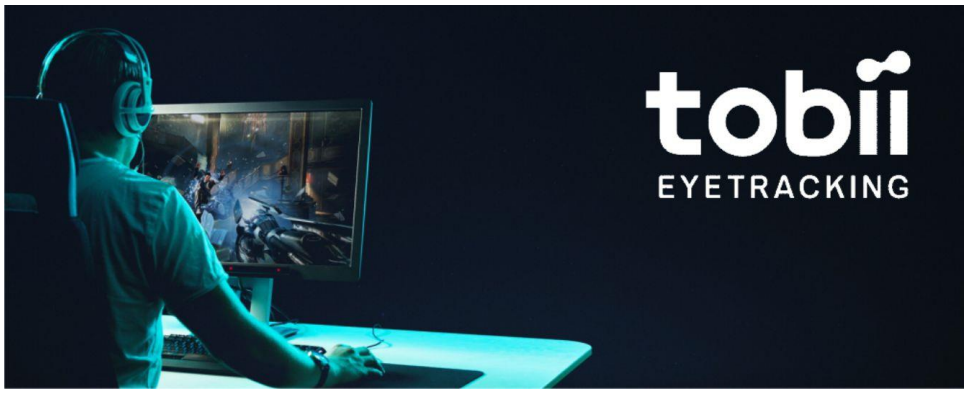
Tobii Eye Tracker

Úvod

Zariadenie Tobii Eye tracker slúži na zistenie, kam sa človek pozerá na monitore počítača či iného zariadenia pomocou infračerveného svetla, ktoré svieti priamo na človeka. V reálnom čase sa do počítača odosielajú informácie o polohe zrenice, vektore pohľadu pre každé oko a bod pohľadu. Vďaka tejto technológií dokážeme ovládať počítač bez použitia myši či klávesnice.



Tvorba prvej hry (Bludisko)

Tobii podporuje tvorbu hier napríklad v hernom frameworku Pygame či v hernom engine Unity/Unreal pomocou SDK (- súbor nástrojov pre vývoj softvéru). Pri tvorbe tejto hry použijeme Unity.

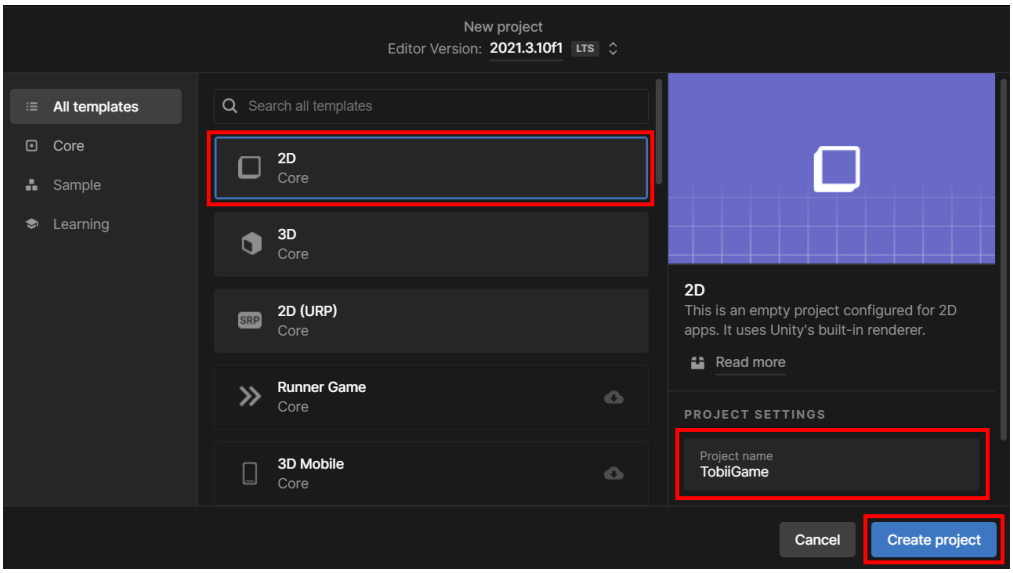
Čo budeme potrebovať:

- Unity 2019.4 alebo vyšší (odkaz na stiahnutie)
- Windows 10, 8.1 alebo 7
- Tobii Experience (odkaz na stiahnutie)
- Tobii Unity SDK (odkaz na stiahnutie)
- Samotný Tobii Eye Tracker 4C/5
- Balíček obrázkov ()

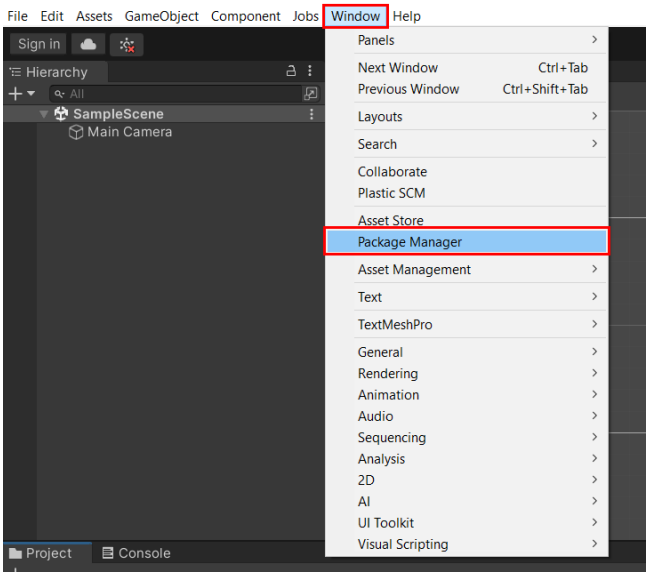
Príprava Unity a Eye Tracker

Po nainštalovaní Unity si v **Unity Hub** vytvoríme nový projekt

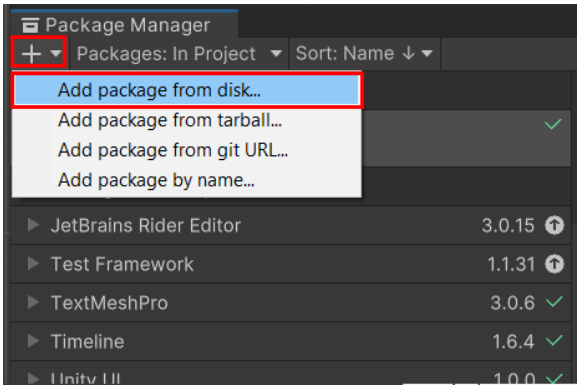
V zozname šablón zvolíme **2D**, projekt si pomenujeme a vytvoríme

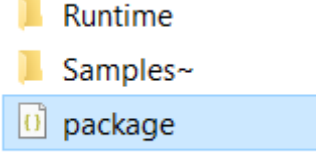


Po vytvorení projektu importujeme **Tobii SDK** cez *Windows -> Package Manager*

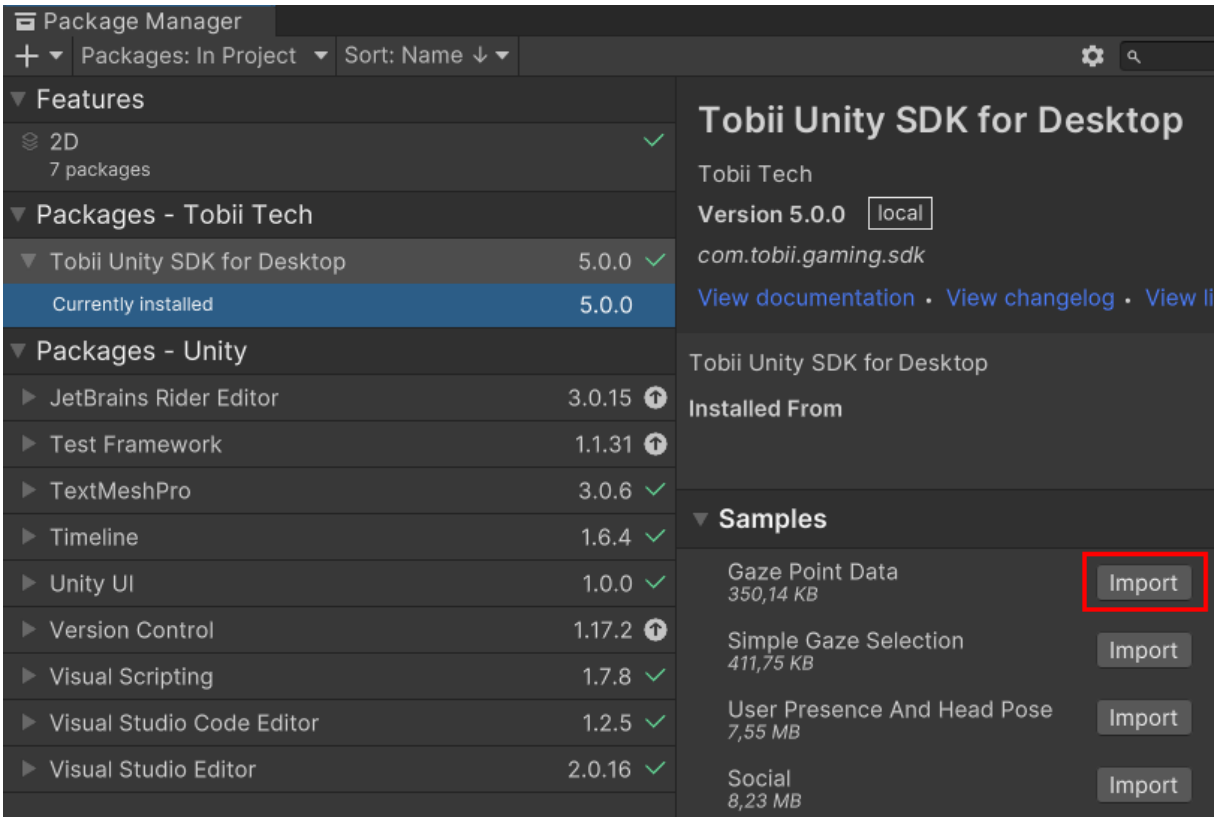


Kde cez **+** zvolíme možnosť pridania z disku a vyberieme **package.json** z rozbaleného **Tobii Unity SDK**, ktorý sme si stiahli na začiatku

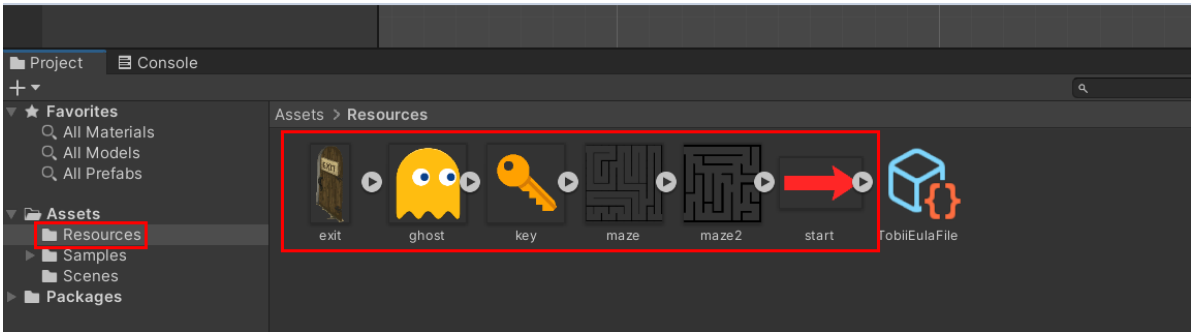




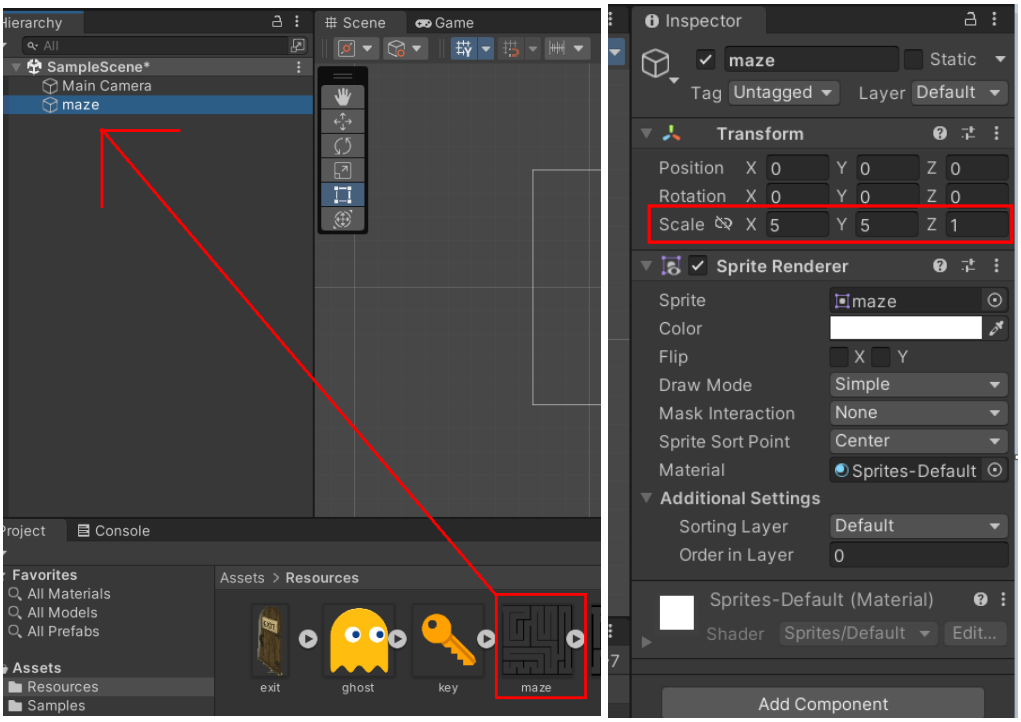
Importujeme **Gaze Point Data**



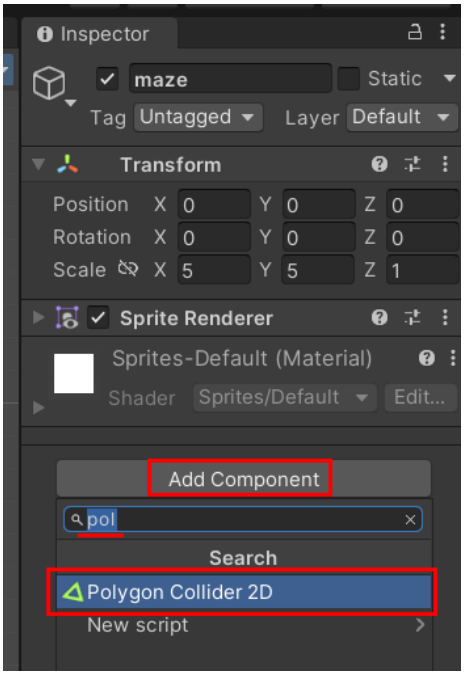
Pridáme obrázky do **Resources** (klasickým presunutím priamo do Unity), ktoré sme si stiahli v úvode



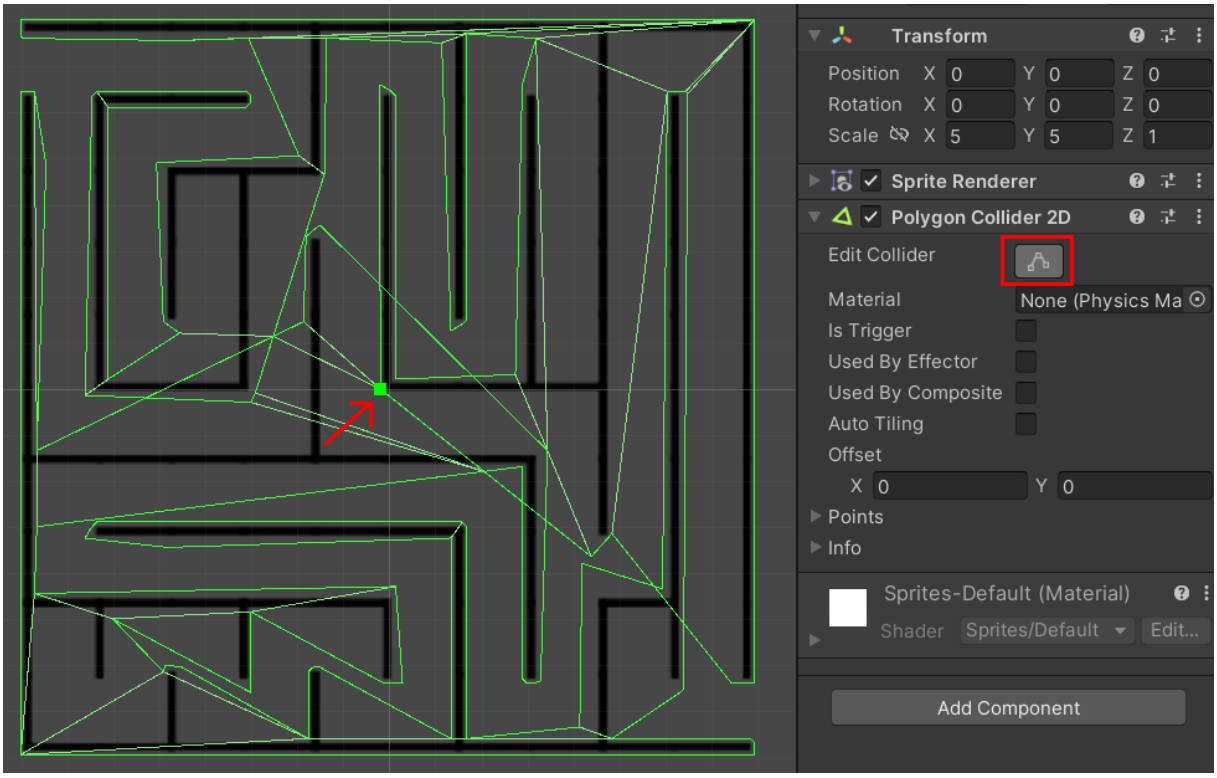
Pridanie bludiska do scény - presunieme obrázok s názvom ***maze*** do ľavej časti (Hierarchy), následne naň klikneme a v pravej časti (Inspector) nastavíme ***Scale*** na X = 5 a Y = 5, čím sa nám bludisko zväčší na presnú kocku



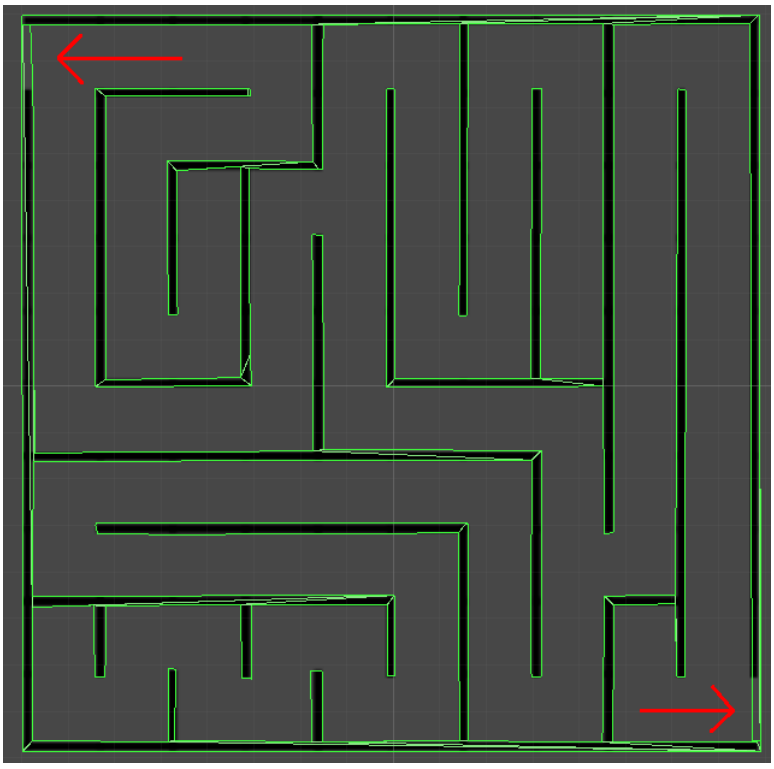
Pridanie bariér pre bludisko – do bludiska je potrebné pridať bariéry aby postavička nemohla chodiť cez steny. Cez tlačidlo **Add Component** pridáme *Polygon Collider 2D*, ktorý nám automaticky vygeneruje bariéry tam kde sú steny bludiska.



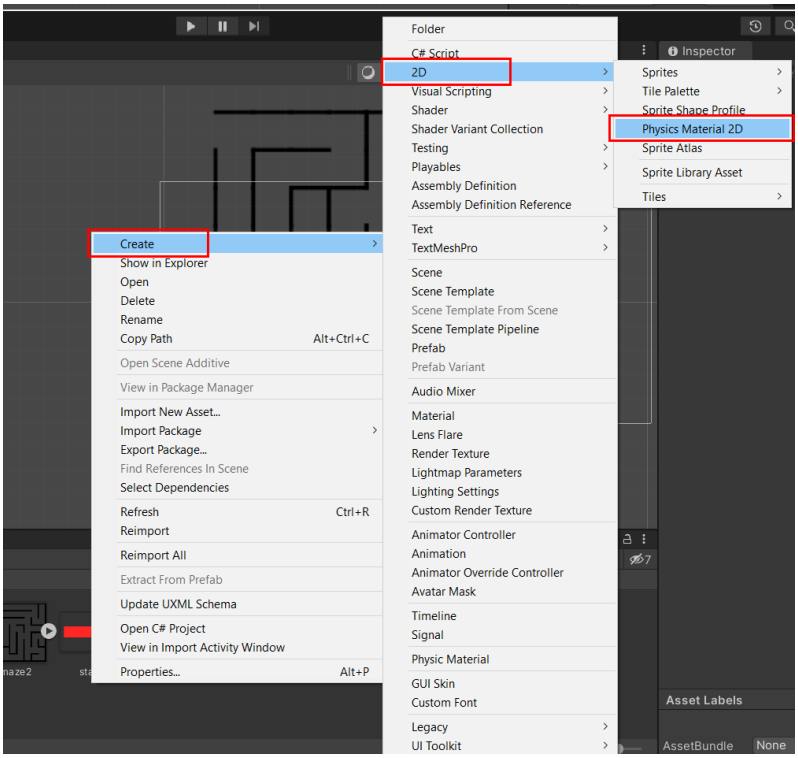
Úprava bariér - Môže sa stať, že niektoré bariéry sa nevytvorili správne a bude ich treba upraviť. V pravej časti zapneme **Edit Collider** a upravíme všetky zle vytvorené steny pomocou **zeleného štvorca**, ktorý sa nám zobrazí po prejdení kurzorom cez čiaru.



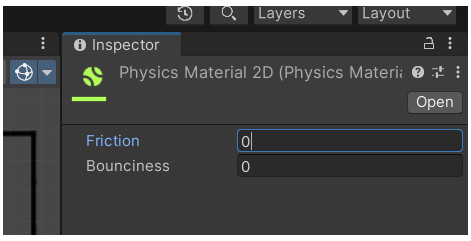
Po úprave ešte zablokujeme otvorené časti bludiska (tak ako vidieť na obrázku), aby postavička nemohla ujsť.

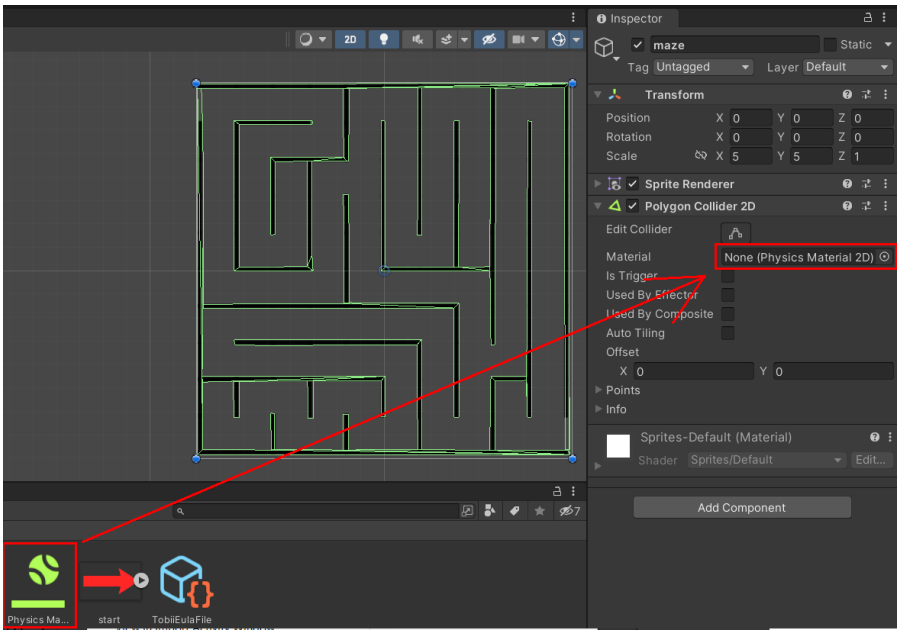


Hladšie steny v bludisku – aby sa postavička nezasekávala pri kolízií so stenou, pridáme **Physics Material 2D**. Klikneme pravým tlačidlom v **resources** a zvolíme *Create -> 2D -> Physics Material 2D*

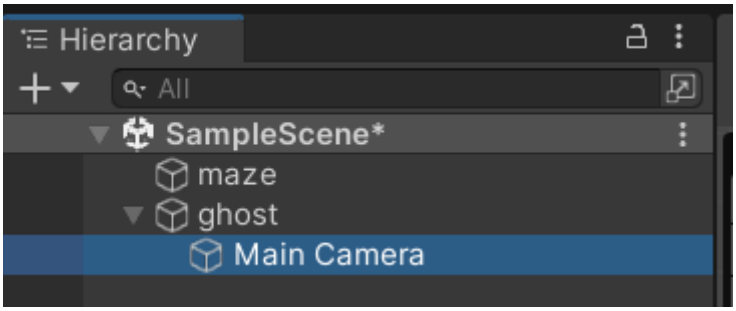


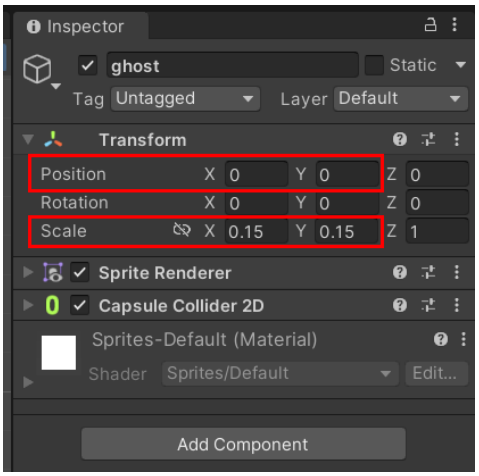
Následne mu nastavíme **Friction** na **0** a presunieme celý materiál do **Material**u bludiska



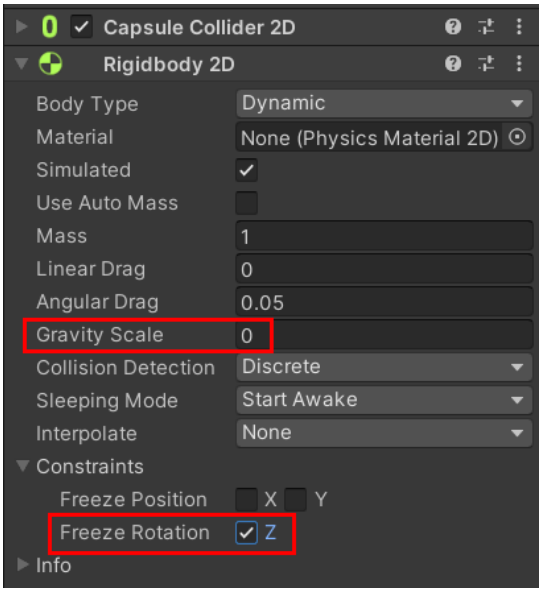


Pridanie postavičky – pridáme do **Hierarchy** obrázok s názvom **Ghost** a rovnako ako pri bludisku mu nastavíme **Scale** na X = 0.15 a Y = 0.15 a **Position** na X = 0 a Y = 0. V *Hierarchy* zvolíme **Main Camera** a presunieme ju na **ghost** (pri kamere tiež nastavíme **Position** na 0). Vďaka tomuto sa bude kamera hýbať zároveň s postavičkou (teraz môžeme zase zvoliť **ghost** a presunúť ho v scéne do ľavého horného rohu (na začiatok bludiska).





Nastavíme bariéru a komponent na ovládanie postavičky – pridáme *Component*y s názvom **Capsule Collider 2D** a **Rigidbody 2D**, pri ktorom nastavíme **Gravity Scale** na 0 (aby postavička v bludisku neklesala) a pod *Constraints* zaškrtneme **Freeze Rotation** (inak by sa postavička začala točiť dookola)



Pohyb postavičky – na pohyb postavičky sa musí použiť script napísaný v jazyku C#.
V **Resources** klikneme pravým a zvolíme *Create -> C# Script*, ktorý pomenujeme: *MovePlayer* a následne ho otvoríme. Automaticky sa nám vytvorila šablóna scriptu kde budeme používať funkciu **Update** (vyvoláva sa každý frame), do ktorej vložíme kód pre pohyb postavičky.

Prvotne si skúsime ovládanie cez WASD/šípky:

Keďže naša hra je 2D, musíme zisťovať len šírku a výšku a to pomocou **Input.GetAxis("")**, kde návratová hodnota je od -1 až po 1. Napríklad keď držíte ľavú šípku tak je -1, keď pravú tak 1 a keď ani jednu tak je hodnota 0.

```
float x = Input.GetAxis("Horizontal");
float y = Input.GetAxis("Vertical");
```

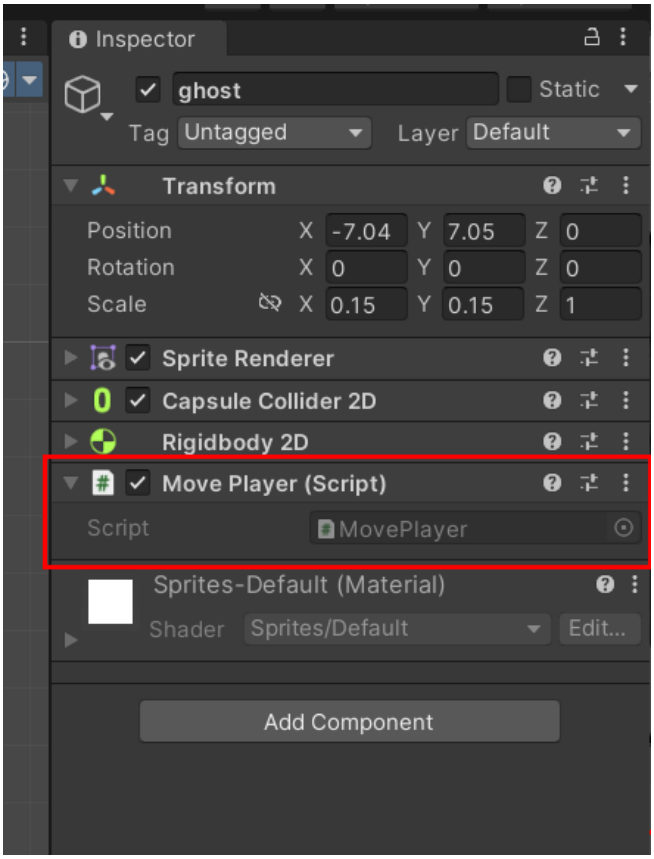
Postavičke budeme nastavovať rýchlosť (velocity) pohybu daným smerom. Ovládať budeme konkrétne komponent RigidBody2D (vytvorili sme pri tvorbe postavičky), ktorý vyhľadáme cez **GetComponent**. Ak chceme zvýšiť rýchlosť postavičky, stačí do Vector2 za x a y dosadiť napríklad ***3**.

```
GetComponent<Rigidbody2D>().velocity = new Vector2(x*3, y*3);
```

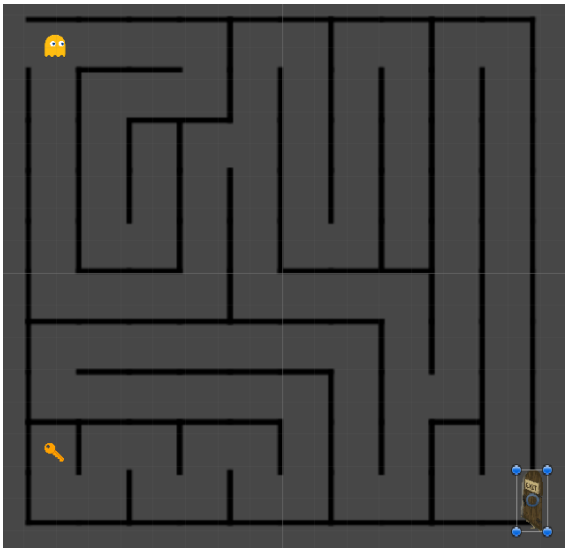
Celý kód

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class MovePlayer : MonoBehaviour
6   {
7       // Start is called before the first frame update
8       void Start()
9       {
10
11      }
12
13      // Update is called once per frame
14      void Update()
15      {
16          // WASD
17          float x = Input.GetAxis("Horizontal");
18          float y = Input.GetAxis("Vertical");
19          GetComponent<Rigidbody2D>().velocity = new Vector2(x, y);
20      }
21  }
```

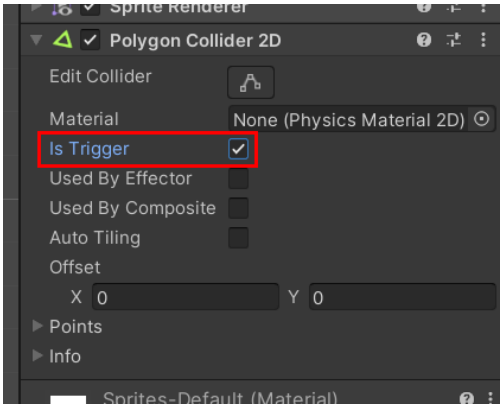
Po spustení by sa nám postavička ešte nedala ovládať, preto potrebujeme priradiť script ku konkrétnemu objektu (**ghost**). Stačí zobrať script z **Resources** a presunúť ku komponentom. Následne môžeme spustiť hru (horné tlačidlo play).



Pridanie kľúča a dverí – na prejdenie bludiska potrebujeme východ (dvere), no aby to nebolo príliš jednoduché, pridáme do bludiska kľúč na odomknutie dverí. Do *Hierarchy* vložíme obrázky **key** a **exit**. Scale **kľúča** nastavíme X a Y na 0.05 (presunieme ho napríklad do ľavého dolného rohu) scale **dverí** nastavíme na 0.1 a presunieme ho do pravého dolného rohu (kde bude náš exit) tak, aby do nich postavička vedela naraziť.



Nastavenie kolízie kľúča a dverí – pri objektoch **key** a **exit** vytvoríme komponent **Polygon Collider 2D**. V komponente kľúča zaškrtneme **Is Trigger**, vďaka čomu budeme vedieť zistiť kolíziu postavičky a kľúča.



V scripte *MovePlayer* pridáme premennú a funkciu čo sa vyvolá po prejdení cez objekt s tagom „key“

```
public static bool haveKey = false;
```

```
private void OnTriggerEnter2D(Collider2D other)
{
	if (other.tag == "key") {
		exitKey = true;
	}
}
```

Taktiež pridáme funkciu, ktorá sa vyvolá pri kolízií hráča a objektu s tagom „exit“

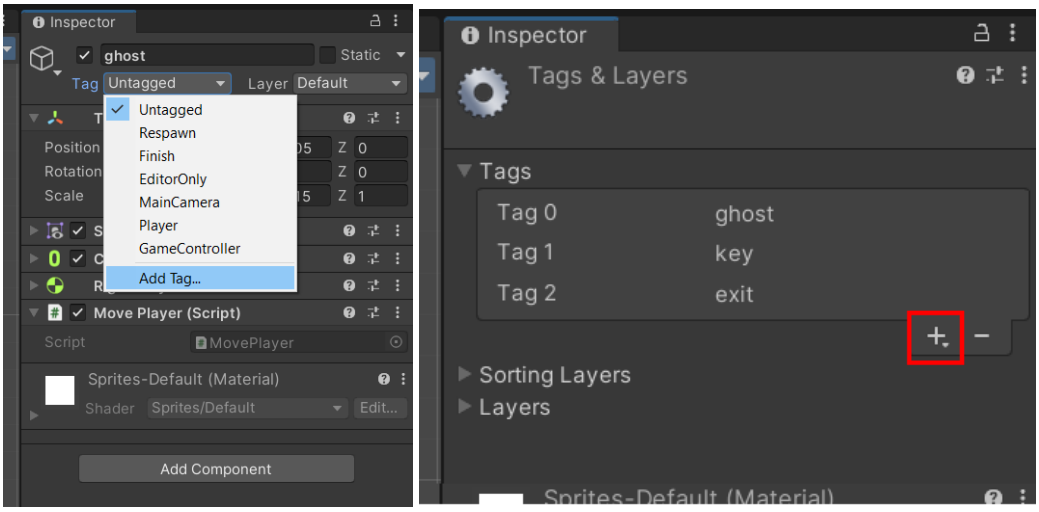
```
private void OnCollisionEnter2D(Collision2D other)
{
	if (other.gameObject.tag == "exit" && exitKey) {
		print("toto je koniec!");
	}
}
```

Celý kód

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MovePlayer : MonoBehaviour
{
    public static bool exitKey = false;

    // Start is called before the first frame update
    void Start() {}

    // Update is called once per frame
    void Update()
    {
        // WASD
        float x = Input.GetAxis("Horizontal");
        float y = Input.GetAxis("Vertical");
        GetComponent<Rigidbody2D>().velocity = new Vector2(x*3, y*3);
    }

    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.tag == "key") {
            exitKey = true;
        }
    }

    private void OnCollisionEnter2D(Collision2D other)
    {
        if (other.gameObject.tag == "exit" && exitKey) {
            print("toto je koniec!")
        }
    }
}
```

Nastavenie tagov – aby sme mohli v scriptoch rozlišovať objekty, musíme im nastaviť tagy.

Pridáme 3 tagy: **ghost**, **key**, **exit**



Po vytvorení tagov nastavíme všetkým trom objektom správny tag zo zoznamu.

Vymazanie kľúča – po tom čo hráč získa kľúč, odstránime celý objekt. Vytvoríme si nový script s názvom *Key*, následne vložíme funkciu, ktorá sa vyvolá len keď cez kľúč prejde objekt s názvom „ghost“, teda postavička.

```
private void OnTriggerEnter2D(Collider2D other)
{
      if (other.tag == "ghost") {
            Destroy(gameObject);
      }
}
```

Po uložení presunieme script ku komponentom objektu **key**. Inak by script nefungoval.

Celý kód

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Key : MonoBehaviour
{
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.tag == "ghost") {
            Destroy(gameObject);
        }
    }
}
```

Teraz môžeme hru otestovať, a v prípade že postavička zobrala kľúč, tak sa po kolízií s exitom zobrazí v konzole text.

Ovládanie pomocou Eye Trackeru – v scripte *MovePlayer* pridáme do funkcie **Update** nasledujúci kód:

```
GazePoint gazePoint = TobiiAPI.GetGazePoint();
if (gazePoint.IsRecent())
{
   // -0.5 z dôvodu že kamera je vždy v strede a gazePoint počíta jednotky z rohu hernej obrazovky

   double x = (gazePoint.Viewport[0]-0.5);
   double y = (gazePoint.Viewport[1]-0.5);
   GetComponent<Rigidbody2D>().velocity = new Vector2((float)x*7, (float)y*7);
}
```

Pridáme **Tobii.Gaming** a zakomentujeme časť kódu pre pohyb WASD

```
using Tobii.Gaming;
```

Celý kód

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Tobii.Gaming;

public class MovePlayer : MonoBehaviour
{
    public static bool exitKey = false;

    // Start is called before the first frame update
    void Start() {}

    // Update is called once per frame
    void Update()
    {
        GazePoint gazePoint = TobiiAPI.GetGazePoint();
        if (gazePoint.IsRecent()) // Use IsValid property instead to process old but valid data
        {
            // -0.5 z dôvodu že kamera je vždy v strede a gazePoint počíta jednoty z rohu hernej obrazovky
            double x = (gazePoint.Viewport[0]-0.5);
            double y = (gazePoint.Viewport[1]-0.5);
            GetComponent<Rigidbody2D>().velocity = new Vector2((float)x*7, (float)y*7);
        }

        // WASD
        //float x = Input.GetAxis("Horizontal");
        //float y = Input.GetAxis("Vertical");
        //GetComponent<Rigidbody2D>().velocity = new Vector2(x*3, y*3);
    }

    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.tag == "key") {
            exitKey = true;
        }
    }

    private void OnCollisionEnter2D(Collision2D other)
    {
        if (other.gameObject.tag == "exit" && exitKey) {
            print("toto je koniec!");
        }
    }
}
```

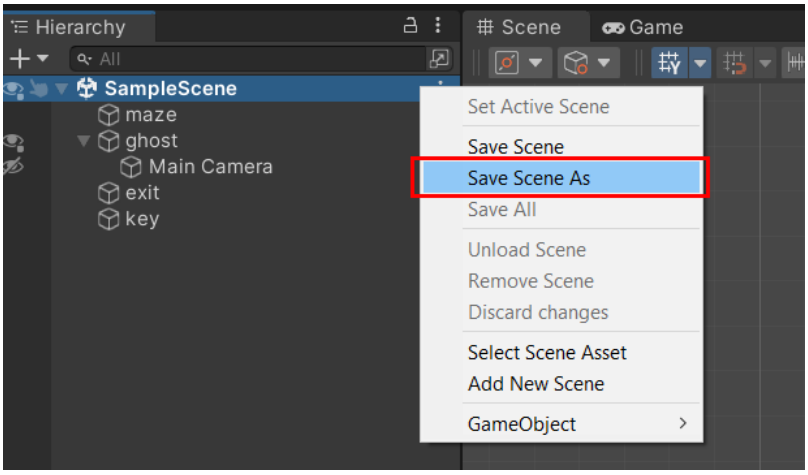
Možnosť ukončenia hry – v scripte *MovePlayer* pridáme do funkcie **Move** podmienku na sledovanie stlačenia ESC (fungovať bude až pri skompilovanej hre)

```
if (Input.GetKeyDown(KeyCode.Escape)) {
      Application.Quit();
}
```

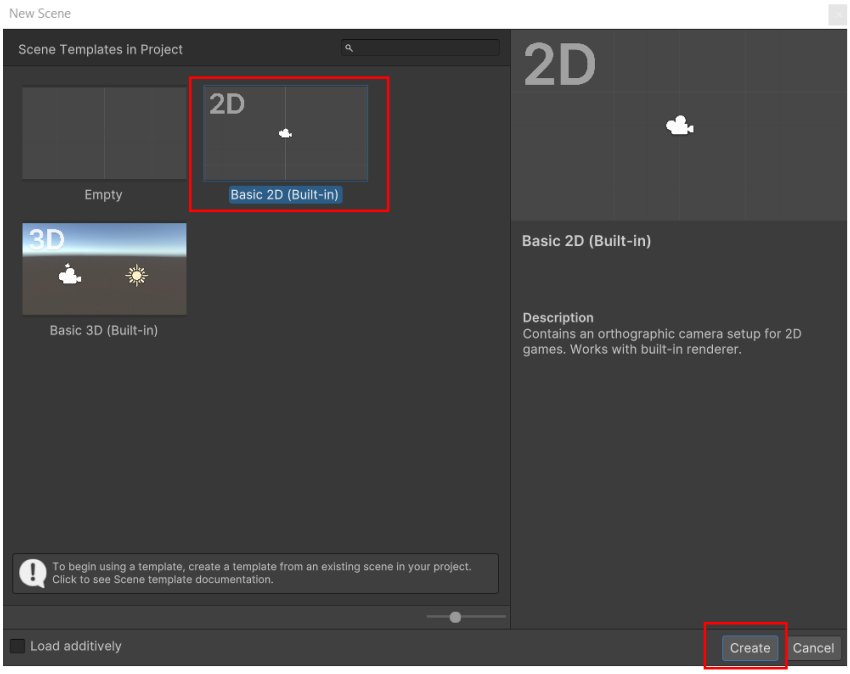
Prejdenie bludiska – po ukončení bludiska sa zobrazí výherná obrazovka. V scripte *MovePlayer* pridáme do funkcie **OnCollisionEnter2D**:

```
SceneManager.LoadScene("winscreen");
```

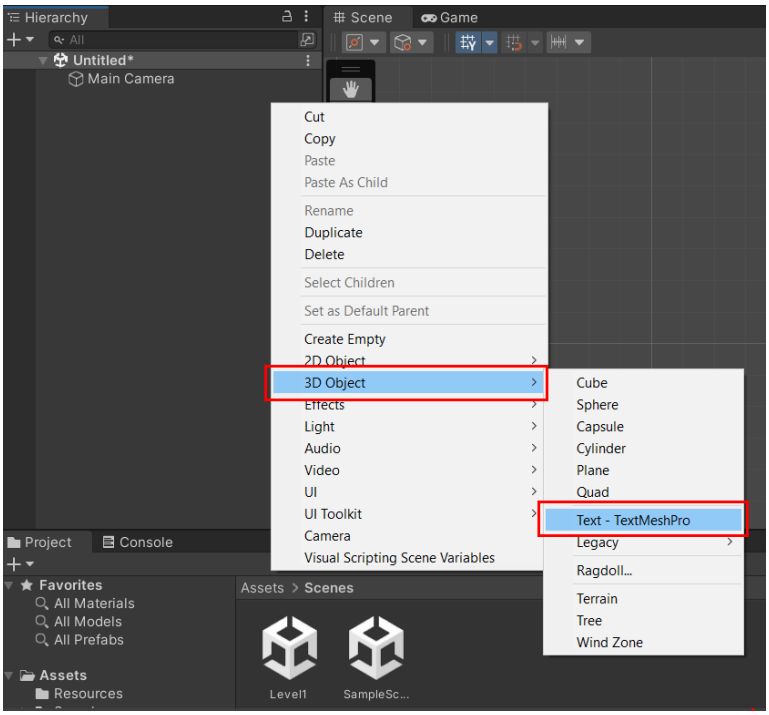
Uložíme aktuálnu scénu pod názvom *Level1* a uložíme do priečinku **Scenes** v projekte



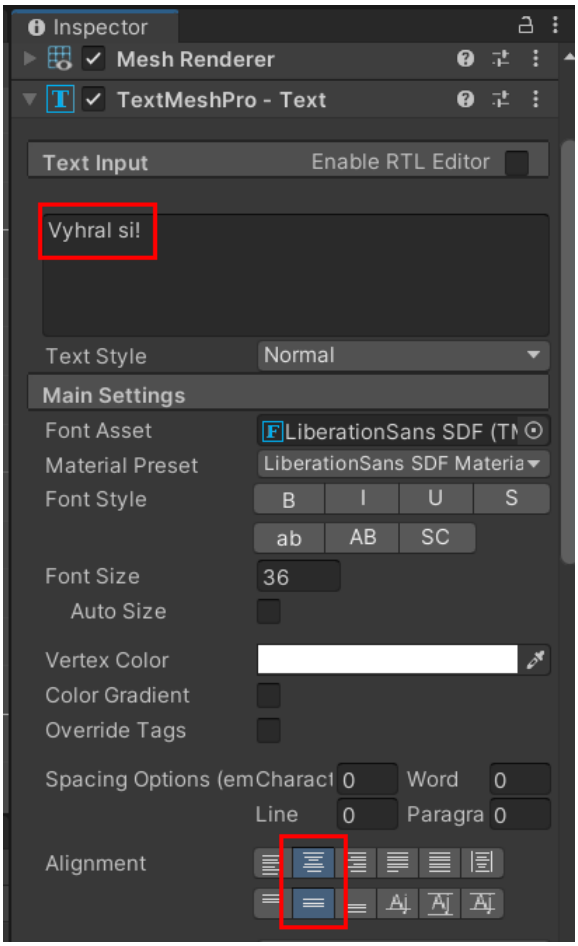
Vytvoríme novú scénu *File -> New Scene*



V **Hierarchy** pridáme *3D object -> Text – TextMeshPro* a následne vyskočí okno kde Importujeme TMP



Upravíme vytvorený nápis a vycentrujeme



Vytvoríme nový script *NewGame* v **Resources**, vďaka ktorému po stlačení medzerníku začneme novú hru. Do scriptu vložíme:

```
using UnityEngine.SceneManagement;
```

a do funckie **Update**

```
if (Input.GetKeyDown("space")) {
      SceneManager.LoadScene("Level1");
}
```

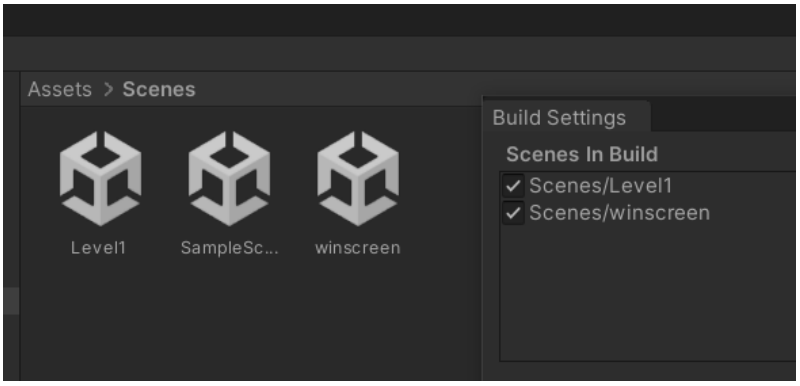
Po uložení tento script presunieme do komponentov Textu.

Celý kód

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class NewGame : MonoBehaviour
{
    // Start is called before the first frame update
    void Start() {}

    // Update is called once per frame
    void Update()
    {
        // Space spustí novú hru
        if (Input.GetKeyDown("space")) {
            SceneManager.LoadScene("Level1");
        }
    }
}
```

Scénu uložíme pod názvom *winscreen*

Aby sme sa vedeli presúvať na inú scénu, potrebujeme všetky pridať do **Build Settings** *File -> Build Settings... V*ymažeme odtiaľ starú scénu a pridáme z pričinka **Scenes** scény **Level1**, **winscreen** a zatvoríme.



Hru už len otestujeme...