Τεχνικό εγχειρίδιο εικονικού βοηθού

Σωτήρης Αναγνωστόπουλος (19390012) Ρεντιόν Κάπο(19390075)
Βασίλης Κούμανης (19390104) Παναγιώτης Φίσκιλης(19390243)

Περιεχόμενα

1 Εισαγωγή **3**

2 Γενική Περιγραφή **3**
2.1 Αναλυτική Παρουσίαση Λειτουργιών 3
2.2 Σχεδιασμός και Υλοποίηση Γραφικών 3

3 Σκηνή 1 **3**

4 Σκηνή 2 **5**
4.1 Καλωσόρισμα του χρήστη και πρώτη επικοινωνία με τον βοηθό 5
4.2 Βασικό μενού ελέγχου 7
4.2.1 Ηλεκτρονικό Ταχυδρομείο (μέσω smtp και gmail) 8
4.2.2 Καιρός 9
4.2.3 Ημερολόγιο και Ραντεβού 11
4.2.4 Τηλέφωνο, Επαφές και Μηνύματα 13
α΄ 13
β΄ 13
γ΄ 14
4.2.5 Τηλέφωνο, Επαφές και Μηνύματαe-Shop με λίστα αγορών 14
4.2.6 Music Player 15
4.2.7 BBC News 16

5 Γενικές Παρατηρήσεις **17**
5.1 Συναρτήσεις Ελέγχου 17
5.2 Γραμματοσειρές 18

6 Online Help **18**

7 Σχεδιαγράμματα **19**

1 Εισαγωγή

Το Unity είναι μια πολυδιάστατη πλατφόρμα ανάπτυξης λογισμικού που χρησιμοποιείται κυρίως για τη δημιουργία παιχνιδιών και διαδραστικού περιεχομένου σε διάφορες πλατφόρμες. Με χρήση γλωσσών προγραμματισμού όπως το C# και το UnityScript, οι προγραμματιστές μπορούν να δημιουργήσουν παιχνίδια με υψηλή ποιότητα και απλότητα. Παρέχεται επίσης ένα ευέλικτο περιβάλλον ανάπτυξης, με εργαλεία που επιτρέπουν τη δημιουργία γραφικών, ήχων, φυσικών χαρακτηριστικών και άλλων στοιχείων που απαιτούνται για τα παιχνίδια.

Εκτός από τα παιχνίδια, χρησιμοποιείται όμως και σε άλλους τομείς, όπως η εκπαίδευση, η αρχιτεκτονική και η ιατρική για τη δημιουργία εκπαιδευτικών παιχνιδιών, εικονικών περιηγήσεων και εκπαιδευτικών προσομοιώσεων.

2 Γενική Περιγραφή

Στα πλαίσια της εργασίας αναπτύχθηκε ένας εικονικός προσωπικός βοηθός μέσω Unity (2021.3.17f1 - Visual Studio Code) ο οποίος αλληλπιδρά με τον χρήστη και του επιτρέπει να εκτελέσει διάφορες λειτουργίες όπως ηλεκτρονικό ταχυδομείο, ειδήσεις, καιρός κα.

Βρείτε τον κώδικα στο ρεπο μας στο Github: https://github.com/Zeref64/UnityProject/tree/master.

2.1 Αναλυτική Παρουσίαση Λειτουργιών

2.2 Σχεδιασμός και Υλοποίηση Γραφικών

Για την υλοποίηση των γραφικών της εφαρμοφής έπρεπε καταρχάς να φτιαχτούν sprites.

Για τον λόγο αυτό κατεβάσαμε αντίστοιχες εικόνες, τις περικόψαμε και τις τοποθετήσαμε στα σωστά αντικείμενα και τη σωστή θέση τους στον καμβά.

Τελικά, καταλήγουμε με ένα ενιαίο background αποτελούμενο από (8*8) 64 boxes, με το κάθε box να είναι φτιαγμένο από ένα περικομμένο sprite.

Για την κίνηση του παικτη στην σκηνή 1 φτιάξαμε επιπλέον και animations χρησιμοποιώντας τα παραπάνω sprites.

3 Σκηνή 1

Σχήμα 1: Στιγμιότυπο σκηνής 1

Η πρώτη σκηνή παρουσιάζει τον εξωτερικό χώρο και περιέχει τον παίκτη ο οποίος μπορεί να κινηθεί ελεύθερα.

Αυτό επιτυγχάνεται μέσω του script 'CharachterController2D' το οποίο μετακινεί τον παίκτη σύμφωνα με τα input (w,s,a,d) του χρήστη.

Listing 1: CharachterController2D.cs - Update function

```
    private void Update()
    {
        // Horizontal movement
        float moveX = Input.GetAxis("Horizontal");
        rb.velocity = new Vector2(moveX * moveSpeed, rb.velocity.y);

        // Flip character
        if (moveX < 0 && isFacingRight)
        {
            Flip();
        }
        else if (moveX > 0 && !isFacingRight)
        {
            Flip();
        }

        // Vertical movement
        float moveY = Input.GetAxis("Vertical");
        rb.velocity = new Vector2(rb.velocity.x, moveY * moveSpeed);
    }
```

Όταν ο χρήστης είναι έτοιμος μπορεί να περπατήσει προς το σπίτι και να μπεί από την πόρτα.

Αυτό επιτυγχάνεται μέσω collider στην πόρτα του σπιτιού και τον παίκτη, τα οποία (με τα παρακάτω script) αλλάζουν σκητή και ο χρήστης μεταβαίνει στην Σκηνή 2.

Listing 2: From_1TO2_Scene.cs

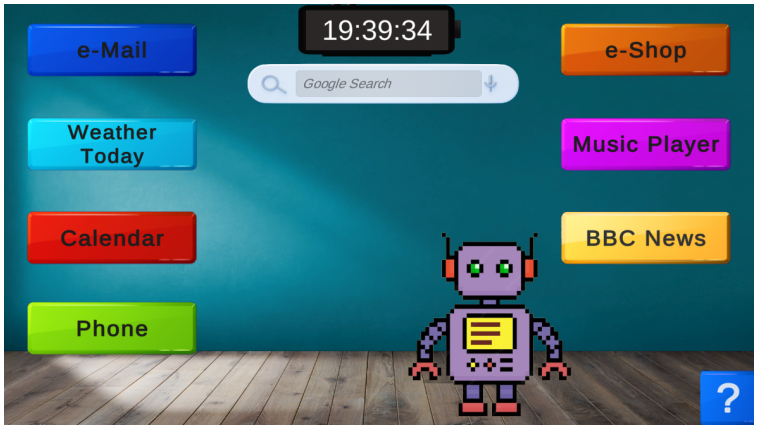
```
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("Player"))
        {
            ChangeScene.Instance.LoadSceneByName("Cabin");
        }
    }
```

Listing 3: ChangeScene.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class ChangeScene : MonoBehaviour
{
    private static ChangeScene instance;

    public static ChangeScene Instance
    {
        get { return instance; }
    }

    private void Awake()
    {
        if (instance != null && instance != this)
        {
            Destroy(this.gameObject);
```

```
        }
        else
        {
            instance = this;
            DontDestroyOnLoad(this.gameObject);
        }
    }

    public void LoadSceneByName(string sceneName)
    {
        SceneManager.LoadScene(sceneName);
    }
}
```

4 Σκηνή 2



Σχήμα 2: Βασικό Μενού

Αυτή είναι η βασική σκηνή της εφαρμογής, μέσω της οποίας αλληλεπιδρούμε με τον βοηθό. Περιέχει 2 καμβάδες:

1. Καλωσόρισμα του χρήστη και πρώτη επικοινωνία με τον βοηθό.
2. Βασικό μενού ελέγχου.

4.1 Καλωσόρισμα του χρήστη και πρώτη επικοινωνία με τον βοηθό

Στην σκηνή αυτή ο εικονικός προσωπικός βοηθός καλωσορίζει τον χρήστη μέσω ενός διαλόγου σε ένα μπαλονάκι το οποίο παρουσιάζεται με typewriter effect.

Αυτό ελέγχεται μέσω των script:

Listing 4: DialogueObject.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[CreateAssetMenu(menuName = "Dialogue/DialogueObject")]
public class DialogueObject : ScriptableObject
```

```
{
    [SerializeField] [TextArea] private string[]  dialogue;

    public string[] Dialogue => dialogue;
}
```

Listing 5: DialogueUI.cs

```
using UnityEngine;
using TMPro;

public class DialogueUI : MonoBehaviour
{
    [SerializeField] private GameObject canvas;
    [SerializeField] private int openObject;
    [SerializeField] private GameObject dialogueBox;
    [SerializeField] private TMP_Text textLabel;
    [TextArea(3, 10)] [SerializeField] private string dialogueText;

    private Typewriter typewriter;

    private void Start()
    {
        typewriter = GetComponentInChildren<Typewriter>();
        OpenDialogueBox();
        StartDialogue();
    }

    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            CloseDialogueBox();
        }
    }

    private void OpenDialogueBox()
    {
        dialogueBox.SetActive(true);
    }

    public void CloseDialogueBox()
    {
        dialogueBox.SetActive(false);
        textLabel.text = string.Empty;
        if (openObject == 1)
            canvas.SetActive(true);
    }

    public void StartDialogue()
    {
        typewriter.Run(dialogueText, textLabel);
        StartCoroutine(CompleteDialogue());
    }
```

```
    private System.Collections.IEnumerator CompleteDialogue()
    {
        yield return new WaitForSeconds(10f);
        CloseDialogueBox();
    }
}
```

Listing 6: Typewriter.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class Typewriter : MonoBehaviour
{
    [SerializeField] private float speed = 100f;

    public Coroutine Run(string textToType, TMP_Text textLabel){
        return StartCoroutine(TypeText(textToType, textLabel));
    }

    private IEnumerator TypeText(string textToType, TMP_Text textLabel){
        yield return new WaitForSeconds(1);
        float t=0;
        int charIndex = 0;

        while (charIndex < textToType.Length)
        {
            t += Time.deltaTime * speed;
            charIndex = Mathf.FloorToInt(t);
            charIndex = Mathf.Clamp(charIndex, 0, textToType.Length);

            textLabel.text = textToType.Substring(0, charIndex);

            yield return null;
        }
        textLabel.text = textToType;
    }
}
```

Όταν τελειώσει το μήνυμα, ή ο χρήστης πατήσει το space και κάνει skip το διάλογο, αλλάζει ο καμβάς και ανοίγει το κύριο μενού.

4.2 Βασικό μενού ελέγχου

Στο κέντρο επάνω βρίσκεται το ρολόι της εφαρμογής με δυνατότητα ρύθμισης αφύπνησης.

Ακριβώς από κάτω βρίσκεται η δυνατότητα αναζήτησης μέσω φυλλομετρητή.

Αριστερά και δεξιά βρίσκονται τα κουμπιά του βασικού μενού. Από εδώ ο χρήστης μπορεί πλέον να επιλέξει όλες τις λειτουργίες του εικονικού προσωπικού βοηθού:

1. Ηλεκτρονικό Ταχυδρομείο (μέσω smtp/submission και gmail)
2. Καιρός

3. Ημερολόγιο με ραντεβού
4. Τηλέφωνο, Επαφές και Μηνύματα
5. e-Shop με λίστα αγορών
6. Music Player
7. Online Help

4.2.1 Ηλεκτρονικό Ταχυδρομείο (μέσω smtp και gmail)

Μέσω της υπηρεσίας αυτής ο χρήστης έχει την δυνατότητα να στείλει μηνύματα ηλεκτρονικού ταχυδρομείου. Η αποστολή γίνεται μέσω SMTP (submission with starttls on 587) πρωτοκόλου και gmail ώς πάροχο.

Χρησιμοποιείται ένα generic email της εφαρμογής (unityapp13@gmail.com).

Listing 7: EmailPanel.cs

```
using UnityEngine;
using UnityEngine.UI;
using System.Net;
using System.Net.Mail;
using TMPro;

public class EmailPanel : MonoBehaviour
{
    public TMP_InputField emailInputField;
    public TMP_InputField subjectInputField;
    public TMP_InputField messageInputField;

    public void SendEmail()
    {
        string senderEmail = "unityapp13@gmail.com";
        string senderPassword = "aeinkugbdzehxyxh";
        string recipientEmail = emailInputField.text;
        string subject = subjectInputField.text;
        string message = messageInputField.text;

        MailMessage mail = new MailMessage();
        SmtpClient smtpServer = new SmtpClient("smtp.gmail.com");

        mail.From = new MailAddress(senderEmail);
        mail.To.Add(recipientEmail);
        mail.Subject = subject;
        mail.Body = message;

        smtpServer.Port = 587;
        smtpServer.Credentials = new NetworkCredential(senderEmail, senderPass
        word);
        smtpServer.EnableSsl = true;

        try
        {
            smtpServer.Send(mail);
            Debug.Log("Email sent successfully!");
        }
        catch (System.Exception e)
```

```
        {
            Debug.LogError("Failed to send email: " + e.Message);
        }
    }

    public void CancelEmail()
    {
        // Clear input fields or perform any desired actions when canceling th
        e email.
        emailInputField.text = "";
        subjectInputField.text = "";
        messageInputField.text = "";
    }
}
```

4.2.2 Καιρός

Εδώ ο χρήστης μπορεί να ενημερωθεί για τον καιρό στην περιοχή του.

Να σημειωθεί ότι έχουμε χρησιμοποίησει και διαφορετικά εικονίδια για την παρουσίαση των καιρικών συνθηκών, όπως βροχή, ήλιος κλπ.

Έτσι ο χρήστης έχει πλήρη ενημέρωση, τόσο της θερμοκρασίας όσο και των συνθηκών.

Listing 8: PanelWeather.cs

```
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.UI;
using TMPro;
using System.Collections;

public class WeatherController : MonoBehaviour
{
    public TextMeshProUGUI temperatureText;
    public Image iconImage;
    public Sprite dayIcon;
    public Sprite nightIcon;
    public Sprite cloudyIcon;
    public Sprite rainyIcon;

    private const string APIKey = "67efa6dcf96741e883692637230507";
    private const string BaseURL = "http://api.weatherapi.com/v1/current.json
";
    private const string Location = "Athens";

    private void Start()
    {
        StartCoroutine(GetWeatherData());
    }

    private IEnumerator GetWeatherData()
    {
        string requestURL = $"{BaseURL}?key={APIKey}&q={Location}";
```

```
        using (UnityWebRequest request = UnityWebRequest.Get(requestURL))
        {
            yield return request.SendWebRequest();

            if (request.result == UnityWebRequest.Result.ConnectionError ||
            request.result == UnityWebRequest.Result.ProtocolError)
            {
                Debug.LogError($"Error: {request.error}");
            }
            else
            {
                string responseJson = request.downloadHandler.text;
                WeatherData weatherData = JsonUtility.FromJson<WeatherData>(r
                esponseJson);
                if (weatherData != null)
                {
                    float temperature = weatherData.current.temp_c;
                    temperatureText.text = $"{temperature}C";

                    string weatherCondition = weatherData.current.condition.
                    text;
                    UpdateWeatherIcon(weatherCondition);
                }
            }
        }
    }

    private void UpdateWeatherIcon(string condition)
    {
        if (condition.Contains("cloudy"))
        {
            System.DateTime currentTime = System.DateTime.Now;
            bool isDaytime = (currentTime.Hour >= 6 && currentTime.Hour <
            18);

            if (isDaytime)
            {
                iconImage.sprite = cloudyIcon;
            }
            else
            {
                iconImage.sprite = cloudyIcon;
            }
        }
        else if (condition.Contains("rain"))
        {
            iconImage.sprite = rainyIcon;
        }
        else
        {
            System.DateTime currentTime = System.DateTime.Now;
            bool isDaytime = (currentTime.Hour >= 6 && currentTime.Hour < 18);

            if (isDaytime)
```

```
            {
                iconImage.sprite = dayIcon;
            }
            else
            {
                iconImage.sprite = nightIcon;
            }
        }
    }
}

[System.Serializable]
public class WeatherData
{
    public CurrentData current;
}

[System.Serializable]
public class CurrentData
{
    public float temp_c;
    public ConditionData condition;
}

[System.Serializable]
public class ConditionData
{
    public string text;
}
```

4.2.3 Ημερολόγιο και Ραντεβού

Εδώ ο χρήστης μπορεί να δεί το ημερολόγιο και να κλείσει (ή να ακυρώσει ραντεβού).

Περιγραφικά, για την λειτουργία του ημερολογίου αρχικά δημιουργούνται δυναμικά τα κουμπιά για τις ημέρες του εκάστοτε μήνα και τοποθετούνται σωστά στην σωστή ημέρα με την οποία ξεκινάει κάθε φορά.

Στον χρήστη επιτρέπεται να κλείσει ραντεβού επιλέγοντας μία ημερομηνία και να το ακυρώσει ξαναπατώντας στην ήδη επιλεγμένη.

Εννοείται ότι το ημερολόγιο δείχνει και την σημερινή ημερομηνία.

Μερικά αποσπάσματα του κώδικα

Listing 9: CaledarController.cs - Snippet 1

```
public class CalendarController : MonoBehaviour
{
    private DateTime currentDate;
    private string[] months;
    public TextMeshProUGUI headerLabel;
    public GameObject buttonPrefab;
    public Transform buttonContainer;
    public float buttonSpacing = 10f;
    public int daysPerRow = 7;
```

Listing 10: CaledarController.cs - Snippet 2

```
    private void UpdateCalendar()
    {
        int daysInMonth = DateTime.DaysInMonth(currentDate.Year, currentDate.
        Month);
        DateTime firstDayOfMonth = new DateTime(currentDate.Year, currentDate.
        Month, 1);
        int startingIndex = GetDays(firstDayOfMonth.DayOfWeek);

        headerLabel.text = months[currentDate.Month - 1] + "␣" + currentDate.
        Year;

        // Clear existing buttons
        foreach (Transform child in buttonContainer)
        {
            Destroy(child.gameObject);
        }

        // Calculate the starting position for the buttons
        RectTransform buttonRect = buttonPrefab.GetComponent<RectTransform>();
        float startX = -daysPerRow / 2f * (buttonRect.rect.width + buttonSpa
        cing) + buttonRect.rect.width / 2f;
        float startY = (buttonRect.rect.height + buttonSpacing) / 2f;

        // Calculate the total height of the buttons
        int numRows = (daysInMonth + startingIndex + daysPerRow - 1) / days
        PerRow;
        float totalButtonHeight = numRows * (buttonRect.rect.height + button
        Spacing);

        // Create buttons for each day
        for (int i = 0; i < daysInMonth + startingIndex; i++)
        {
            int rowIndex = i / daysPerRow;
            int columnIndex = i % daysPerRow;

            int buttonDay = i - startingIndex + 1;

            GameObject buttonGO = Instantiate(buttonPrefab, buttonContainer);
            Button button = buttonGO.GetComponent<Button>();
            button.interactable = true;

            // Check if the buttonDay is within the valid range for the month
            if (buttonDay >= 1 && buttonDay <= daysInMonth)
            {
                button.GetComponentInChildren<TextMeshProUGUI>().text = button
                Day.ToString();
            }
            else
            {
                //Display an empty label for days outside the month's range

                button.GetComponentInChildren<TextMeshProUGUI>().text = string
```

```
                .Empty;
                button.interactable = false;
            }

            // Add an event listener to the button
            button.onClick.AddListener(() => ToggleButton(button));

            // Set button position within the container
            RectTransform buttonRectTransform = buttonGO.GetComponent<
            RectTransform>();
            buttonRectTransform.localScale = Vector3.one;
            buttonRectTransform.localPosition = new Vector3(startX + columnIndex
            * (buttonRect.rect.width + buttonSpacing), startY - rowIndex *
            (buttonRect.rect.height + buttonSpacing), 0f);
        }
    }
```

4.2.4 Τηλέφωνο, Επαφές και Μηνύματα

Πρόκειται για μία σύνθετη λειτουργία του προσωπικού βοηθού, που ουσιαστικά ανοίγει ένα ολόκληρο υπομενού.

Σε Αυτό ο χρήστης μπορεί να επιλέξει το τηλέφωνο, τα μηνύματα ή την λίστα επαφών.

α′ Στο τηλέφωνο, μπορεί να πληκτρολογήσει έναν αριθμό χρησιμοποιώντας το numpad που θα εμφανιστεί και να τον καλέσει. Ακόμη, μπορεί να τον προσθέσει στις επαφές χρησιμοποιώντας το σύμβολο ′+′.

Listing 11: ButtonHandler.cs - UpdateDisplay function

```
public void UpdateDisplay(string value)
    {
        if (displayText != null)
        {
            // Append the clicked button value to the TextMeshPro text
            displayText.text += value;

            // Check if the text length exceeds the maximum characters
            if (displayText.text.Length > MaxCharacters)
            {
                displayText.text = displayText.text.Substring(0, MaxCharacter
                s);
            }

            // Play button click sound
            if (audioSource != null && buttonClickSound != null)
            {
                audioSource.PlayOneShot(buttonClickSound);
            }
        }
    }
```

β′ Στα μηνύματα, μπορεί να πληκτρολογήσει το επιθυμητό μήμυμα και να το στείλει στον αριθμό της επιλογής του.

γ΄ Τέλος στις επαφές μπορεί να προσθέσει ένα όνομα και ένα τηλέφωνο στον κατάλογο.

Listing 12: AddButtonContact.cs - OnAddButtonClick function

```
    public void OnAddButtonClick()
    {
        string name = nameInputField.text;
        string number = numberInputField.text;

        contactManager.AddContact(name, number);

        // Clear input fields after adding the contact
        nameInputField.text = "";
        numberInputField.text = "";
    }
}
```

Οι επαφές και τα τηλέφωνα αποθηκεύονται στον δίσκο (σε json αρχείο) και ανακτώνται σε επόμενη χρήση της υπηρεσίας.

Listing 13: ContactSaveLoad.cs - SaveContacts function

```
   public void SaveContacts(List<ContactData> contacts)
    {
        ContactDataList contactDataList = new ContactDataList();
        contactDataList.contacts = contacts;

        string jsonData = JsonUtility.ToJson(contactDataList);
        File.WriteAllText(savePath, jsonData);
    }
```

Να σημειώσουμε ότι στα κουμπιά όλης της υπηρεσίας έχει προστεθεί ο αντίστοιχος γνωστός μας ήχος (πχ κλήση, πάτημα κουμπιών κλπ).

4.2.5 Τηλέφωνο, Επαφές και Μηνύματαe-Shop με λίστα αγορών

Εδώ ο χρήστης έχει την δυνατότητα να δεί διάφορα αντικείμενα προς πώληση και τις αντίστοιχες τιμές. Πάνω αριστερά βλέπει το υπόλοιπό του και μπορεί με αυτό να πραγματοποιήσει αγορές πατώντας πάνω στα αντίστοιχα προϊόντα.

Listing 14: ShopManagerScript.cs - Buy function

```
public void Buy()
    {
        GameObject ButtonRef = EventSystem.current.currentSelectedGameObject;

        if (ButtonRef != null)
        {
            ButtonInfo buttonInfo = ButtonRef.GetComponent<ButtonInfo>();
            if (buttonInfo != null)
            {
                int itemID = buttonInfo.ItemID;
                if (coins >= shopItems[2, itemID])
                {
                    coins -= shopItems[2, itemID];
                    shopItems[3, itemID]++;
                    CoinsTXT.text = "Coins:" + coins.ToString();
```

```
                    buttonInfo.QuantityTxt.text = shopItems[3, itemID].
                    ToString();
                }
            }
        }
    }
}
```

Με το κουμπί 'καρδιά' πάνω αριστερά, μπορεί να δεί την λίστα επιθυμιών του και να συμπληρώσει σε αυτή.

Listing 15: ShoppingList.cs - Shoping List functions

```
public void AddItem(string item)
    {
        shoppingItems.Add(item);
    }

    public void RemoveItem(string item)
    {
        shoppingItems.Remove(item);
    }

    public void DisplayList()
    {
        Debug.Log("Shopping␣List:");
        foreach (string item in shoppingItems)
        {
            Debug.Log(item);
        }
    }
```

4.2.6 Music Player

Η υπηρεσία αυτή επιτρέπει στον χρήστη να ακούσει μουσική. Υπάρχουν τα γνωστά μας κουμπιά για έλεγχο της αναπαραγωγής (Προηγούμενο, Επόμενο, Παυση/Αναπαραγωγή, Σίγαση κλπ). Τα κομμάτια είναι αποθηκευμένα στον δίσκο.

Listing 16: AudioManager.cs - WaitForMusicEnd cooroutine

```
IEnumerator WaitForMusicEnd() {
        while (source.isPlaying) {

            playTime = (int)source.time;
            showPlayTime();
            yield return null;
        }
        NextTrack();
    }
```

Listing 17: AudioManager.cs - NextTrack function

```
public void NextTrack() {
        source.Stop();
        currentTrack++;
        if (currentTrack > musicClips.Length - 1) {
            currentTrack = 0;
```

```
        }
        source.clip = musicClips[currentTrack];
        source.Play();

        // Show Title
        ShowCurrentTitle();

        StartCoroutine("WaitForMusicEnd");
    }
```

Listing 18: AudioManager.cs - Label Management functions

```
    void ShowCurrentTitle() {
        clipTitleText.text = source.clip.name;
        fullLength = (int)source.clip.length;
    }

    void showPlayTime(){
        seconds = playTime % 60;
        minutes = (playTime / 60) % 60;
        clipTimeText.text = minutes + ":" + seconds.ToString("D2") + "/" +
        ((fullLength / 60) % 60) + ":" + (fullLength % 60).ToString("D2");
    }
```

4.2.7 BBC News

Η υπηρεσία αυτή επιτρέπει στον χρήστη να δεί τις τελευταίες ειδήσεις και φωτογραφίες αν υπάρχουν, του BBC (μέσω API).

Listing 19: NewsPanel.cs - FetchNewsData function

```
    IEnumerator FetchNewsData()
    {
        string url = $"{newsApiUrl}?sources={newsSource}&apiKey={apiKey}";

        UnityWebRequest request = UnityWebRequest.Get(url);
        yield return request.SendWebRequest();

        if (request.result != UnityWebRequest.Result.Success)
        {
            Debug.Log("Failed to fetch news data: " + request.error);
            yield break;
        }

        string jsonResponse = request.downloadHandler.text;
        newsData = JsonUtility.FromJson<NewsData>(jsonResponse);

        if (newsData != null && newsData.articles.Length > 0)
        {
            PopulateNewsButtons();
        }
    }
```

5 Γενικές Παρατηρήσεις

5.1 Συναρτήσεις Ελέγχου

Για όλα τα παραπάνω χρησιμοποιήθηκαν οι εξείς συναρτήσεις ελέγχου των παραθύρων:

Με το πάτημα των κουμπιών καλείται το γενικό σκριπτ διαχείρισης παραθύρων (πανελ χοντρολλερ) το οποίο περιέχει τις μεθόδους που μας επιτρέπουν την διαχείριση των πάνελ σε έναν καμβά. Για παράδειγμα άνοιγμα, κλείσιμο, φαδε-ουτ κλπ.

Listing 20: PanelController.cs

```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class PanelController : MonoBehaviour
{

    public GameObject panelToShow; // Reference to the panel to show
    public GameObject panelToFadeOut; // Reference to the panel to fade out
    public float fadeOutDelay = 2f; // Delay before fading out the second panel
    public float fadeOutDuration = 1f; // Duration of the fade out animation

    public void ShowPanel()
    {
        panelToShow.SetActive(true); // Set the panel to active (visible)
    }

    public void HidePanel()
    {
        panelToFadeOut.SetActive(true);
        StartCoroutine(FadeOutPanel()); // Start the fade out coroutine
    }

    private IEnumerator FadeOutPanel()
    {
        panelToShow.SetActive(false);
        yield return new WaitForSeconds(fadeOutDelay);

        Image panelImage = panelToFadeOut.GetComponent<Image>();
        TextMeshProUGUI panelText = panelToFadeOut.GetComponentInChildren
        <TextMeshProUGUI>();

        Color startColor = panelImage.color;
        Color targetColor = new Color(startColor.r, startColor.g,
        startColor.b, 0f);
        Color start1 = new Color(startColor.r, startColor.g, startColor.
        b, 1f);
        float elapsedTime = 0f;

        while (elapsedTime < fadeOutDuration)
        {
            elapsedTime += Time.deltaTime;
```

```
            float normalizedTime = elapsedTime / fadeOutDuration;
            panelText.color = Color.Lerp(startColor, targetColor, normalized
            Time);
            panelImage.color = Color.Lerp(startColor, targetColor, normalized
            Time);
            yield return null;
        }

        panelToFadeOut.SetActive(false); // Set the panel to inactive (hidden)
        panelText.color= Color.white;
        panelImage.color = start1;
    }


}
```

5.2 Γραμματοσειρές

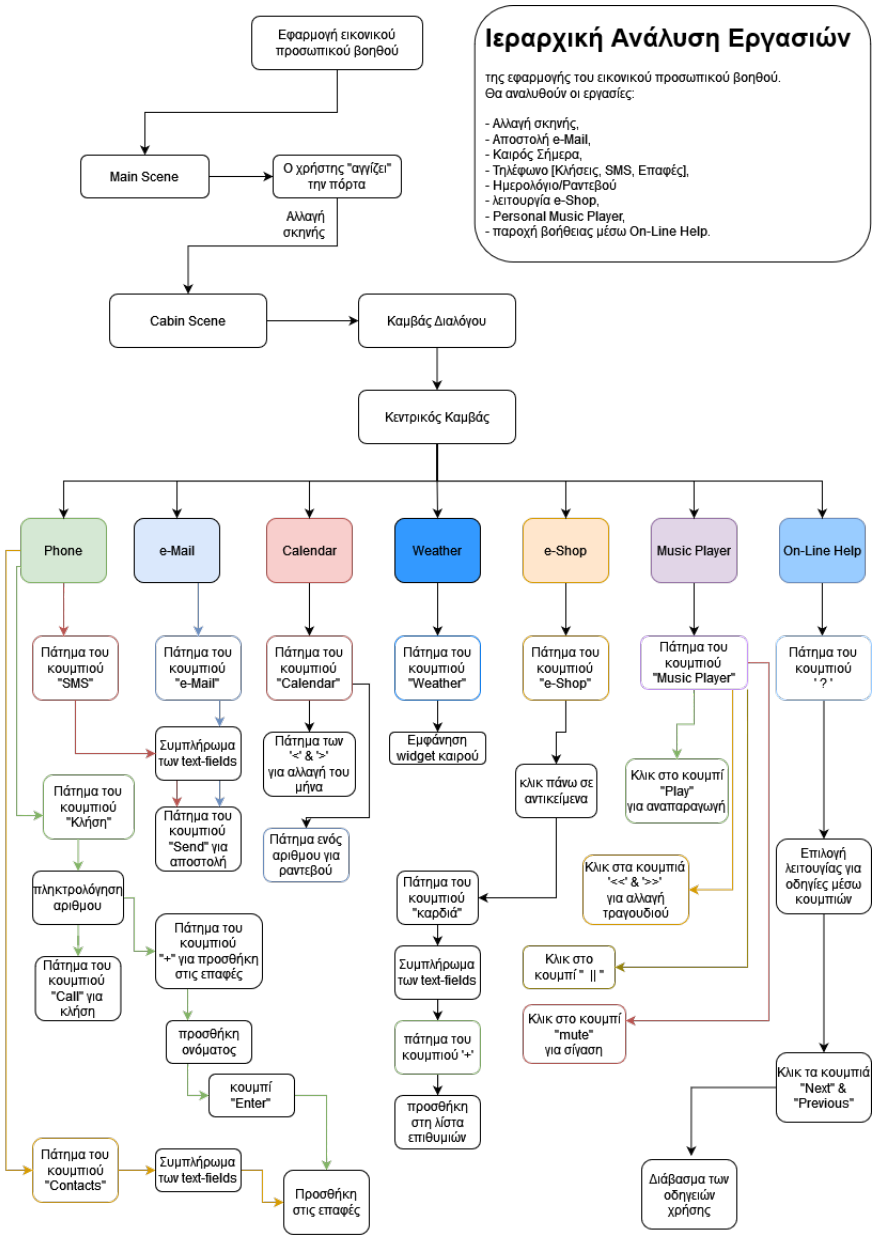
Για τις αγγλικές γραμματοσειρές χρησιμοποιήθηκε το “LiberationSans SDF” ενώ για τις ελληνικές το “GFS Didot-Regular SDF”.

6 Online Help

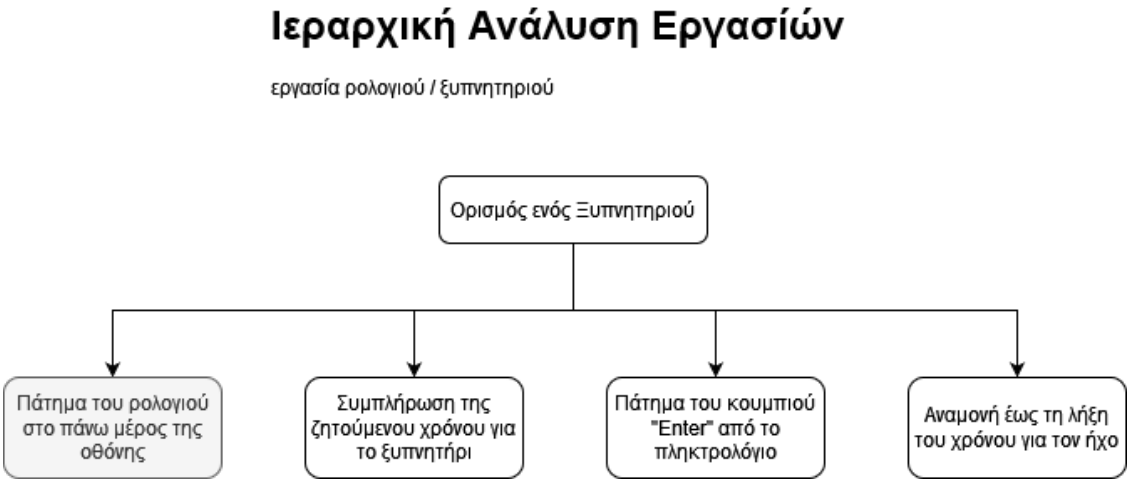
Στην κάτω δεξιά γωνία ο χρήστης μπορεί να βρεί το κουμπί της βοήθειας και να δεί το online help menu, με συνοπτικές οδηγίες για την εφαρμογή.

Σε περίπτωση που θέλει πιο αναλυτική περιγραφή των υπηρεσιών θα πρέπει να ανατρέξει το User Manual.

7 Σχεδιαγράμματα



Σχήμα 3: Ιεραρχική Ανάλυση Εργασιών I



Σχήμα 4: Ιεραρχική Ανάλυση Εργασιών II