



LABORATORIO UNITY3D

#TecHeroes loves #GameDev Tour 2016

Step-by-Step: Come creare una versione 3D del classico Arkanoid con Unity 5

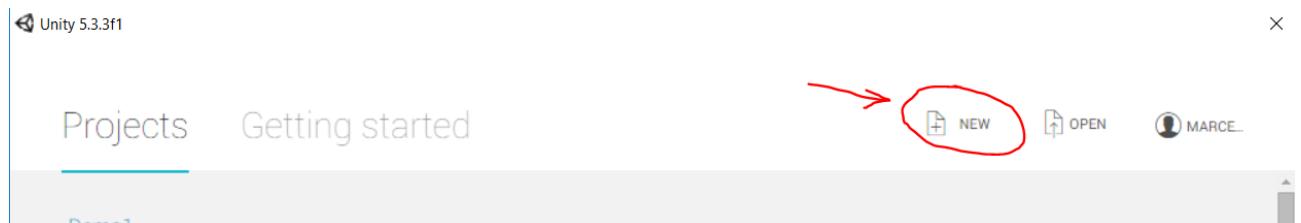
Marcello Marchetti
mamarc@microsoft.com
@marcello_twit

Contents

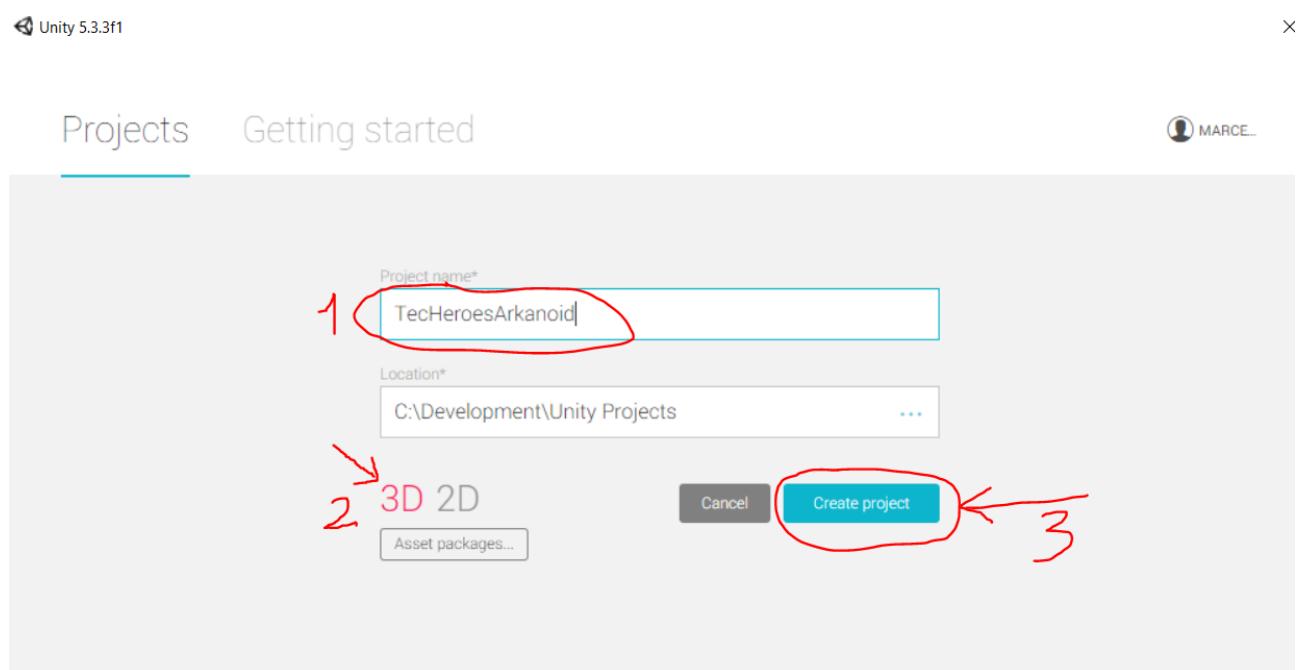
Parte 1 – Creazione progetto e importazione assets	2
Parte 2 – Creazione ambiente di gioco.....	6
Parte 3 – Creazione oggetti di gioco.....	12
Parte 4 – Aggiunta degli script.....	21
Parte 5 – Aggiunta della UI	29
Parte 6 – Creazione del menù	35
Parte 7 – Musica ed effetti sonori	43
Parte 8 – Ritocchi finali e creazione applicazione UWP	47

Parte 1 – Creazione progetto e importazione assets

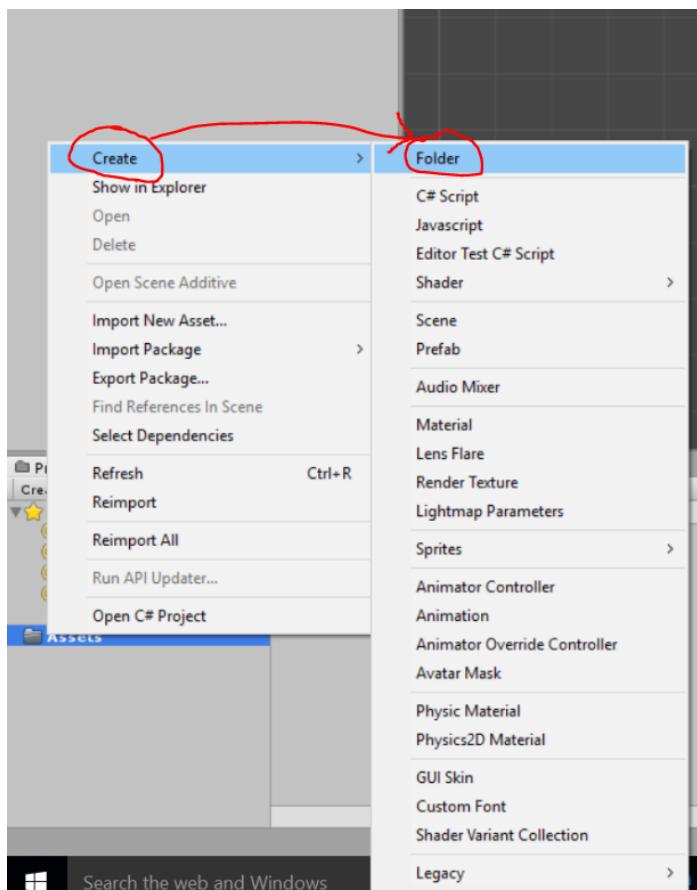
- 1) Lanciare Unity3d
- 2) Sulla pagina iniziale selezionare **New Project**



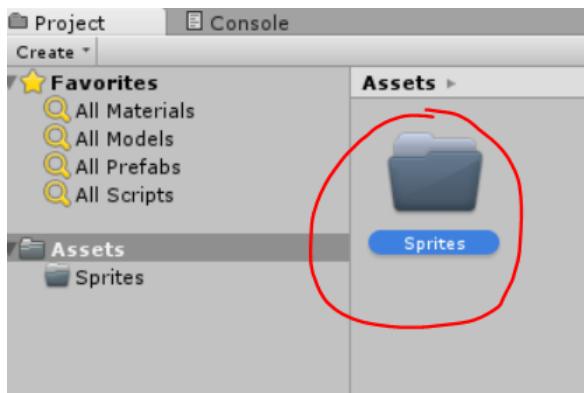
- 3) Impostare il nome del progetto (ad esempio TecHeroesArkanoid), selezionare il tipo di progetto 3D e cliccare su **Create Project**



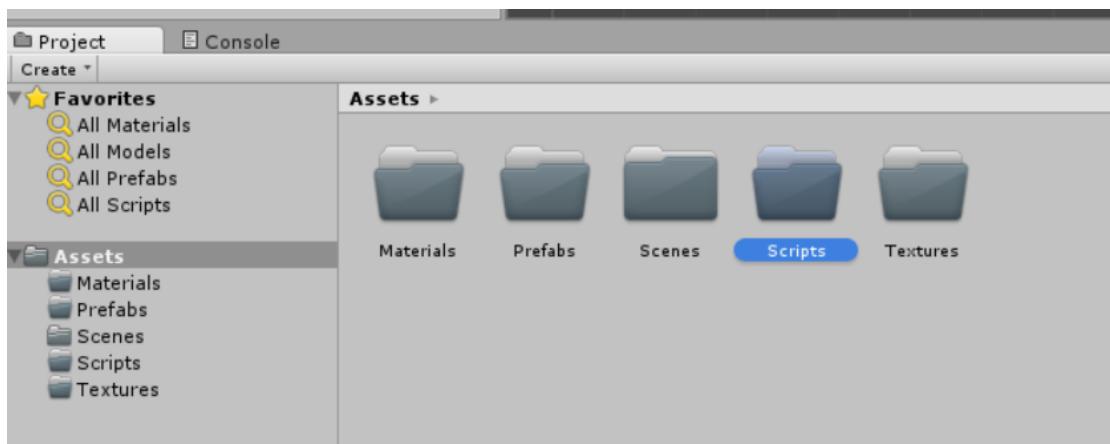
- 4) Nel pannello **Project** cliccare con il tasto destro sulla cartella **Assets** e selezionare **Create→Folder**



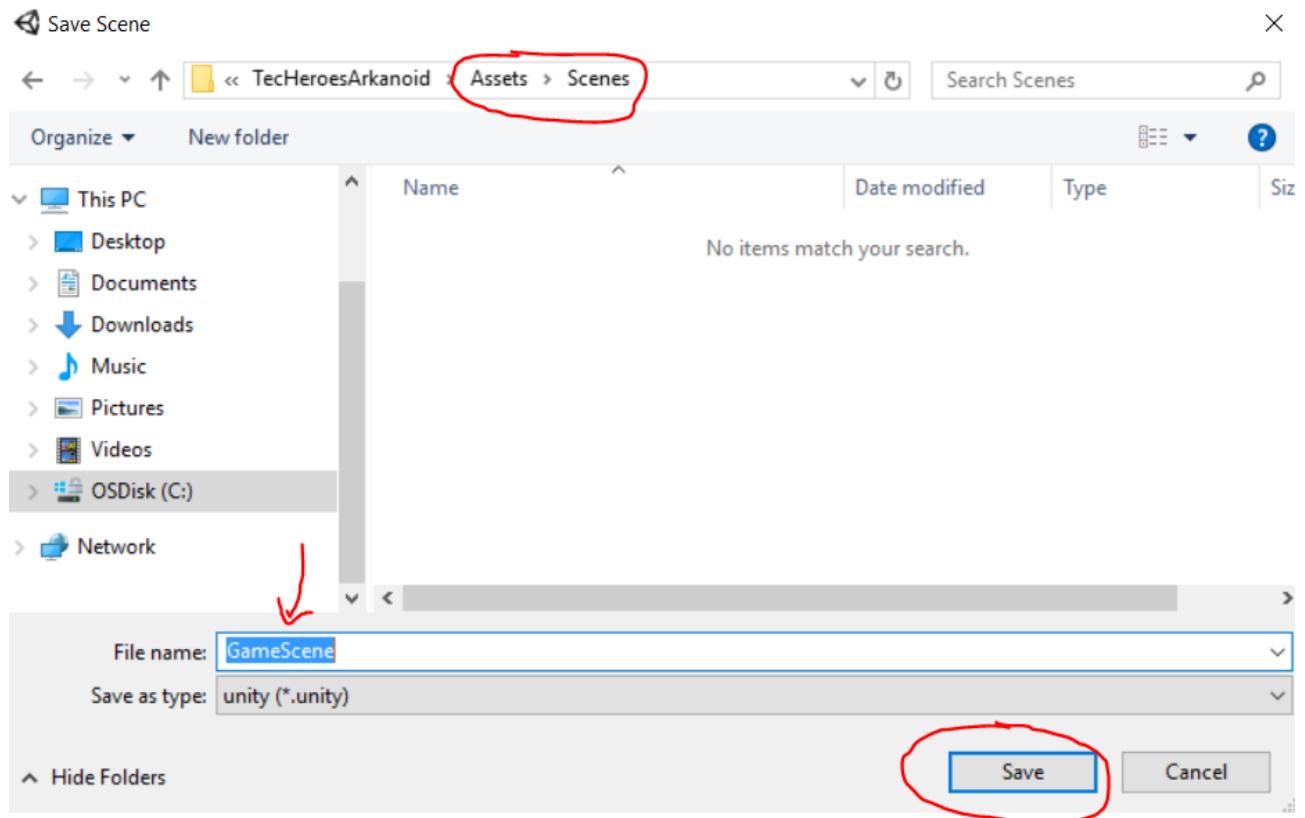
5) Nominare la nuova cartella **Sprites**



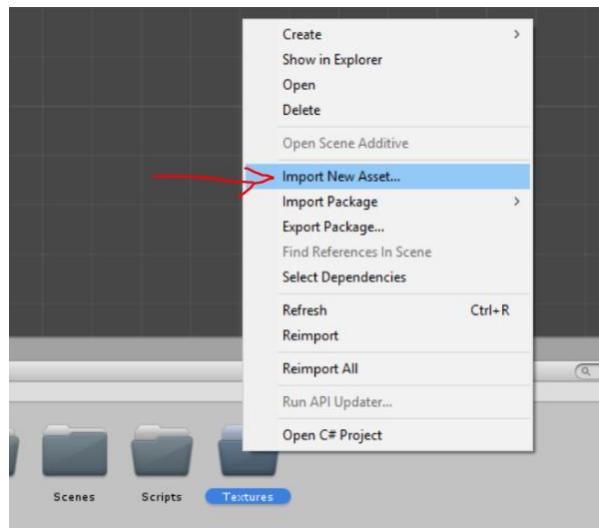
6) Ripetere il punto 5 per creare anche le cartelle **Scenes**, **Prefabs**, **Textures** e **Scripts**



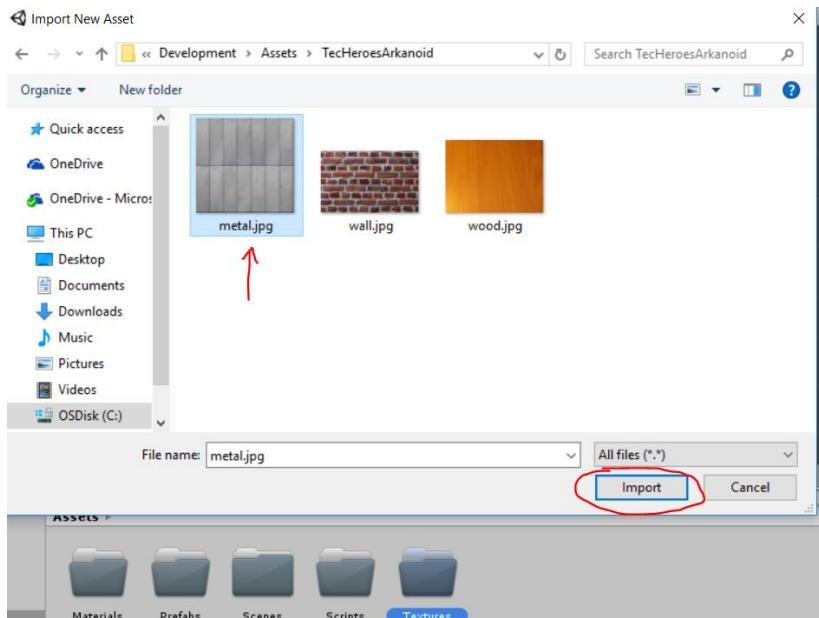
- 7) Salvare la scena corrente cliccando sul menù **File→Save Scene as...**
 8) Selezionare la cartella **Scenes**, impostare **GameScene** com nome della scena e cliccare su **Save**



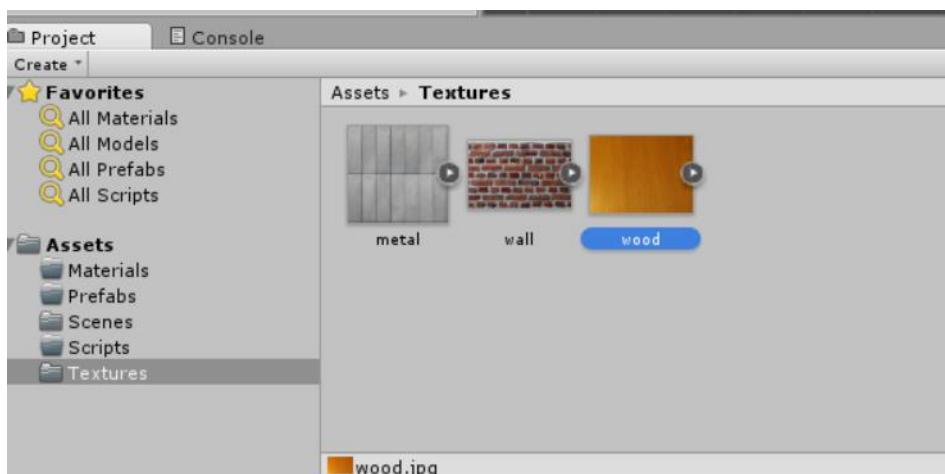
- 9) Importare le textures presenti nelle risorse di questo laboratorio cliccando con il tasto destro sulla cartella **Textures** e selezionando **Import New Assets...**



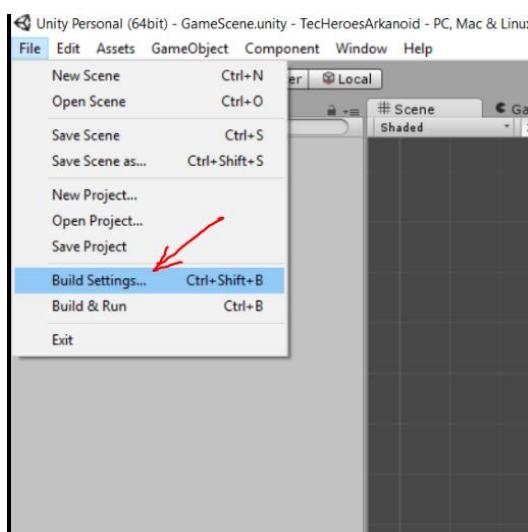
- 10) Selezionare il file **Metal.jpg** e cliccare su **Import**



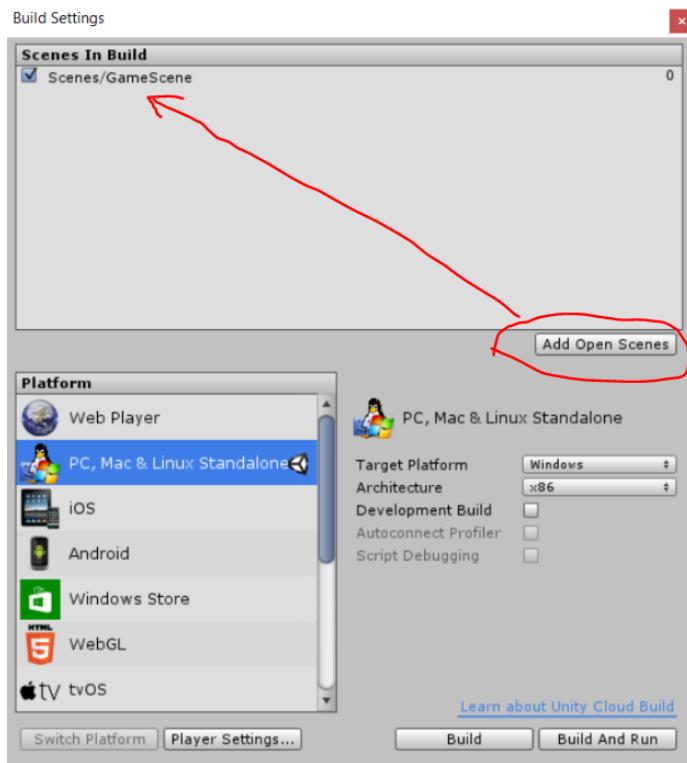
- 11) Ripetere il punto 10) anche per i files **wall.jpg** e **wood.jpg**. Alla fine dell'importazione la cartella **Textures** deve presentarsi in questo modo:



- 12) Aggiungere la scena corrente alla build cliccando sul menuù **File→Build Settings...**

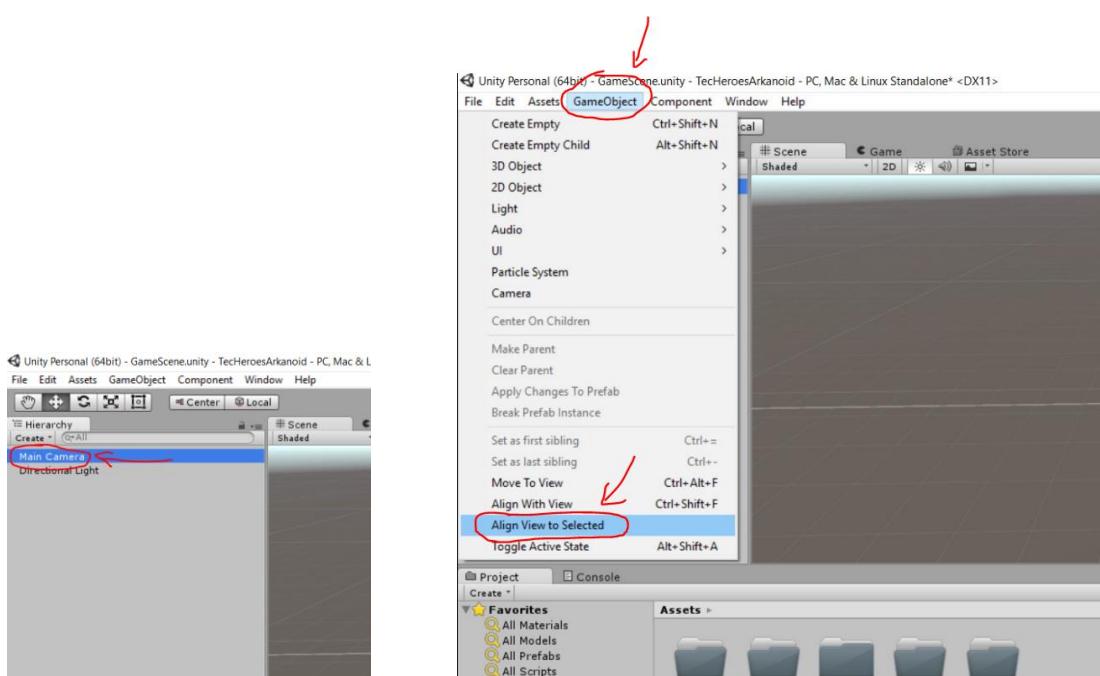


- 13) Nella finestra Build Settings cliccare su **Add Open Scenes** in modo da aggiungere la scena corrente alla build. Poi chiudere la finestra con la X

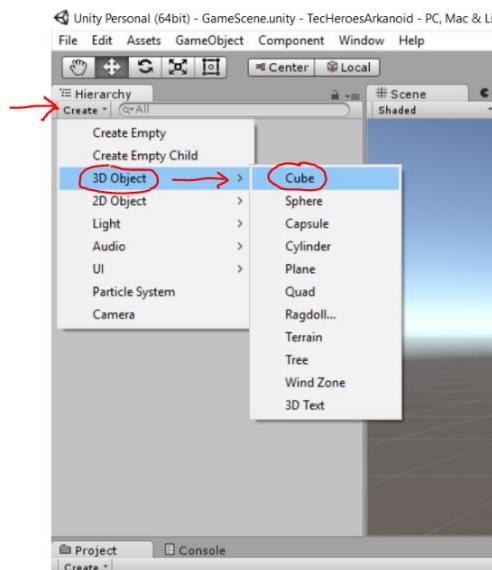


Parte 2 – Creazione ambiente di gioco

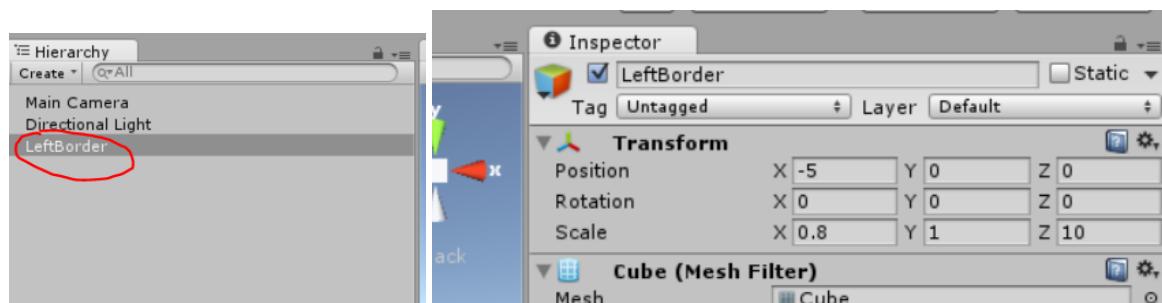
- 1) Allineare la vista alla telecamera principale selezionando la **Main Camera** nel pannello **Hierarchy** e cliccando sul menù **GameObject**→**Allign View to selected**



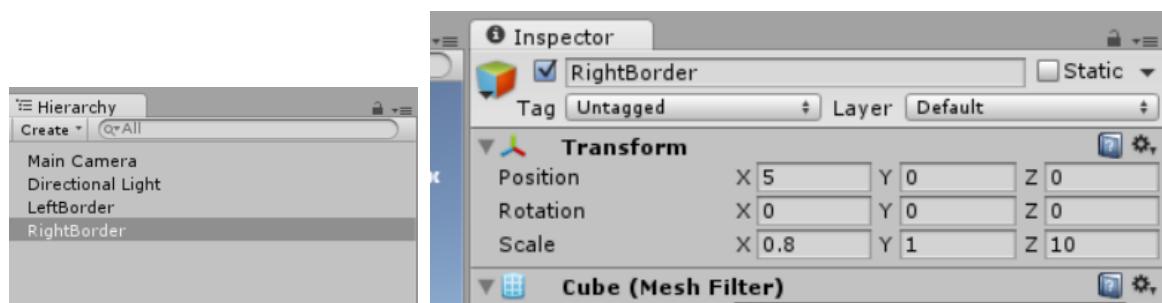
2) Creare un nuovo cubo cliccando sul pulsante **Create** e selezionando **3D Object→Cube**



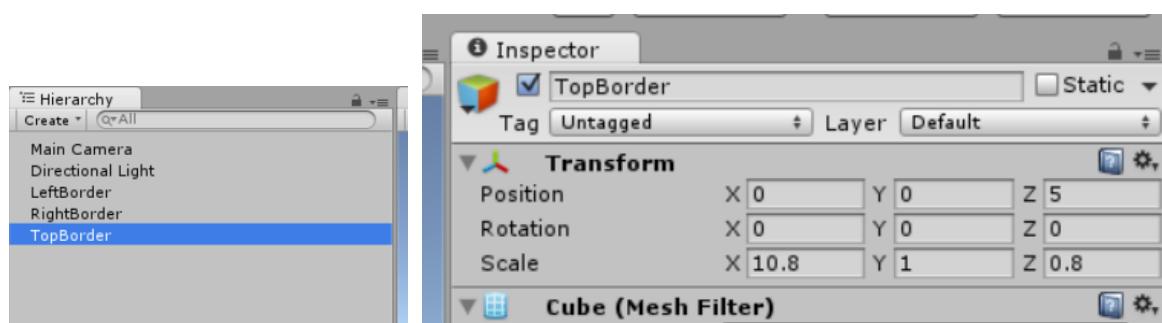
3) Rinominare il cubo appena creato chiamandolo **LeftBorder**, e impostare i valori del componente **Transform** nel pannello **Inspector** come mostrato in figura



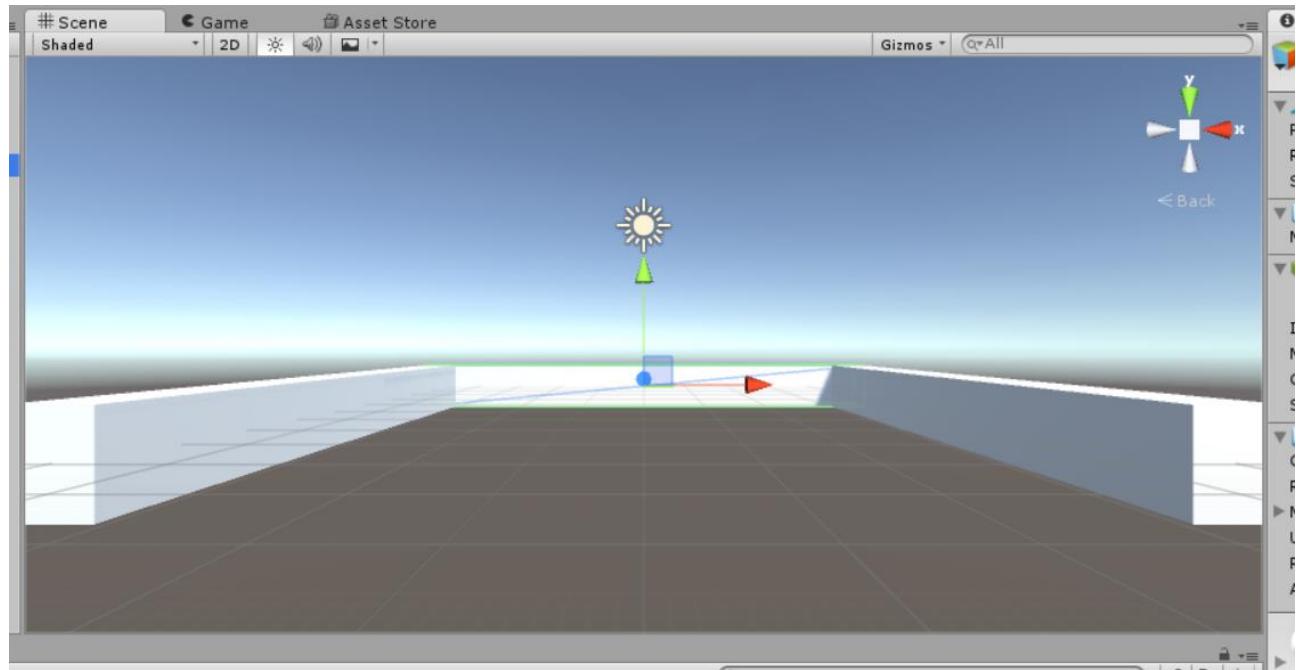
4) Ripetere i passi 2 e 3 per creare **RightBorder** con i seguenti parametri



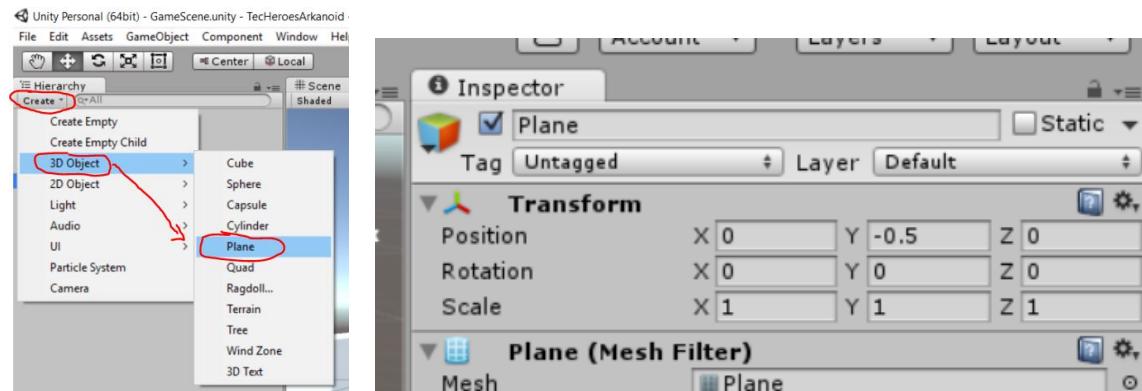
5) Ripetere i passi 2 e 3 per creare **TopBorder** con i seguenti parametri



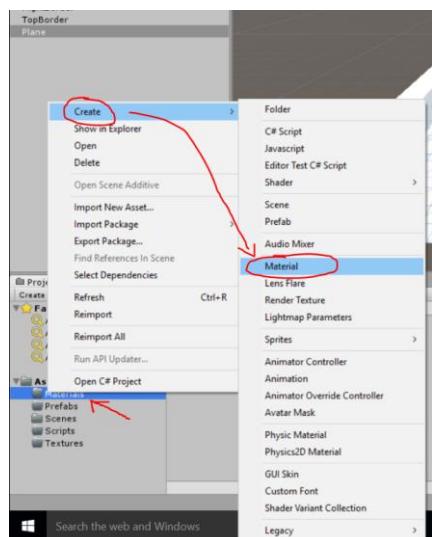
Alla fine la scena deve presentarsi come nella figura seguente



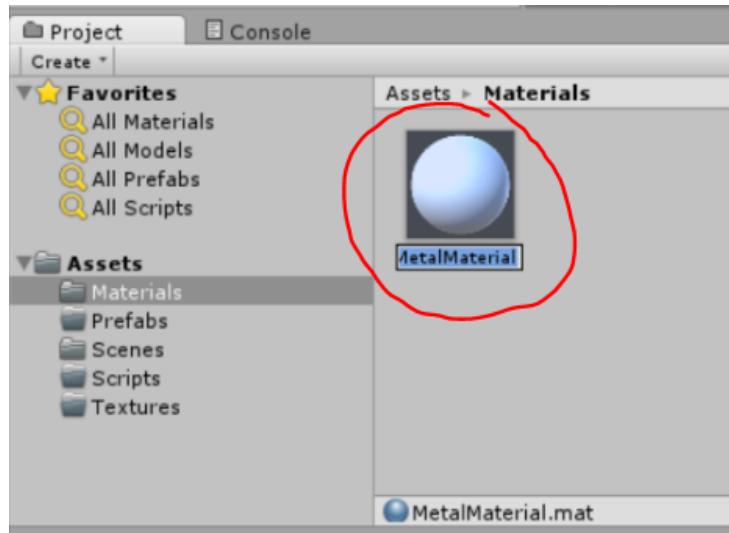
- 6) Aggiungere un piano cliccando sul pulsante **Create** e selezionando **3D Object→Plane** e impostare il componente Transform con i seguenti parametri



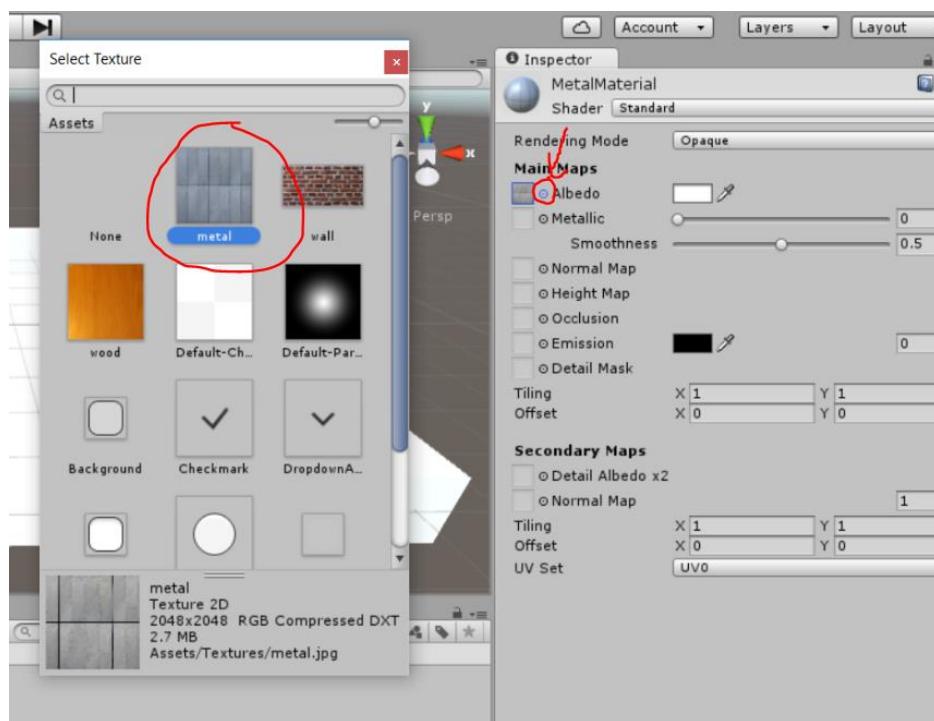
- 7) Creare un nuovo materiale cliccando con il tasto destro sulla cartella **Materials** nel pannello **Project** e selezionando **Create→Material**



8) Impostare il nome del nuovo materiale a **MetalMaterial**



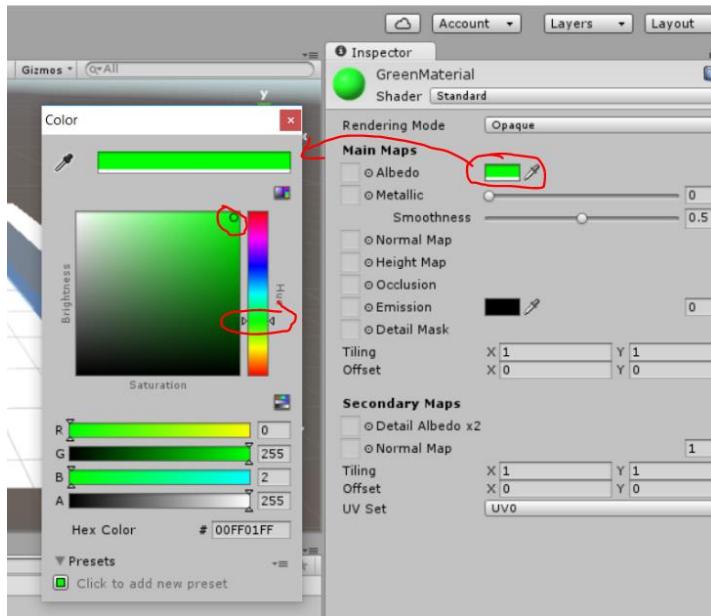
9) Nell'Inspector cliccare sul piccolo bottone circolare vicino alla voce Albedo e dal popup che compare selezionare la texture **metal** tramite un doppio-click



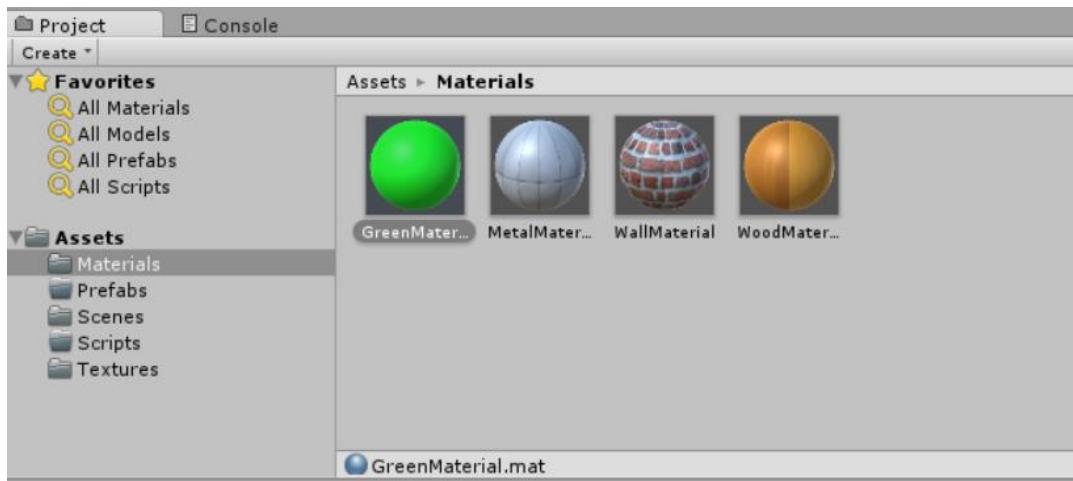
10) Ripetere i passi 7, 8 e 9 per creare il materiale **WoodMaterial** (selezionando la texture **wood**)

11) Ripetere i passi 7, 8 e 9 per creare il materiale **WallMaterial** (selezionando la texture **wall**)

12) Creare anche un materiale chiamato **GreenMaterial**, al quale non va assegnata nessuna texture, ma va semplicemente modificato il colore cliccando sull'apposita area

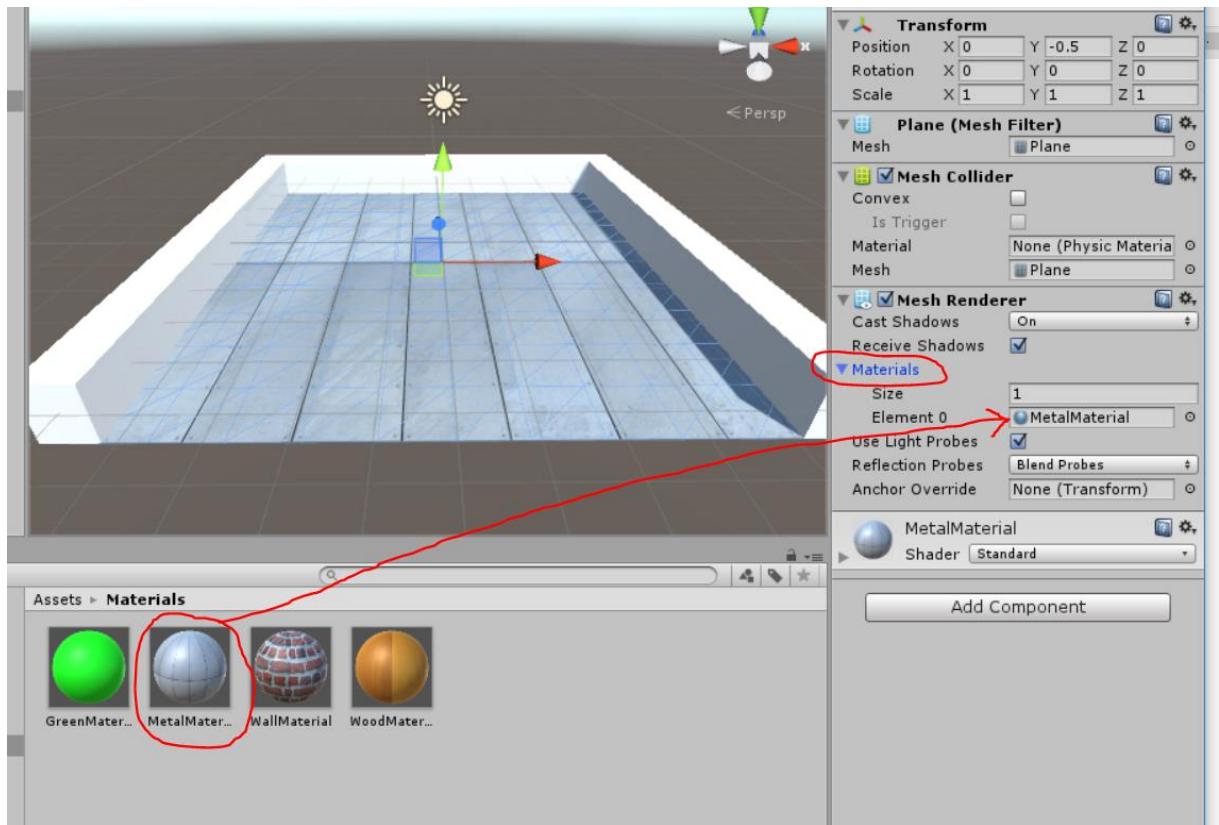


13) Alla fine la cartella Materials deve presentarsi come in figura:

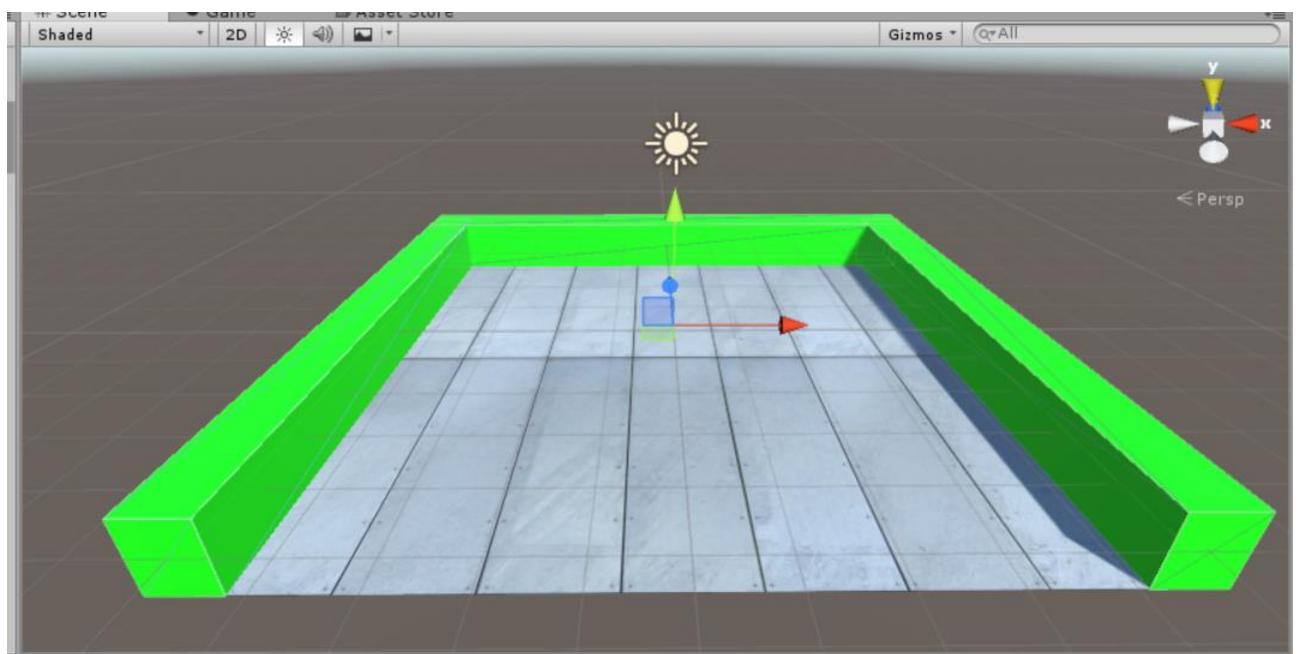


14) Nella **Hierarchy** selezionare **Plane** e nel pannello Inspector espandere la voce **Materials** del componente **Mesh Renderer**

- 15) Trascinare con il mouse il materiale **MetalMaterial** all'interno di **Element 0** per assegnare il materiale al renderer

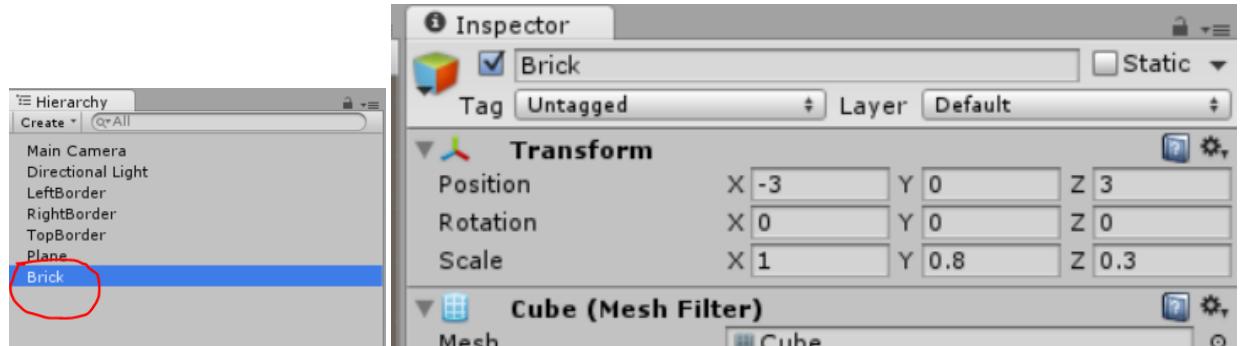


- 16) Ripetere il punto 15 per tutti e 3 gli oggetti del bordo (LeftBOrder, RightBorder e TopBorder), assegnando però GreenMaterial. Alla fine della procedura la scena si deve presentare come in figura

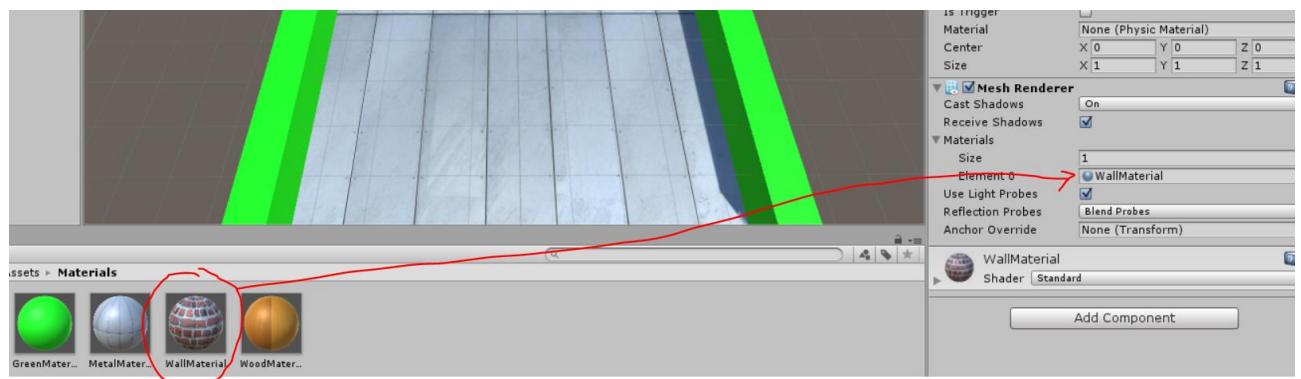


Parte 3 – Creazione oggetti di gioco

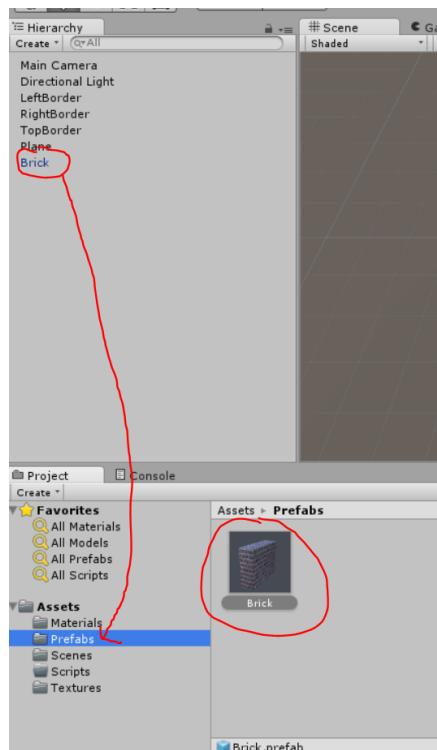
- 1) Aggiungere un nuovo Cube (Create → 3D Object → Cube) e chiamarlo **Brick**. Impostare poi i valori del componente Transform come in figura



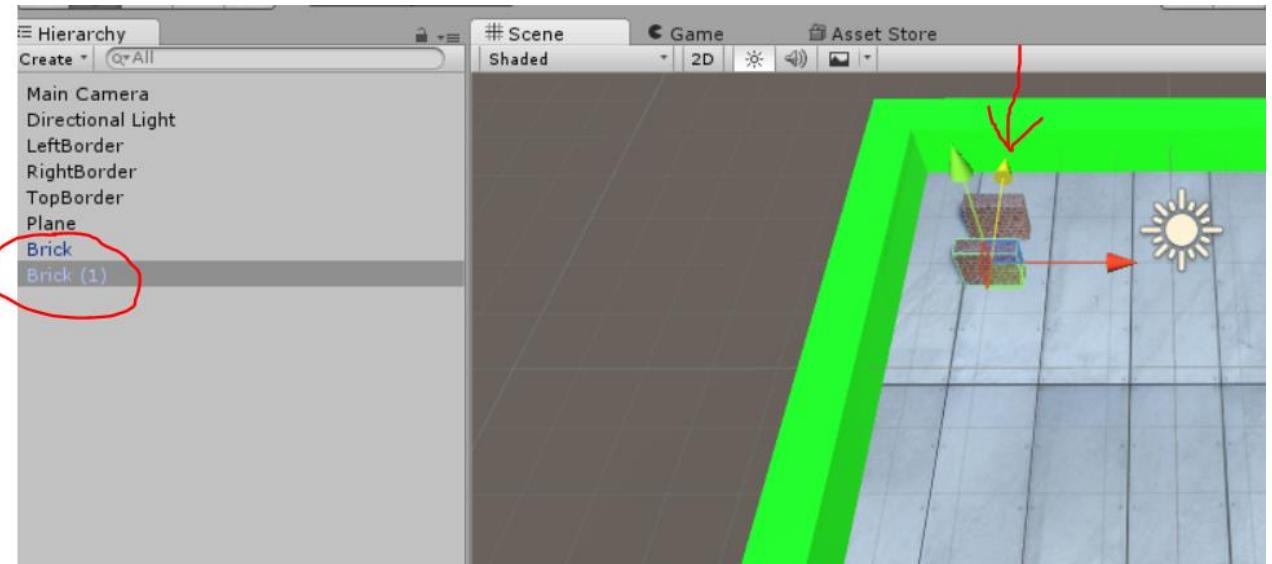
- 2) Impostare il materiale **WallMaterial** come nel punto 15 della precedente sezione



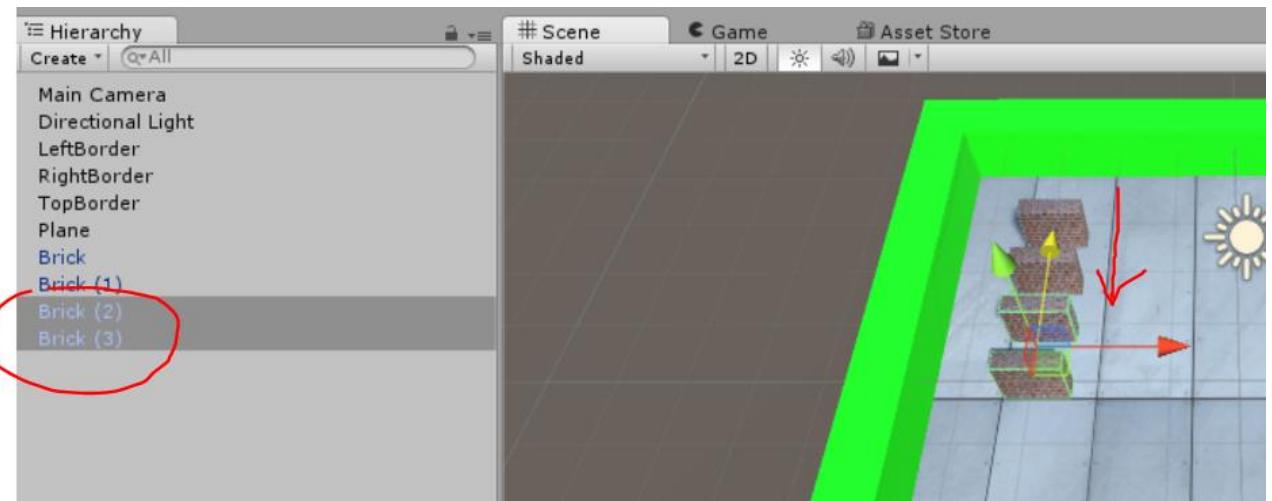
- 3) Creare il prefab dell'oggetto **Brick** trascinandolo dalla **Hierarchy** all'interno della cartella **Prefabs** nel pannello **Project**



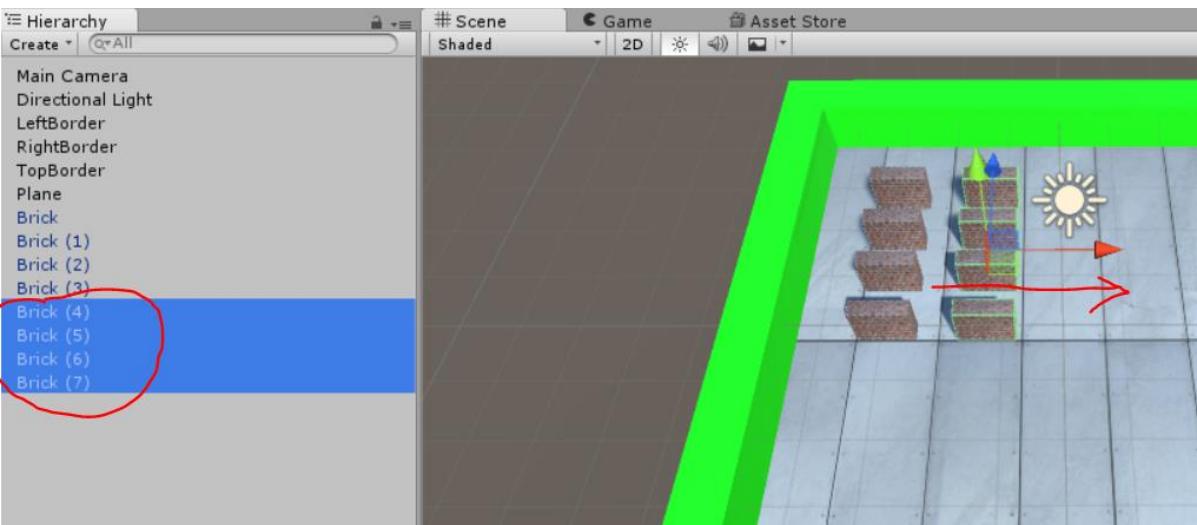
- 4) Duplicare l'oggetto **Brick** selezionandolo e premendo **Ctrl+D**. Dopodiché trascinarlo leggermente sull'asse Z utilizzando la freccia **gialla** del suo gizmo



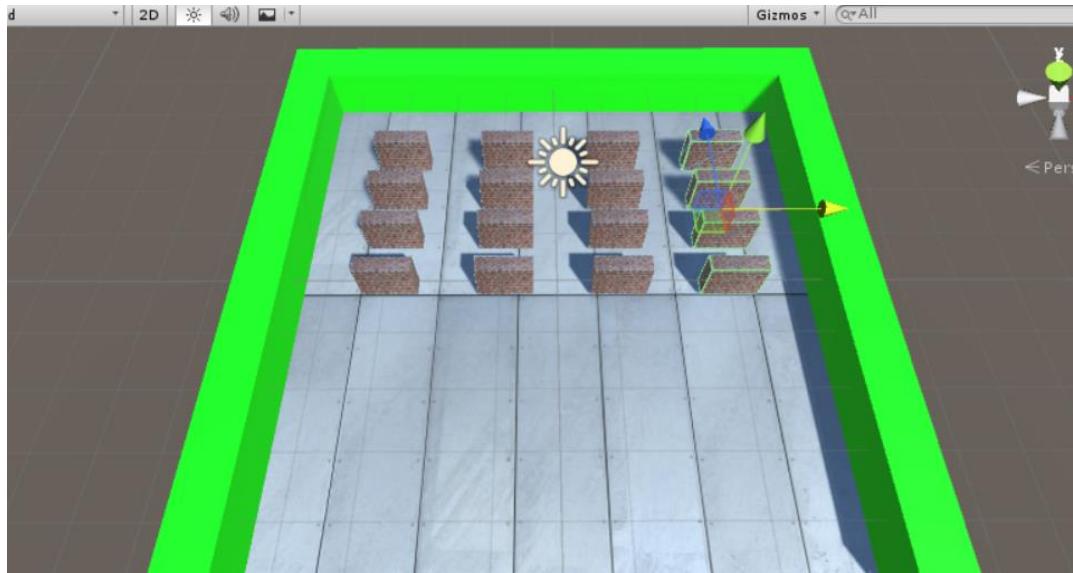
- 5) Ripetere l'operazione ma selezionando contemporaneamente entrambe gli oggetti



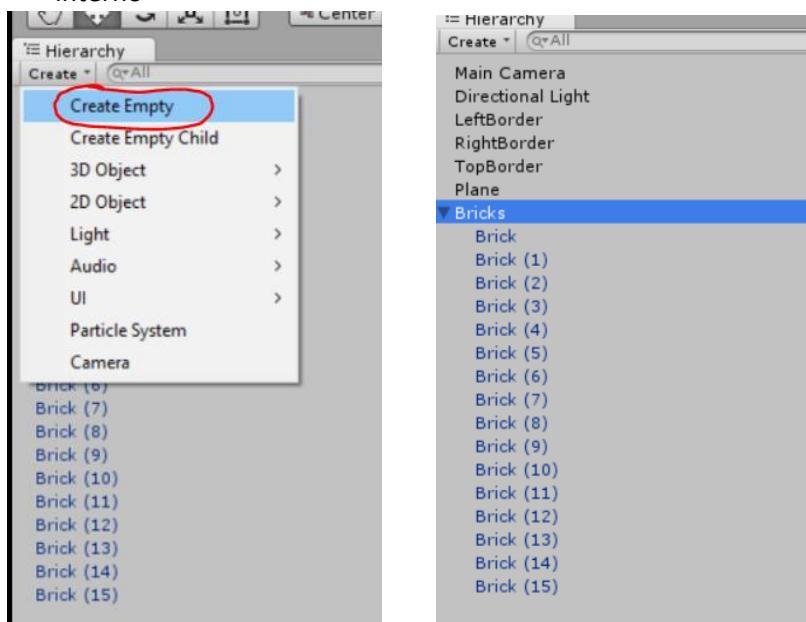
- 6) Ripetere di nuovo l'operazione selezionando tutti e 4 gli oggetti Brick, ma questa volta spostandoli sull'asse X utilizzando la freccia **rossa**



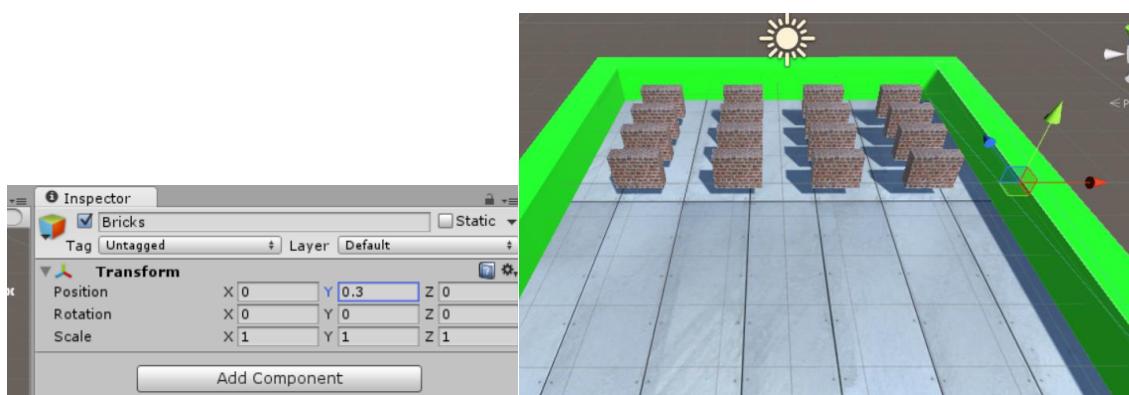
7) Ripetere il punto 6 fino a riempire l'intera area di gioco



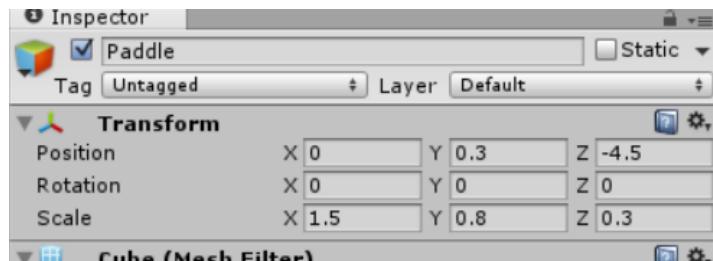
8) Creare un GameObject vuoto chiamato **Bricks** e trascinare tutti gli oggetti appena creati al suo interno



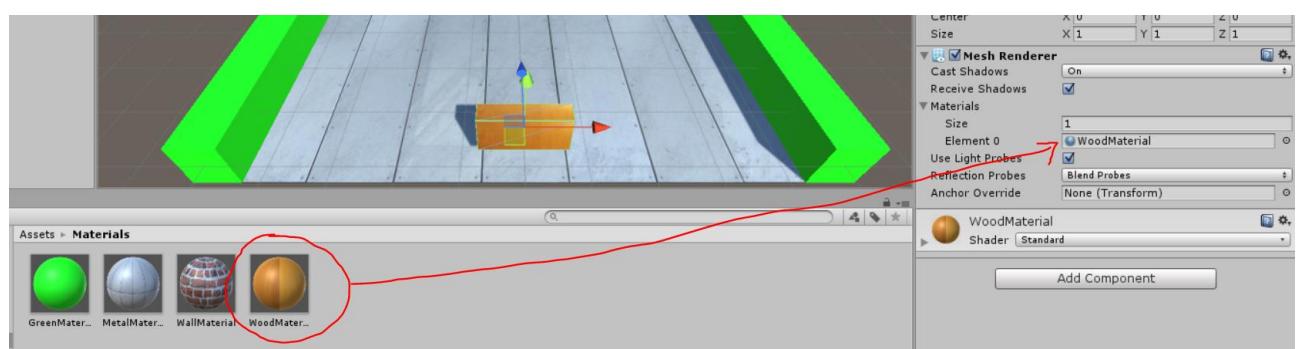
9) Sollevare leggermente tutti i Bricks aumentando leggermente la Y, in modo da ottenere il risultato seguente



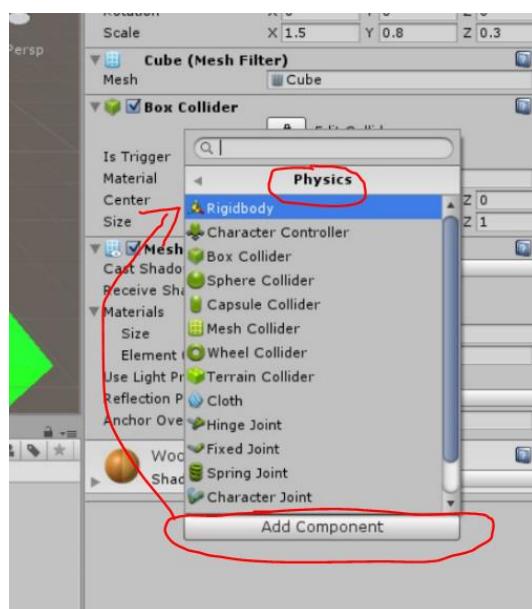
10) Aggiungere un nuovo Cube e chiamarlo **Paddle**. Impostare il componente Transform come mostrato in figura



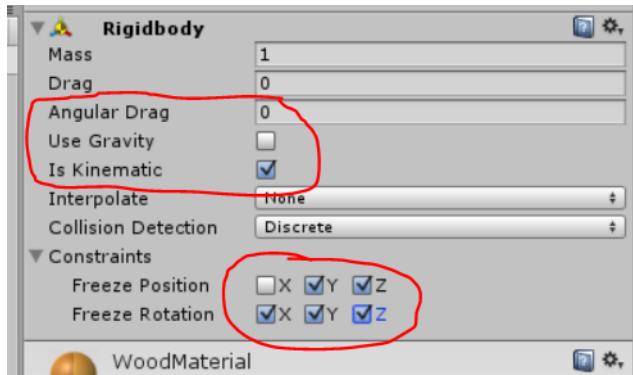
11) Impostare il materiale **WoodMaterial** come illustrato al punto 15 della sezione precedente



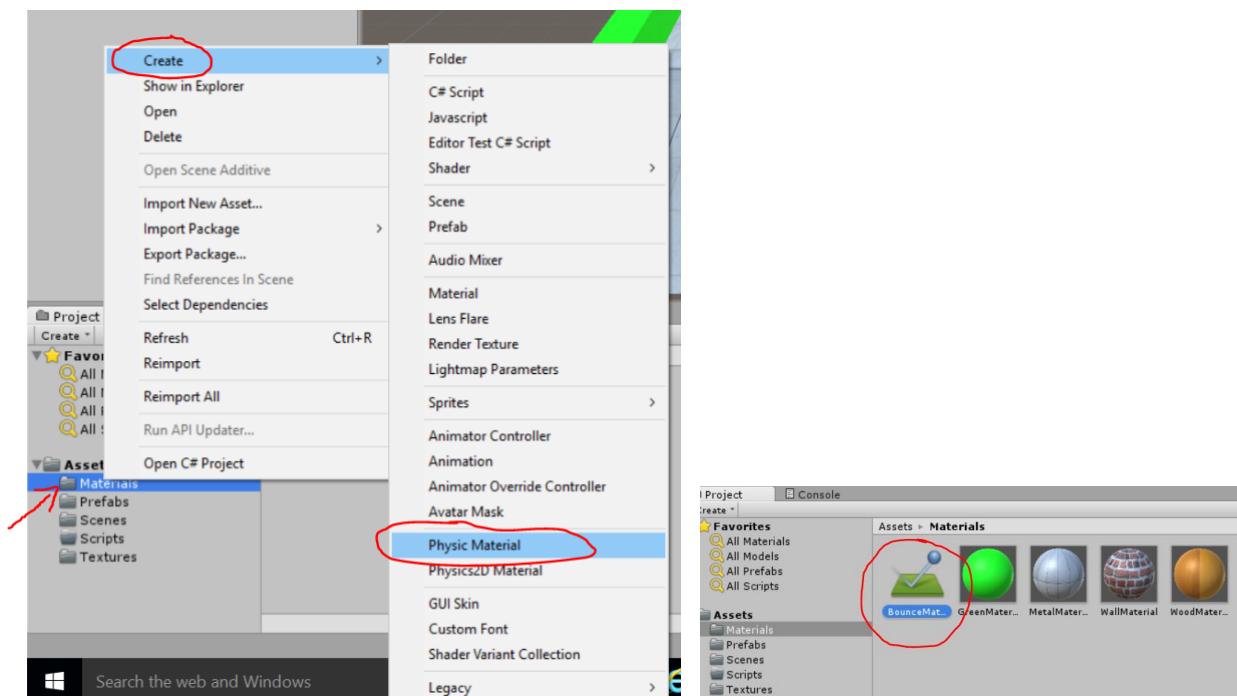
12) Selezionare l'oggetto Paddle nella Hierarchy e aggiungere un componente di tipo **Rigidbody** cliccando sul bottone **Add Component** e selezionando **Physics→Rigidbody**



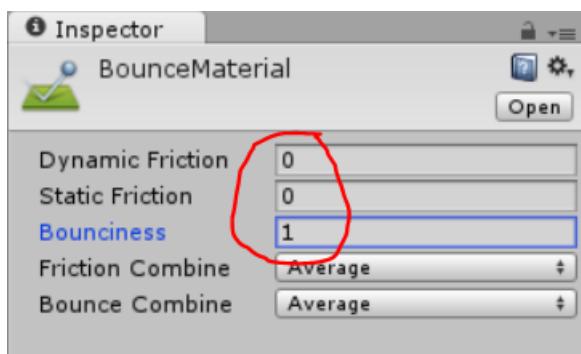
13) Impostare i **Constraints** del Rigidbody come illustrato in figura per bloccare lo spostamento e la rotazione dell'oggetto su tutti gli assi tranne su quello orizzontale. Deselezionare il flag **Use Gravity** e attivare il flag **Is Kinematic** per evitare che il motore della fisica interagisca con l'oggetto (lo muoveremo in seguito via script)



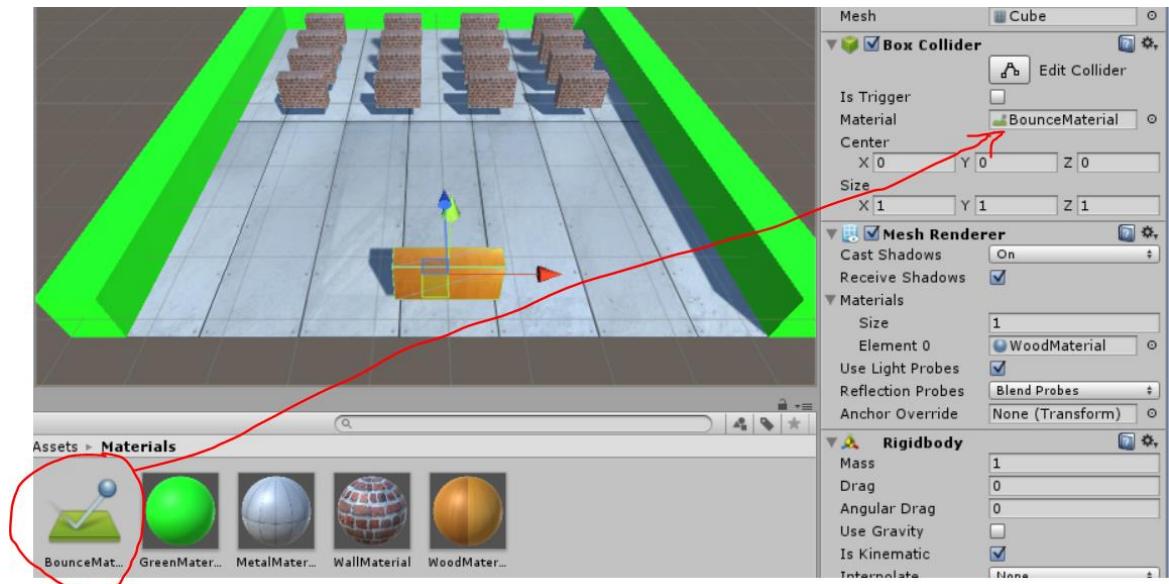
14) Nel pannello Project cliccare con il tasto destro sulla cartella **Materials**, selezionare **Create → Physic Material** ed impostare il nome **BounceMaterial** al nuovo materiale



15) Impostare le proprietà del materiale come mostrato in figura

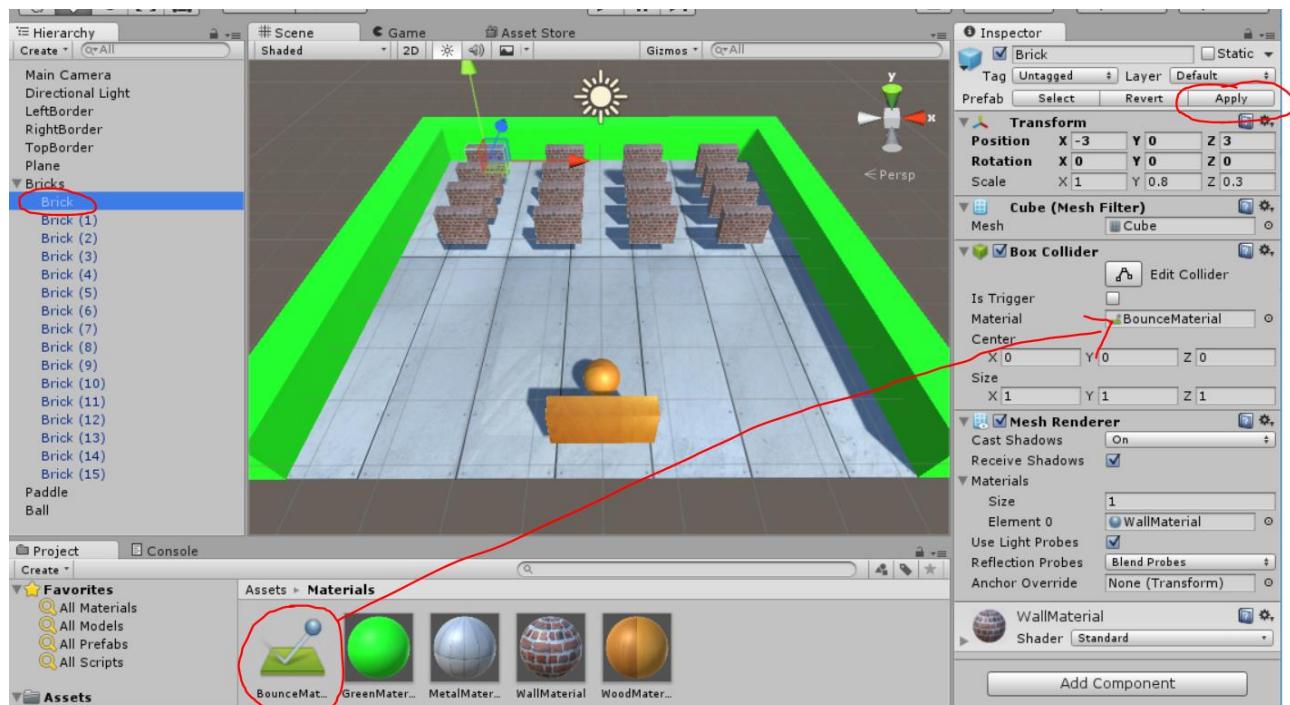


16) Trascinare **BounceMaterial** sulla proprietà **Material** del componente **Box Collider** dell'oggetto **Paddle**



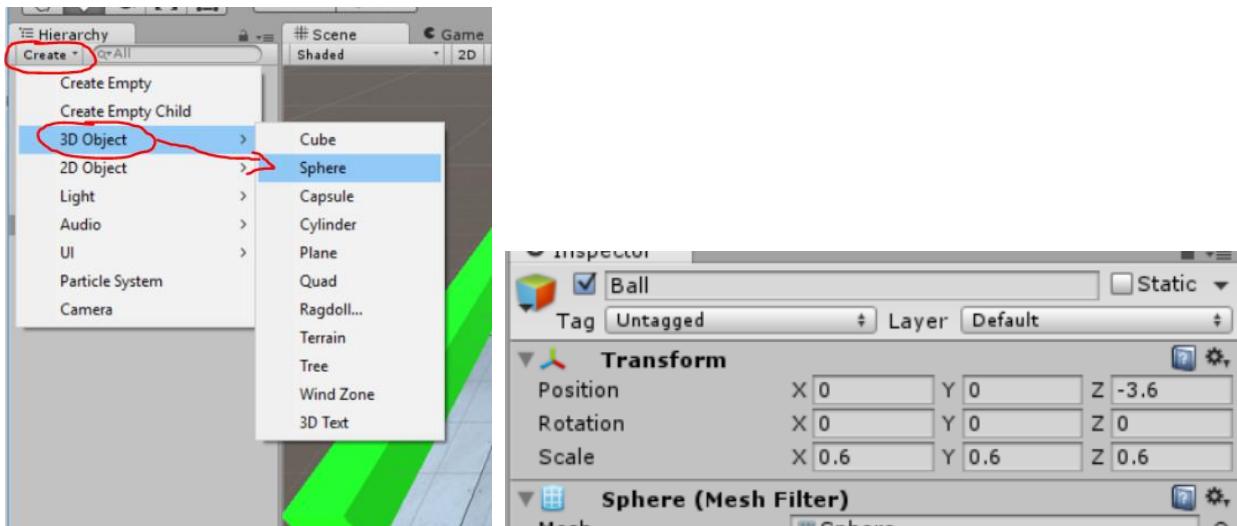
17) Impostare il **BounceMaterial** anche su tutti e 3 i bordi (**LeftBorder**, **RightBorder** e **TopBorder**)

18) Impostare il **BounceMaterial** su uno qualsiasi degli oggetti **Brick** e successivamente cliccare sul bottone **Apply** per propagare la modifica anche su tutte le altre istanze del prefab

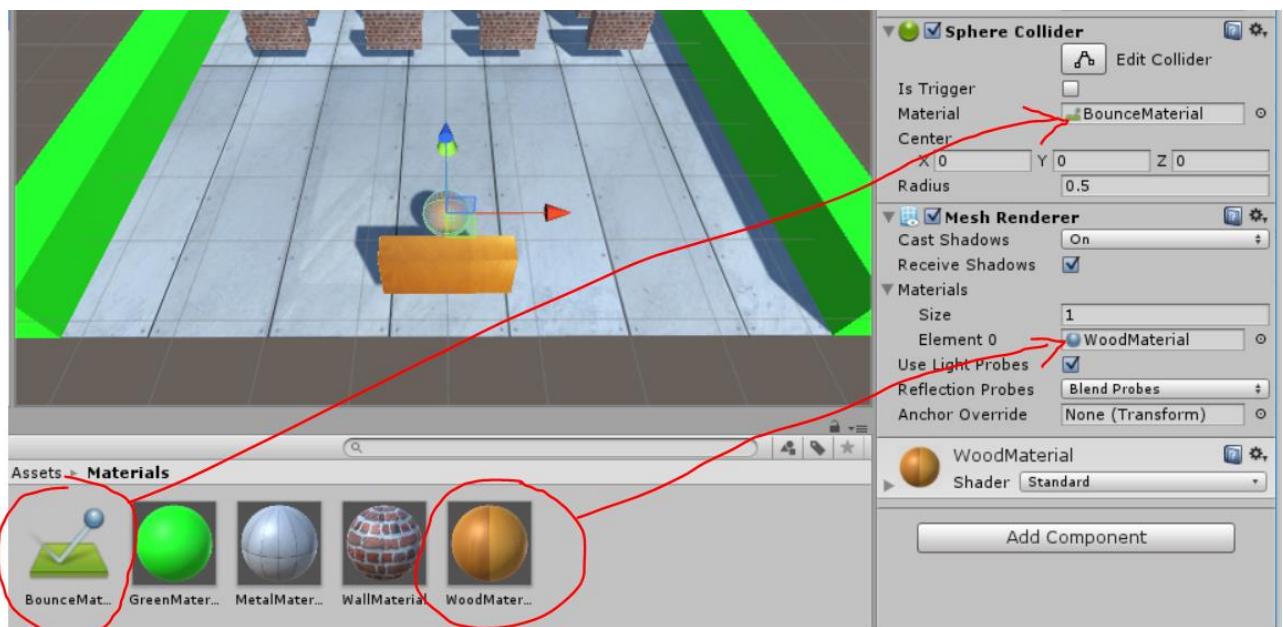


19) Creare una nuova sfera cliccando sul bottone **Create** e selezionando **3D Object→Sphere**.

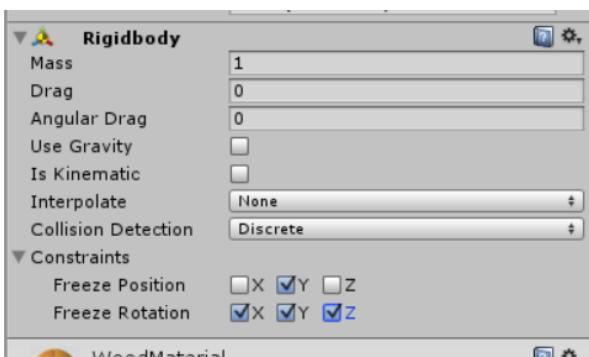
Rinominare l'oggetto come **Ball** e impostare le proprietà del componente Transform come in figura



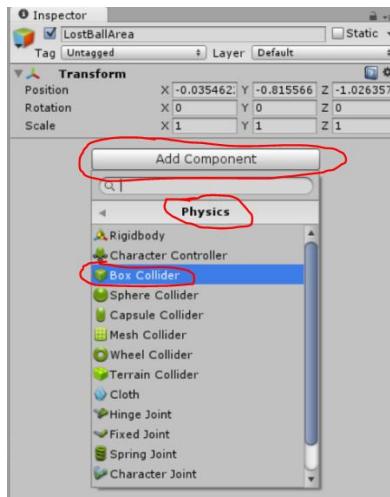
20) Impostare il materiale **WoodMaterial** trascinandolo dalla cartella **Materials** come mostrato in precedenza per il Paddle e impostare il **BounceMaterial** nel collider



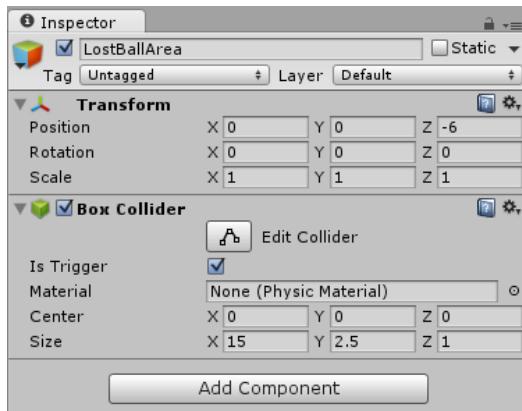
21) Aggiungere il componente **Rigidbody** seguendo la procedura utilizzata per il Paddle e impostare le proprietà come mostrato in figura



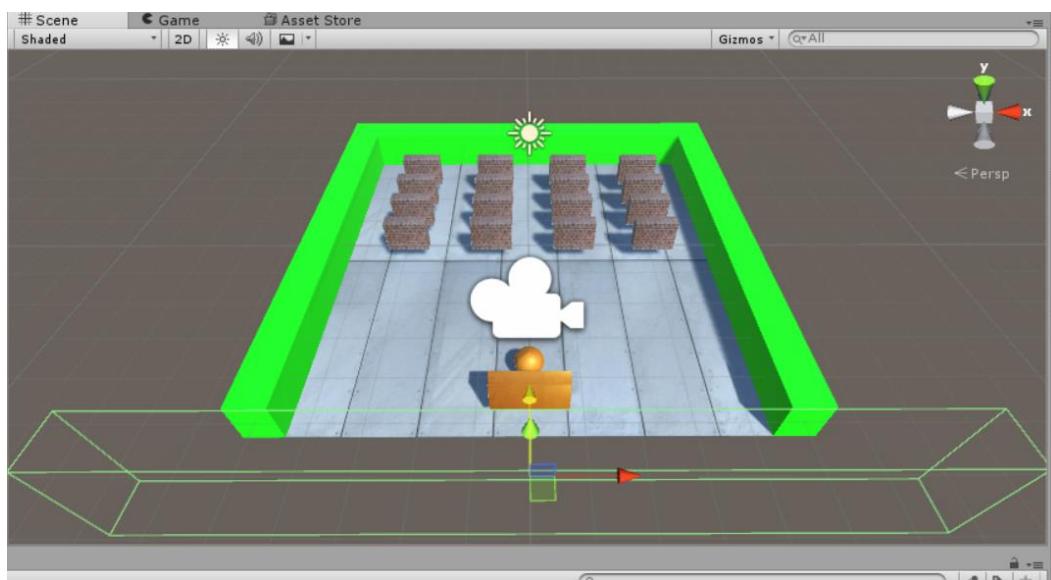
- 22) Creare un GameObject vuoto (Create→Create Empty) e chiamarlo **LostBallArea**
 23) Aggiungere un componente di tipo **BoxCollider** tramite il pulsante **Add Component**



- 24) Impostare i valori del componente **Transform** e del componente **Box Collider** come indicato in figura

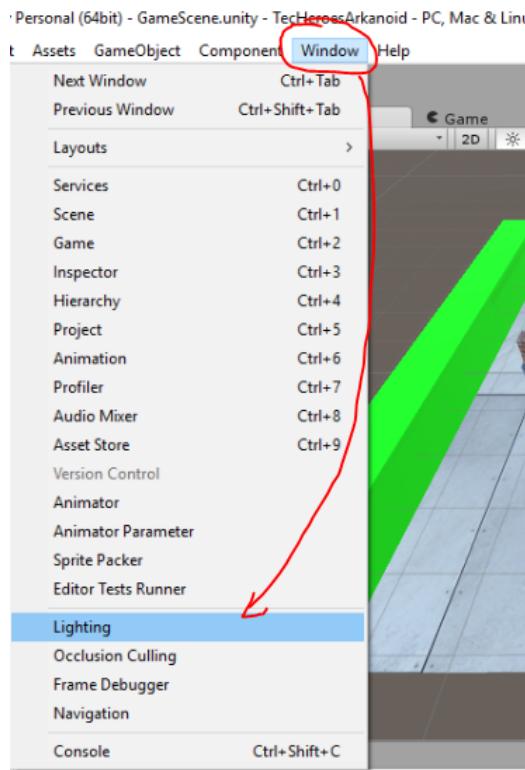


- 25) L'oggetto deve presentarsi in questo modo

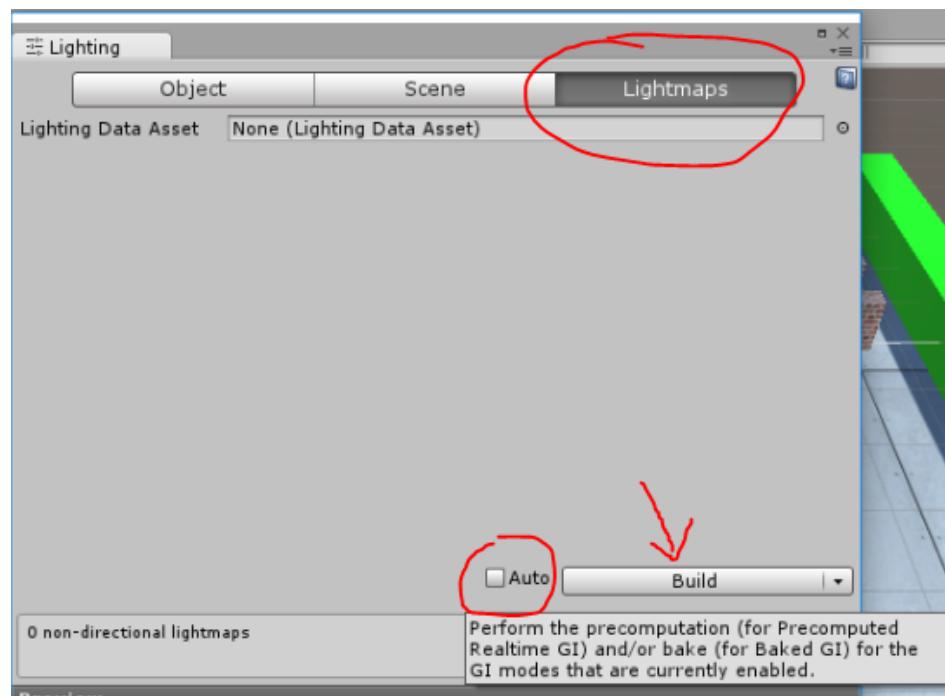


- 26) Salvare la scena dal menù **File→Save Scene** oppure premendo **Ctrl+S**

27) Aprire la finestra Lighting dal menù Windows→Lightning

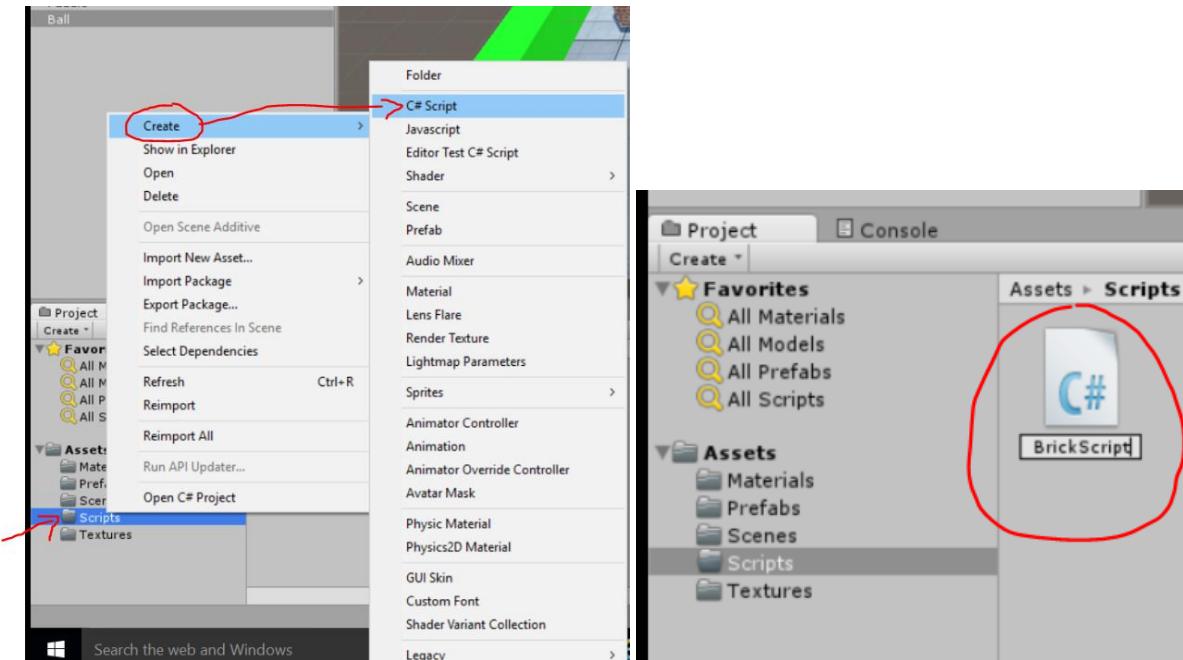


28) Andare nel pannello Lightmap e deselezionare il flag Auto. Infine cliccare sul bottone Build

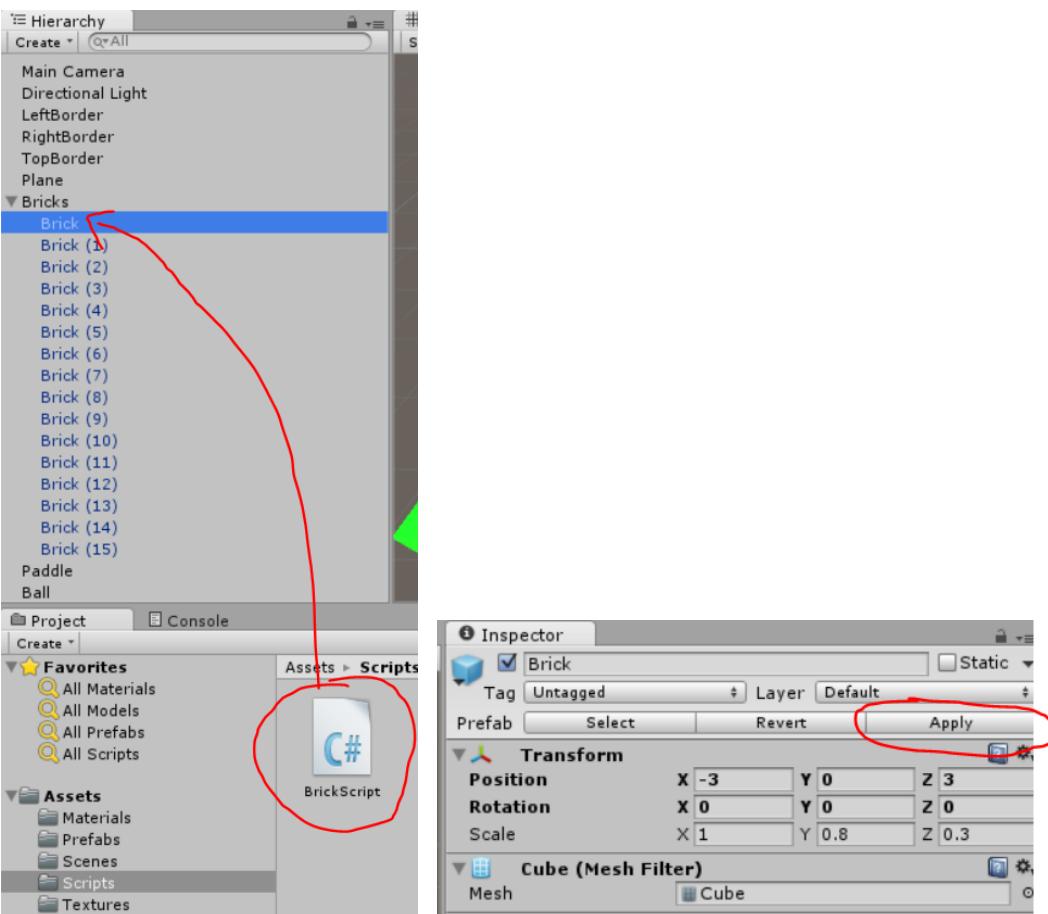


Parte 4 – Aggiunta degli script

- 1) Cliccare con il tasto destro sulla cartella **Scripts** e selezionare **Create→C# Script**. Chiamare il nuovo script **BrickScript**



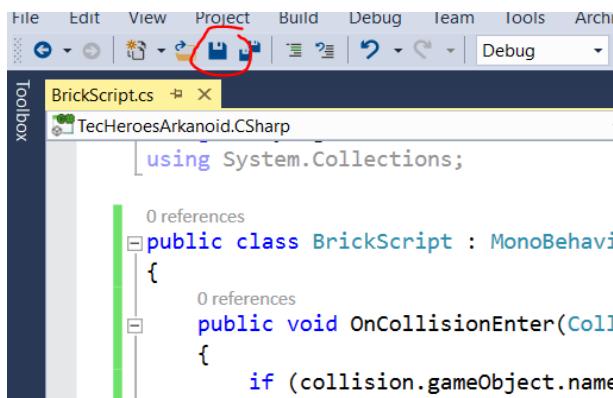
- 2) Trascinare lo script su uno qualsiasi degli oggetti **Brick** nella **Hierarchy** e successivamente cliccare sul bottone **Apply** nell'**Inspector** per propagare il componente a tutte le altre istanze del prefab



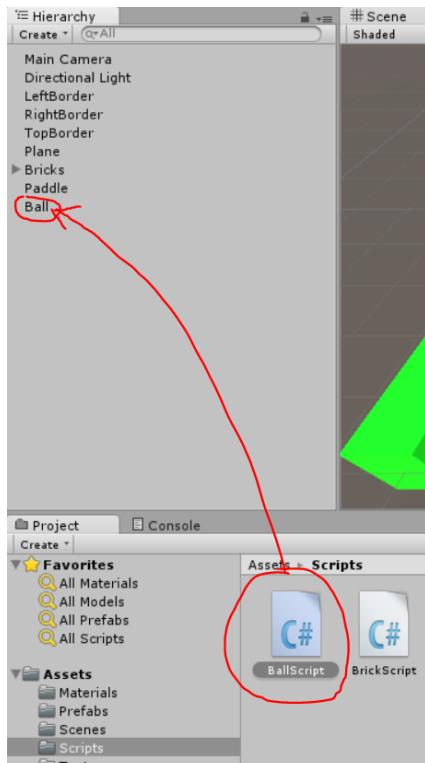
- 3) Aprire lo script in Visual Studio facendo doppio-click sul file
- 4) Rimuovere i metodi Start e Update e aggiungere il metodo OnCollisionEnter in modo che lo script si presenti come riportato qui sotto

```
public class BrickScript : MonoBehaviour
{
    public void OnCollisionEnter(Collision collision)
    {
        //verifica se l'oggetto entrato in collisione è la palla
        if (collision.gameObject.name == "Ball")
        {
            //se si tratta della palla, distrugge il GameObject (ovvero il Brick che è stato colpito)
            Destroy(gameObject);
        }
    }
}
```

- 5) Salvare il file e ritornare in Unity



- 6) Ripetere il passo 1 per creare uno script chiamato **BallScript** e trascinarlo sull'oggetto **Ball** nella Hierarchy

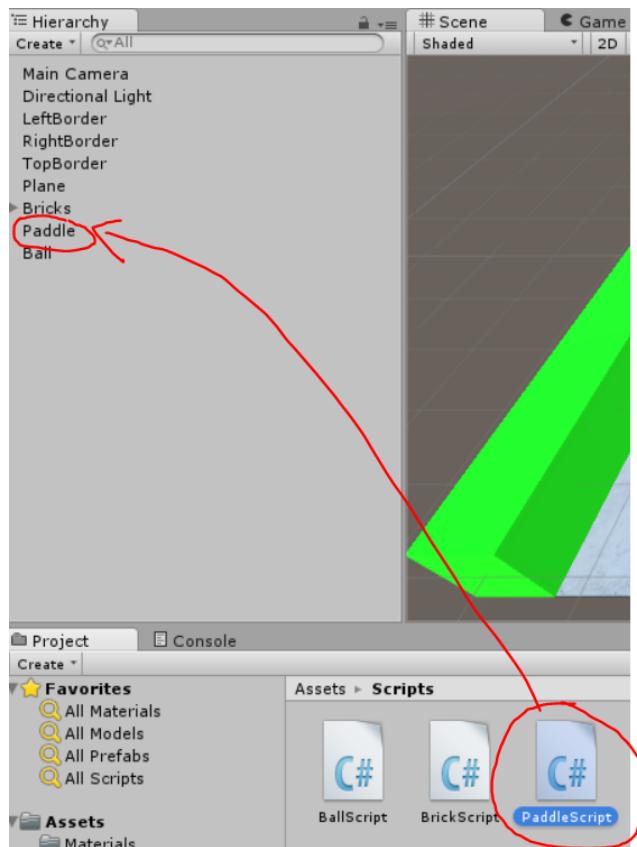


7) Aprire lo script in Visual Studio e modificarlo in questo modo

```
public class BallScript : MonoBehaviour
{
    //proprietà gestibile dall'Inspector per modificare la velocità
    //della palla
    public float speed = 8f;

    //metodo per lanciare la palla
    public void Launch()
    {
        //imposta la velocità del corpo rigido nella direzione della palla
        GetComponent<Rigidbody>().velocity = transform.forward * speed;
    }
}
```

8) Ripetere il passo 1 per creare uno script chiamato **PaddleScript** e trascinarlo sull'oggetto **Paddle** nella Hierarchy



9) Aprire lo script in Visual Studio e modificarlo in questo modo

```
public class PaddleScript : MonoBehaviour
{
    //riferimento all'oggetto Ball
    public GameObject ball;
    //parametro per gestire la velocità di movimento del paddle dall'Inspector
    public float speed = 0.2f;

    //booleano che indica se la palla è "ancorata" al paddle
    private bool ballLocked = true;

    //il metodo Update viene richiamato automaticamente N volte al secondo
    void Update()
    {
        //sposta il paddle nella direzione ricavata dall'oggetto Input sull'asse orizzontale
        transform.position = new Vector3(transform.position.x + (Input.GetAxis("Horizontal") * speed),
                                         transform.position.y,
                                         transform.position.z);

        //blocca il movimento se la coordinata x supera il valore 3.8 verso destra)
        if (transform.position.x > 3.8)
            transform.position = new Vector3(3.8f, transform.position.y, transform.position.z);
        //blocca il movimento se la coordinata x supera il valore -3.8 verso sinistra)
        if (transform.position.x < -3.8)
            transform.position = new Vector3(-3.8f, transform.position.y, transform.position.z);

        //se la palla è "ancorata" imposta la sua posizione in relazione a quella del paddle
        if (ballLocked)
            ball.transform.position = transform.position + Vector3.forward;

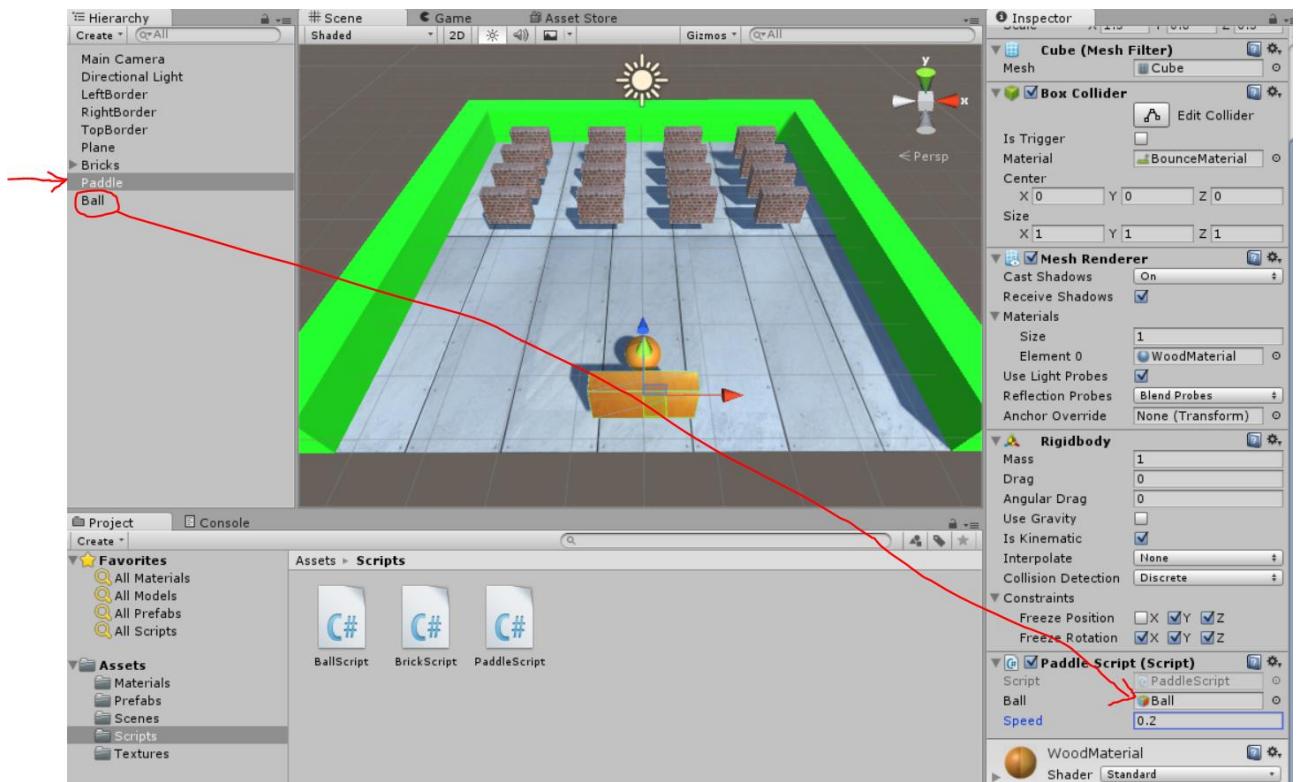
        //se viene premuta la barra spaziatrice, la palla viene lanciata
        if (Input.GetKeyUp(KeyCode.Space) && ballLocked)
            UnlockBall();
    }

    //metodo che "ancora" la palla al paddle
    public void LockBall()
    {
        ballLocked = true;
        ball.transform.position = transform.position + Vector3.forward;
    }

    //metodo che lancia la palla "sbloccandola" dal paddle
    public void UnlockBall()
    {
        ballLocked = false;
        //richiama il metodo Launch dello script BallScript
        ball.GetComponent<BallScript>().Launch();
    }
}
```

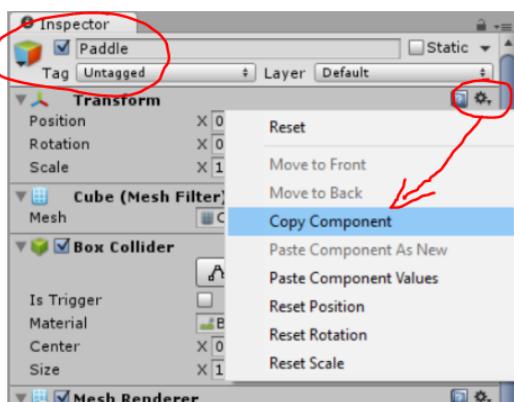
10) Salvare lo script e ritornare in Unity

11) Nella Hierarchy selezionare l'oggetto **Paddle** e successivamente trascinare l'oggetto **Ball** dalla Hierarchy alla proprietà **Ball** dello script nell'Inspector

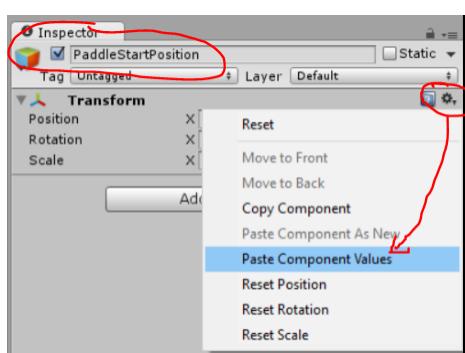


12) Creare un GameObject vuoto (Create→Create Empty), chiamarlo **PaddleStartPosition**

13) Selezionare l'oggetto **Paddle** e copiare i valori del componente Transform cliccando sul piccolo ingranaggio nel componente e selezionando **Copy Component**.



14) Selezionare l'oggetto **PaddleStartPosition**, cliccare sul piccolo ingranaggio nel componente Transform e selezionare **Paste Component Values**



- 15) Creare un GameObject vuoto (Create → Create Empty) e chiamarlo **GameManager**
- 16) Ripetere il passo 1 per creare uno script chiamato **GameManagerScript** e trascinarlo sull'oggetto **GameManager** nella Hierarchy

- 17) Aprire lo script in Visual Studio e modificarlo in questo modo

```
public class GameManagerScript : MonoBehaviour
{
    //riferimento all'oggetto Paddle
    public GameObject paddle;
    //riferimento alla posizione di partenza del paddle
    public Transform paddleStartPosition;

    //membro statico per l'implementazione del pattern singleton
    public static GameManagerScript singleton;

    //metodo richiamato automaticamente da Unity alla creazione dell'oggetto
    void Awake()
    {
        //imposto il membro statico con l'istanza corrente
        singleton = this;
    }

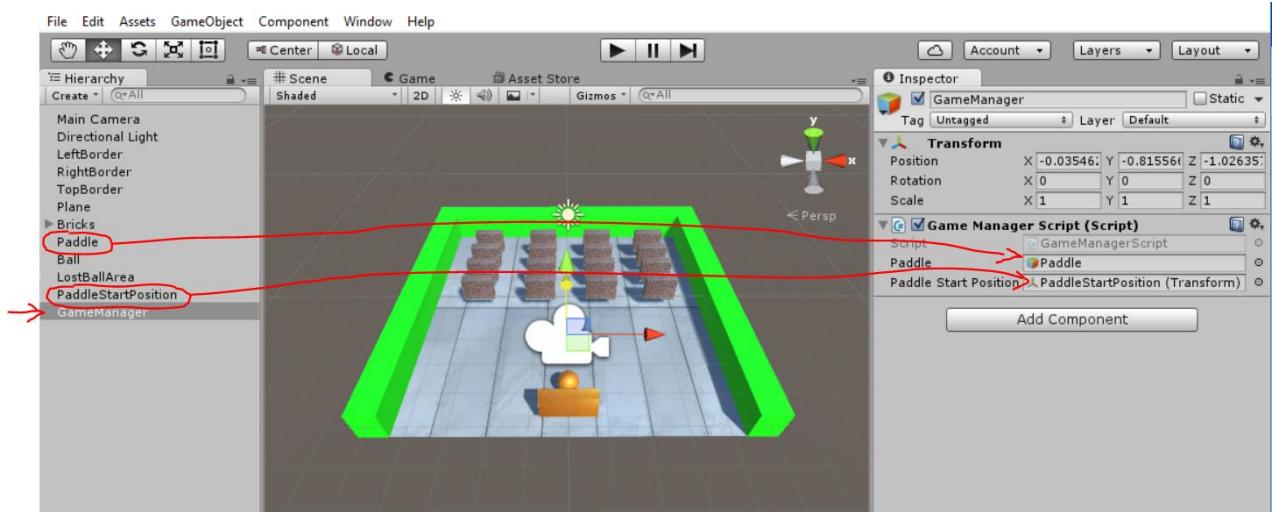
    //metodo richiamato automaticamente da Unity prima del primo ciclo di Update
    void Start()
    {
        //richiama la funzione di inizializzazione del livello
        InitGame();
    }

    //metodo che inizializza gli oggetti
    public void InitGame()
    {
        ResetPaddle();
    }

    //metodo che riporta il paddle allo stato iniziale e "ancora" la palla
    public void ResetPaddle()
    {
        paddle.transform.position = paddleStartPosition.position;
        paddle.GetComponent<PaddleScript>().LockBall();
    }
}
```

- 18) Salvare lo script e tornare a Unity

19) Selezionare l'oggetto **GameManager** e trascinare **Paddle** e **PaddleStartPosition** sulle relative proprietà del componente **Game Manager Script**



20) Ripetere il passo 1 per creare uno script chiamato **LostBallScript** e trascinarlo sull'oggetto **LostBallArea** nella Hierarchy

21) Aprire lo script in Visual Studio e modificarlo in questo modo

```
public class LostBallScript : MonoBehaviour
{
    //evento che viene scatenato dal Box Collider impostato come trigger
    //quando un altro collider entra nella zona di collisione
    void OnTriggerEnter(Collider coll)
    {
        //se il collider è quello dell'oggetto Ball ricomincio
        //più avanti nel laboratorio questa parte verrà modificata
        if (coll.gameObject.name == "Ball")
            GameManagerScript.singleton.InitGame();
    }
}
```

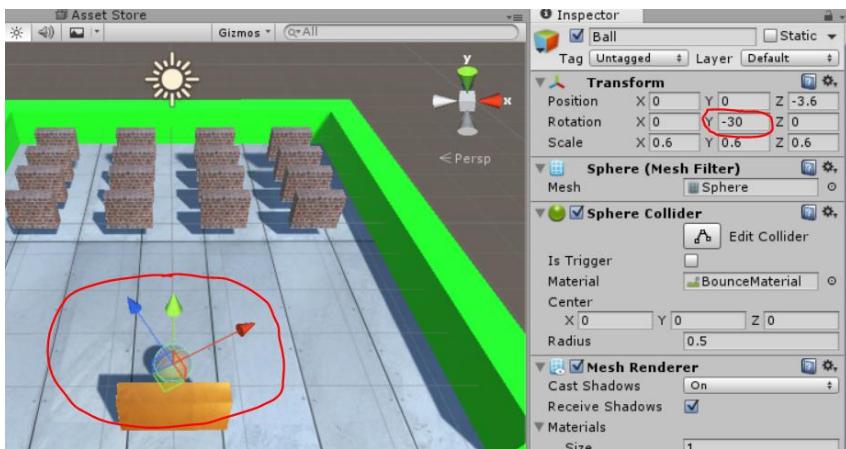
22) Spostare la visualizzazione fino ad ottenere la prospettiva che si desidera avere durante il gioco.

Dopo di che selezionare l'oggetto **Main Camera** e infine cliccare sul menù

GameObject→AlignWith View



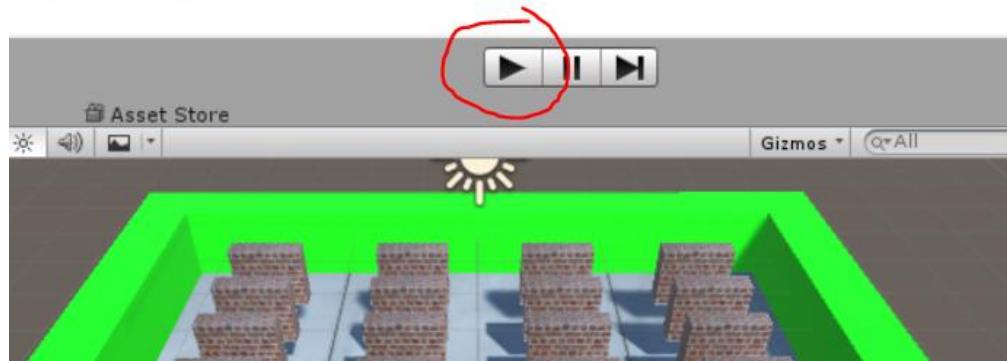
23) Selezionare l'oggetto **Ball** e ruotarlo leggermente da un lato modificando la rotazione sull'asse Y



24) Salvare la scena (Ctrl+S) e provare a mandare in esecuzione il gioco premendo sul pulsante **Play**.

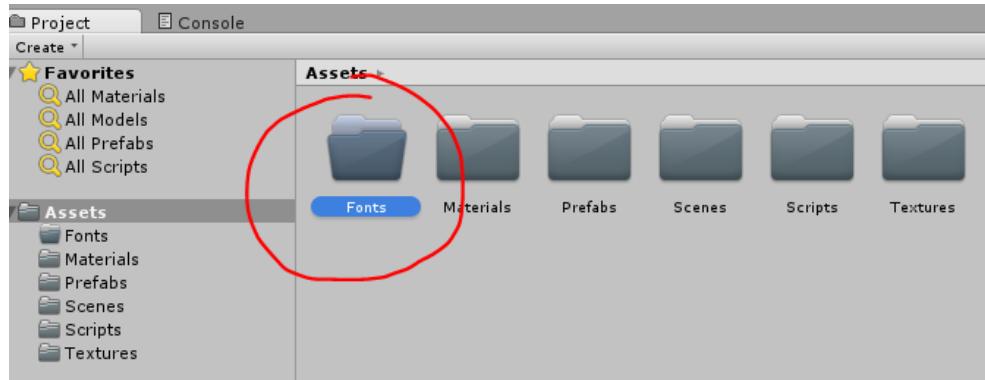
Tramite le frecce direzionali è possibile muovere il paddle, mentre con la barra viene lanciata la pallina. Per uscire dalla modalità gioco e tornare nell'editor è sufficiente cliccare di nuovo sul pulsante Play

tandalone* <UX11>

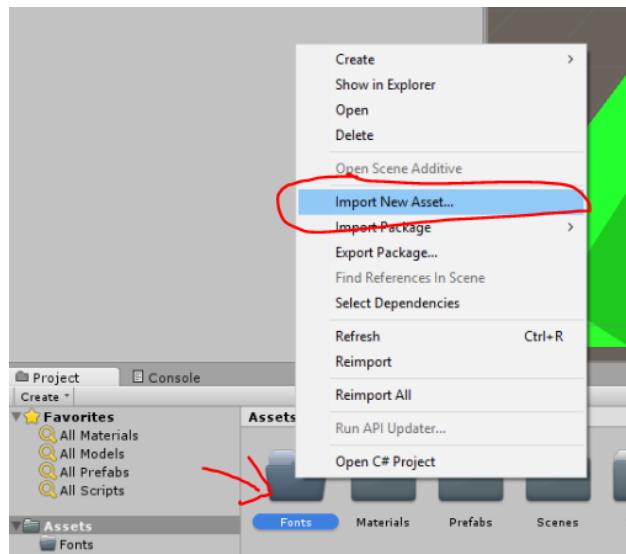


Parte 5 – Aggiunta della UI

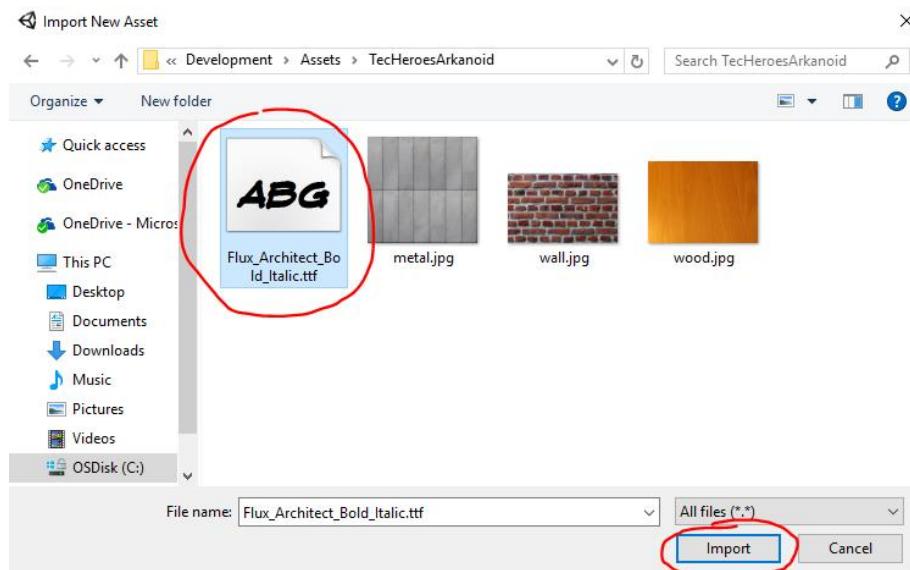
- 1) Creare una nuova cartella chiamata Fonts



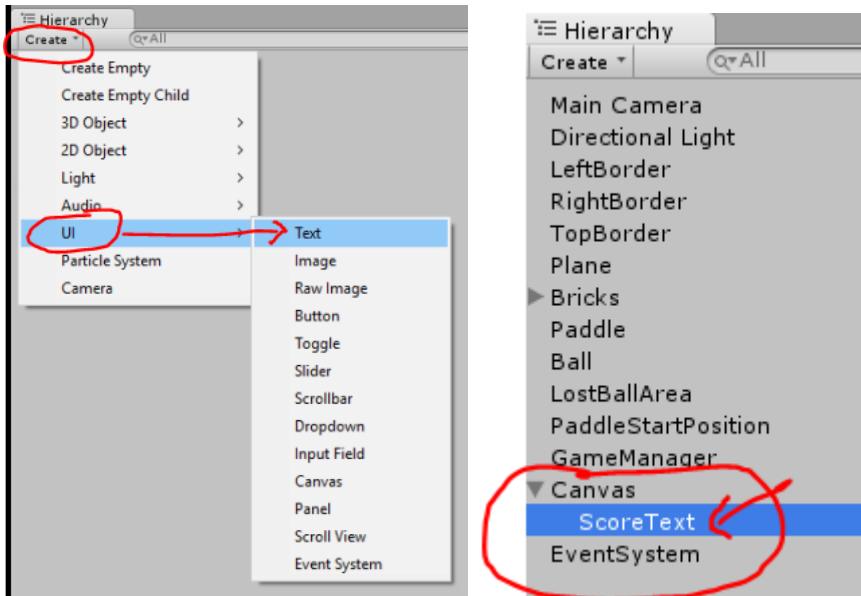
- 2) Cliccare sulla cartella con il tasto destro e selezionare Import New Asset



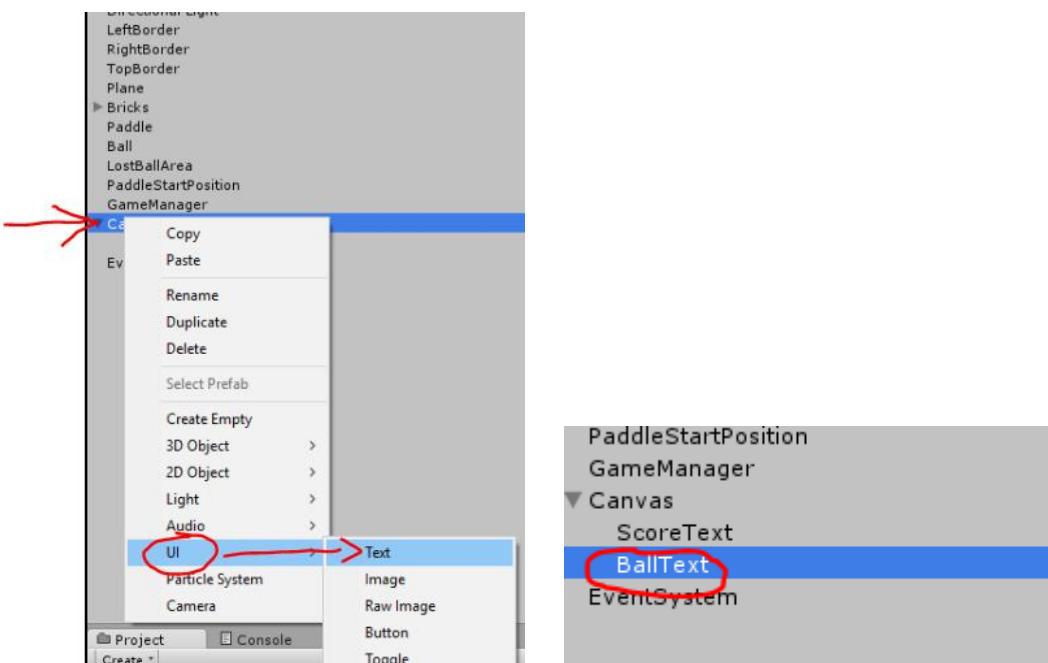
- 3) Selezionare il file Flux_Architect_Bold_Italic.ttf dalla cartella degli assets di questo laboratorio e cliccare su Import



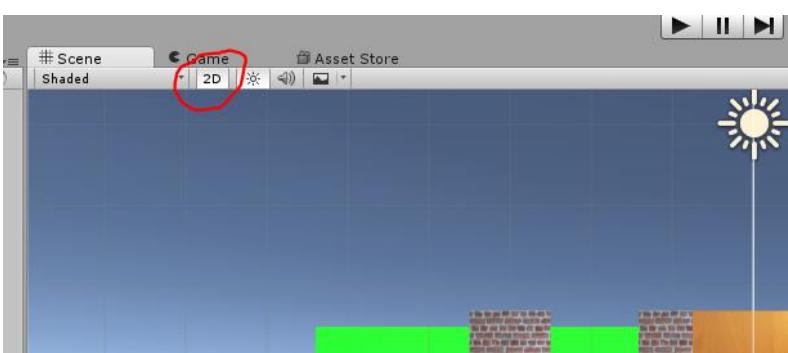
- 4) Creare un nuovo oggetto **Text** cliccando su **Create→UI→Text** e rinominarlo in **ScoreText**. (Oltre al testo saranno aggiunti automaticamente alla scena anche gli oggetti **Canvas** e **EventSystem**)



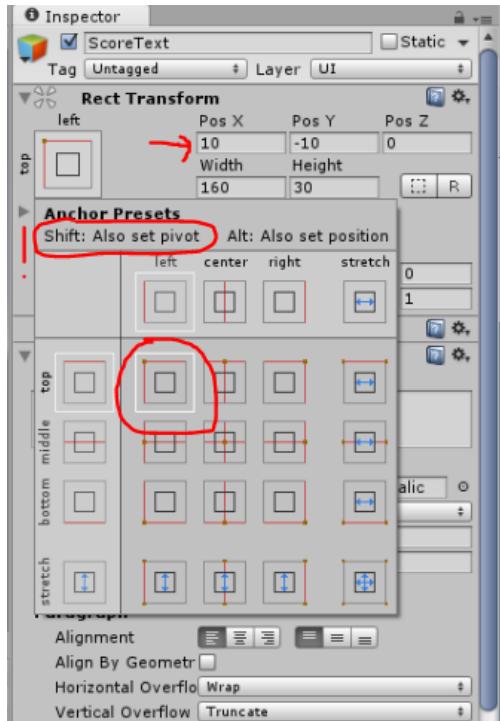
- 5) Cliccare con il tasto destro sull'oggetto **Canvas** e selezionare ancora **UI→Text** per aggiungere un altro testo al canvas che chiameremo **BallText**



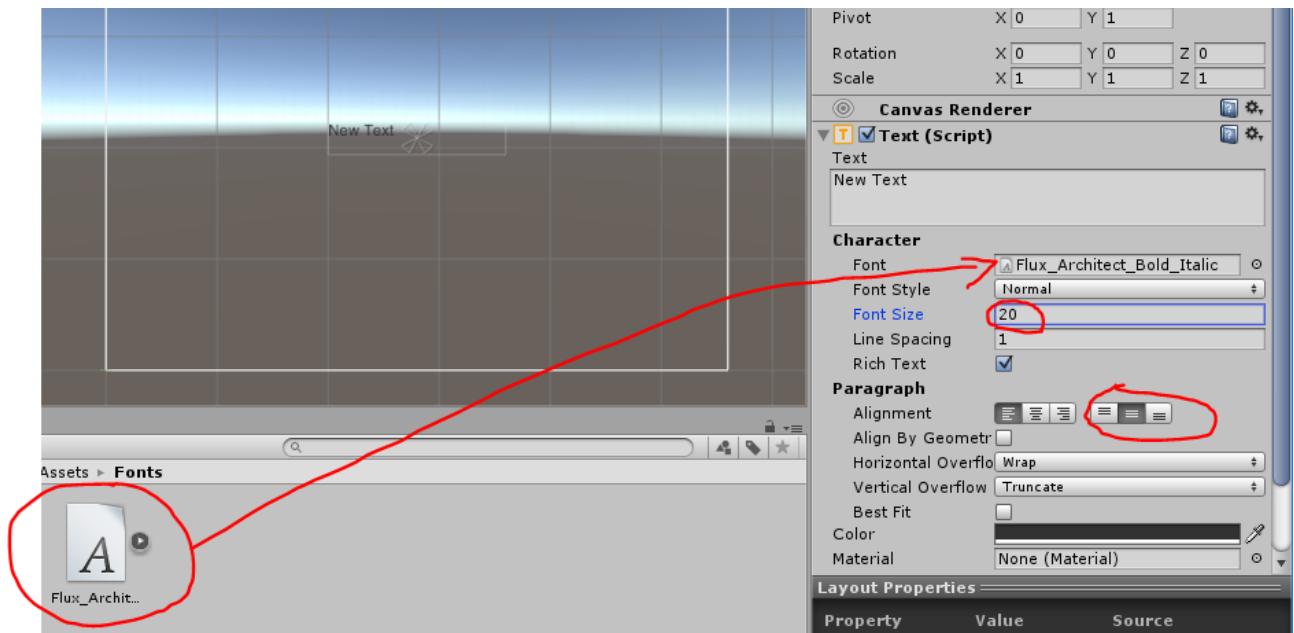
- 6) Sulla toolbar del pannello **Scene** selezionare la modalità **2D** cliccando sull'apposito pulsante



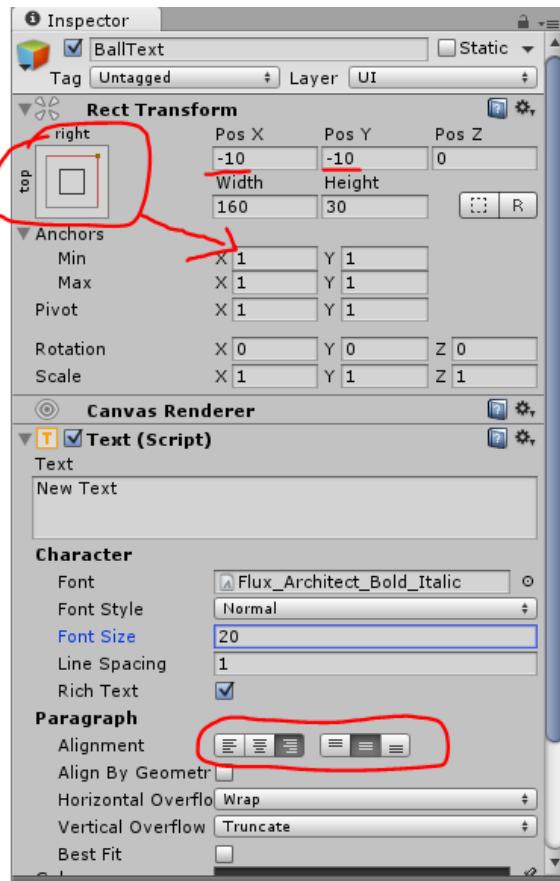
- 7) Fare doppio-click sull'oggetto **ScoreText** per passare rapidamente alla visualizzazione del canvas della UI. Cliccare sul bottone dell'ancoraggio nel componente **Rect Transform** e, tenendo premuto il tasto Shift, selezionare quello in alto a sinistra. Poi impostare i valori di X e Y come mostrato in figura



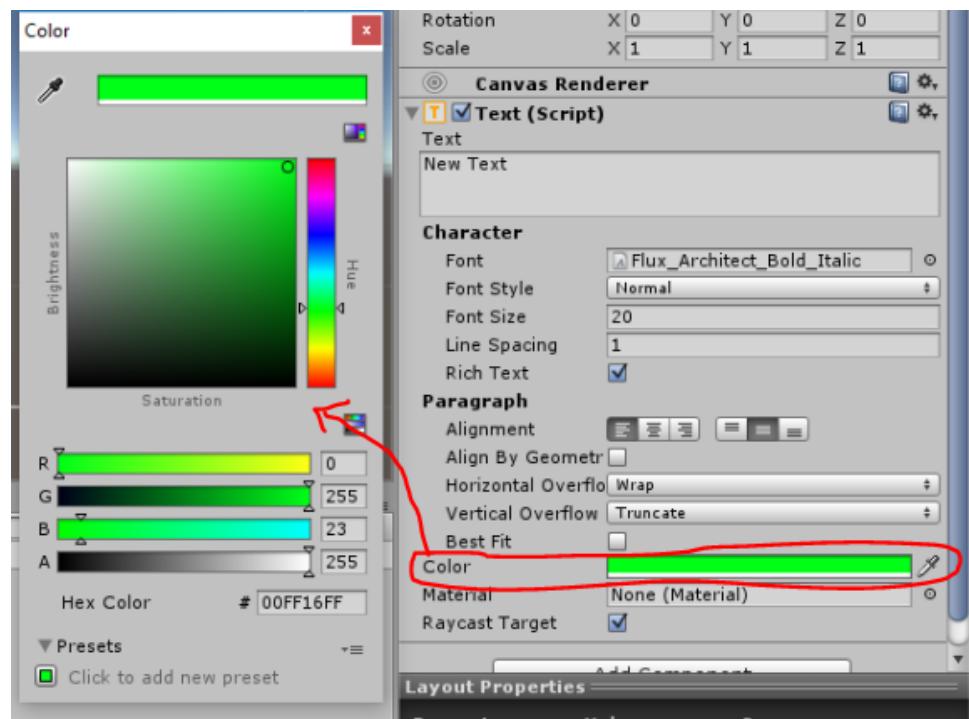
- 8) Impostare il font trascinandolo sulla relativa proprietà e impostare i parametri come da figura



9) Ripetere i passaggi 7 e 8 anche per l'oggetto BallText, ma impostando i seguenti valori



10) Modificare il colore dei 2 testi a piacimento, cliccando sulla proprietà **Color** del componente **Text**



11) Aprire lo script GameManagerScript in Visual Studio e aggiungere le righe evidenziate all'inizio della classe

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class GameManagerScript : MonoBehaviour
{
    //riferimento all'oggetto Paddle
    public GameObject paddle;
    //riferimento alla posizione di partenza del paddle
    public Transform paddleStartPosition;
    //riferimento all'oggetto ScoreText
    public Text scoreText;
    //riferimento all'oggetto BallText
    public Text ballText;

    //variabili per la gestione dei punti e delle palle
    private int currentScore = 0;
    private int balls = 3;
```

12) In fondo allo script (prima dell'ultima parentesi graffa che chiude la classe) aggiungere i seguenti metodi

```
//Metodo per incrementare il punteggio
public void AddScore(int score)
{
    //incremento la variabile del punteggio
    currentScore += score;
    //aggiorno il testo a video
    scoreText.text = string.Format("Score: {0}", currentScore);
}

//Metodo per decrementare le palle a disposizione
public void LoseBall()
{
    //decremento la variabile delle palle
    balls--;
    //aggiorno il testo a video
    ballText.text = string.Format("Balls: {0}", balls);
    //reimposto il paddle nella posizione iniziale
    ResetPaddle();
}
```

13) Infine modificare il metodo InitGame in questo modo

```
//metodo che inizializza gli oggetti
public void InitGame()
{
    ResetPaddle();
    //imposto il numero delle palle a 4 per comodità (vedi riga sotto)
    balls = 4;
    //Richiamo il metodo LoseBall per aggiornare il testo a video
    LoseBall();

    //azzerò il punteggio
    currentScore = 0;
    //aggiorno il testo a video richiamando il metodo e passando 0
    AddScore(0);
}
```

14) Salvare lo script e tornare in Unity

15) Selezionare l'oggetto **GameManager** e trascinare gli oggetti **ScoreText** e **BallText** nelle relative proprietà del componente **Game Manager Script**



16) Aprire lo script **LostBallScript** in Visual Studio e modificare la riga richiamando il metodo LoseBall al posto di InitGame

```
//evento che viene scatenato dal Box Collider impostato come trigger
//quando un altro collider entra nella zona di collisione
void OnTriggerEnter(Collider coll)
{
    //se il collider è quello dell'oggetto Ball ricomincio
    if (coll.gameObject.name == "Ball")
        GameManagerScript.singleton.LoseBall();
}
```

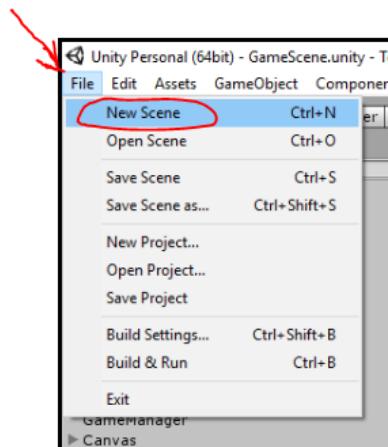
17) Infine aprire lo script **BrickScript** e aggiungere la riga evidenziata nel metodo OnCollisionEnter

```
public void OnCollisionEnter(Collision collision)
{
    //verifica se l'oggetto entrato in collisione è la palla
    if (collision.gameObject.name == "Ball")
    {
        //Aggiorno il punteggio
        GameManagerScript.singleton.AddScore(10);
        //se si tratta della palla,
        //distrugge il GameObject (ovvero il Brick che è stato colpito)
        Destroy(gameObject);
    }
}
```

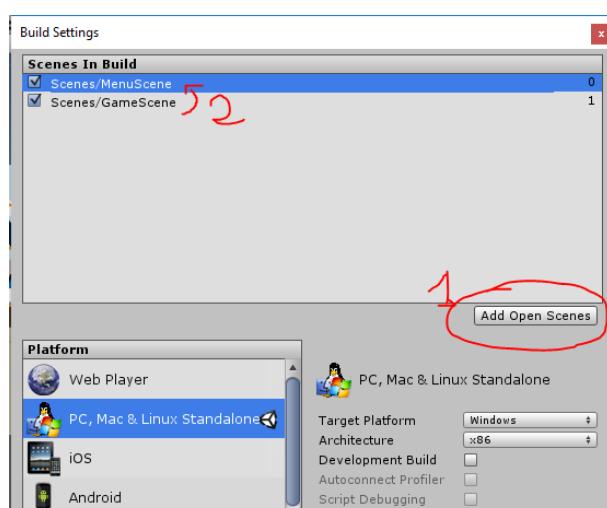
18) Salvare tutto e tornare in Unity

Parte 6 – Creazione del menù

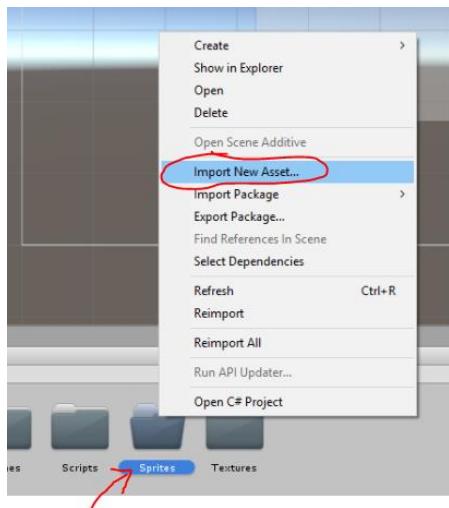
1) Creare una nuova scena cliccando su **File→New Scene**



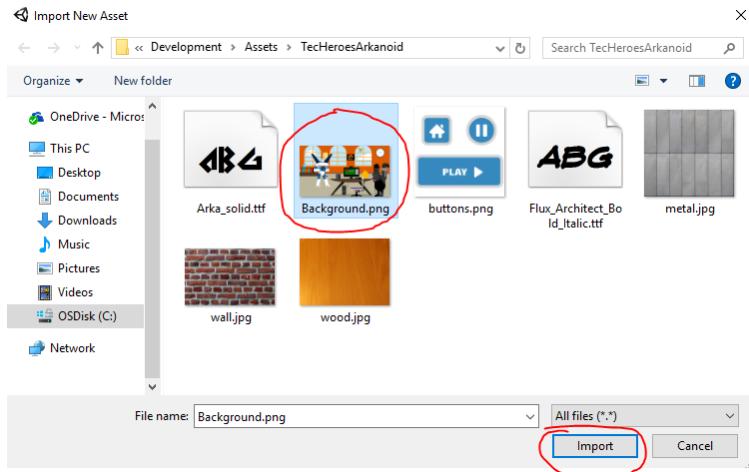
- 2) Salvare la scena nella cartella Scenes (**File→Save Scene as...**) e chiamarla **MenuScene**
3) Aprire la finestra Build Settings (**File→Build Settings...**), cliccare sul bottone **Add Open Scenes** e trascinare la nuova scena al primo posto della sequenza di build. Infine chiudere la finestra.



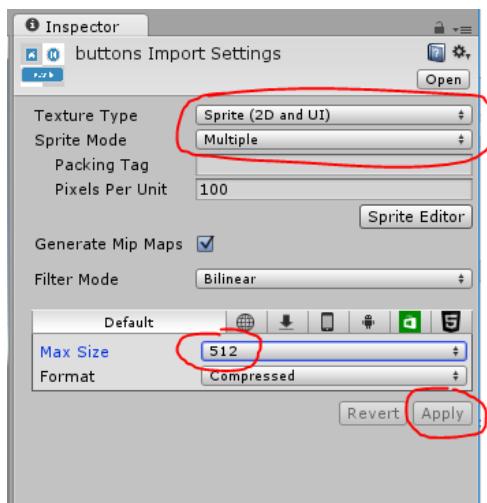
- 4) Nel pannello **Project** creare una nuova cartella chiamata **Sprites**
- 5) Cliccare con il tasto destro sulla nuova cartella e selezionare **Import New Asset...**



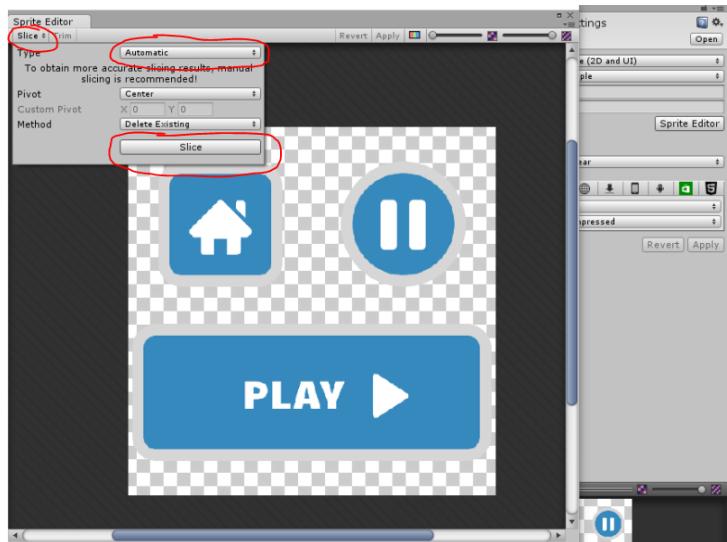
- 6) Selezionare dalla cartella degli assets del laboratorio il file **Background.png** e cliccare su **Import**



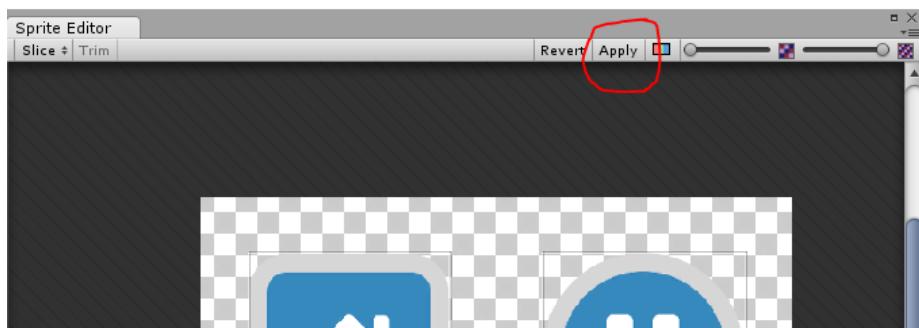
- 7) Ripetere il punto 4 anche per **buttons.png**
- 8) Ripetere di nuovo il punto 4 ma sulla cartella Fonts, e importare il file **Arka_solid.ttf**
- 9) Nel pannello Project selezionare il file **buttons.png** e impostare le proprietà nell'Inspector come mostrato in figura dopo di che cliccare su **Apply**



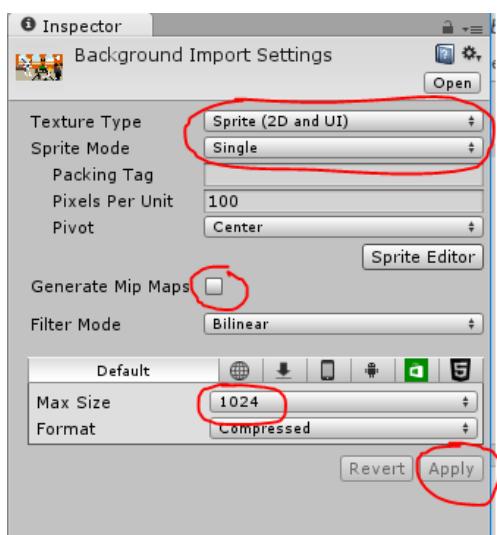
10) Cliccare su **Sprite Editor** per aprire il relativo editor. Cliccare sul menù **Slice** e selezionare **Automatic**. Infine premere sul pulsante **Slice**



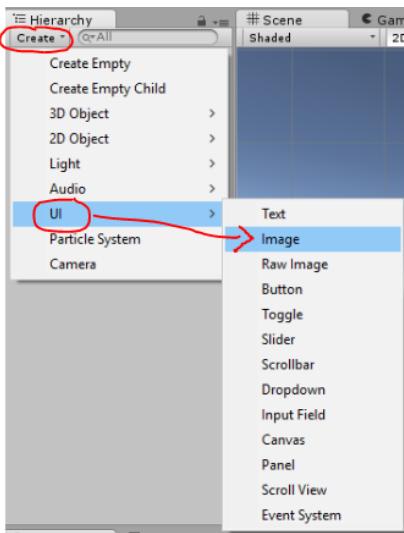
11) Confermare cliccando su **Apply** e chiudere la finestra dell' editor



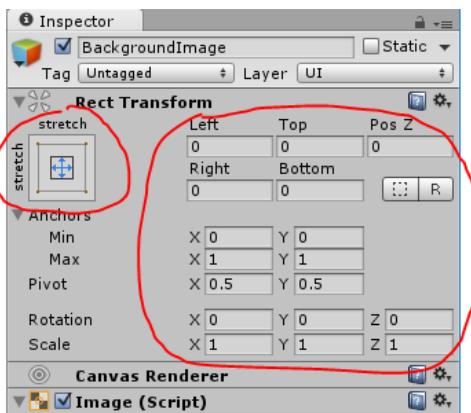
12) Selezionare il file **Background.png** e impostare le proprietà come in figura. Confermare cliccando su **Apply**



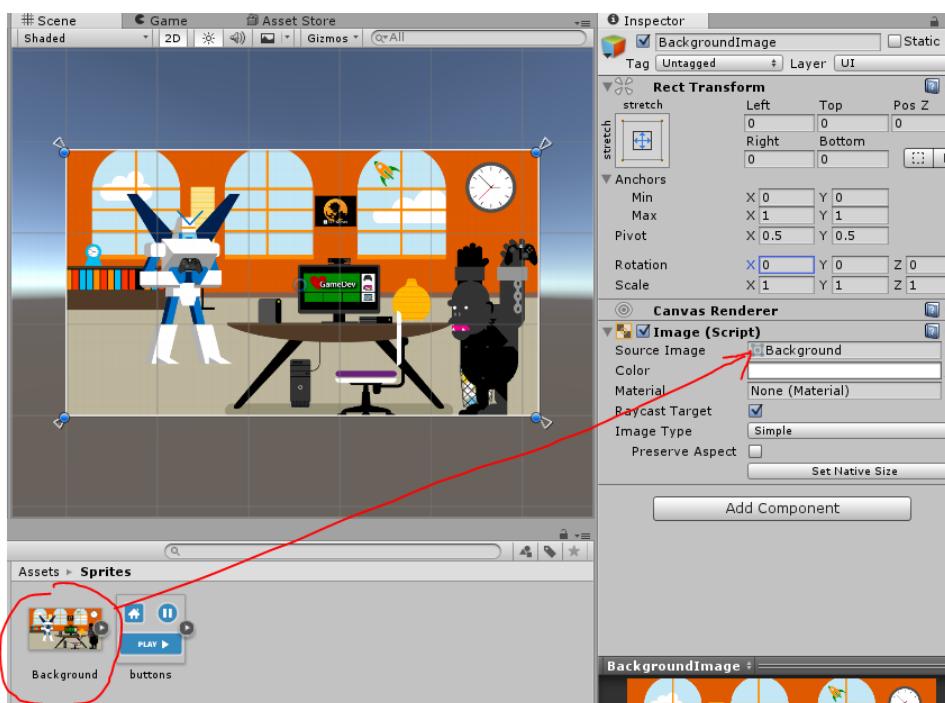
13) Nella Hierarchy creare un nuovo oggetto **Image** cliccando su **Create→UI→Image** e chiamarlo **BackgroundImage**



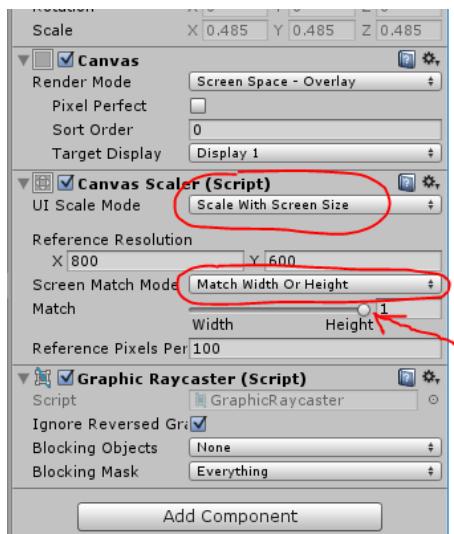
14) Impostare il suo componente **Rect Transform** come indicato in figura



15) Trascinare il **Background** nella proprietà **Source Image** del componente **Image**



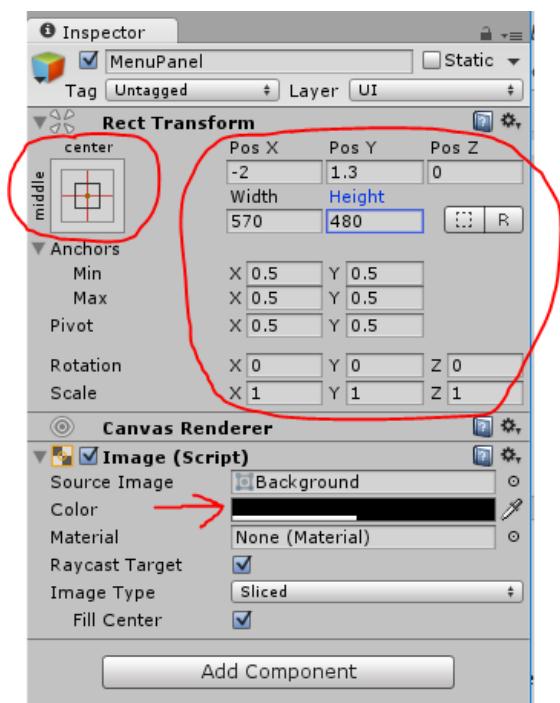
16) Selezionare l'oggetto **Canvas** e impostare il suo componente **Canvas Scaler** come indicato in figura



17) Creare un pannello cliccando con il tasto destro sull'oggetto **Canvas** e selezionando

Create→UI→Panel. Chiamare il nuovo bottone **MenuPanel**

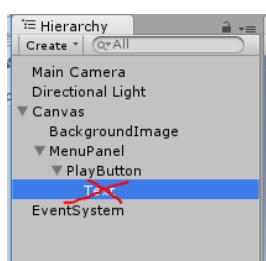
18) Impostare le proprietà del pannello come mostrato in figura



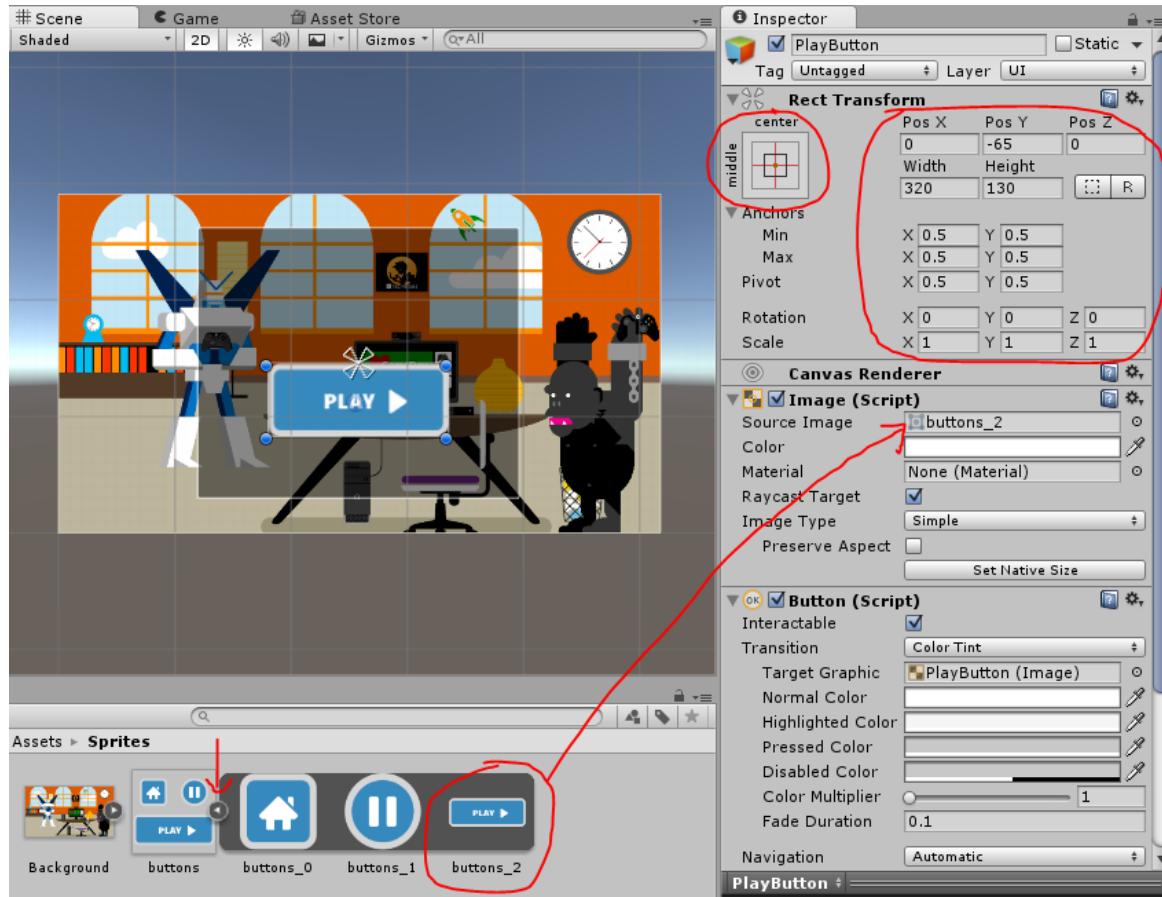
19) Creare un nuovo Bottone cliccando con il tasto destro sull'oggetto **MenuPanel** e selezionando

Create→UI→Button. Chiamare il nuovo bottone **PlayButton**

20) Espandere l'oggetto **PlayButton** ed eliminare l'oggetto **Text** al suo interno premendo il tasto **Canc** della tastiera

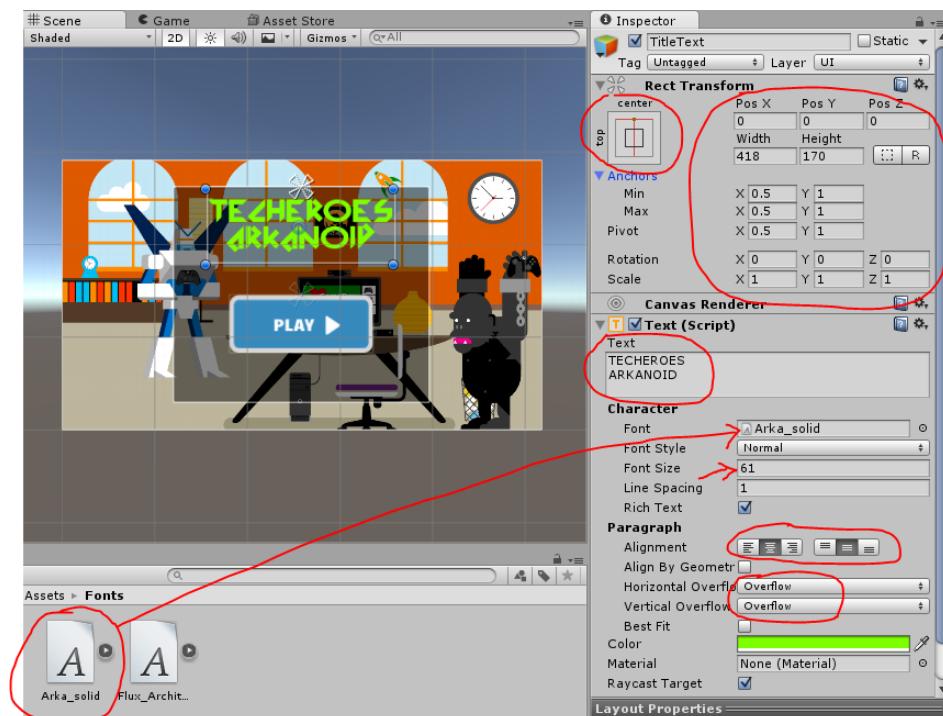


21) Selezionare **PlayButton** e trascinare l'immagine del bottone **Play** nella proprietà **Source Image** del componente **Image**. Dopo di che impostare i valori dei componenti come illustrato in figura

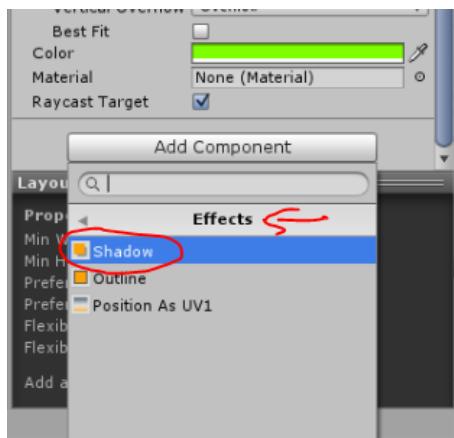


22) Creare un nuovo Testo cliccando con il tasto destro sull'oggetto **MenuPanel** e selezionando **Create→UI→Text**. Chiamare il nuovo testo **TitleText**

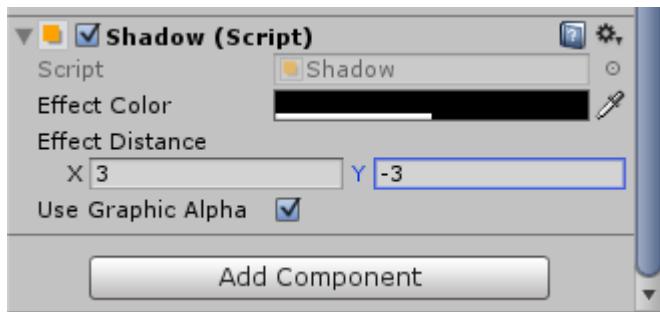
23) Selezionare **TitleText** e trascinare il font **Arka_solid** nella proprietà **Font**. Poi impostare i valori come in figura



24) Aggiungere un componente **Shadow** al testo cliccando sul bottone **Add Component** e selezionando **UI→Effects→Shadow**



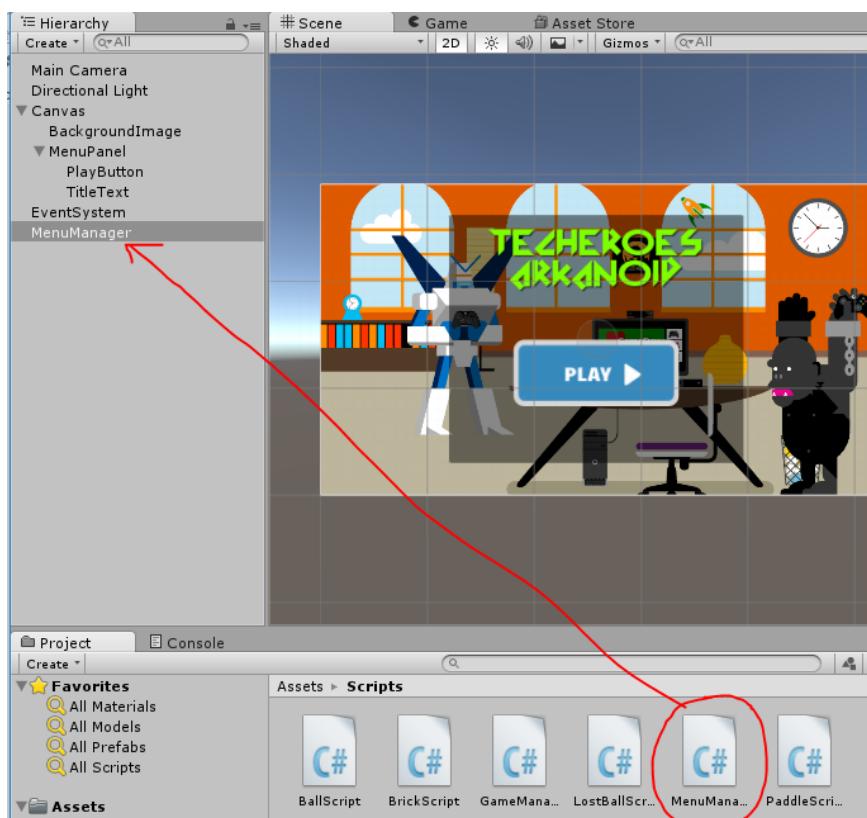
25) Impostare i valori come in figura



26) Creare un GameObject vuoto (Create→Create Empty) e chiamarlo **MenuManager**

27) Creare un nuovo script come illustrato nelle sezioni precedenti e chiamarlo **MenuManagerScript**

28) Trascinare lo script **MenuManagerScript** sull'oggetto **MenuManager** nella Hierarchy



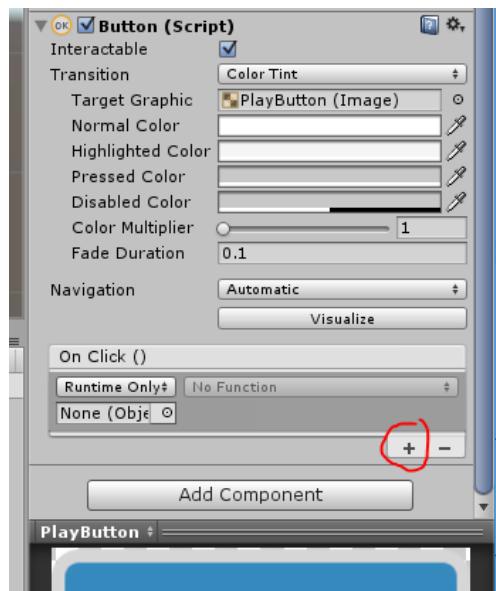
29) Aprire lo script in Visual Studio e aggiungere il metodo Play

```
public void Play()
{
    UnityEngine.SceneManagement.SceneManager.LoadScene("GameScene");
}
```

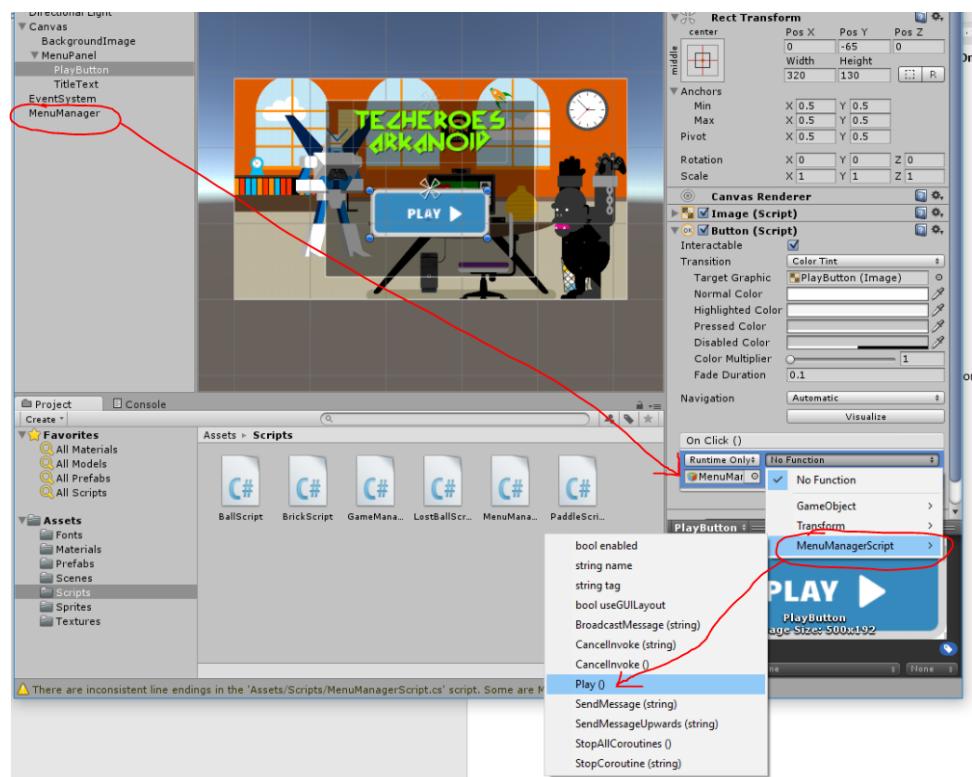
30) Salvare il file e tornare in Unity

31) Salvare la scena (Ctrl+S)

32) Selezionare l'oggetto **PlayButton** e aggiungere un handler per l'evento **OnClick** cliccando sul simbolo '+' nel componente Button

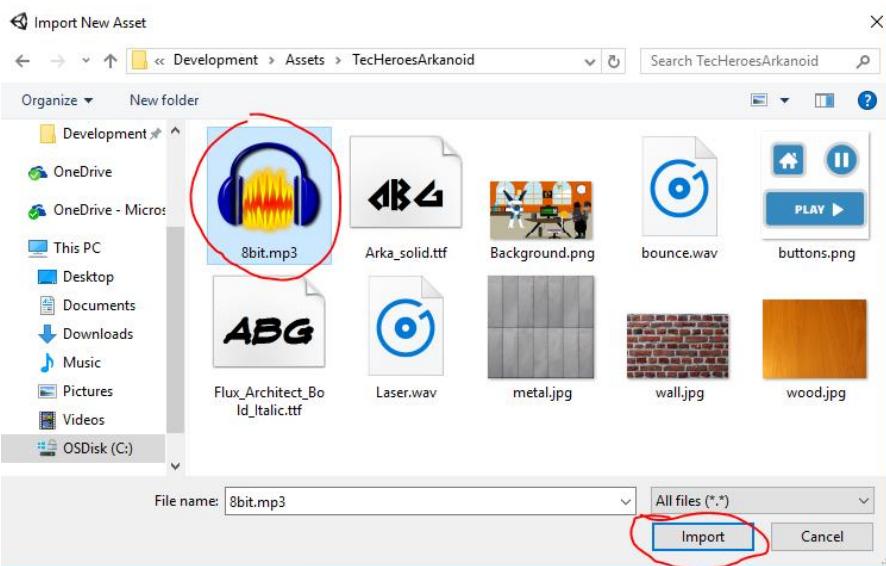


33) Trascinare l'oggetto **MenuManager** dalla Hierarchy all'interno del componente. Dopo di che selezionare il metodo Play dal menù a tendina (**MenuManagerScript**→**Play()**)

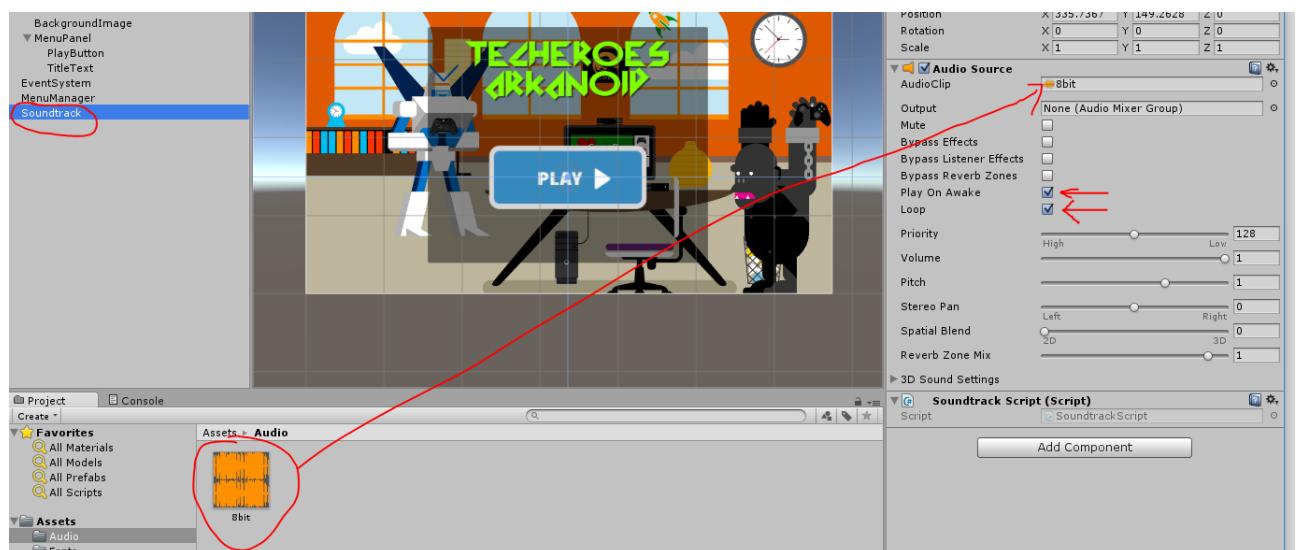


Parte 7 – Musica ed effetti sonori

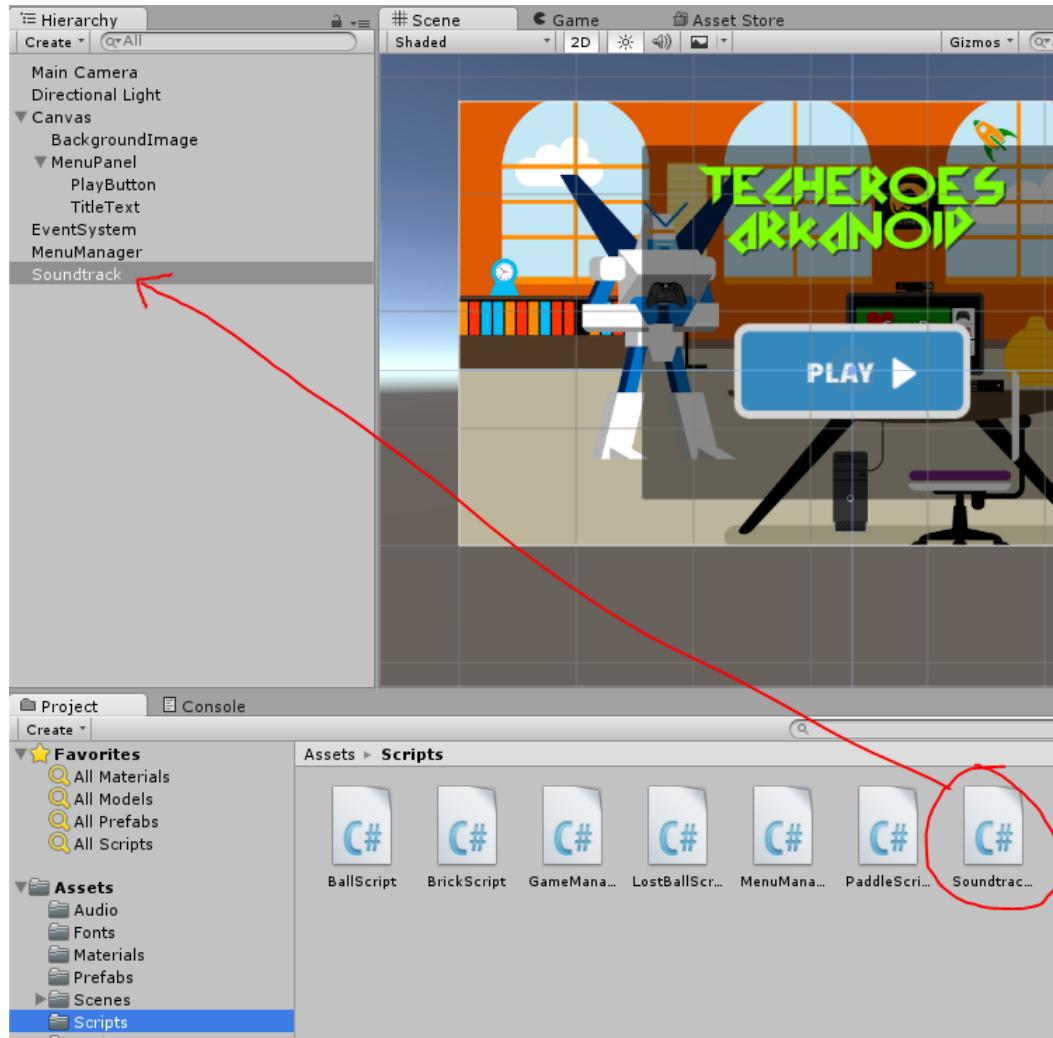
- 1) Nel pannello Project creare una nuova cartella **Audio**
- 2) Cliccare con il tasto destro sulla cartella **Audio** e selezionare **Import new Asset**
- 3) Importare il file **8bit.mp3** dalla cartella degli assets di questo laboratorio



- 4) Creare un GameObject vuoto e chiamarlo **Soundtrack**
- 5) Selezionare **Soundtrack** e trascinare il file **8bit.mp3** nella proprietà **Audio Clip** del componente **Audio Source**. Infine selezionare il flag **Loop**



- 6) Creare un nuovo script chiamato **SoundtrackScript** e trascinarlo sull'oggetto **Soundtrack**



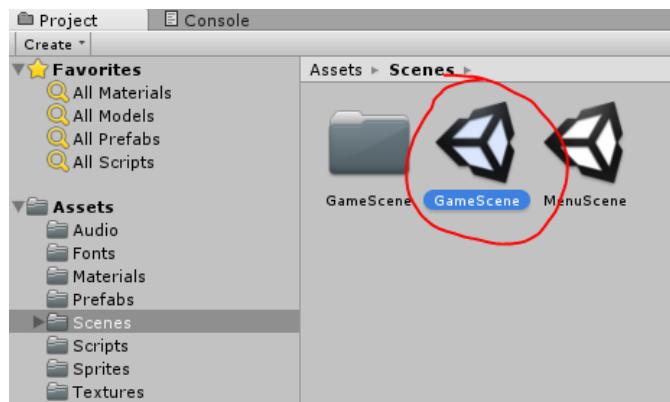
- 7) Aprire lo script in Visual Studio e modificarlo come segue

```
public class SoundtrackScript : MonoBehaviour
{
    public static SoundtrackScript singleton = null;

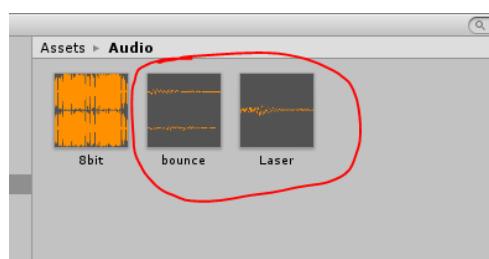
    void Awake()
    {
        if (singleton != null && singleton != this)
        {
            Destroy(gameObject);
            return;
        }
        else
        {
            singleton = this;
        }
        DontDestroyOnLoad(gameObject);
    }
}
```

- 8) Salvare lo script e tornare in Unity
 9) Salvare la scena corrente (Ctrl+S)

10) Aprire la scena **GameScene** facendo doppio click sul relativo asset nel pannello Project



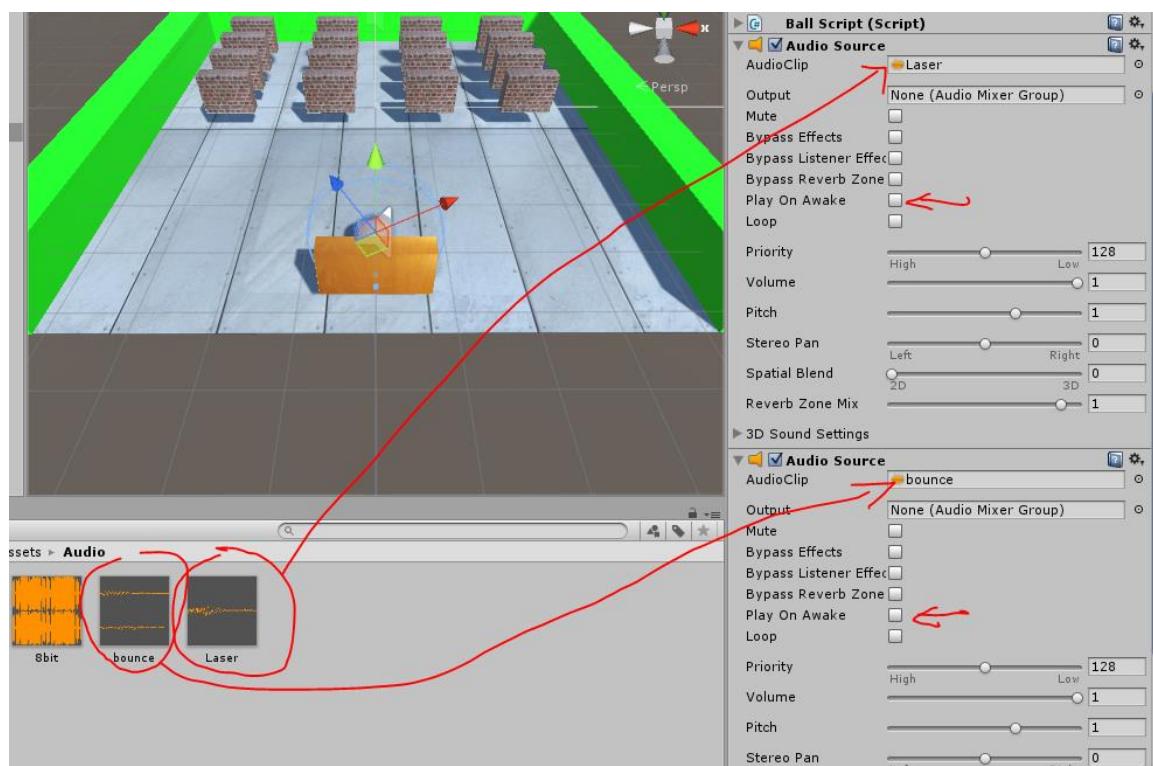
11) Ripetere i passi 2 e 3 per importare i files **laser.wav** e **bounce.wav** dalla cartella degli assets del laboratorio



12) Selezionare Ball e aggiungere 2 componenti di tipo Audio Source cliccando sul bottone **Add Component**→**Audio**→**Audio Source**

13) Trascinare il file **laser.wav** nella proprietà **Audio Clip** del primo componente **Audio Source**. Dopo di che disabilitare i flag **Play on Awake** e **Loop**

14) Ripetere il punto 13 trascinando il file **Bounce.wav** nel secondo componente



15) Aprire lo script **BallScript** in Visual Studio e aggiungere 2 variabili di tipo **AudioSource**

```
public class BallScript : MonoBehaviour
{
    //proprietà gestibile dall'Inspector per modificare la velocità
    //della palla
    public float speed = 8f;

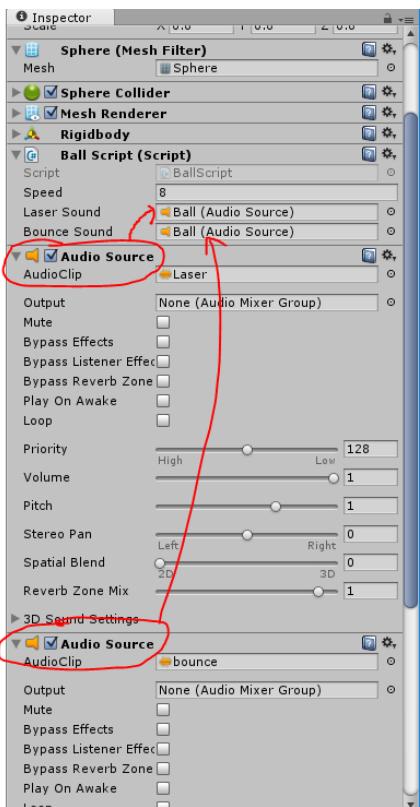
    //riferimenti ai componenti Audio Source della palla
    public AudioSource laserSound;
    public AudioSource bounceSound;
```

16) Aggiungere poi il metodo OnCollisionEnter

```
public void OnCollisionEnter(Collision collision)
{
    //se collide con un Brick riproduco il suono
    if (collision.gameObject.name.Contains("Brick"))
        laserSound.Play();
    else
        //altrimenti riproduco l'altro
        bounceSound.Play();
}
```

17) Salvare lo script e tornare in Unity

18) Selezionare l'oggetto **Ball** e trascinare i componenti **AudioSource** nelle relative proprietà del componente **Ball Script**



- 19) Salvare la scena (Ctrl+S) e avviare il gioco per testare il corretto funzionamento dell'audio

Parte 8 – Ritocchi finali e creazione applicazione UWP

- 1) Aprire lo script GameManagerScript in Visual Studio e aggiungere una variabile di tipo GameObject chiamata menuPanel

```
public class GameManagerScript : MonoBehaviour
{
    //riferimento all'oggetto Paddle
    public GameObject paddle;
    //riferimento alla posizione di partenza del paddle
    public Transform paddleStartPosition;
    //riferimento all'oggetto ScoreText
    public Text scoreText;
    //riferimento all'oggetto BallText
    public Text ballText;
    //riferimento al pannello dei pulsanti
    public GameObject menuPanel;
```

- 2) Aggiungere la riga evidenziata nel metodo Start

```
//metodo richiamato automaticamente da Unity prima del primo ciclo di Update
void Start()
{
    menuPanel.SetActive(false);
    //richiama la funzione di inizializzazione del livello
    InitGame();
}
```

- 3) Aggiungere in fondo alla classe questi 3 nuovi metodi

```
void GameOver()
{
    menuPanel.SetActive(true);
}
public void Restart()
{
    UnityEngine.SceneManagement.SceneManager.LoadScene("GameScene");
}
public void Home()
{
    UnityEngine.SceneManagement.SceneManager.LoadScene("MenuScene");
}
```

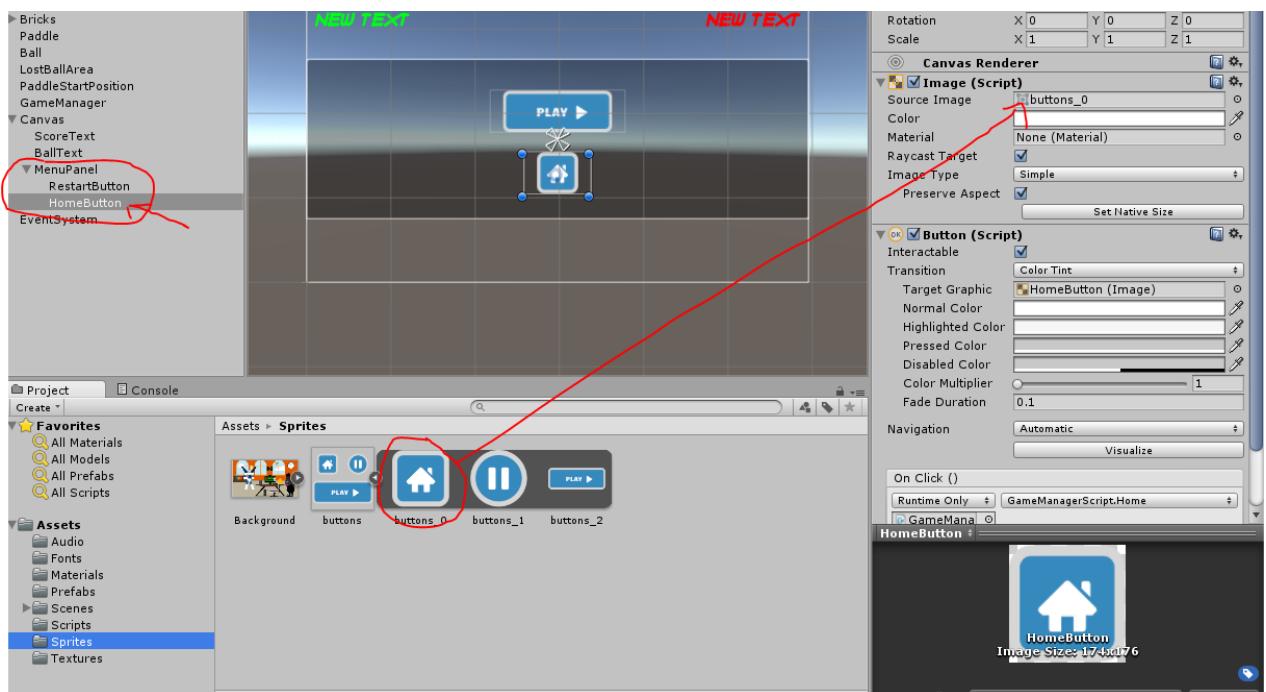
- 4) Modificare il metodo **LoseBall** come segue

```
//Metodo per decrementare le palle a disposizione
public void LoseBall()
{
    //decremento la variabile delle palle
    balls--;
    //aggiorno il testo a video
    ballText.text = string.Format("Balls: {0}", balls);

    if (balls < 0)
        GameOver();
    else
        //reimposta il paddle nella posizione iniziale
        ResetPaddle();
}
```

- 5) Salvare lo script e ritornare in Unity

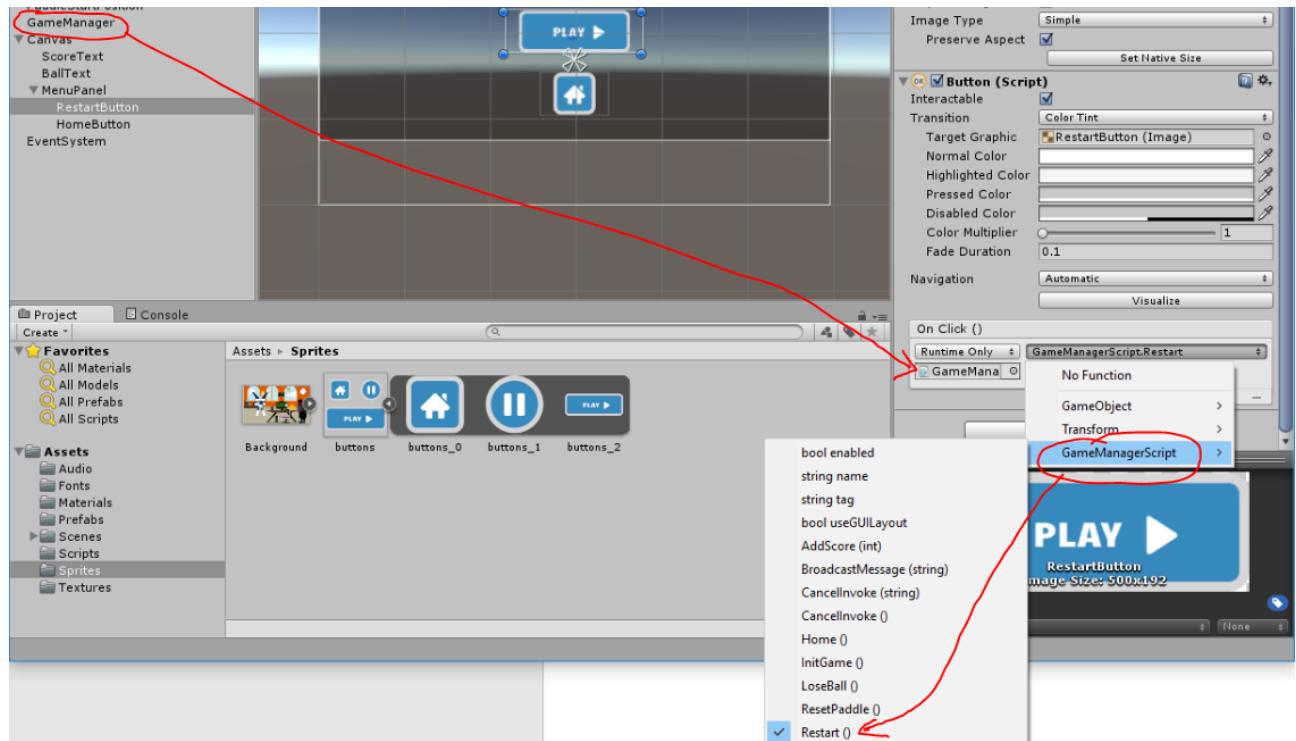
- 6) Creare un pannello con 2 bottoni esattamente come fatto in precedenza nella scena **MenuScene**, con la sola differenza che in questo caso c'è un secondo bottone da impostare con l'immagine della Home (chiamare i bottoni **RestartButton** e **HomeButton**)



- 7) Selezionare l'oggetto **GameManager** e trascinare il pannello nella proprietà **Menu Panel** del componente **Game Manager Script**



- 8) Selezionare il bottone **RestartButton** e aggiungere un handler per l'evento **OnClick**. Traslinare l'oggetto **GameManger** e selezionare dalla tendina il metodo **Restart()**

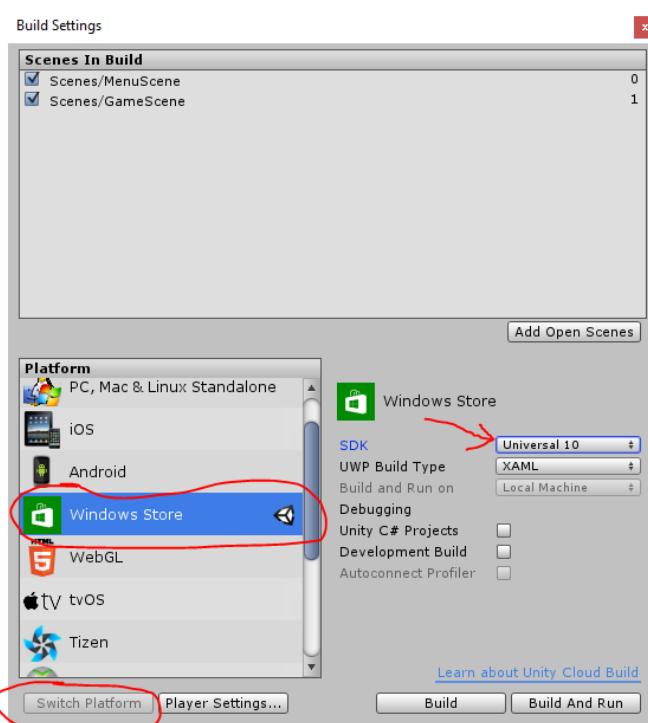


- 9) Ripetere il punto 8 anche per il bottone **HomeButton**, ma selezionando il metodo **Home()**

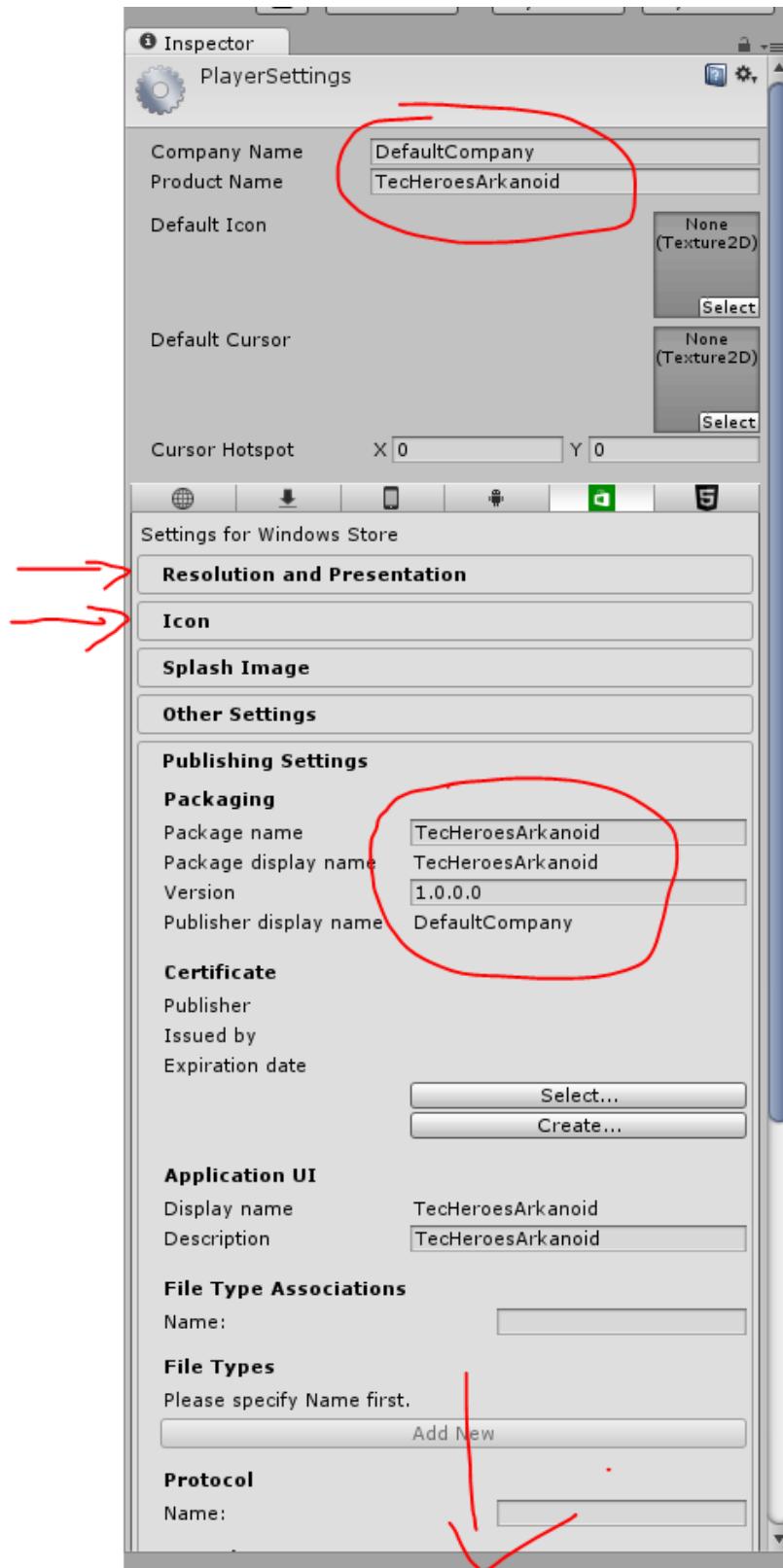
10) Salvare la scena (Ctrl+S) e ritornare alla scena **MenuScene**

11) Eseguire il gioco per testare il funzionamento completo prima di generare il progetto UWP

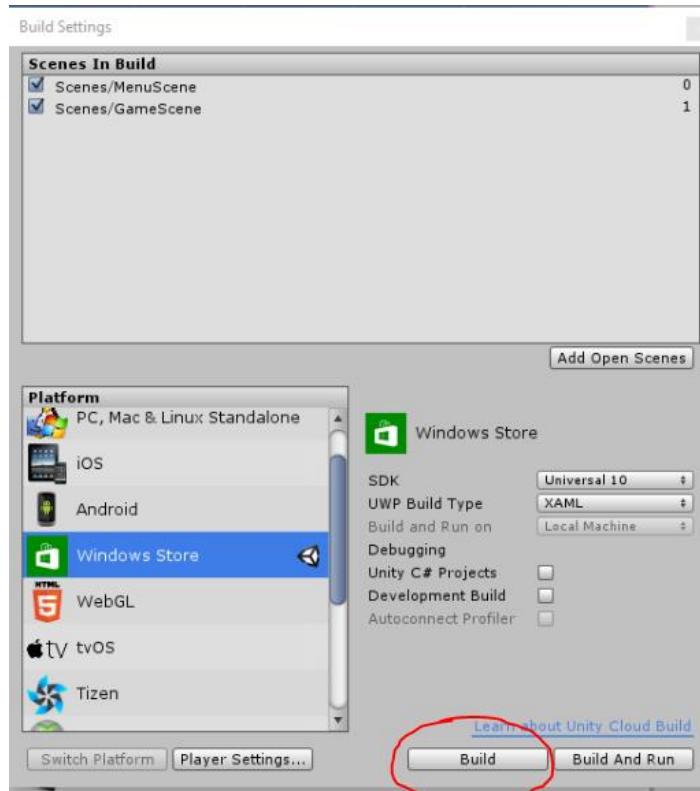
- 12) Aprire i **Build Settings** dal menù **File**, selezionare **Windows Store** dall'elenco delle Platform, impostare l'SDK a Universal 10 e cliccare su **Switch Platform**



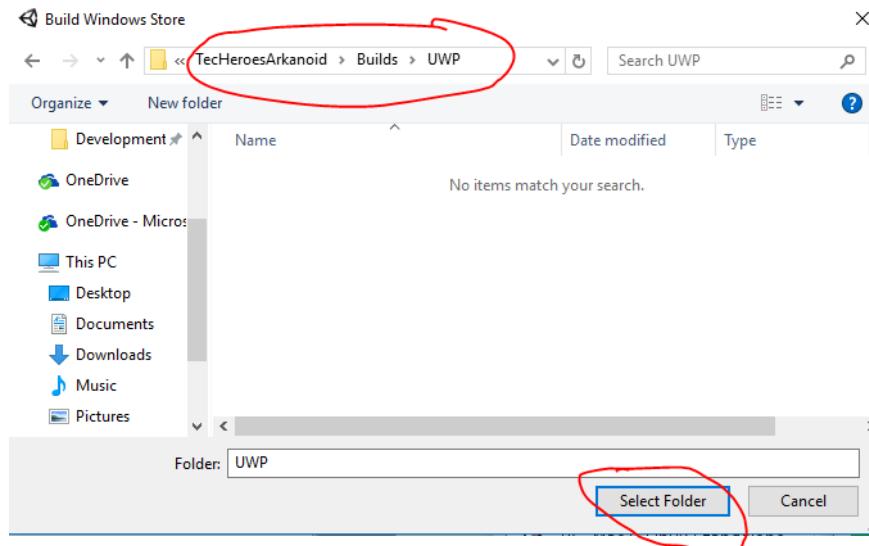
13) Cliccare sul pulsante **Player Settings...** e impostare le proprietà relative alle applicazioni UWP dal pannello Inspector (nome, publisher, icone, orientamento, capabilities ecc...)



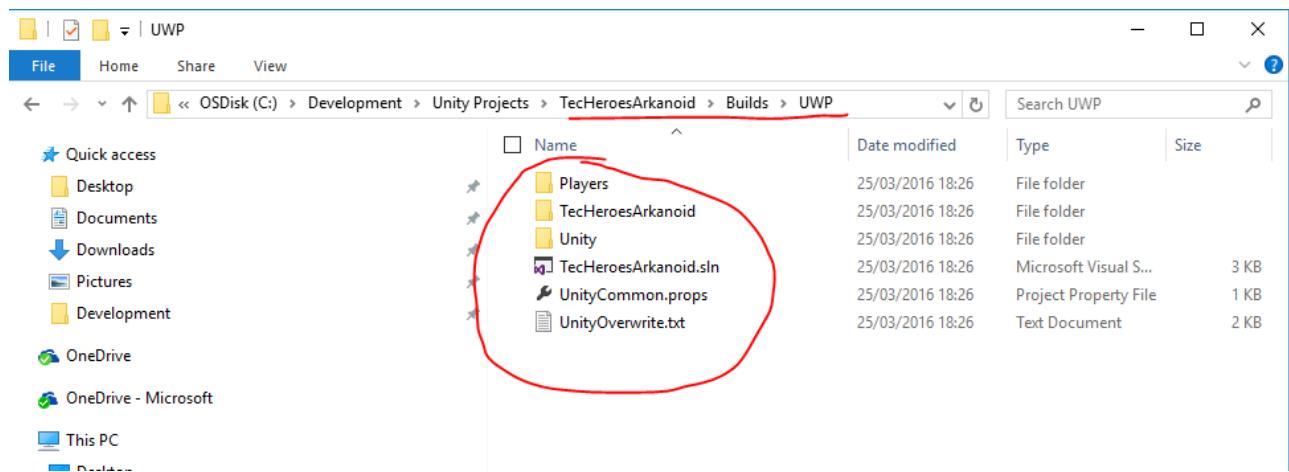
14) Una volta completato l'inserimento dei dati di pubblicazione cliccare sul pulsante **Build** per avviare il processo di generazione della soluzione Visual Studio per la UWP app



15) Verrà richiesto di indicare una cartella nella quale generare la solution



- 16) Una volta selezionata la cartella partirà la creazione di una nuova soluzione Visual Studio che vi permetterà di compilare il gioco come applicazione UWP nativa



FINE DEL LABORATORIO
