D.A. BUTTON

MANUAL FOR DEVELOPERS
1.0.0

Introduction

1. **D.A. Button** is a button with support for **multiple target graphics** and the ability to create **animation** for each separate target graphic.

   With D.A. Button, you can:

   1. Set up **color**, **scale**, and **position** animations for each individual target graphic;
   2. Create animations that will be executed on **click**, **hover** and when **disabling interaction**;
   3. Create **looped** animations.

2. If you encounter any errors while working with the asset, please write me about it at provided contacts.
   I typically respond quickly to messages, offer assistance on an individual basis, and address any identified bugs in the upcoming updates.
   You can leave comments about the features that you want to see in the asset - it's will also be considered.

   Discord Server: https://discord.com/invite/ZsnDffV5eE

   Telegram Group: https://t.me/da_assets_publisher

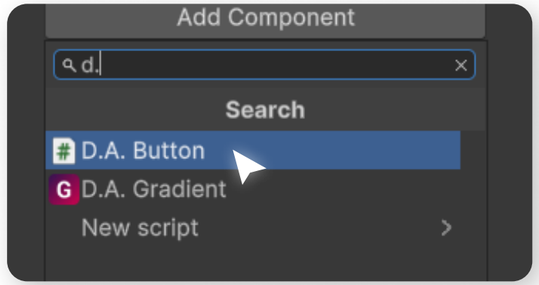
   Email Support: da.assets.publisher@gmail.com

   Website: https://da-assets.github.io/site/

3. You can earn a percentage from sales of this asset through the Unity Affiliate Program.
   If this interests you, please contact me via PM or email.

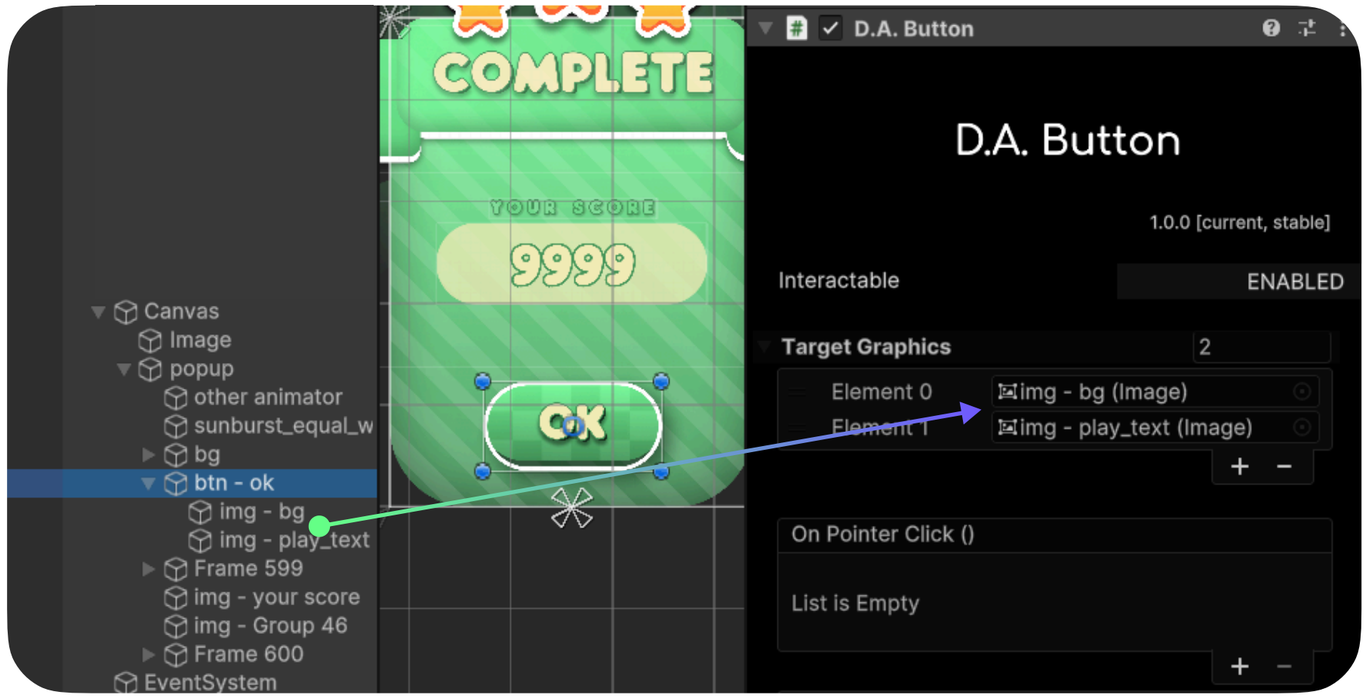
Scene

1. Import the asset using the Unity Package Manager.

2. 

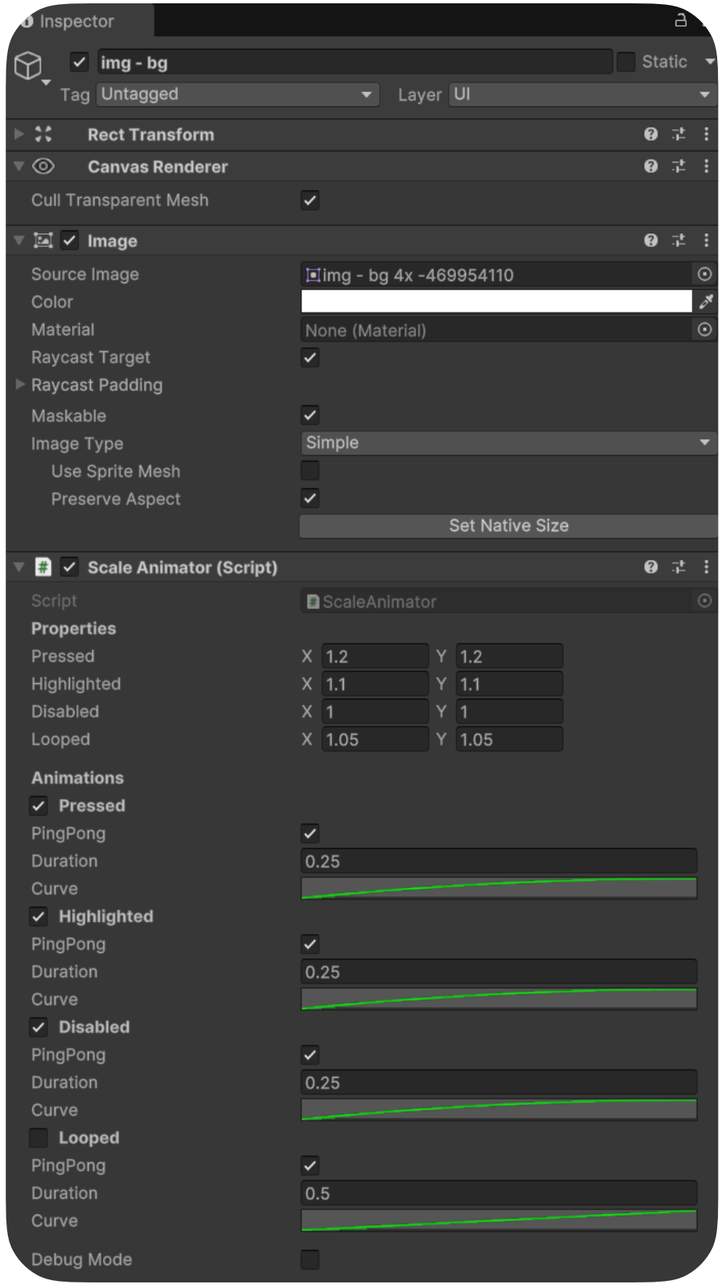
Create an empty GameObject on the scene and add the "**D.A. Button**" script to it.
You will see the asset's interface.

3. Serialize the GameObjects with Graphic that you want to animate in the TargetGraphics array (e.g., Image or Text). All TargetGraphics inside the parent GameObject with the D.A. Button script will respond to pointer events such as IPointerClickHandler, IPointerEnterHandler, and IPointerExitHandler.



Screenshot of the asset's demo scene, which includes a popup with background and button animations.

Animations



1. To animate a specific Graphic, you can use one of the following animator scripts, each serving a specific function:
   - **ColorAnimator** – Animates color changes.
   - **PositionAnimator** – Animates movement.
   - **RotationAnimator** – Animates rotation.
   - **ScaleAnimator** – Animates scaling.
   - **SpriteAnimator** – Animates sprite transitions.

2. Animator settings are divided into two sections: **Properties** and **Animations**.

3. **Properties** defines animation values applied to the **Graphic** over time based on the **Curve** from the **Animations** section.

4. **Animations** are triggered by specific UI events:
   - **Pressed** – Triggered by "IPointerClickHandler".
   - **Highlighted** – Triggered by "IPointerEnterHandler".
   - **Disabled** – Activated when "**Interactable**" is set to "**false**".
   - **Looped** – If enabled, a looping animation starts automatically at game start.

5. Each animation has configurable parameters:

   Duration – Defines the time for the animation to play when an event occurs and the time required for the **Graphic** to return to its initial state.

   Curve – Specifies the animation curve for smooth transitions.

   PingPong – If **enabled**, the animation plays twice:
   Once when the event occurs, and again in reverse when returning to the initial state.
   If **disabled**, the curve plays once. Ensure the **Y** values at the start and end match to prevent abrupt transitions.

Infinity rotation

1. To set up infinite rotation for an object, add a RotationAnimator to it, set the "Looped" parameter to 360 for right rotation or -360 for left rotation.

2. Enable "Looped" in Animations, set the desired curve (e.g., Linear for uniform rotation), and specify the Duration for a single full rotation.

3. If another event is triggered during the playback of a looped animation, the looped animation will pause, the event animation (e.g., a click) will play, and then the looped animation will resume.