# WordClone: A Microsoft Word Clone Application

## FileManager Script

This code serves as a file manager in the Unity game engine. It provides basic file operations such as opening, saving, and saving with different names to the user. The functionality is described below:

```
Unity Betiği (3 varlık başvurusu) | 0 başvuru
public class FileManager : MonoBehaviour//, IPointerDownHandler
{
    // Sample text data
    public TMP_InputField dataText;
```

The FileManager class is derived from the MonoBehaviour class, indicating that it can be used on a game object.

A TMP_InputField variable named dataText is used to store text data. Users input data into this text field or data is retrieved from an opened file.

```
0 başvuru
public void OnSaveButtonClicked()
{
    if (PlayerPrefs.GetInt("isSaved") == 1)
    {
        string filePath = PlayerPrefs.GetString("localPath");

        // Veriyi metin belgesine yaz
        File.WriteAllText(filePath, dataText.text);

        Debug.Log("File saved to desktop: " + filePath);
    }
    else
    {
        OnSaveAsButtonClick();
    }

}
```

The OnSaveButtonClicked() function is called when the "Save" button is clicked. If a file has been previously saved, it writes the current data to the saved file. If no file has been saved before, it calls the OnSaveAsButtonClick() function.

```
1 başvuru
public void OnSaveAsButtonClick()
{
    var path = StandaloneFileBrowser.SaveFilePanel("Title", "", "sample", "txt");
    if (!string.IsNullOrEmpty(path))
    {
        File.WriteAllText(path, dataText.text);
        Debug.Log("path: " + path);
        PlayerPrefs.SetString("localPath", path);
        PlayerPrefs.SetInt("isSaved", 1);
    }
}
```

The OnSaveAsButtonClick() function is called when the "Save As" button is clicked. It opens a file selection dialog for the user to choose a file name and location. The content of dataText is written to the selected location, and the path of the file and whether it's saved or not is stored in PlayerPrefs.

```
0 başvuru
public void OnOpenFileClick()
{
    var paths = StandaloneFileBrowser.OpenFilePanel("Title", "", "txt", false);
    if (paths.Length > 0)
    {
        StartCoroutine(OutputRoutine(new System.Uri(paths[0]).AbsoluteUri));
        PlayerPrefs.SetString("localPath", "");
        PlayerPrefs.SetInt("isSaved", 0);
    }
}
```

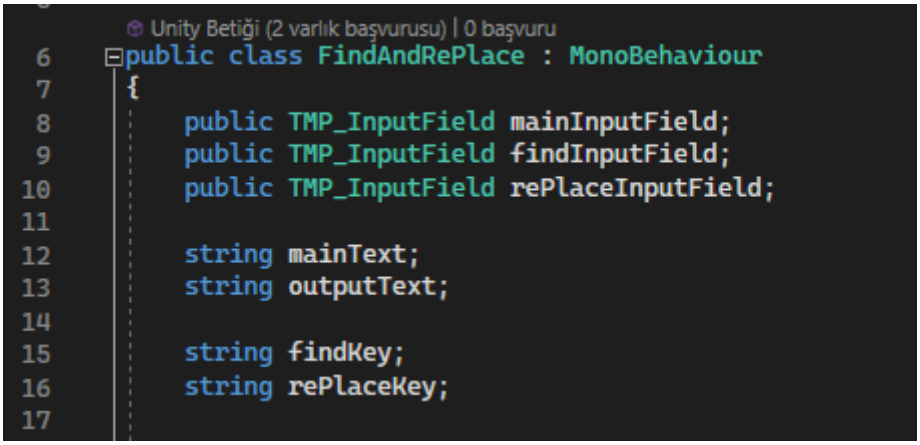
The OnOpenFileClick() function is called when the "Open File" button is clicked. It opens a file selection dialog for the user to choose a file, and if a file is selected, its content is loaded into the dataText field.

```
1 başvuru
private IEnumerator OutputRoutine(string url)
{
    var loader = new WWW(url);
    yield return loader;
    dataText.text = loader.text;
}
```

The OutputRoutine() function takes a file path and reads the content of the file at that path. However, this is done using the WWW class, which is now a deprecated method. It is recommended to use UnityWebRequest instead.

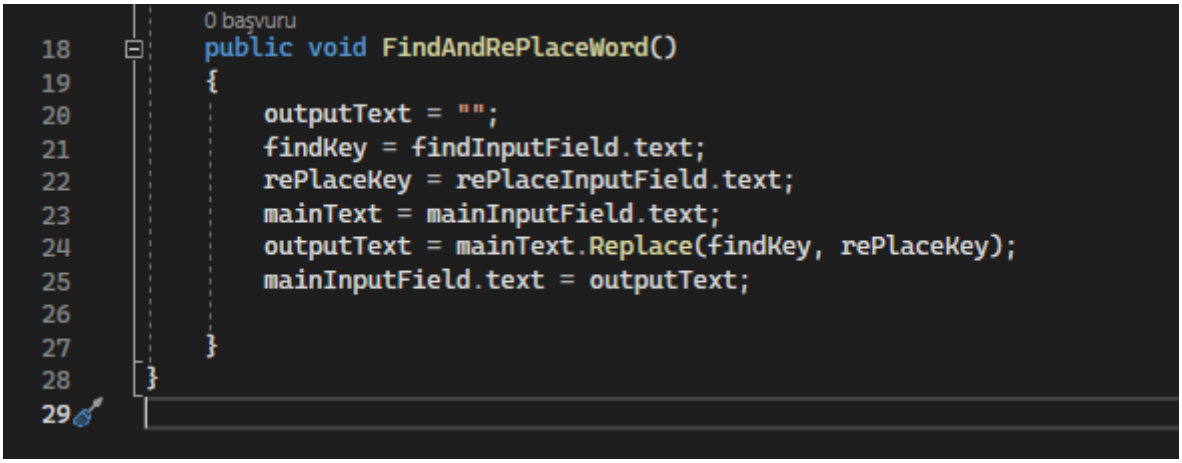
# FindAndRePlace

The script defines a class called FindAndRePlace that inherits from MonoBehaviour, which means it can be attached to a GameObject in Unity.




Public fields are declared for three TMP_InputField variables (mainInputField, findInputField, and rePlaceInputField). These fields likely correspond to the main input field, find input field, and replace input field in the Unity editor.

Four private string variables are declared (mainText, outputText, findKey, and rePlaceKey). These are used for storing the text content of the input fields and the find and replace strings.




The FindAndRePlaceWord() method is a public method that will be called when some action triggers it (e.g., a button click). Inside this method:

The outputText variable is initialized to an empty string.

The text from the findInputField and rePlaceInputField is stored in the findKey and rePlaceKey variables, respectively.

The text from the mainInputField is stored in the mainText variable.

The Replace() method is used to find and replace occurrences of findKey within mainText with rePlaceKey. The result is stored in the outputText variable.

The outputText is then assigned back to mainInputField.text, updating the main input field with the replaced text.

Overall, this script provides functionality to find and replace text within a Unity UI text input field. When FindAndRePlaceWord() is called, it replaces occurrences of a specified text in the main input field with another specified text.

# TextColorChanger

```
Unity Betiği (2 varlık başvurusu) | 0 başvuru
public class TextColorChanger : MonoBehaviour
{
    public TMP_InputField inputField;
    public Color targetColor;
    private int selectionStartIndex; // Seçimin başlangıç indeksi
    private int selectionEndIndex; // Seçimin bitiş indeksi
    private bool colorChanged = false; // Metnin rengi değiştirildi mi?
    public FlexibleColorPicker fCP;
    public bool isColorButtonClicked;
    public float delay = 5.0f;
    public GameObject applyColorButton;

```

The script defines a class called TextColorChanger that allows changing the color of selected text within a TMP_InputField.

Public fields are declared for the TMP_InputField, the target color, a color picker (FlexibleColorPicker), a boolean flag to determine if the color button is clicked, a delay value, and a game object for the apply color button.

```
Unity İletisi | 0 başvuru
private void Update()
{
    targetColor = fCP.color;
}
```

The Update() method updates the target color from the color picker.

```
0 başvuru
public void ChangeSelectedTextToColor()
{
    if (inputField != null)
    {
        //colorButtonBackGround.color = targetColor;
        // Seçimin başlangıç ve bitiş indekslerini al
        selectionStartIndex = inputField.selectionStringAnchorPosition;
        selectionEndIndex = inputField.selectionStringFocusPosition;


        Debug.Log(selectionStartIndex + " " +selectionEndIndex);
        // Başlangıç indeksini bitiş indeksinden küçük yap
        if (selectionStartIndex > selectionEndIndex)
        {
            int temp = selectionStartIndex;
            selectionStartIndex = selectionEndIndex;
            selectionEndIndex = temp;
            Debug.Log("test");
        }

        // Seçili metni değiştir
        string originalText = inputField.text;
        string selectedText = originalText.Substring(selectionStartIndex, selectionEndIndex - selectionStartIndex);
        string coloredText = "<color=#" + ColorUtility.ToHtmlStringRGB(targetColor) + ">" + selectedText + "</color>";
        string newText = originalText.Substring(0, selectionStartIndex) + coloredText + originalText.Substring(selectionEndIndex);

        // Metni güncelle
        inputField.text = newText;
        colorChanged = true;
    }
}
```

The ChangeSelectedTextToColor() method changes the color of the selected text to the target color. It first retrieves the start and end indices of the selection, then gets the

selected text, wraps it with HTML color tags, and replaces the original text with the colored text.

```
0 başvuru
public void OpenColorPicker()
{
    fCP.gameObject.SetActive(true);
    applyColorButton.gameObject.SetActive(true);
}
```

The OpenColorPicker() method activates the color picker and the apply color button.

```
0 başvuru
public void RestoreOriginalText()
{
    if (colorChanged)
    {
        // Metni orijinal haline geri döndür
        inputField.text = inputField.text.Replace("<color=#" + ColorUtility.ToHtmlStringRGB(targetColor) + ">", "");
        inputField.text = inputField.text.Replace("</color>", "");
        colorChanged = false;
        fCP.gameObject.SetActive(false);
        applyColorButton.gameObject.SetActive(false);
    }
}
}
```

The RestoreOriginalText() method restores the original text by removing the HTML color tags. It then deactivates the color picker and the apply color button.

# TextStyleChangerDogan

```
Unity Betiği (3 varlık başvurusu) | 0 başvuru
public class TextStyleChangerDogan : MonoBehaviour
{
    public TMP_InputField inputField, sizeInputField;
    private int selectionStartIndex;
    private int selectionEndIndex;
    public bool styleBoldChanged = false;
    public bool styleItalicChanged = false;
    public bool styleUnderLineChanged = false;
    public int fontSize;
    public GameObject textInMainInputField;
    public TMP_Text tmpTextInMainInputField;
    public GameObject textObject;
    public TMP_Text text1;

    public bool isFontSizeChanged;
```

This script allows changing text styles (bold, italic, underline) and font size of the selected text within a TMP_InputField.

Public fields are declared for the TMP_InputField for text input, another TMP_InputField for changing font size, selection indices, flags to track style changes, font size, and related game objects.

```
Unity İletisi | 0 başvuru
private void Start()
{
    textObject = GameObject.FindGameObjectWithTag("TextInMainInputField");
    text1 = textObject.gameObject.GetComponent<TMP_Text>();
    sizeInputField.onValueChanged.AddListener(OnInputFieldValueChanged);
}
```

The Start() method initializes the references to the text object and subscribes to the onValueChanged event of the font size input field.

```
0 başvuru
public void ChangeSelectedTextToItalic()
{
    if (inputField != null)
    {
        selectionStartIndex = inputField.selectionStringAnchorPosition;
        selectionEndIndex = inputField.selectionStringFocusPosition;

        if (selectionStartIndex > selectionEndIndex)
        {
            int temp = selectionStartIndex;
            selectionStartIndex = selectionEndIndex;
            selectionEndIndex = temp;
        }

        string originalText = inputField.text;
        string selectedText = originalText.Substring(selectionStartIndex, selectionEndIndex - selectionStartIndex);
        string italicText = "<i>" + selectedText + "</i>";
        string newText = originalText.Substring(0, selectionStartIndex) + italicText + originalText.Substring(selectionEndIndex);

        inputField.text = newText;
        styleItalicChanged = true;
    }
}
```

```
0 başvuru
public void ChangeSelectedTextToBold()
{
    if (inputField != null)
    {
        selectionStartIndex = inputField.selectionStringAnchorPosition;
        selectionEndIndex = inputField.selectionStringFocusPosition;

        if (selectionStartIndex > selectionEndIndex)
        {
            int temp = selectionStartIndex;
            selectionStartIndex = selectionEndIndex;
            selectionEndIndex = temp;
        }

        string originalText = inputField.text;
        string selectedText = originalText.Substring(selectionStartIndex, selectionEndIndex - selectionStartIndex);
        string boldText = "<b>" + selectedText + "</b>";
        string newText = originalText.Substring(0, selectionStartIndex) + boldText + originalText.Substring(selectionEndIndex);

        inputField.text = newText;
        styleBoldChanged = true;
    }
}
```

```
0 başvuru
public void ChangeSelectedTextToUnderline()
{
    if (inputField != null)
    {
        selectionStartIndex = inputField.selectionStringAnchorPosition;
        selectionEndIndex = inputField.selectionStringFocusPosition;

        if (selectionStartIndex > selectionEndIndex)
        {
            int temp = selectionStartIndex;
            selectionStartIndex = selectionEndIndex;
            selectionEndIndex = temp;
        }

        string originalText = inputField.text;
        string selectedText = originalText.Substring(selectionStartIndex, selectionEndIndex - selectionStartIndex);
        string underlineText = "<u>" + selectedText + "</u>";
        string newText = originalText.Substring(0, selectionStartIndex) + underlineText + originalText.Substring(selectionEndIndex);

        inputField.text = newText;
        styleUnderLineChanged = true;
    }
}
```

Methods like ChangeSelectedTextToBold(), ChangeSelectedTextToItalic(), and ChangeSelectedTextToUnderline() change the selected text to the corresponding style by wrapping it in HTML tags (, <i>, <u>). Similar methods exist to restore the original text style.

```
0 başvuru
public void RestoreBoldToUnBold()
{
    if (styleBoldChanged)
    {
        inputField.text = inputField.text
            .Replace("<b>", "").Replace("</b>", "");
        styleBoldChanged = false;
    }
}
```

RestoreBoldToUnBold() Method:

This method is used to remove the bold style. If a text fragment is in bold style, when this method is called, it removes the  and  HTML tags, effectively removing the bold style from the text.

If the styleBoldChanged variable is true, meaning that the text style has been changed to bold, the process continues.

inputField.text represents the current text in the input field.

Using the Replace() method, the  and  HTML tags are removed.

The styleBoldChanged variable is set to false as the process of removing the bold style is completed.

The input field is updated, and the text is now displayed without the bold style.

```
0 başvuru
public void RestoreItalicToNonItalic()
{
    if (styleItalicChanged)
    {
        inputField.text = inputField.text.Replace("<i>", "").Replace("</i>", "");
        styleItalicChanged = false;
    }
}
```

RestoreItalicToNonItalic() Method:

This method is used to remove the italic style. If a text fragment is in italic style, when this method is called, it removes the  and  HTML tags, effectively removing the italic style from the text.

If the styleItalicChanged variable is true, meaning that the text style has been changed to italic, the process continues.

inputField.text represents the current text in the input field.

Using the Replace() method, the  and  HTML tags are removed.

The styleItalicChanged variable is set to false as the process of removing the italic style is completed.

The input field is updated, and the text is now displayed without the italic style.

```
0 başvuru
public void RestoreUnderlineToText()
{
    if (styleUnderLineChanged)
    {
        inputField.text = inputField.text
            .Replace("<u>", "").Replace("</u>", "");
        styleUnderLineChanged = false;
    }
}
```

RestoreUnderlineToText() Method:

This method is used to remove the underline style. If a text fragment is underlined, when this method is called, it removes the <u> and </u> HTML tags, effectively removing the underline style from the text.

If the styleUnderLineChanged variable is true, meaning that the text style has been changed to underline, the process continues.

inputField.text represents the current text in the input field.

Using the Replace() method, the <u> and </u> HTML tags are removed.

The styleUnderLineChanged variable is set to false as the process of removing the underline style is completed.

The input field is updated, and the text is now displayed without the underline style.

# WordCountDogan

```
Unity Betiği (2 varlık başvurusu) | 0 başvuru
public class WordCountDogan : MonoBehaviour
{
    public TMP_InputField inputField;
    public TMP_Text wordCountText;

    int wordCount;

```

It is responsible for counting the number of words in a text input field (TMP_InputField) and displaying the count in a text component (TMP_Text).

inputField: A public field to reference the input field where the text is entered.

wordCountText: A public field to reference the text component where the word count will be displayed.

wordCount: An integer variable to store the calculated word count.

```
1 başvuru
public void CountWords()
{
    string text = inputField.text;
    string[] words = text.Split(new char[] { ' ', '\n', '\t' }, System.StringSplitOptions.RemoveEmptyEntries);
    wordCount = words.Length;
    //Debug.Log("Word count: " + wordCount);
}
```

This method calculates the word count.

It first retrieves the text from the input field (inputField.text).

Then, it splits the text into an array of words using space (' '), newline ('\n'), and tab ('\t') characters as delimiters. It removes any empty entries using StringSplitOptions.RemoveEmptyEntries.

Finally, it assigns the length of the resulting array to the wordCount variable, representing the total number of words.

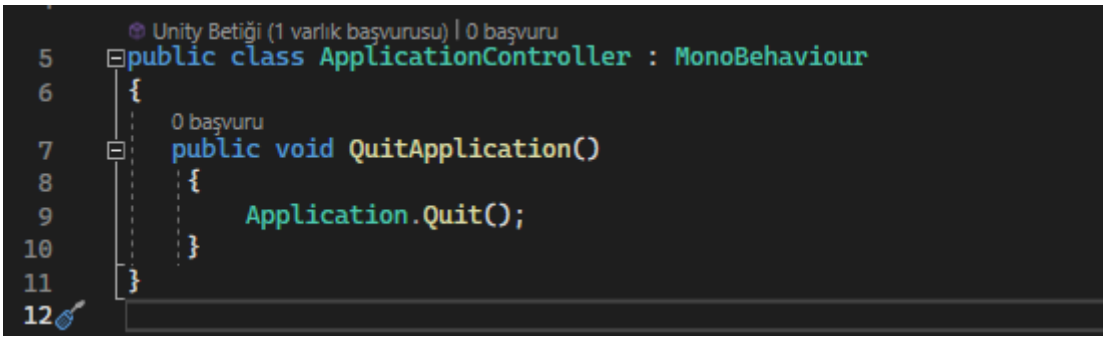
```
Unity İletisi | 0 başvuru
private void Update()
{
    CountWords();
    wordCountText.text = "Word Count: " + wordCount.ToString();
}
}
```

The Update() method is called every frame.

In this method, CountWords() is called to update the word count.

Then, the word count is displayed in the wordCountText text component by setting its text to "Word Count: " followed by the value of wordCount.

## ApplicationController




The ApplicationController class in the provided script is a simple MonoBehaviour that contains a single method QuitApplication().

This is a MonoBehavior class named ApplicationController.

It is responsible for handling application-level functionalities.

This method is a public void method named QuitApplication.

It is called when the application needs to be quit.

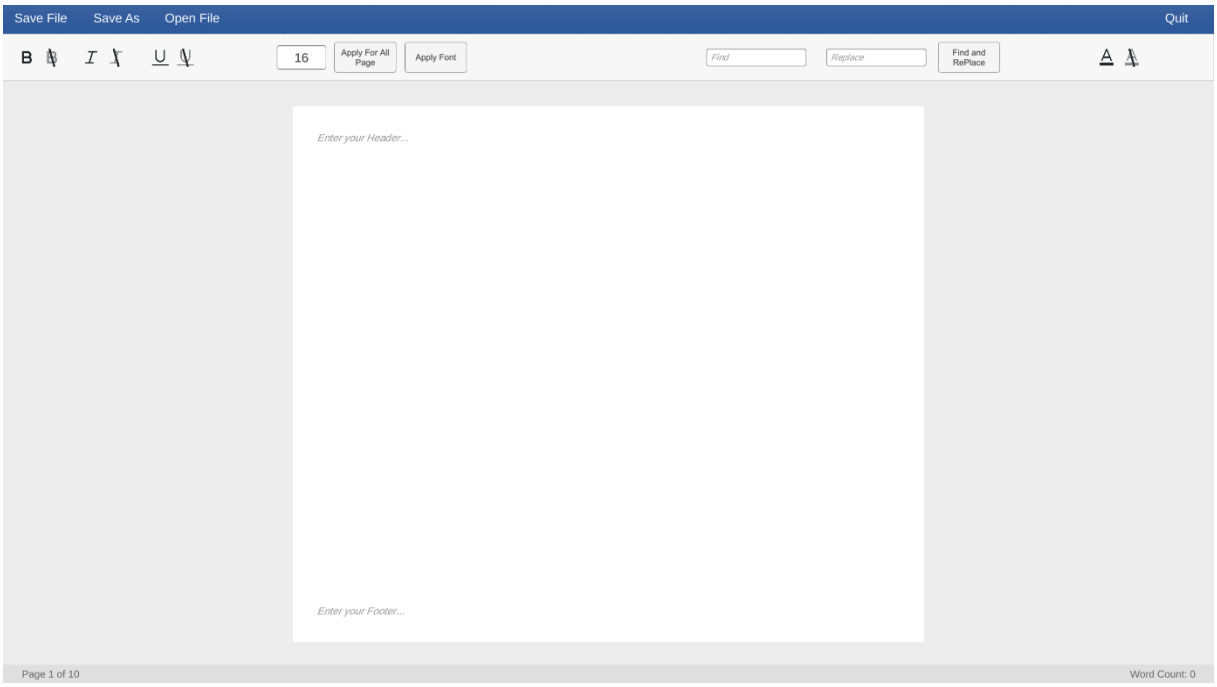
Inside the method, it calls Application.Quit().

Application.Quit() is a static method of the Application class provided by Unity. It is used to quit the application.

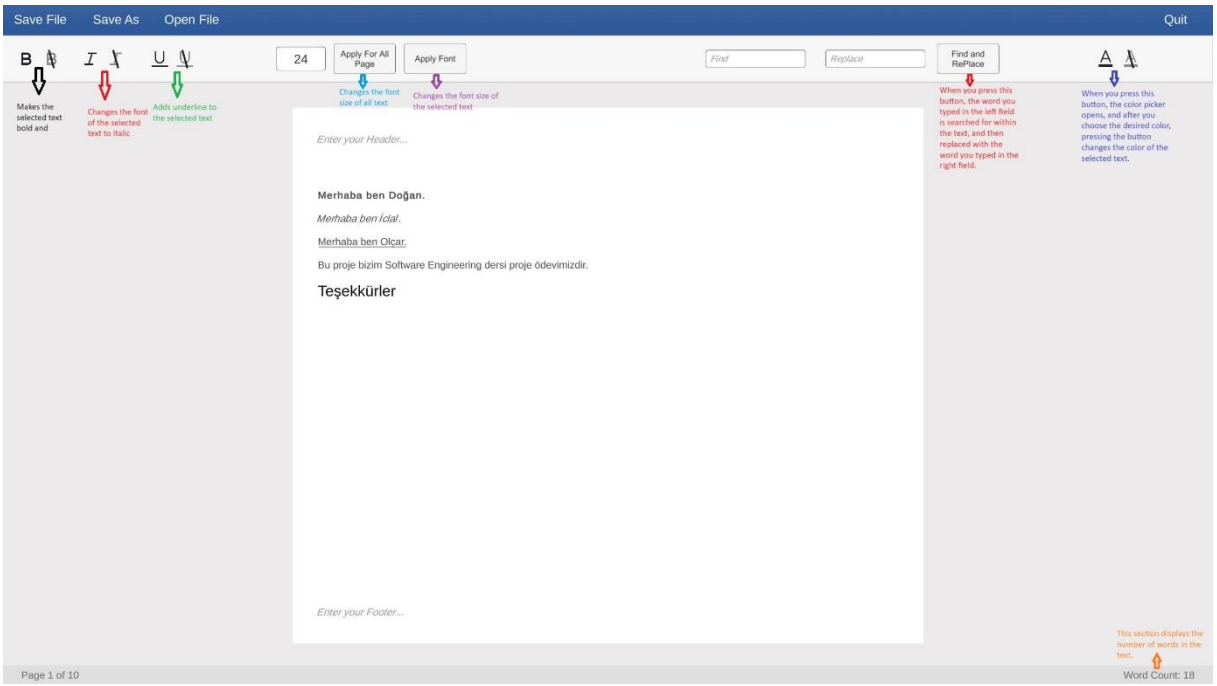
This class provides a simple way to quit the application when necessary. When the QuitApplication() method is called, it invokes Unity's Application.Quit() method, which terminates the application. This can be useful for providing an exit option in the user interface or for handling quit commands.

## Screen Shoots About Running App

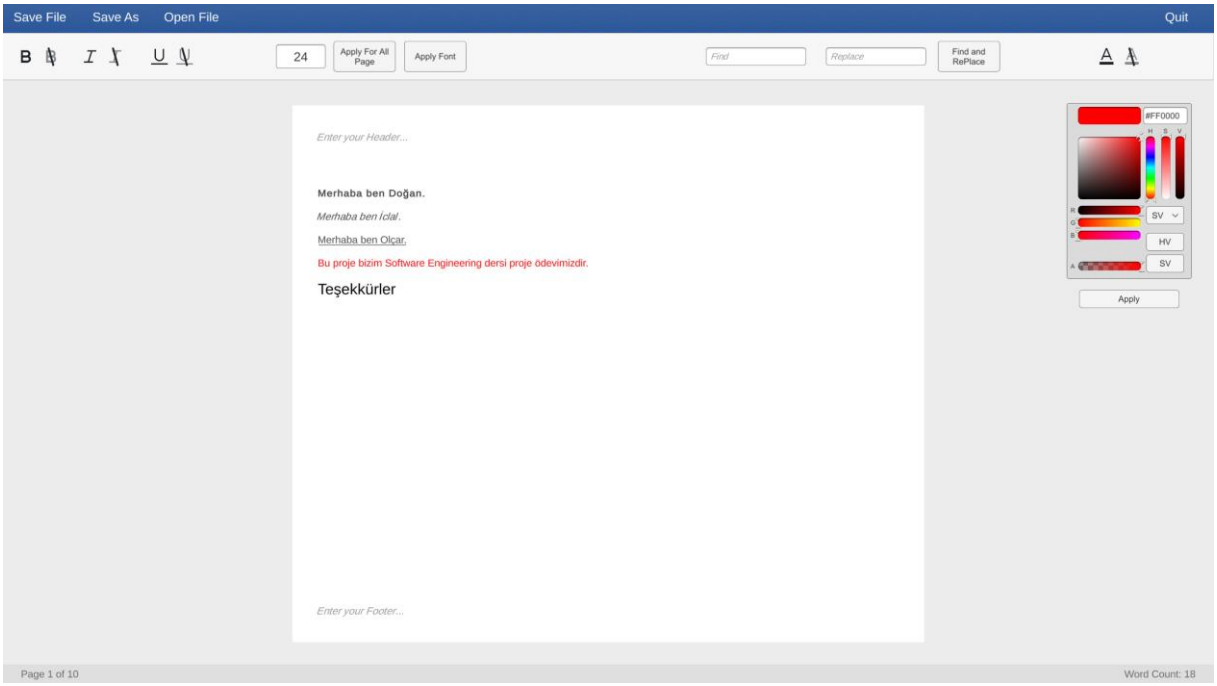
Main Scene



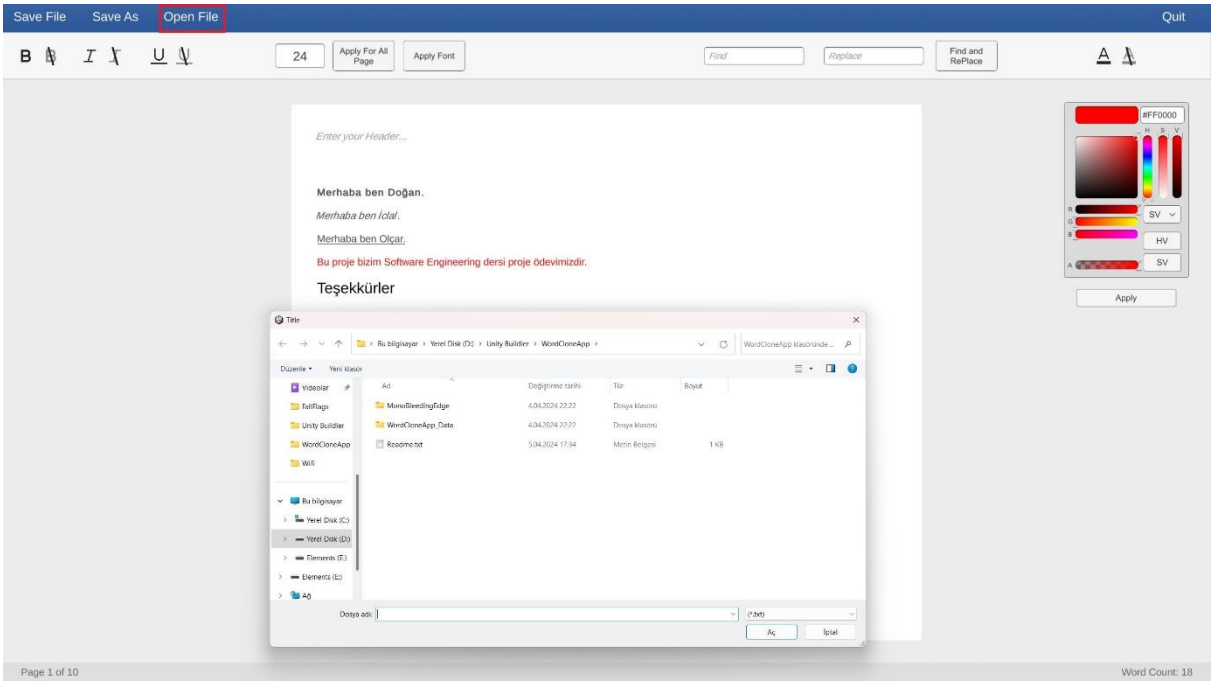

Main Scene With Descriptions



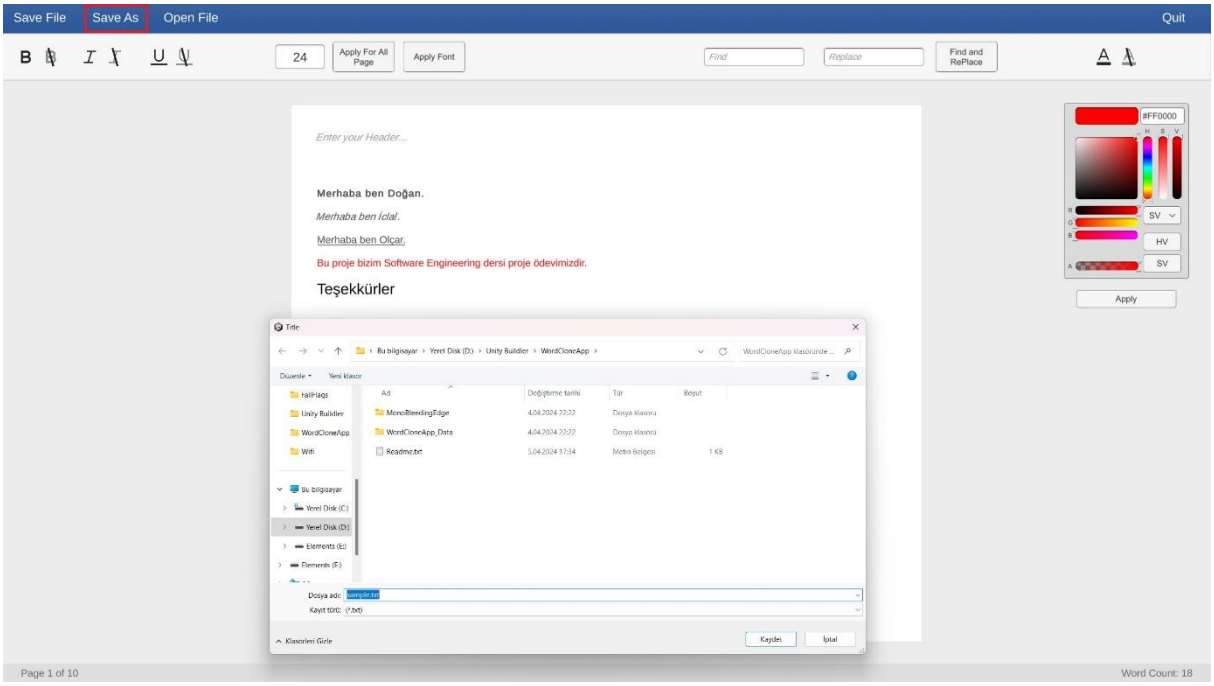
Test




Click Open File Button



Click Save As Button




Note:

If you haven't saved the file previously, when you click the Save button, it functions as if the Save As button does. However, if you have saved it before, clicking the Save button will overwrite the previously saved file.