
IMPROVED PARTICLE SYSTEM LITE

An improved particle system for Torque3D, featuring graph based emitters and compatibility for Twillex animation.

INDEX

Version 1.0	1
Introduction	2
Installing the new files	3
Changes to the particleEmitter class	4
The graphEmitter class	4
The physics	6
Adding more objects	7
Syntax	8
Functions	8
Binary operators	9
Other operators	9
Dynamic variables	10
Callbacks	10
Composite functions	11
The meshEmitter class	12
IPS Lite 1.0	14
Update 2.0.0	14
Update 1.0.2	14
Update 1.0.1	14
Examples	15
2 Dimensional	15
3 Dimensional	18

VERSION 1.0

INTRODUCTION

This resource is a result of my work with extending Torque's particle system.

Check out my blogs [Adding animated particles and animated shaders to Torque](#)¹ and [Spheres, spheres it's all spheres!](#)²
For some insight in the development of these particle features!

I want to thank Ingo Berg for the muParser, it is an excellent tool and very fast! It is fantastic to find gems like these released publicly!

And I want to thank Charles Patterson, for writing the Twillex engine which inspired me to all of this!

Also, I want to thank you! And everyone else who uses this resource! Please, if you create something fantastic with this resource, do show off for everyone to see!

Preferably at this thread: <http://www.garagegames.com/community/forums/viewthread/130709>

¹ <http://www.garagegames.com/community/blogs/view/21696>

² <http://www.garagegames.com/community/blogs/view/21704>

INSTALLING THE NEW FILES

The folder structure matches the one in the source folder so just move the files into the folders it should be easy to figure out.

After moving the files into your engine folder, remember to include them in visual studio.

In Visual Studio add a new filter in “engine/t3d/fx” called “ImprovedParticle” and add all the files in the ImprovedParticle folder.

Then overwrite the particle.h, there is only one change in this file. I added the relPos point so it is easy to merge with your existing files.

CHANGES TO THE PARTICLEEMITTER CLASS

THE GRAPHEMITTER CLASS

The graphEmitter class is what makes this resource unique to my previous ones.

The graphEmitter class is pretty similar to the particleEmitter class, except it have another set of values.

standAloneEmitter (Boolean)




If true, this datablock is not connected other datablocks of the same type

[This resource](#)³ covers what this change is and why it is usefull.

When this Boolean is on, the sa settings from the particleEmitterNode overrules those from the particleEmitterDatablock.

▼ Independent emitters		
standAloneEmitter	<input checked="" type="checkbox"/>	
sa_ejectionPeriodMS	1	▲▼
sa_periodVarianceMS	0	▲▼
sa_ejectionVelocity	0	
sa_velocityVariance	0	
sa_ejectionOffset	0.08	
sa_thetaMin	0	
sa_thetaMax	0	
sa_phiReferenceVel	0	
sa_phiVariance	0	

³ <http://www.garagegames.com/community/resources/view/21697>

Expression	
xFunc	0
yFunc	0
zFunc	0
funcMax	2000 
funcMin	0 
timeScale	1
ProgressMode	
Reverse	<input type="checkbox"/>
Loop	<input type="checkbox"/>

xFunc, yFunc and zFunc

These 3 values defines the equations which tells the graphEmitter where the next particle should be spawned.

E.g. if xFunc is set to 1 it will spawn 1 meter from the graphEmitterNodes center.

The distance can also be scaled via the sa_ejectionOffset value.

$$dst = Func * sa_ejectionOffset$$

funcMax and funcMin

These 2 values defines the boundary of the graphEmitter, if you set Min to 1000 and Max to 2000 the t value will raise between those two.

$$funcMax > t > funcMin$$

timeScale

This value defines the rate at which the t value increases.

ProgressMode

This setting defines how the graphEmitter increases the t-value.

Options is:

- ParticleCount
- ByTime

As the names suggest, ParticleCount increases the t value based on the number of particles that have been emitted. So for every particle that gets emitted it increases by 1.

ByTime increases based on the amount of milliseconds elapsed since the last particle was emitted.

In most cases, these two modes gives the same outcome, however if something slows the engine down this setting have an influence on how the emitter should continue, should it continue from where it was slowed down? Or continue as it have been running properly the whole time?

Reverse

This setting will let the t value descend instead of ascend. You can see the example g_downSpiralNode for an example of how it works.

Loop

This setting will define whether the emitter should loop between the two boundary settings, or just keep going even after it reaches a boundary.

Note: Loop must be set to false if you want to handle the boundaries yourself (See Callbacks)

THE PHYSICS

The physics system is the bread and butter of version 2.0.0

▼ Physics	
☐ attractedObjectID (2)	
[0]	4169
[1]	0
☐ Attraction_offset (2)	
[0]	0 0 0
[1]	0 0 0
☐ AttractionMode (2)	
[0]	attracted ▼
[1]	attracted ▼
☐ Amount (2)	
[0]	1
[1]	1
attractionrange	50
sticky	<input type="checkbox"/>

attractedObjectID

Here you can specify one or more objects that can affect the particles physics (all objects can collide with particles tho)

Attraction_offset

The offset from the origin of the object which the particles should seek to.

AttractionMode

Got 3 settings:

Attract

No attraction

And repulse.

No attraction does nothing.

Attract draws the particles closer.

Repulse pushes the particles away.

Amount

The amount of attraction force to apply to the particles. You can consider this as the amount of speed to apply.

attractionRange

The max range at which particles will be attracted, notice that particles move slower the farther away they are from the object, therefore this will also affect the speed of the particles as they will seem closer to the object the bigger the range is.

Sticky

If this is true, particles will stick to the emitter, this is useful if you don't want the particles to leave a trace when moved, especially in high velocity objects can this come in handy.

ADDING MORE OBJECTS

Adding more objects for the emitters to interact with, doesn't slow the engine or take up network traffic until you utilize them in each specific emitter. The fewer you have, the better overview you will have.

To create space for more objects just go to the top of GraphEmitterNode.h

Find: `static const int attobjectCount = 2;` and change it to the amount of objects you want.

SYNTAX

I am using [muParser](http://muparser.sourceforge.net)⁴ for parsing the equations, the syntax for these equations is pretty straight forward.

FUNCTIONS

Name	Argc.	Explanation
<code>sin</code>	1	sine function
<code>cos</code>	1	cosine function
<code>tan</code>	1	tangens function
<code>asin</code>	1	arcus sine function
<code>acos</code>	1	arcus cosine function
<code>atan</code>	1	arcus tangens function
<code>sinh</code>	1	hyperbolic sine function
<code>cosh</code>	1	hyperbolic cosine
<code>tanh</code>	1	hyperbolic tangens function
<code>asinh</code>	1	hyperbolic arcus sine function
<code>acosh</code>	1	hyperbolic arcus tangens function
<code>atanh</code>	1	hyperbolic arcur tangens function
<code>log2</code>	1	logarithm to the base 2
<code>log10</code>	1	logarithm to the base 10
<code>log</code>	1	logarithm to the base 10
<code>ln</code>	1	logarithm to base e (2.71828...)
<code>exp</code>	1	e raised to the power of x
<code>sqrt</code>	1	square root of a value
<code>sign</code>	1	sign function -1 if x<0; 1 if x>0
<code>rint</code>	1	round to nearest integer
<code>abs</code>	1	absolute value
<code>min</code>	var.	min of all arguments
<code>max</code>	var.	max of all arguments
<code>sum</code>	var.	sum of all arguments
<code>avg</code>	var.	mean value of all arguments

⁴ <http://muparser.sourceforge.net>

BINARY OPERATORS

Operator	Meaning	Priority
=	assignement	-1
&&	logical and	1
	logical or	2
<=	less or equal	4
>=	greater or equal	4
!=	not equal	4
==	equal	4
>	greater than	4
<	less than	4
+	addition	5
-	subtraction	5
*	multiplication	6
/	division	6
^	raise x to the power of y	7

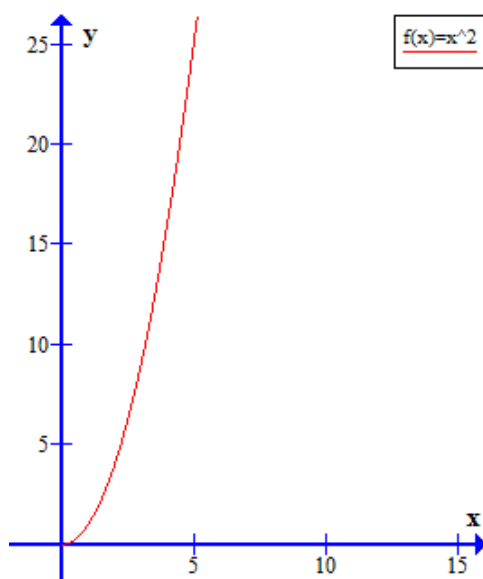
OTHER OPERATORS

Operator	Meaning	Remarks
? :	if then else operator	C++ style syntax

These tables is taken from http://muparser.sourceforge.net/mup_features.html#idDef2

The increasing variable is defined by the letter t.

Writing $x_{\text{Func}} = t^2$ results in a curve like this



DYNAMIC VARIABLES

To add dynamic variables, you have to add a dynamic field with a special naming pattern.

The naming pattern goes like this:

CoordinateVarIndexToken = value

xVar1a = 2

First you specify what coordinate uses this variable, then you specify what index it has and lastly you set it to the token which you use to refer to variable and the value that token has.

You can only use values here you can't use expressions like $2*2$.

Supports 2 digits in index and 2-3 characters in token. 3 characters if only 2 digits was used (5 in total)

This is easy to change in the source code if you want more characters or digits. Look for onDynamicModified.

(Syntax updated in v1.0.2)

CALLBACKS

Callbacks is datablock based, if you have a datablock like this:

```
datablock GraphEmitterNodeData(g_spiralNode : g_DefaultNode)
{
    xFunc = "cos(t/50)*t*0.02";
    yFunc = "sin(t/50)*t*0.02";
    zFunc = "0";
};
```

You would define the callback like this:

```
function g_spiralNode::onBoundaryLimit(%this, %node, %Max)
{
    if(%Max)
        %node.Reverse = true;
    else
        %node.Reverse = false;
}
```

%this is the datablock that is calling the callback.

%node is the GraphEmitterNode which triggered the callback

%Max is a boolean specifying whether it was the top boundary (true) or the bottom bounday (false)

COMPOSITE FUNCTIONS

If you need to create composite functions, use the if then else operator in muParser.

An example:

```
datablock GraphEmitterNodeData(g_compSpiralNode : g_DefaultNode)
{
  xFunc = "t<500 ? cos(t/50)*t*0.02 : t<1000 ? cos(t/50)*t*0.1 : t<1500 ? cos(t/50)*t*0.02 : cos(t/50)*t*0.1";
  yFunc = "t<500 ? sin(t/50)*t*0.02 : t<1000 ? sin(t/50)*t*0.1 : t<1500 ? sin(t/50)*t*0.02 : sin(t/50)*t*0.1";
  zFunc = "t/20";
};
```

This create a small spiral when:

$$t < 500 \vee 1000 \leq t < 1500$$

And it will create a way bigger spiral when:

$$t > 1500 \vee 1000 > t \geq 500$$

THE MESH_EMITTER CLASS

The meshEmitter positions the particles on the vertices or faces of a mesh.

▼ MeshEmitter	
emitMesh	4189
evenEmission	<input checked="" type="checkbox"/>
emitOnFaces	<input checked="" type="checkbox"/>

emitMesh

The name or ID of the object which mesh should be used for emission.

evenEmission

Bool, if true the emission of the particles is evened out over the whole mesh.

This has two different effects depending on whether you are emitting on vertices or on faces.

As vertices do not have an area, one cannot be bigger than another. This means that instead of having the emitter emit linearly from the first vertex to the last, it will instead pick a random vertex.

If you have emitOnFaces turned on, faces with a larger area will have a bigger chance of having a particle emitted on them. So a face twice the size of another, will receive twice as many particles. To avoid taking too much memory / performance it has been simplified a lot tho so a face 1/3 bigger than another face wont emit 1/3 more particles. But it works very well anyways, it is hardly noticeable that it is not precise.

See an example on page 23.

emitOnFaces

If true particles will be emitted on the faces of the mesh if false they will be emitted on the vertices of the mesh.

A practical usage for meshemitters is especially things on fire. Drop a meshemitter on a building and it will look like it is on fire as the particles will emit from the building.

IPS PRO AND OTHER INFORMATIONS

The IPS Pro is the commercial version of the IPS. The 'full' version which will contain several other emitters, although I haven't publicly announced any information about it yet.

I will release information about the IPS Pro gradually through FuzzyVoid Studios news channels. As I am writing this, the primary news channel is facebook on our page: <https://www.facebook.com/pages/FuzzyVoid-Studio/255103567940730?ref=hl> This is the primary news channel, and all news about the IPS Pro will come there first as well as running development statuses. If you are interested in having transparency in the development you should follow our facebook page. I will make the occasional blog as always of course!

The official FuzzyVoid website: <http://fuzzyvoidstudio.com/> (Currently under development)

You should check out the T3D Library tho:

<http://fuzzyvoidstudio.com/t3d/browse>

My youtube channel:

<http://www.youtube.com/user/ijhujhu/videos>

You can find various videos about the IPS on my youtube channel.

If you make some cool particle effects, you can post them here:

<http://www.garagegames.com/community/forums/viewthread/130709>

IPS Lite won't be extended with any significant features from this patch, as I will focus my work in IPS Pro. But if you encounter any bugs or missing documentations in the IPS Lite you can email me at:

LukasPJ@FuzzyVoidStudio.com

And I will fix it ASAP

CHANGELOG

IPS LITE BUGFIX 1

- Fixed some urgent bugs with some models in some builds that I hadn't experienced at first.
- Added the optimization options from the CE

IPS LITE 1.0

- Added meshemitters.
- Fixed a bug where GraphEmitters wouldn't load values that were specified in the new GraphEmitter function but not in the datablock.
- You can now choose between either putting the id of the object or the name of the object in: AttractedObjectID.
- Cleaned the folder structure a lot.
- Added a little more documentation in GraphEmitter class, still needs to be cleaned.

UPDATE 2.0.0

- Added lots of particle physics.
- Raycast based particle collision.
- Attracted particles
- Repulsive particles
- Easy to customize

UPDATE 1.0.2

- Added comments and documentation to the c++ code
- Simplified the c++ code a bit
- Updated the dynamic variables syntax

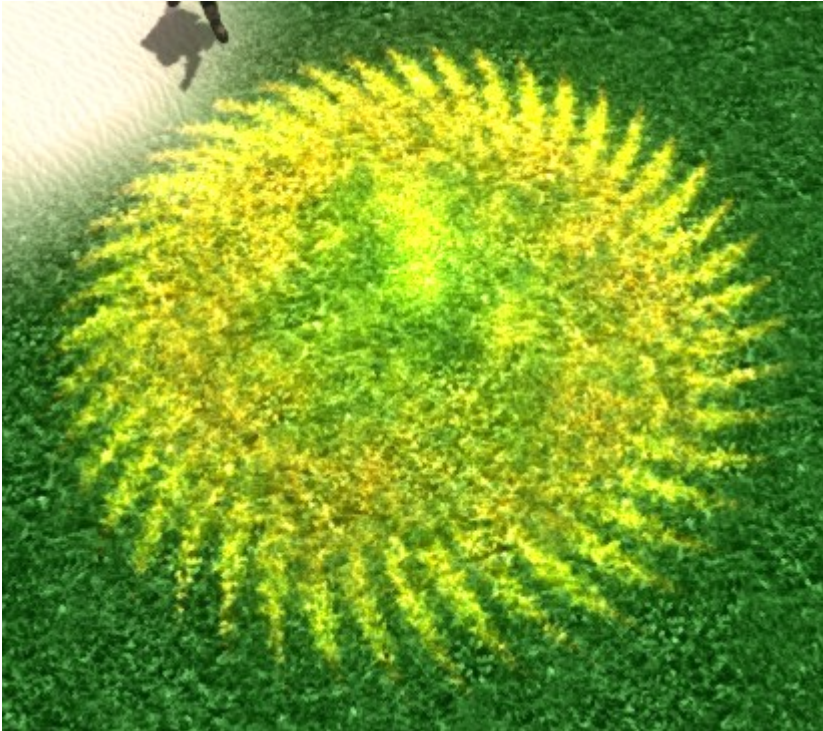
UPDATE 1.0.1

- Now takes the emitter rotation into account when emitting new particles
- Fixed a possible issue where emitters wouldn't delete when you called %emitter.delete()
- Added dynamic variables
- Added reverse Boolean
- Added loop Boolean
- Fixed timescale
- Added onBoundaryLimit callbacks

EXAMPLES

2 DIMENSIONAL

g_wavesNode

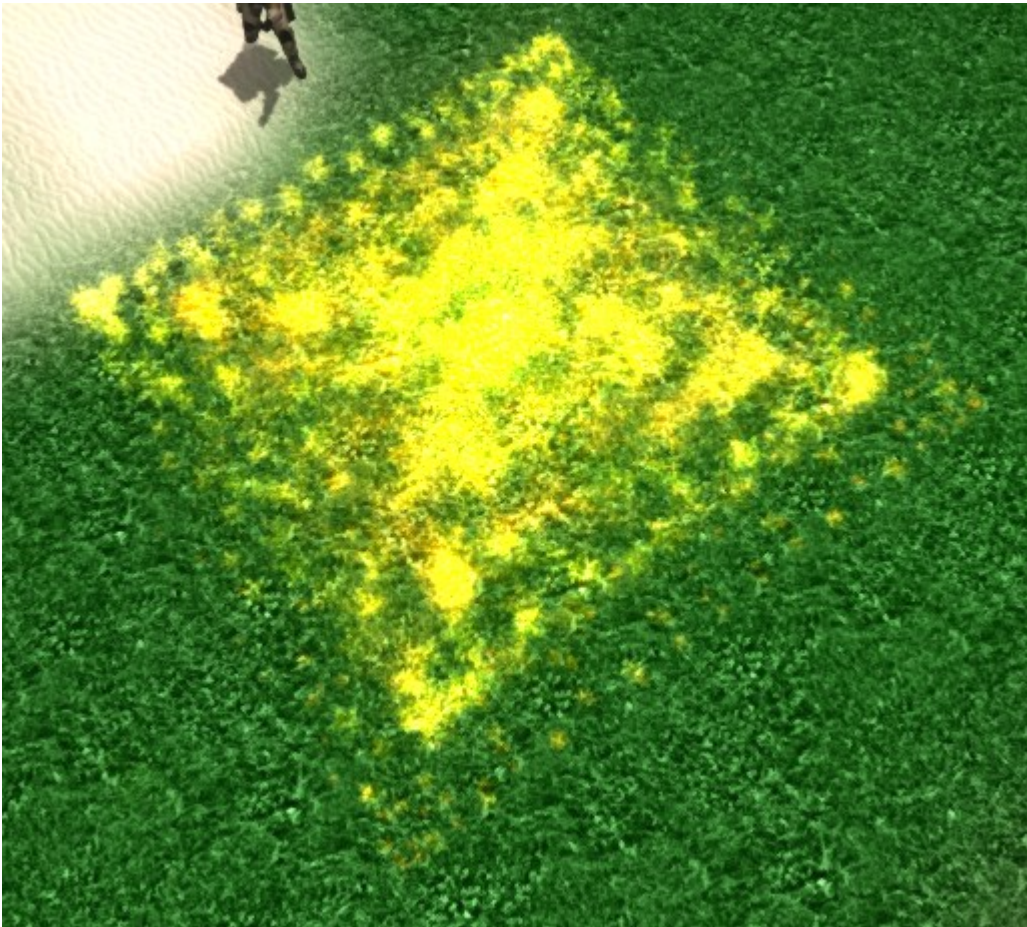


$$x = \cos(t) * t * 0.02$$

$$y = \sin(t) * t * 0.02$$

$$z = 0$$

g_squareSpiralNode

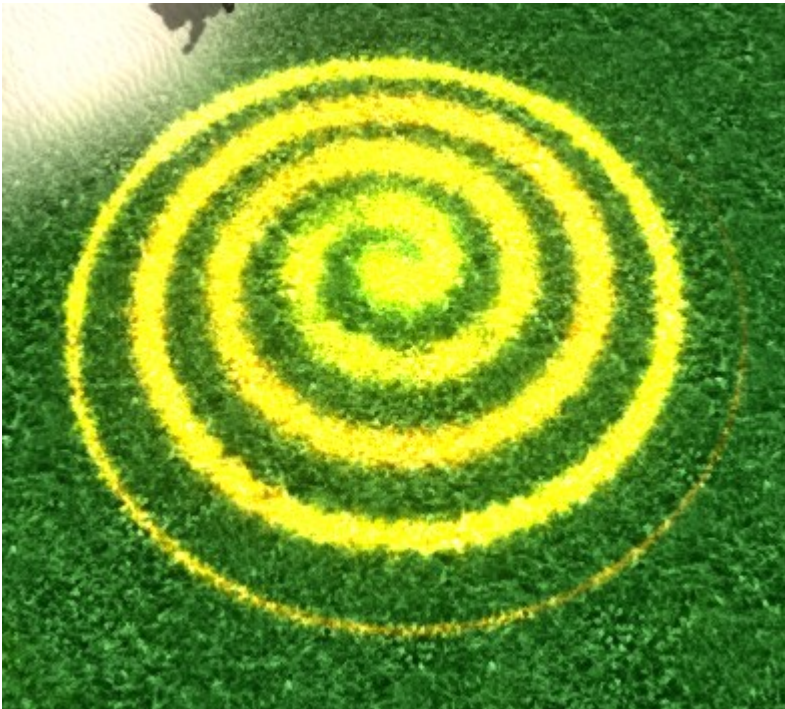


$$x = \cos(t/50) * t * 0.02$$

$$y = \sin(t) * t * 0.02$$

$$z = 0$$

g_spiralNode



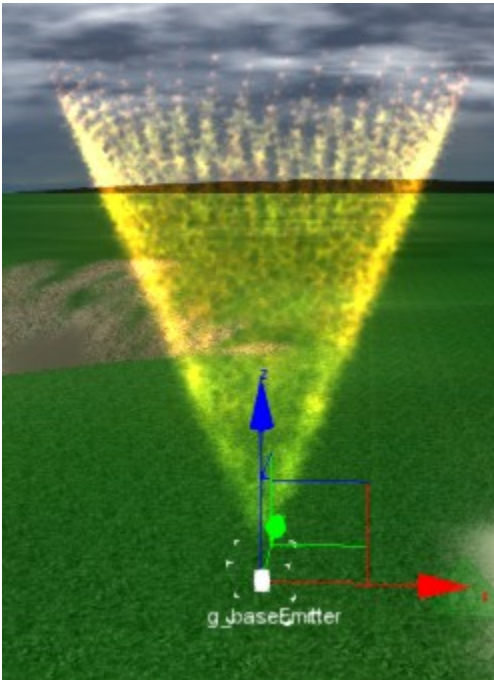
$$x = \cos(t/50) * t * 0.02$$

$$y = \sin(t/50) * t * 0.02$$

$$z = 0$$

3 DIMENSIONAL

g_upWavesNode



$$x = \cos(t) * t * 0.02$$

$$y = \sin(t) * t * 0.02$$

$$z = t/20$$

g_upSpiralNode

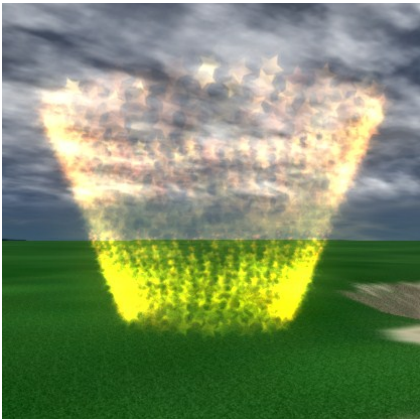


$$x = \cos(t/50) * t * 0.02$$

$$y = \sin(t/50) * t * 0.02$$

$$z = t/20$$

g_upWavesNode with boundaries



$$x = \cos(t) * t * 0.02$$

$$y = \sin(t) * t * 0.02$$

$$z = t/20$$

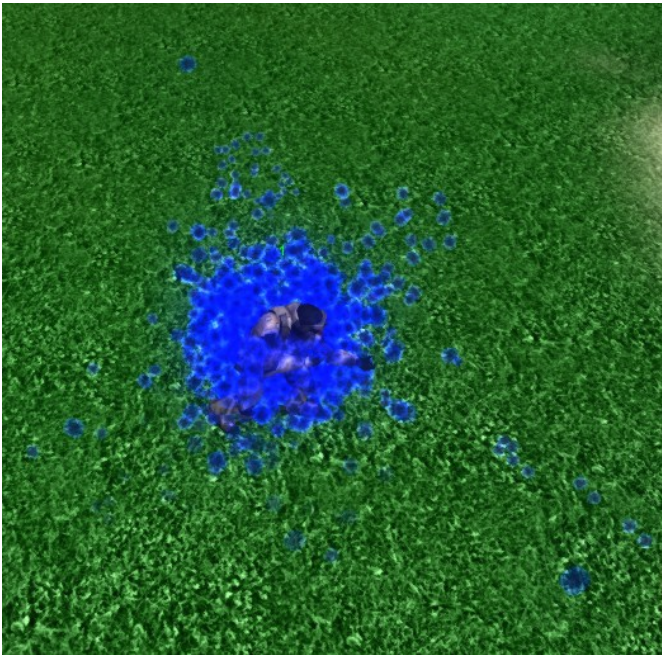
$$funcMin = 1000$$

$$funcMax = 2000$$

flameSpiralSpellNode : Callback usage example



Attracted particles



Meshemitter

