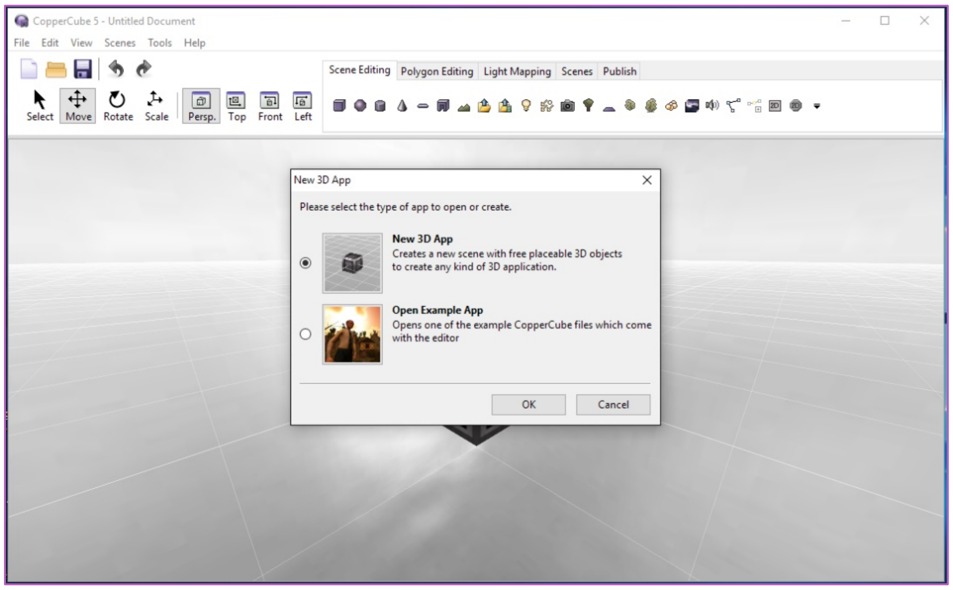
Script	:	CopperCube behavior Script CharacterOnMap
By	:	Ahmed ABDEL SALAM
Contact	:	a.m.abdelsalam@techno-valley.com
Download link	:	http://www.techno-valley.com/ambiera/behavior_CharacterOnMap.zip
Example file	:	http://www.techno-valley.com/ambiera/ccmap_tutorial.zip

Let’s figure out together how this script works.

Creating a new CopperCube Scene

Create a new CopperCube scene.


CopperCube 5 - Untitled Document
File Edit View Scenes Tools Help
Scene Editing Polygon Editing Light Mapping Scenes Publish
Select Move Rotate Scale Persp. Top Front Left
OK
Cancel

New 3D App

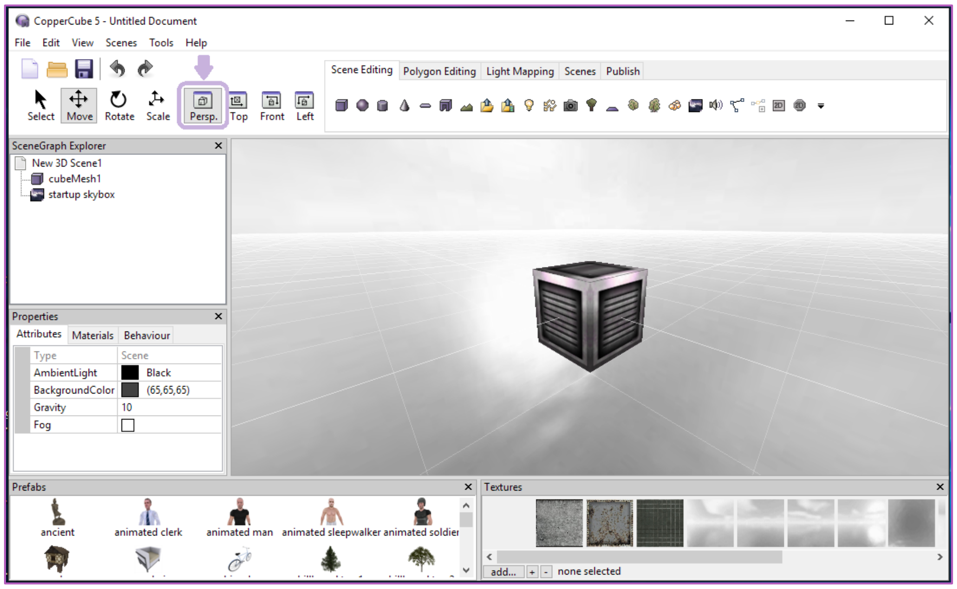
Please select the type of app to open or create.

[✓ New 3D App]
Creates a new scene with free placeable 3D objects to create any kind of 3D application.

Open Example App
Opens one of the example CopperCube files which come with the editor

Figure 1 - New CopperCube Scene

The default view will be the (Perspective View)



Type	Scene
AmbientLight	■ Black
BackgroundColor	■ (65,65,65)
Gravity	10
Fog	☐

Figure 2 - CopperCube Default View

The default scene named “New 3D Scene1” contains the following objects:

- Cube : *CubeMesh1*
- Skybox : *startup skybox*

CopperCube Scene Overview

Let’s have a quick clarification about the axis already defined in CopperCube.

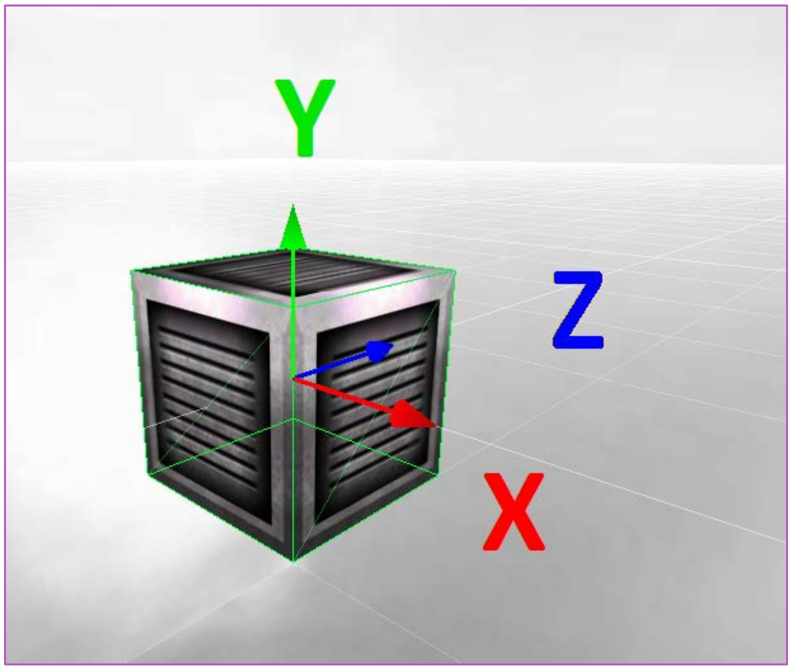
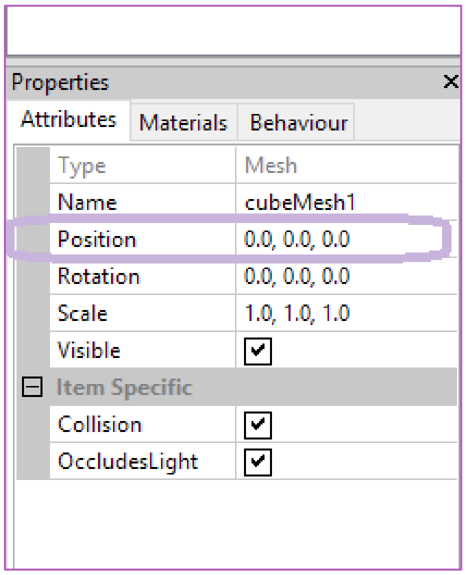


Figure 3 - CopperCube defined XYZ Axis



Properties

Attributes | Materials | Behaviour

Type	Mesh
Name	cubeMesh1
Position	0.0, 0.0, 0.0
Rotation	0.0, 0.0, 0.0
Scale	1.0, 1.0, 1.0
Visible	✓
Item Specific	
Collision	✓
OccludesLight	✓

Figure 4 - Object Properties Window

The Red Arrow represents the X-Axis

The Green Arrow represents the Y-Axis

The Blue-Arrow represents the Z-Axis

On the left pane of the application there is a window called "Properties". It contains all the characteristics of each selected object. In this case for example it shows the properties of the *CubeMesh1*.

Let's select from the topic menu the "Move Icon". This icon instructs the engine to move your selected object.

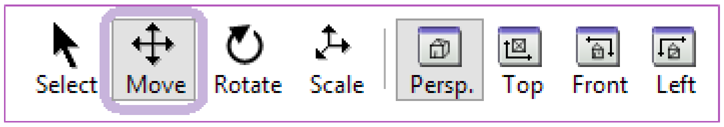


Figure 5 - CopperCube Shortcuts Menu

By default the *CubeMesh1* is located at (0, 0, 0) position.

Whenever you move the *CubeMesh1*, the position for example will change relatively.

Before working on the scene, let's have quick information about CopperCube Scenes.

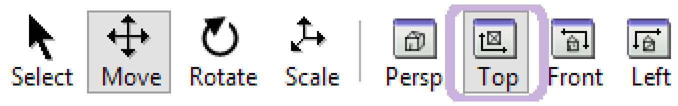


Figure 6 - Change the Current View to Top View

This will allow you to see the scene from above. As you can see in the screenshot below, there are a lot of squares. These squares are called the Grid. Using the grid will let you to be aware about the distances and the size of the scene. (We'll see this later while explaining the script)

If you can't find these white lines (Grid) you can get it from View Menu → "Toggle 3D Grid" or by Pressing Ctrl + G.

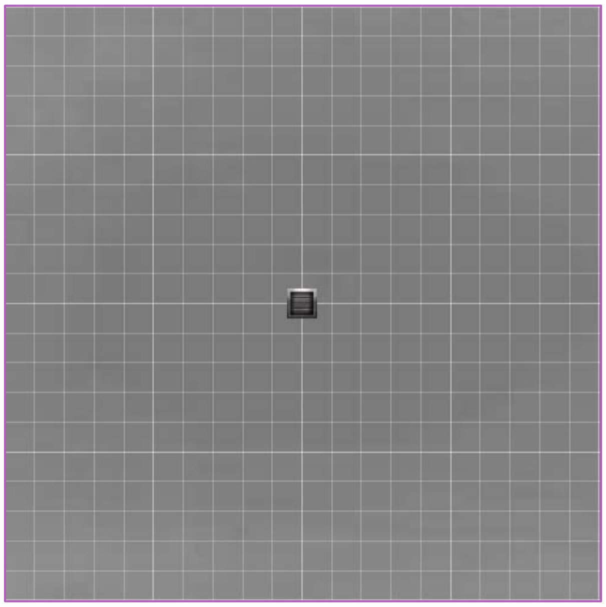


Figure 7 - CopperCube Scene Grid

Each white square is called a Tile. The width is by default 20 units.

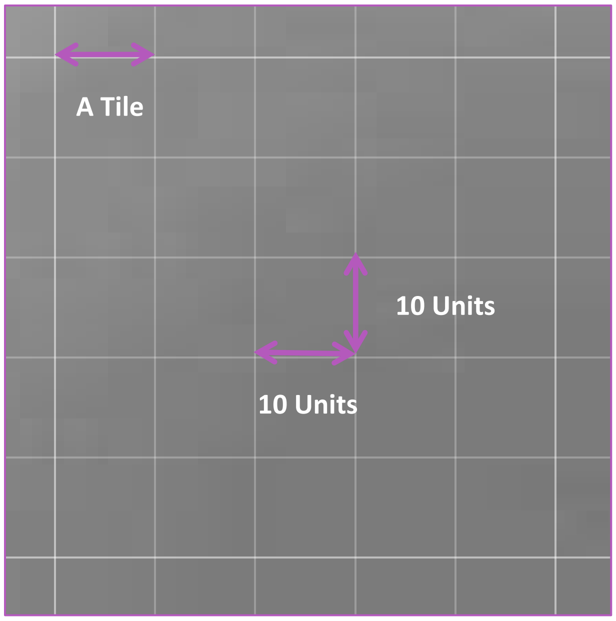


Figure 8 - Grid Definitions

Preparing the CopperCube Scene

Now, let's add a new surface (Plane) to the current scene located at the center of the scene (0, 0, 0) and represents 10 tiles of the scene and as per the screen above we'll use the standard (Width x Height) equals to (20 x 20)

- Add a new plane

- Click on the plane icon under the scene editing tab (Figure 9)

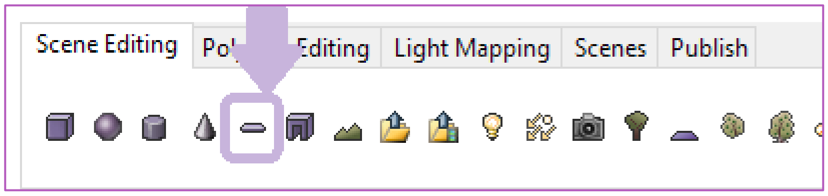
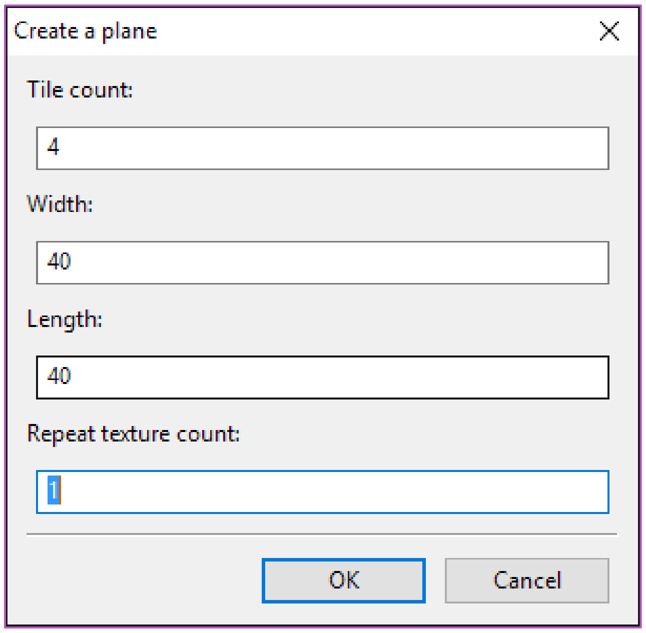


Figure 9 - Creating a plane

- If we're going to add a new surface represented by 10 Tiles, size of 100 units of width and length of 100 units and the shape and the texture of the plane are repeated only once, so we need to enter the input as follow:

Tile count	: 4 (4 Tiles to be used at least for width and height)
Width	: 40 (Using 40 Units for the whole width)
Length	: 40 (Using 40 Units for the whole length)
Repeat Texture Count	: 1 (Means the plane texture will be repeated only once)



Create a plane

Tile count: 4

Width: 40

Length: 40

Repeat texture count: 1

Figure 10 - Creating a New Plane

Please note that the more logical input the more correct output that it will appear. For example you can't say 1 Tile count and width is 40 otherwise the real output will use of course 4 tiles. The final shape will look like the next figure (Figure 11)

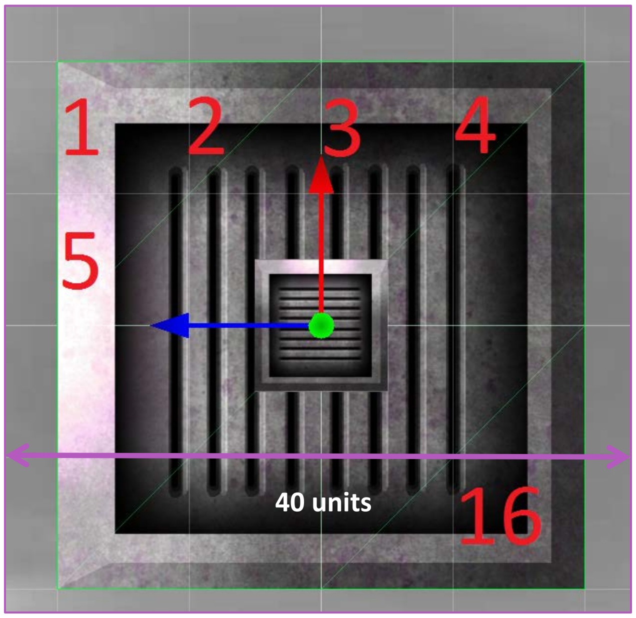


Figure 11 - Generated Plane

From the generated plane above we can conclude the following:

Total Number of tiles used for the whole width is 4 Tiles (40 divided by 10).

Total Number of tiles used for the whole length is 4 Tiles (40 divided by 10).

The total number of tiles used = Total Number of tiles used for the whole length x Total Number of tiles used for the whole width.

The next step is to change the plane texture for *planeMesh1* to be able to differentiate between it and the *CubeMesh1*

Make sure that you've selected the *planeMesh1.*

At the bottom of CopperCube Interface there is a little window called texture. So, we can use one of the existing textures by double clicking it or browse for a texture you already have it by clicking the add button.

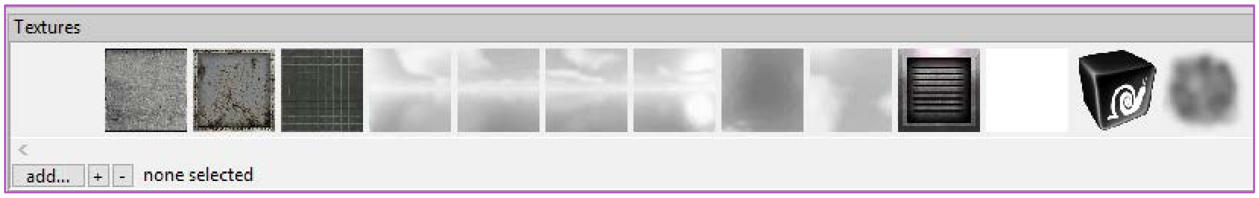


Figure 12 - Select a texture

In the scene explorer, you will have a root node named "*New 3D Scene1*" and 3 sub objects.

A Cube named "*cubeMesh1*", a Plane named "planeMesh1" and a Skybox named "*startup skybox*" which is actually the scene background.

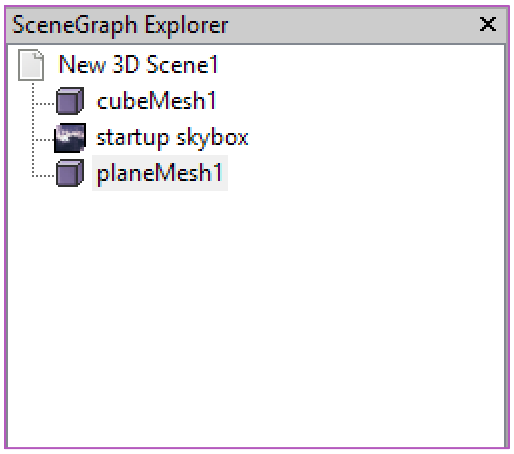


Figure 13 - Scene objects

Finally, the scene from the top view will look like the figure below (Figure 14)

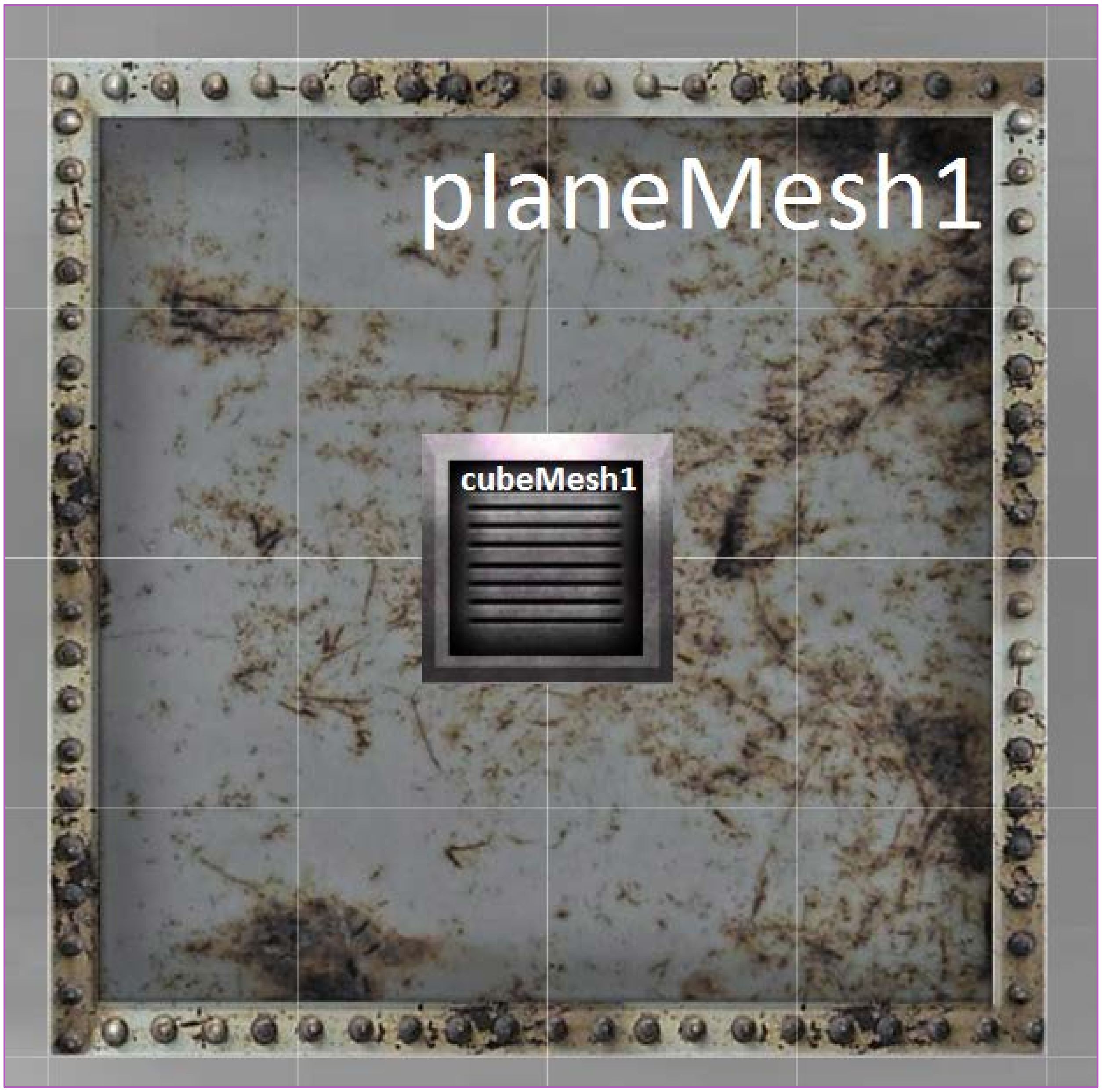


Figure 14 - Scene from Top View

Now, we need to create a larger plane. So, delete the plane *planeMesh1* you've just created. Then create a new one with the following characteristics.

Tiles	: 100
Width	: 1000
Length	: 1000
Repeat Texture Count	: 1

This will create the *planeMesh1* again on the whole grid.

Script overview

This script allows you to follow a selected character in your game on a part of the screen. This part might contain the map for the scene you're watching.

Before applying the script, let's see the whole script and its explanation.

This is a behavior script.

The first part contains simple comments. (Comments are simply descriptive text)

```
// Script : Follow Main Character On Scene Map
// Created by : Ahmed Abdel Salam   (Techno-Valley)
// e-mail : a.m.abdelsalam@techno-valley.com || support@letterskingdom.com
// Inspired by Radar Script (Jaime A. Zegpi B.)
```

This code below contains the integration part with the CopperCube Engine. (Figure 15)

```
/*  <behavior jsname="behavior_CharacterOnMap" description="Follow Main
Character On Scene Map">
      <property name="scene_width" type="int" default="1000"
description="Maximum Scene Width" />
      <property name="scene_height" type="int" default="1000"
description="Maximum Scene Height" />
      <property name="radar_width" type="int" default="20" description="In
Percent in relation to the whole scene width on screen" />
      <property name="radar_height" type="int" default="20" description="In
Percent in relation to the whole scene width on screen" />
      <property name="target_character" type="scenenode" description="Select
the character to follow on screen" />
    </behavior>
*/
```

This behavior has to be added to the scene root node.

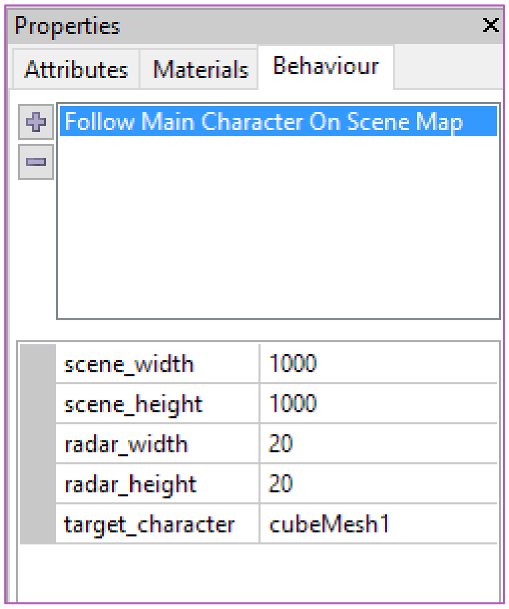
Then make sure you entered the following.

You will have to enter the following in CopperCube:

Scene Width, Scene Height, Radar Width, Radar Height and the Target Character

Scene width = 1000 is the default scene width of the scene.
Scene Height= 1000 is the default scene height of the scene.

(Figure 16)



scene_width	1000
scene_height	1000
radar_width	20
radar_height	20
target_character	cubeMesh1

Figure 15 - Using the Behavior Script

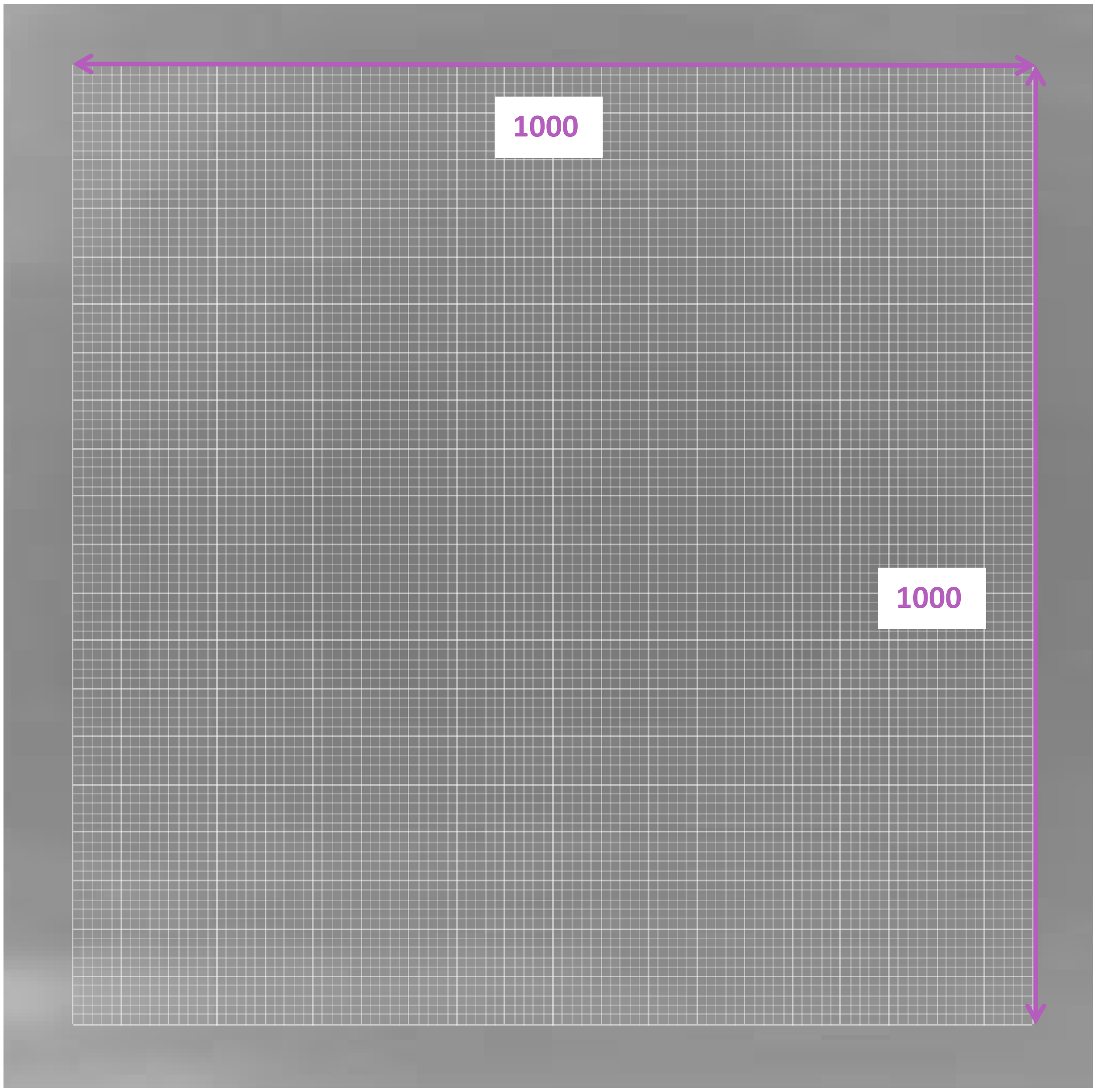
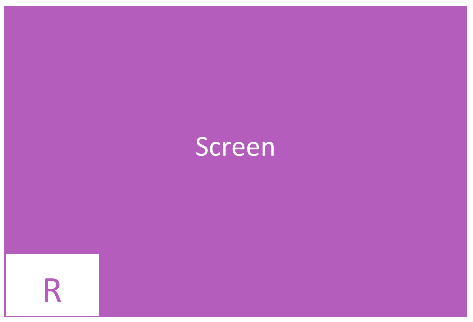


Figure 16 - Full width and Height

The radar width and height is the radar appearance on the output screen and how much will be the percentage of the usage of the whole screen. (In this case 20% of height and 20% of width)



```
behavior_CharacterOnMap = function()
{
};


behavior_CharacterOnMap.prototype.onAnimate = function(node, timeMs){

// Set the value of maximum scene width in v_scene_width (v_ means value)
   var v_scene_width  = this.scene_width;
// Set the value of maximum scene height in v_scene_height (v_ means value)
   var v_scene_height = this.scene_height;

// Get the scene main character name property to plot and follow on the map
   var character_objectname =
ccbGetSceneNodeProperty(this.target_character,"Name");

// The radar width percentage in proportion to the whole screen width (v_
means value)
    var v_radar_width   = this.radar_width;
// The radar height percentage in proportion to the whole screen height (v_
means value)
    var v_radar_height  = this.radar_height;


function ShowOnMap(){

// Set the level height used
   var level_length = v_scene_height;
// Set the level width used
   var level_width = v_scene_width;

// Get the Screen Width Resolution in Pixels
   var screen_width = ccbGetScreenWidth();
// Get the Screen Height Resolution in Pixels
   var screen_height = ccbGetScreenHeight();
// Calculate the radar width in Pixels
   var radar_width = v_radar_width*screen_width/100;
// Calculate the radar height in Pixels
   var radar_height = v_radar_height*screen_height/100;

// Set the radar top position to left bottom you can change the map location
as you wish.
// Radar Top position = Screen Height Max Resolution - Radar Full Length
   var radar_top = screen_height - radar_height;

// Radar left position = 0 oriented to the left of the screen
   var radar_left    = 0;

// Draw a square at the target position to act as the map.
// The line below draws a colored square
      // Use the example values below
      // 0x77ffffff = White Square
      // 0x77000000 = Black Square
      // 0x77ff0000 = Red Square
      // 0x7700ff00 = Green Square
      // 0x770000ff = Blue Square
// You can use your desktop calculator to convert from decimal to hexadecimal
values
// Draw the square using the following line – just remove the comment
// ccbDrawColoredRectangle(0x77ff0000, radar_left , radar_top , radar_width,
screen_height);
// You can also add your custom map drawn in relevance to your
scene/environment as well using the below command instead
// add the ccbDrawTextureRectangle command line
// [ without Alpha Channel (No Transparency) ]
// ccbDrawTextureRectangle (filename, radar_left, radar_top, radar_width,
w_height);

// or add the ccbDrawTextureRectangleWithAlpha command line
// [ with Alpha Channel (With Transparency) ]
   ccbDrawTextureRectangleWithAlpha ("map-new.png", radar_left, radar_top,
radar_width, screen_height);
// Where filename is the image for your scene from top view for example.
// Make sure to set the picture full width and full height from top view

// sc_pos_x : scene position on x axis
// sc_pos_z : scene position on z axis
// plot the character relative position on the map
// The character is shown as a little yellow square
// The square color can be changed from the ccbDrawColoredRectangle command
line as usual.
// Square size can be changed sc_pos_x-3 can be sc_pos_x-5 for example (the
square will be bigger)

// Get the target character object from its name already selected.
   var scene_character = ccbGetSceneNodeFromName(character_objectname);
// Get the target character object position
   var pos = ccbGetSceneNodeProperty(scene_character, "Position");
// Plot the square on the relative position
   var sc_pos_x            = ((pos.x*-1)*radar_height) / (level_length);
   var sc_pos_z            = ((pos.z*-1)*radar_width) / (level_width);
   var pin_top_left_w      = ((sc_pos_x-3)+(radar_width/2));
   var pin_top_left_h      = ((sc_pos_z-3)+(radar_top+(radar_height/2)));
   var pin_bottom_right_w  = ((sc_pos_x+3)+(radar_width/2));
   var pin_bottom_right_h  = ((sc_pos_z+3)+(radar_top+(radar_height/2) ));
   ccbDrawColoredRectangle (0x77ffffff, pin_top_left_w, pin_top_left_h,
pin_bottom_right_w, pin_bottom_right_h);
   }
// Refresh frames
      ccbRegisterOnFrameEvent(ShowOnMap);
}
```