DREAMOS

from MICHSKY

Documentation - v1.0.0b

Scroll down for more

[For an updated and better experience, click here to use online documentation.]

Content

1. Frequently Asked Questions (FAQ)

• Does it support URP/HDRP rendering?

Yes, except for one thing. Blur shader only works with the standard pipeline, it's not compatible with URP and HDRP. All of the other features are compatible with every pipelines.

• Does it support the new input system?

Not fully. This asset is using the old input system, but you can also use the whole thing with the new input system by changing the active input handling to **'Both'** in player settings.

• I can't modify the object, it doesn't let me change them or revert back. Why?

Almost all of the elements are managed by **Theme Manager**. Some of the values in the manager are universal, which means it'll affect any object that has the **'Theme Manager Element'** component. If you don't want that, you can delete the component from specific elements. It'll take its own unique values as soon as deleting the component.

• What platforms can I use it for?

Tested for **Windows**, **WebGL** and **macOS**. The other platforms are not fully supported. Mobile support is coming in the future. Console support with virtual cursor is in development.

• Is this package compatible with Modern UI Pack?

Not fully. Some scripts may conflict, so I wouldn't suggest using DreamOS with Modern UI Pack.

• Are you going to support and add new stuff to the package?

Of course! You can find the roadmap by clicking here.

• I need help / I have a problem, what can I do?

If you can't find a solution for your problem in this doc, **contact me!** I'd gladly help to solve your issue. Click here for contact links.

2. Quick Start

First of all, thanks for purchasing the package! If you need some help to get started, this is the right place.

2.1. How To Use

If you won't use the pack in world space, it is recommended to use the demo scenes as a starting point. Everything you need is in the 'Canvas' object, so you can copy that object and then paste into your scene.

If you're going to use the pack in world space, then you can create the necessary resources from **Tools > DreamOS > Create World Space Resources**. It'll basically create everything you need for a world space scene, including the player controller. To replace the player controller with yours, you can delete the 'Player Controller' object and then assign the main camera object to **World Space Manager > Main Camera**.

Some of the UI objects are using the **Canvas Group** component (mostly windows and panels). You'll have to set their alpha value to 1 in order to see them, and set 0 to make them invisible. If you want to see a specific object/window while editing, you can simply set its alpha value to 1 to make it visible. You can do the opposite to make it invisible.

You can manage the content or add new content by using **Tools > DreamOS** menu and managers. You can see those managers by expanding **Canvas > Managers**. Each app and system feature has its own manager, you can customize or change their settings as you want.

2.2. Theme Manager

Do you want to change the appearance of the entire UI at the same time? Well, we got you covered. Theme Manager will basically change **every** single element, meaning that you won't have to change stuff one by one.

You can open the window by clicking **Tools > DreamOS > Show Theme Manager** to open it up. You can now expand the categories and start to change values. If you want, you can also add the **'Theme Manager Element'** component to any of your **Image** or **Text** objects to change their values via Theme Manager.

Update Values

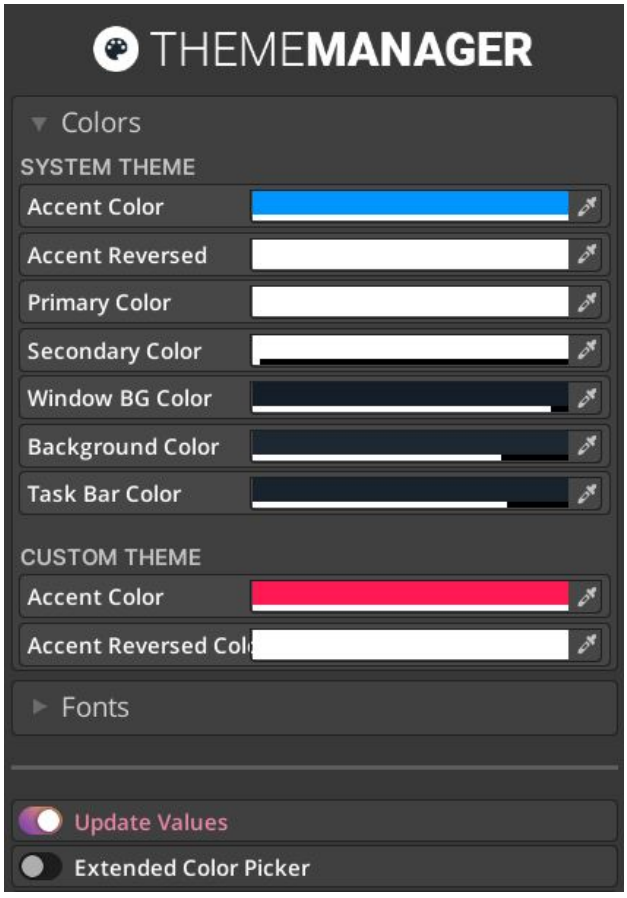
While this option is checked, UI Manager will be updating UI elements dynamically. If not, you won't be able to see any changes until you hit play (runtime). You can turn this off to gain more performance on the editor, just don't forget to enable it while changing stuff. This feature is disabled in build mode, so it won't have any effect on builds.

Extended Color Picker

If you want to see a more detailed color picker, enable this. This will be adding a hex code and an alpha slider right next to the color picker.

UI Manager Hints

If you want to see some tips about the manager, then you can enable this.



Note that Theme Manager values are universal and will affect any object that contains a Theme Manager component.

3. UI Elements

There are lots of UI elements in DreamOS. While some of them are custom made, the others are developed through the default Unity UI elements. In this section, you'll learn more about the custom ones.

3.1. Changing Content

In most cases, you can change the content from the inspector. For example, instead of trying to find the text object, you can simply change the button title by using the **Button Manager** via inspector. This will save some time while changing complex elements. If you want, you can disable this feature in most cases and manually change it.

3.2. Editing UI Elements

Every UI object was made with native Unity UI, so you can customize or change them as much as you want. All of the objects are properly categorized, so if you want to change a specific thing, just search for it via hierarchy.

3.3. UI Elements

There are some custom elements that are not included in the default Unity UI. In this section, you can learn more things about them.

3.3.1. Buttons

There's no usage difference, you can use DreamOS buttons just like the default one. Icon and text variables can be changed via Button Manager. If you don't want to change them by using Button Manager, you can enable **'Use Custom Content'**. For scripting, you can still continue to use **UnityEngine.UI.Button** class.

3.3.2. Context Menu

Simple yet functional pop-up menu which opens with a right click. It only requires a manager (Context Menu Manager) to work, which was already created in the ready to use scene(s).

To use it, you can simply add **'Context Menu Content'** to any of your UI object, assign **'Button Item'** (Context Menu Button) and **'Context Manager'** fields.

Items
You can add context items to this list, change its name, icon or add events. Currently, only button is supported, but more item types are planned for the future releases.

API / Scripting
```
using Michsky.DreamOS; // Namespace required

public ContextMenuContent myContextMenu;

void YourFunction()
{
    // Adding a new button
    myContextMenu.CreateNewButton("Title", spriteVariable);
}
```

3.3.3. Horizontal Selector
Basically, think of this thing as a dropdown, but the navigation is managed by only with next and previous controls. In my opinion, it fits better than dropdown in most cases.

Items
You can add horizontal selector items to this list. If you wish, you can add functions to each item as well. As long as 'Indicator' is enabled, each button will generate an indicator item at runtime.

Saving
You can save the last selected value by checking this box. Note that every selector should has its own unique **Selector Tag** value. Otherwise, there might be conflict(s) between selectors.

API / Scripting
It can be used with OnClick or you can call it from your script. **HorizontalSelector.ForwardClick();** or **HorizontalSelector.PreviousClick();** is what you're looking for.

```
using Michsky.DreamOS; // Namespace required

public HorizontalSelector mySelector; // Your selector variable

void YourFunction()
{
    // Creating a new item
    mySelector.CreateNewItem("Item Title");
    mySelector.SetupSelector();

    // Creating items within a loop
    for (int i = 0; i < yourIndexOrVariable; ++i)
    {
        mySelector.CreateNewItemFast("Item Title");
    }

    // Adding a new dynamic event
    mySelector.selectorEvent.AddListener(YourEventHere);
}
```

3.3.4. Input Field

Input fields in DreamOS are basically the same as the default TMP Input Field. The only difference is the animation. Other than that, you can use them just like the default one. For scripting, you can reference **'TMP_InputField'** and use its functions.

3.3.5. Modal Window

In order to change Modal Window content, you just need to change some values via **Modal Window Manager** (which is attached to the modal object).

At runtime, your window will be changing depending on the values. If you want to change those values manually, you can enable **'Use Custom Content'**. You can call the window via OnClick or within your script.

```
public ModalWindowManager myModalWindow; // Your window variable

void YourFunction()
{
   myModalWindow.icon = spriteVariable; // Change icon
   myModalWindow.titleText = "New Title"; // Change title
   myModalWindow.descriptionText = "Description"; // Change desc
   myModalWindow.UpdateUI(); // Update UI
   myModalWindow.OpenWindow(); // Open window
   myModalWindow.CloseWindow(); // Close window
   myModalWindow.AnimateWindow(); // Close/Open window auto
}
```

3.3.6. Slider

DreamOS is using the native slider that comes with Unity UI, but we've added some new cool features into it. For scripting, you can use **UnityEngine.UI.Slider**.

Saving

You can save the last selected value by checking this box. Note that every selector should has its own unique **'Slider Tag'** value. Otherwise, there might be conflict(s) between sliders.

3.3.7. Switch

Basically, think of this thing as a toggle, but 'modernized'.

Saving

You can save the last selected value by checking the **'Save Value'** box. Note that every switch should has its own unique **'Switch Tag'** value in order to save correctly.

API / Scripting

```
public SwitchManager mySwitch; // your switch variable
mySwitch.AnimateSwitch(); // set on or off
```

3.3.8. Native UI Elements

DreamOS is using the native Unity UI and its elements. You can use your own UI tools or third party tools with it without any problem. For more information about Unity UI, I'd suggest reading the official documentation or watching tutorials: https://docs.unity3d.com/2017.3/Documentation/Manual/UISystem.html

4. Apps & Windows

DreamOS features a set of apps and windows. In this section, you'll learn how to create new apps, delete or change the existing ones.

4.1. Creating New Apps

To create a new app, you can simply duplicate one of the existing app. Here's a step-by-step guide:

1. Select **Canvas > Main Screen > Apps & Windows > [Your Choice]** and duplicate the object.

2. After duplicating, you'll see a component called **'Window Manager'**. It is recommended to delete all **OnEnable** and **OnExit** events after duplicating the object.

3. **(Optional)** You can also duplicate the taskbar/desktop button and assign it into the window manager component. Later on, you can link the new app with your button and call **'WindowManager > OpenWindow'** function. If you won't use this feature, leave **'Taskbar Button'** field null.

4. **(Optional)** If you want to add new panels for the app you've just created, you can use **'Window Panel Manager'**. You'll see some button and panel objects attached to it, you can delete them or add new panels as well. After deleting, make sure to delete those objects from the scene to increase the performance.

If you want, you can also watch this quick tutorial-ish video to learn how to create a new app: https://youtu.be/ils-keY3a7A

4.2. Deleting Apps

To delete an existing app, you can select and delete all objects belonging to the app. Here's a step-by-step guide:

1. Select **Canvas > Main Screen > Apps & Windows > [Your App]** and delete the object.

2. **(Optional)** Select **Canvas > Main Screen > Desktop List > [Your App]** and delete the object.

3. **(Optional)** Select **Canvas > Main Screen > Task Bar > Shortcut List > [Your App]** and delete the object.

4.3. Messaging

Functional messaging app which supports getting and sending messages at runtime.

Workflow

Messaging app depends on its manager and scriptable objects. In order to add a new chat, you have to create a new chat asset from **Assets > Create > DreamOS > New Messaging Chat**. You can now select **Canvas > Managers > Messaging** and add hit add a new chat, and then assign the asset you've created. Each item has its own unique title and person data. All of the messages in the chat asset will be generated at runtime.

API / Scripting

```
public MessagingManager messagingApp;

void YourFunction()
{
   // Instantiate message at runtime and set parent
   messagingApp.CreateMessage(parentTransform, "Message here", "12:00
AM");
   // Instantiate individual message at runtime and set parent
   messagingApp.CreateIndividualMessage(parentTransform, "Message
here", "12:00 AM");
}
```

4.4. Music Player

Functional music player app which supports essential player features, including playlists, shuffle, repeat and so on.

Workflow

Music Player app depends on its manager and scriptable objects. It supports multiple playlists which you can manage by using the manager. You can add new music by using **Tools > DreamOS > Show Music Playlists**. All of the playlists and their items will be generated at runtime. Items can be edited by selecting and opening the main prefab from **Managers > Music Player > Resources** tab.

API / Scripting

```
using Michsky.DreamOS; // Namespace required

public MusicPlayerManager musicPlayer; // Your mp variable

void YourFunction()
{
    musicPlayer.PlayMusic(); // Start playback
    musicPlayer.PauseMusic(); // Pause playback
    musicPlayer.StopMusic(); // Stop playback
    musicPlayer.MuteMusic(); // Mute or unmute audio
    musicPlayer.NextTitle(); // Next track from the current
playlist
    musicPlayer.PrevTitle(); // Next track from the current
playlist
    musicPlayer.PlayCustomMusic(index); // Play specific music
from the current playlist
    musicPlayer.PlayCustomClip(audioClip, spriteCover, "name",
"author"); // Play specific music
    musicPlayer.shuffle = true; // Change shuffle state
    musicPlayer.repeat = false; // Change repeat state
}
```

4.5. Notepad

Functional notepad app which supports essential notepad features, including editing and creating.

Workflow

Notepad app depends on its manager and scriptable objects. Notes can be added or edited by using **Tools > DreamOS > Show Notepad Library**. All of the note items will be generated at runtime.

API / Scripting

```
using Michsky.DreamOS; // Namespace required

public NotepadManager notepadApp;

void YourFunction()
{
   notepadApp.OpenNote(index); // Show specific note from the library
   notepadApp.OpenCustomNote("title", "note content"); // Show specific note
}
```

4.6. Photo Gallery

Functional photo gallery app which supports showing the images in the library or streaming images from URL.

Workflow

Photo Gallery app depends on its manager and scriptable objects. Photos can be added or edited by using **Tools > DreamOS > Show Photo Library**. All of the items will be generated at runtime. You can also get the image from a specific URL by adding **'Retrieve From URL'** component.

API / Scripting

```
using Michsky.DreamOS; // Namespace required
```

```
public PhotoGalleryManager photoApp;

void YourFunction()
{
   photoApp.OpenCustomPicture(index); // Open specific photo
from the library
   photoApp.OpenCustomSprite(spriteVariable, "title",
"description"); // Open specific photo
}
```

4.7. Reminder

Functional alarm/reminder app integrated with Global Time events.

Workflow

Reminder app depends on its manager and reminder item component. All reminder items assigned to the manager component will be updated at runtime. Currently, it is limited to a total of 4 reminders, this will change in the future releases.

4.8. Settings

Functional settings app to change system and user settings.

Workflow

Settings app depends on various managers. User Manager to change the user data, Theme Manager to change the theme, Reboot Manager to format the system.

API / Scripting

```
using Michsky.DreamOS; // Namespace required

public SettingsManager settingsApp;
```

```
void YourFunction()
{
   settingsApp.SetWallpaper(imageVariable); // Set the wallpaper
from an image
   settingsApp.SetWallpaperDirectly(spriteVariable); // Set the
wallpaper by using a sprite
   settingsApp.SelectSystemTheme(); // Select the default theme
   settingsApp.SelectCustomTheme(); // Select the custom theme
   settingsApp.WipeUserData(); // Wipe the data and then reboot
   settingsApp.DeleteUserData(); // Wipe the user data
}
```

4.9. Video Player

Functional video player app which supports essential player features, including playback, streaming, loop, seeking and so on.

Workflow

Video Player app depends on its manager and scriptable objects. You can add new clips by using **Tools > DreamOS > Show Video Library**. You can also stream videos by enabling **'Play From URL'** and setting a proper url. All of the playlist items will be generated at runtime. Items can be edited by selecting and opening the main prefab from **Managers > Video Player > Resources** tab.

API / Scripting

```
using Michsky.DreamOS; // Namespace required

public VideoPlayer videoPlayer;

void YourFunction()
{
   videoPlayer.Play(); // Play the current clip
   videoPlayer.Pause(); // Pause the current clip
   videoPlayer.SeekForward(); // Seek forward depending on
```

```
seekTime
    videoPlayer.SeekBackward(); // Seek backward depending on
seekTime
    videoPlayer.seekTime = 10f; // Change the seek time value
    videoPlayer.loop = true; // Enable or disable looping
    videoPlayer.IncreasePlaybackSpeed(); // Increase playback
speed by double
    videoPlayer.DecreasePlaybackSpeed(); // Decrease playback
speed by double
    videoPlayer.PlayVideoClip(videoClipVariable, "title"); //
Play specific video
    videoPlayer.PlayCustomVideo(index); // Play specific video
from the library
    videoPlayer.PlayVideoURL("url address"); // Stream specific
video from URL
}
```

4.10. Web Browser

Functional web browser app which supports basic web browser features. Integrated with network system and downloads.

Workflow

Web Browser app depends on its manager, its library and network manager. You can add new websites or downloadable files by using **Tools > DreamOS > Show Web Library**. All website items require a prefab to create the page content and an unique URL address. You can exclude ‘www.’ from the address as Web Browser checks for ‘www.’ on search.

You can specify a file type for downloadable files. After selecting a file type, assign / enter a value in the appropriate field. ‘None’ file type doesn’t require any assignment. You can later on download the file by calling its name via **Web Browser Manager** or **Download Helper**. **Download Helper** component is used in the example website(s), as it doesn’t require any assignment in the scene. You can simply call this line within your script or button to download a file: **DownloadHelper.DownloadFile(“downloadable file name”);**

API / Scripting

```
using Michsky.DreamOS; // Namespace required

public WebBrowserManager webBrowser;

void YourFunction()
{
   webBrowser.OpenHome(); // Go to the homepage
   webBrowser.OpenPage("page url"); // Go to a specific page
   webBrowser.GoBack(); // Previous page
   webBrowser.GoForward(); // Next page
   webBrowser.Refresh(); // Refresh the current page
   webBrowser.SetFavorite(); // Add page to favorites
   webBrowser.DownloadFile("File Name"); // Download a specific
file from Web Library
   webBrowser.DeleteDownloadFile("File Name"); // Delete a
specific downloaded file
}
```

5. Audio Manager

A simple component responsible for managing the master volume of the system. It only requires a slider and an audio mixer. If you want to change the volume manually, you can use something like this:
AudioManager.VolumeSetMaster(0.5f);

6. Boot Manager

A component responsible for managing the system power. It requires a boot screen/animator. To process the events, InvokeEvents() needs to be called. You can also add your own events to **'Events After Booting'** area. If you don't want to see the boot screen, you can set **'Boot Time'** to 0.

7. Date & Time Manager

A component responsible for managing the system time/date. It doesn't require any external resources and it's a simulated time solution. It supports timed events, changing the time speed and saving. In order to fetch the data, you can add **'Date and Clock'** component to any UI object.

API / Scripting

```
using Michsky.DreamOS; // Namespace required

public GlobalTime timeManager;

void YourFunction()
{
   timeManager.currentSecond = 30f; // Change second (0-60)
   timeManager.currentMinute = 25; // Change minute (0-60)
   timeManager.currentHour = 10; // Change hour (0-24)
   timeManager.currentDay = 15; // Change day (1-30)
   timeManager.currentMonth = 6; // Change month (1-12)
   timeManager.currentYear = 2020; // Change year
   timeManager.UpdateTimeData(); // Update the stored data
   timeManager.enableAmPm = true; // Enable AM/PM label
}
```

8. Events

There are a few event components that you can use in various situations. You can basically add any of these components to get started.

8.1. Double Click Event

Process double click events depending on time factor. It is mainly used for desktop buttons, but it also can be used for any other purposes by adding **'Double Click'** component.

8.2. Hold Key Event

Process hold & release events depending on the hotkey. It can be used by adding **'Hold Key'** component.

8.3. Press Key Event

Process key events depending on the hotkey. It can be used by adding **'PressKey'** component.

8.4. Timed Event

Process timed events depending on the timer. It can be used by adding **'Timed'** component.

9. Network Manager

A component responsible for managing the system network. It requires a parent for listing the network items. It is integrated with Web Browser and Downloads, but can be also integrated with any external tools/objects.

API / Scripting

```
using Michsky.DreamOS; // Namespace required

public NetworkManager networkManager;

void YourFunction()
{
    networkManager.CreateNetwork("title", 2, "password"); // Create a network (title, signal power, password)
    networkManager.ListNetworks(); // Create or refresh network items
    networkManager.hasConnection = true; // Change network connection state
}
```

10. Notification Manager

A component responsible for managing the notifications. It requires a parent for listing the notification items and a notification button. Supports both pop-up and standard notifications. In order to create a new notification, you need **'Notification Creator'** component.

API / Scripting (Notification Creator)

```
using Michsky.DreamOS; // Namespace required

public NotificationCreator notificationCreator;

void YourFunction()
{
   notificationCreator.notificationTitle = "title"; // Set
notification title
   notificationCreator.notificationDescription = "description";
// Set notification description
   notificationCreator.popupDescription = "description"; // Set
popup notification description
   notificationCreator.notificationIcon = spriteVariable; // Set
notification icon
   notificationCreator.enablePopupNotification = true; // Enable
or disable popup notification
   notificationCreator.popupDescription = "description"; // Set
popup notification description
   notificationCreator.enableButtonIcon = true;
   notificationCreator.CreateButton("title", spriteVariable,
null, true); // Create buttons (title, icon, button event, close
on click)
   notificationCreator.CreateNotification(); // Create a
notification depending on variables
}
```

11. User Manager

A component responsible for managing and creating the user. It is connected with boot screen, setup screen, lock screen and desktop screen. While you can let users to create their own profile, you can also disable user creating and set a pre-made user by enabling **'Disable User Creating'** using the manager.

11.1. Getting User Data

If you want to fetch the user data, you can use this component. It doesn't require any external resources. You can enable **'Get Information At Start'** to fetch the data at start.

API / Scripting

```
using Michsky.DreamOS; // Namespace required

public GetUserInfo userInfo;

void YourFunction()
{
    userInfo.getInformation = Reference.FIRST_NAME; // Reference
    userInfo.GetInformation(); // Fetch the data
}
```

11.2. Setting User Data

You can use this component to change/update the user data. If you're going to use it within your script, use 'Change', not 'Set' prefix. For the other cases (such as buttons), use 'SetFirstName' etc.

API / Scripting

```
using Michsky.DreamOS; // Namespace required

public SetUserInfo setUserInfo;

void YourFunction()
```

```
{
   setUserInfo.ChangeFirstName("name"); // Change first name
   setUserInfo.ChangeLastName("name"); // Change last name
   setUserInfo.ChangePassword("password"); // Change password
   setUserInfo.ChangeProfilePicture(spriteVariable); // Change
profile picture
}
```

12. Widget Manager

A component responsible for creating widget items. It only requires a library, which can be found on **Tools > DreamOS > Show Widget Library**. You can add or create your own widgets via Widget Library. Here's a step-by-step guide:

1. Select one of the existing widget prefab (by using Widget Library).
2. Duplicate the prefab, delete/change its content and rename it as you want.
3. Add a new widget by using Widget Library and change the title etc.
4. Assign the duplicated prefab to **'Widget Prefab'**.

That's it! You'll now see your widget at runtime by using the **'Widgets'** app. If you want, you can change some settings on your widget item.

13. World Space Manager

A component responsible for handling 3D/world space canvas rendering. It requires lots of resources, such as a main camera, an external camera for OS Canvas, a render texture and an enter mount. Instead of assigning/creating those things one by one, you can simply use **'Tools > DreamOS > Create World Space Resources'**.

World Space Resources includes a player controller. To get rid of it and implement with your current character controller, here's a step-by-step guide:

1. Select **World Space Resources > Player Controller** and delete the object.

2. Add your main camera to **'Main Camera'** on World Space Manager.

3. In order to move our camera to the mount, we need to disable the camera control **On Enter** and then enable **On Exit**. This kind of depends on your camera controller, but you can disable the camera controller **On Enter**, disable the whole **Main Camera** object **On Enter End**, enable the camera object **On Exit**, and then lastly enable the camera controller **On Exit End**. This might sounds a bit confusing, but you'll understand the logic when looking at the example player controller.

4. World Space Resources comes with 3D objects. Those are only for Demonstration, you can delete them as you like. Just don't delete **'Glass OS Renderer'**, which you can find at **World Space Resources > Environment > Monitor**.

5. There's an object called **'Enter Mount'**. It's basically an object that helps make the transition between canvas and camera. You can change its position for your needs.

If you don't want to use floating icon, you can delete its resources and then leave **'Use Floating Icon'** field blank. You can also change the transition speed and some other settings via **World Space Manager**. It requires a collider in order to make the transition, which is attached to **World Space Manager**.

14. Window Manager

A component responsible for managing app windows. It is mostly used with **'Window Panel Manager'** component, which manages the in-app window / nav button states and allows creating multi-tabs. In order to add new in-app window, you can follow these steps:

1. Select an app and create a new panel item via **Window Panel Manager**.

2. Set an unique panel name, duplicate one of the existing panel object and button object, then assign those objects to **'Panel Object'** and **'Button Object'**. If you don't need a button, you can just assign an empty object.

3. You can now call your in-app window with its unique name by calling Window Panel Manager > Open Panel() > Panel Name.

4. If you're going to call it within your script: **WindowPanelManager.OpenPanel("Panel Name");**

API / Scripting

```
using Michsky.DreamOS; // Namespace required

public WindowManager windowManager;

void YourFunction()
{
    windowManager.OpenWindow(); // Open window
    windowManager.CloseWindow(); // Close window
    windowManager.MinimizeWindow(); // Minimize window
    windowManager.FullscreenWindow(); // Maximized window
    windowManager.FocusToWindow(); // Focus to window
    windowManager.NavDrawerAnimate(); // Open or minimize nav
drawer
}
```

15. Others

15.1. Editing Animations

You can manage or modify the animations by using the **‘Animator’** tab (Window > Animation > Animator). Click an object that has an animator component and you’ll see the animation states. Now, you can select an animation and change or edit them as you want. To edit a specific animation, you can select an object, click CTRL / CMD + 6 and then start tweaking by using the animation window.

15.2. Localization

Since DreamOS was built with the native Unity UI and TextMesh Pro, you can change the text and allow localization easily. However, some objects are managed by its managers (buttons, modal windows, etc). For most cases, you can enable **‘Use Custom Content’** via their managers to allow changing the values/strings manually. If you can’t enable localization for some objects, you can contact me for help. It is recommended using Unity Localization, as it supports smart strings.

16. Contact & Licence

Contact & socials:

E-mail

Discord

Twitter

YouTube

Website

If you have any problems, questions, suggestions or feedback, please feel free to contact me.

Licence