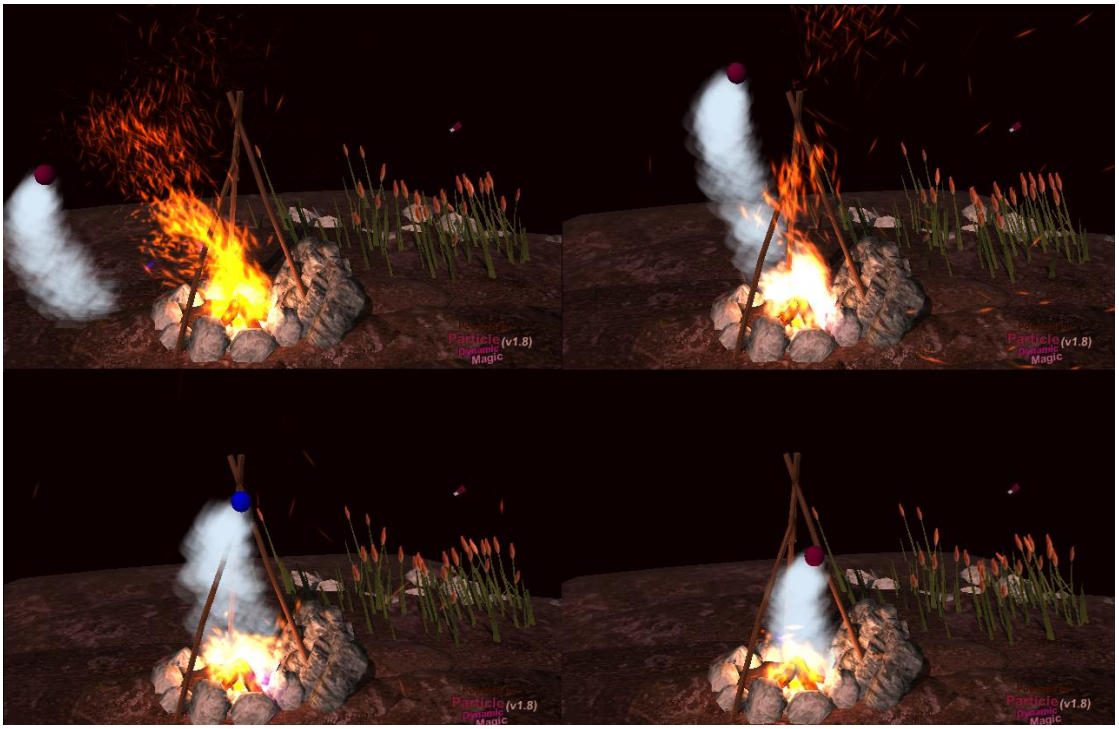
Particle Dynamic Magic 2



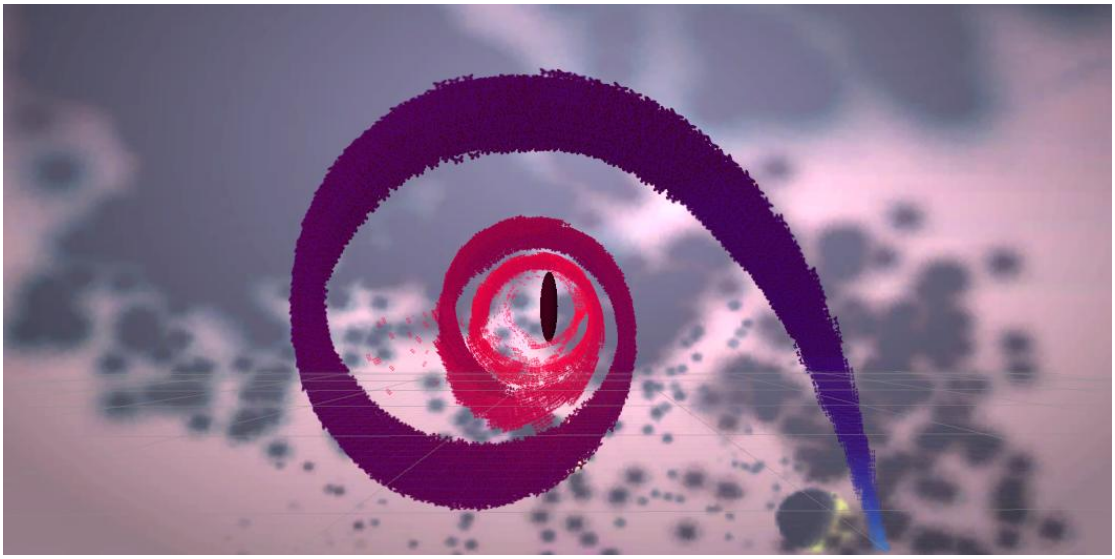
This is a pack devoted to dynamic – interactive particle effects with a focus on magical effects like fire shields-swords-staves and dynamic nature effects like waterfalls and growing grass-flower fields. A global dynamic attractor/deflector class has been developed in order to allow particles to be manipulated on a whole scene level from any point the user may choose. Orbital functions and the particle dynamics of Unity are also used to complete the effects. The pack contains more than 40 scenes with ready to use effects for your game. Load the proper scene and copy elements to your project or create new prefabs. Prefabs for all effects are also provided. **Version 1.2 update adds Splines, Turbulence, Skinned, Projection and Gameobject emission modes, 45 new prefabs and 3 mini game demos. Version 1.3 update adds a Transition manager, a Music module and masked mesh emission. Version v1.5 update takes the pack to a whole new level, with movie style dynamic effects, like dynamic particle and gameobject spreading, ice melting and advanced collision and fire propagation effects, all optimized to use a single particle pool for each behavior. Also many new features are added to existing modules, the Transition Manager can now change particle type dynamically and allow scheduling of transitions and with the new Prefab Manager all effects are one click away.**

The new Version 2.0 of the pack adds a whole range of new systems, enhancements and effects that almost double the pack value and abilities.

Instructions

In order to use the dynamic attractor/deflector class, the user must add the "**AttractParticles"** script to the gameobject that needs to attract or repel particles.

In v1.2 effects of Turbulence, dynamic vortex creation and Gravity are also available.



Global attract/deflect manipulators

Options:

By default the script will attract near particles. Enable Turbulence to create dynamic vortexes (Vortex option) and sinusoidal motion.

Turbulence options:

Disable Force (v2.0): Disable attract/repel forces.

Turbulence: Add Turbulence effect. "Use Exponent" must be disabled.

Turbulence strength: Amplify the effect reach

Turbulence frequency: Adjust the effect frequency

Axis affected: Affect particles with turbulence by axis, add weights for all three axis

Splash effect: Add a splash like effect

Vortex motion: Add dynamic vortexes to recreate turbulent behavior

Vortex count: Control the number of vortexes created dynamically

Vortex life: Control the life of dynamically created vortexes

Vortex angular velocity: The spin each vortex will rotate close particles with. **This setting also applies to Gravity effect.**

Vortex scale: The magnitude of the effect. **This setting also applies to Gravity effect.**

Limit influence: Will limit the effect to near particles only.

Vortex center size: Add a visual representation of the vortex center by adjusting its particle size

Vortex center color: Color the vortex centers

Show vortex: Show the vortex centers

Coloring options:

Color force: Add color to the particles affected by the script forces

Force color: The color for the above option

Forces options (Gravity – Repel – Attract):

Use exponent: Amplify the global forces effects to cover various world scales and gravity. **Gravity only works in "Use exponent" mode.**

Gravity pull: Add gravity effect to particles.

Swirl gravity: Add a swirl effect to gravity mode

Bounce gravity: Add a bounce effect when particles get very close to the attractor, good for emulating atom like behaviors.

Bounce factor: Amplify the bounce effect.

Affect tres: The range the effect affects particles

Gravity planar: Planar gravity effect, fix effect in certain axis

Gravity plane: The axis of rotation for planar gravity and radius factor for each axis.

Gravity factor: The gravitational pull amplitude

Dist Factor: Cutoff distance for particle systems. Systems further than this distance will not be affected even if their individual particles reach the attractor/deflector object.

Enable paint: Paint near the mouse particles in collision surfaces, using the mouse pointer

Affect Distance: Distance from attractor individual particles get affected, keep this number low to increase performance.

Dampen: Dampen the attract/repel and gravity effects. Use higher values when "Use exponent" mode is on.

Smooth attraction: When checked particle motion is smoothed, disable for faster or instant particle attraction

Lerp Velocity: Smoother motion for all effects, but slower.

Repel: Repels near particles

Limit to Y axis: This is the same as adding the "Grass" tag to a particle, but for all particles.

Hit dist: Fine tune painting mode hit distance when mouse is clicked. Use with "Enable paint" option.

Noise options (v1.4):

Vary Turbulence: Apply Perlin or Simplex noise to particles.

Affect by distance (v2.0): Limit perlin-simplex turbulence based on distance (by default applied to whole particle).

Distance factor (v2.0): Distance to apply turbulence.

Perlin enable: Use Perlin noise (Simplex noise is the default).

Splash noise: Create local concentrations of particles.

Noise strength: Affect noise scale

Noise turbulence strength: Affect noise strength

Axis deviation: Apply noise per axis factor.

Splash noise threshold: Control splash effect.

Spiral effect options:

Make moving star: Create rotating trailing star like effects. Affected by Trail options.

Star trail dist: Cutoff distance between the manipulator and scene particle systems

Trail options:

Star trails: Number of circles in the effect

Trail distance: Used for debug purposes

Speed of trail: Rotation speed

Distance of trail: Gradual drop of circle radius

Trail length out : Circle radius

Size of Trail out : Particle size

Distance between trail: Gradual drop of circle radius

Vertical trail separation: Distance between circles

Smooth trail: Smoother trail motion

Optimization and preview options:

Affect by tag: Affect only tagged objects

Tags: Define tags to be affected by the effects.

Exclude tags (v1.4): Use with Affect by tag off, limit influence to particles with the specified tags.

Affect specific (v1.4): Affect only the specified particle systems.

Time to update: Update every nth frame, use to increase performance

Multithreaded: Uses multithreading for speed increase

Enable Preview: Enables editor preview of the effects, must click and then select the particle to see the effect in editor

End life of affected (v1.6): End particle life right away, affects only particles in influence radius of the attractor. Use to enable sub emitters instantly based on attractor location and influence area.

End life (v1.6): Define particle lifetime to assign when the above option is used. Use a value above zero to allow a smoother transition.

Calculations per frame (v2.0): Calculations spread over frames, for performance increase.

Emit in transforms (v2.0): Emit affected particles in selected transforms.

Emit transforms (v2.0): The transforms to emit from.

Transform emit factor (v2.0): Control emission along transform. Higher numbers will conform particles toward the transform emission, lower will allow particles to be governed by other forces.

Keep in position factor (v2.0): Factor controlling how long particles will remain on emit position before it moves along the transform.

Transform velocity (v2.0): Speed along the transform vector.

Length along forward (v2.0): Spread particles along transform forward while they are in “keep in position” mode.

Emit randomize (v2.0): Randomize emission.

Emit specific transform (v2.0): Choose a specific transform to emit from.

Transform emitted (v2.0): Transform to emit from, number corresponds to Transform list IDs.

Front Right Up Factor (v2.0): Change the emission forward vector with respect to transform forward. Front of one is the default forward vector of the transform.

TIP: Tag objects as "VortexSafe” to stop vortexes effect.

NOTE: For grass there is an extra requirement to tag the grass particles as “Grass”, so particles are repelled downwards to the y-axis, rather than in the exact opposite direction.

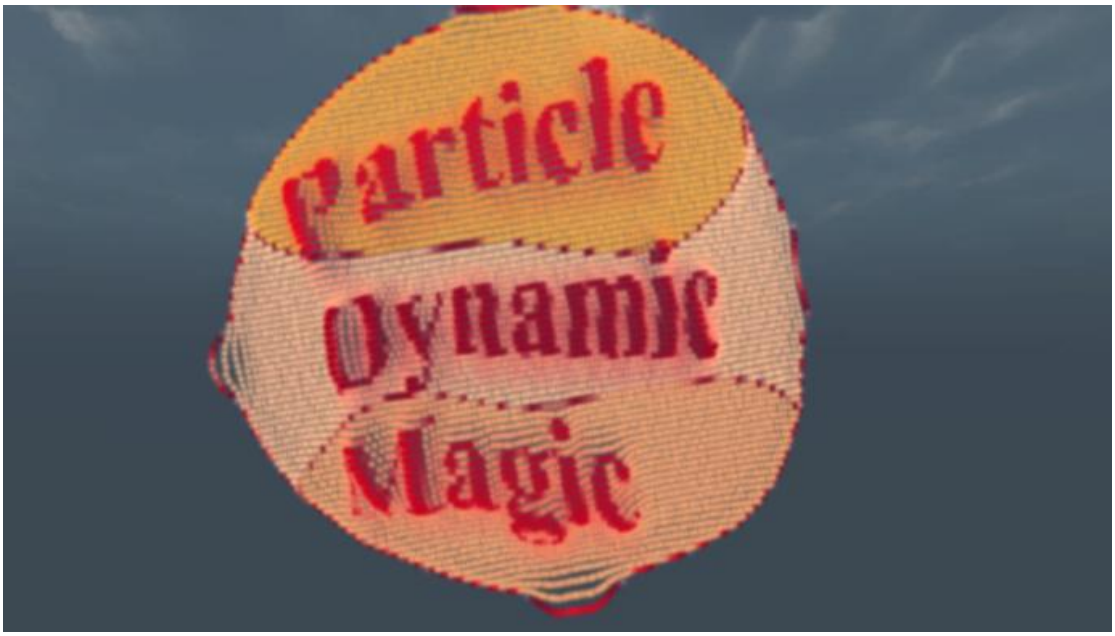


Image based emission

Use **“ImageToParticles”** script on a particle system. Particles are not affected by the particle system options.

Options:

Scale Factor: Scale the image and all particles with it (individual particle size remains the same)

Particle size: Scale individual particles, without scaling the overall system

Image texture: Desired image to use for the effect.

Give depth: Assign a depth based on grayscale of the image.

Depth factor: Depth multiplier.

NOTE: The image must be imported as “Advanced” and have Read/Write flag enabled.

NOTE: Set the particle to “World” simulation and max particles to the image width x height. Set to lower to assemble a smaller part of the image (use for gradual appearance of image).

Dynamic Image based emission

Use **“ImageToParticlesDYNAMIC”** script on a particle system. The particles can be affected by the particle options. Options are similar to the non dynamic version with the following additions:

Additional Options:

Depth image texture: Add a depth map to derive depth from.

Use depth map: Define depth by Depth Image than grayscale.

Changing texture: Allow the texture change during play mode.

Expose highlights: Add grayscale depth on top of Depth map.

Expose factor: Amount of exposure.

Extend Life: Use with “Changing texture” to extend particle life.

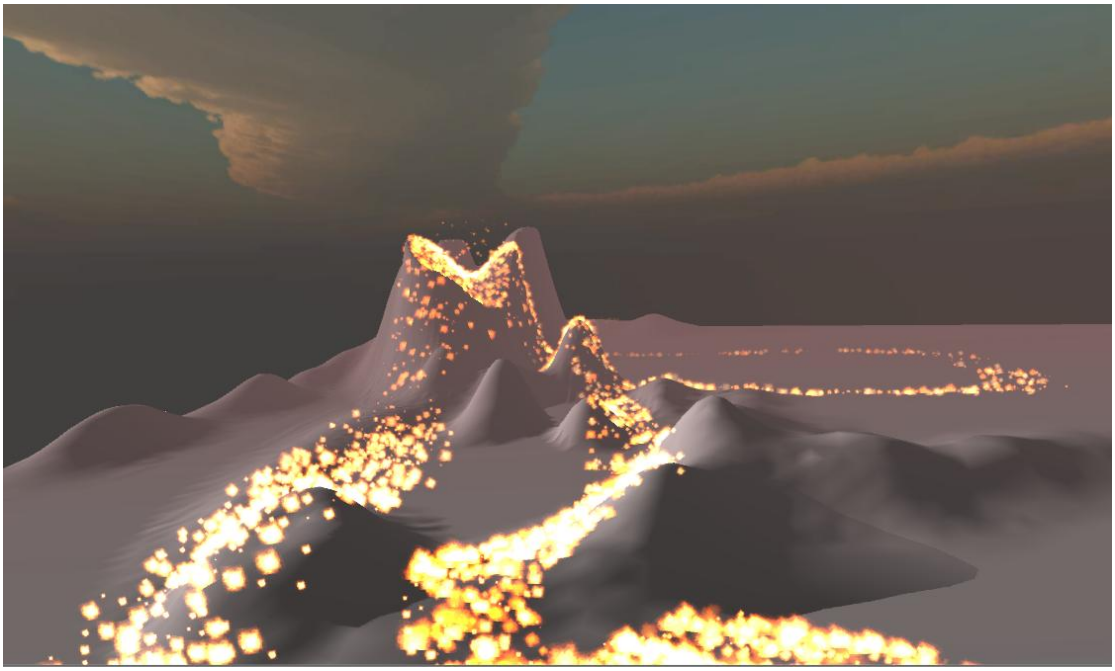
Gravity (v2.0): Gravitate back to the original emit positions

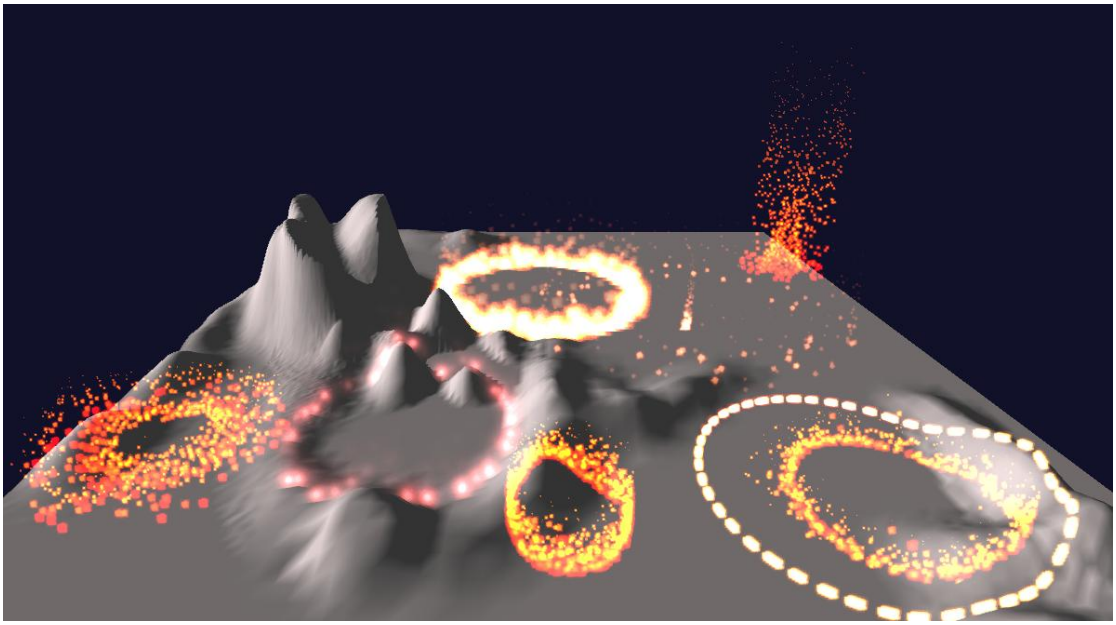
Gravity factor (v2.0): Gravitate back force control.

The system has been developed and tested in Unity 4.3.4 and should work on any Unity 4 version.

Particle Dynamic Magic Version 1.2 Update

In this new version, five new elements are introduced. These include a Spline editor and particle follow system, a system to emit particles from skinned meshes, a gameobject emission set of scripts, particle projection and a terrain conform script to make particles follow the current terrain.





Conform to terrain

Use the "**PlaceParticleOnGround**" script on the particle system which is required to follow the terrain. The effect works with Shuriken particles. The particle must be set to "World" Simulation Space mode.

Options:

P11: Shows the current particle system for debug purposes.

Make circle: Arrange particles in a circular form around the particle object center.

Circle radius: Define the radius of the circle

Loosen circle: Add a probability that the circle particles will leave the ground, use for effects like up going energy or fire, or extra collisions. 1 means zero probability particles will leave the ground (does not apply in circle case, unless Is target is activated).

Is target: Define the particle should be used as a target following the terrain. The particles in the circle will not leave the ground, even if Loosen circle is set above 1.

Spread: The spread of particles across the circle line. Small spread will keep particles in a strict line.

Grass Up-Low threshold is used to limit the relax factor of grass mode. To use grass mode, tag the particle with "Grass" tag.

Relaxed: Same function as loosen circle, is used for non circle setup ("Make circle" is not checked).

Dist above terrain: The distance particles will spawn above terrain. Used for both circular and standard setups of the script.

Outwards (v2.0): Outwards rotation for circle particles

Angled (v2.0): Use an angle in the outward rotation

Radial velocity (v2.0): Apply outwards velocity

Outward speed (v2.0): Speed of the outwards motion

Invert direction (v2.0): Move towards the opposite direction

Conform to terrain, texture sheet varied particles

The "**PlaceParticleSheetOnGround**" script will fix particles to the texture sheet individual textures, it can be used for flowers and particles that need to keep their position and specific initial texture from the sheet.

Note that this can be in a prefab, but will only update the proper preview positions in the editor mode when the OnEnable is activated (when click on the object).

The particle must have the "texture sheet animation" on and the Tile settings must be inserted in the script as well.

The particle must be set to "World" Simulation Space mode.

Options:

Start pos: Shows the registered start position of the system.

Particle Count: How many particles we need emitted.

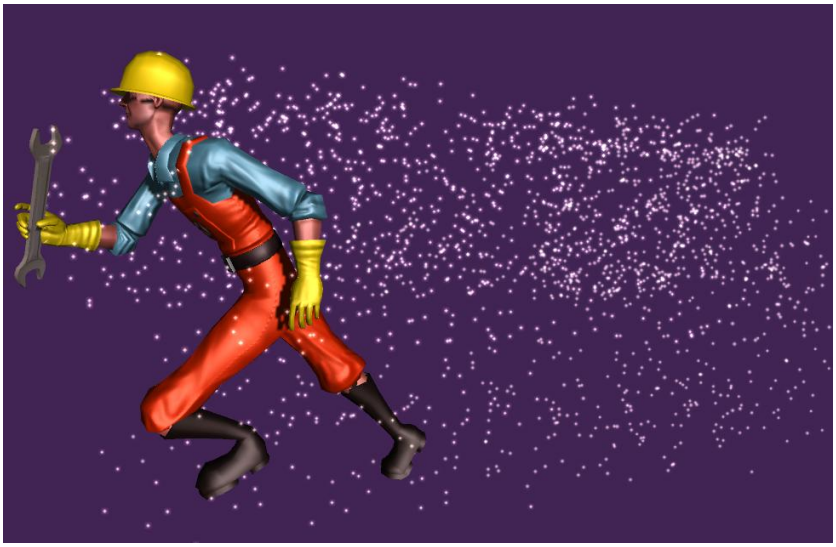
Nbb: The particle system attached for debug purposes.

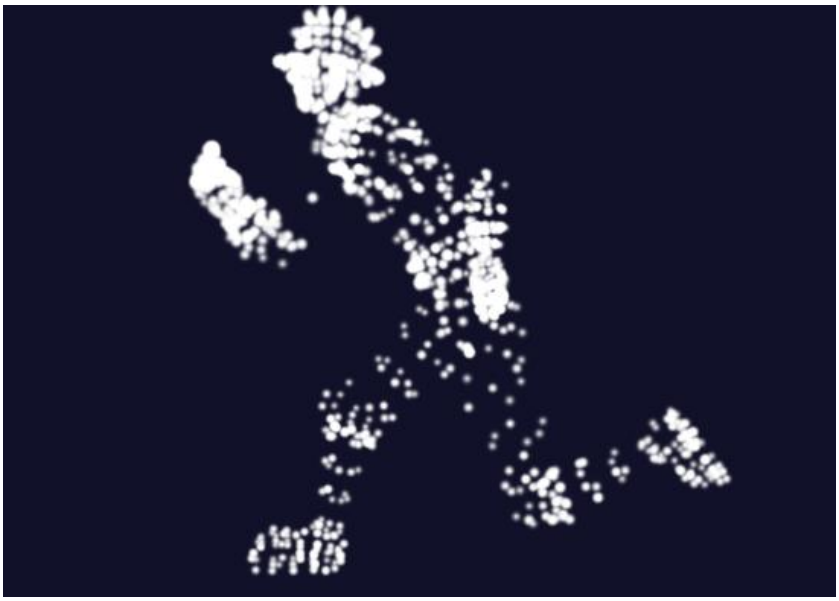
Let loose: Make particles inherit the particle system properties, forces etc

Gravity mode: Add a constant force downwards toward the ground.

Tiles X,Y: The number of tiles in the texture sheet.

Skinned mesh emission





Skinned and mesh emission requires a special setup, so the mesh emitter can inherit the skinned mesh position, orientation and scaling. The setup has the following steps.

1. Create the desired particle effect that needs to be emitted from the skinned mesh. Put "Mesh" and "Triangle" in the Shape parameter.

2. Parent the particle to the object with skinned mesh component (e.g. the "Bip001 Pelvis" in the constructor character that comes with Unity case). Reset its relative position, rotation to zero.

3. Add the "**SKinnedPArticleEmit_StaticNonPrefab**" script to the particle system and the parent (object with skinned mesh component) to the "Emitter" script input.

4. Set initial speed to zero to get particles conformed to the skin.

5. Notes: **This effect can be part of a prefab, but to enable it must first be enabled in the editor. The effect will not work if the object is instanced in play mode directly. This effect is faster than the next one that can be in a prefab normally and auto activates.**

Options:

Start size: Define particle size that will be used in play mode.

P11: Particle system for debug purposes.

Color by vertex: Put a blur hue to the effect.

Emitter: Object with the skinned mesh component, that should also be parent of the particle system with the "**SKinnedPArticleEmit_StaticNonPrefab**" script.

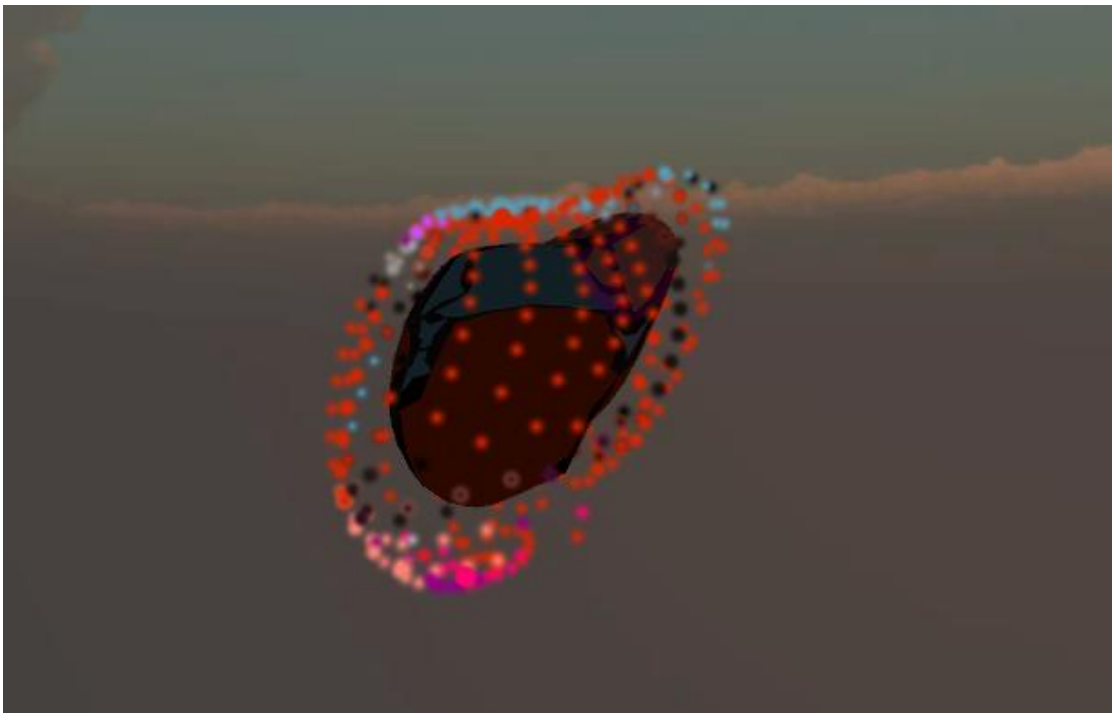
Mesh: The relevant skinned mesh, for debug purposes and editor.

Simple mesh: The relevant simple mesh, for debug purposes and editor.

Animated mesh: The mesh the animation is baked in, for debug purposes and editor.

Coloring: Color for "**Color by vertex" option.**

TIP: After entering the skinned object in Emitter parameter, deselect and re-select the particle to update the particle Mesh.



Skinned Animated-Procedural mesh emission, texture colored

This is the same as the previous script, with two major differences. One is that can be prefabed without the need to be activated in the editor to work and will work without any restrictions and the other that inherits colors from the skinned mesh or simple mesh texture. Attach the "**SkinColoredParticles**" script as described below or use one of the ready prefabs with the setup.

Steps:

1. Create particle system and import mesh to attach to. Use "world" in simulation space parameter and "Mesh" and "Triangle" in "Shape" parameter of the particle.

2. Attach the "SkinColoredParticles" script to the particle. If mesh material has no texture attached, uncheck the "Colored" option. Drag the object with the meshfilter or skinnedmesh to the "Emitter" parameter in the script.

3. Parent particle to the object with the mesh and move to 0,0,0 and reset any rotation to match the shape. Use scale factor to adjust the offset. Put particle speed to zero to make sure particles are on the object at start or for the preview.

4. Check "Let loose" to allow particle motion and adjust with "keep in position" and "keep alive" factors.

Options:

Scale factor: Offset particles from the body

Start size: Define particle size that will be used in play mode.

Emitter: Object with the skinned mesh component that should also be parent of the particle system with the "**SKinnedPArticleEmit_StaticNonPrefab**" script.

Colored: Color by texture (default, use with default particle material or change it for extra blend effects), if unchecked inherits from particle material.

Face Emit: Emit from Shuriken positions. **If this feature is enabled, the prefabs that have the script must be in the scene and enabled in the editor in order to get the shape of the object in the particle system. Do not use this setting for objects that need to be instantiated. If more particles are needed it is better to use the "Every other vertex" option with values under one, in order to increase the particle count.**

Every other vertex: Divide vertices by this number, to scatter gameobjects. In order to enable more particles from each vertex, give it a below one value. The particle count will appear in the Shuriken max particles property.

Particle count: The desired number of particles.

P11: Particle system for debug purposes.

Y Offset: Letloose and Gravity mode height.

Fix initial: Check to keep the current gameobject position

Let loose: Allow particle to inherit from particle system, add gravity and life between 0 and a desired amount to get a constant emission effect (emulate fire etc)

Keep in position factor: Controls the time particle will stay in initial emitted position.

Keep alive factor: Percent of particle life where start lifetime of the particle is restored.

Extend life: Restore each particle life so it never goes out, useful to keep gameobjects static when let loose is used (otherwise they will keep spawning).

Gravity mode: Restore position on skinned body, use to restore shape after a melt effect (let loose plus gravity). Uses "Return speed" parameter to define speed of lerp to the skin position.

TIP: Set speed and size to zero and low respectively before applying the effect.

TIP: Set the particle size between two numbers to get a constant emission effect.

TIP: If particle size cant be changed, disable the script, change size and re-enable.

Skinned-Animated-Procedural Masked emission – deprecated as of v2.0

This is the same as the previous script, with the difference that allows emission from a masked area of the main texture, by defining a mask texture. This script is optimized so that only the unmasked areas will create and emit particles. Also allows for surface normal conformation of mesh particles.

Use the same procedure as the above "SkinColoredParticles" script to setup the system, adding the "SKinColoredMasked" script instead.

Options:

Scale factor: Offset particles from the body

Start size: Define particle size that will be used in play mode.

Mask: Texture used as mask for emission, use transparent mode in texture. PNG file with transparency is recommended.

Low mask threshold: The threshold for transparency, above 255 will cut off all particles, below zero will not mask any particles.

Size by mask: Use mask grayscale to define particle size.

Emitter: Object with the skinned mesh component that should also be parent of the particle system with the "**SKinnedPArticleEmit_StaticNonPrefab**" script.

Colored: Color by texture (default, use with default particle material or change it for extra blend effects), if unchecked inherits from particle material.

Custom color: Use a custom color, instead of the texture colors.

Lerp color: Smooth transition of colors between current and custom, use for Transitions.

End color: Custom color to use.

Face Emit: Emit from Shuriken positions. **If this feature is enabled, the prefabs that have the script must be in the scene and enabled in the editor in order to get the shape of the object in the particle system. Do not use this setting for objects that need to be instantiated. If more particles are needed it is better to use the "Every other vertex" option with values under one, in order to increase the particle count.**

Follow particle: Make particles appear on the particle position, rather than the emitter mesh position.

Velocity towards normal (v1.4): Apply an initial velocity towards the surface normal. Control strength with local speed for each axis with 'Normal Velocity" parameter and use the keep alive and keep in position factors to determine how long the velocity will be applied.

Normal Velocity (v1.4): Control the velocity towards normal strength per axis.

Every other vertex: Divide vertices by this number, to scatter gameobjects. In order to enable more particles from each vertex, give it a below one value. The particle count will appear in the Shuriken max particles property (and is also adjusted based on mask texture).

Particle count: The desired number of particles.

P11: Particle system for debug purposes.

Y Offset: Letloose and Gravity mode height.

Fix initial: Check to keep the current gameobject position

Let loose: Allow particle to inherit from particle system, add gravity and life between 0 and a desired amount to get a constant emission effect (emulate fire etc)

Keep in position factor: Controls the time particle will stay in initial emitted position.

Keep alive factor: Percent of particle life where start lifetime of the particle is restored.

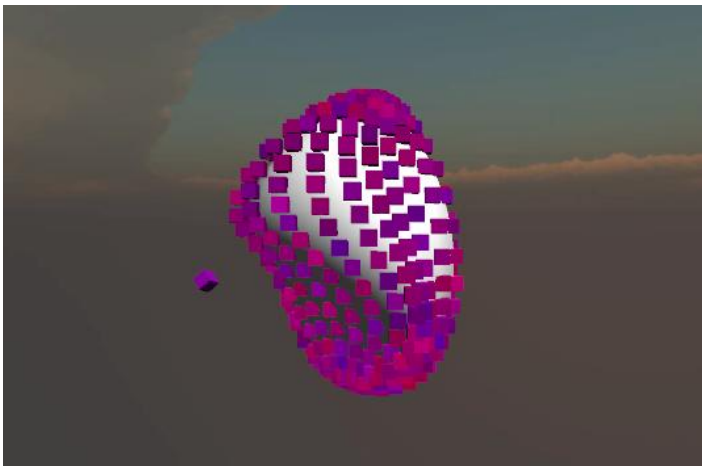
Extend life: Restore each particle life so it never goes out, useful to keep gameobjects static when let loose is used (otherwise they will keep spawning).

Gravity mode: Restore position on skinned body, use to restore shape after a melt effect (let loose plus gravity). Uses “Return speed” parameter to define speed of lerp to the skin position.

Return Speed: Speed to apply to “Gravity” mode.

Follow normals: Use for mesh particles, allows mesh particles to conform to the surface normals.

Transition: Use to reset the system when a transition is required.



Skinned-Animated-Procedural Gameobject emission – deprecated as of v2.0

Steps:

1. Create the gameobject that will be emitted from the mesh. Make sure the size is proper.

2. Create a instances holder, name it “Object_pool” or similar.

3. Create the particle system to emit the gameobjects and attach to the emitter mesh (meshfilter or skinned mesh) **with the procedure described for the previous script** ("**SkinColoredParticles**"). Attach the **"SKinnedGAmeobjEmit"** script instead.

4. Add the Skinnedmesh to "Emitter", the gameobject to "Gameobj" and the Pool gameobject to "ParentOBJ" entries in the script.

Options:

Scale factor: Offset particles from the body

Start size: Define particle size that will be used in play mode.

Emitter: Object with the skinned mesh component that should also be parent of the particle system with the "**SKinnedPArticleEmit_StaticNonPrefab**" script.

Keep in position factor: Controls the time particle will stay in initial emitted position.

Keep alive factor: Percent of particle life where start lifetime of the particle is restored.

Face emit: Choose to either emit through the vertices (particle number is the vertices divided by "Every other vertex" parameter number) or through the Shuriken initial particle positions on the mesh, in which case the number of particles is the number in the particle system. **If this feature is enabled, the prefabs that have the script must be in the scene and enabled in the editor in order to get the shape of the object in the particle system. Do not use this setting for objects that need to be instantiated.**

Preview mode: Enable preview of gameobjects, **disable before entering play mode**.

Every other vertex: Divide vertices by this number, to scatter gameobjects

Particle count: The desired number of gameobject to create.

P2: Particle system for debug purposes.

Y Offset: Letloose and Gravity mode height.

Fix initial: Check to keep the current gameobject position

Let loose: Free particles from skin and let inherit particle properties and forces

Extend life: Restore each particle life so it never goes out, useful to keep gameobjects static when let loose is used (otherwise they will keep spawning).

Gravity mode: Restore position on skinned body, use to restore shape after a melt effect (let loose plus gravity). Uses "Return speed" parameter to define speed of lerp to the skin position.

Parent_OBJ: The gameobject to be used as a parent pool holder for emitted gameobjects.

Angled: Unlock from initial gameobject rotation.

Asign rot: Assign rotation to gameobjects. Useful for grass effects etc.

Local rot: Assign a perlin rotation factor for each axis.

Wind speed: If bigger than zero, will add a perlin wind to gameobjects, use for grass.

Follow particles: Will attach gameobjects to particle position

Remove colliders: Removes all gameobjects colliders, use to enhance speed after an effect has taken place (like objects caught fire and there is no more need for the colliders)

TIP: Close the rendered to only get the gameobjects shown.

TIP: Apply any effect on gameobjects, like randomize coloring with the Procedural script.

Skinned-Animated-Procedural Masked Gameobject and Particle emission (v2.0) – Previous versions for masked particles and gameobject emission deprecated, this new version combines the masked emission and gameobject emission in a single system.

This is the same as the previous scripts, with the difference that allows emission from a masked area of the main texture, by defining a mask texture and gameobject emission as well as particle emission. This script is optimized so that only the unmasked areas will create and emit particles. Also allows for surface normal conformation of mesh particles.

Use the "SKinColoredMaskedCol" script to make use of the new in v2.0 masked gameobject emission.

Options:

Collision triangle indices (v2.1): For debug purposes. Reserved for v2.2 update, where the collisions system will enable skinned emission on the impact point, for precise procedural surface follow of sprayed sticky particles, using a changing collider. Coming in v2.1 of Particle Dynamic Magic.

Gameobject mode: Use gameobject emission along with particle emission.

Preview mode: Show gameobjects in editor

Parent OBJ: Pool to hold instantiated gameobject particles

Gameobj : Gameobject to instantiate as particle.

Angled: Allow gameobjects to get an angle different than the original.

Assign rot: Assign a rotation per axis.

Local rot: Rotation to be assigned per axis.

Wind Y offset: Offset the pre-defined motion in Y axis.

Wind speed: If above “1” will give a windy effect, speed defines the effect magnitude

Follow particles: Make gameobjects follow the particles, use with “let loose” option.

Remove colliders: Remove the colliders from all emitted gameobjects.

Look at normal (v2.0): Rotate to face the emitter surface normal.

Use lerp (v2.0): Use lerp in motion calculations like rotation towards motion or normal.

Quantize factor (v2.0): Use to discretize the gameobject sizes when the “scale by mask” option is used.

Scale factor: Offset particles from the body

Start size: Define particle size that will be used in play mode.

Mask: Texture used as mask for emission, use transparent mode in texture. PNG file with transparency is recommended.

Low mask threshold: The threshold for transparency, above 255 will cut off all particles, below zero will not mask any particles.

Size by mask: Use mask grayscale to define particle size.

Scale by mask (v2.0): Use mask alpha to define particle scale, between low-high defined below.

Scale low-high (v2.0): Low and high scale margins when “Scale by mask” option is used.

Emitter: Object with the skinned mesh component that should also be parent of the particle system with the skinned emission script.

Colored: Color by texture (default, use with default particle material or change it for extra blend effects), if unchecked inherits from particle material.

Custom color: Use a custom color, instead of the texture colors.

Lerp color: Smooth transition of colors between current and custom, use for Transitions.

End color: Custom color to use.

Face Emit: Emit from Shuriken positions. **If this feature is enabled, the prefabs that have the script must be in the scene and enabled in the editor in order to get the shape of the object in the particle system. Do not use this setting for objects that need to be instantiated. If more particles are needed it is better to use the “Every other vertex” option with values under one, in order to increase the particle count.**

Follow particle: Make particles appear on the particle position, rather than the emitter mesh position.

Velocity towards normal (v1.4): Apply an initial velocity towards the surface normal. Control strength with local speed for each axis with ‘Normal Velocity” parameter and use the keep alive and keep in position factors to determine how long the velocity will be applied.

Normal Velocity (v1.4): Control the velocity towards normal strength per axis.

Every other vertex: Divide vertices by this number, to scatter gameobjects. In order to enable more particles from each vertex, give it a below one value. The particle count will appear in the Shuriken max particles property (and is also adjusted based on mask texture).

Particle count: The desired number of particles.

P11: Particle system for debug purposes.

Y Offset: Letloose and Gravity mode height.

Fix initial: Check to keep the current gameobject position

Let loose: Allow particle to inherit from particle system, add gravity and life between 0 and a desired amount to get a constant emission effect (emulate fire etc)

Keep in position factor: Controls the time particle will stay in initial emitted position.

Keep alive factor: Percent of particle life where start lifetime of the particle is restored.

Extend life: Restore each particle life so it never goes out, useful to keep gameobjects static when let loose is used (otherwise they will keep spawning).

Gravity mode: Restore position on skinned body, use to restore shape after a melt effect (let loose plus gravity). Uses “Return speed” parameter to define speed of lerp to the skin position.

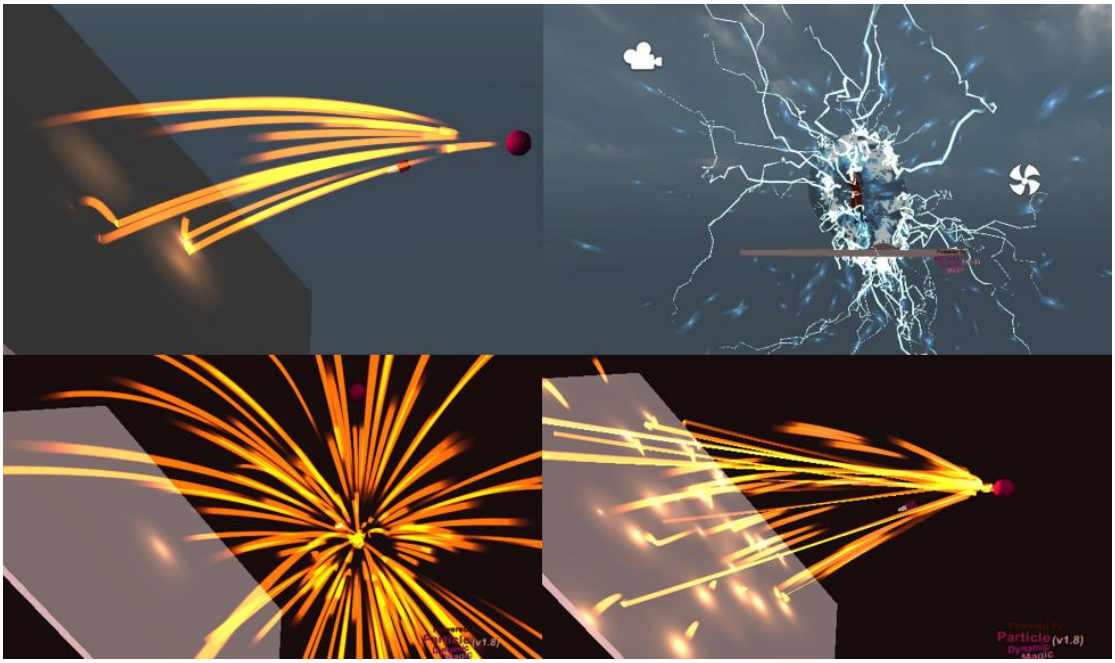
Return Speed: Speed to apply to “Gravity” mode.

Follow normals: Use for mesh particles, allows mesh particles to conform to the surface normals.

Transition: Use to reset the system when a transition is required.

Lerp with normal (v2.0): Lerp the motion vector when in wind mode with the surface normal vector.

Lerp amount (v2.0): Amount to lerp towards the surface normal vector vs the wind direction vector in wind mode.



Ribbons

The ribbon system allows the use of trail renderer to connect successive particle positions. It requires a line renderer to be attached to a gameobject particle and the script **“RibbonsPDM.cs”** to be added to that renderer and tweaked to properly stop the trail as the particle is re-emitted from the source and start it over for a new trail effect.

Then the emitted gameobject particles (using any of the Particle Dynamic Magic gameobject emission systems) will create trails over the particle trajectory. The particle renderer may be disabled to allow only ribbons to be rendered or enabled for combination with particles effects.

Options:

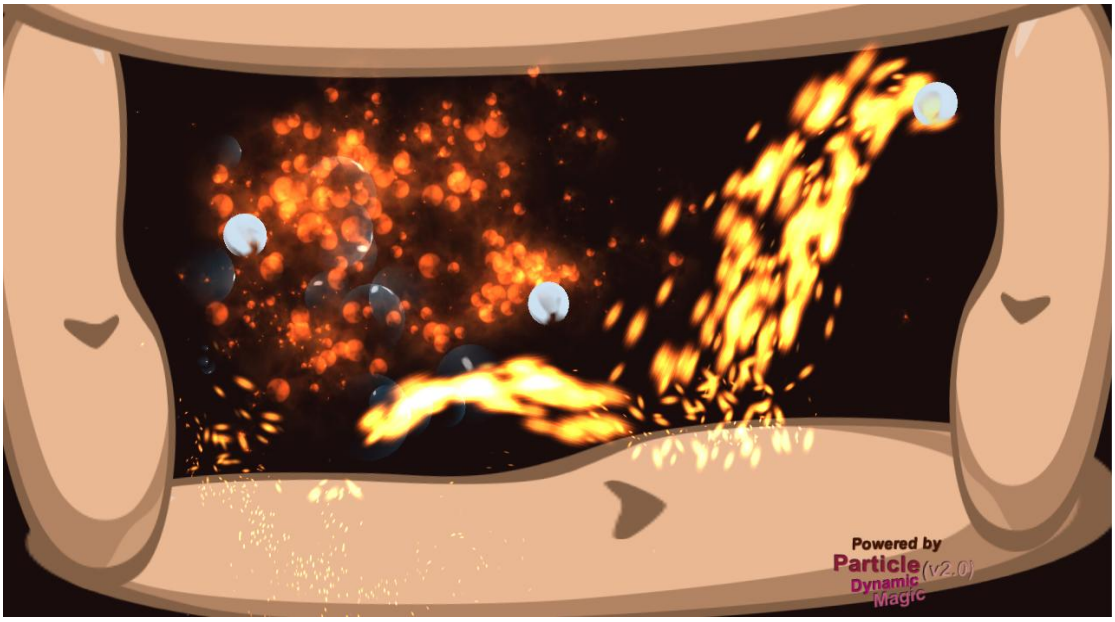
Emitter: Transform of the gameobject that emits the particles.

Ribbon: The trail renderer to be used.

Min-max distance: Distances from the source to renew the ribbon.

Time: Time to set to the ribbon creation, after that time the ribbon part that exceeded the margin will be retracted (trail tail fade).

Skip frames: Skip the defined number of frames when creating the ribbon trail.



Particle 2D collisions PDM

The 2D particle collisions system allows the emulation of collisions between particles and 2D colliders. The script “Particle2DCollisionsPDM.cs” must be attached to the particle that needs to collide with the 2D colliders and the particles can then also be locked to a specific axis (Z by default) in order to properly apply the 2D collisions and restrictions. Each particle is locked to the Z axis of the particle emitter.

Extend life: Keep particles alive while checked.

Keep alive factor: Percent of particle max lifetime to apply the revival of particle life.

Bounce range: Bounce strength factor randomized between two factors.

Bounce factor: Bounce strength

Min collision distance: Min distance to the 2D collider when a collision will happen.

Friction: Enable friction

Friction loss: Factor applied to Bounce strength due to friction, zero is maximum loss, one is no loss, above one is gain.

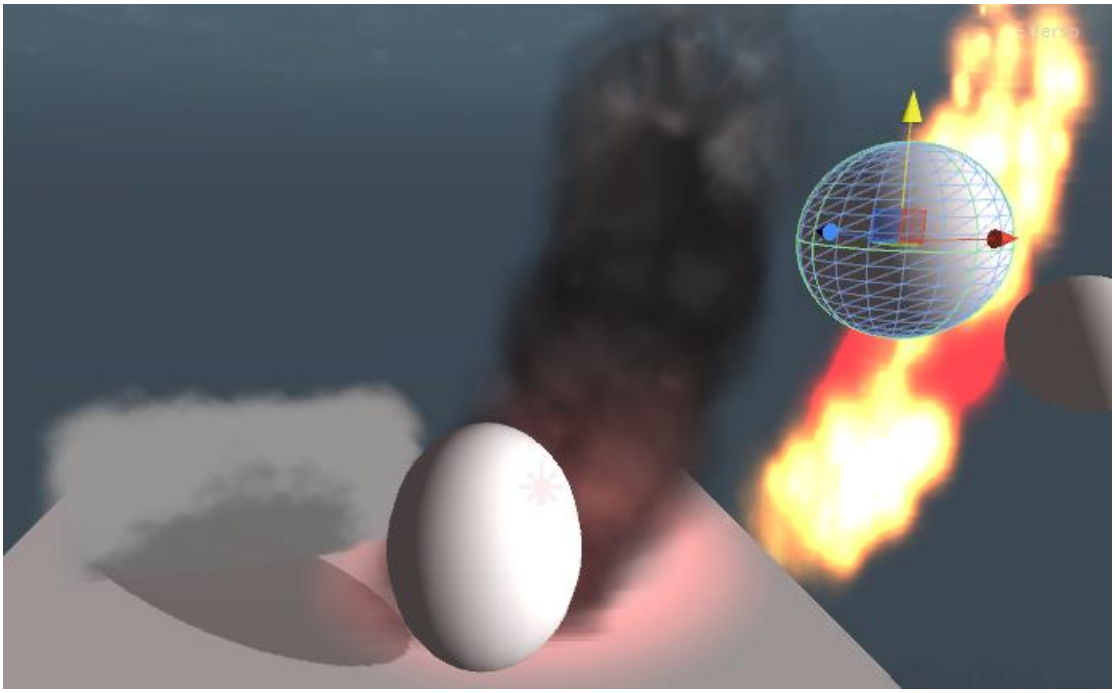
Size loss: Factor applied to particle size after collision, zero will make particles vanish after impact, one will keep their initial size. This system will work in addition to any size over lifetime curve values active from the Shuriken system.

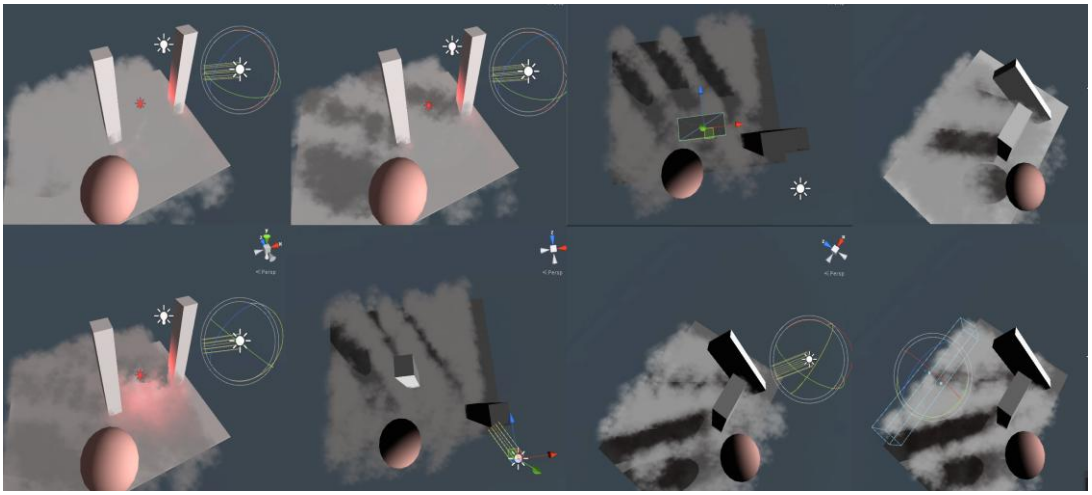
Min size: Minimum size of particles after size loss.

Size on collision: Resize particles based on collisions.

Size on velocity: Resize particles based on velocity.

GUI on: Enable test gui.





Particle Shadows

Particle Dynamic Magic v2.0 offers an option for receiving particle shadows. There is a system to shadow particles based on a reference directional light. The script (**"ParticleShadowsPDM.cs")** uses a reference light and is attached to a particle that needs to be shadowed with respect to that reference light.

Options

Reference light: The sun light transform.

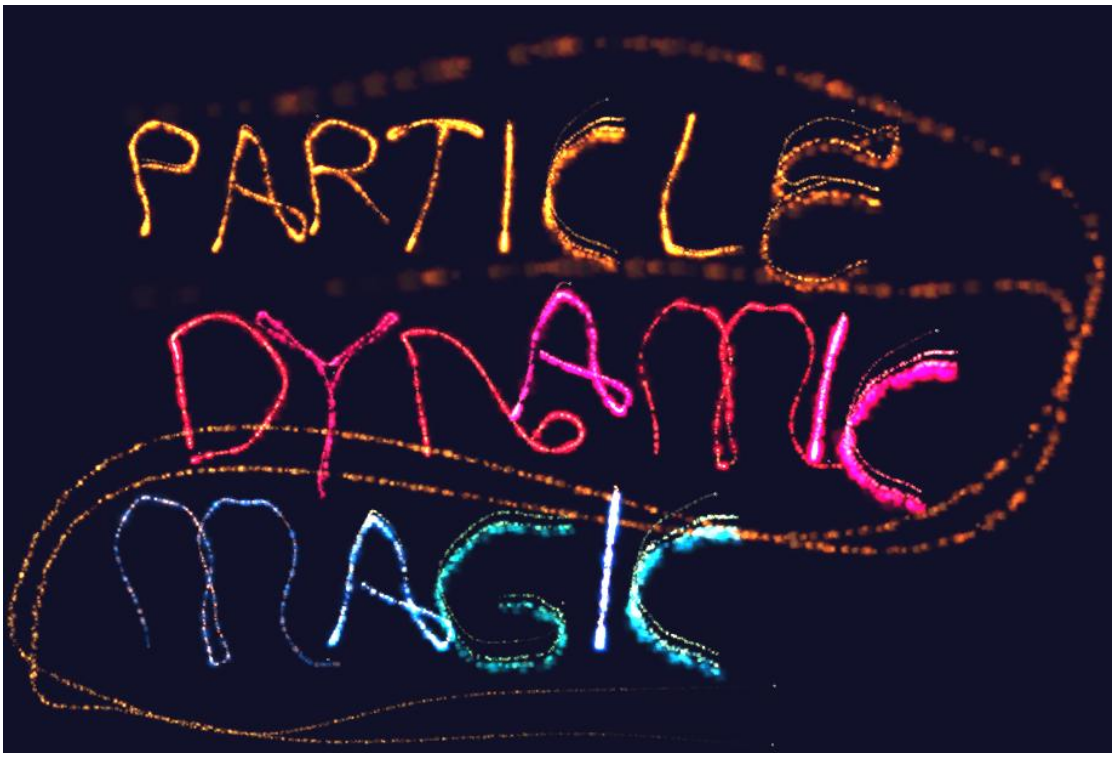
Debug on: Debug the raycasting used to shadow the particles.

Shadow factor: Shadow strength.

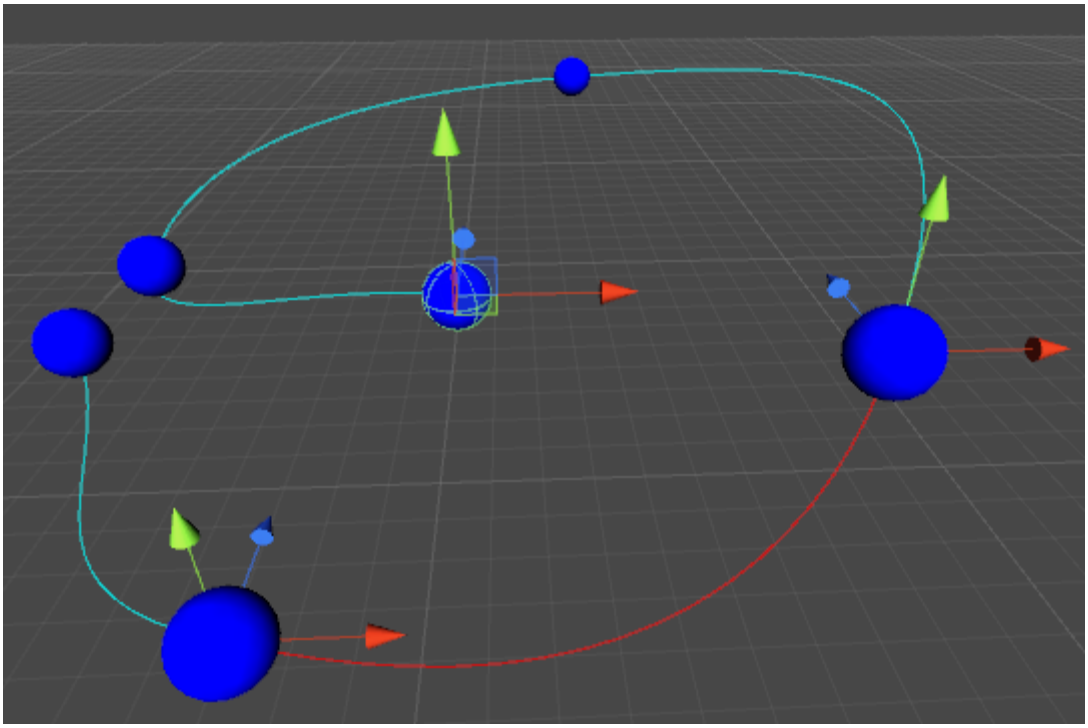
Shadow max distance: Distance after which shadows won't be calculated.

Shadow color: The color of the shadow that falls on particles.

Spline editor and particle follow



The editor allows for the creation of Splines, with basic functions like add and remove points and repositioning of its nodes.



In order to create a Spline object, create an empty gameObject and attach the "**SplinerP**" script. A basic Spline will be created and can be used as starting point to defining the desired one.

As shown in the figure, clicking on one of the spline segments will highlight it and give controls for the position of the control nodes. The starting node control will move the whole object around.

Use the right mouse button to deselect the spline part and return control to the whole object. Use the "Add another point" to add one extra point after the last and "Remove last point" to remove the end point. The last two points cant be removed (if deleted will be auto restored).

Each node also has a sphere gameobject attached, that can be moved and spline will follow. This enables all kind of physics and manipulation effects using the spheres to manipulate the spline and particles that emitted. Erase a sphere to delete the corresponding spline node.

Options:

Multithreaded (v2.1): Use multithreading for spline calculations. Best used for distant to player splines.

Define Rotations (v2.1): Define a rotation per spline node, for use as extra information by external systems (e.g. for mesh generation)

Curve quality: The segments detail between two control nodes. This number is same for any spline segment length, so it is advised to keep spline segments evenly separated to keep the same distances between curve points.

Follow curve: Make the gameObject defined in "Object" variable to follow the spline from its starting point to the end and loop from the start. If the spline has sharp corners, the alter mode should be used for smooth motion.

Parent moving: The gameObject to follow the spline. Place the item to follow the spline as child in the spline object and reset its position, rotation to zero. Make sure curve follow is checked in the script.

Motion speed: Speed for the motion on the spline.

Handle Scale: Scale the spline handles to match the scene scale.

Node toggle distance: The distance from the spline node that will trigger the move to the next. Use to refine the close to node motion depending on scene scale.

Show controls play mode: Show control spheres in play mode, so they can be manipulated.

Game mode on: Turn on to recalculate the spline in play mode. Moving the spheres around (or applying physics based motion) in play mode will change the spline shape in real time.

Alter mode: Make gameobject running on spline face its direction.

Always on: Use in editor, when need to connect two splines, this option will show the spline even when not selected. The option is disabled when entering play mode for all splines.

Show controls edit mode: Show control spheres in editor, useful if many splines are in the scenes.

Alter mode 2: Alternative speed mode, use accelerate with it.

Accelerate: Accelerate the follow spline motion.

Curve scale: This will always default to “1”, use to scale spline without Unity handle.

Add point: Check or enable through code to add a point to the spline during play mode.

With manip: Add a manipulation script for the point, so it is ready to move inside the game.

Add mid point: Adds a point in the mid of the segment chosen in “Add in segment” option.

Add in segment: Choose segment to add point in the middle

Override seg detail: Use to override the global spline quality for individual segments. Zero will make segment inherit the global quality value, above zero will override the global quality. Use value “1” to get a straight line.

Spline points (v2.1): Preview spline points and alter rotation & custom directions for the new in v2.1 split and custom angle modes

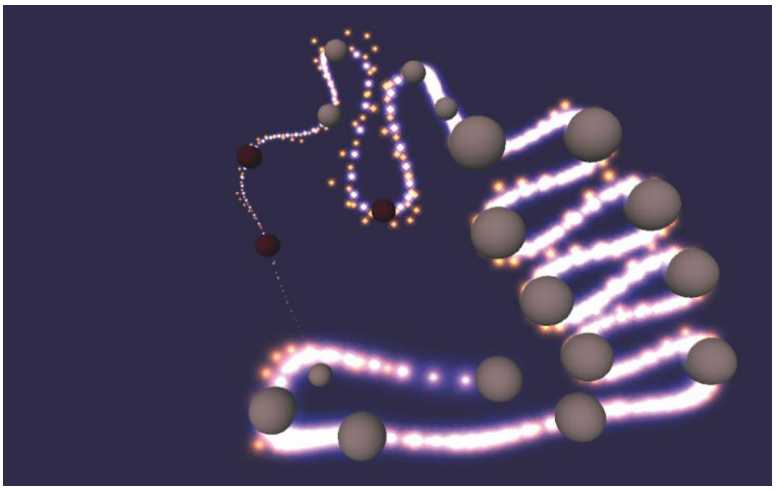
Change to auto (v2.1): Automatic curvature calculation in the selected spline segment end node.

Change to custom angle (v2.1): Define custom curvature in the selected spline segment end node, with a single angle.

Change to split (v2.1): Define custom curvature in the selected spline segment end node, on both sides with two separate angles.

Add point - remove point - add point between: Button to add-remove spline points in the editor.

TIP: When scaling or moving the spline with a segment selected, all other segments will move relative to this segment.



TIP: Splines control GUI is provided in samples showcase demo, for in game spline editing and manipulation. The above spline was created during play mode from the basic spline.

Spline particle emission

Use the "**PlaceParticleOnSpline**" script on a particle system.

Steps:

1. Create the particle system to be emitted from the spline. Use "world" in simulation space parameter and set particle speed to zero to get a preview on the spline.

2. Attach the "**PlaceParticleOnSpline**" script on the particle system. Parent the system to the spline start.

3. Add the gameobject with the spline script on the SplineP_OBJ parameter in the "**PlaceParticleOnSpline**" script.

4. Remove the option "Relaxed" in the script for preview (use in play mode to let particles leave the spline).

5. Disable the "Shape" checkbox on the particle.

6. Increase particle emission to see particles populate the spline. Use the interpolation options to get various effects and particle distributions on the spline.

Options

P2: Linked Particle system, for debug purposes

Gameobject mode (v2.0): Enable gameobject emission. Use the pool defined in "Parent_OBJ" and item defined in "Gameobj" parameters.

Interpolate: Add a particle between spline points, so less detailed splines can be filled with particles.

Angled interpolate: Use an angular effect on interpolation.

Variable step: Enable this mode so that particles are evenly distributed on the spline.

Interpolate steps: Steps between curve points to add particles to.

Reduce factor: Scale particle depending on spline position (type "1" to reset)

Look at spline (v2.0): Make gameobjects look at spline direction.

Rotation offset (v2.0): Offset the rotation of gameobjects using the below axis.

Rotation axis (v2.0): Main axis to rotate particle with, default is Y.

Rotation angle axis (v2.0): Change the axis based on spline direction.

Angle split (v2.0): Define the distance to look for previous point on spline to define the spline direction locally. One by default, use a low number to not exceed spline points.

SplineP_OBJ: The gameobject with the "**SplinerP**" script.

Gravity (v.1.3): Use to make particles move back to spline, with speed defined in "Return Speed"

Return Speed(v.1.3): Speed to use when "Gravity" option in on.

Relaxed: Allow particle to inherit forces and properties from Shuriken particle.

Keep in position factor(v.1.3): Controls the time particle will stay in initial emitted position.

Keep alive factor(v.1.3): Percent of particle life where start lifetime of the particle is restored.

Hold Emission (v.1.3): Keep particle emission on.

Extend life (v.1.3): Restore each particle life before going out.

Transition (v.1.3): Reset system, used for Transition effects.

No overflow (v.1.3): Use to limit particle numbers to the spline in Transition effects.

Gameobject Offset (v2.0): Offset for gameobjects emission.

Preview mode (v2.0): Show gameobjects in editor

Parent OBJ (v2.0): Pool to hold instantiated gameobject particles

Gameobj (v2.0): Gameobject to instantiate as particle.

Angled(v2.0): Allow gameobjects to get an angle different than the original.

Particle count (v2.0): Maximum gameobject particles.

Assign rot(v2.0): Assign a rotation per axis.

Local rot(v2.0): Rotation to be assigned per axis.

Wind speed(v2.0): If above "1" will give a windy effect, speed defines the effect magnitude

Follow particles(v2.0): Make gameobjects follow the particles, use with “let loose” option.

Remove colliders(v2.0): Remove the colliders from all emitted gameobjects.

Let loose (v2.0): Let particle inherit forces from Shuriken particle system.

Place every Nth step (v2.0): Control gameobject emission spread density along the spline.

Spline Gameobject emission (deprecated as of v2.0 – merged with above script)

Steps:

1. Create the gameobject that will be emitted from the mesh. Make sure the size is proper.

2. Create an instances holder, name it “Object_pool” or similar.

3. Create the particle system to emit the gameobjects and parent it to the spline to keep them together. **Use “World” in Simulation space and follow the spline emission procedure of the previous paragraph.**

4. Instead of the “**PlaceParticleOnSpline**” script, place the “**PlaceGameobjectOnSpline**” script to the particle system instead, the gameobject to “Gameobj” and the Pool gameobject to “ParentOBJ” entries in the script.

Options:

SplinerP_OBJ: Place Spline gameobject in this field.

Preview mode: Enable preview mode for the gameobjects. **Preview mode must be turned off before play mode is entered (or gameobjects will have to be manually erased from the pool parent after play mode).**

Particle Count: Particle count can be adjust to be lower than the curve points, but will not go above the spline curve points. Adjust the particles per segment by adjusting the spline quality.

P2: Linked particle system, for debug purposes.

Gameobj: Place the gameobject (scene object) to be emitted in this field.

Y_offset: Give an offset from the spline.

Let loose: Let particle inherit forces from Shuriken particle system.

Gravity mode: Make particles return to the spline with speed defined in “Return Speed”.

Parent_OBJ: The gameobject used as pool for the emitted gameobjects.

Angled: Allow gameobjects to get an angle different than the original.

Assign rot: Assign a rotation per axis.

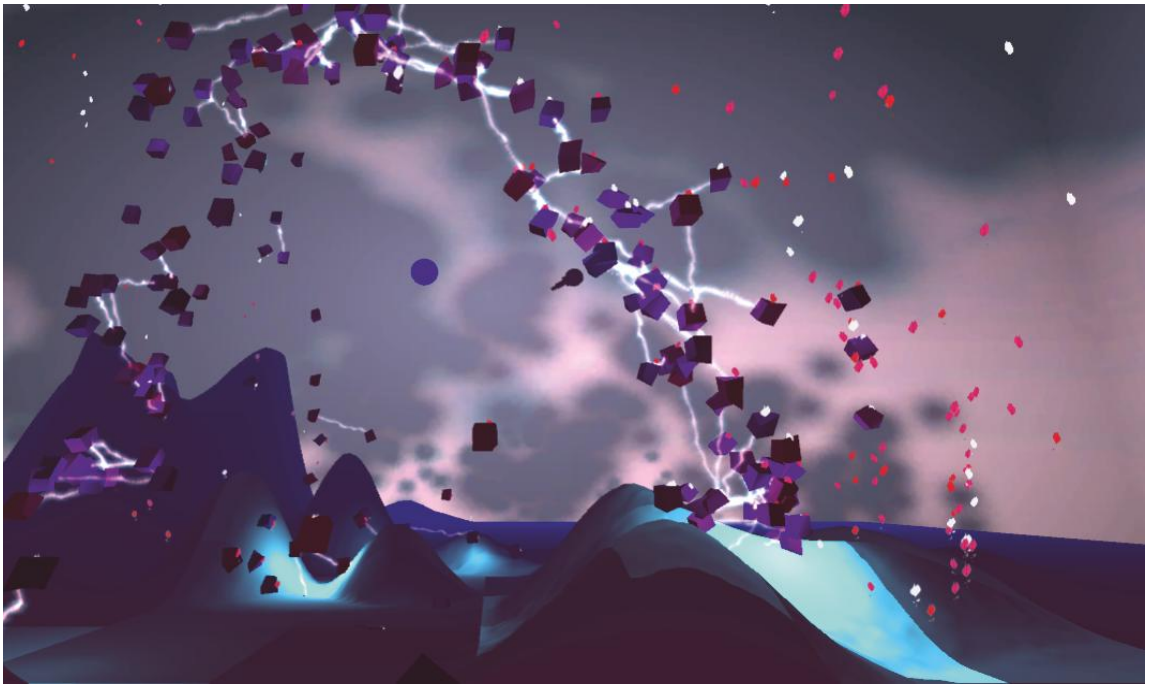
Local rot: Rotation to be assigned per axis.

Wind speed: If above “1” will give a windy effect, speed defines the effect magnitude

Return speed: The speed used for the “Gravity” mode

Follow particles: Make gameobjects follow the particles, use with “let loose” option.

Remove colliders: Remove the colliders from all emitted gameobjects.



Particle projection

Drag the “**Particle projector**” prefab in the scene. It includes a ready setup for the effect. This effect gives a dynamic projection of the attached particle system and is the most basic of the three projection modes.

Options (Particle Projection script):

P2: Linked particle system, for debug purposes

Randomize: Add a jitter effect

Extend: Increase particle spread

Particle projection, texture sheet varied particles

Drag the “**PARTICLE SHEET PROJECTOR**” prefab in the scene. This will give the basic setup for the script (projector arrangement, with Gizmo and manipulator).

Options (Particle Sheet Projection script):

Particle Count: Adjust particle number.

Go Random: Randomize positions.

P2: Linked particle system, for debug purposes.

Extend: Spread particles.

Y Offset: Offset from projection.

Fix initial: Keep first projection positions.

Let loose: Allow particles to inherit forces from particle system.

Gravity Mode: Make particles return to initial projected position

Tiles X-Y: Number of texture sheet rows and columns.

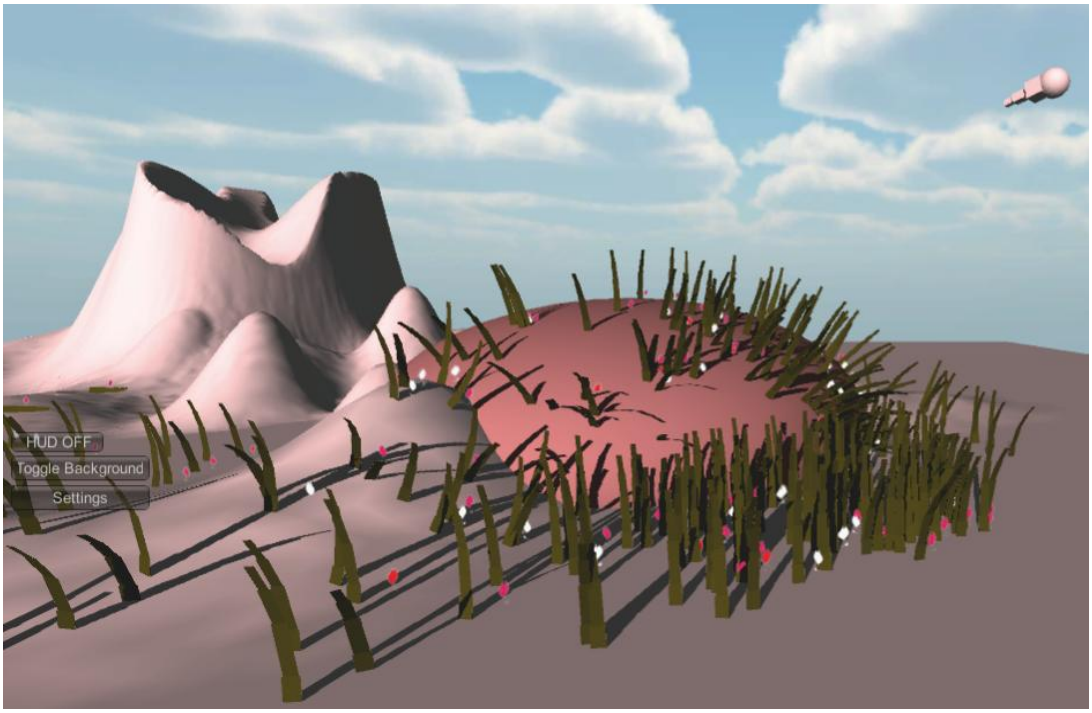
Origin at Projector (v1.3): Particle can move based on the parent projector or on its own.

Projector OBJ (v1.3): Choose the projector gameobject to follow.

Follow normals (v1.3): Align mesh particles to the surface normal.

Transition (v1.3): Use to reset the system, used for Transition effects.

Return Speed(v1.3): Speed to return to original positions when "Gravity" option is on.



Gameobject Projection

Drag the "**GAMEOBJECT PROJECTOR**" prefab in the scene. This will give the basic setup for the script (pool, emitted object and projector arrangement).

Add the "**GameobjectProjection**" script to the "Projection particle" system. Rotate the "MANIPULATOR" gameobject to make the Gizmo point to the required projection position.

Options:

Preview mode: Enable preview mode for the gameobjects. **Preview mode must be turned off before play mode is entered (or gameobjects will have to be manually erased from the pool parent after play mode).**

Particle count: Change the particle count. Use in editor mode.

Go random: Randomize the particle positions.

P2: Linked particle system, for debug purposes.

Extend: Increase distance between particle positions, use to spread particles.

Gameobj: Place the gameobject (scene object) to be emitted in this field.

Y_offset: Give an offset from the projection point.

Fix Initial: Keep the first projected position, use to keep the positions if another collider comes between the projection and the projected objects.

Let loose: Let particle inherit forces from Shuriken particle system.

Gravity mode: Make particles return to the projection positions with speed defined in "Return Speed".

Parent_OBJ: The gameobject used as pool for the emitted gameobjects.

Angled: Allow gameobjects to get an angle different than the original.

Assign rot: Assign a rotation per axis.

Local rot: Rotation to be assigned per axis.

Wind speed: If above "1" will give a windy effect, speed defines the effect magnitude

Return speed: The speed used for the "Gravity" mode

Follow particles: Make gameobjects follow the particles, use with "let loose" option.

Remove colliders: Remove the colliders from all emitted gameobjects.

Look at Direction: Make gameobjects look at their motion direction.

TIP: The prefab contains a transform manipulator to allow in game manipulation of the projection.

TIP: Enable the "**CombineChildrenPDM**" script in the "GAMEOBJECT_POOL" in play mode, to allow gameobjects become a single mesh if no motion is needed.

TIP: Remove colliders will work on the first level of the emitted gameobject, will not remove colliders from the children.

TIP: Use "Fix initial" when colliders are on, so particles will keep the initial projection, otherwise they will be cast on the gameobject colliders.

TIP: When preview gameobjects in editor, select “Remove colliders” to increase preview speed.

Effect combinations

Combination of spline emission and ground conformation.

In order to use the splines with particles, simply attach the desired particle to the gameObject that will follow the spline. Various effects can be made with this. Also can be used together with terrain conform script to make the particles follow the ground, for effects like flowing lave or rivers.

Another combination is possible with place particles on terrain with texture sheet and the place particles on terrain script, the former is used to control the particles texture and ground conformation and the second to control the circular effect.

Combination of gameobject projection, gravity (planar mode) and fire propagation.

Paint particle emission and propagation

Create a particle system to be used a pool for the painted and propagated particles. Attach the "**PlaceParticleFREEFORM**" to the particle system.

Create an item that needs to be painted with particles and add a collider. Tag the item with "PPaint" and select the particle, then paint on the object with the right mouse button. Tag an item as "Flamer" to make it propagate the particle to near objects tagged as "Flammable"

Particles will propagate to objects tagged as "Flammable" from all near objects tagged as "Flammable" if they are on fire (or any other particle), from "Flamer" tagged objects and from "PPaint" tagged objects that have painted positions on them. The painted positions will follow the object transform and will emit from the particle pool.

Options:

Brush Size: Increase the area right click will affect during position erase.

Erase mode: Enable to erase spheres with the right mouse button.

Marker size: The size of the painted position marker.

P11: Linked particle system for debug purposes.

Delay: Interval to check for particle propagation.

Optimize: Delay check for near objects to propagate particles, for optimized performance

Relaxed: Allow particles to inherit forces from the particle system

Draw in sequence: Check to make particles draw in the sequence they were drawn with, if not checked particles will evenly spread along all painted/propagated positions.

TIP: Increase the particle count to allow more positions to be painted.
TIP: To draw or delete positions, must have the object collider enabled.

Paint gameobjects

Create a particle system to be used a pool for the painted and propagated particles. Attach the "**PlaceGameobjectFREEFORM**" to the particle system.

Create an item that needs to be painted with particles and add a collider. Tag the item with "PPaint" and select the particle, then paint on the object with the right mouse button.

Options:

Brush Size: Increase the area right click will affect during position erase.

Erase mode: Enable to erase spheres with the right mouse button.

Marker size: The size of the painted position marker.

P11: Linked particle system for debug purposes.

Delay: Interval to check for particle propagation.

Optimize: Delay check for near objects to propagate particles, for optimized performance

Relaxed: Allow particles to inherit forces from the particle system

Draw in sequence: Check to make particles draw in the sequence they were drawn with, if not checked particles will evenly spread along all painted/propagated positions.

TIP: Increase the particle count to allow more positions to be painted.
TIP: To draw or delete positions, must have the object collider enabled.

Transitions and Music Manager

Options:

Use audio: Enable the music module.

Use GUI: Enable the sample GUI to control particles and transitions.

Loop off (v1.7): Stop looping after last target is reached.

Loop Ended (v1.7): Signals loop stop.

Destroy After loop (v1.7): Destroy target after loop is over.

Disable after loop (v1.7): Disable target after loop is over.

Disable Destroy After (v1.7): Time until the Destroy/Disable take effect.

Start Count Down (v1.7): Count down to destroy, for debug purposes.

Trigger ice grow: Make particles grow for an amount of time, based on" grow speed" and "max particle size" parameters.

Auto cycle targets: Auto cycle targets.

Auto cycle time: Time between auto target changing transitions.

Last Auto Cycle (v1.7): Current cycle time, for debug purposes.

Cycle in order: Cycle targets in the order they appear in the Targets list.

ICE SYSTEM: The main particle pool and scripts to use for the Transitions.

Grow speed: Speed to grow particles when “Trigger ice grow” is selected.

Chosen target: Current target, for debug purposes.

Targets: List of targets to use for the transitions.

Use Timer (v1.4): Apply separate duration for each transition.

Time to next (v1.4): Fill in the time to wait for each target, the list must have the same size as the number of targets.

Use force states (v1.6): Define a different force to be used at each transition target.

Attractors per step (v1.6): Define the Attractor scripts to be enabled per transition target. Size must be equal to the target cycle list.

Per step type (v1.6): Define a different particle system to be used at each transition target.

Type per step (v1.6): Define the particle system to be used at each transition target.

Randomize all (v1.6): Randomly select particle types and forces at each target.

Last Particle type ID (v1.6): For debug purposes. Shows the number of the last used particle type in the Particle types list.

Per step mask (v1.7): Add a mask texture per target.

Mask per step (v1.7): Mask texture per target.

Size growth speed (v1.7): Speed to change particle growth with for grow effects.

Gravity return speed (v1.7): Speed to control the fall back to mesh of particles.

Size growth speed per step (v1.7): Allow size growth per step.

Gravity return speed per step (v1.7): Allow settle to mesh speed return per step.

Per step size growth speed (v1.7)**:** Define size growth speed per step.

Per step gravity return speed (v1.7)**:** Define size growth speed per step.

Exponential Gravity return speed (v1.7)**:** Make return speed slower as target is approached.

Distance based speed (v1.7)**:** Calculate speed of transition based on distance from target.

Distance from first (v1.7)**:** Use the first target for above distance calcs.

Distance Squared (v1.7): Check against the square of the above distances.

Distance speed grow factor (v1.7): Extra factor to control speed based on distance.

Let loose on mesh per step (v1.7): Allow particles to leave the targeted mesh per step.

Per step let loose on mesh (v1.7): Define which meshes the effect will apply to.

Keep previous per step (v1.7): Allow previous particle to be kept alive per step.

Per step keep previous (v1.7): Define which targets will allow previous particles to co-exist with the new.

Change time per step (v1.7): Allow time of transition between the two particle systems be defined per step.

Per step change time (v1.7): Define the time of transition between the two particle systems per step.

Remove gravity per step (v1.7): Allow gravity return to mesh be controlled at each step.

Per step remove gravity (v1.7): Define the removal of gravity per target.

Reach size time per step (v1.7): Allow the control of time to reach full size per step.

Per step reach size time (v1.7): Define the time when the above option is active, for each step.

Inject 3D particles (v1.6): Inject the current particle system to the Propagator script that controls gameobject particles, allowing them to follow the particles from the current transitioning system.

Max particle size: Set the max size the particle can grow to when “Trigger ice grow” is selected.

Attractor: Choose a forces node to use (Attractor script)

Attractor Planar: Choose a second forces node to use.

Coloration: Enable particle coloration, when using a mesh target and “Colored” is checked on the masked mesh emission script.

Start Color: Color to use for coloring particles.

Detail: Define audio detail to be used.

Amplitude: Magnify music effect on particles.

Audio source: Gameobject with Unity AudioSource.

Toggle Interpolation time: Use for music effects, change interpolation steps every N secs.

Dancing splines: Allow spline nodes to move according to the music.

Switch particle (v1.4): Enable this option in order to switch to another particle system on the fly. The variable is reset afterwards.

Target particle (v1.4): The particle system to switch to.

Keep previous (v1.4): Keep the previous particle alive.

Change time (v1.4): The duration particles will co-exist (if Keep previous is not selected). If set to zero, will apply an instant change to the other system.

Smooth change (v1.4): Smooth change, affecting size of particles to give a fluid change effect.

Reach size speed (v1.4): The speed particles reach their size in smooth transition.

Switch on transition: Switch particle system on every transition.

Let loose on mesh: Let particle loose when a mesh target is reached. Use to allow forces affect the particle after it reaches its target.

Remove Gravity: Removes the return force to the initially painted or mesh defined position.

Return to mesh speed: Return force speed for mesh targets. Use to control particle speed.

NOTE: The coloration and resize effects are available when a mesh is selected as target (as they are controlled by the masked mesh script). To recolor the particle when it is on a spline use the Shuriken Color over lifetime parameter.

Provide Transition Target Script (v1.7)

Use the "Provide_Transition_Target_PDM" script to provide targets for the Transition Manager. It must be attached to a Transition Manager script. The script will search for SkinnedMesh and Mesh in the provided gameobjects and assign them to the Transition Manager at start. It will also inject the freeze script to the currently affected target ("FreezeBurnControl_DUAL_SHADER_PDM.cs").

Options:

Dual Shader: Shader to assign to the target. Dual Shader asset is used in the script, code can be altered for any shader.

Dual Shader Ice: Texture to use when item freezes.

Disable Mesh: Disable the mesh after defrost. Use to emulate item breaking into ice.

Thaw speed: Speed to thaw the ice in freezing effect.

Freeze speed: Speed to freeze target.

BEm amount increase speed: Increase Dual Shader freeze over speed.

BEm amount decrease speed: Decrease Dual Shader freeze over speed.

AFlood increase speed: Increase Dual Shader freeze over speed.

AFlood decrease speed: Decrease Dual Shader freeze over speed.

Max freeze amount: Max freeze amount.

Animation Speed drop: Make hero move slower when frozen by this speed.

Animation Speed restore: Make hero move faster when defrost by this speed.

Restore animation to speed: Speed to restore animations to.

Low freeze animation speed: Lowest speed to reach when frozen. Use zero for complete freeze and immobility.

Delay freeze animation factor: Delay the time when animation slow down takes place.

Check time: Interval to check for update.

Transition Target: Use for Single Target.

Find by Code: Use to give target list by code. Needs code changes to assign the targets.

Multi target: Hit more than one targets.

Transition Targets: Provide targets to target.

Injection Targets: Dictate which targets will be injected with the freeze script to control their shader and animation.

Use Dual Shader: Check if Dual Shader Asset is used.

Inject Freeze: Check to enable injection of freeze script overall. Control with Injection Targets individually.

Injection delay: Delay the injection by an amount.

Dist to previous: Use the distance from previous target to calculate the speed of the transition, otherwise the distance from the source is used.

Freeze Burn Control DUAL SHADER script (v1.7)

Options:

Character root: The root gameobject of the Skinned Mesh.

Dual Shader: Shader to assign to the target. Dual Shader asset is used in the script, code can be altered for any shader.

Dual Shader Ice: Texture to use when item freezes.

Freeze Amount: Current freeze amount.

Burn amount: For future work.

Max burn-freeze amount: Max freeze amount.

Thaw speed: Speed to thaw the ice in freezing effect.

Freeze speed: Speed to freeze target.

Check time: Interval to check for update.

Current Time: Debug.

Use Dual Shader: Check if Dual Shader Asset is used.

Start Delay: Delay effect by an amount.

BEm amount increase speed: Increase Dual Shader freeze over speed.

BEm amount decrease speed: Decrease Dual Shader freeze over speed.

AFlood increase speed: Increase Dual Shader freeze over speed.

AFlood decrease speed: Decrease Dual Shader freeze over speed.

Disable Mesh: Disable the mesh after defrost. Use to emulate item breaking into ice.

Transition Manager: The assigned Transition Manager.

This Chosen Target: Current Target.

Animation Speed drop: Make hero move slower when frozen by this speed.

Animation Speed restore: Make hero move faster when defrost by this speed.

Restore animation to speed: Speed to restore animations to.

Low freeze animation speed: Lowest speed to reach when frozen. Use zero for complete freeze and immobility.

Delay freeze animation factor: Delay the time when animation slow down takes place.

Procedural Mesh script

Options (Procedural Noise PDM script):
Scale: Scale of the effect.

Speed: Speed of the effect.

Recalculate Normals: Check to recalculate normals.

Colorize: Assign random colors to the object.

Color cycle speed: The speed of the color cycling.

Change Factor: Add extra randomization to the object shape change.

Deactivate after: Check to deactivate the effect after a time specified in “Seconds” field.

Seconds: Time after which the object will be deactivated.

Lightning Bolt dynamic script

Options (Lightning Bold Free PDM script):

Target: Dynamic current target acquired, for debug purposes.

Zigs: Number of lightning zigs.

Speed: Speed of lightning procedural creation.

Scale: Scale of effect.

Start/End light: Light assigned to be used when lighting starts/ends.

Random Target: Randomize the target selection, otherwise choose the first one found.

Affect dist: Distance at which to check for targets (Object tagged as "Conductor")

Change target delay: time after a new target will be searched.

Particle Energy: Define energy of Legacy particle.

Optimize factor: Optimize light use, decrease chance to see a light when the lightning strikes.

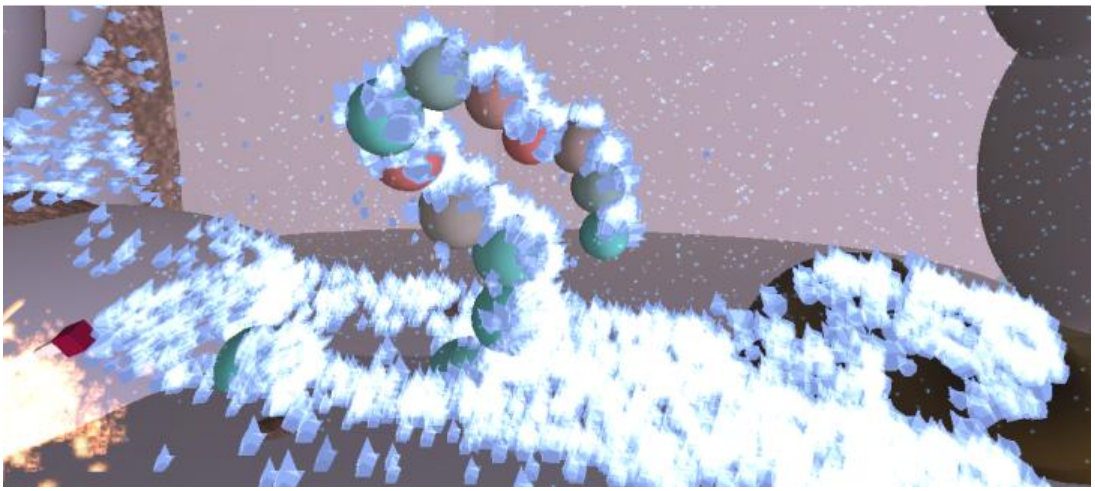
Version 1.5 – Advanced collision, painting and propagation effects, fire/ice dynamics.

The core of the new dynamic effect and collisions system in Version 1.4 are two scripts. The main script ("**ParticlePropagationPDM**") that is attached to the particle pool (for example ice or fire or gameobjects like animated butterflies) and the helper script that is attached to any particle system we need to paint collision positions with ("**ParticleCollisionsPDM**").

These two scripts working together allow for all the new dynamic effects and local overrides. The idea behind the scheme is using particle systems to paint collision positions (any system will do as long as collisions are enabled), that are handled by a common pool of particles which allows the pool particles to appear for a specific time, melt away (ice behavior), emit from the positions (fire), propagate (near object may catch fire) and leave the painted positions based on rules that allow gravitating back or complete freedom to follow near forces and any combination between. The pool allows painting particles (or gameobjects) before the game starts, in case we need pre existing fire/ice etc in the scene.

The collision script can be attached to a pool system to get its properties from or can optionally act on its own by allowing local overrides independent of a particle and also provides advanced particle to particle collision effects, which may use a pool to make particles stick to the environment after collisions and many more effects.

Fire & Ice: Dynamic effects propagation, spraying, collisions, local overrides, emission control and painting.



Create a particle system to be used a pool for the painted upon collision and propagated particles. Attach the **"ParticlePropagationPDM"** to the particle system.

The collision based positions will be populated by the particles that have the helper collision script and apply to all objects with collider setup.

Create an item that needs to be painted with particles and add a collider. Tag the item with "PPaint" and select the particle, then paint on the object with the right mouse button.

Options:

Use pooling (v2.0): Use pooling for decals.

Systems to override: Inherited by the collision script and used locally by the pool particle to allow local melt/go out effects by close by fire/ice systems. Apply **Local Overrides** to the systems that are hit by particle collisions of the linked to the pool particle emitters with the

collision script. Also enables local overrides at the ice/fire pool system, if the other systems are fire/ice respectively.

Cut off distance: Distance at which painted positions of the system to override will be affected by the positions of the pool particle. Use to make fire melt ice locally, by near painted/propagated fire (without using collisions) or fire go out by spread ice. The pool system must have “Enable melt” and “is_fire” or “is_ice” checked and “Enable overrides” option checked. The system to over

Reset overrides: Reset all overrides of the pool particle. Use this option to make butterflies return to their position with gravity, after they have been locally affected and take to the air. Set gravity to an amount that allows free flight first, then reset overrides and gradually increase return speed to get a smooth transition back to the original painted/sprayed positions.

Gameobject mode: Allows the use of gameobjects, together with the particles of the pool system. Particle renderer may be switched off if only gameobjects are required. This options requires two more parameters to be defined, an empty gameobject to be used as the pool for the instantiated gameobjects and the item to be instantiated. Any item can be used and scripts can control its complexity and variance at the time of instantiation. This option allows any object, animated, complex and procedural to be used as particle.

Free Objects (v2.0): Let gameobjects move freely.

Lerp Motion (v2.0): When not free, lerp gameobjects to their destined position.

Follow particle velocity (v2.0): Rotate gameobjects towards the corresponding particle velocity.

Scale by texture: Use in play mode, painted positions will scale according to image grey scale (**experimental feature**).

Color by texture: Use in play mode, painted gameobjects will be painted by the texture colors. This is expensive to use since each color requires a new material and batching is not possible in this mode.

Grow trees: Make gameobjects grow from smaller scale (**experimental feature**).

Grow time: Time scale for the transition.

Preview mode: Enable painted gameobject preview in editor.

Particle Count: Max number of gameobject particles allowed.

Gameobj: The item to be instantiated, when gameobject emission is active.

Gravity mode: Enable local gravity towards the initially painted – projected – propagated - sprayed position for each particle (and gameobject).

Gravity factor: The amount of return speed to be used in Gravity mode.

Y offset: Give an initial offset in Y axis.

X-Z offset factor (v1.7): Multiply offset for X-Z axis. Use to give small difference to each instance.

Let loose: Allow particles to move away from the painted positions. Gravity mode will balance them in between the free and original positions.

Parent Object: The empty pool gameobject to be used for gameobject mode.

Angled: For gameobject mode, use a custom angle or wind.

Assign rotation: Use custom rotation angle.

Local rot: Rotation per axis

Wind speed: Assign wind and adjust speed

Follow particles: Make gameobject follow particles.

Remove colliders: Remove and add colliders from instantiated gameobjects.

Look at direction: Make gameobject face the direction of their motion, use for making creatures look at their direction of movement.

Normal on Y (v2.0): Use when decal is facing the Y axis instead of Z axis. If unchecked the system will align the decal Z axis to the surface normal.

Enable Combine (v1.6): Use with the new dynamic batching system, allows the dynamic use of batching when objects move from their positions. It offers extra performance to the static batching system introduced in v1.6.

Release Gravity: The gravity to be used in local overrides mode. If a particle or gameobject is overridden due to a close by collision by a disturber system, will change its gravity to this value. Use to make hit butterflies take to the air locally.

Use stencil: Use an image as painting brush. Otherwise paint with mouse one point per time (drag is also possible).

Stencil: Image to be used for painting brush.

Tex scale: Scale of the projected brush image (brush size)

Coloration amount: Amount of coloration when “Color by texture” is selected.

Real time painting: Allow painting with the mouse on colliders during play mode. Note that multiple pools will apply if this is active in all of them at once, so enable only one pool with this option at any time to get paint from the specific system.

Color effects: Color particles by brush image colors.

Lerp color: Lerp between color of image and of particle system.

Keep color: Keep initial brush color for the whole emission duration.

Stencil Divisions (v2.0): Divide the current stencil by this amount, for down sampling the brush.

Propagation: Allow near objects tagged as “Flammable” to receive painted positions from the pool if near other painted positions. Use to make near object catch fire or freeze.

Brush size: Size of point brush erase area. Use to erase bigger area when not in brush mode.

Erase mode: Erase painted positions, works in editor and play mode. Use with brush or point painting.

Marker size: The size of marker shown in editor for painted positions.

P11: Pool particle, auto assigned when assign the script to the pool. For debug purposes.

Stay time: Time to let painted items stay on scene. If “keep alive” is not active.

Projectors: Gizmos of projectors to use for projecting painted positions.

Min-Max propagation distance: The min and max distances to a painted position at which a near object will get a propagated position. Use to let fire be propagated at a certain distance. Also the min distance allows near points to not keep propagating the same fire forever.

Go random: For projection mode, use randomized array for the projected rays.

Extend: For projection mode, make the projected area wider.

Follow normals: Make mesh particles follow surface normal.

Density dist factor: Use for propagation, min distance to near painted positions to start the propagation.

Propagation chance factor: Decide propagation rates, higher factor means less chance for propagation.

Use projection: Use once and disable, get painted position though the projection gizmos defined in the Gizmos parameter.

Use particle collisions: Inherited by the collisions script. Allows painted positions to be registered by collisions on any particle system that is linked to the pool and has the collisions helper script assigned.

By layer (v2.0): Apply pool items only on specified layers.

Layers (v2.0): Layers to apply effects to.

Keep alive: Keep particles alive

Grow ice mesh: Use together with Ice Grow script to enable mesh emission on collision.

Variant size: Vary particle size. Must be checked for all size dependent effects.

Vary gameobject size (v2.0): Use for variance on decals.

Random size upper bound: Upper bound for variant size.

Random size lower bound: Lower bound for variant size.

Debug rotation: Assign a starting rotation.

Is Ice: Use to allow fire systems to melt the ice pool particles. “Enable melt” must be on. Use Variant size to get a smoother looking melting, with size getting smaller gradually. If “keep alive” is on will not work, unless it is an override with flag = 1 (ice melt flag). Overridden particles will use the “fast melt speed” parameter. Normal melt uses the “melt speed” parameter. Also use to signal other particles that the pool is ice (or fire), in order to make fire go out (or ice melt) if “Enable melt” is enabled in the pools of the “systems to override” parameter.

Fire: Same use as ice above, for the opposite element.

Butterfly: Use to signal the pool has particles that will take to the air locally, this is checked in the collisions script, if the collision is close to a painted position with a butterfly and the disturb particle is linked to the disturber pool (or is locally assigned as disturb particle without a pool), if the butterfly pool is in the “systems to override” parameter, butterflies will take to air locally near the collision. Override flag for this behavior is 2.

Enable overrides: Enable local overrides, these are assigned per particle and will give different properties than the rest of the pool particles. Like a disturbed butterfly can take to the air, while the rest stay to the ground with a bigger gravity factor, or ice melt faster when fire collisions happen near than the rest of the pool particles.

Enable melt: Use to allow particles melt away as time passes, with “melt speed”.

Enable freeze-burn: Use to allow gameobjects freeze and burn by ice and fire, respectively.

Max freeze-burn amount: The amount of registered burn or freeze before a gameobject starts changing color.

Thaw speed: The speed of ice effect go away after freeze.

Freeze speed: The opposite of thaw speed, this governs how fast ice will freeze a gameobject.

Enable flyaway: This is used during collisions, to determine if the disturb particle will make a butterfly pool particle take to the air. This can also be set locally to a collision script, if there is no pool assigned.

Melt speed – Fast melt speed: Speed with which ice melts normally, and speed when override happens.

Enable local wind: This is used during collisions, to determine if the disturb particle will make grass behave differently than the rest of the pool.

Is grass: Enable to signal the pool is grass, so can be overridden if a collision enabled particle with “Enable local wind” hits near.

Override grass angle and speed: Use to locally override grass properties.

Delay: Time to do recalculations.

Optimize: Use above time to optimize speed. Less propagations will happen depending on time set above.

Use lerp: Use lerp instead of slerp for motion interpolation.

Delay calcs: Delay grass motion calculation by an amount, for speed optimization.

Optimize calcs: Use the above timer to optimize performance.

Relaxed: Allow particles to emit from painted positions.

Draw in sequence: Disable to get particle emit evenly in time and distribution from all painted positions.

Velocity towards normal: Allow an initial velocity towards the painted surface normal.

Normal velocity: Define velocity strength per axis.

Keep in position factor: Determines how long the particle will remain in the painted position before emission in relaxed mode.

Keep alive factor: Determines how long the particle will stay alive before lifetime reset.

NOTE: Objects will be painted as long as the pool sizes and max particle size are not overcome. Then positions must be erased (ice melt etc) to get new positions.

TIP: Increase the particle count to allow more positions to be painted.
TIP: To draw or delete positions, must have the object collider enabled.
TIP: Use rigid bodies on colliders for better and more accurate results.

Particle AI (v2.0)

Particle AI is a new script that allows particles to sense the surrounding environment and avoid obstacles, even when other forces dictate to go through them. There is a balance between the applied force and the script control where the items won't go through surfaces and this requires tweaking per case of use. There are multiple samples provided for this functionality, with attractive forces applied to the particles and tweaked to avoid surfaces encountered.

This script must be attached to the propagation pool to grab the items to control. All gameobject particles in the pool will be controlled by the AI.

Options:

Cast to enemy: Use spell casting on a target.

Cast distributer: Assign an "AttactParticles" script that will handle the multi transform emission through the new system in v2.0. The AI system will add the transform of the gameobject particle that needs to cast (when for example sees an enemy) to the emission transforms in the distributer and the cast will happen though a single pool (or any number of particles) the distributer is assigned to.

Cast enemy distance: Distance from the target to start casting.

P2: The propagator particle system, for debug purposes.

Leader speed: The speed at which the sample leader of the pack (first gameobject by default) will move with. The default motion is a circular one.

Leader frequency: Frequency at which the leader will move up/down offseted from the circular path.

Speed multiplier: Multiply the speed of the pack by this amount.

Swarm center: Center around which the leader will move.

Flock speed: The speed of the flocking around leader followers.

Avoid speed: The speed to avoid obstacles.

Rotation speed: The speed to rotate while trying to avoid obstacles.

Min object distance: Threshold distance at which motion will stop when very close an obstacle and a rotation only mode is initiated.

Min ground distance: The min distance to ground to start avoidance.

Raycast distance: Distance of raycast to the ground.

Cast distance forward: Forward raycast distance.

Cast distance left: Left/right raycast distance when sweeping the view field for obstacles.

Rotate to motion: Rotate items towards their motion vector.

Debug on: Enable debug lines to show the raycasting for obstacle discovery.

Log debug: Enable extra debug messages.

Start after: Time after which to start AI control of gameobject particles. Use when particles need to first be sprayed and there is a delay until all appear on the map.

Constant motion: Apply to keep moving while avoiding at the near threshold. Provides smoother motion.

Free leader: Let leader out of the pre-defined circular motion.

Look at leader: Make flock members look at the leader path while moving.

Min – max speed: Vary flock speed by this factor range.

Leader min distance: Minimum distance to leader that the follow motion will stop.

Side avoid speed: Speed when moving left/right for avoidance.

Front left/right check: Do a left/right sweep check in the same time of a forward found obstacle.

Do sweep: Sweep left/right continuously for obstacles.

Max slope: Slope after which the forward raycast hit is considered as an obstacle encounter. This allows flock members to go over near flat angled surfaces.

To leader factor: Motion speed factor along the vector connecting the flock item to the leader current position.

To forward: Motion speed factor along the forward vector of the flock item.

Rotate towards update: Update interval of the avoidance towards left/right vector. Use this to control how long the object will keep avoiding using the current left/right avoidance vector with speed defined in “Side avoidance speed”, before a new left/right vector (of its current orientation) is used for the avoidance.

Left right speed: Speed used when avoiding by moving left/right.

Cut motion left right: Stop motion towards leader when left/right obstacle is discovered and “constant motion” is disabled, to focus on avoidance.

Stay in place: Stay to the same place when encounter an obstacle below the threshold distance. Do not sweep/move left/right.

Forward collisions speed: Speed with which obstacle is avoided towards the opposite direction.

Use GUI: Use a sample enable/disable AI gui, for testing and demo purposes.

Collision propagator

Attach the script **“ParticleCollisionsPDM**” to a particle spray (or any particle) and link it to the pool system that holds the particles to appear upon collision.

This script can also be used to provide advanced particle to particle collisions. In this case a pool particle is not required (but can be used to provide extra effects upon collisions between particles or with the ground).

Finally collisions may locally override properties of another pool system.

Options:

Normal on Y (v2.0): Use when decal is facing the Y axis instead of Z axis. If unchecked the system will align the decal Z axis to the surface normal.

Decal per material (v2.0): Assign a decal to use per material of surface hit.

Material names (v2.0): String that the material name will include to enable each decal.

Decals for material (v2.0): Gameoject to use as decal, for the specified surface material.

Add effects (v2.0): Define an effect pool to use per decal type. This mode will create particles from the defined pools (propagation script) for each surface hit with the specified material. Use to add impact particle effects upon the decal creation.

Particle Pool: Link particle to a pool. Use this option to let particle collisions spread pool particles on collision. “Enable collisions” must be on in the pool particle. This is not required in order to use the collisions to locally override another particle pool, or for particle to particle collisions. If it is assigned, the properties of the pool will be used for the local overrides. Otherwise options defined locally will be used.

Flame force: Assign a force to apply to rigid bodies hit by the particles. Collisions must be enabled in the particle system.

Systems to override: Use this if no pool is linked, the collisions will override the assigned particle pool systems, given the options locally assigned in the script.

Cut off distance: Distance above which collisions will not affect pool systems defined in “systems to override” locally or in the linked pool system.

Inner particle collisions: Allow particle to particle collisions. This will emit collider objects and enable colliders on them at specific intervals.

Collider object: Object with collider to use for particle to particle collisions. Object must have a collider assigned and no mesh shown (unless needed otherwise).

Gameobject Instances: Instances of the colliders for particle to particle collisions. For debug purposes.

Divide factor: Fraction of particle count for emitted collider objects.

P11: Particle system for collisions, for debug purposes.

End of life: Enable colliders based on factors below, related to the particle life.

Life factor upper-lower: Percent of particle life collider will be enabled on.

Min source distance: Distance of particle from source in order to allow collider enabled.

Is fire: Locally override “systems to override” pool particles if they have “is_ice” and “enable overrides” enabled.

Is ice: Locally override “systems to override” pool particles if they have “is_fire” and “enable overrides” enabled.

Enable local wind: Disturb “systems to override” pool particles locally. Pool must have “is_grass” enabled.

Enable fly away: Disturb “systems to override” pool particles locally. Pool must have “is_butterfly” enabled.

Override color (v1.6): Color the pool particles locally, using the “New color” variable.

New color (v1.6): Color to use for the above option.

Batching (static and dynamic) for gameobject particles.

The new batching system is applied to the instantiated particle gameobjects pool root (assign “ControlCombineChildrenPDM” script to the root object). It allows for static batching as standalone and also dynamic batching can be used when the pool particle has the “Enable combine” option enabled in the Propagator script.

Options:

Generate triangle strip: Option for mesh combination.

Auto disable: Disable automatically after enabled. Use when an external script needs to activate and not have to check if the script got disabled. Used in Dynamic Batching by Propagator script to control the use of the script, so it can be applied only when objects have moved to maximize performance.

Skip every N frame: Performance control.

Make Active: Set script as active. The script will combine meshes to a single mesh at every update and recombine.

Self Dynamic Enable (v1.7): Allow the script to be used to any gameobject pool for dynamic batching (no need for Propagation script to control it). The script checks for position changes to activate the re-batching.

Self Dynamic Check Rotation (v1.7): Dynamically re-batch items when rotation is sensed.

Self Dynamic Check Scale (v1.7): Dynamically re-batch items when scale is sensed.

Is Active (v2.0): For debug purposes, shows when batching is enabled.

Minimum distance (v2.0): Distance from last position of any of the gameobject particles, at which batching will be recalculated. Use this for optimization so batching will happen only if one of the objects moves at a visible distance.

Stop self dynamic after (v2.0): Stop self dynamic batching mode after the specified time.

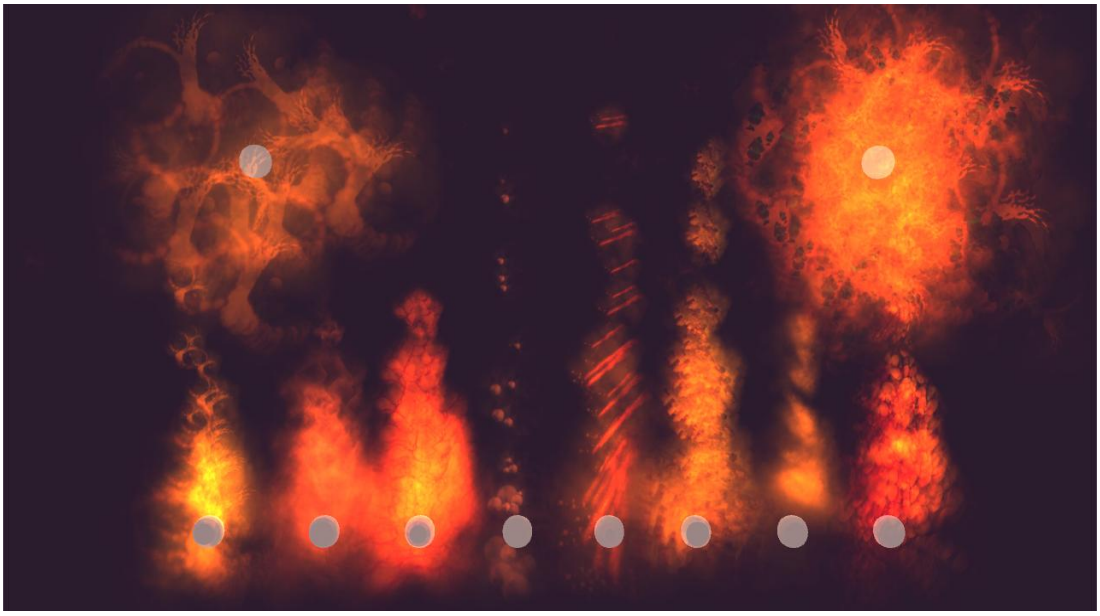
Decombine (v2.0): Turn batching off.

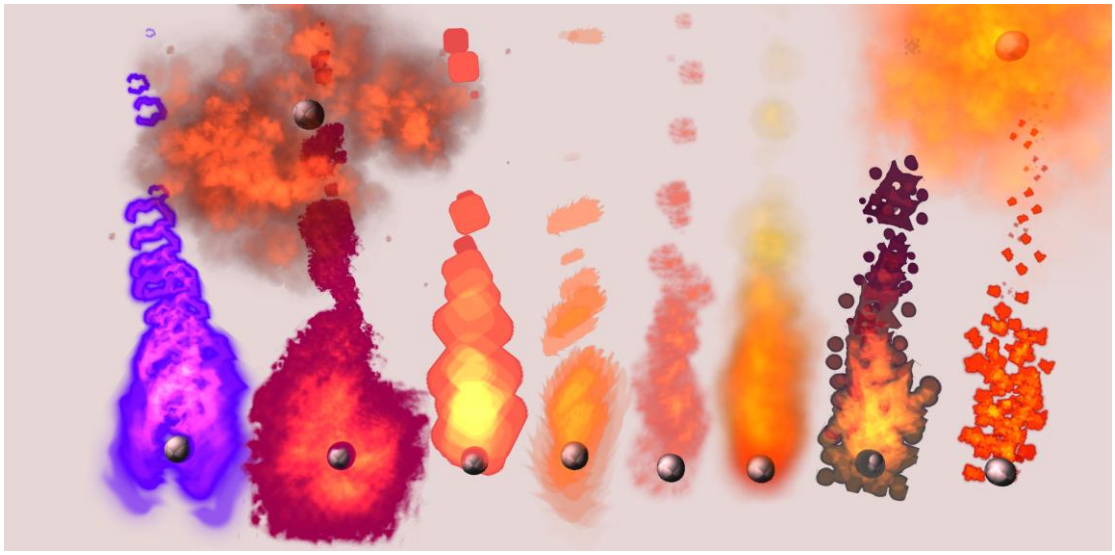
Put item in Batch Pool script

Use the "Put_item_in_PDM_Batch_pool.cs" script on a instantiate gameobject to automatically create a pool of batched objects with the batching script and assign every new instantiation of the assigned prefab to the object pool.

Use to instantly and automatically move objects to the pool as they are instantiated in the game. The prefab should be in a folder named "Resources".
Options:

Prefab in Resources: The prefab of the gameobject to instantiate and add to the batching pool. A pool will be created if it does not already exist. **The tag to use is "PDM_Batch_Pool" for the pool, this tag has to be added for the pool to be created.**





Special shaders (v2.0)

Special shaders are provided in v2.0 of Particle Dynamic Magic that allow detailed effects with minimal particle use and also shader based grass decal motion and effects.

These systems can be used together with the gameobject emission and custom batching system for spectacular grass and fire effects with powerful shader based motion and details.

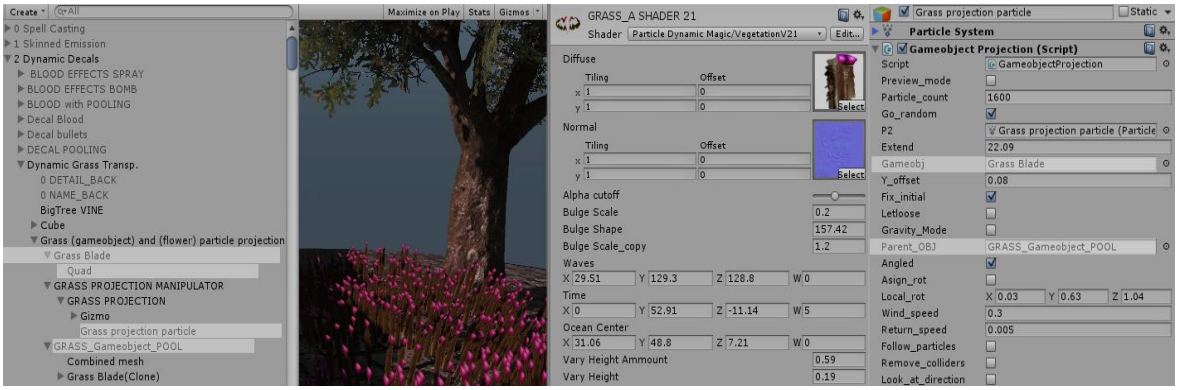
Also a special shader set is included that allows additive alpha blending of particles, for great look on dark and bright backgrounds and also includes a border feature for emulating toon like effects and bordered particles.

Grass shader

The grass shader comes in two varieties. One is to be applied in instantiated quads (through gameobject particles that follow surface normals) and supports transparency and the other is the opaque variety that is faster and must be applied to the provided custom grass blades for best result.

There are multiple samples, implemented using the projection system. In the system below, a projector (“GRASS PROJECTION”) casts gameobject particles (“grass blade”) using the “GameobjectProjection” script in “Grass projection particle” gameobject and creates a pool of instantiated items in “GRASS_Gameobject_POOL”, which applies the “ControlCombineChildrenPDM” script for the batching of grass. The grass blade consists of a single quad where the “Vegetation21” shader is applied.

Transparent grass



Options:

Alpha cutoff: Transparency cutoff.

Buldge scale & shape: Play with shaping of grass.

Waves: Control grass waving

Time: Control grass speed (“x” and “w” parameters)

Ocean center: Center of the grass ocean waving.

Vary height amount: Scale vertex height in Y axis

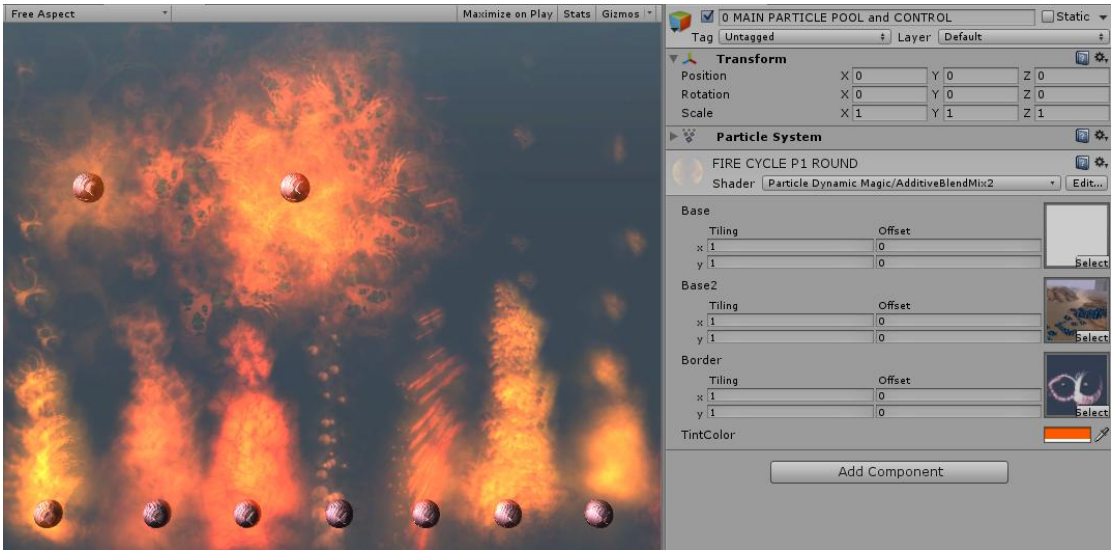
Vary height: Scale vertex height by time factor in cosine, factor of zero will return unit scale.

NOTE: “Vary height amount” of one and “Vary height” of zero will enable no height variation (default).

NOTE: All the above same options and setup procedure apply to the opaque shader as well, the only difference is that the grass is made from a custom grass blade, than a standard quad, shaped in pointy grass form.

Detail & inner motion particle shaders

This system allows the use of one or two main textures (Base & Base 2) that move in parallax and a mask texture (Border) that will create a smooth look around them. This arrangement makes it possible to use very detailed look and inner motion within each particle, thus creating richer and more spectacular effect using far less particles. The most complex of the shaders is the AdditiveBlendMix2 and is the recommended one for best result.



Helper scripts

Other scripts can be added, like the **“Circle Around Particle”** script, that makes an object orbit around a target.

Options:

Speed mult. = Speed of rotation

Up down motion = enable the orbital motion in Y axis

Shock effect = add a jittering effect

Up down speed = speed of motion in Y axis

Up down multiply = magnitude of Y axis motion

Jitter = amount of jittering

Sphere Object = scene object to orbit around

Cycle Shield UVs script

A script for animating texture UVs.

Options:

Speed: Speed to cycle texture UVs with.

Randomize (v2.0): Randomize speed.

Random factor (v2.0): Randomize factor.

Animate Trail Texture PDM script

A script for animating texture sheets on materials.

Options:

Animated texture: Animate texture, requires a texture sheet.

Texture rows-columns: Row and column count of the animation sheet frames.

FPS: Update rate for the animation.

Light Effects PDM script

A sample script of light handling, for flickering emulation, impacts etc

Options:

Reset: Reset light flicker.

Curve: Curve for defining light intensity over time.

Delay: Delay the application of the curve defined intensity.

Preview: Allow preview in editor.

Start-end light color: Start and end light color during the light effect play.

Loop: Loop the effect.

Lerp speed: Speed of lerp between colors.

Collider Message PDM

A sample script of calculating Spline proximity to a transform. It returns the closest curve point ID.

Options:

SplinerP Object: Spline to check for proximity, the check is done by a distance check against the transform position of the desired object.

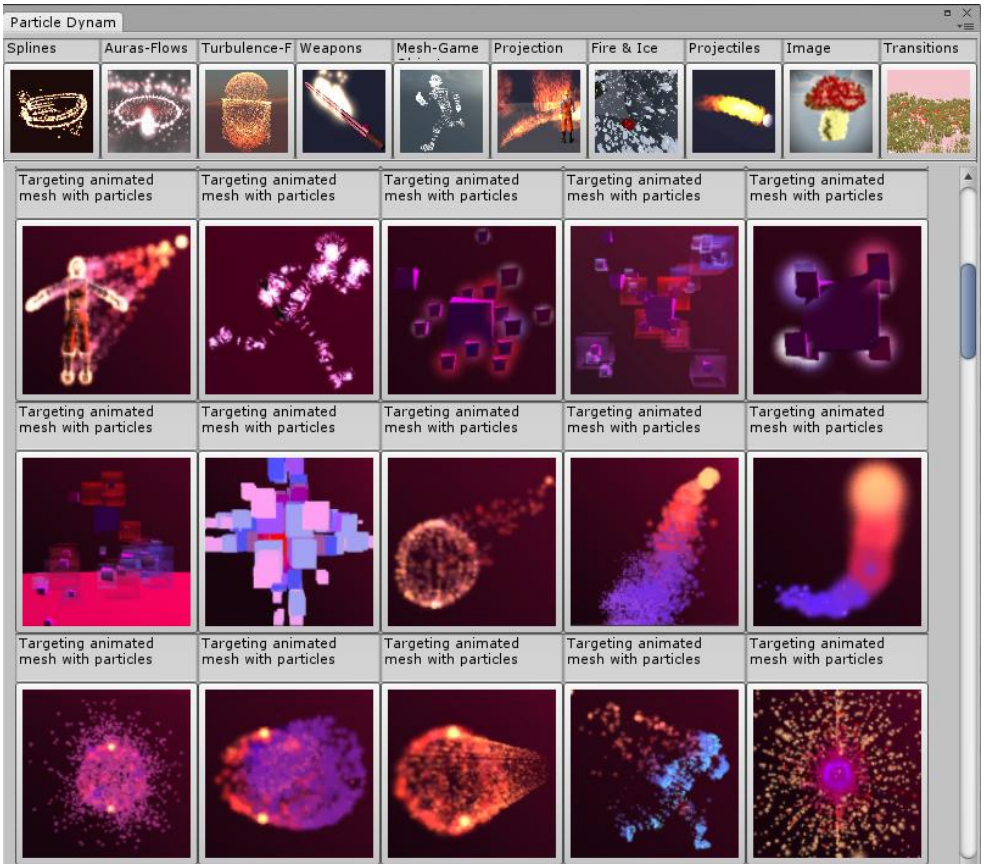
BOX: Transform to check proximity against, with distance check.

Prefab Manager

The Prefab Manager can be found in Window/Particle Dynamic Magic pull down menu. It contains an extensive collection of 200+ prefabs for the available effects. Click the icons to insert the corresponding effect in the scene. Particle Dynamic Magic v2.0 offers 100 extra ready to use effects (for a total of 300+ effects) in Collection 2 Prefab Manager, with an updated editor menu, that allows particle scaling and easier insertion of custom prefabs.

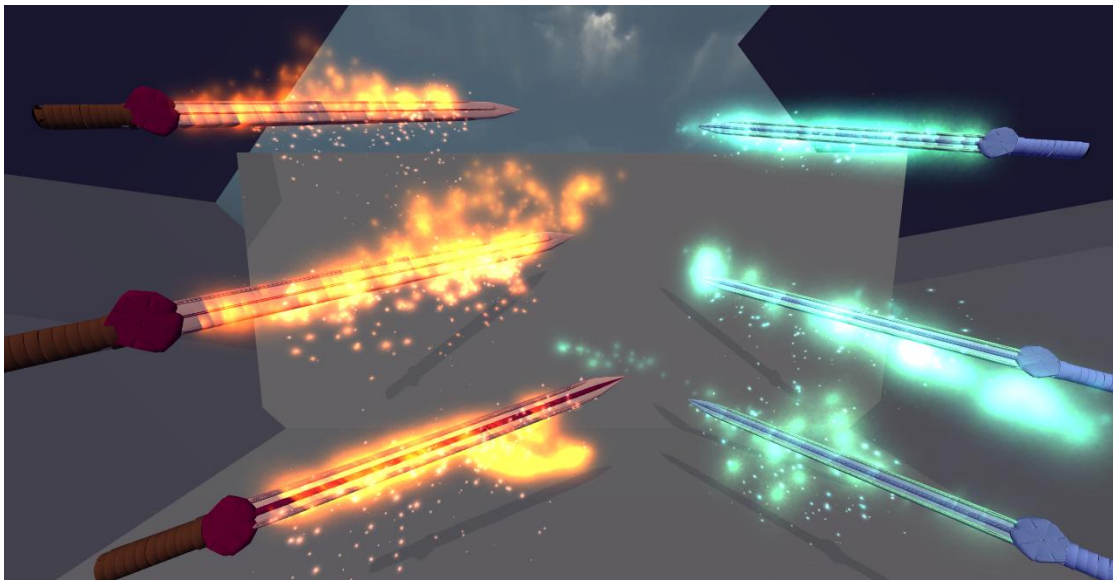
To insert a custom prefab in Collection 2 Prefab Manager.

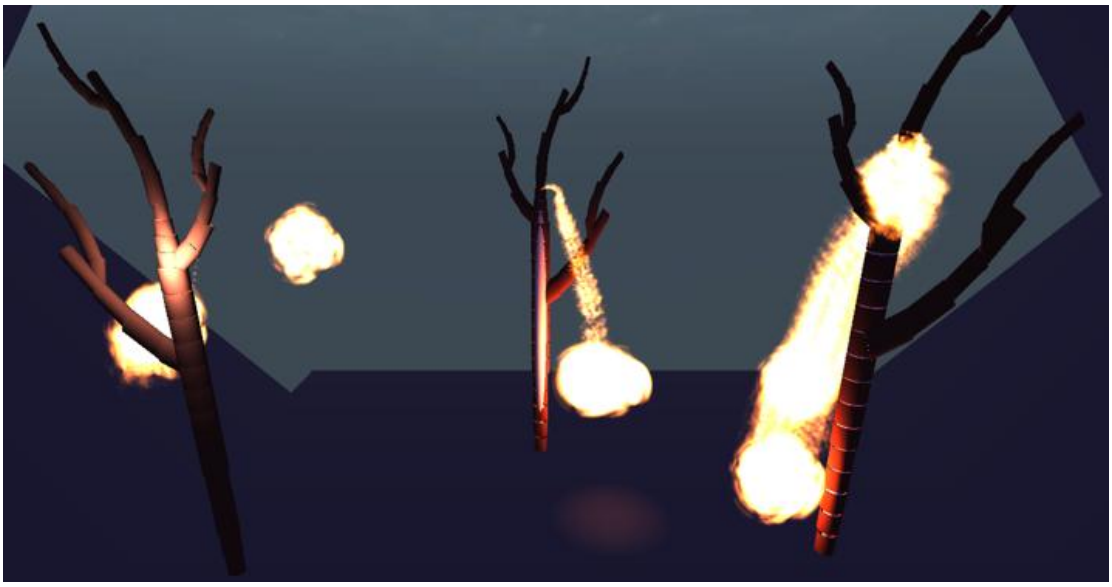
Name the prefab as “CAT_A- prefab name” where CAT_A is the first category (CAT_B for 2ond etc) and place it in the “PrefabWizard/Effects20/Prefabs” folder and add a similar in name JPG picture to be used as icon in the “PrefabWizard/Effects20/Icons” folder. The rest is automated and the preset will appear in the specific category and use the content after the CAT_A for description.

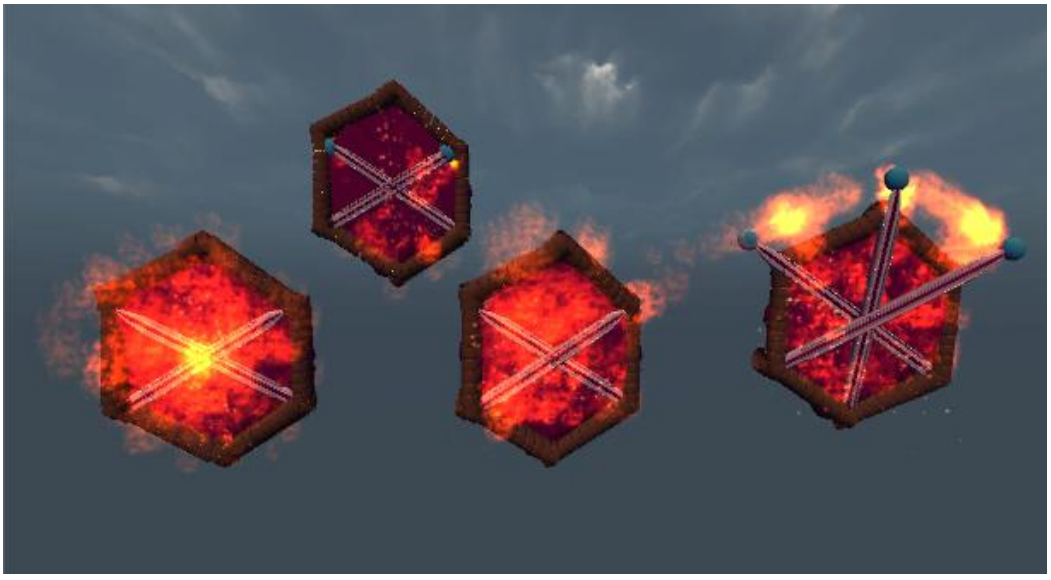


Effect samples (v1.01)

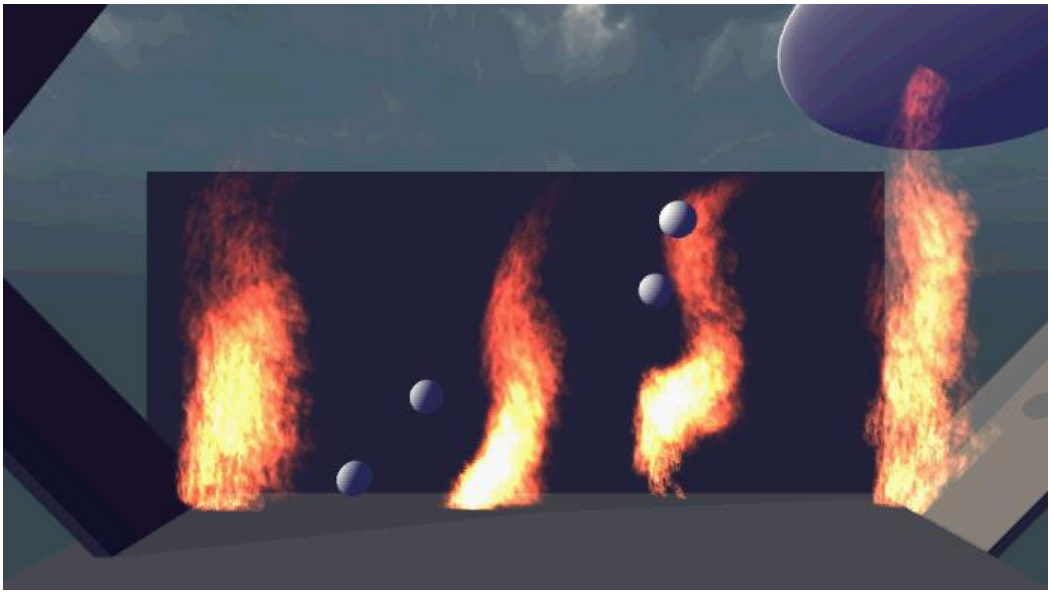
1. Magic weapon effects (fire – ice swords, staves, shields)



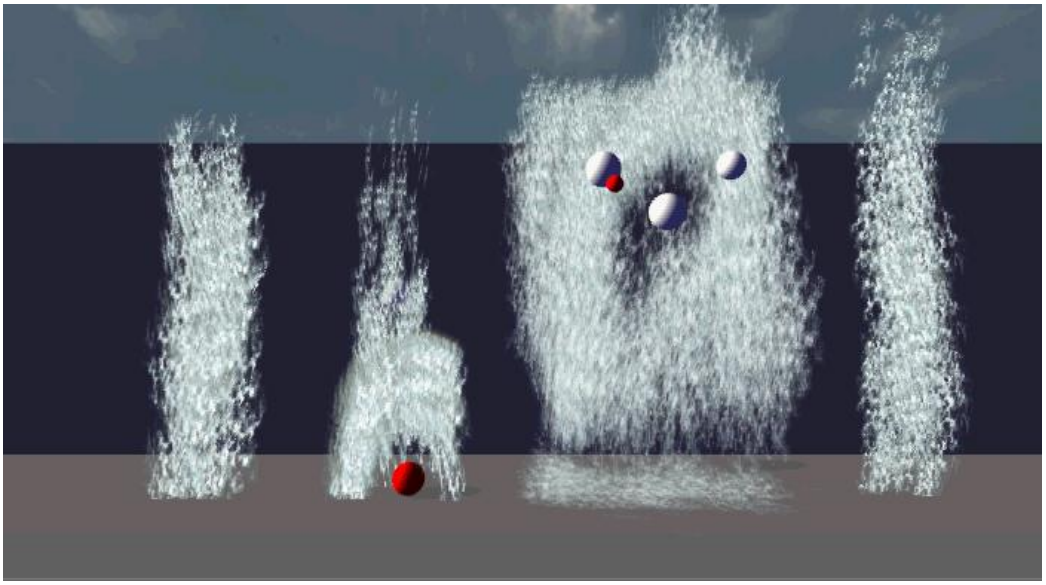




2. Spell effects (fire wall – fire – ice)

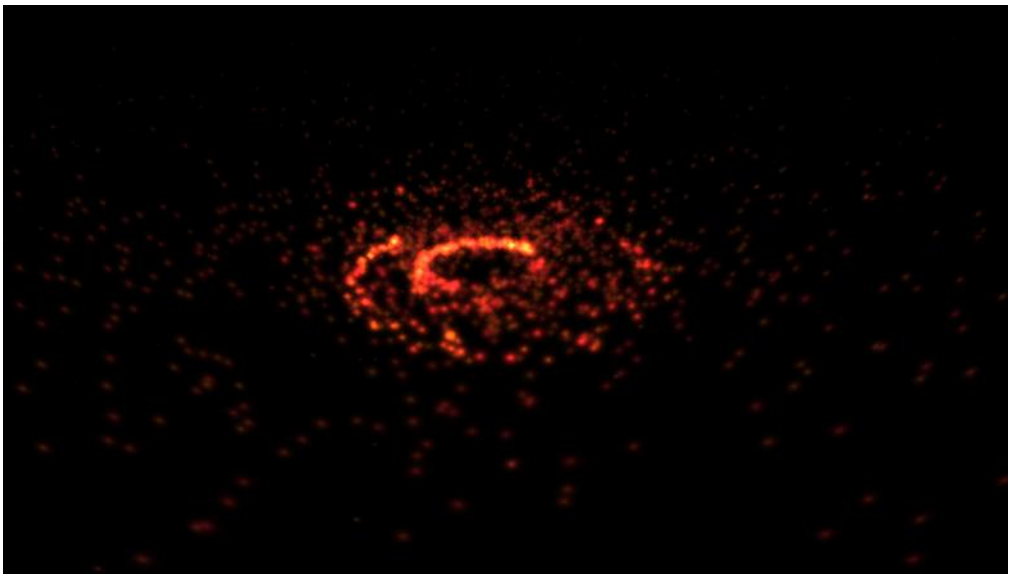


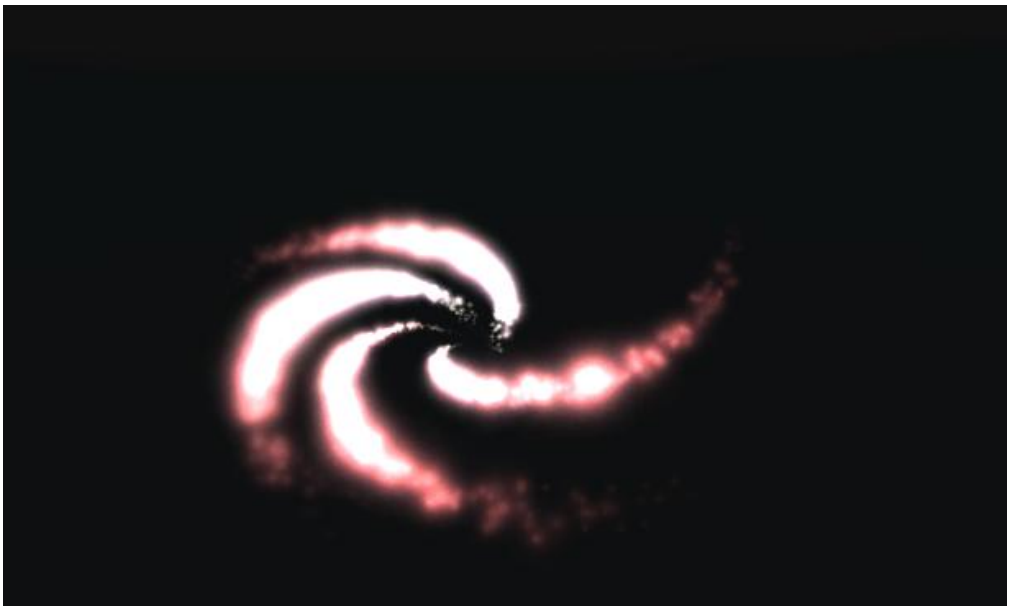
3. Dynamic waterfalls

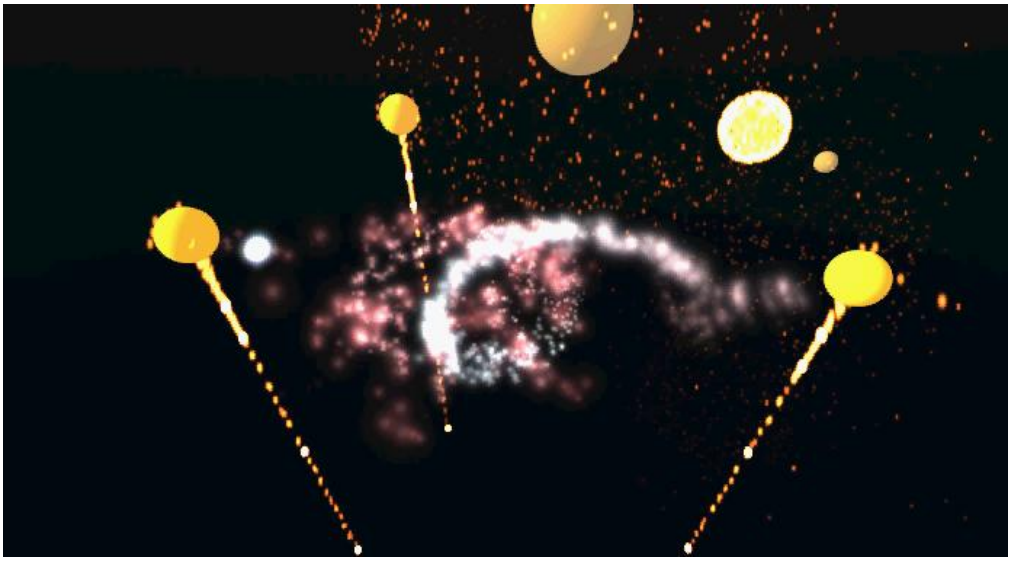


4. Growing nature – grass – flowers

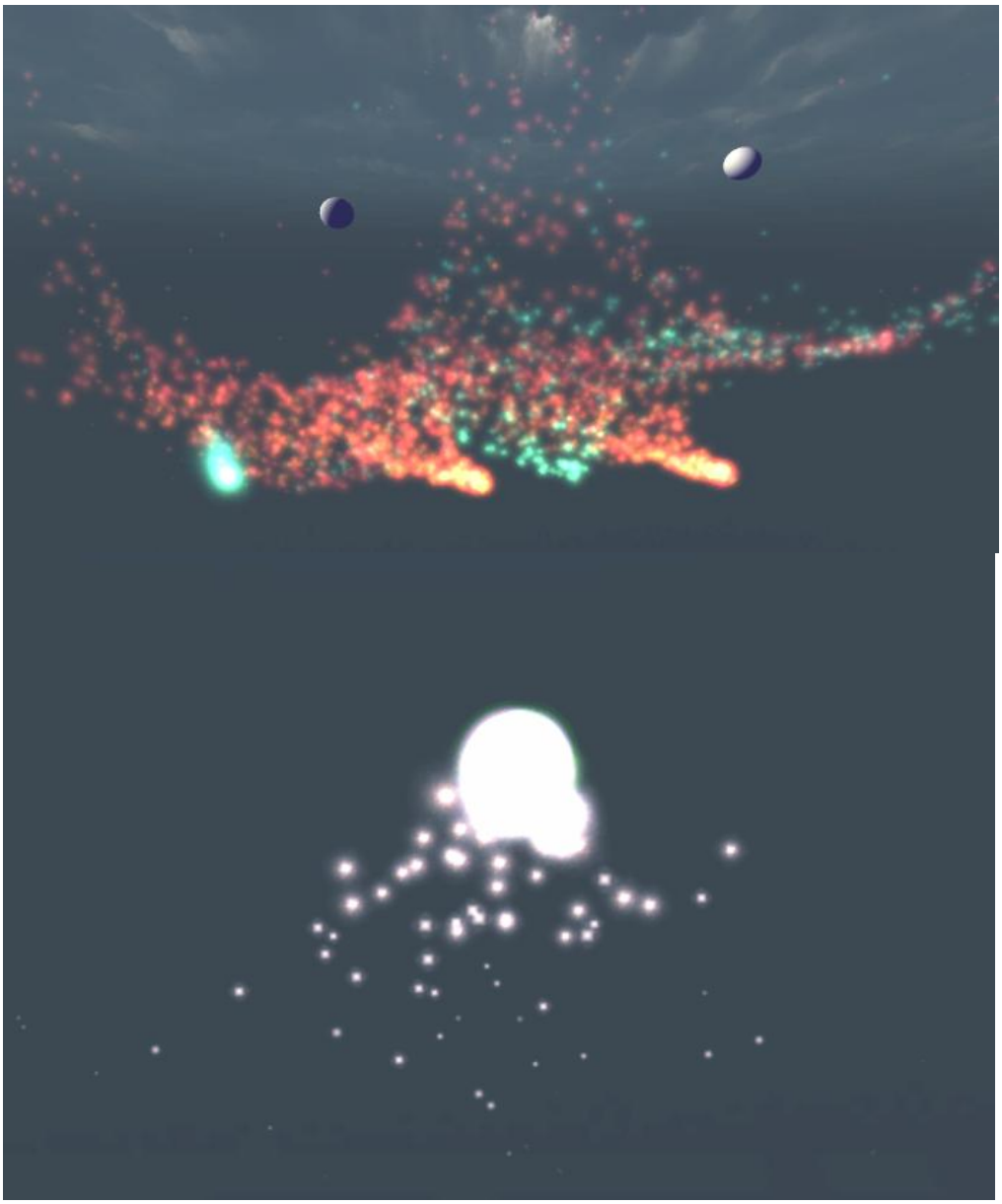
5. Spirals and space - galaxy formations



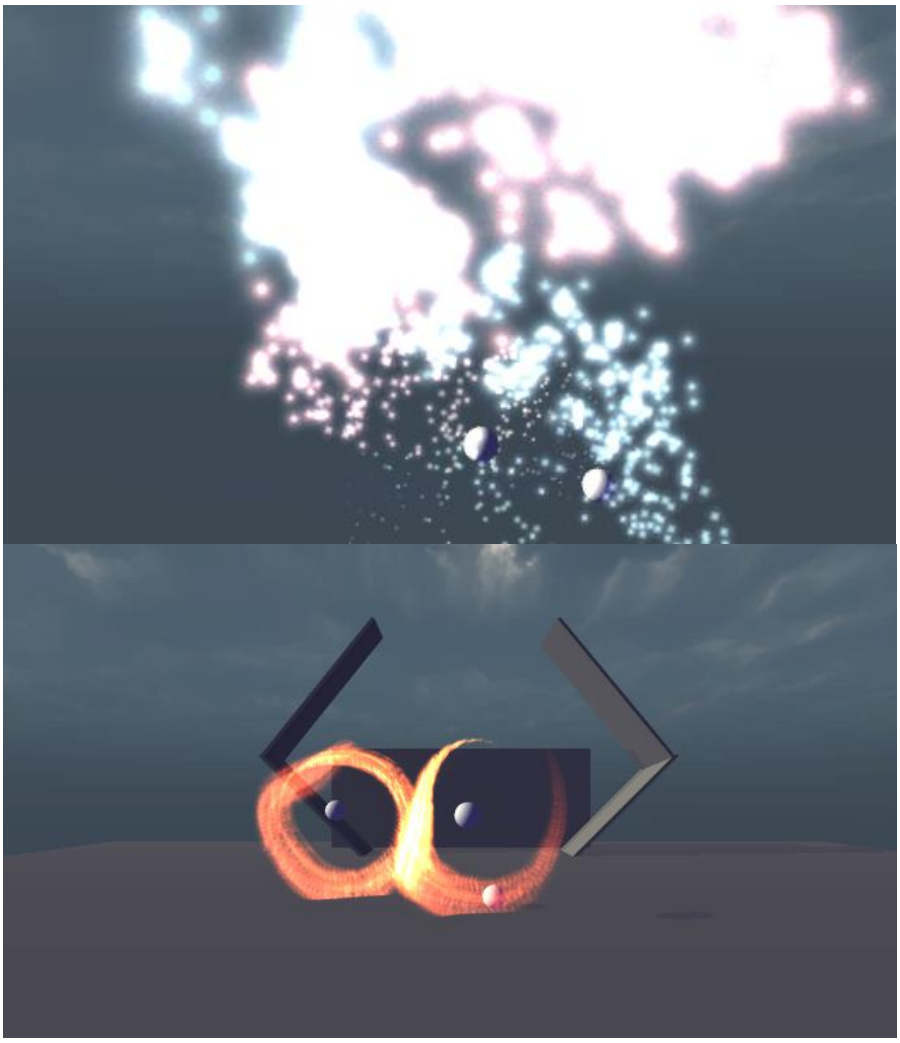


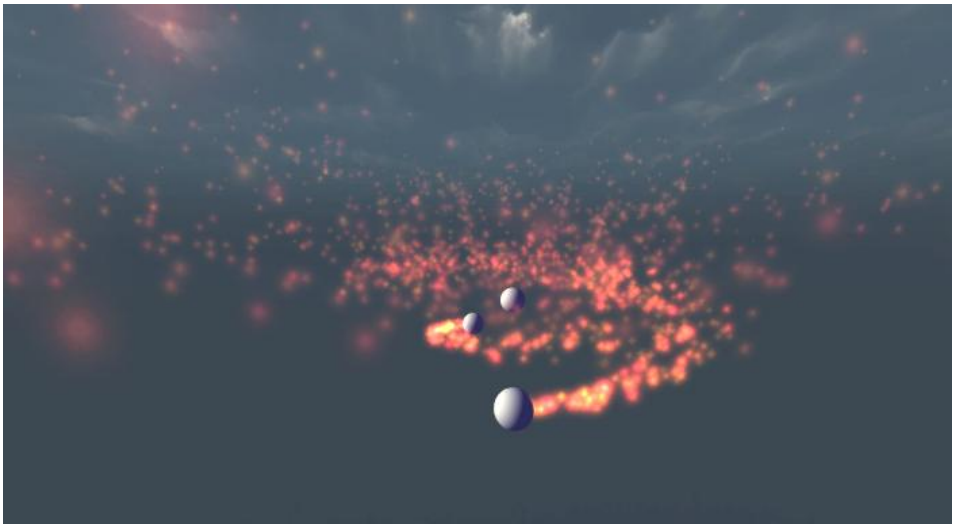


6. Global attractor/deflector class

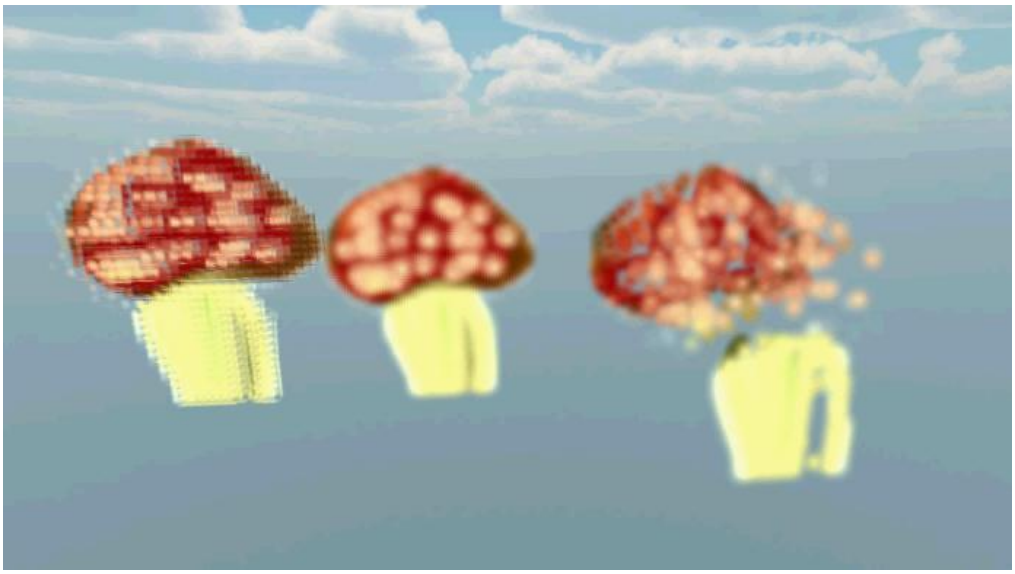


7. Orbital effects, vortex





8. Image based emission, control depth, particle size



9. Real time particle paint (draw from near particles)

10. Dynamic fog