UNIT1

INTRODUCTION TO GAME ENGINES

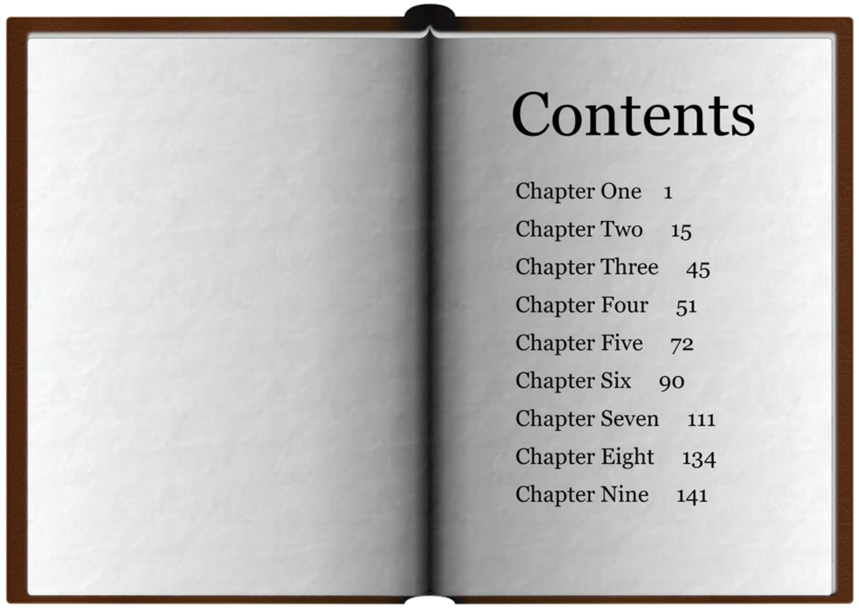
PMDM - 2DAM

Àngel Olmos (a.olmosginer@edu.gva.es)

Jose Pascual Rocher (jp.rochercamps@edu.gva.es)

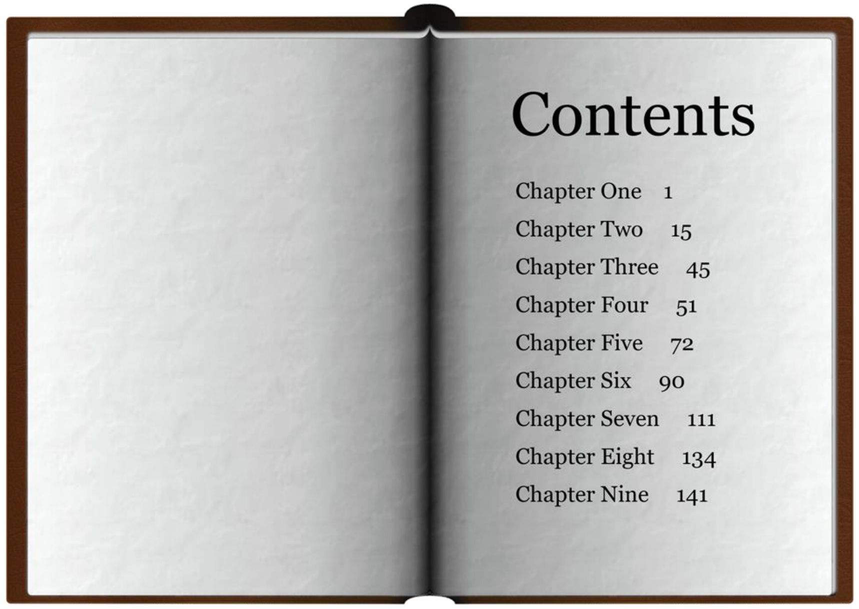
Content

1. What is a game engine?
2. Unity environment installation
3. Explore the Unity Editor
4. Working with 3D GameObjects
5. Create using primitives
6. Add components
7. Add physical properties
8. Manage GameObjects with prefabs
9. Get 3D assets
10. Publish your project
11. ACTIVITY



Content

1. What is a game engine?
2. Unity environment installation
3. Explore the Unity Editor
4. Working with 3D GameObjects
5. Create using primitives
6. Add components
7. Add physical properties
8. Manage GameObjects with prefabs
9. Get 3D assets
10. Publish your project
11. ACTIVITY

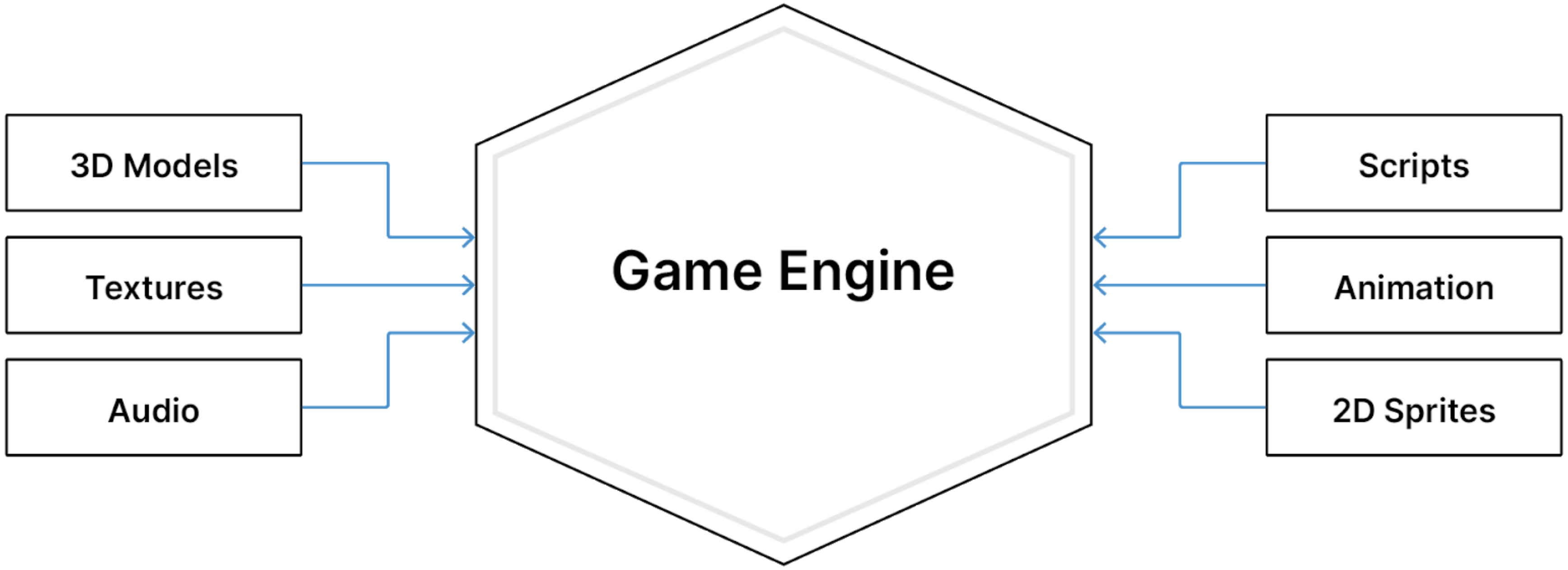


What is a game engine?

- A **game engine** is the point of convergence for all aspects of creating a game
- Games are made of smaller pieces like **3D models, scripts, and audio files**
- When put together, they create the full user experience
- If 3D models, scripts, and audio files were ingredients, a game engine would be the stockpot you dropped them into!

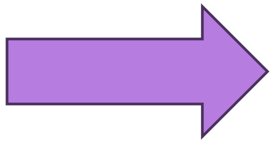
What is a game engine?

- A game engine is the **point of convergence for all components** that go into making a game



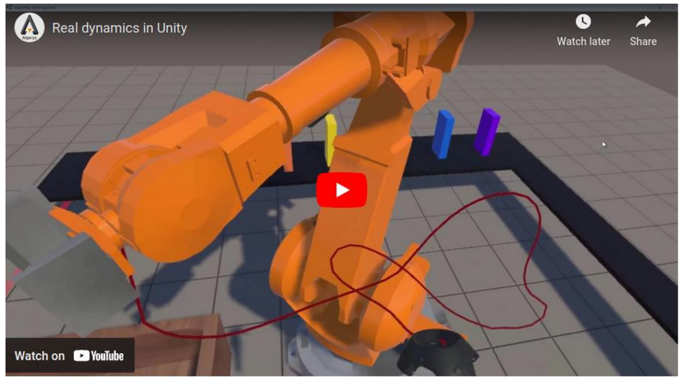
What is a game engine?

- Game engines make sure that your game will display on the screen, objects will be able to interact with other objects, sounds will be audible ...
- And that your **application will be publishable !!**
- You provide the content, and the game engine provides the tools to implement it in an environment that will just work



What is a game engine?

- Game engines can be used **not only to create games**
- They are also used to create interactive simulations and other experiences
- Game engines has been adopted by industries such as **film, automotive, architecture, engineering, construction …**

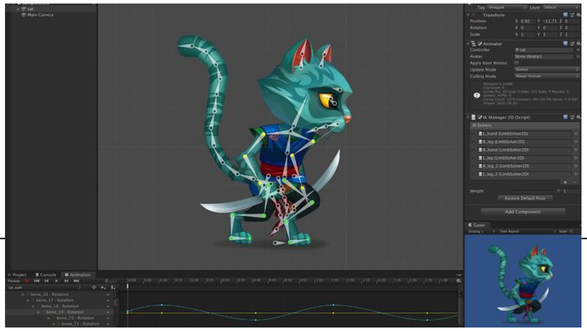


https://www.codinblack.com/what-can-you-do-with-unity-game-engine/

What is a game engine?

What do you do in a game engine?

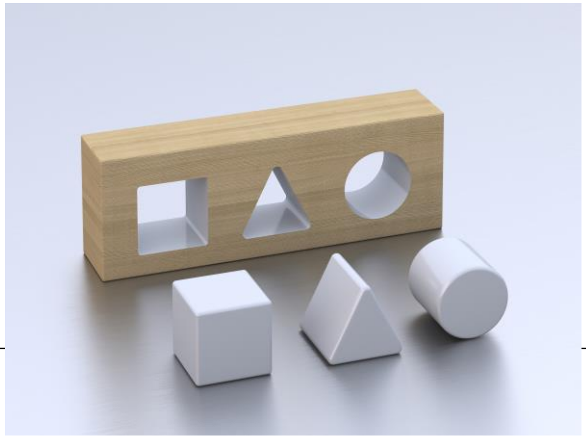
- Putting together everything that the user will experience in the final product
- If that product is a game, the creator designs **gameplay** such as jumping on platforms
- If it is an animation, the creator spawns the **action being recorded**
- If it is a VR architectural visualization, the creator builds a **photorealistic environment** that the user will walk through ...

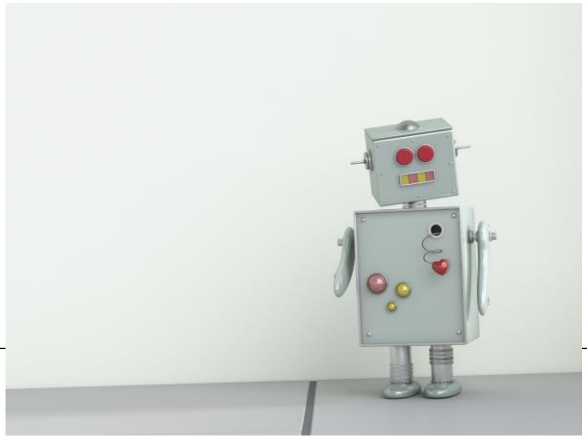


What is a game engine?

What **don't** you do in a game engine?

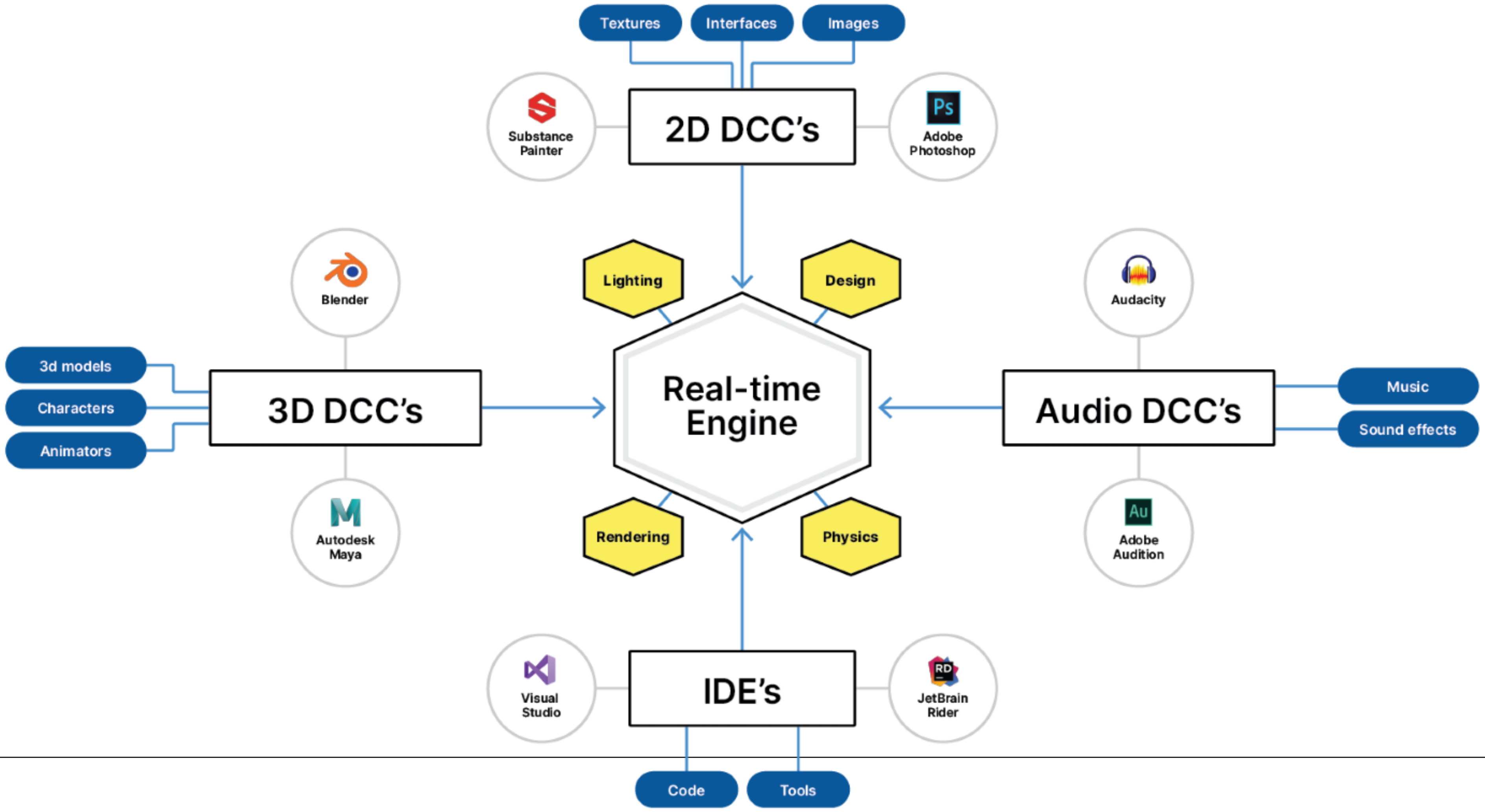
- Inside a game engine, **you don’t create assets** — the objects and sounds that are the building blocks of the project
- Instead, assets are created in specialized external programs called Digital Content Creation (DCC) tools
- Many **DCCs are integrated with game engines** to ease the process of importing them





What is a game engine?

What don't you do in a game engine?



What is a game engine?

What **don't** you do in a game engine?

Common types of DCC tools used in real-time development include:

- 3D DCCs for creating 3D models, animated characters, and environments: **Maya, ZBrush,** and **Blender.**
- 2D DCCs for creating 2D images, illustrations, textures, and interfaces: **Photoshop, Illustrator, Substance Painter,** and **Gimp**
- Audio DCCs for recording, editing, and mixing sound effects and music: **Audition, Logic Pro, Reaper,** and **Audacity**
- IDEs for writing code: **Visual Studio** and **Rider**
- Real-time Engines: programs for real-time development, rendering, and publishing of 3D content or applications: **Unity** and **Unreal**

What is a game engine?

Assets Store

- Learning to use a DCC to create assets is out of the scope of this module
- Hundreds of ready-to-use assets created with DCCs are available to you through so-called Assets Stores
- You can download and import assets directly into your game engines

Años de videojuegos me han enseñado que seguro hay algo detrás de este muro.

https://assetstore.unity.com/

What is a game engine?

Made with Unity!

https://unity.com/made-with-unity

What is a game engine?

Few words about Unity

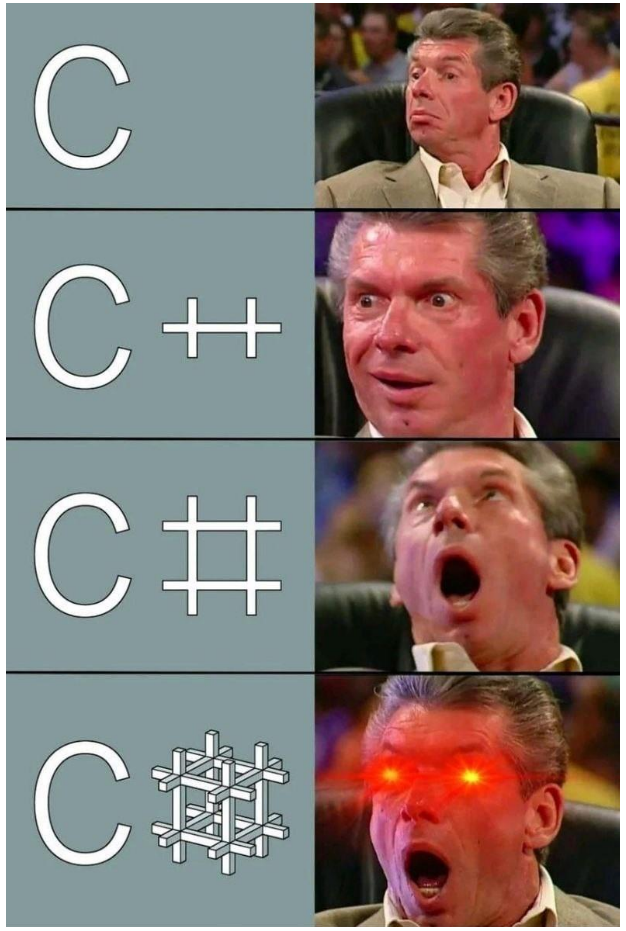
- Unity is a tool developed by Unity Technologies whose engine includes different modules to manage both **2d and 3d physics, audio, animations and rendering**

- The tool allows you to generate projects for various platforms, both **mobile and desktop, web or consoles**

What is a game engine?

Few words about Unity

- The **programming language** it uses is **C#**, very similar to Java and developed by Microsoft
- Over time, the language has been standardized, and **free implementations** have been created
- One such implementation, **Mono**, is what Unity is based on



What is a game engine?

Few words about Unity

- In order to work with Unity, we need to have a **user account**, to which we will link services, purchases and downloads
- In terms of licenses, Unity has a subscription model:

PLAN	ELIGIBILITY	COST
Student	Enrolled in an accredited educational institution and can provide consent to the collection and processing of their personal information	Free
Personal	Revenue or funding less than $100K in the last 12 months	Free
Plus	Less than $200K of revenue or funds raised in the last 12 months	369€ / year
Pro	-	1877€ / year

What is a game engine?

Few words about Unity

What is a game engine?

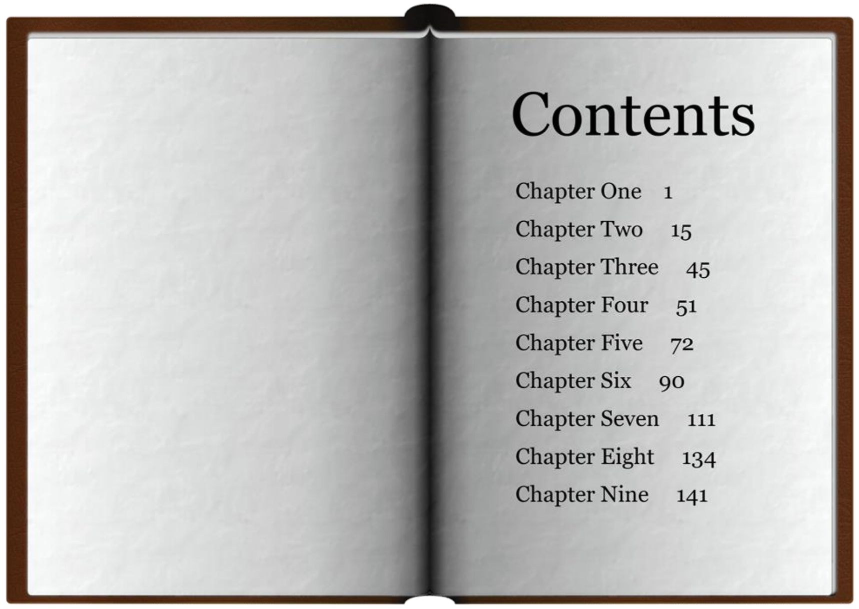
Few words about Unity

Unity Hub is a tool that allows us to manage, in a centralized way:

- Our Unity account
- Licenses
- Our projects
- Different installations (versions) of the Unity Editor
- Different modules of Unity Editor

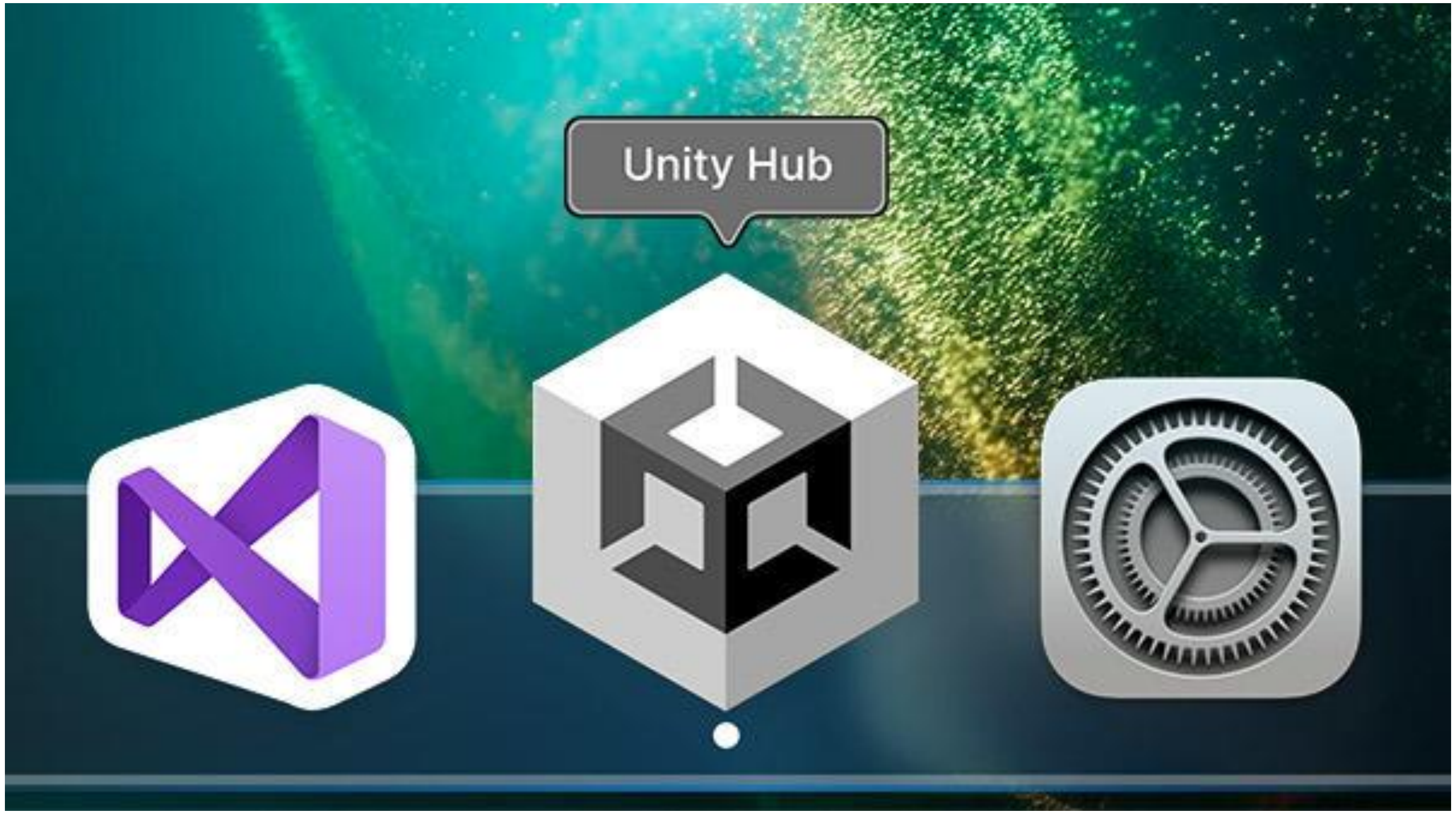
Content

1. What is a game engine?
2. Unity environment installation
3. Explore the Unity Editor
4. Working with 3D GameObjects
5. Create using primitives
6. Add components
7. Add physical properties
8. Manage GameObjects with prefabs
9. Get 3D assets
10. Publish your project
11. ACTIVITY



Unity environment installation

What will you need?



Unity environment installation

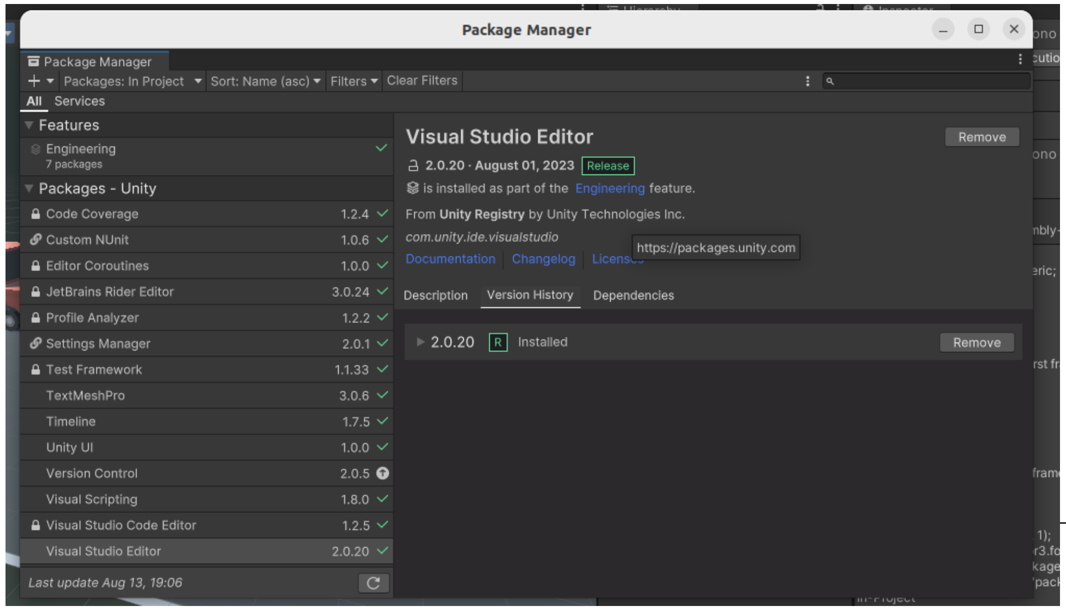
What will you need?

- UNITY:
  1) Unity Hub
  2) Unity Editor 2022.3.7f1 LTS + WebGL module
- VS CODE:
  1) Extensions: "Unity", "C#" and "C# Dev Kit"
  2) Extension ".NET install tool for …" (it should be automatically installed together with previous) --> Check it
  3) .NET SDK --> how-to-link

Unity environment installation

What will you need?

- Once all the previous is done and working, you've to:
  - Set VSC as your default Unity external code editor
  - Update the package "Visual Studio Editor" to 2.0.20 to make it compatible with VSC Unity extension



Package Manager

Package Manager

Packages: In Project | Sort: Name (asc) | Filters | Clear Filters

All | Services

Features

Engineering ✓
7 packages

Packages - Unity

Code Coverage	1.2.4 ✓
Custom NUnit	1.0.6 ✓
Editor Coroutines	1.0.0 ✓
JetBrains Rider Editor	3.0.24 ✓
Profile Analyzer	1.2.2 ✓
Settings Manager	2.0.1 ✓
Test Framework	1.1.33 ✓
TextMeshPro	3.0.6 ✓
Timeline	1.7.5 ✓
Unity UI	1.0.0 ✓
Version Control	2.0.5
Visual Scripting	1.8.0 ✓
Visual Studio Code Editor	1.2.5 ✓
Visual Studio Editor	2.0.20 ✓

Last update Aug 13, 19:06

Visual Studio Editor | Remove

2.0.20 · August 01, 2023 | Release

is installed as part of the Engineering feature.

From Unity Registry by Unity Technologies Inc.

com.unity.ide.visualstudio

Documentation | Changelog | Licenses

https://packages.unity.com

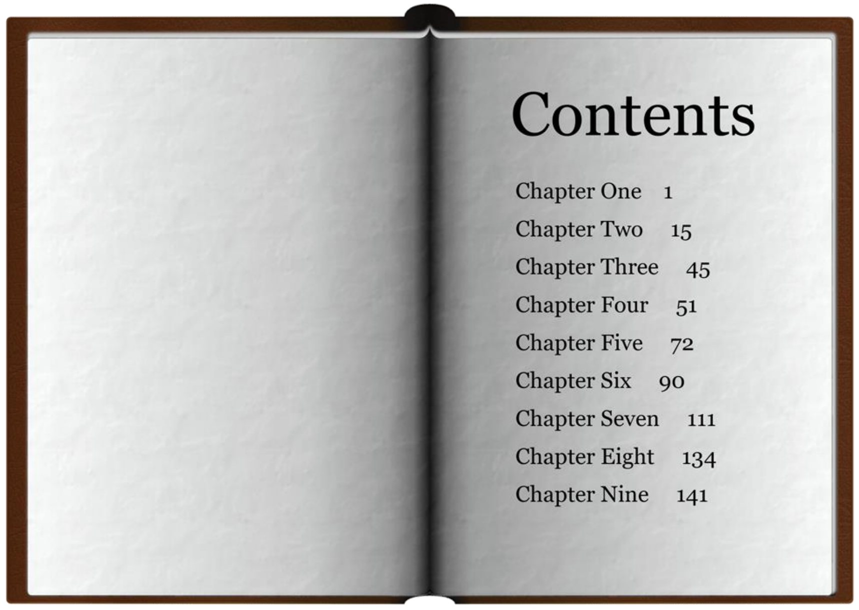
Description | Version History | Dependencies

2.0.20 | R | Installed | Remove

https://code.visualstudio.com/docs/other/unity

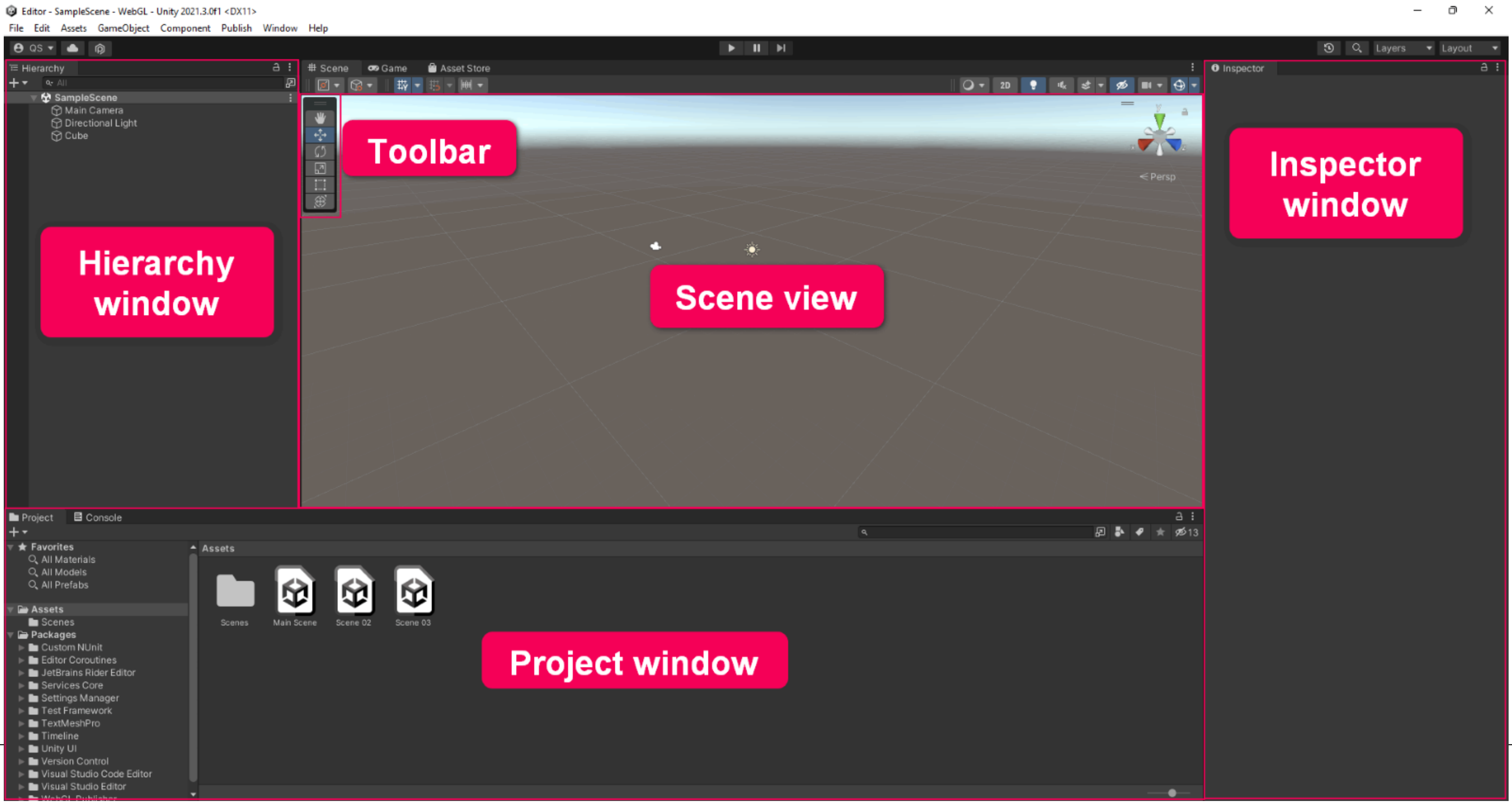
Content

1. What is a game engine?
2. Unity environment installation
3. Explore the Unity Editor
4. Working with 3D GameObjects
5. Create using primitives
6. Add components
7. Add physical properties
8. Manage GameObjects with prefabs
9. Get 3D assets
10. Publish your project
11. ACTIVITY



Explore the Unity Editor

Introduction to the Unity Editor



Explore the Unity Editor

Introduction to the Unity Editor

Scene view and Game view

- Scene is your interactive window into the world you are creating
- You'll use the Scene view to manipulate objects and view them from various angles
- You'll use the Game view to playtest your game

Hierarchy window

- Where you can organize all the things in your project. These things are called **GameObjects**

Explore the Unity Editor

Introduction to the Unity Editor

Project window

- Where you can find all the files (**assets**) available for use in your project, whether you use them or not
- Works like a file explorer, organized in folders. You can drag assets directly from the Project window into the Scene view to add them to the scene
- Difference between the Project and Hierarchy windows ???

Explore the Unity Editor

Introduction to the Unity Editor

Inspector window

- You'll find and configure detailed information about GameObjects
- When you select a GameObject, you'll see its **components** in the Inspector. Components describe the properties and behaviors of GameObjects

Toolbar

- To change your point of view in the scene and start/stop the Play Mode
- Scene navigation functions: move, rotate and scale your selected GameObjects

Explore the Unity Editor

Using Scenes in your project

- Projects in Unity Editor are organized into **scenes**
- Scenes are containers for everything in the experience you are creating
- One way to think about scenes is as discrete experiences
- For example, each level in a game could be a separate scene, and the game’s main menu could be another
- A Unity project can have one scene or more than a hundred (at least one scene)

Explore the Unity Editor

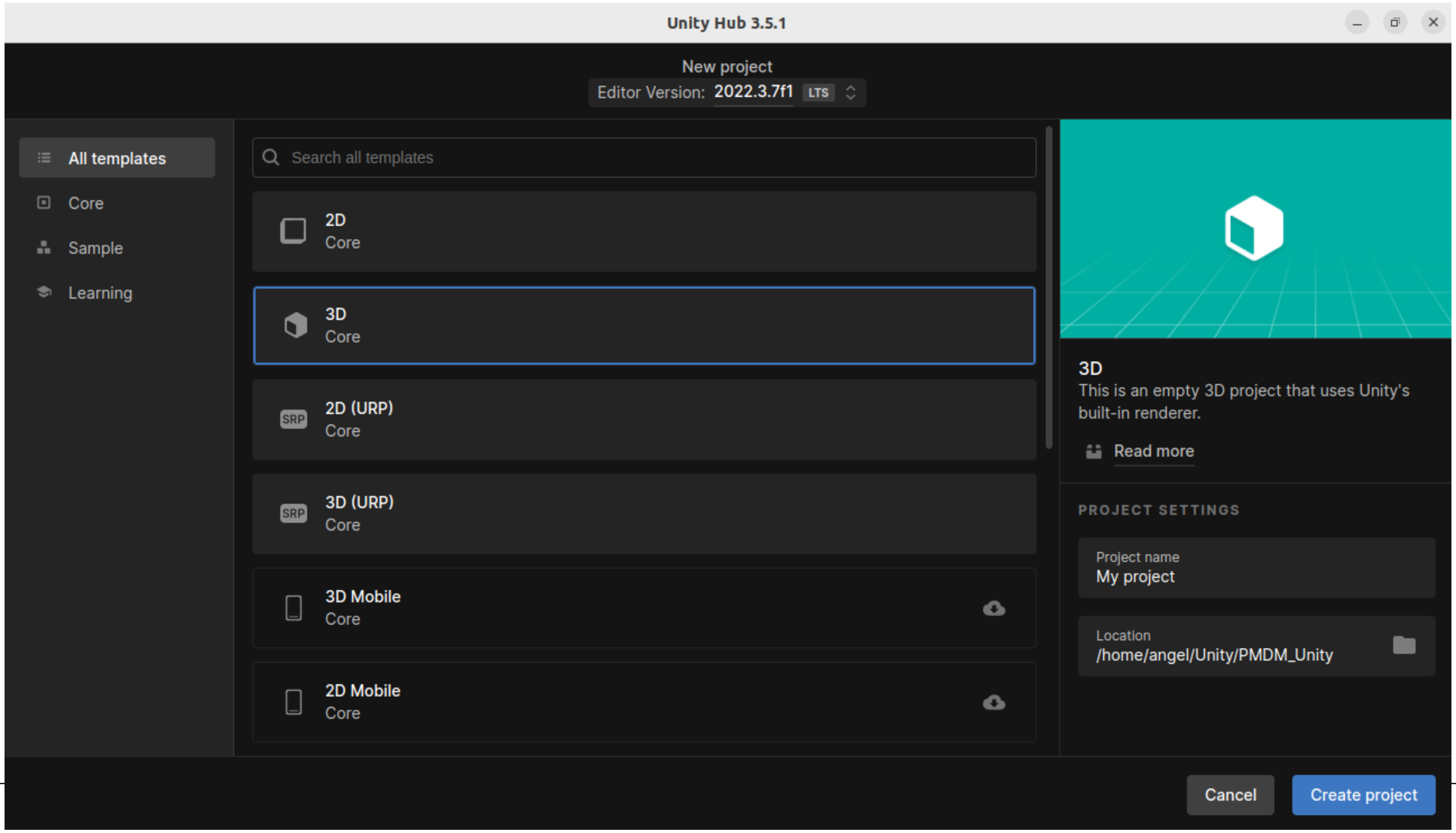
Navigating the scene

- Is like operating a drone camera
- With practice, you can learn to navigate with ease <-- Very important !!
- **Pan**: Select the Hand tool and click and drag
- **Zoom**: Holding Alt + right-click + drag (also Mouse Wheel)
- **Orbit**: Holding Alt + left-click + drag to orbit around the pivot point
- **Focus**: When a GameObject is selected, press F to focus your view on that GameObject
- **Flythrough mode**: Click and hold the right mouse button

Explore the Unity Editor

Navigating the scene

Let's start navigating the scenes by creating a new Unity 3D project


Unity Hub 3.5.1

New project
Editor Version: 2022.3.7f1 LTS

3D
This is an empty 3D project that uses Unity's built-in renderer.

PROJECT SETTINGS

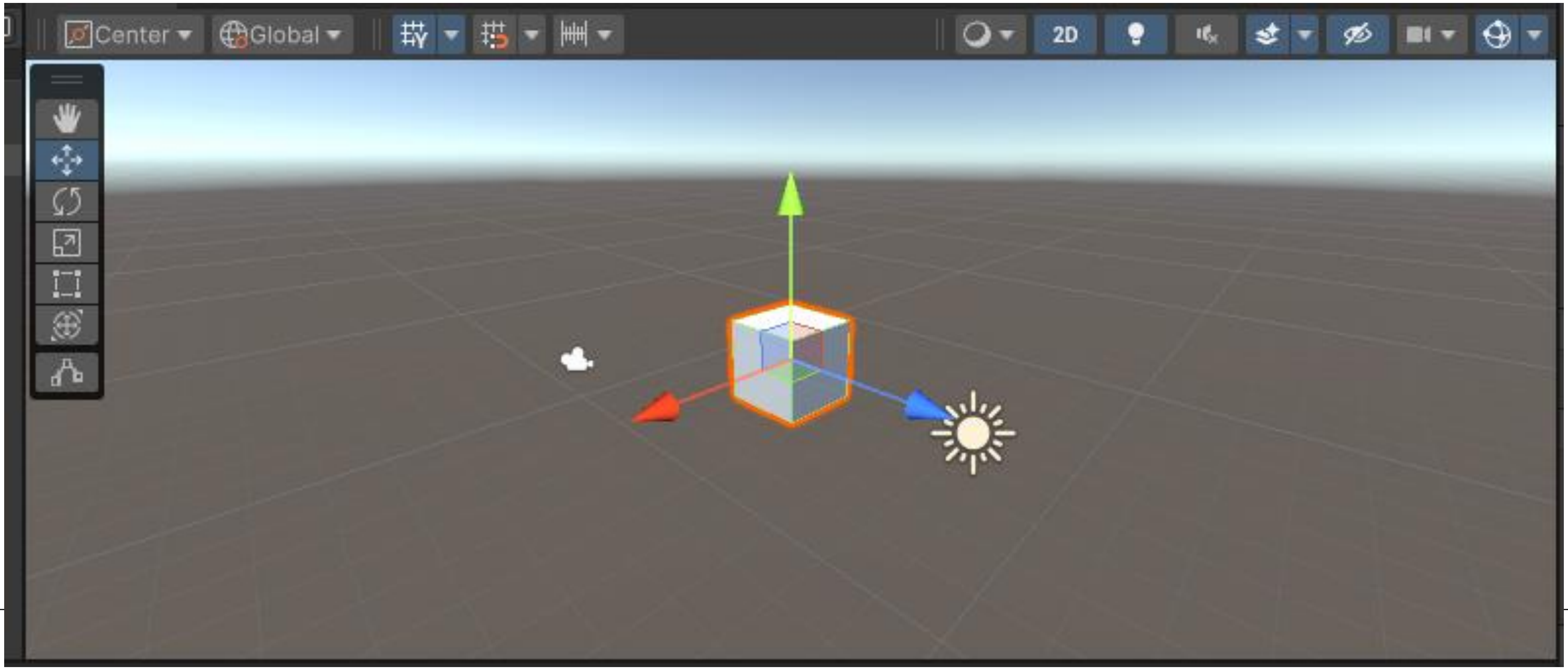
Project name
My project

Location
/home/angel/Unity/PMDM_Unity

Explore the Unity Editor

Navigating the scene

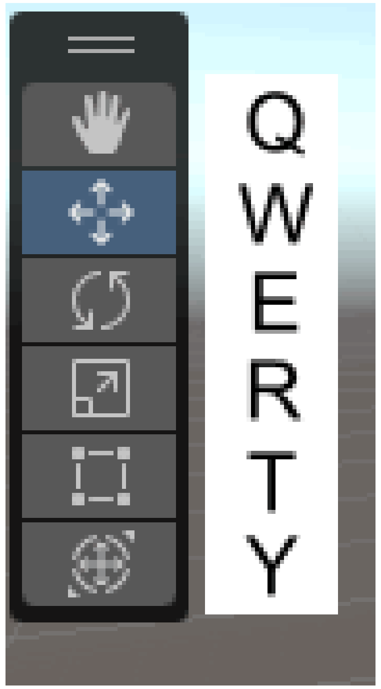
Create your 1st GameObject to practice --> right-click in the Hierarchy window and select **3D Object > Cube**



Explore the Unity Editor

Navigating the scene

- The keyboard shortcuts for the toolbars correspond to QWERTY keys
- Using these keys, you can switch quickly between the tools

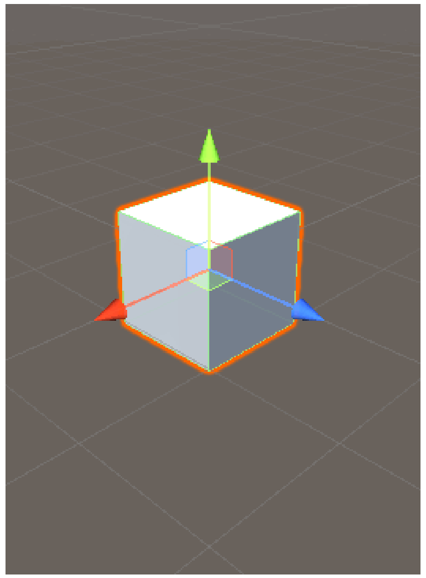
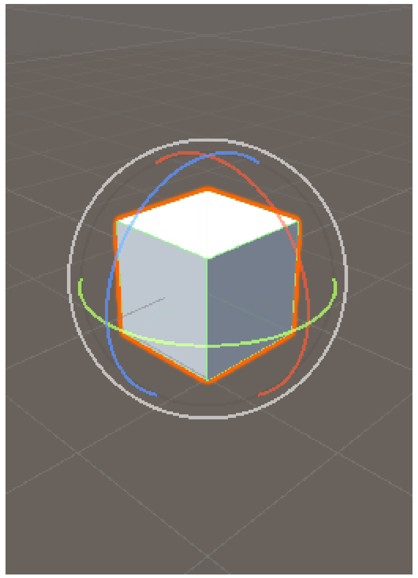
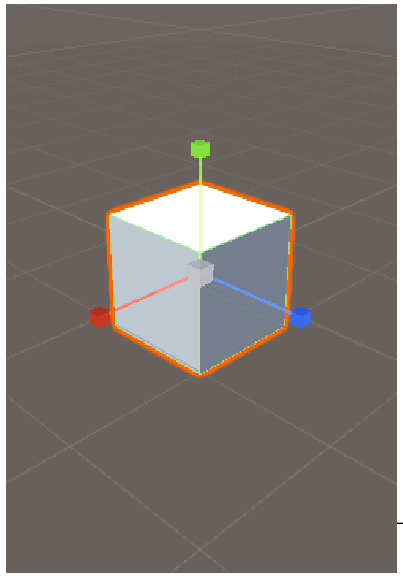
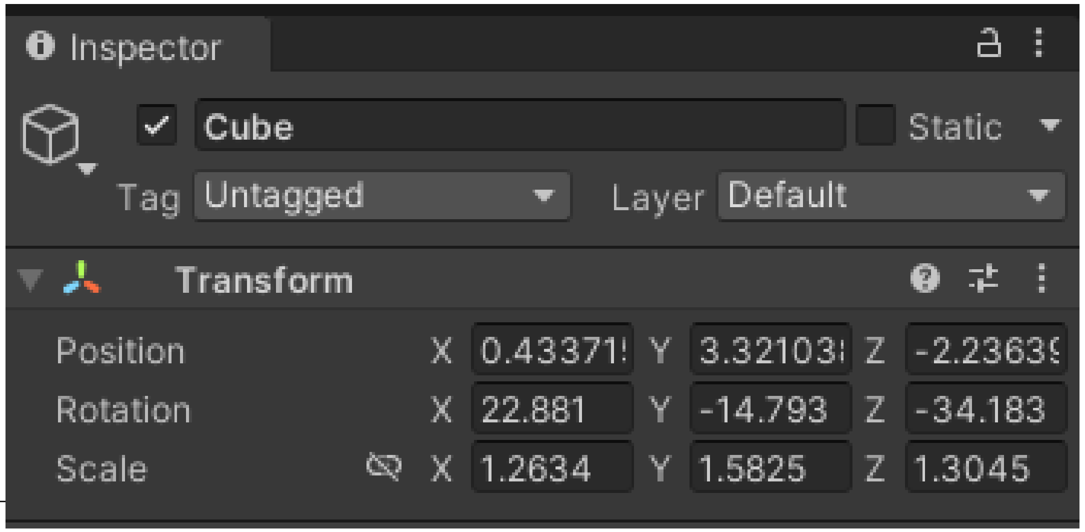


Q: Hand tool, to pan your view
W: Move tool, to select and change position
E: Rotate tool, to select and rotate
R: Scale tool, to select and change size
T: Rect Transform tool, to scale in 2D
Y: Transform tool, to move, scale, and rotate with one Gizmo

Explore the Unity Editor

Navigating the scene

- For each of the transform tools, a **Gizmo** appears that allows you to manipulate the GameObject along each specific axis
- As you manipulate these controls, the values in the **Transform Component** change accordingly

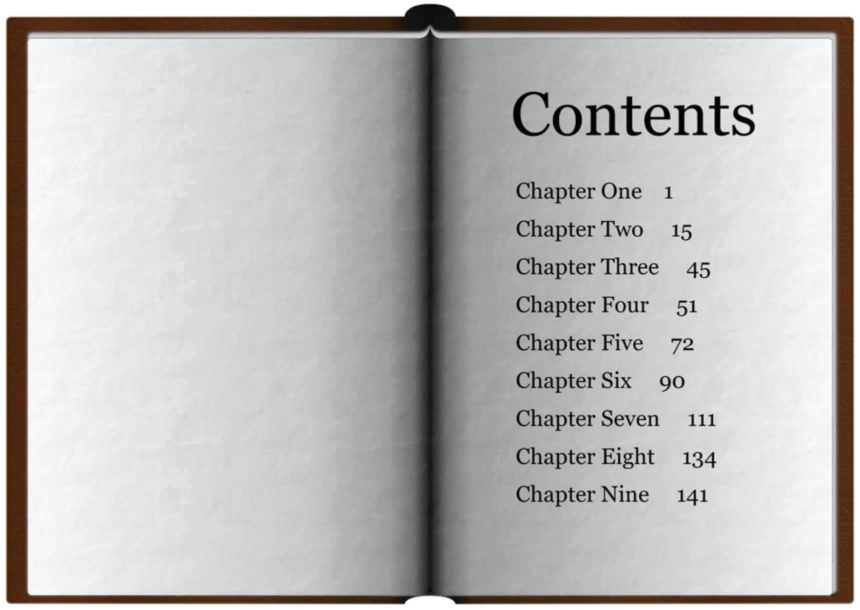





Cube
Tag Untagged Layer Default

Position	X	0.43371!	Y	3.32103!	Z	-2.23639
Rotation	X	22.881	Y	-14.793	Z	-34.183
Scale	X	1.2634	Y	1.5825	Z	1.3045

Content

1. What is a game engine?
2. Unity environment installation
3. Explore the Unity Editor
4. Working with 3D GameObjects
5. Create using primitives
6. Add components
7. Add physical properties
8. Manage GameObjects with prefabs
9. Get 3D assets
10. Publish your project
11. ACTIVITY



Contents

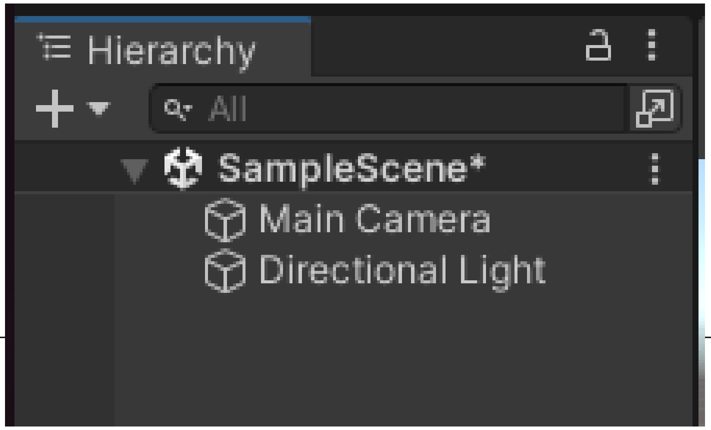
Chapter One	1
Chapter Two	15
Chapter Three	45
Chapter Four	51
Chapter Five	72
Chapter Six	90
Chapter Seven	111
Chapter Eight	134
Chapter Nine	141

Working with 3D GameObjects

Default 3D scene

The default 3D scene comes equipped with two important GameObjects, which are listed in the Hierarchy window:

- **Main Camera:** controls what your players will see in the Game view (Play mode)
- **Directional Light:** simulates the sun and provides light that will reflect off your 3D GameObjects to create realistic visual effects

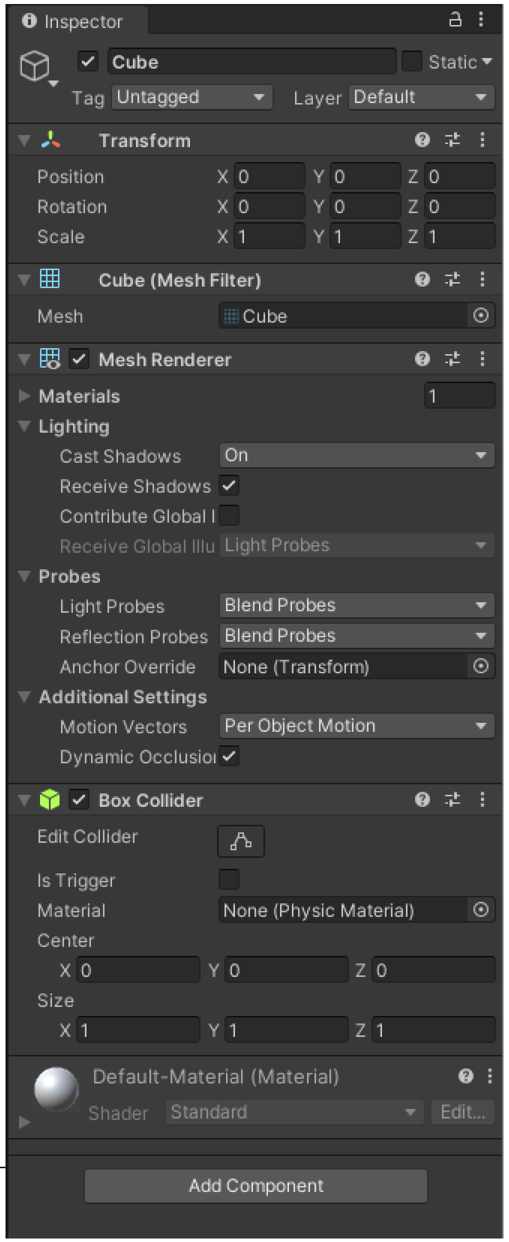


Working with 3D GameObjects

The Inspector

- To view and change the properties of GameObjects
- Each section of the Inspector represents a **component**, which is a set of properties and behaviors of the selected GameObject
- Some components are built-in to primitives like the ones you see here
- Later, we will add more components to give GameObjects more sophisticated properties and behaviors

Note: Unity uses a **Y-up coordinate system**



Inspector
✓ Cube — 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

Cube (Mesh Filter)
Mesh Cube

✓ Mesh Renderer
Materials 1
Lighting
Cast Shadows On
Receive Shadows ✓
Contribute Global I
Receive Global Illu Light Probes
Probes
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Additional Settings
Motion Vectors Per Object Motion
Dynamic Occlusioı ✓

✓ Box Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 0 Y 0 Z 0
Size X 1 Y 1 Z 1

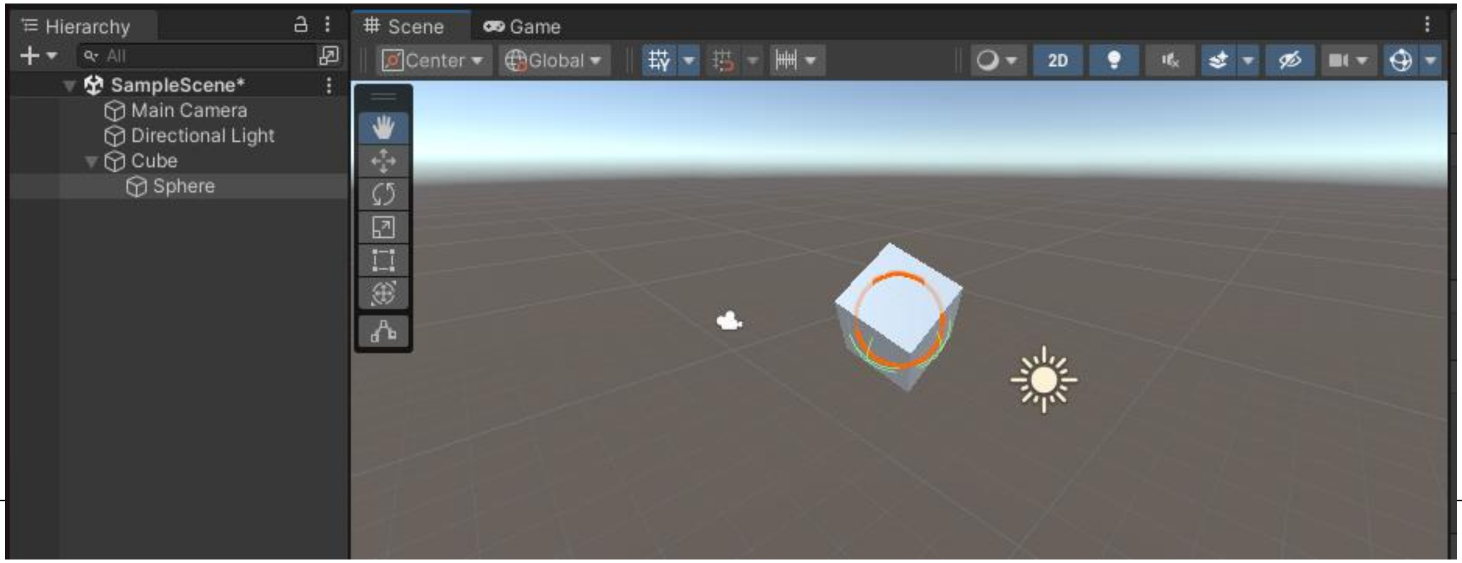
Default-Material (Material)
Shader Standard — Edit...

Add Component

Working with 3D GameObjects

Organize GameObjects in the hierarchy

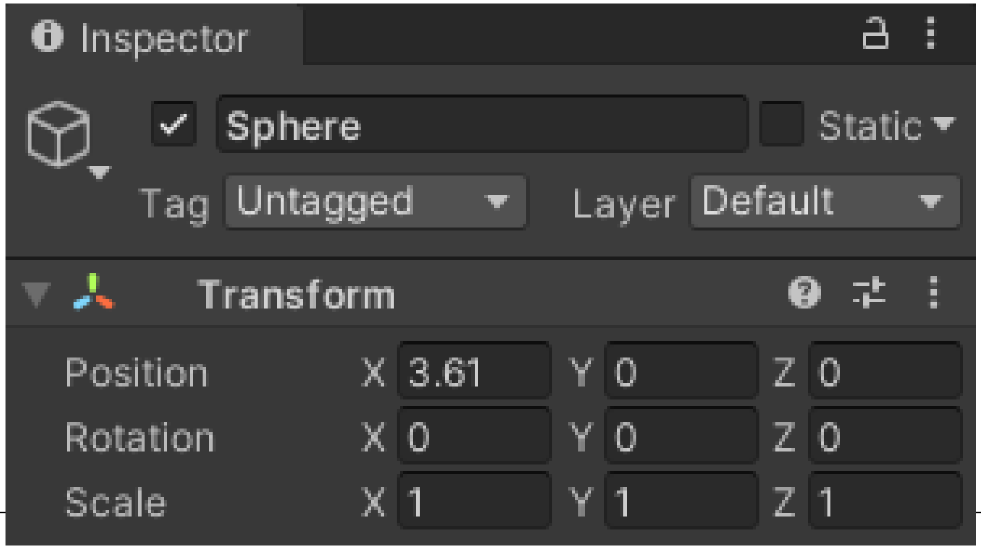
- Use the Hierarchy to define the relationships between GameObjects
- You can group them to create more complex GameObjects (parent-child)
- Create a child sphere --> right-click the Cube + Select 3D Object > Sphere



Working with 3D GameObjects

Organize GameObjects in the hierarchy

- Move, Scale and Rotate the Cube --> What happens to the Sphere?
- Select the Sphere and notice its coordinates in the Transform component --> These are **relative coordinates**, which store the Sphere's relationship to Cube (not Sphere's position, rotation, and scale in the scene)



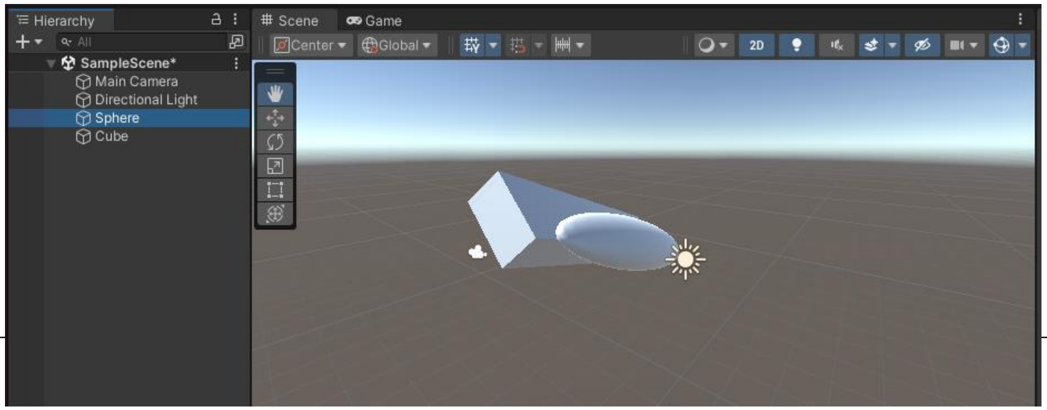
Sphere ✓ Static
Tag Untagged Layer Default

	X	Y	Z
Position	3.61	0	0
Rotation	0	0	0
Scale	1	1	1

Working with 3D GameObjects

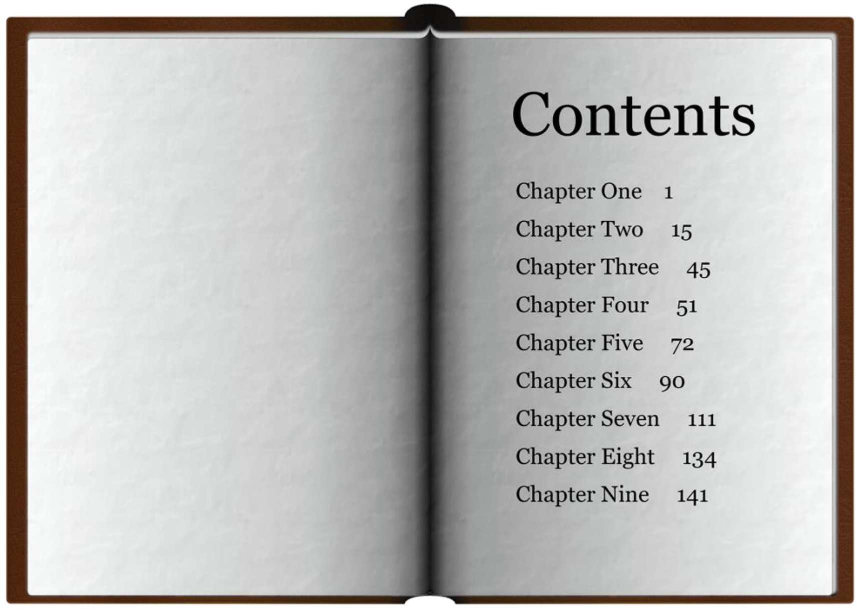
Organize GameObjects in the hierarchy

- Move, Scale and Rotate the Sphere--> What happens to Transform component?
- To remove the parent-child relationship, drag Sphere in the Hierarchy to the root level
- Both GameObjects now appear as independent, and Sphere's Transform component displays coordinates in the scene space



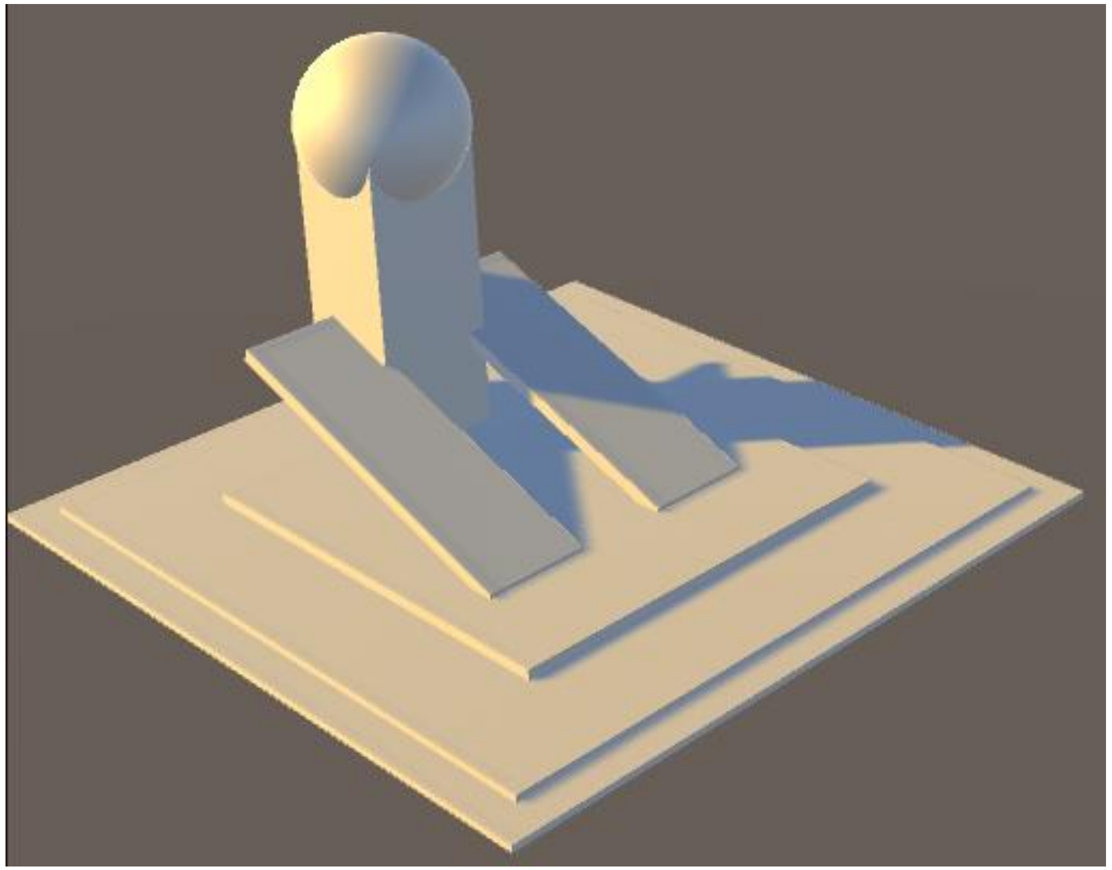
Content

1. What is a game engine?
2. Unity environment installation
3. Explore the Unity Editor
4. Working with 3D GameObjects
5. Create using primitives
6. Add components
7. Add physical properties
8. Manage GameObjects with prefabs
9. Get 3D assets
10. Publish your project
11. ACTIVITY



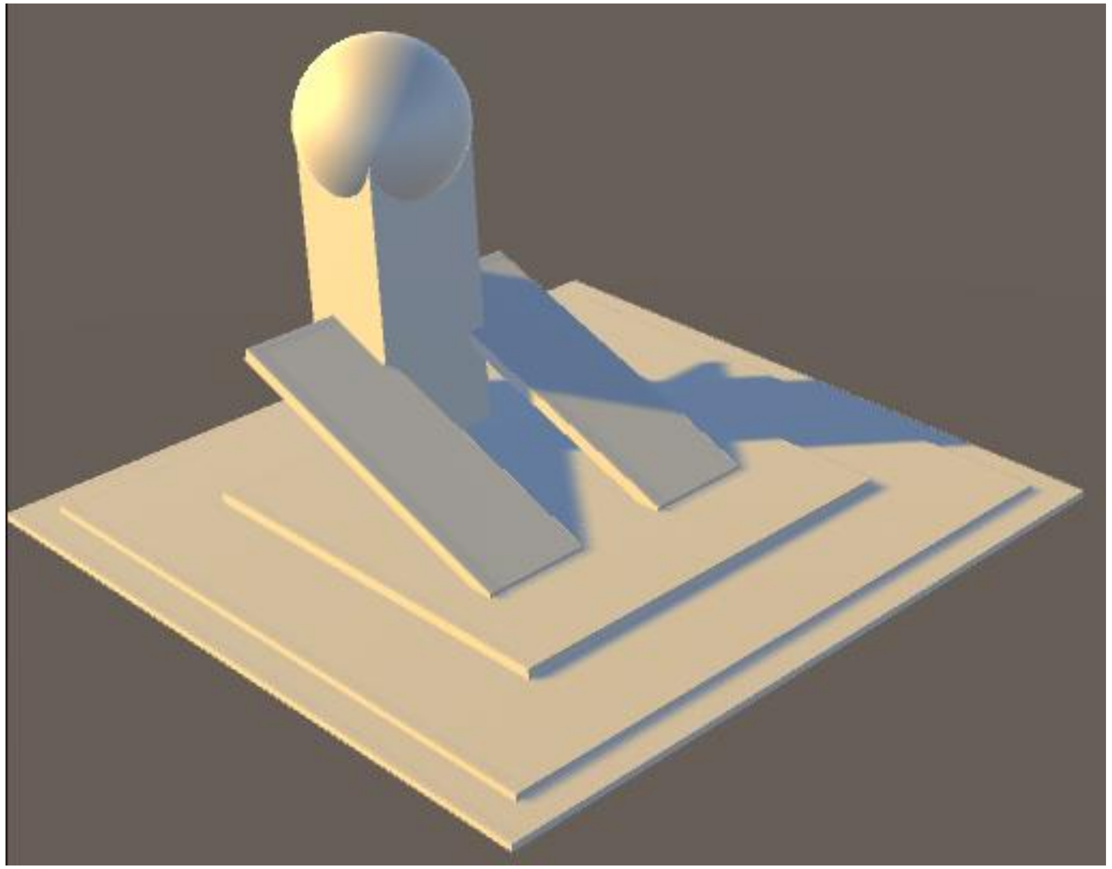
Create using primitives

- It's time to build a **GameObject made up of multiple primitives**
- Here you'll learn how to create and manipulate GameObjects
- It will also give you some practice in navigating around a scene <-- tough and important



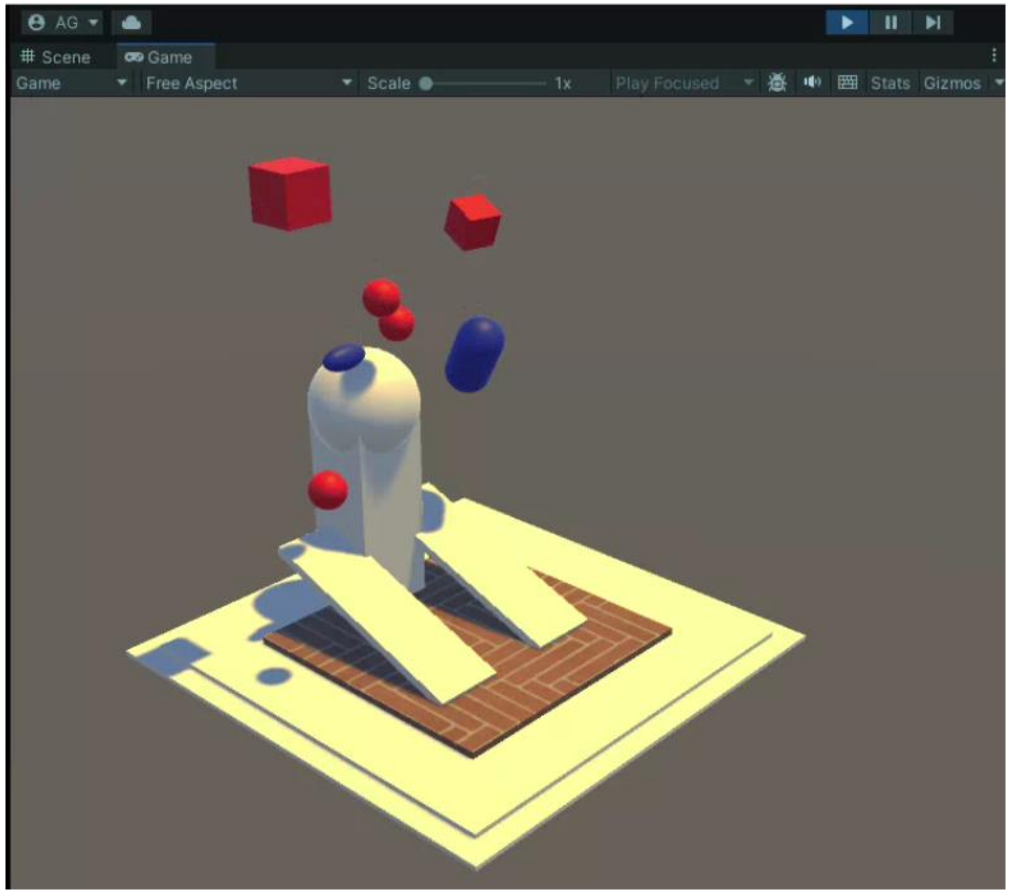
Create using primitives

- You will use this structure later as a surface to catch and deflect falling objects
- The structure you'll build can be **anything you want**, as long as it can be built from primitive cubes, spheres, cylinders, and capsules
- Each student must create **its own structure**



Create using primitives

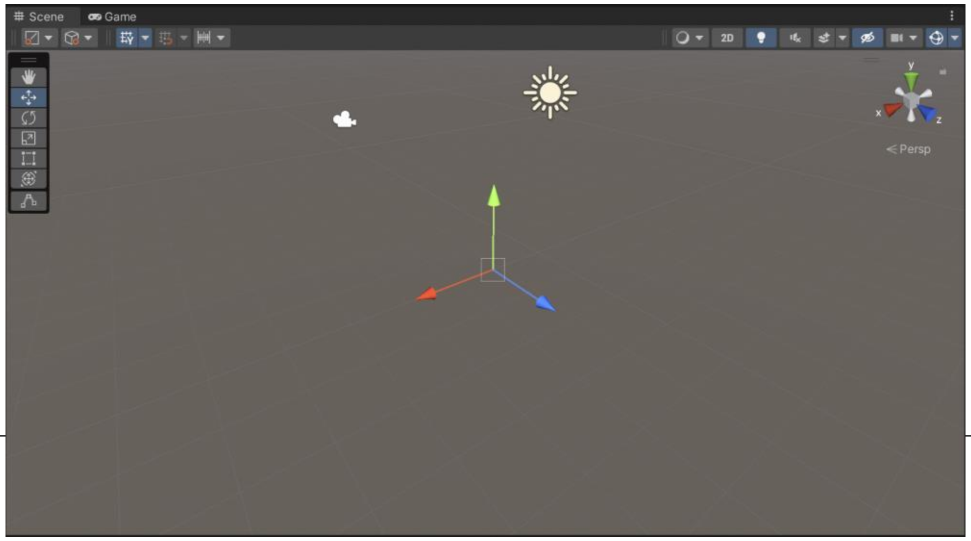
RESULT VIDEO – Falling Objects



Create using primitives

1- Add an empty GameObject

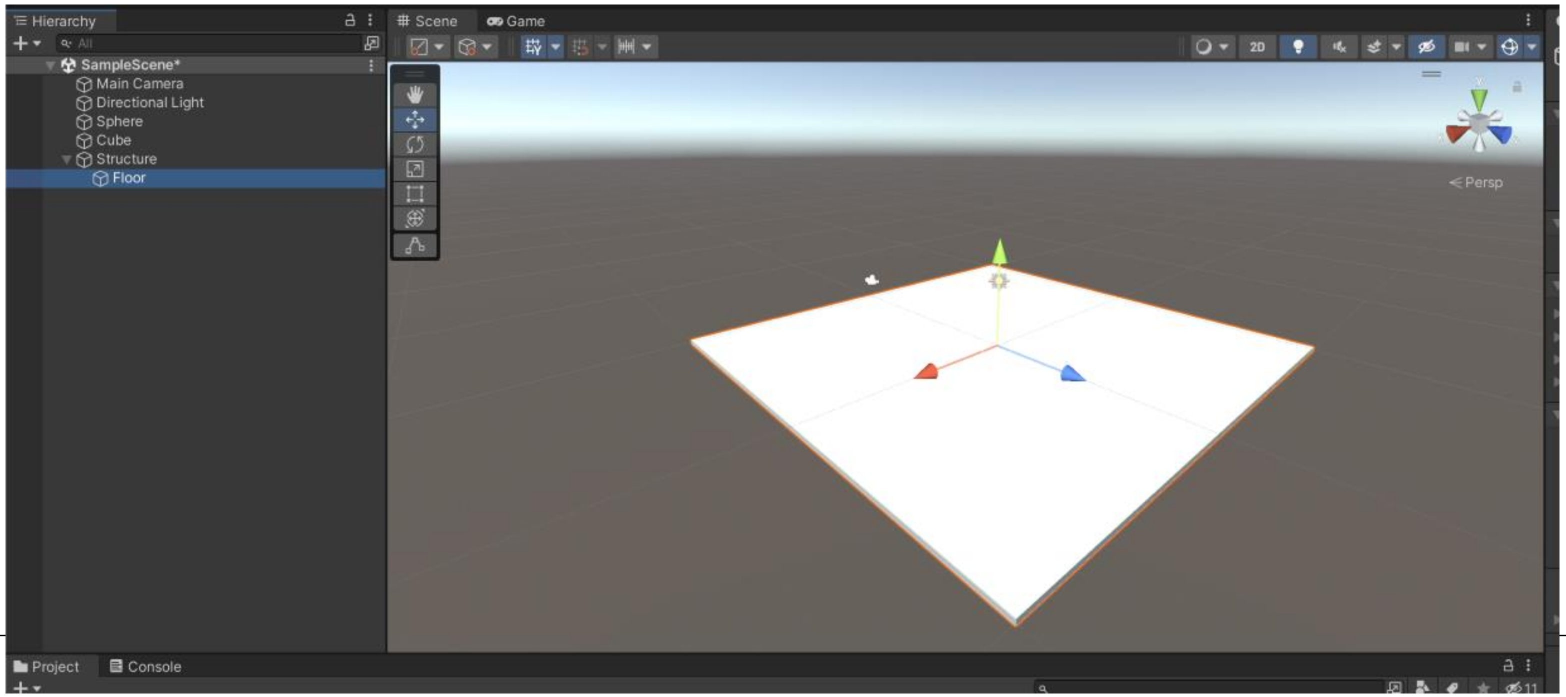
- An empty GameObject is a **placeholder** object that can be created in the Hierarchy
- It does not have a visible representation in the scene, and it can act as a container for other GameObjects
- Rename the GameObject to "Structure" and reset its position



Create using primitives

2- Create a floor

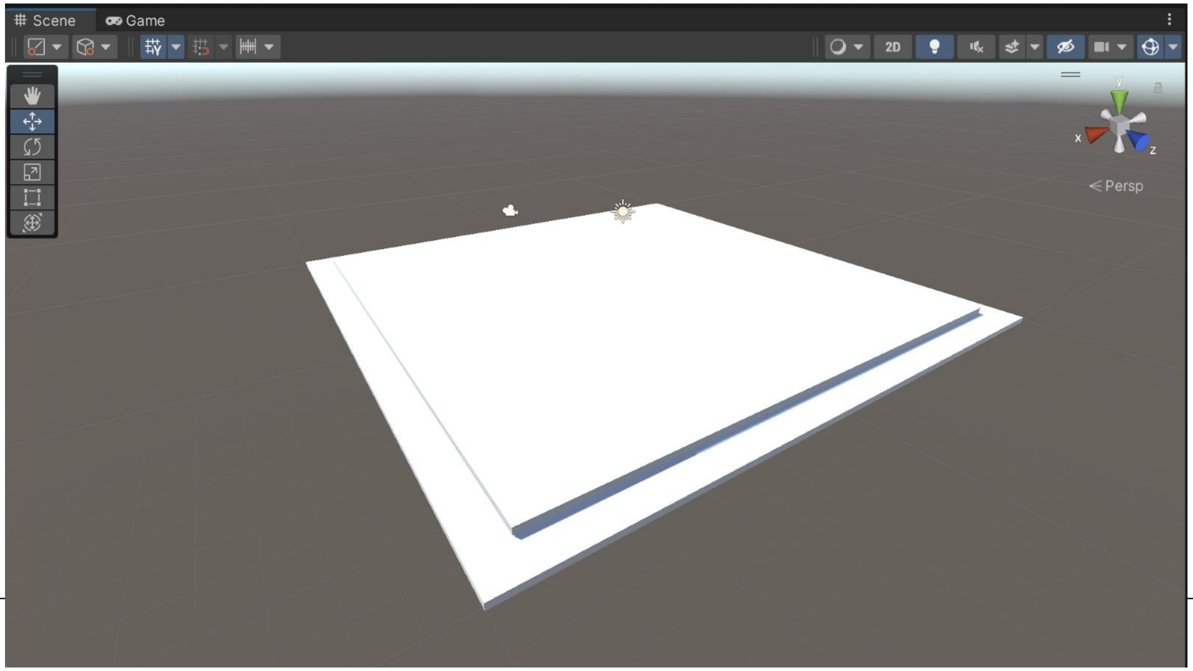
- Create a new cube primitive as a child of the empty GameObject and rename it
- Scale the cube to make it look like a floor



Create using primitives

2- Create a floor

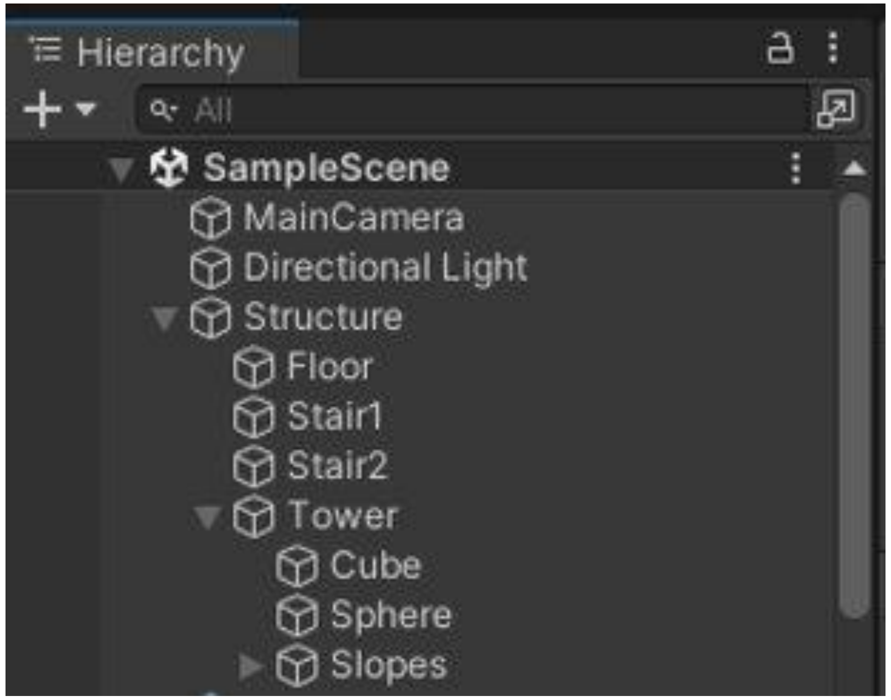
- Duplicate the floor and replace it to create stairs
- Use Ctrl + D to duplicate GameObjects
- Remember to give proper names to all GameObjects



Create using primitives

3- Nest and Combine

- Feel free to change the shapes and transforms for your own structure. The following are just examples
- Create a separate "Tower" from an empty object by combining cubes, spheres …
- Make the entire Tower a child of the Structure
- Rotate, scale, and position the new child GameObject onto your structure

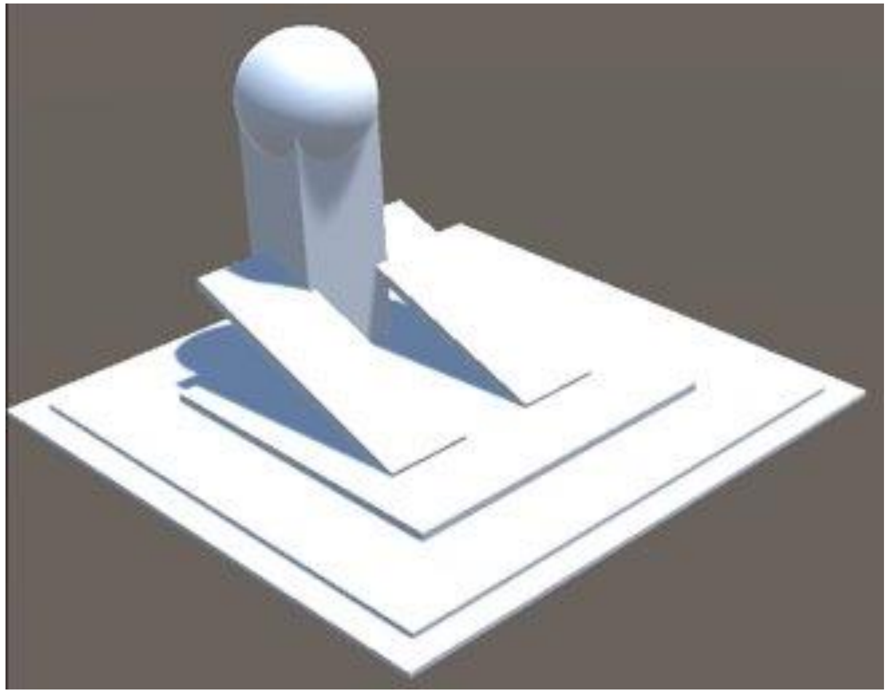


```
Hierarchy
+ ▾  All
▼ SampleScene
    MainCamera
    Directional Light
  ▼ Structure
      Floor
      Stair1
      Stair2
    ▼ Tower
        Cube
        Sphere
      ▶ Slopes
```

Create using primitives

4- Complete your structure

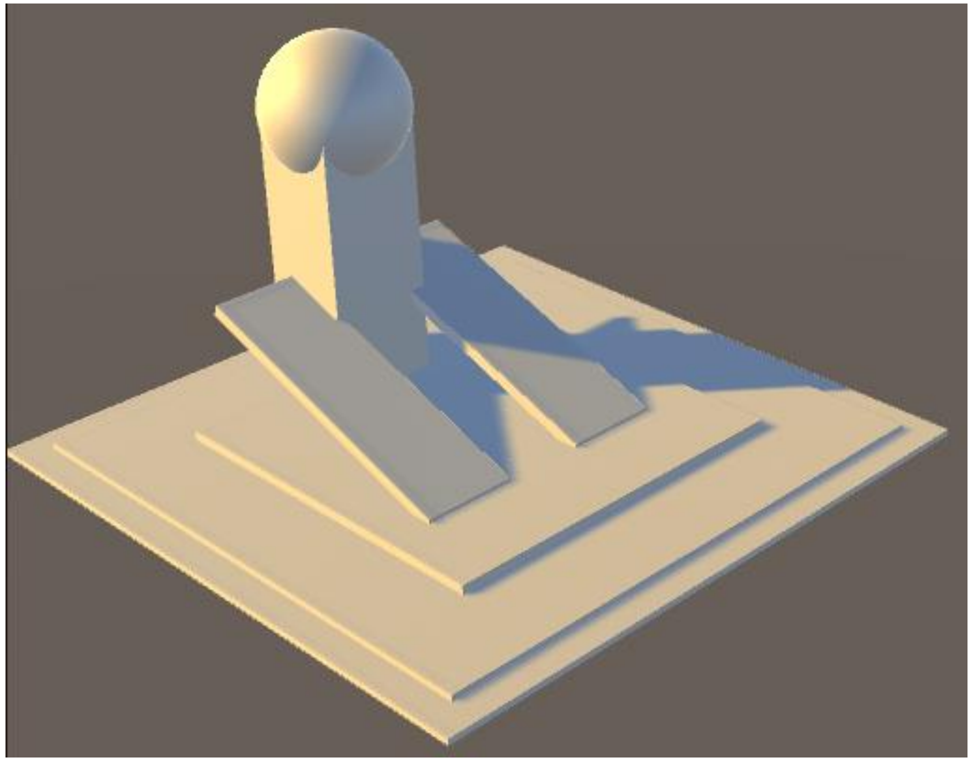
- Add spheres, cylinders, or capsules to create your final structure
- Every new GameObject should be a child of the Structure
- Keep in mind that you will use this structure as a surface for falling objects
- Create surfaces where objects can bounce, roll, and tumble



Create using primitives

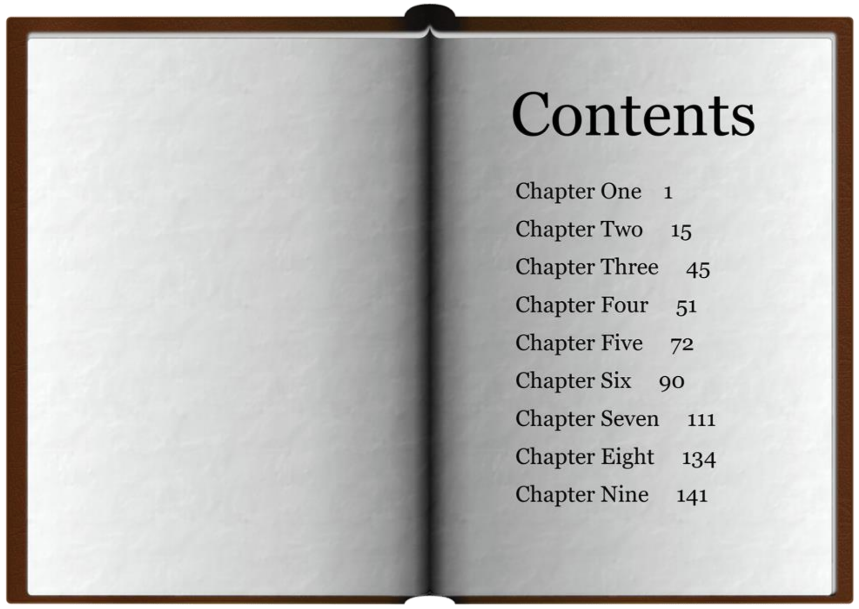
5- Adjust the lighting

- You can manipulate the direction of the light by rotating the **Directional Light** GameObject
- If you like, you can also change the color of the directional light
- In the Inspector, select the color picker in the Light component



Content

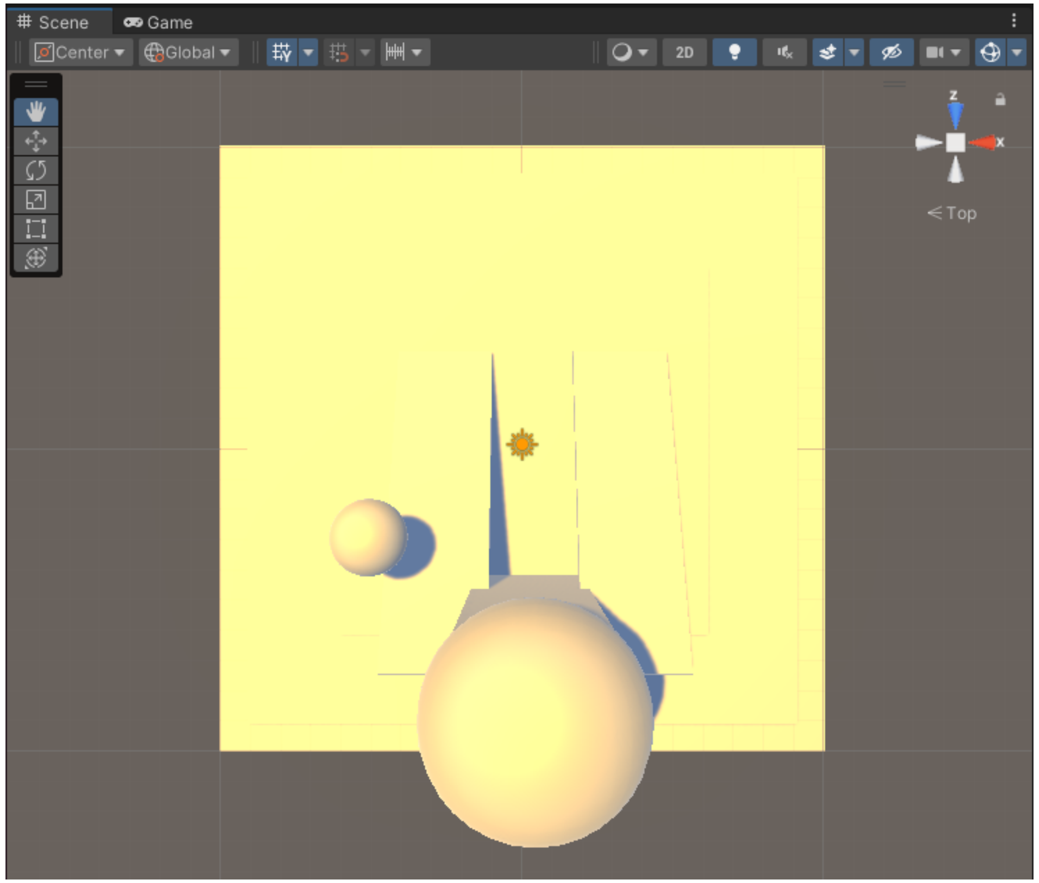
1. What is a game engine?
2. Unity environment installation
3. Explore the Unity Editor
4. Working with 3D GameObjects
5. Create using primitives
6. Add components
7. Add physical properties
8. Manage GameObjects with prefabs
9. Get 3D assets
10. Publish your project
11. ACTIVITY



Add components

6- Place a falling object

- Create a new sphere primitive (not child of any other GameObject)
- Move it to the space above your structure so that it is positioned in “mid-air” (use the **Gizmo – Y**)



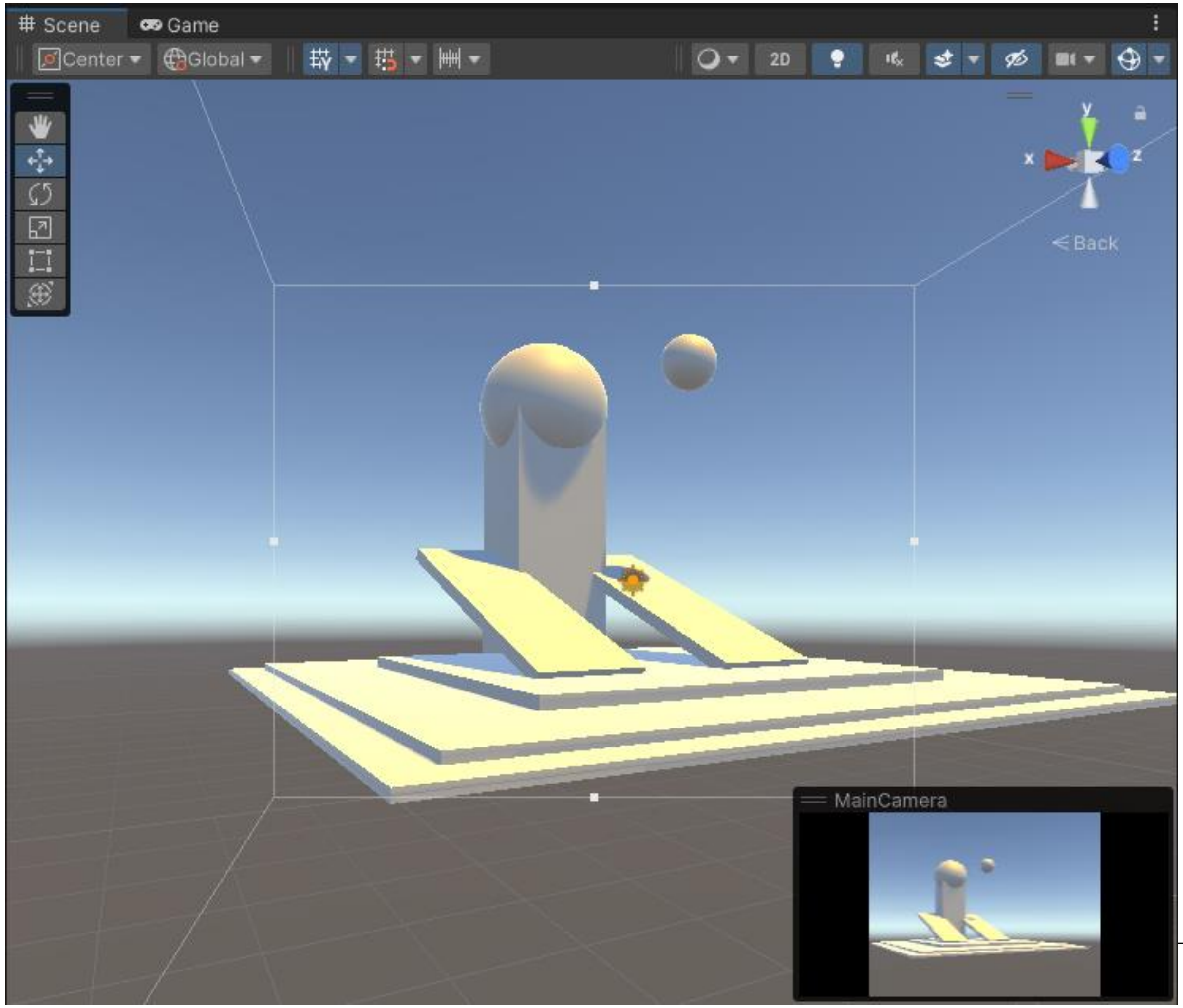
Add components

7- Position the camera

- The **Main Camera** GameObject captures and displays your scene to the user in the Game view
- Move and rotate the camera to get a view of the sphere and the structure below it
- You can also move the camera to **align with your current Scene** view by selecting it in the Hierarchy window and then pressing **Ctrl+Shift+F**
- Manipulate the camera until you get a good view of the structure and the sphere above it

Add components

7- Position the camera



- The **frustum** shows you what part of your scene the camera is viewing
- Use the handles on the sides of the frustum to narrow or widen the view

Now run your application

What has happened?

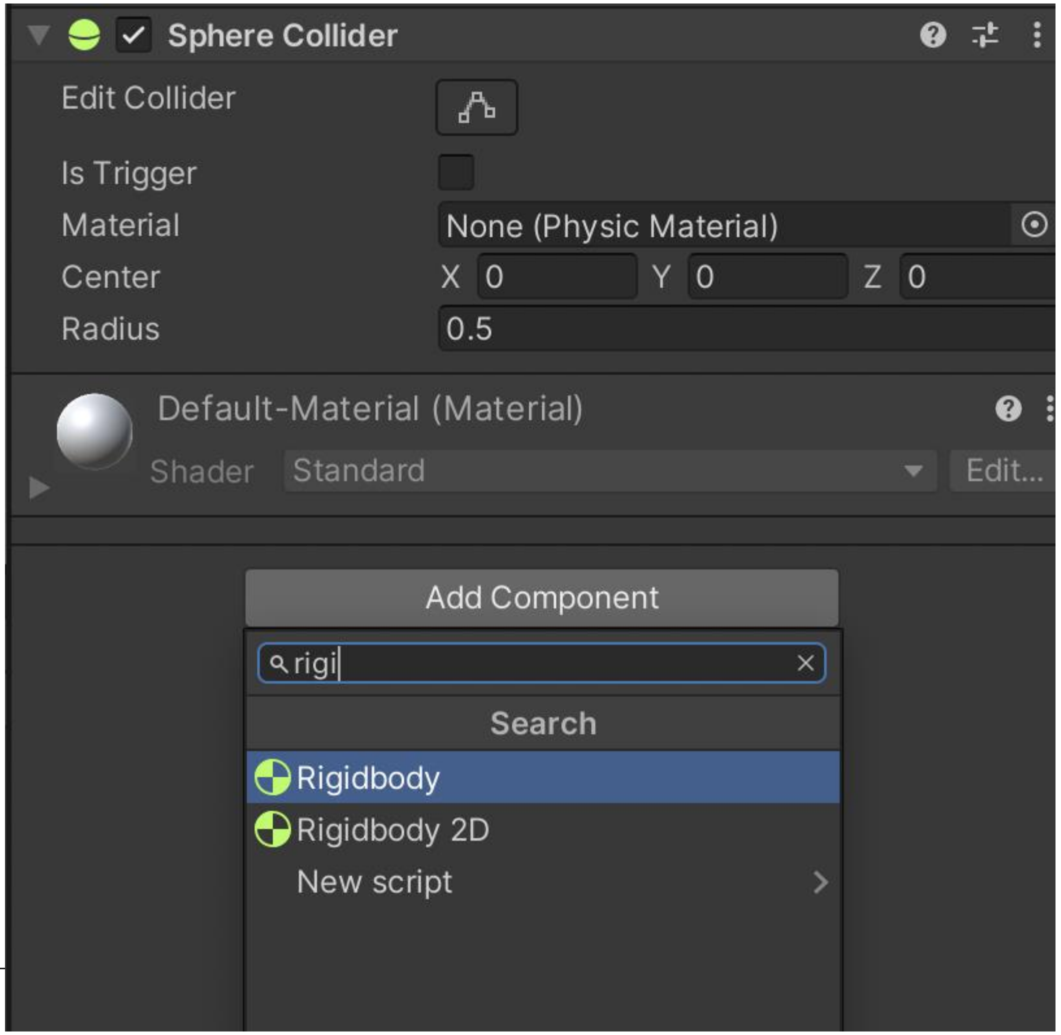
Add components

8- Give mass to GameObjects

- Objects in the physical world don’t hover in mid-air …
- … but in a Unity scene, by default, GameObjects don’t have mass or respond to gravity
- One can give it physical properties by adding a **Rigidbody component** (not the 2D!!)

Run your application again

What has happened now?


Sphere Collider

Edit Collider

Is Trigger

Material: None (Physic Material)

Center: X 0 Y 0 Z 0

Radius: 0.5

Default-Material (Material)

Shader: Standard

Edit...

Add Component

rigi

Search

Rigidbody

Rigidbody 2D

New script

Add components

8- Give mass to GameObjects

Experiment with objects, components and properties

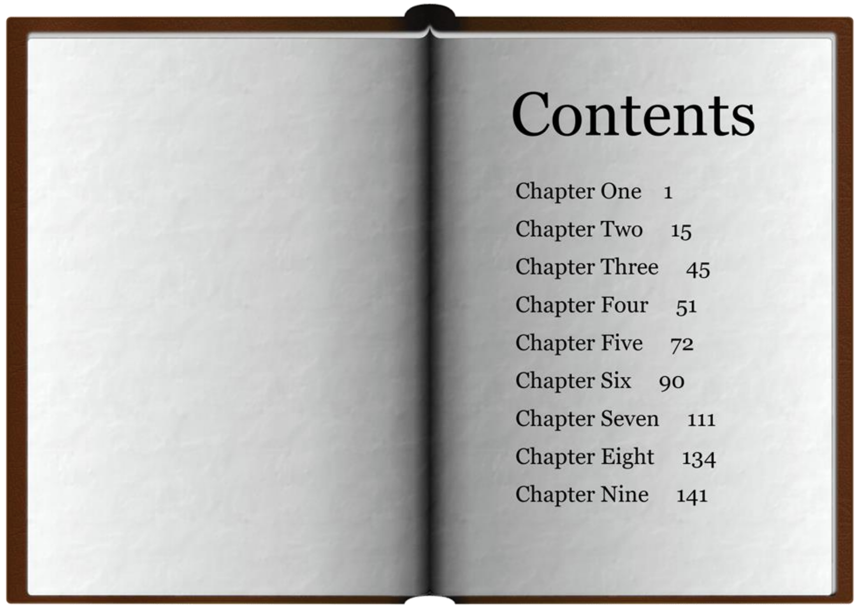
a) Duplicate your sphere and position the duplicates to fall from varying heights and locations on your structure

b) Add other primitives with a Rigidbody component and see how they behave (cubes, capsules...)

c) Experiment with the **Mass property** of the Rigidbody component. (If you make a cube very heavy, it won't bounce or tumble as much)

Content

1. What is a game engine?
2. Unity environment installation
3. Explore the Unity Editor
4. Working with 3D GameObjects
5. Create using primitives
6. Add components
7. Add physical properties
8. Manage GameObjects with prefabs
9. Get 3D assets
10. Publish your project
11. ACTIVITY



Contents

Chapter One 1
Chapter Two 15
Chapter Three 45
Chapter Four 51
Chapter Five 72
Chapter Six 90
Chapter Seven 111
Chapter Eight 134
Chapter Nine 141

Add physical properties

9- Materials

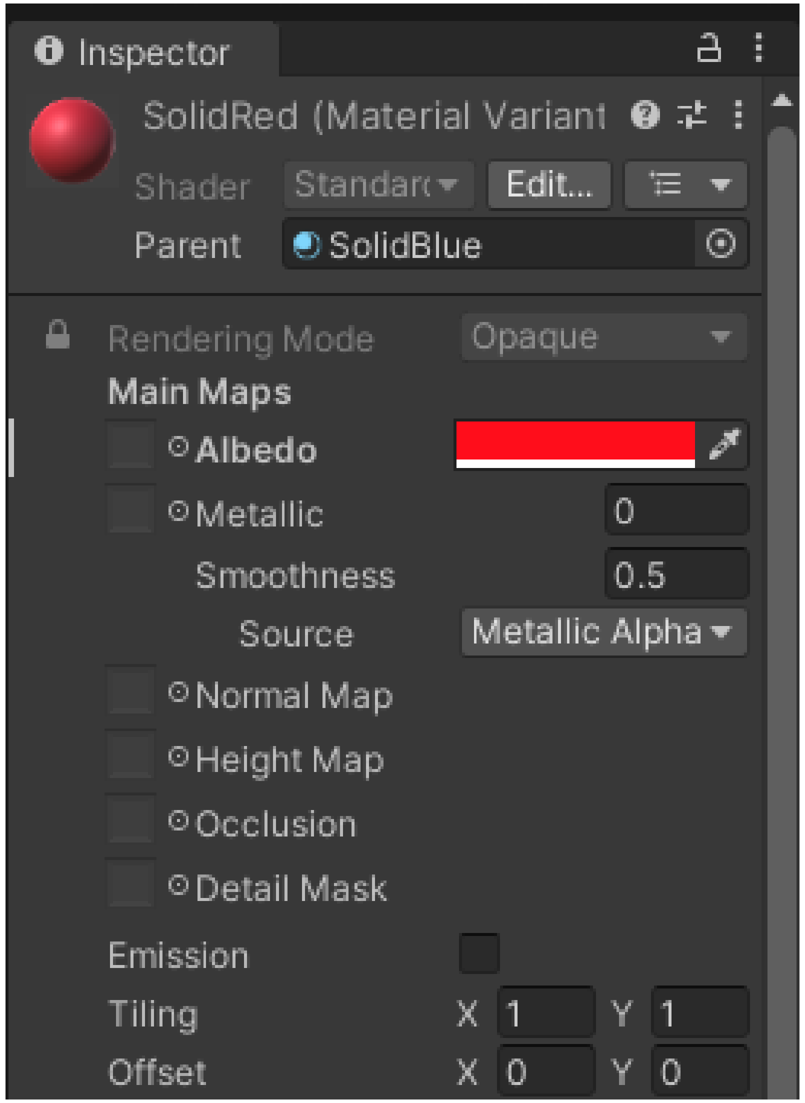
- **Materials** are components that define the surface characteristics of objects
- One can create simple materials to change the visual appearance of GameObjects and apply different materials to different GameObjects to manage the ways the objects look

1. Create a Materials folder inside your Assets folder
2. Then add a new material by: Right-click + **Create** > **Material**
3. Give a proper name (e.g. SolidRed)

Add physical properties

9- Materials

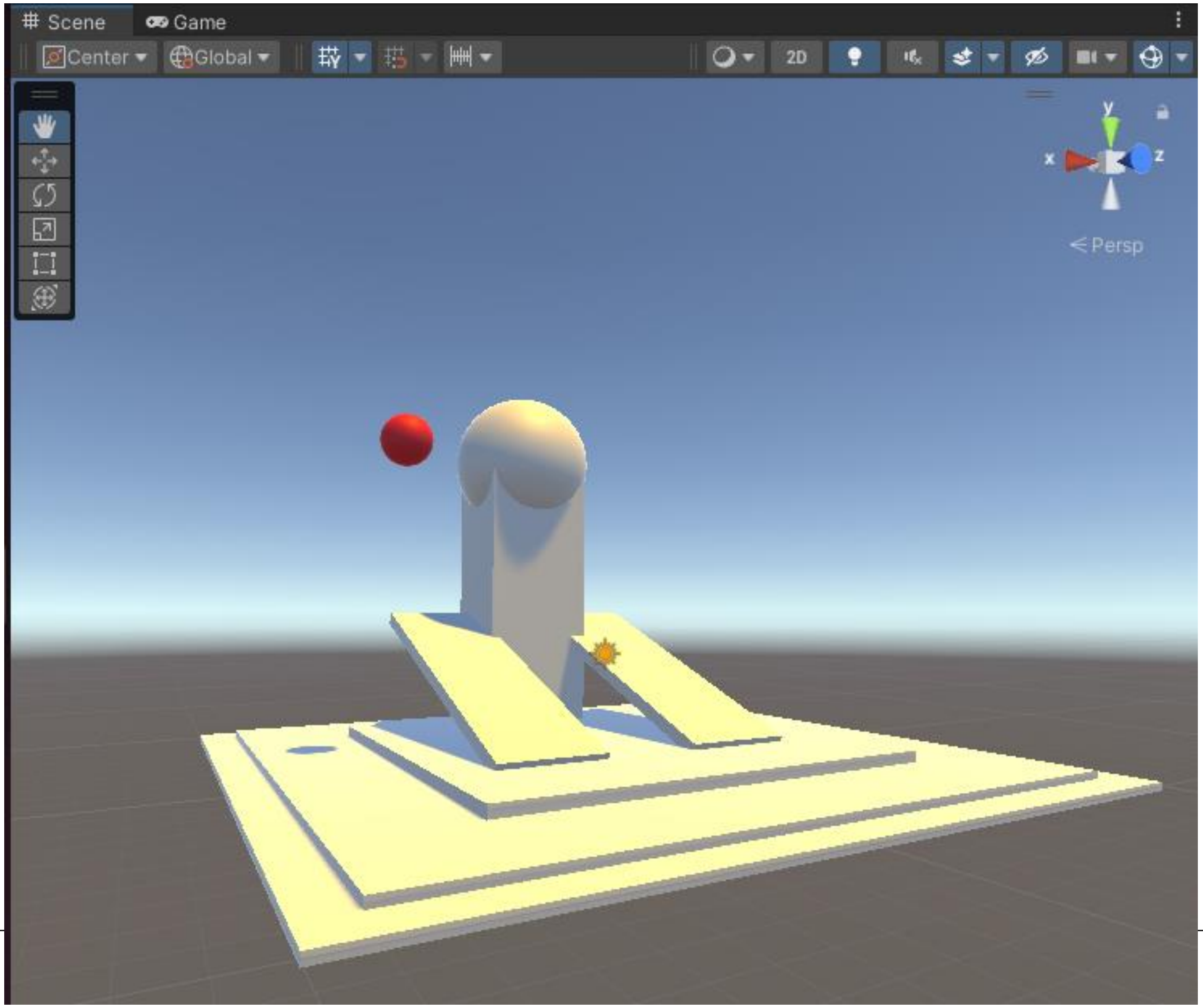
1. In the Inspector, locate the **Albedo** property and change the color using the color picker
2. Drag your material from your Project window to any of the GameObjects in your scene



SolidRed (Material Variant
Shader Standard Edit...
Parent SolidBlue
Rendering Mode Opaque
Albedo 🟥
Metallic 0
Smoothness 0.5
Source Metallic Alpha
Normal Map
Height Map
Occlusion
Detail Mask
Emission
Tiling X 1 Y 1
Offset X 0 Y 0

Add physical properties

9- Materials

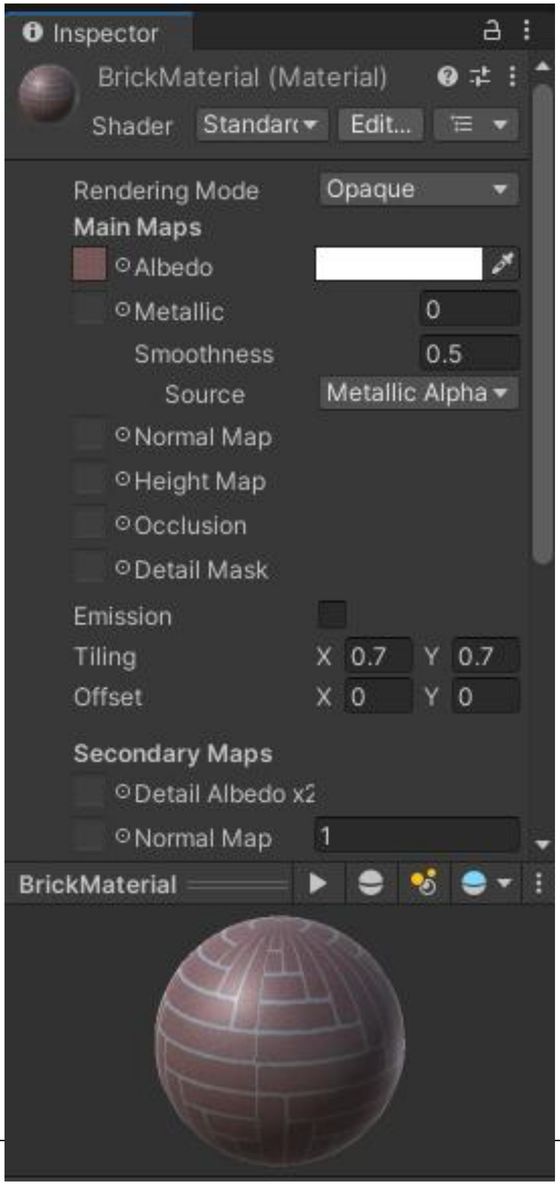


Add physical properties

10- Texture Map Material

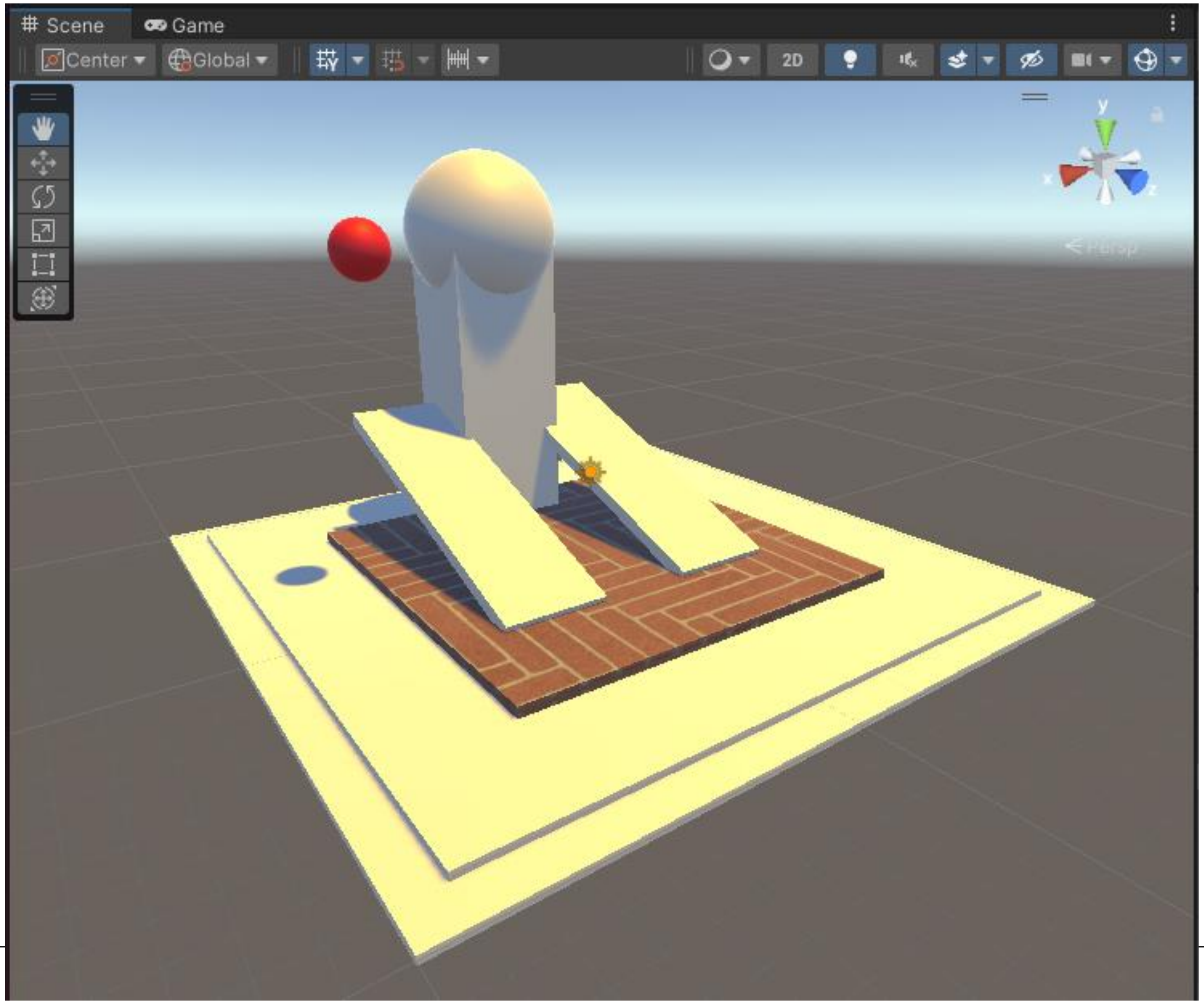
A **texture map** is an image file, such as a PNG or JPG, that you apply to a material

1. Save the **Bricks.png** into the Materials folder
2. Create a new material and give a proper name
3. In the Inspector, select the **circle icon** next to the Albedo property and select the bricks texture file
4. Drag the material to some part of the ground


Inspector
BrickMaterial (Material)
Shader Standard Edit...
Rendering Mode Opaque
Main Maps
Albedo
Metallic 0
Smoothness 0.5
Source Metallic Alpha
Normal Map
Height Map
Occlusion
Detail Mask
Emission
Tiling X 0.7 Y 0.7
Offset X 0 Y 0
Secondary Maps
Detail Albedo x2
Normal Map 1
BrickMaterial

Add physical properties

10- Texture Map Material

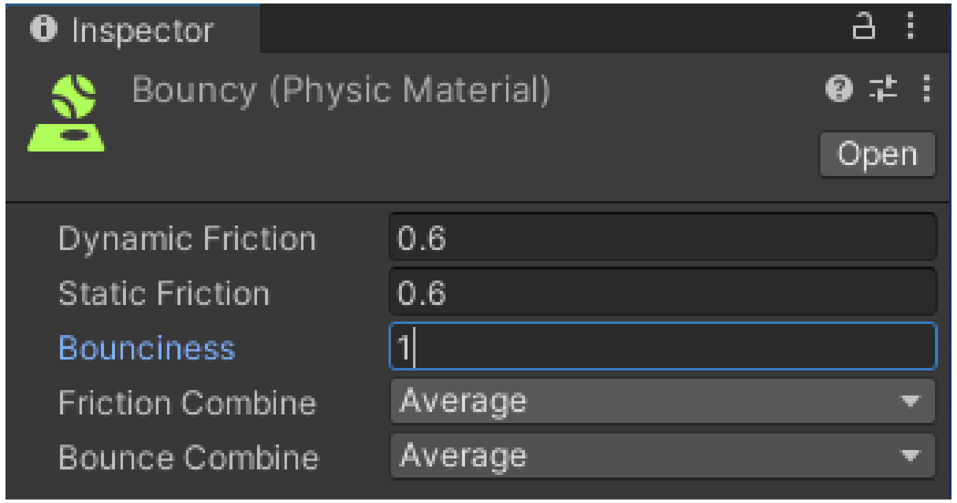


Add physical properties

11- Physic Material

- A **Physic material** is a different type of material that makes an object bounce and changes its friction and drag properties
- These properties take effect when the object is under the effects of gravity

1. Inside the Materials folder, Right-click + **Create > Physic Material**
2. Change its name to "Bouncy" and its **Bounciness** to "1"



Bouncy (Physic Material)

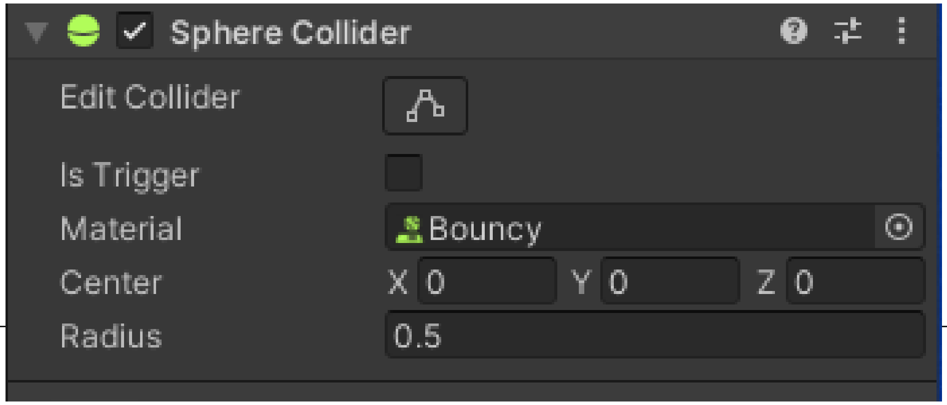
Dynamic Friction	0.6
Static Friction	0.6
Bounciness	1
Friction Combine	Average
Bounce Combine	Average

Add physical properties

11- Physic Material

- In the Inspector window of the sphere, notice the Sphere **Collider component**
- A collider component is automatically included when you create a 3D primitive, such as this sphere

1. Drag the new Bouncy Physic material you created earlier into the **Material property** in the Sphere **Collider component**



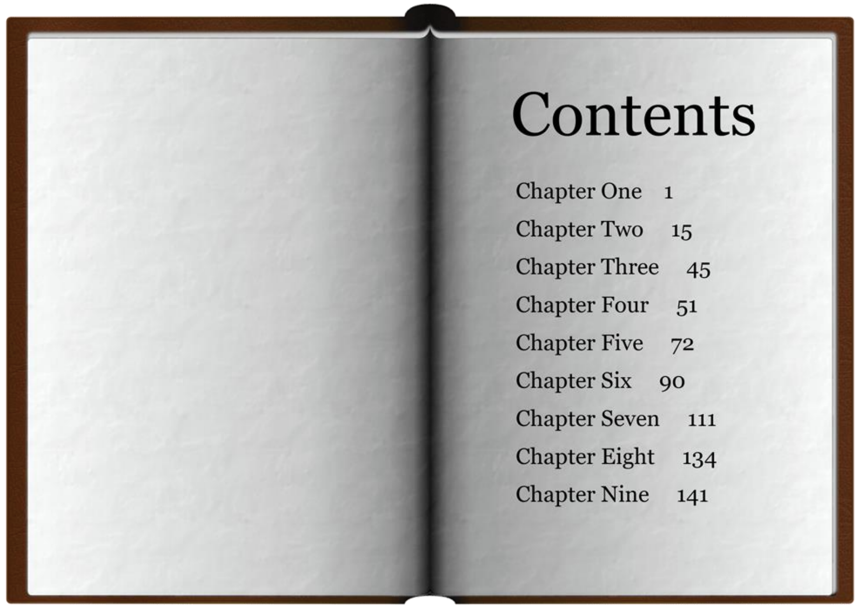
Sphere Collider

Edit Collider	
Is Trigger	
Material	Bouncy
Center	X 0 Y 0 Z 0
Radius	0.5

Create more falling objects of different primitives, set Materials and Physic Materials … and RUN

Content

1. What is a game engine?
2. Unity environment installation
3. Explore the Unity Editor
4. Working with 3D GameObjects
5. Create using primitives
6. Add components
7. Add physical properties
8. Manage GameObjects with prefabs
9. Get 3D assets
10. Publish your project
11. ACTIVITY

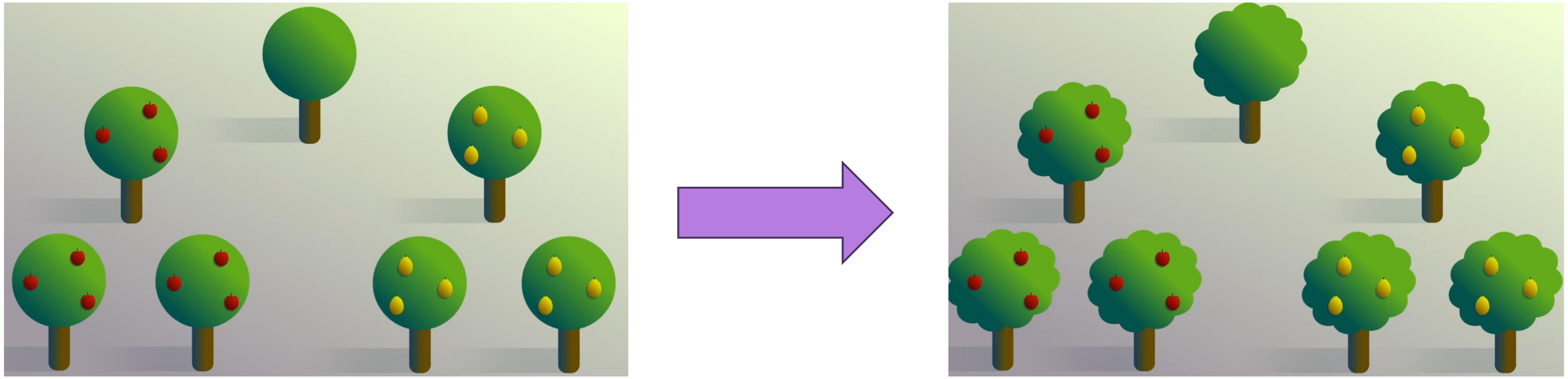


Manage GameObjects with Prefabs

- You’ll typically have many GameObjects to manage in a project
- In many cases your GameObjects will be copies of others:
    - copies of the same enemy character
    - the same trees
    - the same objects to collect
- As you design these GameObjects, you might want to make changes to all copies of one item
- Instead of managing many copies of items, you can organize your duplicated GameObjects using **prefabs**

Manage GameObjects with Prefabs

- A prefab **is an asset that acts as a template** of a GameObject
- From the prefab, you can create multiple copies, called **instances**
- A change to the prefab asset causes all of its instances to change as well

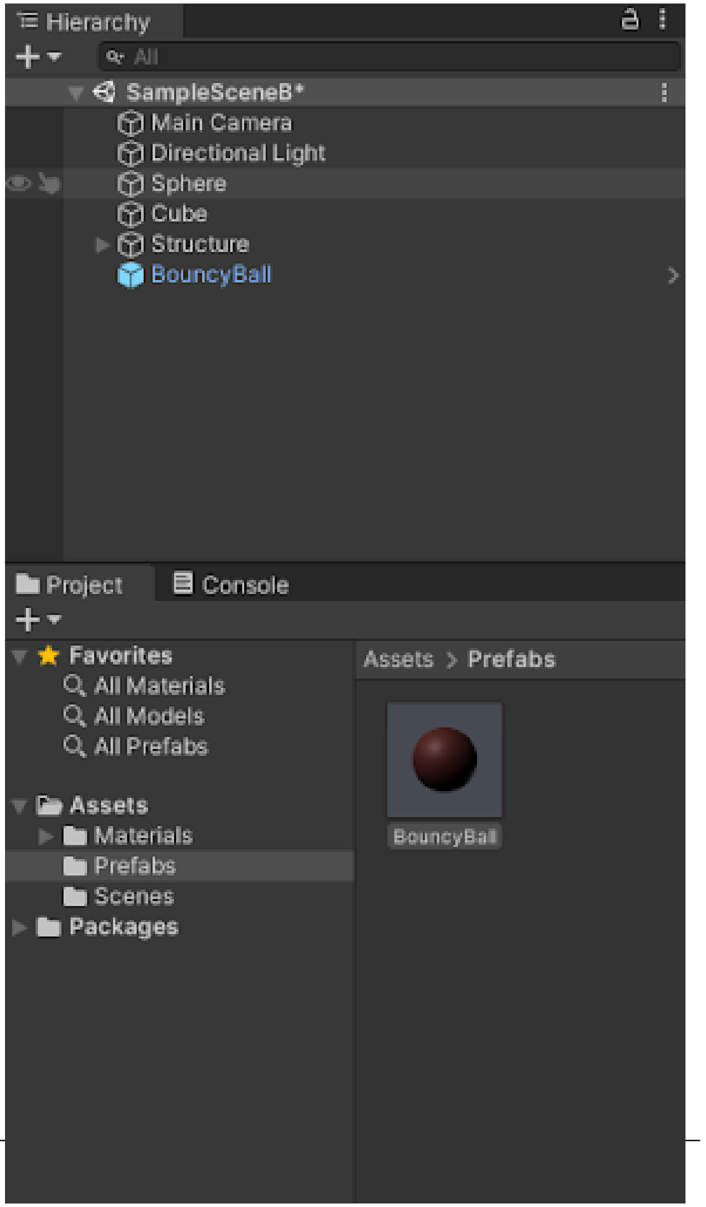


Do you see any relation of this wrt the OOP world?

Manage GameObjects with Prefabs

12- Create a Prefab

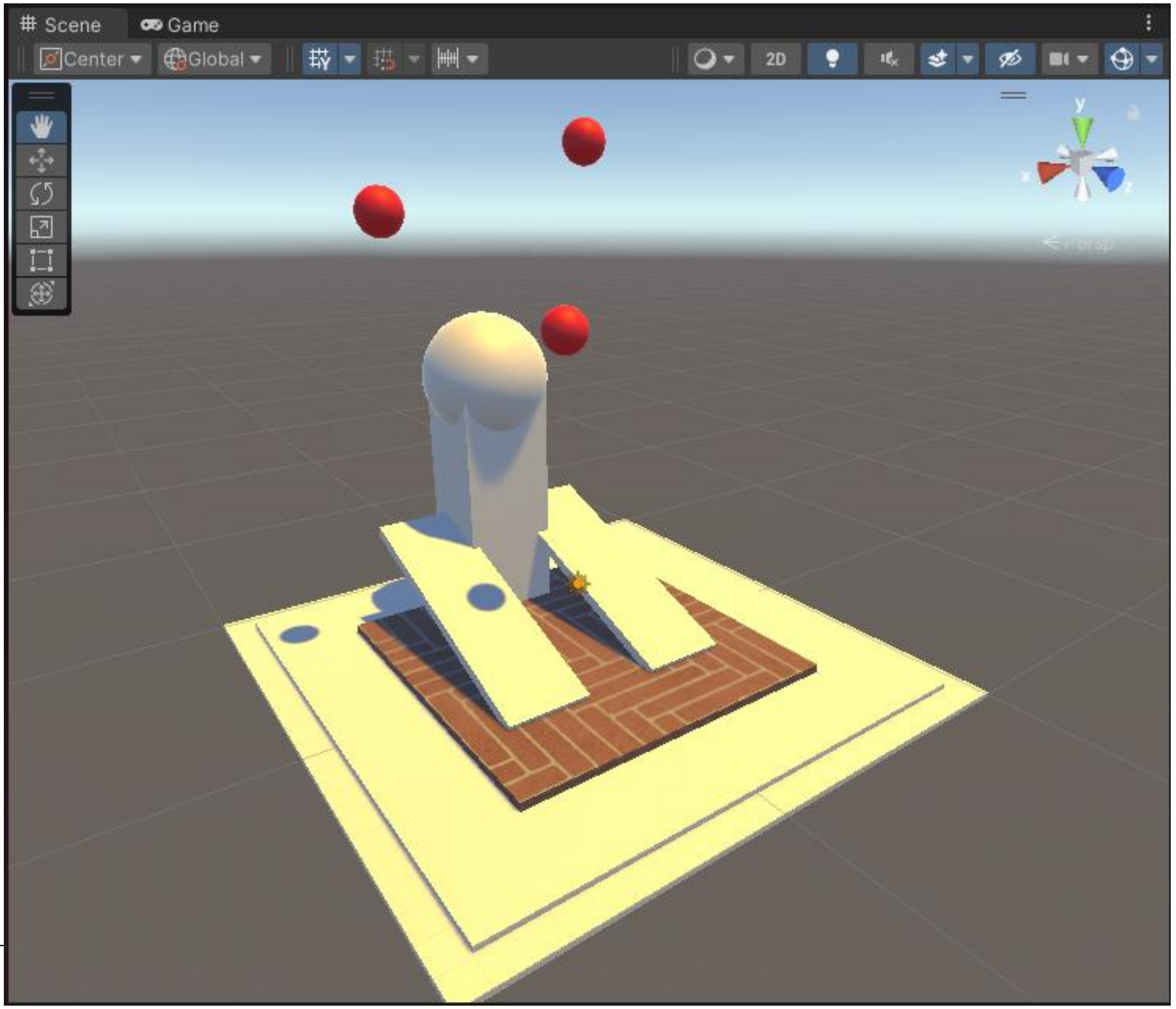
1. Choose a GameObject and make sure it has a regular material and a Physic material
2. Create a Prefabs folder inside your Assests directory
3. Drag your object from the Hierarchy into the Prefabs folder. The new asset in your Prefabs folder is your prefab
4. In the Hierarchy, the GameObject is now blue to let you know that this object is an instance of your prefab



Manage GameObjects with Prefabs

13- Create prefab instances

1. From the Prefabs folder in the Project window, drag some prefabs into your scene
2. For more bouncing, position them higher

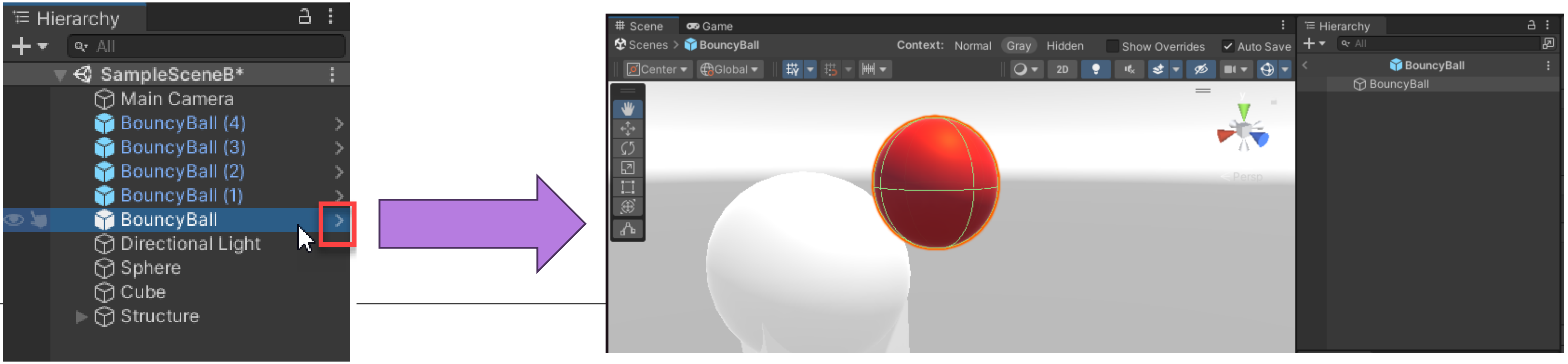


Manage GameObjects with Prefabs

14- Update prefab instances in prefab mode

- You can update properties all at once by updating your prefab in **Prefab Mode**
- It is a special editing state in which you can change a prefab in the Scene window

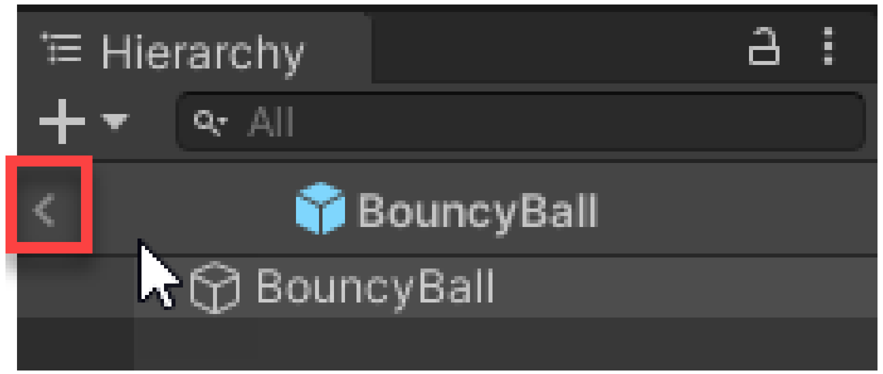
1. Select a prefab instance and then select the arrow on the right



Manage GameObjects with Prefabs

14- Update prefab instances in prefab mode

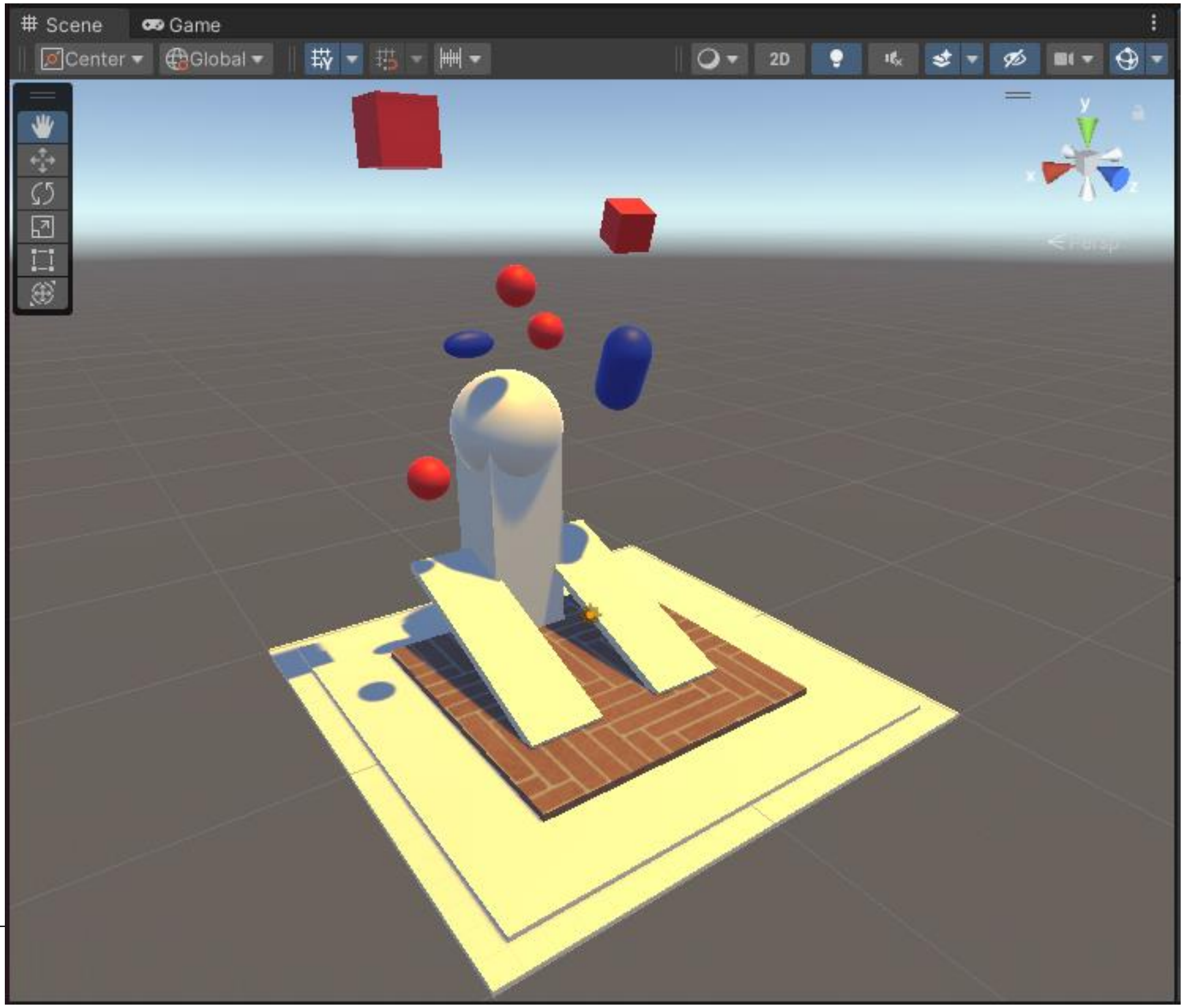
1. In the Project window, open your Materials folder and drag a different material onto the prefab in the Scene window
2. One can also change scale, meshing, collider properties …
3. To exit Prefab Mode, select the arrow at the top left of the Hierarchy window



Manage GameObjects with Prefabs

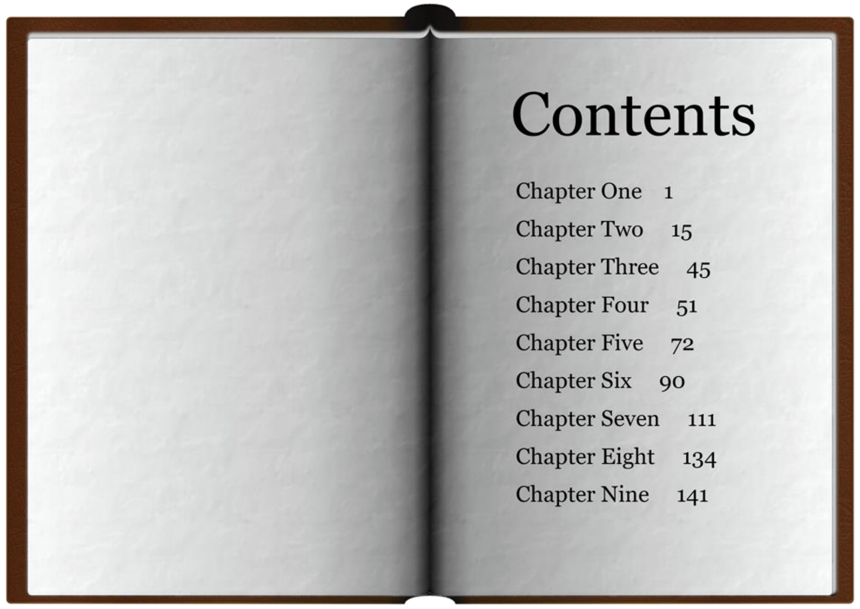
Challenge yourself

1. Make sets of falling objects with different colors (materials), mass, and bounciness, all based on prefabs
2. Make complex shapes with nested prefabs and then change a prefab that you nested



Content

1. What is a game engine?
2. Unity environment installation
3. Explore the Unity Editor
4. Working with 3D GameObjects
5. Create using primitives
6. Add components
7. Add physical properties
8. Manage GameObjects with prefabs
9. Get 3D assets
10. Publish your project
11. ACTIVITY

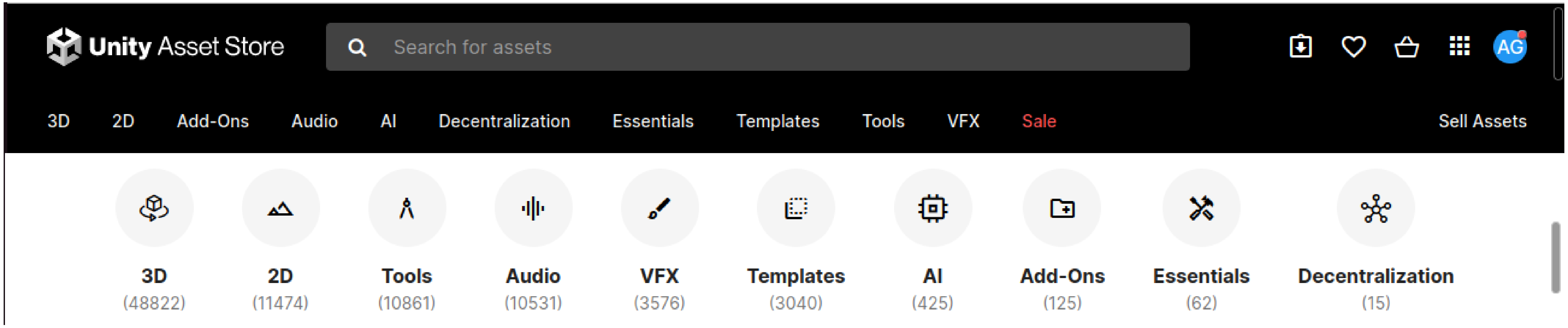


Contents

Chapter One	1
Chapter Two	15
Chapter Three	45
Chapter Four	51
Chapter Five	72
Chapter Six	90
Chapter Seven	111
Chapter Eight	134
Chapter Nine	141

Get 3D assets

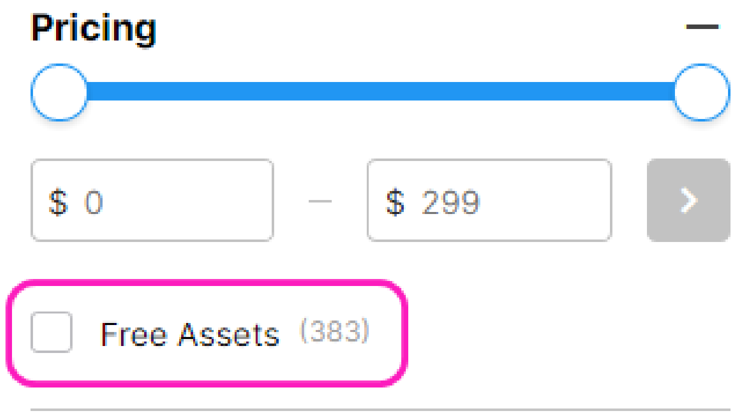
- Not everyone has the time to learn a 3D DCC or the budget to hire someone to create 3D art for them
- But there are many online resources for obtaining high quality assets(CGTrader and Turbosquid)
- The Unity Asset Store is an invaluable resource for art specifically designed with Unity in mind



Get 3D assets

Let's get some stuff

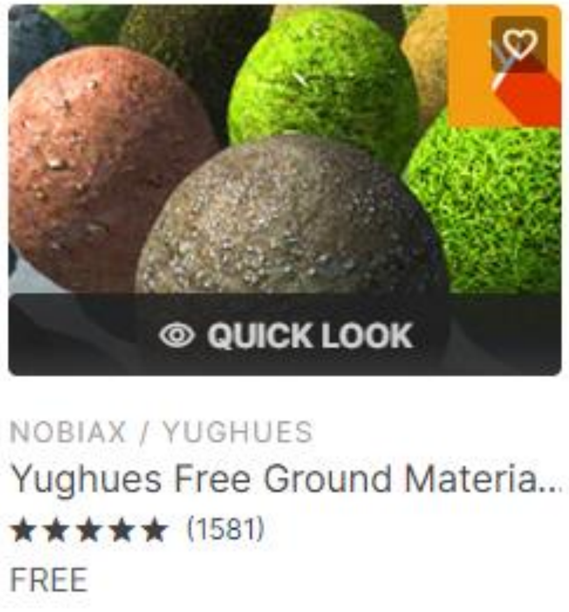
- Go to the Unity Asset store and login with your Unity ID (or create one)
- Anything you acquire through the Asset Store will be linked to your Unity account and will be available in the Editor as long as you are signed in with the same Unity ID
- In the Asset Store search tab, type “materials” to search for available Material assets
- Use the checkbox to view only the free assets



Get 3D assets

Let's get some stuff

Select the **Yughues Free Ground Materials**

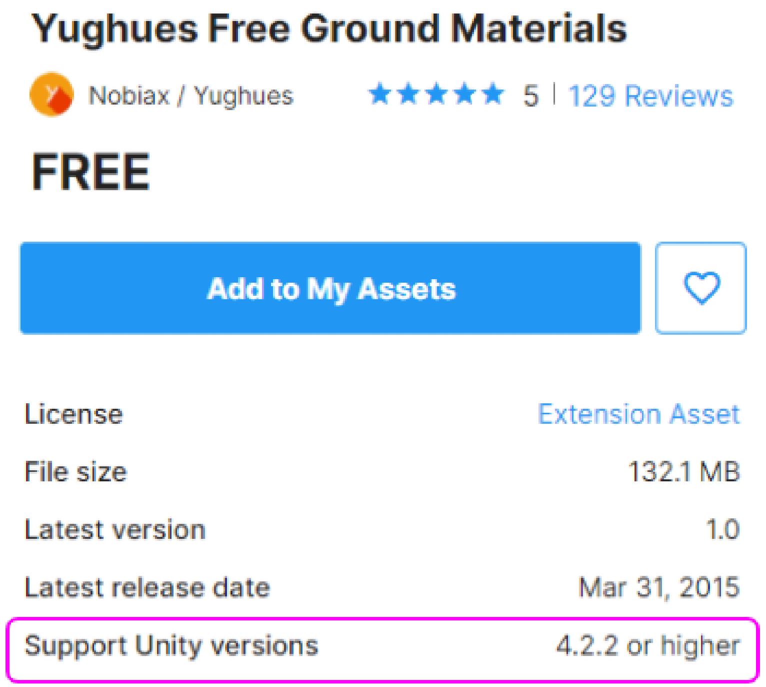


NOBIAX / YUGHUES

Yughues Free Ground Materia...

★★★★★ (1581)

FREE



Yughues Free Ground Materials

Nobiax / Yughues ★★★★★ 5 | 129 Reviews

FREE

License	Extension Asset
File size	132.1 MB
Latest version	1.0
Latest release date	Mar 31, 2015
Support Unity versions	4.2.2 or higher

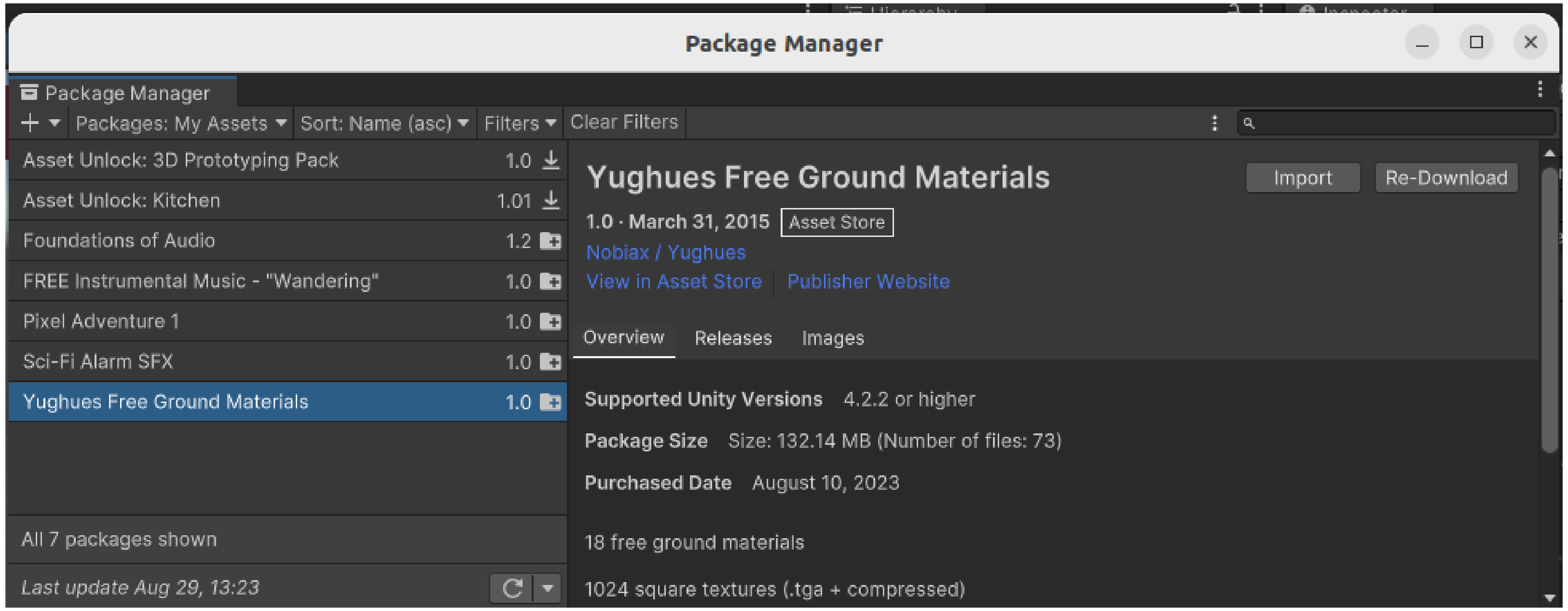
Check the Unity version of the assets you have selected. It is critical to make sure that the assets will be compatible with your version of Unity

Add to My Assets

Get 3D assets

Import assets to Unity

Go to **Window > Packet Manager** and select **Packages: My Assets**



Package Manager

Packages: My Assets | Sort: Name (asc) | Filters | Clear Filters

Asset Unlock: 3D Prototyping Pack	1.0
Asset Unlock: Kitchen	1.01
Foundations of Audio	1.2
FREE Instrumental Music - "Wandering"	1.0
Pixel Adventure 1	1.0
Sci-Fi Alarm SFX	1.0
Yughues Free Ground Materials	1.0

All 7 packages shown

Last update Aug 29, 13:23

Yughues Free Ground Materials — Import | Re-Download

1.0 · March 31, 2015 Asset Store

Nobiax / Yughues

View in Asset Store | Publisher Website

Overview | Releases | Images

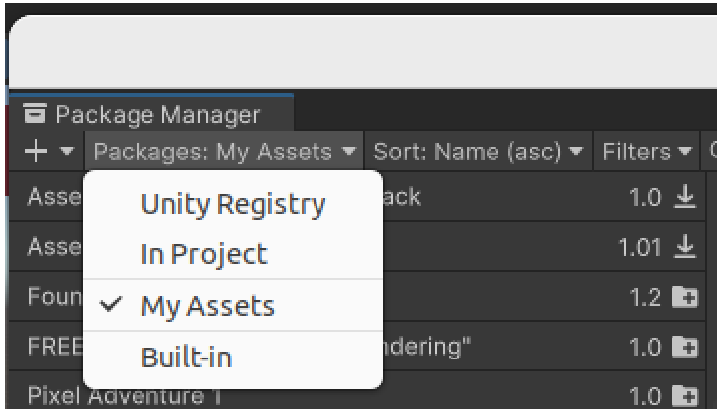
Supported Unity Versions 4.2.2 or higher

Package Size Size: 132.14 MB (Number of files: 73)

Purchased Date August 10, 2023

18 free ground materials

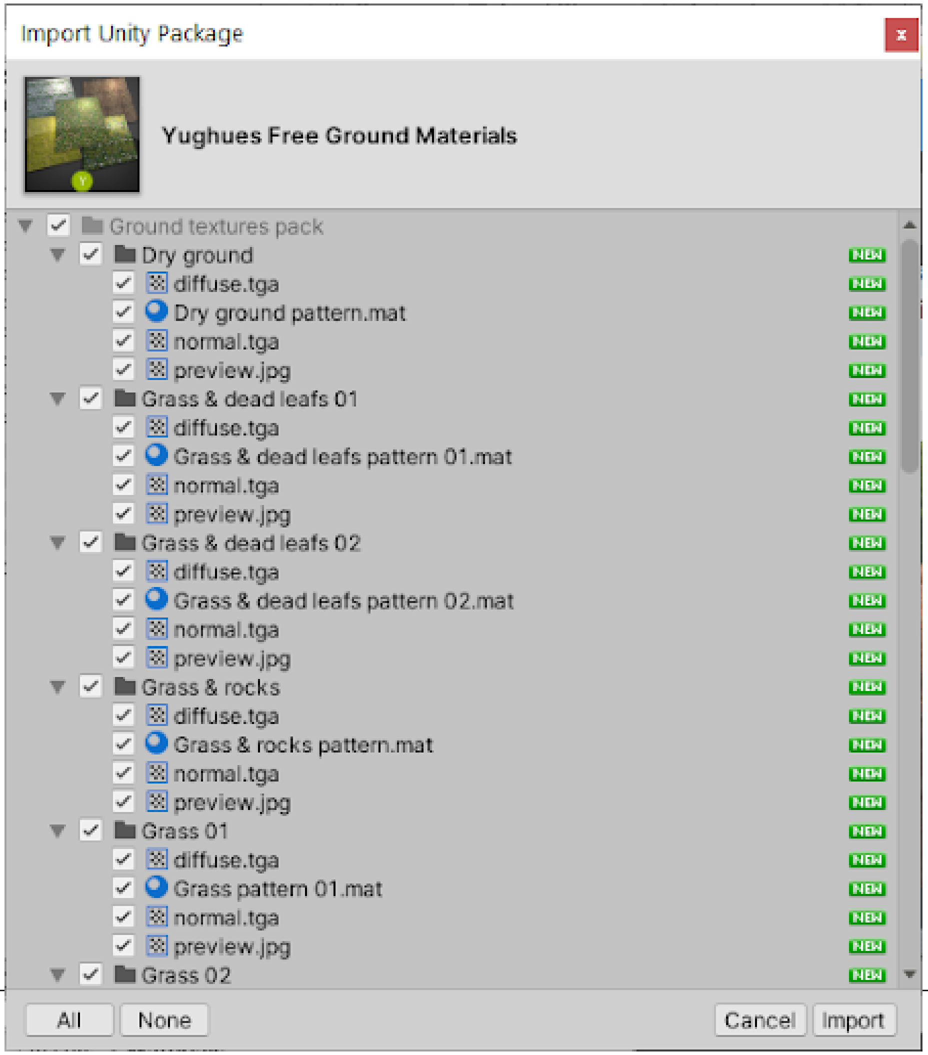
1024 square textures (.tga + compressed)



Download and then **Import**

Get 3D assets

Import assets to Unity



Import Unity Package

Yughues Free Ground Materials

```
▼ ✓ Ground textures pack
  ▼ ✓ Dry ground                                  NEW
      ✓ diffuse.tga                               NEW
      ✓ Dry ground pattern.mat                    NEW
      ✓ normal.tga                                NEW
      ✓ preview.jpg                               NEW
  ▼ ✓ Grass & dead leafs 01                       NEW
      ✓ diffuse.tga                               NEW
      ✓ Grass & dead leafs pattern 01.mat         NEW
      ✓ normal.tga                                NEW
      ✓ preview.jpg                               NEW
  ▼ ✓ Grass & dead leafs 02                       NEW
      ✓ diffuse.tga                               NEW
      ✓ Grass & dead leafs pattern 02.mat         NEW
      ✓ normal.tga                                NEW
      ✓ preview.jpg                               NEW
  ▼ ✓ Grass & rocks                               NEW
      ✓ diffuse.tga                               NEW
      ✓ Grass & rocks pattern.mat                 NEW
      ✓ normal.tga                                NEW
      ✓ preview.jpg                               NEW
  ▼ ✓ Grass 01                                    NEW
      ✓ diffuse.tga                               NEW
      ✓ Grass pattern 01.mat                      NEW
      ✓ normal.tga                                NEW
      ✓ preview.jpg                               NEW
  ▼ ✓ Grass 02                                    NEW
```

The package will be extracted and placed in your project's Assets folder

Surf the different materials and apply them on some of your GameObjects

Get 3D assets

IMPORTANT: Before importing assets

- **Very detailed assets** (characters, scenes, templates, materials …) will increase your project weight and, what it is worse, **will make your game very very very slow** (even "unplayable")
- Professional developers have huge machines with pro graphical cards … that unfortunately we don't have
- Please **check the technical details of the assets** in the Assests Store before adding them to your project
- **Better use Minecraft-style characters** ("pixelated") than super detailed ones

Get 3D assets

IMPORTANT: Before importing assets

Adventure Character

Overview   Package Content

Adventure Character

PBR textures.

-Albedo 7
-Ao 3
-Metallic 3
-Normal 3
(all 4096-4096 size)

Polys: 119,538
Tris: 236,200
Verts: 118,900

Quirky Series - FREE Animals Pack

Overview   Package Content   Releases   R

Features

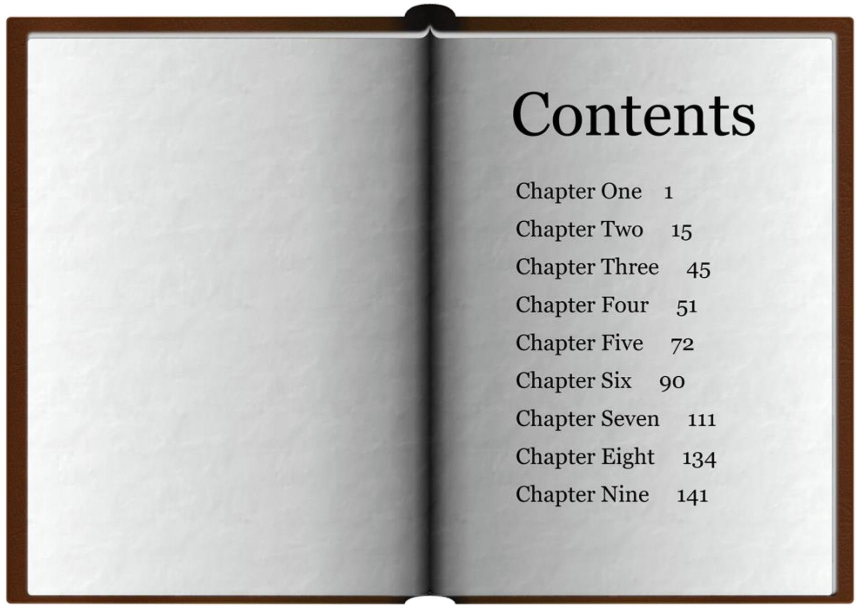
✅ Eight (8) animal
✅ Tiny 16×4 px texture [diffuse map only]
✅ Rigged/Skeleton
✅ 18 animations
✅ 4 Levels of Detail [min 300 up to 9k tris]
✅ Mobile, AR/VR ready
✅ Sample URP Shader included
❌ Vertex color
❌ Clean (non-overlapping) UV mapping

Get 3D assets

IMPORTANT: Before importing assets

Content

1. What is a game engine?
2. Unity environment installation
3. Explore the Unity Editor
4. Working with 3D GameObjects
5. Create using primitives
6. Add components
7. Add physical properties
8. Manage GameObjects with prefabs
9. Get 3D assets
10. Publish your project
11. ACTIVITY



Contents

Chapter One	1
Chapter Two	15
Chapter Three	45
Chapter Four	51
Chapter Five	72
Chapter Six	90
Chapter Seven	111
Chapter Eight	134
Chapter Nine	141

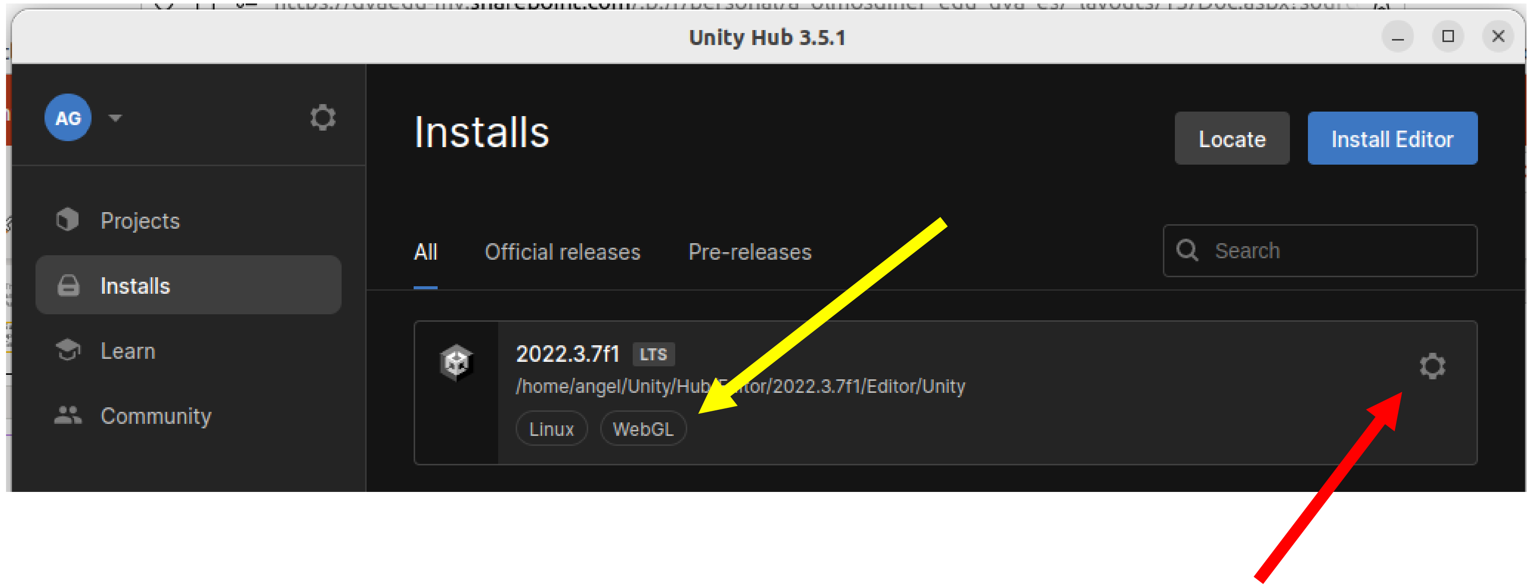
Publish your project

- When your game is ready, it is time to create a **build**, which is a standalone, playable version of your game
- Unity supports most popular platforms, including mobile (Android and iOS)
- The game developer must **make sure that the game can run on the target platforms** --> the resources available on a smartphone are vastly different than those on a Playstation
- Here we'll publish to **WebGL** (HTML5) and then share it on the web with the Foundations community

Publish your project

What do we need to publish to WebGL?

1- The WebGL Build Support module added to your Unity installation



Unity Hub 3.5.1

Installs

2022.3.7f1 LTS

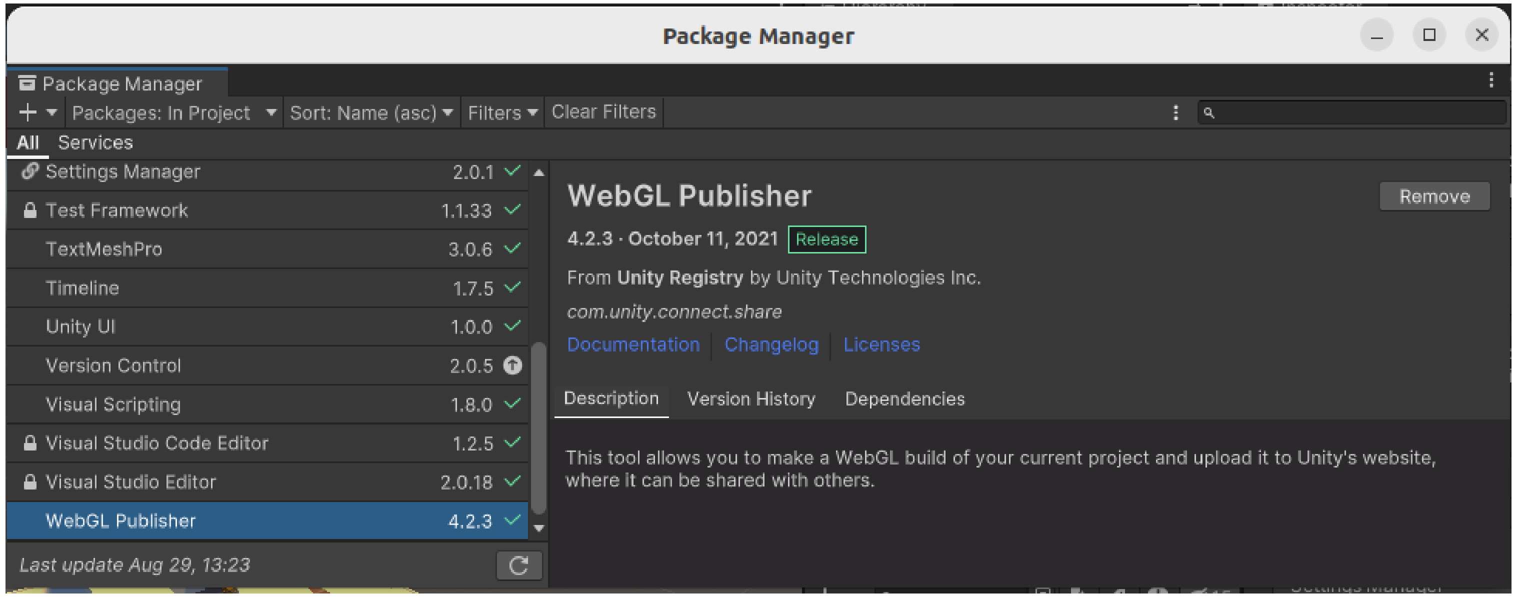
/home/angel/Unity/Hub/Editor/2022.3.7f1/Editor/Unity

If the WebGL module is not there, add it

Publish your project

What do we need to publish to WebGL?

2- The WebGL Publisher package installed via the **Package Manager**



Settings Manager	2.0.1 ✓
Test Framework	1.1.33 ✓
TextMeshPro	3.0.6 ✓
Timeline	1.7.5 ✓
Unity UI	1.0.0 ✓
Version Control	2.0.5
Visual Scripting	1.8.0 ✓
Visual Studio Code Editor	1.2.5 ✓
Visual Studio Editor	2.0.18 ✓
WebGL Publisher	4.2.3 ✓

Last update Aug 29, 13:23

WebGL Publisher

4.2.3 · October 11, 2021 Release

From **Unity Registry** by Unity Technologies Inc.

com.unity.connect.share

Documentation | Changelog | Licenses

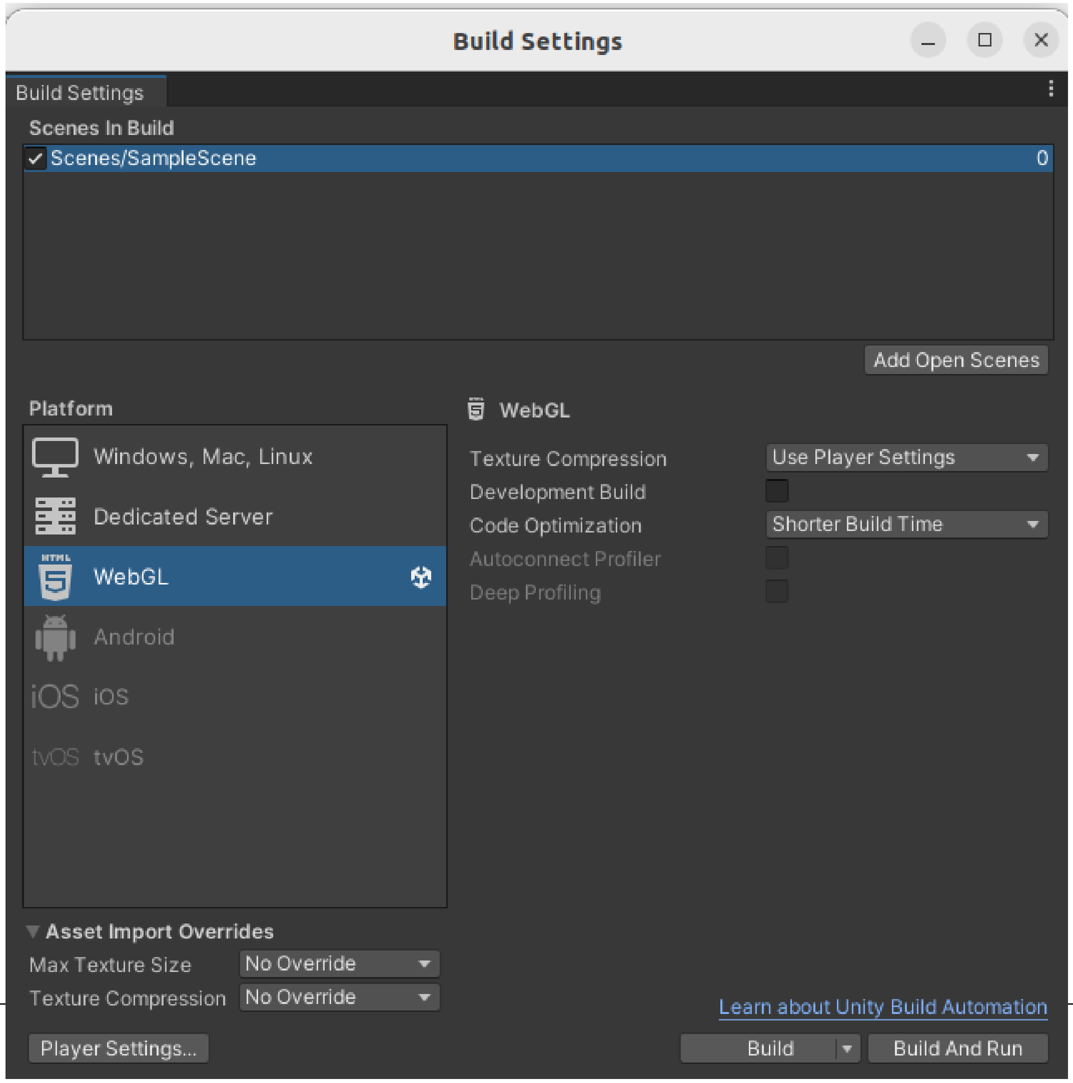
This tool allows you to make a WebGL build of your current project and upload it to Unity's website, where it can be shared with others.

If the WebGL package is not there, **install it**

Publish your project

Setup the Build

1. Open the Build Settings by **File > Build Settings**
2. Select the **Add Open Scenes** button to set up the current Scene as your starting Scene for the build
3. Select **WebGL** from the list on the left



Build Settings

Build Settings

Scenes In Build

✓ Scenes/SampleScene 0

Platform

WebGL

Texture Compression	Use Player Settings
Development Build	
Code Optimization	Shorter Build Time
Autoconnect Profiler	
Deep Profiling	

Asset Import Overrides

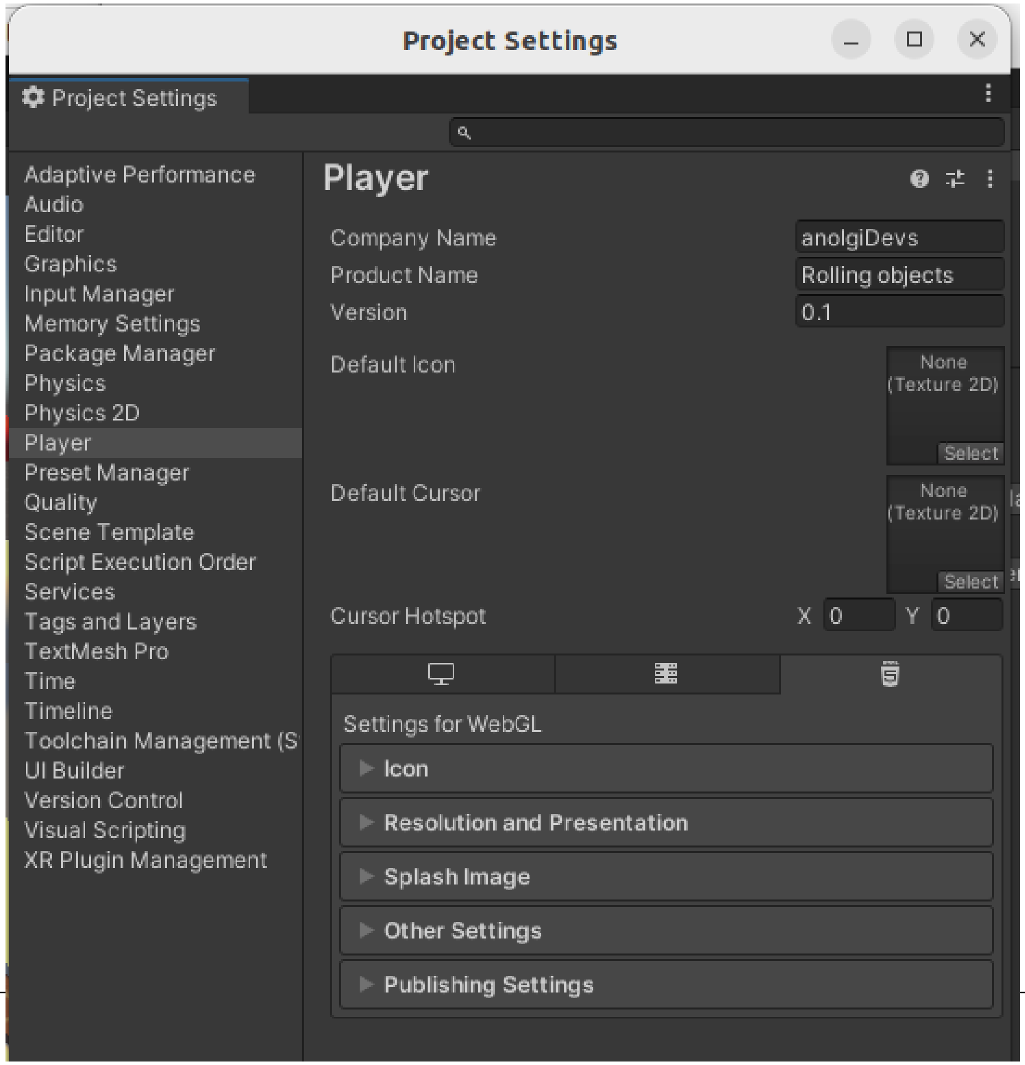
Max Texture Size	No Override
Texture Compression	No Override

Learn about Unity Build Automation

Publish your project

Setup the Build

1. Select the **Player Settings** button
2. Change the **Company Name** to a name of your choice, and enter a **Product Name**
3. Close the player settings window



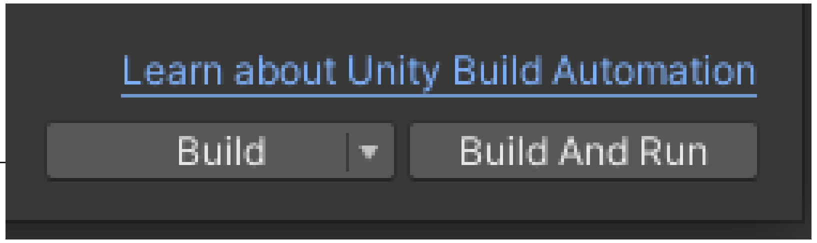
Company Name: anolgiDevs
Product Name: Rolling objects
Version: 0.1
Default Icon
Default Cursor
Cursor Hotspot X 0 Y 0

Publish your project

Setup the Build

1. Depending on your settings, you might see a **Switch Platform** button at the bottom. If you do, select it, and it will change to the Build button
2. Select the **Build** button to start the build process
3. When prompted, select a folder where you would like to save the game.

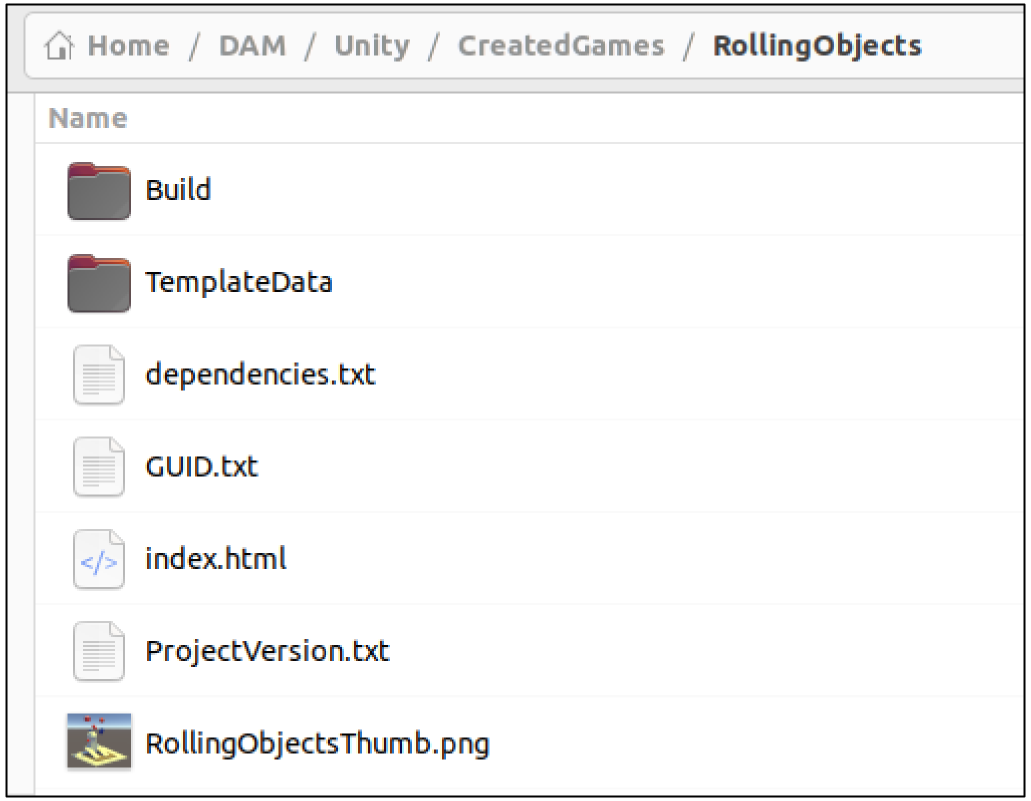
<u>Note:</u> Do not save it in the same folder as your project



Publish your project

Setup the Build

- Unity will create an *index.html* file, as well as other folders, files and assets that the game will need to run
- The files are dependent on each other and should not be moved or changed
- If you need to move the game to a different location, make sure to move the index.html file and all the folders together

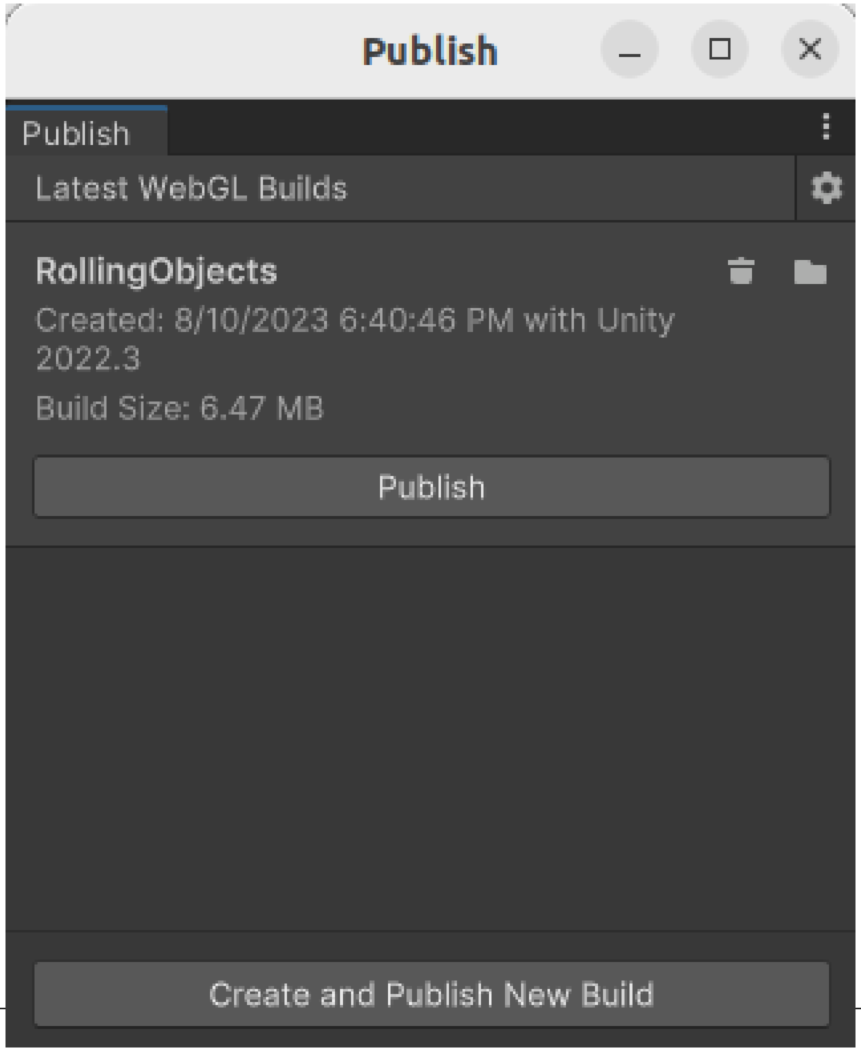

Home / DAM / Unity / CreatedGames / RollingObjects

Name
Build
TemplateData
dependencies.txt
GUID.txt
index.html
ProjectVersion.txt
RollingObjectsThumb.png

Publish your project

Publish your project

- We will publish the project on Unity Play, and provide a shareable link to your project
- From the main menu, select **Publish > WebGL Project**
- Locate, add the build and **Publish** it
- Once the publishing is done, copy and share the link

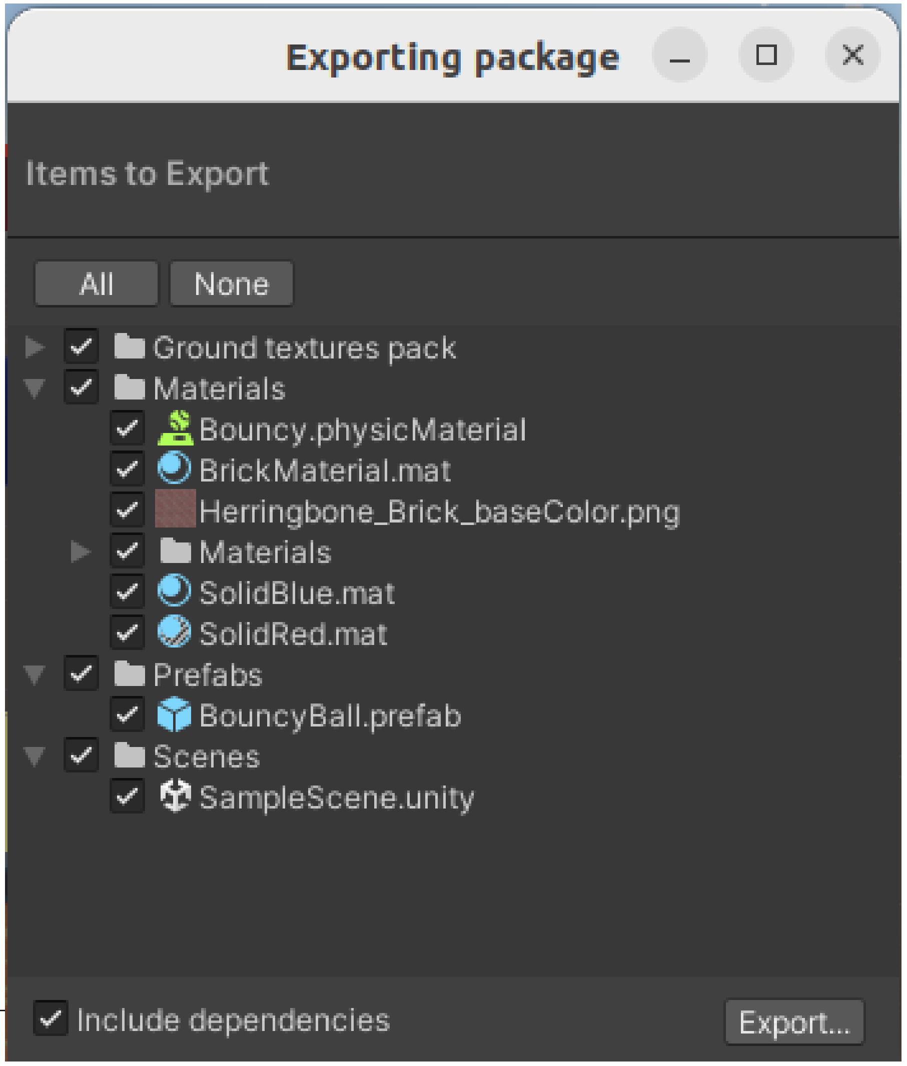


RollingObjects
Created: 8/10/2023 6:40:46 PM with Unity 2022.3
Build Size: 6.47 MB

Publish your project

Export your project as a Unity package

- Make sure the **Assets folders** is selected in the Project window
- Go to **Assets > Export package ...**
- Make sure all prefabs, materials and scenes are selected and export to a folder outside the Unity project folder
- The exported file can be then imported to a new project through **Assets > Import package > Custom package ...**



Exporting package

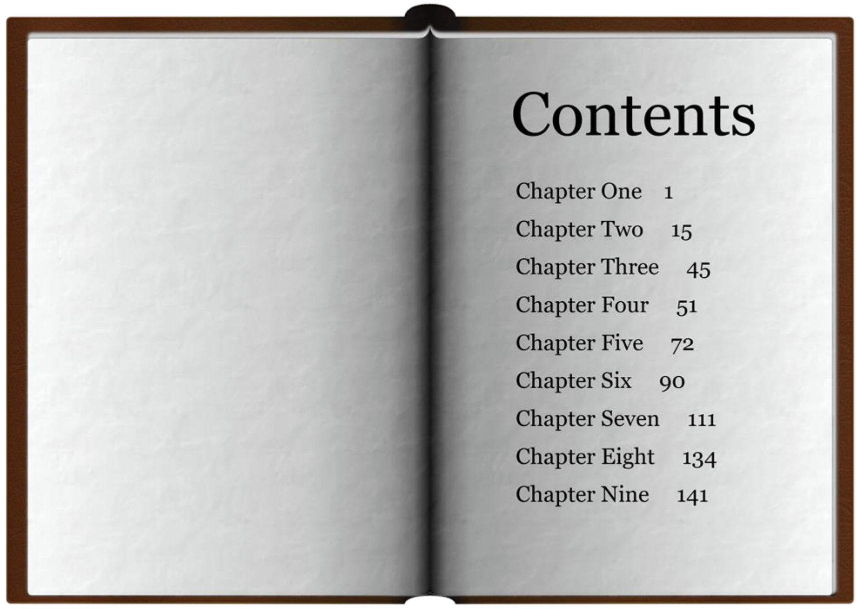
Items to Export

- ✓ Ground textures pack
- ✓ Materials
  - ✓ Bouncy.physicMaterial
  - ✓ BrickMaterial.mat
  - ✓ Herringbone_Brick_baseColor.png
  - ✓ Materials
  - ✓ SolidBlue.mat
  - ✓ SolidRed.mat
- ✓ Prefabs
  - ✓ BouncyBall.prefab
- ✓ Scenes
  - ✓ SampleScene.unity

✓ Include dependencies

Content

1. What is a game engine?
2. Unity environment installation
3. Explore the Unity Editor
4. Working with 3D GameObjects
5. Create using primitives
6. Add components
7. Add physical properties
8. Manage GameObjects with prefabs
9. Get 3D assets
10. Publish your project
11. ACTIVITY



ACTIVITY

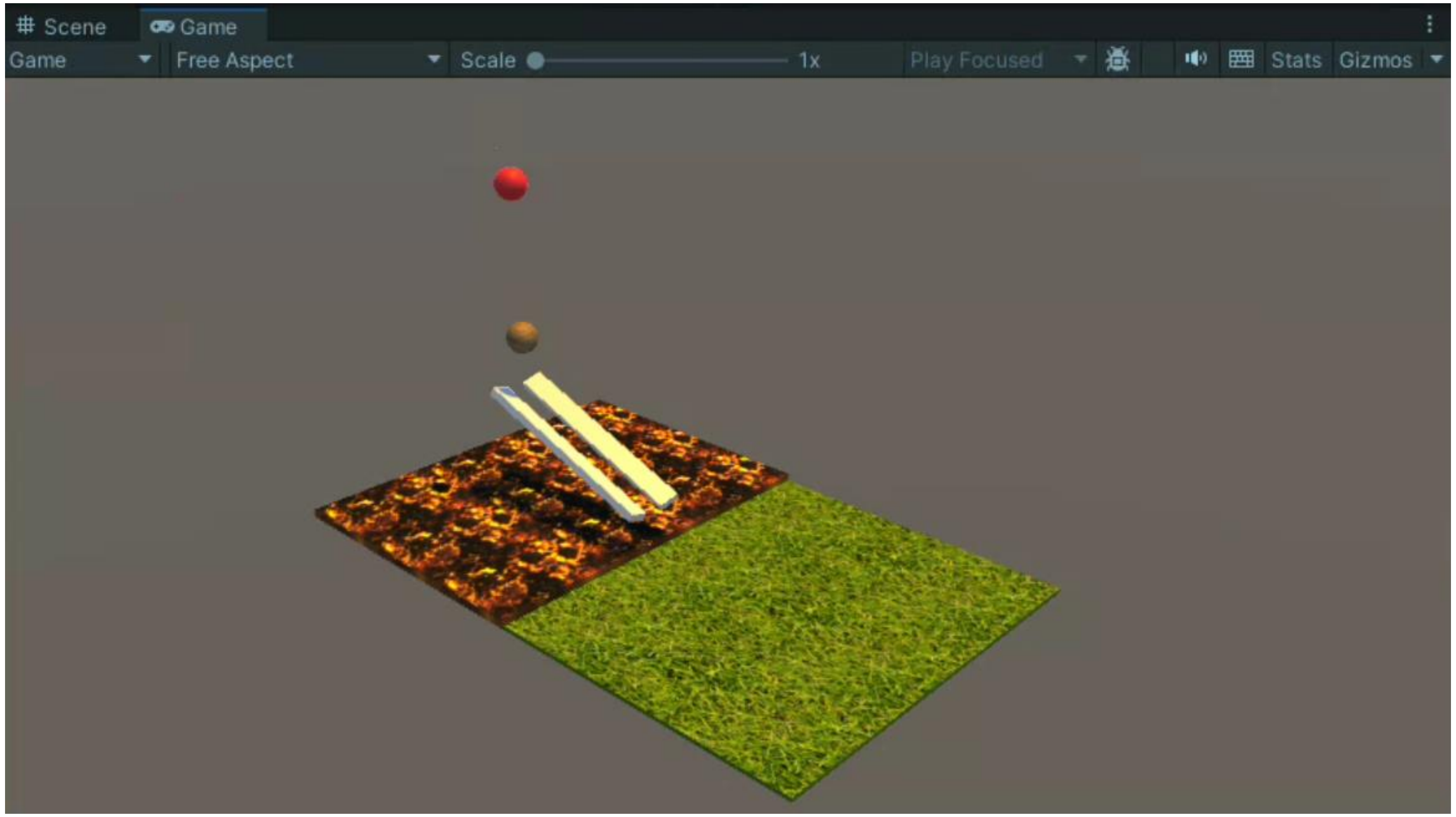
ROCK that ROLLS

Create a Unity project that accomplishes the following requirements:

1. There must be a ground structure with a texture material made by the combination of at least 2 primitives
2. A second structure, child of the ground, must include a rolling path for a falling ball
3. Create a ball prefab with bouncy characteristics
4. Instantiate a ball from the prefab and give a texture material to it
5. Place the ball above the path. The ball will fall and must continue along the path without falling to the ground
6. Instantiate a second ball from the prefab with a different material and place it above the first one
7. It doesn't matter if the second ball falls to the ground
8. Export the project as a Unity package

ACTIVITY

RESULT VIDEO – RockThatRolls



LICENSE