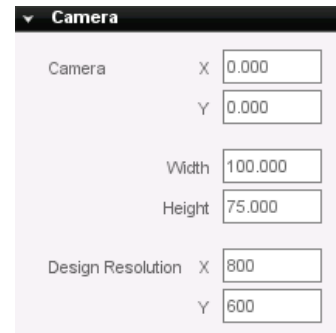


Camera and Mouse Tutorial

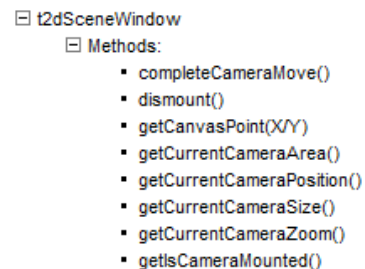
The Basics:

To access the camera properties, deselect all objects and click the edit tab while in the TGB editor. Here you will find three properties, x and y position, width and height, and design resolution. The last property, design resolution, determines the size of the objects you are importing into the editor along with the default size of the application when executed.



Using the Documentation:

There are a lot of methods that are premade for use with the camera. You can find a list in the documentation. In the table of contents open Reference/TGB Reference/t2dSceneWindow/Methods and you will find a list of premade methods for the camera. I've picked two methods to use out of this list:



`getMousePosition()`

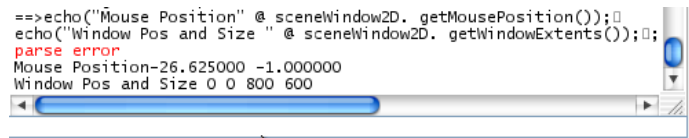
`getWindowExtents()`

Open the console and enter the following code. This will display the values retrieve from the above two methods.

Console

```
echo("Mouse Position" @ sceneWindow2D. getMousePosition());  
echo("Window Pos and Size " @ sceneWindow2D. getWindowExtents());
```

The first method outputs the mouse position in world units. The second method outs the position of the camera and dimension of the screen. If you would like to get individual values from these methods, you can use the `getWord()`



method. Now that you are familiar with finding methods in the documentation it is just a matter of testing out different methods and finding out what they do.

Making the Camera Follow an Object:

You should now have a small idea on how to use the camera so let's do something practical with it. Let's make a behavior that mounts the camera to an object. Create a new script file and name it CameraMount.cs and add the following code.

Console

```
if (!isObject(CameraMountBehavior))
{
    %template = new BehaviorTemplate(CameraMountBehavior);
    %template.friendlyName = "Camera Mount";
    %template.behaviorType = "Camera";
    %template.description = "Mounts Camera to object owning this behavior";

    %template.addBehaviorField(offset, "offset of mount "x y", string, "0 0");
    %template.addBehaviorField(force, "force of mount", float, 50.0);
}

function CameraMountBehavior::onLevelLoaded(%this)
{
    sceneWindow2D.mount(%this.owner,%this.offset,%this.force,true);
}
```

Attach this behavior to any object you would like the camera to follow. Also experiment with the force of the mount because it will change the camera mount effect greatly.

Using Mouse Callbacks:

If you search the documentation for Reference/TGB Reference/t2dSceneWindow/Callbacks there will be a list of callbacks. These callbacks are used differently than the methods used earlier in this chapter. To use them simply use the following format.

Mouse Callbacks Example

```
function sceneWindow2D::Callback(%arguments)
{
    //code
}
```

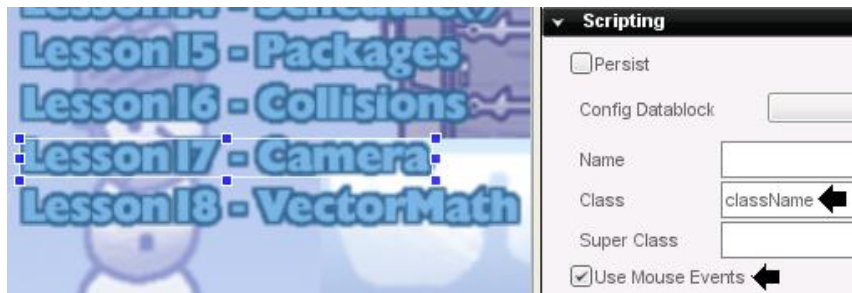
Callbacks are methods that you can declare that are called when a certain event happens. The next example is displaying “you clicked” every time the mouse is pressed down.

Mouse Click Example

```
function sceneWindow2D::onMouseUp( %this, %modifier, %worldPos, %mouseClicks )
{
    echo("You Clicked");
}
```

Clicking an Object:

To make an object clickable you must change some fields in the scripting tab. Select an object, the one below is from the Torque Script Interactive Tutorial. Once selected change its class to the desired name and enable the use mouse events checkbox.



After that is done you can now program callbacks for this object like onMouseMove() and onMouseDown(). In the next example we will make it so we can click the object.

Clicking an Object Example

```
function className::onMouseDown(%this, %modifier, %worldPosition, %clicks)
{
    echo("You click this object" @ %this);
}
```

The above code will display you clicked this object followed by the object ID of the object clicked when an object has the class, “className”, and its use mouse events active.