

Zajęcie 1. Modelowanie gry komputerowej 3D





Roll-a-ball – w oparciu na: <http://unity3d.com/learn/tutorials/projects/roll-a-ball/introduction>

Cele:

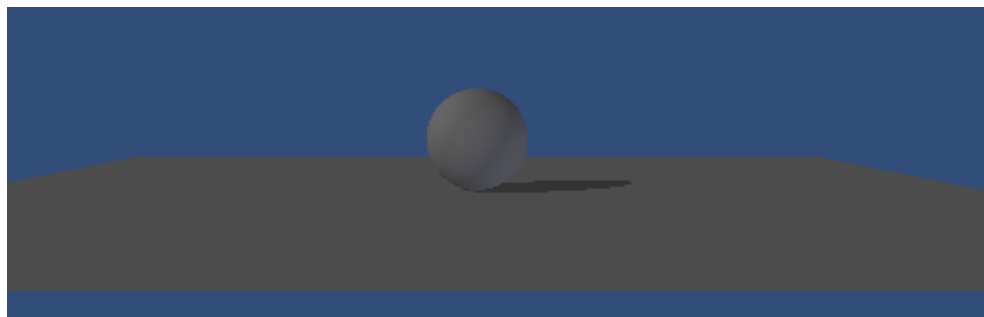
- Stworzyć grę Roll a Ball która wykorzystuje fizykę
- Realizować wejście z klawiatury
- Kolizje między obiektami
- Brak multimedia

Ustawienia gry

1. File-> New Project (Stworzyć folder projektu, np. lab1)
2. Unity 3D stworzy następujące foldery w projekcie

 Assets	1/21/2015 12:07 PM	File folder
 Library	1/21/2015 12:07 PM	File folder
 ProjectSettings	1/21/2015 12:07 PM	File folder
 Temp	1/21/2015 12:07 PM	File folder

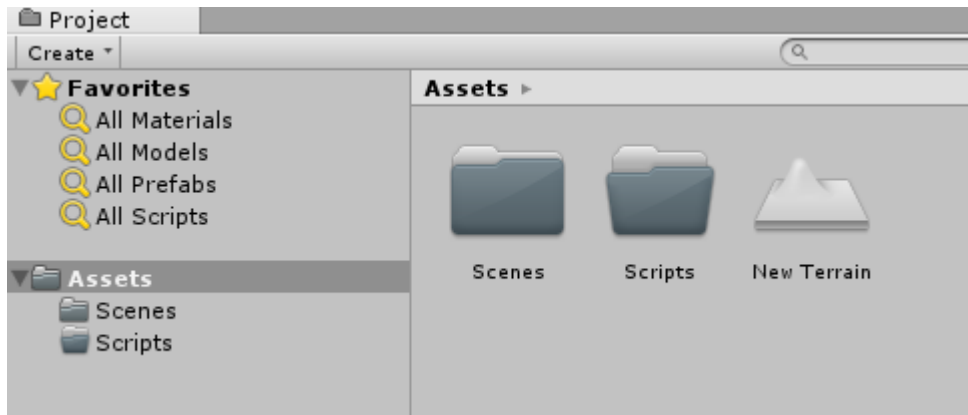
3. File->Save Scene, i stworzyć nowy folder w Assets jako "Scenes", i zachować plik jako "scene1"
4. Stworzyć powierzchnię
5. Stworzyć obiekt gry jako kulę. W tym celu GameObject->Sphere
 - a. Zmienić nazwę obiektu na 'Player'
 - b. Jeżeli obiekt jest niewidoczny naciśnij 'f' żeby sfokusować na ten obiekt.
 - c. Ustawić pozycję obiektu na 0,1,0
6. Stworzyć światło kierunkowe dla sceny GameObject->Light->Directional light.
 - a. Intensywność 0.1.
 - b. Pozycja 0,5,0
 - c. Kąt 30,60,0
 - d. Shadow Type -> Soft Shadows
 - e. Resolution -> Very High Resolution
7. Stworzyć jeszcze jedno źródło światła za pomocą duplikacji (Ctrl+D) bieżącego światła kierunkowego
 - a. Zmienij nazwę na fill light
 - b. Shadowtype-> No Shadows
 - c. Ustalij kolor na blue-ish tint.
 - d. Zmienij kąty na -30,-60,0.



Zachowaj scenę.

Ruchy playera

1. Wybierz Player. W inspektorze "Add Component" Wybierz Physics->Rigid Body
2. Teraz stworzymy foldery dla zachowania skryptów. Na panelu Project stworzymy nowy folder jako 'scripts'.



3. Teraz stworzymy dla obiektu komponentę który będzie skryptem C#.
4. Dla edycji skryptu wykorzystujemy silnik skryptowy MonoDevelop.
 - a. Zauważymy 2 metody w skryptu
 - a.i. Start() – działania w ciągu inicjalizacji (podobnie jako konstruktor)
 - a.ii. Update() – działania w każdej chwili gry dotyczące obiektu.
 - b. Możemy teraz usunąć Start albo zachować go dla przyszłości.
5. Oto script

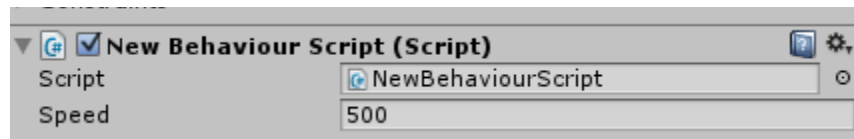
```
public class NewBehaviourScript : MonoBehaviour {

    public float speed;

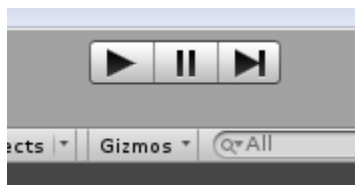
    // Update is called once per frame
    void Update () {
        // Record input from our player.
        float moveHorizontal = Input.GetAxis ("Horizontal");
        float moveVertical = Input.GetAxis ("Vertical");

        // Use the input we get to move the rigid body.
        Vector3 movement = new Vector3 (moveHorizontal, 0.0f, moveVertical);
        // Make a smooth movement by multiplying by Time.deltaTime.
        // Adjust our 'movement' vector by 'speed' to make game playable.
        rigidbody.AddForce (movement * speed * Time.deltaTime);
    }
}
```

- a. Zachowaj script w MonoDevelop. Unity kompiluje ciągle.
- b. Wróć do Unity3D. Zachowaj scenę
- c. Zauważ że obiekt 'Player' ma pole dla prędkości. Ustalimy go 500.



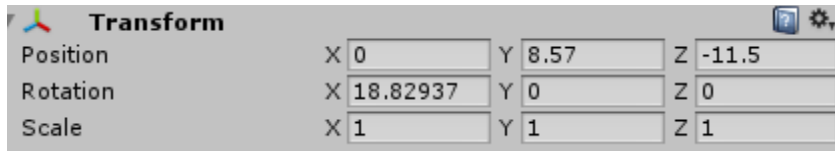
- d. Spróbujmy grę naciskając przycisk play.



Kamera

Stworzymy nową kamerę w celu wyświetlania gracza (player).

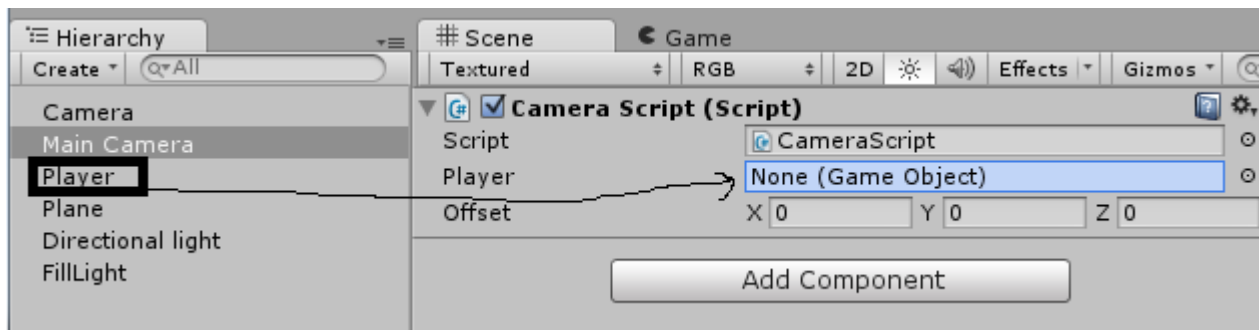
1. Game Object -> Camera



2. Stworzymy script (Add Component->Script->CSharp, i zachowajmy go w folderze Scripts)

```
public class CameraScript : MonoBehaviour {  
  
    // Reference the player  
    public GameObject player;  
    // Position of our camera  
    public Vector3 offset;  
  
    void Start() {  
        // Where we moved the camera in our scene.  
        offset = transform.position;  
    }  
  
    // LateUpdate is called once per frame  
    // But at the end of a frame.  
    void LateUpdate () {  
        // Our initial position, plus the updated player position.  
        transform.position = player.transform.position + offset;  
    }  
}
```

3. Przeciągnij Player do właściwości Player dla Camera Scripts. W taki sposób stworzymy referencję do GameObject (w tym przypadku our Sphere).

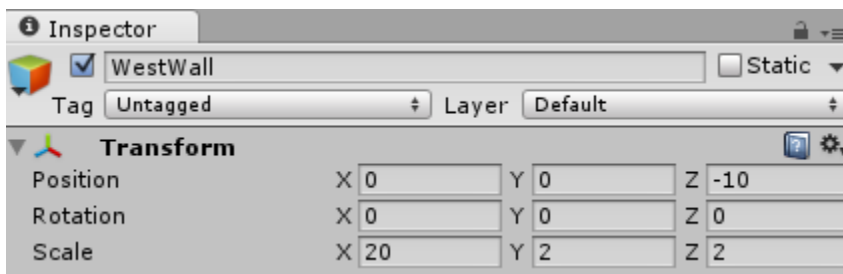


Tworzymy ściany

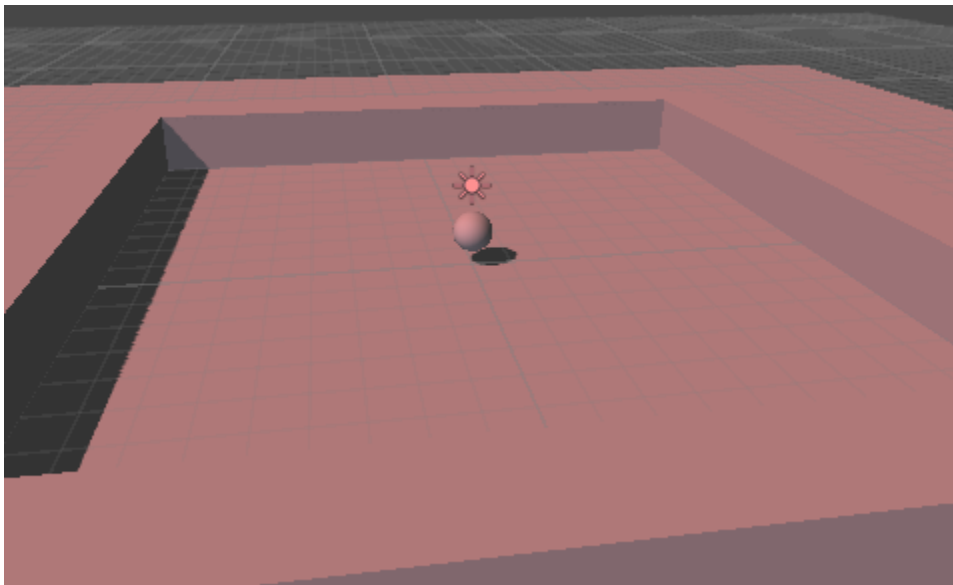
1. Stworzymy nowy GameObject jako Walls. Poniżej w hierarchii stworzymy cztery sześciany i modykujemy je żeby stworzyć ściany.



2. Dla West Wall to są parametry:

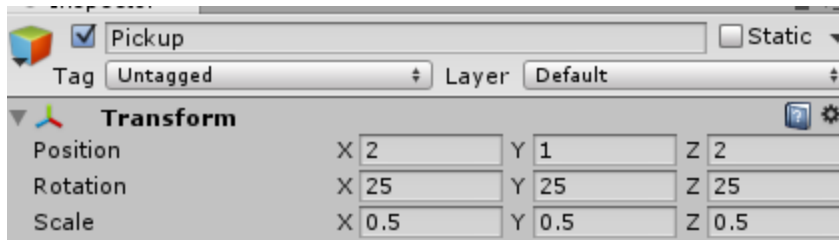


3. Scena ma taką postać



Stworzymy cele

1. Stworzymy nowy GameObject który jest sześcianem i nazywa się 'pickup'
2. Parametry są

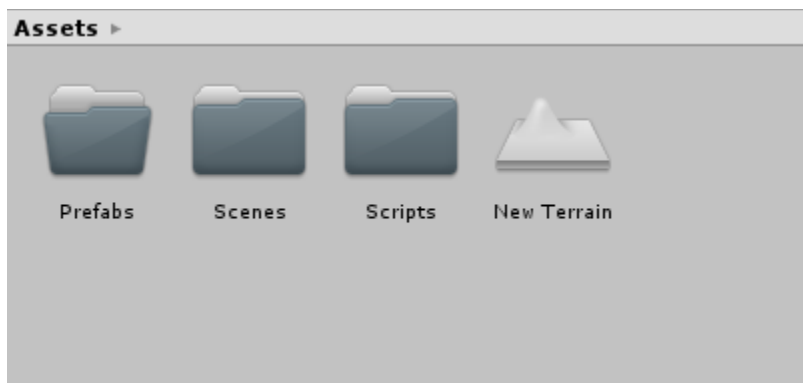


3. Lets now create a script called 'RotatingCube' for the pickup object to make it spin. Lets now open it up in MonoDevelop Stworzymy teraz skrypt o nazwie "RotatingCube 'dla obiektu pickup, aby kręcić go. Otworzymy go w MonoDevelop

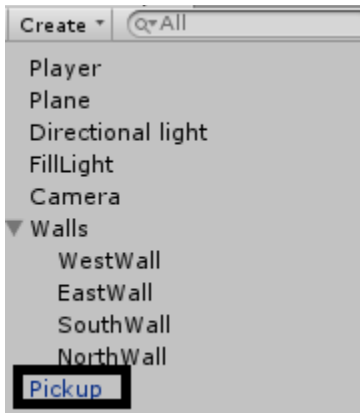
4. Oto skrypt.

```
public class RotatingCube : MonoBehaviour {  
  
    // Update is called once per frame  
    void Update () {  
        transform.Rotate (new Vector3 (15, 30, 45) * Time.deltaTime);  
    }  
}
```

- 5.
6. Po stworzeniu jednego obiektu stworzymy jeszcze kilka z nich. Mamy zamiar zrobić to na podstawie "prefabrykatów". "Prefabrykat" jest wielokrotnego użytku, który występuje jako plan obiektu. Możemy używać go w dowolnym miejscu w naszym projekcie!
7. W tym celu stworzymy nowy folder o nazwie 'prefabs'.

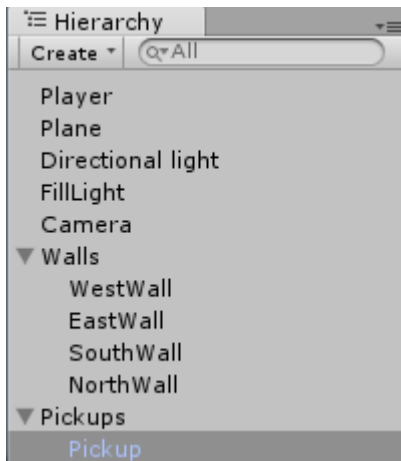


8. Przeciągnij obiekt w folder z hierarchii obiektów.

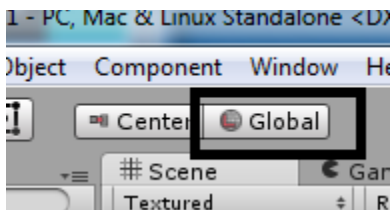


9.

10. Stworzymy pusty GameObject o nazwie 'pickups'. Ulokujcie ten obiekt do początku układu współrzędnych. Teraz przesuniecie nasz pick-up do obiektu, w taki sposób, że to - dziecko.



11. Now we want to duplicate this object, and move it around. To move it around in a global coordinate space, toggle the coordinate space to global.

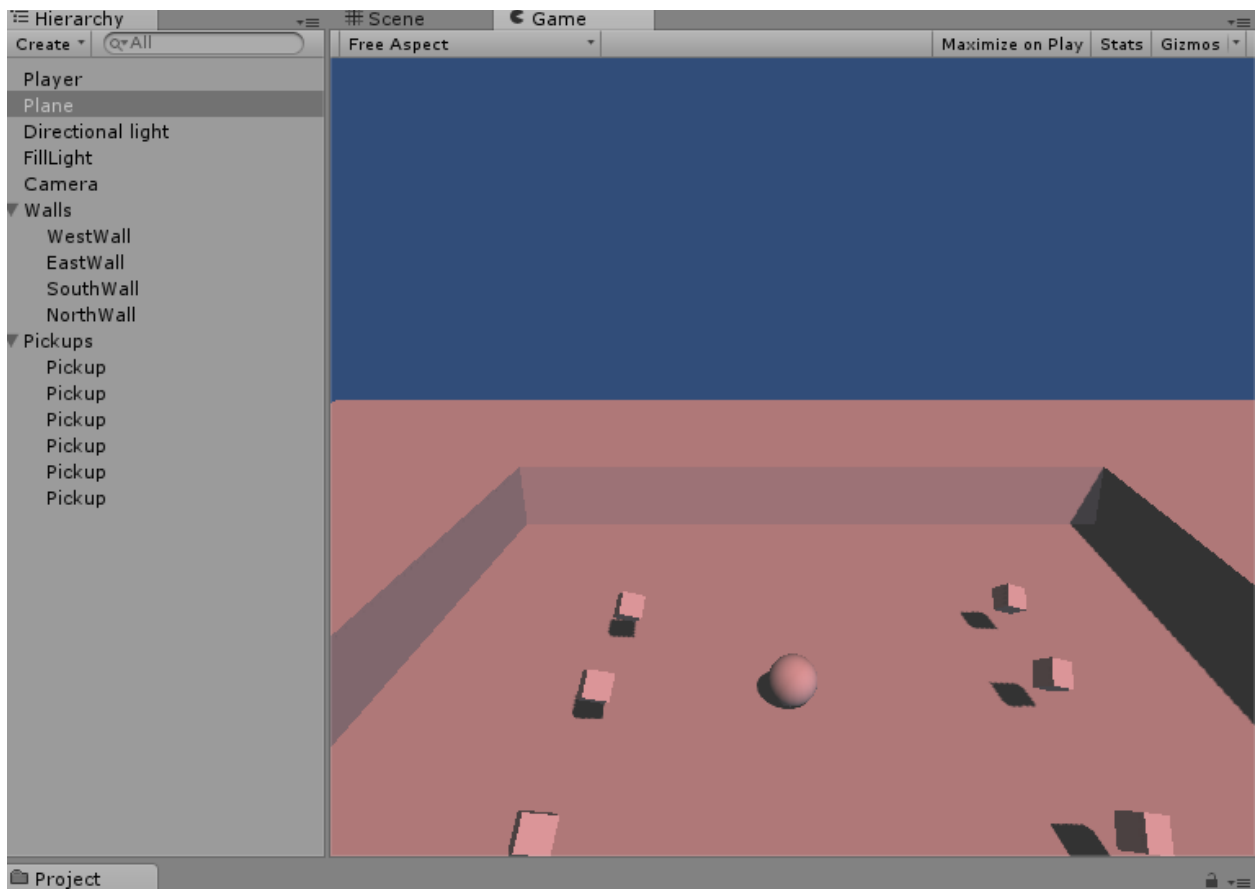


12. Now we manipulate the object, in regards to the world, instead of its own rotation, scale, and position.

13. Create several of them by duplicating them (note, since they are prefabs, they will already have scripts attached to them and be ready to go!)

14. Your scene should look something like this:

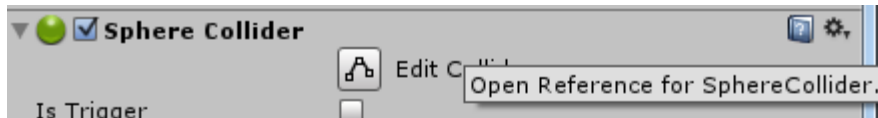
15.



Collecting Objects

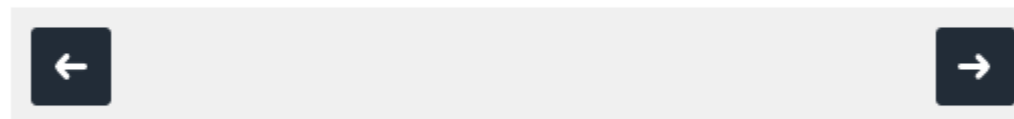
In order to win the game, the player will collide with the objects and pick them up. When they have picked up all the objects, the game is over. In order to do this, we will learn about colliding with the correct object, and picking it up.

1. Open up our original player script that we wrote.
2. (Stepping back, this is how to learn about what methods are available)



Click on the 'Book' for help

[Unity Manual](#) / [Physics](#) / [3D Physics Reference](#) / Sphere Collider



Sphere Collider

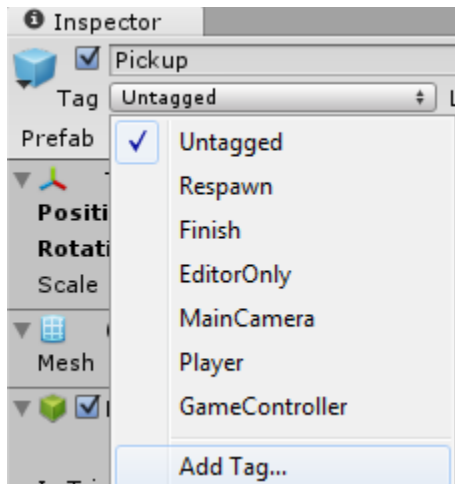
SWITCH TO SCRIPTING

Your browser brought you to the reference page. Then click on this to learn about class methods, variables, messages, etc.

- 3.
4. In our script, add the following method.

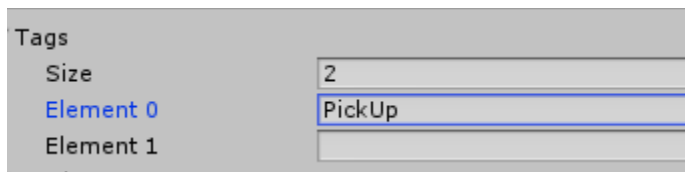
```
void OnTriggerEnter(Collider other){  
    // Check if our object collides with a pickup object.  
    if (other.gameObject.tag == "PickUp") {  
        other.gameObject.SetActive(false);  
    }  
}
```

- 5.
6. Now we must 'tag' our object so that Unity knows about it. To do this, select one of our 'pickup' objects and then find the 'Tag' item, and 'Add Tag'



7.

8. We then add the name of our tag (NOTE: This is case sensitive, and must exactly match what is in our code, because C# is a case sensitive language).



9.

10. Then return to our 'pickup' object

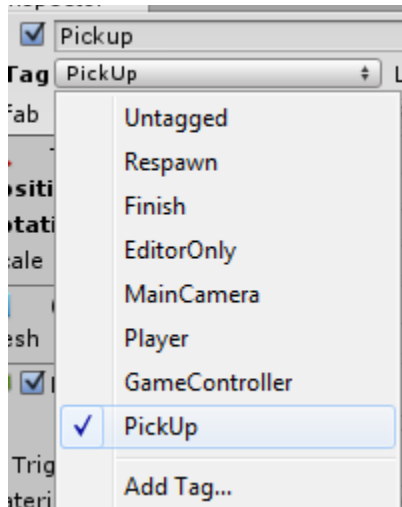
11. We then hit 'select' for our prefab.



12.

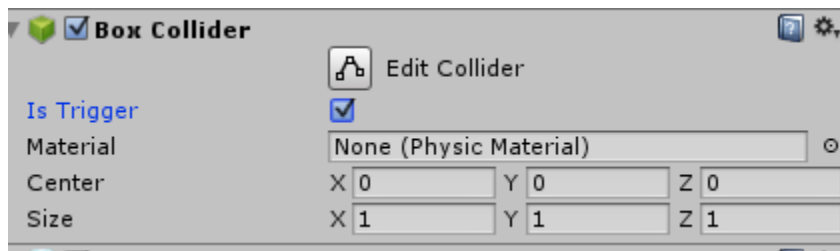
13. This ensures we make changes to every game object that is of this type.

14. Finally we tag our object.



15.

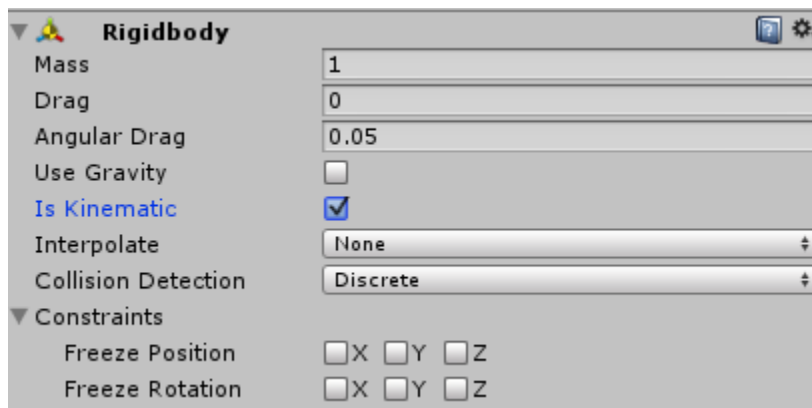
16. The final step for our prefab, is to then make sure that it is a trigger. We want our objects geometry to act as a trigger, as opposed to physical collisions.



17.

18. Side note: Small performance optimization to add a rigid body for our pickup object.

- a. Is Kinematic makes objects ignore physics simulations.
- b. Rigid body avoids excess computation for every frame on dynamic objects.



19.

Winning the Game

1. When we have picked up all of the objects, we will have won the game. Lets make a private variable in our player script that gets updated every time we collide with an object.
2. Final code:

```
public float speed;
private int count;

void Start(){
    count = 0;
}

// Update is called once per frame
void Update () {
    // Record input from our player.
    float moveHorizontal = Input.GetAxis ("Horizontal");
    float moveVertical = Input.GetAxis ("Vertical");

    // Use the input we get to move the rigid body.
    Vector3 movement = new Vector3 (moveHorizontal, 0.0f, moveVertical);
    // Make a smooth movement by multiplying by Time.deltaTime.
    // Adjust our 'movement' vector by 'speed' to make game playable.
    rigidbody.AddForce (movement * speed * Time.deltaTime);
}

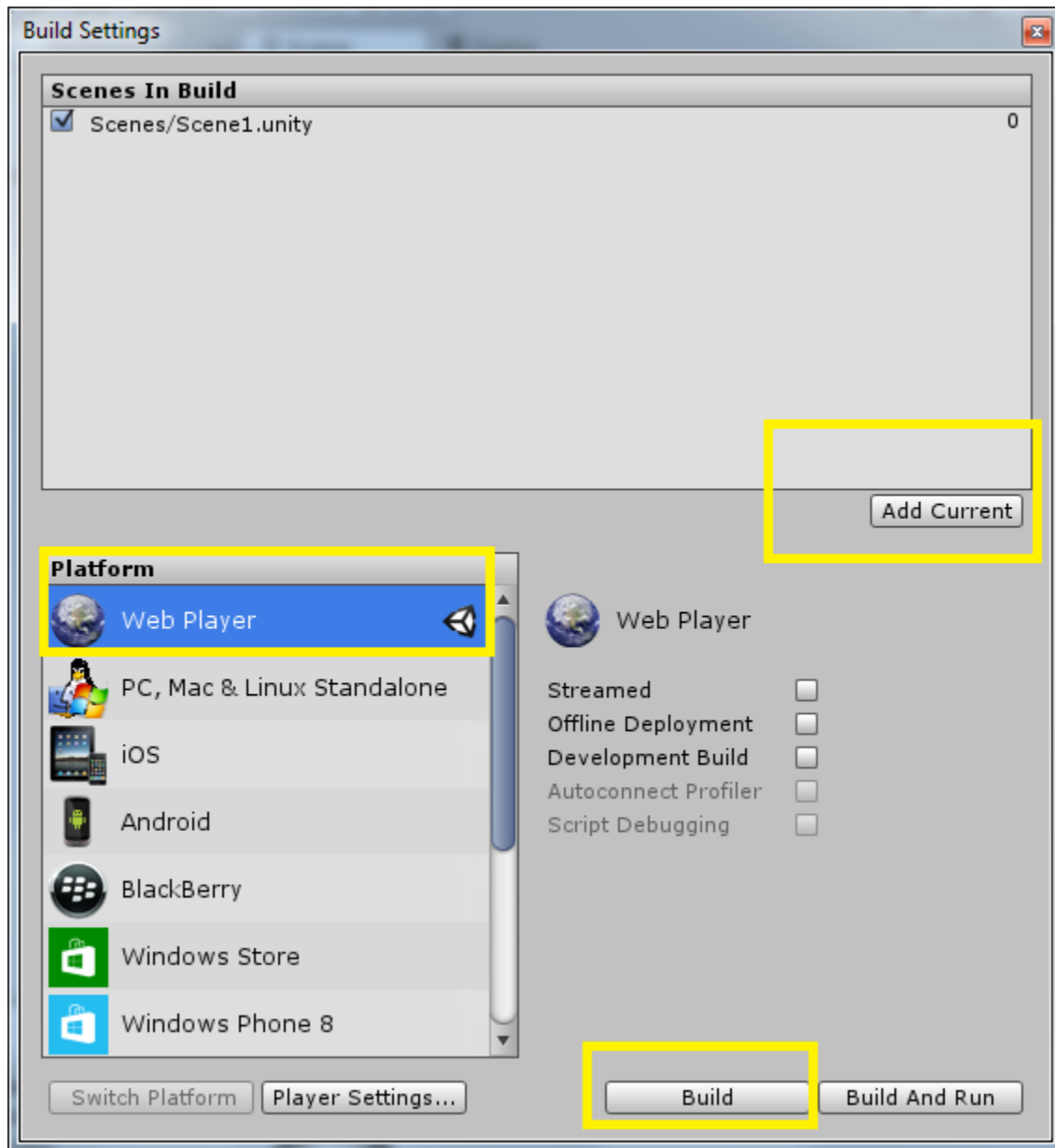
void OnTriggerEnter(Collider other){
    // Check if our object collides with a pickup object.
    if (other.gameObject.tag == "PickUp") {
        other.gameObject.SetActive(false);
        count += 1;
    }
}
```

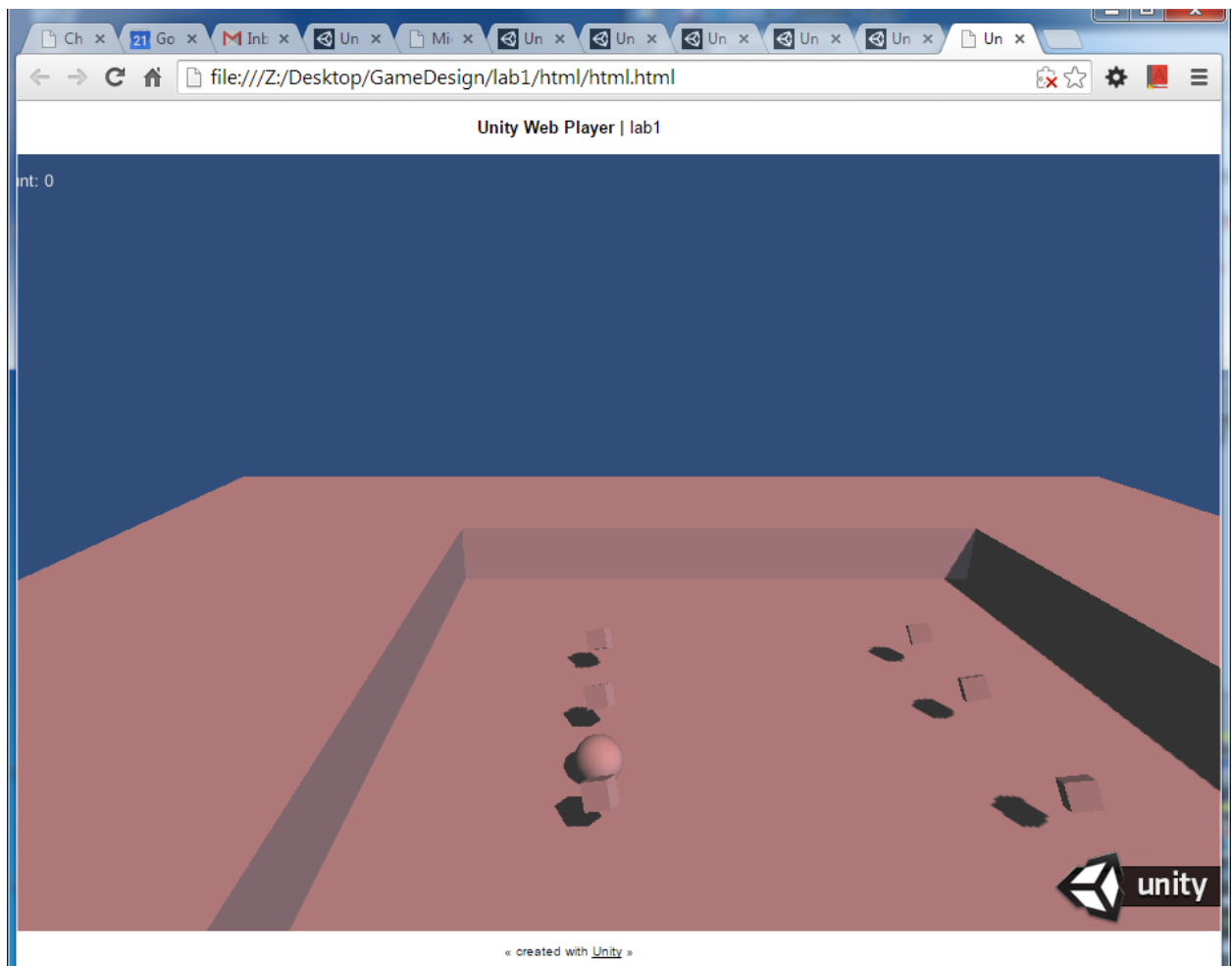
3. Lets now display the code for our user to see how well they are doing.

4.

5. SAVE all of your scripts and your Unity3D scene.
6. Play your game!!

For an Extra Cool Factor (Show your friends, export to the web!)





Do go home and install the android SDK or if you're on a Mac, install the iOS XCode Tools and deploy!

Warianty zadania

Opracować grę typu Roll a Ball z takimi parametrami

Wariant 1. Kolor materiału „playera” – niebieski,

obiekty „pick up” dwóch typów

1 typ obiektu „pick up” - cylinder,

2 typ obiektu „pick up” - capsule,

ilość obiektów „pick up” typu 1 - 9,

ilość obiektów „pick up” typu 2 - 3,

Kolor materiału obiektów „pick up” typu 1 – żółty,

Kolor materiału obiektów „pick up” typu 2 – czerwony,

Kolor materiału „ścian” - zielony

Reguły gry:

trafianie w obiekt typu 1 – 1 punkt

trafianie w obiekt typu 2 – 3 punkty

warunek zakończenia gry – 9 punktów

Wariant 2. Kolor materiału „playera” – czerwony,

obiekty „pick up” dwóch typów

1 typ obiektu „pick up” - sześciąt,

2 typ obiektu „pick up” - kula,

ilość obiektów „pick up” typu 1 - 8,

ilość obiektów „pick up” typu 2 - 4,

Kolor materiału obiektów „pick up” typu 1 – niebieski,

Kolor materiału obiektów „pick up” typu 2 – brązowy,

Kolor materiału „ścian” - pomarańczowy

Reguły gry:

trafianie w obiekt typu 1 – 1 punkt

trafianie w obiekt typu 2 – 2 punkty

warunek zakończenia gry – 10 punktów

Wariant 3. Kolor materiału „playera” – zielony,

obiekty „pick up” dwóch typów

1 typ obiektu „pick up” - kapsle,

2 typ obiektu „pick up” - cylinder,

ilość obiektów „pick up” typu 1 - 7,

ilość obiektów „pick up” typu 2 - 5,

Kolor materiału obiektów „pick up” typu 1 – żółty,

Kolor materiału obiektów „pick up” typu 2 – brązowy,

Kolor materiału „ścian” - niebieski

Reguły gry:

trafianie w obiekt typu 1 – 1 punkt

trafianie w obiekt typu 2 – 4 punkty

warunek zakończenia gry – 9 punktów