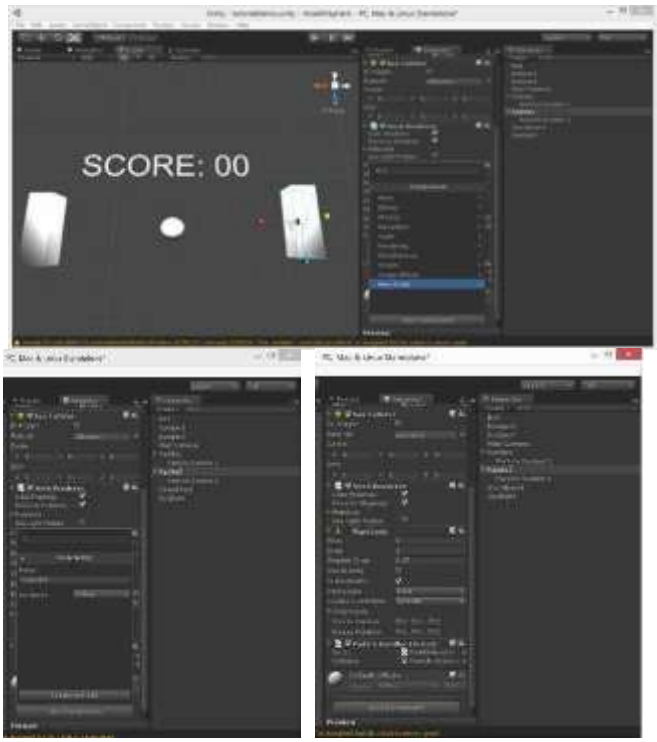
Βασικές εισαγωγικές έννοιες

Το λογισμικό Unity υποστηρίζει scripts για προγραμματισμό παιχνιδιών στις γλώσσες C++ και Java. Η λέξη προγραμματισμός δεν πρέπει να "τρομάζει", μιας και είναι αρκετά εύκολο για τους χρήστες του λογισμικού να προσαρμοστούν στη δομή κα τις εντολές στην αντίστοιχη γλώσσα αν έχουν κατανοήσει το μηχανισμό δημιουργίας και εφαρμογής τμημάτων κώδικα σε αντικείμενα και έχουν στο μυαλό τους το πλάνο λειτουργικότητας του εκάστοτε παιχνιδιού. Ένας εύχρηστος οδηγός εκμάθησης είναι το περιβάλλον C# Language Primer Microsoft. Κάποιος που έχει γνώσεις προγραμματισμού, μπορεί πολύ εύκολα να προσαρμοστεί στους συντακτικούς και γραμματικούς κανόνες της κάθε γλώσσας, καθώς και στις διαθέσιμες συναρτήσεις μέσω των αντίστοιχων βιβλιοθηκών.

Η βασική ιδέα του αντικειμενοστραφούς προγραμματισμού (object oriented programming) είναι πως τα παραγόμενα προγράμματα αποτελούνται από δομικές μονάδες που ονομάζονται αντικείμενα, κάθε ένα από τα οποία διαθέτει τις δικές του ιδιότητες, μεταβλητές και συναρτήσεις, μέσω των οποίων μπορεί να αλληλεπιδρά με άλλα αντικείμενα. Τα αντικείμενα (Objects) προέρχονται από κλάσεις (classes), δηλαδή δομές-αρχεία τα οποία "δηλώνουν" τα χαρακτηριστικά τους. Στο περιβάλλον της Unity 3D, είναι πολύ εύκολη διαδικασία να εφαρμοστεί/συνδεθεί κώδικα (script) σε/με ένα αντικείμενο της σκηνής μέσω του κουμπιού Add component που βρίσκεται κάτω δεξιά στο περιβάλλον εργασίας:



- Πρώτα επιλέγουμε το αντικείμενο της σκηνής στο οποίο θέλουμε να εφαρμοστεί το script και μέσω της εργαλειοθήκης **inspector** κάνουμε κλικ στο κουμπί. Then, click on the **'Add Component'**.
- Επιλέγουμε **'new script',** δίνουμε ένα όνομα και πατάμε το κουμπί **'create and add'.**
- Στην περιοχή Assets εμφανίζεται ένα νέο "αντικείμενο-αρχείο", το οποίο αποτελεί το τμήμα κώδικα που δημιουργήθηκε. Κάνοντας διπλό κλικ πάνω του, μπορούμε να το επεξεργαστούμε μέσω του ενσωματωμένου editor του λογισμικού Unity 3D

Η δομή ενός αρχείου class π.χ. για τον έλεγχο της κατάστασης υγείας ενός παίχτη θα ήταν η ακόλουθη:

```
using UnityEngine;
public class Mook : MonoBehaviour
{
   private float health;
     void Start ()
     {
        health = 100;
     }

     void Update()
      {
         if (health > 0)
                {
                   //search for player
                   //if you encounter the player on the road, kill him
                   //if you get shot, remove a random amount of health
                 }
        }
}
```

Ας εξηγήσουμε λίγο τα εμπλεκόμενα στοιχεία:

- ***using UnityEngine;*** *-Η γραμμή αυτή καθορίζει στη γλώσσα C τη χρήση της συγκεκριμένης βιβλιοθήκης της Unity που περιέχει τη "αναπαραγωγής" του παιχνιδιού".*
- ***public class Mook :*** MonoBehaviour - *Ορίζεται μια κλάση τύπου MonoBehaviour με το όνομα ("Mook")*
- ***private float health;*** *-Ορίζεται μια private μεταβλητή κλάσης (class variable) η οποία μπορεί να χρησιμοποιηθεί μόνο από τα στοιχεία εντός της κλάσης*
- ***void Start ()*** *- Ορίζεται η "ειδική" μέθοδος 'Start', οποία τρέχει μόνο μια φορά (στην αρχή του παιχνιδιού) κι ουσιαστικά καθορίζει την κατάσταση εκκίνησης του παιχνιδιού (π.χ. θέσεις αντικειμένων, χρώμα φόντου, κλπ)*
- ***void Update()*** *- Άλλη μια ειδική μέθοδος, η οποία όμως είναι η πιο βασική γιατί "τρέχει" συνέχεια, σε κάθε frame και εντός των ορίων της (ο χώρος ανάμεσα στα σύμβολα {}) γράφεται ο κώδικας που θα καθορίζει την εξέλιξη του παιχνιδιού!*
- *// -> Οι γραμμές που ξεκινούν με αυτά τα σύμβολα είναι σχόλια, βοηθητικά λόγια για τον προγραμματιστή, τα οποία όμως αγνοούνται τελείως από τον Compiler.*

Εκτός από την παραπάνω βασική δομή, ο προγραμματιστής μπορεί να δηλώσει δικές του μεθόδους, δίνοντας δική του ονομασία και χρησιμοποιώντας μεταβλητές που θα ορίσει και θα χρησιμοποιήσει ο ίδιος με δικό του τρόπο. Είναι σημαντικό να μπορούμε να καταλαβαίνουμε ποιες μέθοδοι και μεταβλητές είναι του λογισμικού και ποιες προέρχονται από τον προγραμματιστή-χρήστη.

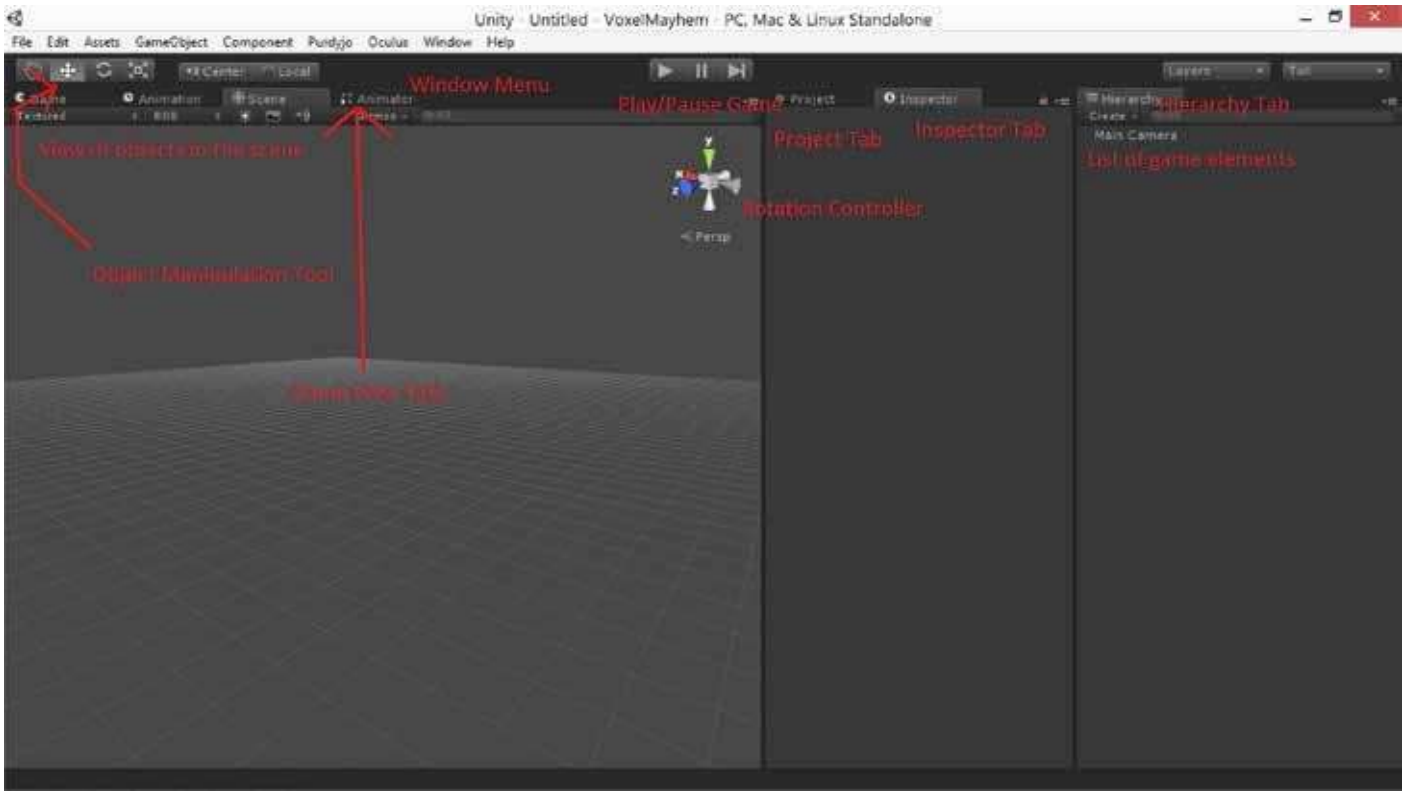
Για παράδειγμα η παρακάτω συνάρτηση δηλώνεται από το χρήστη και υλοποιεί την πρόσθεση δύο αριθμών:

```
public float addTwoNumbers(float a, float b){ return a+b; }      Πρωτότυπο συνάρτησης
float result = addTwoNumbers(1,2);                               Κλήση συνάρτησης
myClass instance; float result = instance.addTwoNumbers(1, 2);   Κλήση συνάρτησης που
υλοποιήθηκε σε μια άλλη κλάση
```

Σημαντικό επίσης είναι να καταλάβουμε πως αν η μέθοδος έχει συνδεθεί με ένα αντικείμενο με ειδικές ιδιότητες οι οποίες δεν μπορούν να προσπελαστούν από την έτοιμη μέθοδο/συνάρτηση **GameObject** , τότε μπορεί να χρησιμοποιηθεί η εξειδικευμένη μέθοδος **GetComponent**:

```
GetComponent<ParticleSystem>().Play();
```

Προκειμένου να περάσουμε στο δημιουργικό κομμάτι, ας θυμηθούμε πάλι τις εργαλειοθήκες του περιβάλλοντος Unity 3D.



Στην άσκηση αυτή θα υλοποιηθεί ένα απλό, διαδραστικό παιχνίδι μεταξύ υπολογιστή-χρήστη,κατά το οποίο δύο μπάρες ελέγχου που ανεβοκατεβαίνουν προσπαθούν να ανταλλάξουν "μπαλιές" με σκοπό η μία να προλαβαίνει τις επιστροφές μπάλας της άλλης. Αξίζει τον κόπο να δείτε στο ακόλουθο σύνδεσμο Pong, πώς αυτό το παιχνίδι μπορεί να κατασκευαστεί με τεχνικά μέσα, προγραμματίζοντας ένα μικροελεγκτή.