UnityLibrary - Documentation

https://github.com/Luca00IT/UnityLibrary/tree/main

Description

UnityLibrary contains many different libraries for Unity to speed up the process of developing your own video game. Later you can consult the documentation in order to understand how to implement these libraries in your projects and use the functions

Prerequisites

- Unity Engine v2021.3.4f1 or higher
- Visual Studio 2022 Community + "Game Development with Unity" extension
- Import the library at the beginning of each script:

```
// using libraryname
```

FadeTools

The FadeTools library allows you to use functions that can create and edit a RawImage by creating a "FadeIn" and "FadeOut" effect across the entire screen. You can use this library when, for example, you need this effect for loading a layer or other occasions when you will need this effect.

Import the library

```
using FadeTools;
```

Initialize your variables

```
RawImage rawImage;
float duration;
```

Create RawImage

Creates a black-colored image that covers the entire **Canvas** GameObject. Make sure to create one before using this method.

```
FadeTools.Fade.CreateRawImage();
```

Fade In

Makes the newly created image darker and darker in a time determined by the duration variable.

```
FadeTools.Fade.FadeIn(rawImage, duration);
```

Fade Out

Makes the newly create image lighter in a time determined by the duration variable.

```
FadeTools.Fade.FadeOut(rawImage, duration);
```

ActionTools

The ActionTools library allows you to speed up the first person action process by making two functions that get all the interactive tags of your game and checks if your character is pointing at one of these GameObjects.

Import the library

```
using ActionTools;
```

Initialize your variables

```
int n; // this is the size of your array of tags
int tagIndex; // this will store the index of the interactive object found
float raycastDistance; // this is the distance of the raycast
string[] tags = new string[n]; // this is the array of interactive tags of your project
```

Add interactable tags

```
void Start()
{
    ActionTools.Action.AddInteractableTags(tags)
}
```

You can also call this function in the Awake() function

```
void Awake()
{
    ActionTools.Action.AddInteractableTags(tags)
}
```

Is object interactable

```
void Update()
{
    if(ActionTools.Action.IsObjectInteractable(tagIndex, raycastDistance))
    {
        switch(tagIndex)
        {
            case 0:
                // do stuff
            break;
        }
    }
}
```

License