ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ Β

Γραφικά Υπολογιστών και Συστήματα Αλληλεπίδρασης

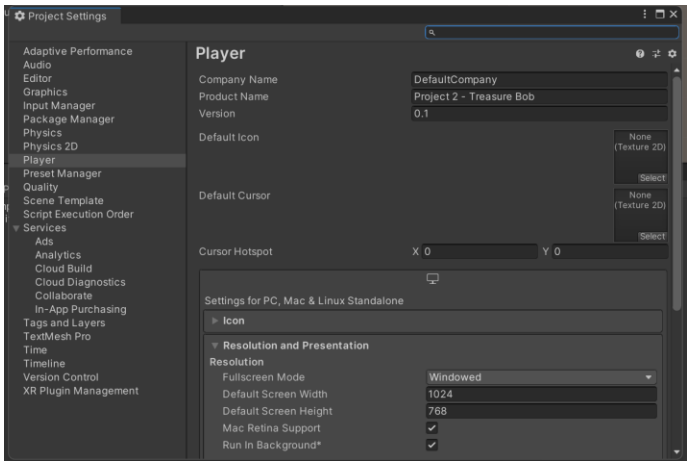
Όμηρος Χατζηιορδάνης – 5388

21/12/2024

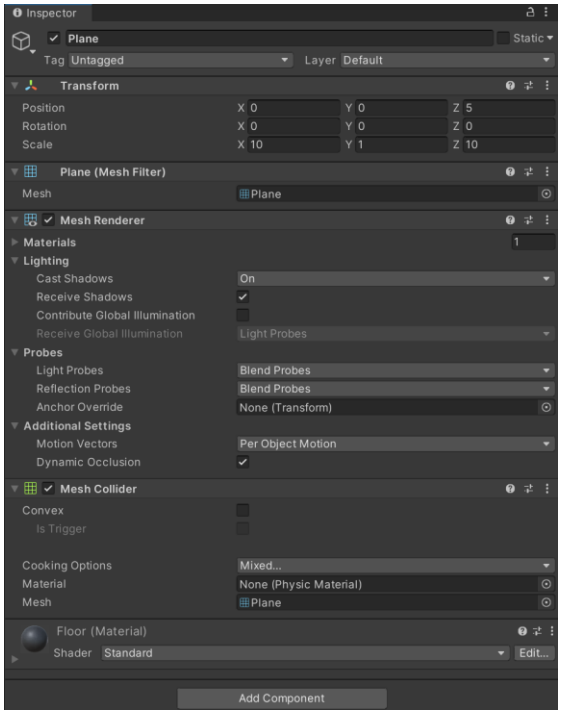
1.Περιγραφή Εργασίας:

Ερώτημα i:

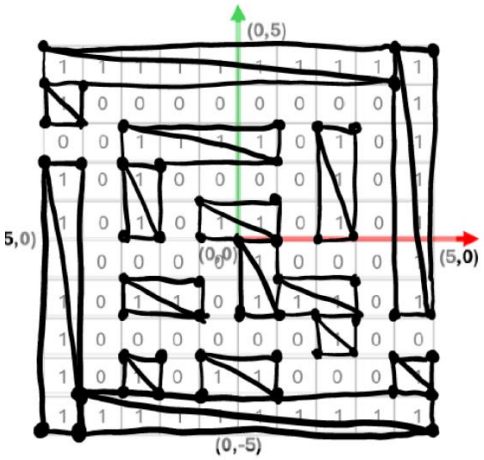
Το project ονομάστηκε με τον τίτλο που ζητείται στην εκφώνηση, η ανάλυση καταχωρήθηκε στις ρυθμίσεις του player, όπου και επιλέχθηκε η προβολή σε παράθυρο.

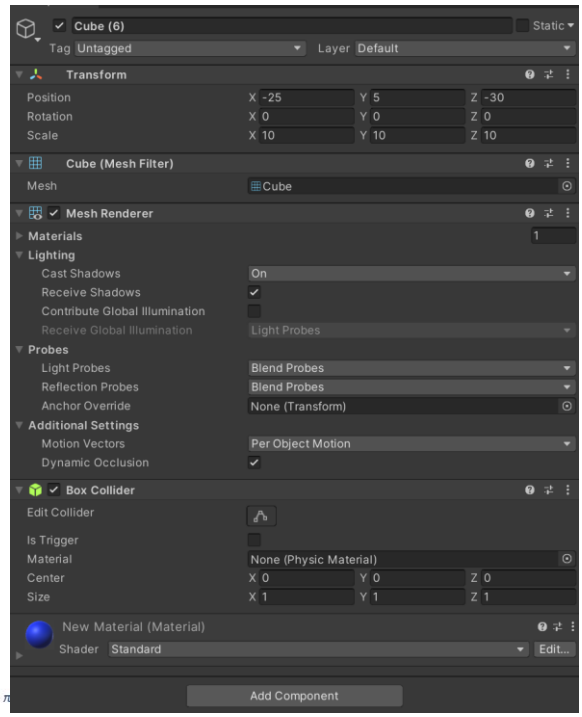


Για το έδαφος του λαβυρίνθου δημιουργήθηκε ένα αντικείμενο plane το οποίο έχει scale (10,1,10), ώστε να αποτελείται από 100x100 τετραγωνάκια με την ιδιαιτερότητα στην διερμηνεία των τιμών του scale στα αντικείμενα plane στη unity, και το κέντρο του είναι τοποθετημένο στο σημείο (0,0,5) όπως ζητείται στην εκφώνηση. Η υφή δημιουργήθηκε κάνοντας drag and drop την εικόνα floor.jpg στο plane.



Για τους τείχους του λαβυρίνθου δημιουργήθηκαν πολλαπλά αντικείμενα cube ακολουθώντας παρόμοιες τιμές με τον κύβο στην παρακάτω εικόνα. Το μέγεθος τους καθορίζεται βάζοντας 10 στις τιμές των αξόνων στο scale. Σε άλλους κύβους οι τιμές των x και z μεταβάλλονται και στο position και στο scale, ανάλογα με το ορθογώνιο που πρέπει να δημιουργηθεί με βάση τον σχεδιασμό του λαβυρίνθου από τις προηγούμενες εργασίες. Για το χρώμα δημιουργήθηκε ένα νέο material που το χρώμα του να είναι μπλε με βάση τις γνωστές τιμές rgb, το οποίο τοποθετήθηκε σε όλους τους κύβους με drag and drop.

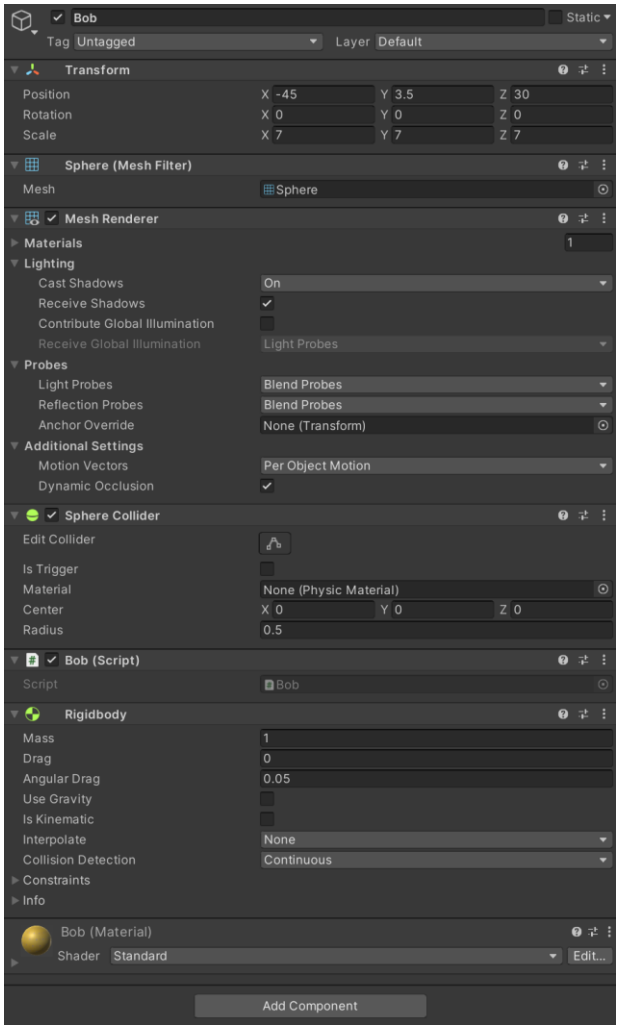




ίνα 4: Η αναπαράσταση του λαβύρινθου με τη χρήση του καρτεσιανού συστήματος και του π

Ερώτημα ii:

Για τον παίχτη, δημιουργήθηκε ένα αντικείμενο σφαίρας με όνομα bob, με το scale να έχει τιμή 7 και στους τρεις άξονες, και τοποθετημένο στην αρχή του λαβυρίνθου με βάση το κέντρο (0,0,5). Η υφή τοποθετήθηκε όπως και με το plane. Για να μην περνάει ο bob μέσα από τους κύβους του λαβυρίνθου χρησιμοποιήθηκε το RigidBody της unity το οποίο περιέχει τη δικιά του φυσική, μέσω των colliders στα υπόλοιπα αντικείμενα θα εμποδίζει την εσωτερική προσπέλαση της σφαίρας.



Για την κίνηση δημιουργήθηκε ένα script. Υπάρχει το ιδιωτικό πεδίο για το rigidbody ώστε να γίνεται διαθέσιμο για την κίνηση στο update, και μια τιμή για την ταχύτητα του bob. Στην αρχή της εφαρμογής γίνεται η ανάθεση του component rigidbody από το αντικείμενο, έλεγχος ότι έγινε σωστά η ανάθεση, και η απενεργοποίηση της επιλογής isKinematic στο rigidbody ώστε να μπορεί να λειτουργεί σίγουρα η κίνηση.

```
5   public class Bob : MonoBehaviour
6   {
7       private Rigidbody rb;
8       private float speed = 25f;
9
10      void Start()
11      {
12          rb = GetComponent<Rigidbody>();
13          if (rb == null)
14          {
15              Debug.LogError("Rigidbody is missing on this GameObject!");
16          }
17          rb.isKinematic = false;
18      }
```

Σε κάθε στιγμή, εφόσον ο παίκτης δεν έχει χάσει (περισσότερα στο ερώτημα iv), υπάρχει εντοπισμός για τη χρήση των πλήκτρων i,j,k και l, όπου ανάλογα τον συνδυασμό των πλήκτρων τροποποιείται ένα διάνυσμα, το οποίο αντιπροσωπεύει

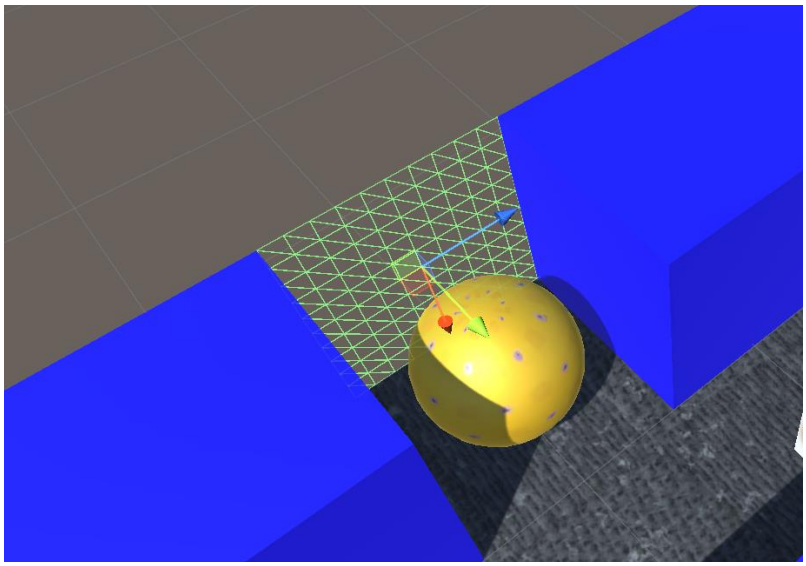
τη κατεύθυνση στους τρεις άξονες. Το διάνυσμα της κατεύθυνσης χρησιμοποιείται πολλαπλασιασμένο με τη ταχύτητα ώστε να αλλάξει το velocity του rigidbody, και η σφαίρα του bob να αποκτήσει ταχύτητα και να κινηθεί κατάλληλα.

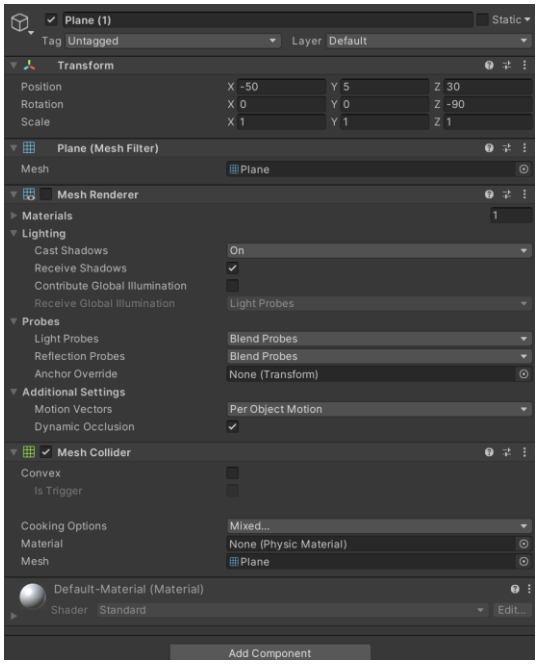
```
29      if (GlobalVariables.gameOn)
30      {
31          if (Input.GetKey("j")) moveDirection += new Vector3(-1, 0, 0);
32          if (Input.GetKey("l")) moveDirection += new Vector3(1, 0, 0);
33          if (Input.GetKey("i")) moveDirection += new Vector3(0, 0, 1);
34          if (Input.GetKey("k")) moveDirection += new Vector3(0, 0, -1);
35
36          if (Input.GetKeyDown("m")) speed += 25;
37          if (speed > 125) speed = 125;
38          if (Input.GetKeyDown("n")) speed -= 25;
39          if (speed < 25) speed = 25;
40
41          rb.velocity = moveDirection * speed;
42      }
```

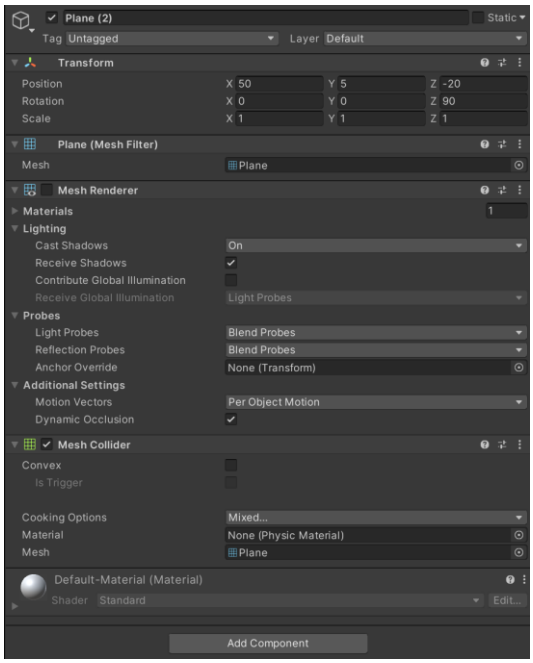
Επίσης λόγω της φυσικής του rigidbody, αν ο bob ακουμπήσει κάποιο άλλο αντικείμενο μπορεί να βρεθεί σε διαφορετικό ύψος (y), οπότε για αυτό υπάρχει ένας συνεχόμενος έλεγχος αν το y του bob έχει αλλάξει από το 3.5 (η ακτίνα της σφαίρας), όπου αν συμβεί αυτό επαναφέρει τον bob στο σωστό y.

```
24      Vector3 currentPosition = rb.position; //Rigidbody's position
25      if (currentPosition.y > 3.5f || currentPosition.y < 3.5f)
26      {
27          rb.MovePosition(new Vector3(currentPosition.x, 3.5f, currentPosition.z));
28      }
```

Τέλος για να μην βγει ο bob εκτός λαβυρίνθου, δημιουργήθηκαν δύο planes, τοποθετημένα στην αρχή και στο τέλος του λαβυρίνθου, όπου έχει απενεργοποιηθεί ο mesh renderer τους.

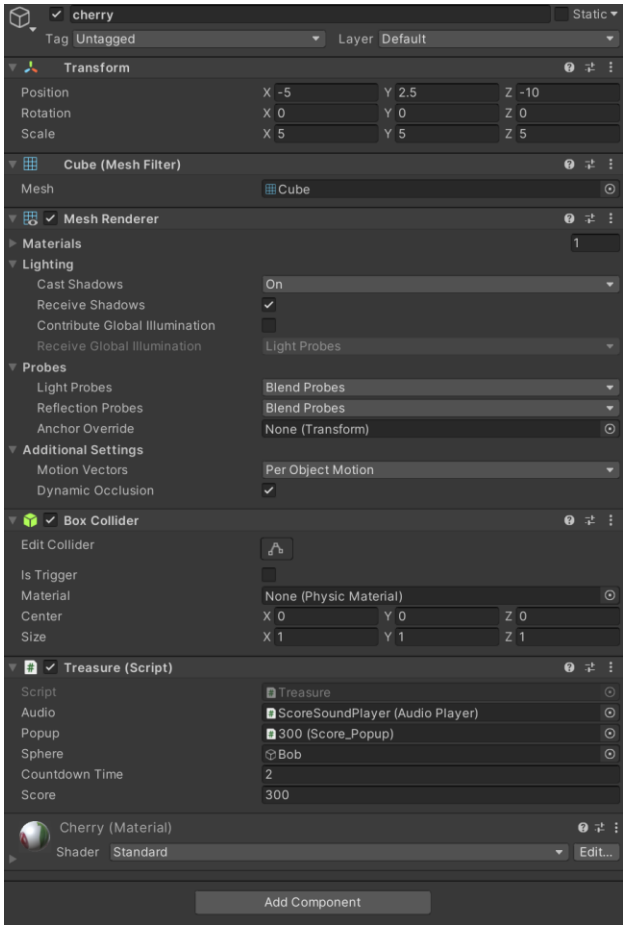






Ερώτημα iii:

Για τους θησαυρούς δημιουργήθηκαν τρεις παρόμοιοι κύβοι cherry, orange, lemon, με τα scale τους να έχουν την τιμή 5 και στους τρεις άξονες. Και για τους τρεις κύβους χρησιμοποιείται το ίδιο script με όνομα treasure.



Στο script δημιουργούνται μεταβλητές που χρησιμοποιούνται στις παρακάτω μεθόδους μαζί και με πεδία για αντικείμενα όπως ο audio player, τα αντικείμενα για τα οπτικά εφέ (περισσότερα στην εξήγηση του bonus a) και του bob στα οποία μπορούν να γίνουν ανάθεση από τον editor. Παρόμοια μπορεί να αλλάξει και η τιμή για τον χρόνο που θα εμφανίζονται οι θησαυροί, ώστε να υπάρχει διαφοροποίηση για τον καθένα.

Στην αρχή της εφαρμογής θέτονται κατάλληλα οι μεταβλητές για την εμφάνιση και τη σμίκρυνση του θησαυρού, καλείται η μέθοδος για την τοποθέτησης των θησαυρών, τοποθετείται η τιμή για τον χρόνο της εμφάνισης του θησαυρού και ξεκινάει η υπορουτίνα για την εμφάνιση και την εξαφάνιση των θησαυρών.

```
5   public class Treasure : MonoBehaviour
6   {
7       private int treasure_x;
8       private int treasure_z;
9
10      private bool treasure_to_appear;
11      private bool treasure_to_small;
12      private bool add_score = true;
13
14      public AudioPlayer audio;
15      public Score_Popup popup;
16
17      public GameObject sphere; // Reference to Bob
18      private float sphereRadius = 3.5f;
19      private float cubeHalfSize = 2.5f;
20
21      public float countdownTime;
22      private float currentTime;
23      public int score;
24
25      private float dissapearCountdownTime = 3f;
26
27      void Start()
28      {
29          treasure_to_appear = true;
30          treasure_to_small = false;
31
32          PlaceTreasure();
33          currentTime = countdownTime;
34          StartCoroutine(TreasureLogic());
35      }
```

Για την τοποθέτηση, έχει δημιουργηθεί ένα script global variables το οποίο περιέχει δημόσιες μεταβλητές που μπορούν όλα τα αντικείμενα να χρησιμοποιήσουν για τα δικά τους script. Οι θησαυροί χρησιμοποιούν τον πίνακα labyrinth, όπως και στις προηγούμενες εργασίες, όπου αντιπροσωπεύονται οι ελεύθερες θέσεις μέσα στον λαβύρινθο.

```
public class GlobalVariables : MonoBehaviour
{

    public static int score;
    public static bool gameOn = true;

    public static int[] labyrinth = new int[]
    {
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
        0, 0, 1, 1, 1, 1, 0, 1, 0, 1,
        1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
        1, 0, 1, 0, 1, 1, 0, 1, 0, 1,
        1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
        1, 0, 1, 1, 0, 1, 1, 1, 0, 1,
        1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
        1, 0, 1, 0, 1, 1, 0, 0, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1
    };
```

Η μέθοδος για την τοποθέτηση περιέχει έναν έλεγχο για τις αναφορές από τα εξωτερικά αντικείμενα ώστε να μην προκαλείται κλείσιμο της εφαρμογής σε περίπτωση σφάλματος, και τη κύρια επανάληψη για την τοποθέτηση. Η επανάληψη βασίζεται στην μεταβλητή Boolean placed, η οποία αντιπροσωπεύει αν έχει τοποθετηθεί ο θησαυρός. Στην επανάληψη ανατίθονται τυχαίες τιμές για τα x και z στο πεδίο τιμών [0,9], οι τιμές αυτές χρησιμοποιούνται για τον έλεγχο της ελεύθερης θέσης στον λαβύρινθο μέσω του index και για τις συντεταγμένες του θησαυρού, μέσω των κατάλληλων υπολογισμών. Για το παγκόσμιο σύστημα συντεταγμένων οι τυχαίες τιμές πολλαπλασιάζονται με το δέκα και προστίθεται το κατάλληλο offset, ώστε να αντιστοιχεί η θέση του πίνακα στο περιβάλλον της εφαρμογής. Επειδή στον πίνακα το 0 στις γραμμές αντιπροσωπεύει τις θετικές τιμές του z, σε αντίθεση με το x, ο παρόμοιος υπολογισμός για τη θέση στον z άξονα πρέπει να έχει το αντίθετο πρόσημο από αυτόν του x. Στον πίνακα του λαβυρίνθου το z αντιπροσωπεύεται στις γραμμές και το x τις στήλες. Αν βρεθεί διαθέσιμη θέση, αποθηκεύονται οι τυχαίες τιμές για τη χρησιμοποίηση τους σε επόμενη αλλαγή θέσης, μαρκάρεται ο θησαυρός στον πίνακα του λαβυρίνθου, αλλάζει η θέση του θησαυρού αναθέτοντας νέο διάνυσμα με τις τιμές που υπολογίστηκαν στο position του αντικειμένου και αλλάζει η τιμή του placed ώστε να σταματήσει η επανάληψη. Επίσης για να τοποθετηθεί ο θησαυρός πρέπει να μην βρίσκεται ο bob σε αυτή τη θέση, αυτό γίνεται παίρνοντας το τη θέση του bob και τη πιθανή νέα θέση, και ελέγχεται αν η απόσταση τους είναι μεγαλύτερη του αθροίσματος της ακτίνας της σφαίρας και του κύβου.

```
void PlaceTreasure()
{
    if (sphere == null || GlobalVariables.labyrinth == null)
    {
        Debug.LogError("Required references are missing. Check sphere and labyrinth initialization.");
        return;
    }

    bool placed = false;

    while (!placed)
    {
        int randomInt_x = Random.Range(0, 10);
        int randomInt_z = Random.Range(0, 10);

        float world_x = randomInt_x * 10 - 45;
        float world_z = -(randomInt_z * 10 - 50);

        Vector3 spherePosition = sphere.transform.position;
        Vector3 treasurePosition = new Vector3(world_x, 2.5f, world_z);

        int index = randomInt_z * 10 + randomInt_x;

        if (GlobalVariables.labyrinth[index] == 0 &&
            Vector3.Distance(spherePosition, treasurePosition) > (sphereRadius + cubeHalfSize))
        {
            treasure_x = randomInt_x;
            treasure_z = randomInt_z;

            GlobalVariables.labyrinth[index] = 1;
            transform.position = new Vector3(world_x, 2.5f, world_z);
            placed = true;
        }
    }
}
```

Για την κύρια λειτουργία των θησαυρών δημιουργήθηκε η μέθοδος τύπου IEnumerator TreasureLogic. Η χρήση αυτού του τύπου στη μέθοδο αντί της update έγινε λόγω προβλημάτων επίδοσης της εφαρμογής που οδηγούσαν σε πάγωμα και στη μη ανταπόκρισης της. Η μέθοδος αυτή είναι υπεύθυνη κυρίως για την εμφάνιση των θησαυρών και για τη σμίκρυνσή τους εάν τους έχει ακουμπήσει ο bob. Οι δύο λειτουργίες αυτές βασίζονται στις μεταβλητές treasure_to_appear και treasure_to_small αντίστοιχα.

Όταν η επανάληψη εκτελεί τον κώδικα όπου η τιμή του treasure_to_appear είναι αληθής, χρησιμοποιείται το currentTime, το οποίο λειτουργεί ως ο χρονοδιακόπτης για το πόσο θα εμφανίζεται ένας από τους θησαυρούς σε μία θέση. Αν φτάσει στο μηδέν ξεμαρκάρεται η θέση του θησαυρού στον πίνακα του λαβύρινθου, ο θησαυρός μετακινείται προσωρινά σε ένα σημείο εκτός του λαβυρίνθου του περιβάλλοντος ώστε να μην υπάρξει καμία αλληλεπίδραση με τον bob, καλείται η μέθοδος τοποθέτησης και η currentTime παίρνει ξανά την τιμή για τον χρόνο εμφάνισης. Αν δεν είναι μηδέν απλά μειώνεται η τιμή με βάση τον πραγματικό χρόνο μέσω του Time.deltaTime.

```
84          if (treasure_to_appear)
85          {
86              if (currentTime > 0)
87              {
88                  currentTime -= Time.deltaTime;
89              }
90              else
91              {
92                  int oldIndex = treasure_z * 10 + treasure_x;
93                  GlobalVariables.labyrinth[oldIndex] = 0;
94                  transform.position = new Vector3(-70, -2.5f, -70);
95                  PlaceTreasure();
96                  currentTime = countdownTime;
97              }
98          }
```

Αν ο θησαυρός δεν πρέπει να εμφανίζεται, τότε το script βρίσκεται στη διαδικασία της σμίκρυνσης του, όπου με παρόμοιο τρόπο με τον χρονοδιακόπτη η εφαρμογή συρρικνώνει τον εκάστοτε κύβο μέχρι να φτάσει στο μηδέν, όπου ο κύβος πάλι τοποθετείται σε σημείο εκτός λαβυρίνθου και ενεργοποιείται πάλι ο χρονοδιακόπτης αλλά ώστε να ξεκινήσει η διάρκεια της εξαφάνισης.

```
99          else
100         {
101             if (treasure_to_small)
102             {
103                 if (transform.localScale.x > 0)
104                 {
105                     transform.localScale -= new Vector3(0.1f, 0.1f, 0.1f);
106                 }
107                 else
108                 {
109                     treasure_to_small = false;
110                     transform.position = new Vector3(-70, -2.5f, -70);
111                     currentTime = dissapearCountdownTime;
112                 }
113             }
```

Η διαδικασία της εξαφάνισης απλά χρησιμοποιεί το currentTime πάλι ως χρονοδιακόπτη, με τη τιμή έναρξης να είναι 3 δευτερόλεπτα για όλους τους θησαυρούς. Αν φτάσει στο μηδέν επαναφέρεται η λειτουργία εμφάνισης του θησαυρού και το αρχικό του scale.

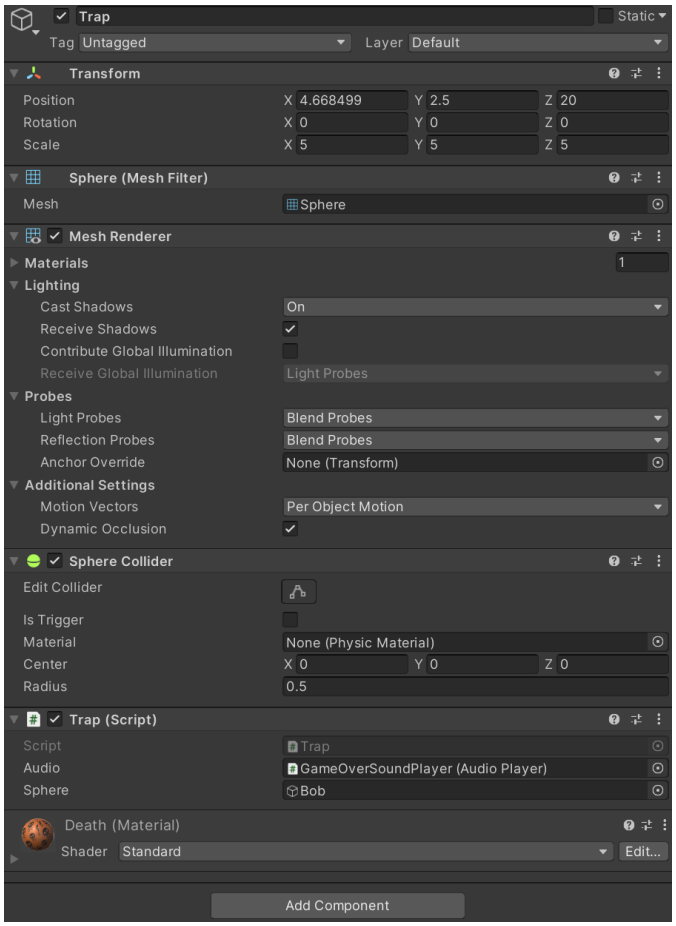
```
114             else
115             {
116                 if (currentTime > 0)
117                 {
118                     currentTime -= Time.deltaTime;
119                 }
120                 else
121                 {
122                     treasure_to_appear = true;
123                     add_score = true;
124                     transform.localScale = new Vector3(5, 5, 5);
125                 }
126             }
127         }
128
129         yield return null;
130     }
131 }
```

Για τις λειτουργίες της εξαφάνισης και της σμίκρυνσης δημιουργήθηκε η μέθοδος για τα collision, όπου αν οι θησαυροί συγκρουστούν με τον bob αλλάζουν ανάλογα οι Boolean μεταβλητές που χρησιμοποιούνται στην treasure logic. Οι υπόλοιπες λειτουργίες που κάνει η μέθοδος εξηγούνται στα bonus.

```
133     void OnCollisionEnter(Collision collision)
134     {
135         if (collision.gameObject.name == "Bob")
136         {
137             treasure_to_appear = false;
138             treasure_to_small = true;
139             if (add_score)
140             {
141                 GlobalVariables.AddScore(score);
142                 audio.PlaySound();
143                 popup.ShowPopup(transform.position + new Vector3 (0,0,5));
144                 add_score = false;
145             }
146         }
147     }
148 }
```

Ερώτημα iv:

Η παγίδα είναι μια σφαίρα με scale (5,5,5). Η υφή τοποθετήθηκε όπως σε προηγούμενα ερωτήματα.



Το script για την παγίδα χρησιμοποιεί την ίδια λογική με τους θησαυρούς στο προηγούμενο ερώτημα. Η διαφορά είναι στην μέθοδο TrapLogic όπου η

currentTime πλέον παίρνει τυχαίες τιμές. Οι τιμές για την εμφάνιση είναι από 1 εώς 7 δευτερόλεπτα και για την εξαφάνιση από 0 έως 5 δευτερόλεπτα, εκτός από την αρχή της εφαρμογής όπου είναι από 3 έως 7 δευτερόλεπτα. Στην επαφή με τον bob δεν υπάρχει σμίκρυνση, στη μέθοδο για το collision αλλάζει η μεταβλητή game on στα GlobalVariables, η οποία στην ουσία επηρεάζει όλα τα υπόλοιπα αντικείμενα εκτός της κάμερας, και σταματάει τη λειτουργία της εφαρμογής. Στην εφαρμογή μετά το collision του bob και της παγίδας πλέον δεν μπορεί ο χρήστης να κουνήσει τον bob και οι θησαυροί και η παγίδα δεν αλλάζουν την τοποθεσία τους. Η εφαρμογή δεν κλείνει, αφήνεται στον χρήστη το πως θα το χειριστεί.

```
IEnumerator TrapLogic()
{
    while (true)
    {
        if (!GlobalVariables.gameOn)
        {
            yield return null;
            continue;
        }

        if (appear)
        {
            if (currentTime > 0)
            {
                currentTime -= Time.deltaTime;
            }
            else
            {
                transform.localScale = new Vector3(0, 0, 0);
                allow_death = false;
                appear = false;

                transform.position = new Vector3(0, -10, 0);

                int oldIndex = trap_z * 10 + trap_x;
                if (oldIndex >= 0 && oldIndex < GlobalVariables.labyrinth.Length)
                {
                    GlobalVariables.labyrinth[oldIndex] = 0;
                }
                currentTime = Random.Range(0, 5);
            }
        }
        else
        {
            if (currentTime > 0)
            {
                currentTime -= Time.deltaTime;
            }
            else
            {
                appear = true;
                allow_death = true;
                PlaceTrap();
                currentTime = Random.Range(1, 7);
            }
        }

        yield return null;
    }
}
```

```
    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.name == "Bob")
        {
            if (allow_death)
            {
                audio.PlaySound();
                allow_death = false;
                GlobalVariables.gameOn = false;
                Debug.Log("Player collided with trap. Game Over.");
            }
        }
    }
}
```

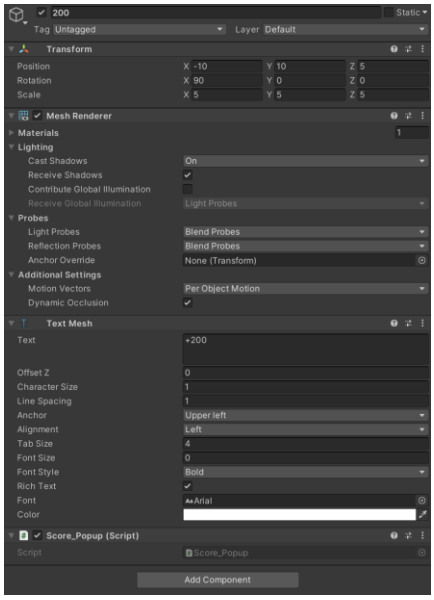
Ερώτημα v:

Για την κάμερα δημιουργήθηκε το script MainCamera όπου στο update υπάρχει έλεγχος για τα βελάκια, τα + και – και τα r και e. Στα βελάκια το position της κάμερας αλλάζει προσθέτοντας τα διανύσματα transform.right και up ώστε να η κίνηση να γίνεται ανάλογα τη θέση αλλά και το που βλέπει η κάμερα, πολλαπλασιασμένα με τη ταχύτητα που είναι ίση με 0,1. Τα βελάκια χρησιμοποιούν τα ανάλογα διανύσματα. Τα + - χρησιμοποιούν παρόμοια το transformation.forward. Τα e και r στριφογυρίζουν την κάμερα γύρω από τον y άξονα μέσω του transform.Rotate με παρόμοια ταχύτητα όπως τα βελάκια.

```
5   public class MainCamera : MonoBehaviour
6   {
7       private float speed = 1f;
8
9       // Update is called once per frame
10      void Update()
11      {
12          if (Input.GetKey("left")) transform.position += -transform.right * speed;
13          if (Input.GetKey("right")) transform.position += transform.right * speed;
14          if (Input.GetKey("up")) transform.position += transform.up * speed;
15          if (Input.GetKey("down")) transform.position += -transform.up * speed;
16          if (Input.GetKey(KeyCode.Plus) || Input.GetKey(KeyCode.KeypadPlus)) transform.position += transform.forward * speed;
17          if (Input.GetKey(KeyCode.Minus) || Input.GetKey(KeyCode.KeypadMinus)) transform.position += -transform.forward * speed;
18          if (Input.GetKey("r")) transform.Rotate(0, 1f, 0); // Rotating around the Y-axis
19          if (Input.GetKey("e")) transform.Rotate(0, -1f, 0);
20      }
21  }
```

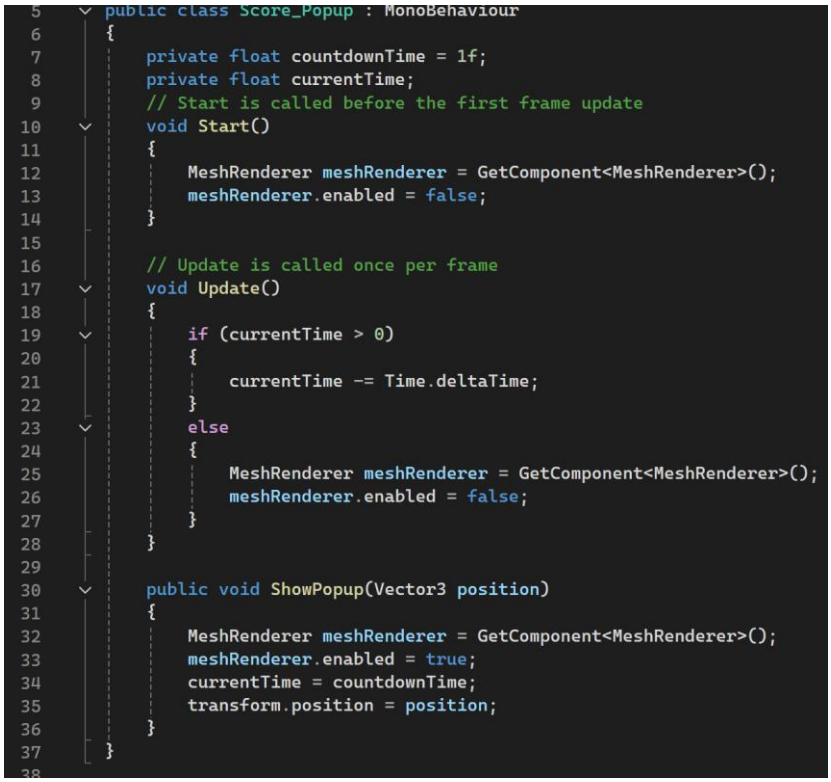
Bonus a:

Για τα οπτικά εφέ δημιουργήθηκαν τρία αντικείμενα text τα οποία αναγράφουν το σκορ για κάθε θησαυρό (εξηγείται παραπάνω στο bonus c). Τα αντικείμενα αυτά περιέχουν το script Score_Popup.



Στην αρχή της εφαρμογής το mesh renderer απενεργοποιείται ώστε να μην εμφανίζεται το αντικείμενο στη κάμερα. Η μέθοδος ShowPopup είναι υπεύθυνη για την επανεργοποίηση του mesh renderer, το ξεκίνημα ενός παρόμοιου χρονοδιακόπτη με τα προηγούμενα ερωτήματα και με χρόνο 1 δευτερόλεπτα, και

τη τοποθέτηση του κειμένου με το διάνυσμα που δίνεται στα ορίσματα. Όταν ο χρονοδιακόπτης φτάνει στο μηδέν, ο mesh renderer απενεργοποιείται ξανά.



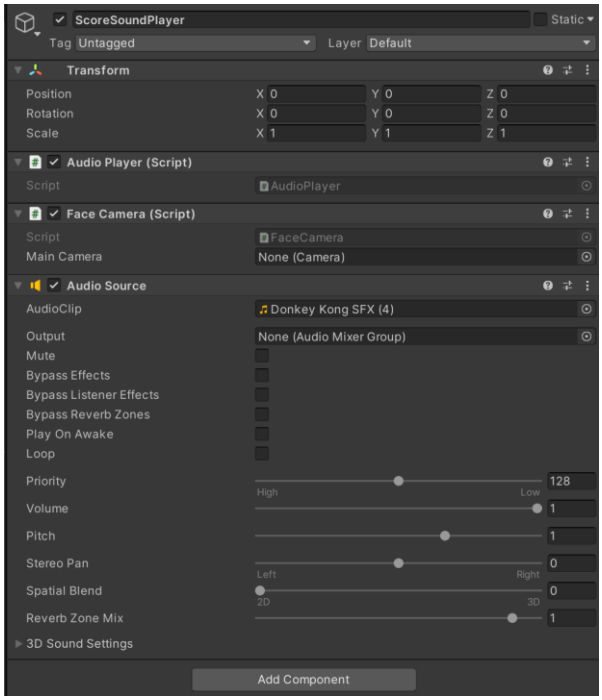
```
public class Score_Popup : MonoBehaviour
{
    private float countdownTime = 1f;
    private float currentTime;
    // Start is called before the first frame update
    void Start()
    {
        MeshRenderer meshRenderer = GetComponent<MeshRenderer>();
        meshRenderer.enabled = false;
    }

    // Update is called once per frame
    void Update()
    {
        if (currentTime > 0)
        {
            currentTime -= Time.deltaTime;
        }
        else
        {
            MeshRenderer meshRenderer = GetComponent<MeshRenderer>();
            meshRenderer.enabled = false;
        }
    }

    public void ShowPopup(Vector3 position)
    {
        MeshRenderer meshRenderer = GetComponent<MeshRenderer>();
        meshRenderer.enabled = true;
        currentTime = countdownTime;
        transform.position = position;
    }
}
```

Στους θησαυρούς, έχοντας τοποθετήσει ένα από αυτά τα αντικείμενα μέσω του editor, στο collision με τον bob καλείται η μέθοδος ShowPopup δίνοντας την τοποθεσία του θησαυρού με ένα μικρό offset στον άξονα z, για καλύτερη προβολή στον χρήστη.

Για τα ηχητικά εφέ δημιουργήθηκε ένα άδειο αντικείμενο ScoreSoundPlayer το οποίο περιέχει ένα Audio Source component της unity το οποίο μπορεί να παίξει αρχεία ήχου, συγκεκριμένα για τους θησαυρούς χρησιμοποιείται ένα sound effect από το παιχνίδι donkey kong.



Το αντικείμενο αυτό περιέχει και το script Audio Player. Στο script αυτό υπάρχει η μέθοδος PlaySound όπου εκτελείται η μέθοδος play στο component audio source. Η μέθοδος γίνεται διαθέσιμη στους θησαυρούς όπως και τα αντικείμενα κειμένου.

```
3   public class AudioPlayer : MonoBehaviour
4   {
5       private AudioSource audioSource;
6
7       void Start()
8       {
9           // Get the AudioSource component attached to this GameObject
10          audioSource = GetComponent<AudioSource>();
11      }
12
13      public void PlaySound()
14      {
15          audioSource.Play();
16      }
17  }
```

Επίσης για να μην εμφανίζονται πολλαπλές φορές τα οπτικά και ηχιτικά εφέ από τυχών πολλαπλά collision που γίνονται λόγω της φυσικής του rigid body του bob, χρησιμοποιείται μια Boolean μεταβλητή που το επιτρέπει μόνο μία φορά.

Bonus b:

Στο script του bob υπάρχει έλεγχος για τα κουμπιά m και n τα οποία αυξάνουν και μειώνουν την μεταβλητή της ταχύτητας. Η αύξηση/μείωση γίνεται με τη τιμή 25 και υπάρχουν έλεγχοι ώστε να μην ξεπερνάει η ταχύτητα τη τιμή 125 και να μην είναι πιο κάτω από τη τιμή 25, διασφαλίζοντας έτσι τις πέντε διαβαθμίσεις.

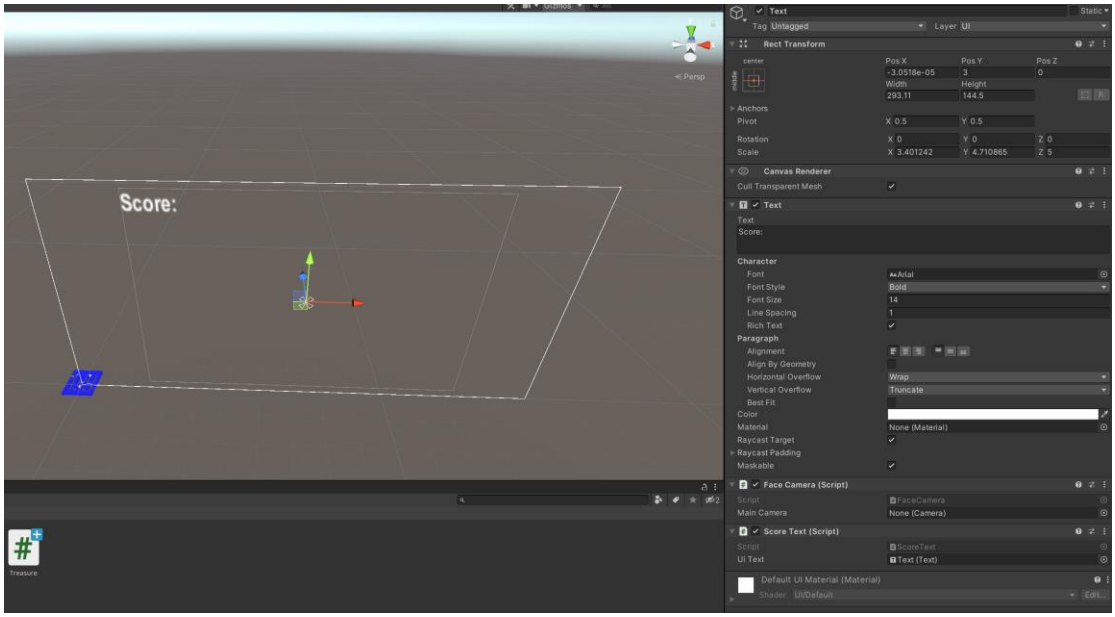
```
36              if (Input.GetKeyDown("m")) speed += 25;
37              if (speed > 125) speed = 125;
38              if (Input.GetKeyDown("n")) speed -= 25;
39              if (speed < 25) speed = 25;
```

Bonus c:

Για το σκορ, υπάρχει η μεταβλητή score στο GlobalVariables, η οποία όταν ξεκινάει η εφαρμογή ξεκινάει έχει τιμή 0. Στα collision των θησαυρών καλείται η AddScore ώστε να προστεθεί το σκορ του ανάλογου θησαυρού. Το κεράσι επειδή βρίσκεται το λιγότερο χρόνο σε μία θέση έχει σκορ 300, το πορτοκάλι 200 και το λεμόνι που βρίσκεται το περισσότερο σε μία θέση έχει σκορ 100.

```
26      // Start is called before the first frame update
27      void Start()
28      {
29          score = 0;
30      }
31
32      public static void AddScore (int add)
33      {
34          score += add;
35      }
36  }
37
```

Το σκορ εμφανίζεται στην αριστερά πάνω γωνία του παραθύρου μέσω ενός αντικείμενου ui text



Το αντικείμενο περιέχει δύο script, το FaceCamera και το ScoreText. Το ScoreText είναι υπεύθυνο αν ο bob δεν έχει πέσει στη παγίδα, να αλλάζει το κείμενο με τη τιμή του score στο Global Variables. Το FaceCamera είναι υπεύθυνο για την σωστή τοποθέτηση του ui text μπροστά στην κάμερα παίρνοντας τα διανύσματα

της κάμερας για το look at του αντικειμένου. Το script αυτό χρησιμοποιήθηκε και στα αντικείμενα για τον ήχο.

```
public class ScoreText : MonoBehaviour
{
    public Text uiText; // Reference to the Text component

    void Update()
    {
        if (GlobalVariables.gameOn)
        {
            uiText.fontSize = 14;
            uiText.alignment = TextAnchor.UpperLeft;
            uiText.text = "Score: " + GlobalVariables.score;
        }
        else
        {
            uiText.alignment = TextAnchor.MiddleCenter;
            uiText.fontSize = 30;
            uiText.text = "GAME OVER\nFinal Score: " + GlobalVariables.score;
        }

    }
}
```

```
public class FaceCamera : MonoBehaviour
{
    public Camera mainCamera;

    void Update()
    {
        if (mainCamera != null)
        {
            // Always face the camera
            transform.LookAt(mainCamera.transform);
        }
    }
}
```

Bonus d:

Για το τέλος του παιχνιδιού αφού το game on παίρνει τιμή false, το κείμενο που εμφανίζεται βρίσκεται στο κέντρο της οθόνης και εκτός από το game over εμφανίζει και το σκορ που έχει μαζευτεί. Επίσης έχει δημιουργηθεί και ένα άλλο αντικείμενο Audio Player το οποίο έχει διαφορετικό αρχείο ήχου από το παιχνίδι sonic the hedgehog το οποίο χρησιμοποιείται από την παγίδα στο collision με τον bob.

2.Πληροφορίες σχετικά με την υλοποίηση

Η εργασία υλοποιήθηκε στο λειτουργικό Windows 11, χρησιμοποιώντας την έκδοση unity 2020.3.24f1. Στο zip υπάρχει ο φάκελος Project 2 - Treasure Bob ο οποίος περιέχει τα αρχεία που χρησιμοποιεί ο editor, και εκεί μέσα περιέχεται ο φάκελος build που έχει το .exe και τα υπόλοιπα αρχεία από τη διαδικασία του build. Το build έγινε για windows σε αρχιτεκτονική x86_64, οπότε η εφαρμογή δεν μπορεί

να τρέξει σε άλλα λειτουργικά ή επεξεργαστές ARM, απλών x86 των 32 bit ή άλλων αρχιτεκτονικών.

Παρατηρήθηκε ότι η εφαρμογή δουλεύει διαφορετικά στον editor και στο εκτελέσιμο σε θέματα ταχύτητας στην σμίκρυνση και στη κίνηση της κάμερας. Για αυτό η ταχύτητες που χρησιμοποιήθηκαν είναι προσαρμοσμένες να αντιστοιχούν σε καλύτερη εμπειρία του εκτελέσιμου.

Το md5 του zip:

61798BDBB0BCFA1DA92AEC5BA7ABED01

Το link στο google drive:

https://drive.google.com/file/d/1AQQHWUWFRM_asDyBqI0-QwVB94YBj9cK/view?usp=sharing

3. Σύντομη αξιολόγηση της λειτουργίας της ομάδας σας και της συνεργασίας

Στην εργασία δεν υπήρξε συνεργασία με κάποιο άλλο άτομο, οπότε δεν είναι δυνατός ο σχολιασμός λειτουργίας ομάδας.

4.Αναφορές – Πηγές

- https://ecourse.uoi.gr/mod/resource/view.php?id=117453
- https://discussions.unity.com/t/coroutines-vs-update/428844
- https://discussions.unity.com/t/ienumerator-or-update/705042
- https://www.reddit.com/r/Unity3D/comments/evbq0v/update_vs_coroutine_elegance_performance_and/
- https://discussions.unity.com/t/what-are-ienumerator-and-coroutine/143510
- https://stackoverflow.com/questions/64258574/what-is-an-ienumerator-in-c-sharp-and-what-is-it-used-for-in-unity
- https://docs.unity3d.com/6000.0/Documentation/ScriptReference/AudioSource.html
- https://docs.unity3d.com/2017.3/Documentation/ScriptReference/UI.Text-text.html
- https://web.archive.org/web/20201111215913/http://wiki.unity3d.com/index.php?title=CameraFacingBillboard