MaxScript

This page provides information on how MaxScript can be used with Chaos Phoenix.

Overview

During the simulation process you can directly access the Simulator's content using Phoenix's MaxScript functions.

You can use these functions to script advanced simulation rules, or to e.g. control a simulation on a remote machine that runs using only a Simulation License.

Callback Functions

These functions are available when you enable **Use Script** in the Simulation rollout of a Phoenix Simulator.

At different moments of the simulation, Phoenix will call them and you can add your custom MaxScript code inside.

This MaxScript code is separate for each different Simulator and is saved to the 3ds Max scene file.

For example, it is possible to add script for starting another Simulator automatically once a simulation ends, or start of a sequence or single frame rendering (see this example).

The following functions are available:

Function	Description
OnSimulationBegin	Called after the initialization of the simulator is done and before first execution.
OnSimulationStep	Called before each simulation step, after the interaction with the scene.
OnSimulationEnd	Called after the end of the simulation. The simulation core, referred to by the 'this' variable, would be already destroyed during this callback so it should not be accessed.
OnNewFrame	Called after each frame export.

Global Variables

The following global variables are initialized before entry in the callback functions:

Variable	Description
this:<simulator>	Points to the simulator that calls the callback function
t:<float>	The simulator's internal time
dt:<float>	The simulator's internal step duration

Global Functions

These are the Phoenix-specific functions you can call from 3ds Max's MaxScript listener, or from the Callback Functions above, or for example from a MaxScript file that you pass to 3ds Max on startup.

Each of the simulation grid channels (temperature, velocity, smoke etc.) exists in two instances - one instance is in the simulation core, while the simulation is running, and the other instance is in the loaded simulation cache files, which you can also access even when no simulation is running. The functions in this section can access both the simulation core and cache files. If the first argument passed to the function specifies a Phoenix Simulator node, then the function accesses the cache file data. If there is no explicit Phoenix Simulator node specified, the function accesses the currently running simulation core, and it's not ambiguous because only one Simulator can be started at a time.

Note that the simulation core exists only during the simulation and can be accessed only in the Callback Functions using the **this** global variable.

Functions	Description
A_SetSystem Parameters: system: <integer> Available options are: 0 - Objectspace 1 - Worldspace 2 - Grid(voxel)space Return value: none	Specifies which coordinate system will be used.

A_Inject Parameters: where:<point3> amount:<float> [temperature:<float>] [smoke:<float>] [velocity:<point3>] [RGB:<point3>] Return value : none	Injects fluid in a given point. Using this function, you can create your own procedural sources. The result of the function CAN NOT be achieved by calling one or more A_SetX functions, because they do not affect the quantity of the fluid, but only the parameters carried by the fluid. The injection of fluid in some point causes changes in the content only of the nearest 8 cells, but produces an outgoing flow in the entire grid. Nevertheless the function is not slower than the ordinary A_SetX function, because the outgoing flow appears later, when the simulation is executed. If A_GetV function is executed immediately after A_Inject in some near point, the velocity will not be changed.
A_SetV Parameters: x:<integer> y:<integer> z:<integer> velocity:<point3> Return value : none	Sets the velocity of a cell. If a Simulator name is used, the function will write to the loaded cache, otherwise the function will write into the grid of the running simulator, if any.
A_SetRGB Parameters: x:<integer> y:<integer> z:<integer> RGB:<point3> Return value : none	Sets the RGB of a cell. If a Simulator name is used, the function will write to the loaded cache, otherwise the function will write into the grid of the running simulator, if any.

A_SetT Parameters: x: <integer> y: <integer> z: <integer> temperature:<float> Return value : none	Sets the Temperature of a cell. If a Simulator name is used, the function will write to the loaded cache, otherwise the function will write into the grid of the running simulator, if any.
A_SetSm Parameters: x: <integer> y: <integer> z: <integer> smoke:<float> Return value : none	Sets the Smoke of a cell. If a Simulator name is used, the function will write to the loaded cache, otherwise the function will write into the grid of the running simulator, if any.
A_SetFl Parameters: x: <integer> y: <integer> z: <integer> fuel:<float> Return value : none	Sets the Fuel of a cell. If a Simulator name is used, the function will write to the loaded cache, otherwise the function will write into the grid of the running simulator, if any.
A_GetFl Parameters where: <Point3> Return Value: <float>	Gets the Fuel in a given point. If a Simulator name is used, the function will read from the loaded cache, otherwise the function will read from the running simulator, if any.

A_GetV Parameters where: <Point3> Return Value: <point3>	Gets the Velocity in a given point. If a Simulator name is used, the function will read from the loaded cache, otherwise the function will read from the running simulator, if any.
A_GetRGB Parameters where: <Point3> Return Value: <point3>	Gets the RGB in a given point. If a Simulator name is used, the function will read from the loaded cache, otherwise the function will read from the running simulator, if any.
A_GetT Parameters where: <Point3> Return Value: <float>	Gets the Temperature in a given point. If a Simulator name is used, the function will read from the loaded cache, otherwise the function will read from the running simulator, if any.
A_GetSm Parameters [node:<Simulator>] where: <Point3> Return Value: <float>	Gets the Smoke value in a given point. If a Simulator name is used, the function will read from the loaded cache, otherwise the function will read from the running simulator, if any.
A_StartSim Parameters node:<Simulator> [cache: <String>] [startframe: <integer>]	Starts the simulation. Passing just the simulator node will start a new simulation. If you pass the path to a cache file, the effect is that of the Load & Start button in the Simulation rollout: the simulation state will be loaded from the cache and the simulation will continue from the specified Start Frame. If you manually pass the **startframe** index too, it takes precedence over the Load function and the simulation will be **restored** from the given frame, the same way that the Restore button works. This function will decide between simulation and re-simulation depending on the state of the Particle Resimulation and Grid Resimulation switches.
A_StopSim Parameters node:<Simulator>	Stops the simulation
A_Wait Parameters node:<Simulator>	This function halts the execution of the script until the specified simulator has finished running. Usually this function is used along A_StartSim when you want to run certain actions after the simulation is finished. ⚠ Use this function extremely carefully because it does not block the GUI of 3ds Max.

A_CreateParticle Parameters Particle group: <string> where: <Point3> [Radius: <float>] [Velocity: <Point3>] Return Value: none	Creates a new particle in a given position with given properties.
A_Freeze Parameters: x: <integer> y: <integer> z: <integer>	Freezes the given cell. The frozen cell acts as a Solid object.
A_Unfreeze Parameters: x: <integer> y: <integer> z: <integer>	Unfreezes the given cell. Keep in mind that the simulator counts the freezing operations and you have to execute the same number of unfreezing operations to successfully unfreeze a cell.
A_QuickSetup Parameters: setup: <integer> Available options are:	Creates a Quick Setup with the selected objects (the Quick Setup presets can be applied over a selection of several objects).

0
-
fire

1
-
fu
el
fire

2
-
e
x
pl
o
si
on

3
-
g
a
s
ol
in
e
e
x
pl
o
si
on

4
-
la
rg
e
s
m
o
ke

5
-
c
ol
d
s
m
o
ke

6
-
ci
g
ar
et
te
s
m
o
ke

7
-
c
a
n
dle

8
-
cl
o
u
ds

9
-
ta
p
w
at
er
1
0
-
m
ilk
1
1
-
b
e
er
1
2
-
c
of
fee

1
3
-
h
o
n
ey

1
4
-
li
q
ui
d
c
h
o
c
ol
ate

1
5
-
bl
o
od

1
6
-
p
ai
nts

1
7
-
in
k
in
w
at
er
1
8
-
w
at
er
fa
ll
1

9 - o c e an Return Value: none	
A_LoadRenderPreset Parameters node: <Simulator> preset path: <String>	Loads the specified render preset file. This is the same as using the Render Presets... menu from the Rendering rollout of the Simulator. The preset path can be a full path or use the *$(dir)* macro, which denotes the current scene file's directory. For example, the following will load "preset.tpr" from the current scene directory: A_LoadRenderPreset (getnodebyname "PhoenixFDFire001") "$(dir)\preset.tpr"
A_SaveRenderPreset Parameters node: <Simulator> preset path: <String>	Saves a Phoenix render preset to a file with the current render and preview settings. This is the same as using the Render Presets... menu from the Rendering rollout of the Simulator. The preset path can be a full path or use the *$(dir)* macro, which denotes the current scene file's directory.
A_LoadSimPreset Parameters node: <Simulator> preset path: <String>	Loads the specified simulation preset file. This is the same as using the Simulation Presets... menu from the Simulation rollout of the Simulator. The preset path can be a full path or use the *$(dir)* macro, which denotes the current scene file's directory. For example, the following will load "preset.tpr" from the current scene directory: A_LoadSimPreset (getnodebyname "PhoenixFDFire001") "$(dir)\preset.tpr"
A_SaveSimPreset Parameters node: <Simulator> preset path: <String>	Saves a Phoenix simulation preset to a file. This is the same as using the Simulation Presets... menu from the Simulation rollout of the Simulator. The preset path can be a full path or use the *$(dir)* macro, which denotes the current scene file's directory.
[available since Phoenix FD 4.10.04] A_GetGridSize Parameters node: <Simulator> channel name: <String> Return Value: <point3>	Retrieves the grid size of the running Simulator or the loaded cache file if a simulation is not running. Useful when using adaptive grid, in which case this.xc, this.yc and this.zc will return only the initial grid size. The name of the grid channel is reserved for future use. Currently all grid channels are the same size.

[available since Phoenix 5.01.02 Nightly, Build ID: 2022120531780] A_ExportSimscene Parameters export path: <String>	Exports a simscene file which you can use for simulation at a later time or on a different machine. See this page for more information on exporting .simscene files for Phoenix Standalone simulation.

Global Interface

Phoenix also provides a global interface which can be used to obtain data which is not specific to any particular Phoenix node.

Example usage:
IPhoenix.getCopyrightsString()

The available functions are:

getVersionString Returns a string with the exact Phoenix version.
getTargetString Returns a string with the 3ds Max version and Phoenix build type.
getCopyrightsString Returns a string with all credits and copyrights for Chaos Phoenix and all 3rd party software used.

Per-Node Functions

Phoenix also provides an interface for getting grid data and loading render presets by typing in directly the Simulator node.

Example usage:
$PhoenixFDFire001.getFuel (Point3 15 15 15)

The available functions are:

getVersion – *[available since Phoenix FD 4.10.04]* Gets the currently installed Phoenix version.
setCoordSys – Specifies which coordinate system will be used.

Available options are:

0 - Object space

1 - World space

2 - Grid (voxel) space

getGridSize – *[available since Phoenix FD 4.10.04]* Retrieves the grid size of the running Simulator or the loaded cache file if a simulation is not running.
loadRenderPreset – Loads a previously saved Phoenix render preset from a file.
saveRenderPreset – Saves a Phoenix render preset to a file with the current render and preview settings.
loadSimPreset – Loads a previously saved Phoenix simulation preset from a file.
saveSimPreset – Saves a Phoenix simulation preset to a file with the current render and preview settings.
getVelocity – Gets the Velocity value in a given point.
getRGB – Gets the RGB value in a given point.
getTemperature – Gets the Temperature value in a given point.
getSmoke – Gets the Smoke value in a given point.
getFuel – Gets the Fuel value in a given point.
reloadFrame – Forces a load on the cache frame for the current timeline time.
getFrameInfo – Returns a string with information on the currently loaded cache frame. This is the info that you can find in the Simulation rollout's *Cache File Content* box.