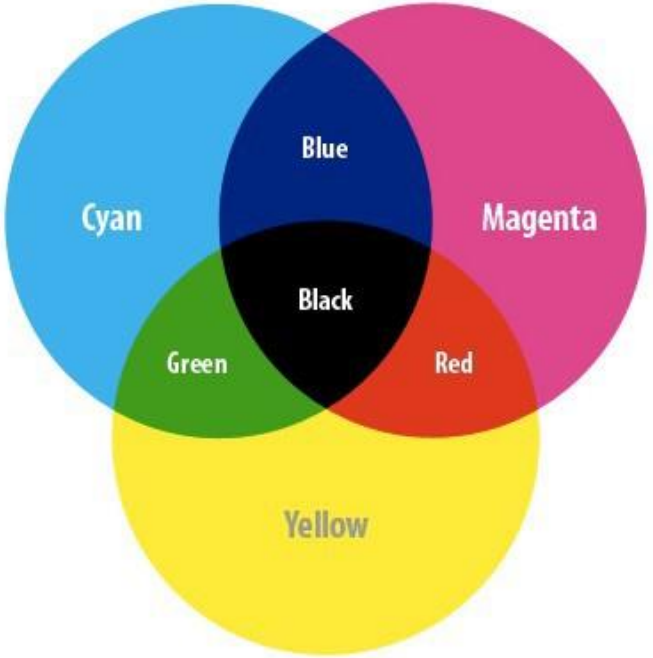
CS/DES 427/450
Unity Color /Components

Daria Tsoupikova
Michael Papka
Hal Brynteson

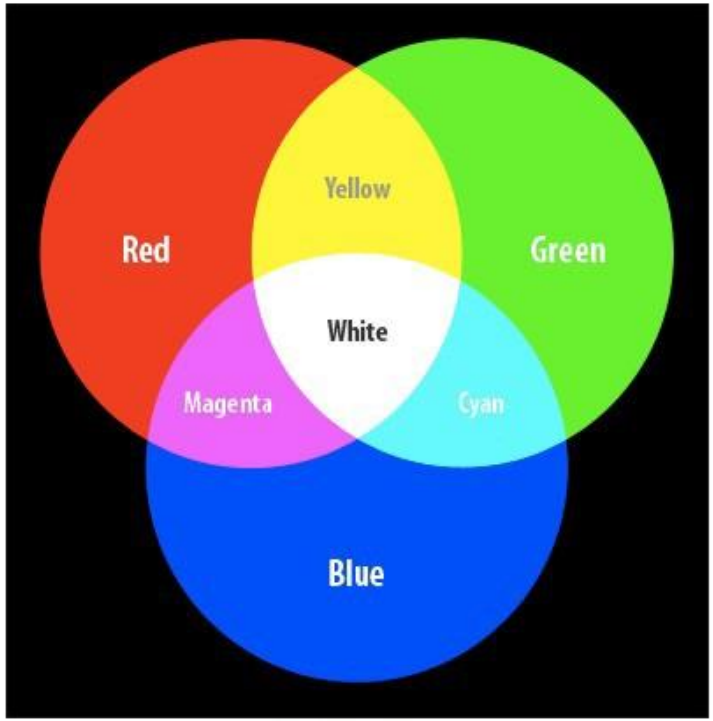
Color Systems

CMYK – The Subtractive System



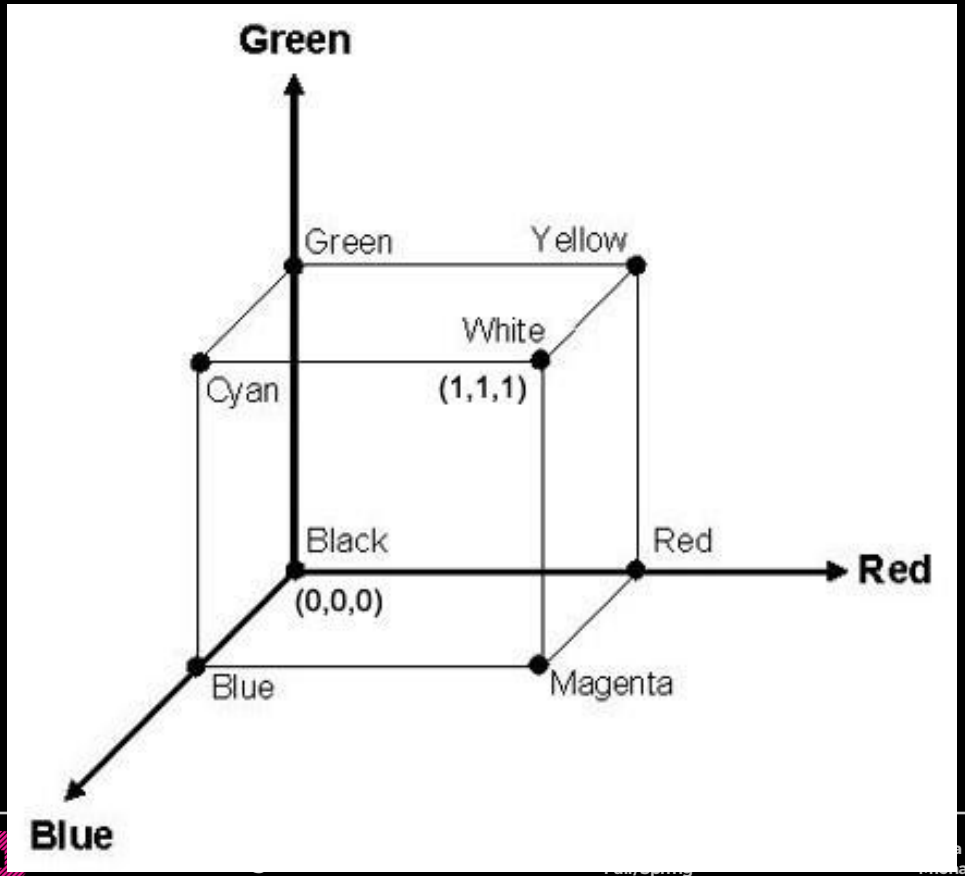
CMYK Color Model

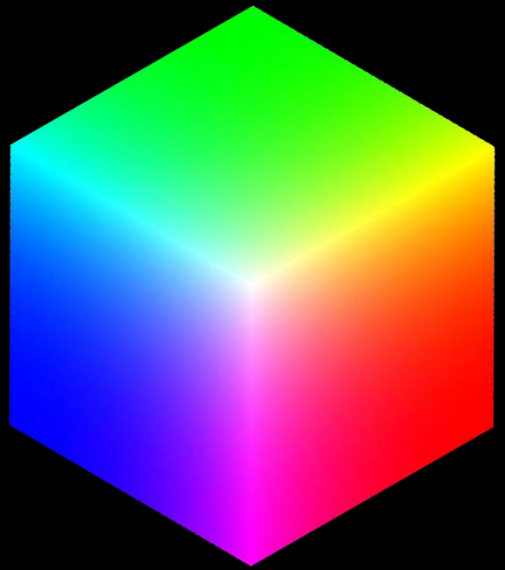
RGB – The Additive System



RGB Color Model

Color Systems





RGBA color

Values for red, green, blue and alpha are floating point values
with a range from 0 to 1 (ex. 0.3, 0.77)

Alpha component (a) defines transparency
Alpha of 1 is completely opaque, alpha of zero is completely transparent

Black RGBA is (0, 0, 0, 1)

Blue RGBA is (0, 0, 1, 1)

Gray RGBA is (0.5, 0.5, 0.5, 1)

Clear Completely transparent RGBA is (0, 0, 0, 0)

RGBA color

Black RGBA is (0, 0, 0, 1)

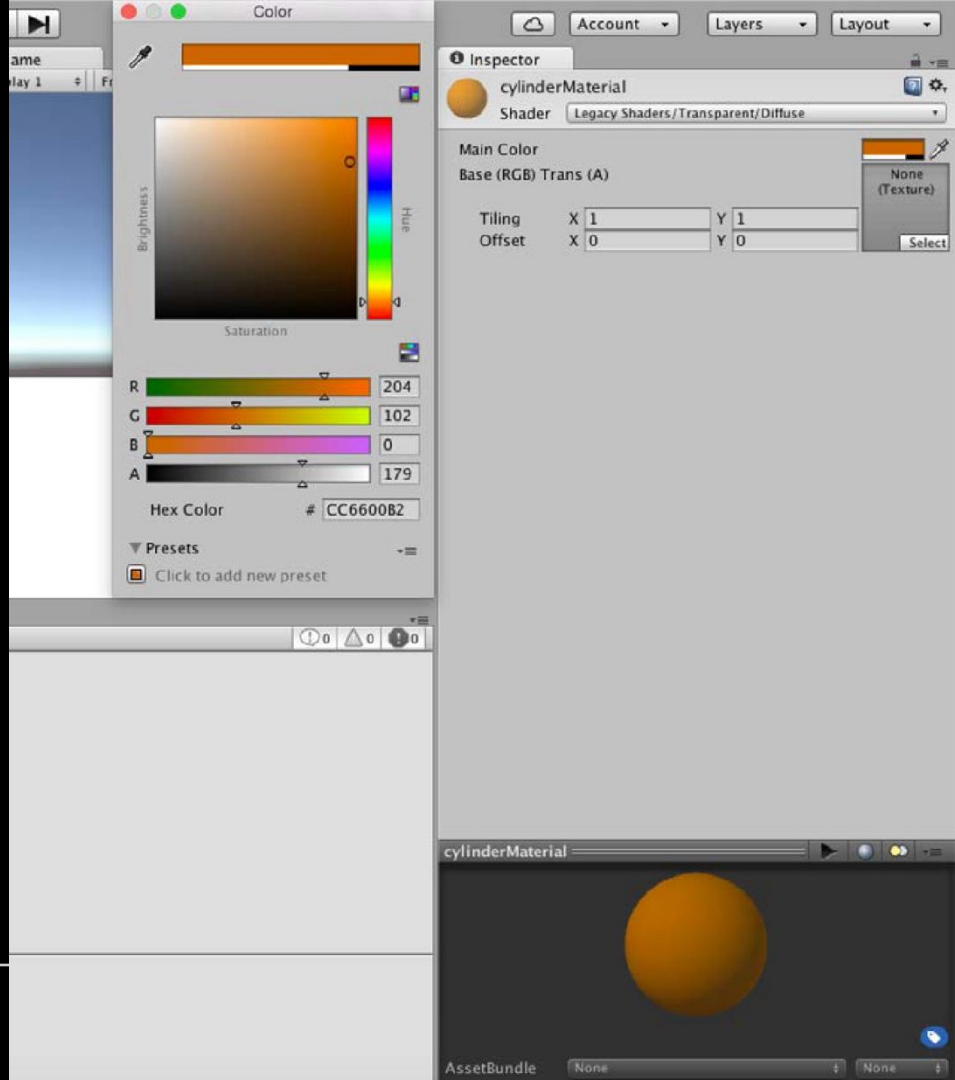
Blue RGBA is (0, 0, 1, 1)

Gray RGBA is (0.5, 0.5, 0.5, 1)

Clear Completely transparent. RGBA is (0, 0, 0, 0)

Magenta? Yellow? cyan?

Unity color


Color
Brightness
Hue
Saturation
R 204
G 102
B 0
A 179
Hex Color # CC6600B2
Presets
Click to add new preset
Account
Layers
Layout
Inspector
cylinderMaterial
Shader Legacy Shaders/Transparent/Diffuse
Main Color
Base (RGB) Trans (A)
None (Texture)
Select
Tiling X 1 Y 1
Offset X 0 Y 0
cylinderMaterial
AssetBundle
None
None

If statement

```
public class Ifstatement : MonoBehaviour { void Update()      {

if (Input.GetKeyDown(KeyCode.C)) {
        GetComponent<Renderer> ().material.color = Color.cyan;
}
if (Input.GetKeyDown(KeyCode.M)) {
        GetComponent<Renderer>().material.color = Color.magenta;
}
if (Input.GetKeyDown(KeyCode.Y)) {
        GetComponent<Renderer>().material.color = Color.yellow;

}
}}
```

If statement

```
public class Ifstatement : MonoBehaviour { void Update()      {

if (Input.GetKeyDown(KeyCode.C)) {
        GetComponent<Renderer>().material.color=new Color(0.0f, 0.5f, 0.2f, 0.9f);

}
if (Input.GetKeyDown(KeyCode.M)) {
        GetComponent<Renderer>().material.color = Color.magenta;
}
if (Input.GetKeyDown(KeyCode.Y)) {
        GetComponent<Renderer>().material.color = Color.yellow;

}
}}
```

Interaction - Key and Button input

The input manager allows to name an input and specify a key or button for it.

You can access keys on the keyboard and mouse buttons through scripting interface.

Edit > Project Settings >Input

Interaction - Key and Button input

The input manager allows to name an input and specify a key or button for it.

You can access keys on the keyboard and mouse buttons through scripting interface.

Edit > Project Settings >Input

Interaction - Key and Button input

Input.GetButtonDown("Fire1"))
Mouse button left mouse 0

Input.GetButtonDown("Fire2"))
Mouse button right mouse 1

Input.GetKeyDown(”Jump"))

space key space

Button input interaction

```
public class buttonInput : MonoBehaviour { void Update()      {
if  (Input.GetButtonDown("Fire1"))
{
GetComponent<Renderer>().material.color=new Color(0.0f, 0.5f, 0.2f, 0.9f);
}
if  (Input.GetButtonDown("Fire2"))
{
GetComponent<Renderer>().material.color = Color.magenta;
}
if (Input.GetButtonDown("Jump"))
{
GetComponent<Renderer>().material.color = Color.yellow;
}
}}
```

Unity documentation - KeyCode

unity | DOCUMENTATION Manual Scripting API Search scripting... unity3d.com

Version: **2018.2** (switch to 2018.3b or 2017.4) C#

- ImagePosition
- IMECompositionMode
- JointDriveMode
- JointLimitState2D
- JointProjectionMode
- KeyCode
- LegDof
- LightmapBakeType
- LightmapsMode
- LightmapsModeLegacy
- LightRenderMode
- LightShadowCasterMode
- LightShadows
- LightType
- LineAlignment
- LineTextureMode
- LocationServiceStatus
- LODFadeMode
- LogType

Properties

None	Not assigned (never returned as the result of a keystroke).
Backspace	The backspace key.
Delete	The forward delete key.
Tab	The tab key.
Clear	The Clear key.
Return	Return key.
Pause	Pause on PC machines.
Escape	Escape key.
Space	Space key.
Keypad0	Numeric keypad 0.
Keypad1	Numeric keypad 1.
Keypad2	Numeric keypad 2.
Keypad3	Numeric keypad 3.

Interaction - Key and Button input

Create new object Sphere, add rigidbody component

Create new Script KeyInput.cs

Assign script to the sphere

Key input

```
void Update () {

if (Input.GetKey("up")) {

Debug.Log("up arrow key is held down");
}

if (Input.GetKey("down")) {

Debug.Log("down arrow key is held down");
}

if (Input.GetKeyDown(KeyCode.Space)) {

GetComponent<Rigidbody>().AddForce(transform.forward * 200f);
}

}}
```

Interaction - Key and Button input

Uncheck Use Gravity in the inspector

Test the scene

Interaction - Key and Button input

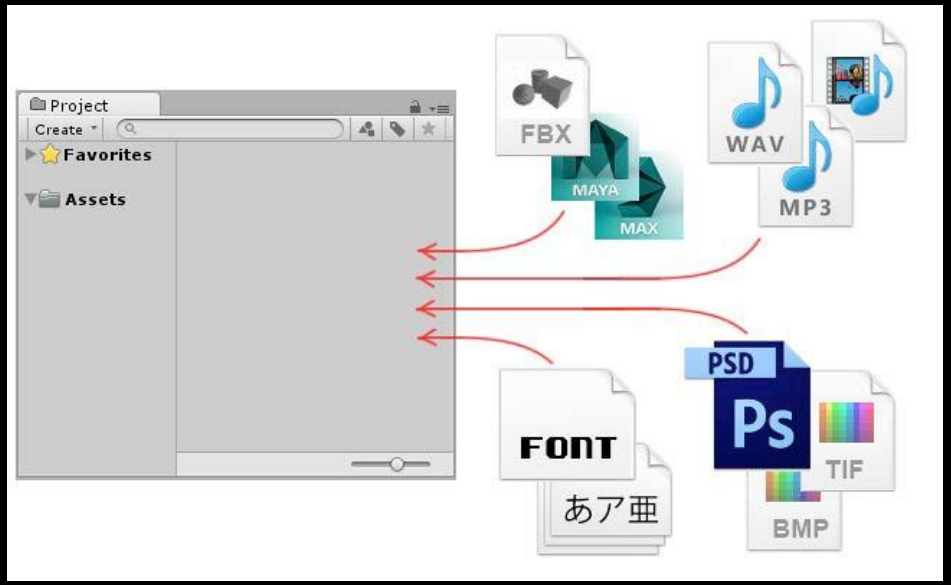
Add useGravity to Rigidbody component in the script

```
GetComponent.<Rigidbody>().AddForce(transform.forward * 500f);

GetComponent.<Rigidbody>().useGravity = true;

}
```

Project Organization

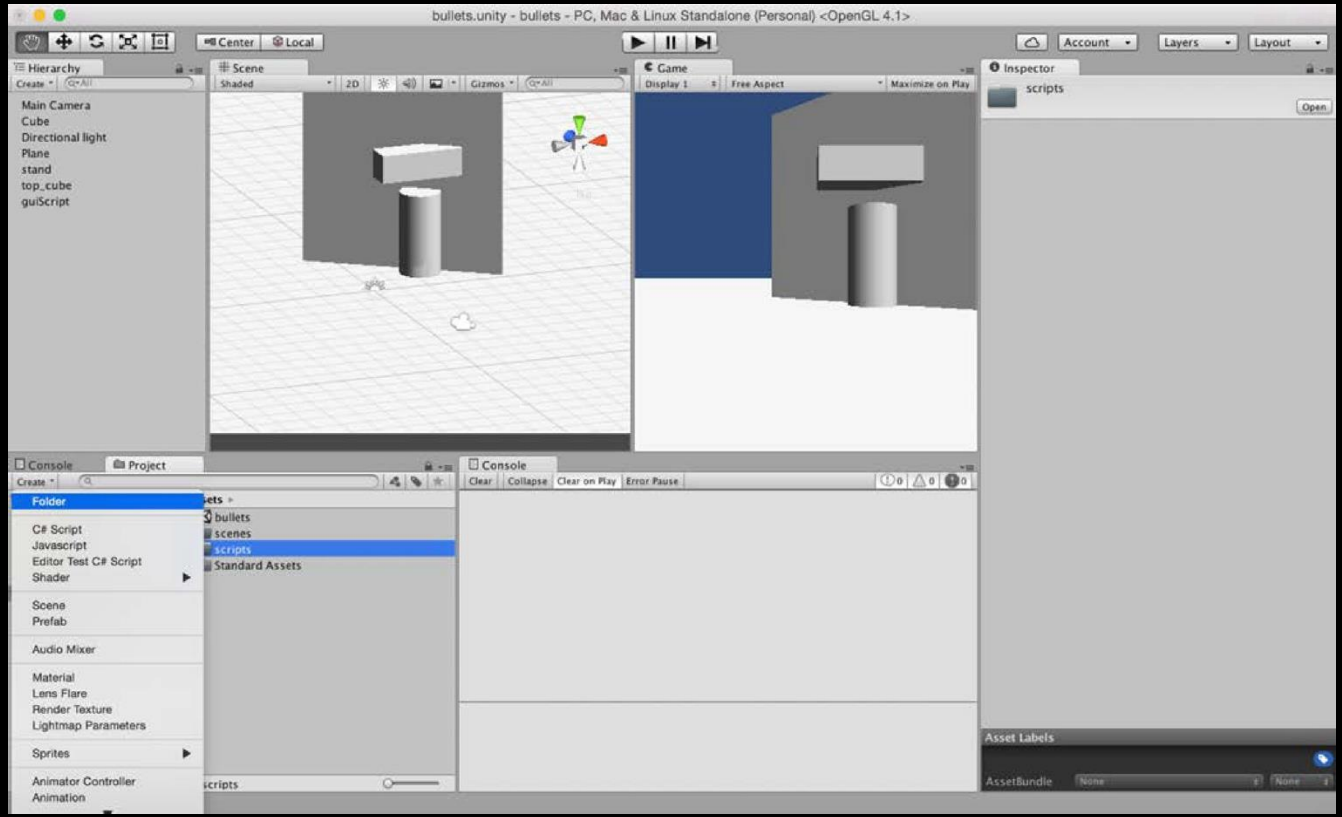
Assets are the models, textures, sounds and all other media files

- Audio Files
- Materials
- Meshes
- Textures
- Prefabs
- Scripts



Project Organization

RMC
Project>
Create Folder



Project Organization

Assets > Folder > Rename

Materials

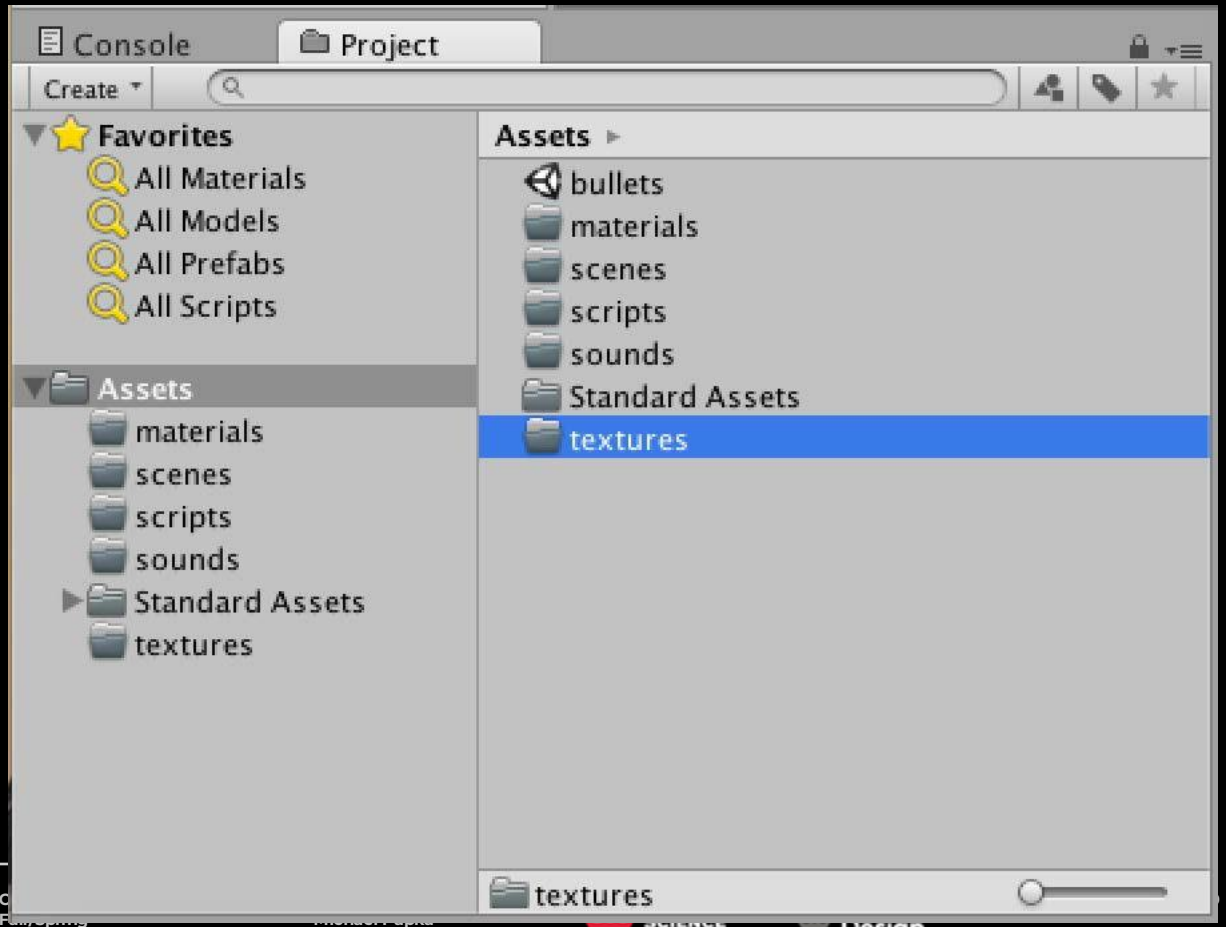
Scripts

Scenes

Prefabs

Textures

Sounds



Project Organization

Keep your assets organized

Be consistent with naming. If you decide to use camel case
for directory names and low letters for assets, stick to that convention.

Use lowercase folder names

Use camelCase naming convention if needed

Sort all the assets in the corresponding folders

Project Organization

Do not store any asset files in the root directory.
Use subdirectories whenever possible.

Do not create any additional directories in the root directory,
unless you really need to.

Use 3rd-Party to store assets imported from the Asset Store.
They usually have their own structure that shouldn't be altered.

Project Organization

Audio
Music
SFX
Materials
Models
Plugins
Prefabs

Resources
Textures
Sandbox
Scenes
Levels
Scripts
Shaders

Unity Components

Components are the nuts & bolts of objects and behaviors in a game.

A GameObject is a container for many different Components.

Unity Components

Audio components
Physics components
The GameObject
Image effect scripts
Settings managers
Animation components
Asset components
Mesh components
Network group
Particle components
Rendering components
Transform component
UnityGUI components

Unity Components

Audio components Implement sound in Unity

- Audio Listener

Add this to a Camera to get 3D positional sound.

- Audio Source

Add this Component to a GameObject to make it play a sound

- Audio Effects

Unity Components

The Game Object

GameObjects are containers for all other Components

All objects in your scene are inherently GameObjects
a way to group objects

They are containers that hold Components,
which implement actual functionality.

Ex. a Light is a Component which is attached to a GameObject.

Unity Components

Mesh components

3D Meshes are the main graphics primitive of Unity

Various components exist in Unity to render regular or skinned meshes, trails or 3D lines

Mesh Filter
Mesh Renderer
Skinned Mesh Renderer
Text Mesh

Unity Components

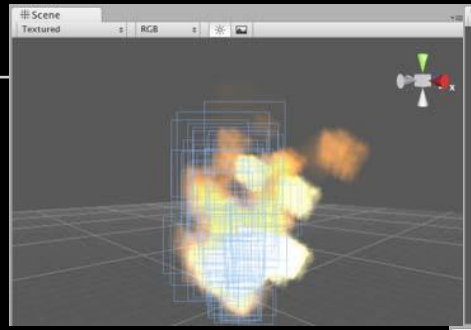
Particle components

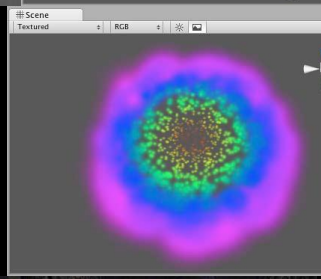
Particle Systems are used to make effects

Smoke
Steam
Fire
Atmospheric effects
Rain
Fog

Particle systems in Unity work by using one or two textures (2D), and drawing them many times, creating a chaotic random effect

Particle Components





Ellipsoid
- spawns particles inside a sphere

Line Renderer
- draws a straight line between 2+ points in 3D space

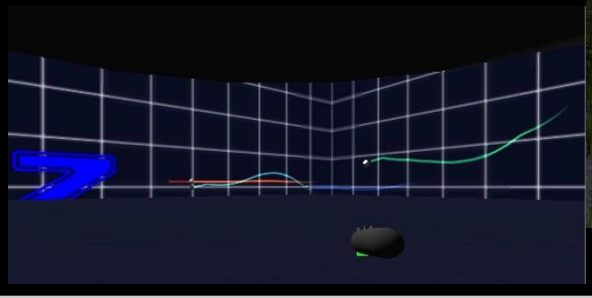
Trail Renderer
- makes trails behind moving objects in the scene

Mesh Particle Emitter
- emits particles from the surface of the mesh

Particle Animator
- Moves particles over time

Particle Renderer

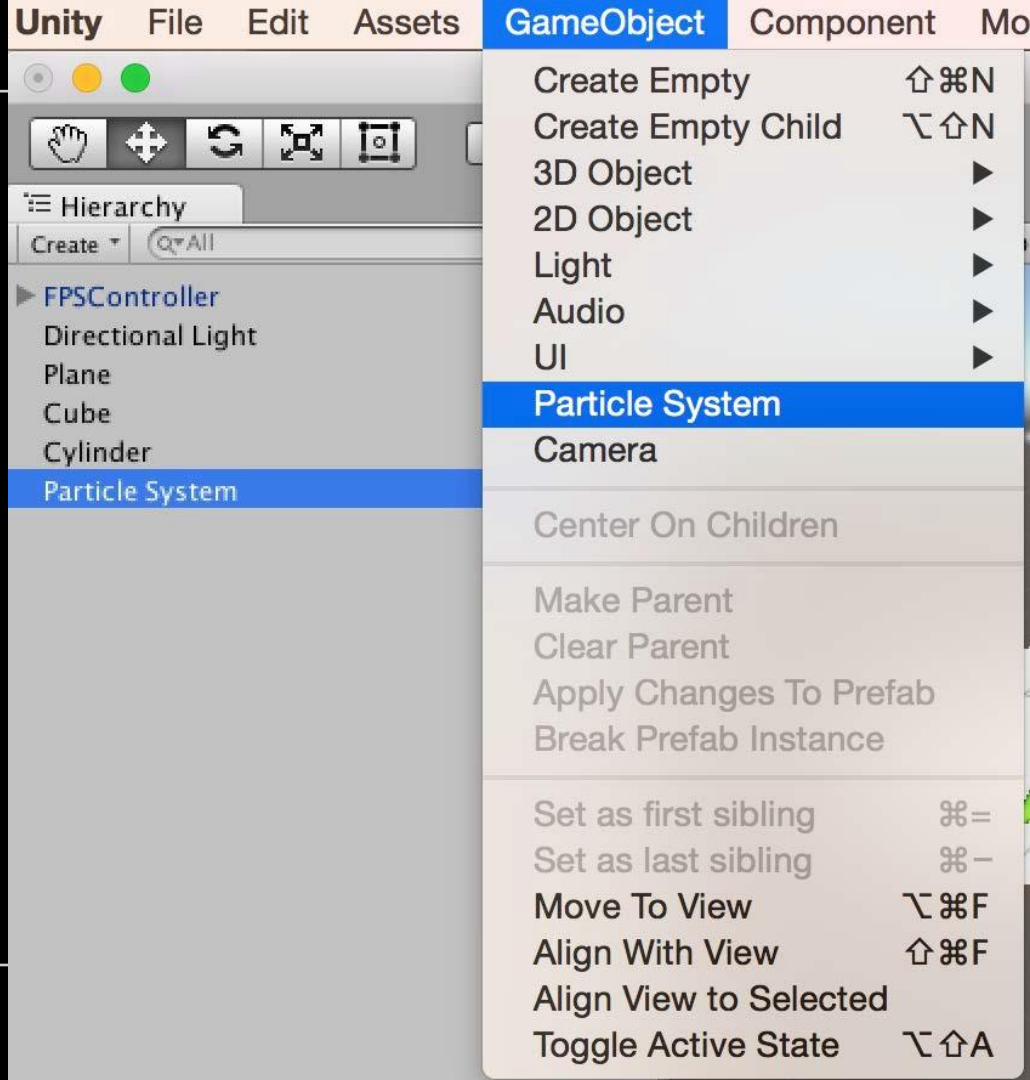
Particle Collider



Particle Components

Open your Unity scene
GameObject >Particle System

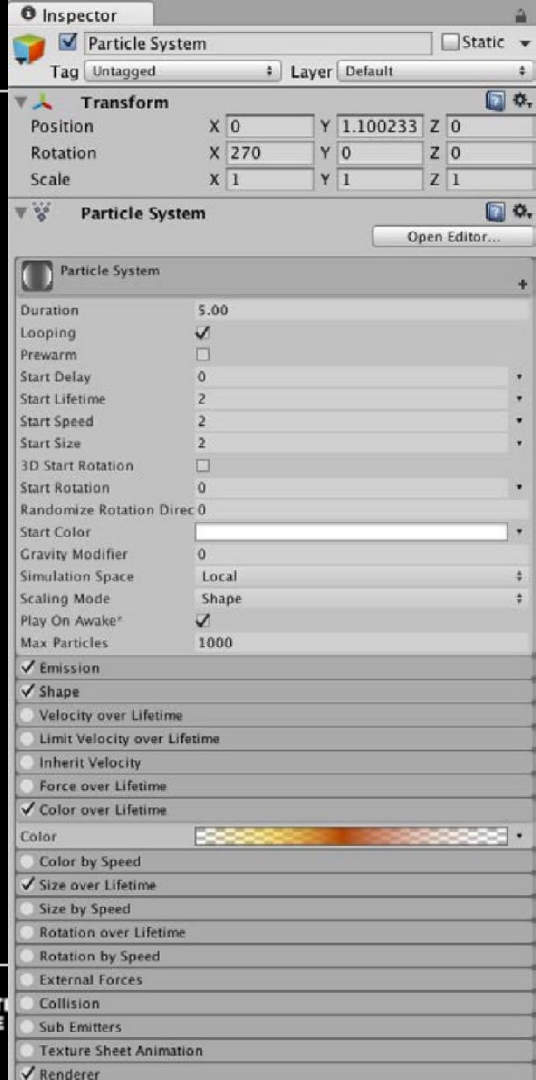
Rename it “Confetti”



Particle Components

Open your Unity scene
GameObject >Particle System

Rename it “Confetti”



Particle script

```
public class Confetti: MonoBehaviour {

public ParticleSystem confettiEmitter;
void Start () {

}

void Update () {

if (Input.GetButtonDown("Fire1")) {
confettiEmitter.Emit(30);
  }
 }
}
```

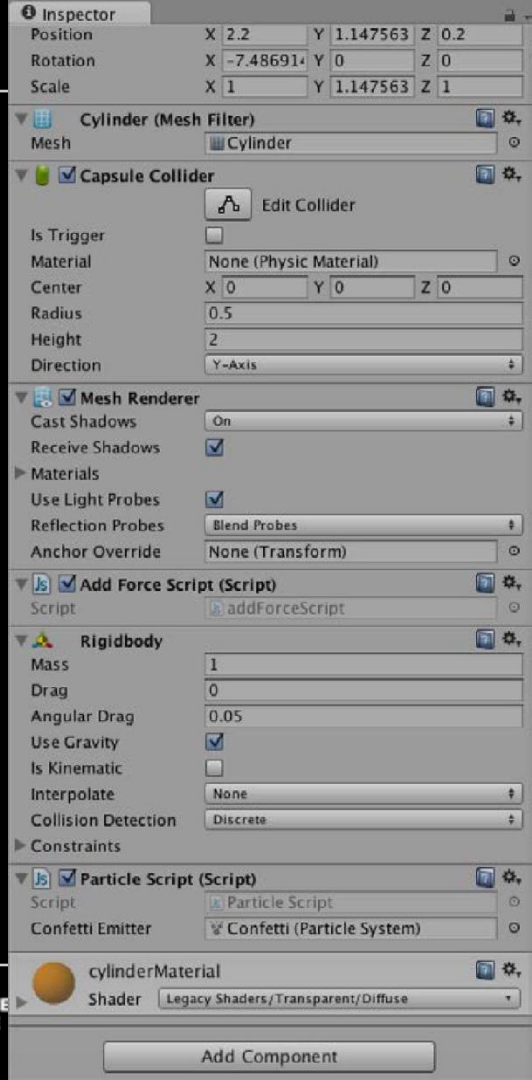
Particle script

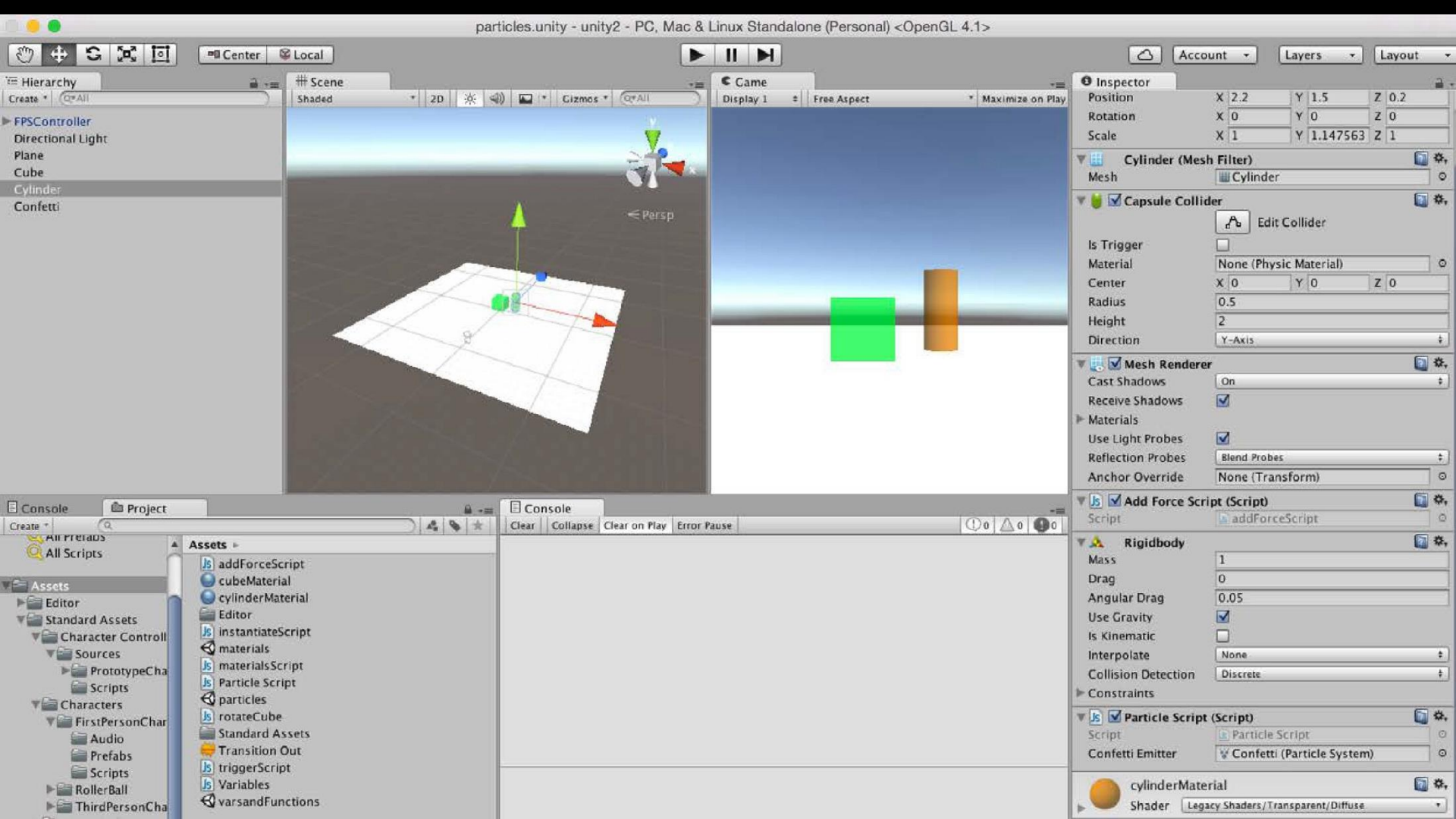
Assign “ParticleScript” to a cylinder
(drag and drop)

Assign Confetti Particles to
Confetti Emitter variable in the Inspector
(drag and drop from Hierarchy)

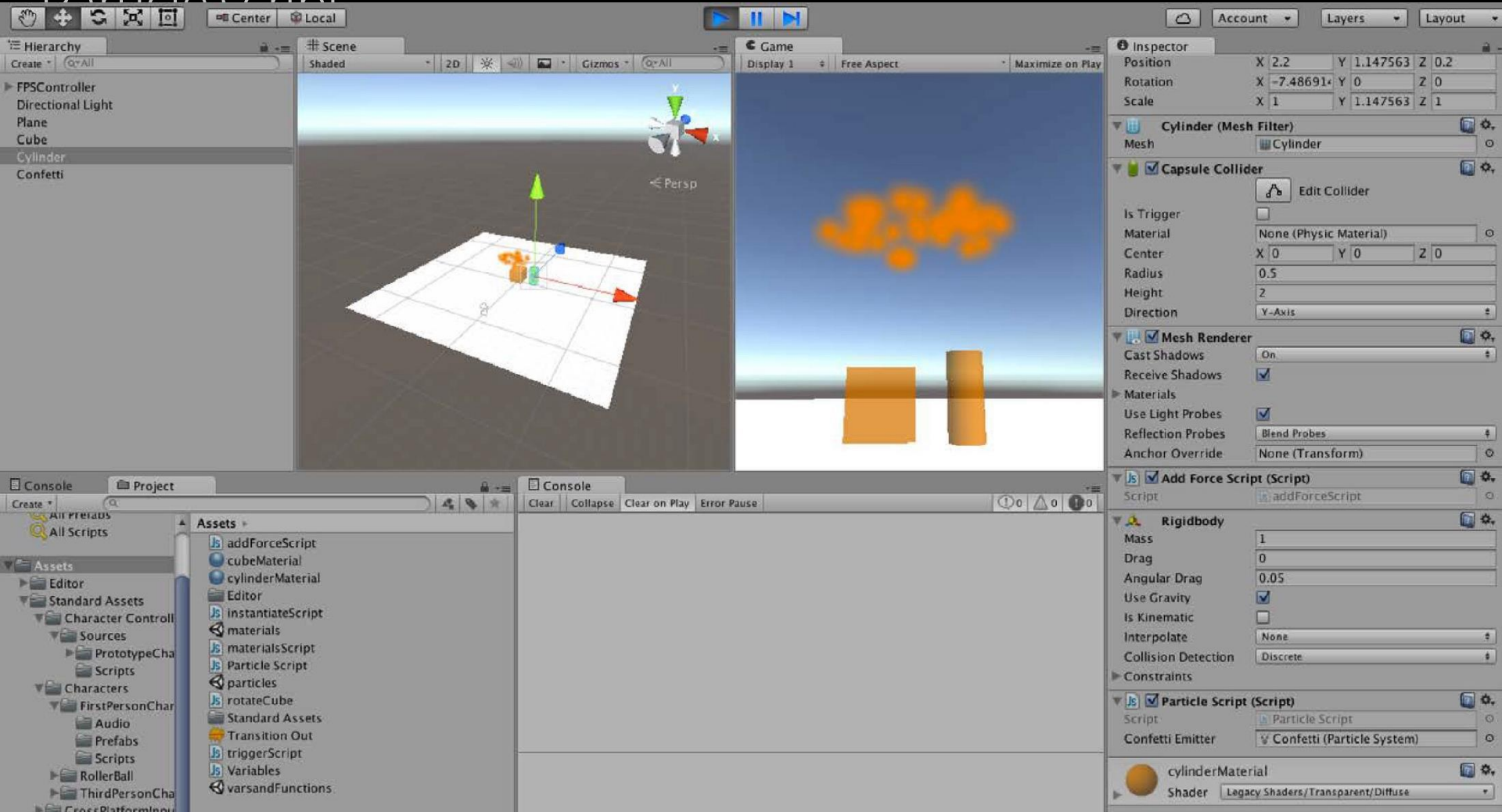
Uncheck “Play on Awake” box in the Inspector
Uncheck “Looping” box

With selected Confetti object in Hierarchy




particles.unity - unity2 - PC, Mac & Linux Standalone (Personal) <OpenGL 4.1>
Center
Local
Account
Layers
Layout
Hierarchy
Create
FPSController
Directional Light
Plane
Cube
Cylinder
Confetti
Scene
Shaded
2D
Gizmos
Persp
Game
Display 1
Free Aspect
Maximize on Play
Inspector
Position
Rotation
Scale
Cylinder (Mesh Filter)
Mesh
Cylinder
Capsule Collider
Edit Collider
Is Trigger
Material
None (Physic Material)
Center
Radius
Height
Direction
Y-Axis
Mesh Renderer
Cast Shadows
On
Receive Shadows
Materials
Use Light Probes
Reflection Probes
Blend Probes
Anchor Override
None (Transform)
Add Force Script (Script)
Script
addForceScript
Rigidbody
Mass
Drag
Angular Drag
Use Gravity
Is Kinematic
Interpolate
None
Collision Detection
Discrete
Constraints
Particle Script (Script)
Particle Script
Confetti Emitter
Confetti (Particle System)
cylinderMaterial
Shader
Legacy Shaders/Transparent/Diffuse
Console
Project
Create
All Prefabs
All Scripts
Assets
Editor
Standard Assets
Character Controll
Sources
PrototypeCha
Scripts
Characters
FirstPersonChar
Audio
Prefabs
Scripts
RollerBall
ThirdPersonCha
addForceScript
cubeMaterial
cylinderMaterial
Editor
instantiateScript
materials
materialsScript
Particle Script
particles
rotateCube
Standard Assets
Transition Out
triggerScript
Variables
varsandFunctions
Console
Clear
Collapse
Clear on Play
Error Pause

Particle script



Particle script

Test the mouse button input and interaction

The particle should emit 30 particles on mouse click

Parenting

Parent-Child Relationship in Unity

The Child Game Object will inherit the behavior of its parent
It will move, rotate, scale exactly as its Parent does.

Similar to Arm/Body relationship- whenever your body moves, your arm moves along with it

This is a way of grouping objects together

Children can have their own children and etc.

Complex parent-child structure

Represented in the Hierarchy window

Particle script

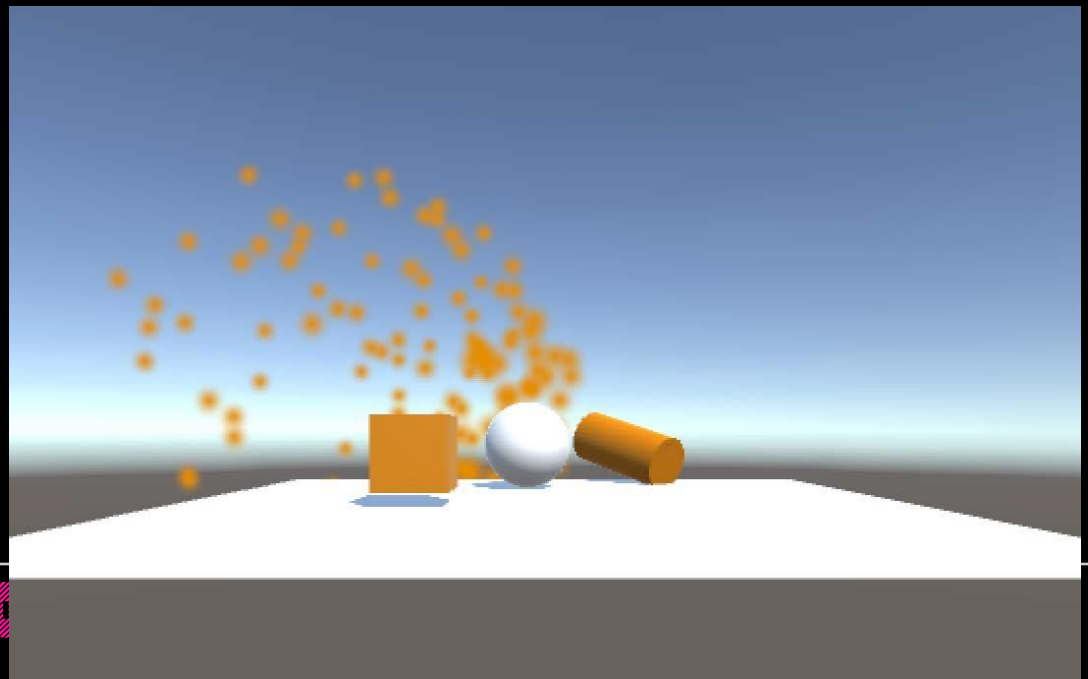
Add another if statement to add force to push cylinder forward:

```
public ParticleSystem confettiEmitter;

void Update () {

if (Input.GetButtonDown("Fire1")) {
confettiEmitter.Emit(30);
  }
if (Input.GetKeyDown(KeyCode.Tab)) {
GetComponent<Rigidbody>().AddForce(transform.forward * 200f);
}
 }
}
```

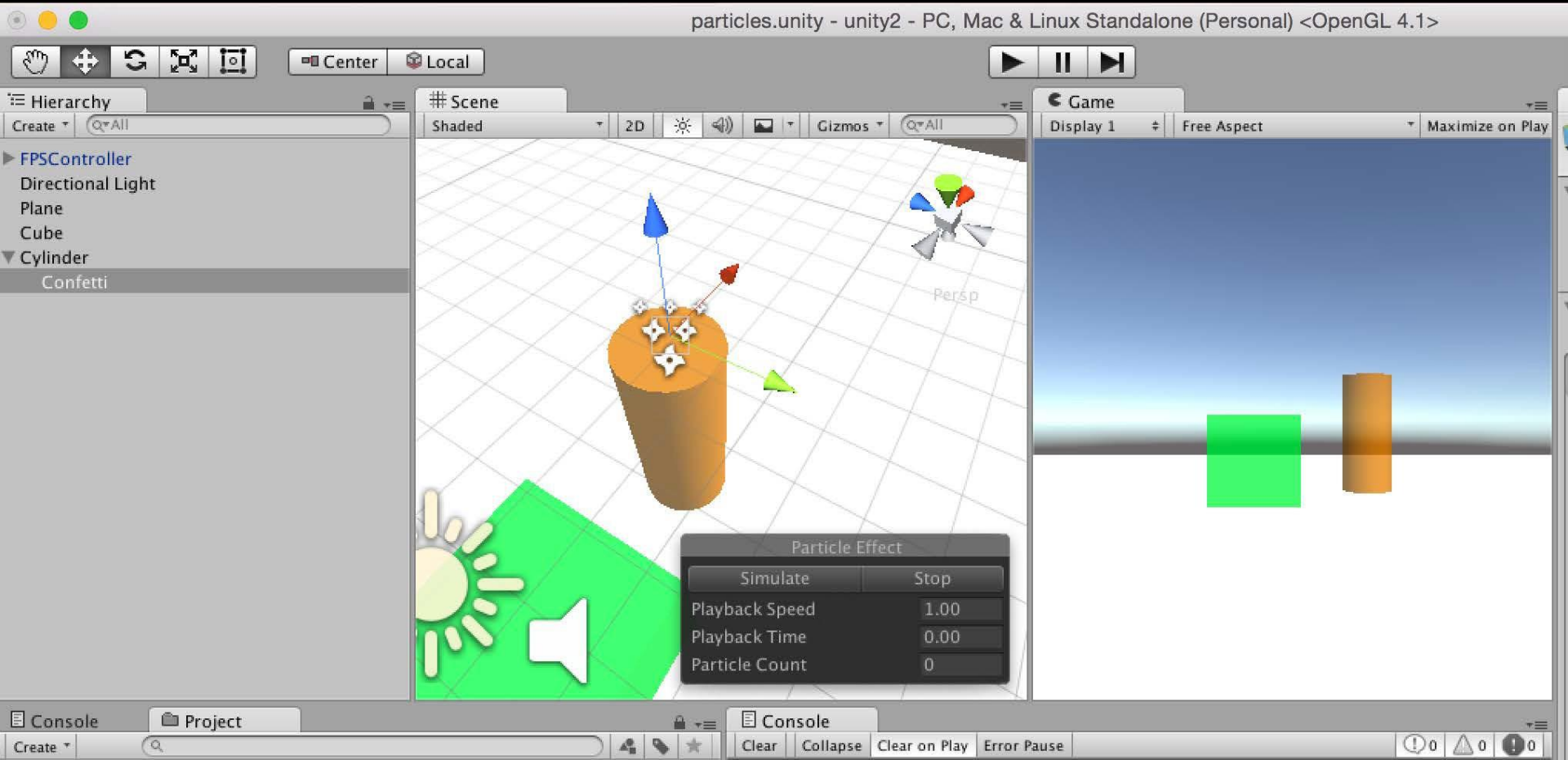
Parenting

Drag and drop Confetti particles inside the Cylinder object in the Hierarchy window

Test the game



Parenting


particles.unity - unity2 - PC, Mac & Linux Standalone (Personal) <OpenGL 4.1>
Center
Local
Hierarchy
Create
All
FPSController
Directional Light
Plane
Cube
Cylinder
Confetti
Scene
Shaded
2D
Gizmos
All
Persp
Particle Effect
Simulate
Stop
Playback Speed
1.00
Playback Time
0.00
Particle Count
0
Game
Display 1
Free Aspect
Maximize on Play
Console
Project
Create
Console
Clear
Collapse
Clear on Play
Error Pause
0
0
0

Prefabs

Prefabs are prefabricated objects

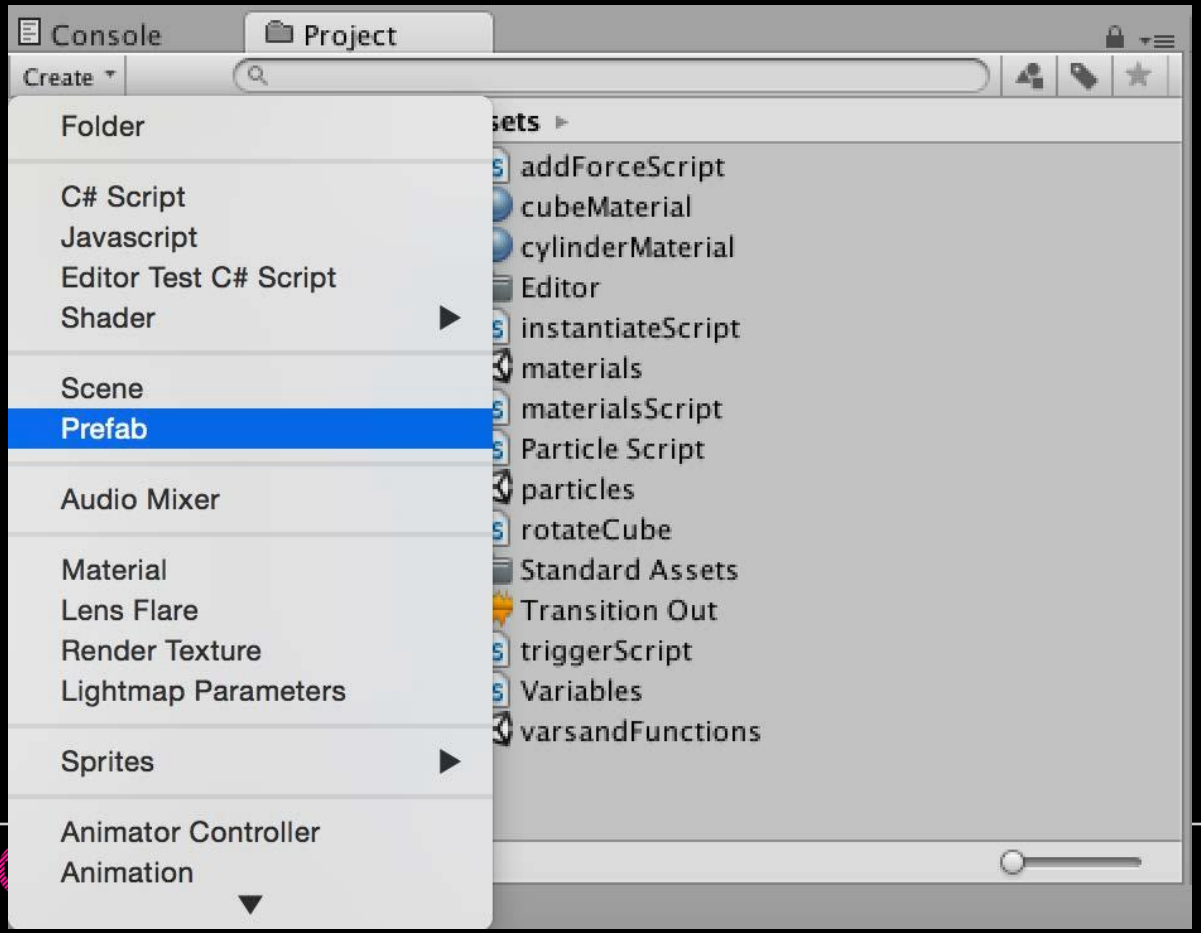
Game Object > 3D > Sphere

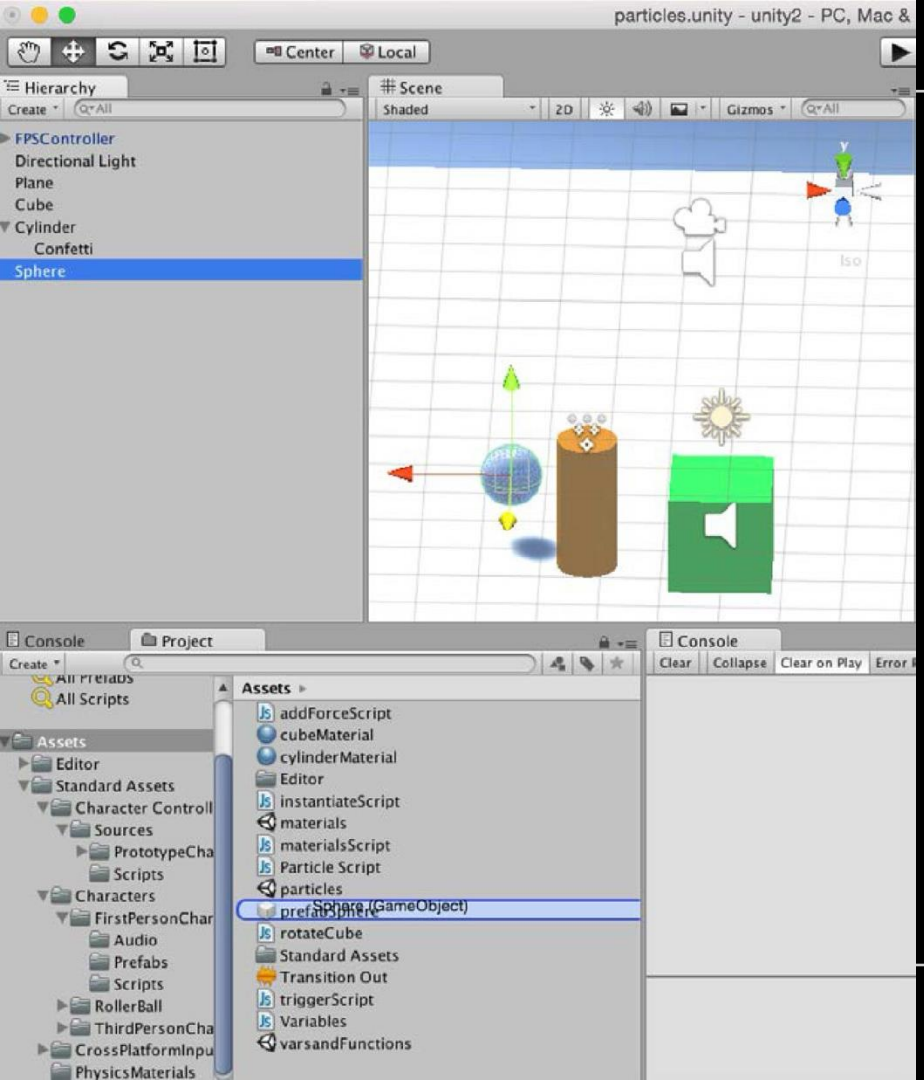
Create Prefab “PrefabSphere” (drag and drop to project window)

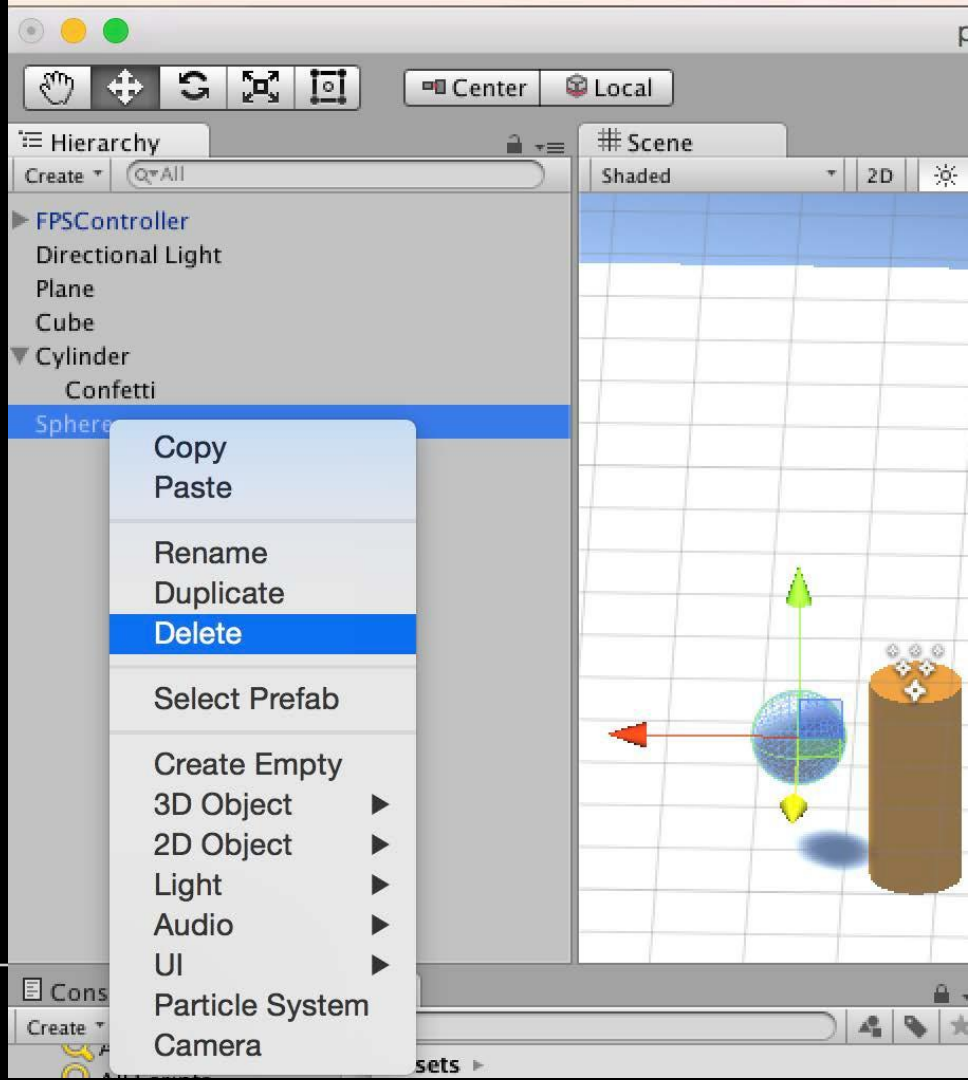
Inspector : Add RigidBody Component

Delete Sphere in the Hierarchy window / scene

Prefabs


Console
Project
Create
Folder
C# Script
Javascript
Editor Test C# Script
Shader
Scene
Prefab
Audio Mixer
Material
Lens Flare
Render Texture
Lightmap Parameters
Sprites
Animator Controller
Animation
addForceScript
cubeMaterial
cylinderMaterial
Editor
instantiateScript
materials
materialsScript
Particle Script
particles
rotateCube
Standard Assets
Transition Out
triggerScript
Variables
varsandFunctions


particles.unity - unity2 - PC, Mac &
Center
Local
Hierarchy
Create
FPSController
Directional Light
Plane
Cube
Cylinder
Confetti
Sphere
Scene
Shaded
2D
Gizmos
Console
Project
Clear
Collapse
Clear on Play
All Prefabs
All Scripts
Assets
Editor
Standard Assets
Character Controll
Sources
PrototypeCha
Scripts
Characters
FirstPersonChar
Audio
Prefabs
Scripts
RollerBall
ThirdPersonCha
CrossPlatformInpu
PhysicsMaterials
addForceScript
cubeMaterial
cylinderMaterial
Editor
instantiateScript
materials
materialsScript
Particle Script
particles
prefabSphere
Sphere (GameObject)
rotateCube
Standard Assets
Transition Out
triggerScript
Variables
varsandFunctions


Center
Local
Hierarchy
Create
All
FPSController
Directional Light
Plane
Cube
Cylinder
Confetti
Sphere
Scene
Shaded
2D
Copy
Paste
Rename
Duplicate
Delete
Select Prefab
Create Empty
3D Object
2D Object
Light
Audio
UI
Particle System
Camera

Instantiate

Instantiate is to create objects during run-time
(as the game is being played)

Instantiate function takes three parameters;

- the object we want to create
- the 3D position of the object
- the rotation of the object

Instantiate

1. Which object to instantiate?

the best way is to expose a variable

We can state which object to instantiate by using drag and drop to assign a game object to this variable prefab

2 Where to instantiate it?

create the new game object to locate new prefab whenever the Fire2 button is pressed.

Destroy script

To clean generated prefabs from the scene, attach the following script “PrefabDestruction” to the prefabSphere

```
public class PrefabDestruction : MonoBehaviour {

void Start()

{

Destroy (gameObject, 5.5f);

}
}
```

Instantiate

Create UsingInstantiate script and assign it to the cylinder:

```
public class UsingInstantiate : MonoBehaviour {
public Rigidbody spherePrefab; public Transform placetoStart;
void Update () {

if(Input.GetButtonDown("Fire2")) {

Rigidbody newInstance;
newInstance = Instantiate (spherePrefab, placetoStart.position,
placetoStart.rotation);
newInstance.AddForce(placetoStart.forward * 50);
}
} }
```