Scripting σε Unity

Μέρος 1ο

Κανόνας

class = component = class = script

Ορατότητα πεδίων

- Κάθε script που φτιάχνουμε αποτελείται από μια public κλάση η οποία περιέχει πεδία που μπορεί να είναι public, private ή σε άλλο επίπεδο ορατότητας.
- Στα scripts του Unity η ορατότητα έχει και πιο πρακτική σημασία, καθώς αν κάποιο πεδίο σημειωθεί ως public θα είναι προσβάσιμο από το αντίστοιχο component του στον editor.

Awake(), Start() & Update()

- Κατά τη δημιουργία ενός script το αρχείο έχει ήδη μέσα δύο άδειες συναρτήσεις: τη συνάρτηση Start() και τη συνάρτηση Update().
- Όπως δηλώνουν και τα σχόλια πάνω από κάθε συνάρτηση, η Start() καλείται μια φορά όταν ξεκινάει το παιχνίδι ενώ η Update() καλείται σε κάθε frame.
- Είναι προφανές πως καθώς η Update() εκτελείται σε κάθε frame είναι καλό να αποφεύγονται χρονοβόρες και κοστοβόρες διαδικασίες εντός της συνάρτησης.

Awake(), Start() & Update()

- Η συνάρτηση Awake() επίσης προσφέρεται από το API και λειτουργεί παρόμοια με την Start() αλλά με την εξής διαφορά: Αν το component είναι disabled, η Awake() θα κληθεί, ενώ η Start() θα κληθεί τη πρώτη φορά που το component γίνει enabled.
- Σημειώνεται πως η Awake() καλείται πριν την Start().

Awake(), Start() & Update()

- Η Update() έχει επίσης δύο παραλλαγές: τη FixedUpdate() και τη LateUpdate(). Ο τρόπος λειτουργίας τους είναι παρόμοιος (καλούνται σε τακτά χρονικά διαστήματα) αλλά έχουν κάποιες διαφοροποιήσεις που τις καθιστούν χρήσιμες για πιο συγκεκριμένες διαδικασίες.
- Η FixedUpdate() καλείται σε σταθερά χρονικά διαστήματα ανάμεσα σε frames και χρησιμοποιείται κυρίως για διαδικασίες που αφορούν physics.
- Η LateUpdate() καλείται σε κάθε frame αλλά αφού κληθεί η Update.

Awake(), Start() & Update()

- Video tutorial για τις συναρτήσεις Awake() & Start(): Awake() & Start()
- Video tutorial για τις συναρτήσεις Update() & FixedUpdate(): Update() & FixedUpdate()

Vectors

- Πολλά στοιχεία στο API της Unity χρησιμοποιούν κάποια μορφή διανυσμάτων για την αναπαράστασή τους.
- Αν και τα διανύσματα εξ ορισμού διαθέτουν κατεύθυνση και μέτρο, στο περιβάλλον της Unity χρησιμοποιούνται και για να αναπαραστήσουν σημεία, ή απλά ως μια δομή που περιέχει τρεις πραγματικούς αριθμούς.
- Οι κλάσεις που χρησιμοποιεί η Unity για αναπαράσταση των διανυσμάτων είναι οι Vector2, Vector3, Vector4 που αντιστοιχούν σε διανύσματα 2,3 και 4 διαστάσεων.

Vectors

- Για παράδειγμα, σε ένα Transform, η θέση του (position) είναι ένα αντικείμενο τύπου Vector3 που δηλώνει την απόστασή του από το (0,0,0) σε κάθε άξονα. Ωστόσο, και η περιστροφή του δηλώνεται σαν Vector3 (περίπου) που αναπαριστά τις γωνίες περιστροφής του σε κάθε άξονα, αλλά και η κλίμακά του αντιστοιχεί σε ένα Vector3 ως το μέγεθός του σε κάθε άξονα.
- Τόσο τη περιστροφή όσο και τη κλίμακα δεν έχει νόημα να τις ταυτίσουμε με διανύσματα με φορά και μέγεθος αλλά απλά σαν μια συλλογή τριών αριθμών.

Vectors

- Βασικά πεδία των κλάσεων που θα μας απασχολήσουν είναι αυτά που αφορούν την κατεύθυνση και αποτελούν static πεδία της κάθε κλάσης.
- Τέτοια πεδία είναι τα Vector3.up/down, Vector3.right/left, Vector3.forward/back
- Σημειώνεται πως η κλάση Vector3 είναι μια πιο ειδική μορφή της κλάσης Vector2 και μάλιστα κληρονομεί από αυτή.

Vectors

- Κανονικοποιημένο διάνυσμα: διάνυσμα το οποίο έχει μέτρο ίσο με 1. Κάθε αντικείμενο τύπου Vector2/Vector3 έχει ένα πεδίο “normalized” το οποίο επιστρέφει διάνυσμα ίδιου τύπου και κατεύθυνσης, αλλά με μοναδιαίο μέτρο.
- Το μέτρο (magnitude) υπολογίζεται μέσω του πυθαγορείου θεωρήματος, οπότε είναι ίσο με τη ρίζα του αθροίσματος των τετραγώνων κάθε συντεταγμένης. Μέσα από την κλάση μπορεί να προσπελαστεί από το πεδίο “magnitude”. Ωστόσο, επειδή η πράξη της τετραγωνικής ρίζας είναι πολύ κοστοβόρα, για συγκρίσεις χρησιμοποιείται το πεδίο “sqrMagnitude” που αντιστοιχεί στο τετραγωνισμένο μέτρο.

Vectors

- Στην κλάση επίσης εμπεριέχονται πολλές συναρτήσεις που αφορούν στα διανύσματα όπως:
  - Γωνία μεταξύ διανυσμάτων: Vector3/2.Angle(…)
  - Εσωτερικό γινόμενο: Vector3/2.Dot(…)
  - Εξωτερικό γινόμενο: Vector3/2.Cross(…)
  - Απόσταση:Vector3/2.Distance(…)

Vectors

- Video tutorials για τον τρόπο λειτουργίας των vectors:
  - Game Math Theory - VECTORS από Brackeys
  - VectorMaths από Unity

Quaternions

- Αν και εντός του editor η περιστροφή ενός αντικειμένου εκφράζεται ως το σύνολο των γωνιών περιστροφής του ως προς κάθε άξονα, εντός του API η περιστροφή ενός αντικειμένου (rotation) εκφράζεται μέσω της κλάσης Quaternion (τετραδόνιο).
- Το τετραδόνιο αποτελεί γενικευμένη μορφή των μιγαδικών αριθμών και γενικότερα είναι μια πολύπλοκη μαθηματική έννοια με την οποία δεν χρειάζεται να ασχοληθούμε.

Quaternions

- Αντί για quaternions, δίνεται η δυνατότητα να χρησιμοποιηθούν οι γωνίες ως προς κάθε άξονα, όπως είναι και εντός του editor.
- Αν και το πεδίο rotation ενός Transform είναι Quaternion, υπάρχει ένα πεδίο EulerAngles το οποίο είναι Vector3 και αντιστοιχεί ακριβώς σε αυτό που αναπαριστούν τα πεδία του editor.
- Ένα Quaternion μπορεί να επιστρέψει το αντίστοιχο Vector3 διάνυσμα μέσω του πεδίου “Euler”.

Instantiate() & Destroy()

- Η συνάρτηστη Instantiate() του API χρησιμοποιείται για να εμφανίσει ένα GameObject στη σκηνή σε δεδομένη τοποθεσία και με δεδομένη κλίση.
- Αντίστοιχα, η συνάρτηση Destroy() χρησιμοποιείται για να καταστρέψει ένα αντικείμενο από τη σκηνή, είτε άμεσα είτε αφού περάσει κάποιο χρονικό διάστημα.
- Οι συναρτήσεις αυτές, αν και χρήσιμες, καλό θα ήταν να χρησιμοποιούνται με φειδώ. Ο λόγος είναι πως το να δημιουργούνται ή να καταστρέφονται άμεσα, στη διάρκεια ενός frame, αντικείμενα που είναι αρκετά περίπλοκα μπορεί να έχει αρνητικό αντίκτυπο στην απόδοση.

Instantiate() & Destroy()

- Video tutorial για τη συνάρτηση Instantiate(): Instantiate() από Unity
- Video tutorial για τη συνάρτηση Destroy(): Destroy() από Unity

GameObject

- Κάθε αντικείμενο στη σκηνή είναι ένα GameObject, και, συνεπώς έχει το αντίστοιχο component/κλάση με το ίδιο όνομα.
- Η συγκεκριμένη κλάση έχει πεδία που αφορούν το κάθε αντικείμενο ως οντότητα, οπότε περιλαμβάνει το όνομα του αντικειμένου, αν είναι ενεργό ή όχι μέσα στη σκηνή και λοιπές πληροφορίες τέτοιας φύσεως.
- Σε κάθε script που δημιουργείται, υπάρχει το πεδίο “gameObject” το οποίο είναι ένα αντικείμενο τύπου GameObject και αναφέρεται στο αντικείμενο το οποίο έχει το παρόν script.

GameObject

- Video tutorial πάνω στην ενεργοποίηση των game objects: Activating Game Objects από Unity

Transform

- Η συγκεκριμένη κλάση αποτελεί μία από τις πιο κοινές κλάσεις, καθώς κάθε σχεδόν αντικείμενο περιέχει το component Transform.
- Η κλάση αυτή καθορίζει τις ιδιότητες ενός αντικειμένου ως ένα αντικείμενο στον χώρο. Δηλαδή, προσδιορίζει στοιχεία όπως η θέση, η περιστροφή και η κλίμακα του αντικειμένου, σε τοπικές και παγκόσμιες συντεταγμένες, ενώ περιέχει και λειτουργίες που αφορούν στη μετακίνηση ή περιστροφή του.

Transform

- Όπως και με τη κλάση gameObject, σε κάθε script που δημιουργείται υπάρχει ένα πεδίο “transform” το οποίο είναι ένα αντικείμενο της κλάσης Transform το οποίο αφορά στο αντικείμενο που περιέχει το script.
- Τα βασικά πεδία ενός Transform ως προς τις ιδιότητές του στο χώρο είναι τα εξής:

	Παγκόσμιες συντεταγμένες	Τοπικές συντεταγμένες
Θέση	transform.position	transform.localPosition
Περιστροφή	transform.rotation transform.eulerAngles	transform.localRotation transform.localEulerAngles
Κλίμακα	transform.lossyScale	transform.localScale

Transform

- Όμοια με τα διανύσματα, υπάρχουν πεδία της κλάσης transform τα οποία δηλώνουν κατεύθυνση ως προς το ίδιο το αντικείμενο.
- Τέτοια πεδία είναι ως εξής: transform.forward, transform.right, transform.up
- Τα back, left, down δεν δίνονται, ωστόσο μπορούν να υπολογιστούν πολλαπλασιάζοντας τα παραπάνω διανύσματα με -1.

Transform

- Χρήσιμες συναρτήσεις της κλάσης Transform είναι οι εξής:
  - transform.Translate(Vector3 translation): μετακινεί (πρακτικά τηλεμεταφέρει) το αντικείμενο στη κατεύθυνση του διανύσματος translation και τόσες μονάδες όσες το μέτρο του διανύσματος translation.
  - transform.Rotate(Vector3 eulerAngles): περιστρέφει το αντικείμενο ως προς τον άξονα που ορίζει το διάνυσμα eulerAngles κατά τόσες μοίρες όσες το μέτρο του διανύσματος.
- Σημειώνεται πως αυτές οι συναρτήσεις έχουν και ορισμένα overloads, οπότε δεν είναι απαραίτητο πως θα καλούνται όπως αναφέρθηκε.

Transform

- Video tutorial για τις συναρτήσεις Translate και Rotate της κλάσης Transform: Translate and Rotate από Unity

Rigidbody

- Το Rigidbody αποτελεί βασικό component καθώς είναι αυτό που προσδίδει φυσικές ιδιότητες σε ένα αντικείμενο, όπως τη δυνατότητα να δέχεται και να ασκεί δυνάμεις.
- Η αντίστοιχη κλάση περιλαμβάνει πεδία που αφορούν αυτές τις φυσικές ιδιότητες, όπως την επιτάχυνση του αντικειμένου, καθώς και συναρτήσεις για άσκηση δυνάμεων στο αντικείμενο.

Rigidbody

- Επειδή δεν είναι απαραίτητο πως κάθε αντικείμενο θα έχει ένα rigidbody component, δεν υπάρχει πια κάποιο έτοιμο πεδίο που να απευθύνεται σε αυτό το component για κάθε αντικείμενο, όπως τα transform/gameObject.
- Το πεδίο που χρησιμοποιείται πιο συχνά είναι το rb.velocity το οποίο επιστρέφει την ταχύτητα του αντικειμένου σε μορφή Vector3.
- Η συνάρτηση της κλάσης που αφορά στην άσκηση δύναμης στο αντικείμενο είναι η εξής:
  - rb.AddForce(Vector3/2 force)

η οποία ασκεί δύναμη στο αντικείμενο στην κατεύθυνση του διανύσματος force και με μέγεθος αντίστοιχο του μέτρου του.

Rigidbody

- Σημειώνεται πως επειδή η προσομοίωση φυσικών ιδιοτήτων διαφέρει από τον 3D στον 2D χώρο, για 2D στοιχεία χρησιμοποιείται το component Rigidbody2D με αντίστοιχα πεδία και συναρτήσεις.

Inputs

- Ίσως το πιο σημαντικό κομμάτι της διάδρασης με το παιχνίδι είναι η δυνατότητα εκχώρησης εισόδου από τον παίκτη, μέσω του πληκτρολογίου, ποντικιού ή άλλου μέσου.
- Δίνονται πολλοί διαφορετικοί τρόποι ανίχνευσης κάποιας εισόδου τόσο σε μορφή κλήσεων συναρτήσεων (static συναρτήσεις της κλάσης Input) όσο και σε μορφή ορισμού συναρτήσεων με συγκεκριμένο όνομα.
- Οι έλεγχοι για είσοδο μέσω των συναρτήσεων γίνονται συνήθως στην Update καθώς θέλουμε ο έλεγχος για είσοδο να γίνεται σε κάθε frame.

Inputs

- Οι πιο συχνοί τρόποι ανίχνευσης εισόδου είναι με την κλήση των εξής συναρτήσεων:
  - Input.GetButton(string buttonName)
    - Input.GetButtonDown(string buttonName)
    - Input.GetButtonUp(string buttonName)
  - Input.GetAxis(string axisName)
    - Input.GetAxisRaw(string axisName)
  - Input.GetKey(KeyCode key)
    - Input.GetKeyDown(KeyCode key)
    - Input.GetKeyUp(KeyCode key)
  - Input.GetMouseButton(int button)
    - Input.GetMouseButtonDown(int button)
    - Input.GetMouseButtonUp(int button)

Inputs – Input.GetButton

- Η συγκεκριμένη συνάρτηση επιστρέφει bool του οποίου η τιμή εξαρτάται από τη παραλλαγή της συνάρτησης, και παίρνει σαν όρισμα το όνομα ενός κουμπιού, όπως έχει οριστεί από τον input manager του editor.
- Οι παραλλαγές είναι οι εξής:
  - Input.GetButton(string buttonName): Καλείται όσο το κουμπί που αντιστοιχεί στο buttonName μένει πατημένο.
  - Input.GetButtonDown(string buttonName): Καλείται όταν πατηθεί το κουμπί που αντιστοιχεί στο buttonName.
  - Input.GetButtonUp(string buttonName): Καλείται όταν ο παίκτης σταματήσει να πατάει το κουμπί που αντιστοιχεί στο buttonName.

Inputs – Input.GetButton

- Βασικά ονόματα κουμπιών είναι τα εξής:
  - “Jump” : Αντιστοιχεί στο space
  - “Fire1” : Αντιστοιχεί στο αριστερό κλικ του ποντικιού
  - “Fire2” : Αντιστοιχεί στο δεξί κλικ του ποντικιού
  - “Fire3” : Αντιστοιχεί στο μεσαίο κλικ του ποντικιού
- Η χρήση των συναρτήσεων GetButton είναι γενικά καλύτερη γιατί δίνει τη δυνατότητα στον παίκτη να αλλάξει τα bindings των πλήκτρων.

Inputs – Input.GetAxis

- Αυτή η συνάρτηση επιστρέφει πραγματικό αριθμό από το -1 μέχρι το 1 και παίρνει σαν όρισμα το όνομα ενός άξονα.
- Ο αριθμός που επιστρέφει αντιστοιχεί στην είσοδο που δόθηκε βάσει του δεδομένου άξονα. Π.χ. με τον άξονα “Horizontal” επιστρέφεται -1 αν ο παίκτης πατάει το αριστερό βελάκι ή το “A” και το 1 αν πατάει το δεξί βελάκι ή το “D”. Τις ενδιάμεσες τιμές τις παίρνει ως γρήγορη μετάβαση προς τις τιμές -1 και 1.
- Στη μορφή Input.GetAxis(string axisName) επιστρέφει τιμές σε όλο το φάσμα τιμών [-1,1].
- Με τη παραλλαγή Input.GetAxisRaw(string axisName) επιστρέφονται μόνο οι τιμές -1, 0 και 1.

Inputs – Input.GetAxis

- Συχνά χρησιμοποιούμενοι άξονες είναι:
  - “Horizontal” : “A”/⬅ για αριστερά, “D”/➡ για δεξιά. Η GetAxis επιστρέφει 1 για δεξιά και -1 για αριστερά.
  - “Vertical”: “W”/⬆ για πάνω, “S”/⬇ για κάτω. Η GetAxis επιστρέφει 1 για πάνω και -1 για κάτω.
  - “MouseX”: Άξονας που αφορά στη κίνηση του ποντικιού στον άξονα Χ. Η GetAxis επιστρέφει 1 αν το ποντίκι κινείται προς τα δεξιά και -1 αν κινείται προς τα αριστερά.
  - “MouseY”: Άξονας που αφορά στη κίνηση του ποντικιού στον άξονα Υ. Η GetAxis επιστρέφει 1 αν το ποντίκι κινείται προς τα πάνω και -1 αν κινείται προς τα κάτω.

Inputs – Input.GetKey

- Αυτή η οικογένεια συναρτήσεων λειτουργούν με παρόμοιο τρόπο με την οικογένεια Input.GetButton, με τη διαφορά ότι αφορούν μόνο είσοδο από πληκτρολόγιο και μάλιστα με hard-coded τρόπο, αφού παίρνουν σαν όρισμα συγκεκριμένο πλήκτρο από το KeyCode enumeration. Επιστρέφουν bool που εξαρτάται από τη παραλλαγή.
- Δεν προτιμάται επειδή ακριβώς δεν έχει την ευελιξία της GetButton.
- Έχει ως παραλλαγές τις εξής:
  - Input.GetKey(KeyCode key)
  - Input.GetKeyDown(KeyCode key)
  - Input.GetKeyUp(KeyCode key)

  οι οποίες δουλεύουν με αντίστοιχο τρόπο όπως αυτές της GetButton.

Inputs – Input.GetMouseButton

- Αυτή η οικογένεια συναρτήσεων είναι κατά αντιστοιχία η hard-coded πλευρά της εισόδου από το ποντίκι. Επιστρέφει bool που εξαρτάται από τη κάθε παραλλαγή, ενώ σαν όρισμα παίρνει έναν ακέραιο (0, 1 ή 2) που αντιστοιχούν στο αριστερό, δεξί και μεσαίο κουμπί του ποντικιού αντίστοιχα.
- Και εδώ υπάρχουν οι αντίστοιχες παραλλαγές με τη λειτουργικότητα που αναφέρθηκε:
  - Input.GetMouseButton(int button)
  - Input.GetMouseButtonDown(int button)
  - Input.GetMouseButtonUp(int button)

Inputs

- Παράδειγμα χρήσης των παραπάνω:

```
//…
void Update() {
        float move = Input.GetAxis(“Horizontal”);
        if (move > 0) {//Move right}
        else if (move < 0) {//Move left}
        if (Input.GetButtonDown(“Jump”)) {//Jump}
}
```

Inputs

- Video tutorial πάνω στις συναρτήσεις GetButton και GetKey: GetButton & GetKey από Unity
- Video tutorial πάνω στη συνάρτηση GetAxis: GetAxis από Unity

Inputs

- Επίσης πολύ χρήσιμοι τρόποι αναγνώρισης εισόδου είναι με τις εξής συναρτήσεις:
  - void OnMouseDown()
  - void OnMouseUp()
  - void OnMouseDrag()
  - void OnMouseOver()
  - void OnMouseEnter()
  - void OnMouseExit()
- Οι συναρτήσεις αυτές δεν καλούνται μέσα από το script. Ο προγραμματιστής ορίζει το σώμα μέσα στο script και η Unity τις καλεί ανάλογα με τη κάθε περίπτωση.

Inputs

- Οι συναρτήσεις αυτές αφορούν συγκεκριμένα το αντικείμενο στο οποίο βρίσκεται το script, όχι γενικότερα την είσοδο από τον χρήστη.
- Συγκεκριμένα, οι συναρτήσεις OnMouseEnter/Over/Exit καλούνται όταν το ποντίκι μπαίνει, βρίσκεται και βγαίνει αντίστοιχα από τη περιοχή που ορίζει ο collider του αντικειμένου που έχει το script.
- Κατά αντιστοιχία, οι συναρτήσεις OnMouseDown/Drag/Up καλούνται όταν ο παίκτης κάνει κλικ, κρατάει πατημένο το κλικ ή σταματάει να κάνει κλικ αντίστοιχα πάνω στο αντικείμενο.

Inputs

- Video tutorial πάνω στη χρήση της OnMouseDown: OnMouseDownαπόUnity
- Λόγω παλιότερης έκδοσης το συγκεκριμένο script που εμφανίζεται στο video δεν θα δουλέψει ακριβώς όπως είναι, ωστόσο στο κάτω μέρος της σελίδας έχει documentation για κάθε συνάρτηση που αναφέρθηκε, μαζί με παραδείγματα.
- Το ίδιο ισχύει και για άλλα video tutorials από τη Unity.

Find & GetComponent

- Καθώς δεν θα είναι πάντα εφικτό ή πρακτικό να περνιούνται αντικείμενα και components μέσω public πεδίων, δίνεται η δυνατότητα εύρεσής τους με τη χρήση των συναρτήσεων GameObject.Find() και της συνάρτησης GetComponent<T>().
- Η συνάρτηση Find είναι static συνάρτηση της κλάσης GameObject και επιστρέφει ένα game object με βάση το όνομα που δόθηκε σαν όρισμα.
- Η συνάρτηση GetComponent<T> βρίσκεται τόσο σε game objects όσο και σε transforms και αποτελεί μια generic συνάρτηση που επιστρέφει το πρώτο component τύπου T που βρίσκει στο ζητούμενο αντικείμενο.

Find & GetComponent

- Video tutorial πάνω στη χρήση της συνάρτησης GetComponent: GetComponent από Unity

Colliders & Triggers

- Ίσως το δεύτερο πιο σημαντικό στοιχείο της διαδραστικότητας στα παιχνίδια μετά τα inputs είναι η δυνατότητα να εκτελούνται γεγονότα με βάση το αν ο παίκτης βρίσκεται σε κάποια συγκεκριμένη περιοχή ή αν ήρθε σε σύγκρουση με κάποιο αντικείμενο.
- Για το σκοπό αυτό χρησιμοποιείται το component του collider, το οποίο χρησιμοποιείται για να ορίσει μια περιοχή γύρω από ένα αντικείμενο η οποία συμπεριφέρεται σαν εμπόδιο για τον παίκτη.
- Εναλλακτικά, ορίζοντας μέσα από τον editor τον collider ως trigger, η περιοχή δεν εμποδίζει με κάποιο τρόπο τον παίκτη.

Colliders & Triggers

- Είτε το component λειτουργεί σαν collider ή σαν trigger, υπάρχει η δυνατότητα μέσα από τον κώδικα να αναγνωρίζεται πότε κάποιο αντικείμενο εισέρχεται σε ένα trigger ή συγκρούεται με ένα collider και να διατηρούνται κάποιες πληροφορίες σχετικά με το γεγονός αυτό.
- Για αυτή τη λειτουργία χρησιμοποιούνται οι εξής συναρτήσεις:
  - void OnCollisionEnter(Collision collision)
    - void OnCollisionStay(Collision collision)
    - void OnCollisionExit(Collision collision)
  - void OnTriggerEnter(Collider collider)
    - void OnTriggerStay(Collider collider)
    - Void OnTriggerExit(Collider collider)

Colliders & Triggers

- Ο τρόπος λειτουργίας αυτών των συναρτήσεων είναι αντίστοιχος με αυτόν που αναφέρθηκε για τις συναρτήσεις OnMouseButton κλπ. Δηλαδή δεν καλούνται κάπου, απλά ορίζονται από τον προγραμματιστή μέσα στο script και η Unity τις καλεί ανάλογα με την περίσταση.
- Σημειώνεται πως ενώ οι συναρτήσεις OnCollision παίρνουν ως όρισμα ένα Collision (κλάση που περιέχει πληροφορίες ως προς τη σύγκρουση αντικειμένων), οι συναρτήσεις OnTrigger παίρνουν ως όρισμα ένα Collider, συγκεκριμένα τον collider που ανήκει στο αντικείμενο που εισήλθε/εξήλθε στη περιοχή που ορίζει το trigger.

Colliders & Triggers - OnCollision

- Η οικογένεια συναρτήσεων OnCollision περιέχει τις εξής συναρτήσεις:
  - void OnCollisionEnter(Collision collision): Το σώμα της εκτελείται όταν συγκρουστεί ένα αντικείμενο με το αντικείμενο στο οποίο βρίσκεται το script με τη συνάρτηση.
  - void OnCollisionStay(Collision collision): Καλείται όσο το τρέχον αντικείμενο είσαι σε επαφή με άλλο αντικείμενο.
  - void OnCollisionExit(Collision collision): Καλείται τη στιγμή που το τρέχον αντικείμενο σταματήσει να είναι πλέον σε επαφή με κάποιο άλλο αντικείμενο.
- Σημειώνεται πως για να κληθεί επιτυχώς κάποια από αυτές τις συναρτήσεις θα πρέπει τουλάχιστον ένα από τα αντικείμενα που συμμετέχουν να έχει non-kinematic RigidBody component.

Colliders & Triggers - OnTrigger

- Κατά αντιστοιχία, οι συναρτήσεις αυτής της οικογένειας είναι:
  - void OnTriggerEnter(Collider collider): Καλείται τη στιγμή που ένα αντικείμενο εισέρχεται στο trigger του αντικειμένου που έχει το script με τη συνάρτηση αυτή.
  - void OnTriggerStay(Collider collider): Καλείται όσο ένα άλλο αντικείμενο βρίσκεται εντός του trigger του τρέχοντος αντικειμένου.
  - void OnTriggerExit(Collider collider): Καλείται τη στιγμή που ένα άλλο αντικείμενο βγει από τη περιοχή του trigger.
- Σημειώνεται πως για να κληθεί επιτυχώς κάποια από αυτές τις συναρτήσεις θα πρέπει τουλάχιστον ένα από τα συμμετέχοντα αντικείμενα να έχει ένα RigidBody component.

Colliders & Triggers

- Επειδή η ανίχνευση για collisions και για είσοδο σε trigger σε 2D περιβάλλον γίνεται με διαφορετικό τρόπο, υπάρχουν ξεχωριστές συναρτήσεις για 2D colliders με την εξής μορφή:
  - void OnCollisionEnter2D(Collision2D collision)
  - void OnCollisionStay2D(Collision2D collision)
  - void OnCollisionExit2D(Collision2D collision)
  - void OnTriggerEnter2D(Collider2D collider)
  - void OnTriggerStay2D(Collider2D collider)
  - void OnTriggerExit2D(Collider2D collider)

διατηρώντας φυσικά την ίδια λειτουργικότητα.

Colliders & Triggers

- Επειδή συχνά θέλουμε να ενεργοποιούνται οι συναρτήσεις μόνο από συγκεκριμένα αντικείμενα χρησιμοποιούνται τα tags για έλεγχο. Τα tags σε κάθε αντικείμενο ορίζονται μέσα από τον editor.
- Για παράδειγμά:

```
void OnTriggerEnter(Collider other) {
        if (other.tag == “Player”) {
                //Do stuff
        }
}
```

```
void OnCollisionExit2D(Collision2D col) {
        if (col.collider.tag == “Player”) {
                //Do stuff
        }
}
```

Colliders & Triggers

- Video tutorial σχετικά με τη χρήση της συνάρτησης OnCollisionEnter: Detecting collisions with OnCollisionEnter από Unity
- Video tutorial σχετικά με τη χρήση της συνάρτησης OnTriggerEnter: Colliders as triggers από Unity