# Notes

These instructions are meant as a guide, and may be modified as desired or required. You may run this at your own pace. These are intended for instructors who know the concepts already and assume that the content provided in the slides associated with this document has also been presented.

**Code Blocks are in the Appendix and are numbered for reference.**

# UI Walkthrough

Begin by resetting your layout to the default:

Window > Layouts > Default

Walk through each section of the UI:

* Hierarchy
  + GameObject create, select
* Inspector
  + Edit transform for an object
  + Show camera as example of components and their editors
* Scene
  + Use tools to move the Main camera object around
  + MMB for Pan
  + RMB for Pivot (rotate)
  + LMB for select and drag select
  + WSAD for movement
* Game
  + Triggered on pressing play
* Project
  + Filters
  + Create options
  + Searching
* Console
* Menus
* Main/Top Toolbar
  + Hand, Translate, Rotate, Scale
  + Global/Local Transform Tools
  + Play/Pause

Demo that the UI can be dragged around to adjust as desired and saved in the Window > Layouts menu, or using the top rightmost drop-down (which should display Default at the moment unless customized).

# Demo #1

1. Create a Plane
   * Pos 0,0,0
   * Scale 10,10,10
   * Name “Ground”
2. Add a directional light
   * 0,0,0
3. Add a sphere
   * Name “PlayerSphere”
   * Position it above the ground, in front of the camera
4. Select the light and change shadow type to Hard Shadows
5. Add a folder to assets named Materials
6. Add a material to that folder named GroundMat
   * Colour 73, 206, 129
7. Add a new material named PlayerMat
   * Colour 206, 82, 73
8. Drag material onto sphere in scene view
9. Select the player sphere and add a rigid body component
10. Edit > Project Settings > Input
11. Expand Axes, change size to 3
    * Show and explain Horizontal/Vertical options
12. Change 3rd item to Jump
    * Change buttons, set a single button to space
    * Clear other options
13. Add a scripts folder
14. Add a C# script named PlayerMovement
15. Assets > Sync MonoDevelop project
16. Open PlayerMovement.cs in the generated solution
17. At the top of the script add “public float Power =100;”
18. Inside Update() add Code Block 1
19. In the editor drag the script onto PlayerSphere
20. Change Angular Drag of sphere to 1.0
21. Add “public float JumpPower = 300;” to the top of PlayerMovement
22. Add Code Block 2 to Update()
23. Create a new Physics Material in Assets, name it BouncyMaterial
24. Drag BouncyMaterial onto the ground plane
25. Create a new material in the Materials folder named CoinMaterial
    * Specular shader
    * Colour 225, 210, 12
    * Specular Colour 225, 225, 225
    * Increase shininess to a little over 50%
26. Create a Capsule, name it Coin
    * Scale 0.5, 0.5, 0.5
    * Position 0, 0.75, 0
27. Drag CoinMaterial onto Coin
28. Drag Coin from Hierarchy into Assets to create a prefab
29. Move the one in the scene in front of the sphere (between cam and player)
30. Create a new script named CoinPickup
31. Add this script to the coin and click Apply
32. Add to the top of CoinPickup: “public float RotationSpeed = 50;”
    * Tweak as desired
33. Add Code Block 3 to Update()
34. Add Code Block 4 below Update()
35. Add a new script named FollowPlayer and attach to Main Camera
36. Add to the top of FollowPlayer: “public GameObject Player”
37. In the editor, drag the player into the inspector for that field
38. Add Code Block 5 under our Player field
39. Add Code Block 6 to Update()
40. Add some more coins based on the prefab around the area
    * When dragging onto the plane, it will go to Y=0.5 (snap to plane)
41. Add some more primitives in as obstacles, colour with new materials as desired

# Code Blocks

## 1

if (Input.GetAxis("Vertical") > 0)

{

var forward = transform.position - Camera.main.transform.position;

forward.y = 0;

forward.Normalize();

rigidbody.AddForce(forward \* (Power \* Time.deltaTime));

}

## 2

else if (Input.GetAxis("Vertical") < 0)

{

var backward = Camera.main.transform.position - transform.position;

backward.y = 0;

backward.Normalize();

rigidbody.AddForce(backward \* (Power \* Time.deltaTime));

}

if (Input.GetButtonDown("Jump"))

{

rigidbody.AddForce(Vector3.up \* JumpPower);

}

## 3

transform.Rotate(Vector3.forward, Time.deltaTime \* RotationSpeed);

## 4

void OnCollisionEnter(Collision c)

{

if (c.gameObject.GetComponent<PlayerMovement>() != null)

{

// This script is only on the player

Destroy(gameObject);

}

}

## 5

public float RotationSpeed = 50;

private float \_rotation;

## 6

transform.position = Player.transform.position + new Vector3(0, 5, -8);

transform.LookAt(Player.transform);

transform.RotateAround(Player.transform.position, Vector3.up, \_rotation);

if (Input.GetAxis("Horizontal") > 0)

{

// Right

\_rotation += Time.deltaTime \* RotationSpeed;

}

else if (Input.GetAxis("Horizontal") < 0)

{

// Left

\_rotation -= Time.deltaTime \* RotationSpeed;

}