**Unity Tutorial**

1. **Unity User Interface**

- Scene View (see everything that makes up the view)

- Game View (whatever the player should see)

- Hierarcy (on the left pane, showing the inventory list of **everything in your scene**)

- Main Camera

- Directional Light

- Player

- Environment

- Tile 1

- Tile 2…

🡪 Click to select in the scene view

In more complex games, we might want to dynamically generate and then delete the objects.

- Project Window (On the bottom) default view is 2 column layout. This is where the objects are stored (in the warehouse)

- Inspector (On the right): To change properties of the objects

1. **Game Objects, Transforms, and Components**

- Everything in Unity is a Game Object (Camera, Light, Player, etc.)

🡪 A collection of a transform and components

- A transform is a Position, Rotation, and Scale of a game object

- Components: a mesh filter tells the graphic card which type of mesh it should be rendering (3D objects)

- Mesh render: how to render its mesh (shadows, materials, etc.) of a object

- Box collider: how the object reacts when it's collided.

\*\*Avoid changing the scale, from Blender

**Hotkeys**

**x is left/right y is up/down z is back/forth**

**Hold alt & click left mouse** 🡪 Rotate around

**q** view tool **e** rotate tool **w** move tool **r** scale tool

**F2**: rename object

**Shift + F**: frame scene view to object

**Ctrl + D**: A copy of the components (at the exact position). Can also be applied to materials

**Hold Ctrl while moving** 🡪 snap movements to 0.25 increments

**Hold Alt** while expanding a Prefab object in hierarcy 🡪 Expanding ALL child objects.

**Hierarcy**:

The child follows the parent if the parent moves, but the parent will not move if child moves.

The child will have relative position to the **parent object** (offsets)

**Prefabs**

These are like **templates**. It's recommended to use these while creating objects.

We can make an object an instance of a prefab by dragging it into a Prefab folder. This makes the object to become a template, which can be used to create more instances (by dragging to the screen/or Ctrl-D)

Instances of a prefab can be put together inside an Empty Game Object (in Hierarcy). **It is recommended that we keep the empty game object position as 0,0,0** because this will help with the positioning (offset) of child object instances.

It is also recommended that for objects we do not need to care about collision, we should delete the box collider (to optimize running time)

🡪 We can modify all instances of a prefab if we modify one in the **prefab editing mode**, enter by clicking into the Prefab in the folder

We can overwrite a prefab by clicking on an object separately.

**Create a Player**

We can make a player prototype by using a Capsule 3D object

- We apply physics to the player by selecting the player, and then choose **Add Component -> Rigidbody**

**User Input**

Create a Folder in the root directory, create a **Scripts** folder.

- Create a C# script

- The name of the class (inside the script) must match the name of the C# script (and should match the name of the Unity object)

- Click and drag the script to the object in Unity, to "connect" the script to that object.

- Debug.log("") 🡪 Log to console (used to debug)

- **Monobehaviour** is Unity predefined class (that covers most features you need) that contains a bunch of predefined methods.

To make the Player object move/jump, we need to access the Rigidbody component within C# code

GetComponent<Rigidbody>().*Dosomething*;

e.g., For player to jump (regardless of weight)

GetComponent<Rigidbody>().AddForce(Vector3.up \* 5, ForceMode.VelocityChange);

We add the function FixedUpdate. By default, Unity runs physic update at 100hz (100 times per second), regardless of the computer 🡪 proper physics.

Therefore, **all keypresses should be in the Update() loop, but all physics should be in FixedUpdate()**.

Text

Description automatically generated

**Make Player Move**