Лабораторно упражнение N3

Инструкции към игра "Box Shooter"

(с материали на Браян Уин, Мичигански държавен университет)

А. Предварителни указания за разработка на скриптови компоненти

- Платформа Unity използва C# и/или JS езици за създаване на скриптови асети. И двата езика се поддържат от вградената среда MonoDevelop.
- При създаване на нов скрипт (софтуерен компонент) средата създава скелет на класа с името на скрипта,
- Скриптовете могат да се задават за управление на игрови обекти (т.е. да им се присвояват) с добавяне в инспектора или с влачене в долната му част.
- При присвояване на скриптов компонент на игрови обект се създава екземпляр (обект) на този клас, напр.:

```
using UnityEngine;
using System.Collections;

public class Chaser : MonoBehaviour {
	// Use this for initialization
	void Start () {
	}

	// Update is called once per frame
	void Update () {
	}
}
```

- Публичните променливи на класа или на екземпляр (обект) на класа представят като видими полета (свойства) на скрипта в инспектора, със зададените им начални стойности (ако има такива).
- Публичните свойства на екземпляра на класа (т.е. на обекта) могат да се променят както в скрипта (програмно), така и в инспектора.
- Методът **`Start()`** се извиква еднократно при стартиране на играта заедно с **`Awake()`** и **`OnEnable()`**, докато **`Update()`** се изпълнява за всеки фрейм, т.е. десетки пъти в секунда.
- Моделът на изпълнение на методите е с управление по събития, като с опростен вид е показан на фигурата по-долу.

За повече информация относно реда на изпълнение на методите, вижте https://docs.unity3d.com/Manual/ExecutionOrder.html

- Обръщението към игровия обект, към който е прикачен екземпляра на класа, става или с името на обекта (напр. **`gameObject`**), или с референцията към самия екземпляр на класа, напр. **`this.gameObject`**.
- Всички свойства на игровия обект могат да се променят програмно. Напр. можем да скрием (деактивираме) игрови обект по следния начин:

```
this.gameObject.setActive(False); //makes the game object invisible
```

- Можем да търсим игрови обекти или по име, или по маркер (таг)

```
private gameObject target1;

private gameObject target2;

voif Start() {

    target1.Find("Ball"); //finds the game object named Ball

    target2.FindWithTag("Floor"); //finds object tagged as Floor

    target1.setActive(False); //set as invisible

    target2.transform.position.x +=1; //moves it 1 meter in X

}
```

- Можем да местим/завъртаме/скалираме игрови обект подобно на показания по-горе начин за **`target2`**
- Можем да достъпваме други софтуерни компоненти на игрови обекти от тип **`TYPE`** чрез

```
gameObject.getComponent<TYPE>();
```

Например:

```
rb = gameObject.getComponent<RigidBody>();

rb.useGravity = true; //turns gravity on
```

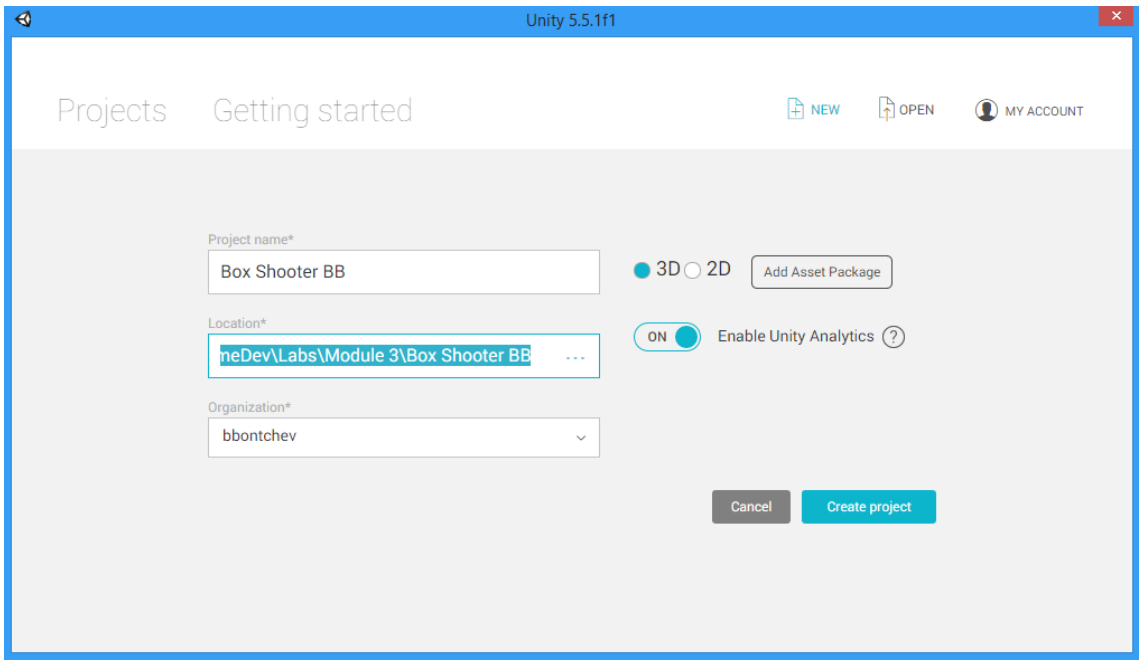
За повече информация: https://docs.unity3d.com/Manual/UnityManual.html

Б. Указания за разработка на игра Box Shooter – подготовка и създаване на мишена

Играта Box Shooter е FPS игра със стрелба по движещи се мишени от различни типове. В зависимост от типа на мишената, резултатът може да се повиши или намали. Играта продължава фиксирано време.

Стъпка 1 – Отворете Unity и създайте нов проект с име Box Shooter от тип 3D.

Изберете папка за проектните файлове, но не добавяйте стандартни вградени Unity асети (активи) с цел по-бърза разработка на играта.

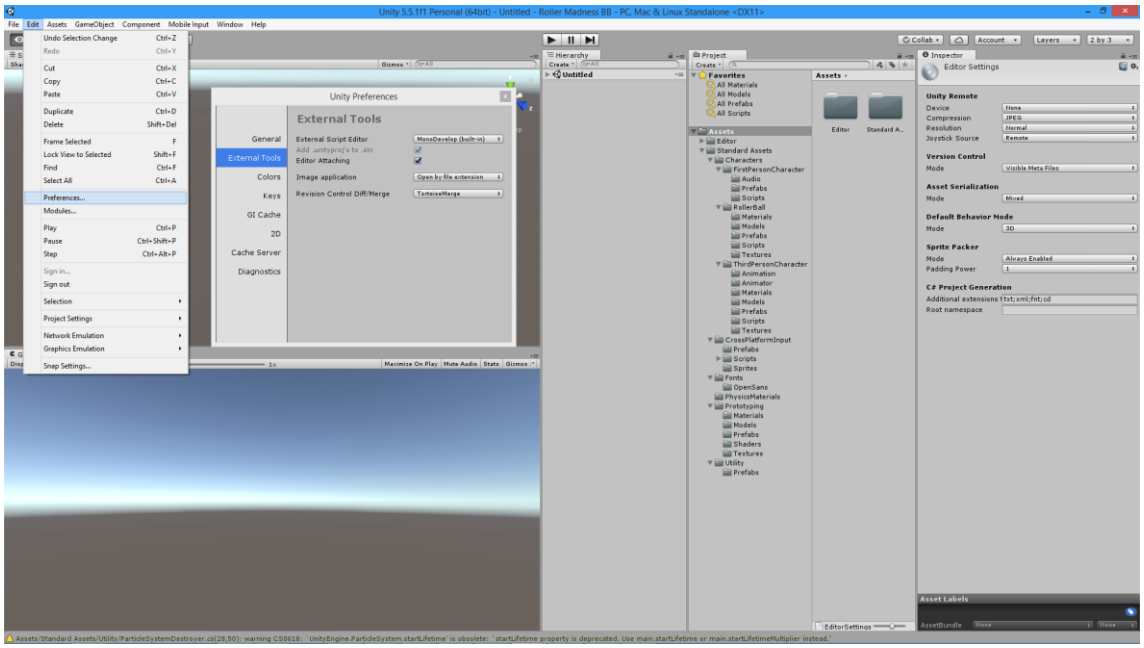


След това щракнете на бутона Done и после на Create project.

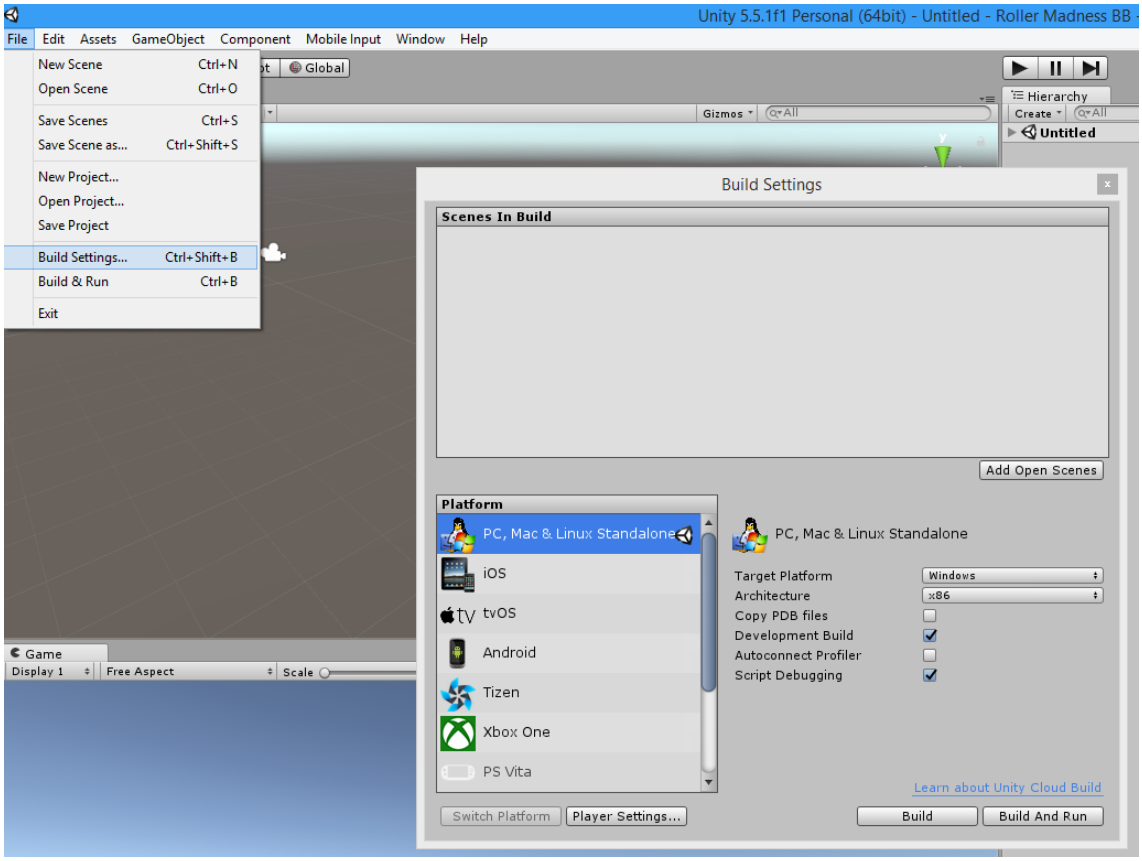
Стъпка 2 – изберете layout 2 by 3.

Стъпка 3 (опционална) – ако желаете да ползвате средата MonoDevelop и не сте я избрали при инсталацията, направете следното:

1. В меню Edit > Preferences в панела External Tools изберете за External Script Editor средата MonoDevelop.
2. След това, Unity проектът ви ще се отвори/синхронизира в/с MonoDevelop средата чрез меню Assets > Open C# Project.

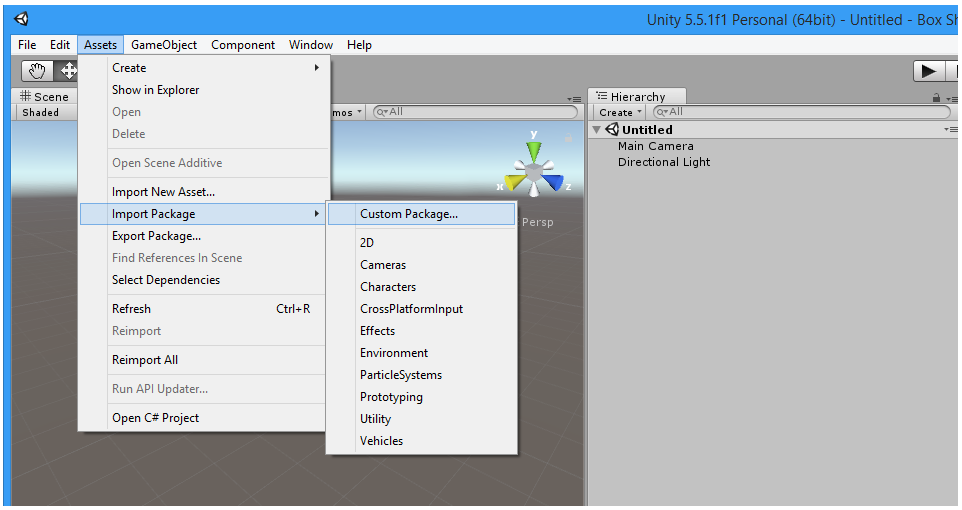


3. Проверете дали опциите Development Build и Script Debugging са избрани в Build Settings за избраната от вас платформа (меню: File > Build Settings).

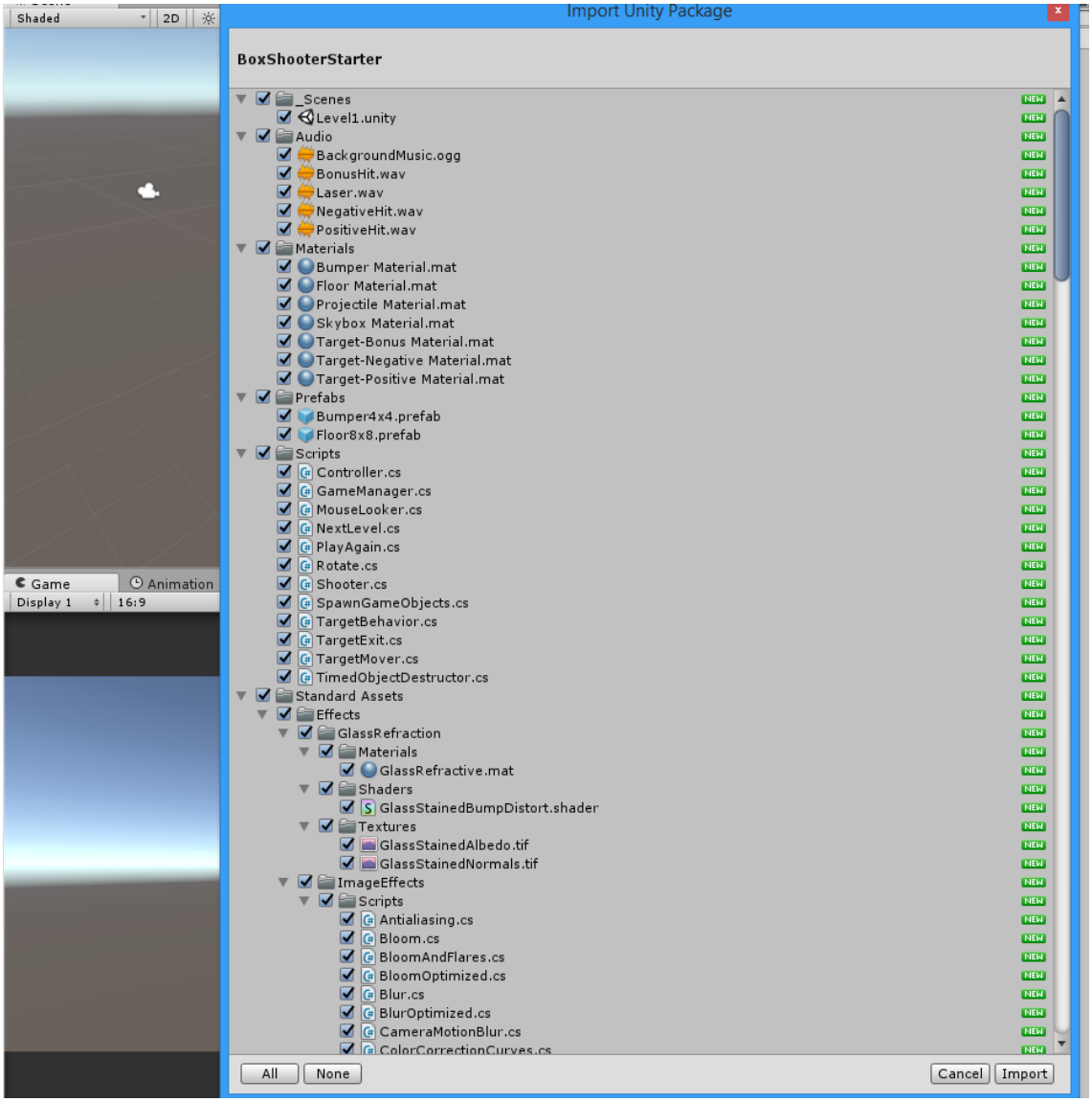


Стъпка 4 – импортирайте пакета със скрипт асети (*на Браян Уин, Мичигански държавен университет*) с име BoxShooterStarter.unitypackage (от архива BoxShooterAssets.zip, наличен в сайта на курса), с влачене на файла в папка Assets в Project панела и последващо натискане на бутона Import. Ако е нужно, задайте Update за скриптовете.

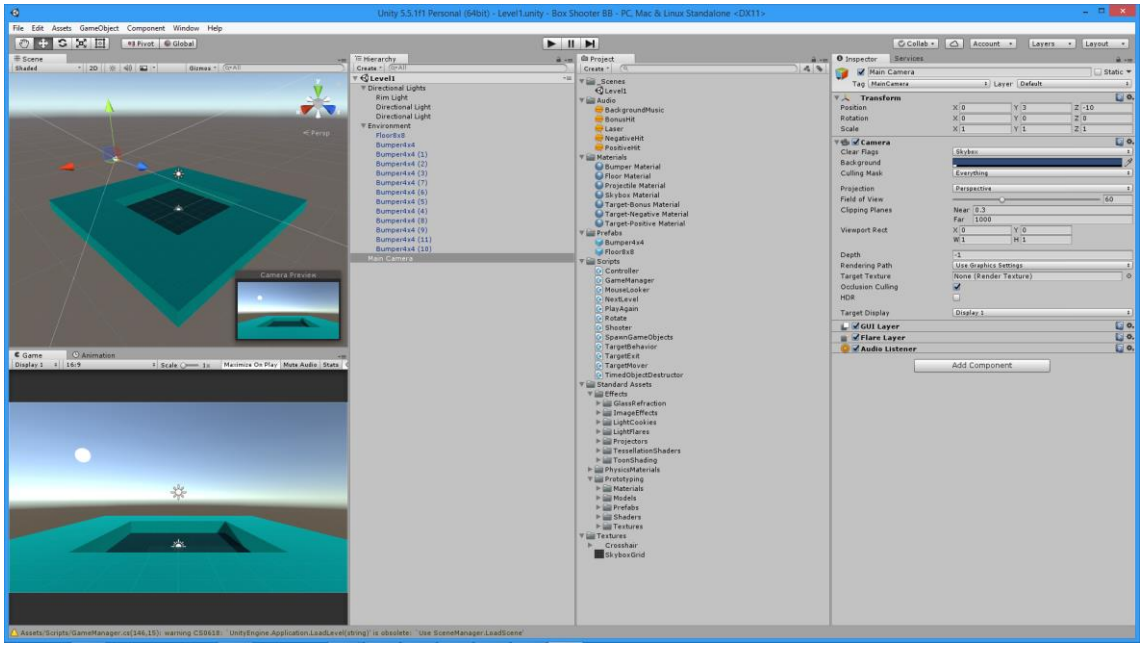
Друг начин да ги импортирате е чрез меню Assets > Import Package > Custom Package.



Проверете, че скриптовете са импортирани в папка Assets > Scripts. Забележете, че са импортирани и стандартни асети на платформата.

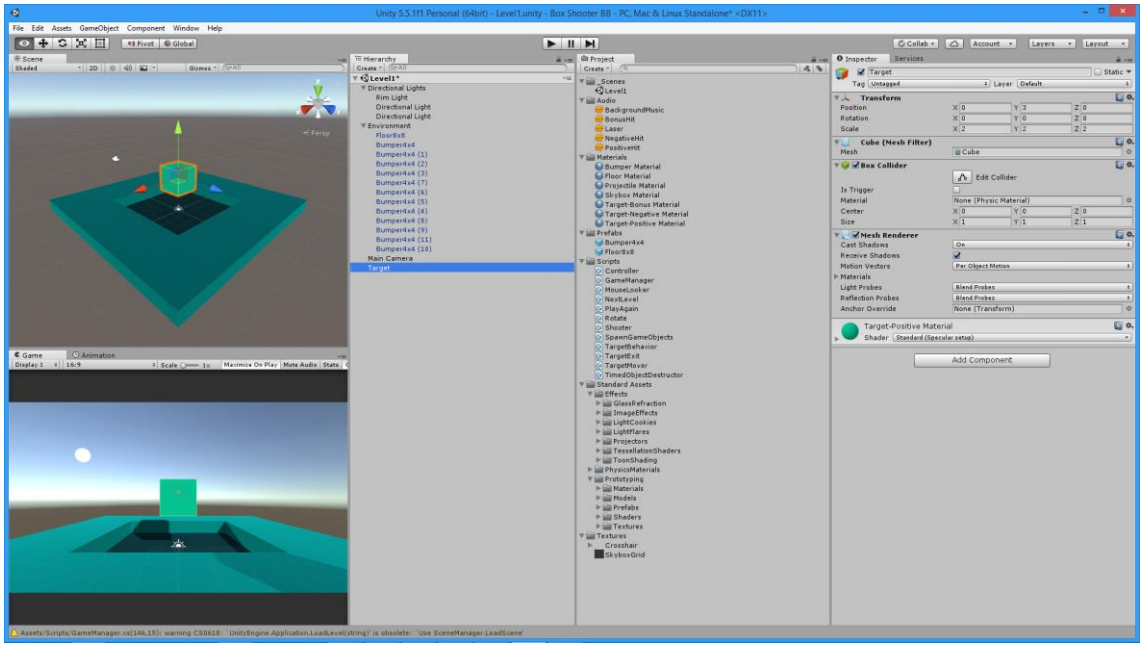


Стъпка 5 – в папка _Scenes щракнете два пъти на сцената Level1, за да я заредите в дизайнерския и игровия прозорец. Разгледайте игровите обекти и техните материали и други асети.



Б. Базов дизайн на първа сцена

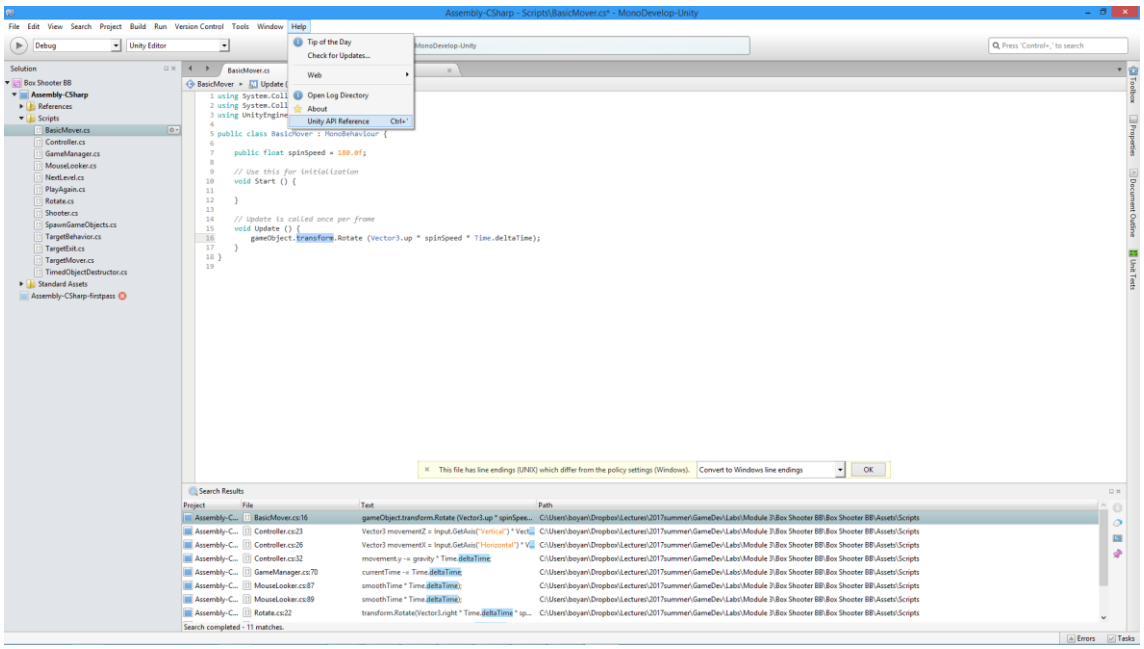
Стъпка 6 – създайте мишена (на нивото на главната камера), която да е куб с име Target и с координати (X, Y, Z) = (0, 3, 0). Скалирайте го (чрез Scale в Inspector) два пъти (т.е. с мащаб 2:1) и по трите координати. Добавете към куба материала с име Target-Positive Material.



Стъпка 7 – създайте нов скриптов компонент на езика C# за движение на обекта с име BasicMover и го преместете в папката Scripts. Забележете, че Unity автоматично създава по шаблон следния скрипт:

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

public class BasicMover : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
}
```

Стъпка 8 – преработете скрипта с цел ротация на игровия обект, към който той е прикрепен, съгласно фигурата по-долу.



За всеки компонент, добавен към игровия компонент, можете да разгледате онлайн помощ относно Scripting API, чрез маркиране на компонента (напр. Transform) и с използване на показаното по-горе меню (или чрез CTRL+‘). Можете и директно да отворите в браузър помощта за този компонент на адрес https://docs.unity3d.com/ScriptReference/Transform.html.

Стъпка 9 – проверете какво означава методът Transform.Rotate, както и Vector3.up, по описания в Стъпка 8 начин.

Стъпка 10 – съхранете скрипта и проверете в инспектора за обекта Target дали и *как* се вижда променливата spinSpeed и нейната начална стойност 180.0f (градуси за секунда). Проиграйте скрипта и се убедете в ротацията на обекта Target. Опитайте в режим на игра (не на дизайн!) с 360 и 10 градуса за секунда.

Стъпка 11 – упражнете се, като направите грешка в

```
gameObject.transform.Rotate (Vector3.up * spinSpeed * Time.deltaTime);
```

– напр. замените spinSpeed с spinSped. Опитайте се да проиграете играта и разгледайте рапортуваните грешки с двойно щракване на съобщението за грешка. Поправете грешката.

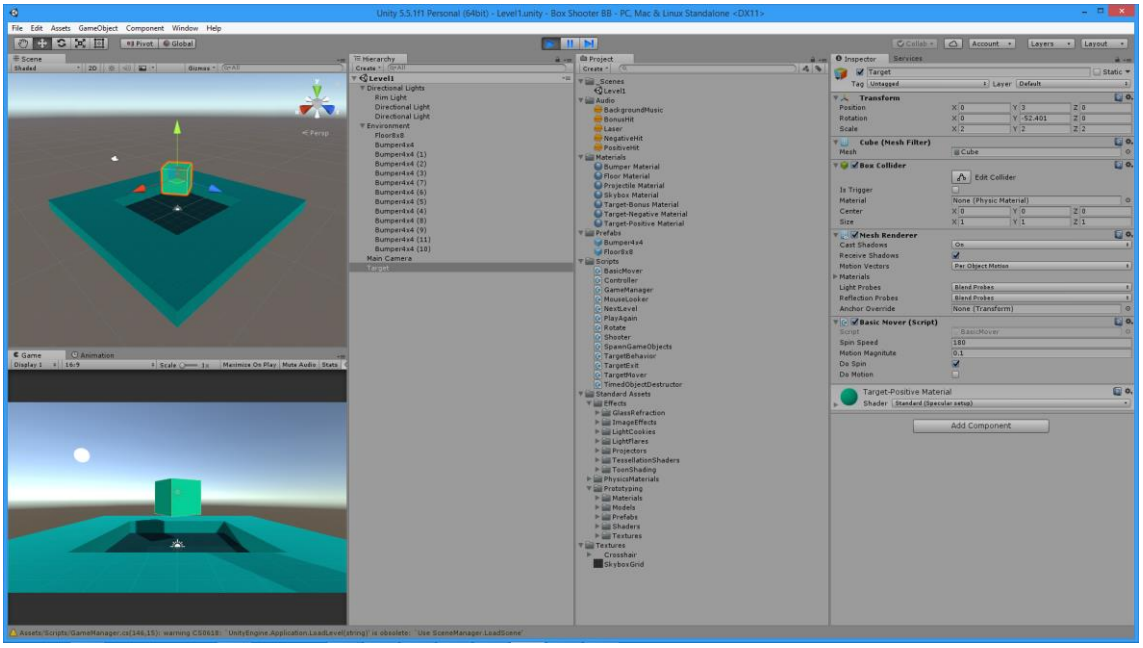
Стъпка 12 – преработете скрипта с цел транслация на игровия обект, към който той е прикрепен, съгласно кода по-долу. Запазете промените и вижте как в инспектора за обекта-мишена се представя променливата motionMagnitute. Проиграйте скрипта и обяснете поведението на обекта-мишена.

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

public class BasicMover : MonoBehaviour {

    public float spinSpeed = 180.0f;
    public float motionMagnitute = 0.1f;

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        // rotate around the up axis
        gameObject.transform.Rotate (Vector3.up * spinSpeed * Time.deltaTime);

        // move up and down
        gameObject.transform.Translate(Vector3.up * Mathf.Cos(Time.timeSinceLevelLoad
) * motionMagnitute);
    }
}
```

Стъпка 13 – преработете скрипта с цел условна ротация и транслация на игровия обект, към който той е прикрепен, съгласно кода по-долу. Запазете промените и вижте как в инспектора за обекта-мишена се представят двете нови променливи. Проиграйте скрипта и в режим на игра маркирайте и демаркирайте всеки един от двата чек-бокса.

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

public class BasicMover : MonoBehaviour {

    public float spinSpeed = 180.0f;
    public float motionMagnitute = 0.1f;

    public bool doSpin = true;
    public bool doMotion = false;

    // Update is called once per frame
    void Update () {
        if (doSpin) {
            // rotate around the up axis
            gameObject.transform.Rotate (Vector3.up * spinSpeed * Time.deltaTime);
        }
        if (doMotion) {
```

```
            // move up and down
            gameObject.transform.Translate(Vector3.up * Mathf.Cos(Time.timeSinceLevel
Load) * motionMagnitute);
        }
    }
}
```



Стъпка 14 – запазете създадената дотук сцена под име Level1.

Стъпка 15 – преименувайте BasicMover скрипта в BasicTargetMover и се опитайте да стартирате играта. Къде е грешката?

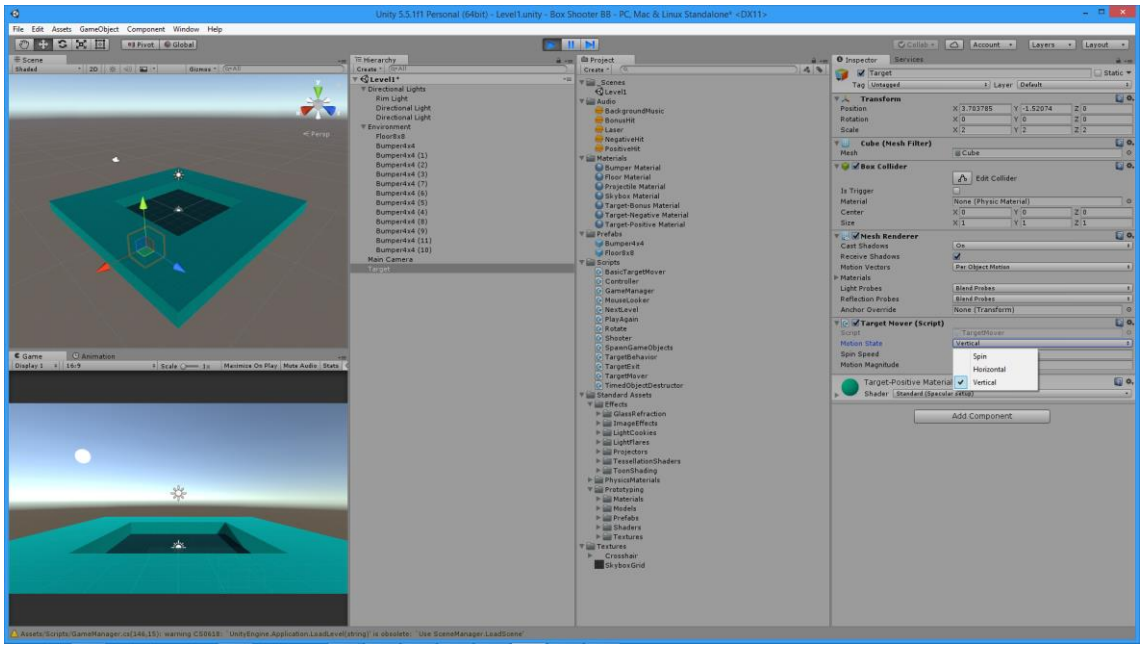
Указание: променете също така името на класа.

Стъпка 16 – Скалиране на времето - играта може да се показва с 15, 30 или 60 фрейма за секунда на бавен, среден и бърз компютър. За да скалираме скоростта на въртене така, че да е еднаква за различните честоти на показване на кадрите, използваме `Time.deltaTime` – дава в милисекунди времето от последния показан фрейм!

Стъпка 17 – премахнете компонента BasicTargetMover от обекта Target и добавете съществуващия компонент TargetMover. В режим на игра експериментирайте със стойностите на Motion State, както е показано на фигурата по-долу.

Разгледайте в редактора кода на BasicTargetMover.

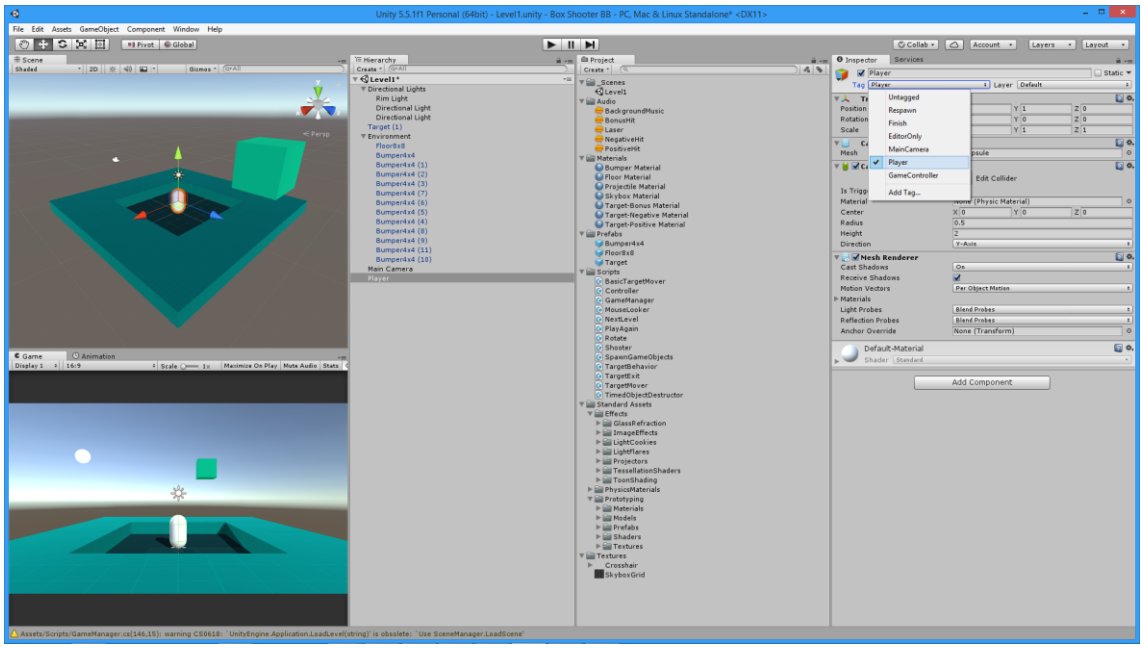
Съхранявайте периодично сцената.



Стъпка 18 – създайте префаб от обекта Target без да сменяте името му. Изтрийте обекта Target и съхранете сцената.

С. Указания за разработка на игра Box Shooter – играч и камера

Стъпка 19 – създайте играч с аватар капсула (Capsule) и го позиционирайте в точка (X, Y, Z) = (0, 1, 0). Преименувайте го на Player и изберете за Tag предефинираната стойност Player (както е показано на фигурата по-долу).



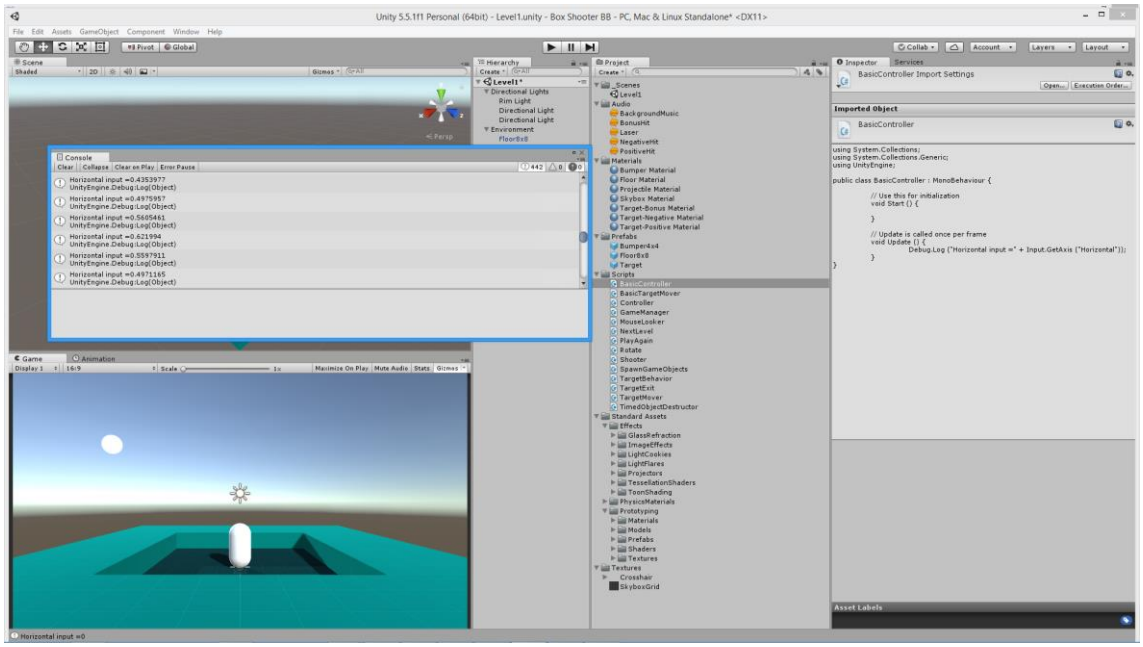
Добавете предефинирания компонент с име Character Controller от пакета Physics. Забележете, че той не приема събития от клавиатура или мишка с цел движението на играча – трябва сами да се погрижим за това.

Съхранете сцената и я проиграйте.

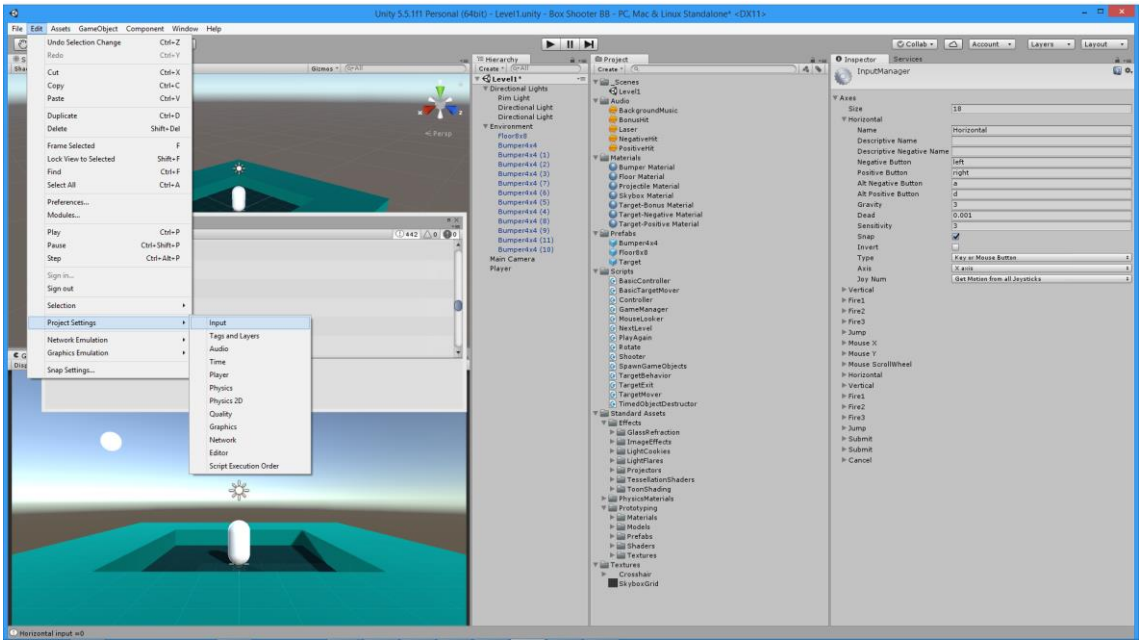
Стъпка 20 – в папката Scripts създайте нов скрипт с име BasicController, по кода по-долу. Съхранете го в редактора и го присвоете на Player.

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

public class BasicController : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        Debug.Log ("Horizontal input =" + Input.GetAxis ("Horizontal"));
    }
}
```

В режим на игра вижте изхода от `Debug.Log` в конзолата при вход с клавиши-стрелки или *a* и *d*.



Стъпка 21 – разгледайте Input Manager компонента чрез меню Edit > Project Settings > Input, в дяла му Horizontal, както е показано по-долу.

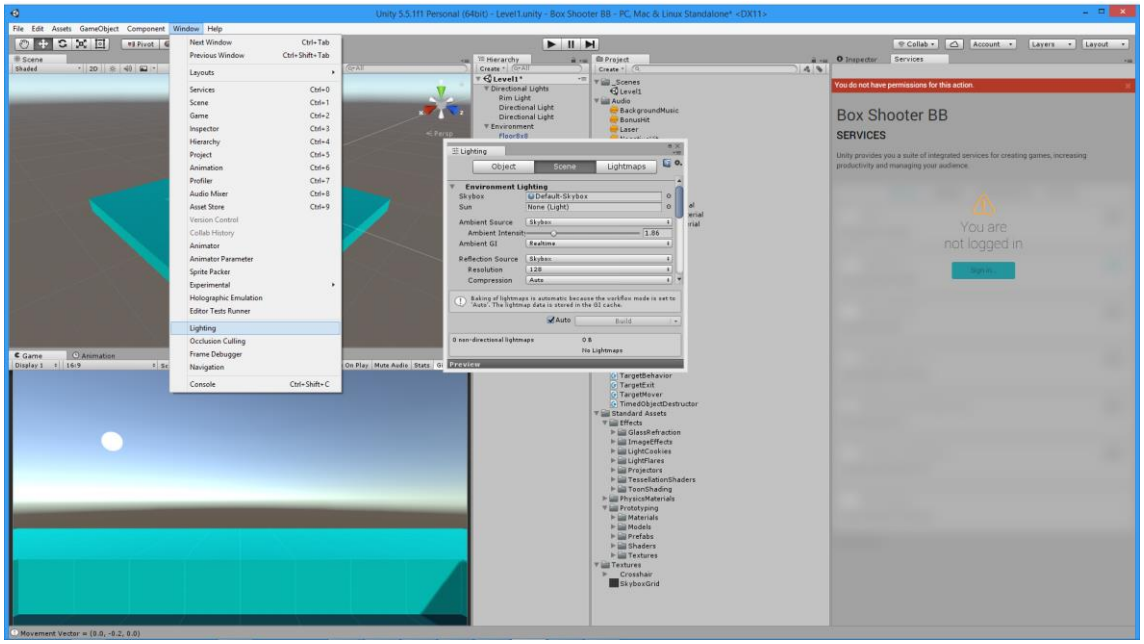


Стъпка 22 – за играча, заменете скрипта BasicController с друг – с име Controller. Проиграйте с управление чрез стрелковите клавиши. Прегледайте изхода в конзолата. Разгледайте как кодът на Controller използва входа (от `Input.GetAxis("Vertical")` и `Input.GetAxis("Horizontal")`, за да движи обекта-играч.

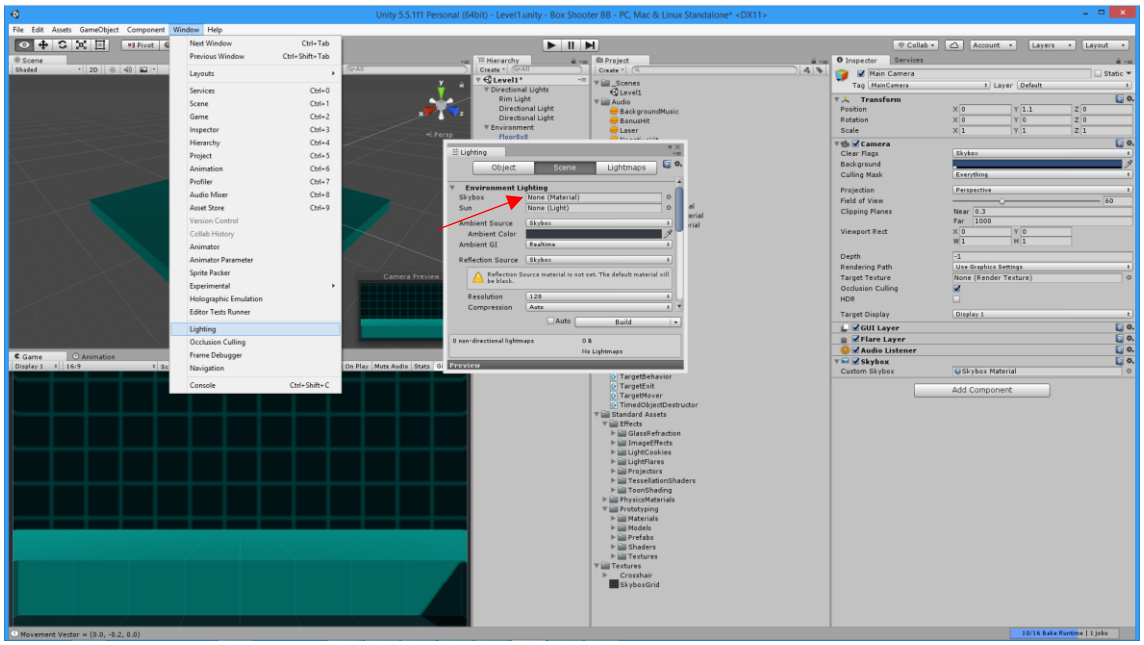
Стъпка 23 – променете позицията на главната камера на (X, Y, Z) = (0, 2.1, 0). Направете я под-възел в йерархията на играча. Проиграйте и се убедете в транслацията на камерата чрез стрелките или клавишите wasd.

Стъпка 24 – добавете към играча съществуващия скрипт MouseLooker. Проиграйте и се убедете в ротацията на камерата чрез мишката. Вече имате FPS управление на камерата.

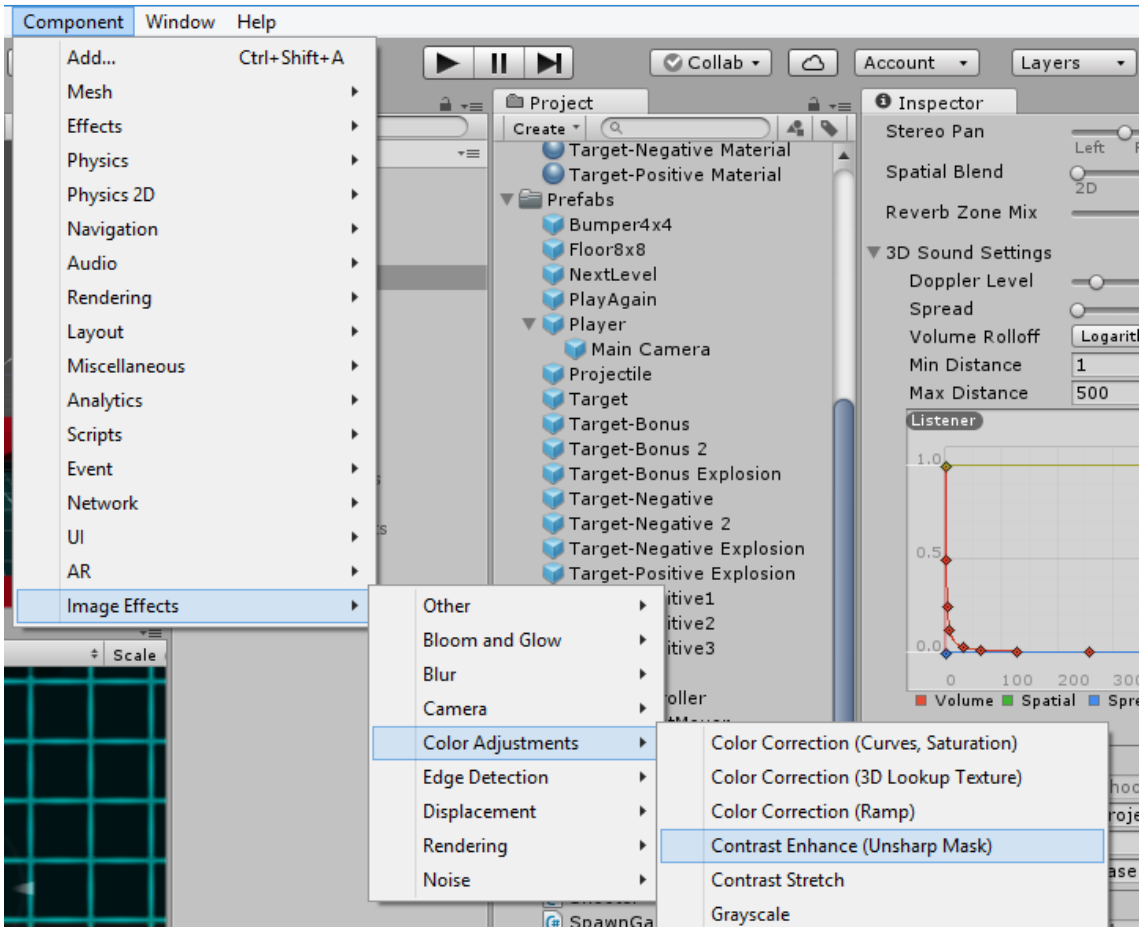
Стъпка 25 – за упражнение - подобрете осветеността на виртуалния свят (интензивност на осветлението, отражения, слънце, небе и др.) чрез менютата от фигурата по-долу.



Стъпка 26 – изберете Main Camera и добавете към нея компонент Rendering > Skybox. Добавете наличния материал Skybox Material към този компонент и вижте промяната (само) в режим на игра – но не и в изгледа на сцената. Премахнете Skybox в изгледа на сцената по указания по-долу начин.



Стъпка 27 – добавете към главната камера два компонента за ефекти на картината (Image Effects):



- Components > Image Effects > Color Adjustments > Contrast Enhance (Unsharp Mask). Задайте Intensity да е равно на 1 или 2.
- Чрез меню Components > Image Effects > Bloom and Glow > Bloom. Задайте Bloom Intensity да е равно на 2.

Проиграйте.

Стъпка 28 – добавете към главната камера Audio Source Задайте свойството AudioClip да е равно на клипа BackgroundMusic, Loop=yes и още Volume=0.25 и Pitch=1.2. Проиграйте.

Стъпка 29 – до момента при проиграване често се вижда част от капсулата или от сянката ѝ. За да не се вижда, деактивирайте Mesh Renderer за обекта Player. Създайте префаб за обекта Player чрез влачене на обекта в префаб папката. Съхранете сцената и проекта.

Стъпка 30 – нека създадем снаряд (projectile) във формата на малка сфера. Създайте сферата под име Projectile. Добавете към сферата Projectile Material. Скалирайте я по X, Y и Z с коефициент 0.2. Добавете Rigid Body компонент с цел да взаимодейства с другите тела. Задайте Use Gravity като демаркирано, за да няма гравитация за сферата.

Стъпка 31 – с цел оставане на следа (опашка), добавете към обекта Projectile скрипта Effects > Trail Renderer; добавете материала Default-Particle. Задайте графично за начална ширина 0.1, а за крайна - 0. Задайте за цвят морско-синьо с увеличаване на

прозрачността. Запазете Projectile като префаб и изтрийте самия обект с име Projectile. Съхранете сцената и проекта.

Стъпка 32 – за да изравните изгледа на сцената с изгледа на играта, селектирайте главната камера и изберете Game Object > Align View to Selected (което е обратното на Align with View!).

Стъпка 33 – създайте мишена чрез влачене на префаба Target в изгледа на сцената и задаване на координати за него (0, 6, 10). Задайте за свойството Motion State на компонента Target Mover стойността Spin. Дублирайте Target и преместете по X новия обект 4 м надясно, след което му задайте за свойството Motion State на компонента Target Mover стойността Vertical. Дублирайте пак Target и преместете по X новия обект 4 м наляво, след което му задайте Horizontal. Съхранете сцената и проекта. Проиграйте.

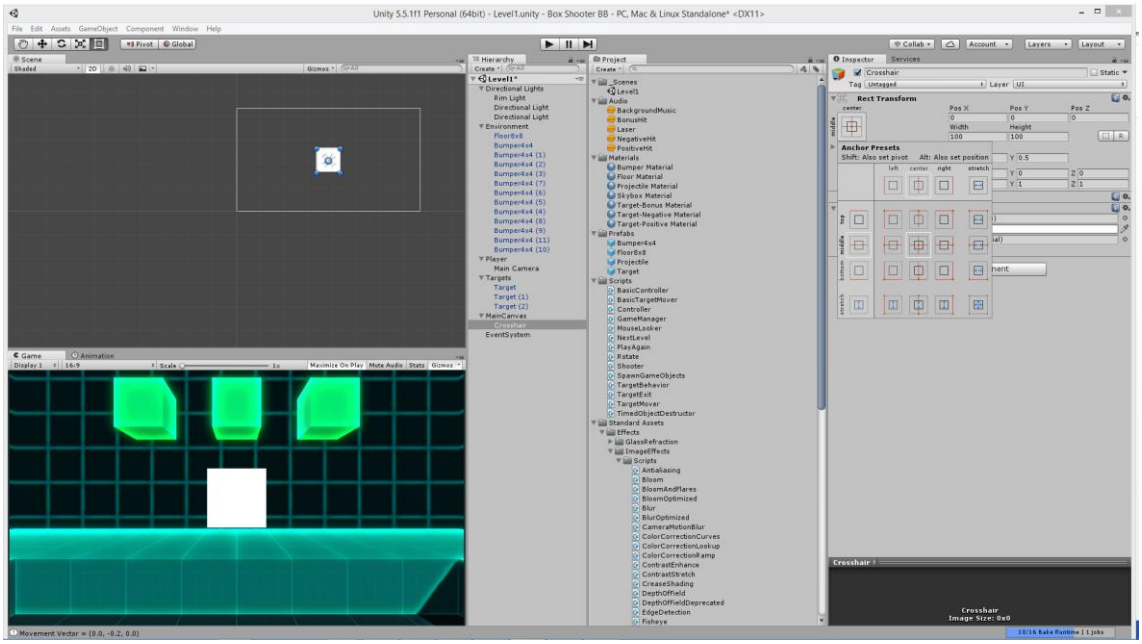
Стъпка 34 – създайте йерархията Targets като празен обект с нулеви координати. Преместете в него всички мишени.

Стъпка 35 – време е за стрелба. Най-смислено е да стреляме от самата камера. Добавете към камерата компонента Shooter. За Projectile задайте едноименния префаб! А за Shoot Sound Effect (Shoot SFX) задайте Laser. Проиграйте. С цел по-бърза стрелба, задайте Power=60 или повече.

Стъпка 36 – разгледайте в редактора скрипта (компонента) Shooter.

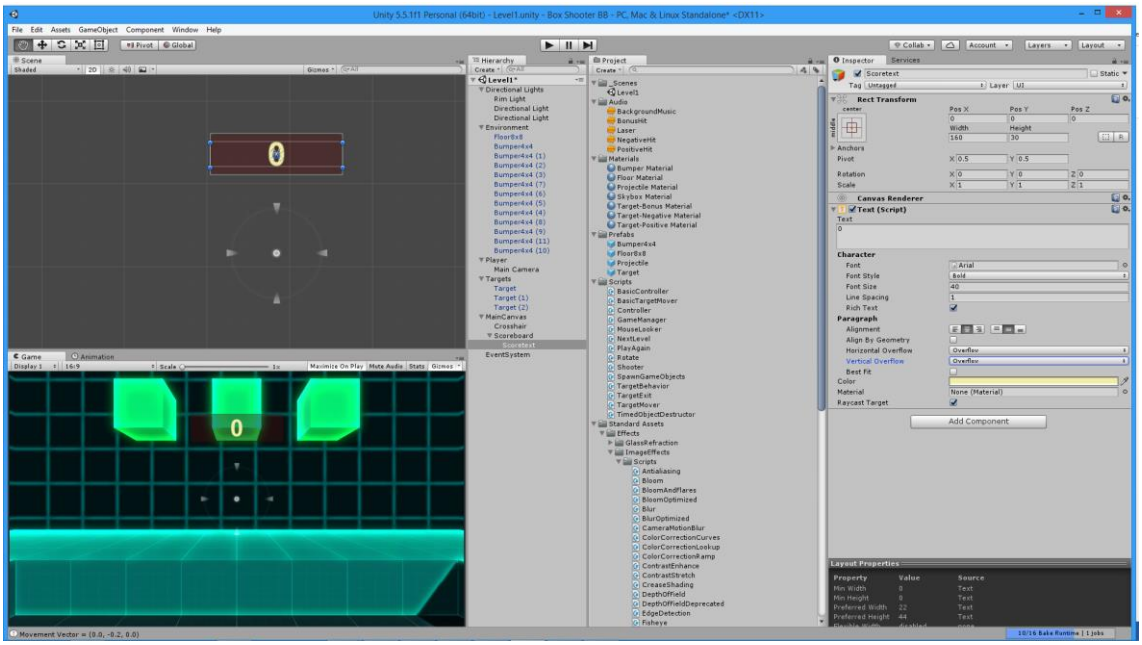
Стъпка 37 – трябва да унищожаваме снарядите след известно време, защото иначе остават видими в играта. За целта към префаба Projectile добавяме скрипта Time Oject Destructor и задаваме време на живот на снаряда 1-2 секунди.

Стъпка 38 – Създайте платно чрез меню GameObject > UI > Canvas и го преименувайте на MainCanvas. Под него (в йерархията му) добавете Image с име Crosshair и превключете в 2D mode. Позиционирайте картинката в средата на платното чрез задържане на Ctrl+Shift и избор на центъра в Ancor Presets (в инспектора). Задайте за Source Image картинката с име Crosshair, и установете Width=Height=144. Проиграйте.

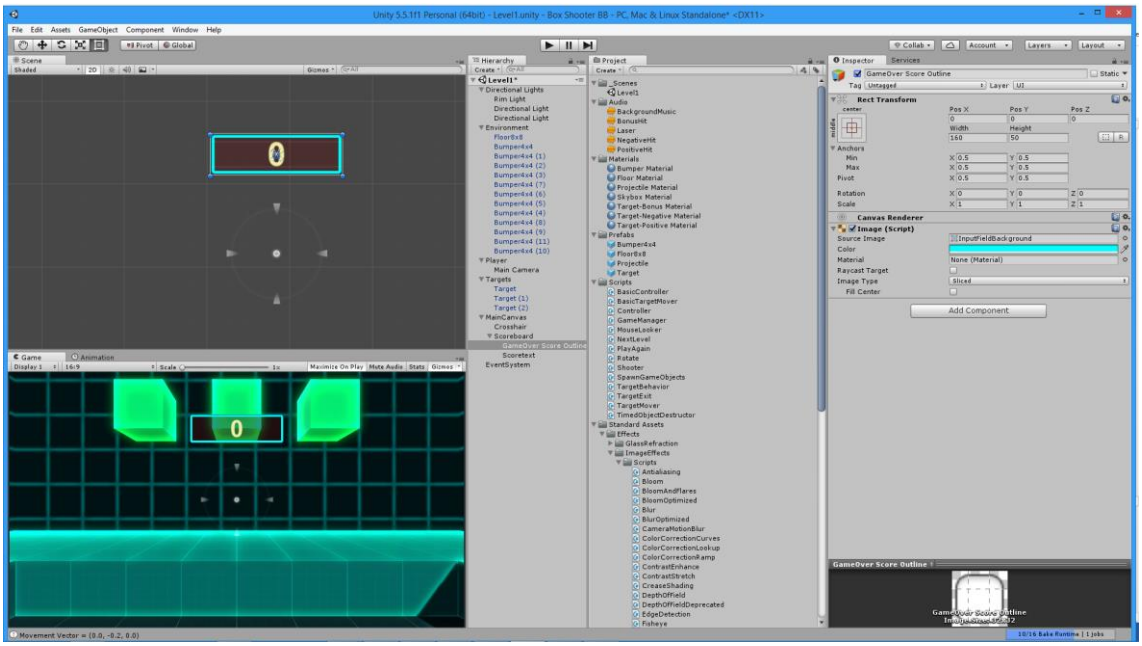


Стъпка 39 – Под MainCanvas (в йерархията му) добавете Image с име Scoreboard и превключете в 2D mode. Позиционирайте картинката в средата на платното чрез задържане на Ctrl+Shift и избор на центъра в Ancor Presets (в инспектора). Задайте за Source Image картинката с име InputFieldBackground (Color = Dark red, Alpha=100), и установете Y=120, Width=160, Height=50. Проиграйте.

Стъпка 40 – Под MainCanvas (в йерархията му) добавете Text с име Scoretext. Преместете го в йерархията на Scoreboard. Позиционирайте го в средата на платното чрез задържане на Ctrl+Shift и избор на центъра. Задайте за текста стойност „0“, удебелен шрифт, размер 40 и го центрирайте и сменете цвета му.

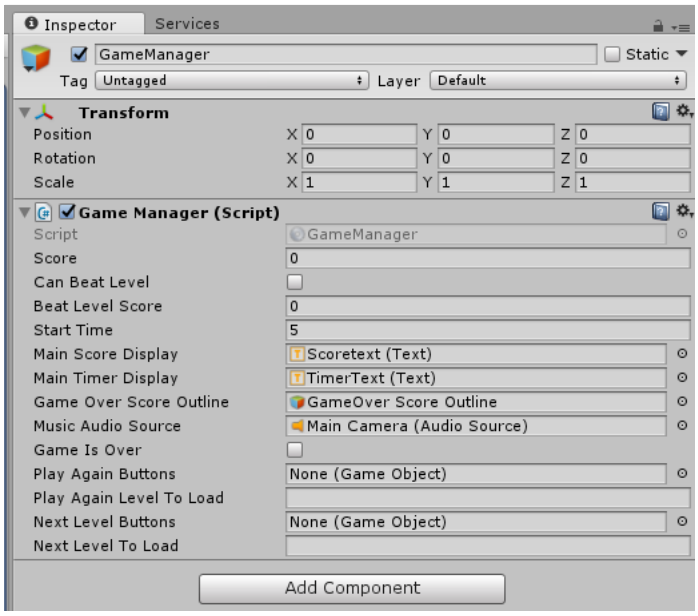


Стъпка 41 – Под MainCanvas (в йерархията му) добавете Image с име GameOver Score Outline и превключете в 2D mode. Преместете го в йерархията на Scoreboard. Позиционирайте го в средата на платното чрез задържане на Ctrl+Shift и избор на центъра в Ancor Presets (в инспектора). Задайте за Source Image картинката с име InputFieldBackground и останалите свойства както е показано по-долу. Накрая скрийте рамката GameOver Score Outline.



Стъпка 42 – Под MainCanvas (в йерархията му) добавете Text с име TimerText. Преместете го в йерархията на Scoreboard. Позиционирайте го в средата на платното чрез задържане на Ctrl+Shift и избор на центъра. Задайте Y=160 за текста стойност „15.0“, удебелен шрифт, размер 20 и го центрирайте и сменете цвета му.

Стъпка 43 – създайте празен обект с нулеви координати под името GameManager. Добавете към него скрипта Game Manager и го инициализирайте както е показано по-долу. Проиграйте.



Стъпка 44 – разгледайте в редактора скрипта (компонента) GameManager.

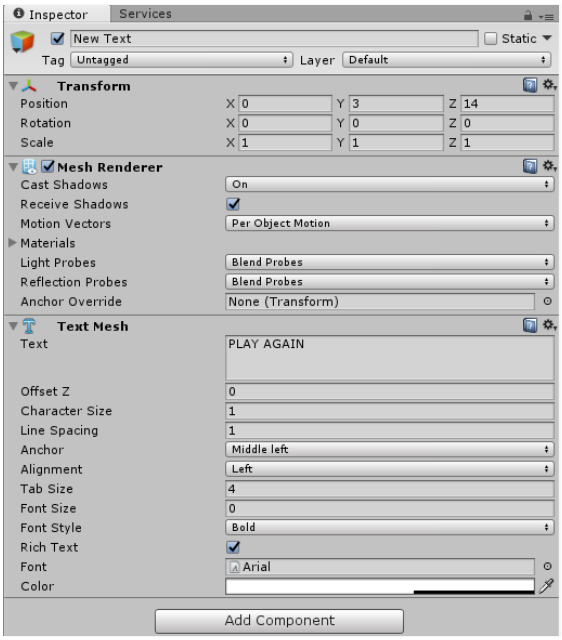
Стъпка 45 – превключете отново на 3D визуализация. Добавете в префаба на мишената Target компонента Target Behaviour. Понеже промяната е направена в префаба, тя се разпростира до всички негови екземпляри. За централната мишена задайте Time Amount=3, за лявата мишена - Score Amount=5, а за дясната - Score Amount=-10.

Стъпка 46 – разгледайте в редактора скрипта (компонента) Target Behavior. Обърнете внимание на редовете:

```
// only do stuff if hit by a projectile
        if (newCollision.gameObject.tag == "Projectile") {
                ....
```

За целта трябва да добавим маркер (таг) с име `Projectile` за префаба с име Projectile. Създайте такъв таг и го добавете. По време на игра забележете, че не всички точни попадения на снаряда поразяват целта. Затова променете свойството Collision Detection в RigidBody на Projectile така, че да има за стойност Continuous вместо Discrete (изисква повече процесорна мощност, но работи и за бързо движещи се обекти).

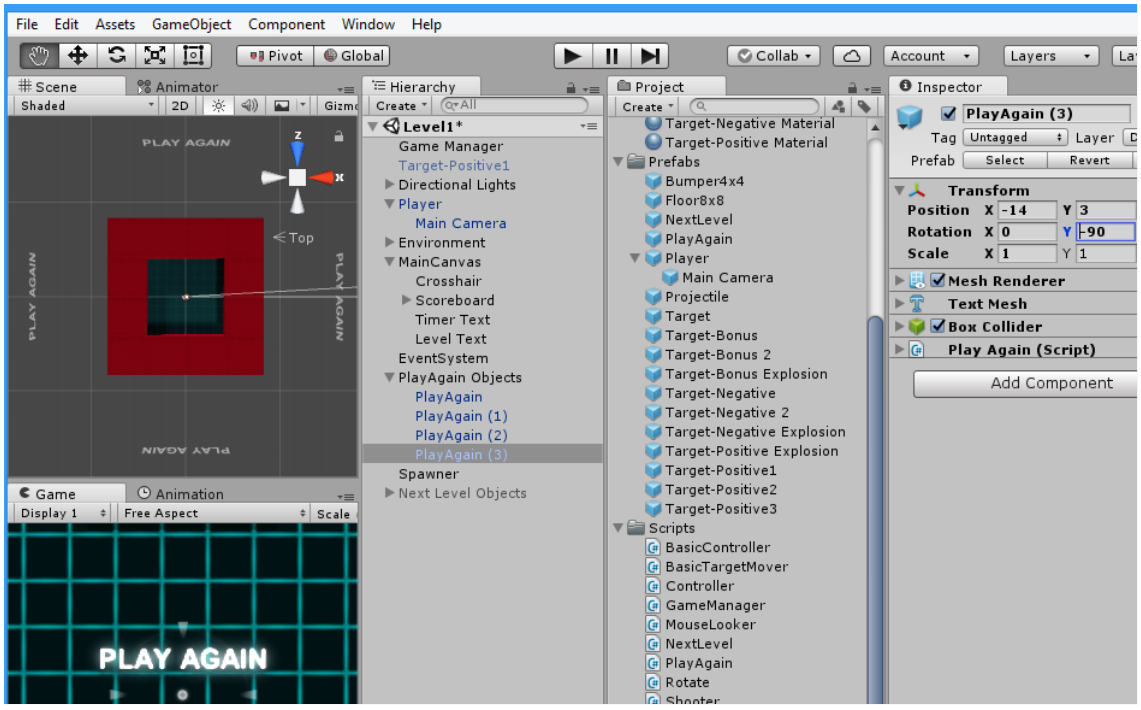
Стъпка 47 – Play Again: създайте триизмерен текст чрез менюто GameObject > 3D > 3D Text с име PlayAgain, и с координати (X, Y, Z)=(0, 3, 14). Задайте останалите свойства така, както е показано по-долу.



Стъпка 48 – за PlayAgain добавете компонент Box Collider, и за него задайте Z=1. За PlayAgain адайте скриптов компонент с име Play Again.

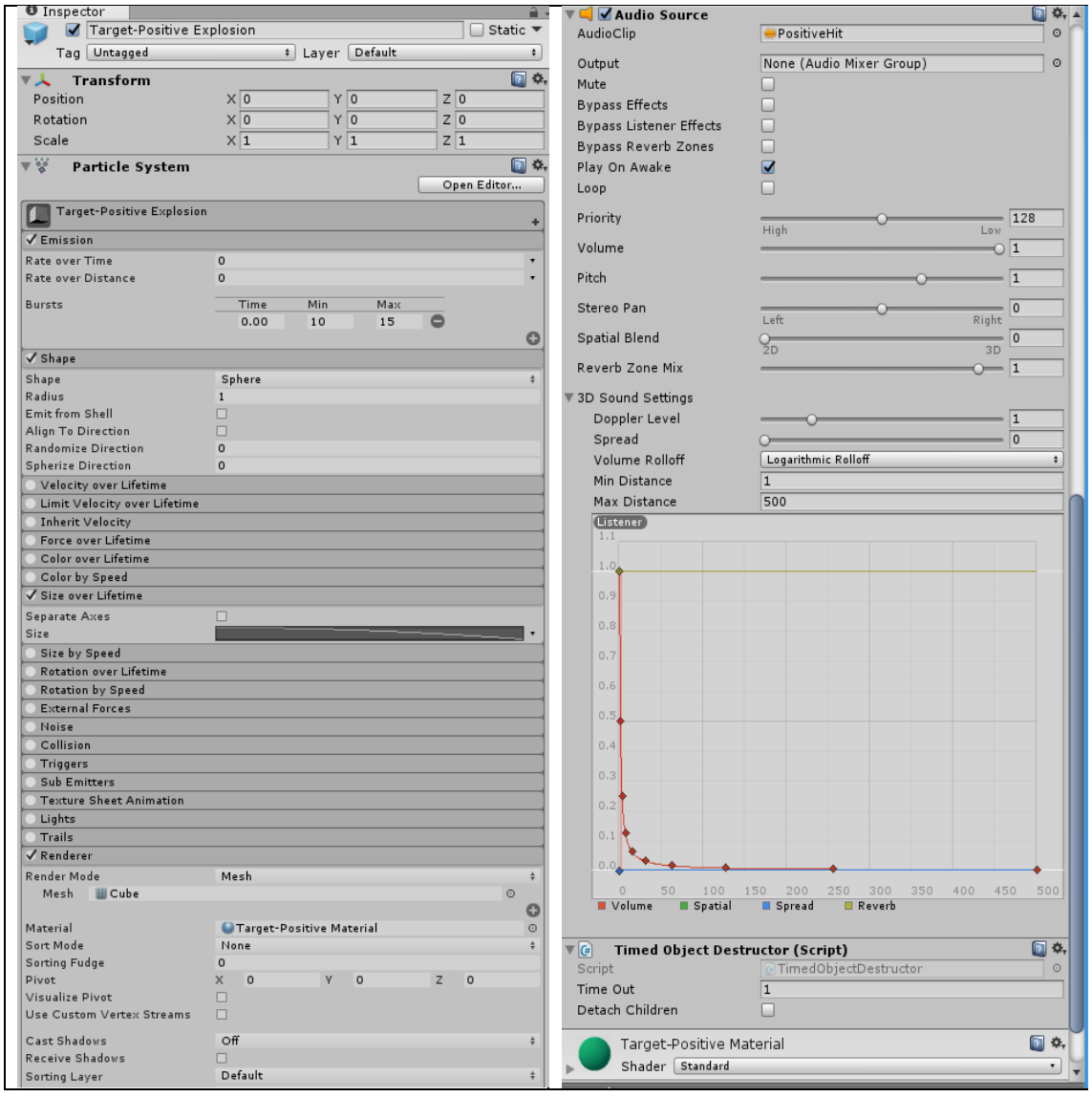
В GameManager за свойството Play Again Level to Load задайте текста „Level1“. Направете от обекта PlayAgain префаб, за да създадем после повече такива обекти при край на играта.

Създайте още три PlayAgain обекта по останалите три страни на играта, поставете всички PlayAgain обекти в йерархия, която да се върти чрез скрипта Rotate (Way=Around Y). Дезактивирайте ги и нека GameManager ги активира след края на играта.



Стъпка 49 – създайте система от частици чрез менюто GameObject > Particle System. Преименувайте я на Target Positive Explosion. Задайте им свойства съгласно фигурите по-долу.

Добавете и компонентите Audio Source с аудио клип PositiveHit, както и Time Object Destructor с цел разрушаването на системата от частици след 1 сек, като премахнете цикъла. Инициализирайте ги съгласно фигурите.



Стъпка 50 – създайте в Prefabs префаб на Target Positive Explosion (ще служи за обекти, добавящи точки)и след това изтриите обекта за този префаб.

Дублирайте Target Positive Explosion и го именовайте като Target Negative Explosion (ще служи за обекти, отнемащи време). Променете само два параметъра:

- в Renderer - Target Negative
- в Material Audio Source - NegativeHit.

Дублирайте Target Negative Explosion и го именовайте като Target Bonus Explosion (ще служи за обекти, добавящи време). Променете само два параметъра:

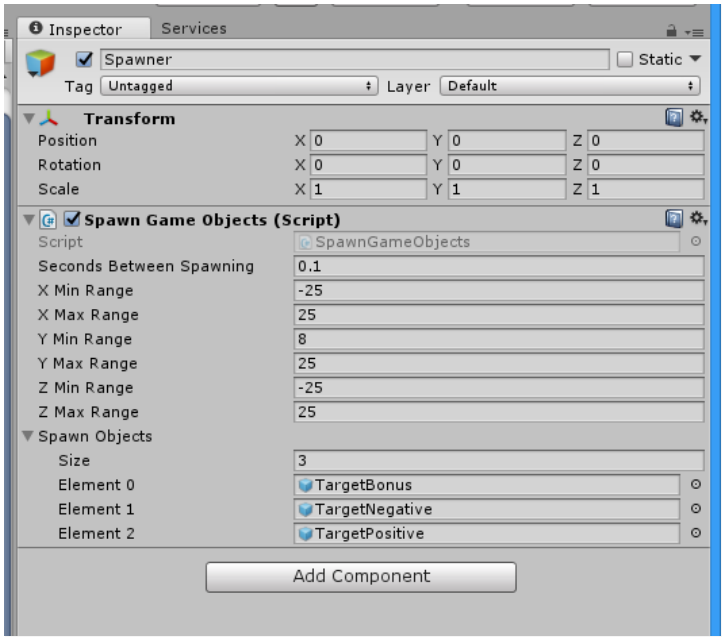
- в Renderer - Target Bonus
- в Material Audio Source - BonusHit.

Стъпка 51 – преименувайте трите мишени така:

- за централната (Time Amount=3) мишена задайте име TargetBonus
- за лявата мишена (Score Amount=5), задайте име TargetPositive
- за дясната - Score Amount=-10, задайте име TargetNegative

Стъпка 52 – направете прафаби от трите мишени и ги изтрийте. За трите префаба задайте съответни материали, както и параметри Explosion Prefab в скрипта Target Gehavior. За трите префаба добавете скриптовия компонент Timed Object Destructor, с Time Out = 3.

Стъпка 53 – променете в GameManager стартовото време от 5 на 15 сек. Създайте празен обект с нулеви координати под името Spawner (размножител) и добавете към него скрипта за размножаване на обектите с име SpawnGameObjects. Инициализирайте го съгл. фигурата по-долу.



Стъпка 54 – Съхранете, тествайте играта и разгледайте скрипта за размножаване на обектите с име SpawnGameObjects.