CAMBRIDGE SYSTEMATICS

Think Forward

GISDK Basics

presented to

Caltrans

presented by

Cambridge Systematics, Inc.

1/10/2017

GISDK ~~Programming~~ Scripting Language

➤ C-like?
  » Do … end instead of { … }
    ▪ Macro … EndMacro
    ▪ Dbox … EndDbox
    ▪ Etc…
  » Variable type is handled automatically
    ▪ Single-element variables do not need to be declared
  » Code ~~can~~ must be contained in
    ▪ Macros (similar to functions)
    ▪ Dialog Boxes
    ▪ Objects
      • Can have methods and properties
  » Must be complied prior to running

Editing Resource code

➤ Syntax Highlighting
  » Function Names
  » Item Start/End
  » Strings
  » Numbers
  » Comments

➤ Various programs available
  » Notepad++ (Free, open source)
  » Ultra Edit (not free)
  » Various others…

➤ Basic, no highlighting: notepad.exe

CAMBRIDGE SYSTEMATICS

Language Elements

> Variables

» Very flexible (dynamic typing)

» Integer, string, real

» Many more…

> Arrays

// indicates end of line comments

```
MyArray = {2, 4, 6}    //MyArray[1] = 2
MyArray[5] = 1  //This will result in an error (out of bounds)
MyArray[0] = 1  //This will always result in an error (GISDK is one based)
```

» Each element can have a different type

```
MyArray = {1, "Two", 3}
```

» Arrays can be nested

```
MyArray = {{1, 2, 3}, {"a", "b", "c"}}           //MyArray[2][1] = "a"
```

Language Elements

➤ Arrays

» Arrays may need to be defined:

```
MyArray = {2, 4, 5}   //Creates a new array
YourArray[1] = 2   //Does not work unless YourArray is already an array
Dim YourArray[10]   //Creates an empty 10-element array
Dim YourArray[10, 10] // Creates a new 10x10 array
```

» Array Length

```
MyArray = {2, 4, 6}
x = MyArray.Length //x = 3
x.Length = 5   //This will result in an error
```

» Arrays are pointers!

```
MyArray = {{2, 4, 6}, {8, 10, 12}}
RefArray = MyArray[1]
RefArray[1] = "Two"
x = MyArray[1][2]   //x = "Two"

MyArray = {{2, 4, 6}, {8, 10, 12}}
RefArray = CopyArray(MyArray[1])
RefArray[1] = "Two"
x = MyArray[1][2]   //x = 2
```
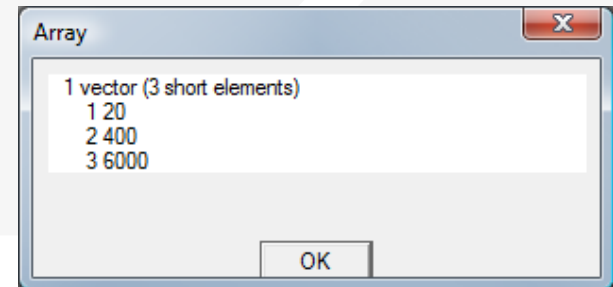
This can get you in trouble!

CAMBRIDGE SYSTEMATICS

Language Elements

Vectors

» Contain a row or column of data (1 dimension only)

» All elements are the same type

» Allow for efficient storage and fast operations

```
//V2 and V3 are already vectors...
V1 = ArrayToVector({2, 4, 6})
V2 = ArrayToVector({10, 100, 1000})
V = V1 * V2
ShowArray({V})
```

```
Array                                    [x]

    1 vector (3 short elements)
      1 20
      2 400
      3 6000

                        OK
```

» Read and set vectors from matrices and data tables

CAMBRIDGE SYSTEMATICS

Language Elements

- Matrices
  - » Referenced with "Matrix Currencies"
  - » Point to a matrix file, core, and index
  - » efficient Two dimensional data storage

```
MC := MC1 + MC2   //Perform element-by-element addition of two matrices
```

  - » New in TransCAD 6: Memory Only matrices can be used for scratch data
    - Can be much faster
    - Can run out of memory with large matrices

CAMBRIDGE SYSTEMATICS

Language Elements

➤ Functions

» Built-in functions

```
ShowMessage("Hello World!")    //Show a simple message
ShowArray({2, 4, 6})           //Display the contents of an array
RunMacro("My Macro")           //Runs a macro
RunDbox("My Dialog")           //Runs a dialog box
Return(True)                   //Returns (ends) a macro or dialog box
//... many many more...
```

» Functions may return values

```
x = Round("2.1", 0)  //x = 2.0
```

➤ Operators

» *Most* standard math operators apply:

```
x = 1 / 2 + 5 * 2    //x = 10.5
x = Pow(10, 2)       //x = 100 (x = 10^2 does not work)
```

CAMBRIDGE SYSTEMATICS

Language elements

➤ Loops

» For Loops:

```
dim arr[5]
for i = 1 to arr.Length do
    arr[i] = i * 2
end
//arr[4] = ??
```

> Avoid looping over long arrays or vectors to perform simple operations. Vector math is much quicker

» While Loops:

```
ans = "Yes"
While ans = "Yes" do
    ans = RunMacro("Again?")  //User input macro
end
```

Language Elements

➤ **If Statements**

» Evaluate a True/False condition

» Uses = in a different context

- There is not a separate assignment operator (except for matrices)

```
MyVal = True
if MyVal = True then do
        ShowMessage("It's True!")
end
```

- Relational Operators

```
a = b   a < b      a > b      a <= b     a >= b     a <> b
1 between 1 and 5                 //returns True
"GIS" Like "GI?"                  //returns True
"TransCAD" contains "CAD"   //returns True
```

- and, or, & not (Lazy evaluation or call by need)

```
if x = "Platypus" or (x <= 100 and y > 500) then do
    if not x = "Platypus" then ShowMessage("It's not a platypus...")
    else ShowMessage("It's a platypus, but x and y were not checked")
end
//Shortcuts:  and --> &   or --> |   not --> !
```

CAMBRIDGE SYSTEMATICS

Macros

- Macros contain code
  - » Operations are performed in sequence as called
  - » Macros end when a value is returned
  - » Arguments can be passed to macros

```
1   Macro "Hello World"
2
3       ShowMessage("Hello World")
4
5   EndMacro
6
7   Macro "Multiply 5" (num)
8
9       ShowMessage("Your Number multiplied by 5 is " + string(num * 5))
10      Return(num * 5)
11
12  EndMacro
13
14  Macro "Call Multiply"
15      x = RunMacro("Multiply 5", 5)   //Can have up to 8 arguments
16      //x = 25
17  EndMacro
```

CAMBRIDGE SYSTEMATICS

Dialog Boxes

➤ Dialog boxes interact with the user

» Dialog box items contain code

```
20 Dbox "Interface" Title: "Five Times"
21
22     //This section is run when the dialog box starts
23     init do
24         ShowMessage("Starting up!")
25     enditem
26
27     //This edit item can only contain an integer value.
28     //The value typed in the box is stored in the variable num
29     //A prompt is placed to the left of the edit field
30     //the field is placed at location x=1, y=1
31     //the field is 10 characters wide and 1 character high
32     edit integer "Num" 1, 1, 10, 1 variable: num prompt: Number
33
34     //The do items (optional) are executed with the item is activated
35     button "Multiply" 1, 3, 10, 1.5 do
36         RunMacro("Multiply 5", num)
37     enditem
38
39     button "Exit" 12, 3, 10, 1.5 do
40         Return()  //Returns null and exits the dialog box
41     enditem
42 EndDbox
```

12

CAMBRIDGE SYSTEMATICS

Objects

➤ GISDK is allows creation and use of Objects

» Objects behave like physical objects

- Each object is of a certain type: **classes** of objects
- They have attributes or characteristics: **properties**
- They have things they can do: **methods**

Objects – Example: A car as an object

- **Class** "Car"
  - » I can create a new object of type or **class** "Car"
  - » In GISDK: MyCar = CreateObject("Car")
- A car can have **properties**
  - » Each car can have different attributes
    - ▪ Color (Red, Blue, Green, etc)
    - ▪ Make, Model, etc.
  - » In GISDK:
    - ▪ MyCar.Color = "Green"
    - ▪ MyCar.Make = "Volkswagen"
    - ▪ MyCar.Model = "New Beetle"

Objects – Example: A car as an object

➤ A car can have **Methods**

  » Methods let the car actually *do something*

    ▪ Accelerate, decelerate, lock, unlock, "Roll down the windows"

  » In GISDK:

    ▪ MyCar.Accelerate(15) //Speed up by 15 mph

    ▪ MyCar.Decelerate(5) //Slow down by 5 mph

➤ Check the property "Speed"

  » ShowMessage("Speed is: " + String(MyCar.Speed))

    ▪ If the car started at 0 mph, what is the speed?

CAMBRIDGE SYSTEMATICS

Objects – Example: A car as an object

```
1    //StartClass
2    Class "Car"
3
4        //This acts as an object "Constructor" and is run each time a new instance
5        //   of the object is created
6        init do
7
8            //An object can have Properties. Properties are attributes of the
9            //   object.
10           //
11           //All properties must be specified or set to null from within the
12           //   object before they can be set or accessed externally.
13           self.Color = "Beige"
14           self.Make = null
15           self.Model = null
16           self.Speed = 0 //Start at rest
17
18       enditem
19
20       //A class can have Methods that perform tasks
21       macro "Accelerate" (AddSpeed) do
22           if AddSpeed > 0 then self.Speed = self.Speed + AddSpeed
23       enditem
24
25       macro "Decelerate" (DecSpeed) do
26           if DecSpeed > 0 then self.Speed = Max(self.Speed - DecSpeed, 0)
27       enditem
28
29   EndClass
```

CAMBRIDGE SYSTEMATICS

Objects – Example: A car as an object

➤ Create a new car object and adjust the speed
  » See what happens if we try to access an undefined property

```
Macro "Go"
    MyCar = CreateObject("Car")
    MyCar.Accelerate(15)
    MyCar.Decelerate(5)
    ShowMessage("Speed is: " + String(MyCar.Speed))

    //This will cause an error.  FuelType is not a valid property for
    //  the "car" class.
    MyCar.FuelType = "Diesel"

EndMacro
```

CAMBRIDGE SYSTEMATICS

Objects – Example: A car as an object

➤ New properties can be defined from within the object, but not externally

```
31          macro "InitFuelType" do
32              self.FuelType = null
33          enditem
```

```
50      MyCar.InitFuelType()
51      MyCar.FuelType = "Diesel"
52      ShowMessage("Fuel Type is: " + String(MyCar.FuelType))
```
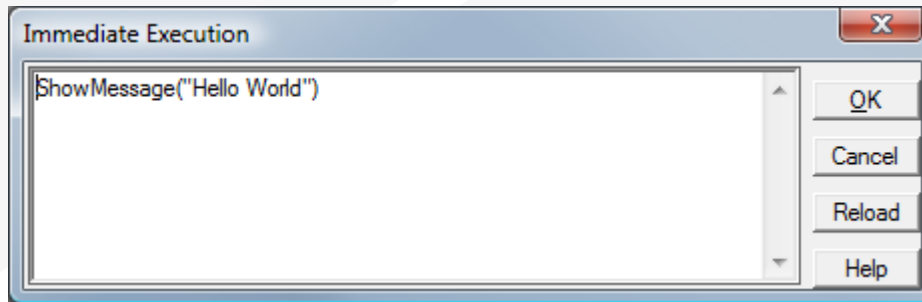
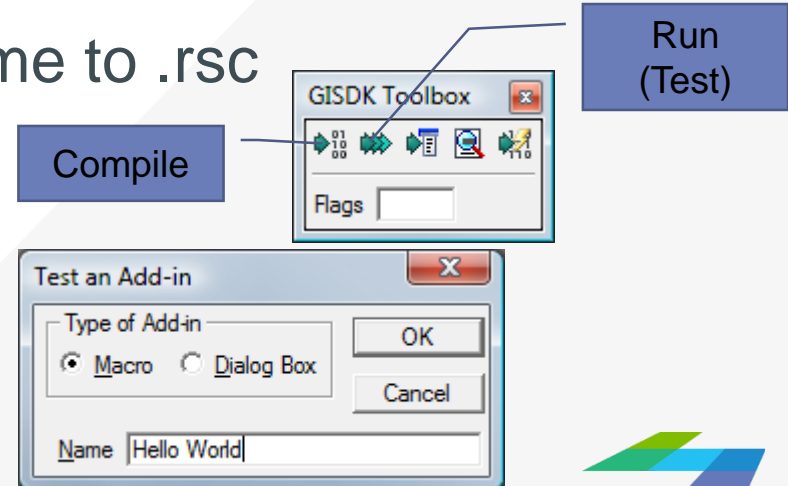➤ A constructor are local and cannot be accessed from another method

➤ The **self** variable in a class macro provides access to object properties and methods

CAMBRIDGE SYSTEMATICS

Try It!

➤ ## Write a Hello Word line

» Tools → GIS Developer's Kit

» Immediate Execution

Immediate Execution

```
ShowMessage("Hello World")
```

OK
Cancel
Reload
Help

➤ ## Write a Hello World macro

» Create a new text file, rename to .rsc

» Write a macro

- Use the example above
- Compile the macro
- Run the macro

Run
(Test)

Compile

GISDK Toolbox

Flags

Test an Add-in

Type of Add-in
● Macro   ○ Dialog Box

OK
Cancel

Name Hello World

CAMBRIDGE SYSTEMATICS

Options Arrays

➤ Organized function input

```
1  Macro "Question"
2
3      Opts = null
4      Opts.Caption = "Please answer:"
5      Opts.Buttons = "YesNo"
6      Opts.Icon = "Question"
7      Opts.Default = 1
8
9      ans = MessageBox("Do you have a platypus?", Opts)
10
11     if ans = "Yes" then ShowMessage("How strange...")
12     else ShowMessage("Why not?")
13
14 EndMacro
```

Options Arrays

➤ Simply a set of {name, value} pairs

```
//Preferred syntax (usually)
Opts = null
Opts.Caption = "Please answer:"
Opts.Buttons = "YesNo"
Opts.Icon = "Question"
Opts.Default = 1

//Alternate syntax
Opts = {{"Caption",     "Please answer:"},
        {"Buttons",     "YesNo"},
        {"Icon",        "Question"},
        {"Default",     1}}
```

» Options are often optional
  ▪ The whole array can be null
  ▪ Certain options can be missing (will revert to default)
» Option names with spaces are allowed:
  ▪ Opts.[Two Words]

CAMBRIDGE SYSTEMATICS

Batch Mode

➤ Planning functions are run in Batch Mode:
- » Record GUI input using the batch recorder
  - ▪ Planning → Batch Editing
- » This will create a rsc file
  - ▪ Adjust option values as desired
  - ▪ Replace option values with variables
- » Compile and run a new rsc file

CAMBRIDGE SYSTEMATICS

Batch Mode: Example

```
1  Macro "Batch Example"
2
3
4      //Initialize the batch processor:
5      //This must be done prior to calling batch procedures
6      RunMacro("TCB Init")
7
8      //(Put the directory in a variable for convenience)
9      dir = "C:\\Users\\Sean\\Work\\OCTA\\Training\\SCAGLOCAL\\TripDistribution\\RunGrav\\"
10
11     //Create an options array:
12     Opts = null
13     Opts.Input.[Input Matrix] = dir + "OfPkSkims.mtx"
14     Opts.Input.[Target Core] = "Table 3"
15     Opts.Input.[Core Name] = "Was Table 3"
16
17     //run the procedure or operation:
18     ret_val = RunMacro("TCB Run Operation", "Rename Matrix Core", Opts)
19
20     //did the operation complete successfully or fail?
21     // --> If success, the macro will continue (e.g., more functions)
22     // --> After failure, the macro will goto quit: and exit
23     if !ret_val then goto quit
24
25
26     quit:
27
28     //Close the batch processor
29     //Argument 1 (ret_val) is True/False and indicates success/failure
30     //Argument 3 shows the report on screen if true
31     RunMacro("TCB Closing", ret_value, True)
32
33  EndMacro
```

Practice: Batch Recorder

➤ Make a simple model dialog box
- » Rename a matrix core interactively
- » Save the recorded results
- » Create a dialog box
  - Add a text box asking what the "DA" core should be named
  - Add an OK and Cancel button
- » Rename the core on OK

```
8
9      button "Exit" 12, 3, 10, 1.5 do
0          Return()   //Returns null and exits the dialog box
1      enditem
```

- » Compile and run

➤ Extra Credit: Ask the user which core to rename as well

CAMBRIDGE SYSTEMATICS

Debugging the Code

➤ TransCAD has an interactive debugger

➤ Two ways to use:

» Set a break in the code

» Choose to debug after a crash

> You must have the resource code to use the debugger!

➤ Try It: Set a break on the example script

» Identify a line number in your editor

» Go to the same line in the debugger, then add a break

» Leave the debugger open and run the script

CAMBRIDGE SYSTEMATICS

Debugging the Code

➤ In the stopped code, step through each line

» Check the variable window, the watch window

CAMBRIDGE SYSTEMATICS

Challenge Exercise

➤ Use GISDK to write a script that computes total VMT for all trips with at least one end in Orange County

- » Use functions:
  - OpenMatrix()
  - AddMatrixCore()
  - CreateMatrixCurrency() and/or CreateMatrixCurrencies()
  - CreateMatrixIndex()
  - GetMatrixVector() → Get a vector of row/col sums
  - VectorStatistic() → Get the total from the row or col sum

- » Use matrix operations
  - (e.g., mat_cur := cur1 * cur2)

CAMBRIDGE SYSTEMATICS