

## Purpose

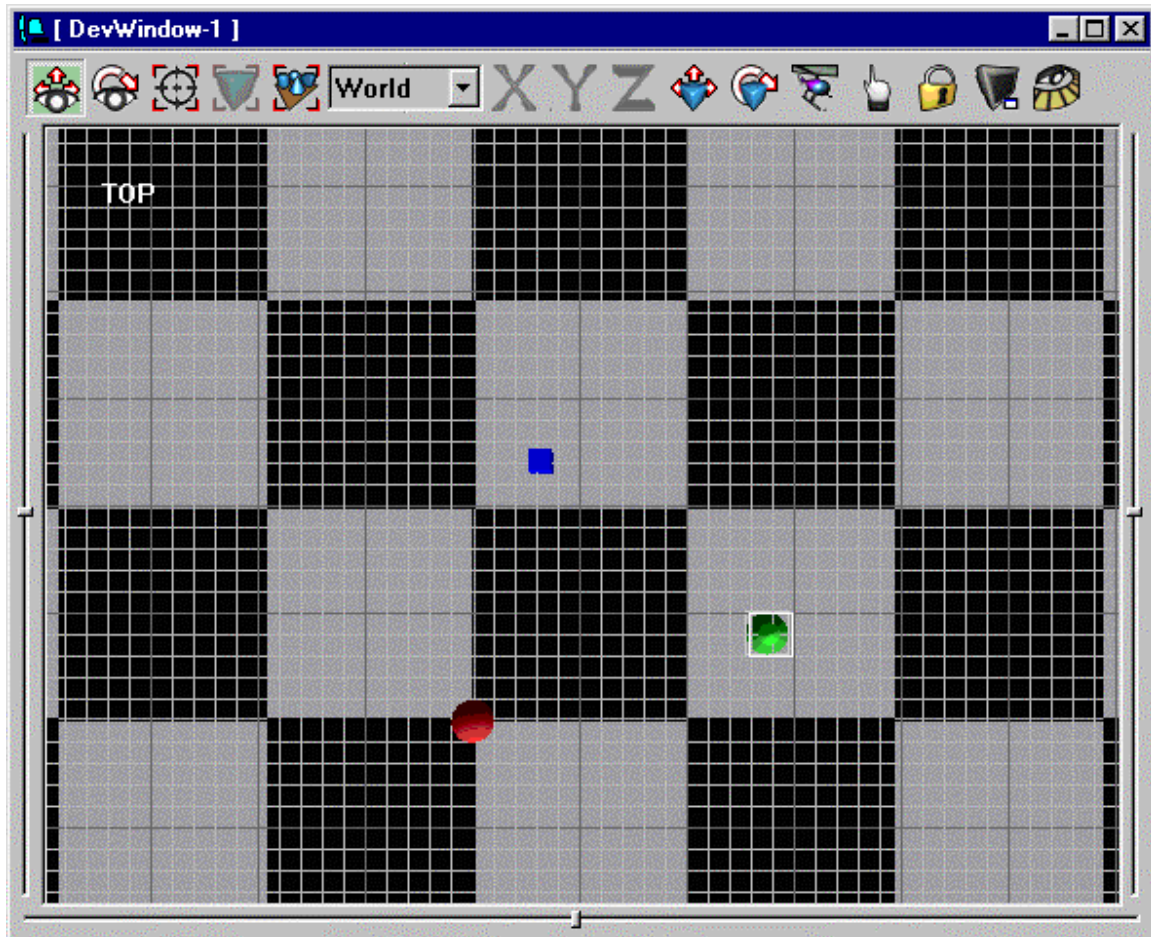
The purpose of this scenario is to show how VRo can incorporate the use of both the MATLAB Plug-in and WorldUp's scripting language to control objects.

This scenario hopes to show:

- ◆ Expose an object to MATLAB and Simulink.
- ◆ Write a WorldUp subroutine script to control an object from MATLAB and Simulink.
- ◆ Add a disabled block to control our mousing viewpoint with a task.
- ◆ Load our objects in MATLAB.
- ◆ Control our objects in MATLAB with a simple function m-file.
- ◆ Load our objects in Simulink.
- ◆ Control our objects in Simulink through the use of simple sources.

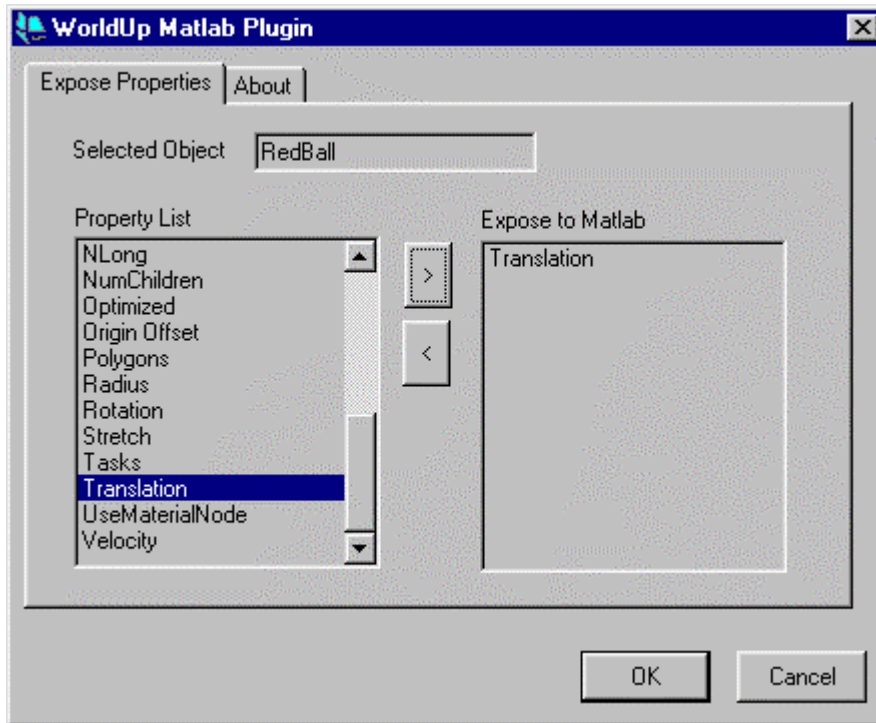
## Load sample combo world

In WorldUp, open the combo.up file located in the VRo/scenarios/combo folder.



## Use MATLAB Plug-in to expose RedBall

1. Select the RedBall in the Scene Graph.
2. Select File►Plug-in Tools►Expose to Matlab
3. Expose the Translation property.
4. Press OK.



# Write subroutine script to manipulate BlueBlock

Select File ► New Script

Enter the following lines:

```
sub moveblock(InputString as string)

'input:Block X
'input:Block Y
'input:Block Z

dim InputDouble(2) as double

dim i as integer
Token = ","

for i = 0 to 2
    InputDouble(i) = 0
next i

StringLength = Len(InputString)
i = 0
while StringLength > 1
    InputDouble(i) = val(InputString)
    TokenPosition = Instr(InputString, Token)
    InputString = Mid(InputString, TokenPosition+1, StringLength -
(TokenPosition))
    StringLength = Len(InputString)
    i = i + 1
wend

dim BlueBlock as geometry

dim Position as vect3d

set BlueBlock = getgeometry("BlueBlock")

Position.x = InputDouble(0)
Position.y = InputDouble(1)
Position.z = InputDouble(2)

BlueBlock.SetTranslation Position
```

sub keyword

Name of script

Name of input string (Do not change this)

Names of input values.  
Must have the format:  
'input:<input name>

Only change the dimension value of this line to match the number of input values. In this case, we have 3 inputs, therefore, the dimension is 2.

Only change the loop end value to match the number of input values. In this case, we have 3 inputs, therefore, the end loop value is

Dimension for our BlueBlock variable

Dimension for our Position variable

Set our BlueBlock variable

Assign our input values

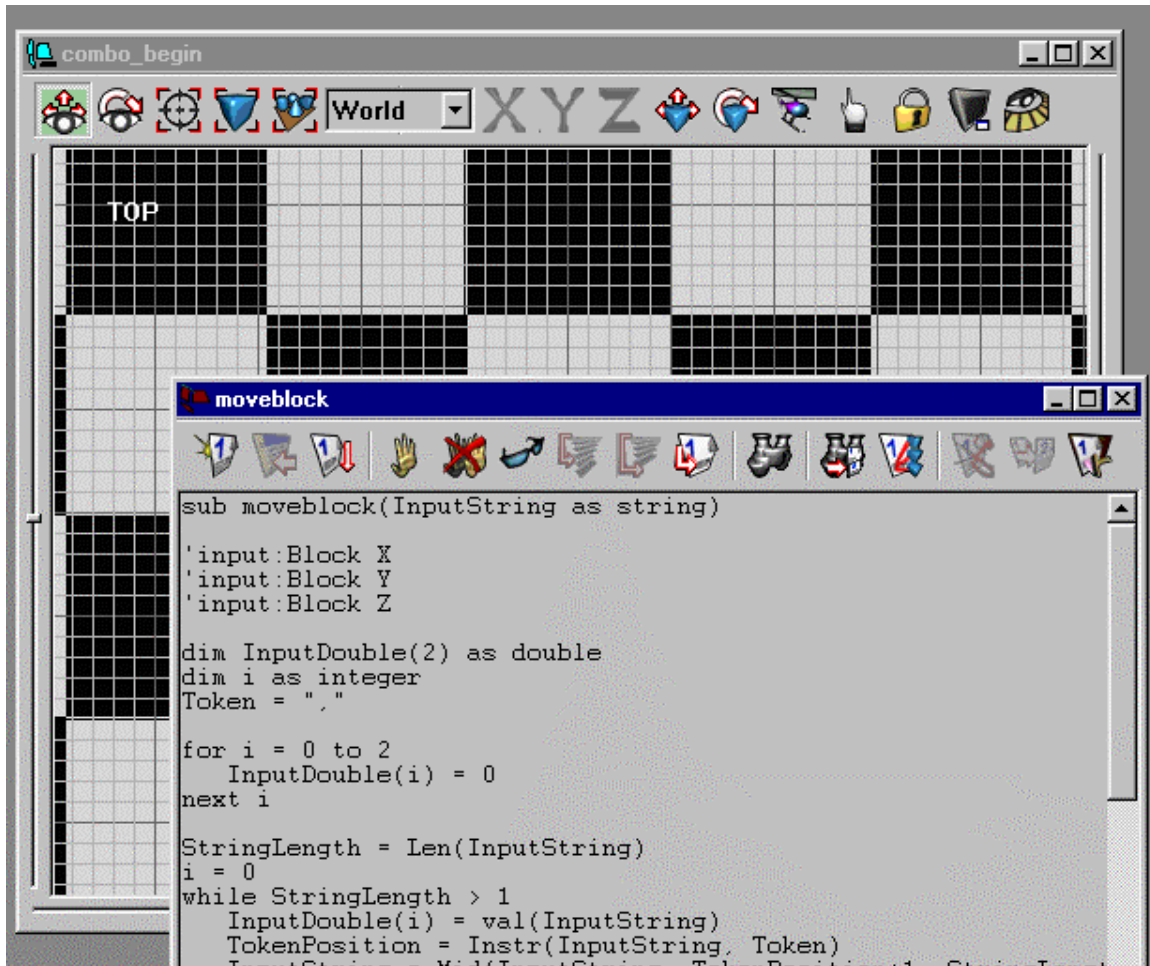
Set the position of BlueBlock

end sub ←

End the subroutine

**The code in red is what you will copy to every subroutine you write that communicates to MATLAB and Simulink.**



Save the script as **moveblock.ebs**.

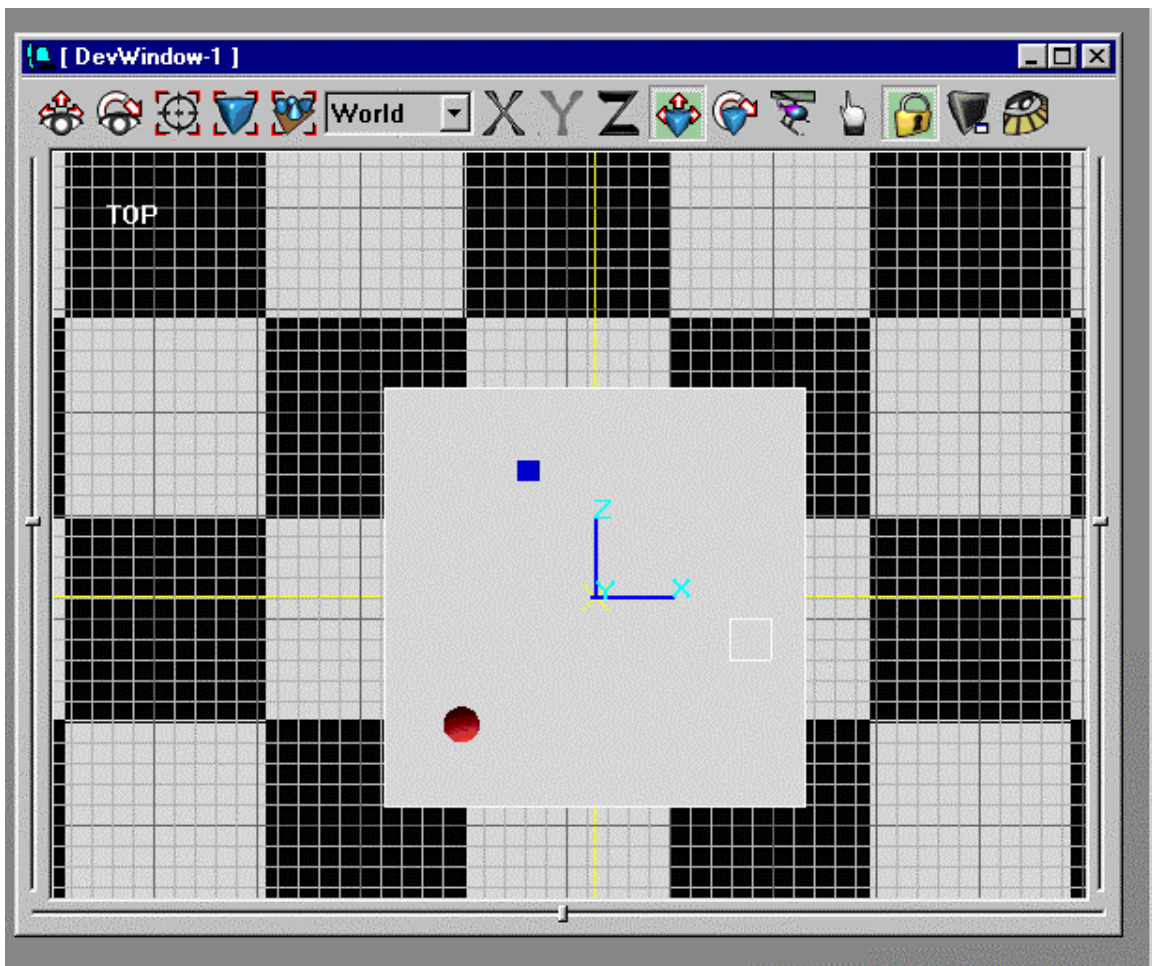




## Create Block geometry for viewpoint control

So that our viewpoint is easy to control with the mouse, we are going to add a Block geometry to lock the viewpoint to.

1. Drag and drop a Block subtype over to the Scene Graph.
2. Change the name to ViewLockBlock.
3. Set the Depth to 20.
4. Set the Width to 20.
5. Press the **Lock Selected** toolbar button. 
6. Press the **Object Translate** toolbar button. 
7. Move the block so it entirely encompasses all three objects.
8. Set Enabled to False so it is not rendered.



## Write task script to lock the viewpoint to ViewLockBlock

Select File ► New Script.

Enter the following:

```
sub task

dim w as window
dim View as viewpoint
dim ViewDirection as vect3d
dim ViewOrientation as orientation

set w = getfirstwindow()
set View = getviewpoint("Viewpoint-1")

dim ViewLockBlock as geometry
set ViewLockBlock = getgeometry("ViewLockBlock")

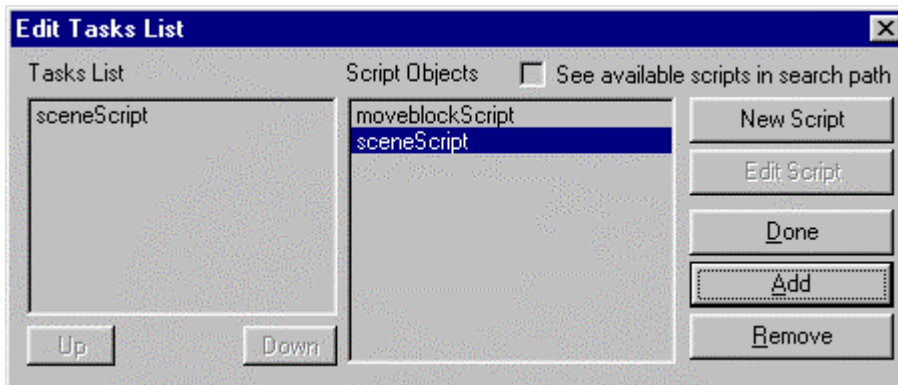
w.ZoomToNode ViewLockBlock

end sub
```

Save the task script as **scene.ebs**.

Because this is a task, it needs to be attached to an item in our scene graph. Let's attach it to our Light-1 object.

1. Right-click on Light-1 and select Edit Tasks.



2. Select SceneScript and press Add.
3. Press Done.

Select Simulation ► Run to test that our task works. You should be able to fly around the three objects in a spherical fashion. Close the created window when finished.

## Load into Matlab

At the MATLAB prompt, pass the path to your combo.up file into [vroload](#):

```
my_objects = vroload('c:\mfiles\vro\scenarios\combo\combo.up');
```

Let's see what objects we have by using the overloaded [set](#) command:

```
set(my_objects(1))  
    Type: Sphere  
    Name: RedBall  
    Properties: Translation (Vect3D)
```

This tells us `my_objects(1)` is of type Block named MouseViewpointBlock with Translation as the exposed property.

```
set(my_objects(2))  
    Type: Script  
    Name: moveblock  
    Inputs: Block X  
           Block Y  
           Block Z
```

This tells us `my_objects(2)` is of type Script. This is because we wrote a subroutine script called moveblock to control our BlueBlock. The inputs to that script are then listed.



## Write m-file to control objects

Here is a simple m-file that controls the various objects in our world:

```
function combo(my_objects)

% assign our objects
RedBall = my_objects(1);
BlueBlock = my_objects(2);

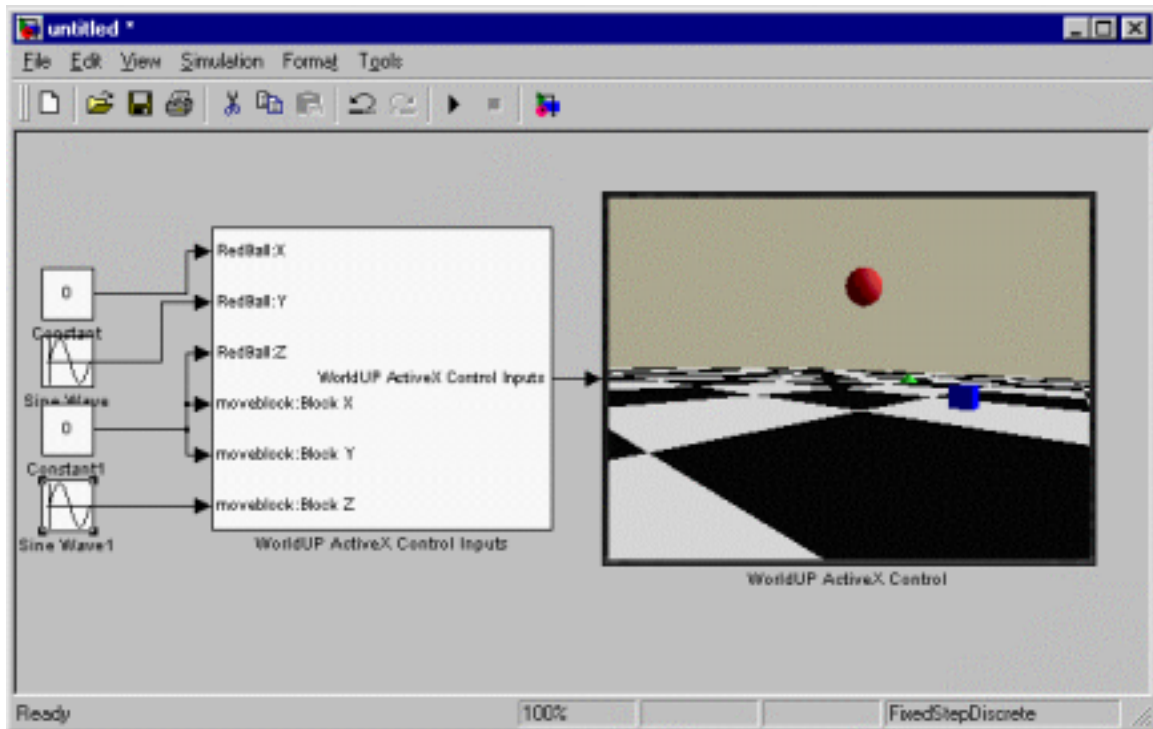
% make sphere bounce
TimeVector = 0:0.1:20;
YDisplacement = abs(6*sin(TimeVector));

% make block move back and forth
ZDisplacement = 10*sin(TimeVector);

% move our objects (remember that vertical displacement is positive
% down)
for k = 1:length(TimeVector),
    set(RedBall, 'Translation', [0, -Ydisplacement(k), 0]);
    set(BlueBlock, 'Inputs', [0, 0, ZDisplacement(k)]);
end
```

## Load into Simulink

1. Open Simulink and create a new model window.
2. From the VRo Blockset, drag the WorldUp ActiveX Control into the new model.
3. Double-click on the thin black border surrounding the control to open the [mask](#).
4. In the **World Up File** edit field, enter the full path to the location of combo.up.
5. Press OK.
6. Add some simple sources.



If you notice your simulation finishes too quickly, in **Simulation** ► **Parameters**, set the Solver Type to Fixed-Step and specify a Fixed Step size of 0.01.

## Conclusion

From this scenario we learned how to:

- ◆ Expose an object to MATLAB and Simulink.
- ◆ Write a WorldUp subroutine script to control an object from MATLAB and Simulink.
- ◆ Add a disabled block to control our mousing viewpoint.
- ◆ Load our objects in MATLAB.
- ◆ Control our objects in MATLAB with a simple function m-file.
- ◆ Load our objects in Simulink.
- ◆ Control our objects in Simulink through the use of simple sources.

If you have any further questions, do not hesitate to contact [Terasoft Support](#).