CT3536 Games Programming

Terrains
Particle Emitters
Some C# Collection Classes

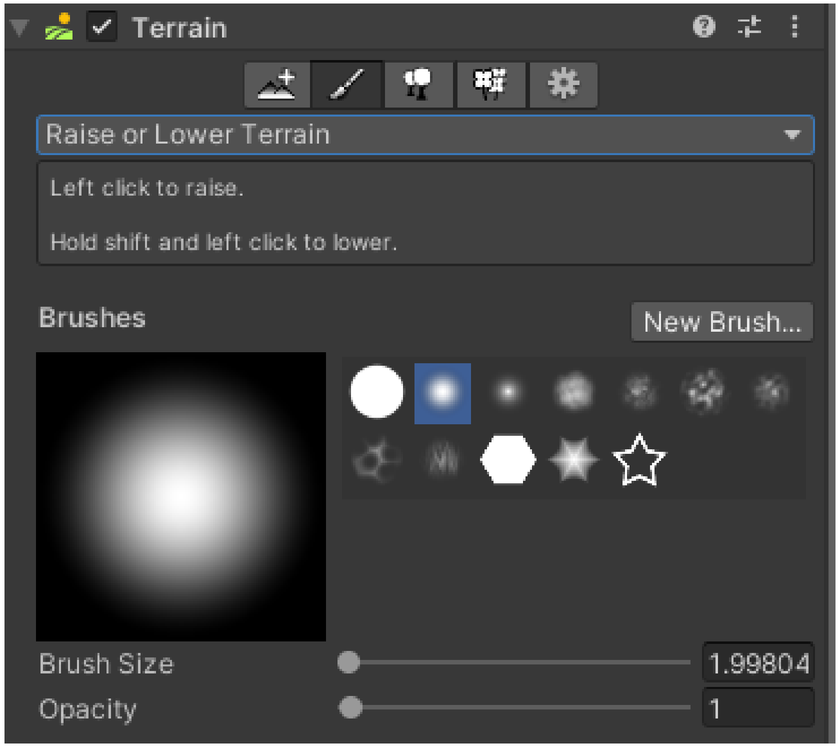
Terrains.

Terrains

https://docs.unity3d.com/Manual/script-Terrain.html

Unity terrains offer a lot of features:

- Heightmap import
- Heightmap randomisation and sculpting tools
- Terrain texture painting
- Trees
- Plants



Left click to raise.

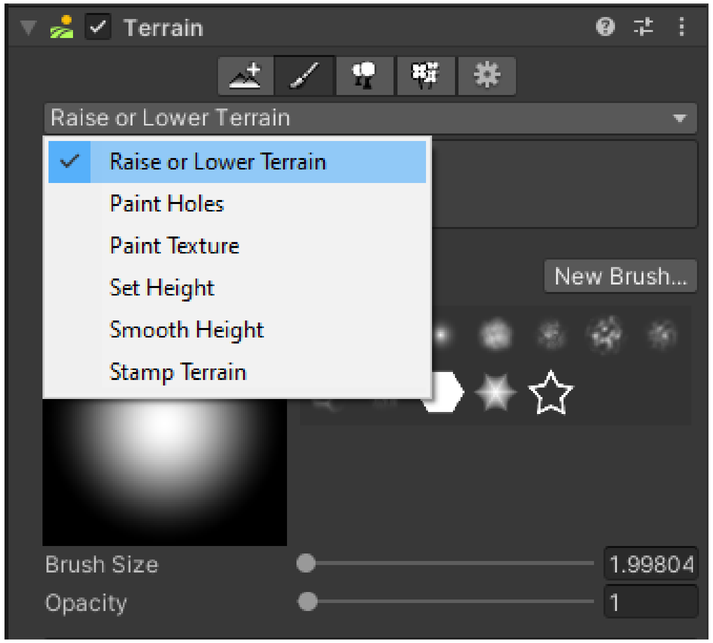
Hold shift and left click to lower.

Brush Size 1.99804

Opacity 1

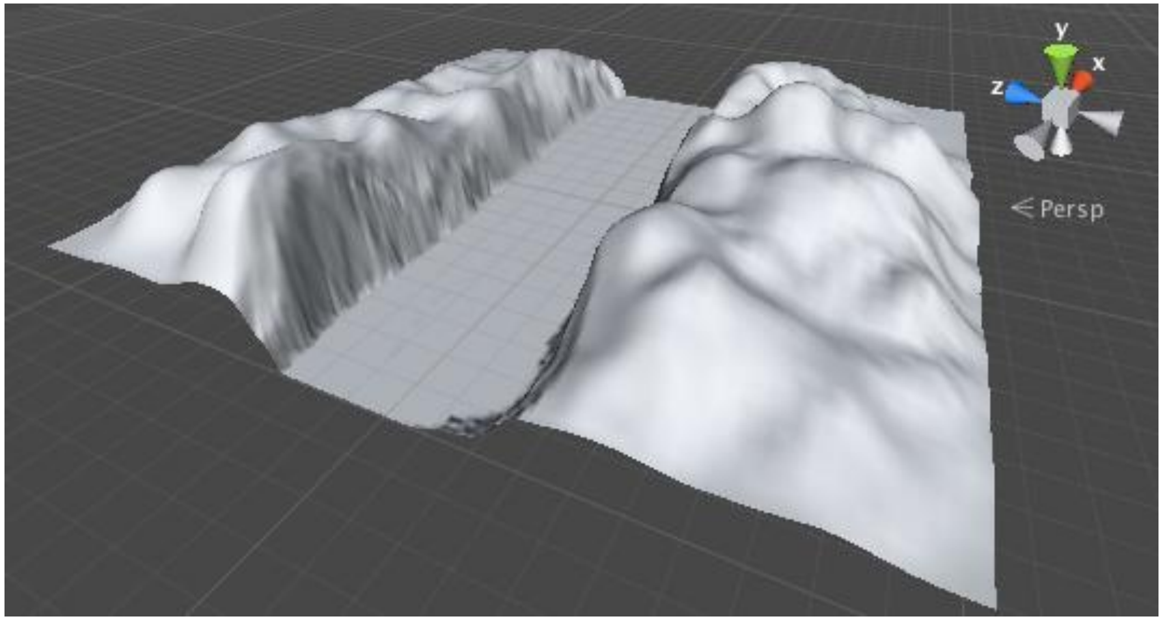
- To add a Terrain GameObject to a Scene, select GameObject > 3D Object > Terrain. This also adds a corresponding Terrain Asset which stores data about it.
- The landscape is initially a large, flat plane.
- With the exception of the tree placement tool and the settings panel, all the tools on the toolbar provide a set of "brushes" and settings for brush size and opacity.

Terrain Height Tools



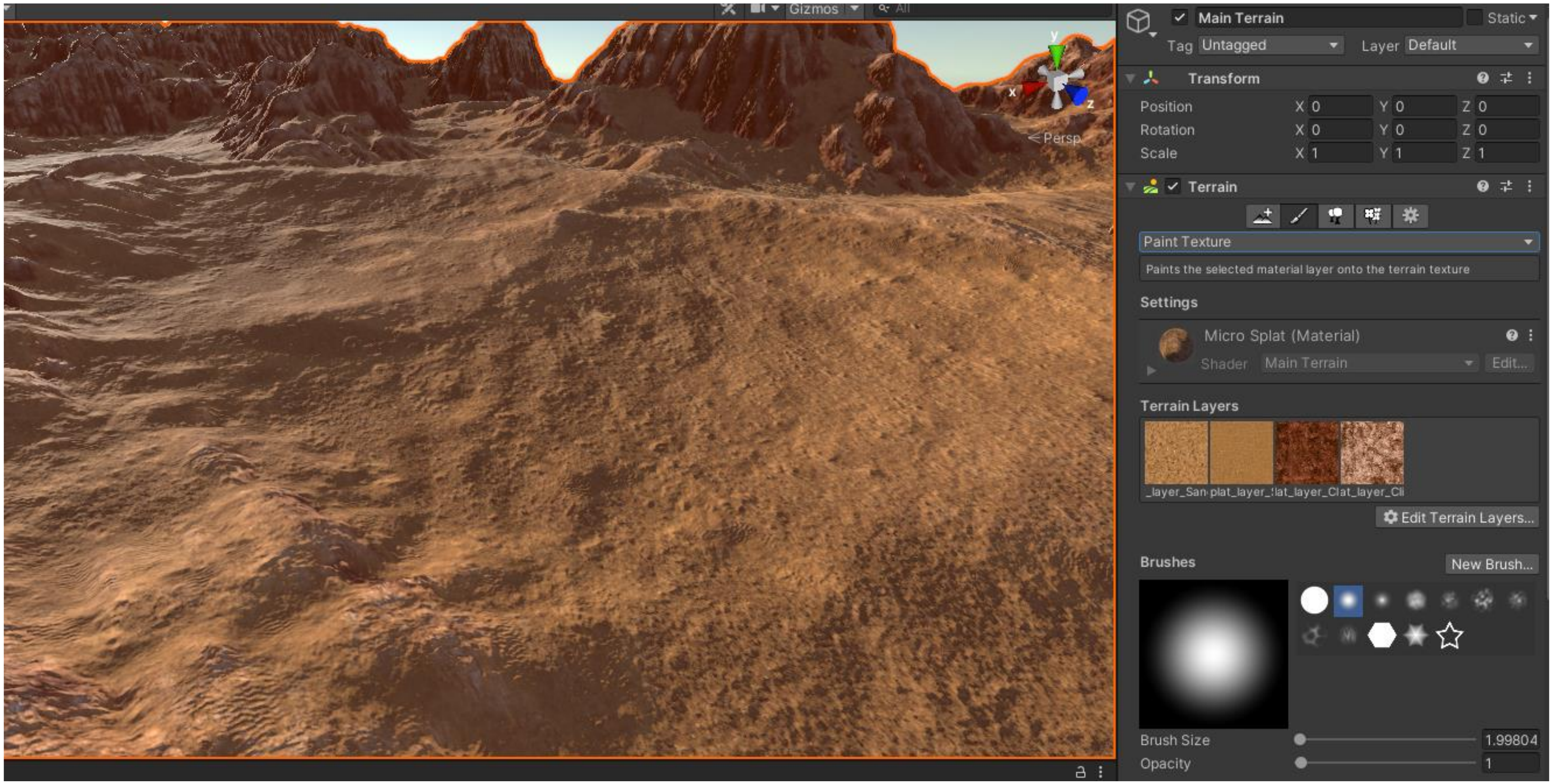
Brush Size 1.99804
Opacity 1

- Raise/Lower Height
- Paint Holes
- Set Height
- Smooth Height
- Different brushes can be used to create various effects:
  - For example, you can create rolling hills by increasing the height with a soft-edged brush and then cut steep cliffs and valleys by lowering with a hard-edged brush.



Terrain Textures

- Terrain textures should tile seamlessly
- One texture acts as the base for the whole terrain
- Other textures are painted on top, with variable opacity ('splat maps')



Main Terrain Static

Tag Untagged Layer Default

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Terrain

Paint Texture

Paints the selected material layer onto the terrain texture

Settings

Micro Splat (Material)

Shader Main Terrain Edit...

Terrain Layers

_layer_San plat_layer_!lat_layer_Clat_layer_Cli

Edit Terrain Layers...

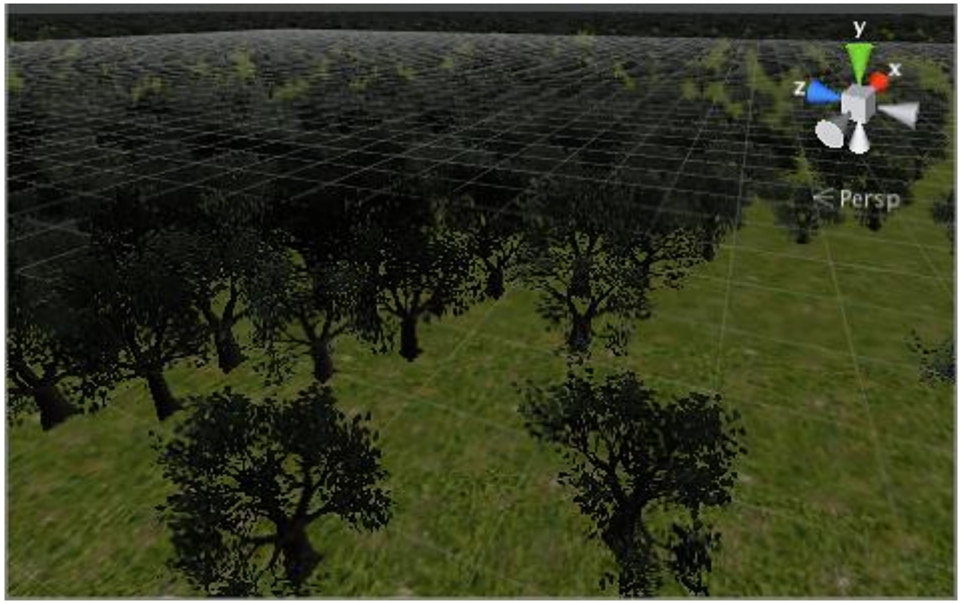
Brushes New Brush...

Brush Size 1.99804

Opacity 1

Trees

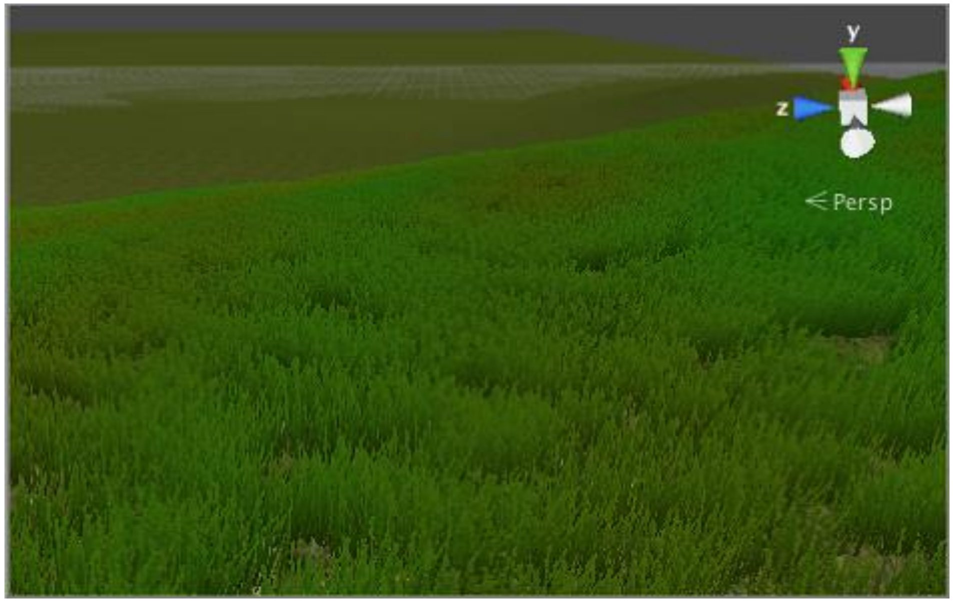
- Trees are "painted" onto the terrain
- You must define tree prefabs to use

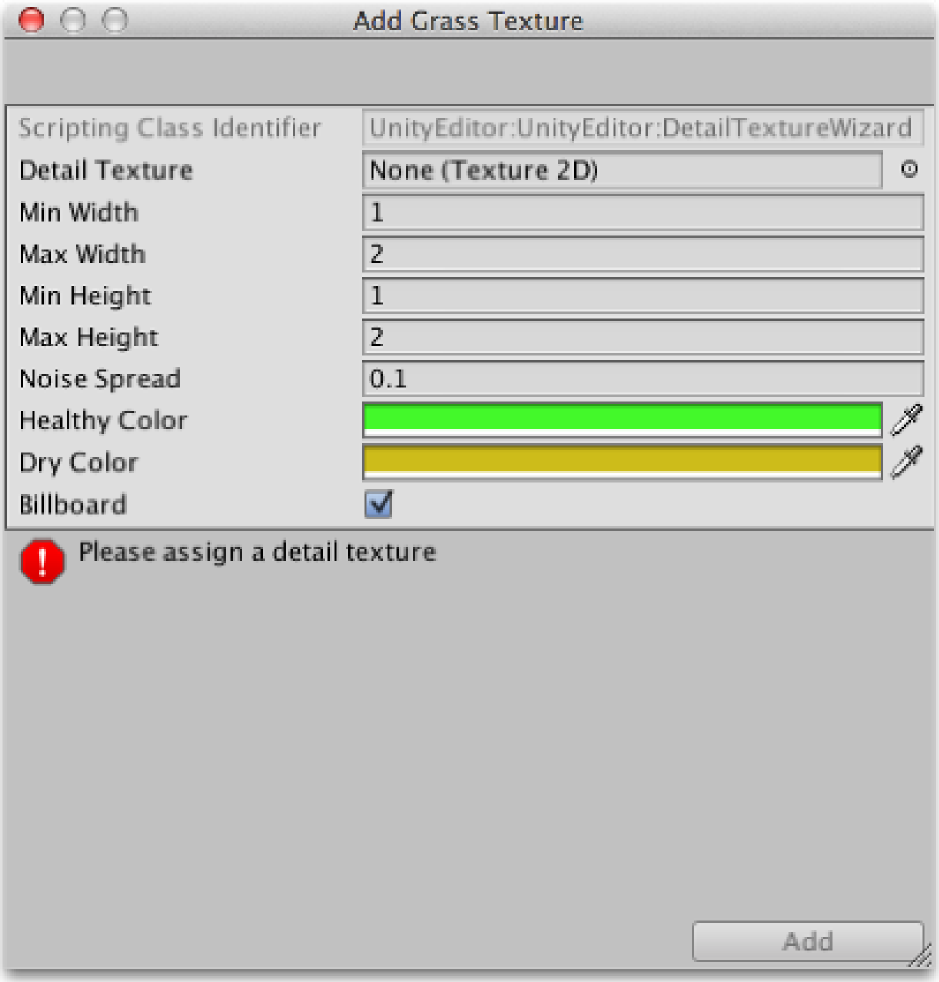


- There are several tree formats supported (e.g. SpeedTree) which allow such things as randomised variations, and wind effects
- See: https://docs.unity3d.com/Manual/terrain-Trees.html https://docs.unity3d.com/Manual/class-Tree.html

Grass/Plants

These are textures (with transparency) rendered as *billboards,* i.e. single quads which rotate to face the camera





Scripting Class Identifier	UnityEditor:UnityEditor:DetailTextureWizard
Detail Texture	None (Texture 2D)
Min Width	1
Max Width	2
Min Height	1
Max Height	2
Noise Spread	0.1
Healthy Color	🟩
Dry Color	🟨
Billboard	✓

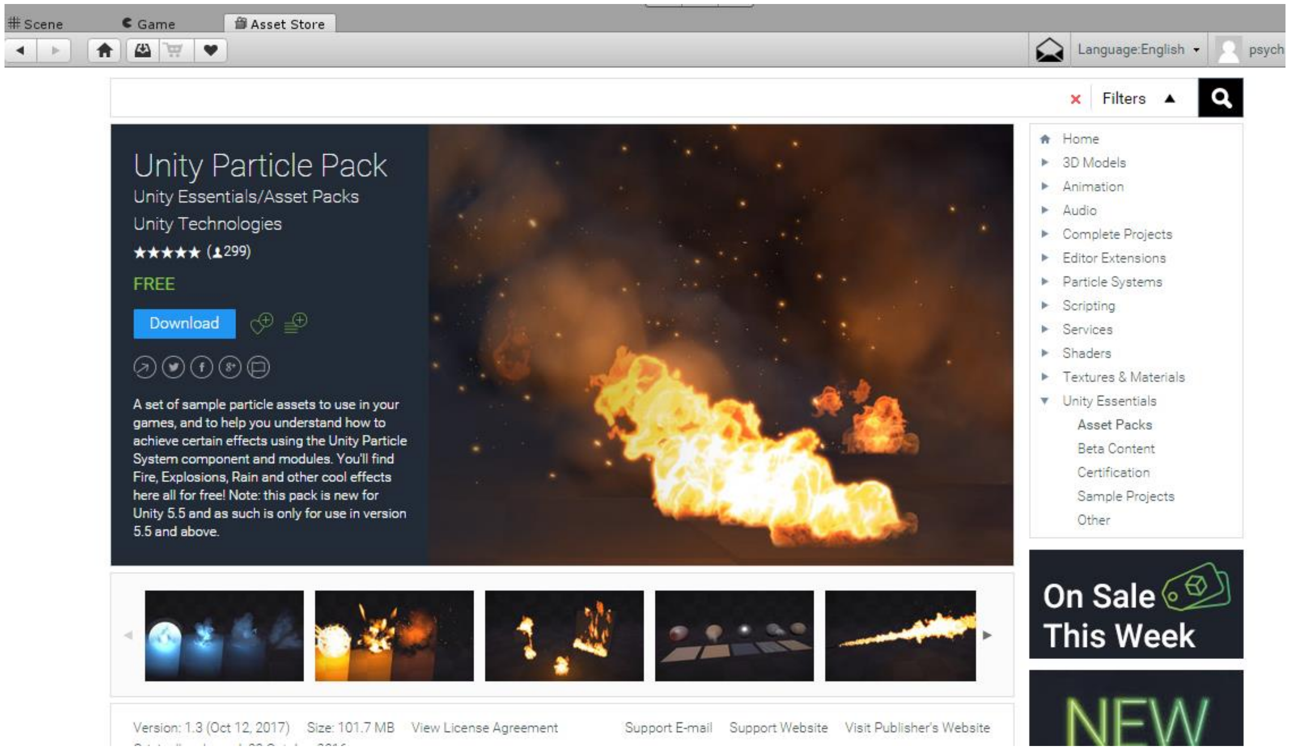
Please assign a detail texture

Particle Emitters

- Particles are a standard approach in games for flexible and efficient creation of special effects such as fire, smoke, explosions, sparks, rain, etc..
- Based on the concept of textured billboards which are emitted according to a defined shape and pattern, with defined behaviour / changes to each particle's size, colour, position over time
- In Unity, use the new **ParticleSystem** component, rather than the older ParticleEmitter component (but Particle Emitter is the more usual term for the concept, in game development)
- https://docs.unity3d.com/ScriptReference/ParticleSystem.html
- https://docs.unity3d.com/Manual/ParticleSystems.html

Particle Emitters

- Add a ParticleSystem component to a Game Object
- A particle system has a lot of settings! But this makes it hugely flexible and powerful (+ it has been heavily optimised in the core game engine for efficient rendering)
- A good place to start is the free "Unity Particle Pack" asset on the Unity asset store



Unity Particle Pack
Unity Essentials/Asset Packs
Unity Technologies
★★★★★ (299)
FREE

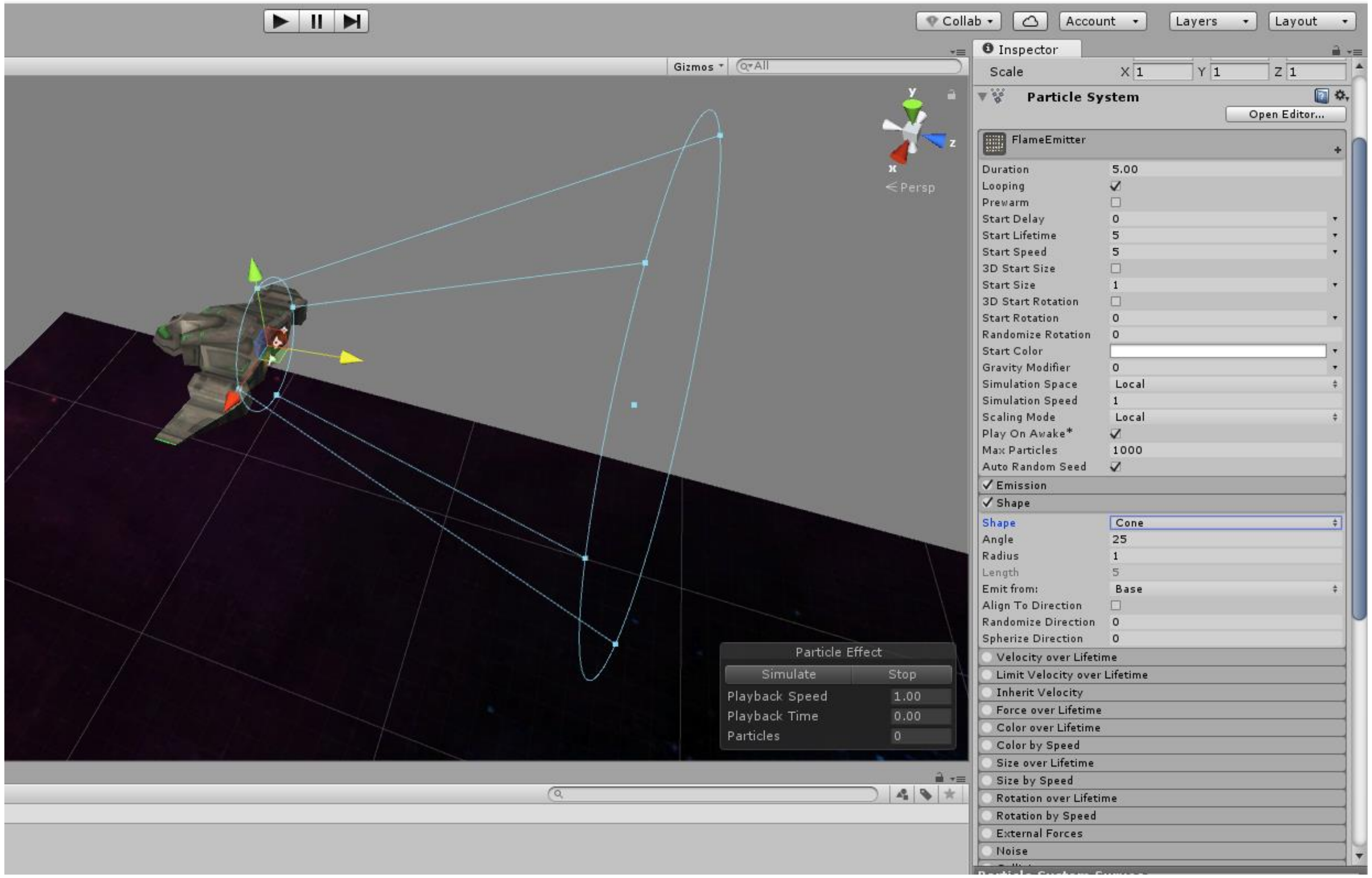
A set of sample particle assets to use in your games, and to help you understand how to achieve certain effects using the Unity Particle System component and modules. You'll find Fire, Explosions, Rain and other cool effects here all for free! Note: this pack is new for Unity 5.5 and as such is only for use in version 5.5 and above.

Version: 1.3 (Oct 12, 2017) Size: 101.7 MB View License Agreement Support E-mail Support Website Visit Publisher's Website

ParticleSystem Example

Adding a flame emitter to use when our Asteroids game Spaceship has its engines engaged

N.B. I have attached the ParticleSystem component to an otherwise empty Game Object nested inside the main spaceship. This allowed me to rotate and position it correctly



Scale X 1 Y 1 Z 1

Particle System

FlameEmitter

Duration	5.00
Looping	✓
Prewarm	☐
Start Delay	0
Start Lifetime	5
Start Speed	5
3D Start Size	☐
Start Size	1
3D Start Rotation	☐
Start Rotation	0
Randomize Rotation	0
Start Color	
Gravity Modifier	0
Simulation Space	Local
Simulation Speed	1
Scaling Mode	Local
Play On Awake*	✓
Max Particles	1000
Auto Random Seed	✓

✓ Emission

✓ Shape

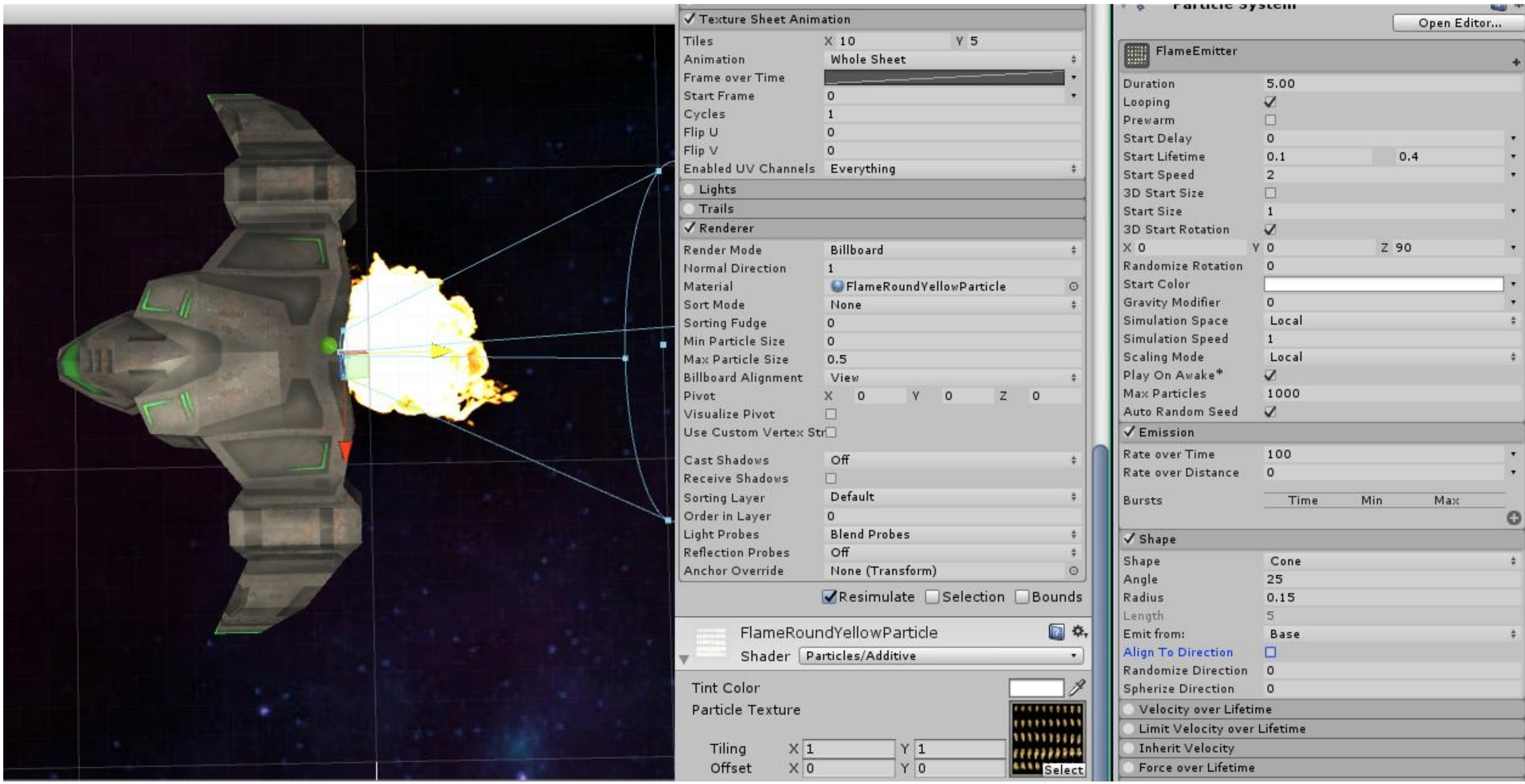
Shape	Cone
Angle	25
Radius	1
Length	5
Emit from:	Base
Align To Direction	☐
Randomize Direction	0
Spherize Direction	0

Velocity over Lifetime
Limit Velocity over Lifetime
Inherit Velocity
Force over Lifetime
Color over Lifetime
Color by Speed
Size over Lifetime
Size by Speed
Rotation over Lifetime
Rotation by Speed
External Forces
Noise

Playback Speed	1.00
Playback Time	0.00
Particles	0

Using a cone-shaped emission

Final Settings



✓ Texture Sheet Animation

Tiles	X 10 Y 5
Animation	Whole Sheet
Frame over Time	
Start Frame	0
Cycles	1
Flip U	0
Flip V	0
Enabled UV Channels	Everything

Lights

Trails

✓ Renderer

Render Mode	Billboard
Normal Direction	1
Material	FlameRoundYellowParticle
Sort Mode	None
Sorting Fudge	0
Min Particle Size	0
Max Particle Size	0.5
Billboard Alignment	View
Pivot	X 0 Y 0 Z 0
Visualize Pivot	☐
Use Custom Vertex Str	☐
Cast Shadows	Off
Receive Shadows	☐
Sorting Layer	Default
Order in Layer	0
Light Probes	Blend Probes
Reflection Probes	Off
Anchor Override	None (Transform)

✓ Resimulate ☐ Selection ☐ Bounds

FlameRoundYellowParticle

Shader Particles/Additive

Tint Color	
Particle Texture	Select
Tiling	X 1 Y 1
Offset	X 0 Y 0

Particle System

Open Editor...

FlameEmitter

Duration	5.00
Looping	✓
Prewarm	☐
Start Delay	0
Start Lifetime	0.1 0.4
Start Speed	2
3D Start Size	☐
Start Size	1
3D Start Rotation	✓
X 0	Y 0 Z 90
Randomize Rotation	0
Start Color	
Gravity Modifier	0
Simulation Space	Local
Simulation Speed	1
Scaling Mode	Local
Play On Awake*	✓
Max Particles	1000
Auto Random Seed	✓

✓ Emission

Rate over Time	100
Rate over Distance	0
Bursts	Time Min Max

✓ Shape

Shape	Cone
Angle	25
Radius	0.15
Length	5
Emit from:	Base
Align To Direction	☐
Randomize Direction	0
Spherize Direction	0

Velocity over Lifetime

Limit Velocity over Lifetime

Inherit Velocity

Force over Lifetime

Code Changes (Spaceship class)

- Turns the flames on/off as user presses/releases the Up arrow

Add inspector setting:

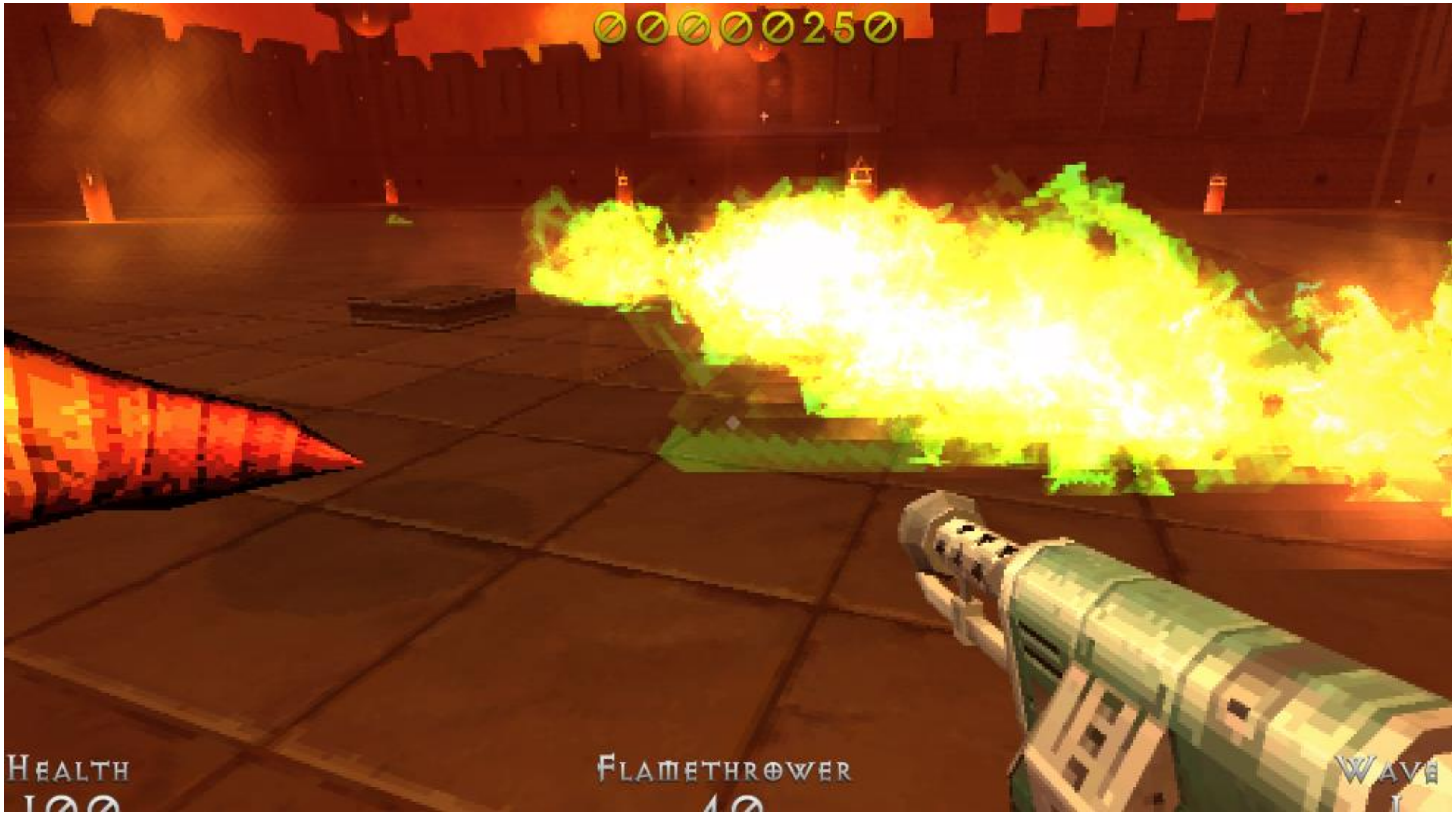
```
public ParticleSystem flameParticleSystem;
```

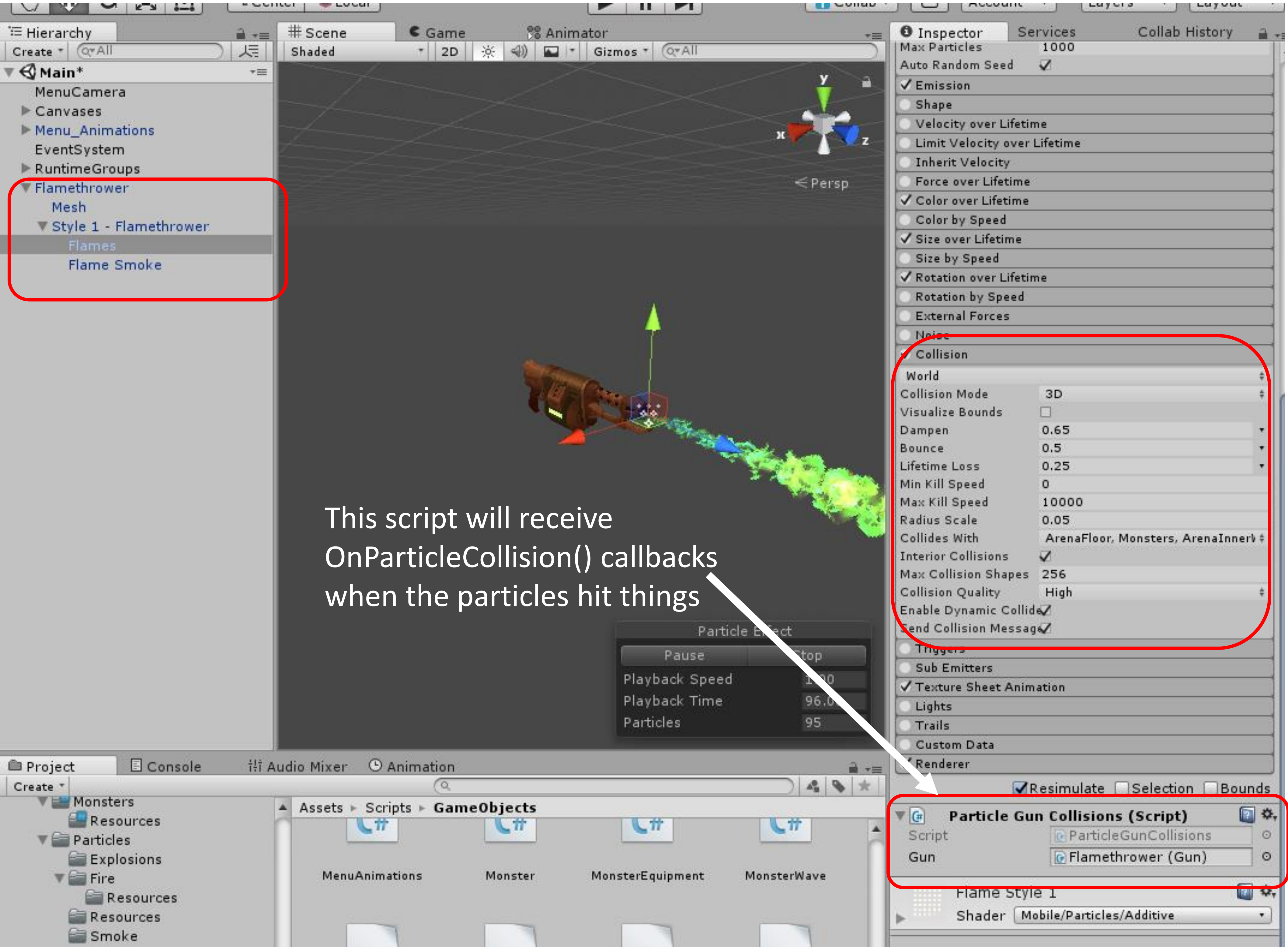
Change at the start of the Update() method:

```
    void Update () {
        if (Input.GetKey (KeyCode.UpArrow)) {
            rigidBody.AddForce (transform.forward * (rigidBody.mass *
Time.deltaTime * 500f));
            if (!flameParticleSystem.isPlaying)
                flameParticleSystem.Play ();
        }
        else if (flameParticleSystem.isPlaying)
            flameParticleSystem.Stop ();
```

Particle Collision Callbacks

We can use particle collision callbacks to calculate flamethrower damage to enemies, and spread fire on the floors and walls





This script will receive OnParticleCollision() callbacks when the particles hit things

Playback Speed	1.00
Playback Time	96.0
Particles	95

Max Particles	1000
Auto Random Seed	✓

✓ Emission
Shape
Velocity over Lifetime
Limit Velocity over Lifetime
Inherit Velocity
Force over Lifetime
✓ Color over Lifetime
Color by Speed
✓ Size over Lifetime
Size by Speed
✓ Rotation over Lifetime
Rotation by Speed
External Forces
Noise
✓ Collision

World	
Collision Mode	3D
Visualize Bounds	☐
Dampen	0.65
Bounce	0.5
Lifetime Loss	0.25
Min Kill Speed	0
Max Kill Speed	10000
Radius Scale	0.05
Collides With	ArenaFloor, Monsters, ArenaInnerW
Interior Collisions	✓
Max Collision Shapes	256
Collision Quality	High
Enable Dynamic Collide	✓
Send Collision Messag	✓

Triggers
Sub Emitters
✓ Texture Sheet Animation
Lights
Trails
Custom Data
✓ Renderer

✓ Resimulate ☐ Selection ☐ Bounds

Particle Gun Collisions (Script)

Script	ParticleGunCollisions
Gun	Flamethrower (Gun)

Flame Style 1
Shader Mobile/Particles/Additive

Some C# Collection Classes

My three favourite collection classes provided by C#:

- **List**
  - Dynamically add/remove items from a list
  - Iterate the list
  - Random-access the list by index
- **LinkedList**
  - Dynamically add/remove items from a list where this will be done very frequently (i.e. add/remove is very efficient)
  - Iterate the list
- **Dictionary**
  - Dynamically add/remove items from a collection which is indexed by any nominated data type, e.g. String or (sparse) Integer, or even Object