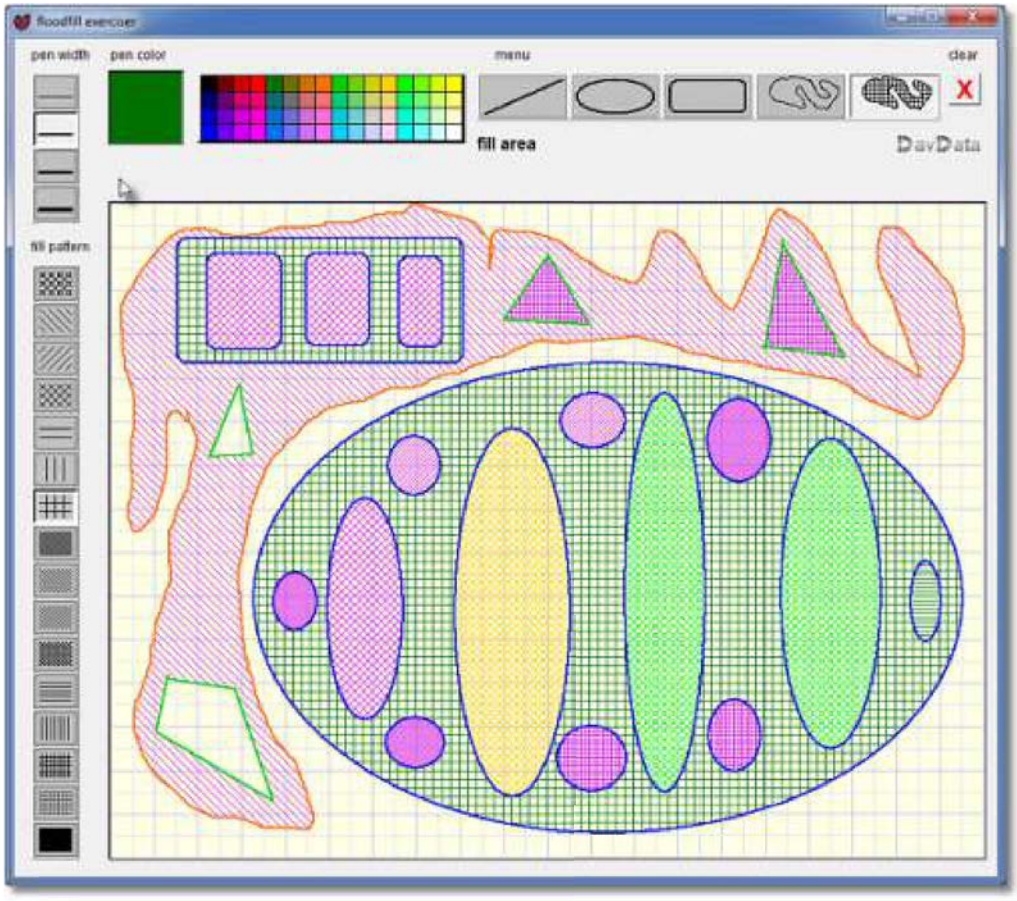


Figure 1: The floodfill exerciser at work.

Introduction

This article describes the translation of several types of mathematical equations into a sequence of basic arithmetic operations. Such a process is also known as "parsing" and is necessary when plotting functions or using them in spreadsheets. Years ago, I designed the algorithm for my equations grapher **Graphics-Explorer**. **(see chapter 36)**
However, at a second glance, this Delphi source code is rather hard to read.

Formula Translation

So I have rewritten the code into a more comprehensible form. The Delphi-7 project consists of 3 units:

- *unit1*:
  form with buttons, TEdit for entering formulas and a paint-box to show graphics and tables
- *xlate*:
  holds the code for the translation and the calculations
- *eqdrawer*:
  holds procedures to draw the different type of equations using the xlate unit

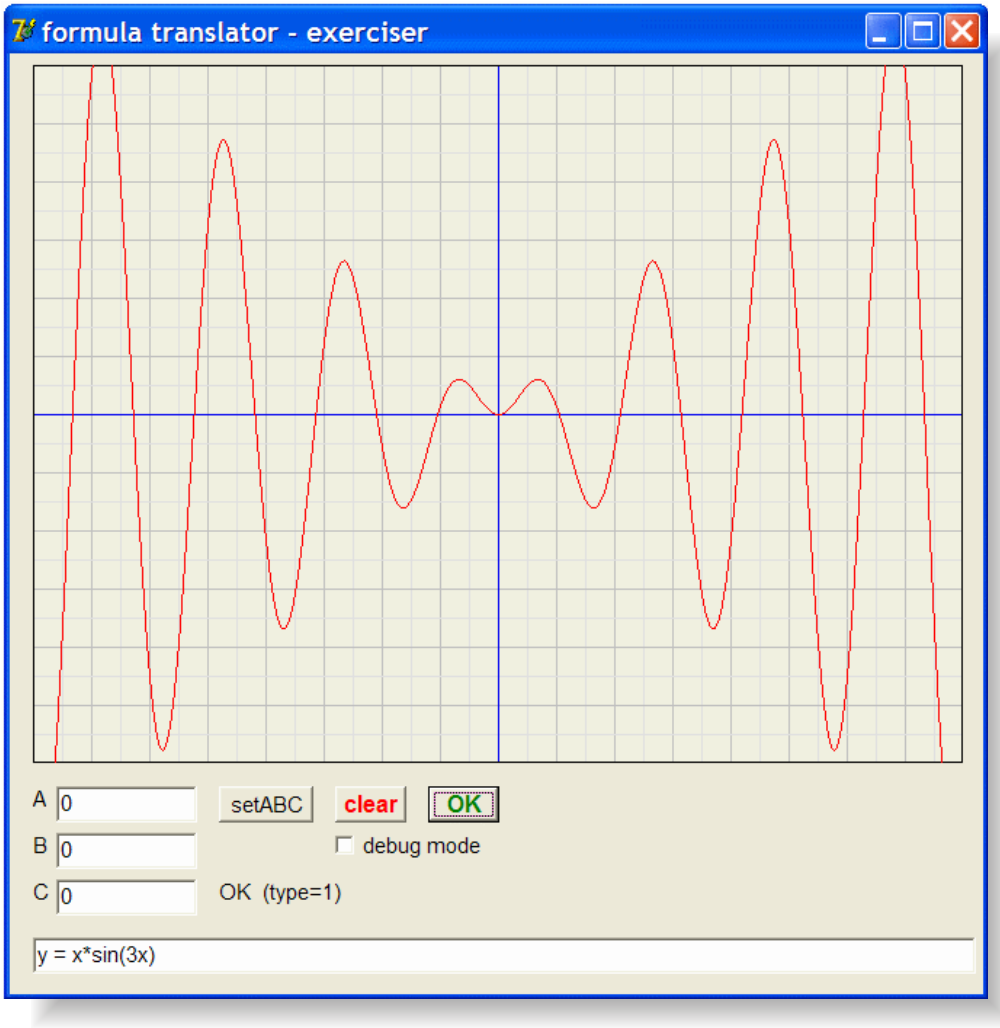


Figure 1: The exerciser, showing the function y = x*sin(3x)

CHAPTER 87 ICT

Introduction
In a previous chapter „formula translation" I discussed how functions or equations are dissected into basic arithmetic operations allowing the value to be calculated.
The result was the Director Table, which is a list of basic arithmetic operations sorted from high to low priority. To plot the functions I included unit eqdrawer and its code is described below.

NOTE:

- a `formula` has the form **... x ... y ...** where **x** and **y** are **variables** and ..... are **operators**
- an `equation` has the form **....x....y = ...x.....y**
- a `function` has the form **y = ...x.....**

So, a function is a special form of an equation. Functions are used widely because, yielding only one result, they may be used in formulas.
An example is `y = 3sin(x)`
In the **fxlate** project, 4 type of functions or equations are supported. The options to draw the graph are intentionally kept very simple because the only purpose here is to illustrate the proper translation. The coordinate system is in paintbox1 on form 1, the size is 640 * 480 pixels. The origin of the coordinate system is at pixel position (320,240). The domain of x is -8 ... + 8, the domain of y is -6...+6. The scale is fixed to 40 pixels per cm. In general, equation plotting is calculating consecutive (x,y) number pairs and connecting them by straight lines. Keep in mind, that certain values of x, y result in arithmetic errors if we try to calulate the square root or logarithm of a negative number or if we divide by zero.
The `calculate(var OK : boolean)` procedure will set OK to false in this case and the plotting software must act accordingly.

EQUATION GRAPHER DESCRIPTION

Supporting functions

```
function y2pix(y : double):longInt;
// convert y double value to screen
pixel position

function x2pix(x : double):longInt;
//convert x double to screen pixel position

function pix2x(p : longInt):double;
//convert screen pixel x position to x coordinate

function pix2y(p : longInt):double;
//convert screen pixel y position to
```

Below I discuss the drawing procedure per function type. Refer to the **eqdrawer source code** for more details.

Type1 functions {y = ...x.....}
Variable `i` is incremented from 0 to 639. `i` is converted to a `x` value by calling `pix2x(i)`. Then `setX(x)` sets x and calculate (`valid`) is called to calculate y. `y := getY` yields the calculated value of y. Counter `vcount` represents the number of valid points received. It counts 0,1,2,2,2... At value 1, a `moveto(i, y2pix(y))` takes place, at `vcount = 2` a `lineto(i, y2pix(y))` takes place. If `valid = false`, `vcount` is reset to zero. Also a check is made for the difference of two consecutive y values: if too large, painting is suppressed. This avoids painting asymptotes as in `y = tan(x)`.
This asymptote suppression is somewhat primitive, more sofisticated but more complicated procedures are possible.

Type2 functions {x = ...y.....}
Basically the plotting is the same as for type1, but x and y are traded.