UTeleApp Documentation

Overview

This is a Telegram Web App / Mini App plugin for Unity that allows developers to integrate Telegram's web functionalities directly into their Unity applications, particularly for WebGL builds. This document provides a comprehensive guide on how to utilize the plugin effectively.

Getting Started

Installation

- Download the plugin files from the Unity Store / provided links.
- Import the downloaded files into your Unity project.

Platform Support

- The plugin is designed to work specifically with **WebGL** builds. Ensure that your project is set to build for this platform.

Examples

Please Check Assets/OmnInteractive/Examples/.., we have 1 scenes to demo the plugin usage. Please use the UTeleApp's **WebGLTemplates** to build with and test. If you want to user your own **WebGLTemplates,** please add

```
<script src="https://telegram.org/js/telegram-web-app.js"></script>
```

to your html code/ js code for injecting Telegram Web App API.

To test the example, you need to setup your own bot and host the build with http server and set your mini app url on Telegram's Bot Father.

Key Features and Functions

All the functions from **https://core.telegram.org/bots/webapps#initializing-mini-apps** is supported.

Properties

- InitData: Retrieves initialization data as a string.
- InitDataUnsafe: Gets unsafe initialization data as a JSON object.
- Version: Retrieves the current version of the Telegram Web App.
- Platform: Gets the platform information of the Telegram Web App.
- ThemeParams: Accesses theme-related parameters such as colors.
- ColorScheme: Retrieves the current color scheme.
- HeaderColor: Gets the header color.
- BackgroundColor: Retrieves the background color.
- ViewportHeight: Gets the viewport height.
- ViewportStableHeight: Gets the stable viewport height.
- IsExpanded: Checks if the viewport is expanded.
- IsClosingConfirmationEnabled: Checks if closing confirmation is enabled.
- IsVerticalSwipesEnabled: Checks if vertical swipes are enabled.

Methods

The following methods are available in the `TelegramWebApp` class:

1. Navigation Methods
    - `Ready()`: Marks the app as ready for interaction.
    - `Expand()`: Expands the Telegram Web App view.
    - `Close()`: Closes the Telegram Web App.
    - `EnableClosingConfirmation()`: Enables closing confirmation in the app.
    - `DisableClosingConfirmation()`: Disables closing confirmation in the app.
    - `EnableVerticalSwipes()`: Enables vertical swipes in the app.
    - `DisableVerticalSwipes()`: Disables vertical swipes in the app.
2. Utility Methods
    - `IsVersionAtLeast(string version)`: Checks if the current version is at least the specified version.
3. UI Methods
    - `ShowConfirm(string message)`: Shows a confirmation dialog with a specified message.
    - `ShowAlert(string message)`: Displays an alert with a specified message.

4. Main Button Methods
    - `ShowMainButton()`: Displays the main button in the app.
    - `HideMainButton()`: Hides the main button in the app.
    - `SetMainButtonText(string text)`: Sets the text of the main button.
5. Back Button Methods
    - `ShowBackButton()`: Displays the back button in the app.
    - `HideBackButton()`: Hides the back button in the app.
6. Secondary Button Methods
    - `ShowSecondaryButton()`: Displays a secondary button in the app.
    - `HideSecondaryButton()`: Hides the secondary button in the app.
    - `SetSecondaryButtonText(string text)`: Sets text for the secondary button.
7. Settings Button Methods
    - `ShowSettingsButton()`: Displays settings button in the app.
    - `HideSettingsButton()`: Hides settings button in the app.
8. Link and Invoice Methods
    - `OpenLink(string url)`: Opens a link in an external browser.
    - `OpenTelegramLink(string url)`: Opens a Telegram link.
    - `OpenInvoice(string url)`: Opens an invoice.
9. Media Sharing and Popups
    - `ShareToStory(string mediaUrl)`: Shares media to a user's story.
    - `ShowPopup(PopupParams parameters)`: Shows a popup with specified parameters.
10. Clipboard and Access Requests
    - `ReadTextFromClipboard()`: Reads text from clipboard.
    - `RequestWriteAccess()`: Requests write access to storage.
    - `RequestContact()`: Requests a contact from user.
11. Color Settings
    - `SetBottomBarColor(string color)`: Sets color for bottom bar.
    - `SetHeaderColor(string color)`: Sets header color.
    - `SetBackgroundColor(string color)`: Sets background color.

Events

List of Events

1. OnThemeChanged
    - Description: Triggered when the theme of the app changes.
    - Usage: Subscribe to this event to update UI elements or settings based on the new theme.
2. OnViewportChanged

- Description: Triggered when the viewport changes, with a parameter indicating the change details.
- Parameter: `ViewportChangedReturnType` - Contains information about the new viewport state.
- Usage: Use this event to adjust layouts or UI components when the viewport size or orientation changes.

3. OnMainButtonClicked
    - Description: Triggered when the main button is clicked by the user.
    - Usage: Subscribe to this event to execute specific actions when the main button is interacted with.
4. OnSecondaryButtonClicked
    - Description: Triggered when the secondary button is clicked.
    - Usage: Handle actions related to secondary button interactions.
5. OnBackButtonClicked
    - Description: Triggered when the back button is clicked.
    - Usage: Use this event to manage navigation or other actions when the user chooses to go back.
6. OnSettingsButtonClicked
    - Description: Triggered when the settings button is clicked.
    - Usage: This event can be used to open settings menus or perform related tasks.
7. OnInvoiceClosed
    - Description: Triggered when an invoice is closed, with a parameter indicating the closure details.
    - Parameter: `InvoiceClosedReturnType` - Contains information about the closed invoice.
    - Usage: Handle cleanup or state updates after an invoice has been closed.
8. OnPopupClosed
    - Description: Triggered when a popup is closed, with a parameter indicating the closure details.
    - Parameter: `PopUpClosedReturnType` - Contains information about the closed popup.
    - Usage: Use this event for managing UI states after popups are dismissed.
9. OnQrTextReceived
    - Description: Triggered when QR text is received from a QR code scan.
    - Parameter: `QrTextReceivedReturnType` - Contains the scanned text data.
    - Usage: Handle actions based on QR code scans, such as navigating to a URL or processing data.
10. OnScanQrPopupClosed
    - Description: Triggered when the QR scanner popup is closed.

- Usage: Manage UI states or cleanup after the QR scanner popup is dismissed.

11. OnClipboardTextReceived
    - Description: Triggered when text is received from the clipboard.
    - Parameter: `ClipboardTextReceivedReturnType` - Contains the clipboard text data.
    - Usage: Use this event to process or display clipboard text within your app.
12. OnWriteAccessRequested
    - Description: Triggered when write access is requested by the app.
    - Parameter: `WriteAccessRequestedReturnType` - Contains information about the access request.
    - Usage: Handle user responses to access requests, allowing for custom behavior based on user input.
13. OnContactRequested
    - Description: Triggered when a contact request is made by the app.
    - Parameter: `ContactRequestedReturnType` - Contains information about the contact request.
    - Usage: Manage interactions related to contact requests, such as displaying contact information.
14. OnBiometricManagerUpdated
    - Description: Triggered when there are updates related to biometric management.
    - Usage: Use this event to refresh biometric-related UI elements or settings.
15. OnBiometricAuthRequested
    - Description: Triggered when biometric authentication is requested by the app.
    - Parameter: `BiometricAuthRequestedReturnType` - Contains information about the authentication request.
    - Usage: Handle biometric authentication processes and responses from users.
16. OnBiometricTokenUpdated
    - Description: Triggered when a biometric token is updated.
    - Parameter: `BiometricTokenUpdatedReturnType` - Contains information about the updated token.
    - Usage: Manage any changes needed in your app based on biometric token updates.

Example Usage

Here's an example demonstrating how to utilize several functions from the Telegram Web App class within a Unity script:

```
using UTeleApp;
using UnityEngine;

public class TelegramIntegration : MonoBehaviour
{
    void Start()
    {
        // Marking as ready
        TelegramWebApp.Ready();

        // Displaying an alert
        TelegramWebApp.ShowAlert("Welcome to our Telegram Web App!");

        // Setting up main button
        TelegramWebApp.ShowMainButton();
        TelegramWebApp.SetMainButtonText("Click Me");

        // Checking platform version
        if (TelegramWebApp.IsVersionAtLeast("1.0"))
        {
            Debug.Log("You are using a compatible version of Telegram Web App.");
        }

        // Opening an external link
        TelegramWebApp.OpenLink("https://example.com");

        // Sharing media to story
        string mediaUrl = "https://example.com/media.jpg";
        TelegramWebApp.ShareToStory(mediaUrl);
    }

    void Update()
    {
        // Example of handling events (pseudo-code)
        if (Input.GetKeyDown(KeyCode.Space)) // Assuming space triggers main
button click
        {
            OnMainButtonClick();
        }
    }

    private void OnMainButtonClick()
    {
```

```
        Debug.Log("Main Button Clicked!");
        // Additional logic here
    }
}
```

Events Example Usage

Here's an example demonstrating how to utilize several functions from the Telegram Web App class within a Unity script:

```
using UTeleApp;
using UnityEngine;

public class TelegramIntegration : MonoBehaviour
{
    void Start()
    {
        // Subscribe to events
        TelegramWebApp.OnThemeChanged += HandleThemeChanged;
        TelegramWebApp.OnViewportChanged += HandleViewportChanged;
        TelegramWebApp.OnMainButtonClicked += HandleMainButtonClick;
        TelegramWebApp.OnSecondaryButtonClicked +=
HandleSecondaryButtonClick;
        TelegramWebApp.OnBackButtonClicked += HandleBackButtonClick;
        TelegramWebApp.OnSettingsButtonClicked += HandleSettingsButtonClick;
        TelegramWebApp.OnInvoiceClosed += HandleInvoiceClosed;
        TelegramWebApp.OnPopupClosed += HandlePopupClosed;
        TelegramWebApp.OnQrTextReceived += HandleQrTextReceived;
        TelegramWebApp.OnClipboardTextReceived +=
HandleClipboardTextReceived;
        // Add other subscriptions as needed
    }

    private void HandleThemeChanged()
    {
        Debug.Log("Theme has changed!");
        // Update UI elements based on new theme
    }

    private void HandleViewportChanged(ViewportChangedReturnType
viewportInfo)
    {
        Debug.Log("Viewport has changed!");
        // Adjust layouts or UI components based on viewportInfo
    }

    private void HandleMainButtonClick()
```

```
    {
        Debug.Log("Main Button Clicked!");
        // Add logic for main button click
    }

    private void HandleSecondaryButtonClick()
    {
        Debug.Log("Secondary Button Clicked!");
        // Add logic for secondary button click
    }

    private void HandleBackButtonClick()
    {
        Debug.Log("Back Button Clicked!");
        // Add logic for back button click
    }

    private void HandleSettingsButtonClick()
    {
        Debug.Log("Settings Button Clicked!");
        // Add logic for settings button click
    }

    private void HandleInvoiceClosed(InvoiceClosedReturnType invoiceInfo)
    {
        Debug.Log("Invoice Closed!");
        // Process invoice closure
    }

    private void HandlePopupClosed(PopUpClosedReturnType popupInfo)
    {
        Debug.Log("Popup Closed!");
        // Manage UI states after popup closure
    }

    private void HandleQrTextReceived(QrTextReceivedReturnType qrData)
    {
        Debug.Log($"QR Text Received: {qrData.Text}");
        // Process scanned QR text
    }

    private void HandleClipboardTextReceived(ClipboardTextReceivedReturnType
clipboardData)
    {
        Debug.Log($"Clipboard Text Received: {clipboardData.Text}");
        // Process clipboard text
    }
}
```

Telegram Mini App Payment Integration Guide

Overview

This guide explains how to implement Telegram payments in a Unity-based Mini App using the provided PaymentDemoController.cs example.

Prerequisites

1. A Telegram Bot Token (obtained from @BotFather), replace **your_bot_token** in PaymentDemoController.cs

2. UTeleApp plugin integrated into your Unity project

Payment Flow

```
A[User clicks Pay Button] --> B[Generate Invoice Link]
B --> C[Open Invoice in Telegram]
C --> D[User Reviews Payment]
D --> E[Pre-checkout Query]
E --> F[Answer Pre-checkout Query]
F --> G[Payment Processing]
G --> H[Receive Success/Failure]
```

Security Considerations

⚠️ Important Security Note:

The example code processes payments on the client side for demonstration purposes.

In a production environment, payment processing should be handled by a secure backend server. The client app should only do `openInvoiceLink` and process the payment result.

Helps

If you encounter any issues, please reach omninteractive@gmail.com