유니티(Unity)를 활용한

그래픽스 프로그래밍

13 Physics, Collision

1

Physics, Collision

Unity Built-in 3D Physics Engine

- Unity helps you simulate physics in your Project to ensure that the objects correctly accelerate and respond to collisions, gravity, and various other forces.

- Unity provides different physics engine implementations which you can use according to your Project needs
  - 3D/2D, object-oriented/data-oriented

- Built-in physics engines for object-oriented projects
  - If your project is object-oriented, use the Unity's built-in physics engine that corresponds to your needs
  - Built-in 3D physics (Nvidia PhysX engine integration)
  - Built-in 2D physics (Box2D engine integration)

Unity Built-in 3D Physics Engine

- Physics engine packages for data-oriented projects
  - If your project uses Unity's Data-Oriented Technology Stack (DOTS), you need to install a dedicated DOTS physics package. The available packages are
  - Unity Physics package
    - the DOTS physics engine you need to install by default to simulate physics in any data-oriented project.
  - Havok Physics for Unity package
    - an implementation of the Havok physics engine for Unity, to use as an extension of the Unity Physics package. Note that this package is subject to a specific licensing scheme.

Built-in 3D Physics

- Built-in 3D physics (Nvidia PhysX engine integration)

	Description
Character control	Configure physics-based character control for first-person and third-person characters.
Rigidbody physics	Apply physics-based behavior to **GameObjects**
Collision	Use **colliders** to configure collisions between GameObjects.
Joints	Apply and configure **joints** that connect GameObjects and simulate physical forces for pivoting, movement, and restriction.
Articulations	Configure complex systems of rigid bodies and joints.
Ragdoll physics	Configure **ragdoll** physics for characters.
Cloth	Simulate fabric movement for character clothing and other in-application textiles.
Multi-scene physics	Manage different physics contexts in one project with multiple dedicated physics **scenes**.

Built-in 3D Physics

- Built-in 3D physics (Nvidia PhysX engine integration)

	Description
Character control	Configure physics-based character control for first-person and third-person characters.
Rigidbody physics	Apply physics-based behavior to **GameObjects**
Collision	Use **colliders** to configure collisions between GameObjects.
Joints	Apply and configure **joints** that connect GameObjects and simulate physical forces for pivoting, movement, and restriction.
Articulations	Configure complex systems of rigid bodies and joints.
Ragdoll physics	Configure **ragdoll** physics for characters.
Cloth	Simulate fabric movement for character clothing and other in-application textiles.
Multi-scene physics	Manage different physics contexts in one project with multiple dedicated physics **scenes**.

Built-in 3D Physics

- Built-in 3D physics (Nvidia PhysX engine integration)

	Description
Character control	Configure physics-based character control for first-person and third-person characters.
Rigidbody physics	Apply physics-based behavior to **GameObjects**
Collision	Use **colliders** to configure collisions between GameObjects.
Joints	Apply and configure **joints** that connect GameObjects and simulate physical forces for pivoting, movement, and restriction.
Articulations	Configure complex systems of rigid bodies and joints.
Ragdoll physics	Configure **ragdoll** physics for characters.
Cloth	Simulate fabric movement for character clothing and other in-application textiles.
Multi-scene physics	Manage different physics contexts in one project with multiple dedicated physics **scenes**.

Built-in 3D Physics

- Built-in 3D physics (Nvidia PhysX engine integration)

	Description
Character control	Configure physics-based character control for first-person and third-person characters.
Rigidbody physics	Apply physics-based behavior to **GameObjects**
Collision	Use **colliders** to configure collisions between GameObjects.
Joints	Apply and configure **joints** that connect GameObjects and simulate physical forces for pivoting, movement, and restriction.
Articulations	Configure complex systems of rigid bodies and joints.
Ragdoll physics	Configure **ragdoll** physics for characters.
Cloth	Simulate fabric movement for character clothing and other in-application textiles.
Multi-scene physics	Manage different physics contexts in one project with multiple dedicated physics **scenes**.

Built-in 3D Physics

- Built-in 3D physics (Nvidia PhysX engine integration)

	Description
Character control	Configure physics-based character control for first-person and third-person characters.
Rigidbody physics	Apply physics-based behavior to **GameObjects**
Collision	Use **colliders** to configure collisions between GameObjects.
Joints	Apply and configure **joints** that connect GameObjects and simulate physical forces for pivoting, movement, and restriction.
Articulations	Configure complex systems of rigid bodies and joints.
Ragdoll physics	Configure **ragdoll** physics for characters.
Cloth	Simulate fabric movement for character clothing and other in-application textiles.
Multi-scene physics	Manage different physics contexts in one project with multiple dedicated physics **scenes**.

Built-in 3D Physics

- Built-in 3D physics (Nvidia PhysX engine integration)

	Description
Character control	Configure physics-based character control for first-person and third-person characters.
Rigidbody physics	Apply physics-based behavior to **GameObjects**
Collision	Use **colliders** to configure collisions between GameObjects.
Joints	Apply and configure **joints** that connect GameObjects and simulate physical forces for pivoting, movement, and restriction.
Articulations	Configure complex systems of rigid bodies and joints.
Ragdoll physics	Configure **ragdoll** physics for characters.
Cloth	Simulate fabric movement for character clothing and other in-application textiles.
Multi-scene physics	Manage different physics contexts in one project with multiple dedicated physics **scenes**.

Built-in 3D Physics

- Built-in 3D physics (Nvidia PhysX engine integration)

	Description
Character control	Configure physics-based character control for first-person and third-person characters.
Rigidbody physics	Apply physics-based behavior to **GameObjects**
Collision	Use **colliders** to configure collisions between GameObjects.
Joints	Apply and configure **joints** that connect GameObjects and simulate physical forces for pivoting, movement, and restriction.
Articulations	Configure complex systems of rigid bodies and joints.
Ragdoll physics	Configure **ragdoll** physics for characters.
Cloth	Simulate fabric movement for character clothing and other in-application textiles.
Multi-scene physics	Manage different physics contexts in one project with multiple dedicated physics **scenes**.

Built-in 3D Physics

- Built-in 3D physics (Nvidia PhysX engine integration)

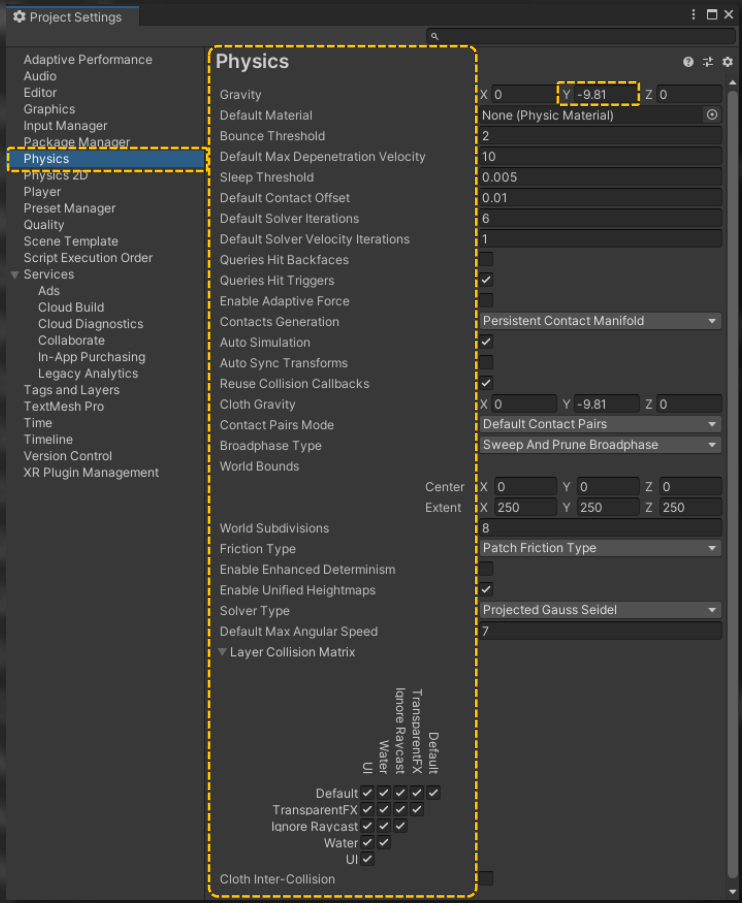
	Description
Character control	Configure physics-based character control for first-person and third-person characters.
Rigidbody physics	Apply physics-based behavior to **GameObjects**
Collision	Use **colliders** to configure collisions between GameObjects.
Joints	Apply and configure **joints** that connect GameObjects and simulate physical forces for pivoting, movement, and restriction.
Articulations	Configure complex systems of rigid bodies and joints.
Ragdoll physics	Configure **ragdoll** physics for characters.
Cloth	Simulate fabric movement for character clothing and other in-application textiles.
Multi-scene physics	Manage different physics contexts in one project with multiple dedicated physics **scenes**.

Built-in 3D Physics

- Built-in 3D physics (Nvidia PhysX engine integration)

	Description
Character control	Configure physics-based character control for first-person and third-person characters.
Rigidbody physics	Apply physics-based behavior to **GameObjects**
Collision	Use **colliders** to configure collisions between GameObjects.
Joints	Apply and configure **joints** that connect GameObjects and simulate physical forces for pivoting, movement, and restriction.
Articulations	Configure complex systems of rigid bodies and joints.
Ragdoll physics	Configure **ragdoll** physics for characters.
Cloth	Simulate fabric movement for character clothing and other in-application textiles.
Multi-scene physics	Manage different physics contexts in one project with multiple dedicated physics **scenes**.

Default Settings for Unity Physics Engine



Collider

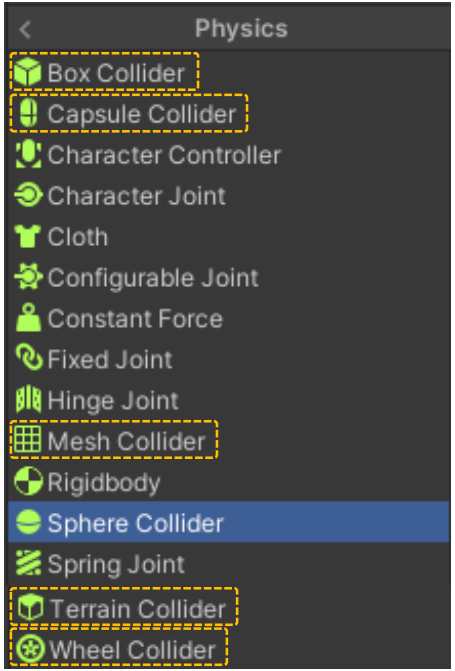
- Unity handles collision between GameObjects with colliders.
- GameObjects must have a RigidBody component attached to them for collisions to occur.
- A collider is invisible, and does not need to be the exact same shape as the GameObject’s mesh. A rough approximation of the mesh is often more efficient and indistinguishable in gameplay.

Collider

Collider

Type of Colliders

- Box Collider
- Capsule Collider
- Mesh Collider
- Sphere Collider
- Terrain Collider
- Wheel Collider

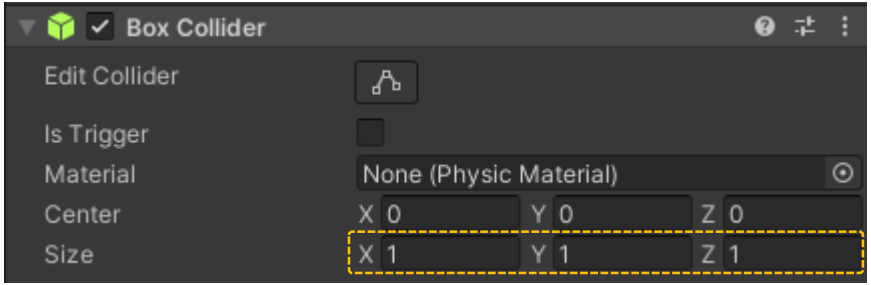


이미지 출처 : Unity

Box Collider

Box collider

- The Box Collider is a basic cuboid-shaped collision primitive.



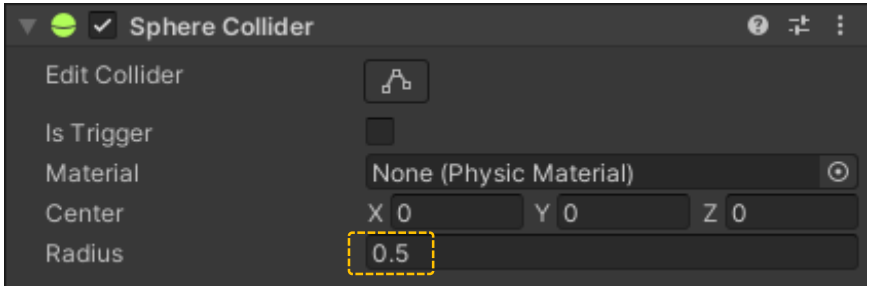
이미지 출처 :Unity

Property	Function
Is Trigger	If enabled, this Collider is used for triggering events, and is ignored by the physics engine.
Material	Reference to the Physics Material that determines how this Collider interacts with others.
Center	The position of the Collider in the object's local space.
Size	The size of the Collider in the X, Y, Z directions.

Sphere Collider

Sphere collider

- The Sphere Collider is a basic collision primitive.



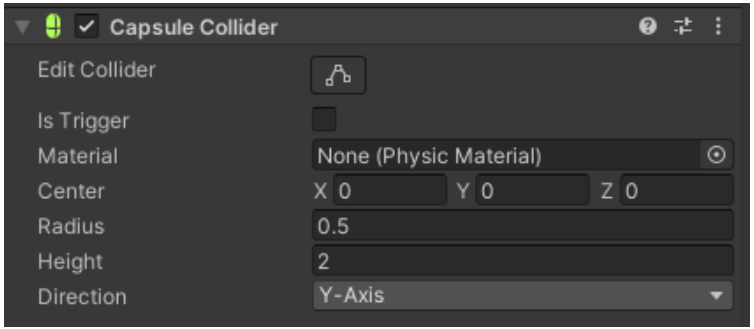
이미지 출처 :Unity

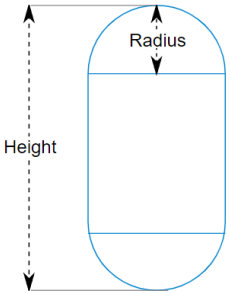
Property	Function
Is Trigger	If enabled, this Collider is used for triggering events, and is ignored by the physics engine.
Material	Reference to the Physics Material that determines how this Collider interacts with others.
Center	The position of the Collider in the object's local space.
Radius	The size of the Collider

Capsule Collider

Capsule collider

- The Capsule Collider is made of two half-spheres joined together by a cylinder.





이미지 출처 :Unity

Capsule Collider

Capsule collider

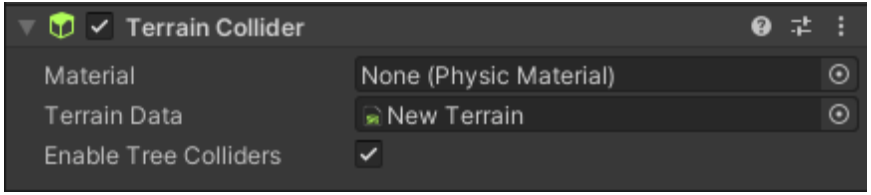
- The Capsule Collider is made of two half-spheres joined together by a cylinder.

Property	Function
Is Trigger	If enabled, this Collider is used for triggering events, and is ignored by the physics engine.
Material	Reference to the Physics Material that determines how this Collider interacts with others.
Center	The position of the Collider in the object's local space.
Radius	The radius of the Collider's local width.
Height	The total height of the Collider.
Direction	The axis of the capsule's lengthwise orientation in the object's local space.

Terrain Collider

Terrain collider

- The Terrain Collider implements a collision surface with the same shape as the Terrain object it is attached to.



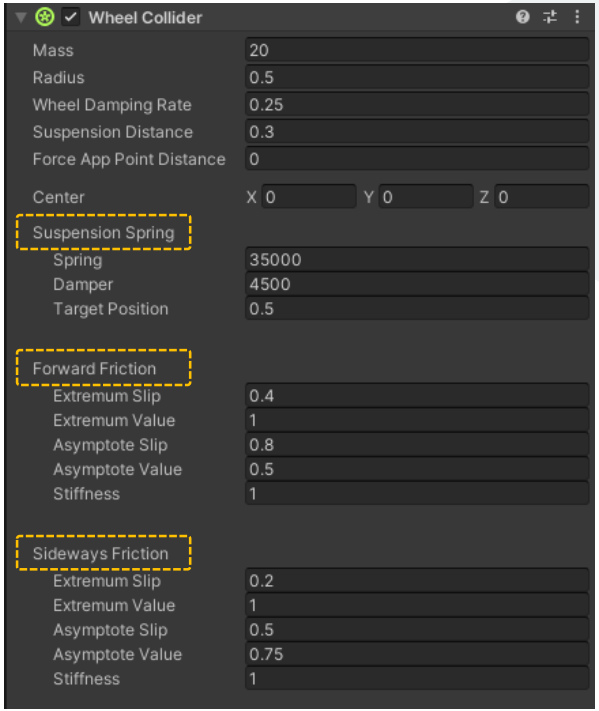
이미지 출처 :Unity

Property	Function
Material	Reference to the Physics Material that determines how this Collider interacts with others.
Terrain Data	The terrain data.
Enable Tree Colliders	When selected Tree Colliders will be enabled.

Wheel Collider

Wheel collider

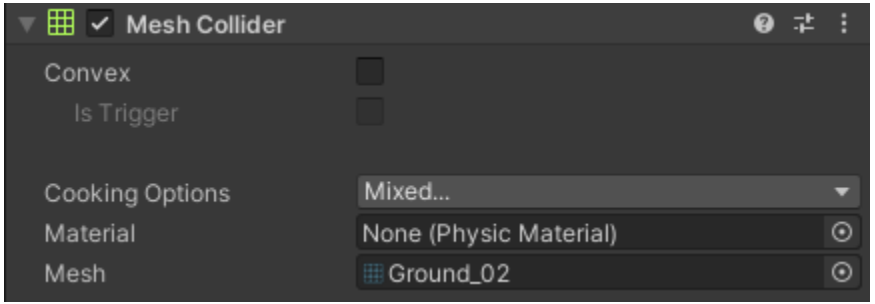
- The Wheel Collider s a special collider for grounded vehicles.
- It has built-in collision detection, wheel physics, and a slip-based tire friction model.
- It can be used for objects other than wheels, but it is specifically designed for vehicles with wheels.



이미지 출처 :Unity

Mesh Collider

Mesh collider



이미지 출처 :Unity

Mesh Collider

Mesh collider

- There are some cases, however, where even compound colliders are not accurate enough.
- In 3D, you can use Mesh Colliders to match the shape of the GameObject's mesh exactly. In 2D, the Polygon Collider 2D does not match the shape of the sprite graphic perfectly but you can refine the shape to any level of detail you like.
- The Mesh Collider builds its collision representation from the Mesh attached to the GameObject, and reads the properties of the attached Transform to set its position and scale correctly.
- The benefit of this is that you can make the shape of the Collider exactly the same as the shape of the visible Mesh for the GameObject, which creates more precise and authentic collisions.

Mesh Collider

Mesh Collider

Property		Function
Convex		Enable the checkbox to make the Mesh Collider collide with other Mesh Colliders. Convex Mesh Colliders are limited to 255 triangles.
Is Trigger		Enable this checkbox to make Unity use this Collider for triggering events, and the physics engine ignore it.
Cooking Options		Enable or disable the Mesh cooking options that affect how the physics engine processes Meshes.
	None	Disables all of the Cooking Options listed below.
	Everything	Enables all of the Cooking Options listed below.
	Cook for Faster Simulation	Makes the physics engine cook Meshes for faster simulation. When enabled, this runs some extra steps to guarantee the resulting Mesh is optimal for run-time performance. This affects the performance of the physics queries and contacts generation. When this setting is disabled, the physics engine uses a faster cooking time instead, and produces results as fast as possible. Consequently, the cooked Mesh Collider might not be optimal.

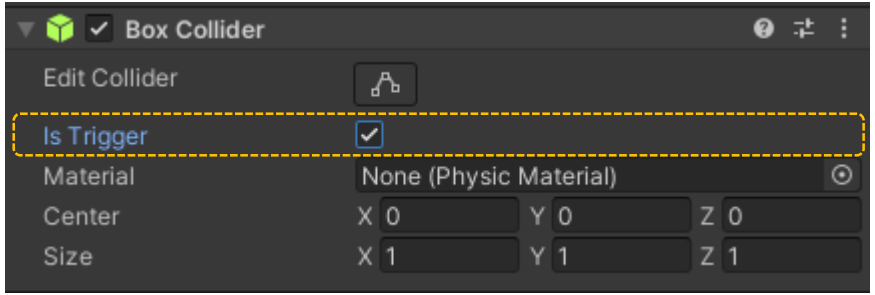
Mesh Collider

Property		Function
	Enable Mesh Cleaning	Makes the physics engine clean Meshes. When enabled, the cooking process tries to eliminate degenerate triangles of the Mesh, as well as other geometrical artifacts. This results in a Mesh that is better suited for use in collision detection and tends to produce more accurate hit points.
	Weld Colocated Vertices	Makes the physics engine remove equal vertices in the Meshes. When enabled, the physics engine combines the vertices that have the same position. This is important for the collision feedback that happens at run time.
	Use Fast Midphase	Makes the physics engine use the fastest mid-phase acceleration structure and algorithm available for your output platform. When you enable this option, the physics engine uses a faster algorithm that doesn't require any R-Trees for spatial access. If you encounter mid-phase issues at runtime on some platform, you can still disable this option to use the slower legacy mid-phase algorithm instead.
Material		Reference to the Physics Material that determines how this Collider interacts with others.
Mesh		Reference to the Mesh to use for collisions.

Trigger

Trigger

- Triggers are enabled with the “IsTrigger” checkbox selected.
- This functions the same as a Collider, but disables Physics on the component, enabling objects to pass through it via a zone.
- Events can be called when objects enter or exit the Trigger.



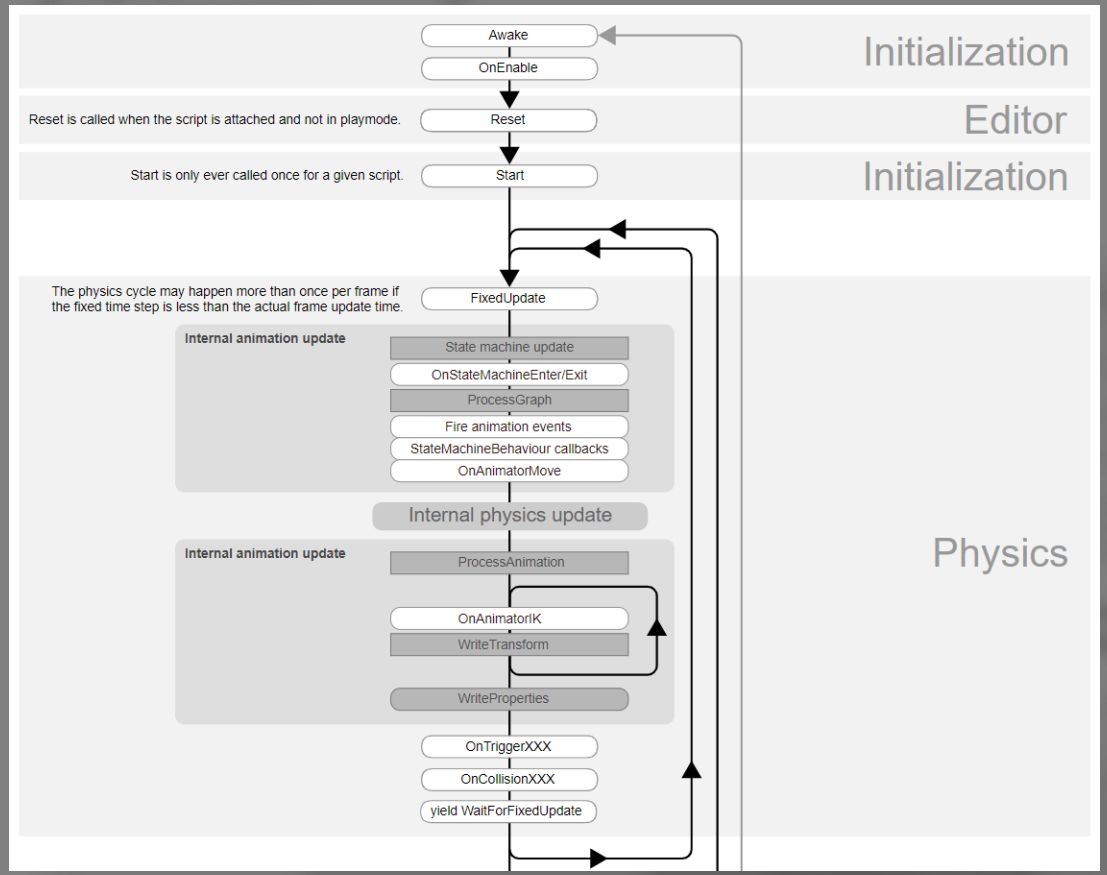
이미지 출처 :Unity

Trigger/Collision Event

- Trigger Event
  - OnTriggerEnter
  - OnTriggerStay
  - OnTriggerExit
- Collision Event
  - OnCollisionEnter
  - OnCollisionStay
  - OnCollisionExit

Trigger/Collision Event

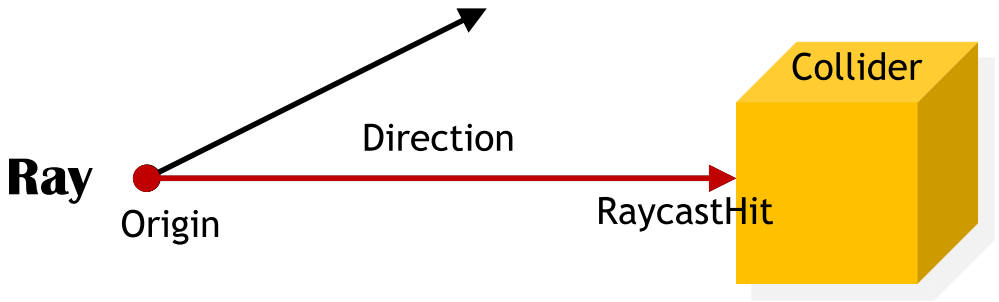
Trigger/Collision Event



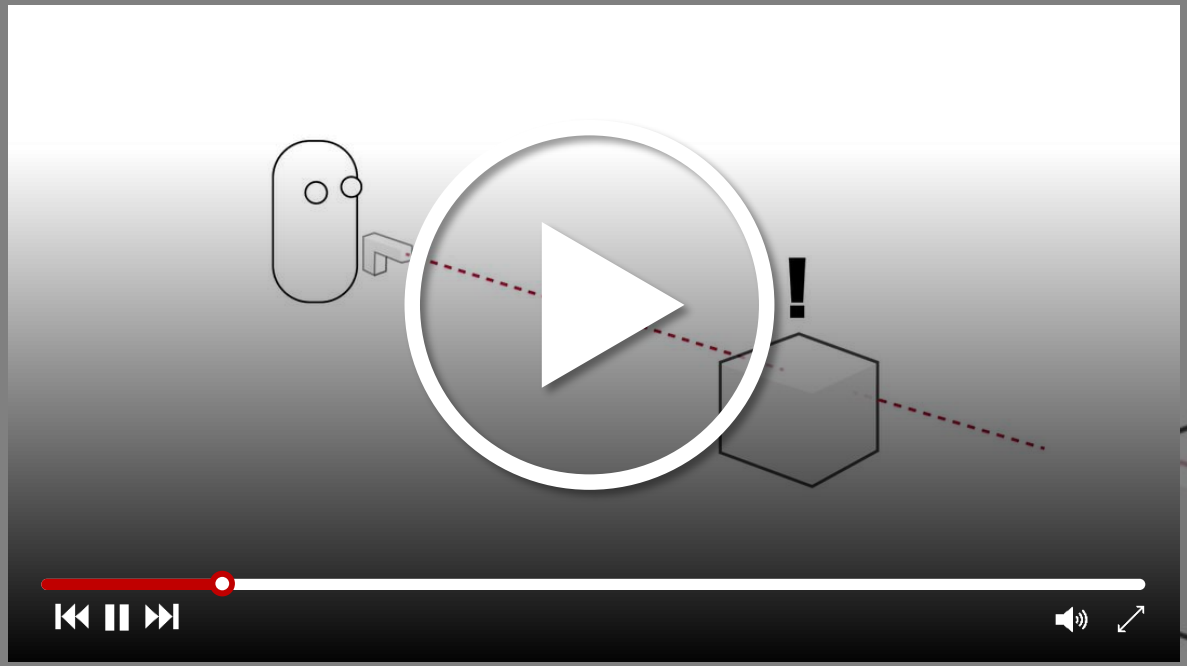
이미지 출처 :Unity C# Script Lifecycle Flowchart
https://docs.unity3d.com/2019.3/Documentation/Manual/ExecutionOrder.html

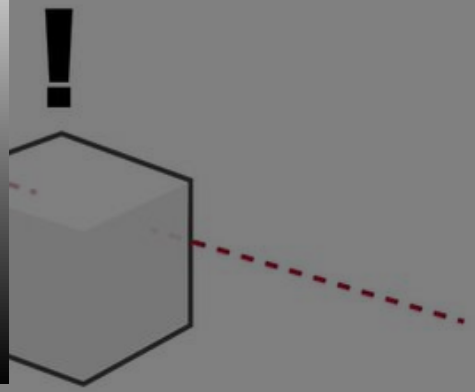
Raycasting

- Cast a ray, or invisible connection between one Physics object and another.
- The colliding GameObject can be assigned a new or existing tag so a RaycastHit will be easier to identify.
- Physics.Raycast(Vector3 Origin, Vector3 Direction)
- Physics.Raycast(Vector3 Origin, Vector3 Direction, RaycastHit Info, float Distance, int LayerMask)
- Physics.Raycast(Ray RayName, RaycastHit Info, float Distance, int LayerMask)



Raycasting

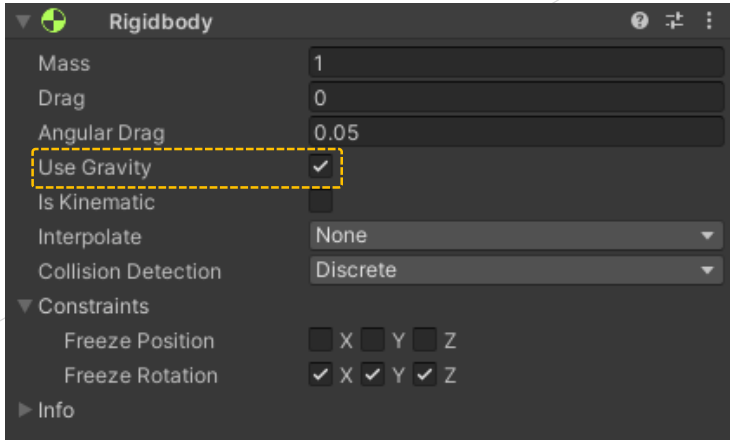




영상 출처 : https://www.youtube.com/watch?v=EINgIoTG8D4

Rigidbody

- A Rigidbody is the main component that enables physical behavior for a GameObject. With a Rigidbody attached, the object will immediately respond to gravity.
- If one or more Collider components are also added, the GameObject is moved by incoming collisions.



이미지 출처 :Unity

Rigidbody

- Objects of large mass (in kilograms by default) are less affected by objects with lower mass and vice versa.
- Drag affects the dampening of velocity over time. 0 means no air resistance.
- Angular Drag affects angular velocity.
- If UseGravity enabled, the object is affected by gravity.
- If IsKinematic enabled, the object will not be driven by the physics engine, and can only be manipulated by its Transform.
  - This is useful for moving platforms or if you want to animate a Rigidbody that has a HingeJoint attached.
  - IsKinematic also affects objects controlled by the Animation Engine. If IsKinematic is selected, the Animation Engine affects objects. If deselected, the Physics Engine retains control.

Rigidbody

Rigidbody

- Interpolate setting detects how collision are checked.
  - None
    - No interpolation is applied.
  - Interpolate
    - Smooth movements of objects are based on information from the previous frame in an animation's timeline.
  - Extrapolate
    - Smooth movements of objects are based on a guess of the next frame.

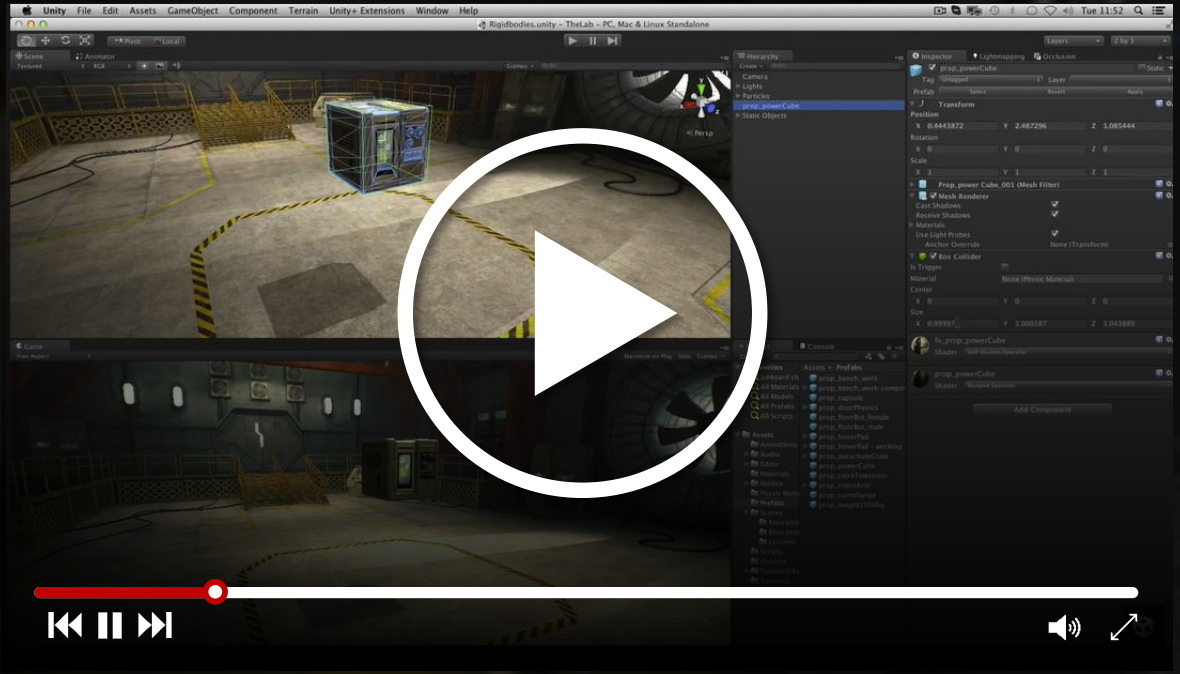
Rigidbody

Rigidbody

- Collision Detection sets the rate at which collisions are checked.
  - Discrete
    - Default. Use discrete collision detection against all other Colliders in the Scene.
  - Continuous
    - Fast objects that interact with static objects
  - Continuous Dynamic
    - Fast objects that interact with other fast objects
  - Continuous Speculative
    - Predictive collision checking
- Constaints for X, Y, Z axes defines which axis should not move.

Rigidbody

Rigidbody



영상 출처 : https://www.youtube.com/watch?v=WTGcs10Sj34

Rigidbody Class

Properties

Properties	Function
angularVelocity	The angular velocity vector of the rigidbody measured in radians per second.
position	The position of the rigidbody.
rotation	The rotation of the Rigidbody.
velocity	The velocity vector of the rigidbody. It represents the rate of change of Rigidbody position.
isKinematic	Controls whether physics affects the rigidbody.
useGravity	Controls whether gravity affects this rigidbody.

Rigidbody Class

Methods

Methods	Function
AddExplosion Force	Applies a force to a Rigidbody that simulates explosion effects.
AddForce	Adds a force to the Rigidbody.
AddTorque	Adds a torque to the Rigidbody.
GetPointVelocity	The velocity of the Rigidbody at the point worldPoint in global space.
MovePosition	Moves the kinematic Rigidbody towards position.
MoveRotation	Rotates the Rigidbody to rotation.

Rigidbody Class

Update vs FixedUpdate

Update

Update called once per frame.

FixedUpdate

FixedUpdate called multiple times per frame. Most physics calculations will be called in FixedUpdate. The time between calculations is fixed, as the name implies.

Set GameObject Physics Component

- Use only Collider on GameObject
  - Used for static GameObjects (e.g., wall, structure, etc)
- Use only IsTrigger-enabled Collider on GameObject
  - Used as a trigger to detect if a specific object has entered the area
- Use Rigidbody and Collider on GameObject
  - GameObject is driven solely by the physics engine.
  - Be careful not to move using Transform in the script
- Use IsKinematic-enabled Rigidbody and Collider on GameObject
  - Suitable for moving only script without being affected by physics engine.

Set GameObject Physics Component

- Use Rigidbody and IsTrigger-enabled Collider on GameObject
  - The trigger area is moved by the physics engine
  - Be careful not to move using Transform in the script
- Use IsKinematic-enabled Rigidbody and IsTrigger-enabled Collider on GameObject
  - When moving the trigger area to a script

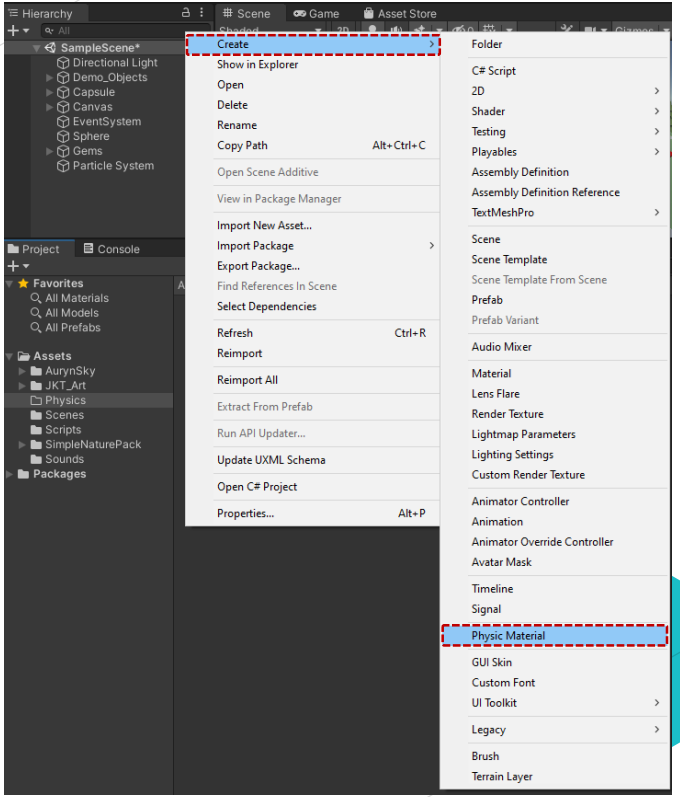
Force Modes

- Acceleration
  - Applies a force that increases at a constant rate.
- Force
  - Default, gradually applies a force accounting for its mass.
- Impulse
  - Applies an instant force instead of one that gradually builds up over time.
- VelocityChange
  - Applies instant forces in different directions.
  - Disregards mass.

Force Modes

Physic Material

- The Physic Material adjusts friction and bouncing effects of colliding GameObjects.
- Create a custom Physic Material (e.g. rubber) and then drag it to the GameObject's Material property in the inspector.



이미지 출처 :Unity

Physic Material

Physic Material Inspector

- Dynamic Friction
- Static Friction
- Bounciness
- Friction Combine
- Bounce Combine

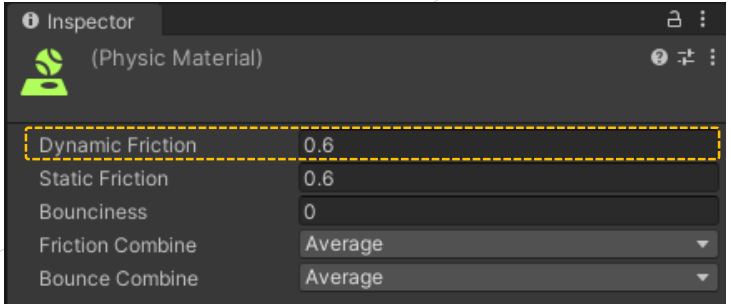
이미지 출처 :Unity

Physic Material

Physic Material Inspector

Dynamic Friction (0-1)

- How much friction applied to the object when in motion. The higher the friction the more outside force (like gravity or an explosion) impacts it;
  0 is ice, 1 is super-glue sticky.



이미지 출처 :Unity

Physic Material

Physic Material Inspector

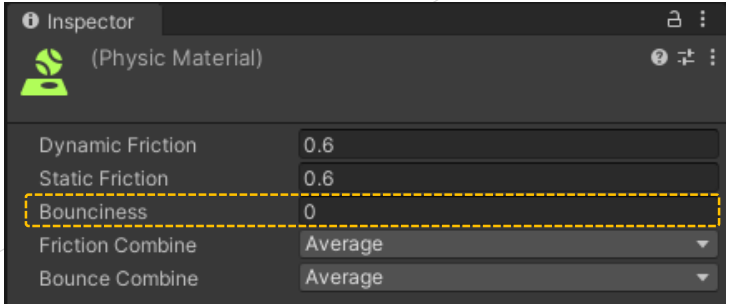
- Static Friction (0-1)
  - How much force is needed to get the object moving in the first place basically; 0 means anything gets it going, 1 means it require a heavy amount of push

이미지 출처 :Unity

Physic Material

Physic Material Inspector

- Bounciness (0-1)
  - How bouncy the surface is when something collides with it (or it collides with something);
    0 is your surface made of mud, 1 it is made of rubber.



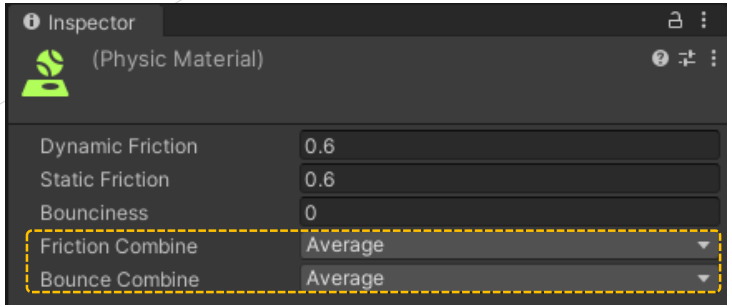
이미지 출처 :Unity

Physic Material

Physic Material Inspector

Friction / Bounce Combine

- This tells Unity which physics material takes priority when making the calculation. Defaults to “average” where it tries to work out a middle ground, but sometimes it is useful to use minimum (where the lowest value of the two objects colliding is used) or maximum (where the highest value is used),
  e.g. when a rubber ball hits a pile of mud, you don’t want to bouncing away, so use “Minimum”.



이미지 출처 :Unity

Physic Material

Physic Material

Rubber Material

- Dynamic : 0.8
- Static : 0.9
- Bounciness : 0.8
- Friction Combine : Maximum
- Bounce Combine : Average

Ice Material

- Dynamic : 0.05
- Static : 0.1
- Bounciness : 0.05
- Friction Combine : Multiply
- Bounce Combine : Multiply

자료 출처 : https://medium.com/sun-dog-studios/rapid-unity-tutorials-1-physics-materials-68758351fd8a

Physic Material

Physic Material

Physic Material

Wood Material

- Dynamic : 0.475
- Static : 0.475
- Bounciness : 0
- Friction Combine : Average
- Bounce Combine : Average

Metal Material

- Dynamic : 0.15
- Static : 0.2
- Bounciness : 0
- Friction Combine : Minimum
- Bounce Combine : Average

자료 출처 : https://medium.com/sun-dog-studios/rapid-unity-tutorials-1-physics-materials-68758351fd8a

Physic Material

Physic Material

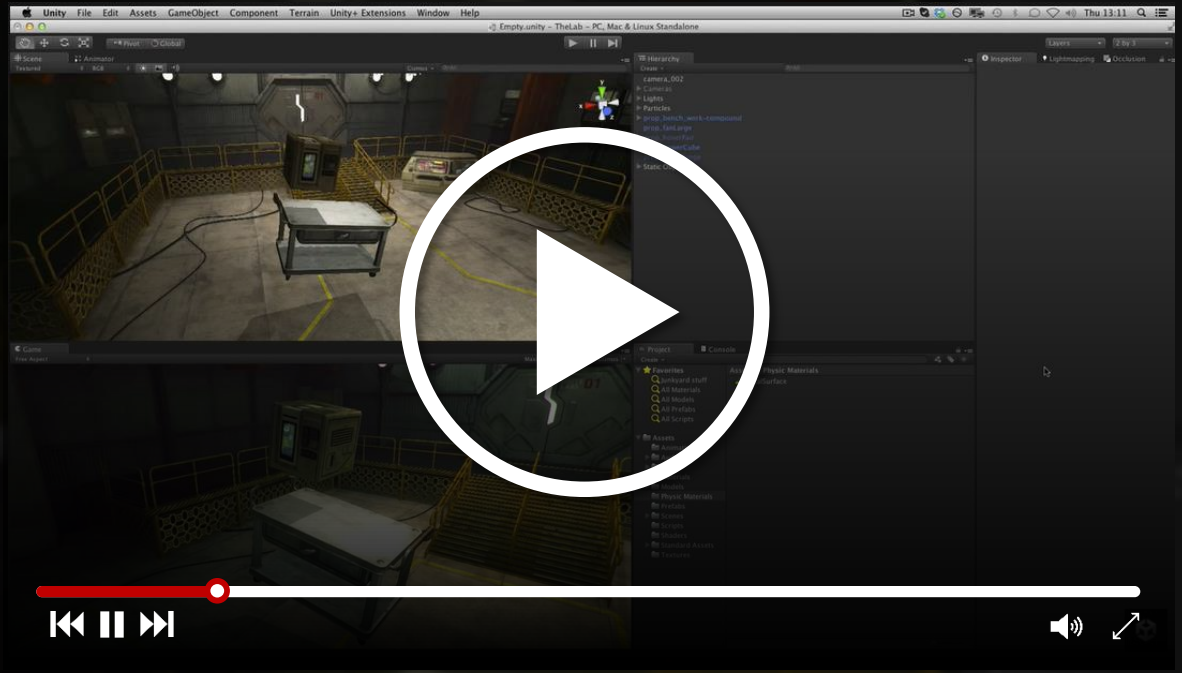
Mud Material

- Dynamic : 1
- Static : 0.9
- Bounciness : 0
- Friction Combine : Minimum
- Bounce Combine : Minimum

자료 출처 : https://medium.com/sun-dog-studios/rapid-unity-tutorials-1-physics-materials-68758351fd8a

Physic Material

Physic Material



영상 출처 : https://www.youtube.com/watch?v=SuUNnwswH94

Character Control

Character Control

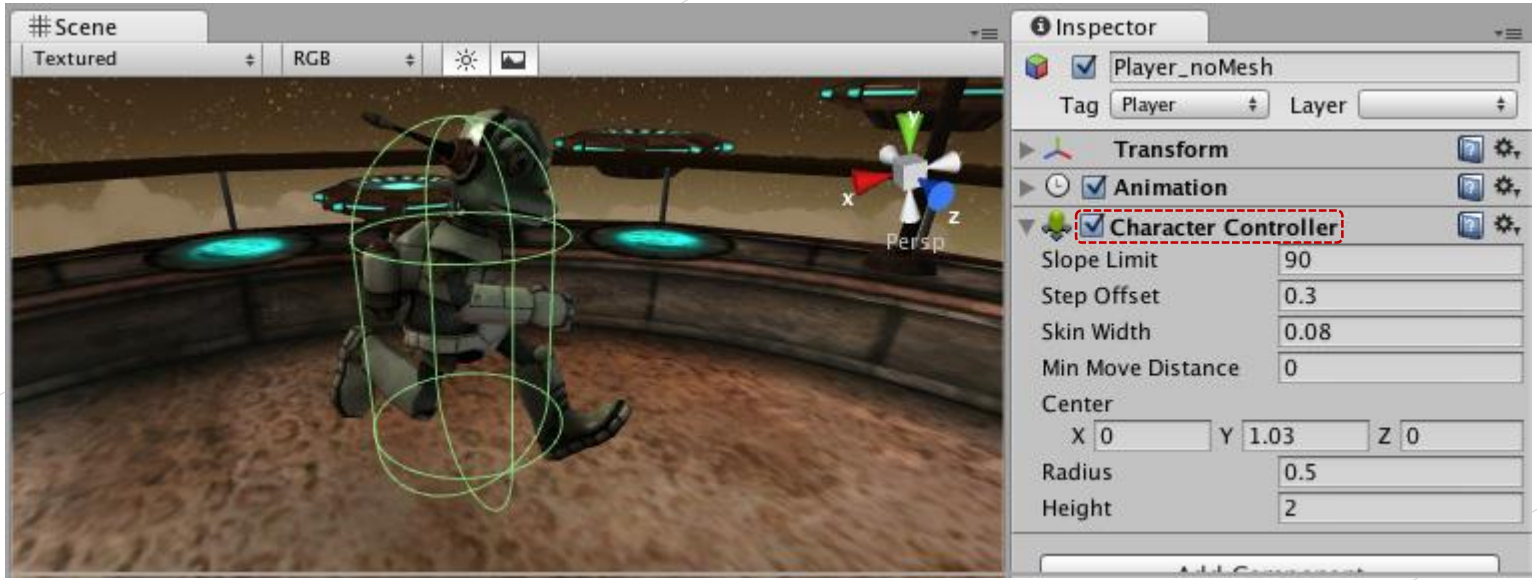
- In a first-person or third-person application, a user's character or avatar usually needs some collision-based physics, so that it doesn't fall through the floor or walk through walls.
- In 3D physics, you can create and configure character physics and control with a Character Controller.
- You can configure character control via the CharacterController class, or the corresponding Character Controller component.
- This component gives the character a simple, capsule-shaped collider that is always upright. The controller has its own special functions to set the object's speed and direction but unlike true colliders, a rigidbody is not needed and the momentum effects are not realistic.

Character Control

Character Controller

Character Controller Component

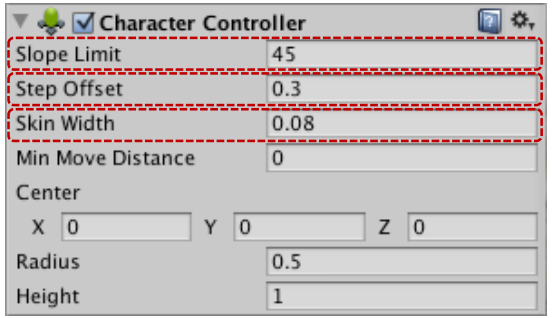
- The Character Controller is mainly used for third-person or first-person player control that does not make use of Rigidbody physics.



이미지 출처 :https://docs.unity3d.com/Manual/class-CharacterController.html

Character Controller

Character Controller

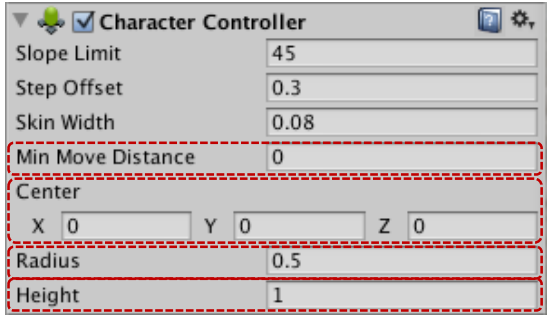


이미지 출처 :Unity

Slope Limit	Limits the collider to only climb slopes that are less steep (in degrees) than the indicated value.
Step Offset	The character will step up a stair only if it is closer to the ground than the indicated value. This should not be greater than the Character Controller's height or it will generate an error.
Skin width	Two colliders can penetrate each other as deep as their Skin Width. Larger Skin Widths reduce jitter. Low Skin Width can cause the character to get stuck. A good setting is to make this value 10% of the Radius.

Character Controller

Character Controller



이미지 출처 :Unity

Min Move Distance	If the character tries to move below the indicated value, it will not move at all. This can be used to reduce jitter. In most situations this value should be left at 0.
Center	This will offset the Capsule Collider in world space, and won't affect how the Character pivots.
Radius	Length of the Capsule Collider's radius. This is essentially the width of the collider.
Height	The Character's Capsule Collider height. Changing this will scale the collider along the Y axis in both positive and negative directions.

Physics Joints

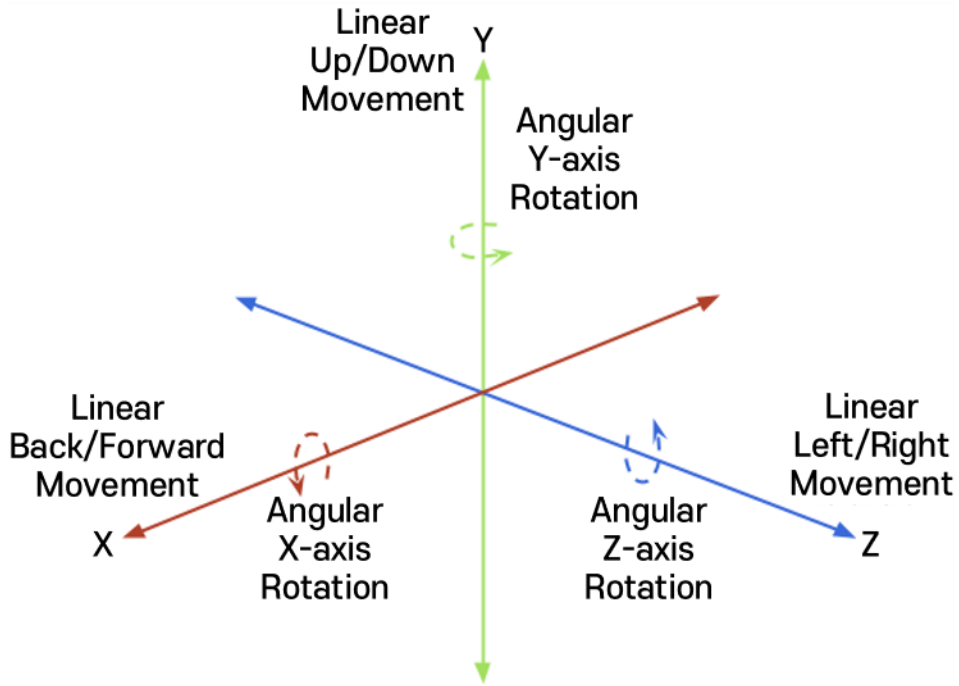
- Joint
  - A Joint connects a Rigidbody to another Rigidbody, or to a fixed point in space.
  - Joints can apply forces that move rigid bodies, and joint limits can restrict that movement.

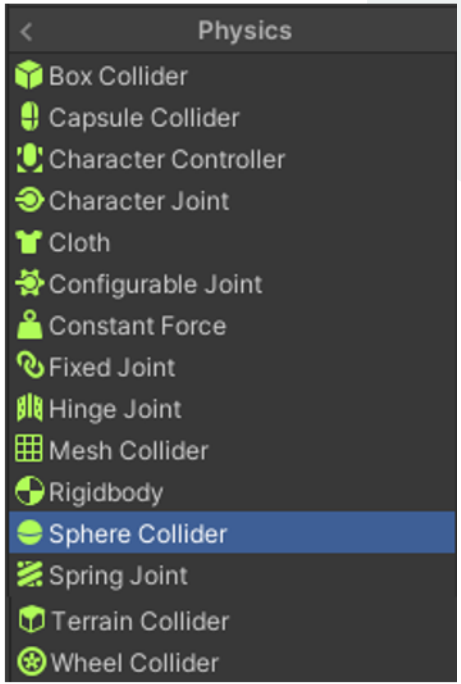
Physics Joints

Physics Joints

Joint

- Joints give Rigidbodies the following degrees of freedom

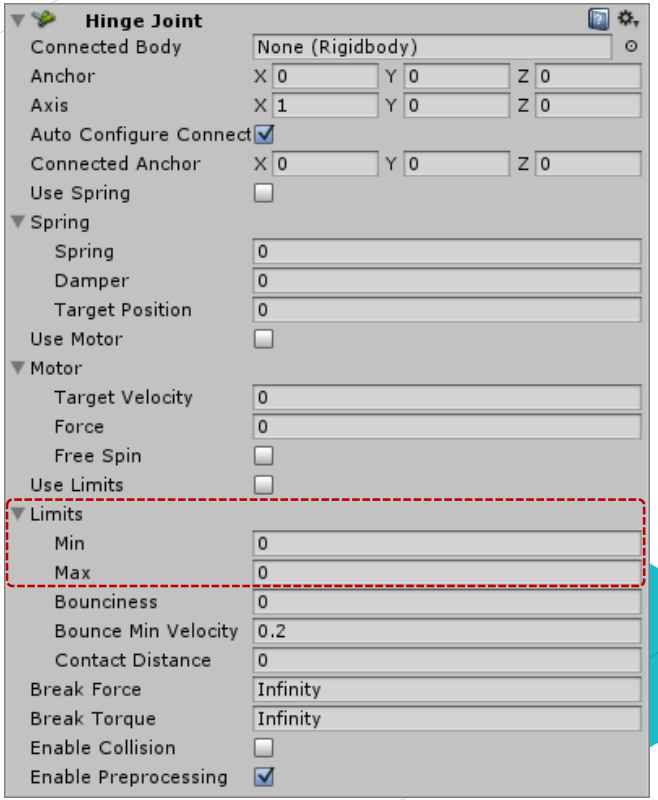




이미지 출처 :https://docs.unity3d.com/Manual/Joints.html

Hinge Joint

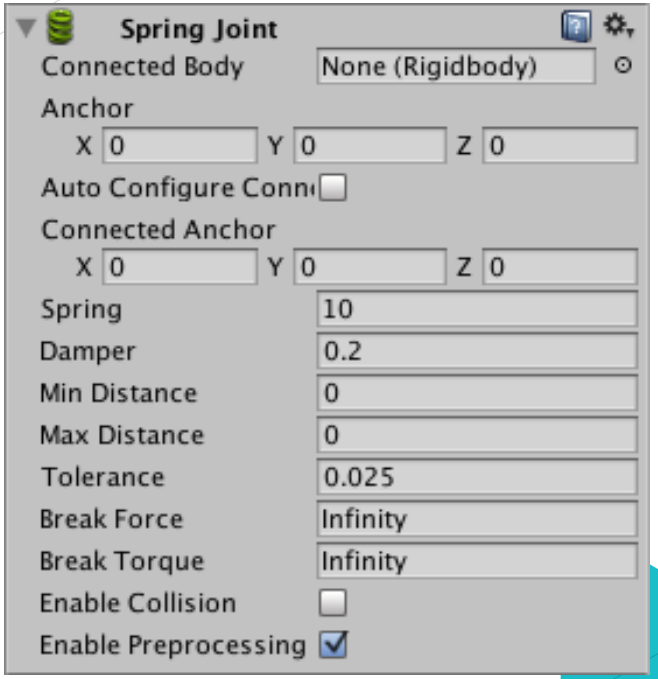
- Attaches a rigid body to another rigid body or a point in space at a shared origin and allows the rigid bodies to rotate around a specific axis from that origin.
- Hinge Joint groups together two Rigidbodies, constraining them to move like they are connected by a hinge.
- It is perfect for doors, and can also be used to model chains, pendulums.



이미지 출처 :Unity

Spring Joint

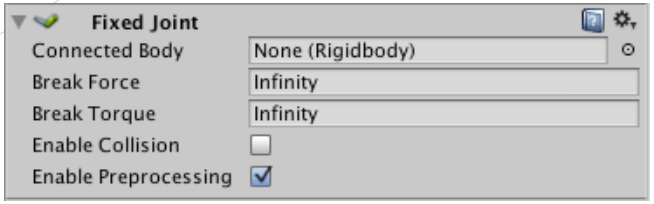
- Keeps rigid bodies apart from each other but lets the distance between them stretch slightly. The spring acts like a piece of elastic that tries to pull the two anchor points together to the exact same position.
- Spring Joint joins two Rigidbodies together but allows the distance between them to change as though they were connected by a spring.



이미지 출처 :Unity

Fixed Joints

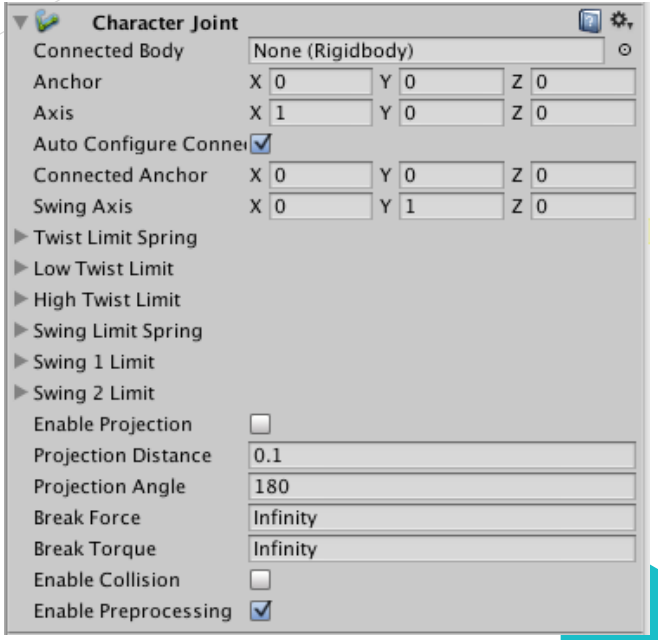
- Restricts the movement of a rigid body to follow the movement of the rigid body it is attached to. This is useful when you need rigid bodies that easily break apart from each other, or you want to connect the movement of two rigid bodies without parenting in a Transform hierarchy.



이미지 출처 :Unity

Character Joints

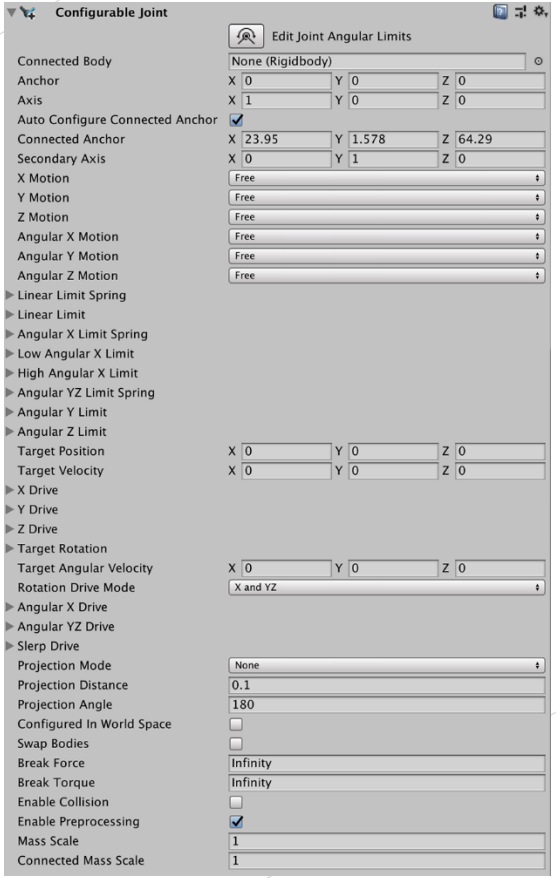
- Character Joints are mainly used for Ragdoll effects. They are an extended ball-socket joint which allows you to limit the joint on each axis.



이미지 출처 :Unity

Configurable Joints

- Configurable Joints incorporate all the functionality of the other joint types and provide greater control of character movement.
- Configurable Joints are particularly useful when you want to customize the movement of a ragdoll and enforce certain poses on your characters. You can also use them to adapt joints into highly specialized joints of your own design.



이미지 출처 :Unity

Joints

영상 출처 : https://www.youtube.com/watch?v=f4xikqJdkwM

Articulations

- A physics articulation is a set of Articulation Bodies organized in a logical tree, in which each parent-child relationship reflects mutually constrained relative motion.
- The main purpose of physics articulations is to provide a realistic physics behavior for industrial and commercial non-gaming applications that involve joints.
- For example, they make it a lot easier than the regular Joints to simulate robotic arms and kinematic chains.

Articulations

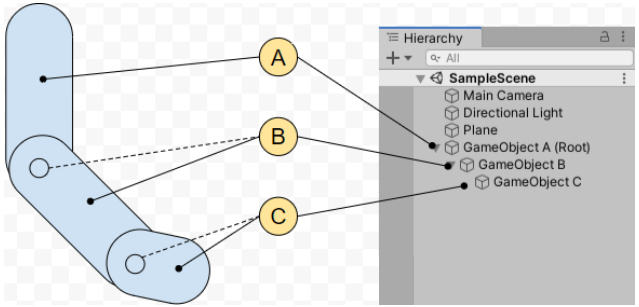
Building an Articulation in Unity

- To build a physics articulation in Unity, you must add an Articulation Body component to each GameObject that the articulation consists of.
- Each Articulation Body component allows you to configure in one single place
  - The physical body properties of the corresponding GameObject. Basically, its mass and the way it responds to the physics environment.
  - The type and properties of the joint that links the GameObject to its parent GameObject (except for the root of the articulation).

Building an Articulation

Building an Articulation in Unity

- The example of a simple physics articulation that involves 3 physical bodies and 2 joints



to. 교수님
출처 확인
부탁드립니다.

이미지 출처 :---

GameObject	Articulation Body component configuration
A (root)	You can only define physical body properties for GameObject A.
B	You can define • Physical body properties for GameObject B. • The type and properties of the joint with GameObject A.
C	You can define • Physical body properties for GameObject C. • The type and properties of the joint with GameObject B.

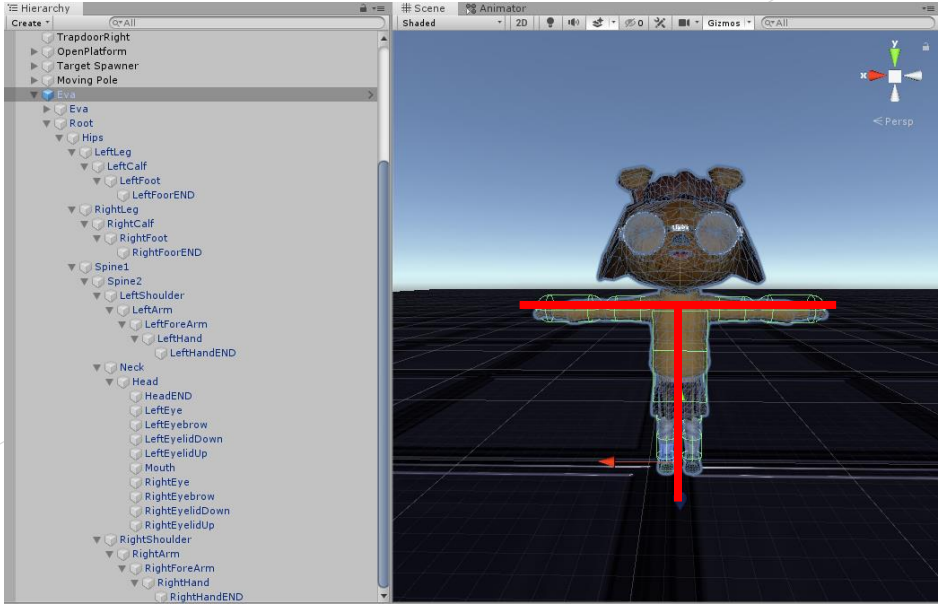
Articulation Joint types

- Four types of articulation joints
  - Fixed joint
    - sets a rigid, unbreakable and unstretchable link between bodies.
  - Prismatic joint
    - prevents all motion except sliding along a particular axis.
  - Revolute joint
    - allows rotation around a particular axis (like a hinge).
  - Spherical joint
    - anatomical joint, which allows two swings and one twist.

Articulation Joint types

Ragdoll Physics

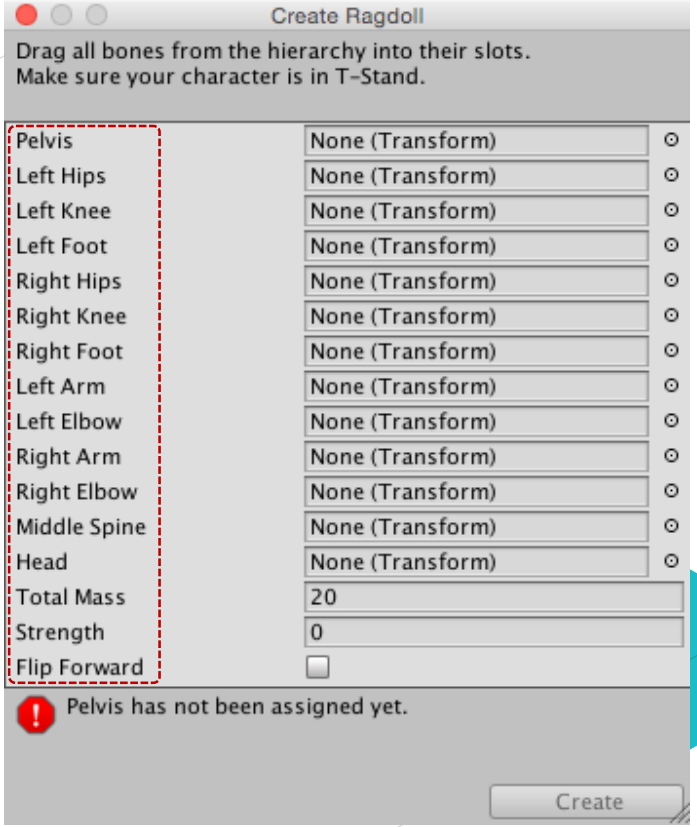
- Ragdoll physics are a set of colliders, rigid bodies and joints that you can apply to a humanoid character, to simulate behavior such as impact collisions and character death.



이미지 출처 :https://learn.unity.com/tutorial/creating-ragdolls-2019

Ragdoll Physics

- Create a ragdoll
  - Unity has a simple wizard that lets you quickly create your own ragdoll.
  - You simply have to drag the different limbs on the respective properties in the wizard.
  - Then select create and Unity will automatically generate all Colliders, Rigidbodies and Joints that make up the Ragdoll for you.
  - Ragdolls make use of Skinned Meshes, that is a character mesh rigged up with bones in the 3D modeling application (such as Maya).

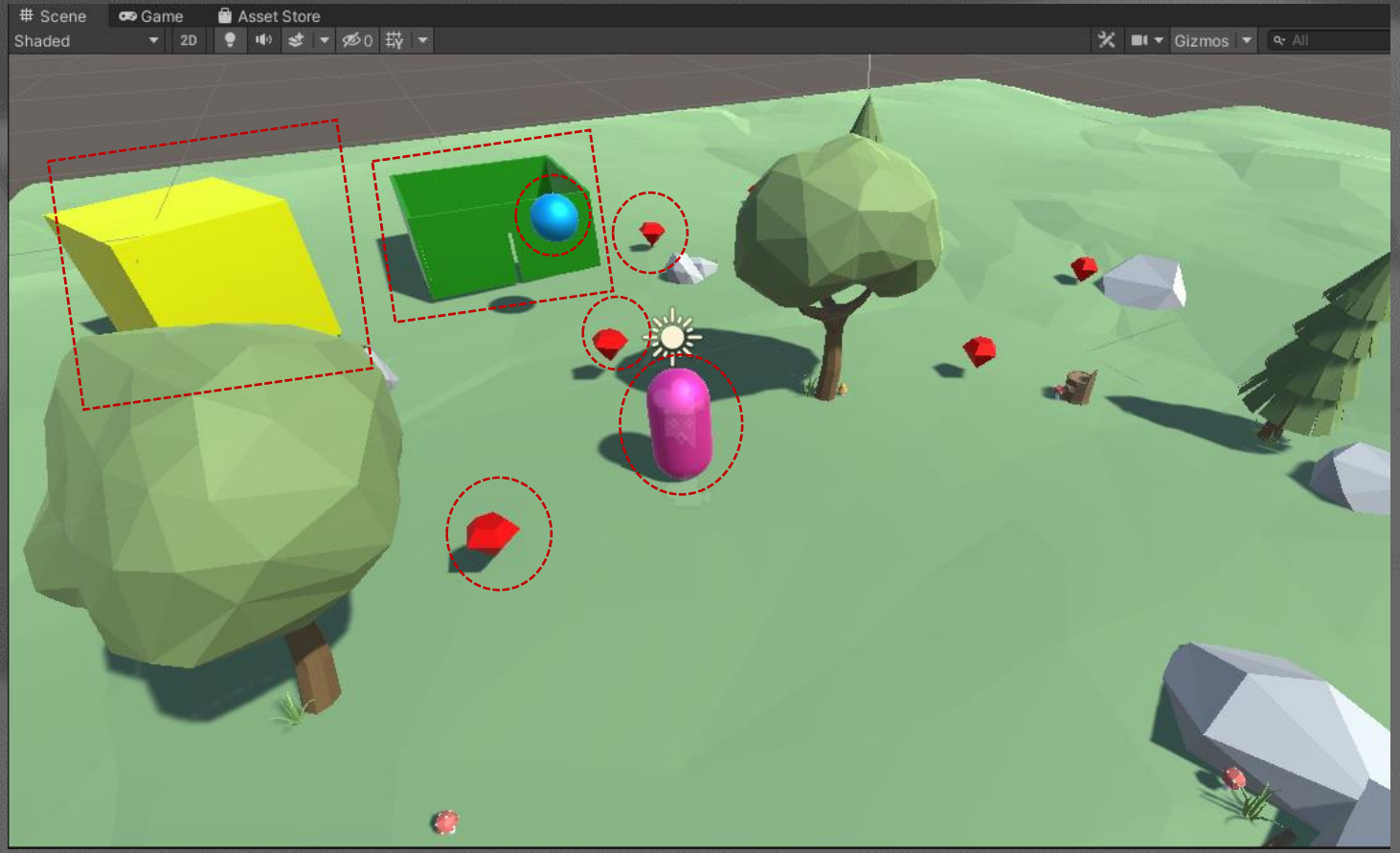


이미지 출처 :https://docs.unity3d.com/Manual/wizard-RagdollWizard.html

2

Unity Physics Example

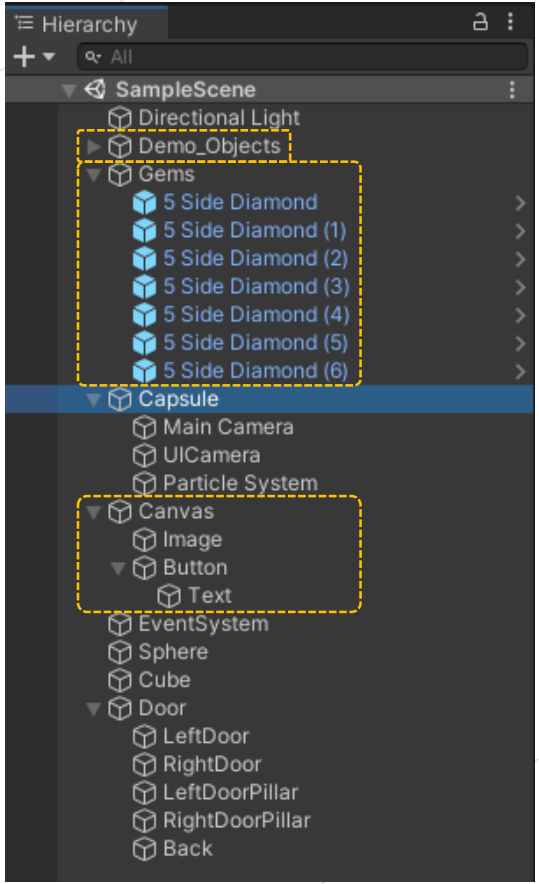
Physics Examples



이미지 출처 :Unity

Physics Examples

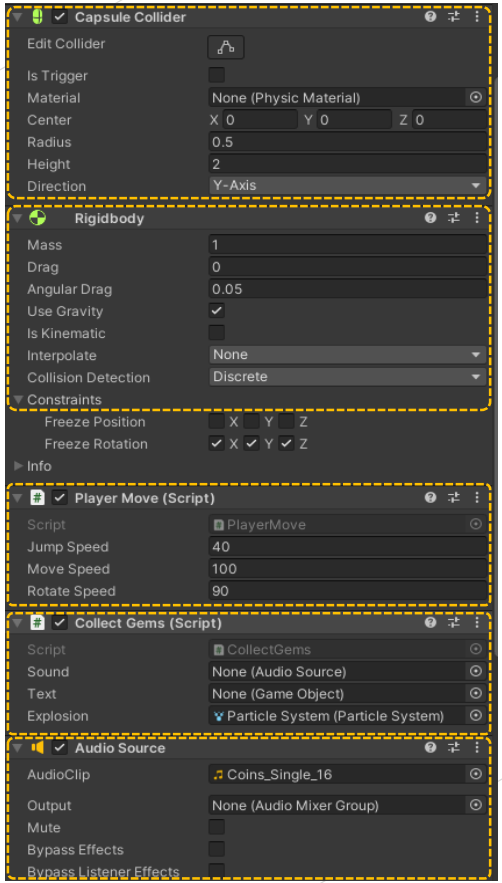
- Create SimpleNaturePack Demo_Objects
  - Mesh Collider
- Create 7 Gems using SimpleGemsUltimateAnimatedCustomizable
  - Box Collider with IsTrigger enabled
  - Tag named Gem
- Create UI with Image and Button
  - Image with go_2 image
  - Button image with box_11 image & text with Gems: 0



이미지 출처 :Unity

Physics Examples

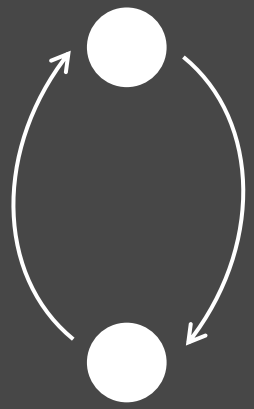
- Create Capsule (myself)
  - MainCamera(AudioListener) & UICamera (no AudioListener) & ParticleSystem children
  - AudioSource (Coins_Single_16 sound)
  - Capsule Collider & Rigidbody with UseGravity enabled
  - C# Scripts (PlayerMove & CollectGems)



이미지 출처 :Unity

Physics Examples

```
[RequireComponent(typeof(Rigidbody))]
Unity Script (1 asset reference) | 0 references
public class PlayerMove : MonoBehaviour
{
    public float jumpSpeed = 100f;
    public float moveSpeed = 100f;
    public float rotateSpeed = 100f;
    private Vector3 rotation;
    private Rigidbody rb;

    Unity Message | 0 references
    private void Start()
    {
        rotation = transform.eulerAngles;
        rb = GetComponent<Rigidbody>();
    }

    Unity Message | 0 references
    void FixedUpdate()
    {
        if (Input.GetKey(KeyCode.W))
        {
            rb.velocity = transform.forward * moveSpeed * Time.deltaTime;
        }
        if (Input.GetKey(KeyCode.S))
        {
            rb.velocity = -transform.forward * moveSpeed * Time.deltaTime;
        }
        if (Input.GetKey(KeyCode.A))
        {
            rotation += new Vector3(0, -rotateSpeed, 0) * Time.deltaTime;
            rb.MoveRotation(Quaternion.Euler(rotation));
        }
        if (Input.GetKey(KeyCode.D))
        {
            rotation += new Vector3(0, rotateSpeed, 0) * Time.deltaTime;
            rb.MoveRotation(Quaternion.Euler(rotation));
        }
        if (Input.GetKey(KeyCode.Space))
        {
            rb.AddForce(Vector3.up * jumpSpeed);
        }
```



이미지 출처 :Unity

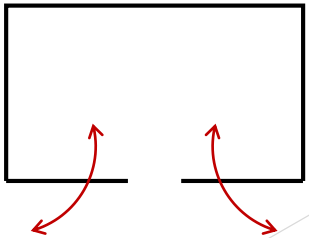
Physics Examples

```
public class CollectGems : MonoBehaviour
{
    3 references
    public int numberOfGems { get; private set; }
    public AudioSource sound;
    public GameObject text;
    public ParticleSystem explosion;

    Unity Message | 0 references
    void Update()
    {
        if (numberOfGems >= 7)
        {
            print("Game Success!");
        }
    }

    Unity Message | 0 references
    void OnTriggerEnter(Collider other)
    {
        if (sound) sound.Play();
        if (explosion)
        {
            explosion.transform.position = transform.position;
            explosion.transform.rotation = transform.rotation;
            explosion.Play();
        }
        if (other.tag == "Gem")
        {
            CollectGem();
            if (text) text.GetComponent<Text>().text = "Gems: " + numberOfGems;
            other.gameObject.SetActive(false);
        }
    }

    1 reference
    public void CollectGem()
    {
        numberOfGems++;
    }
}
```

이미지 출처 :Unity

Physics Examples

- Create Sphere
  - Sphere Collider & Rigidbody with UseGravity enabled
  - C# Scripts (BallMove)
- Create Cube
  - Box Collider with IsTrigger enabled
  - C# Scripts (FadeOut)
- Create Door
  - LeftDoor & RightDoor with BoxCollider & HingeJoint
  - LeftDoorPillar & RightDoorPillar & Back



Physics Examples

```
public class BallMove : MonoBehaviour
{
    public float speed = 5.0f;
    private Rigidbody rb;

    Unity Message | 0 references
    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }

    Unity Message | 0 references
    void FixedUpdate()
    {
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");
        Vector3 movement = new Vector3(moveHorizontal, 0, moveVertical);
        rb.AddForce(movement * speed);
    }
}
```

이미지 출처 :Unity

Physics Examples

```
public class FadeOut : MonoBehaviour {
    public float fadeOutSpeed = 0.01f;

    Unity Message | 0 references
    void OnTriggerEnter(Collider other)
    {
        StartCoroutine("RunFadeOut");
        other.gameObject.SetActive(false);
    }

    0 references
    IEnumerator RunFadeOut()
    {
        Renderer renderer = gameObject.GetComponent<Renderer>();
        Color color = renderer.material.color;
        while (color.a > 0.0f)
        {
            color.a -= fadeOutSpeed;
            renderer.material.color = color;
            if (color.a <= 0.0f) color.a = 0.0f;
            Debug.Log("RunFadeOut fadeOutMaterialColor " + color);
            yield return new WaitForSeconds(fadeOutSpeed); // wait for
        }
    }
}
```

이미지 출처 :Unity

Physics Examples

이미지 출처 :Unity

Reference

- https://learn.unity.com/tutorial/intro-to-the-unity-physics-engine-2019-3?uv=2019.4
- https://docs.unity3d.com/kr/530/Manual/class-PhysicMaterial.html
- https://docs.unity3d.com/Manual/Joints.html
- https://medium.com/sun-dog-studios/rapid-unity-tutorials-1-physics-materials-68758351fd8a
- https://learn.unity.com/tutorial/creating-ragdolls-2019
- https://www.youtube.com/watch?v=f4xikqJdkwM
- https://www.youtube.com/watch?v=SuUNnwswH94
- https://www.youtube.com/watch?v=WTGcs10Sj34
- https://www.youtube.com/watch?v=EINgIoTG8D4