
**CONTENTS**

# INTRODUCTION

The system for managing the interface of virtual windows, folders and other elements of the operating system. This asset has a flexible architecture that can be integrated into your project, ranging from simulating an operating system to developing a file manager and cloud storage. The asset contains the UI package and the functionality of the main elements of interaction with the interface.

**Peculiarities:**

1. **Manage windows on the desktop**
   Resizing, moving windows within the limited space of the desktop, transferring focus to the selected window, collapsing, expanding to full screen, closing.

2. **Desktop**
   Files and folders can be dragged and dropped within the desktop along a given grid. This grid is flexibly configured and cell sizes are automatically calculated.

3. **Folder system**
   The folder system is designed in such a way that it stores information about all its ancestors and child elements that are contained within it. The folder contains a workspace where other folders/files can be placed. There is also a check to see if the folder moves into itself or into its child.

4. **Bottom taskbar**
   The taskbar displays open folders or files. On the panel, you can move the focus to the required window, as well as minimize / maximize the window.

5. **System Alerts**
   A flexible pop-up message constructor with the ability to send an event depending on which button the user presses in this window. For example, calling an event after pressing the "Confirm" button. Created pop-up messages in the constructor can be called from any part of the code.

6. **Central system manager**
   You probably won't need to search for the right script and try to figure out what it does. All basic settings can be made using a simple and flexible configurator of all the features of this asset in one place.

7. **Network communication and local data storage**
   It is very easy to upgrade this asset for networking or local data storage. You will need to convert the serializable classes to the json format, which already stores all the

information about folders, location, and so on. And work with this json file as you wish

8. **Basic set of UI elements**
   There is a stylish UI for the main elements of the operating system, which you can use in any of your projects.

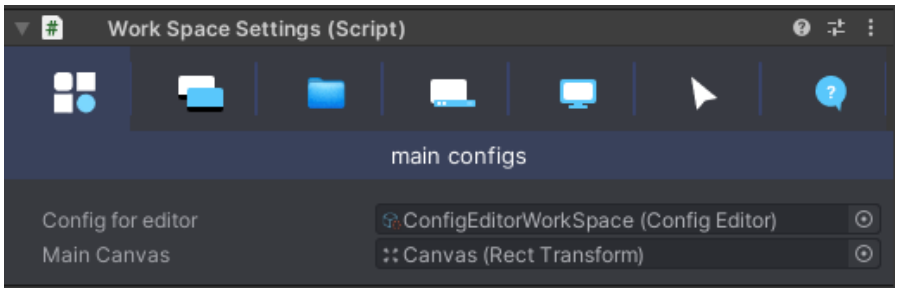
## DEMO SCENES

1. **DemoDesctop** – contains a demonstration of all the features of the asset.
2. **DemoWindowStyles** – demo scene with windows in different styles. Windows can be interacted with.
3. **DemoAlerts** – demonstration of pop-up messages and their variety.

## ASSET SETTING

The settings for all the features of the asset are made after adding the WorkSpaceSettings.cs script to your scene, which is also the workspace configurator.

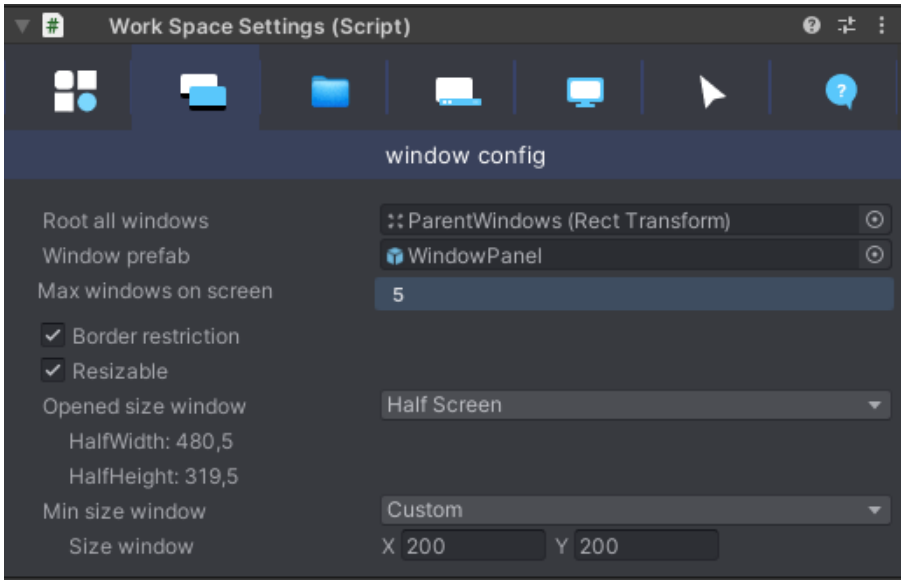
1. **Main settings**




Component inspector window

| | |
|---|---|
| **Config for editor** | Here we specify a link to the "ConfigEditorSpace" object. This is a ScriptableObject with all the basic settings for rendering the configurator panel. |
| **Main Canvas** | Specify the main Canvas of our workspace. |

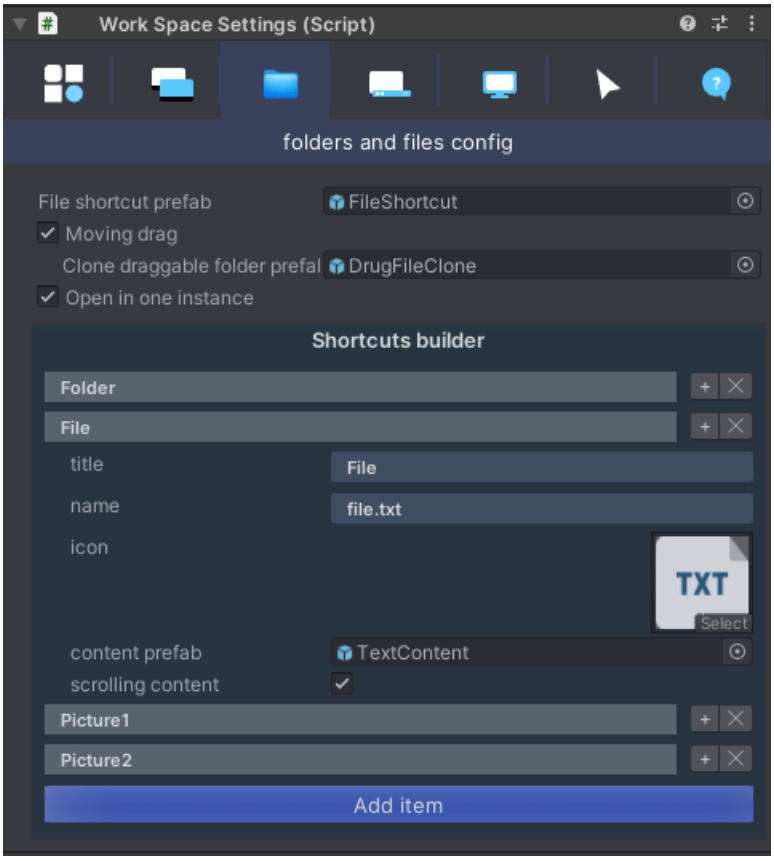
## 2. Window settings




Component inspector window

| | |
|---|---|
| **Root all windows** | The parent object that will contain all open windows. |
| **Window prefab** | The prefab of the window that will be created when opened. |
| **Max windows on screen** | The maximum number of windows that can be open at the same time. If the maximum number is exceeded, after opening the window, the very first one will be closed, and so on. |
| **Border restriction** | If you enable this mode, then moving windows off the screen will be impossible. |
| **Resizable** | If enabled, open windows can be resized. |
| **Opened size window** | Select the size of windows when opening. |
| **Min size window** | Select the minimum window size when resizing. |

## 3. Folder and file settings (shortcuts)




Component inspector window

| | |
|---|---|
| **File shortcut prefab** | A prefab of an icon (shortcut) that will be generated on the desktop. |
| **Moving drag** | If enabled, desktop icons/folders can be moved around. |
| **Clone draggable folder prefab** | A prefab of a draggable icon that will follow the mouse while dragging. |
| **Open in one instance** | If enabled, when you double-click on the icon, only one window will open, otherwise the windows will be duplicated. |
| **Shortcuts builder*** | Icon/badge constructor. This constructor creates a list of all icons that can be immediately added to the desktop from the inspector by pressing the “+” button next to the element name. |

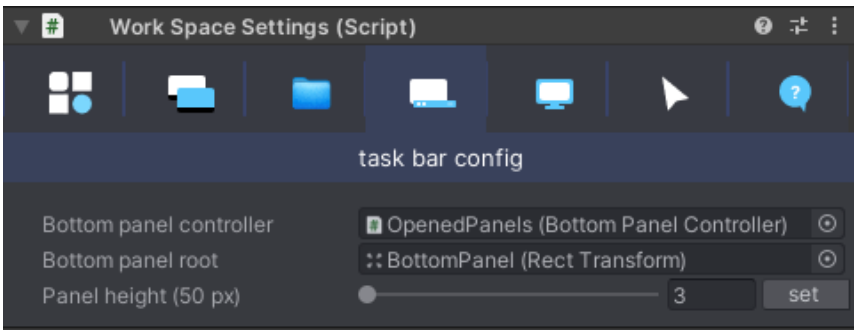
---

***Important!*** *Generate icons directly from the inspector or in your scripts with a method call* ***obj.GetComponent<GetDataFile>().Init(prop).*** *Since each icon has a unique hash code that is generated in the* ***Init****() method. This hash code is the shortcut identifier and is used in some checks. If you ignore this recommendation and place shortcuts in the unity editor, then the logic will be violated and, for example, it will not work out the algorithm that prohibits moving the folder to itself and to its child folders!*

---

In the icon builder, the "content prefab" field contains the main content prefab. See the demo for how it's done, and following this example, you can create your own prefabs, where you can place any content for the contents of a folder or file.

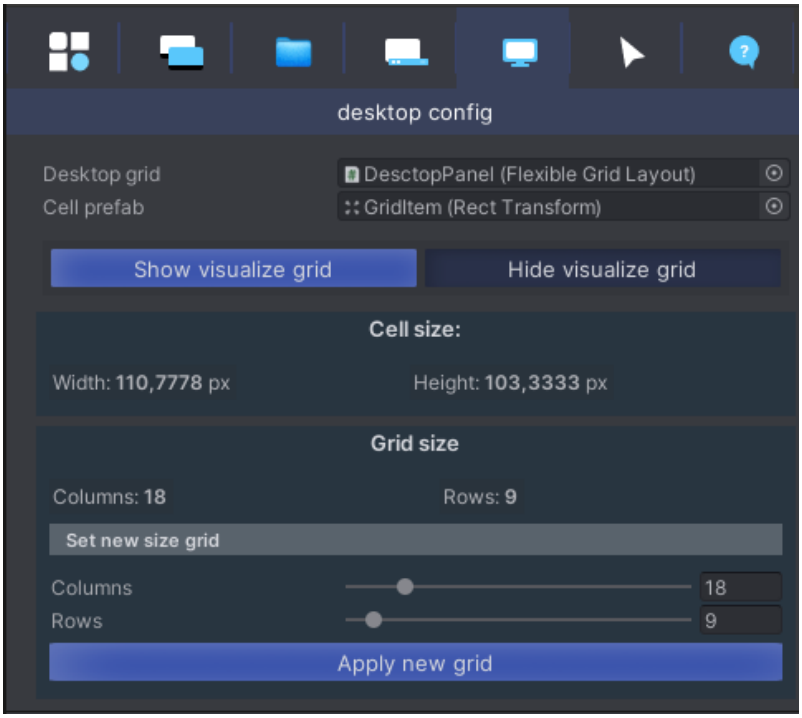
### 4. Button taskbar settings




Component inspector window

| | |
|---|---|
| **Bottom panel controller** | The lower taskbar controller object. |
| **Bottom panel root** | Transform of the parent object of the bottom bar, which defines the height of this bar. |
| **Panel height (n px)** | Indicates in brackets the current height of the bottom bar. And also here you can define a new value for the height of the bottom panel. For the changes to take effect, click the "Set" button, which is located to the right of the slider. |

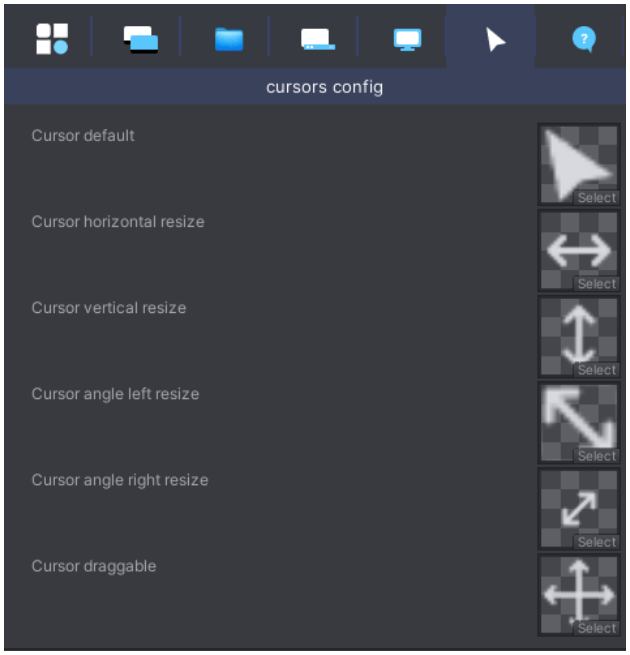
## 5. Desktop settings




Component inspector window

| | |
|---|---|
| **Desktop grid** | A Flexible Grid Layout that resides on the desktop grid's parent. |
| **Cell prefab** | The cell prefab that is generated when defining the desktop grid on which the icons (shortcuts) will move. |
| **Buttons «Show visualize grid» and «Hide visualize grid»** | The names of these buttons speak for themselves - disable and enable the visualization of the grid on the desktop. This is useful when debugging and customizing the desktop. |
| **Cell size** | This displays the current cell size in the desktop grid in real time. |
| **Grid size** | The number of columns and rows of the grid is displayed. |
| **Set new size grid** | Set a new grid size. After specifying the new values, don't forget to apply the changes by clicking "Apply new grid". |

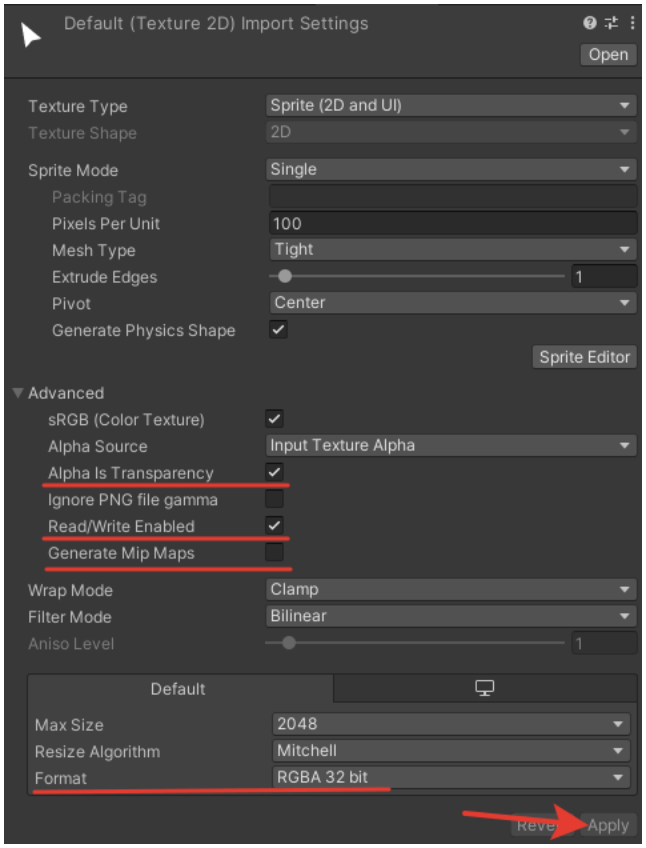
## 6. Cursor settings



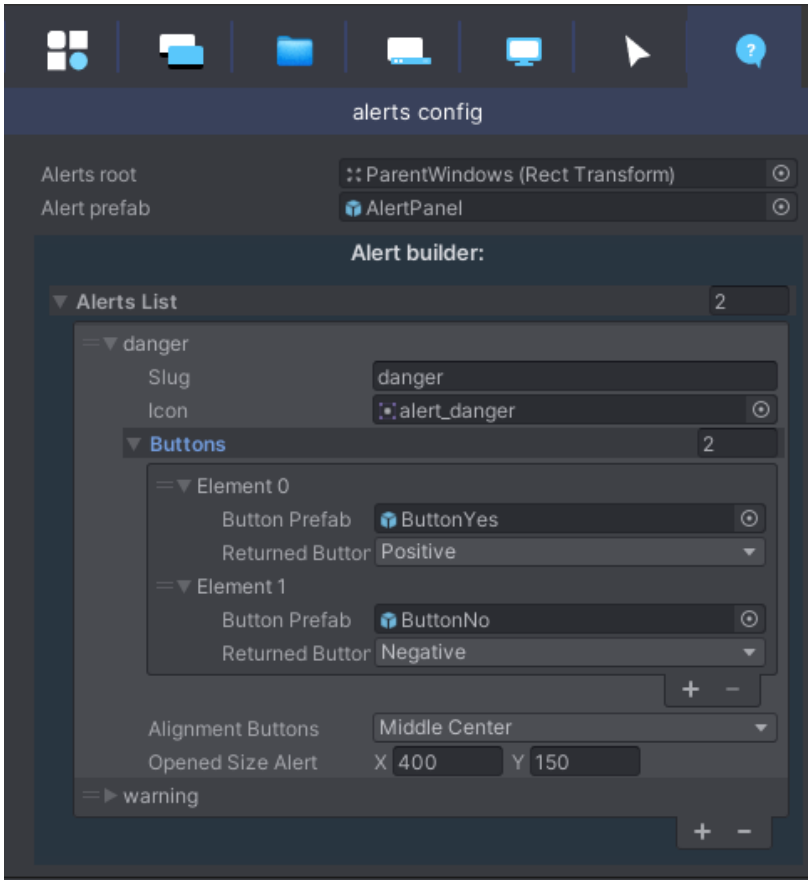

In this panel, specify the textures of your cursors. If the field is left blank, the standard cursor of your operating system will be used.

If you are using your own textures for cursors, then it is important to properly set up the texture itself in unity so that there is no warning in the console:


I recommend using the settings as in the screenshot:



## 7. Alert settings




Component inspector window

| | |
|---|---|
| **Alerts root** | The parent object where pop-up messages will be generated. |
| **Alert prefab** | Popup prefab. |
| **Alert builder** | The popup constructor, which is a list.<br>More details below. |

Alert builder

| | |
|---|---|
| **Slug** | The key identifier of the popup message, which we will use from scripts. So make sure you come up with a unique slug. |
| **Icon** | Popup message icon, it usually displays the message type. |
| **Buttons** | The list of buttons that will be displayed in the popup panel. The button acts as a prefab. Note that you also need to specify the return type for the button after the button is clicked. Negative – button type that returns the execution of the method if the user decides negatively. Positive - positive response. None - returns nothing. |

**How to call a popup message?**

The message is called anywhere in your script:

```
WorkSpaceSettings.Instance.Alert("question", "Question alert", "Do you like this asset?", QuestionYes, QuestionNo);
```

Where, the first parameter is slug, a unique key that is used to select from the list of all alerts created through the constructor.

The second parameter is title, the title.

The third is content text, the body of the message.

The fourth parameter is the name of the method that will be called upon a positive user response.

The fifth parameter is the name of the method that will be called upon a negative user response.

If you need to call only the method, with a negative response, but there is no positive response, then specify null as the fourth parameter.

## SUPPORT

If you are confused by something, you can write to denis@proger.xyz and we will do our best to contact you immediately. We hope you enjoy our asset and good luck in creating your games