Earth Engine Towns

Memory Game

This Memory Game Master will be the Tutorial 195 from maXbox Starter Series.

You can program a simple memory game in maXbox using its built-in Object Pascal (Delphi-like) scripting engine and the precompiled VCL components. maXbox even includes a proof-of-concept memory game example using a grid of bitmaps. [sourceforge][sourceforge]

What maXbox is

maXbox is a self-contained scripting tool, compiler, and source library in one executable. It uses an Object Pascal/Delphi engine (RemObjects PascalScript) with hundreds of precompiled units and over 1,400 examples.[blogs.embarcadero] [sourceforge][maxbox6.wordpress]

Memory game approach in maXbox

The documented approach uses a `TStringGrid`/`TDrawGrid` (or `QGrid` on Linux/CLX) filled with bitmap images to represent cards. The basic structure follows a typical game loop:[sourceforge][sourceforge]

1. Configure and open window – create the form and grid
2. Initialize resources – load card images, shuffle pairs
3. Register input callbacks – handle mouse clicks on grid cells
4. Event processing loop – manage game state (flipping, matching, scoring) [softwareschule]

Example structure

Here's a minimal runnable skeleton you can adapt in maXbox:

```
const OPENDELAY = 500;

var
  Form1: TForm;
  Grid1: TDrawGrid;
  cCards: array of Integer;  // card values
  Flipped: array of Boolean;
  FirstCard, SecondCard: Integer;
  MatchCount, TotalPairs: Integer;

procedure FormGridCreate(Sender: TObject);
var i, j, k: Integer;
begin
  TotalPairs:= 8;  // 4x4 grid = 16 cards
  SetLength(cCards, TotalPairs * 2 +0);
  SetLength(Flipped, TotalPairs * 2 +0);

  // Initialize pairs
  for i:= 0 to TotalPairs - 1 do begin
    cCards[i * 2]:= i;
    cCards[i * 2 + 1]:= i;
  end;

  // Shuffle cards
  Randomize;
  writ('debug ccards count: '+itoa(high(ccards)+1))
  for i:= High(cCards) downto 1 do begin
    j:= Random(i + 1);
    k:= cCards[i];
    cCards[i]:= cCards[j];
    cCards[j]:= k;
  end;
  grid1.Font.Style2:=fsBold2;
  Grid1.font.size:= 20;
  Grid1.font.name:= 'Wingdings'
  Grid1.ColCount:= 4;
  Grid1.RowCount:= 4;
  // No fixed header row !
  Grid1.FixedRows:= 0;
  Grid1.FixedCols:= 0;
  Grid1.DefaultColWidth:= 80;     //60
  Grid1.DefaultRowHeight:= 80;
  //Grid1.Font.Color:= clred;
  Grid1.Options:= Grid1.Options+ [goFixedVertLine,goFixedHorzLi
end;

procedure Grid1DrawCell(Sender: TObject; ACol, ARow: Integer;
```

```
                                                Rect: TRect; State: TGridDrav
var Index: Integer;
begin
  Index:= ARow * Grid1.ColCount + ACol;
  with Grid1.Canvas do begin
    Brush.Color:= clSilver;
    FillRect(Rect);
    if Flipped[Index] then begin
      TextOut(Rect.Left+22, Rect.Top+16, IntToStr(cCards[Index])
    end else
      TextOut(Rect.Left+ 22, Rect.Top+ 16, '*');  //or ?,!
  end;
end;

procedure Grid1SelectCell(Sender: TObject; ACol, ARow: Integer;
                                                  var CanSelect: Boo
var Index: Integer;
begin
  Index:= ARow * Grid1.ColCount+ ACol;
  if Flipped[Index] or (FirstCard >= 0) and (SecondCard >= 0) tl
    Exit;
  Flipped[Index]:= True;
  Grid1.Invalidate;

  if FirstCard < 0 then
    FirstCard:= Index
  else begin
    SecondCard:= Index;
    if cCards[FirstCard] = cCards[SecondCard] then begin
      Inc(MatchCount);
      writ('debug found: '+itoa(matchcount));
      Grid1.Font.Color:= clgreen; //clyellow; //clnavy;
      if MatchCount = TotalPairs then
        ShowMessage('You got it Mem Master, Gewonnen!');
    end else begin
      // Delay unflip (use Timer in real implementation or sleep
      Flipped[FirstCard]:= False;
      Flipped[SecondCard]:= False;
      Grid1.Font.Color:= clblack;
      sleep(OPENDELAY)
      Grid1.Invalidate;
    end;
    FirstCard:= -1;
    SecondCard:= -1;
  end;
end;

procedure memoryMainRoutine;
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Form1.Caption:= 'Memory Game by maXbox5';
  form1.setbounds(10,10,420,390)
  Grid1:= TDrawGrid.Create(Form1);
  Grid1.Parent:= Form1;
  Grid1.Align:= alClient;
  Grid1.OnDrawCell:= @Grid1DrawCell;
  Grid1.OnSelectCell:= @Grid1SelectCell;
  FirstCard:= -1;
  SecondCard:= -1;
  MatchCount:= 0;
  FormGridCreate(Form1);
  form1.show;
```

```
    //Application.Run;
  end;
```

recent posts

Google Earth Engine API

Getting started

WebView Injection

Cité du Train 2026

GeoJSON in OSM

GeoJSON in GMaps

GeoJSON

Barcode API

Railuino II

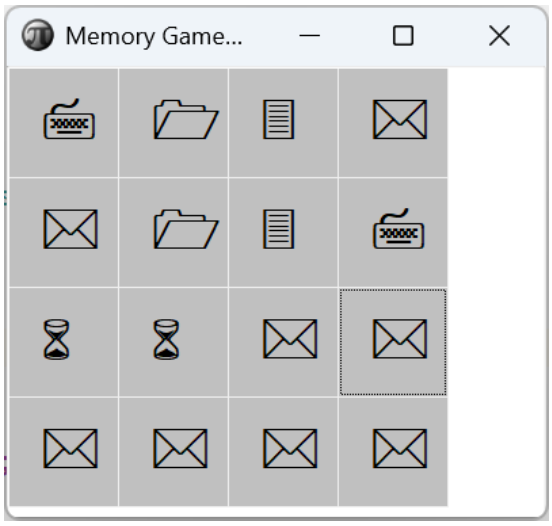
about

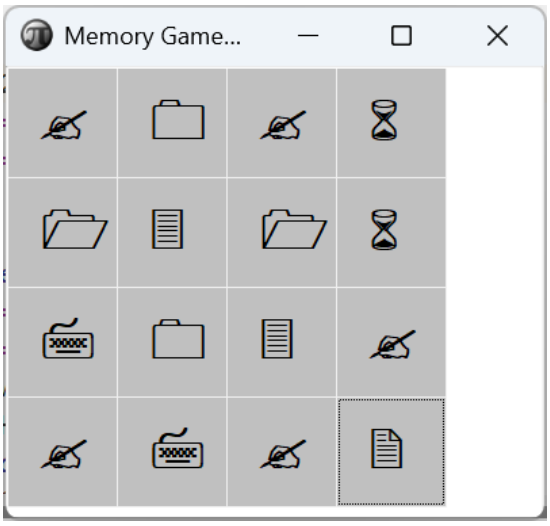
1. Download maXbox from SourceForge and unzip (preserve folder structure) [sourceforge][softwareschule]
2. Open maXbox, create a new script, paste the code above
3. Press F9 to compile and run[softwareschule][softwareschule]
4. Extend with actual bitmap images instead of text, add timers for flip delays, and scoring
5. We use the wingdings font as a straight graphical library

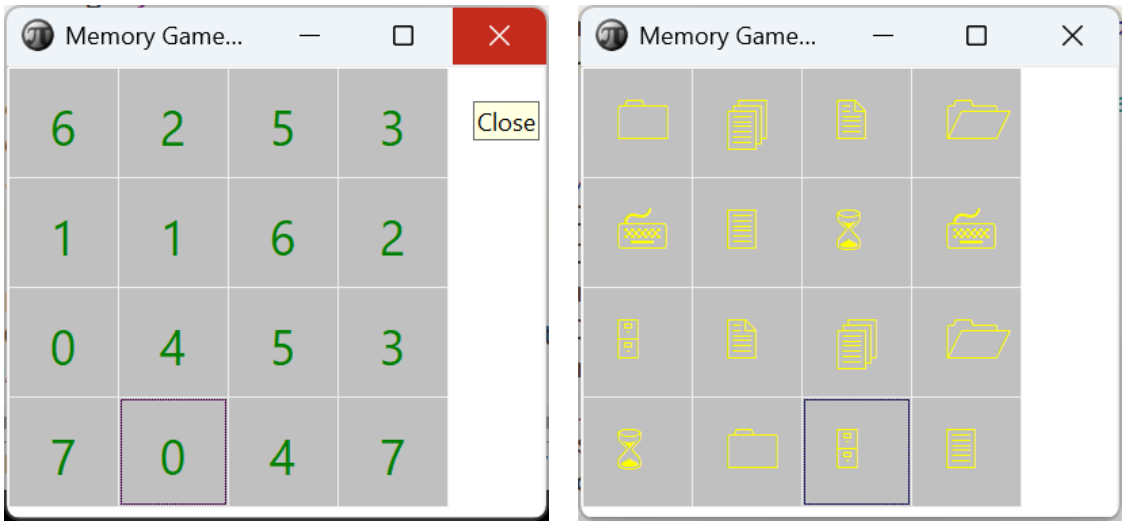
By changing the font.name or comment out

```
grid1.Font.Style2:=fsBold2;
Grid1.font.size:= 20;
Grid1.font.name:= 'Wingdings'
```

For more examples, browse the 1,415+ examples in maXbox or check the Pascal Education Program (PEP) examples on SourceForge.[sourceforge] [maxbox6.wordpress]







Memory Master Decks and Variants

https://sourceforge.net/projects/maxbox5/files/examples/1495_memorygame.txt/download

Note:
In Delphi, if you want to draw text in a TDrawGrid (or TStringGrid) with a custom font color, you need to handle the OnDrawCell event.
This event gives you a Canvas to draw on, where you can set Font.Color before calling TextOut or TextRect.

Key Points:

- Canvas.Font.Color — controls the text color.
- Canvas.Brush.Color + FillRect — paints the background.
- TextRect — draws text clipped to the cell rectangle.
- gdSelected — lets you detect if the cell is selected and adjust colors accordingly.

Avoid flicker — always fill the background before drawing text.

✅

If you want different colors per cell, just add conditions on ACol and ARow inside OnDrawCell.

In Delphi, a TDrawGrid does not have a built-in "header row" like TStringGrid does — it's just a blank canvas for you to draw cells.
If you want the first row to be treated as normal data (not as a header), you simply need to:

Set FixedRows to 0 in the Object Inspector or in code.

```
// No fixed header row !
Grid1.FixedRows:= 0;
Grid1.FixedCols:= 0;
```


Adjust your OnDrawCell logic so that it draws all rows the same way.

FormGridCreate() Explainer

This procedure `FormGridCreate` initializes and configures a 4×4 memory game grid in maXbox. Here's what it does, section by section:

Array initialization

```
TotalPairs := 8;  // 4x4 grid = 16 cards
SetLength(cCards, TotalPairs * 2 + 0);
SetLength(Flipped, TotalPairs * 2 + 0);
```

- Sets up 8 pairs (16 cards total)
- Allocates dynamic arrays `cCards` (card values) and `Flipped` (visibility state) [dev][scribd]

Card value assignment

```
for i := 0 to TotalPairs - 1 do begin
  cCards[i * 2] := i;
  cCards[i * 2 + 1] := i;
end;
```

Creates matching pairs: cards at positions 0–1 get value 0, positions 2–3 get value 1, etc.

Shuffling (Fisher-Yates algorithm)

```
Randomize;
writ('debug ccards count: ' + itoa(high(ccards) + 1))
for i := High(cCards) downto 1 do begin
  j := Random(i + 1);
  k := cCards[i];
  cCards[i] := cCards[j];
  cCards[j] := k;
end;
```

- `Randomize` seeds the random number generator
- `writ` outputs a debug message (maXbox's console output function) [maxkleiner1.medium]
- `itoa` converts the integer count to a string for display[dev][cplusplus]

- The loop implements an in-place shuffle by swapping each card with a random earlier position

Grid visual configuration

```
grid1.Font.Style2 := fsBold2;
Grid1.font.size := 20;
Grid1.font.name := 'Wingdings'
Grid1.ColCount := 4;
Grid1.RowCount := 4;
Grid1.FixedRows := 0;
Grid1.FixedCols := 0;
Grid1.DefaultColWidth := 80;
Grid1.DefaultRowHeight := 80;
Grid1.Options:= Grid1.Options+[goFixedVertLine,goFixedHorzLine,goVe
```

- Sets font to bold Wingdings at size 20 (Wingdings renders numbers as symbols, useful for card icons)
- Creates a 4×4 grid with 80×80 pixel cells
- Removes fixed header rows/columns (`FixedRows := 0`, `FixedCols := 0`)
- Enables grid lines for visual separation between cells

Note on syntax

There's a missing semicolon after `'Wingdings'` — it should be:

```
Grid1.font.name := 'Wingdings';
```

This procedure is typically called from `FormCreate` or similar initialization code to set up the game board before the player starts clicking cards.

Wingdings Explainer

In the memory game code, Wingdings is used as a symbol font where each card's numeric value (0–7) is displayed as a distinct pictogram instead of a number. [en.wikipedia][text-case]

How it works

When you set:

```
Grid1.Font.Name:= 'Wingdings';
Grid1.Font.Size:= 20;
```

and then draw a number like `0`, `1`, `2`, etc. on the grid cell, Wingdings renders those ASCII codes as symbols:

Card value	ASCII char	Wingdings symbol	Description
0	`0`	☔	Umbrella with rain
1	`1`	☕	Hot beverage
2	`2`	☖	White shogi piece
3	`3`	☗	Black shogi piece
4	`4`	☘	Shamrock
5	`5`	❧	Reversed floral heart
6	`6`	☚	Left pointing index
7	`7`	☛	Right pointing index

[text-case][texttoolz][dcode]

Display logic in the game

In the `OnDrawCell` event (not shown in your snippet but implied), you'd render the card like this:

```
if Flipped[Index] then
  Grid1.Canvas.TextOut(Rect.Left+20, Rect.Top+20, Chr(Index+48)) //
else
  Grid1.Canvas.TextOut(Rect.Left + 20, Rect.Top + 20, '?');
```

- Hidden state: Shows `?` (question mark)
- Flipped state: Shows the Wingdings symbol for that card's value[wingdingstranslator][text-case]

Why Wingdings?

- No external images needed – symbols are built into the font

- **Consistent sizing** – scales with font size like text
- **Easy pairing** – numeric values 0–7 map directly to distinct symbols[en.wikipedia][text-case]

Alternative mappings

You could also use other ranges for more variety:

- A–H → ✌, ✍, ✈, ✉, ☺, ♣, ♠, ♦ (victory hand, writing hand, airplane, envelope, smiley, card suits)[text-case][texttoolz]
- Circled numbers ⓪–⑦ for a numbered look[text-case]

This approach is lightweight and works well for simple memory games without needing bitmap resources.

Flipped Array Logic

The `Flipped` array is a **boolean state tracker** that records which cards are currently face-up (visible) in the memory game.[docwiki.embarcadero][thoughtco]

Declaration and initialization

```
SetLength(Flipped, TotalPairs * 2 + 0);
```

This creates a dynamic boolean array with 16 elements (for a 4×4 grid). In Delphi/Object Pascal:

- Dynamic arrays are **zero-based** (indices 0–15)
- Boolean elements are initialized to `False` by default when allocated[docwiki.embarcadero][docwiki.embarcadero][docwiki.embarcadero]
- So initially, all 16 cards are face-down (not flipped)

How it's used in gameplay

The array works as a lookup table during the game loop:

State	`Flipped[Index]`	What the player sees
Face-down	`False`	Hidden symbol (e.g., `?`)
Face-up	`True`	Card symbol visible (Wingdings character)

Typical usage in `OnDrawCell`:

```
if Flipped[Index] then
  // Draw the card's symbol (Wingdings)
  Grid1.Canvas.TextOut(Rect.Left, Rect.Top, Chr(Cards[Index] + 48))
else
  // Draw hidden marker
  Grid1.Canvas.TextOut(Rect.Left, Rect.Top, '?');
```

Flipping logic

When a player clicks a cell:

```
procedure Grid1SelectCell(Sender:TObject;ACol,ARow:Integer;var CanS
var
  Index: Integer;
begin
  Index := ARow * Grid1.ColCount + ACol;

  // Prevent clicking already-flipped cards or during match-check d
  if Flipped[Index] then
    Exit;

  // Flip this card
  Flipped[Index] := True;
  Grid1.Invalidate;  // Triggers OnDrawCell to show symbol
  ...
end;
```

Match-checking flow

1. First click: `Flipped[FirstCard] := True`
2. Second click: `Flipped[SecondCard] := True`
3. Compare: If `Cards[FirstCard] = Cards[SecondCard]`, leave both `True` (permanently revealed)
4. No match: After a delay, set both back to `False` (hide again)[thoughtco] [drbob42]

Why a separate array?

You could store state in `Cards` (e.g., negative values for hidden), but a dedicated `Flipped` array:

- Keeps card values clean (0–7 for pairs)
- Makes logic clearer (`if Flipped[i]` is self-documenting)

- Allows easy extension (e.g., `FlippedOnce`, `MatchedCount`) [delphibasics.50webs][delphibasics.50webs]

The `+ 0` in `SetLength(Flipped, TotalPairs * 2 + 0)` is redundant—it's just `16`—but doesn't hurt.[docwiki.embarcadero][delphibasics.co]

How does Grid1SelectCell calculate the 1D index from grid coordinates?

`Grid1SelectCell` receives a cell's two-dimensional coordinates—column and row—and converts them into the position of that card in the one-dimensional arrays `cCards` and `Flipped`.

```
Index := ARow * Grid1.ColCount + ACol;
```

This is row-major order: all cards of row 0 come first, then all cards of row 1, and so on. Delphi grid coordinates are zero-based: the top-left card is column 0, row 0. [docwiki.embarcadero][docwiki.embarcadero]

Formula

```
\text{Index} = \text{ARow} \times \text{number of columns} + \text{
```

With four columns:

```
Index := ARow * 4 + ACol;
```

`ARow * 4` skips all cards in the rows above; `ACol` selects the card within the current row. This is the standard conversion from a two-dimensional grid coordinate to a one-dimensional array index.[stackoverflow][stackoverflow]

4×4 mapping

Grid cell	Calculation	Array index
(ACol=0, ARow=0)	0 * 4 + 0	0
(ACol=3, ARow=0)	0 * 4 + 3	3
(ACol=0, ARow=1)	1 * 4 + 0	4
(ACol=2, ARow=1)	1 * 4 + 2	6
(ACol=3, ARow=2)	2 * 4 + 3	11

Grid cell	Calculation	Array index
(ACol=3, ARow=3)	3 * 4 + 3	15

So the visual grid corresponds to the arrays like this:

```
Grid coordinates              cCards / Flipped index

(0,0) (1,0) (2,0) (3,0)      0   1   2   3
(0,1) (1,1) (2,1) (3,1)      4   5   6   7
(0,2) (1,2) (2,2) (3,2)      8   9  10  11
(0,3) (1,3) (2,3) (3,3)     12  13  14  15
```

Why it matters

After the calculation, both arrays use exactly the same index:

```
Index := ARow * Grid1.ColCount + ACol;

if not Flipped[Index] then begin
  Flipped[Index] := True;     // Reveal this clicked card
  writ(IntToStr(cCards[Index])); // Its matching-pair value
end;
```

For example, clicking the third column of the second row gives `(ACol=2, ARow=1)`, therefore `Index = 6`. The game reads `cCards[6]` to identify its symbol and changes `Flipped[6]` to decide whether that symbol should be drawn. [docwiki.embarcadero][docwiki.embarcadero]

Full Source Code

https://sourceforge.net/projects/maxbox5/files/examples/1495_memorygame.txt/download