

Περιεχόμενα

Περιεχόμενα	i
1 Εγχειρίδιο Χρήστη	2
1.1 Πλοήγηση	2
1.1.1 Βασικές κινήσεις	2
1.1.2 Ο χώρος του Μουσείου	3
1.2 Επικοινωνία με τον Επιστάτη του Μουσείου	5
1.3 Ανασκόπηση εκθεμάτων	7
1.4 Αγορά εκθεμάτων	8
1.4.1 Προσθήκη εκθέματος στο καλάθι	8
1.4.2 Αφαίρεση εκθέματος από το καλάθι	8
1.4.3 Παραγγελία και αγορά εκθέματος	9
1.5 Προβολή ταινίας	11
1.6 Online Help	12
2 Εγχειρίδιο Προγραμματιστή	14
2.1 Παίκτης και εκθέματα	14
2.1.1 Παίκτης	14
2.1.2 Ανασκόπηση εκθεμάτων	18
2.2 Αγορά εκθεμάτων	24
2.2.1 Ιεραρχική ανάλυση εργασιών	24
2.2.2 Καλάθι αγορών	24

2.2.3	Εναλλαγή Panel	31
2.2.4	Ολοκλήρωση αγοράς	33
2.3	Εικονικός πράκτορας	38
2.4	Προβολή ταινίας	41
2.5	Online Help	47

Σημείωση

Ο κώδικας υλοποίησης του project βρίσκεται στο GitHub και συγκεκριμένα στο Branch Margarita2.

Πατήστε **εδώ** ή ακολουθήστε το URL που ακολουθεί:

<https://github.com/Georgegipa/Museum/tree/Margarita2>

Κεφάλαιο 1

Εγχειρίδιο Χρήστη

Το εγχειρίδιο χρήστη αποτελεί ένα βιομητικό έγγραφο που εξηγεί αναλυτικά πως ο χρήστης μπορεί να αλληλεπιδράσει με την εφαρμογή του μουσείου. Περιλαμβάνονται screenshots με όλες τις οιδόνες που μπορούν να προβληθούν στον χρήστη. Σε κάθε οιδόνη εξηγούνται οι επιλογές του χρήστη, κάθε επιλογή μπορεί είτε να οδηγήσει τον χρήστη σε επόμενη ή προηγούμενη οιδόνη είτε να παρουσιάσει ένα νέο panel που θα προσδιορίζονται νέες επιλογές.

1.1 Πλοήγηση

1.1.1 Βασικές κινήσεις

Μπορείτε να πλοηγηθείτε σε όλους τους χώρους του μουσείου με τα ακόλουθα πλήκτρα:

W Κίνηση προς τα εμπρός

S Κίνηση προς τα πίσω

A Κίνηση αριστερά

D Κίνηση δεξιά



Μπορείτε επίσης να αλλάξετε την οπτική γωνία της κάμερας χρησιμοποιώντας το ποντίκι.

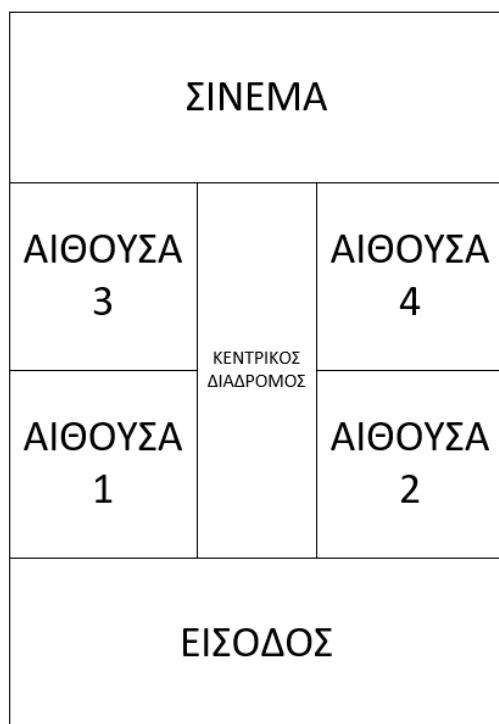


Κρατώντας πατημένο το πλήκτρο shift ενώ μετακινείστε, έχετε την δυνατότητα να μετακινηθείτε με αυξημένη ταχύτητα.



1.1.2 Ο χώρος του Μουσείου

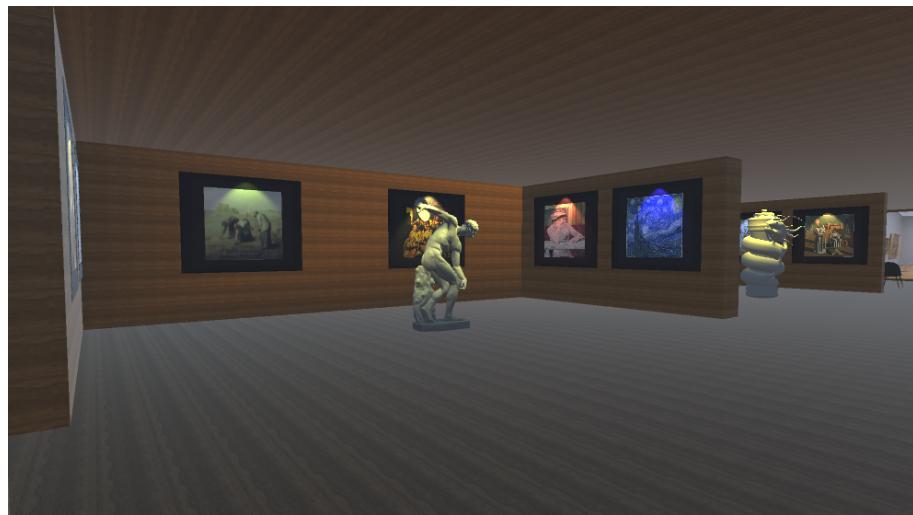
Το μουσείο αποτελείται από την είσοδο, τον κεντρικό διάδρομο, τέσσερεις αίθουσες με εκθέματα και την αίθουσα σινεμά.



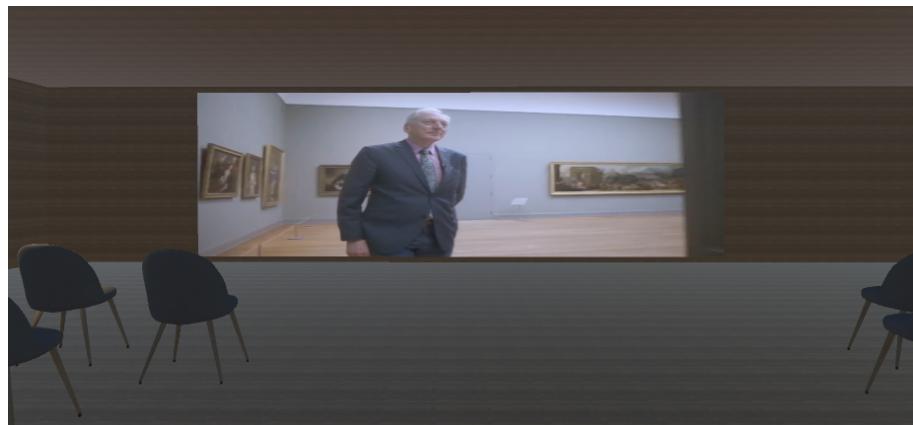
Η πλοήγησή σας ξεκινά από την είσοδο του μουσείου.



Έχετε την δυνατότητα να μετακινηθείτε κατά μήκος του κεντρικού διαδρόμου και να εισέλθετε στις αίθουσες στα δεξιά και αριστερά, ώστε να δείτε τα εκθέματα.



Αν μεταβείτε στο τέλος του διαδρόμου, όπου γίνεται προβολή ταινίας.

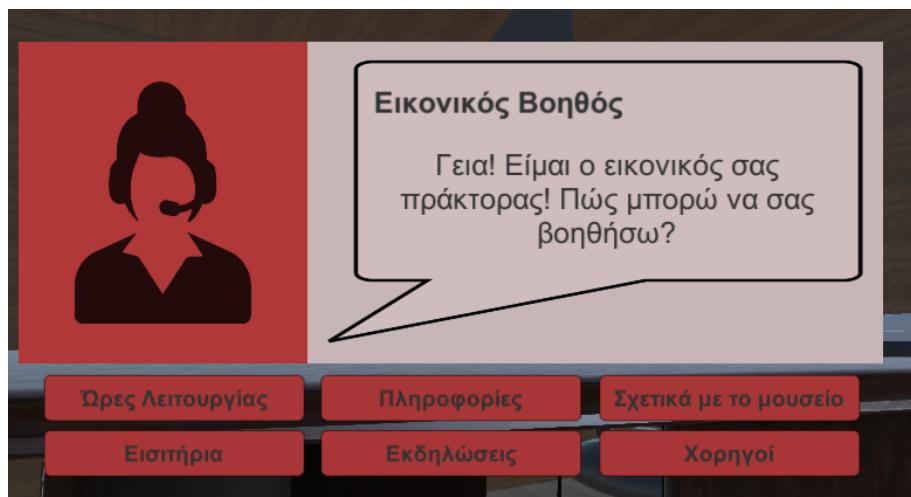


1.2 Επικοινωνία με τον Επιστάτη του Μουσείου

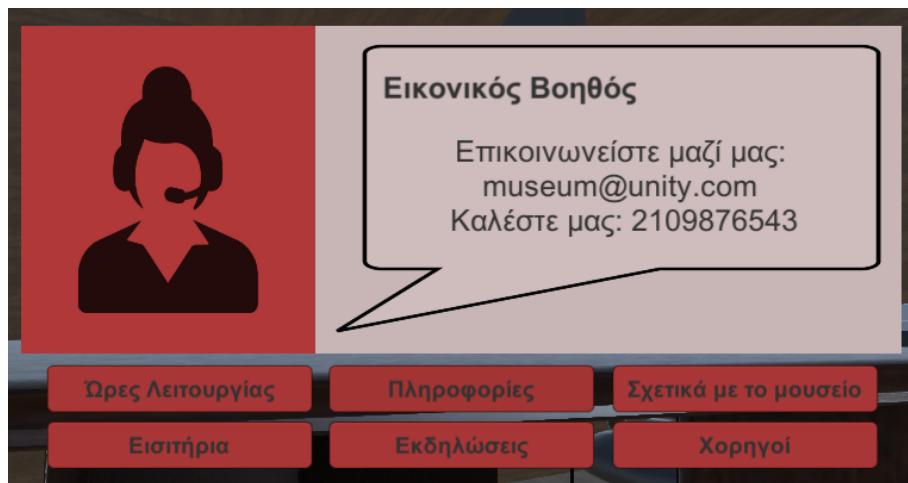
Στην είσοδο του μουσείου, υπάρχει η επιλογή να κάνετε ερωτήσεις για το μουσείο. Αυτό γίνεται μέσω του γραφείου που υπάρχει στην είσοδο.



Όταν πλησιάσετε το γραφείο, θα εμφανιστεί το ακόλουθο μενού επιλογών:



Μπορείτε να κάνετε κλικ σε μια από τις επιλογές που δίνονται, οι οποίες αποτελούν τις πιο συχνές ερωτήσεις για το μουσείο. Μόλις κλικάρετε την επιλογή που επιθυμείτε θα εμφανιστεί η απάντηση στην ερώτησή σας.

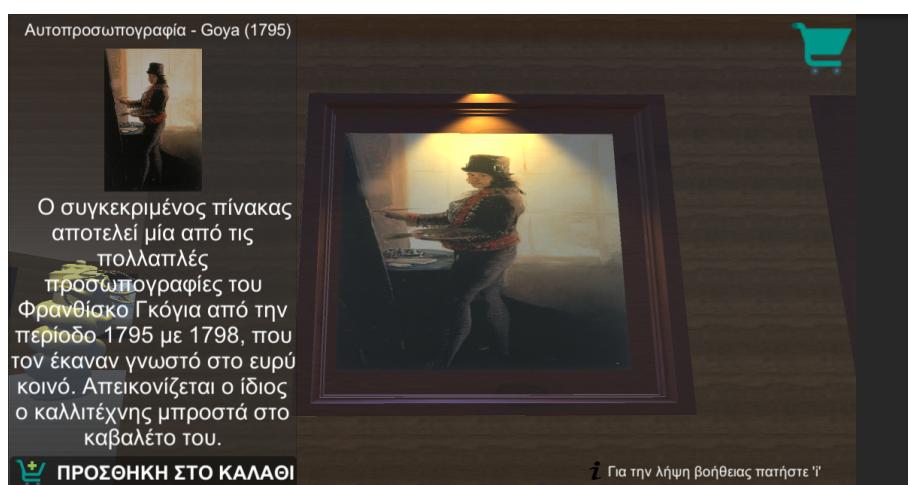


Αν απομακρυνθείτε από το γραφείο, το μενού επιλογών θα εξαφανιστεί και θα μπορέσετε να συνεχίσετε την πλοήγησή σας.

1.3 Ανασκόπηση εκθεμάτων

Μόλις εισέλθετε σε μία από τις τέσσερεις αίθουσες του μουσείου, έχετε μπροστά σας τα εκθέματα της αίθουσας αυτής.

Όταν πλησιάσετε το έκθεμα που επιθυμείτε, θα εμφανιστούν στο αριστερό μέρος της οθόνης σας πληροφορίες για το έκθεμα αυτό.



1.4 Αγορά εκθεμάτων

Έχετε τη δυνατότητα να αγοράσετε ένα ψηφιακό αντίγραφο των εκθεμάτων που επιθυμείτε. Αρχικά, πρέπει να προσθέσετε τα εκθέματα στο ψηφιακό καλάθι αγορών. Στη συνέχεια, πρέπει να ολοκληρώσετε την αγορά σας, τοποθετώντας τα προσωπικά σας στοιχεία στα αντίστοιχα πεδία.

1.4.1 Προσθήκη εκθέματος στο καλάθι

Όταν κάνετε ανασκόπηση ενός εκθέματος, υπάρχει το κουμπί



στην κάτω αριστερή πλευρά της οθόνης σας. Κάντε κλικ στο κουμπί αυτό για να προσθέσετε το παρόν έκθεμα στο ψηφιακό καλάθι αγορών σας.



1.4.2 Αφαίρεση εκθέματος από το καλάθι

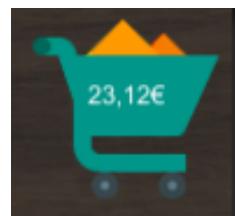
Μόλις προσθέσετε ένα αντικείμενο στο καλάθι, θα παρατηρήσετε ότι το κουμπί έχει, πλέον, την επιγραφή



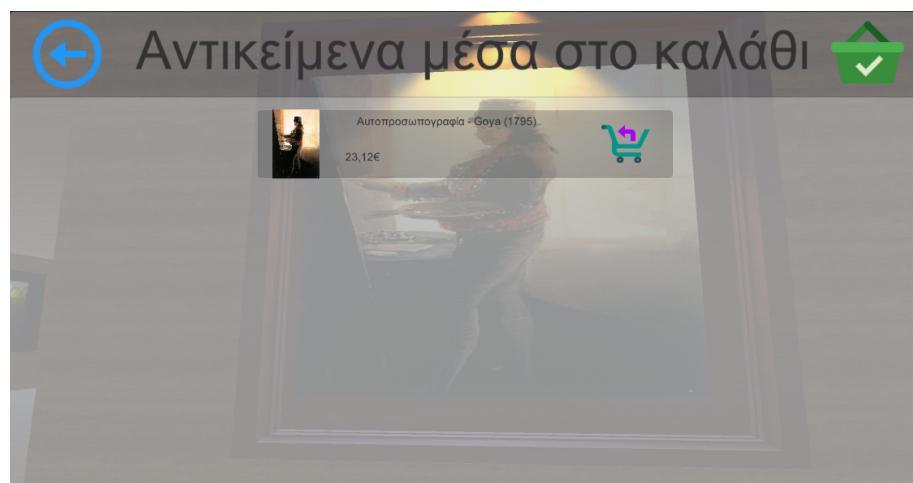
Μπορείτε να πατήσετε το κουμπί αυτό, για να αφαιρέσετε το παρόν έκθεμα από το καλάθι σας.

1.4.3 Παραγγελία και αγορά εκθέματος

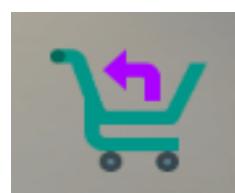
Όταν έχετε προσθέσει ένα έκθεμα στο καλάθι σας και επιθυμείτε να ολοκληρώσετε την αγορά, πρέπει να κλικάρετε το εικονίδιο του καλαθιού που βρίσκεται στην πάνω δεξιά γωνία της οθόνης σας.



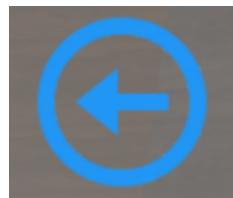
Μόλις πατήσετε το εικονίδιο, θα εμφανιστεί στην οθόνη σας η λίστα με όλα τα εκθέματα που έχετε τοποθετήσει στο καλάθι σας.



Για άλλη μια φορά έχετε την δυνατότητα να αφαιρέσετε κάτι από το καλάθι σας πατώντας το εικονίδιο της ΑΦΑΙΡΕΣΗΣ.



Αν θέλετε να συνεχίσετε την πλοιήγησή σας στο μουσείο, πατήστε το κουμπί:



Αν επιθυμείτε να ολοκληρώσετε την αγορά σας, πατήστε το κουμπί:

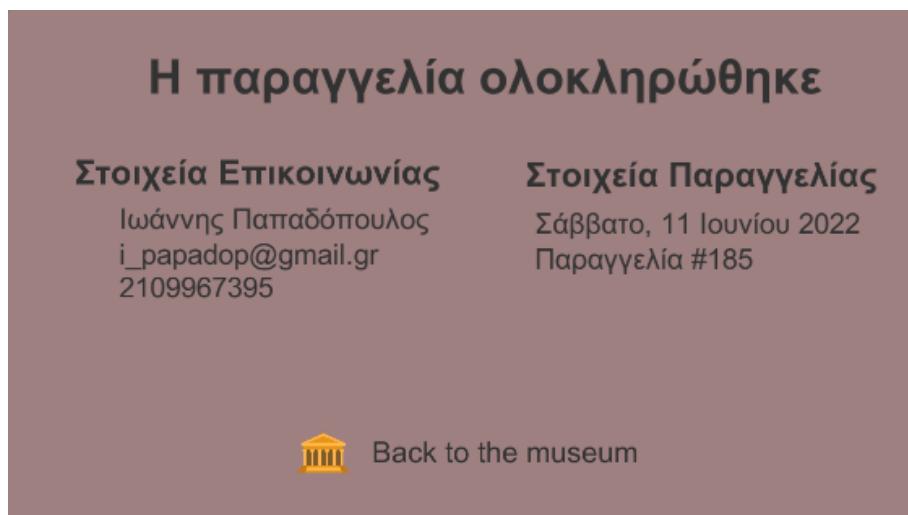


Μόλις πατήσετε το παραπάνω κουμπί, θα εμφανιστεί στην οθόνη σας το ακόλουθο παράθυρο:

Για να ολοκληρώσετε την αγορά σας, πρέπει να τοποθετήσετε τα προσωπικά στοιχεία που ζητούνται στα αντίστοιχα πεδία. Στη συνέχεια πατήστε το κουμπί “Πληρωμή” στο κάτω δεξιά σημείο της οθόνης σας.

The screenshot shows a payment interface for a purchase of 23,12 €. It includes fields for shipping address (Στοιχεία Παραγγελίας), delivery method (Επιλογές Παραλαβής), payment method (Τρόπος πληρωμής), and card details (Στοιχεία Κάρτας). The payment method section shows 'Παραλαβή από το Μουσείο' (Pickup at the Museum) selected. Payment methods shown are MasterCard and VISA. Card details include a card number (475829405724), name (Papadopoulos), CVV (***), and expiration date (05/2026). A blue 'Πληρωμή' (Pay) button is at the bottom right.

Αν η παραγγελία σας ολοκληρωθεί με επιτυχία θα εμφανιστεί το ακόλουθο παράθυρο στην οθόνη:



Πατώντας "Back to museum", επιστρέψετε στον χώρο του μουσείου.

1.5 Προβολή ταινίας

Υπάρχει δυνατότητα να παρακολουθήσετε ταινία στην αίθουσα σινεμά του μουσείου.

Μόλις μεταβείτε στην αίθουσα σινεμά στο τέλος του κεντρικού διαδρόμου θα ξεκινήσει αυτόματα να παίζει η ταινία στην οθόνη.



Έχετε την δυνατότητα να σταματήσετε προσωρινά την ταινία:



και να την ξαναρχίσετε:



ή να την σταματήσετε αντίστοιχα.



1.6 Online Help

Παρέχεται online βοήθεια για όλες τις δυνατότητες που αναφέρονται προηγουμένως, πατώντας το πλήκτρο I στο πληκτρολόγιό σας.

i Για την λήψη βοήθειας πατήστε 'i'

Μόλις πατήσετε το πλήκτρο I στο πληκτρολόγιό σας θα εμφανιστεί στο δεξί μέρος της οθόνης σας το ακόλουθο παράθυρο:



Πατώντας το πλήκτρο E στο πληκτρολόγιό σας μπορείτε να εμφανίσετε τις επόμενες σελίδες με οδηγίες.



Αν πατήσετε ξανά το πλήκτρο I, μπορείτε να κλείσετε το παράθυρο του εικονικού βοηθού.

Κεφάλαιο 2

Εγχειρίδιο Προγραμματιστή

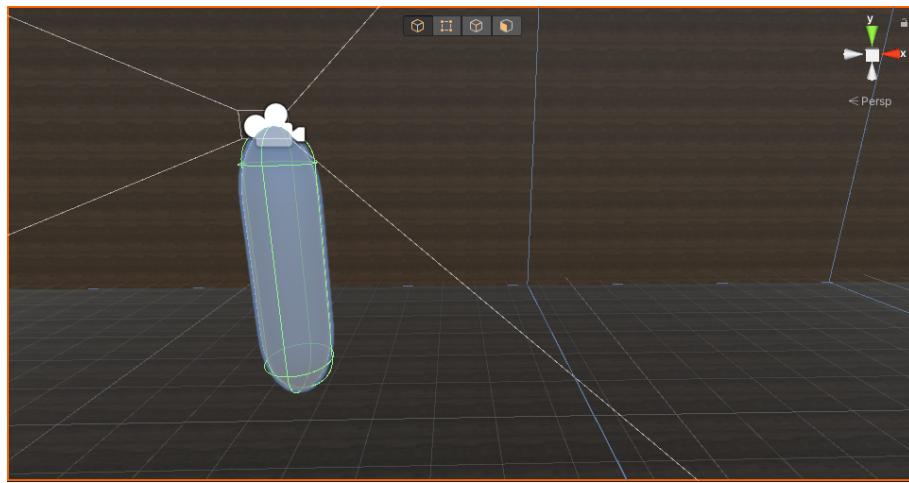
Στο εγχειρίδιο προγραμματιστή αναλύονται τα βασικά προγραμματιστικά μέρη της εφαρμογής μας. Για κάθε λειτουργία που προέβλεπε η εκφώνηση της εργασίας έχουν δημιουργηθεί ξεχωριστά script που είτε λειτουργούν ατομικά είτε συνεργάζονται με άλλα script με σκοπό την πλήρη λειτουργικότητα του Μουσείου.

Καθώς η ανάπτυξη της εφαρμογής έγινε σε Unity ορισμένα αντικείμενα-κλάσεις των script έχουν ανατεθεί στα αντίστοιχα αντικείμενα του περιβάλλοντος Unity.

2.1 Παίκτης και εκθέματα

2.1.1 Παίκτης

Ο player που αντιπροσωπεύει τον κανονικό χρήστη είναι τοποθετημένος στη αρχή του παιχνιδιού στον είσοδο του μουσείου.



Σαν GameObject είναι συνδεδεμένος με το script PlayerController, το οποίο βρίσκεται στον φάκελο: Assets/Scripts/PlayerScripts.

```
public class PlayerController : MonoBehaviour
{
    [SerializeField] Transform playerCamera = null;
    [SerializeField] float mouseSensitivity = 3.5f;
    [SerializeField] float walkSpeed = 6.0f;
    [SerializeField] float maxSprintSpeed = 12.0f;
    [SerializeField] float gravity = -13.0f;
    [SerializeField] float speedScale = 0.01f;
    public static bool move = true;
    private float defaultSpeed;
    bool lockMouse = false;

    float cameraPitch = 0.0f;
    float velocityY = 0.0f;

    [SerializeField] [Range(0.0f, 0.5f)] float moveSmoothTime = 0.3f;
    [SerializeField] [Range(0.0f, 0.5f)] float mouseSmoothTime =
        0.03f;

    CharacterController controller = null;
    Vector2 currentDir = Vector2.zero;
    Vector2 currentDirVelocity = Vector2.zero;

    Vector2 currentMouseDelta = Vector2.zero;
    Vector2 currentMouseDeltaVelocity = Vector2.zero;
```

```
void Start()
{
    controller = GetComponent<CharacterController>();
    defaultSpeed = walkSpeed;
    Cursor.lockState = CursorLockMode.Locked;
    Cursor.visible = false;
    Cursor.lockState = CursorLockMode.Locked;
}

void Update()
{
    UpdateMouseLook();
    UpdateMovement();
    if (Input.GetKey(KeyCode.LeftShift))
    {
        if (walkSpeed <= maxSprintSpeed)
            walkSpeed += speedScale;
    }
    else
    {
        if (walkSpeed != defaultSpeed)
            walkSpeed -= speedScale;
    }
}

void UpdateMouseLook()
{
    if (!lockMouse)
    {
        Vector2 targetMouseDelta = new
            Vector2(Input.GetAxis("Mouse X"),
                    Input.GetAxis("Mouse Y"));
        currentMouseDelta = Vector2.SmoothDamp(currentMouseDelta,
                                                targetMouseDelta, ref currentMouseDeltaVelocity,
                                                mouseSmoothTime);

        cameraPitch -= currentMouseDelta.y * mouseSensitivity;

        cameraPitch = Mathf.Clamp(cameraPitch, -90.0f, 90.0f);

        playerCamera.localEulerAngles = Vector3.right *
            cameraPitch;
    }
}
```

```
        transform.Rotate(Vector3.up * currentMouseDelta.x *
                         mouseSensitivity);
    }
}

void UpdateMovement()
{
    if (move)
    {
        Vector2 targetDir = new
            Vector2(Input.GetAxisRaw("Horizontal"),
                    Input.GetAxisRaw("Vertical"));
        targetDir.Normalize();

        currentDir = Vector2.SmoothDamp(currentDir, targetDir,
                                         ref currentDirVelocity, moveSmoothTime);

        if (controller.isGrounded)
            velocityY = 0.0f;

        velocityY += gravity * Time.deltaTime;

        Vector3 velocity = (transform.forward * currentDir.y +
                            transform.right * currentDir.x) * walkSpeed +
                           Vector3.up * velocityY;
        controller.Move(velocity * Time.deltaTime);
    }
}

private void OnTriggerEnter(Collider other)
{
    if (other.tag != null)
    {
        Cursor.visible = true;
        lockMouse = true;
        Cursor.lockState = CursorLockMode.None;
    }
}

private void OnTriggerExit(Collider other)
{
    if (other.tag != null)
```

```

    {
        Cursor.visible = false;
        lockMouse = false;
        Cursor.lockState = CursorLockMode.Locked;
    }
}
}

```

To script αυτό ορίζει κάποια τεχνικά χαρακτηριστικά όπως είναι:

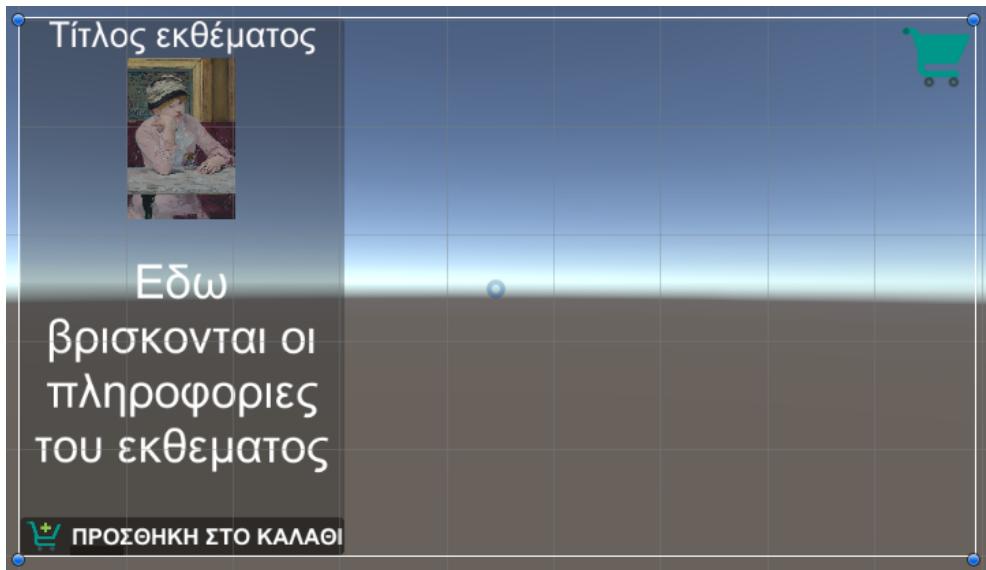
- η ταχύτητα του χρήστη σε συνδυασμό με το κουμπί shift που την αυξάνει περαιτέρω,
- η λειτουργία της κάμερας και η αλλαγή του προσανατολισμού της από το ποντίκι,
- η κίνηση του παίκτη στον χώρο,
- η βαρύτητα κα.

2.1.2 Ανασκόπηση εκθεμάτων

Η σημαντικότερη λειτουργία του Μουσείου είναι να παρουσιάσει τα εκθέματα του με τον καλύτερο δυνατό τρόπο. Για αποφυγή συνωστισμού πληροφορίας έχει φτιαχτεί ένα UI που παρουσιάζει τις πληροφορίες του κάθε εκθέματος ξεχωριστά. Το UI αυτό αποτελεί ένα GameObject τύπου Panel και ενεργοποιείται με την είσοδο του παίκτη στα σημεία που έχουν τοποθετηθεί οι αντίστοιχοι Colliders δηλαδή μπροστά από κάθε έκθεμα. Οι πληροφορίες όλων των εκθεμάτων βρίσκονται στο αρχείο Exhibits.json στον φάκελο Assers/Extra Files/.Ενδεικτικά, οι πληροφορίες για το κάθε έκθεμα είναι της μορφής:

```
{
  "exhibits": [
    {
      "tag": "ROpsarades",
      "title": "Ψαράδες στην Θάλασσα - J. M. W. Turner (1796)",
      "path": "Images/ROpsarades",
      "info": "Οι ψαράδες στην θάλασσα αποτελούν πίνακα του Τζόζεφ Μάλλορντ Ουίλλιαμ Τέρνερ. Ο καλλιτέχνης αφιερώθηκε στο τοπίο και ταξίδεψε σ' όλα τα μήκη και πλάτη της Βρετανίας και της Ευρώπης, ώστε να αποκτήσει εμπειρίες που αποτέλεσαν πηγή έμπνευσης για πλήθος των έργων του.",
      "quantity": 34,
      "price": 23.12
    },
    {
      "tag": "ROautoproswpografia",
      "title": "Αυτοπροσωπογραφία - Goya (1795) ",
      "path": "Images/ROautoproswpografia",
      "info": "Ο συγκεκριμένος πίνακας αποτελεί μία από τις πολλαπλές προσωπογραφίες του φρανθίσκο Γκόγια από την περίοδο 1795 με 1798, που τον έκαναν γνωστό στο ευρύ κοινό. Απεικονίζεται ο ίδιος ο καλλιτέχνης μπροστά στο καβαλέτο του.",
      "quantity": 34,
      "price": 23.12
    }
  ]
}
```

Το UI είναι της μορφής:



To script το οποίο αναλαμβάνει να εμφανίσει της πληροφορίες κάθε εκθέματος είναι το PlayerLookAt.cs που βρίσκεται στον φάκελο Assets/Scripts/. Ξεκινώντας με το Script PlayerLookAt.cs βλέπουμε τον ορισμό της εμφαλευμένης κλάση Excibit που περιέχει σαν attributes τις πληροφορίες κάθε εκθέματος. Η συνάρτηση JSONToDictionary παίρνει όλες τις πληροφορίες από το JSON που είδαμε παραπάνω και το αναθέτει στην δομή δεδομένων excibitDictionary τύπου Dictionary.

```

[Serializable]
public class Exhibit
{
    public string title;
    public string path;
    public string info;
    public int quantity;
    public float price;
}

void JSONToDictionary()
{
    DataSet dataSet =
        JsonConvert.DeserializeObject<DataSet>(jsonFile.text);
    DataTable dataTable = dataSet.Tables["exhibits"];
    exhibitDictionary = new Dictionary<string, Exhibit>();
    foreach (DataRow row in dataTable.Rows)
    {
        Exhibit exhibit = new Exhibit();
        exhibit.title = row["title"].ToString();
        exhibit.path = row["path"].ToString();
        exhibit.info = row["info"].ToString();
        exhibit.quantity = int.Parse(row["quantity"].ToString());
        exhibit.price = float.Parse(row["price"].ToString());
        exhibitDictionary.Add(row["tag"].ToString(), exhibit);
    }
}

```

Στην συνέχεια, σε ότι αφορά τον Collider, αναφερόμαστε στις μενόδους OnTriggerEnter και OnTriggerExit. Στον OnTriggerEnter, δηλαδή μόλις ο χρήστης εισέλθει στον Collider, καλείται η συνάρτηση JSONtoInfoPanel η οποία αναθέτει τις κατάλληλες τιμές στα πεδία του Panel πληροφοριών. Στον OnTriggerExit το panel πληροφοριών εξαφανίζεται ενώ παράλληλα βοηθητικές μεταβλητές όπως οι insideExhibitCollider και currentExhibit τροποποιούνται ανάλογα.

```

private void OnTriggerEnter(Collider other)
{
    if (other.tag != null)
    {
        JSONtoInfoPanel(other.tag);
    }
}

```

```

}

private void OnTriggerExit(Collider other)
{
    if (other.tag != null)
    {
        insideExhibitCollider = false;
        moreinfoPanel.SetActive(false);
        currentExhibit = null;
    }
}

void JSONtoInfoPanel(string exhibit_tag = "Exhibit 1")
{
    if (exhibitDictionary.ContainsKey(exhibit_tag))
    {
        insideExhibitCollider = true;
        moreinfoPanel.SetActive(true);
        infoText.GetComponent<Text>().text =
            exhibitDictionary[exhibit_tag].info;
        infoTitle.GetComponent<Text>().text =
            exhibitDictionary[exhibit_tag].title;
        infoPicture.GetComponent<Image>().sprite =
            Resources.Load<Sprite>(exhibitDictionary[exhibit_tag].path);
        currentExhibit = new Tuple<string, Exhibit>(exhibit_tag,
            exhibitDictionary[exhibit_tag]);
        changeAddToCartBtn();
    }
}

```

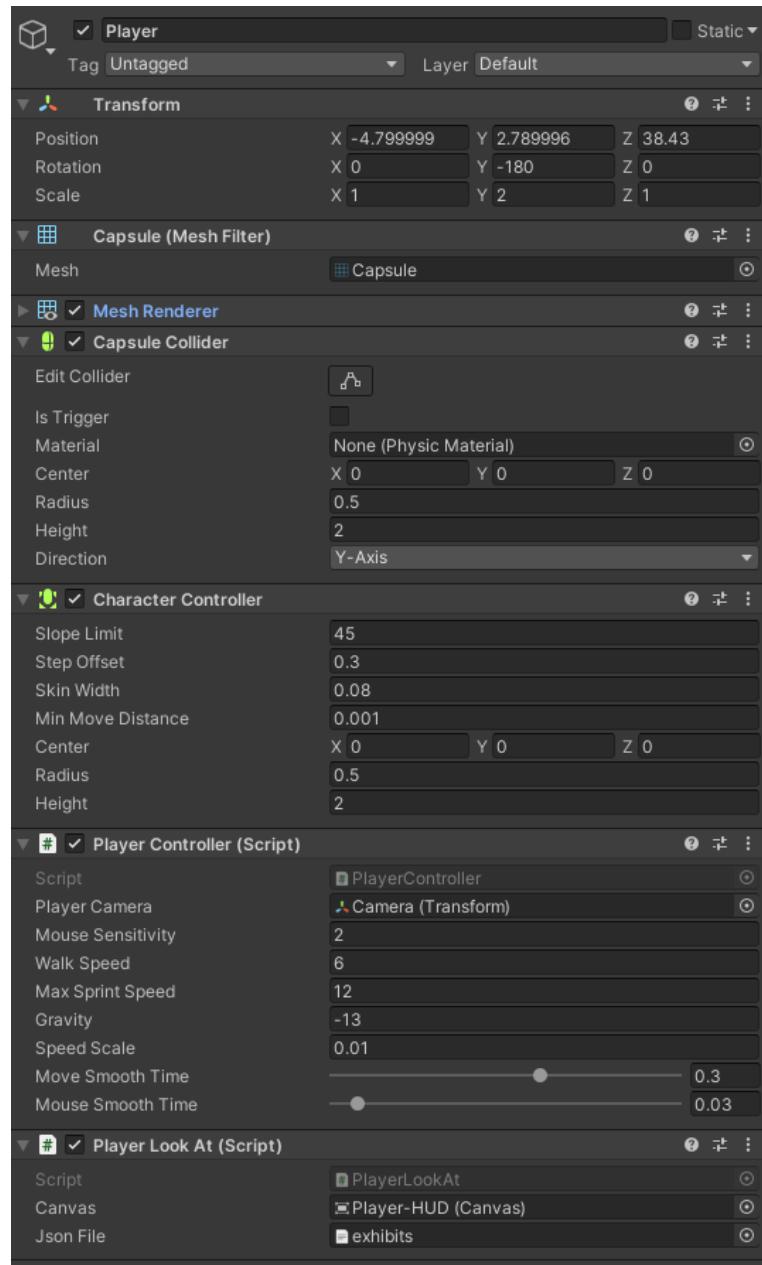
Στην τελευταία συνάρτηση του Script PlayerLookAt.cs στην changeAddToCartBtn ισχύουν τα εξής:

- Ελέγχεται μέσω μεθόδων του Script CartContents που θα δούμε στην συνέχεια αν το καλάθι αγορών είναι άδειο. Σε περίπτωση που ο έλεγχος είναι επιτυχής τότε ανανεώνεται η φωτογραφία του καλαθιού στο πάνω δεξιά μέρος της οθόνης του χρήστη και από άδειο δείχνει γεμάτο. Παράλληλα, ανανεώνεται η συνολική τιμή που αναγράφεται και αυτή πάνω στο καλάθι.
- Στον δεύτερο έλεγχο If μέσω πάλι μίας μεθόδου που έχουμε δανειστεί από

το script CartContents ελέγχεται αν το έκθεμα που παρουσιάζεται έχει ηδή τοποθετηθεί στο καλάθι. Αν αυτό αληθεύει στη ύση του κουμπιού "Προσθήκη στο καλάθι" πλέον εμφανίζεται το μήνυμα "Αφαίρεση από το καλάθι". Είναι σημαντικό να σημειώσουμε ότι ο χρήστης έχει δικαίωμα να προσθέσει στο καλάθι του μόνο 1 αντίγραφο από το κάθε έκθεμα.

```
public static void changeAddToCartBtn()
{
    if (CartContents.getNumItems()>0)
    {
        topCornerCartIcon.GetComponent<Image>().sprite =
            Resources.Load<Sprite>("Icons/items_in_cart");
        topCornerCartPriceText.GetComponent<Text>().text =
            CartContents.getTotalPrice().ToString() + "";
    }
    else
    {
        topCornerCartPriceText.GetComponent<Text>().text = "";
        topCornerCartIcon.GetComponent<Image>().sprite =
            Resources.Load<Sprite>("Icons/empty_cart");
    }
    if(insideExhibitCollider)
    {
        if (CartContents.itemExists(currentExhibit.Item1))
        {
            addToCartbtn.GetComponentInChildren<Text>().text =
                " ";
            addToCartIcon.GetComponent<Image>().sprite =
                Resources.Load<Sprite>("Icons/remove_from_cart");
        }
        else
        {
            addToCartbtn.GetComponentInChildren<Text>().text =
                " ";
            addToCartIcon.GetComponent<Image>().sprite =
                Resources.Load<Sprite>("Icons/add_to_cart");
        }
    }
}
```

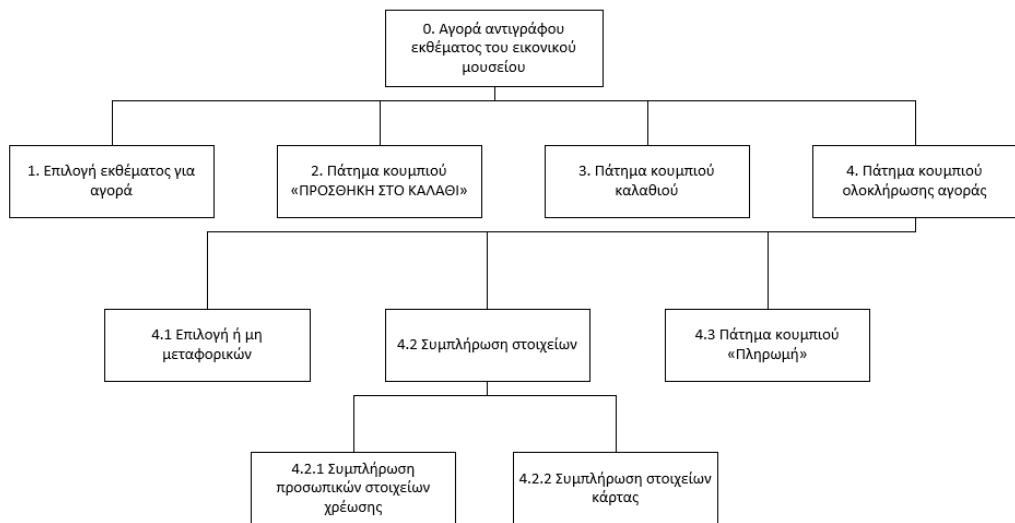
Τελειώνοντας το κεφάλαιο που αφορά τον παίκτη και την προβολή των εκθεμάτων είναι σημαντικό να δούμε το πως όσα scripts αναλύσαμε σε αυτή την ενότητα αποκτούν ρόλο και σημασία στο περιβάλλον του Unity:



2.2 Αγορά εκθεμάτων

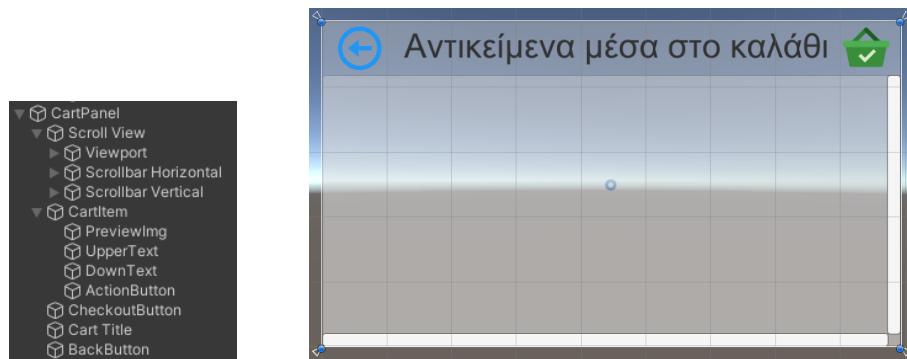
2.2.1 Ιεραρχική ανάλυση εργασιών

Για την λειτουργία «Αγορά αντιγράφου εκθέματος του εικονικού μουσείου» έχει γίνει ιεραρχική ανάλυση εργασιών. Η διαδικασία που πρέπει να ακολουθήσει ο χρήστης προκειμένου να αγοράσει ένα έκθεμα από το Μουσείο φαίνεται στο παρακάτω διάγραμμα:



2.2.2 Καλάθι αγορών

Στον χρήστη προσφέρεται η δυνατότητα να δει τα περιεχόμενα του καλαθιού αγορών του. Πατώντας λοιπόν το καλάθι αγορών (μόνο όταν βρίσκεται μέσα σε κάποιον Collider εκθέματος) εμφανίζεται το GameObject CartPanel που στο Unity έχει την ακόλουθη δομή και εμφάνιση:



Όπως βλέπουμε μέσα στο panel CartPanel περιέχονται το scroll view (στο οποίο εμφανίζονται ένα προς ένα τα εκθέματα που έχουν προστεθεί στο καλάθι , το checkoutButton (που οδηγεί στην πληρωμή με την προϋπόθεση ότι στο καλάθι περιέχεται τουλάχιστον ένα έκθεμα) , ο τίτλος , και άλλο ένα κουμπί που οδηγεί στην προηγούμενη οθόνη, δίνοντας την δυνατότητα ο χρήστης να συνεχίσει την πλοήγηση στο Μουσείο από το σημείο που την σταμάτησε.

Ακόμα, υπάρχει το CartItem που αποτελεί το template που θα χρησιμοποιήσουν τα αντικείμενα προκειμένου να προβληθούν στον χρήστη ως περιεχόμενα:



Παρατηρούμε ότι στον χρήστη παρέχεται εκ νέου η δυνατότητα να αφαιρέσει, εφόσον το επιθυμεί, ένα έκθεμα από το καλάθι αγορών,

Αν τα δούμε όμως λίγο πιο αναλυτικά το script το οποίο αναλαμβάνει να παρουσιάσει στον χρήστη τα περιεχόμενα του καλαθιού είναι το ToCart2.cs που βρίσκεται στον φάκελο Assets/Scripts/PlayerScripts/.

```
public class ToCart2 : MonoBehaviour
{
    public GameObject panel;
    public GameObject scrollView;
    void clearScrollView()
    {
```

```
foreach (Transform child in scrollView.transform)
{
    GameObject.Destroy(child.gameObject);
}
}

public void AddItemsToScrollView()
{
    clearScrollView();
    int i = 0;
    foreach (KeyValuePair<string, PlayerLookAt.Exhibit> item in
        CartContents.getCartContentsDict())
    {
        GameObject newPanel = Instantiate(panel);
        var prevImg =
            newPanel.GetComponent<Transform>().GetChild(0).gameObject;
        var upperText =
            newPanel.GetComponent<Transform>().GetChild(1).gameObject;
        var lowerText =
            newPanel.GetComponent<Transform>().GetChild(2).gameObject;
        var btn =
            newPanel.GetComponent<Transform>().GetChild(3).gameObject;
        btn.name = i.ToString();
        btn.GetComponent<Button>().onClick.AddListener(() =>
            RemoveItemFromCart(btn.name));
        prevImg.GetComponent<Image>().sprite =
            Resources.Load<Sprite>(item.Value.path);
        upperText.GetComponent<Text>().text = item.Value.title;
        lowerText.GetComponent<Text>().text =
            item.Value.price.ToString() + "\u20AC";
        newPanel.transform.SetParent(scrollView.transform);
        i++;
    }
}

private void RemoveItemFromCart(string btnName)
{
    int item = int.Parse(btnName);
    Destroy(scrollView.transform.GetChild(int.Parse(btnName)).gameObject);
    CartContents.removeItem(item);
    PlayerLookAt.changeAddToCartBtn();
}
```

Όπως βλέπουμε η μέθοδος AddItemsToScrollView διατρέχει την λίστα που είναι αποθηκευμένα όλα τα εκθέματα που έχει προσθέσει ο χρήστης στο καλάθι και τα εμφανίζει ένα ένα στο scroll view. Παράλληλα, χρησιμοποιείται για το κουμπί Remove ένας Listener που περιμένει τον χρήστη να πατήσει το Remove button και στην συνέχεια να καλέσει την συνάρτηση RemoveItemFromCart().

Με την σειρά της η συνάρτηση RemoveItemFromCart() 1. αφαιρεί από τα περιεχόμενα του καλαθιού το έκθεμα που πρόκειται να αφαιρεθεί, 2. χρησιμοποιεί την μέθοδο removeItem του script CartContents για να αφαιρέσει το έκθεμα επίσης από το Dictionary και 3. καλεί την συνάρτηση changeAddToCartBtn() του script PlayerLookAt.cs προκειμένου να αλλάξει το "Αφαίρεση από το καλάθι" σε "Προσθήκη στο καλάθι".

Είδαμε ότι το script ToCart2.cs χρησιμοποιεί πολλές μεθόδους του script CartContents. Το script CartContents φαίνεται παρακάτω:

```
public class CartContents : MonoBehaviour
{
    private static Dictionary<string, PlayerLookAt.Exhibit>
        cartItems = new Dictionary<string, PlayerLookAt.Exhibit>();

    public static void addItem(string tag, PlayerLookAt.Exhibit item)
    {
        if (!cartItems.ContainsKey(tag))
        {
            cartItems.Add(tag, item);
        }
    }

    public static void removeItem(string tag)
    {
        if (cartItems.ContainsKey(tag))
        {
            cartItems.Remove(tag);
        }
    }

    public static void removeItem(int index)
    {
        print(cartItems.Keys.ToArray()[index]);
        cartItems.Remove(cartItems.Keys.ToArray()[index]);
    }
}
```

```
public static float getTotalPrice()
{
    float totalPrice = 0;
    foreach (KeyValuePair<string, PlayerLookAt.Exhibit> item in
        cartItems)
    {
        totalPrice += item.Value.price;
    }
    return totalPrice;
}

public static int getNumItems()
{
    return cartItems.Count;
}

public static Dictionary<string,PlayerLookAt.Exhibit>
getCartContentsDict()
{
    return cartItems;
}

public static void clearCart()
{
    cartItems.Clear();
}

public static bool itemExists(string tag)
{
    return cartItems.ContainsKey(tag);
}
```

Ουσιαστικά αποτελεί μία βοηθητική κλάση που κρατά αποθηκευμένα στο Dictionary την λίστα με τα εκθέματα στα οποία ο χρήστης πάτησε στην προγούμενη ουδόνη "Προσθήκη στο καλάθι" και παράλληλα διαχειρίζεται την προβολή των περιεχομένων του καλαθιού.

Το τελευταίο Script που λαμβάνουμε υπόψη στην υλοποίηση του καλαθιού είναι το CartButtons.cs:

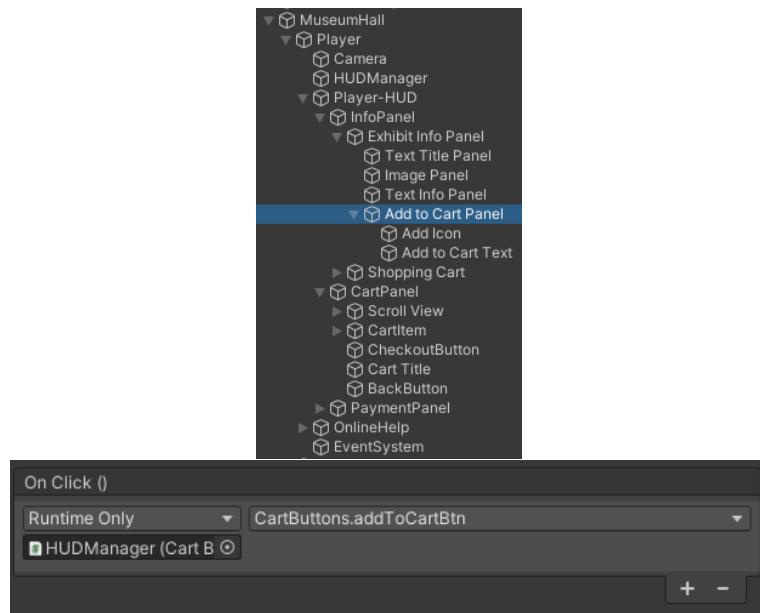
```
public class CartButtons : MonoBehaviour
{
    public void addToCartBtn()
    {
        var e = PlayerLookAt.currentExhibit;
        if(e == null)
            return;

        if (CartContents.itemExists(e.Item1))
        {
            CartContents.removeItem(e.Item1);
        }
        else
            CartContents.addItem(e.Item1,e.Item2);
        Debug.Log("Items in cart: " + CartContents.getNumItems());
        PlayerLookAt.changeAddToCartBtn();
    }

    public void checkoutDone()
    {
        CartContents.clearCart();
        PlayerLookAt.changeAddToCartBtn();
    }
}
```

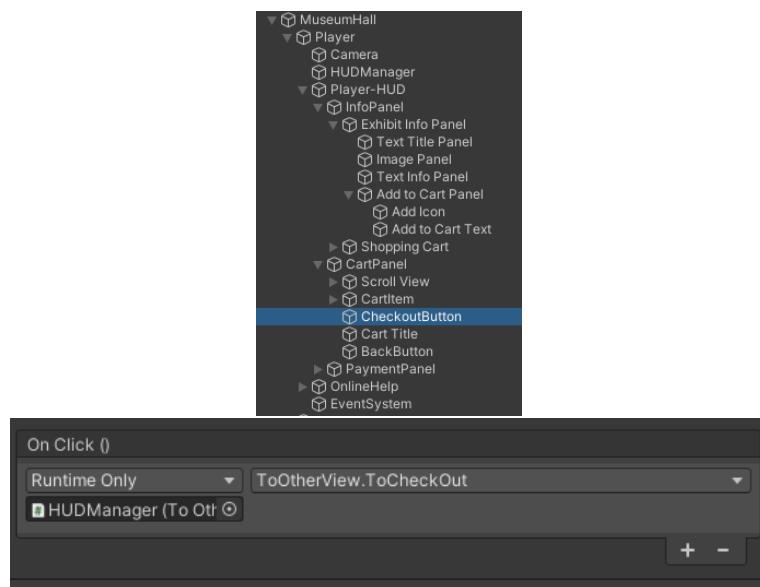
To script αυτό έχει τις εξής εφαρμογές:

- Συνάρτηση addToCartBtn:
Χρησιμοποιείται στο GameObject Add To Cart Panel. Δουλειά του είναι να προσθέτει το currentExcibit στο Dictionary.



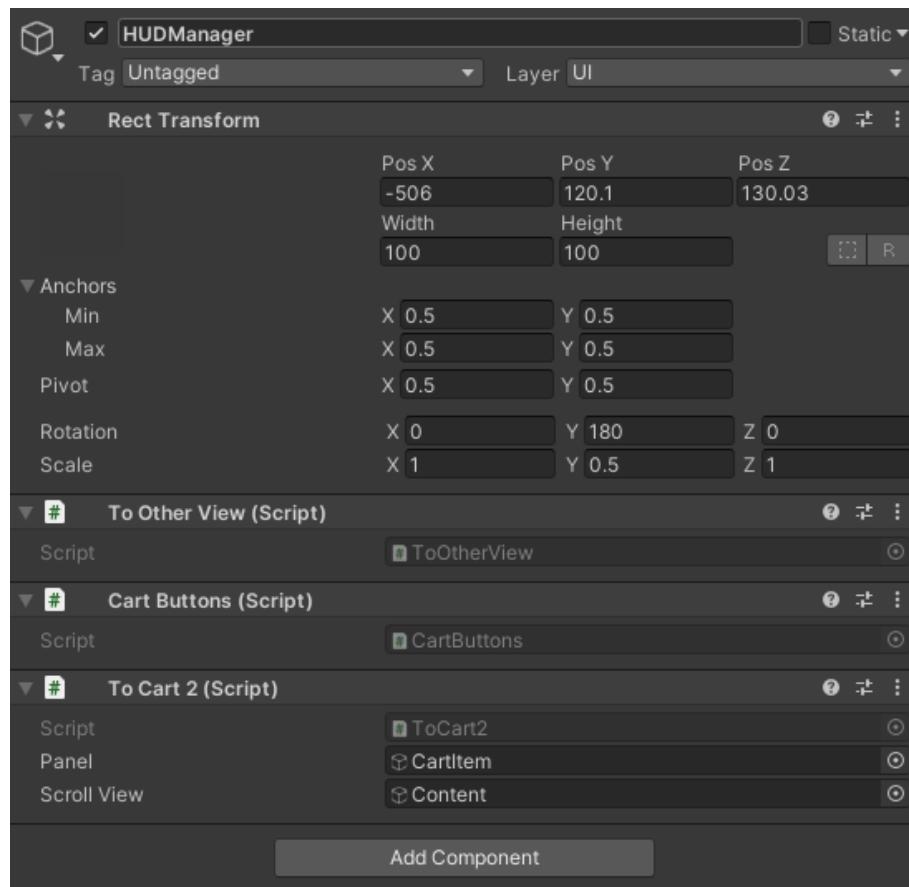
- Συνάρτηση `checkOutDone()`:

Χρησιμοποιείται στο GameObject `CheckoutButton`. Δουλειά του είναι να καθαρίζει τα περιεχόμενα του καλαθιού αγορών μόλις ολοκληρωθεί η αγορά.



Όλα τα script που αναφέρθηκαν σε αυτή την ενότητα καθώς και το script που υπάρχει στην ενότητα "Εναλλαγή Panel" ενσωματώνονται στο

περιβάλλον του Unity χρησιμοποιώντας το GameObject HUDManager. Όπως φαίνεται παρακάτω το αντικείμενο HUDManager χρησιμοποιεί τα scripts και όπου χρειάζεται έχουν συμπληρωθεί τα public πεδία.



2.2.3 Εναλλαγή Panel

Λόγω της ύπαρξης πολλών UI, έχει αναπτυχθεί το Script `ToOtherView.cs` που βρίσκεται στον φάκελο `Assets/Scripts/PlayerScripts/`. Το συγκεκριμένο script βοηθά στην διαδοχή των panel που αφορούν το panel προβολής πληροφοριών για το κάθε έκθεμα, το panel προβολής των περιεχομένων του καλαθιού αγορών και το panel πληρωμής οπού ολοκληρώνεται η παραγγελία.

```
public class ToOtherView : MonoBehaviour
{
```

```

public static void ToInfoPanel()
{
    PlayerLookAt.infoPanel.SetActive(true);
    PlayerLookAt.cartPanel.SetActive(false);
    PlayerLookAt.paymentPanel.SetActive(false);
    PlayerController.move = true;
}

public static void ToCheckOut()
{
    if (CartContents.getNumItems() == 0)
    {
    }
    else
    {
        PlayerLookAt.infoPanel.SetActive(false);
        PlayerLookAt.cartPanel.SetActive(false);
        PlayerLookAt.paymentPanel.SetActive(true);
        PlayerController.move = false;
    }
}

public static void ToCart()
{
    PlayerLookAt.infoPanel.SetActive(false);
    PlayerLookAt.cartPanel.SetActive(true);
    PlayerLookAt.paymentPanel.SetActive(false);
    PlayerController.move = false;
}

```

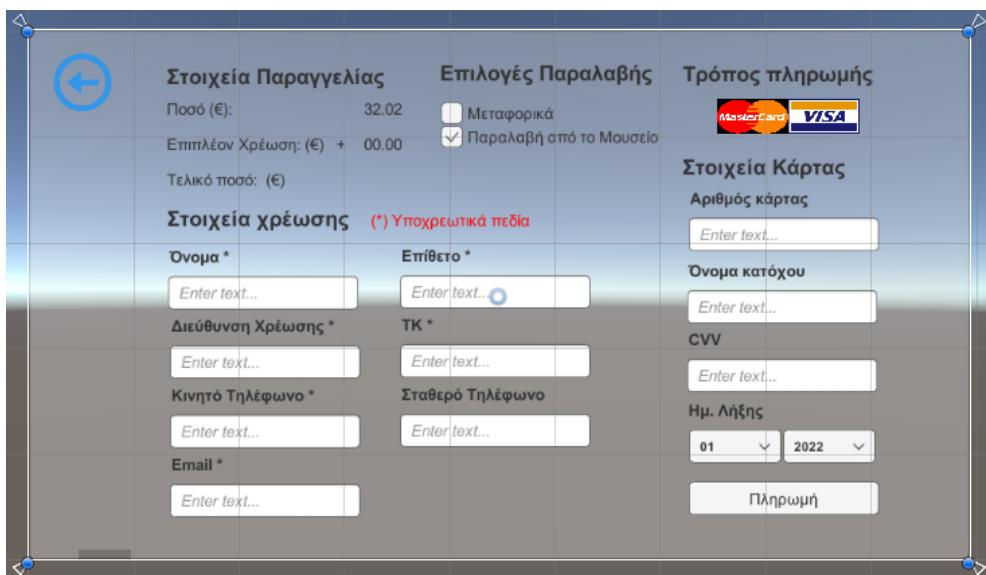
Συγκεκριμένα, η συνάρτηση ToInfoPanel καλείται πατώντας το κουμπί Back στο panel προβολής των περιεχομένων του καλαθιού αγορών και οδηγεί τον χρήστη στο σημείο όπου είχε σταματήσει πριν την περιήγηση του στο Μουσείο.

Η συναρτηση ToCheckOut καλείται καλείται πατώντας το κουμπί CheckOut στο panel προβολής των περιεχομένων του καλαθιού αγορών και οδηγεί τον χρήστη στο panel πληρωμής με σκοπό να ολοκληρώσει την παραγγελία του. Η πληρωμή πραγματοποιείται μόνο αν υπάρχει τουλάχιστον ένα έκθεμα στο καλάθι αγορών.

Η συνάρτηση ToCart καλείται πατώντας το κουμπί με εικόνα το καλάθι αγορών και οδηγεί στο panel προβολής των περιεχομένων του καλαθιού αγορών. Ο χρήστης μπορεί να το χρησιμοποιήσει όταν επιθυμεί να δει το καλάθι αγορών του και από εκεί και έπειτα να σκεφτεί αν θέλει να το τροποποιήσει ή αν θέλει να προχωρήσει απευθείας στην ολοκλήρωση της παραγγελίας.

2.2.4 Ολοκλήρωση αγοράς

Ο χρήστης μπορεί να φτάσει στο Panel πληρωμής αφού περάσει από το καλάθι αγορών, μέσω του CheckOutButton. Αμέσως εμφανίζεται η παρακάτω οθόνη:



To panel έχει υλοποιηθεί χρησιμοποιώντας GameObjects όπως Text, InputFields, Toggles και φυσικά το Button επιβεβαίωσης πληρωμής. Το script που διαχειρίζεται το Credit Card System είναι το CreditScript.cs που βρίσκεται στον φάκελο Assets/Scripts/CreditCardScript/.

```
public class CreditScript : MonoBehaviour
{
    public GameObject Panel;
    public Text AmountText, TotalAmountText, ExtraAmount;
```

```
public InputField cardnumberText,cardholderText,cvvText;
public InputField
    nameText,surnameText,addressText,zipcodeText,mobileText,phoneText,emailText;
public Toggle option1, option2;
public Text
    panel_name,panel_email,panel_phone,panel_ordercode,panel_ordertime;
public Text notification;
double amount,totalamount;
private static string cardnumber,cardholder,cvv;
private static string
    name,surname,address,zipcode,mobile,phone,email;
private static string ordercode,ordertime;
private static bool toggle1, toggle2;
private static double shippingextra = 30.00;

void Start()
{
    Panel.SetActive(false);
    float temp = CartContents.getTotalPrice();
    AmountText.GetComponent<Text>().text = temp.ToString("F");
    TotalAmountText.GetComponent<Text>().text = AmountText.text;
}

void Update()
{
    float temp = CartContents.getTotalPrice();
    AmountText.GetComponent<Text>().text = temp.ToString("F");
    TotalAmountText.GetComponent<Text>().text = AmountText.text;
    if (option1.isOn)
    {
        String amount_str = AmountText.text;
        amount = Double.Parse(amount_str);
        amount = amount + shippingextra;
        TotalAmountText.GetComponent<Text>().text =
            amount.ToString("F");
        ExtraAmount.GetComponent<Text>().text =
            shippingextra.ToString("F");
    }
    else
    {
        String zero = "00.00";
        TotalAmountText.GetComponent<Text>().text =
            AmountText.text;
    }
}
```

```
        ExtraAmount.GetComponent<Text>().text = zero;
    }
}

public void FinishOrder()
{
    bool OrderOk=true;
    initialiseObjects();
    if (string.IsNullOrEmpty(name) ||
        string.IsNullOrEmpty(surname) ||
        string.IsNullOrEmpty(address) ||
        string.IsNullOrEmpty(zipcode) ||
        string.IsNullOrEmpty(mobile) ||
        string.IsNullOrEmpty(email))
    {
        OrderOk = false;
        notification.text = " ";
    }
    if(string.IsNullOrEmpty(cardnumber) ||
       string.IsNullOrEmpty(cardholder) ||
       string.IsNullOrEmpty(cvv))
    {
        OrderOk = false;
        notification.text = " ";
    }
}

if(Panel != null && OrderOk)
{
    notification.text = "";
    panel_name.GetComponent<Text>().text = name + " " +
        surname;
    panel_email.GetComponent<Text>().text = email;
    panel_phone.GetComponent<Text>().text = phone;
    int rand_num = UnityEngine.Random.Range(100, 200);
    ordercode = "#" + rand_num.ToString();
    panel_ordercode.GetComponent<Text>().text = ordercode;
    DateTime now = DateTime.Now;
    panel_ordertime.GetComponent<Text>().text =
        now.ToString("F");
    Panel.SetActive(true);
}
```

```

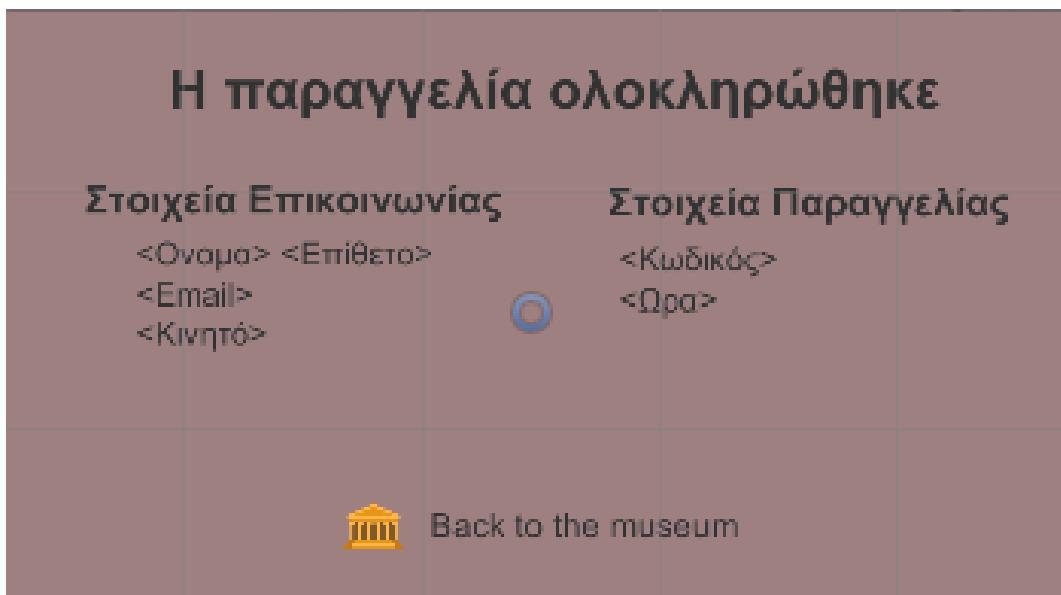
    }
}

```

Δινεται στον χρήστη η δυνατότητα να επιλέξει τον τρόπο παραλαβής μεταξύ παραλαβής από το μουσείο και μεταφορικών. Σε κάθε περίπτωση μπορεί να ενημερωθεί για το τελικό κόστος (με ή χωρίς μεταφορικά) και καλείται να συμπληρώσει τα στοιχεία του συμπεριλαμβανομένων των στοιχείων χρέωσης και των στοιχείων της πιστωτικης του κάρτας. Η παραγγελία δεν δύναται να ολοκληρωθεί αν δεν έχουν συμληρωθεί όλα τα απαιτούμενα πεδία από τον χρήστη.

Με την ολοκλήρωση της παραγγελίας του χρήστη, εμφανίζεται ένα νέο Panel επιβεβαίωσης ολοκλήρωσης της παραγγελίας. Η ολοκλήρωση της παραγγελίας πραγματοποιείται με το πάτημα του Button SubmitButton και την κλήση της συνάρτησης FinishOrder του script CreditScript που έχει ανατεθεί στο Button Submit, με χρήση της συνάρτησης onClick().

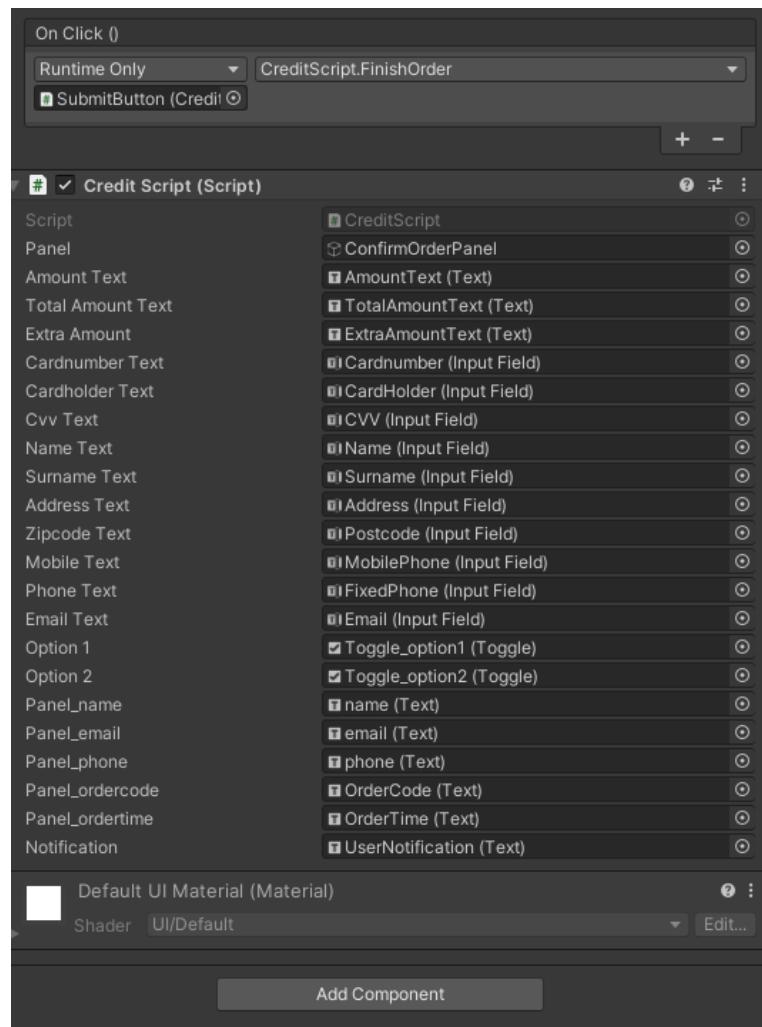
Το επιβεβαιωτικό Panel έχει την παρακάτω μορφή:



Στο Panel περιλαμβάνονται τα στοιχεία επικοινωνίας που υπέβαλε ο χρήστης ενώ το σύστημα πληρωμής έχει παράξει έναν μοναδικό κωδικό παραγγελίας τον οποίο επιδεικνύει μαζί με την ώρα επιβεβαίωσης της παραγγελίας.

Με το κουμπί "Back to the Museum" ο χρήστης μπορεί να επιστρέψει πίσω στο Μουσείο.

Ακολουθεί ο τρόπος με τον οποίο συνδεόμετε το script CreditScript με το περιβάλλον του Unity:



To script έχει ανατεθεί στο κουμπί SubmitButton του Panel πληρωμής.

2.3 Εικονικός πράκτορας

Ο εικονικός πράκτορας παρέχει πληροφορίες στον χρήστη σχετικά με το Μουσείο. Το μοντέλο αλληλεπίδρασης είναι της μορφής ερώτηση-απάντηση με τις ερωτήσεις που μπορεί να θέσει ο χρήστης να είναι προκαθορισμένες. Ο χρήστης απευθύνει την ερώτηση πατώντας το αντίστοιχο κουμπί.

Τα scripts που αναλύονται σε αυτή την ενότητα βρίσκονται στον φάκελο Assets/Scripts/AgentScripts/.

Για τον εικονικό πράκτορα συνδυάζονται τα scripts:

- **Dialogue.cs**

Στο script Dialogue.cs βρίσκεται ο ορισμός του αντικειμένου Dialogue. Το αντικείμενο Dialogue αποτελεί βοηθητική χλάση του Μουσείου και χρησιμοποιείται κυρίως στο script DialogueTrigger αλλά και στο DialogueManager.

```
[System.Serializable]
public class Dialogue
{
    [TextArea(3, 10)]
    public string[] sentences;
}
```

Ο πίνακας sentences φιλοξενεί τις προτάσεις-απαντήσεις του Agent προς την ερώτηση του χρήστη.

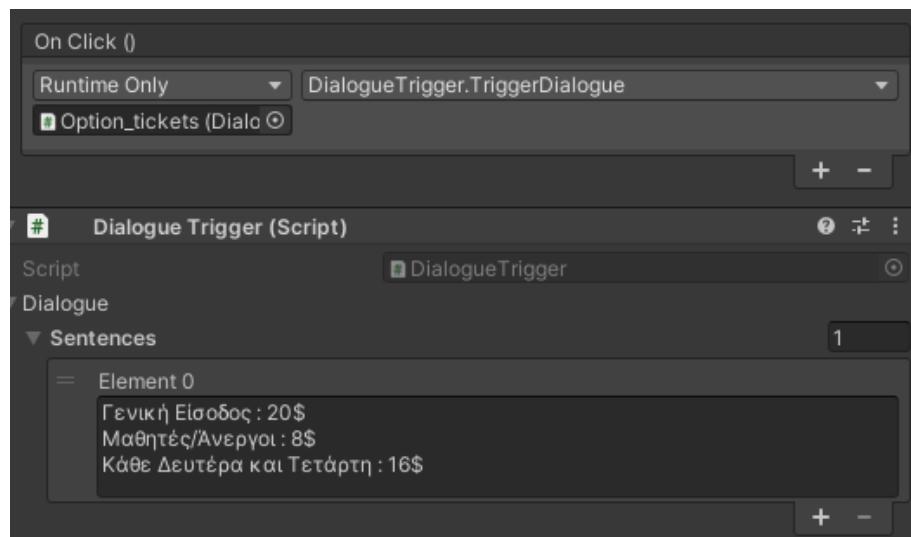
- **DialogueTrigger.cs**

Το script DialogueTrigger λαμβάνει σαν είσοδο το αντικείμενο Dialogue του script Dialogue.cs. Περιέχει μια συνάρτηση την TriggerDialogue η οποία λειτουργεί σαν μεσάζοντας και με αυτό τον τρόπο καλείται η συνάρτηση StartDialogue του script DialogueManager με όρισμα το στιγμιότυπο dialogue τύπου Dialogue (public μέλος).

```
public class DialogueTrigger : MonoBehaviour
{
    public Dialogue dialogue;
```

```
public void TriggerDialogue ()
{
    FindObjectOfType<DialogueManager>().StartDialogue(dialogue);
}
```

Το αντικείμενο AgentTrigger ενσωματώνεται σε κάθε κουμπί που φέρει μια επιλογή του χρήστη. Έτσι είμαστε σε θέση για κάθε επιλογή να συνδέσουμε ένα string το οποίο θα απαντάει στον χρήστη με το πάτημα του αντίστοιχου κουμπιού. Για παράδειγμα το κουμπί "Εισιτήρια" έχει ρυθμιστεί ως εξής:



Όπως βλέπουμε, για το button με την επιλογή "Εισιτήρια" υπάρχει η συνάρτηση onClick() η οποία, με το πάτημα του κουμπιού καλεί την μέθοδο TriggerDialogue του script DialogueTrigger. Το script DialogueTrigger έχει ανατεθεί στο συγκεκριμένο κουμπί και χρειάζεται (public attribute) ένα αντικείμενο Dialogue. Το αντικείμενο Dialogue έχει ως attribute έναν πίνακα με sentences. Το πεδίο sentence[0] συμπληρώνεται απευθείας μέσω του Unity Editor.

- **DialogueManager.cs**

Το script DialogueManager είναι υπέυθυνο για την εμφάνιση της απάντησης στον χρήστη. Η απάντηση δίνεται με την ανάθεση του string sentence[0] στο πεδίο messageText.text που είναι public μέλος του script.

```

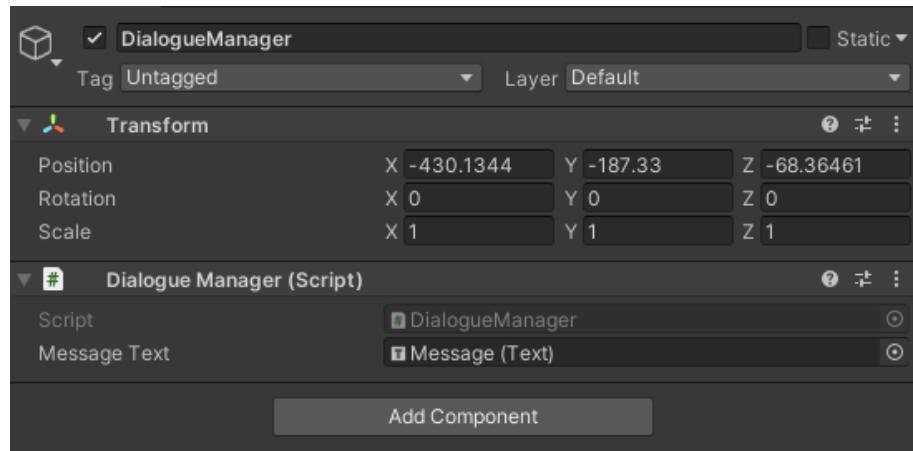
public class DialogueManager : MonoBehaviour
{
    public Text messageText;

    public void StartDialogue (Dialogue dialogue)
    {
        StopAllCoroutines();
        StartCoroutine(TypeSentence(dialogue.sentences[0]));
    }

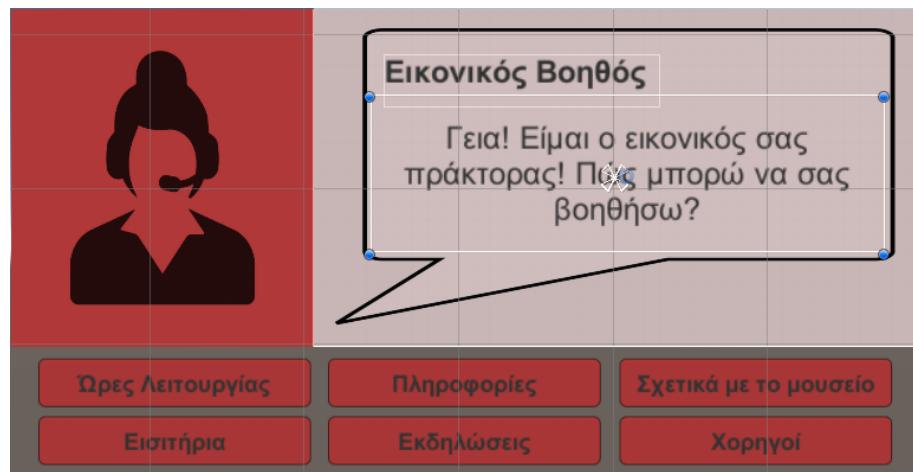
    IEnumerator TypeSentence (string sentence)
    {
        messageText.text = "";
        foreach (char letter in sentence.ToCharArray())
        {
            messageText.text += letter;
            yield return new WaitForSeconds(0.02F);
        }
    }
}

```

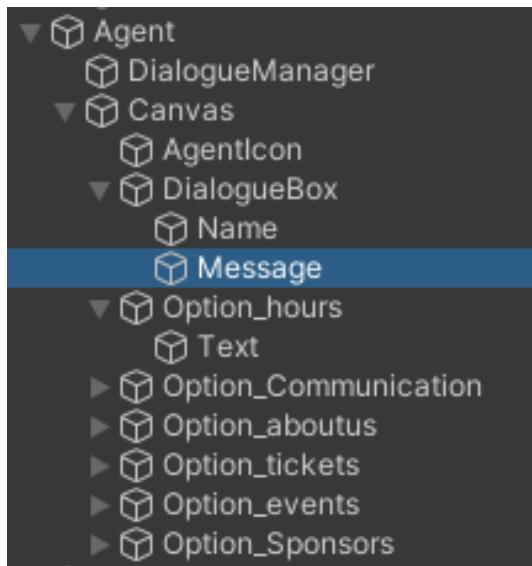
To Object messageText τύπου Text έχει συνδεθεί με το αντικείμενο παιδί του DialogueBox της σκηνής του Agent, και συγκεκριμένα με το Object Message.



Στην εικόνα που ακολουθεί φαίνεται ο χώρος που καταλαμβάνει το Message στην σκηνή του Agent.



Στην πρώτη εμφάνιση του Agent, το πεδίο Text είναι αρχικοποιημένο με το μήνυμα καλωσορίσματος.



2.4 Προβολή ταινίας

Προχωρώντας στο βάθος της αίθουσας, στην τελευταία αίθουσα του Μουσείου υπάρχει το Cinema.



Στην αίθυσα αυτή ο χρήστης μπορεί να παρακολουθήσει ταινία, η οποία εκκινεί μέσω Collider. Η συμπεριφορά του Collider περιγράφεται από το Script VideoTrigger.cs που βρίσκεται στον φάκελο Assets/Scripts/CinemaScripts/.

```
public class VideoTrigger : MonoBehaviour
{
    [SerializeField] Canvas _canvas;
    private GameObject infoPanel;
    public VideoPlayer video;

    void CanvasToPanel(string exhibit_tag = "cinema")
    {
        infoPanel.SetActive(true);
    }

    void InfoPanelObjects()
    {
        infoPanel = _canvas.GetComponent<Transform>().gameObject;
    }

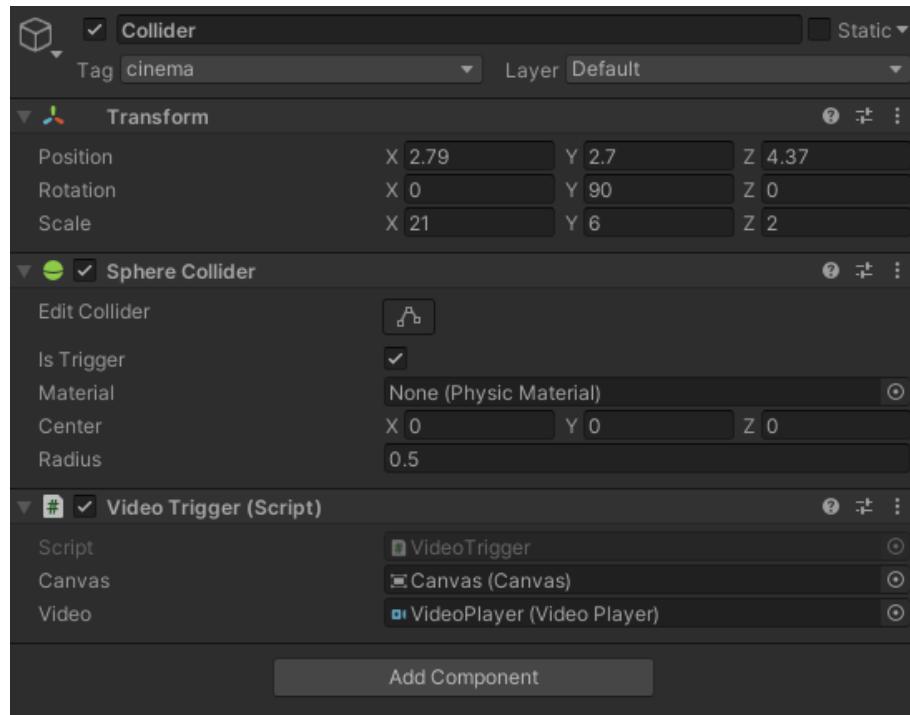
    void Start()
    {
        InfoPanelObjects();
        infoPanel.SetActive(false);
    }
}
```

```
private void OnTriggerEnter(Collider other)
{
    if (other.tag != null)
    {
        CanvasToPanel(other.tag);
        video.Play();
    }
}

private void OnTriggerExit(Collider other)
{
    if (other.tag != null)
    {
        infoPanel.SetActive(false);
        video.Pause();
    }
}

void Update()
{
}
}
```

To Script VideoTrigger ενσωματώνεται στο GameObject Collider ως εξής:



Πιο συγκεκριμένα, σε ότι αφορά τους Collider αναφερόμαστε στις μεθόδους OnTriggerEnter και OnTriggerExit. Στον OnTriggerEnter, δηλαδή μόλις ο χρήστης εισέλθει στον Collider, ο Canvas εμφανίζεται και το Video ξεκινά να παίζει. Στον OnTriggerExit ο Canvas εξαφανίζεται και το Video παγώνει. Ο Canvas αποτελεί το GameObject που περιέχει τα εξής:



To script που ελέγχει την λειτουργία των κουμπιών του Canvas είναι το PlayPauseVideo.cs.

```
public class PlayPauseVideo : MonoBehaviour
{
    private VideoPlayer video;

    private void Awake()
    {
        video = GetComponent<VideoPlayer>();
    }
}
```

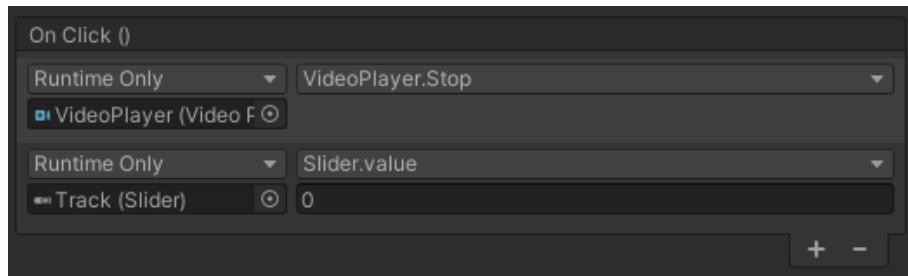
```

public void PlayVideo()
{
    video.Play();
}

public void PauseVideo()
{
    video.Pause();
}
}

```

Συγκεκριμένα, η συνάρτηση Play καλείται όταν ο χρήστης πατήσει το αντίστοιχο κουμπί. Αντίστοιχα, όταν ο χρήστης πατήσει το pause, καλείται η συνάρτηση Pause. Το τρίτο κουμπί (stop) εκτελεί την συνάρτηση Stop (της κλάσης VideoPlayer) μέσω της συνάρτησης onClick:



Ο Canvas εκτός των κουμπιών play pause περιλαμβάνει επίσης την μπάρα αναπαραγωγής (track). Έχει δοθεί η δυνατότητα ο χρήστης να μπορεί να αλλάξει το σημείο αναπαραγωγής του Video. Η δυνατότητα αυτή υπάρχει λόγω του script track.cs που βρίσκεται στον ίδιο φάκελο με όσα scripts αναφέρονται στην λειτουργικότητα του Cinema.

```

public class track : MonoBehaviour, IPointerDownHandler,
    IPointerUpHandler
{
    public AudioSource audio;
    public Slider audioVolume;
    public VideoPlayer video;
    Slider tracking;
    bool slide = false;
    private float musicVolume = 1f;

```

```

void Start()
{
    tracking = GetComponent<Slider>();
}

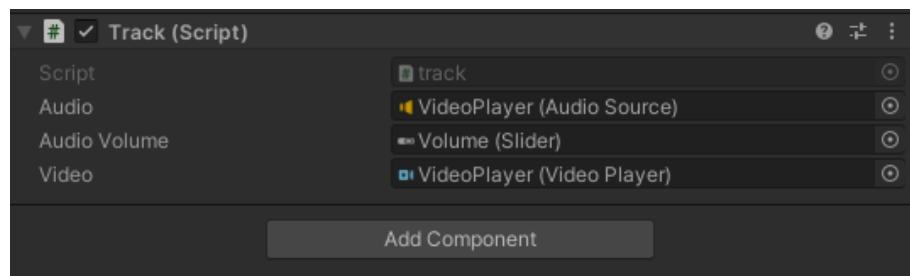
public void OnPointerUp(PointerEventData a)
{
    float frame = (float) tracking.value * (float)
        video.frameCount;
    video.frame = (long)frame;
    slide = false;
}

public void OnPointerDown(PointerEventData a)
{
    slide = true;
}

void Update()
{
    if(!slide && video.isPlaying)
    {
        tracking.value = (float)video.frame / (float)
            video.frameCount;
    }
}
}

```

Ακολουθεί ο τρόπος σύνδεσης Script και αντικειμένων του Unity για το GameObject track που χρησιμοποιεί το script track.cs:



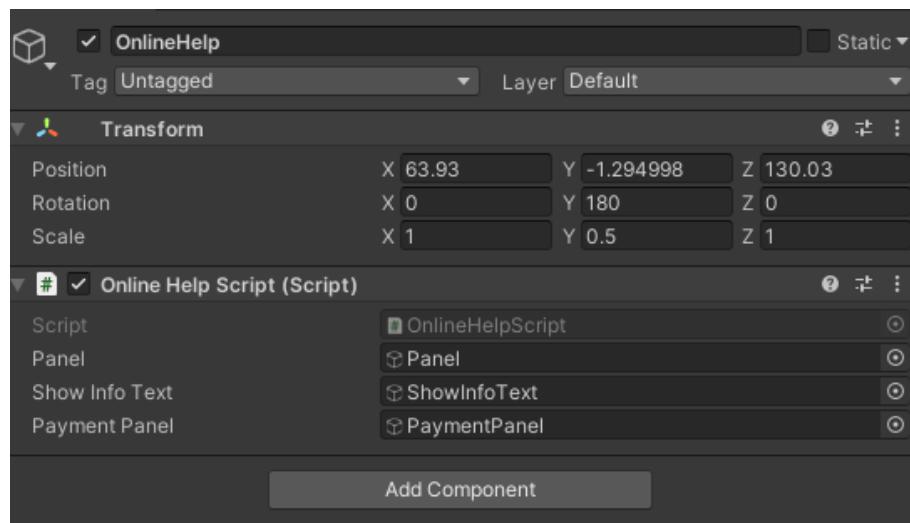
2.5 Online Help

Το OnlineHelp είναι διαθέσιμο στον χρήστη κάθε στιγμή που περιεγείται στους χώρους του μουσείου εκτός από όταν βρίσκεται στο panel πληρωμής. Ο χρήστης μπορεί να ανοίξει το Online Help πατώντας στο πληκτρολόγιο του το κουμπί 'i' και να πραγματοποιεί ταυτόχρονα όποια ενέργεια θέλει στους χώρους του μουσείου. Όταν πια δεν το χρειάζεται άλλο μπορεί να πατήσει ξανα το 'i' και το Online Help θα εξαφανιστεί.

Για την υλοποίηση του Online Help έχουν επιλεχθεί 3 panel που περιέχουν Text Objects με τις πληροφορίες που θέλουμε να εμφανιστούν. Το ένα panel διαδέχεται το άλλο με χρήση του κουμπιού 'e'.



Μέλη του GameObject OnlineHelp αποτελούν:



Όπως βλέπουμε όλα τα Objects βρίσκονται μέσα σε ένα Canvas. Το πεδίο ShowInfoText χρησιμεύει στην εμφάνιση της προτροπής:

Για την λήψη βοήθειας πατήστε '?'

Με το πάτημα του ή όπως είπαμε ανοίγει το πρώτο Panel. Ταυτόχρονα στην θέση του ShowInfoText εμφανίζεται η προτροπή:

Για έξοδο από τον οδηγό πατήστε '?'

Μέσα στο Object Panel του OnlineHelp περιλαμβάνονται τα Text Title (για την εμφάνιση του τίτλου), PagesText (για την αρίθμηση των σελίδων) και NextText (για την πληροφορία ότι με πάτημα του ε αλλάζει η σελίδα).

Όσον αφορά τα Panels οι πληροφορίες είναι καρφωτές, με χρήση του Unity.

Το μοναδικό Script που σχετίζεται με το Online Help είναι το OnlineHelpScript που βρίσκεται στον φάκελο Assets/Scripts/:

```
public class OnlineHelpScript : MonoBehaviour
{
    public GameObject panel;
    public GameObject showInfoText;
    private GameObject panel1;
    private GameObject panel2;
    private GameObject panel3;
    private GameObject pages;
    private GameObject Next;

    void Start()
    {
        panel.SetActive(false);
        panel1 =
            panel.GetComponent<Transform>().GetChild(1).gameObject;
        panel2 =
            panel.GetComponent<Transform>().GetChild(2).gameObject;
        panel3 =
            panel.GetComponent<Transform>().GetChild(3).gameObject;
        pages =
            panel.GetComponent<Transform>().GetChild(4).gameObject;
        Next =
            panel.GetComponent<Transform>().GetChild(5).gameObject;
        panel1.SetActive(false);
        panel2.SetActive(false);
        panel3.SetActive(false);
    }

    void Update()
    {
        if (!paymentPanel.activeSelf)
        {
            if (Input.GetKeyDown("i"))
            {
                if (panel.activeSelf)
                {
                    panel.SetActive(false);
                    panel1.SetActive(false);
                    panel2.SetActive(false);
                    panel3.SetActive(false);
                    showInfoText.GetComponent<Text>().text = "
```

Για την λειτουργία του Script είναι απαραίτητο να κάνουμε Drop τα Objects του Unity στα πεδία του Script. Η σύνδεση γίνεται αναθέτοντας το script OnlineHelpScript.cs στο GameObject OnlineHelp:

