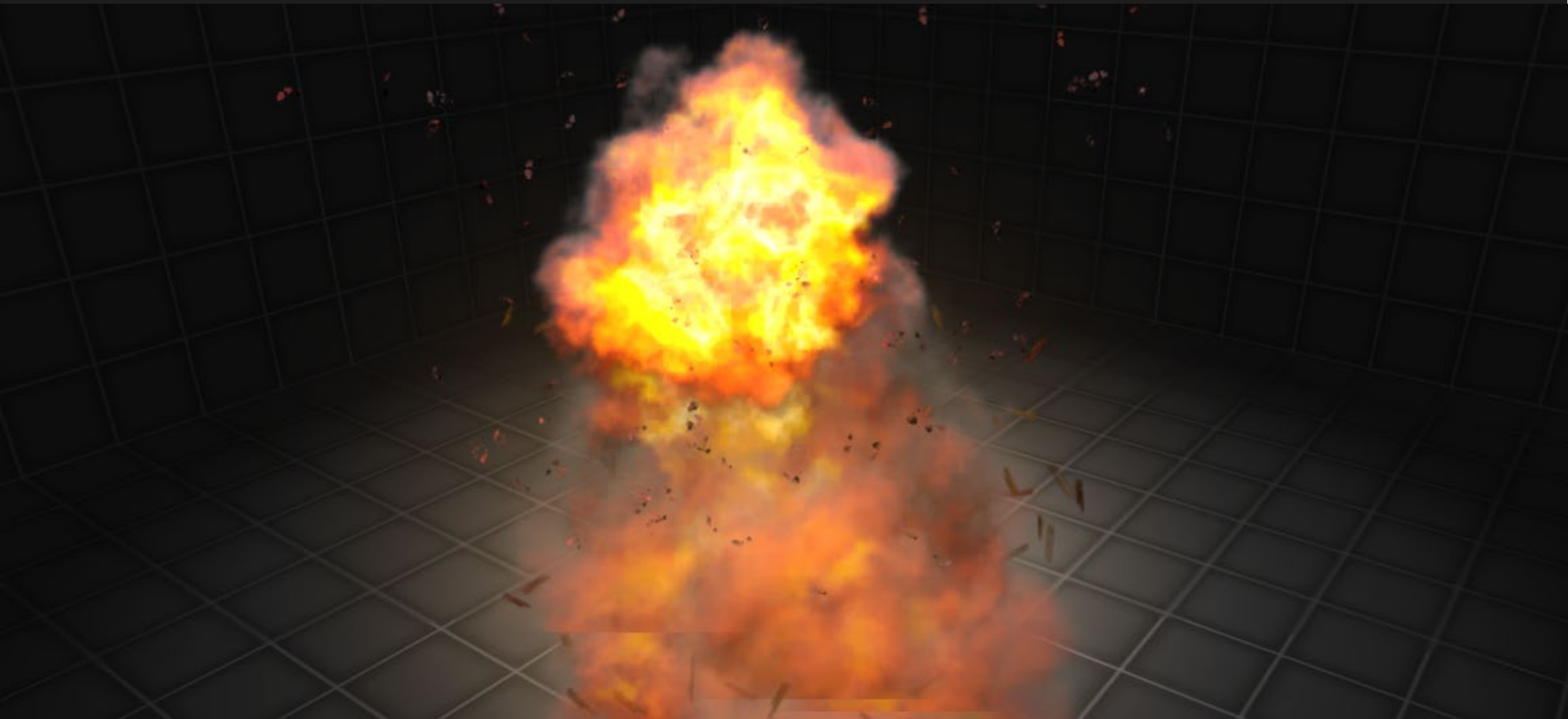
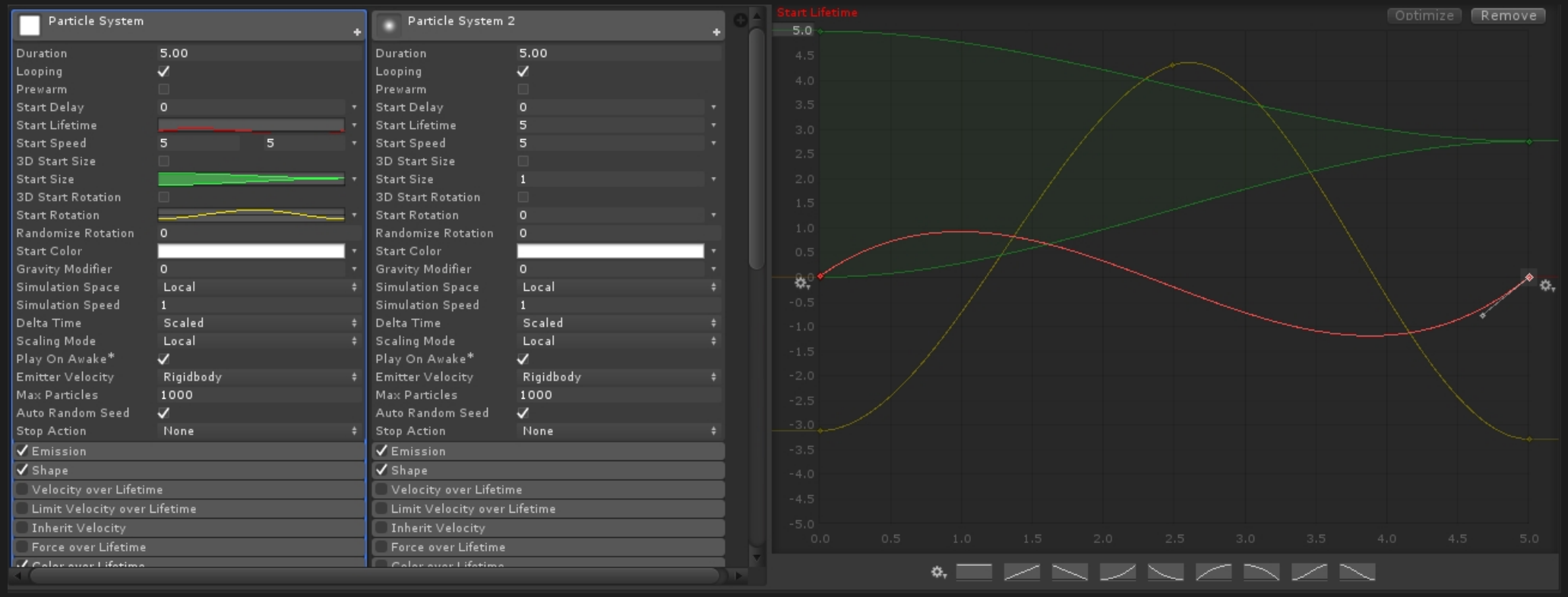
Particles



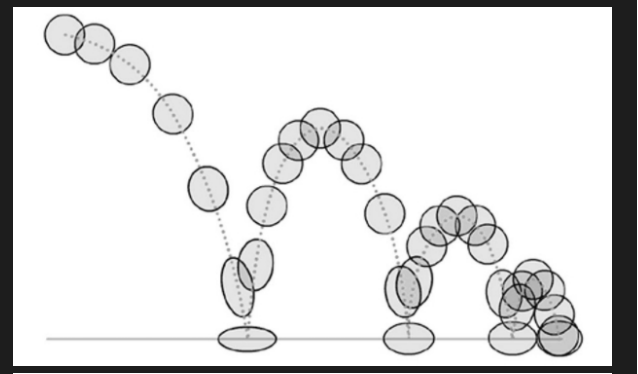
Particle System

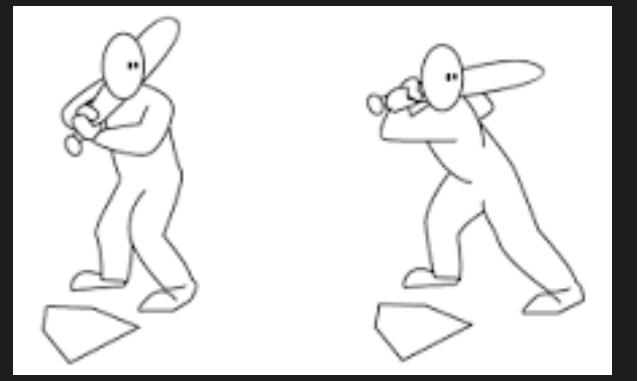
- Used to solve visual problems not solvable in a “standard” way (models/textures, etc)
- Composed by Emitter & N Parameters
  - Particles as 2D textures or 3D obj, emission rate, duration, velocity, color/size/opacity over time, bouncing off colliders, child particles emission
- ParticleSystem (PS) vs ParticleEffect
- In Unity, the Particle Editor allows to edit one Particle effect, composed by one or more PS

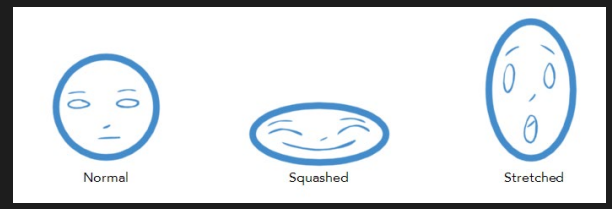


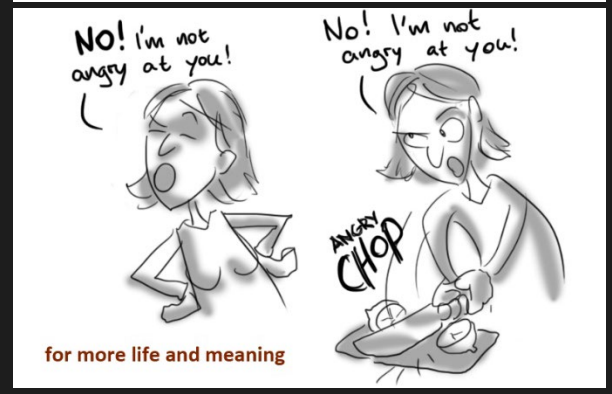
Particle System Artist

- Understanding of
  - Rendering visual results
  - Animation principles
    - Anticipation, Secondary action What happens before a detonation
    - Exaggeration, Squash&Stretch Push the Fx further than you think
    - Timing Increases emotional intensity
- Ability to view visual reference of an Fx > Deconstruct it > Re-create it in a real-time game engine
- Acts as a bridge between the Artists and Programmers working on a game
  - Art assets must be easily integrated into the game without sacrificing either the artistic vision or exceeding the technical limits of the chosen platform
- Strong knowledge of environmental effects, cloth simulations, lighting and dynamics
- Knowledge of Photoshop, Maya, Adobe After Effects and 3D Studio Max
- Experience in creating real time shaders and post effects
- Usually works at the end of the pipeline: get the most in less time





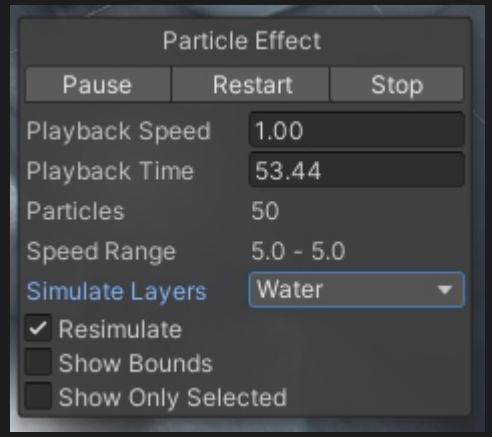




ParticleEffect Scene panel

- To create a PSystem: RClick/Effect/ParticleSystem
- Select it: in the Scene view appears a panel

- Psystem Component
- PEffect One or more Psystem, usally organized as a parent child relationship
- SimulateLayers allows to preview Psystems at the same time, even if they are not in a parenting relationship



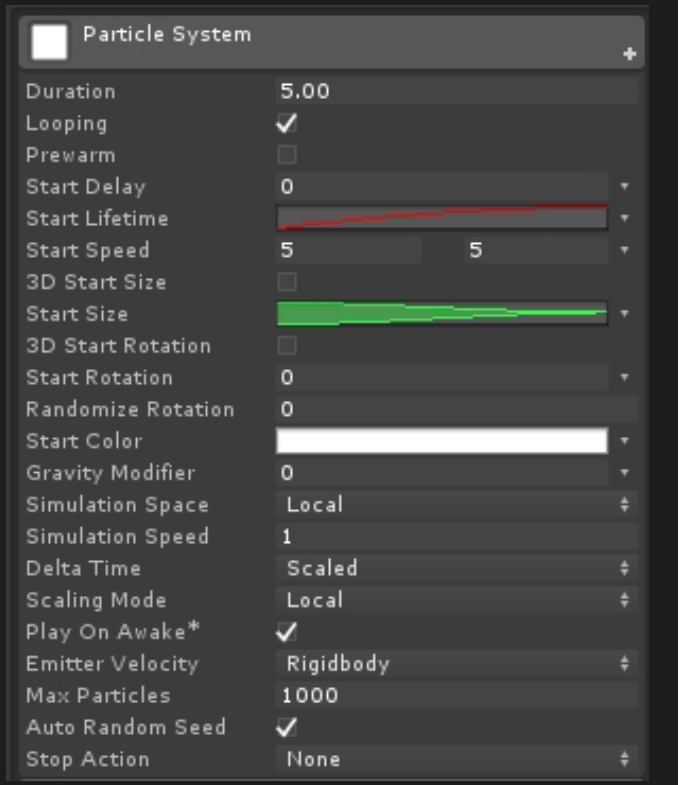
[Particles_05]

Main

Start vs OverLifetime values. If they are based on gradient/Curves, normalized time is

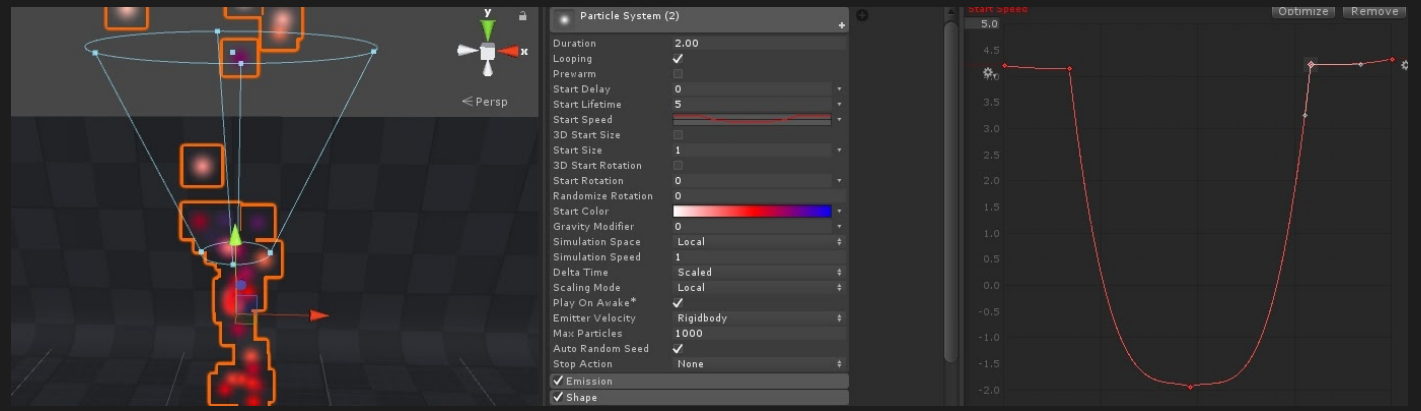
- Start values (SystemAge % Duration) / Duration
- OverLifeTime values ParticleAge / Particle lifetime
  - Multiplied by the start value. This applies for scalar and colors. Eg:
    - Size(t) = StartSize * SizeOverLifeTime (t)
    - If StartSize is 0, even if SizeOverlifetime goes from 0 to 1, size will always be 0
  - StartSpeed isn't Start time for Velocity

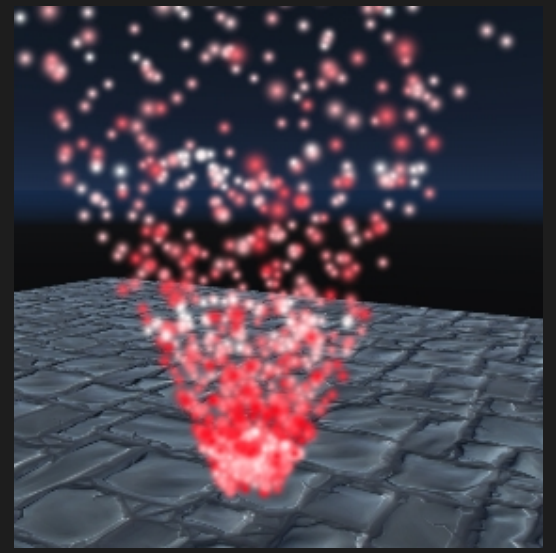
- Duration (Set looping OFF to see it) Time before the start of the next loop
- Prewarm
- SimulationSpace Local/World (Move Psystem on XZ plane to see the effect)
  - Custom: Add empty obj reference and move/Rotate it
- ScalingMode
  - Local if you scale parent, no changes (Psystem should stay at 0,0,0)
  - Hierarchy Both Particles and Shape is scaled
  - Shape Only Shape is scaled
  - Officially supported only uniform scale values
- StartColor Gradient
  - It is not a Color overlifetime, it is the StartColor value! We use this color: (SystemAge%Duration)/Duration
  - Good rule Use a loopable color gradient
  - Particle/Additive Shader R+G+B = White
- EmitterVelocity Choose how the Particle System calculates the velocity used by the Inherit Velocity and Emission modules. The system can calculate the velocity using a Rigidbody component, if one exists, or by tracking the movement of the Transform component
- StopAction The OnParticleSystemStopped callback is sent to any scripts attached to the GameObject



Main

- StartSpeed
  - Same as color: it is not a Speed over lifetime. If we use Startcolor gradient Red as central color, and 0 Value as central StartSpeed, we'll see only Red particles on the floor

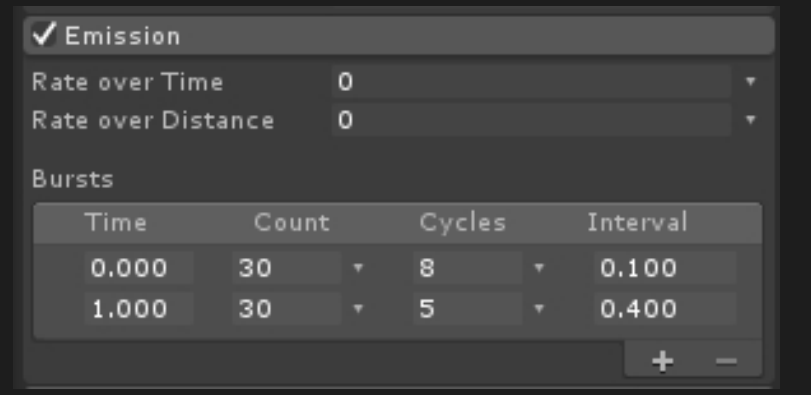




Emission

- Time & Distant emission (can be used together)
  - RateOverDistance How many emitted particles per-unit if they move. A value of 100 means that if I move 1 unit, I'll emit 100 particles. Hence, if 100 is the Max particles limit, I'll have to wait for the DurationTime until next loop begins
    - Try it
      - Activate Emission_RateOverDistance
      - Particles are going Up because of the StartSpeed
  - RateOverTime How many particles per-second are emitted

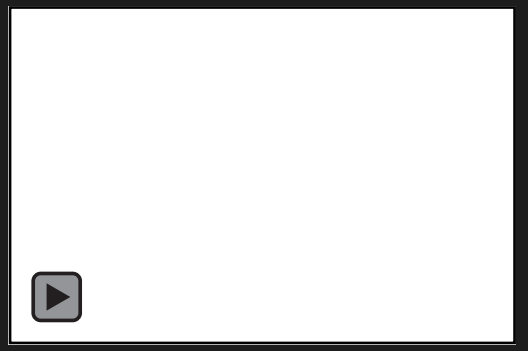
- Burts Simulate a sudden particles emission
  - Cycles/Interval
  - Up to 8 Burts
  - Even if RateOverDistance/Time is 0, particles are emitted



Shape

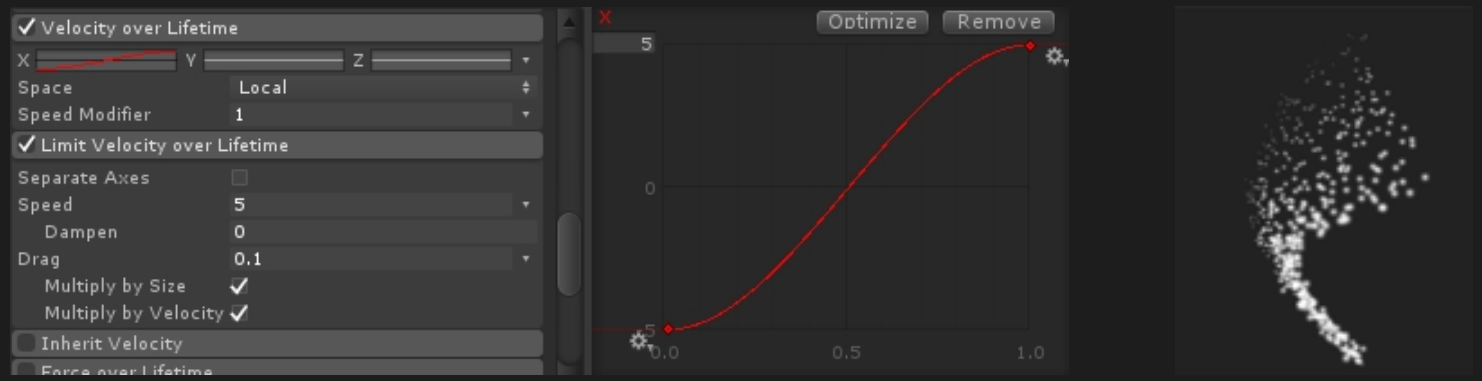
- 2D/ 3D Shapes / Meshes
- Spread How many points particles are emitted
  - 0: Burst Count particles are evenly distributed around the circle
  - 1: Burst Count particles are emitted only from 1 point
- Sequencer (Cone, Donut, Circle, Edge, SkinnedMeshRenderer)
  - Random/Loop/PingPong
- AlignToDirection
- SpherizeDirection (Box / Volume)
- Texture
  - Try to use a texture here (read/writeEnable!)
    - You can set a channel threshold: Under that value, particles are not emitted
    - particles can sample from the texture the color to use

- Mesh
  - Import 3D_Text_Mesh_Emitter.fbx and use this mesh as emitter Shape
  - SingleMaterial Only emit from a specific material of the mesh
  - You can use RandomizePosition together with mesh emitter to create a “particle reveal” mesh animation



Velocity & Force OverTime

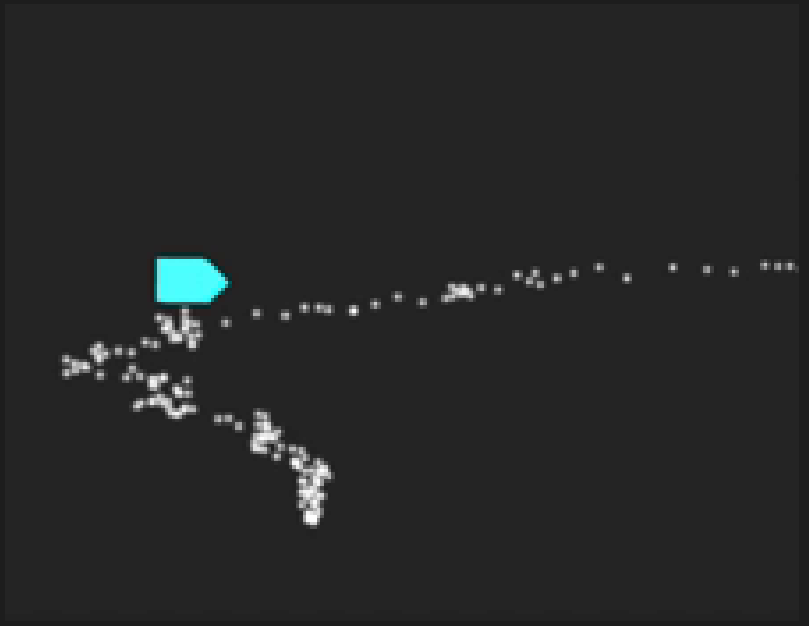
- VelocityOverLifetime We are giving the particles a vel of -5 on the x axis and then of +5 on the same axis during their lifetime: the result is a “C” shape.
- LimitVelocityOverLifetime
  - Drag based on
    - Size: Bigger particles have less velocity
    - Velocity: At start lifetime vel is negative, as soon as it goes towards +5, dragging is more evident
- InheritVelocity
  - Initial The particle emitted at time t takes the emitter instant tangent velocity at time t
  - Current The particle emitted at time t take the current emitter instant velocity (with a multiplier of 1, the particle stays over the emitter: it has the exact same velocity of the emitter)
- ForceOverLifetime
- ColorOverLifetime
  - From ForcesOverTime, try to set a gradient
- ColorBySpeed
  - From ForcesOverTime, try to set a gradient and SpeedRange [5,10]

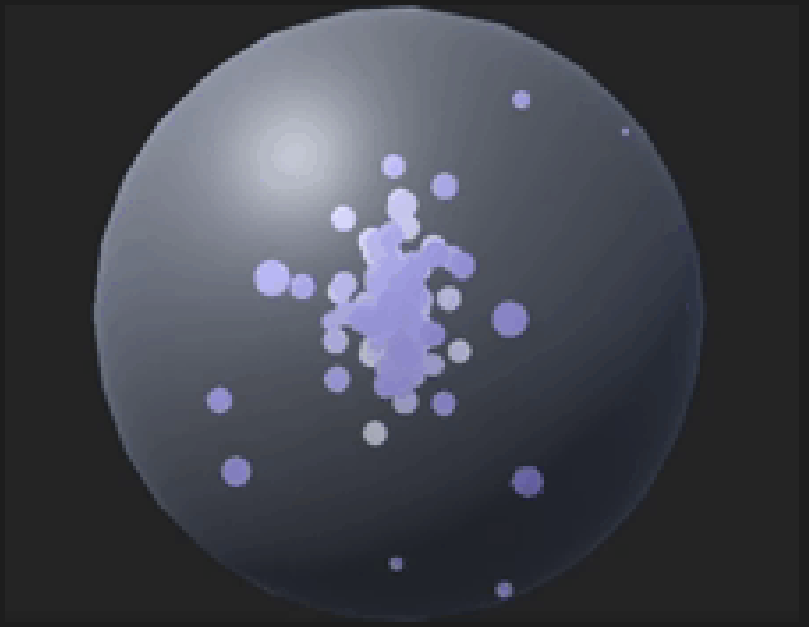


ExternalForces

Multiplier
- Scale the force applied to this particle system from Windzone

WindZones
- Influence area
  - Directional - Limitless
  - Spherical
- Main Main Strength
- Turbolence Main Variation
- PulseMagnitude is an additional force that adds to Main Strenght according to a Frequency, to create a more natural effect
- Set PulseFrequency or PulseMagnitude to 0 to have a uniform constant Wind

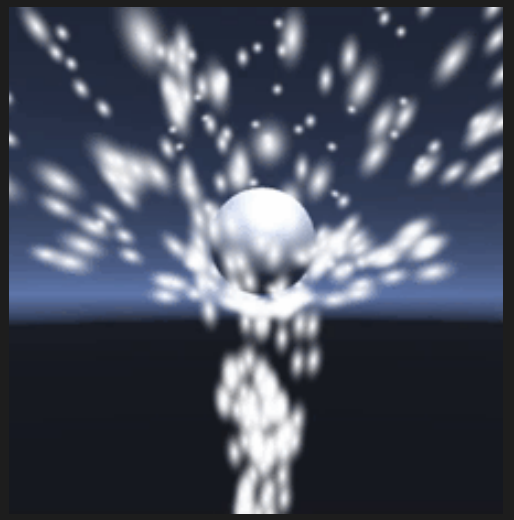


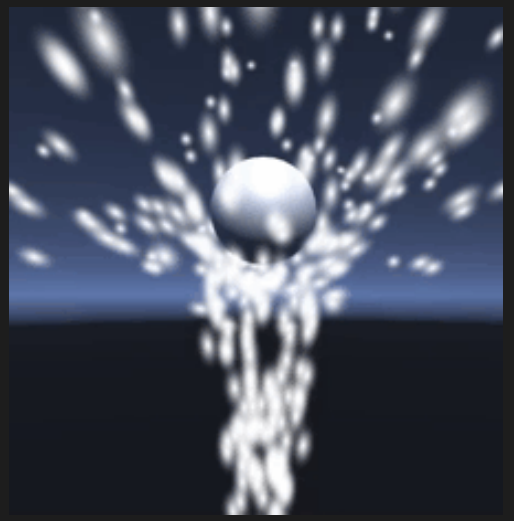


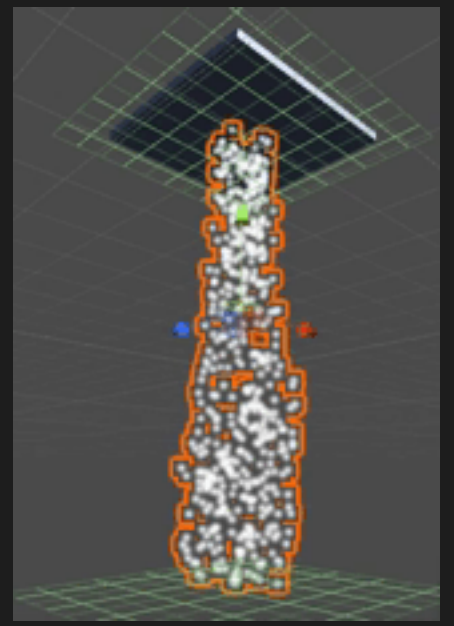
Collisions 1/2

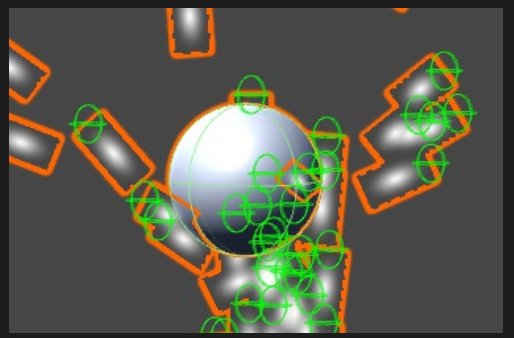
- Performance issues
  - Turn on VisualizeBound to see every collider
- Try it
  - Activate ColliderForce
    - PS_Collissions_ColliderForce has Type: World: every particle will collide with world colliders
    - The Sphere is hitted with a force of ColliderForce, and its weight is determined by its mass
  - Activate Planes
    - Set PS_Collissions Type to Planes: every particle will infinite planes described by transforms you provide

- Particles have sphere colliders
- World/Plane collisions Particles can bounce between more planes
  - NB: Planes are infinite. Plane size is only for Visualization purpose
- Dampen When particle collides, it will lose this fraction of speed
- Bounce
- LifetimeLoss When particle collides, it will lose this fraction of StartLifeTime
- Min/Max Kill speed If outside the [Min,Max] range, particle will die on collision
- Max collision shapes Terrains take priority
- RadiusScale Particle sphere collider are scaled by this value







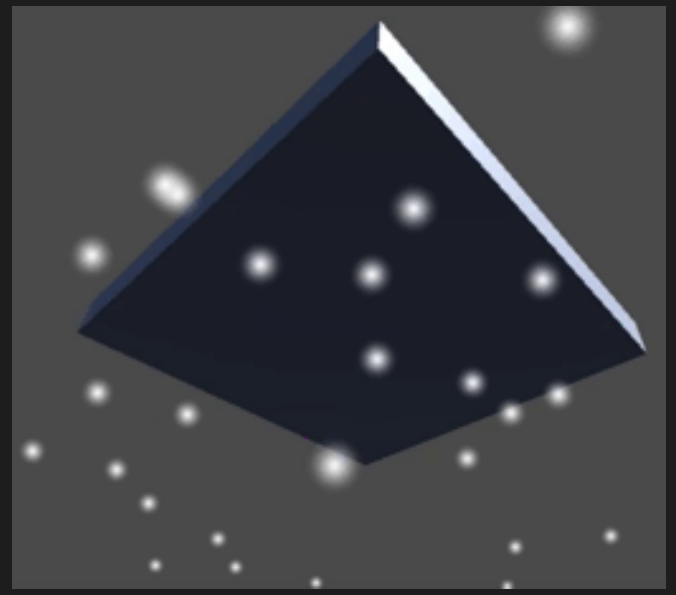


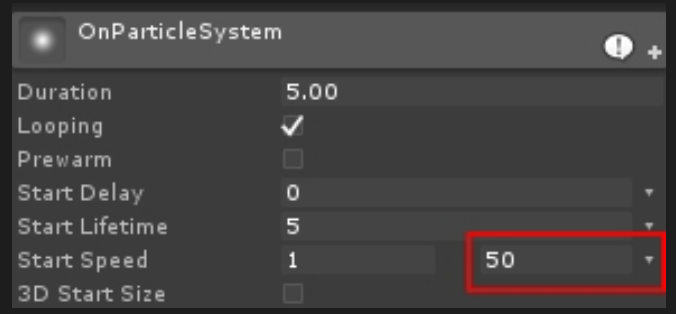
Collisions 2/2

SendCollisionMessage

- OnParticleCollision(GameObj objVal)
  - Invoked from a script attached to a GameObject with a Collider
    - objVal is the PS
  - Invoked from a script attached to a PS
    - objVal is a GameObject with an attached collider struck by the PS

- ParticleSystem.GetCollisionEvents
  (GameObj objVal, List<ParticleCollisionEvent> CEventList)
  can be used to retrieve all the collision incidents on the GameObject.
  - To know more about i-th collision incidents:
    - CEventList[i].intersection/normal/velocity





To retrieve info about the PS, we need to access its modules

- ParticleSystem.main.startSpeed.constantMax access to the Main Module

[PCollisionOnPSystem.cs, PCollisionOnGObj.cs]

Scripting 1/4

How to access PS particles?

- We use an “allocation free” method: the input Array particles is preallocated once
- Only a small part of the particles array might be used as this depends on how many particles are currently alive in the particle system when calling GetParticles()

1. Preallocate a Particle array
   ParticleSystem.Particle[] particles = new ParticleSystem.Particle[ParticleSystem.main.maxParticles]
2. Get Particles
   int numParticlesAlive = ParticleSystem.GetParticles(Particle[] particles)
3. Iterate through them and modify particles behavior/property
4. Write modified particles back to the PS
   ParticleSystem.SetParticles(m_Particles, numParticlesAlive);

[ParticlesBoost.cs]

Scripting 2/4

To change position value of a GameObject, we do

- Vector3 oldPos = GObj.transform.position;
- Vector3 newPos = oldPos*2;
- GObj.transform.position = newPos;

To change PS properties, we need to access its modules

```
// Get the emission module
ParticleSystem.EmissionModule emissionModule = GetComponent<ParticleSystem>().emission;
// Enable it and set a value
emissionModule.enabled = true;
emissionModule.rateOverTime = 15;
```

- Does it seem weird? Why?

Scripting 2/4

To change position value of a GameObject, we do

- Vector3 oldPos = GObj.transform.position;
- Vector3 newPos = oldPos*2;
- GObj.transform.position = newPos;

To change PS properties, we need to access its modules

```
// Get the emission module
ParticleSystem.EmissionModule emissionModule = GetComponent<ParticleSystem>().emission;
// Enable it and set a value
emissionModule.enabled = true;
emissionModule.rateOverTime = 15;
```

- Does it seem weird? Why?
  - We grab the struct and set its value, but never assign it back to the PS. How can the PS ever know about this change?
  - EmissionModule struct is just an interface into the PS internals

[PVT] Scripting 3/4

```
var emissionModule = GetComponent<ParticleSystem>().emission; [1]

public sealed class ParticleSystem : Component
{…
    public EmissionModule emission { get {
        return new EmissionModule(this); } } [2]
…}

public partial struct EmissionModule
{
    // Direct access to the PS that owns this module
    private ParticleSystem m_ParticleSystem;
    EmissionModule(ParticleSystem particleSystem) {
        m_ParticleSystem = particleSystem;
    }
    public MinMaxCurve rateOverTime { set {
        // Here we call down to the C++ side to perform the rate variation
        m_ParticleSystem->GetEmissionModule()->SetRate(value); [3]
    }}
}
```

- PS are in the Unity C++ side
- PS Modules are properties of a PS, they are never shared between different PS: a module will always belong to the same PS

1. PS receives a request for the Emission module
2. The engine create a new EmissionModule struct, passing the owning PS as its only parameter
3. To set the rate, the var m_ParticleSystem is used to access the module and set it directly

- This is why we don’t need to reassign the module to the PS: it is always part of it
- EmissionModule struct is just an interface into the PS internals
- It is not possible for modules to be shared or assigned to different PS (like we do for the Vector3 position of an Object, for example)

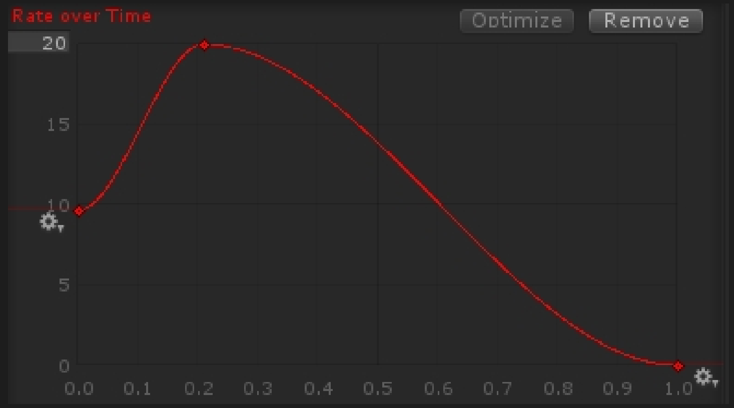
Scripting 4/4

MinMaxCurve class

- Used to describe a change of value over time. 4 supported modes
  - Constant
  - Random Between 2 constants
    - We get our value by performing a lerp between the 2 values using a normalised random parameter as our lerp amount.
  - Curve
    - Stored in 2 ways, depending if it has more than 3 keys (in this case you can optimize the curve)
  - Random Between 2 curves

- It is possible to evaluate MinMaxCurve value even if it is a curve value, with MinMaxCurve.Evaluate()
- In PSystemOps.cs you have CustomMinMaxCurve and CustomMinMaxValue which are evaluated in between the curves and the gradients (Lerp val is 0.5f)
- You can use ParticleSystem.MinMaxCurve and ParticleSystem.MinMaxValue also without Particles, just as value data

[PSystemOps.cs]

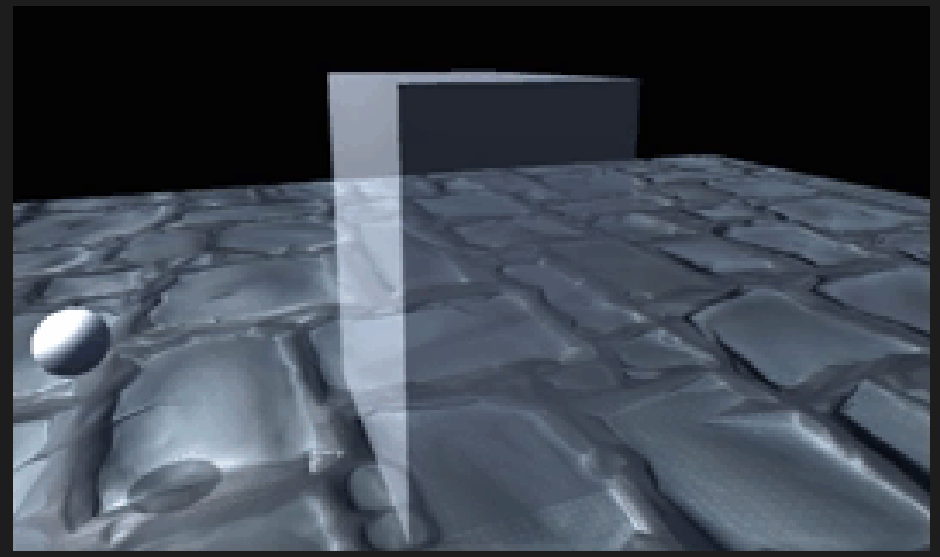


Trigger

- Modify particles when they interact with Physic Colliders
- Allows more control over collision and trigger events compared to Collision module
  - No automatic collision response
  - Modify particles in a script callback OnParticleTrigger()
    - Must be on a script attached to the PS
    - Inside/Outside/Enter/Exit events
- You must activate the Triggers module

How to access single triggered particles?

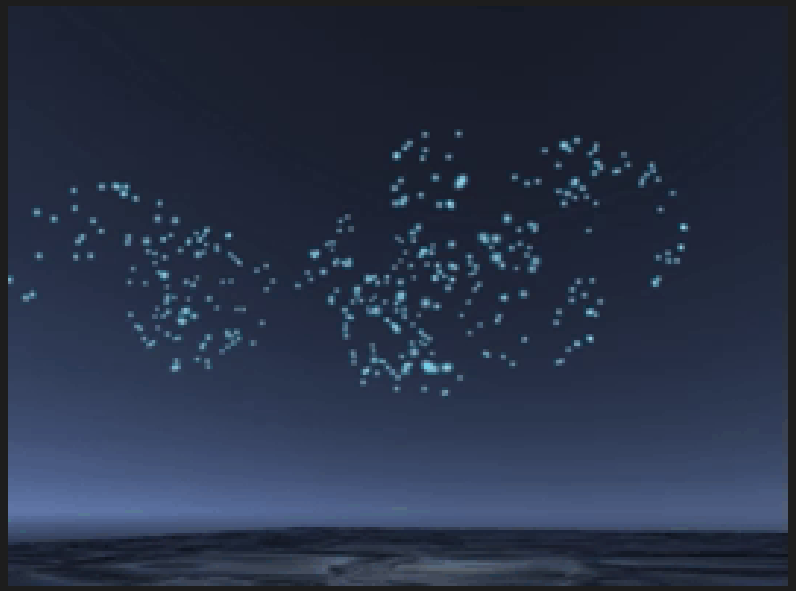
1. Get Particles using
   ParticleSystem.GetTriggerParticles
   (ParticleSystemTriggerEventType.Enter,
   List<ParticleSystem.Particle> enter)
2. Then iterate through them and modify single particle behavior/property
3. Write modified particles back to the PS
   - ParticlePhysicsExtensions.SetTriggerParticles
     (PS, ParticleSystemTriggerEventType, Particle[], offset, count)



[Ptriggers.cs]

Sub Emitters

- Fireworks, ClusterBombs, Explosions, Rain, etc
- Can inherit Parent PS properties of type Start (not OverLifetime) (Start-size/rotation/color/lifetime)
  - Inherited velocity is configured in the InheritVelocity module of the child PS
- SubEmitter trigger
  - Birth
  - Collision
  - Death
  - Manual: Only triggered when requested via script (ParticleSystem.TriggerSubEmitter(int subEmitterIndex)). You can have more than 1 subemitter
- Try it
  - SubEmitter0 is created only when the parent Particle System particles are death. Hence
    - RateOverTime / Burst Cycles and Interval values
      - are not taken into account

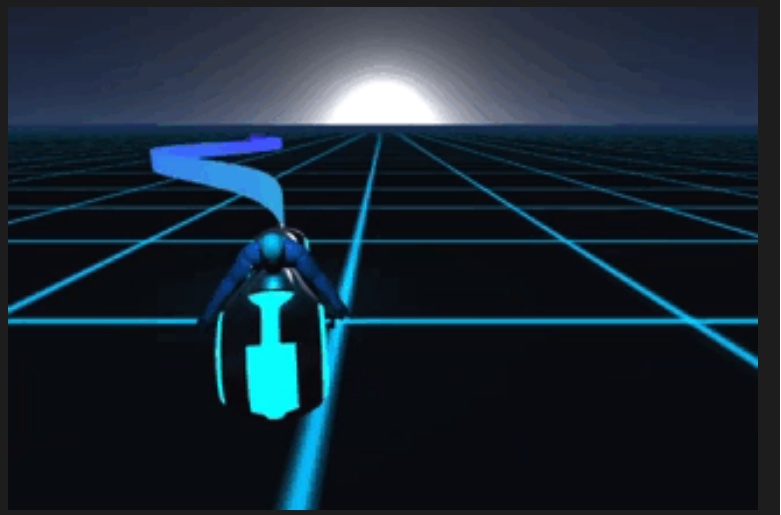


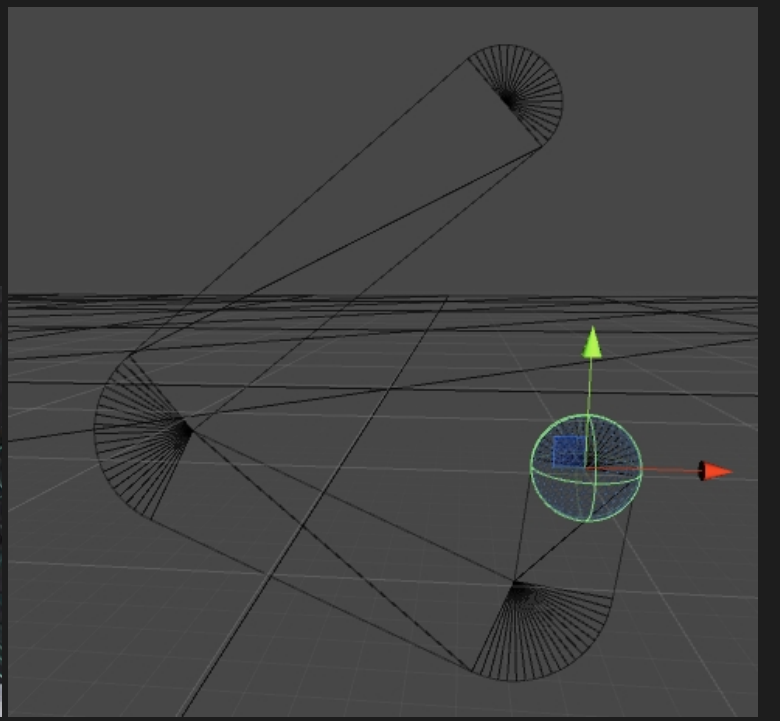
Trail Renderer

- Leave trails behing GameObjs in the Scene as they move
- Highlight path / Emphasize moving object
- Should be the only renderer used on the attached GameObject
  - Create an empty GameObj w Trail Renderer
  - Parent it to the appropriate moving GObj

- MinVertexDistance how far an object that contains a trail must travel before a segment of that trail is solidified
- Corner/EndCap Vetices Refine geometry at the end and in between Trail segments
- Autodestruct Destroy the GameObj if it doesn't move for Time seconds

- Sometimes produce artifacts: see other products in the Asset store





Trails

- Like Trail renderer component, leave ribbons of geometry behind particles
- WorldSpace If ON, Trails move in WSpace even if PS is using LSpace.
- TextureMode How Trail texture will repeat
- GenerateLightingData Calculate also Normals and Tangents
- Trail Material is set in Renderer module

Mode

- Particle
  - Ratio Probability for each particle to have a Trail
  - MinVertexDistance Distance that the particle must travel before its trail receives a new vertex
- Ribbon
  - RibbonCount How many ribbons to render throughout the PS
    - 1 Creates a single ribbon connecting each particle
    - N>1 Creates ribbons that connect every Nth particle
      - Eg. 2 = one ribbon connecting particles 1, 3, 5, another ribbon connecting particles 2, 4, 6 (and so on) The ordering of the particles is decided based on their age
- To render only trails
  - RenderModule/RenderMode: None

ForceField

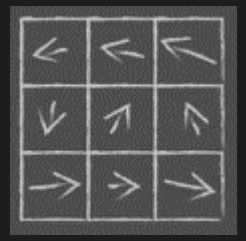
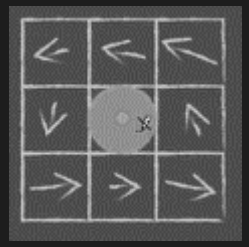
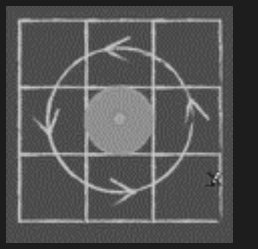
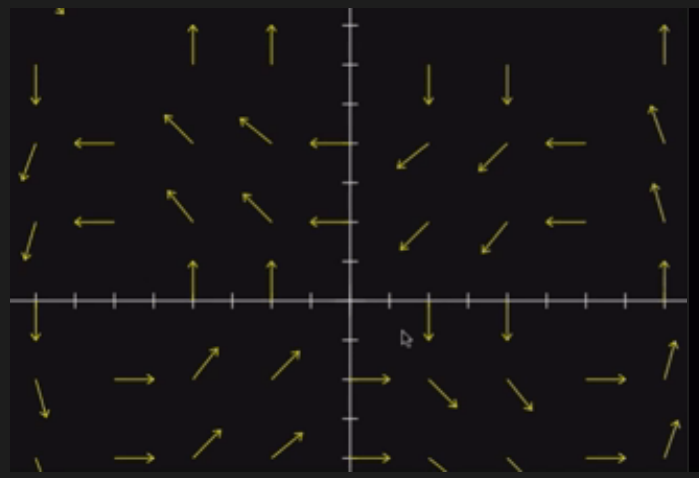
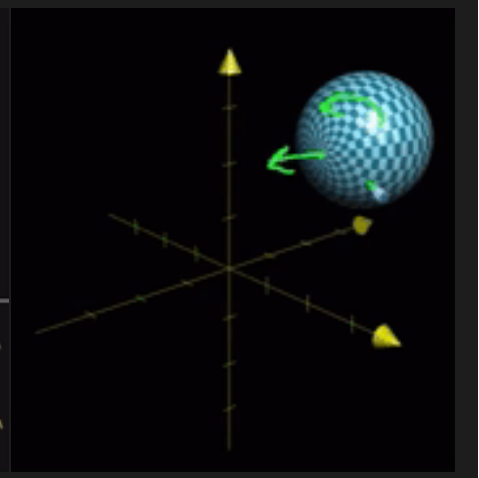
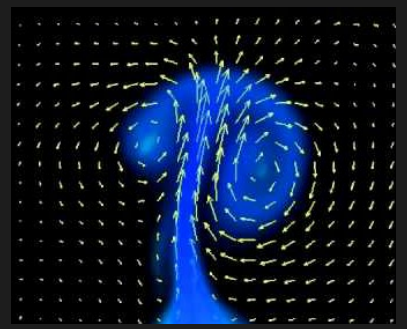
- Psystem must have ExternalForces module ON
- Focus Acts like an inner space where particles are attracted. It is a normalized value
  0: The FField center
  1: EndRange

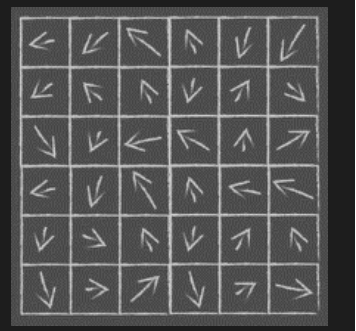
Try it
- Activate TrailsWithForceField and CompositeForceFields
- Tweak ForceField component values

VectorField
- You need a 3DTexture to start with, or some plugin that create that texture
  - VectorFieldMaker

Noise 1/3

- Based on Curl Noise
- Vector field
- A Curl is a vector operator that describes the infinitesimal rotation of a vector field in three-dimensional Euclidean space
- We can calculate curls in space using a Curl function, which samples the neighbors cells
- Curl noise is a Curl function of a Perling noise
  - Perling noise is a type of gradient noise developed by Ken Perlin in ‘80 for Tron (search for his SIGGRAPH 1985 paper)
- Perling noise > Perling noise Vector field > Curl noise Vector field
- Used for Gas & Fluids simulation & fast to compute



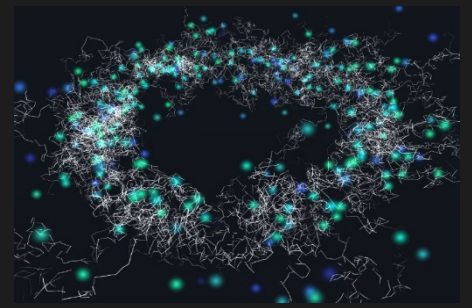
Noise 2/3

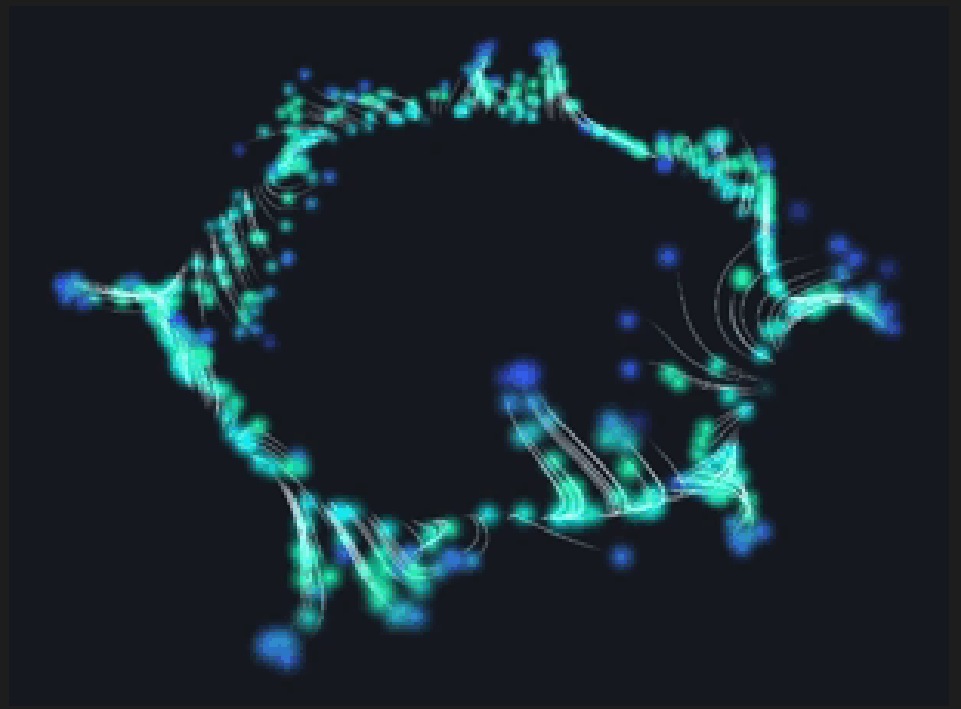
Increase erratic particle movement
Try it

- Start from Noise/Start, and Tweak Noise/Strength value

- Octaves Adds layers of noise (computationally expensive)
- Scroll Move noise field to change noise pattern
- Quality How many unique noise samples are generated.
  - 1D and 2D noise re-use samples across axes (combining them in a different way to hide reuse), but is a significant performance benefit

- Damping If ON, strength is proportional to frequency

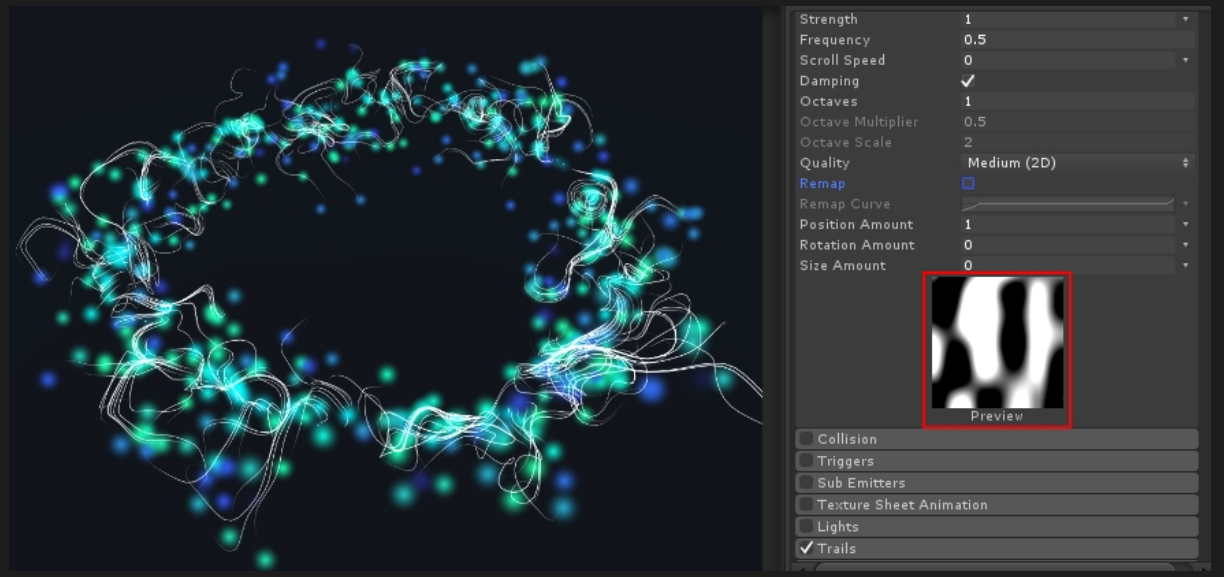


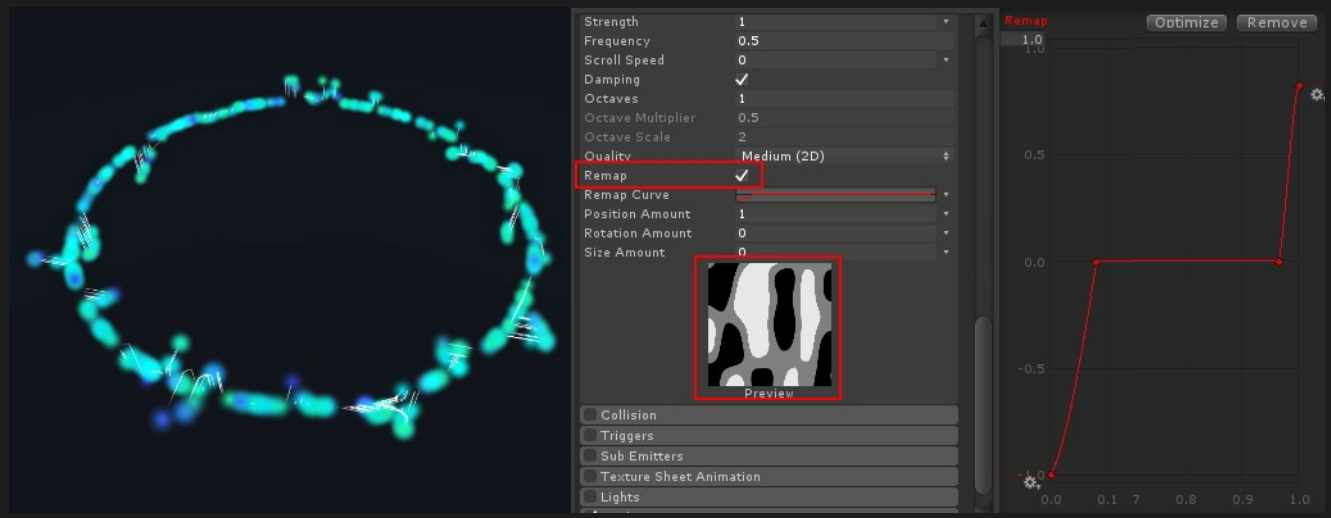


Noise 3/3

Noise Remap

- Noise values range is normalized on X axis, and are remapped using a curve function
- Allows to define custom noise shapes, muting noise in certain range
- Casual curves will result in more unpredictable noise





Renderer

RenderMode
- Billboard For particles representing volumes that look much the same from any dir (clouds)
- Stretched Billboard “Stretch & Squash” technique applied to particles: aligned to face the camera and to their velocity
- Mesh

Normals
- 1 particle Normal will point towards the Cam
- 0 particle Normal will point towards the corner direction of the particle

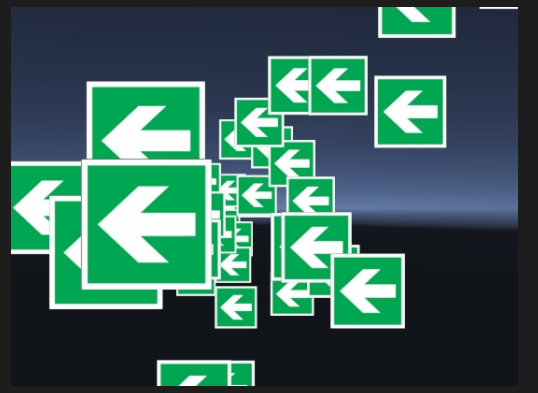
SortingFudge This value should be low if you want high probability that this PS appear in front of other particles or other transparent objects

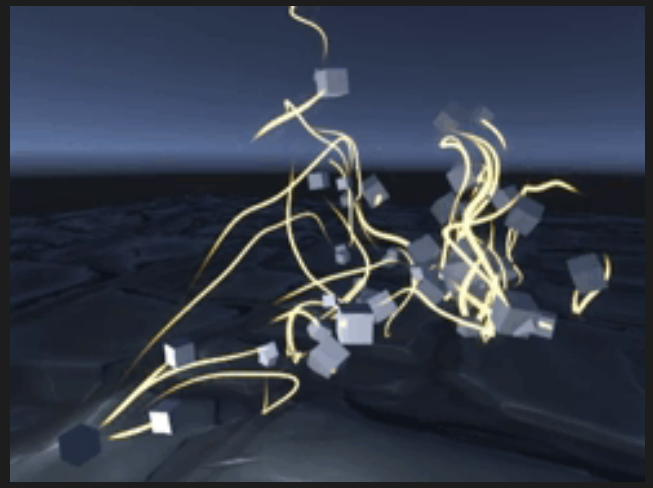
Pivot (use Visualize Pivot to see what happens)
- If Particle size change, and Pivot remains constant => Particle position will change

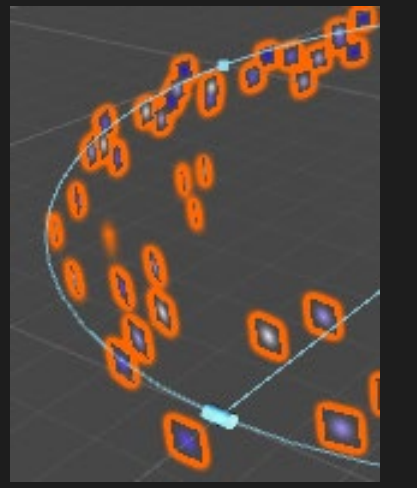
Min/MaxParticleSize This is related to the viewport. If a particle takes all the viewport, there will be a lot of overdraw!

RenderAlignment Facing VS View

Ralignment Facing / NormalDirection 1 (left) 0 (right)







Lights

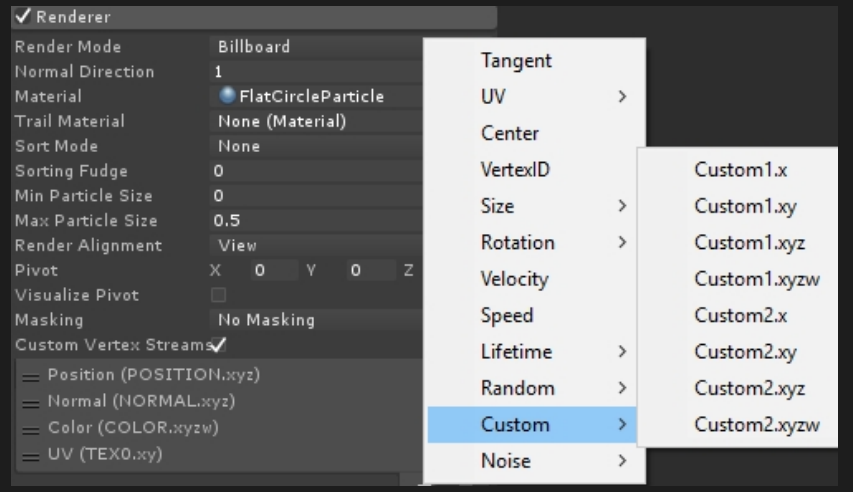
- Add lights to particles
- Supports Shadow casting (very expensive)
- Lights can inherit color from particles
- # of lights emitted from PS depends on:
  - LightModule/Maximum light
  - Material shader used on lit surfaces
  - Edit/ProjectSettings/Quality/PixelLightCount (if Forward rendering is used)
- Ratio
  - RndDistribution OFF Percentage of Light Particles
  - RndDistribution ON Probability for each particle to have a Light
- URP limits
  - Forward rendering: up to 8 lights
  - Deferred rendering: up to 20/30 lights

CustomData

- Define 2 custom data formats (Vector4/Color) streams in the Editor to be attached to particles. Can also be set via script
- Generate custom per-particle data, which can be used either in script or shaders

- Script
  - List<Vector4> CData = new List<Vector4>();
  - PS.GetCustomParticleData(CData, ParticleSystemCustomData.Custom1);
  - Return the list in the same order for particles in GetParticles()
    - This is useful, because you know that if CustomData is a curve from 0 (at particle initial lifetime) to 10 (at particle end lifetime), if particles[i] is a particle at normalized lifetime 0.5 (half its lifetime), it will get a CustomData value of 5
- Shaders
  - Passed via a TEXCOORD Channel
  - Enabled in the Renderer Module

- Eg. The artist wants a second ColorOverLifetime when the avatar is in a specific state



[PSCustomData.cs]

Masking

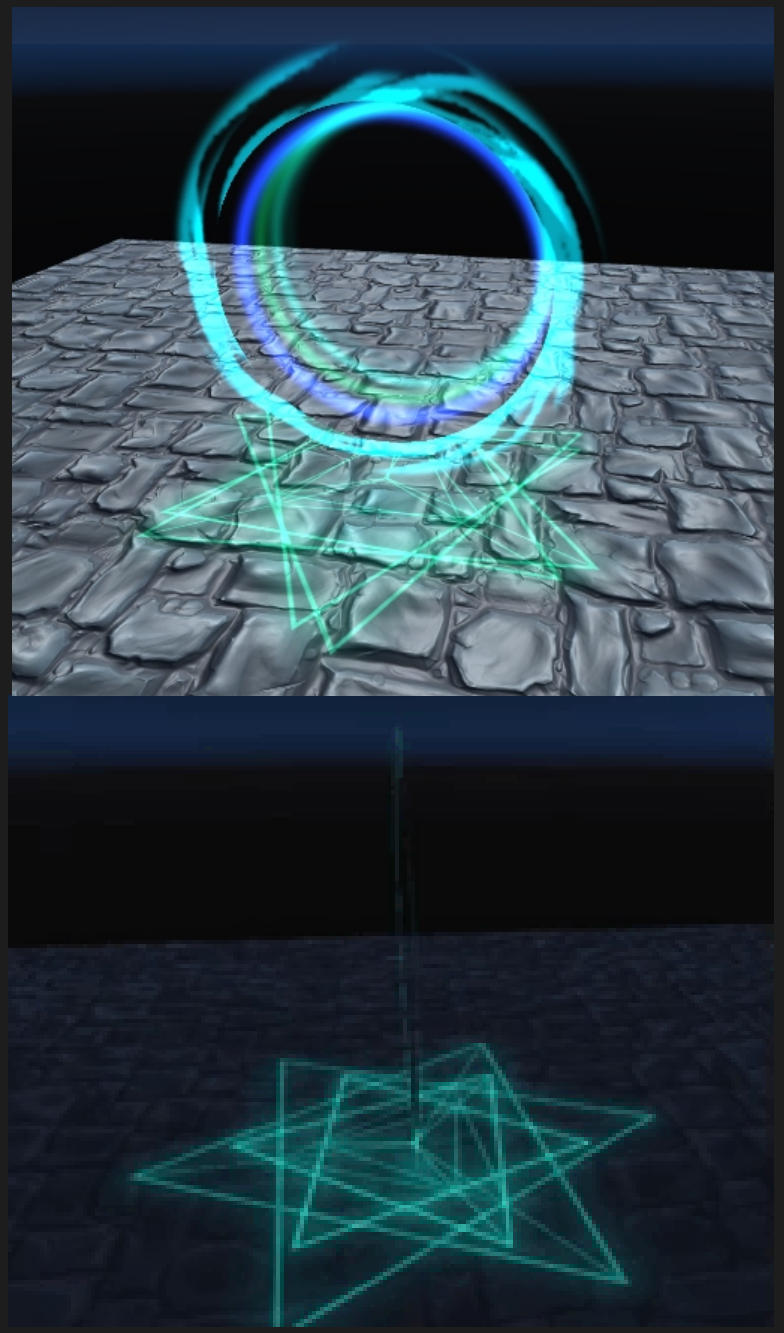
Setup
- Use Forward Rendering

Ground
- PortalGround Material URP/Particles/Unlit-Transparent
  Texture Geom_00
- Rotation.x -90
- Emission RateOverTime: 1
- MaxParticles: 1
- Duration: 5
- Loop
- Render
  - RenderMode: HorizontalBillBoard
  - MaxParticleSize: 10
- StartSpeed: 0
- StartSize: 40
- Shape: OFF
- SizeOverLifeTime: 0.5>1
- RotationOverLifeTime: 45
- ColorOverLifetime: 4 keys
  start/end Alpha 0, 2 inner keys Alpha 255
- Add a pointlight over the ground: Intensity 10, Range 10
- Duplicate the entire particleSystem: Ground2, with these changes:
  SizeOLTime From 1 to 0
  RotationOLTime -45

Circle (Duplicate Ground)
- PortalCircle Material URP/Particles/Unlit-Transparent
  Texture CircleGlow_00
- StartSize: 10
- StartLifeTime: 4
- MaxParticles: 50
- RotationOverLifeTime OFF
- VelocityOLTime.z 1
- Render
- Transform.Rotation.x -90
- Transform.Pos.y 5
- Material PortalCircle
- Render
  - Masking: VisibleOutsideMask
  - RenderMode Billboard
  - RenderAlignement Local

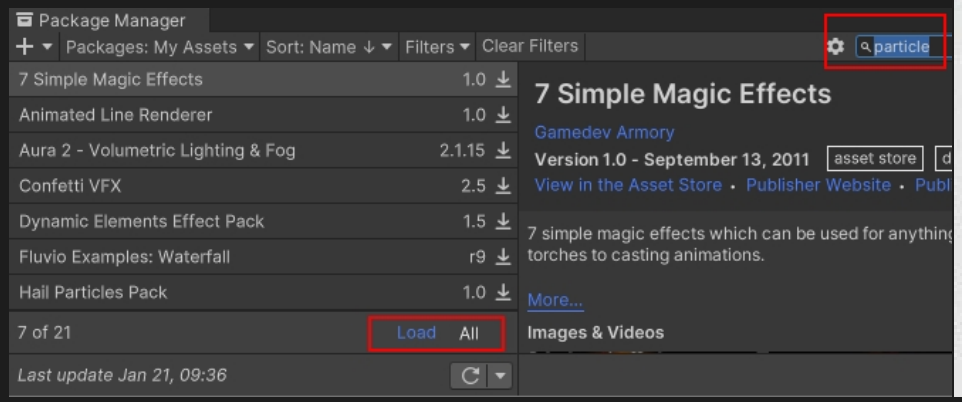
Aura (Duplicate Circle)
- PortalCircle Material URP/Particles/Unlit-Transparent
  Texture Aura_00
- StartSize: 15
- StartLifeTime: 4
- MaxParticles: 1
- RotationOverLifeTime 45
- VelocityOLTime OFF
- Render
  - Masking: VisibleOutsideMask

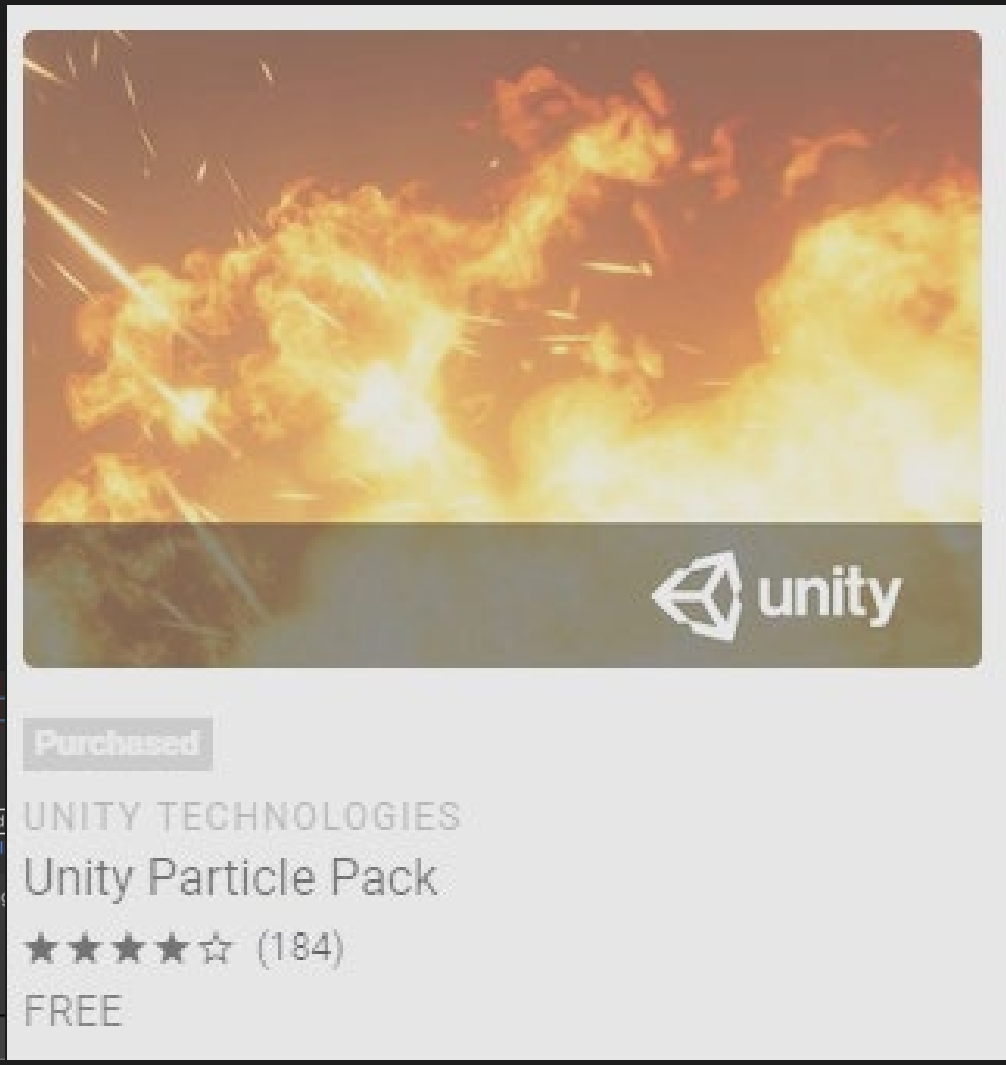
Mask
- Duplicate Circle PS Gobj and start from there
- Remove PS
- Has SpriteRenderer and SpriteMask component with a Circle sprite in it
- SpriteRenderer is used only as a reference: Emitting Circles should have this size. After creating the emitting circles Psystem, you can disable SpriteRenderer
- Set it as parent of all Psystems created



A good reference: Unity particle pack

- Create a new project 2021.2.7f1, 3D URP Template
- Install Unity Particle Pack from PackageManager
  - If it isn't listed, press "Load All" button
  - This will install also postprocessing stuff

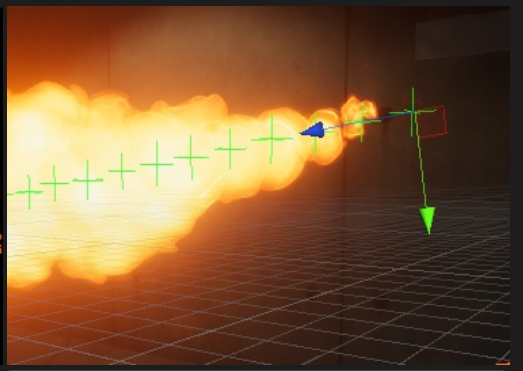


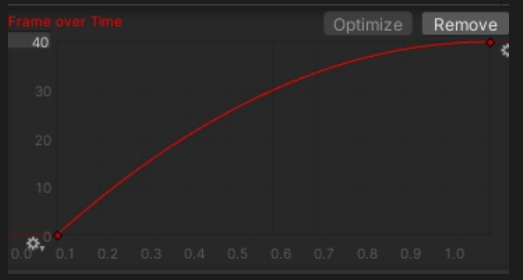


ParticleEffect samples

- Open main scene

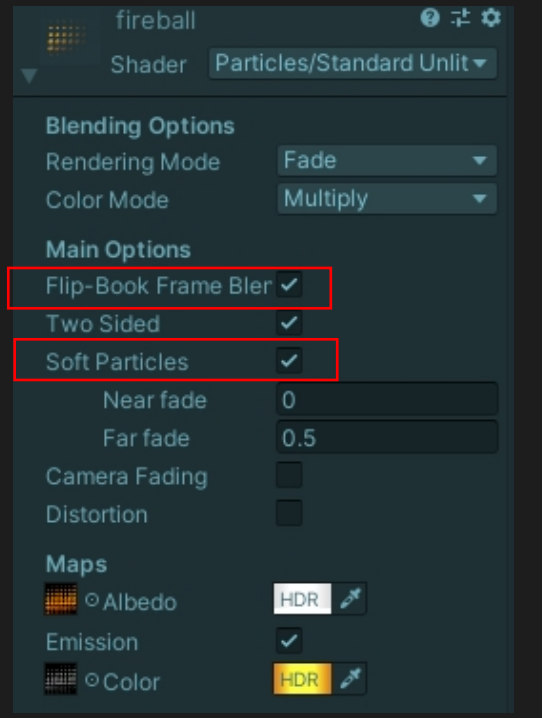
Flame Thrower
- TextureSheetAnimation: 7x7 grid
- For flames, usually is a good idea to use alpha 0 value for end or start/end ColorOverLifetime
- Animation WholeSheet
  - Every sprite in the sheet is take into consideration
  - TimeMode: Use the sprites according to particle Lifetime. The curve FrameOverTime rapresents:
    x: lifetime
    y: # of sprite to use

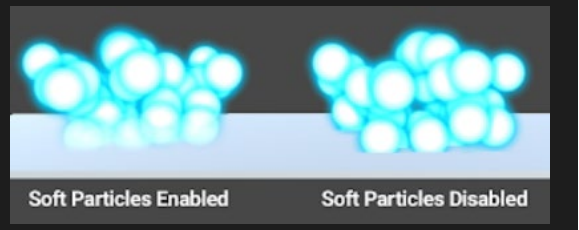


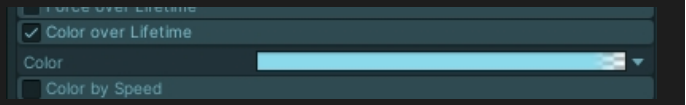


ParticleEffect samples

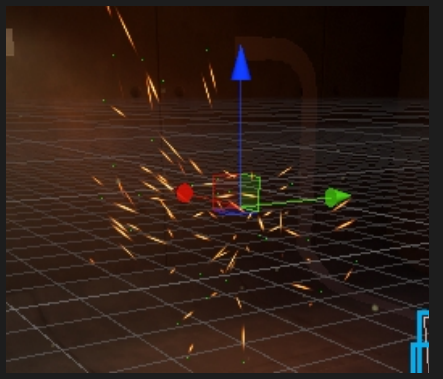
- Fireball Material
  - Flip-books Frame blending Render images as individual frames or blend the frames together to give smoother animations. Set to either:
    - Simple - Render frames in a flip-book as a sequence of individual frames (you can test this setting the TimeScale to 0.2)
    - Blended - Blend the frames in a flip-book to render the flip-book as a smooth animation
  - Soft particles Fade out particles when they get close to the surface of objects written into the depth buffer. This is useful for avoiding hard edges when particles intersect with opaque geometry. For example, by enabling soft particles, you can make the particle system emit particles close to an opaque surface without causing harsh intersections with the surface
  - Fade out particles when they get close to the camera. Set to:
    - Near fade - The closest distance particles can get to the camera before they fade from the camera's view
    - Far fade - The farthest distance particles can get away from the camera before they fade from the camera's view







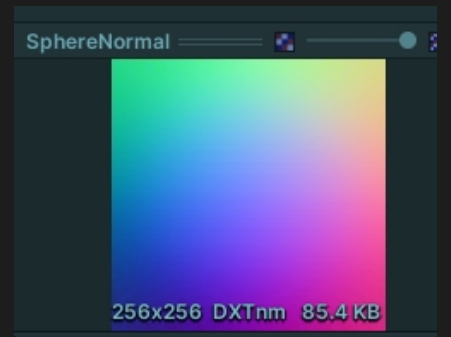
ParticleEffect samples

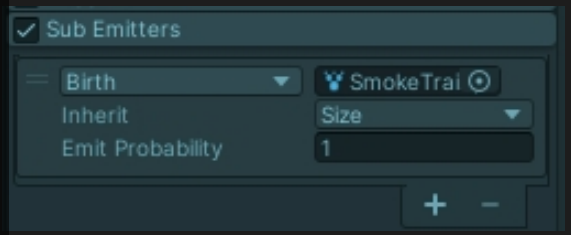


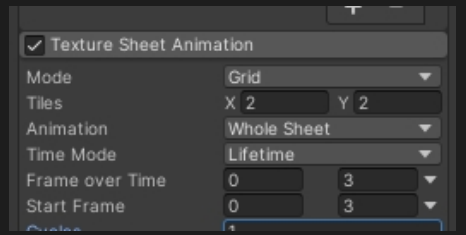
Fireball
- Has a FireEmbers as secondary PS
  - Billboard stretched + Noise

EnergyExplosion
- Embers can explode and implode using Radial velocity value in the Velocity module
- Shockwave has a distortion effect in the shader
  - Use a SphereNormalMap to enhance the Distortion effect

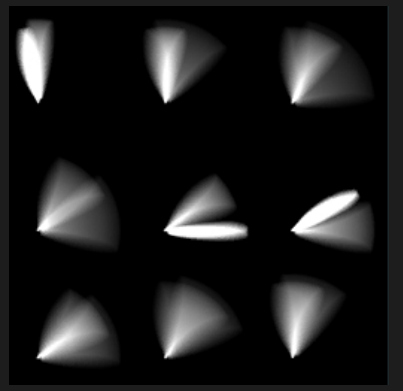
BigExplosion
- Debris
  - SubEmitters at birth emits SmokeTrail Psystem
  - TextureSheetAnimation Choose at start one frame, from 0 to 3, from the Debris texture

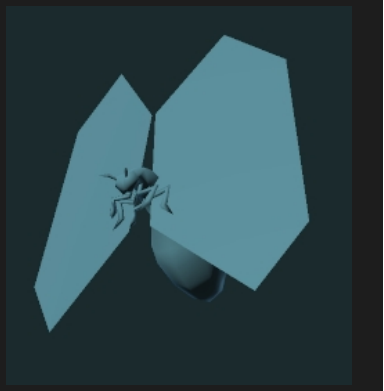






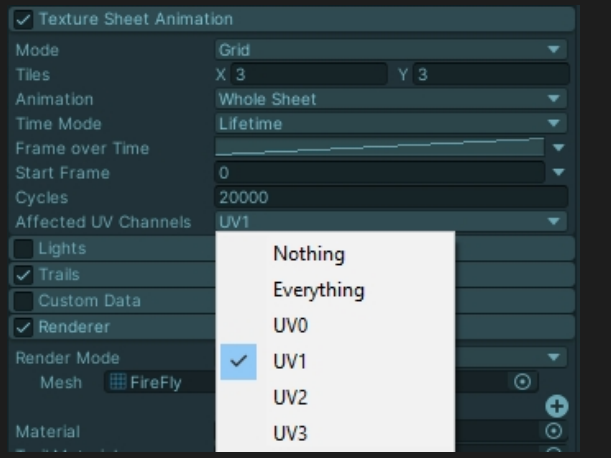
ParticleEffect samples





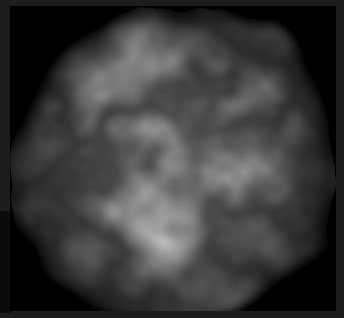
Fireflies

- Renderer use a mesh
- The shader has 2 pass: 1 pass to render the fly body, 2nd pass to render flapping wings
- Wings have a different Uvset, and it is animated by the TextureSheetAnimation module, that affects only UV1



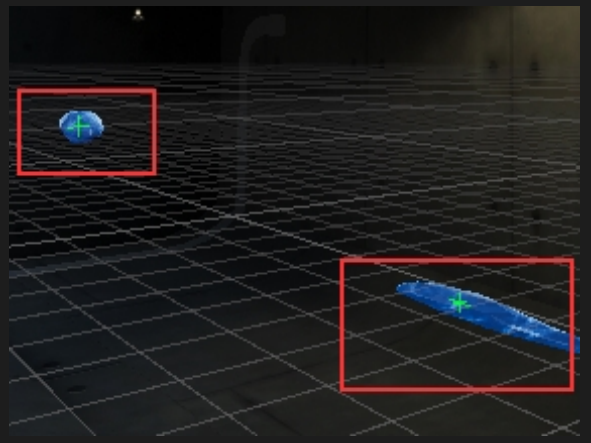
GoopStreamEffect

- GoopStreamEffect and PoolEmitter has the same StartSpeed, hence their particles follow the same path until the collision with the floor, even if their emitter module is very different:
  - PoolEmitter.RateOverTime = 10
  - GoopStreamEffect.RateOverTime = 150
- PoolEmitter.Renderer.RenderMode = None
- We use
  - PoolEmitter collisions to emit subEmitter GoopPool when floor collision occur
  - GoopStreamEffect collisions to emit subEmitter GoopSplash when floor collision occur
- GoopPool animation is handled by color and sizeOverLifeTime, not by a TextureSheetAnimation module
  - ColorOverLifeTime changes its alpha, and the material is a cutOut shader that will make the Goop disappear in a fancy way, because the Alpha channel of the texture is in grayscale
  - SizeOverLifeTime lets the Goop grow and then shrink



GoopPoolParticle material Alpha channel: useful to animate a fadeout via Alpha controlled by ColorOverLifetime

ParticleEffect samples



IceLance

- IceLance
  - emits the big ice 3D mesh. As soon as it collides, it looses its velocity because Collision.Dampen is set to 1
    - Try to set Collision.Dampen to 0 and Collision.Bounce to 1
  - Has a Trail module
  - Has 2 SubEmitters
    - TinyShards, a 3D mesh emitter, at its Death (it is the IceLance that is destroying itself)
    - Emits Mist at its collision, to add some Dust

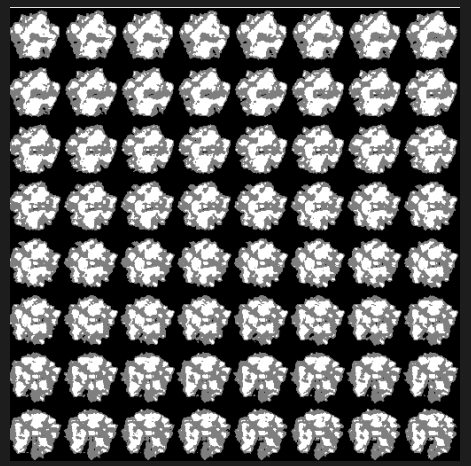
Other particle software

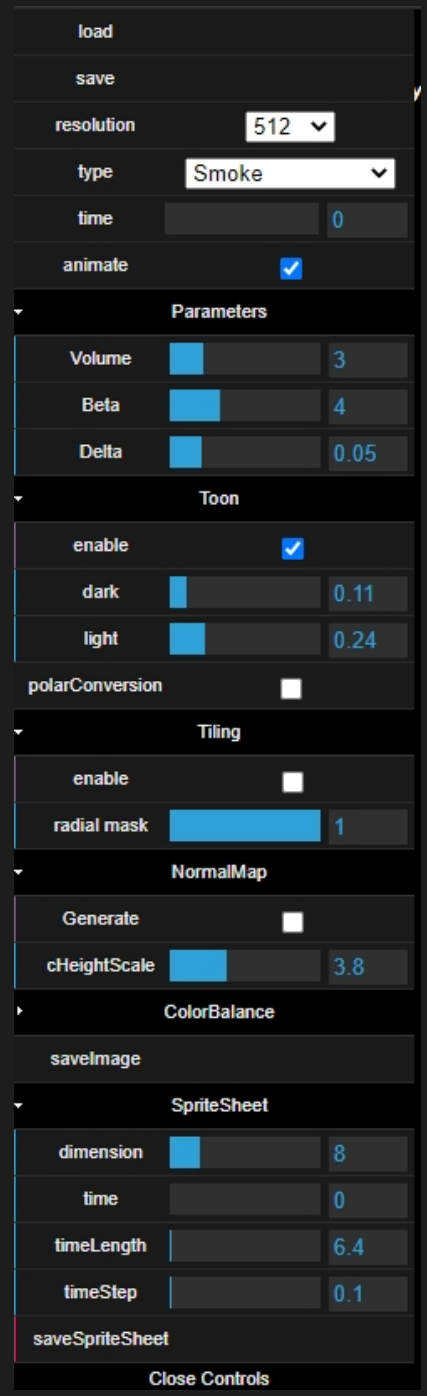
- Generate Noise textures
  - https://mebiusbox.github.io/contents/EffectTextureMaker/

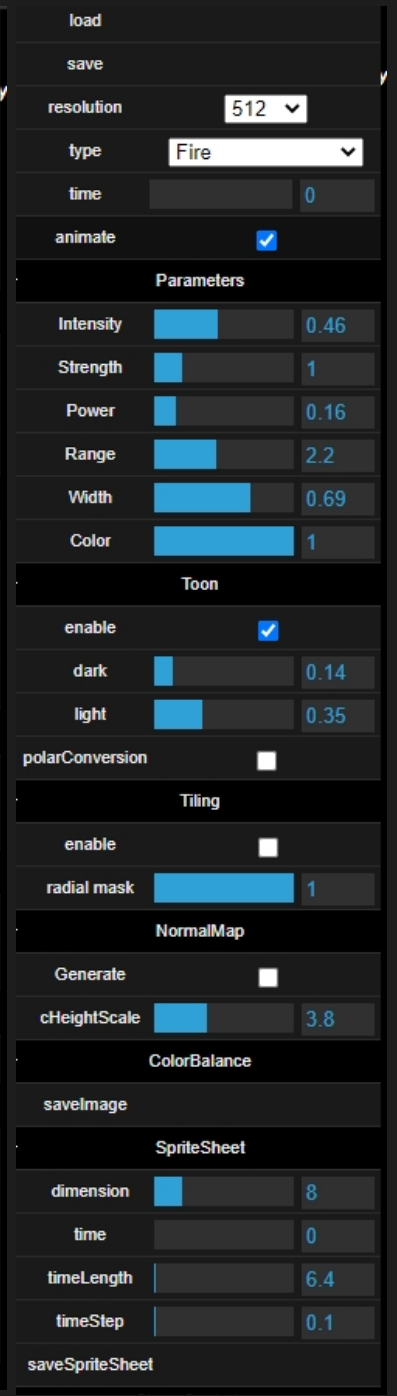
- Other particle software
  - https://www.popcornfx.com/
  - https://codemanu.itch.io/particle-fx-designer
  - https://store.steampowered.com/app/940920/BlastFX/
  - https://jangafx.com/software/embergen/

Other particle software

- Generate Noise textures
  - https://mebiusbox.github.io/contents/EffectTextureMaker/
- Apply to Smoke/Steam PS
  - TextureSheetAnimation is already 8x8
  - Alpha from Grayscale
- Apply to CandleFlame







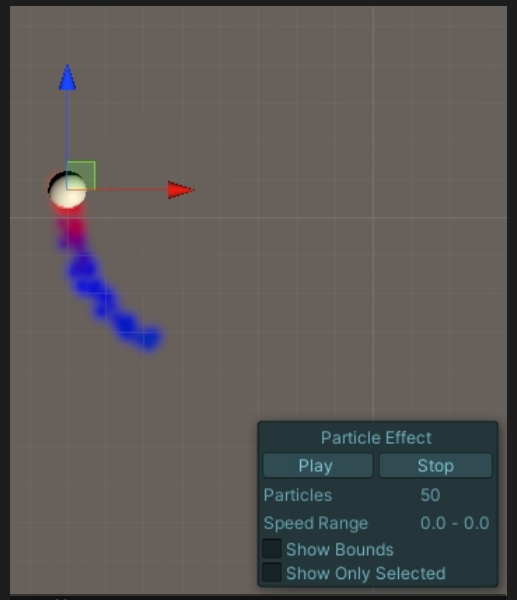
Performances

RingBuffer

- Once the list of particles reaches the maximum num of particles, it can no longer spawn anymore
- In RingBufferMode, rather than dying when their lifetime has elapsed, particles will remain alive until the MaxParticles buffer is full, at which point new particles will replace the oldest
  - PauseUntilReplace Play the normal animation of the particle, but instead of dying at the end and being removed, it just stays alive at its last state it pauses
  - LoopUntilReplace will let you set a loop point, so you'll just continuously play that animation between those two loop points and then when the particle is going to be removed we play the remainder

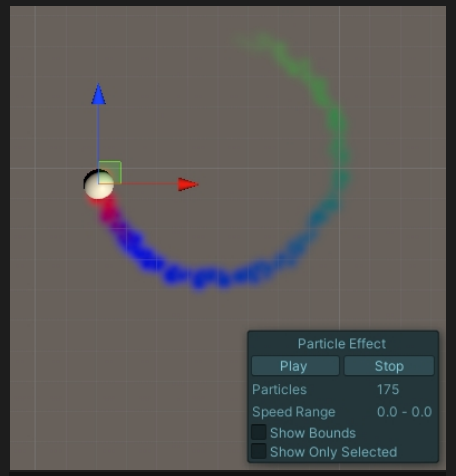
Rbuffer Disabled

After reaching MaxParticles (50), stops emitting until the oldest particles (on the left) begin to die



Rbuffer PauseUntilReplaced

After reaching MaxParticles (50), remove oldest particles and emits new ones



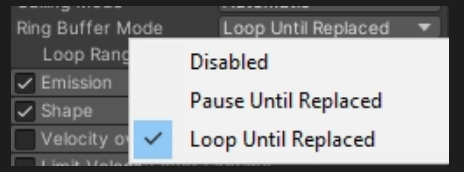
Rbuffer LoopUntilReplace – Loop [0,1]

Emits particles until their lifetime reaches loop range and until MaxParticles is reached (or exceeded), then emits one last complete lifetime range loop

Here PS emits until the entire lifetime (loop range 0,1) is covered: at the end of that we emitted 175 particles, this is why it stops

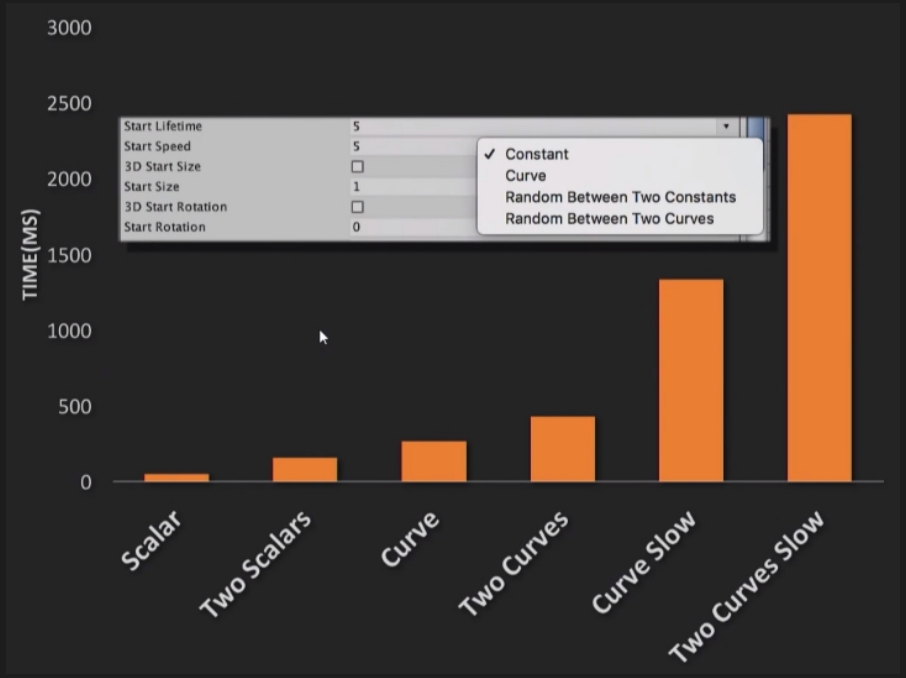
Rbuffer LoopUntilReplace – Loop [0,0.2]

PS emits until 0.2 of the entire lifetime (0.2*5=1) : at the end of that, we emitted less than 50 particles, then emits the last complete loop (175 particles)

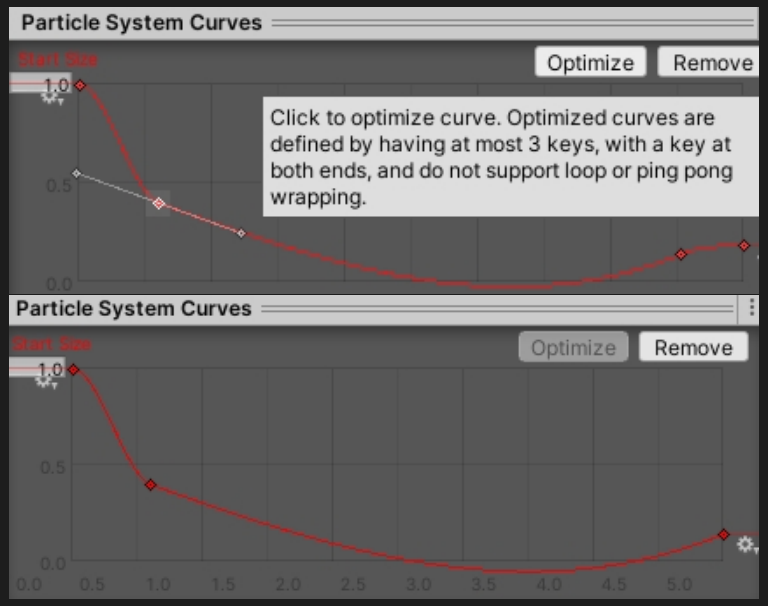


Performances

- Evaluation times are different from “Constant Scalar” to “Random between two curves with more than 3 keyPoints”

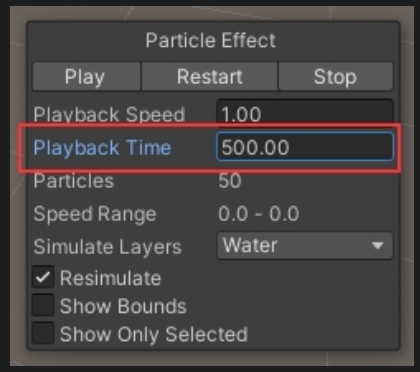


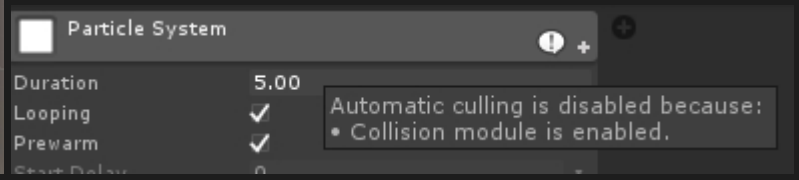
A “slow curve” is a curve with more than 3 keyPoints. Unity lets you to optimize those curves y clicking on “Optimize”

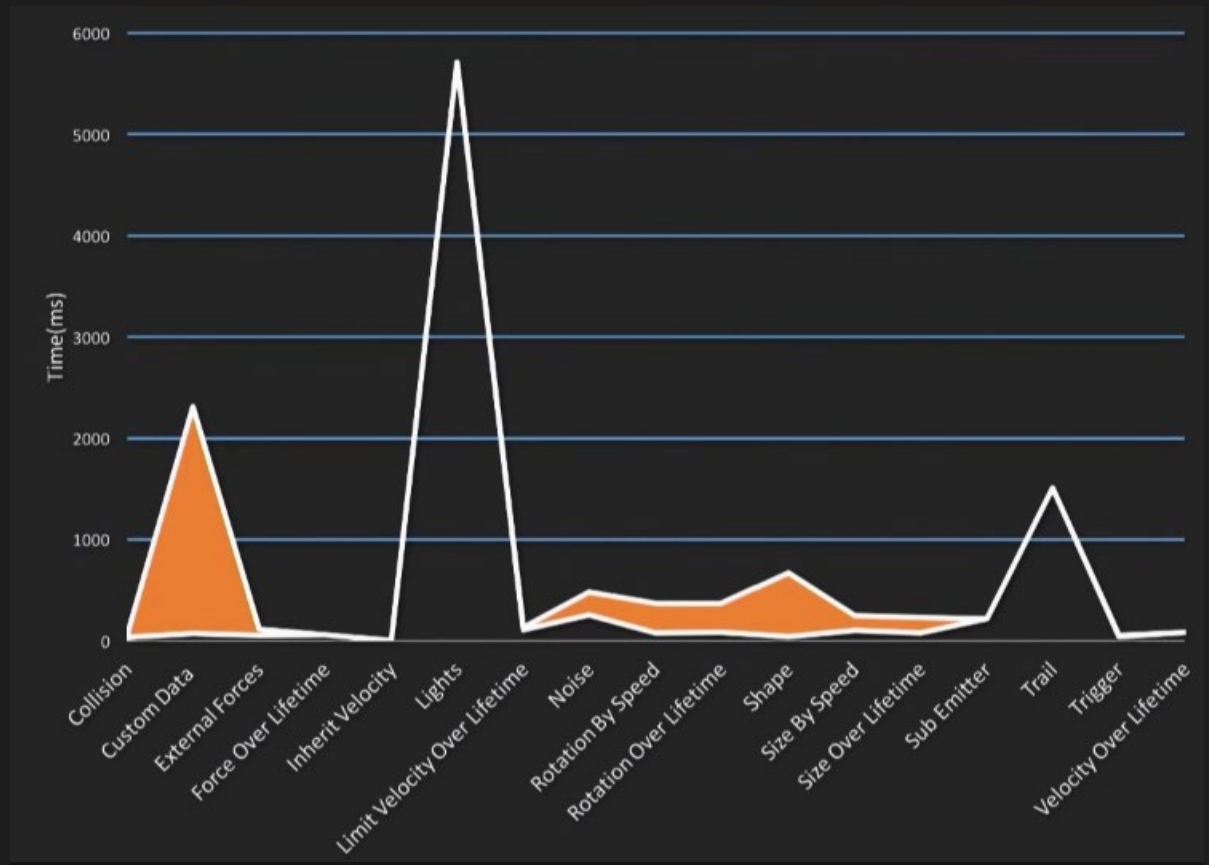


Performances

- Multi-threading
- SIMD Math Library
  - Allows to perform 4D vector calculations with a single instruction. 2.5x faster
- ProceduralMode
  - Can predict the state of a particle system at any time: this means that we don't have to constantly update it when it's not on screen
  - If you want to simulate for a particle system you can jump to that point in time instantly
    - In Editor Mode, add to the PS the collision module, a balloon with an exclamation point will appear
    - When some modules are enabled, this feature is lost
      - Collisions
      - RateOverDistance
      - SimulationSpace World
      - Etc…
- AnimationClips can contain all PSystem properties. Can improve performances (eg. Velocity change)

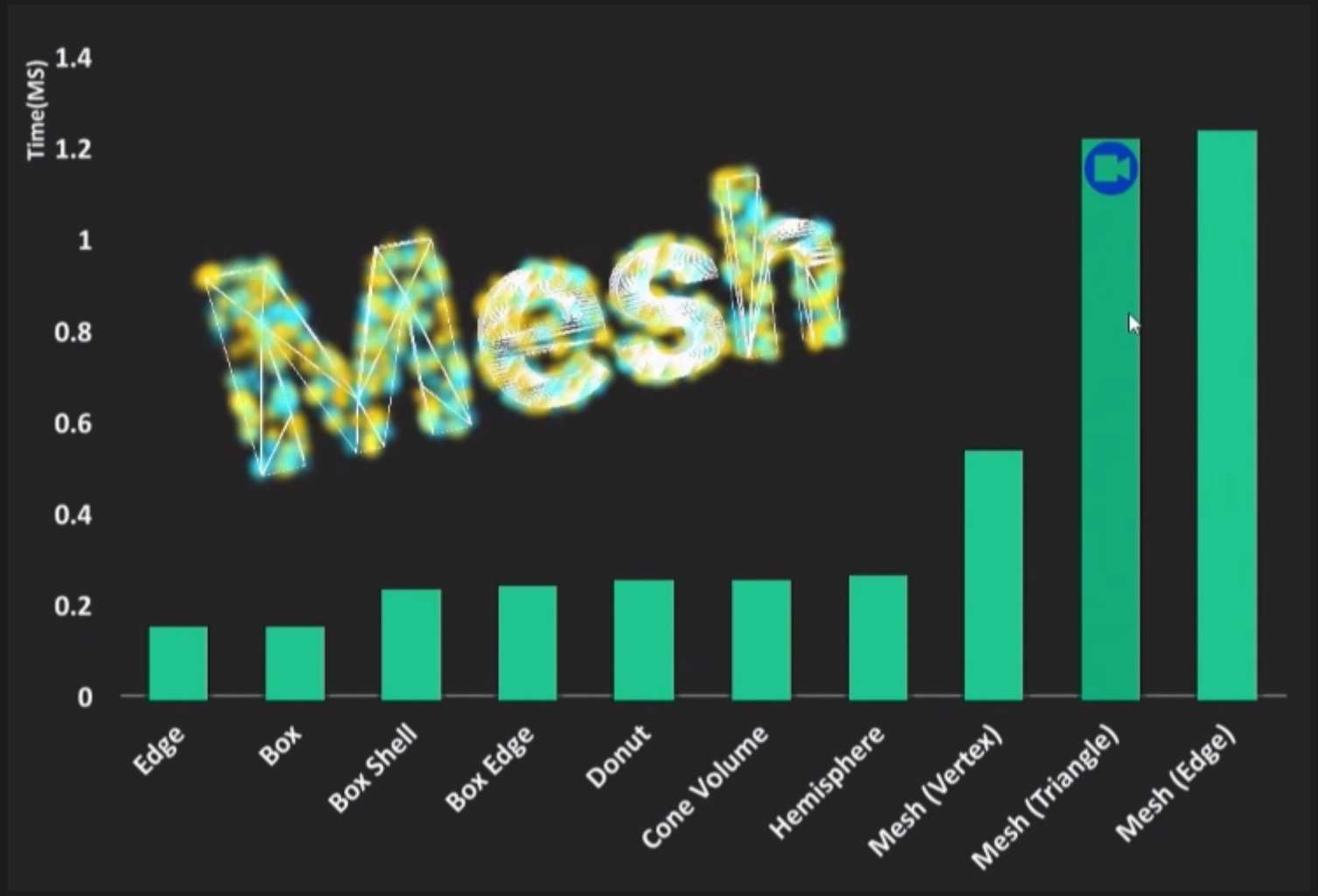




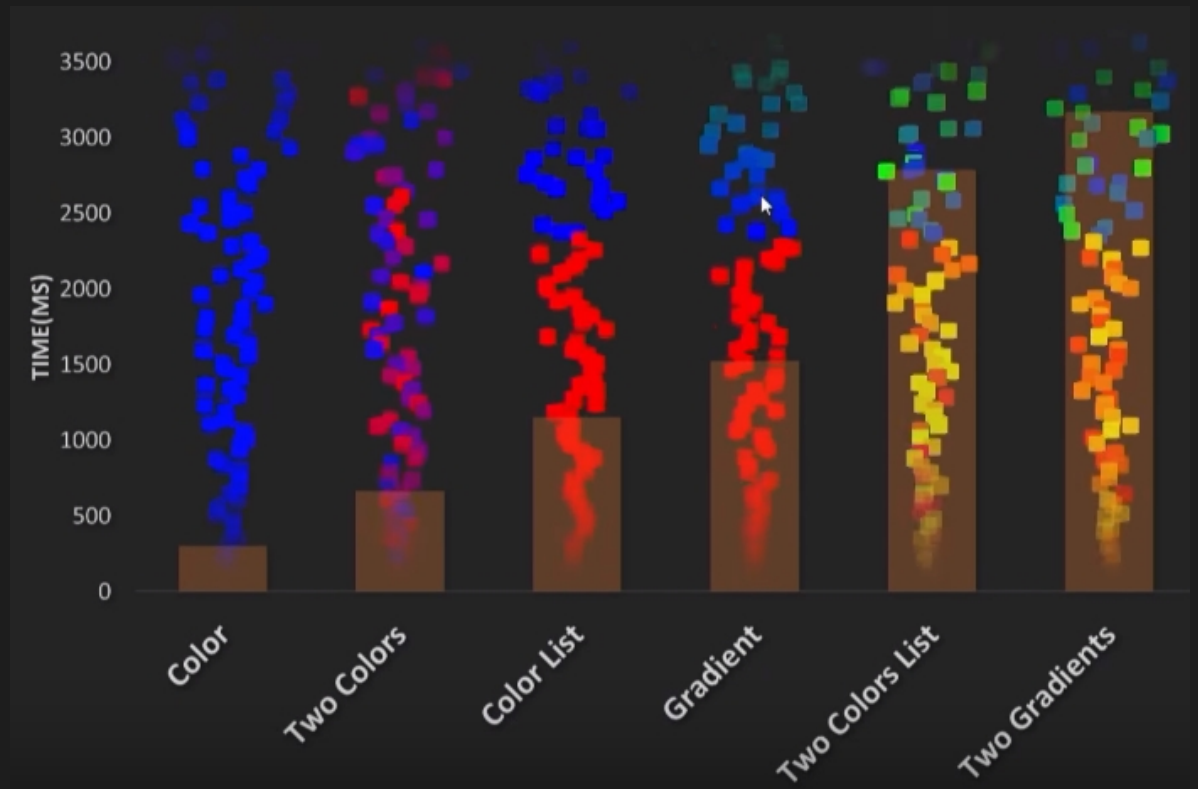


Min/Max performance based on module settings (Eg. CustomData: Vector1 VS Color – Color is more expensive to compute)

Performances



Performances

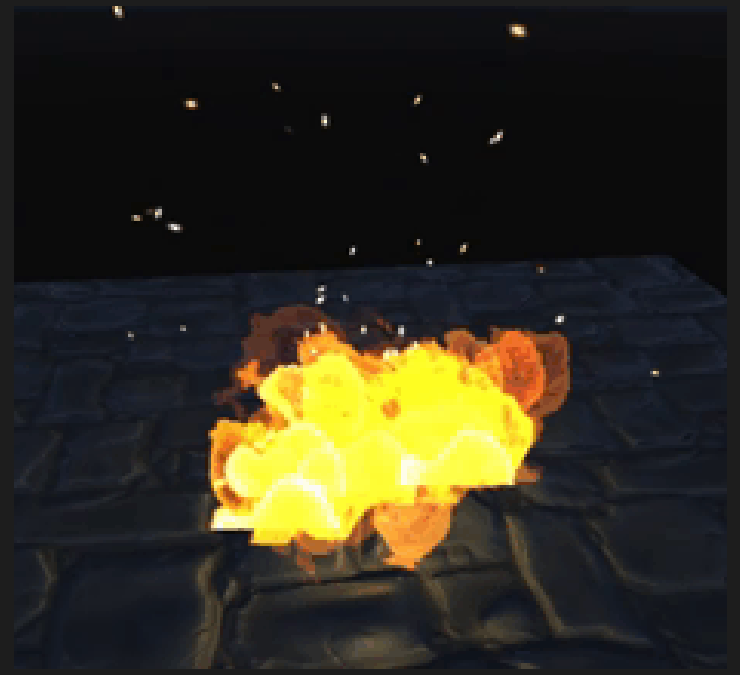


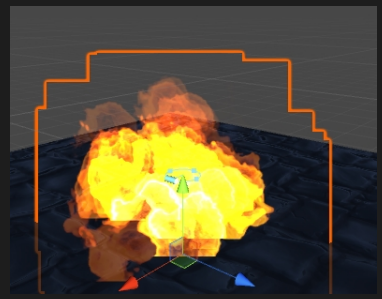
ColorList is Gradient with Mode: Fixed

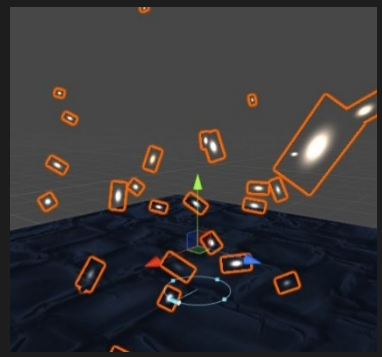
Particle Fx / Fire

We'll use 3 PS

- Flames
  - Texture Sheet Animation
  - Color/SizeOverLifetime
  - Up Velocity
  - Noise
- Ember
  - Small size
  - Noise
  - StretchedBillboard
- Light
  - Emitter at the base
  - 2 or 3 particles with ColorOverLifeTime
  - Render only light







VFXGraph

PARTICLE COUNT	Thousands	Millions
SIMULATION	Simple	Complex
PHYSICS	Underlying Physics System	• Primitives • Scene Representation (SDF…) • Depth Buffer
GAMEPLAY READBACK	Yes	No* ** Potentially small data with latency*
OTHER	-	Can read Frame Buffers

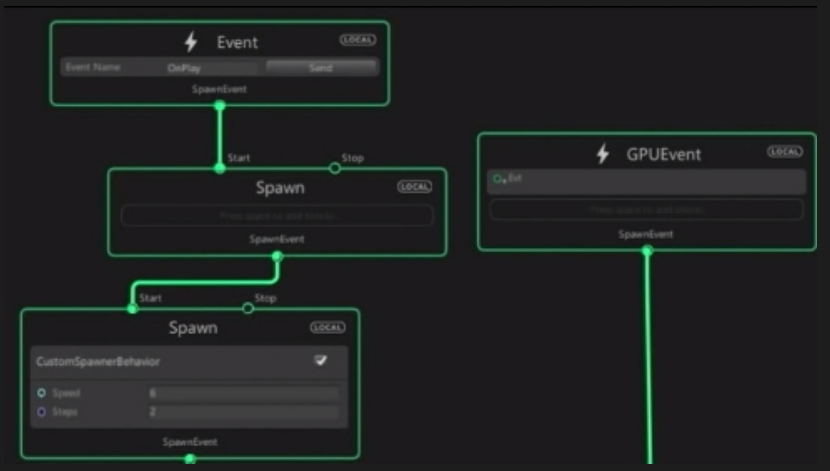
CPU (Left) vs GPU (Right)

- New tool to create VFX in realtime
- Fully progrmamable
- Creates assets with exposed Events and Parameters interface
- GPU Based
- Artist friendly

- Both CPU and GPU particles have pros and cons
- We can have a lot more particles and a much more complex behaviour on GPU, due to the parallel nature of GPU
- If you want to rely on the physics system, it resides in the CPU accessible memory, so for GPU particles you need to rely on other scene representations (primitive, sign distance fields, depth buffer)
- If you want particles interact with gameplay, go for CPU particles: reading back data from GPU to CPU is a very costly operation
- On GPU you've access to all the data in the video memery, like Framebuffers, to achieve cool screen space effects

- 3 Layers of Editing
  - Asset instance configuration You Just tweak exposed values on an already created effect
  - VFX Asset Authoring Use the VFXNode base editor
  - VFX Feature Creation/Editing

VFXGraph



- Context nodes contains Blocks, where you set your attributes
  - The flow is from top to bottom
- You can create and add nodes(s) to affect the simulation of these blocks

Context
- Spawn How many particles
- Initialize It is calles only once (Start C# method)
- Update (Update C# method)
- Output

Events
- You can trigger via Script a spawn, or spawn chains: one spawner plugged into another spawner: its spawning can cause start/stop spawning of another spawner

Output
- We can have multiple output for the same simulation (quad, line, point, mesh)

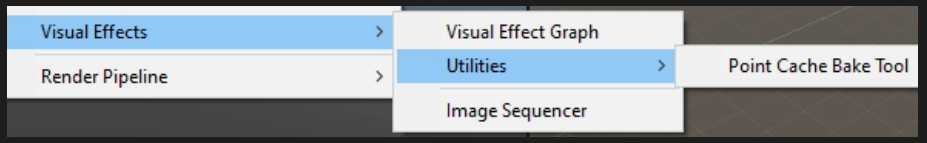
VFXGraph

- From the high level representation Unity generates a low level representation, the Expression graph, which is am abstract syntax tree (AST)
- The Expression graph goes into the VFX Compiler to generate all the runtime data
  - Shaders. Compute shaders for simulation, vs/ps for rendering
  - Bytecode for the VFX CPU interpreter. If you have expressions that have to be evaluated once per frame and not once per particle, you can do it on CPU)
  - Optimized data layout. Only necessary per-particle attributes are stored (if you don't use velocity in your system, velocity is not going to be stored)
  - PropertySheet: Input values/exposed parameters. Accessible from timeline, scripts
  - List of VFX systems descriptions. Bind everything together and issue commands to our VFX manager

VFX ToolBox

- Project that is meant to be a suite of tools to generate data that are usable within the VFXGraph/ShaderGraph/ParticleSystem
- Download from [https://github.com/Unity-Technologies/VFXToolbox]
- Follow “local Package Install” instructions
- Once installed, you should see ImageSequencer under Window/VisualEffects menu

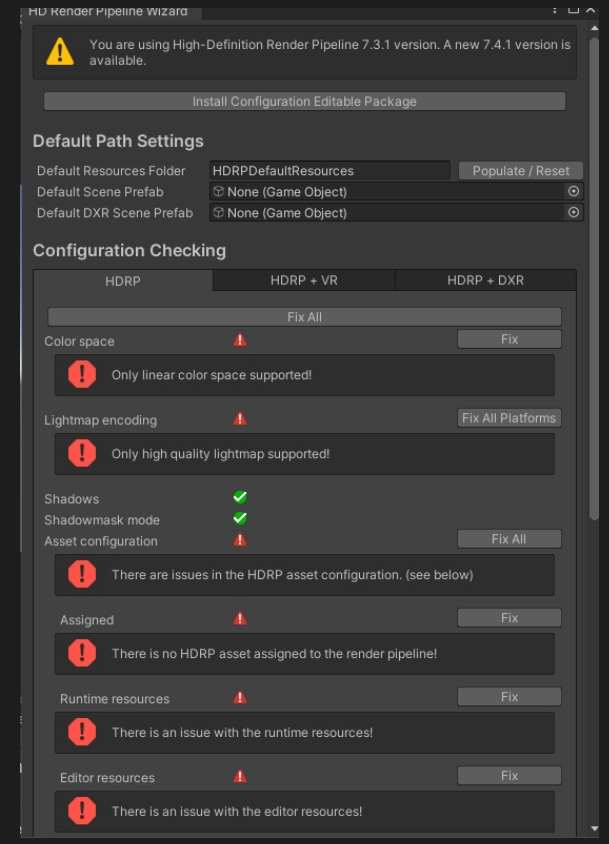
- Image Sequencer (documentation)
  - Allows to assemble flipBooks to be used to render particles
- PointCacheBakeTool
  - A PointCache is a set of points that hold attributes: positions, velocity, normal, Uvs, Custom attributes
  - This tool allows you to create PointCache from meshes, textures
- External DCC (Houdini) to export
  - Vector fields
  - Point Caches

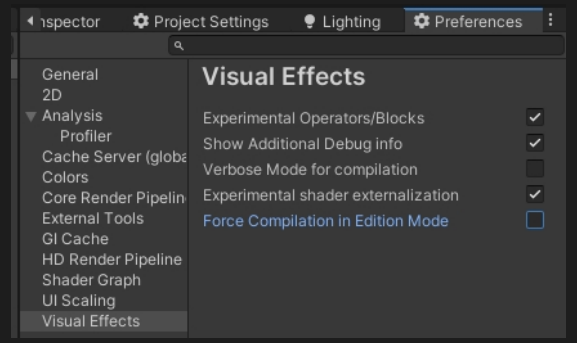


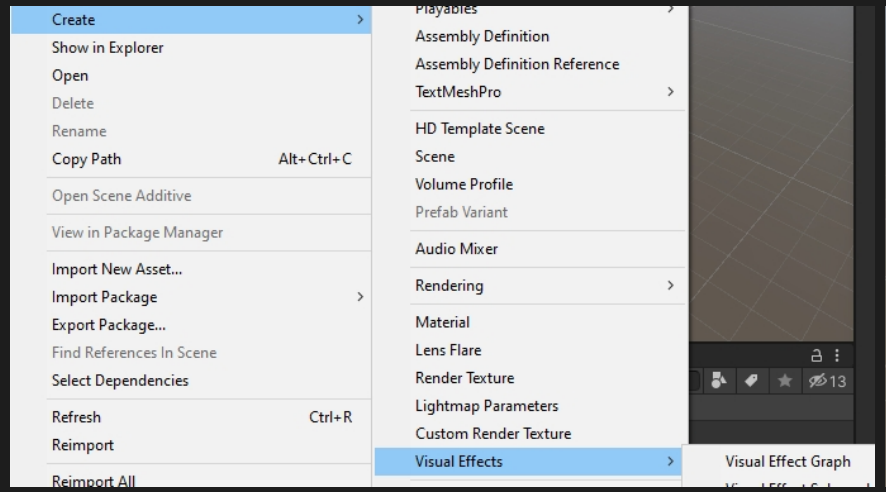
- In the future
  - It will remain independent from VFXGraph, because the generated assets can be used anywhere in Unity
  - UnityPackage
  - More DCC integration (May, Blender)
  - Motion Vectors from optical flow in image sequencer
  - Vector Fields, SignedDistanceFields, FlowMapPainter
  - VFXGraph will be able to spawn CPU particles (using ECS system, replacing Shuriken system)
  - At the moment, URP supports only UnlitShaders particles. In the future, it will be supported

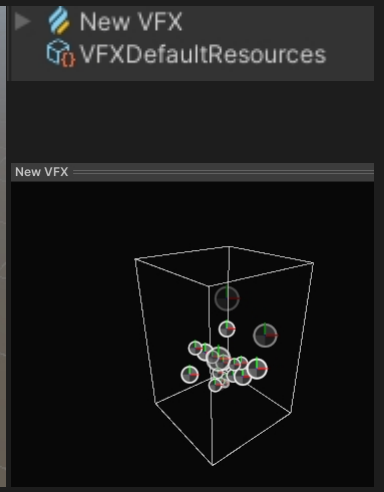
VFXGraph

- Create a new project
- PackageManager / High Definition RP
- Once installed, HDRenderPipelineWizard should appear: Choose FixAll
- Create a new HDRenderPipelineAsset
- Create a new HD default scene
- Enable Preferences/VisualEffects/ExperimentalOperators-Blocks
  - This will let us use Events and Triggers nodes
- Create/VisualEffects/VisualEffectGraph
  - This will create a new .vfx asset we can drag into the level







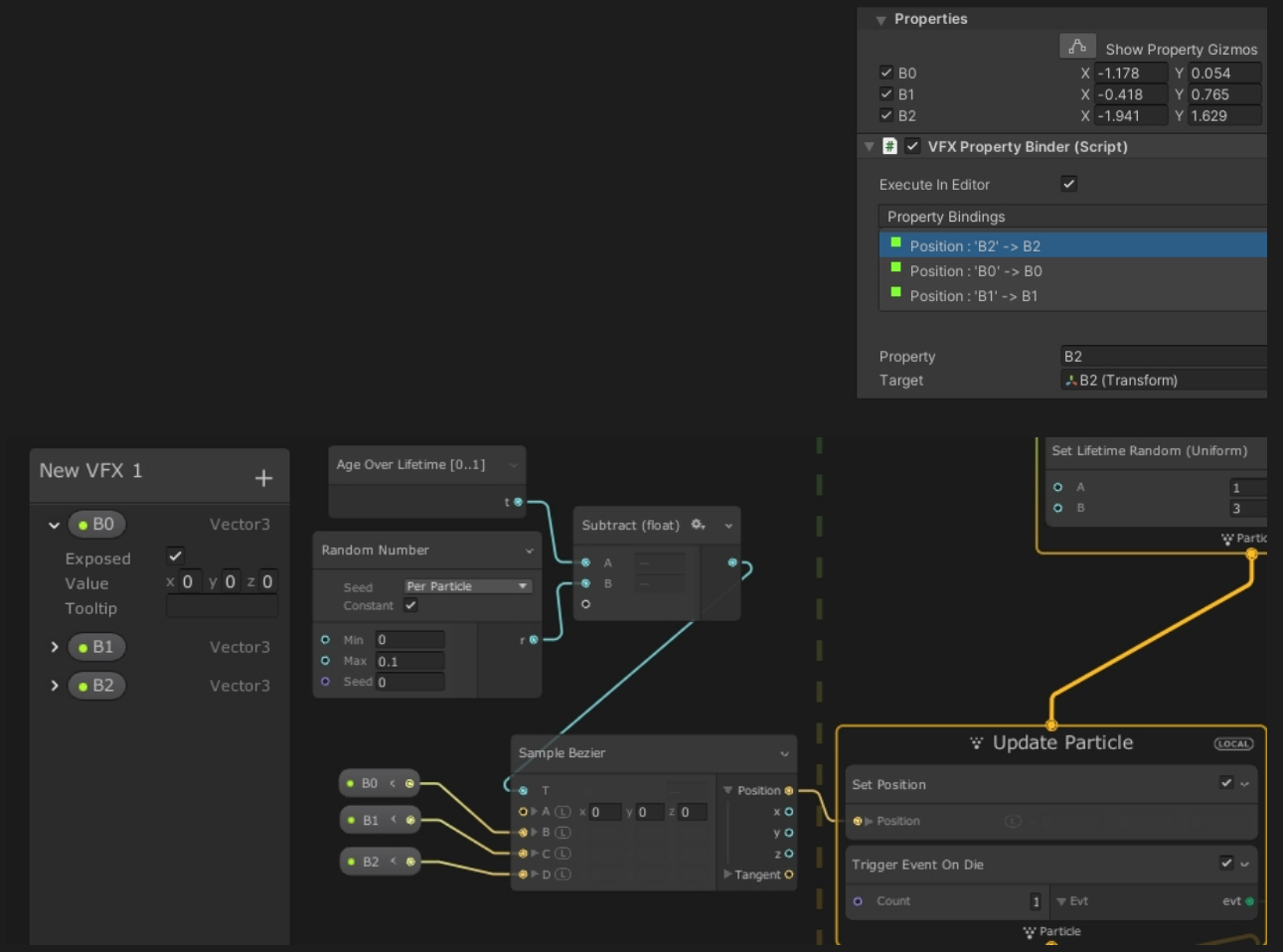


VFXGraph

Try it

- Create a new VFXGraph assets and drag it in the hierarchy, under BezierSubEmitterEvents
- Add a SetPosition Block in Update Context
- Create a BezierCurve and expose 3 Vector3 values: B0,1,2
- In the inspector, add VFXPropertyBinder component at the same level of the VisualEffect component
- Bind Scene Objects Transforms B0,1,2 to VFXPropertyBinder Position values B0,1,2
- Now if you move B0,1,2 Transforms in the scene, you'll change B,C,D arguments of the SampleBezier curve node (A argument is 0,0,0, which is where the VisualEffect component is)

[VFXGraph_Basics_00]



VFXGraph

Try it

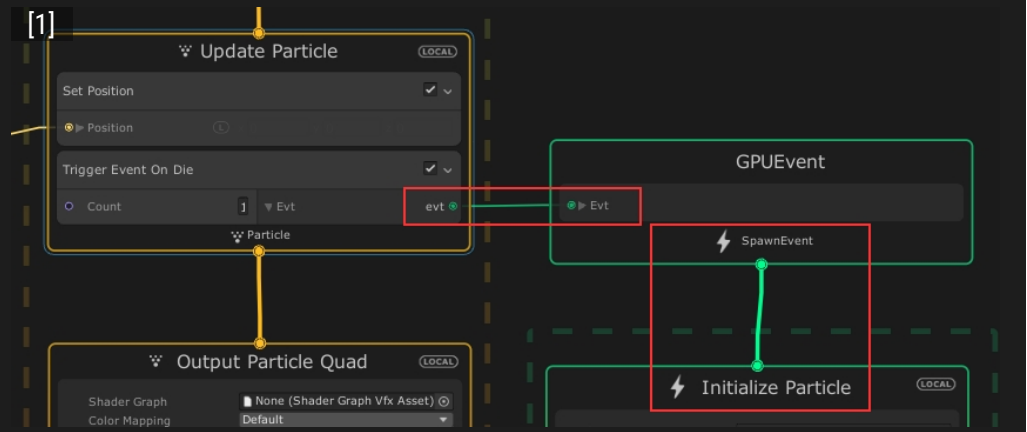
- Create a new SimpleParticleSystem System S2 (Add node/System)
- Add a TriggerEventOnDie Block to UpdateContext, and link it to a GPUEvent and then to the new SimpleParticleSystem [1]
- Now as soon as a particle dies, the other PS Particles are spawned
- Add 2 Event Nodes (with event names Start/Stop) and connect them to Start/Stop Spawn input [2]
- We need to trigger those event from the Scene: add 2 VFXMouseEventBinder components to Start/StopMouseEvents GameObject in the scene: Bind them with a MouseEnter/Exit from this cube
- Now at runtime if the mouse enters the cube the PS is started, and as soon as it exits the PS is stopped!
- How to trigger this events during EditTime?
- Show TargetGameObject Panel in the VFXGraph, then select the Visual Effect GameObject in the hierarchy: a list of events should appear inside the VFXEditor! [3]

Events

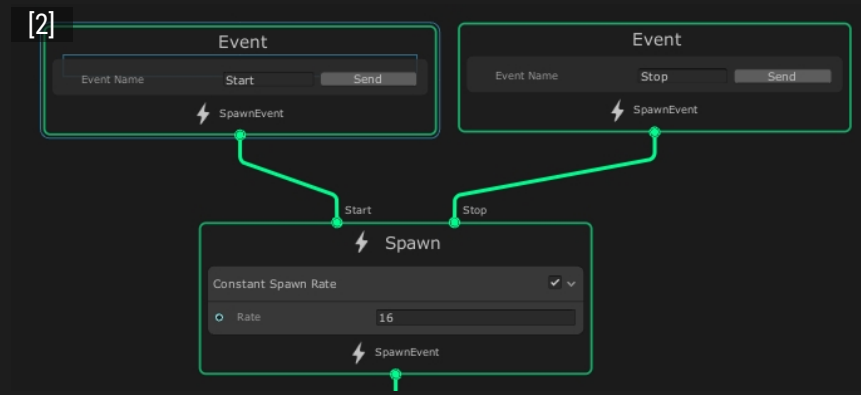
- You can have an “Empty effect”: acts just as a container for all the effects you want to trigger, and you hook them up to the same event

[VFXGraph_Basics_00]

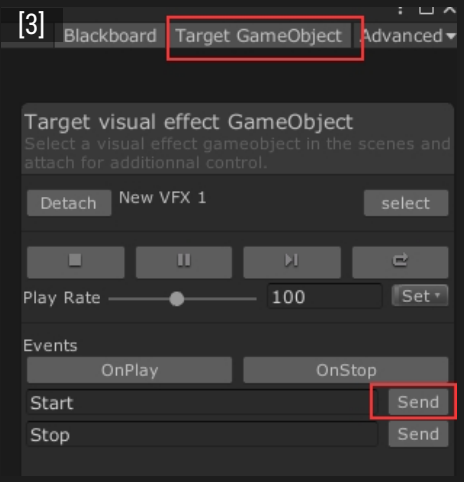
[1]



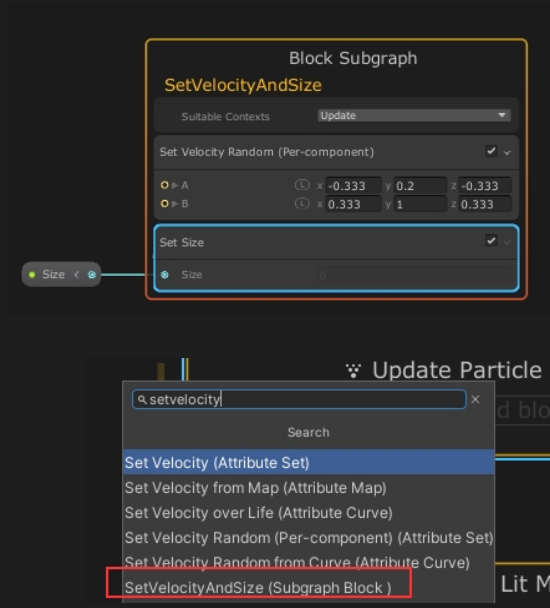
[2]



[3]



SubGraphOperator/Block



SubGraphOperator
Try it

- Create 3 Random Nodes, from 0 to 1
- Add an AppendVector Node
- Add a Vector3 Entry in the BlackBoard, and move it under the Output tab
- Connect AppendVectorOutput with this output
- Create a Scale input and use in a Multiply Node
- Save it: now you can use this subgraph as a graph operator!

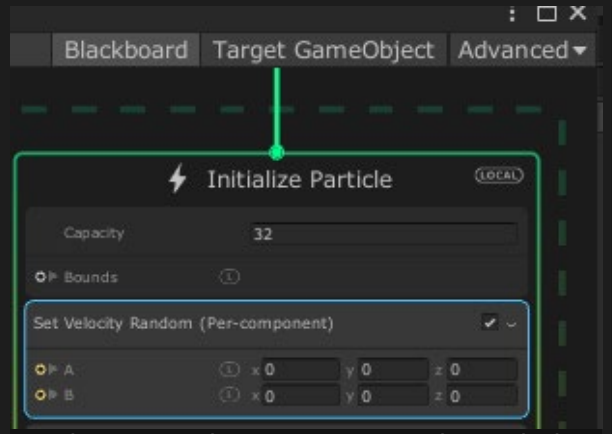
SubgraphBlock
Try it

- RightClick on Update/SetVelocityBlock, ConvertToSubGraphBlock
- Now add SetSize Block and expose Size Value in the blackboard
- Save it and now in Update Context you'll be able to add this Subgraph!
- The name of the new Block is the name of the created asset

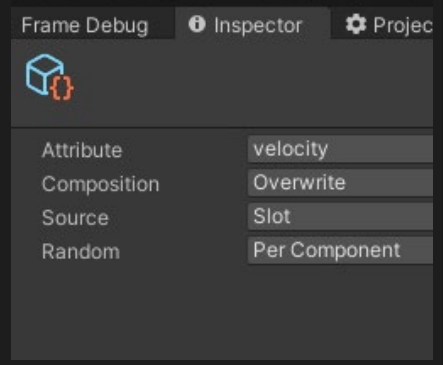
[VFXGraph_Basics_00]

Inspector

- Ispector panel changes accordingly to the selected Context or the selected Block in the VFXGraph



By selecting Inizialize Context or SetVelocity Block, The inspector changes



VFXGraph / ShaderGraph integration

- Create a VFXShadergraph
- The master node used is a VisualEffectMaster. Choose the LitVersion if you use a LitMeshOutput Context n
  VFXShaderGraph