

1. Things you can do with MEL:

- Use MEL commands to bypass Maya's user interface, quickly create shortcuts, and access advanced features.
- Enter exact values for attributes, bypassing any restrictions to precision imposed by the interface.
- Customize the interface for specific scenes, changing default settings to settings you prefer for a particular project.
- Create MEL procedures and scripts that carry out custom modeling, animation, dynamics, and rendering tasks.
-

2. Plugin-Examples (Demo next Monday)

//lots of free plugins

<http://www.highend3d.com/maya/mel/>

<http://www.highend3d.com/maya/plugins/>

//alias-wavefront plugins

<http://www.aliaswavefront.com/en/WhatWeDo/maya/extend/index.html>

//toon shader and tree-generator

<http://www.toonshader.mailru.com/>

3. Two ways to program MAYA, MEL and C++:

MEL

- Easier for people not familiar with C++
- Fast to use (it is an interpreter)
- Used to modify user interface

C++

- Much faster
- Can create user-defined data structures
- More programming power

4. Help on MEL:

- Introduction to MEL programming can be found in:
Help/Library/
Go to MEL under Using Maya section.
- MEL command reference:
Help/Library/
MEL Command Reference under Technical Library

- Introduction to C++ programming for MAYA and the command reference:
Help/Library/
Maya Developer's Tool Kit under Technical Library

- Specific command help:
In script editor type:
help command_name

5. Running MEL commands:

- Command line (at the bottom of the Maya window)
- Command Shell (Window > General Editors > Command Shell)
- Script Editor (Window > General Editors > Script Editor)
- You can type your MEL script in Script Editor and then save it, but it is usually better to use some text editor. (Notepad/Word/C++ under Windows, pico/emacs/vi under linux). For people unfamiliar with linux editors, "pico" is very similar to notepad.

6. To open existing MEL script:

- Script name usually has .mel extension
- Put MEL file into directory specified by script path variable (MAYA_SCRIPT_PATH). You can look at that variable by typing getenv "MAYA_SCRIPT_PATH" in Script Editor.
- In script editor type:
source file_name.mel (that executes all commands in file)
procedure_name (name of the procedure you want to execute)
- Alternatively you can open the file into Script Editor and then execute it as usual.

7. MEL COMMANDS:

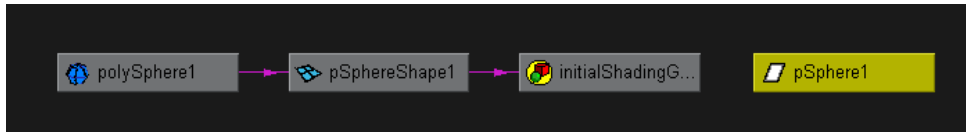
- Anything you do in Maya executes a MEL script
Open Script editor, Choose: edit > clear all
In Maya menu, choose: create > polygon primitive > sphere
Result in script editor:

```
polySphere -r 1 -sx 20 -sy 20 -ax 0 1 0 -tx 1 -ch 1;  
// Result: pSphere1 polySphere1 //
```
- Creating Shelf Button:
Select commands in script editor, drag the text with middle mouse button to the shelf.
- Basic structure of MEL command:
Most commands have flags and arguments:

```
polySphere -radius 5;  
or  
polySphere -r 5;
```

- Many command can be used in “create”, “query” and “edit” modes

```
polySphere -radius 10 -name pSphere1;  
polySphere -q -radius pSphere1;  
polySphere -e - radius 2 pSphere1;
```
- All information in Maya is stored in nodes.



Can modify attributes using setAttr command:

```
setAttr polySphere1.radius 1;
```

- Most MEL commands require the name of the object in the scene.
 Can use ls command to get the name of the object

```
ls; (lists all objects)  
ls -selection; (return names for all objects that are selected)
```
- To capture result of the command in a variable:

```
string $selectionList [] = `ls -selection`;  
or  
string $name[] = `polySphere -radius 10`;  
move 0 10 0 $name[0];
```

8. Variables and Data Types:

- Data types: int, float, vector, string, array, matrix

```
int $a = 14;    int $b = $a + 10;    print ($b); (Result:24)  
float $f=2.5;  
vector $v = <<1, 2, 3>>; print ($v.y); (Rresult: 2)  
string $str1="Testing "; string $str2 = "Strings"; print($str1+$str2);  
int $arrayVar[] = {1, 2, 3, 4};    $arrayVar[4] = 10; print($arrayVar);
```

9. Controlling Flow of Execution:

- Conditions (If-else, switch)

```
int $flag = 1;  
if ($flag == 0)  
{  
    sphere;  
}  
else  
{  
    cone;  
}
```

```
$flag = 0;
```

```

switch($flag)
{
case 0:
    sphere;
    break;
case 1:
    cone;
    break;
};

```

- Loops (for-in, for, while)

```

for ($i=0; $i<10; $i++)
{
    string $name[] = `sphere -r 1`;
    move 0 $i 0 $name[0];
};

```

10. Procedures and Functions:

Simple way to group commands together.

Function is a special type of procedure that return a value

Procedure:

```

proc make_primitive(int $flag)
{
    if ($flag == 0)
    {
        sphere;
    }
    else
    {
        cone;
    }
};
make_primitive(0);

```

Function:

```

proc string make_primitive(int $flag)
{
    string $name[];
    if ($flag == 0)
    {
        $name = `sphere`;
    }
    else
    {
        $name = `cone`;
    }
}

```

```

    return $name[0];
};
string $new_prim = make_primitive(0);
// Result: nurbsSphere1 //

```

11. Global/Local:

All variables are seen only in the scope they are declared unless specifically declared global :

Example1:

```

global int $a = 0;
proc print_a()
{
    $a = 10;
}

```

```

print_a();
print($a);
//prints 0 not 10

```

Example2:

```

global int $a = 0;
proc print_a()
{
    global int $a;
    $a = 10;
}

```

```

print_a();
print($a);
//prints 10

```

12. Example of a crowd system:

Implements crowd behavior and interface for it.
(see screenshot at the end).

- Create a vehicle:
polyCube -name vehicle_1 -width 2 -height 2.5 -depth 2;
- Make vehicles a solid dynamic object:
rigidBody -name rigidVehicle_1 -active
-mass 1 -bounciness 0
-damping 1.5 -position -10 0 0
-impulse 0.15 0.0 0.0 vehicle_1;

- Display velocity vector:
Solvers > Rigid Body Solver, select Display Velocity box.
`setAttr rigidSolver.displayVelocity 1;` (faster with MEL)
- Changing vehicle direction:
`setAttr rigidVehicle_1.impulseZ -0.15;`
- Using Expressions in MEL:
Can change motion of the vehicle using expressions.

Select Window > Animation Editors > Expression Editor
Enter expression for rigidVehicle_1:
`rigidVehicle_1.impulseX = sin(time);`
`//follows sinusoid`

In expression editor choose Select Filter > By Expression Name
choose wander expression
modify as follow:

```
rigidVehicle_1.impulseX = sin(time);
rigidVehicle_1.impulseY = 0;
rigidVehicle_1.impulseZ = cos(time);
//follows circle
```

change again:

```
rigidVehicle_1.impulseX = sin(time);
rigidVehicle_1.impulseY = 0;
rigidVehicle_1.impulseZ = cos(time*2);
//follows figure 8
```

change again to create a wander behavior.

`xMult` and `yMult` are used to create a different behavior for each vehicle.

`rand()` function can be used to set `xMult` and `yMult` differently for each vehicle.

```
float $xMult = 1.5;
float $zMult = 2.0;
rigidVehicle_1.impulseX = sin(time*$xMult);
rigidVehicle_1.impulseY = 0;
rigidVehicle_1.impulseZ = (noise(time)*$zMult);
```

- Orienting vehicle_1 direction with its motion
`setAttr rigidSolver.allowDisconnection 1;`
`disconnectAttr rigidVehicle_1rx.output vehicle_1.rotateX;`
`disconnectAttr rigidVehicle_1ry.output vehicle_1.rotateY;`
`disconnectAttr rigidVehicle_1rz.output vehicle_1.rotateZ;`
see changes in DG in hipergraph (section 13)

Change “wander” expression in Expression Editor:

```
float $fVel[] = `getAttr rigidVehicle_1.velocity`;
vehicle_1.rotateX = 0;
vehicle_1.rotateY = atan2d($fVel[0], $fVel[2]);
vehicle_1.rotateZ = 0;
```

- Vehicle interaction
How will vehicle react to each other?

In this crowd system we will have two behaviors:

1. each vehicle will try to avoid collisions with other vehicles
2. some of the vehicles (followers) will tend to follow other vehicles (leaders)

Add Radial Field to vehicle_1 to Repel Other Vehicles:

```
radial -position 0 0 0 -name vehicleForce_1 -magnitude 50
      -attenuation 0.3 -maxDistance 8.0;
parent vehicleForce_1 vehicle_1;
```

If this vehicle is a “leader” add attractive radial field that is global in influence and has no maximum distance setting:

```
radial -position 0 0 0 -name vehicleLeaderGlobalForce_1
      -magnitude -1 -attenuation 1.2;
parent vehicleLeaderGlobalForce_1 vehicle_1;
```

If we had two vehicles, one follower(vehicleF_1) and one leader (vehicleL_1)we next need to hookup dynamics, both collisions and fields using connectDynamic command (see code for few vehicles in section 15).

Code in section 15 also illustrates how to create expression in MEL (without using Expression Editor).

- Simple user interface:

Predefined dialog boxes:

```
string $retVal = `confirmDialog
      -title "Warning"
      -message "Removing vehicle"
      -button "OK"
      -button "Cancel"`;
```

```
print($retVal);
```

Result: OK (if OK button is pressed)

```
promptDialog -message "Enter number of vehicles:"
      -button "OK"
      -button "Cancel";
```

```
string $numVeh = `promptDialog -q`;
```

(See printout for more details on user interface design)

- The whole system:

Has ability to turn selected objects into obstacles for the vehicles (obstacles are passive rigid bodies).

Has user-controllable global field that affects all of the objects uniformly (4 radial fields located at specified positions in the scene).

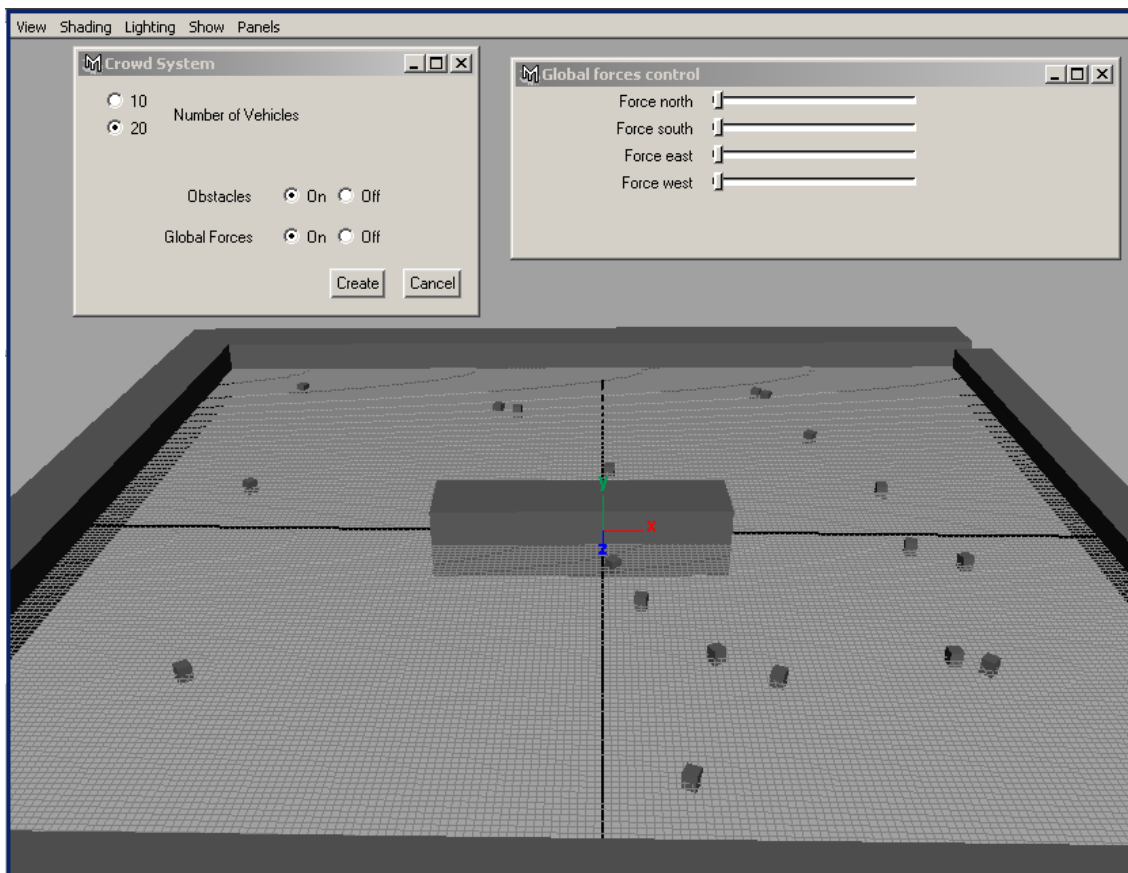
MEL user-interface allows:

- choose number of “followers” and number of “leaders”

- turn global forces ON/OFF

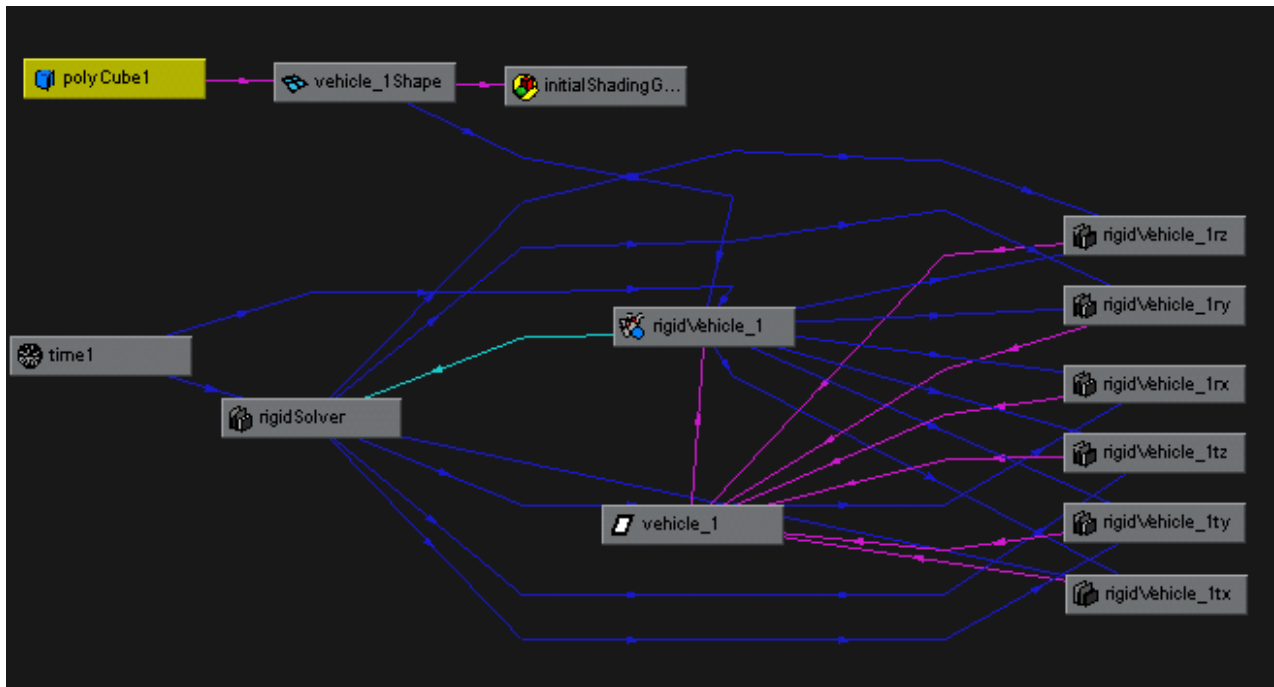
- enable/disable obstacles

- global force control using sliders



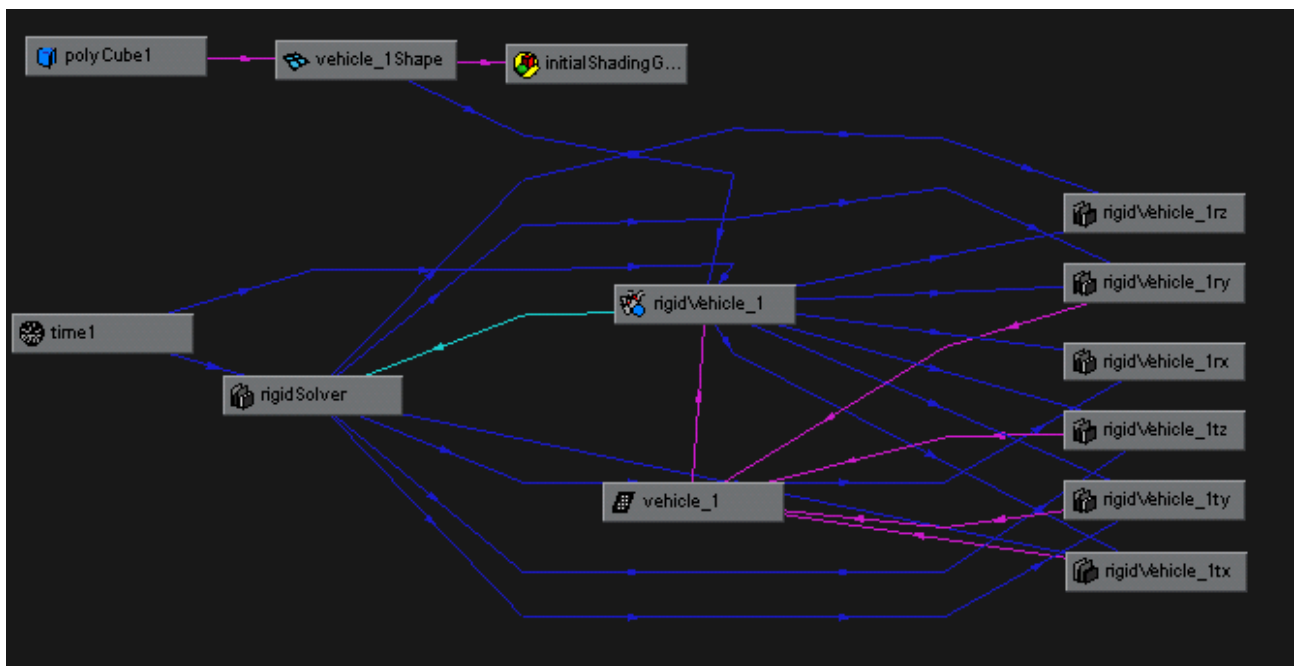
13. DG and DAG:

DG (dependency graph)

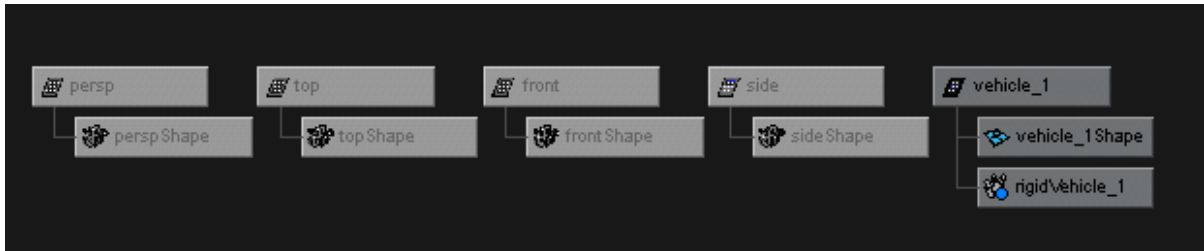


After

```
disconnectAttr rigidVehicle_1rx.output vehicle_1.rotateX;  
disconnectAttr rigidVehicle_1rz.output vehicle_1.rotateZ;
```



DAG (directed acyclic graph)



14. Code for few vehicles (with no interface):

- On ART server in CrowdSystem directory
- To run:
Place crowdSystemNoInterface.mel script into Maya Path as describe before.
Type in script editor:

```
source crowdSystemNoInterface.mel  
createCrowd(10);  
or  
createCrowd(20);
```

15. Code for the whole system (same as above but with user interface):

- On ART server in CrowdSystem directory
- To run:
Create some polygonal objects that will be obstacles.
Select them.
Place crowdSystem.mel script into Maya Path as describe before.
Type in script editor:

```
source crowdSystem.mel  
crowdMain();
```