COMP1150/MMCC1011 Week 11 Prac

Topics covered:

- Particle effects

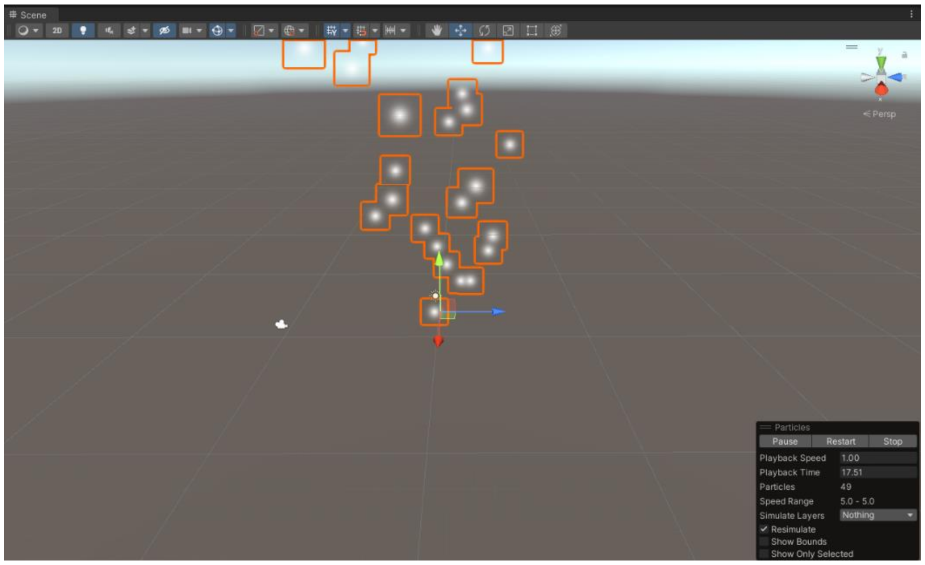
Particle Effects

Particle systems are a useful tool for simulating animated systems that consist of many small similar particles in motion, such as fire, rain, smoke, dust, 'magic' energy effects, and many other things. While a simple idea in principle, used creatively particle systems can produce a wide range of effects and add a lot to the 'feel' of a game.

The idea behind particle effects is fairly simple: An effect is created using one or more **particle systems**. A system consists of an **emitter** and a bunch of **particles**. A particle is usually just a small 2D image (such as a drop of rain or a puff of smoke) drawn in the 3D world. Particles have a limited lifetime. They are created by the emitter and disappear, usually after just a few seconds. Individual particles are usually very simple, but they are cheap to draw so we can have hundreds or thousands of them on the screen at once. The art of a good particle effect comes from the layering of many particles at once.

Creating a Particle System

Create a new empty scene in Unity in your COMP1150-3D-Pracs project, just as you did for Lighting last week (you can call this scene 'Particles'). From the menu select **GameObject > Effects > Particle System** to add a particle system to your project. Unity adds a default particle system and immediately starts animating it. You should see a fountain of white particles.



Playback Speed	1.00
Playback Time	17.51
Particles	49
Speed Range	5.0 - 5.0
Simulate Layers	Nothing

[✓] Resimulate
[ ] Show Bounds
[ ] Show Only Selected

The ‘Particles’ panel in the bottom right corner lets you control the animation of the effect in the Scene view. This has no effect on the game, it is just there for you to examine your system while creating it. If you press **Pause** in the panel you can change the **Playback Time** value to see what the system would look like at any particular point in time. Click on the **Playback Time** label and drag the mouse left and right to scrub back and forwards in time.

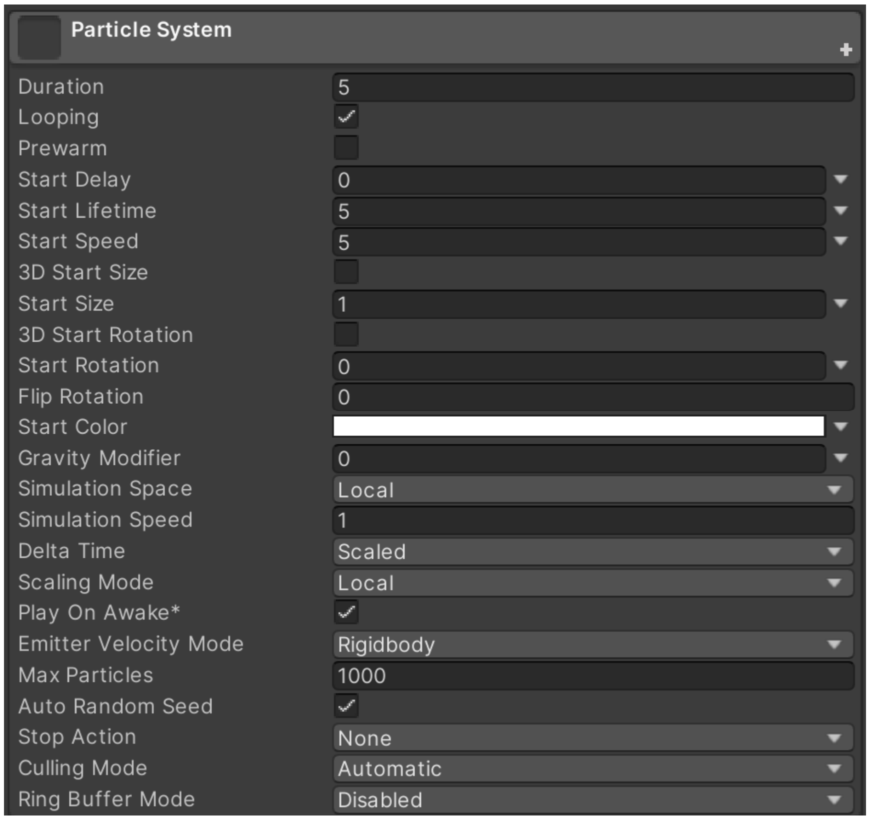
There are some things to note about this system:

- Each individual particle is a simple 2D image, which by default is **billboarded** (similar to grass from the Terrain prac) so that it always faces the camera.
- Particles are created at the **emitter**, which (in this case) is the bottom of the fountain.
- Particles move and **evolve** over time.
- Particles are **destroyed** after a fixed ‘lifetime’.

The way particles are created and evolve over time can be changed in a wide variety of ways. Unity’s particle system is quite sophisticated. We won’t go into every last detail in this prac, but there are several tutorials available online. For this prac we will concentrate on three common effects: fire, explosions and rain, before letting you experiment more freely with the system.

The Components of a Particle System

Before we start making something, take a moment to examine the particle system in the Inspector view.


Particle System

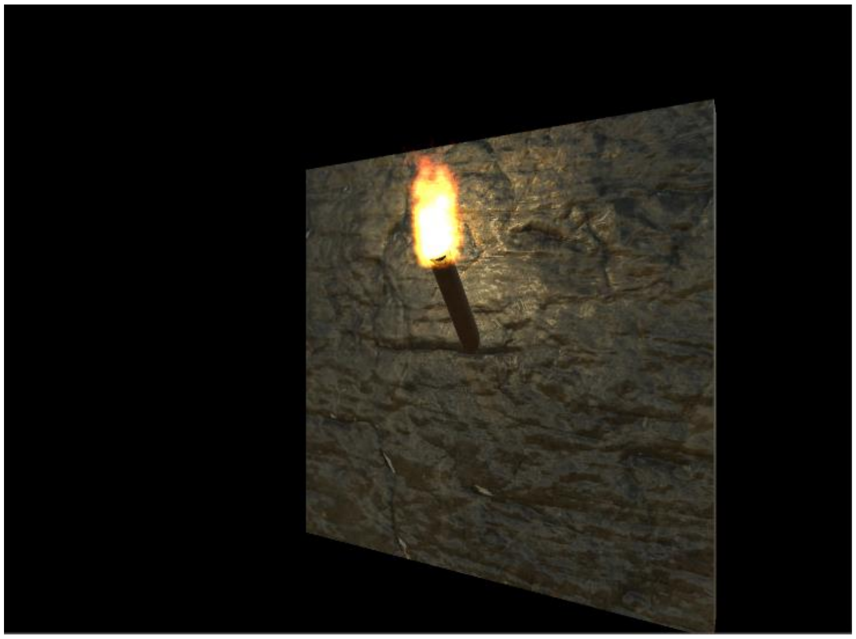
Duration	5
Looping	✓
Prewarm	
Start Delay	0
Start Lifetime	5
Start Speed	5
3D Start Size	
Start Size	1
3D Start Rotation	
Start Rotation	0
Flip Rotation	0
Start Color	⬜
Gravity Modifier	0
Simulation Space	Local
Simulation Speed	1
Delta Time	Scaled
Scaling Mode	Local
Play On Awake*	✓
Emitter Velocity Mode	Rigidbody
Max Particles	1000
Auto Random Seed	✓
Stop Action	None
Culling Mode	Automatic
Ring Buffer Mode	Disabled

You will see that there are a bunch of settings at the top followed by a list of **modules** that can be activated and deactivated (by checking the circle to the left of the module name). By default only the **Emission**, **Shape** and **Renderer** modules are activated. Each module controls some aspect of the particles' evolution. We will activate particular modules as we need them.

Making a Fire

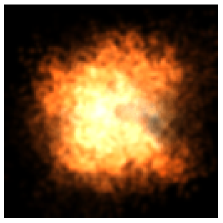
Fire is exactly the kind of thing that particle effects were invented for. It doesn't have a solid form, so modelling it with a 3D mesh would not make a lot of sense. It is better to think of a fire as a lot of small glowing particles rising from a source, changing colour, then disappearing.

First let's try to get the shape right. Let's suppose we're modelling the flame from an RPG-standard: a flaming torch. Ultimately, we'd want it to look something like the following image.



Choosing a particle material

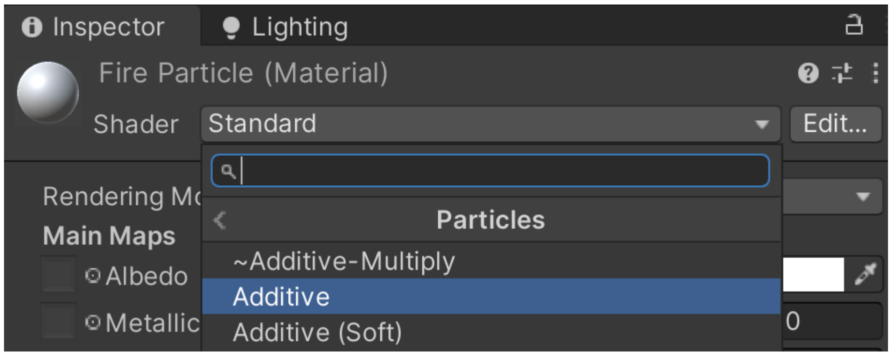
First we need to choose an image for our individual particles. The default particle is a circular blob with faded edges. This actually isn't a bad building block for many effects, but we'll start with something more fire-specific. Download the Flame.png image from iLearn and import it into your project.



To make this texture into a particle, create a new **material** (**Assets > Create > Material**) and call it "**Fire Particle**". Particle materials are simpler than the standard material we use for texturing objects. We do not need to worry about bump maps or shininess or any of the features we used before. We just want to draw the particles onto the scene as quickly as possible.

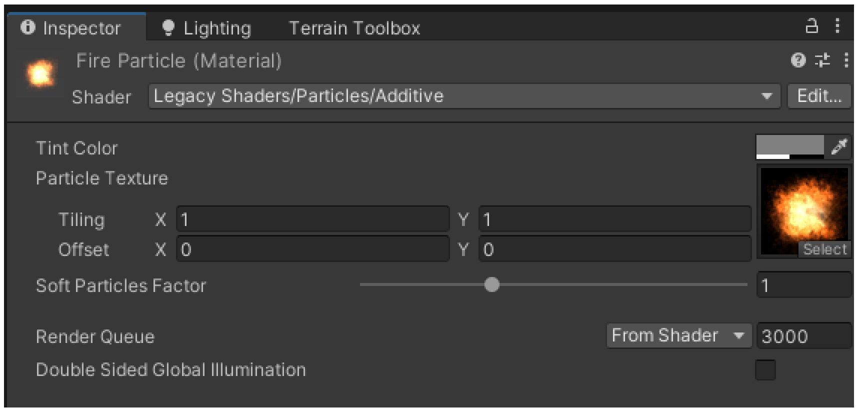
To do this we need to change the **shader** used by the material. Shaders are pieces of code run by the computer's graphics card to render objects. Unity's standard shader is quite sophisticated, but therefore also relatively slow. To render lots of particles, we want a simpler, faster shader.

Click on the **Shader** dropdown in the material (at the top) to get a list of available shaders. Select the **Additive** legacy shader (**Legacy Shaders > Particles > Additive**).



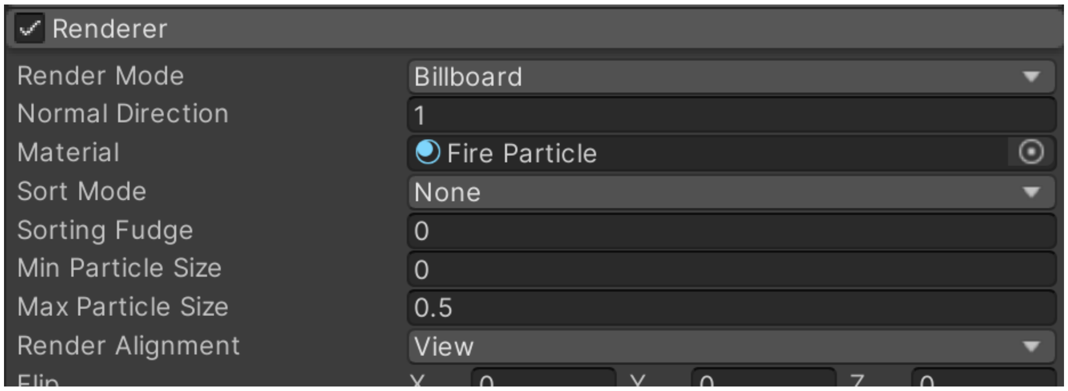
The additive shader adds the texture colours on top of whatever is behind it. With this shader we can draw lots of particles over each other and they will add together for a brighter effect.

Select the **Flame.png** texture in the **Particle Texture** box (or drop the texture onto the box).



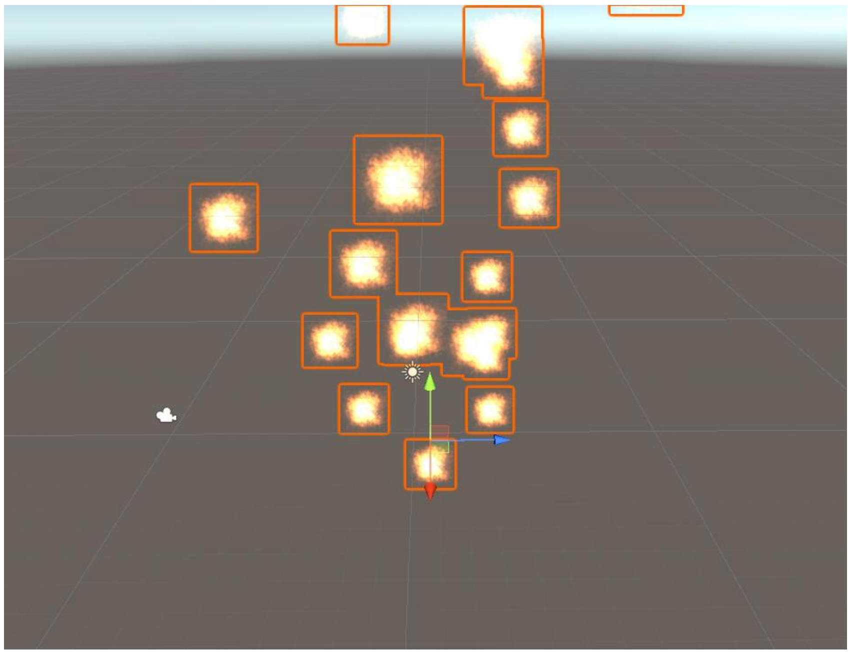
Shader: Legacy Shaders/Particles/Additive
Tint Color
Particle Texture
Tiling X 1 Y 1
Offset X 0 Y 0
Soft Particles Factor 1
Render Queue From Shader 3000
Double Sided Global Illumination

This material is now ready for you to use in the particle system. Open the particle system in the Inspector and click on the **Renderer** module to open it. You should see various options for rendering particles. In particular, there is a **Material** setting which is currently set to Default-Particle. Change this to use your new **Fire Particle** material.



✓ Renderer	
Render Mode	Billboard
Normal Direction	1
Material	Fire Particle
Sort Mode	None
Sorting Fudge	0
Min Particle Size	0
Max Particle Size	0.5
Render Alignment	View
Flip	X 0 Y 0 Z 0

Your particle system should now start emitting fire particles.

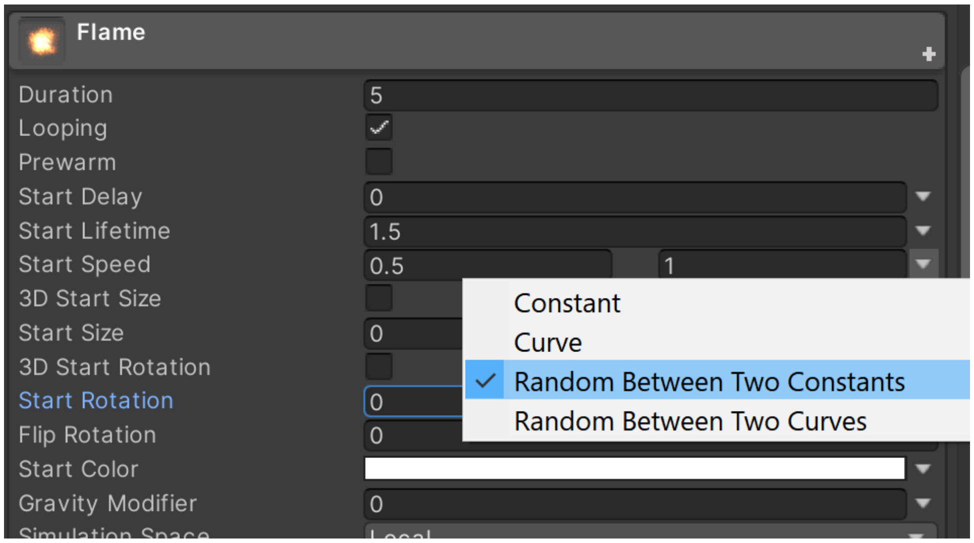


Changing the size and shape

Obviously this still looks nothing like a flaming torch. The particles are too spread out and the shape is wrong. Let's start by making it smaller. The overall size of the effect is determined by the **lifetime** of the particles. The longer they hang around, the taller the effect will be.

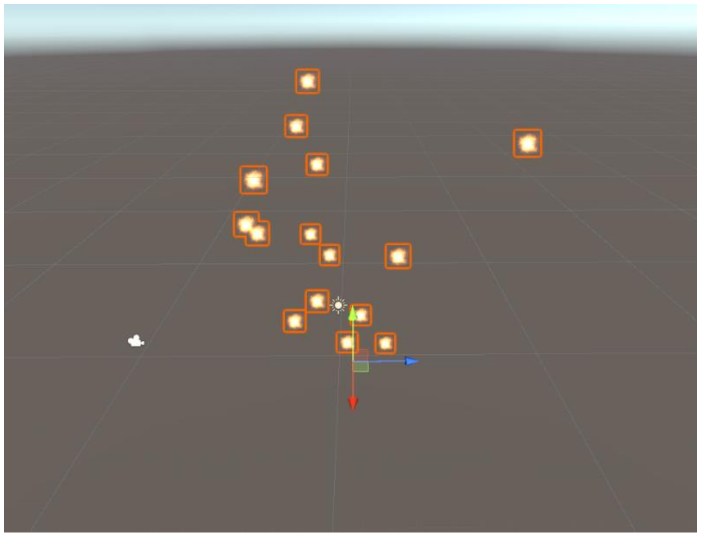
The particle lifetime is one of the standard settings at the top of the 'Particle System' component. You want your particles to start disappearing much sooner, so change the **Start Lifetime** from 5 to **1.5**. This is closer to the flame size we want. You can tweak this number later to get the exact value you want.

We can also make the individual particles smaller by setting the **Start Size** to 0.5. Better yet, we can have Unity *randomise the sizes*. Click the little arrow to the right of Start Size and select **Random Between Two Constants**.

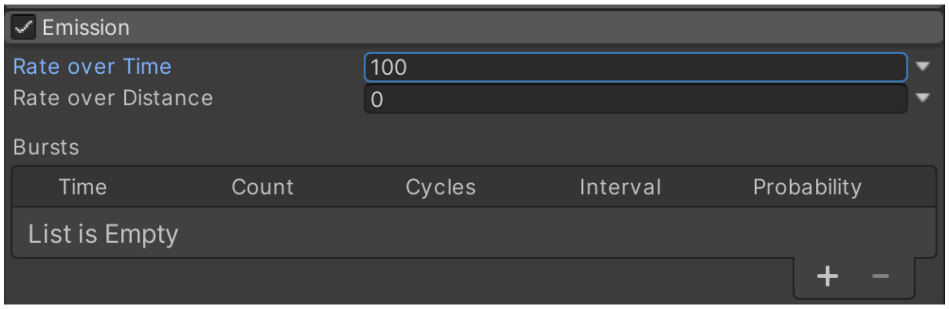

Flame

Duration	5
Looping	✓
Prewarm	
Start Delay	0
Start Lifetime	1.5
Start Speed	0.5 / 1
3D Start Size	
Start Size	0
3D Start Rotation	
Start Rotation	0
Flip Rotation	0
Start Color	
Gravity Modifier	0
Simulation Space	Local

You should now see two boxes next to Start Size, for the minimum and maximum sizes. Unity will randomly choose a value between these two limits for each particle.



Next we want to add a lot *more* particles. The effect we want is not to see individual flame particles, but to add a lot of them together to make the appearance of a single flame. Open the **Emission** module and change the rate from 10 (particles per second) to 100.

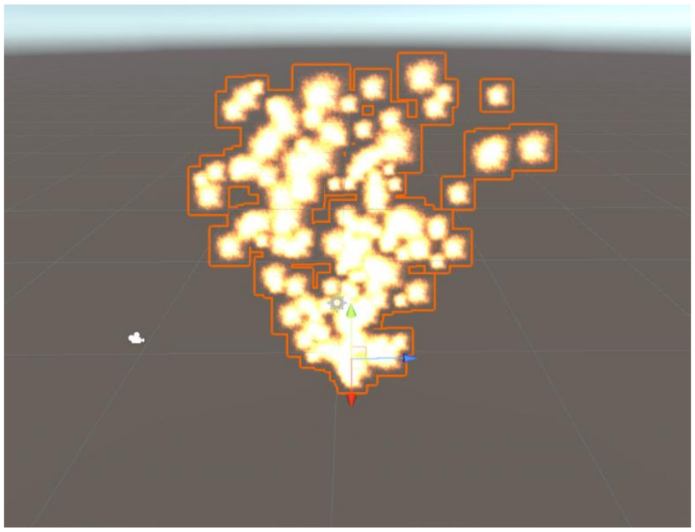

✓ Emission

Rate over Time	100
Rate over Distance	0

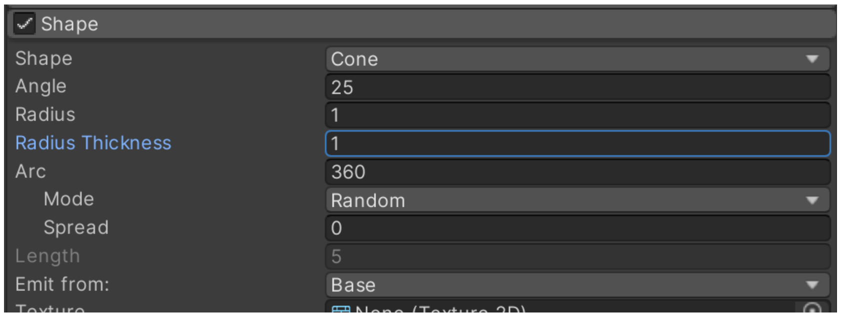
Bursts

Time	Count	Cycles	Interval	Probability
List is Empty				

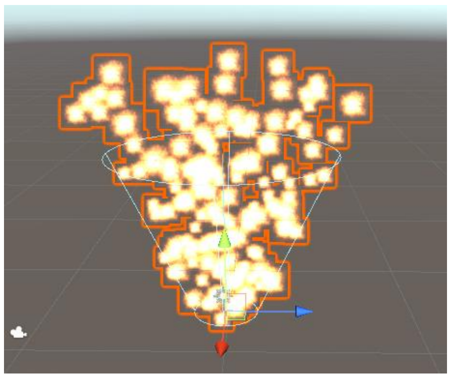
This gives us many more particles but they are still too spread out. We want to make the cone of particles smaller and narrower.



Open the **Shape** module. You will see settings for controlling the shape of your particle emitter. You can also see the emitter drawn in the Scene view:

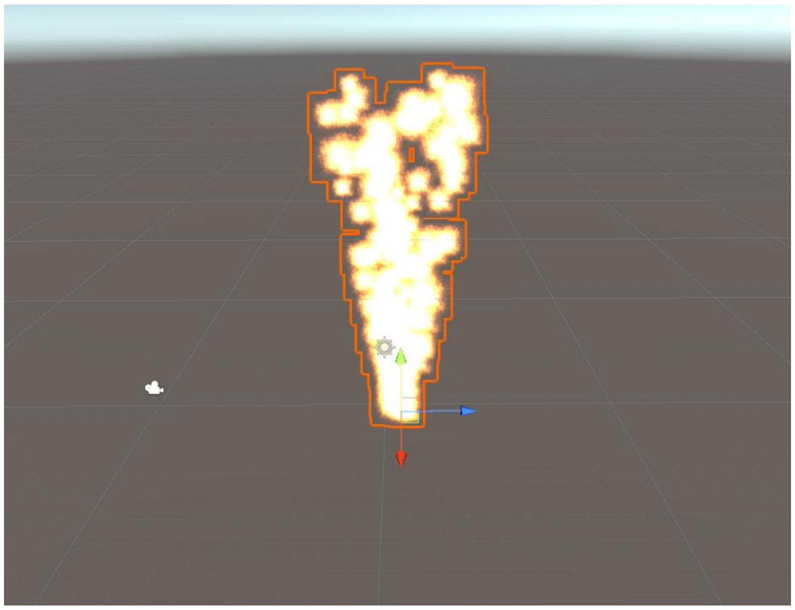


✓ Shape	
Shape	Cone
Angle	25
Radius	1
Radius Thickness	1
Arc	360
Mode	Random
Spread	0
Length	5
Emit from:	Base



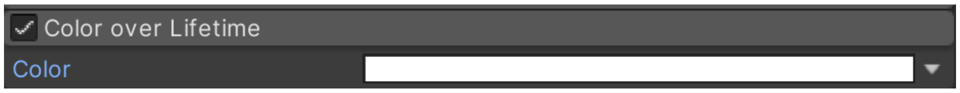
The emitter is the device creating the particles. The size and shape of the emitter determines the initial distribution of particles. The default emitter is a ‘cone’. Particles are created at the base of the cone and move in an upwards direction, expanding outwards with the cone.

To make the cone narrower, we want to decrease the **Angle** and **Radius** of the cone. Set the **Angle** to 10 and the **Radius** to 0.2. Your particles should now emit into a tighter cone.



Evolving Particle Size and Colour

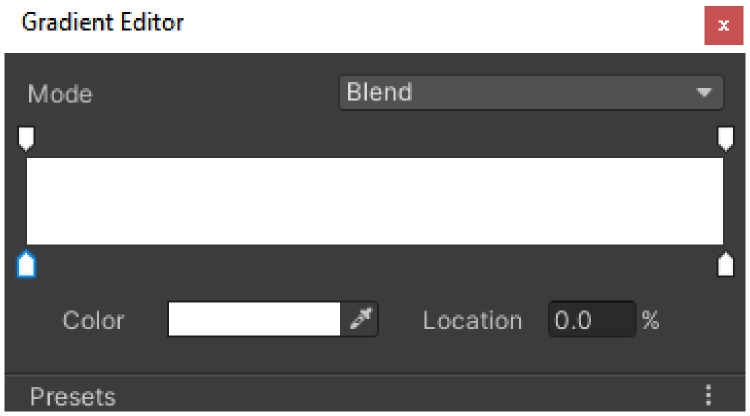
The effect is starting to look better but is still too big and bright. A real flame is brightest at its source and then fades as it grows higher. It also changes colour from bright white/yellow to red. We can create both of these effects by activating the **Color Over Lifetime** module:



✓ Color over Lifetime

Color

This module changes the particles colour over time. If you click the **Color** box you will be presented with Unity's 'Gradient Editor'. This allows you to specify a gradient blend between different colours over time.



Gradient Editor

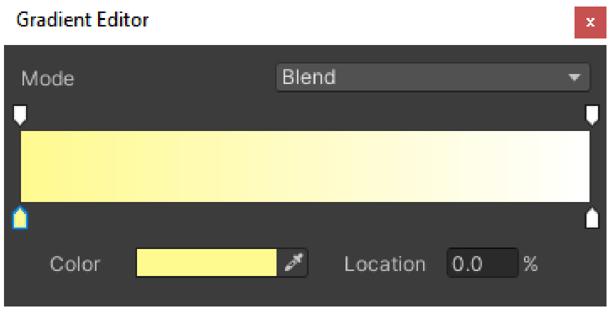
Mode: Blend

Color

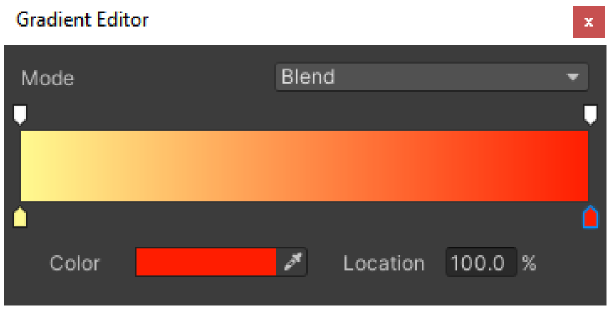
Location 0.0 %

Presets

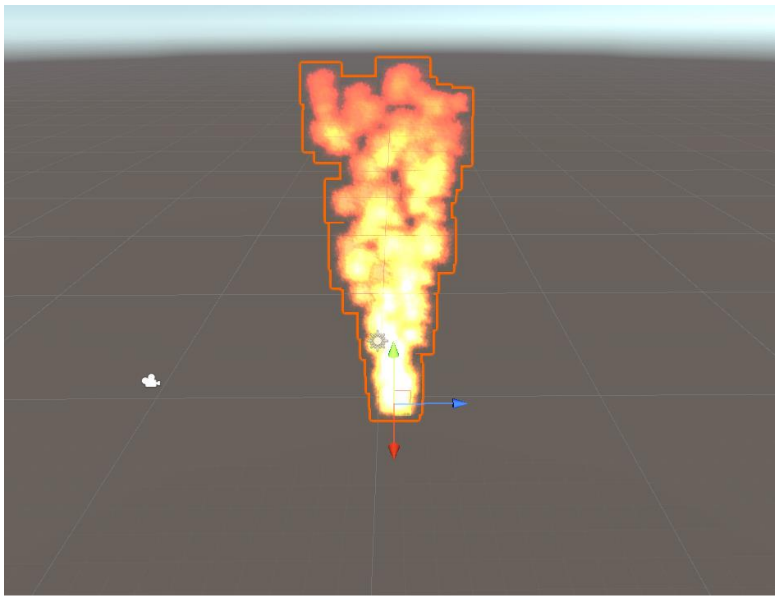
Initially the whole gradient is white, but you can change this by selecting the little arrow tabs **beneath** the colour box. First select the one on the left (as shown above). This is the starting colour for the particles. Click the **Color** box and select a bright, medium-saturation yellow colour.



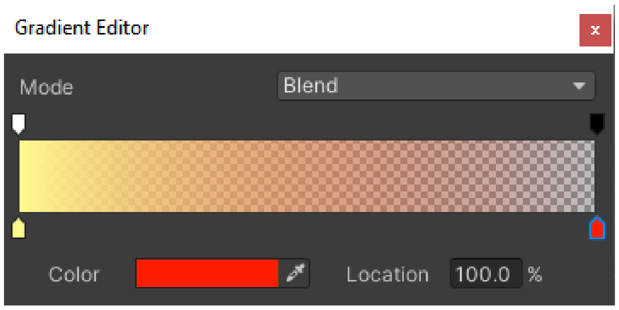
Now click the arrow on the right and select a bright red colour.



You should now notice your flame particles turning from yellow to red as they rise.



The particles at the top are still too big and bright. We want them to fade away. We can do this by setting the ‘transparency’ (aka **alpha**) value in the gradient editor. Select the arrow on the top right of the gradient box. This arrow controls the **Alpha** value. It is initially 255 (completely opaque) set it to **0** (completely transparent).

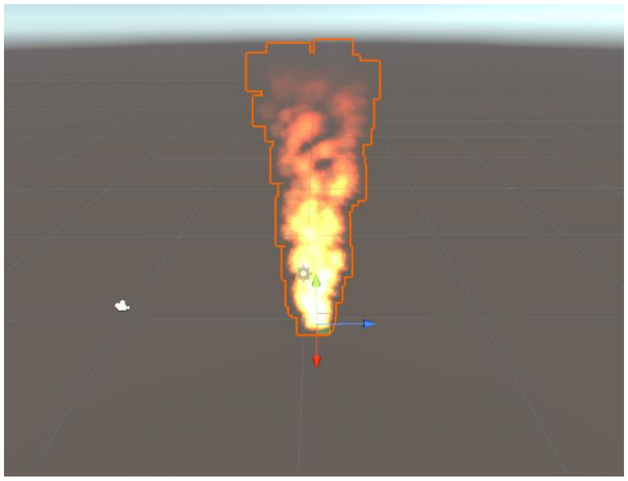


Mode: Blend

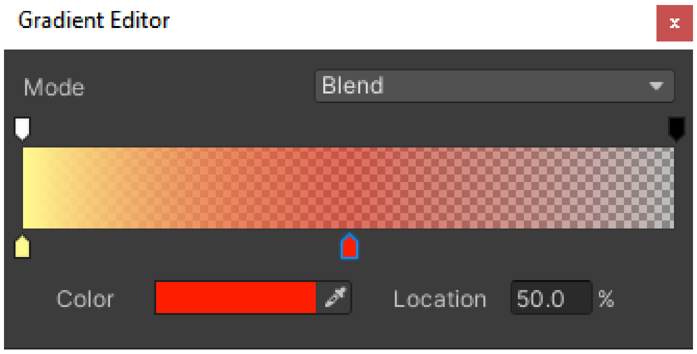
Color

Location 100.0 %

You should now notice your flame is fading out as it rises.



Some of the redness is now lost as the particles fade. You can move the red node from the end of the gradient box to the middle by clicking and dragging it. This will allow you to make the red transition happen before the particles disappear.



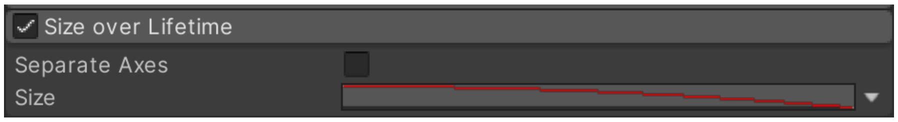
Mode: Blend

Color

Location 50.0 %

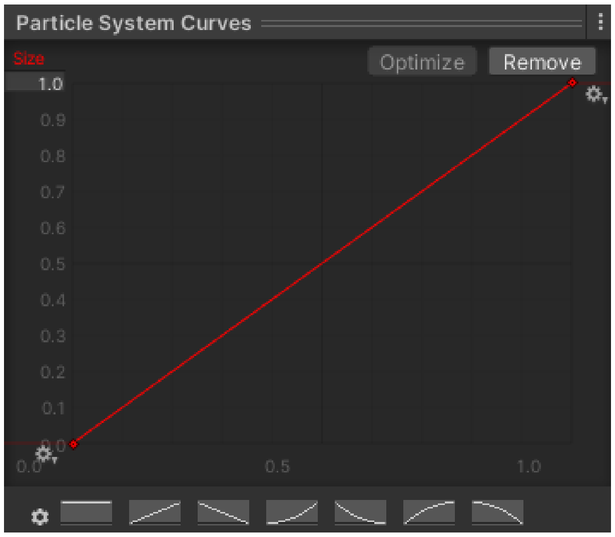
You can also add other nodes by clicking anywhere along the bottom (for colour) or the top (for gradient) of the box. Experiment with different shades or timings for your colour transitions.

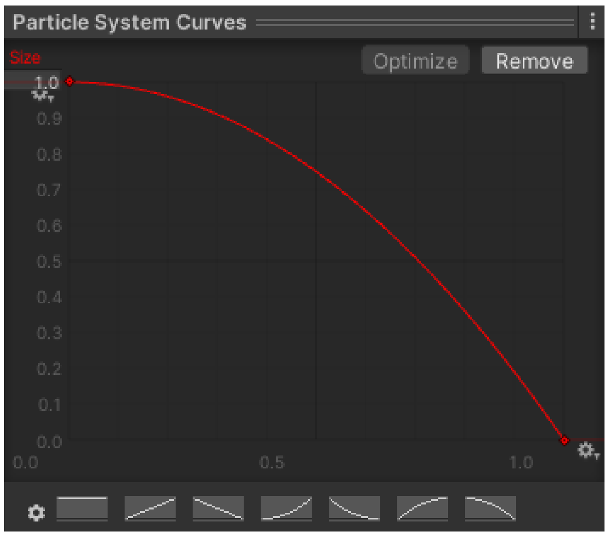
We can also make the size of the particles change over time. Activate the **Size Over Lifetime** module.



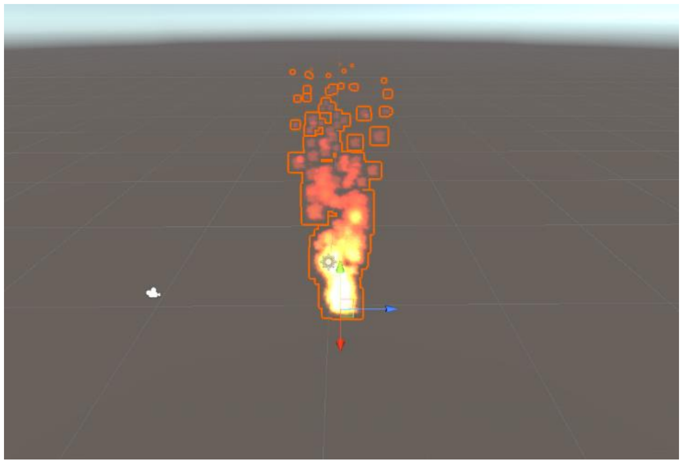
The ‘Size’ box contains a curve which you can edit in the ‘Particle System Curves’ preview at the bottom of the Inspector. You may need to drag this area up using the grip handle next to its title to reveal the preview (or if this is still too hard to see then you can click the three dots and select Convert to Floating Window).

Click on the **Size** curve and you should see that initially the curve is a straight line. The size starts at 0 and transitions linearly to 1 (as shown on the left below). You can change this by selecting one of the standard curves at the bottom (or edit manually by dragging the nodes). **Change the line into a curve** which decays from 1 to 0 (as shown on the right).



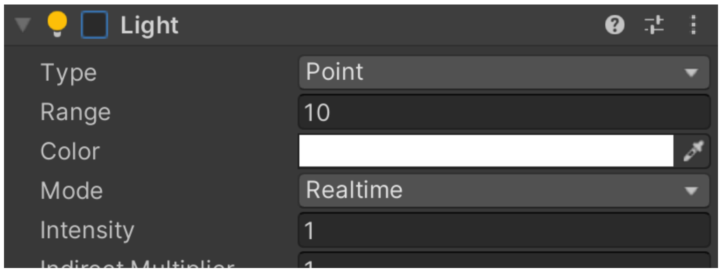


You should now notice your particles shrinking as they get higher.



This is getting closer to a more realistic flame effect. From here it is a matter of tweaking the parameters until it feels right. In the example the **Start Speed** (in the first module of the particle system) is lowered from 5 to 3, making the flame smaller and less vigorous. In the **Emission module**, a burst of 20 particles was added at Time 0.0 so that there are occasional flares of light. Overall, this amounts to a relatively realistic effect, but you can adjust your settings to suit your preferences.

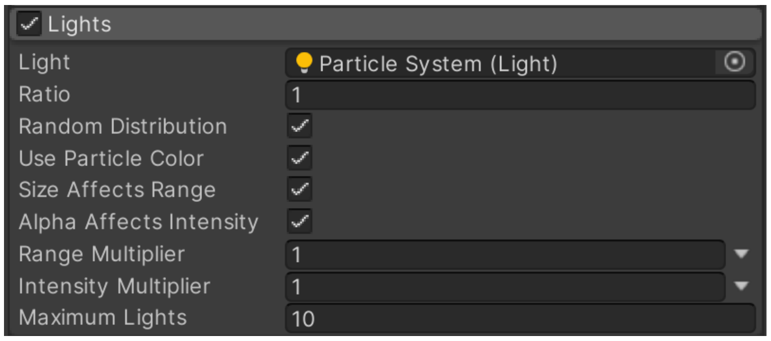
To add light to the particle system scroll down the bottom of the inspector and add a light component (**Add Component** and then **Rendering > Light**) and change its type to **Point** light. This will add a light component to the game object that your particle system belongs to, so that it can be used as a reference for the kind of light you want your particles to create. For now, you can disable this light (by unticking the checkbox next to the title of the component).



Light

Type	Point
Range	10
Color	⬜
Mode	Realtime
Intensity	1

Then turn on the '**Lights**' module of the Particle system. Within the lights module, click the object picker button next to **Light**, select the **Scene** tab, and **attach your particle system's light to itself**. You also want to change the maximum amount of lights and tick all of the options (as below).



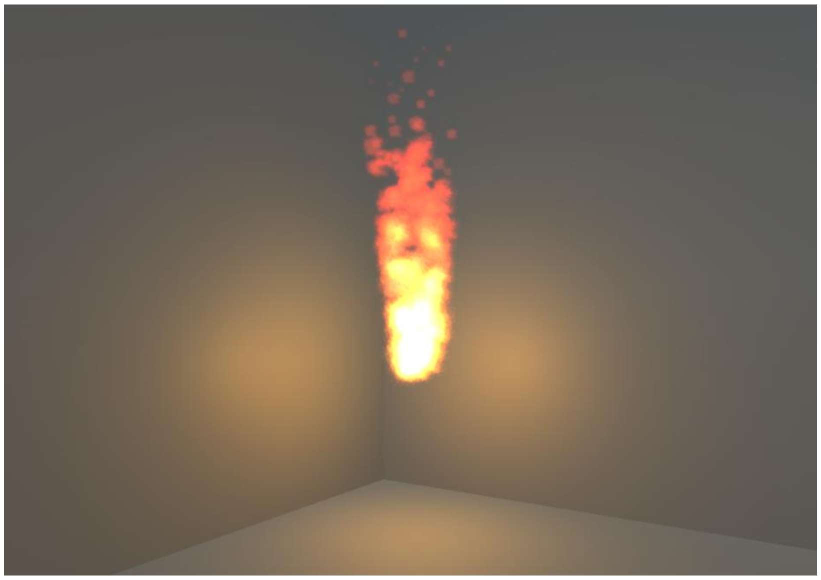
✓ Lights

Light	Particle System (Light)
Ratio	1
Random Distribution	✓
Use Particle Color	✓
Size Affects Range	✓
Alpha Affects Intensity	✓
Range Multiplier	1
Intensity Multiplier	1
Maximum Lights	10

To be able to see the lights from your particles flicker, place the system against a wall, or within a box. We'd recommend having the following settings to make sure the light is subtle, yet effective:

- **Ratio**: 0.3 (otherwise your lights may "stutter")
- **Range Multiplier: 0.5**
- **Intensity Multiplier: 0.5**

The result should look something *light* this….



Note: You will notice that the light only illuminates surrounding objects when the emitted particles are alive. If you want a subtle and steady glow around the particle system, you can turn your light component (attached to the game object) back on. This choice depends on the effect you are trying to create, e.g. a steady glowing torch vs. a more dramatic pulsing flame. If you decide to leave this component on you should also make its 'Color' approximately match the colour of your fire.

Consider how lighting from a particle effect like this might be used to light a scene and/or help guide a player through it.

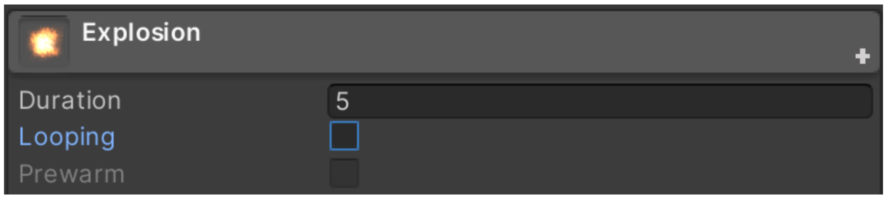
Take this opportunity to give your particle an appropriate name, like "Flame". We will work on this fire particle more towards the end of the prac. Leave this as it is for now and move on with the rest of the prac.

Reminder: Save, Commit and Push to GitHub

Explosions – Short-lived Effects

To create an explosion, we will create a new particle effect that reuses the Fire Particle material we created earlier. Move to another area in your scene (with some distance from your fire effect). **Create a new particle system**, rename it "**Explosion**", and set the **Material** (in the Renderer module) to the **Fire Particle**.

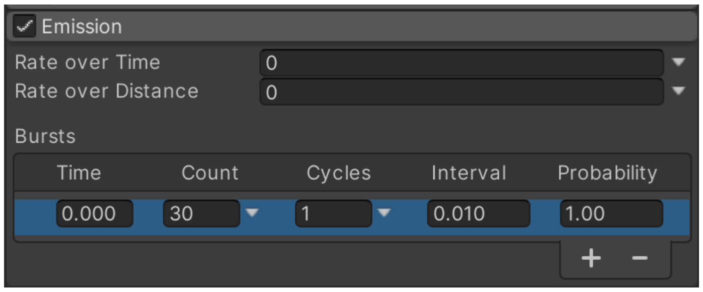
The main difference between an explosion and the torch is that the torch is a *looping* effect that goes forever, while the explosion is a one-off event. We can change this by unticking the **Looping** checkbox in the standard module.



Explosion

Duration	5
Looping	☐
Prewarm	☐

You should notice that the system now runs for a period of time and then stops. Still, the particles are coming out too slowly. We want a lot of particles to be create in one burst. We can do this using the **Bursts** settings in the **Emission** module. First set the **Rate over Time** to zero, then press the **+ button** in the bottom right corner of the module to add a burst.



✓ Emission

Rate over Time	0
Rate over Distance	0

Bursts

Time	Count	Cycles	Interval	Probability
0.000	30	1	0.010	1.00

This means the system will now create 30 particles at time 0.0 (i.e. when the system starts) and no more after that. You may wish to increase the 'Count' for an explosion with more particles.

The other major thing we want to change is shape of the emission. If we assume the exploding object is sitting on the ground, we want the explosion to spread in a hemisphere away from the centre point. In the **Shape** module change the **Shape** from 'Cone' to **Hemisphere**.

Now play with the other settings to make something that feels more like an explosion. As a guide you could:

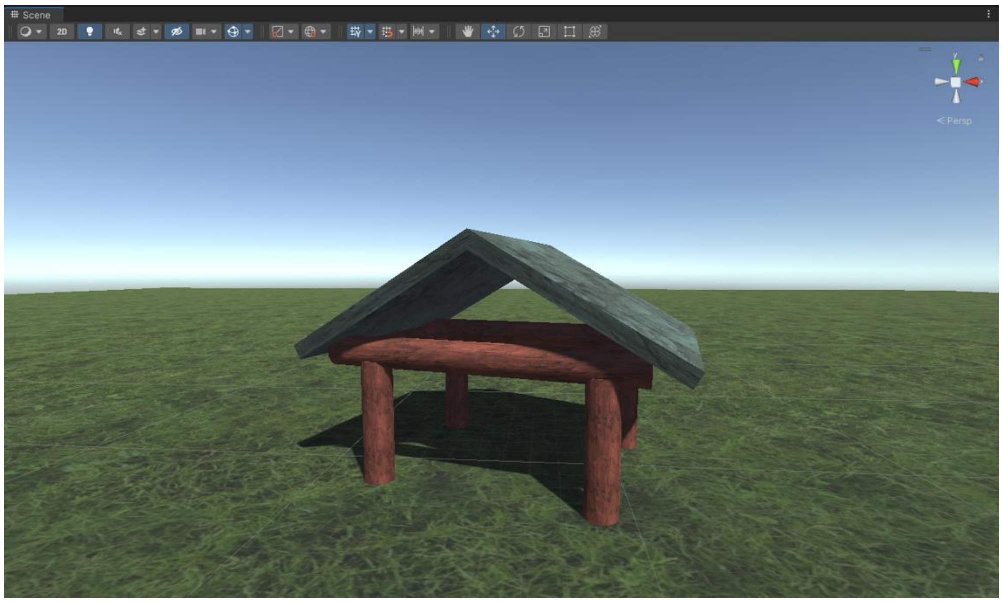
- Reduce the **Start Lifetime** to make the explosion more short-lived.
- Increase the **Start Size** to make the particles bigger
- Use the **Color Over Lifetime** module to make the particles turn red and fade.
- Use the **Size Over Lifetime** module to make the particles expand in size over time.
- Use the **Rotation Over Lifetime** module to add a random rotation to each particle

Reminder: Save, Commit and Push to GitHub

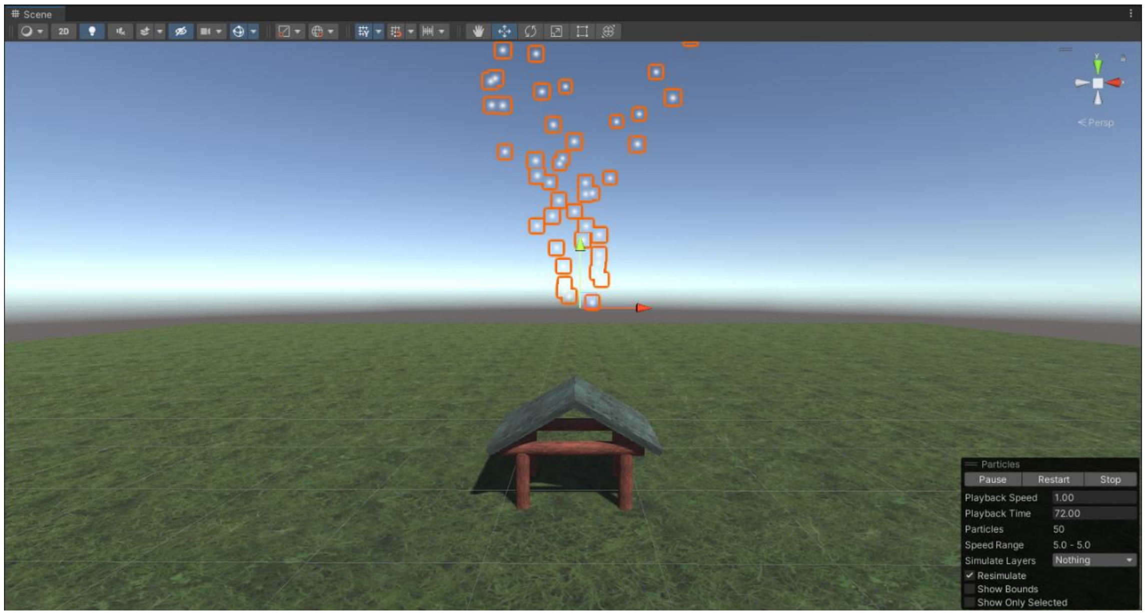
Rain - Interacting with Other Objects

The last effect we are going to create is a rain effect. We will use this to show how particles can interact with objects in the world.

First of all, build a simple landscape out of Unity primitives in your scene (again, off to the side with some distance from your other effects). As an example, the simple hut below is built out of cylinders and cubes, with a plane for the ground. Importantly, all the Unity primitives have colliders, so that later on we will be able to make particles collide with the structure.

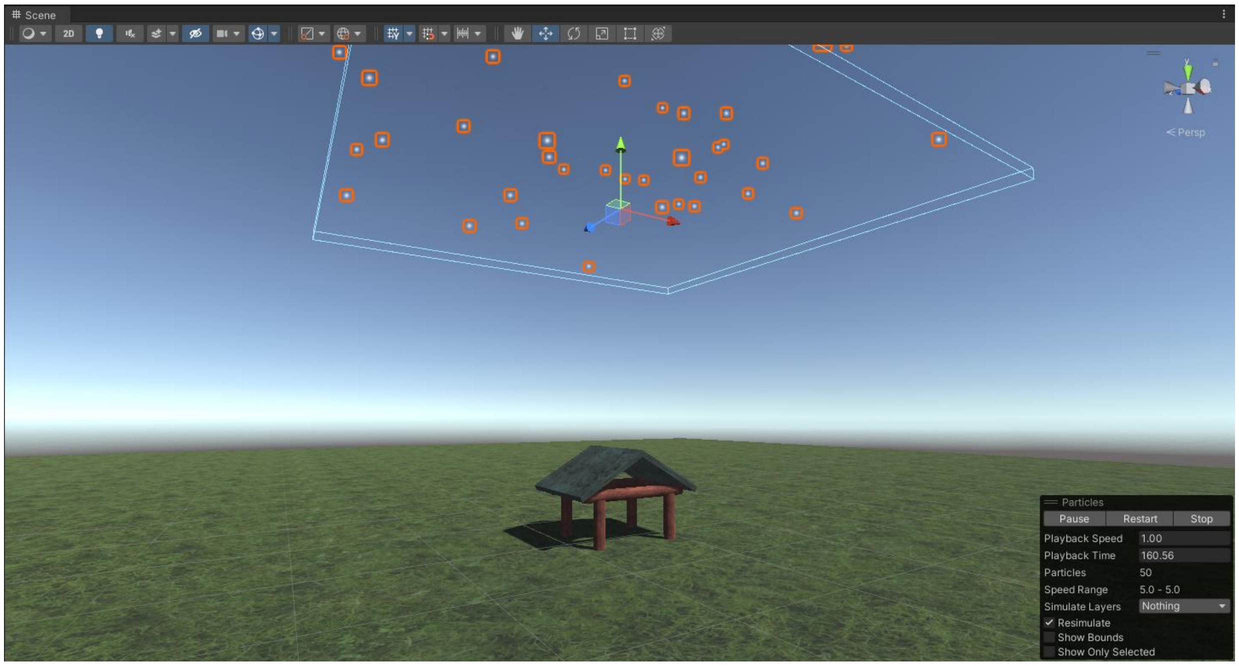


Now **create a new particle system** and rename it "**Rain**". This time put it in the sky above the scene.



```
Playback Speed    1.00
Playback Time     72.00
Particles         50
Speed Range       5.0 - 5.0
Simulate Layers   Nothing
[✓] Resimulate
[ ] Show Bounds
[ ] Show Only Selected
```

In the Shape module, change the shape of the system from a ‘Cone’ to **Box**, and then resize the **Scale X** and **Y** values (within the **Shape** module of the ‘Particle System’, don’t touch the Transform scale component) so that the box comfortably covers a large area above your shelter.

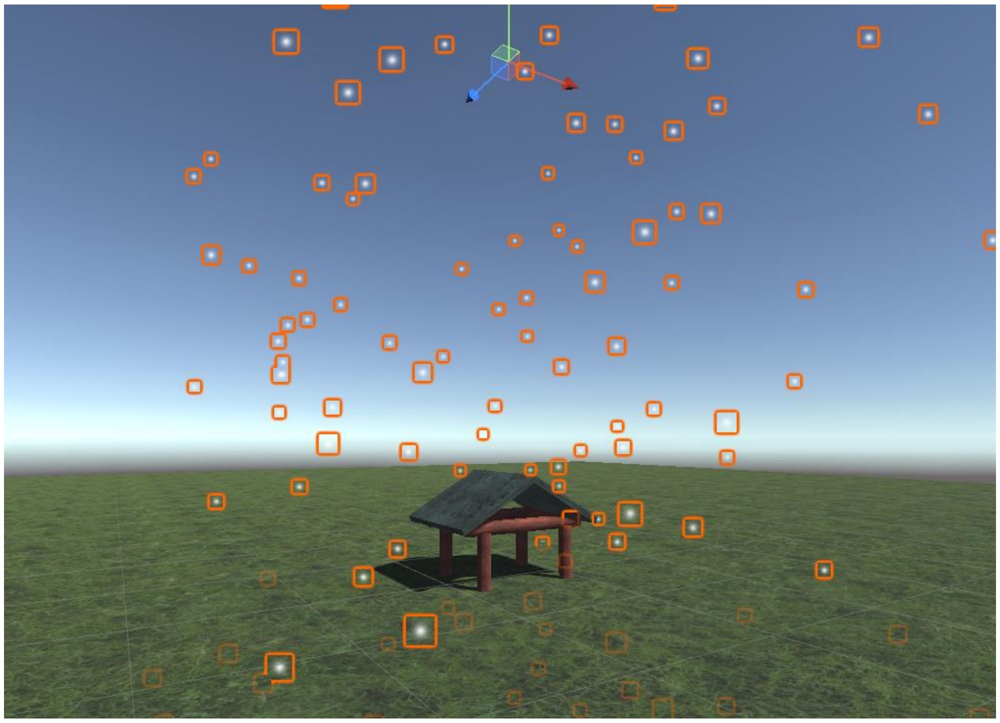


Particles

Pause | Restart | Stop

Playback Speed 1.00
Playback Time 160.56
Particles 50
Speed Range 5.0 - 5.0
Simulate Layers Nothing
✓ Resimulate
Show Bounds
Show Only Selected

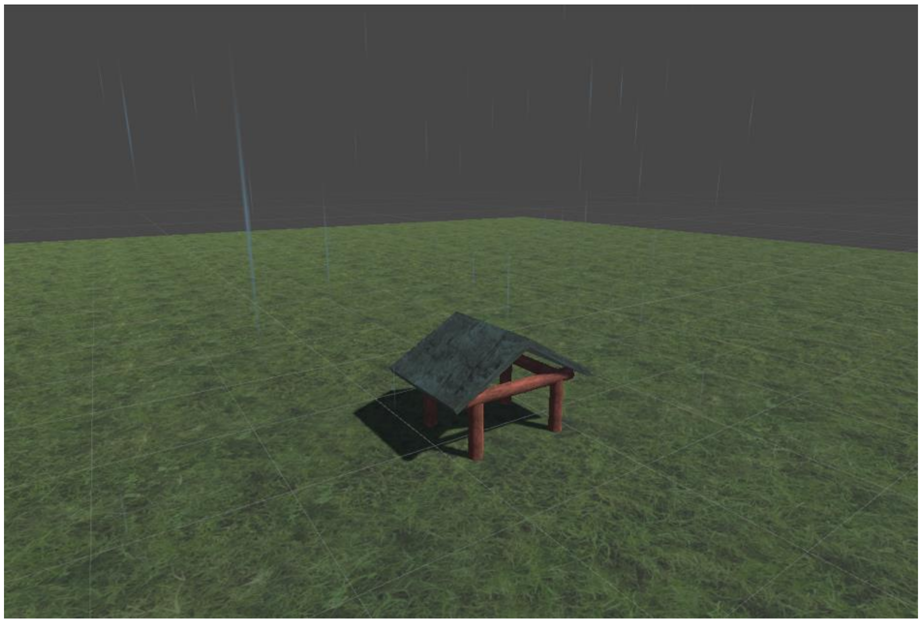
Rotate the system so it is pointing downwards (**Rotation** within **Shape** module). Increase the **Start Speed** and **Emission Rate** so you have a lot of particles rapidly falling on the scene.



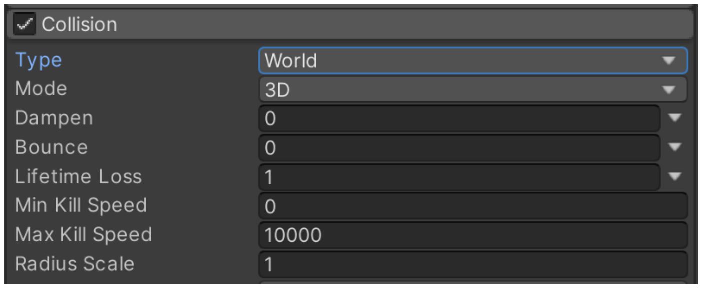
To make the particles look like rain, try the following:

1. Change the **Start Speed** to 60
2. Change the **Start Colour** to greyish blue.
3. Change the **Start Size** to 0.1
4. In the **Emissions** module, change **Rate over time** to be 100
5. In the **Renderer** module, change **Render Mode** from Billboard to **Stretched Billboard**
6. Set the **Speed Scale** (also in the Renderer) to 0.05

The 'Stretched Billboard' mode stretches the particle in its current direction of movement, in this case giving us *streaks* of rain. Tweak the above values to create your desired effect.



Notice that the rain is currently falling straight through the roof of the hut. You can fix this by activating the **Collision** module and setting the collision type to **World** (rather than Planes).



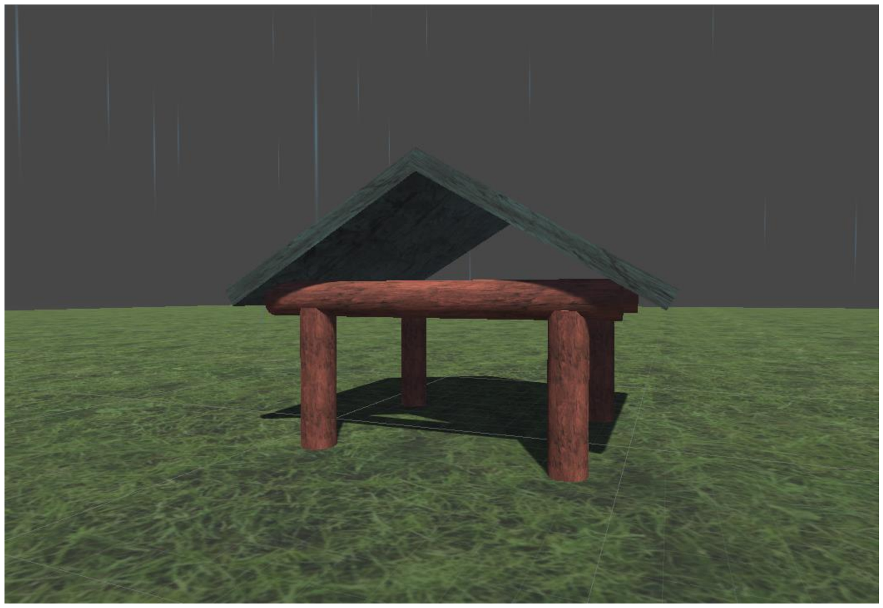
Collision

Type	World
Mode	3D
Dampen	0
Bounce	0
Lifetime Loss	1
Min Kill Speed	0
Max Kill Speed	10000
Radius Scale	1

When you do this, you will get a rather strange effect. The rain bounces off the roof and the ground. This is because the **Bounce** value is set to 1. This is the proportion of velocity that is maintained after collision. For this simulation we don't want particles to bounce, but rather to be destroyed when they collide with the roof.
In the **Collision** module

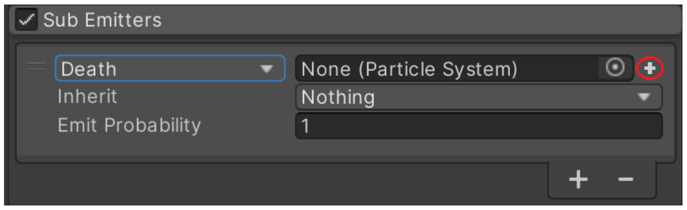
- Set **Bounce** to 0
- Set **Lifetime Loss** to 1

This means that when the collision occurs the particle loses all its remaining lifetime (i.e. it is immediately destroyed). You should now see that the rain disappears when it collides with the roof/ground.



To finish the effect, we want a small splash to occur when the raindrop hits a surface. We can do this using **sub-emitters**. A ‘sub-emitter’ is a new particle effect created when something happens to a particle (birth, collision or death).

Activate the **Sub Emitters** module, change the setting from **Birth** to **Death,** and press the **+ sign** next to the object picker button.



Sub Emitters

Death | None (Particle System)

Inherit: Nothing

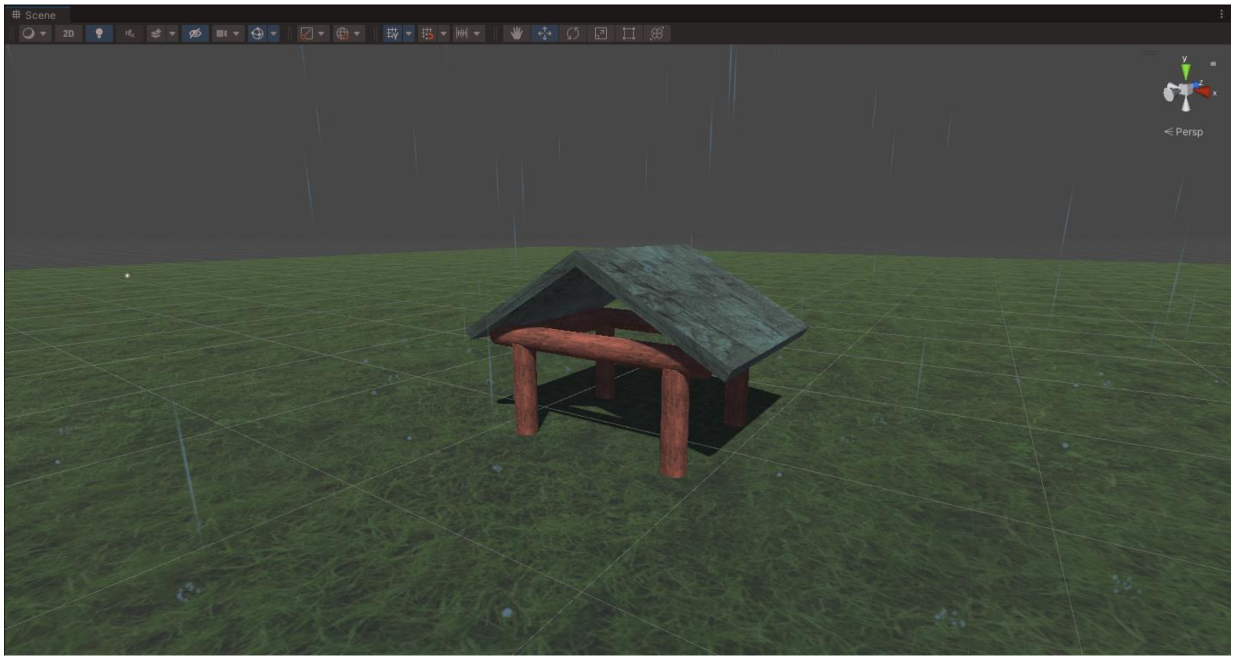
Emit Probability: 1

This creates a new particle system at the location of the existing particle whenever it dies. Notice in your Hierarchy view that the rain Particle System now has a new child attached to it called “SubEmmiter0”. This object contains the particle system component which will be used for the splashes.

Practically, a splash is basically a small, wet explosion. So as we did above for the explosion, change the SubEmmiter’s settings so that:

1. The **Duration** is 0.5
2. The **Looping** flag is off
3. The **Start Lifetime** is short (maybe 0.2s)
4. The **Start Speed** is higher (maybe 3)
5. The **Start Size** is small (maybe 0.1)
6. The **Start Color** matches the colour of your rain
7. In the **Emission** module, ensure the **Rate over Time** is set to **0**, and add a small burst of ~15 particles at time 0
8. In the **Shape** module, ensure the **Shape** is set to **Sphere** and that the radius is small (it may already be set to this).

Because the Particle material we are using is additive, it can be difficult to see particles against a white background (adding more colour to white just gives white). You could fix this by making a new particle material which used the **Alpha Blend** shader instead of the Additive shader. Or as you might have gathered from the above images, removing your skybox and making the rain fall on a dark-coloured plane.



Reminder: Save, Commit and Push to GitHub

Extra

If you have prac time remaining, test out the following ideas, which should help you become more familiar with particle systems and may be claimable features in your Game Design Task. Assuming there is time, your tutor will expect you to **make at least one (1)** of the following for them to assess.

- **Create a snow effect** by duplicating the geometry/particle systems used for your rain effect in another area and **modifying its parameters**. Think about how snow differs from rain as it falls, and what should happen when the snow collides with an object.

- **Insert a global post effects volume and layer**, and add the effect called **Bloom.** Once added, tweak the intensity, and you will notice that your torch and explosions have softer looking particles.

- **Re-visit your fire particle and try and make it fit within a realistic setting**. For example, you could turn your flame effect into a torch by adding a cylinder primitive beneath it and adding a rock wall behind it using a rock texture with normal and height maps. Another option would be to turn it into a little campfire or fireplace scene. Note that for your light component you may need to turn shadows on, using either the hard or soft shadow setting.

- **Make your fire emit temporary puffs of smoke or cinders**. You may need to download a new texture to do this, and there are a couple of different approaches you could take to implement this.

- **Insert a global post effects volume and layer**, and add the effect called **Bloom.** Once added, tweak the intensity, and you will notice that your torch and explosions have softer looking particles.

- **Find a new texture for your fire particle/explosion**, changing the art style and appearance of it. Ensure you have a license for any assets used.

- **Experiment with different fire gradients for your flame/explosion**. Some colour combinations you could try are:
  - Green and purple, for a *spooky looking flame*. These could look even spookier with cyan/dark green or white/purple.
  - Baby blue and pink, for a *neat retro looking flame* (you could probably make this look great with a “glitchy” post-effect).

Extra - Advanced

You will probably not have time to finish these in class, but you could also try:

- Turning your explosion into a firework, by making it a sub-emitter of a missile and experimenting with different colour transitions and further sub-emitters
- Making objects hit by your explosion catch fire temporarily (the same way you made the rain splash when hitting a collider)
- Creating a quick interior and giving it some additive dust particles that slowly and randomly float around the room.
- Creating a particle system to blow leaves in front of the player as they move around the world
- Making "god rays" shine through a window/hole in a structure

Show your Demonstrator

If you still have time you might like to dress your scene a little or further tweak your effects. To demonstrate your understanding of this week's content to your demonstrator you should show:

- Your fire particle system, with lights implemented
- Your explosion particle system
- Your rain particle system
- Anything extra you made this week
- That you have made progress in your separate Game Design Task repo since week 8 (show your updates via your GDT GitHub repo on the web)