Particle Playground - Version History

Version 3.0.3 (Current version)

Adding Playground Trails shadow settings, Windows Universal 10 build support and fixing some smaller issues.

Features

I. **Playground Trails shadow casting & receiving**
Playground Trails can now be set to cast and receive shadows. Note that the shader for the trail material needs to support this to have any effect.

II. **Windows Universal 10 build support**
Particle Playground projects should now be able to build towards Universal 10 apps. The implementation is still experimental where additional updates to the multithreading architecture could be needed in further releases.

Fixes

I. **Example scene issues**
Some example scenes got broken in the Unity 5.3 compatibility update due to a miss in PlaygroundNextScene.cs.

II. **Occasional Editor freezes when having the Playground Wizard open and hitting Play**
Due to the Playground Wizard checking the online version using a non-asynchronous method the Unity Editor could freeze before the returning timeout.

III. **Playground Trails having inconsistent length upon calculation cycle frame skips**
The Playground Trails was relying on a global delta time to update all trail points. This made each trail point inconsistent in their length as time would be updated slower when skipping calculation cycles. This was mostly noticeable on particle systems generating a higher amount of trails (~200+). Now each trail point is now relying on their own delta time which results in a consistent behavior no matter the time between calculation cycles.

Version 3.0.2

A small update with fixes for Snapshots, Playground Trails and compiling through the new MonoDevelop in Unity 5.3.

Fixes

I. **MonoDevelop compilation in Unity 5.3**
A float inside a ternary modification done in UpdateShuriken function due to a bug in MonoDevelop not being able to compile.

II. Snapshots not being able to create
Due to added data in the Playground Cache which could end up null made Snapshots not able to be created successfully. The error was not seen due to an asynchronous thread.

III. Playground Trails leaving parent behind in between play sessions
At occasions the parent of the Playground Trails would be left behind in Editor when going between Play/Stop mode. This is fixed with a Playground Trail Parent component which keeps a reference to the Playground Trails. The parent will now only be created once a new trail is requesting to be generated.

Version 3

Particle Playground 3 brings better performance, improved functionality and new technical features. Particle Playground 3 is compatible with the updated Shuriken particle system in Unity 5.3.

Features

I. Updated Shuriken component (Unity 5.3 support)
Particle Playground is now supporting all Unity versions spanning from 4.3.4 to 5.3. Particle Playground is compatible with the updated Shuriken component in Unity 5.3.

II. Increased performance through the Playground Thread Pool
A self managed thread pool is introduced which will take care of the multithreaded calculations. This is proven to increase performance by up to x1.5 while decreasing the amount of GC allocations by up to x6 from previous versions.

Max Threads will determine the amount of available reusable threads in the Playground Pool. Try to keep this at a low figure (for example 8) for best performance in regular desktop/mobile projects. If you're certain the end-target will have more CPUs you have the possibility of raising the maximum threads up to 128.

The previous .NET Thread Pool is still available, this can be toggled in *Playground Manager > Advanced > Thread Pool Method*.

III. Playground Trails - Draw mesh trails after particles
Particle Playground 3 introduces Playground Trails, with the ability to draw flexible mesh trails after particles.

A Playground Trail's length is defined by a time parameter, from where each created point in the trail will base its starting lifetime upon. The width of the trail is defined by the Width and Time Width parameters. Time Width is a normalized Animation Curve where each trail point will be scaled accordingly to its y-axis value over its lifetime.

Coloring of the trails is done through a gradient, where the Color Mode will determine if colors are distributed over the trail point's Lifetime or linearly through the Point Array. The UVs can also be distributed over the trail based on the point Lifetime or the linear Point Array.

A Playground Trail can use the Render Modes: Billboard, Horizontal, Vertical and Custom Render Scale. Billboard will rotate each trail point towards the assigned Billboard Transform (Main Camera by default). Horizontal will render the trails flat globally on XZ-axis, Vertical renders the trails flat globally on XY-axis. Custom Render Scale creates a custom render rotation and scale specified by a Vector3.

All mesh data in a Playground Trail is cached upon start and then iterated through throughout its lifetime to generate less memory garbage. The amount of cache is determined by the Maximum Points found in the Point Creation tab.

The Playground Trails are multithreaded (Advanced > Multithreading) and will do mesh calculations asynchronously, they will also dynamically batch to reduce draw calls.

IV. Playground Recorder - Particle recording and playback

Introducing the Playground Recorder which makes you able to record, playback and time scrub particle systems at any point. The method used is particle arrays acting as keyframes where current and next keyframe will be interpolated between during playback.

A recording can be stored in a Scriptable Object format in Editor (also while in Play Mode to be stored between sessions, which is preferable due to a more detailed simulation time). Note that serialization of Scriptable Objects outside the Unity Editor is not available, but they are however possible to read back into the Playground Recorder at runtime.

The Playground Recorder is multithreaded and will record and playback asynchronously.

V. More particle effects and preset categories in the Playground Wizard

A bunch of freshly made particle effects are now available along with categories in the Playground Wizard (*Window > Particle Playground*). The categories are based on the folder structure in *Particle Playground/Playground Assets/Presets/*.

The Preset Wizard now also handles categories when creating presets. When selecting 'Uncategorized' the preset will be stored directly in the specified *Assets Preset Path* set in *Settings > Paths* (default: *Particle Playground/Playground Assets/Presets/*). When selecting 'Resources' the preset will be stored in the specified *Resources Preset Path* set in *Settings > Paths* (default: *Resources/Presets/*).

Along with this the naming convention for User/Example presets is changed to Assets/Resources. An 'Asset' preset is contained in the project folder inaccessible until referenced by the project, whereas a 'Resource' preset is contained in a Resources folder and will always be accessible (i.e. can instantiate when using PlaygroundC.InstantiatePreset). Any presets within Resources will always be exported to the final build.

VI. Improved preset search pattern

It is now easier to search for presets using the preset search bar in the Playground Wizard. The search pattern is based on words without any particular order, where you can also type the category name to filter out presets even further.

VII. Custom Mask Sorting

You can now set your own mask sorting by calling `RefreshMaskSorting (int[])`. The passed in mask sorting array should contain all numbers from 0 to particleCount-1, it will then apply non-linear masking depending on the order of the numbers in relation to the actual particle array.

VIII. Particle Lifetime Strength for Manipulators

A Manipulator can now affect particles based on a normalized Animation Curve, where the Manipulator's Strength value will be multiplied based on the particles lifetime.

IX. Infinite Manipulator Shape

A Manipulator can now be defined with Shape: Infinite to bypass distance / containment checks of particles. This is a more performant mode if you want to affect all particles within a

Manipulator. An Infinite Manipulator will be displayed as red in the Scene View.

X. WebGL support

Particle Playground will now run in a browser using webGL. Keep in mind that JavaScript doesn't support multithreading and you will therefore experience less performant particle simulations.

XI. Memory improvements

A previous Particle Playground system was addressing up to 3MB of memory due to a fixed Shuriken particle cache. This is now reworked to stay flexible depending on actual particle count. A standard particle system is now reduced down to 13kB.

XII. Increased color calculation speed

The color conversion is now fully using Color32 to apply colors whereas no unnecessary color conversions from Color to Color32 is being made.

XIII. Lossy- or Local Scale option

Any Source that is using a Transform to base birth positions on now supports lossy scale. This also includes the Playground Spline component where you can change between Local- or Lossy Scale in the Advanced tab.

XIV. Particle Position, Color, Scale and Velocity functions

You can now set particle position, color, scale and velocity in a more convenient way through script without having to dig into the Playground Cache. Use the available functions (on your particle system reference):

```
- ParticlePosition()
- ParticleColor()
- ParticleSize()
- ParticleVelocity()
```

XV. Playground Components Menu

Introducing the Playground Components Menu in the Playground Wizard where you can attach Playground related components to your GameObjects.

XVI. Improved Editor functionality for finding assets

Particle Playground will now cache the location for the Playground Settings.asset file if not found at original path. If you have moved your Particle Playground folder into another location and the Playground Settings.asset file can't be found you will be presented with a dialogue to search for the file in the Playground Wizard. This functionality improves loading times if you have a large project.

XVII. Global or Local Time Scale

The Playground Manager can now determine whether to use global or local time scale within the scene (*Playground Manager > Advanced > Global Time Scale*). If this is turned off the Time.timeScale won't affect the particle simulations.

XVIII. (Bonus) A simpler and faster alternative to Playground Follow

There's now a script called PlaygroundTransformFollow.cs found in the Simple Scripts folder which will make objects follow particles. This version uses a static pool of objects and does not contain event listeners for particle birth/death, which also means that all followers will be active at all times. This is suitable for objects following particles within a looping particle system.

XIX. (Bonus) Spline Meshes w. Perlin Noise

Use the script PlaygroundSplineMesh.cs onto any Playground Spline to construct meshes along your spline. The component also supports Perlin Noise which adds a more natural geometry along the mesh.

Fixes

I. **Emission Rate at 0 and SetParticleTimeNow**
SetParticleTimeNow will now take Emission Rate into account while set to 0.

II. **Wrongly set GUI disables in Inspector**
- Disabled GUI on Source: Script when Emission tab is unfolded.

III. **Spline missing upon particle system instantiation causing null reference**
When instantiating a particle system with a missing Playground Spline null reference will no longer be thrown. You are in that case expected to hook up any missing reference manually through script.

IV. **Scripted emission color conversion fix**
When emitting particles of certain color values such as RGBA(0, 255, >0, 255) the result was RGBA(0, 255, 255, 255). This was due to an erroneous and unnecessary color conversion between Color and Color32.

V. **Fix for Snapshots with Skinned Meshes throwing null reference in Source tab.**
VI. **Fix for Skinned Meshes trying to initialize when GameObject is inactive.**
VII. **Fix for Source Paint Brush not being set correctly.**
VIII. **Fix for mismatched layout group in Playground Wizard while downloading extensions.**
IX. **Fix for collision lifetime loss making particles not rebirth.**

X. **Internet connectivity lookup for checking latest version and caching extension files**
Previously you may have experienced disruptive behavior of the Playground Wizard in Editor and not being connected to the internet. The Playground Wizard will now try the internet connectivity before accessing any server files. *Note: Particle Playground only uses an internet connection to check the latest version and to download the "Extend Your Playground" xml file/ thumbnails (these are cached in the PlaygroundCache folder inside your Project folder).*

Version 2.26

Features

I. **Sticky and improved collisions**
Collisions has become a bit more diverse. You can now choose to have your particles stick onto specified collided surfaces (using a layer mask). Along with this the Collision Cache is introduced, where you can extract information about every collided particle. Through script a couple of new functions are available to work with the collision information:

```
- HasCollisionCache()
- HasCollided()
- GetCollisionPosition()
- GetCollisionNormal()
- GetStickyPosition()
- GetCollisionTransform()
- SetSticky()
- UpdateSticky()
- ClearCollisions()
```

Note that you need to enable *Advanced > Misc > Force Collision Caching* in case you wish to extend your scripts upon the available collision information. Otherwise it will only be used when you enable sticky particles.

In *Advanced > Misc* you'll also find a toggle for Collision Precision, this determines if collision checking should be using Vector3.Distance (if true) or Vector3.SqrMagnitude (if false). Having Collision Precision enabled will ensure more exact collision positions with a slight performance penalty. This can be useful when you have set a small Collision Radius.

II. **Axis Constraints for Manipulators**
You can now control which axis a Manipulator can affect forces and/or repositioning of particles.

III. **Apply random Delta Movement between two values**
Delta Movement Strength can now be applied as a constant (as before) or random between two values. This give particles a more natural slingshot behavior, noticeable when birth rate is high during Delta Movement distribution.

IV. **Multiple Preset export from the Preset Wizard**
Introducing the possibility to export multiple presets using the Preset Wizard's Publish mode.

Fixes

I. **Particles with collision appears jumpy on rest**
By enabling *Advanced > Misc > Sync Particles To Main-Thread* you'll now be able make particles stay perfectly still once they come to a rest.

II. **Global Manipulators tracking collisions**
The index out of range issue for Global Manipulators tracking collisions is now fixed.

III. **Particle system's Force Axis Constraints not cloning upon duplication**
When cloning a particle system the Force Annihilation > Axis Constraints will now be taken into account.

IV. **Non-emitting system with Disable On Done runs before loop exceed upon instantiation**
A particle system will now check that a loop has been exceeded before calling the specified Disable On Done. This mends the issue of a particle system destroying or going disabled when instantiating a non-emitting particle system with Disable On Done.

V. **All initialization of a particle system is moved to Start()**
The initialization of a particle system is now moved away from Awake() and into Start(). This enables you to configure a particle system using custom code in Awake before it initializes. Essentially this opens up for assigning different sources before the system starts to calculate and setting specific values which would otherwise require a second call to rebuild the cache.

Version 2.25

Quick-fix of a deeply nested issue related to events and death properties.

Fixes

I. **Death on collision while sending events on non-looping systems**
The issue where particle events would keep going when death property / collision death had affected the particles on non-looping systems is now fixed.

Version 2.24

Adding the new Math Manipulator, Extend Your Playground gallery and swiping away some bugs.

Features

I. Math Manipulator
A new Property Manipulator is available called *Math*. This can be used to apply common math operations to certain particle properties. Available properties to alter is *position*, *rotation*, *size*, and *velocity*. Algorithms available is *sine*, *cosine*, *linear interpolation*, *add* and *subtract*. You can also *clamp* the in- or out value.

II. Extend Your Playground
A new category has opened up in the Playground Wizard (*Window > Particle Playground*) called Extend Your Playground. This is a gallery of extensions based on the Particle Playground framework which will hopefully grow over time.

III. Emit by quantity overload
There's now an Emit() overload which will emit by passed in quantity. All other values will be as set in previous emission (or as in Inspector).

Fixes

I. State Chroma Key and Depthmap
Chroma Key and Depthmap is now connected when initializing a state.

II. Emit(true) not responding on instantiated non-emitting systems
Emission will now start on an instantiated particle system with Emit set to false. The issue was connected to particles being set to not calculate in the calculation loop while booting up.

Version 2.23

This is a small update adding some mobile example scenes and fixing an issue with particles freezing on mobile devices when turbulence initializes in the middle of simulation.

Features

I. Mobile scene examples
A couple of mobile scene examples has been added. These intend to show how particle systems could be setup for low-end devices. Keep in mind fill-rate will always be an issue for mobile, where rendering transparent objects will have a noticeable impact on performance. Also when using skinned meshes as Source, keeping vertex count as low as possible will help significantly.

Fixes

I. Particles freezing during simulation on mobile
Fixed issue with particles freezing in the middle of simulation. This was due to the simplex algorithm not initializing in the correct order where thread completion check would fail.

Version 2.22

This update is focused on adding Windows Store Apps with SDK 8.1 support and addressing some issues found in the particle calculation routines.

Features

I. Windows Store Apps with SDK 8.1 Support
Particle Playground will now run in applications based on the Windows Store 8.1 SDK. This is the compatibility release candidate 1, which currently doesn't support debug builds. Further improvements in this area will come. Please note that the example scenes in the project isn't adapted for low-end devices.

II. Added LoadAndApplyMask() to Snapshots
You can now transition any Snapshot and edit the particle mask at the same time. This enables flexibility for which particles should be visible when a snapshot transition has finished. Along with this the transitioning will iterate over current to loading masked particles, fading them in or out.

Fixes

I. Particles that won't send death events to Manipulators
Fixed issue with particles that wouldn't send their event information to Manipulators tracking particles with Tracking Method: Manipulator ID.

II. Global Manipulators tracking particles may halt simulation
Each particle system now has a check in their calculation routine to be combined on the same threaded call (ThreadMethod.OneForAll) if a Global Manipulator will affect them. This solves the asynchronous conflict of adding and removing tracked particles when called from different threads.

III. Global Manipulators have inconsistent velocity
The Global Manipulator calculation is now back to iterate internally for each particle system as it is providing a concise behavior. Performance should not be impacted.

IV. Delta Movement isn't connected to Time Scale
Fixed issue with Delta Movement not taking Time Scale into account.

V. Particle death flickering in Unity 5
The standard value for Minimum Shuriken Lifetime is raised to 0.08 from 0.05 resolving death flickering on particles in Unity 5.

VI. Prewarming Nearest Neighbor Lifetime Sorted systems won't have any effect
Fixed issue with prewarming particle systems with Lifetime Sorting: Nearest Neighbor. The method would only prepare the source positions but not scrub to the specified simulation time.

VII. Minimum to Maximum Sliders sometimes turns NaN
The Minimum to Maximum Sliders in Inspector is improved to not cause the value to turn NaN at rare occasions when initially dragging.

Version 2.21

This is a small update addressing some issues found in the 2.2 release.

Fixes

I. **Out Of Memory upon install into large projects**
GetAssetsOfType in PlaygroundSettingsC was consuming a lot of memory, this could crash the Editor on large projects. The method has been reworked to only call the objects needed and is also using a temporary cache.

II. **Boot routine called twice on first run**
Fixed issue with boot routine being called twice on first run (due to previousLifetimeEmission not getting correct value).

III. **Some Nearest Neighbor sorted particle systems is initially delayed**
Fixed issue with Lifetime Sorting: Nearest Neighbor where the particle system would be delayed unnecessary frames at first launch. This was connected to the sorting's automatic Prewarming to find source positions and initialization of the turbulence algorithm.

IV. **Casting error on Unity 5.0 beta 21**
Fixed BezierControlPointMode casting error on Unity 5.0 beta 21.

V. **Casting error on Unity 5.0 beta**
Fixed GUI check for Shadow Casting Mode.

Version 2.2

This is the largest update to date with an array of requested features and fixes. You'll find an example project in the package from now on which is the scene from the 2.2 release video (https://vimeo.com/118170374).

Features

I. **Playground Splines**
Introducing the Playground Spline component to create and edit splines in the scene. A particle system can emit from the spline, use it as an alternative to Lifetime Positioning and target it with a Spline Target Manipulator. A Playground Spline is not exclusive for particles, you can use it for any other component as well. See the example scenes found under the Splines folder for more details how to work with splines and particles.

II. **Multiple Transforms as Source**
Introducing the possibility to create source positions from a list of transforms. You can specify if the particle distribution should see the source as separate or conjoint transforms, this has effect on how particles birth positions are placed and noticeable when working with any method reading the particle array linearly (for instance Overflow Offset, Lifetime Sorting: Linear/Reversed, Particle Mask and Color Method: Particle Array). See the Square Frame preset for example use.

III. **Prewarming**
A particle system can now prewarm its simulation upon creation (Advanced > On Enable > Prewarm). Use Prewarm Lifetime Cycles to determine where in its lifetime it should start and

Prewarm Cycles to determine how many calculation steps should be used (higher means more accurate but will demand more CPU). Prewarming is multithreaded for performance.

IV. Particle Time Scale

You can now set the simulation time seamlessly (Advanced > Time Scale). This also lets you pause an effect by setting Time Scale to 0.

V. Reworked update routines

Preparation, calculation and final particle output has been restructured. This was to create an overall smooth particle repositioning and give better time sync. Each particle also has an individual calculation check to not interfere with final output when Sync Particle To Main-Thread is enabled. Previously you may have experienced jaggy movement especially on heavier scene setups.

VI. Size particle by array

A new size option to particles is introduced where you can set the size depending on the particle's position in the array over a normalized AnimationCurve (X:0 means first particle and X:1 means last particle).

VII. Lifetime Emission

It's now possible to specify the Lifetime Sorting pattern over a normalized Lifetime Emission value (Particle Settings > Lifetime > Lifetime Emission). Essentially this lets you emit particles during a specified period of time and break the otherwise obvious Lifetime Sorting pattern. For instance a Lifetime Emission of 0.5 will emit all particles during the first half of the specified Lifetime. The Lifetime Sorting will still remain intact.

VIII. Transition Back To Source

Every particle can now transition back to their source position during their lifetime. When having transition enabled you determine the amount to transition with a normalized AnimationCurve, where 1 on the X-axis means full lifetime and 1 on the Y-axis means full interpolation time. You will find it in the Forces > Force Annihilation tab.

IX. Spline Target Manipulator

You can now target splines using a Property Manipulator set to Spline Target. Use the Target Method to determine if particles should position over the Spline Time or the Particle Time.

X. State Target Manipulator

Will make particles target a State (pixels/vertices) within a Property Manipulator.

XI. State Chroma Key

You can now filter out a desired color (with spreading option) when creating a State. This can be used to remove any unwanted colors which can come in handy if your texture doesn't have alpha but instead a solid background color.

XII. Color Method

The Color Method lets you determine how particles get their gradient colors distributed. Using *Lifetime* will apply the gradient over a particle's lifetime (as previously). Using *Particle Array* will color the particles linearly using the array of particles as time for evaluating the color gradient, where leftmost color in gradient is the first particle and rightmost color is the last particle. This can be useful when working with any linearly positioned particles (using Playground Splines, Lifetime Positioning or Overflow Offset).

XIII. Particle Mask Sorting

Sorting methods for the particle mask by linear, reversed and scrambled is introduced. For performance and more versatile use you can now enable and disable the particle mask (rather than by setting Particle Mask to other than 0). To refresh the scrambled sorting you can call

yourParticleSystem.RefreshMaskSorting().

XIV. Texture Sheet Animation

Texture Sheet Animation (for particle UVs) is now implemented directly into the Particle Playground Inspector. You can find it in the Rendering tab.

XV. Sort Mode & Sorting Fudge

Sorting particles within the particle system is now available in the Rendering tab.

XVI. New Nearest Neighbor Origin methods

When using Lifetime Sorting: Nearest Neighbor/Reversed you can now specify the origin method by Source Point (as previous), Vector3 or Transform. Using a Vector3 or Transform will determine the sort origin from a point in world space. Using Source Point will determine the origin from a generated particle birth position in world space.

XVII. Cast & Receive Shadows

Cast- and Receive shadows is now available in the Rendering tab. Note that you will need a shader which can work with shadows (particle shaders doesn't by default).

XVIII. Improved calculations

An improvement to certain routines such as distance checks makes parts of the particle calculation loop faster to execute (such as collisions and all Manipulators).

XIX. Distance effect for Size and Color Property Manipulators

Resizing or coloring particles from a Size or Color Manipulator will now apply the Smoothing- and Distance Effect. This ables you to apply the effect additive with the distance from the manipulator taken into account. Note that Manipulator Strength will determine the multiplier over distance when not using a Transition method, where Manipulator Strength will be the speed of the effect over time when applying a Transition effect.

XX. Minimum to maximum spherical shape

A new method to calculate minimum to maximum random Vector3 values is introduced where you can choose between Rectangular (behaves as previously), Rectangular Linear, Spherical, Spherical Linear. This can be applied to the Initial Velocity and Source Scatter values to get a spherical emission/position shape instead of a quadratic. Using any linear method means the distribution will take the array of particles into calculation.

The Range determines the size of the sphere in minimum to maximum random value. When applied to Source Scatter the Range of 0-0 means no scattering, 0-1 means random within one unit from source position, where for instance 1-2 creates a random shell one unit thick. When applied to a force, the Range will amplify the force within a minimum to maximum random spherical direction.

XXI. Collision improvements

- Calculating collisions is faster to execute.
- You can now also invert the affect upon rigidbody collision.
- An exclusion list is introduced, to keep specified transforms from colliding with particles.

XXII. Improved UI

The Inspector of a particle system has been divided into section tabs for easier navigation. Also introducing quick info at the far right of each tab.

XXIII. More Emit() overload options

You can now call custom emission through script by these new overloads:

- Emit (Vector3 givePosition) Other values by Inspector

- Emit (Vector3 givePosition, Vector3 giveVelocity) Color by Inspector
- Emit (float giveLifetime) Set lifetime of particle, other values by Inspector
- Emit (Vector3 givePosition, float giveLifetime) Set lifetime of particle, other values by Inspector
- Emit (Vector3 givePosition, Vector3 giveVelocity, float giveLifetime) Set lifetime of particle, color by Inspector
- Emit (Vector3 givePosition, Vector3 giveVelocity, float giveLifetime, Color32 giveColor)

Any emission call without the full declare of instructions will let the previously scripted value be in control (same as set in the Inspector by the Source tab). Setting independent lifetime of an emitted particle is also introduced.

XXIV.Out Of View - a reworked calculation culling

There's a new section called Out Of View found in Advanced. The previous Auto-Pause Calculation is reworked to get triggered by the camera frustum, along with that trigger options where you can set trigger positions from a Transform, size of the trigger area and its offset. The trigger will only work in Play Mode and will only disable calculation when no camera (including Scene Camera) no longer is rendering the particle system.

XXV.Render toggle

You can now control the render output by toggling Rendering > Render Mode > Renderer.

XXVI.Particle stretching improvements

Lifetime Positioning can now cope with Render Mode: Stretch and its settings. You'll also find a toggle for Start Stretch. Thanks to the reworked update routines stretching will also appear more stable.

XXVII.Sync Particles On Main-Thread

Previous setting Sync Positions On Main-Thread has been reworked to sync particle time, birth positions, rotation and size. This will ensure smooth updates for each particle.

XXVIII.Particle Tracking Method and improvements

A Manipulator tracking its particles can now use a much faster method by comparing each particle by its Manipulator Id. This will only work for non-overlapping Manipulator areas. The original method is done by the Particle Id. This will work for overlapping areas but requires a particle-to-particle comparison, which naturally makes it heavier to compute.

You can now also use IsParticleInsideManipulator (particleId, manipulator) to detect if a certain particle is within a specified Manipulator. This method doesn't require any tracking.

XXIX.Hierarchy icon - an indicator

An icon in the Hierarchy is added to each Playground system. This serves as an indicator if the particle system isn't currently calculated (for instance when out of view or when the simulation has expired) by turning beige. The icon will turn red to indicate that the calculation loop is reporting a slow iteration, this is a good way to detect which particle system is slowing down a scene.

Fixes

I. Skinned Meshes with Optimize Game Objects enabled

A particle system can now compute having a skinned mesh set to optimize its GameObjects at its Import Settings. Note that this requires the skinned mesh's vertices to be extracted each calculated frame, it will produce a lot of GC allocations. Consider to not use Optimize Game Objects on any skinned meshes Particle Playground should calculate.

II. Particle positions upon moving the source

You should now experience a more exact particle positioning when moving the source while using local space or Only Source Positions. Skinned meshes needs to be set to run inside the

particle calculation (Playground Manager > Advanced > Multithreading), otherwise you may experience one frame of teared positioning.

III. Delta Movement no longer responding
Applying Delta Movement as a force is now back to function as expected.

IV. Vector3AnimationCurves causing null reference on HasKey check
A null check will now run for Vector3AnimationCurves upon particle system selection.

V. Manipulator color properties
- Global Manipulators will now change color properties as expected.
- Local Manipulators setting Lifetime Color with Only Color In Range will now work as expected.

VI. Manipulator with Additive Velocity property & Transition gives unwanted velocity scale
Additive Velocity property along with transitions is now disabled as the additive behavior will work as a transition effect within itself.

VII. Assigning a State mesh without normals
The construction of a State will now handle if the mesh doesn't have normals.

VIII. Texture Sheet Animation and scripted emission
Particle time will now reset upon an Emit() call which makes Texture Sheet Animation cope with Source: Script mode.

IX. Simplex turbulence fails to initialize if null during simulation
If the simplex turbulence is null during simulation and strength is set from 0 to above the algorithm will now initialize automatically.

X. Particle stretching on rebirth
Fixed stretching on rebirth where some particles could have their stretch upon death remaining.

XI. No rigidbody affect when Bounciness is 0 upon collision
A particle does no longer need to bounce to affect any rigidbodies upon collision.

XII. Fix for non-uniform emission rate
The emission rate is now linearly distributed thanks to a delta from birth to current time adjustment. This makes a particle's birth position non-depending of frame rate.

XIII. Texture Sheet Animation rate
Fixed issue with time being inconsistent for particle using Texture Sheet Animation. This resulted in particles skipping frames and faulty rendered frames when minimum to maximum lifetime was set.

XIV. Thread Aggregator sending inactive GameObjects to threads
Fixed issue with Thread Aggregator sending inactive GameObjects to bundled calculation in Automatic mode.

XV. Issues when Playground Settings doesn't exist in project
When opening the Playground Wizard and Playground Settings.asset file wasn't found in the project you should no longer see null reference or division by zero errors. An instance of the settings will be loaded into memory instead.

XVI. Fixed particle stretching issues
The issue with particle stretching on prior to first frame around particle rebirth is now fixed.

XVII.Fixed particle death rotation
A particle would at times get the birth rotation upon death. Use Sync Particle To Main-Thread to ensure correct rebirth updates.

XVIII.Calling UpdateEventParticle with non-existing index
Calling for updating an event particle's data which no longer exists is now fixed. This would happen whenever the particle cache had been resized after the event particle was created.

Version 2.14

Features

I. Added presets
Two presets added (Playground Hot Circle and Playground Projected Circle), these are examples how to create a ring of particles. Along with those two new light point textures.

Fixes

I. Changing Inspector settings on prefab in Project View causes errors
When setting Source Scatter or Lifetime Offset from a prefab in Project View it will no longer try to run their function.

II. Bulk destroying particle systems causes halt
When particle systems are destroyed more than one per frame the Playground Manager will now cope with the updated list of particle systems.

III. Multithreading and Only Lifetime Positioning causing particle systems to halt
If using automatic multithreading and having several particle systems in the scene with Only Lifetime Positioning particle systems should no longer halt.

IV. Fix for automatic multithreading initialization causing unnecessary GC Alloc
First calculations will now be sent to a single thread when using automatic multithreading. This has proven to improve loading times when entering the simulation and minimize any initial particle system halts.

Version 2.13

Fixes

I. Particle Playground asset locations
You should now see a warning message in the Playground Wizard (*Window > Particle Playground*) if the path to the Playground assets is incorrect. Edit the *Settings > Paths > Playground Path* to match your current Particle Playground's location in your project. The pointers for all paths are set in the ScriptableObject Playground Assets/Settings/Playground Settings.

Version 2.12

Features

I. Improved performance
The calculation loop has improved routines for unnecessary iterative cycles per particle, improving calculation times and performance. There's also an overall check if no particles are active then no calculation will be called for, this will improve performance greatly when non-looping or non-emitting systems is active within the scene. Initialization of a particle system is also reworked and has improved loading time.

II. Thread aggregator to bundle calculation calls to same thread
You can now choose how particle systems, skinned meshes and turbulence should calculate over multithreaded calls. Bundling calculation calls improves performance for scenes with a high quantity of particle systems. It also helps in generating less memory garbage. Use the *Playground Manager > Advanced > Multithreading* section to alter how calculation should be distributed. Per default it is set to bundle threaded calculation calls onto available processors, in most cases this is a section you'll never have to worry about unless you have a very specific need for how multithreading should be handled.

Available Particle Thread Methods are:
No Threads: No multithreading will be used (everything will calculate on main-thread unless other thread methods are specified for single particle systems or skinned meshes / turbulence).
One For All: One bundled thread for all particle systems.
One Per System: One thread per particle system (no bundling).
Automatic: Bundle calculation calls onto available processors.

Methods for Skinned Mesh and Turbulence calculation are:
Inside Particle Calculation: Runs in the update loop of selected Particle Thread Method.
One Per System: Creates a new single thread per calculation call each frame.
One For All: Bundles all calculation calls into one single thread each frame.

A single particle system can be put aside from the overall Particle Thread Method for separate handling in your particle system's Advanced > Particle Thread Method. This is to give you the option to choose which particle systems should bundle onto a single thread call or if you'd like to give a particle system extra attention in available threads.

Methods available for a single particle system is:
Inherit: Calculation will follow the overall thread aggregation.
No Threads: No multithreading will be used (the particle system will calculate on main-thread).
One Per System: The particle system will calculate on its own thread.
One For All: All particle systems with this option will calculate onto a bundled thread.

Due to this improvement all Global Manipulators will now calculate over its own bundled thread to ensure stability and performance when using events.
Try out how this impacts performance in the Multithreading Skinned Meshes example scene.

III. Send events to the Playground Manager
The PlaygroundC class now contain functions for receiving events from particle systems.

IV. Inactivate or Destroy on emission cycle completion
You can now choose between inactivating or destroying a non-looping particle system when first cycle finishes.

V. **Clear on emission stop**
Wipe out all simulated particles when emission is set to false.

VI. **Edit particle systems from Project View**
You can now edit your particle system prefab settings in Project View. Any features that needs an active particle system in Scene View will be inactivated.

VII. **Choose to instantiate presets with prefab connection**
By enabling Settings > Prefab Connection in the Playground Wizard your instantiated presets will have their prefab connection intact.

Smashed bugs

I. **Velocity Scale and Turbulence**
Fixed issue with turbulence field not following the overall scale of forces.

II. **Death events on non-looping particles**
Fixed issue with particle events being sent repeatedly after a non rebirthing particle has died.

III. **Non-looping particle systems don't reset on emission change**
When changing emission a non-looping particle system will now reset. This differentiates from looping particle systems which will calculate new particle lifetimes based upon their current.

IV. **Nearest neighbor sort origin local gizmo position**
Sorting helper gizmo will now display at correct position when particle system is simulated in local space.

V. **First born particle and any linear lifetime sorting**
Fixed issue with first particle in cycle not always having correct initial velocity values when lifetime sorting is any linear type.

VI. **Lifetime sorting and nearest neighbor origin change during runtime**
Fixed check if lifetime sorting or nearest neighbor origin would change during runtime (previously only triggered from Editor).

VII. **Nearest neighbor initial sorting when particle count is high**
Fixed issue with nearest neighbor sorting
not calculating correctly upon initialization due to threaded call taking too long to execute.

VIII. **Particle system ID on duplicate**
A particle system will now always get a unique ID upon duplication.

IX. **Improved singleton pattern for Playground Manager**
You should no longer see Playground Manager duplicates in your scene when accidentally instantiating prefabs with a Playground Manager.

X. **Event target calling a disabled GameObject or particle system component**
An event will now make sure the target is active before sending any emission information.

XI. **Born particle receiving new position based on velocity**
A just born particle (with life 0) will now remain in first set position and not receive any additional repositioning. This was a noticeable error in scripted events where first emitted particle would offset position with its calculated velocity.

Version 2.11

Features

I. Overall velocity scaling
You'll find a Velocity Scale slider in the Forces tab to apply scaling of all velocities. Use the ParticleSystemScaler.cs script in the Simple Scripts library to overall scale a particle system (where forces, particle size, lifetime positioning, overflow offset and source scatter will be affected).

II. Functions for procedural Mesh- and Skinned Mesh Targets of Manipulators
You can now call SetMeshTarget(), SetSkinnedMeshTarget(), UpdateMeshTarget() and UpdateSkinnedMeshTarget() on a manipulator.property to create and update target positions for your particles. This gives you an easy way of refreshing any procedural meshes used as mesh targets in your scene. Should your mesh target change vertices each frame you can enable ”Mesh Target is Procedural” from the Inspector.

Smashed bugs

I. Fix for GUI errors when a Manipulator is added from script while viewing Inspector
Issue resolved with the previous method to list unfolding of Manipulator's settings and Manipulator property's GUI language being null. Additionally unfolded values will now be remembered upon re-selection of a particle system's Inspector.

II. Box-shaped Manipulators and rotation
Fixed calculation issue with rotated box-shaped Manipulators.

III. Initial Local Velocity and Velocity Shape Scale
Scale is now connected to the Velocity Shape when applying it to Initial Local Velocity.

Version 2.1

Features

I. Localization (multi-language support) and stored Settings
Playground is now localized to support different languages. You have the ability to create new languages as well as install language packs. You will find it in the new Settings part of the Playground Wizard (home screen) where you also can setup your Playground experience. The Editor Limits are now living in Settings and will be serialized throughout your whole project. This goes for any Inspector specific values such as opened and closed tabs, so you will no longer experience closing tabs between Play- and Edit mode (will not apply to lists).

II. Manipulator Events
Manipulators can now broadcast events, this will let you extend your game logic upon particles confined within a manipulator's shape. Whenever a particle enters, leaves, collides, birth or dies inside the manipulator's shape a call will be made to its corresponding Event Delegate. Every manipulator also has a particle list of all active particles currently inside its shape to easily extract their information. Please see the example scene Manipulator Events for further

details.

III. **Turbulence Manipulator**
You can now simulate turbulence locally within a Property Manipulator of type Turbulence. The same settings are available as within Forces > Turbulence except that this type will also be affected by the Smoothing- and Distance effect. Set Distance Effect to 0 to simulate as the global turbulence.

IV. **Particle filter for Manipulators**
Filter out the amount of particles a manipulator can affect. This is set through a minimum to maximum normalized value (0 - 1), where for instance 0.0 - 0.5 is the first half in the list of particles.

V. **Lifetime Positioning**
The new Lifetime Positioning found in Forces will able particles to move to exact positions during their lifetime using normalized AnimationCurves by X, Y and Z values. The value 0 on the Y-axis on any curve represents the source position origin. This is a good way to create controlled movement sequences where you'll know exactly where a particle will be during its lifetime. Having Lifetime Positioning enabled will annihilate any forces. Please see the example scene Lifetime Positioning for further details.

VI. **Lifetime between two values**
You can now set lifetime randomly between two values. Use the drop-down arrow next to the Lifetime value to toggle between Constant and Random Between Two Values.

VII. **Sorting Layers**
When working with 2D you can now set the Sorting Layers in Rendering > Sorting Layer and Order In Layer. This affects if the particle system will render above or behind objects within the layer(s).

VIII. **Snapshot Material**
Upon loading a snapshot you can now transfer its material. Toggle Advanced and unfold the snapshots's settings through the arrow button to determine whether the material should set before or after any transition.

IX. **Improved particle system loading**
Particle systems will now make use of the multithreading upon start.

X. **Only Size In Range for Manipulators**
Manipulators using the Size property can now be set to only affect particles within the bounds of the Manipulator when leaving the area.

Smashed bugs

I. **Fix for procedural World Objects**
Certain procedural meshes could not process due to an exception in calculated normals.

II. **Disabled PlaygroundParticlesC component**
Having the PlaygroundParticlesC component disabled instead of the GameObject would throw initialization errors for certain setups.

III. **Non-looping burst system to looping snapshot**
When loading a looping snapshot set to burst the simulation will now continue as expected.

IV. Manipulator Color ”Only Color In Range” and transitions
When a Color Property Manipulator is set to Only Color In Range while using a transition the color will now transition back to the particle’s original color.

V. Rotate Towards Direction on rebirth
Wrong rotation when using Rotate Towards Direction upon particle rebirth fixed.

VI. Flash upon Snapshot load
Loading a Snapshot with a different particle count than current could result in a one-frame flash where particles weren’t visible.

VII. Assigning Skinned World Object without mesh causes null reference
A Skinned Mesh Renderer with no mesh assigned will no longer cause null reference, but instead give a notification in the Console.

Version 2.03

Features

I. Reworked particle stretching
When using Render Mode: Stretch you now have more control over the final presentation. Particle’s stretching will now work properly with any manipulator targets. Added features are:
Stretch Speed - *The speed to reach direction.*
Start Stretch - *If no initial velocity is set this will determine the starting direction.*
Lifetime Stretch - *The amount of stretching over a particle’s lifetime set in normalized values where X is lifetime and Y is strength.*

II. Reworked Rotate Towards Direction
Rotating a particle towards any manipulator target is now be available. Keep in mind that stretched particles will zero out any rotation.

III. Lifetime Filter for Manipulators
You can now filter particles within a specified lifetime span.

IV. Script improvements
- Ability to kill a specified particle using *PlaygroundParticlesC.Kill (int)*, where *int* is the id of the particle within the *playgroundCache*.
- *PlaygroundEvenParticle* will now have the variable *particleId* stored upon broadcast.

Smashed bugs

I. Snapshots created before version 2.02 halts simulation
Due to the Particle Mask array not being the correct length the simulation would halt on snapshots made before 2.02.

Version 2.02

Features

I. Wireframes
Choose to render wireframes around particles in Scene View through Playground Manager > Advanced > Wireframes.

II. Improved features for Snapshots
- A toggle between Simple and Advanced settings for Snapshots added.
- Transition time multiplier added to each individual snapshot.
- Transition type added to each individual snapshot.
- Load from script by name instead of int added.
- Name is now connected to the snapshot's GameObject in Hierarchy for easier handling.

III. Particle Mask with fade
You can now mask a chosen amount of particles found in Particle Settings > Particle Mask. Use Mask Time to fade in/out the masked particles. Due to the Particle Playground's caching structure of built-in arrays, this is a good alternative for visually changing the current particle count over time. All particles will still be simulated behind the mask.

Smashed bugs

I. Minor fixes
- Skinned Mesh Targets will now copy over when loading from a snapshot.
- Changing snapshot names is now under Advanced control (fixes serialization issues).
- Affecting rigidbodies in 2D using collision fixed.
- Instantiating a preset in Editor adds double references to the Playground Manager.
- Creating a prefab of a particle system halts calculation due to a stuck isYieldRefreshing until you enter Play Mode.
- Fix for snapshots getting stuck upon load at rare occasions.
- Overwriting an existing preset will now open a dialogue window with call to action.

Version 2.01

Features

I. Multithread support
Calculation is revised to distribute over multiple threads and relieve main thread from most of the Playground's heavier computations. This improves performance for devices with multiple CPUs, where each particle system will calculate on its own thread. Due to the current non thread-safe Physics class, collisions which is dependent on raycasting will still calculate on the main thread.

A wrapper used for manipulator transforms is introduced, called PlaygroundTransformC. This is intended to replace Unity's Transform class to be able to calculate on a second thread. Due to this you need to reassign your transforms to your manipulators.

II. Turbulence
Simplex- and Perlin Noise algorithms have been made available to simulate turbulent forces. Turbulence calculates upon a new thread for performance efficiency. You will find the settings in the "Forces" tab.

III. Event System

A new mechanic called ”Events” for particle systems is introduced. An event is a way of letting one particle system talk to another when a certain action occurs. The event receiver must be set to Source: Script to be able to receive events from the controlling particle system. The receiver will list the controller(s) in its Source tab. You can also subscribe any game logic to an event delegate where you can access all information about the particle triggering the event. Current available events are birth, collision, death and timed.

IV. Snapshots

It’s now possible to save and load the state of a particle system within the scene. In the list of snapshots you can cycle through any predetermined state, where you can choose to load particle data, settings or both. This can also be used to start a particle system at an exact position in time. Once a change occur you can transition from current state to next over time using different smoothing options. Note that if you use a different particle count within a snapshot than in your current particle system a rebuild will be issued.

Every snapshot is stored as a child of your current particle system within the Hiearchy. You can edit its settings afterwards by enabling *Playground Manager > Advanced > Show Snapshots*.

V. Multiple Lifetime Colors as Color Source

Introducing a list of lifetime colors as Color Source where you can set as many gradients as you please. Each particle will get paired with a gradient upon birth.

VI. Vortex Manipulator

A new velocity manipulator is introduced which simulates a swirling vortex. The swirl direction is based on the manipulator’s transform upward axis, changing direction is just a matter of rotating the transform.

VII. Mesh Target Manipulator

Now you can send particles toward mesh vertices within the scene. Skinned meshes are supported as well with the Skinned Mesh Target. The Mesh Target have property abilities where you can choose to either immediately set particles towards their target or transition in form of strict linear or linearly interpolated movement. Use Target Sorting to distribute target positions in a linear or scrambled fashion.

VIII. 2D Collisions

A Collision Type is introduced where you can choose between 2D- and 3D collisions. Collision methods for source mode Paint and Projection is also available.

IX. Collision Plane Offset

You can now determine an offset in world space to a collision plane. This is useful if you for instance want to use the particle system’s transform as a plane.

X. Improved Skinned World Objects with procedural support

Skinned World Objects has been improved greatly in terms of calculation time and the previous garbage it produced. Should you have a mesh that is procedurally generated you can now also determine to update mesh and normals just like with regular World Objects.

XI. Velocity Bending types

You can now switch between Source Position and Particle Delta Position when calculating Velocity Bending. Source Position will calculate the difference between each particle’s positioning and its source positioning. Particle Delta Position will calculate the difference between the particle’s current and previous frame’s position. The difference is then passed in as a normal into Vector3.Reflect to readjust current velocity path.

XII. Enhanced GUI and workflow
It's easier to manage your particle systems now through the improved home-screen of Particle Playground. Presets are divided into "User" and "Examples", all "User" presets will be exported to the final build of the project whereas an "Example" won't. Should you want to use InstantiatePreset("preset name") on an "Example" preset you need to press the button "Convert to User" found in the list presentation mode.

XIII. Pause calculation when outside of camera view
To optimize what's being calculated you can now determine if a particle system should stop when not in view. To enable this feature set `pauseCalculationWhenInvisible` to true.

XIV. Lock particle system position, rotation and scale to specified Vector3
You can now lock a particle system's transform to world or local position, rotation and scale by enabling Advanced > "Lock …".

XV. More advanced presets possible
You can now add presets which doesn't necessarily contain a Particle Playground system on the root object in hierarchy. This makes it possible to have several particle systems within one preset - ready to easily instantiate and be published.

XVI. Render source positions in Scene View
For debug purposes you can now render the distributed source positions for each particle in Scene View. Enable this through *Playground Manager > Advanced > Source Positions*.

Smashed bugs

I. Disappearing particle systems when transform is outside of camera view
The behavior of the Shuriken component stop rendering is mended by forcing Play() when particle system ends up outside of the camera's frustrum. To disable this feature set `forceVisibilityWhenOutOfFrustrum` to false. You will still experience this behavior in Editor when not in play mode.

II. Revised Clone- and CopyTo functions
All Clone- and CopyTo functions should now be reliable on passing data to other objects.

Version 1.20

Features

I. Improved procedural mesh support for World Objects
A reworked model of reading procedural meshes where you can choose if a World Object will update its vertices and/or normals during runtime. Non-procedural will also generate less garbage thanks to this method.

II. Better control over what happens on particle rebirth
Through *Advanced > Rebirth Options* you can now determine if random size, random rotation and random scatter should occur upon particle rebirth.

III. Flowerbed preset added
An example of how to work with a static texture sheet to get different textures per particle. Try changing the mode to projection, add a readable texture and project onto a surface!

IV. Simple Script library started
A simple script library is started for both C# and JavaScript. The intention is to get you going faster and have a much better overview of how to interact with a particle system.

Smashed bugs

I. One-frame flash on prefab instantiation
No more one-frame flashes during pooling.

II. Enable/disable GameObject sometimes results in flickering
Particle time is now set inside OnEnable, which will make the initial time correct when enabling a particle playground system and remove the catchup behavior.

III. Emit (true) sometimes results in flickering
Particle time with rest emission is now set inside the overload Emit (bool) when you want to enable emission through script. Upcoming particle rebirths are also set with care for time of enabling, to not loose the predetermined emission pattern.

IV. Target Manipulator fixes
- Could throw null reference upon item removal.
- Lerp and Linear transitions will now work as expected while in local space and ”Only Position In Range” is deselected.

V. Code optimizations
- Meshes created several unnecessary clone instances generating a lot of garbage.
- Reworked World Object updates to not ask for vertices unless needed.
- Moved pool from struct to class. The pool will not generate garbage anymore.
- Chain of functions cleaned up addressing unnecessary repeating initializations.

Huge thanks to movra on the Unity Forums for feedback regarding C# and performance!

Version 1.19

Smashed bugs

I. Minor C# fixes
A couple of cleaned up items lost in translation. These didn’t affect working in the Unity Editor or being able to build the project, but give debug errors and warnings in MonoDevelop.

Version 1.18

Features

I. C# support
Particle Playground now supports JavaScript (UnityScript) and C#. Both languages are distributed within the package where any language specific files are separated into either ”JavaScript”- or ”Csharp” folders. You will see an updated menu in Window > Particle Playground > Playground Wizard (language), which determines the language you prefer to

work with.

If you're upgrading from a previous Particle Playground version it's recommended to remove the previous installation in "Assets/Particle Playground/" first to minimize asset locations in wrong folders. This is due to a new scripts-, presets and brushes structure separating the languages. Remember to store any presets, brushes or other created/modified files within the Particle Playground folder first.

To remove a language simply remove the language specific folders within the project.

Smashed bugs

I. **State offset is wrongly calculated on initiation**
State offset is now appearing the same during particle build and simulation.

II. **Keeping lifetime color alphas broken for Property Manipulator: Color**
Property Manipulator: Color will now remember lifetime alphas when deselecting "Only Color In Range" while using "Keep Color Alphas" during a particle's lifetime cycle.

III. **Property Manipulator: Target won't apply local simulation space positions**
Property Manipulator: Target will now calculate particles simulated in local space correctly.

IV. **Null reference fixes**
- Particles will no longer try to calculate manipulators with a null transform.
- States will no longer try to calculate with a null state transform, the transform of the particle system will be used instead.

V. **Undo of certain settings needs manual pool rebuild**
Particle pool will now rebuild automatically upon undo and unfreeze the particle calculation.

Version 1.17

Features

I. Source Projection
A new source is available. Projection lets you project a texture onto colliders in the scene. Use this to for instance project dust and rain splashes on the ground or gunshot hits on walls - it's really open for interpretation. The projector can do live updates each frame and be called to update when you need to refresh the projection. Through the Surface Offset you can offset from the projected surface with the hit normal direction into account.

II. Initial Velocity Shape
Adding ability to shape your own initial velocity by Vector3 Animation Curves. The shape applies to an emitted particle's force where X is total amount of source positions and Y is multiplier for total initial velocity. Use this to create shapes in how your particles spread out in the scene.

For instance, try a transform with overflow offset to see the basics of how this distributes velocities to each particle's initial velocity at birth with respect to source positions.

III. Custom Lifetime Sorting
Now you can determine your own lifetime sorting by a curve where X 1.0 is total amount of

particles and Y 1.0 is total lifetime.

As an example, to replicate some of the existing behaviors:

1. X1Y1, X0Y0: Linear
2. X0Y0, X1Y1: Reversed
3. X1Y1, X1Y1: Burst

IV. New Manipulator: Combined

The Combined manipulator is a placeholder for adding several manipulator properties into one manipulator call. This is an optimization if you want the same bounding space for several properties. For the sake of usability, all other main manipulator types are available through properties now as well (such as Gravitational and Repellent), you can see this as giving a particle the property of these main manipulator types.

V. Property Manipulators added

Death - Force a sooner death for a particle.
Lifetime Color - Give a particle new lifetime colors (gradient).
Attractor - Attracts particles with a funnel type feature. (Same as main type)
Gravitational - Attracts particles with gravitational features. (Same as main type)
Repellent - Repel particles. (Same as main type)

VI. Local Simulation Space

You can now simulate particles in local space. All global values (manipulators, velocities, collisions etc.) are taken into account for local space from the particle system's transform. Available in *Advanced > Simulation Space*.

VII. Source Scattering

Ability to spread source positions within minimum- and maximum range. Use this if you want to efface any previous vertex or pixel structure which your source provides.

VIII. Source Down Resolution for skinned meshes

You can now determine the source point distribution over the skinned mesh vertices. Use this to scale down the amount of source positions needed to cover the complete mesh. Do note that the extraction of vertices is still the same, but this will allow having fewer source positions, and in return fewer needed calculated particles. Make sure to use this if you target mobile platforms.

Smashed bugs

I. **When using scripted emission and Disable On Done you need to manually re-enable**
Emitting a particle will now enable the particle system as expected when using Disable On Done.

Version 1.16

Features

I. New Property Manipulator: Target

Ability to split and target particles towards transforms in the scene. Every particle affected will

pair up with the manipulator as key, so if the particle interfere with a new target manipulator's space it will get a new pairing key. This gives a particle the ability to wander from node to node. Use linear interpolation- or strict linear transition to move them over time (use no transition to teleport).

II. **Linear transition added to Property Manipulators**
You can now change properties over time in a strict linear fashion without smoothing towards the end.

III. **Inverse bounds of Manipulators**
Inverse the affected area for the manipulator properties. This opens up for a much broader usability, just think outside the box and you'll be able to make use of this feature. The Manipulator will be marked with inverse colors in Scene View when using inverse bounds.

IV. **Force axis constraints added**
Forces can now be constrained on one or several world axises. This is especially good for particles living in 2D. You will find X, Y and Z toggles in the bottom of the Forces tab.

V. **Improved randomization of respawned particles**
Particles will now appear more different when they respawn in their previous PlaygroundCache. This regards positions, velocities and rotations.

VI. **Circle Shot preset added**
An extensive example of emitting particles in a circle from script is added. You can customize its Circle Shot behavior through its attached script (particles, rotation, cycles, yields and what will happen to it when emission is done).

VII. **Editor limits made accessible through Playground Manager**
Previously hidden settings for limits of all controls is now lifted into Playground Manager > Advanced > Editor Limits.

VIII. **Add plane colliders through the Playground wrapper**
You can now add plane colliders to a particle system using the function `Playground.AddCollider(particleSystem, transform)`. This returns a `PlaygroundCollider` object.

IX. **Particle icon gizmo in Scene View**
A Particle Playground System is now marked with a P in the Scene View.

Smashed bugs

I. **Issues with Script Mode**
Each emitted particle will now remember their own emitted colors during their lifetime and not inherit from the last created particle. All velocities will now apply without having to manually issue Update() on the particle system in between loops the same frame.

II. **Playground hindering values to store in the PlayMaker FSM entering Play Mode**
An issue that stemmed from the Unity Editor's update delegate made PlayMaker not save its values when entering Play Mode. Moving the Playground's Editor update method onto MonoBehavior has mended the issue and you should now be able to work with PlayMaker and Particle Playground in the same project.

Due to this update you need to select a particle system to run its calculation when not in Play Mode. This is also a great optimization if you have many particle systems within the scene.

III. **Editor updating particle systems at half speed when returning from Play Mode**
A related issue to having the Playground's Editor update method inside the update delegate made the value Playground.globalTime update at half its speed. You can now expect a particle system to run equally in the Editor regardless of being in Play Mode or not.

IV. **Returning to the Unity Editor after some time makes particle systems flicker**
Due to the Playground.globalTime the particle systems need to iterate through all lifetimes that should have been played whilst not having the Unity Editor active. Returning to the Editor will now issue a reset and calculate times from when you returned instead of when you left, resulting in a non-flickering and smooth appearance.

V. **Random initial refresh in Editor causes recalculation of particle distribution**
The Editor will no longer randomly refresh a particle system during its first cycles. This was due to a behavior caused by the previously used Editor update delegate.

VI. **Clearing out and rebuilding a particle system's pool causes issues**
Clearing out a particle system's particle pool will no longer cause null reference when resetting time. Rebuild will now take lifetime offset into account.

Version 1.15.3

Features

I. Support for particle stretching, mesh particles and horizontal/vertical billboard
Particle Playground will now run together with all the different types of particle render techniques available in Shuriken. Texture Sheet Animation compatibility is also introduced, but due to the restricted access level you need to enable and edit this directly in the Shuriken particle system. Enable the visibility of the Shuriken component in *Playground Manager > Advanced > Show Shuriken*.

II. Script Source Mode made more user friendly
Reworked Source Mode: Script to lift the core features and make them more accessible. Use `PlaygroundParticles.Emit()` to emit a particle, pass in `position(Vector3)`, `velocity(Vector3)`, `color(Color)` and `parent(Transform)` to set the source data of how this particle should behave. The framework will take care of the rest in the calculation loop.

You can see this mode as creating your own source behavior, where the framework will give you a ready to use particle pool with forces-, collision and manipulator features available (Overflow-, Lifetime Sorting features and Delta Movement is not available as they are not applicable behaviors). You just need to tell it when to emit and pass in a couple of variables,

Example:
```
playgroundParticleSystem.Emit(position, velocity, color, parent);
```

III. Vertex Lit Color shader
A new single colored shader of type vertex lit is introduced. This can for instance be used to give particles a single-colored voxel-like look (especially with meshes) - or to let light affect each particle pixel in a state.

IV. Scale of minimum- and maximum size
It's now possible to scale the minimum- and maximum size values without rebuilding the size

array. This gives you an extra control above the lifetime size curve and the particle size range.

V. **Loop and automatic disabling when done**
Added possibility to control if the particle system's emission should loop or turn silent after first lifetime cycle. The loop takes all lifetime sortings into account. Use "Disable On Done" to turn the GameObject inactive when the loop is exceeded.

Version 1.14.2

Features

I. **Improved collisions**

A. The particle collisions are reworked. Collisions will no longer suffer from leaking through colliders at lower frame rates while being much more efficient in their collision checking.

B. Introducing possibility to add infinite plane colliders, great if you want to keep your simulation into a determined space.

C. Added a random bounce method which offsets randomly within minimum and maximum Vector3-values from the collision surface's normal. This is good if you for instance want to simulate an uneven surface.

D. Added Lifetime Loss on particle collision represented by a normalized value of remaining lifetime span. For instance, a lifetime loss of 0.5 on a particle in 50% of its lifetime will set it to 75% of its lifetime.

II. **Local- and Global manipulators**
Introducing Local (particle system-bound) and Global (manager-bound) manipulators. The previous manipulator functionality was only seen as global, affecting all particle systems within range and layer. Being able to set a local manipulator to a particle system makes it serializable within a prefab. This gives you opportunity to create more advanced structures of presets where manipulators don't need to rely on an existing Playground Manager.

III. **Max Velocity**
Ability to clamp particles velocity magnitude in the Forces settings.

IV. **Toon Star preset added**
A star with a toon look added to the preset collection. Along with it a spark material.

Smashed bugs

I. **Additive Velocity Property Manipulator with Local Rotation is frame dependent**
The property manipulator type Additive Velocity is no longer frame dependent when using the setting Local Rotation.

II. **Particle systems set to not emit on start will run one cycle upon simulation**
The particle systems will now stay nice and quiet until you tell them to emit. Along with that you can now inactivate the GameObject to make sure that the particle system won't be updated

through the Manager.

Version 1.13

Features

I. **Publish your own presets**
Introducing a standard for publishing your own created presets to the Unity Asset Store. Use the Preset Wizard (found in Playground Wizard > Presets > Create) and the tab Publish to get started. Please see the Publish Guide at http://playground.polyfied.com/ for additional info.

II. **Preset filter search in Playground Wizard**
You can now filter search your Particle Playground presets in the Playground Wizard. The filter makes use of the prefab name, so keep in mind to meta name your presets.

III. **Lifetime Offset - time particle system sequences with each other**
Added an offset ability to the lifetime sorting. This can be used to set particle systems in sequences to each other. For instance, using the Playground Runway preset you can now determine if two (or more) runways should be similarly synced or offset in their blinking lights. You can also use this to annihilate any fade-ins at first particle cycle by setting negative values. This can for instance be useful in a situation where you want clouds similar to the Cloud preset to be fully visible from first frame in the first particle cycle.

IV. **Playground Bouncing Coin**
A preset with a colliding coin added. Perfect for those jackpot moments.

Smashed bugs

I. **Particle systems inside prefabs doesn't run**
Instantiating a prefab (at any point) that has a Particle Playground System in its hierarchy will now initiate and attach itself onto the Playground Manager as expected.

II. **Lifetime Offset: Nearest Neighbor and a single point source makes emission stop**
Now when using a Lifetime Offset of type nearest neighbor (and reversed) from a single point source (such as a transform with no Overflow Offset) then Lifetime Offset will now be calculated as zero. What happens is that all source points are equal which will make every particle fire at once (similar to burst mode).

Version 1.12.5

Features

I. **Manipulator type Property**
Introducing a new manipulator that can change the properties of a single particle.

Properties available:
None - The flag 'changedByProperty' in the particle pool will be set to true.
Color - Set particle to chosen color.
Velocity - Set particle to chosen velocity. Local Rotation creates a direction from Transform.
Additive Velocity - Add velocity to particle.
Size - Set particle to chosen size.

Use **Transition: Lerp** to change the property over time. Strength will determine how much exposure within the manipulator radius is needed to fully reach the new property.

II. **Manipulator shape Box**
Introducing a box shape to manipulators, set size by editing the bounds of the bounding box. Represented in Scene View with a similar layout as box colliders have.

III. **Manipulator positioning**
Ability to move manipulators in their local- or global axis. Rotation and scale handles are also available. Handles are shown when the Manipulators section is unfolded in Inspector and is connected to the Scene Tool selected.

IV. **Overflow mode Source point**
A new overflow mode is introduced where overflowing from the source position with its normal direction is possible. Available for all sources.

V. **Playground Laser Preset**
Additions to the preset inventory has been made, introducing a "Playground Laser". This preset has a script attached which is running alongside the PlaygroundParticles-script. This gives it additional properties of setting the overflow offset towards any hit collider in forward direction.

VI. **Playground Cloud Preset**
A vertex lit particle in form of a cloud as been introduced as a preset. Try moving around a light source to see the emission in action (and alter the Emissive Color of the material to get different results).

VII. **Playground Fire Preset**
A fire preset added, perfect for torches.

VIII. **Playground Soap Machine Preset**
A bubble preset added, perfect for underwater themes. This makes use of the Velocity Bending to sway the particle's direction.

IX. **Playground Valentine Preset**
A heart preset added. This illustrates the new 'Rotate Towards Direction' ability.

X. **More materials and textures**
The particle library is extended with different shapes and objects.

XI. **Initial Rotation**
Ability to set the initial rotation for particles apart from rotation speed.

XII. **Rotate particle towards direction**
Added the ability to rotate a particle towards its direction. Use the Rotation Normal to determine which vector to rotate around. The normal is setup in world space so if you for instance always want to rotate from the camera perspective you could apply the camera forward to 'rotationNormal'. You can use the Initial Rotation value to offset the rotation.

XIII. **Example scene Thunder**
A new example scene showing thunders in the clouds using a property manipulator.

Smashed bugs

I. **Instantiating state presets with texture and no Manager in scene throws errors**
When instantiating a preset built from a texture and Playground Manager isn't available in the scene the non-existing (at that frame) Playground.reference will no longer throw an error. This was due to trying to access the Playground reference and read the value of buildZeroAlphaPixels (which now by default is false if no Playground Manager is instantiated).

II. **States created from textures sometimes being frozen upon creation**
When creating a state from a texture the state should now always initialize in a correct order to display immediately.

III. **Paint normals don't follow surface rotations**
When rotating an object with painted source positions the normal direction will now follow.

IV. **Skinned mesh normals don't follow animations**
When rotating/moving bones of a skinned mesh the normal direction will now follow.

Version 1.12.4

Smashed bugs

I. **Preset Instantiation from code doesn't run in Manager**
When instantiating a preset from code the preset is now correctly attached to the Playground Manager. If no Playground Manager is present it will instantiate upon creation.

II. **Burst mode makes particle system wait one lifetime cycle before initiation**
When using Lifetime Sorting: Burst the time for all particles is now correctly calculated making all particles burst from initiation.

III. **Changing particle count right after previous frame causes "Array Index out of range"**
Changing the particle count at any time should no longer give "Array Index out of range"-error. This was particularly an issue when changing particle count from OnGUI.

IV. **Update delegate not always get attached to the Editor update routine**
When instantiating presets the Playground Manager and the Editor update delegate should now initiate correctly.

V. **Title in Playground Manager is cut short on Windows platform**
Title is now at its full length in the Inspector on Windows platform (in all glory…).

VI. **State not building data when duplicating in Hierarchy**
States will now duplicate correctly when the particle system is duplicated in hierarchy.

VII. **Class Brush changed to PlaygroundBrush**
To not intervene with other plugins using a namespace called Brush the Brush class is now called PlaygroundBrush.

Version 1.1

Features

I. **Source Paint**
A new Source is available which lets you paint emission positions live in the scene with help of brushes (painting live in Play mode is also available through helper-functions).

Each paint position has information of its parent, initial position and current normal it lives upon. What this practically does is to attach each paint position onto an exact local position of each object. So if the object moves or rotates, the emission will stay put to the origin position of the object. The normal will let you send out particles with an Initial Local Velocity from the parent's mesh normals. All of this is automatically calculated within the Particle Playground Update loop.

Painting live from script is as easy as calling:

```
Playground.Paint(
     particles : PlaygroundParticles,
     position : Vector3,
     normal : Vector3,
     parent : Transform,
     color : Color32,
);
```

For further information please see the Manual and the example scene *Paint* - where raycasting is used to paint live into the scene at runtime.

II. **Brush Presets**
Create your own brushes to paint with. These can be made in form of prefabs and stored in *Resources/Brushes*. They're represented in the Source Paint as buttons but you're also able to create them during runtime and assign them onto a PaintObject. For convenience a Brush Wizard will appear when you choose to create a new brush from the brush preset list.

III. **The Playground Wizard**
The Playground Wizard (*Window/Playground Wizard*) is now Particle Playground's home screen. From here you can create new Particle Playground Systems, both factory-made and customized by you. This is replacing the previous Editor Window for Particle Systems, which has discontinued support due to incompatibility of some of the new and upcoming features.

IV. **Particle Playground System Presets**
Ability to create your own Particle Playground System Presets. These are made in form of prefabs and kept in *Resources/Presets*, which you can instantiate from code or from the Playground Wizard. For convenience a Preset Wizard will appear when you choose to create a new preset from the preset list. This enables you to create pre-made Particle Playground Systems and easily recreate them during runtime.

Instantiating a preset can be done like this:

```
Playground.InstantiatePreset("Name");
```

InstantiatePreset() returns a PlaygroundParticles object.

V. **Edit Manipulators from Scene View**
Ability to visually edit Manipulators from Scene View while Playground Manager or a Particle Playground System is selected in Hierarchy and desired Manipulator is unfolded in Manipulators list. Position is represented by a position handle, Size is represented with an intractable omni and Strength is represented by a vertical slider-handle.

VI. Improved workflow
Overviewing and editing the settings for a Particle Playground System and the Playground Manager is easier now with a new layout. The toolset is also more semantically structured, contains better automatized features and is exposed for a broader user control. For instance, now you can create new Particle Playground Systems directly from the Playground Manager, list the available Particle Playground Systems, from there - sort and duplicate them. The same features are also available for States and Manipulators.

VII. Lifetime Sorting Nearest Neighbor
The new lifetime sorting Nearest Neighbor and Nearest Neighbor Reversed will sort emission positions with comparison to distance of the selected Sort Origin. Imagine a water-ripple effect on any type of Source.

VIII. Control particle bursts with Emission Rate
Emission Rate lets you control the percentage of flow in burst sequences.

IX. Assign transform as parent to state
Possibility to assign a transform in the scene to a state. This gives you the ability to move, rotate and scale a state. Initial Local Velocity and Delta Movement is also available now for states that has a transform assigned.

X. Overflow Mode
Set which method to offset the remaining particles when calculating the overflow of particle count towards source positions. Available methods are Source Transform (transform point), Particle System Transform (transform point) and World (global). Using a transform will make you able to rotate and scale the Overflow Offset.

XI. Color Source
Choose which type of method to colorize your particles with Color Source (found in Rendering). Using a state with a texture or painted positions from a brush with a texture is an example of a Color Source. If no source is used a fallback to Lifetime Color will occur. You also have the possibility to only set alpha from Lifetime Color while the colors are picked up from the source positions.

XII. ”Only Source Positions”-settings
Overrides all velocities and set every particle towards their source position every Update-cycle. See the presets ”Matrix Cube” and ”Holobot” for basic example of usage.

XIII. Initial Velocity Ranges
Possibility to set minimum- and maximum values for Initial Velocity and Initial Local Velocity. This enables spreading particles from their source emission position within a set range.

XIV. Velocity Bending
Possibility to bend a particle’s velocity using the reflected value of current velocity multiplied with bending. The direction from each particle’s source position towards their current is seen as the normal plane. Each particle’s current velocity path is altered with the bended direction. You can use this to create interesting movement patterns without having to use Manipulators

(combine them to create really interesting behavior).

XV. Added the function GetParticles to Playground wrapper
Calling `Playground.GetParticles(int)` will return the PlaygroundParticles at list position.

XVI. Kickstart
When creating a new Particle Playground System the preset is set to easier settings to get started.

Smashed bugs

I. Lifetime size is zero when Particle Playground System Prefab is reset
Leading to invisible particles emitting. Now when instantiating a new Particle Playground System a standard lifetimeSize AnimationCurve with two initial keyframes of value 1 is set.

II. Assigning GameObject without mesh component as World Object throws null reference
Calculation of a World Object's mesh that doesn't have a mesh assigned returns instead of executing mesh calculations leading to null reference.

III. Removed switching source mode if source is null
Previously a feature which is no longer needed. This behavior caused Particle Playground Systems to jump through their sources on initiation (making users have to set source when scripting).

IV. Script no longer default Source mode
When instantiating a Particle Playground System the default Source was set to Script. Now the default Source is set to Transform with the Particle Playground System's own transform assigned. This creates an easier starting point for users and will make the particle system emit upon creation.

V. Selecting Particle System Prefab assigned to Playground Manager
Fixed the faulty behavior of assigning Particle System prefabs in Project View to the Playground Manager upon selection.

VI. Null reference on missing Manipulator transform
When destroying the transform for a Manipulator the Console no longer throws null reference messages.

VII. Inspector thinks object is in Project instead of Hierarchy when Prefab
When selecting a Particle Playground System in Hierarchy that is set to a prefab the Inspector no longer gives a "Please edit this from Hierarchy only"-message.

VIII. Duplicating Playground Manager leads to undesired behavior
When duplicating the Playground Manager the previous Playground Manager is turned into a zombie. Only one Playground Manager is allowed within a scene, trying to initialize a second manager will now destroy the later copy and throw a message in the Console.

IX. Setting lifetime to zero makes particles disappear
Not semantically faulty but breaks the behavior being able to have "sticky" particles. Now setting the lifetime to zero will make the particles behave like they are static objects (but follow their source position).

X. Scene Undo makes particles disappear sometimes
Particles now correctly refresh on undo.

Version 1.0.2

Features

I. **Added procedural mesh support to World Objects**
World Objects now handle meshes that updates during particle calculation. Each particle origin position will update towards the updated vertices positions. Along with the procedural support a checkbox named ”Update Mesh Normals” will appear when selecting WorldObject as Source, which will enable updating the directions of the normals as well. You can also call `Playground.GetNormals(Vector3[], WorldObject)` to update the normals on a World Object.

II. **Added the function GetManipulator to Playground wrapper**
Calling `Playground.GetManipulator(int)` will return the `ManipulatorObject` at list position. If the integer passed is larger than the Manipulator list count it will return the Manipulator at repeated remainder position. It’s a bit easier to work with rather than using the Playground reference (`Playground.reference.manipulators[int]`).

Smashed bugs

I. Adding non-readable Textures and Meshes to a State throws console errors
Now checking the Texture’s and Mesh’s AssetImporter’s variable isReadable when adding a State from Particle Playground Editor Window or Custom Inspector.

Version 1.0.1

Initial release version.

Support

For questions, requests and bug-reporting please send a mail to support@polyfied.com.
Please visit http://playground.polyfied.com/ for more information.

Particle Playground 3 is proudly presented by **Polyfied**,
Stockholm Sweden 2015.

polyfied.com