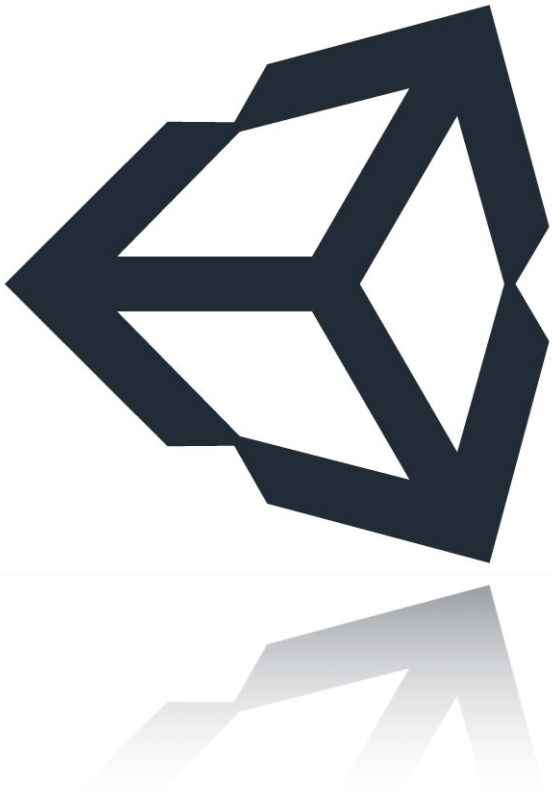
Computer Graphics

Unity3D Project Report

Αντωνίου Σεραφείμ - Ηλίας (AM: 2640)



Εισαγωγή

Στην δεύτερη προγραμματιστική άσκηση κληθήκαμε να υλοποιήσουμε ένα παιχνίδι και να εξοικειωθούμε με την πλατφόρμα γραφικών Unity3D*[1], η οποία παρέχει μία μεγάλη ποικιλία από εργαλεία για ανάπτυξη παιχνιδιών και άλλων αλληλεπιδραστικών εφαρμογών.

Ανάλυση

Αρχικά, όλα τα περιεχόμενα των φακέλων (folder contents) βρίσκονται μέσα στα assets. ΔΕΝ πειράχθηκε κανένα από τα έτοιμα πακέτα (packages) της Unity.

Περιεχόμενα Φακέλων - Folder Contents:

Εκτός των γενικών components που χρησιμοποιεί η Unity (transform, mesh renderer) οι λειτουργίες του παιχνιδιού υλοποιούνται με τα παρακάτω components:

- Animation: Περιέχει 2 animations και 1 animation controller ο οποίος είναι υπεύθυνος για την μετάβαση.
- Audio: Περιέχει τα sound clips για την καταστροφή των κουτιών/κύβων και το ηχητικό εφέ της τηλεμεταφοράς (teleport). Τα εφέ αυτά κατέβηκαν απευθείας από το web.
- Character: Περιέχει το HammerRespawn script και έναν box collider τύπου isTrigger και το HammerPickup script. (αναλυτική επεξήγηση των scripts παρακάτω)
  - HammerRespawn: Περιέχει το σφυρί (MysticHammer) - κατεβασμένο από το assets store.
    - MysticHammer: Περιέχει το HammerRotate script προκειμένου το σφυρί να εκτελεί περιστροφική κίνηση γύρω από τον εαυτό του.
      - HammerRotate (script): Το script για την εκτέλεση περιστροφικής κίνησης γύρω από τον εαυτό του.
  - Player: Τον παίρνω έτοιμο απευθείας από το assets store (standard assets της Unity). Και περιλαμβάνει:
    - FirstPersonCharacter: Είναι η κάμερα μας.
      - PlayerLight: Το φως που ακολουθεί τον παίκτη. Και είναι τύπου “point light” (φαίνεται το reflection όταν στέκεστε κοντά σε ένα κύβο).
      - MysticHammer:

[1] Τα σημεία με αστερίσκο (*) είναι hyperlinks προς το αντίστοιχο αντικείμενο.

- AnimatorController: Υλοποιεί το hit και το idle.
- Hammer (script): (ανάλυση παρακάτω)
- TeleportLight: Το φως (effect) της τηλεμεταφοράς.

- Cubes: Περιέχει το Boundaries δηλαδή την οριοθέτηση του maze.
  - Boundaries:
    - FloorBoundary: Το πάτωμα.
    - MazeBoundary: Τα τοιχώματα.
    - WinBoundary: Το πάνω-πάνω επίπεδο στο οποίο κερδίζει ο παίκτης.
      - GameWinner (script):
      - TriggerBoxCollider: (ανάλυση παρακάτω)
  - Materials: Όλα τα materials που χρησιμοποιήθηκαν για την υλοποίηση/ δημιουργία των κύβων, του Skybox και του FloorBoundary.
  - Prefabs: Εμπεριέχει όλους τους κύβους του παιχνιδού συμπεριλαμβανομένων και των empty κύβων που δημιουργήθηκαν για τα κενά και το teleport. Κάθε κύβος εκτός από το teleport έχει και ένα script το CubeHealth script για την "υγεία" (ζωή) του. Και το teleport έχει το teleport script.
    - CubeFragments: Τα κομμάτια που προκύπτουν μετά από 3 hits και την διάσπαση ενός κύβου.
- ImportedAssets: -Κατεβασμένα απευθείας-
  - GameAssetPacks: Μέσα στον φάκελο αυτό έχει πραγματοποιηθεί η λήψη του σφυριού και στον Mesh folder βρίσκεται το σφυρί.
    - Mesh: Η τοποθεσία του σφυριού.
  - ImagesForMaterial: Περιέχει τα textures που χρησιμοποιήθηκαν για να φτιαχτούν τα materials που χρειάστηκαν.
  - StandardAssets>Characters>FirstPersonCharacter>Prefabs>FPSController: Το πήρα έκανα player στον δικό μου φάκελο.
- Lights: Τα διάφορα φώτα του παιχνιδιού**.
  - MazeLight: Το φως που πέφτει πάνω στην πίστα (στο πάνω επίπεδο - είναι ορατό κατά την εκκ'νηση του παιχνιδιού).
  - PlayerLight: Το φως που ακολουθεί τον παίκτη.
  - TeleportLight: Το φως/εφέ κατά την τηλεμεταφορά.

**Φως: Για να αποφευχθεί το συνεχές bake: Window>Rendering>Lighting settings>Disable Generate και διαλέγω το Generate Lighting. Αποθηκεύει μόνο 1

φορά το φως στη σκηνή για να μην κάνει συνέχεια rendering ο υπολογιστής (path: Assets>Scenes>GameScene).

- MazeRead: Το δοθέν file που δίνεται από την εκφώνηση της άσκησης για τον λαβύρινθο και το initialization που αρχικοποιεί το παιχνίδι, το οποίο επιτυγχάνεται μέσω του script StartWorld.
- Resources: Περιέχει τα 3 materials από τα οποία αποτελείται το σφυρί. Έγινε χρήση του φακέλου αυτού διότι, η Unity MONO μέσω αυτού του φακέλου μου επιτρέπει να φορτώσω τα materials του σφυριού.
- Scenes:
  - GameScene: Τύπου Unity και είναι το level/πίστα του παιχνιδιού μας.
  - GameScene:
    - Skybox: ReflectionProbe (δημιουργείται αυτόματα).
    - LightingData: Κρατάει τα φώτα όπως είναι.
- UI: Το interface του παιχνιδιού.
  - ScoreAndAimCanvas:
    - Aim: Στόχος για hit.
    - ScoreTxt:
      - Score (script)

Ανάλυση των scripts - Script Analysis:

- StartWorld.cs: Αρχικοποιεί το παιχνίδι διαβάζοντας το file που δίνεται μαζί ε την άσκηση.
  - Awake(): Έτοιμη συνάρτηση της Unity, η οποία βάσει της ιεραρχίας *[2] εκτελείται πριν από όλα.
  - InitParents(): Δημιουργεί τα parents των κύβων και τα "κάνει" παιδιά του initialization.
  - InitBoundaries(): Αρχικοποιεί όλα του παιχνιδιού. Δηλαδή, οριοθετεί την πίστα και τα "κάνει" παιδιά του Boundaries (parent).
  - InitFromFile(): Διαβάζω από το αρχείο file.maz, ελέγχω τις 3 πρώτες γραμμές και κάνω cast σε int32 τα levels, το grid (16x16) και τα σφυριά τα οποία είναι static (επειδή χρησιμοποιούνται και σε άλλα scripts). Μετά ελέγχω αν η γραμμή περιέχει L και μετά E και αν το περιέχει αυξάνω +1 το επίπεδο (level). Αλλιώς, δημιουργώ στιγμιότυπο των κύβων μου.

[2] Τα σημεία με αστερίσκο (*) είναι hyperlinks προς το αντίστοιχο αντικείμενο.

- Instantiate(prefab, Vector3(), Quaternion): Φτιάχνει στιγμιότυπο του εκάστοτε prefab στο position Vector3 (3D διάνυσμα) με γωνίες το Quaternion (identity: μοναδιαίος πίνακας ή γωνίες Euler).
- InitRandomPosition(): Θέτω ως χωρητικότητα, το πλήθος των παιδιών του αρχικού empty γονέα. Κάνω ένα iteration, στο οποίο διατρέχω όλα τα ύψη των παιδιών και όσα είναι ίσα με μηδέν, προσθέτω το position τους σε μία λίστα. Στη συνέχεια, παίρνω ένα τυχαίο position και το οριοθετώ με την RoundToInt. Τέλος, φτιάχνω στιγμιότυπο του παίκτη από την λίστα με τα διάφορα positions με το random position σε Quaternion identity.

CubeHealth.cs:

- Start(): Είναι έτοιμη συνάρτηση της Unity η οποία δημιουργεί στιγμιότυπο του HammerSpawn στο position του κάθε κύβου με Quaternion identity και τα θέτει όλα inactive. Παίρνω το component FirstPersonController από τον Player στο audio και το αρχικοποιώ.
- Update(): Είναι έτοιμη συνάρτηση της Unity, η οποία τρέχει από πίσω σε κάθε frame. Ελέγχω την υγεία του κύβου (cube health) και σε περίπτωση που είναι <=0 κάνω στιγμιότυπο των κομματιών του κατεσραμμένου κύβου και παίζω τον ήχο την καταστροφής (εφέ).
- DropChance(): Υπολογίζω την πιθανότητα του να κάνω active το σφυρί που περιέχει κάθε κύβος.
- OnDisable(): Έτοιμη συνάρτηση της Unity, η οποία καλείται με την καταστροφή του κύβου και καταστρέφει τα κομμάτια που προέκυψαν (fragments) μετά το πέρας 1.5 sec.

GameWinner.cs:

- OnTriggerStay(): Είναι έτοιμη συνάρτηση της Unity, η οποία παίρνει σαν όρισμα έναν Collider. Ο collider είναι ένα component το οποίο χρειάζεται ένα game object, προκειμένου να αντιληφθεί πότε υπάρχει σύγκρουση. Η σύγκρουση είναι 2 ειδών: trigger & collision. Απαραίτητη προϋπόθεση για να λειτουργήσει η συνάρτηση είναι ένα από τα 2 σώματα να έχει rigid body. Στη συνέχεια, ελέγχει αν ο collider είναι του παίκτη (tag = 'Player') και αν πατηθεί το πλήκτρο E ενώ βρίσκομαι στον collider τότε καλείται η GameEnd().
- GameEnd(): Μηδενίζει το timescale δηλαδή παγώνουν όλα στο παιχνίδι εκτός από το mouse.

Hammer.cs:

- Start(): ... Αρχικοποίηση του παιχνιδιού (όπως πριν).

- Update(): Ελέγχει αν πατηθεί το πλήκτρο H, τότε παίζει το animation του hit και καλεί την συνάρτηση BoxDetect(). Αν πατηθεί το πλήκτρο X, μηδενίζεται το score και σταματάει το παιχνίδι.
- BoxDetect(): Ελέγχει αν η ακτίνα που κάνει cast η κάμερα στα 0.9 (no measurement unit) χτυπήσει πάνω σε κάποιο object με tag = 'Cube'. Αν γίνει αυτό, τότε παίρνω από το object αυτό την υγεία του (health) και την μειώνω κατά το εκάστοτε damage που προκαλείται. Στη συνέχεια, αλλάζει το χρώμα του σφυριού καλώντας την ChangeHammer() και παίζει το sound clip του hit.
- ChangeHammer(): Ελέγχει την υγεία του σφυριού και αναλόγως αλλάζει το χρώμα του [70% - 40%: RED & 30% - 0%: BLACK]. Αλλιώς, αλλάζω σφυρί, θέτω health = 100%, αφαιρώ -1 σφυρί από αυτά που έχω στην διάθεση μου και ελέγχω αν έμειναν σφυριά (αν ΔΕΝ περίσσεψαν σφυριά τότε Game Over), έπειτα αυξάνω +1 τα σφυριά που ήδη χρησιμοποίησα για το score, μειώνω το health κατά 10% και φορτώνω το αρχικό material.

HammerPickup.cs: Αν ο collider έχει tag = 'Player', τότε καταστρέφω το game object που έχει HammerPickup και προσθέτω +1 σφυρί στο inventory μου.

HammerRotate.cs: Όσο δεν κάνω pick up το σφυρί, περιστρέφεται γύρω από τον άξονα του εαυτού του (δεν δούλευε το να πέφτει στο πάτωμα με rigid body οπότε το άφησα στον αέρα).

Score.cs: Τυπώνεται το τρέχον score με βάση τις κινήσεις του παίκτη, το πέρασμα του χρόνου, το πλήθος των διαθέσιμων σφυριών και το πλήθος των ήδη χρησιμοποιημένων σφυριών.

Teleport.cs:

- Start(): Παίρνω τα απαραίτητα components.
- OnTriggerEnter(): Εξετάζω αν ο collider έχει tag = 'Player', μετά ελέγχω το ύψος του παίκτη. Αν το ύψος του παίκτη είναι μεταξύ -1 και 0 (δηλαδή, βρίσκεται στο 1ο επίπεδο), τότε κάνει το position του παίκτη = teleport position 1ου επιπέδου και μετά καλεί τον iEnumerator όπου κάνει disable τον collider του teleport στον οποίο κάνουμε την τηλεμεταφορά. Περιμένω για t = 5sec και ξαναενεργοποιώ τον collider. Ακολουθώ την ίδια διαδικασία για το φως με την συνάρτηση waitForLightSec(). Αυτή η διαδικασία γίνεται για κάθε επίπεδο

FirstPersonController.cs:

- PlayHitSound(): Παίζει το ηχητικό εφέ του χτυπήματος σε κύβο.
- PlayTeleportSound(): Παίζει το ηχητικό εφέ της τηλεμεταφοράς.

- PlayDestructionSound(): Παίζει το ηχητικό εφέ της καταστροφής κύβων.
