

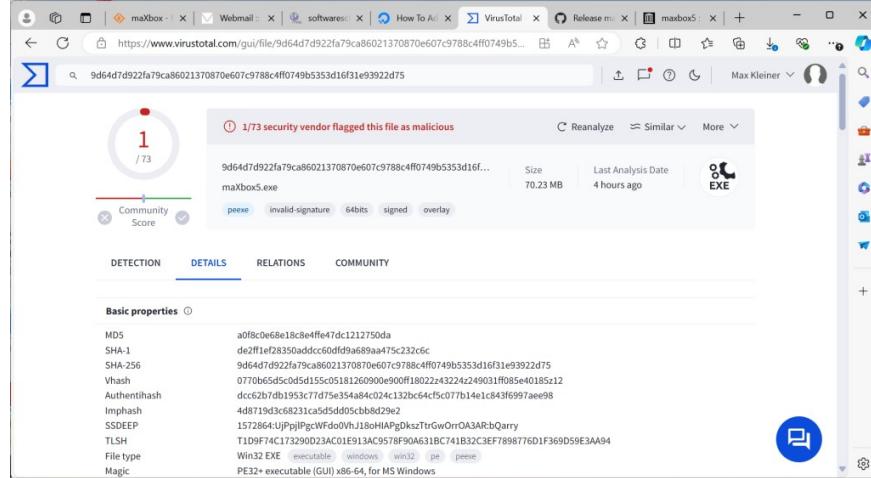
Beyond the Obstacle

Jul 22, 2024 code, maxbox, script

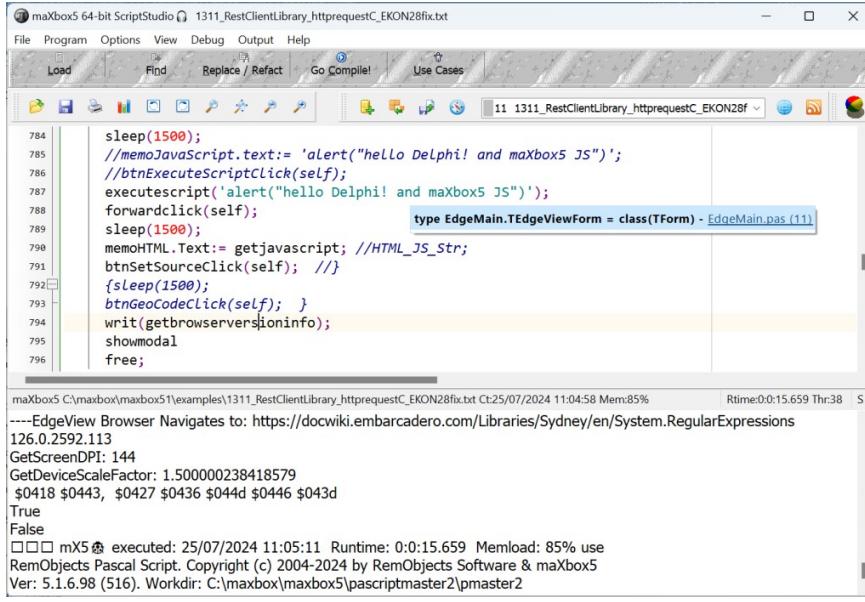
In this article I want to show a few aspects with **perplexity.ai** and works with maXbox code:
maXbox is a script tool engine, compiler and source lib all in one exe to design and code your scripts in a shellbook! It supports **Pascal**, **Delphi** (VCL), **Python** (P4D), **PowerShell** and **Java Script** (Edge WebView).

https://sourceforge.net/projects/maxbox/files/Examples/13_General/1337_15_puzzle_game12.pas/download

Welcome to WordPress and maXbox5! This is a sample post. Edit or delete it to take the first step in your blogging journey. To add more content here, click the small plus icon at the top left corner. There, you will find an existing selection of WordPress blocks and patterns, something to suit your every need for content creation. And don't forget to check out the List View: click the icon a few spots to the right of the plus icon and you'll get a tidy, easy-to-view list of the blocks and patterns in your post.



maXbox5 latest



```

784 sleep(1500);
785 //memoJavaScript.text:= 'alert("hello Delphi! and maxbox5 JS")';
786 //btnExecuteScriptClick(self);
787 executescript('alert("hello Delphi! and maxbox5 JS")');
788 forwardclick(self);
789 sleep(1500);
790 memoHTML.Text:= getjavascript; //HTML_JS_Str;
791 btnSetsourceclick(self); //}
792 {sleep(1500);
793 btnGeoCodeclick(self); }
794 writ(getbrowsersversioninfo);
795 showmodal;
796 free;

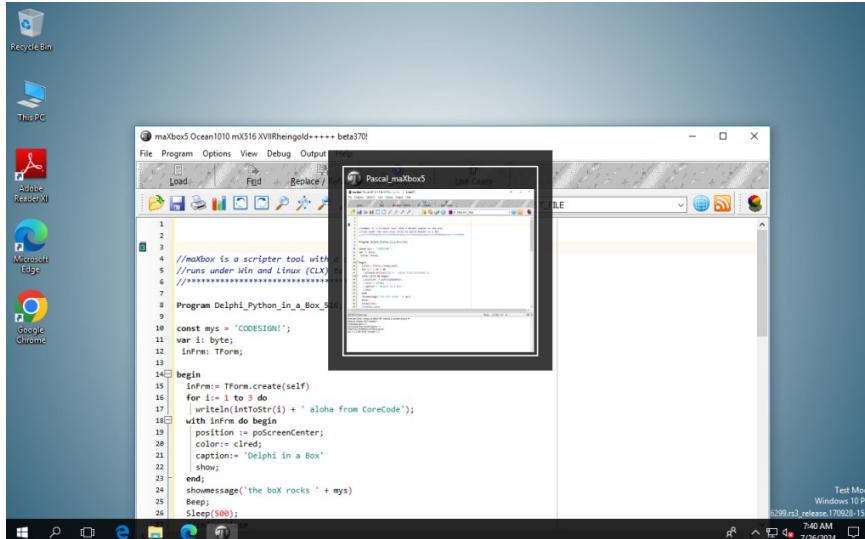
```

maXbox5 C:\maxbox\maxbox5\examples\1311_RestClientLibrary_httprequestC_EKON28fix.txt Ct25/07/2024 11:04:58 Mem:85% Rtime:0:0:15.659 Thr:38 S
---EdgeView Browser Navigates to: https://docwiki.embarcadero.com/Libraries/Sydney/en/System.RegularExpressions
126.0.2592.113
GetScreenDPI: 144
GetDeviceScaleFactor: 1.500000238418579
\$0418 \$0443, \$0427 \$0436 \$044d \$0446 \$043d
True
False
□□□ mX5 executed: 25/07/2024 11:05:11 Runtime: 0:0:15.659 Memload: 85% use
RemObjects Pascal Script. Copyright (c) 2004-2024 by RemObjects Software & maXbox5
Ver: 5.1.6.98 (516). Workdir: C:\maxbox\maxbox5\pascscriptmaster2\pmaster2

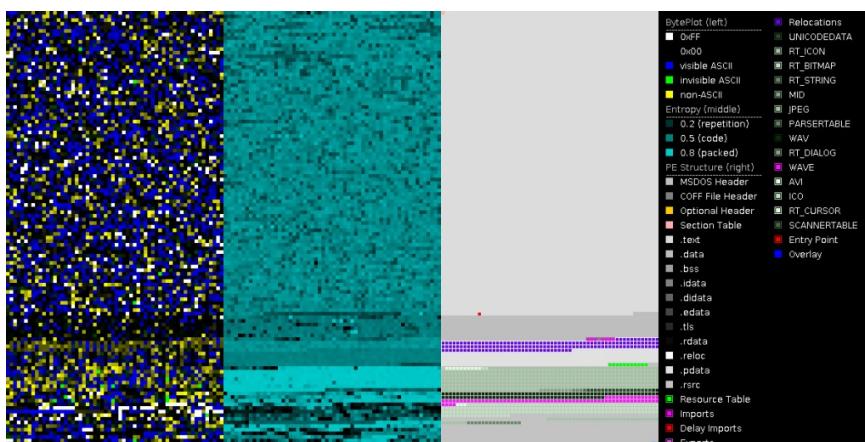
on the edge with path o logic

Path o Logical

Gather signals to trace your threat. Our tools extract suspicious signals such as OLE VBA code streams in Office document macros, invalid cross reference tables in PDFs, packer details in Windows Executables, intrusion detection system alerts triggered in PCAPs, Exif metadata, authenticode signatures and a myriad of other properties. Use these properties as IoCs to hunt down badness in your network.



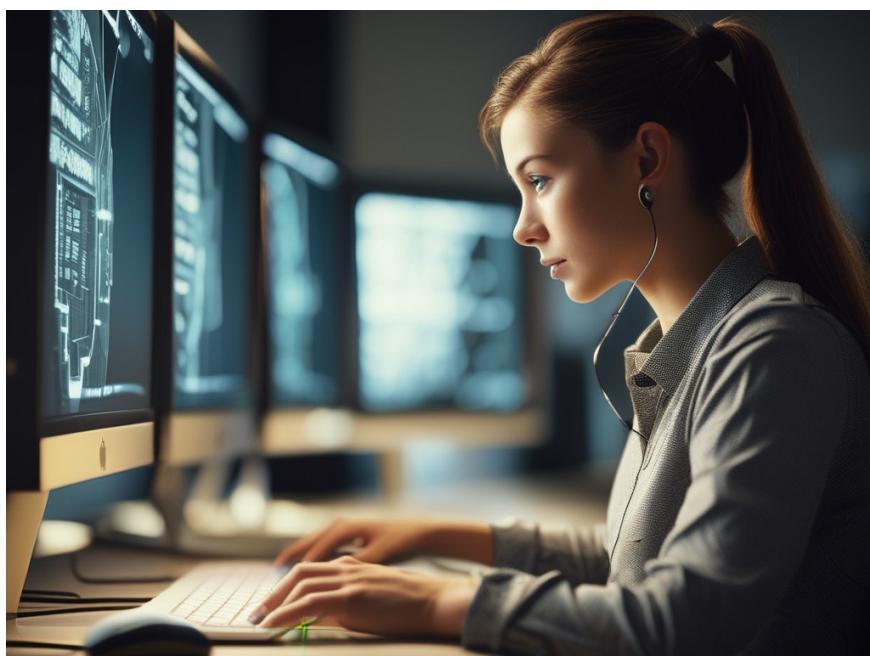
Hybrid



Visualize EXE



S.A.C.M. Elsässische Maschinenbauanstalt Graffenstaden C-Kuppler (2x)
– Baujahr 1900

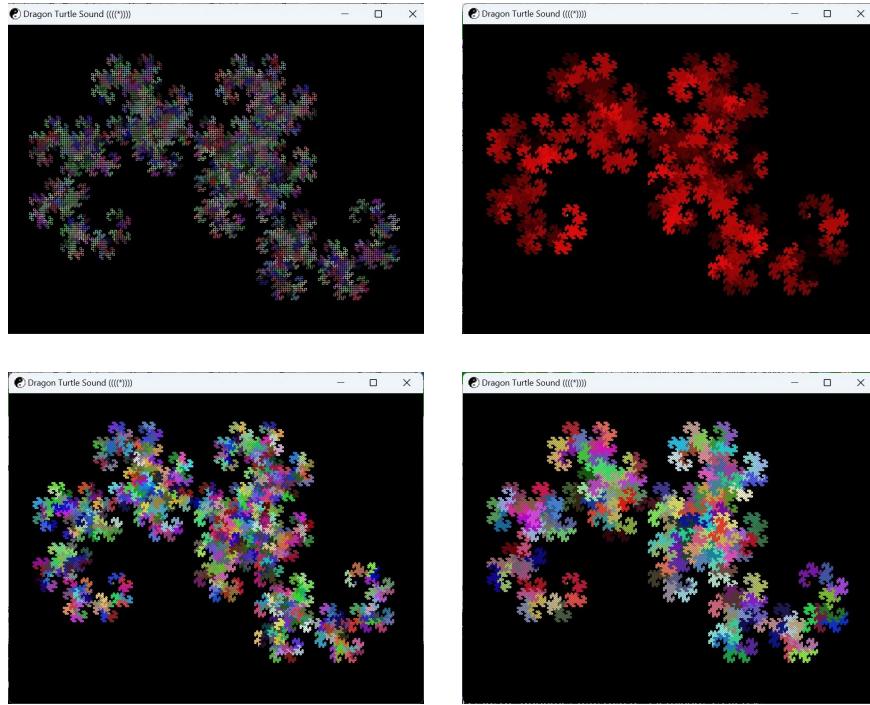


AI prompt: girl is programming with maxbox5 in a screen



coding in maxbox with maxbox logo

sign of all5



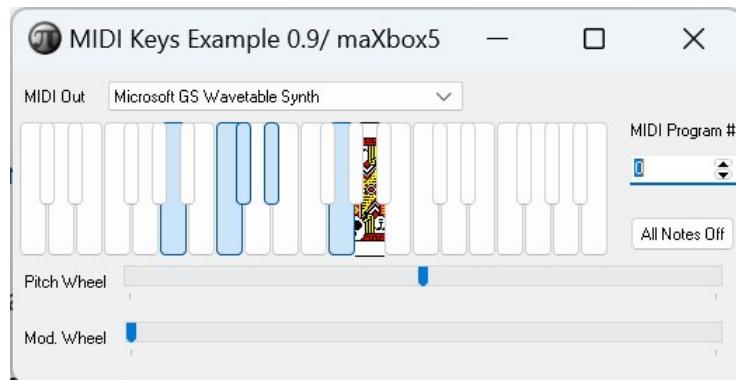
Dragon Curve



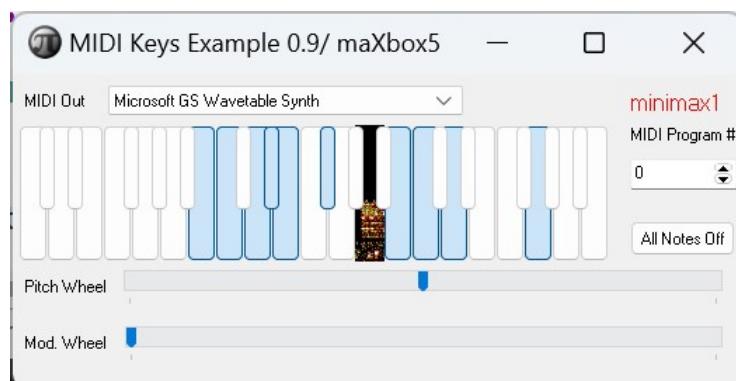
Obstacles V

MIDI Sound Keyboard

May be the is the smallest keyboard on planet earth:



minimax1

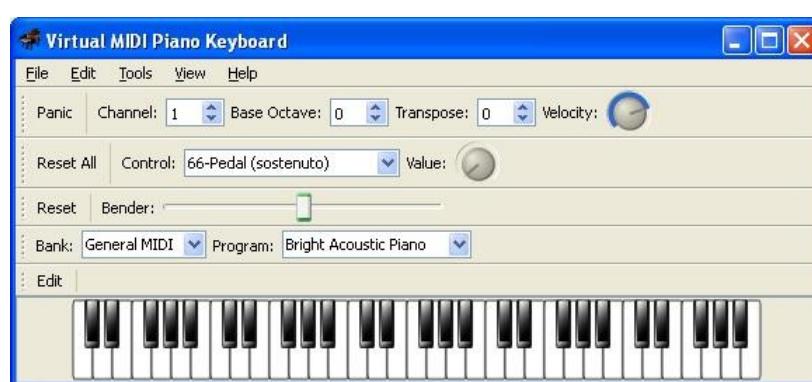


minimax1b

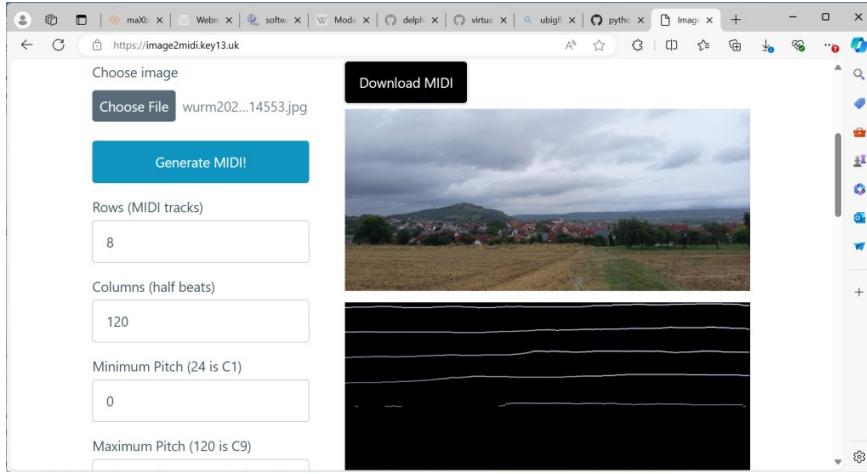
```

1  Const
2    HalftonesPerOctave = 12;
3    MiddleA           = 440.0; // Hertz
4    MidiMiddleA       = 69;    // A3 = 440 Hertz
5    Digits = 6;
6    MIDIFreqMax = 127.99993896;
7    FreqHertzMin = 8.17579892;
8    FreqHertzMax = 13289.70346552;
9
10   function Hertz(MIDINote: Extended): Extended;
11   begin
12     result:= TwoToY((MIDINote - MidiMiddleA) / HalftonesPerOctave) * Middle
13   end;
14
15   writeln(MIDINoteToStr(note+69));
16   writeln(floats(Hertz(69)));
17   A3
18   440.00000000000064
19   Microsoft GS Wavetable Synth

```



MidiMath



Listen that picture

http://www.softwareschule.ch/examples/wurm20240818_114553_f#_harmonicminor.mid

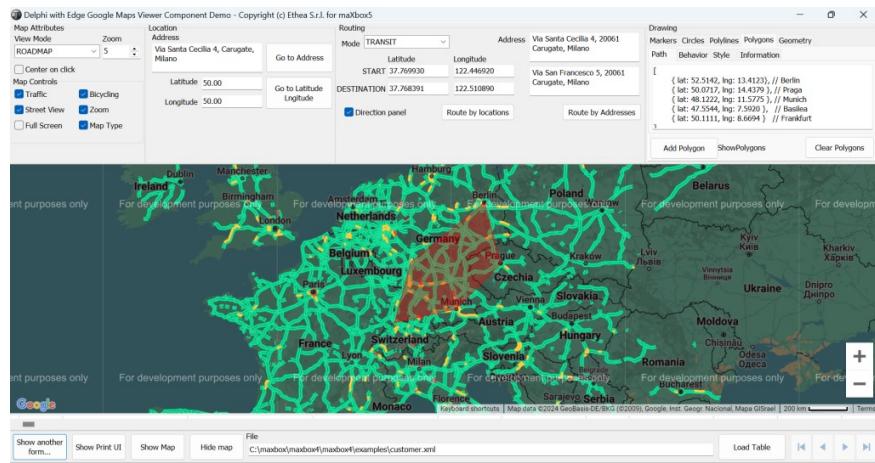
Reference Chart Of Common Chords

Root	Major	m	+	6	m6	7	m7	ma7	o	9
C	C	Cm	C+	C6	Cm6	C7	Cm7	Cma7	C ^o	C9
C#	D	D	D+	D6	Dm6	D7	Dm7	Dma7	D ^o	D9
D	D	Dm	D+	D6	Dm6	D7	Dm7	Dma7	D ^o	D9
D#	E	E	E+	E6	Em6	E7	Em7	Ema7	E ^o	E9
E	E	Em	E+	E6	Em6	E7	Em7	Ema7	E ^o	E9
F	F	Fm	F+	F6	Fm6	F7	Fm7	Fma7	F ^o	F9
F#	G	G	G+	G6	Gm6	G7	Gm7	Gma7	G ^o	G9
G	G	Gm	G+	G6	Gm6	G7	Gm7	Gma7	G ^o	G9
G#	A	A	A+	A6	Am6	A7	Am7	Ama7	A ^o	A9
A	A	Am	A+	A6	Am6	A7	Am7	Ama7	A ^o	A9
A#	B	B	B+	B6	Bm6	B7	Bm7	Bma7	B ^o	B9
B	B	Bm	B+	B6	Bm6	B7	Bm7	Bma7	B ^o	B9

Note Reference



at wylerbad by the end of august 2024 jumping jack max – the splash



TEE Track

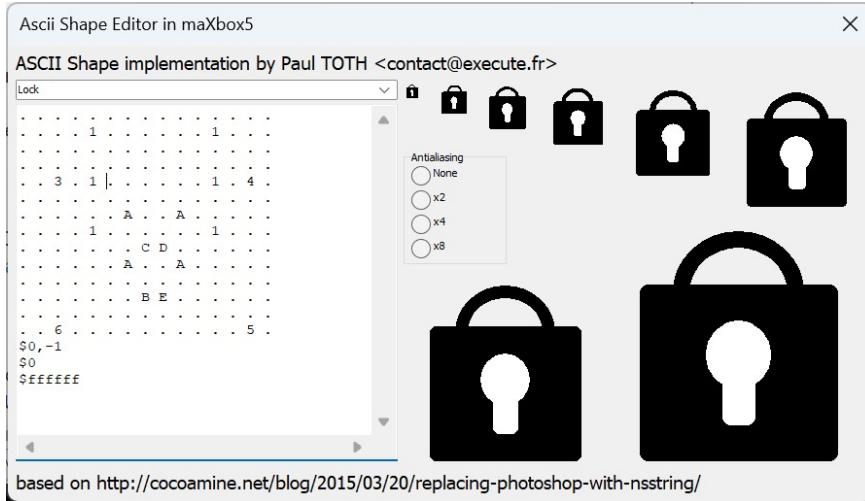


cool tool for programming

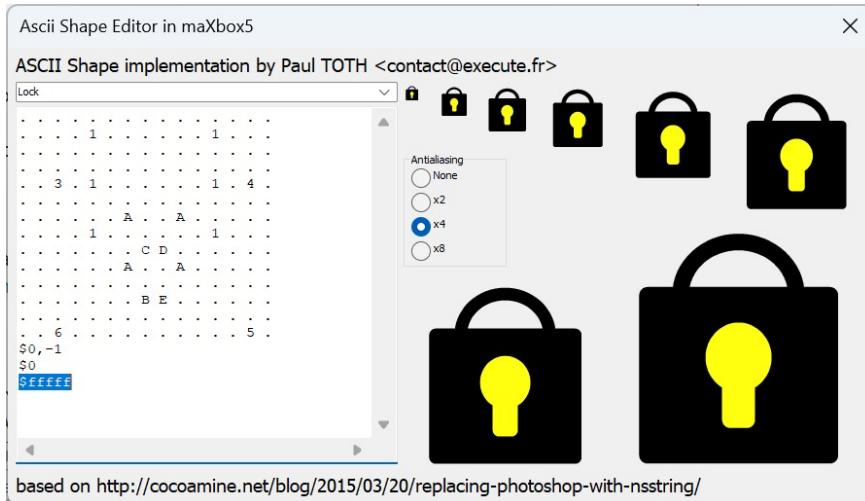


IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

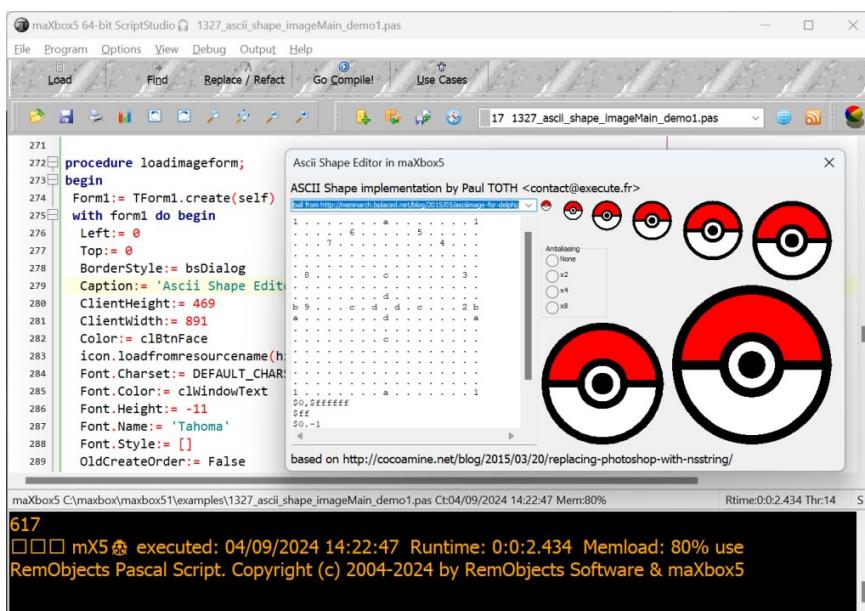
Ascii Shape



The main difference between this implementation and others is the ability to specify colors of the shapes. After the shape you can define as much colors then the number of shapes in the picture.



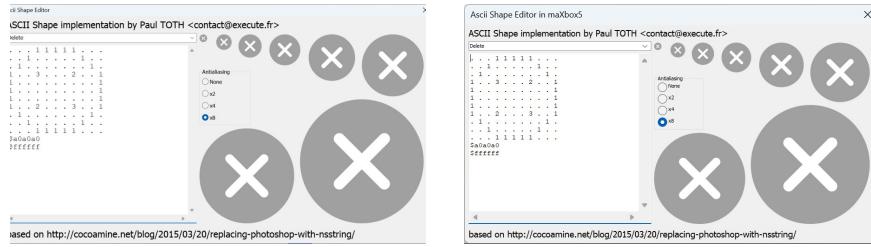
<https://sourceforge.net/projects/maxbox/files/>
Examples/13_General/1327_ascii_imageMain_demo.pas/download



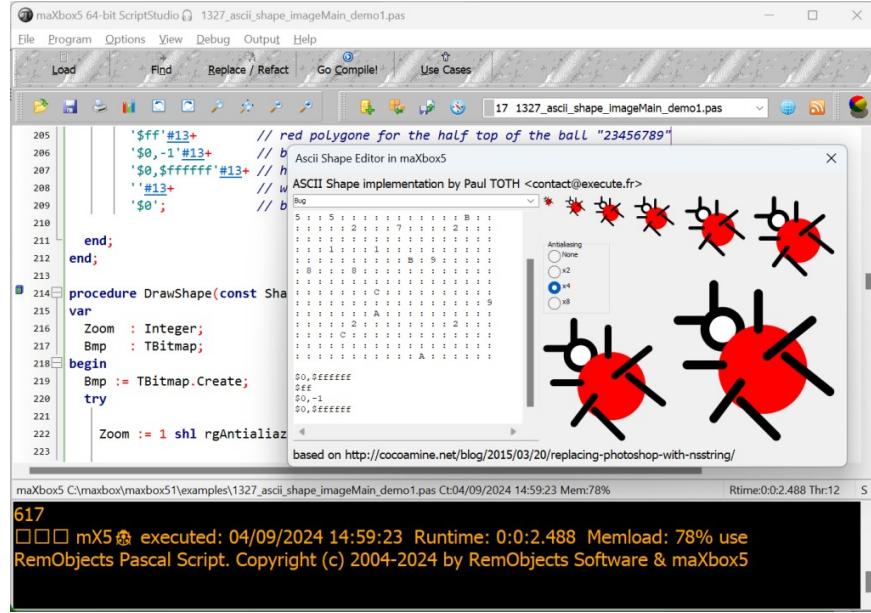
main form with code and shape editor

What's a developer to do? Write code! I don't remember the first time I decided to draw an

image directly in code, but that seemed like a good idea at the moment.



Compiled and Scripted Version



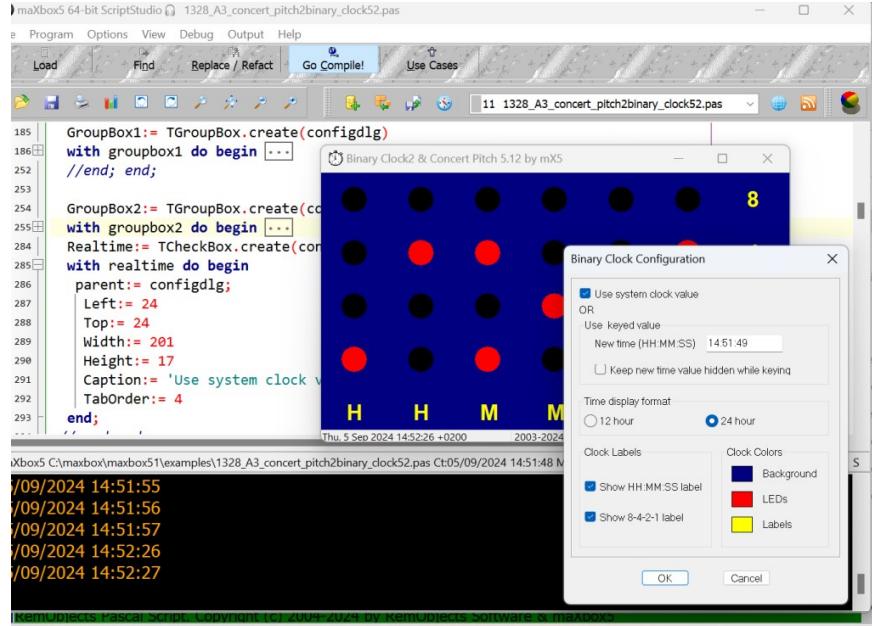
<https://sourceforge.net/projects/maxbox/files/>

Examples/13_General/1327_ascii_shape_imageMain_demo1.pas/download

An other difference with the original code is that the lines do not need to be all equals in lenght, the largest one define the final image width.

Basic properties	Value
MD5	9a92dc62e4a7939ee1de31393aab1ce
SHA-1	cd7a7b794fb78c22cd4d2590074a7947771cecb
SHA-256	f5ddcc172f8693f0a190feb55edcb91ad8d8692f2570b952fd83ce4ac7e2b2e
Vhash	0770b65d5c0d5d155c0518126090090ff1f8022a432424249031ff085e40185212
Authentihash	4b27ca344c7b7c63a114e6514385bfafe0c0dada2e61a4a707ccfecf6675e65
ImpHash	448719d3c6821ca5d5d05ccb8d29e2
SSDeep	T86432-BmpjPGrwOOfwJWx4tgjIrlUGeMifjrxHUh9scr:ajPGaeWx4vgJUGwOrA0A7cr
TLIH	T1DAf7C1B3290C23AC05E9139C8978F80A631BC751B32C3EF785877691F369D59E3AE95
File type	Win32 EXE (executable, windows, win32, pe)

Beyond Virus Total

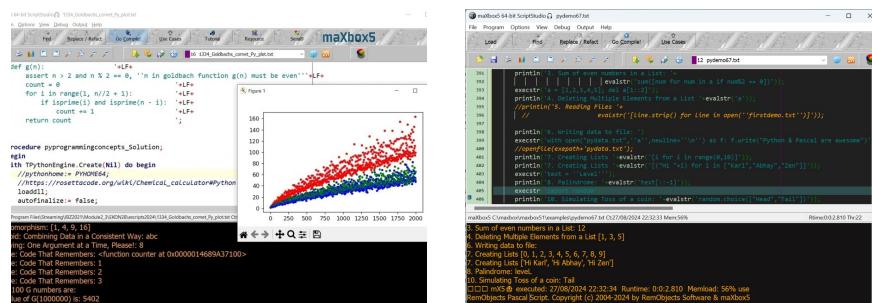


<https://sourceforge.net/projects/maxbox/files/>
[Examples/13_General/1328_A3_concert_pitch2binary_clock52.pas/download](https://sourceforge.net/projects/maxbox/files/Examples/13_General/1328_A3_concert_pitch2binary_clock52.pas/download)

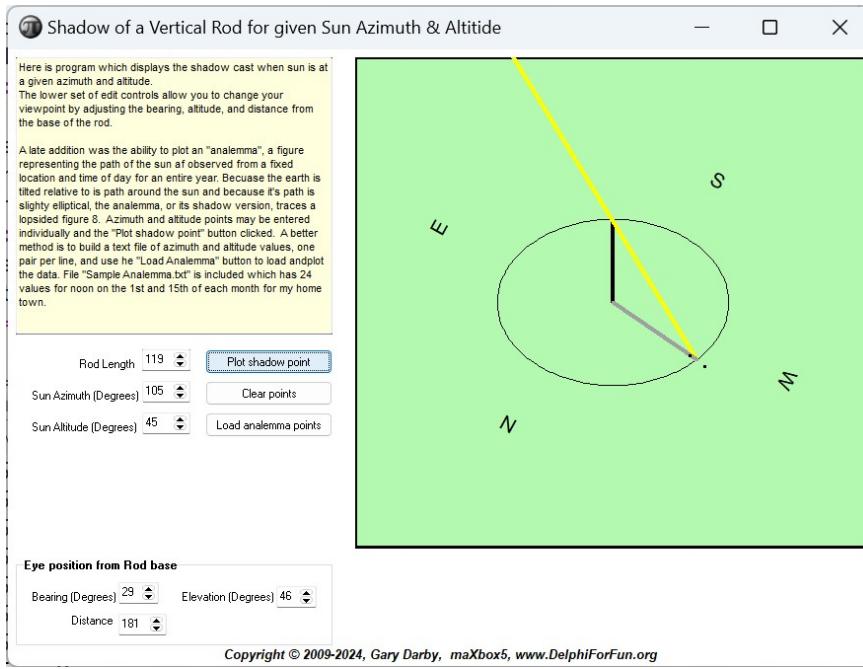
STATION2STATIONDAVIDBOWIE



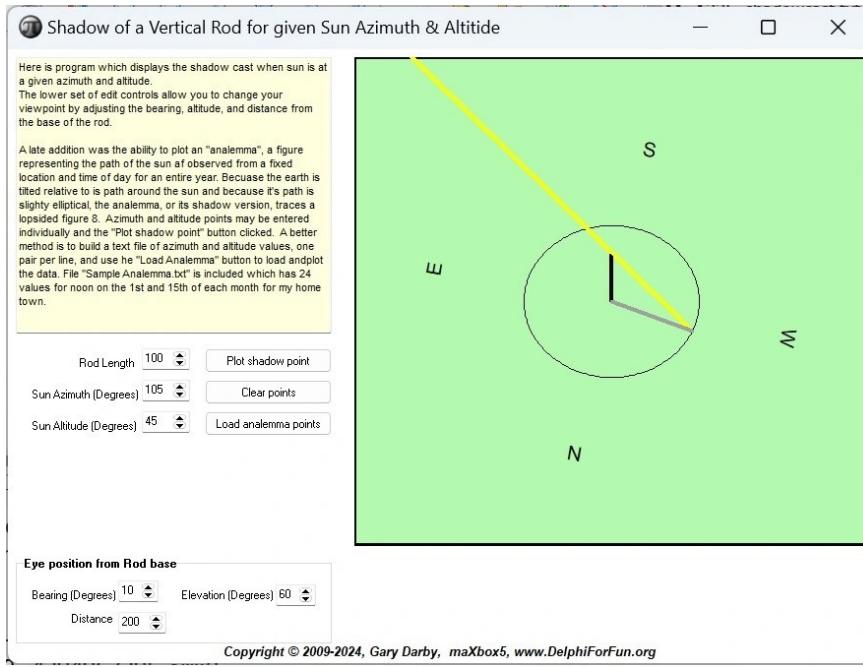
Station2Station



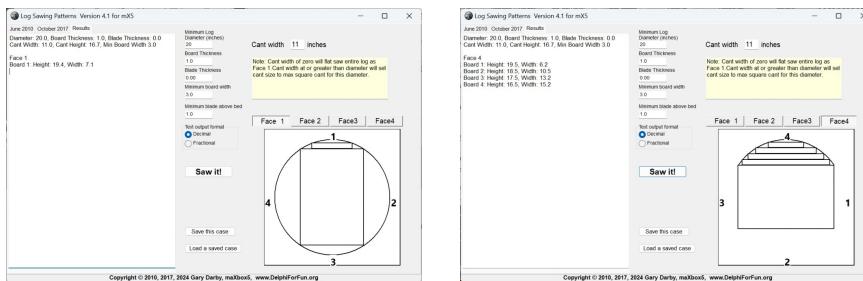
mX5 cool tool



C:\maxbox\maxbox51\examples\1335_shadowcast.txt

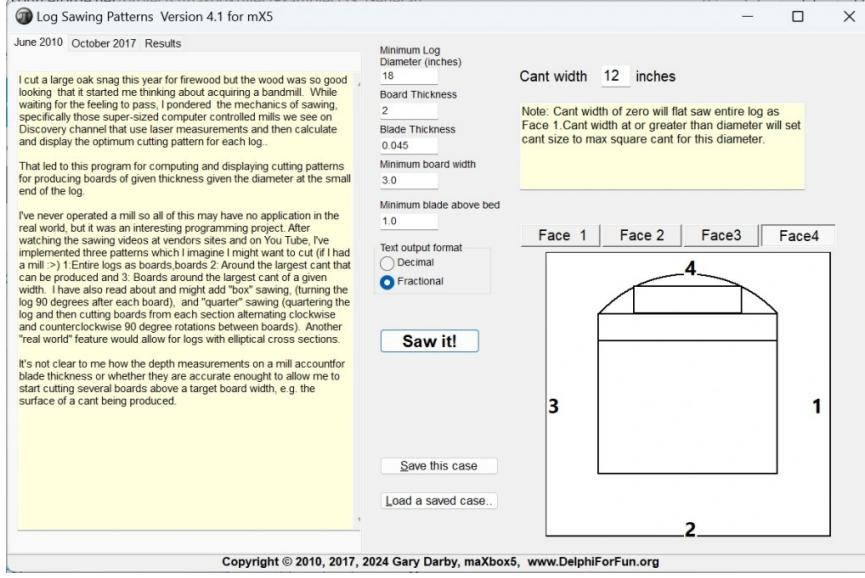
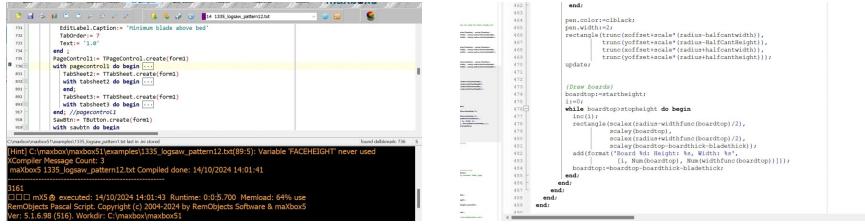


Here's a program which displays potential log sawing patterns for band milling.



<https://sourceforge.net/projects/maxbox/files/>
[Examples/13_General/1335_logsaw_pattern12.txt/download](https://sourceforge.net/projects/maxbox/files/Examples/13_General/1335_logsaw_pattern12.txt/download)





Code Map



<https://sourceforge.net/projects/maxbox/files/>
[Examples/13_General/1337_15_puzzle_game1.pas/download](#)

Certainly! I'll explain this Pascal code function by function:

```

1 | function Iswinner: boolean;
2 | {Check to see if tiles are winning in order}
3 | var I,X,Y: integer;
4 | begin
5 |   Result:=False;
6 |   for I:=1 to CellCount-1 do begin
7 |     Y:=(I-1) div BoardHeight;
8 |     X:=(I-1) mod BoardWidth;
9 |     if GameBoard[X][Y]<>I then exit;
10 |   end;
11 |   Result:=True;
12 | end;

```

Function Overview

This Pascal function named `Iswinner` checks if the game board is in a winning state. It returns

a boolean value (True if it's a winning state, False otherwise).

Code Breakdown

Function Declaration

```
1 | function IsWinner: boolean;
```

This declares a function named `IsWinner` that returns a boolean value.

Variables

```
1 | var I,X,Y: integer;
```

Three integer variables are declared:

- `I`: Used as a loop counter
- `X` and `Y`: Used to calculate positions on the game board

Function Body

1. Initial Result:

```
1 | Result:=False;
```

The function assumes the board is not in a winning state by default.

2. Main Loop:

```
1 | for I:=1 to CellCount-1 do begin
```

This loop iterates through all cells on the board except the last one.

3. Position Calculation:

```
1 | Y:=(I-1) div BoardHeight;
2 | X:=(I-1) mod BoardWidth;
```

These lines calculate the `X` and `Y` coordinates on the board for each cell.

4. Winning Condition Check:

```
1 | if GameBoard[X][Y]<>I then exit;
```

This checks if the value in the current cell (`GameBoard[X][Y]`) matches its expected position (`I`). If not, the function exits early, keeping `Result` as `False`.

5. Winning State:

```
1 | Result:=True;
```

If the loop completes without exiting, it means all cells are in the correct order, so **Result** is set to True.

Function Logic

This function appears to be checking for a specific winning condition where all tiles on the game board are in sequential order. It starts from 1 and checks if each cell contains the number corresponding to its position. If any cell doesn't match, the function immediately returns False. If all cells match their expected values, it returns True.

This type of win condition is common in puzzle games where the objective is to arrange tiles in a specific order, such as a sliding puzzle or a number arrangement game.

Perplexity

The use of **div** for the **board height** and **mod** for the **board width** in the code is related to calculating the 2D coordinates (X and Y) from a 1D index. Let's break down why these operators are used:

Calculating Y coordinate

```
1 | Y := (I-1) div BoardHeight;
```

The **div** operator is used here to calculate the row (Y coordinate) of the cell[3]. It performs integer division, discarding any remainder[1]. This effectively groups the cells into rows.

Calculating X coordinate

```
1 | X := (I-1) mod BoardWidth;
```

The **mod** operator is used to calculate the column (X coordinate) of the cell[3]. It returns the remainder after division[2], which corresponds to the position within a row.

Logic Behind the Calculation

1. The **(I-1)** is used because the loop starts from 1, but array indices typically start from 0.
2. **div BoardHeight** determines how many complete rows have been filled.
3. **mod BoardWidth** determines the position within the current row.

Example

Suppose we have a 3×3 board (**BoardWidth** = 3, **BoardHeight** = 3) and **I** = 5:

1. $Y = (5-1) \text{ div } 3 = 1$ (second row)
2. $X = (5-1) \text{ mod } 3 = 1$ (second column)

This correctly identifies the position of the 5th cell as (1,1) in a zero-based 2D array.

By using these operators together, the function efficiently converts a 1D index (**I**) into 2D coordinates (X,Y) for the game board, allowing it to check if each cell contains the correct value for a winning condition.

Citations:

- [1] <https://smartpascal.github.io/help/assets/div.htm>
- [2] <http://ctp.mkprog.com/en/pascal/modulo/>
- [3] <https://stackoverflow.com/questions/16608612/modulo-operator-in-pascal>
- [4] <https://www.freepascal.org/docs-html/ref/refsu44.html>
- [5] <https://www.bio.vu.nl/thb/course/comp/pascal/pas1e.html>
- [6] https://www.tutorialspoint.com/pascal/pascal_arithmetic_operators.htm
- [7] <https://www.freepascal.org/docs-html/rtl/math/divmod.html>
- [8] <https://smartpascal.github.io/help/assets/mod.htm>

Answer

Perplexity AI is an innovative **AI-powered search engine** designed to provide concise, real-time answers to user queries. Founded in 2022 by a team of former engineers from Google and OpenAI, it aims to blend the functionalities of traditional search engines like Google with the conversational capabilities of chatbots such as ChatGPT.

Source Code and Microtonal Music

While there isn't a direct correlation between source code and microtonal music in the traditional sense, there are several interesting connections and applications where programming intersects with microtonal music creation and exploration:

Electronic Music and Synthesis

Programming plays a crucial role in electronic music production, which often incorporates microtonal elements:

- Karlheinz Stockhausen used electronic means to create works with scales ranging from 7 to 60 equal divisions of the octave⁵.
- Easley Blackwood composed etudes exploring equal temperaments from 13 to 24 notes per octave using electronic music media⁵.
- Wendy Carlos used custom tuning scales in her album "Beauty In the Beast," likely employing programming techniques to achieve these tunings⁵.

MIDI and Microtonal Limitations

While the standard MIDI 1.0 specification doesn't directly support microtonality, developers have found ways to emulate microtonal scales using pitch bending and other programming techniques⁵. This demonstrates how coding can be used to overcome limitations in existing music technology standards. In conclusion, while source code itself isn't inherently microtonal, programming and software development play significant roles in creating tools, instruments, and platforms for exploring, composing, and performing microtonal music. The flexibility and precision offered by digital technologies, implemented through code, have greatly expanded the possibilities for microtonal music creation and study.

Designed with [WordPress](#)