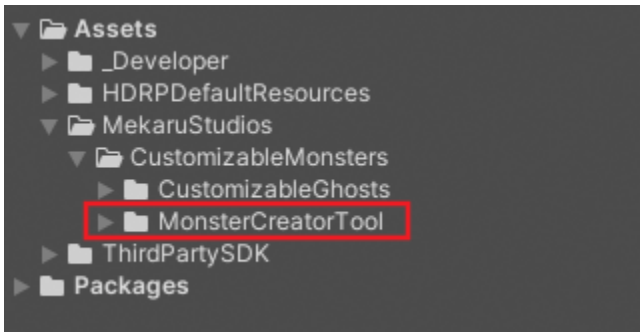
Monster Creator Tool

BASICS

This section of the document provides a straightforward guide on getting started with the Monster Creator Tool. You'll learn how to effortlessly initiate the tool, personalize your monster, and conveniently save it as a prefab, all without the need for extensive exploration of the tool's intricate settings and customizations.

IMPORT AND UPDATE

The Monster Creator Tool is a complementary inclusion in every packet from the Customizable Monsters series. Moreover, our commitment to enhancing your monster-building experience is unwavering, as we regularly introduce fresh cosmetics, materials, and face textures through consistent updates.
To ensure you have access to the latest and greatest version, follow these steps after downloading any package from the "Customizable Monsters" series: Obtain the Monster Creator Tool separately as a distinct package and seamlessly integrate it into your project. This way, you'll always stay on the cutting edge of monster customization.



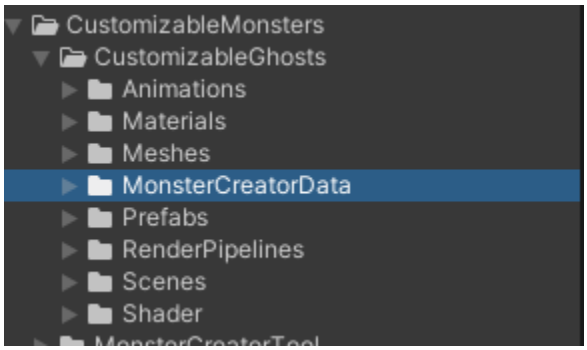
REMOVING MONSTER CREATOR

While some developers may opt not to utilize the Monster Creator Tool and refrain from integrating any third-party scripts into their projects, rest assured that our monsters remain entirely customizable through manual means. The Creator Tool simply streamlines and expedites this process, enhancing efficiency.

If you find the need to remove the Monster Creator Tool from your project, here's how to go about it:

1. Begin by navigating to the following path within your project directory: *Assets/MekaruStudios/CustomizableMonsters/MonsterCreatorTool*
2. In this directory, locate the "MonsterCreatorTool" folder and remove it from your project entirely.
3. Next, delve into any of the customizable monster packets you've imported into your project.
4. Within each of these packets, you will find a "MonsterCreatorData" folder. Select and delete this folder from within each packet to ensure complete removal.

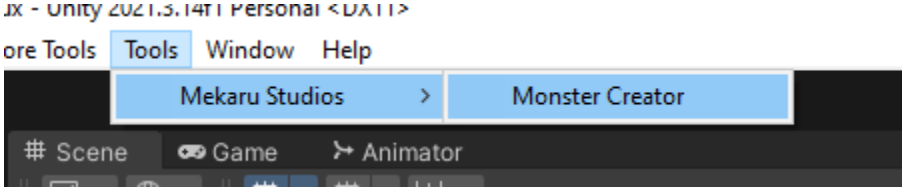
By following these steps, you will successfully remove the Monster Creator Tool and its associated data from your project, should you choose to do so.



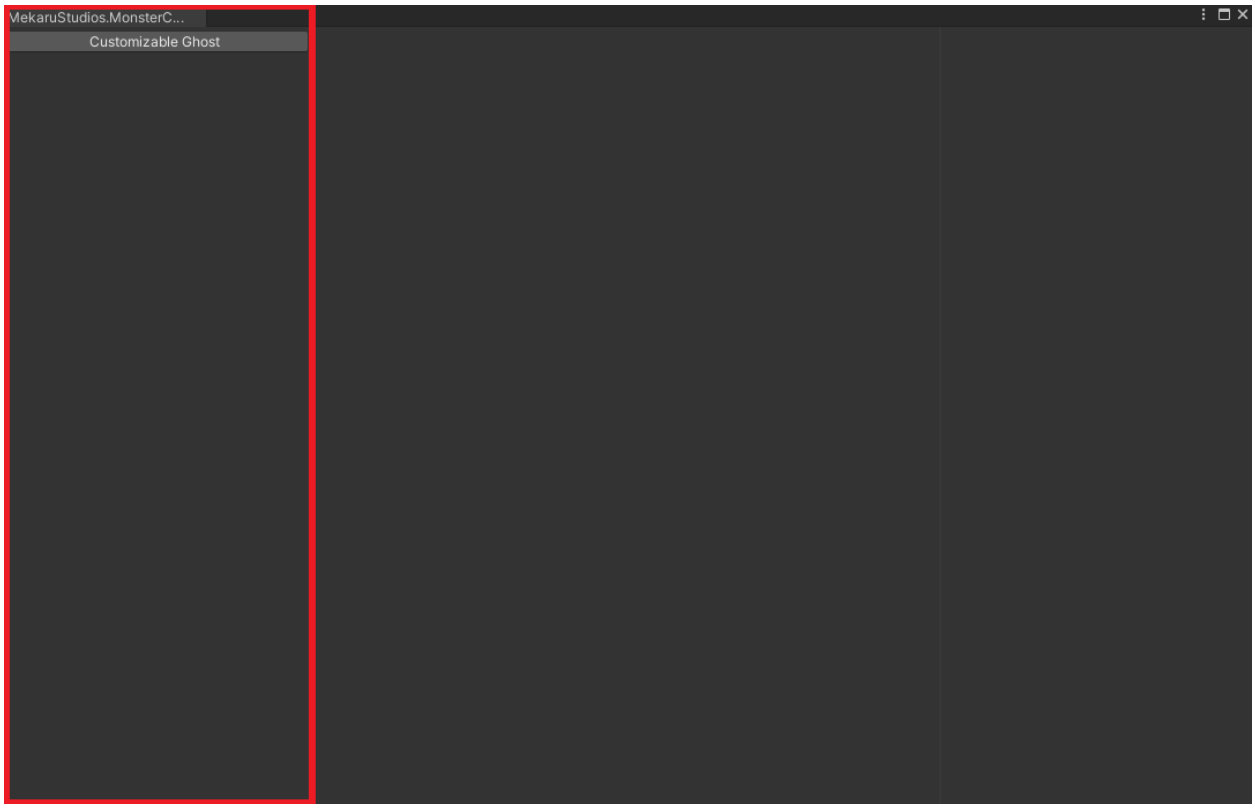
START USING TO MONSTER CREATOR TOOL

At this part we assume you got at least one monster packet from Customizable Monsters series, and downloaded the MonsterCreatorTool.

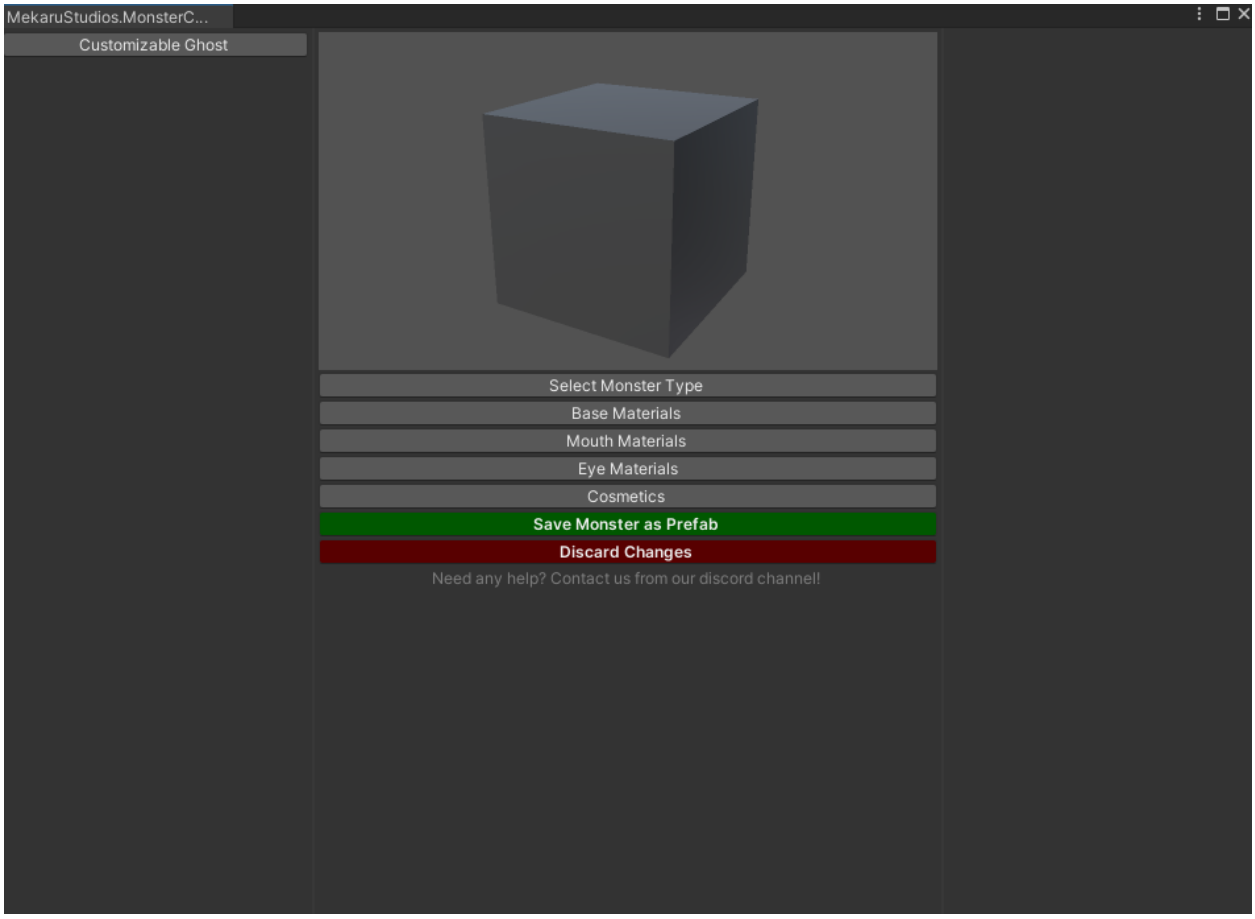
1. First, navigate to *Tools>Mekaru Studios>Monster Creator*

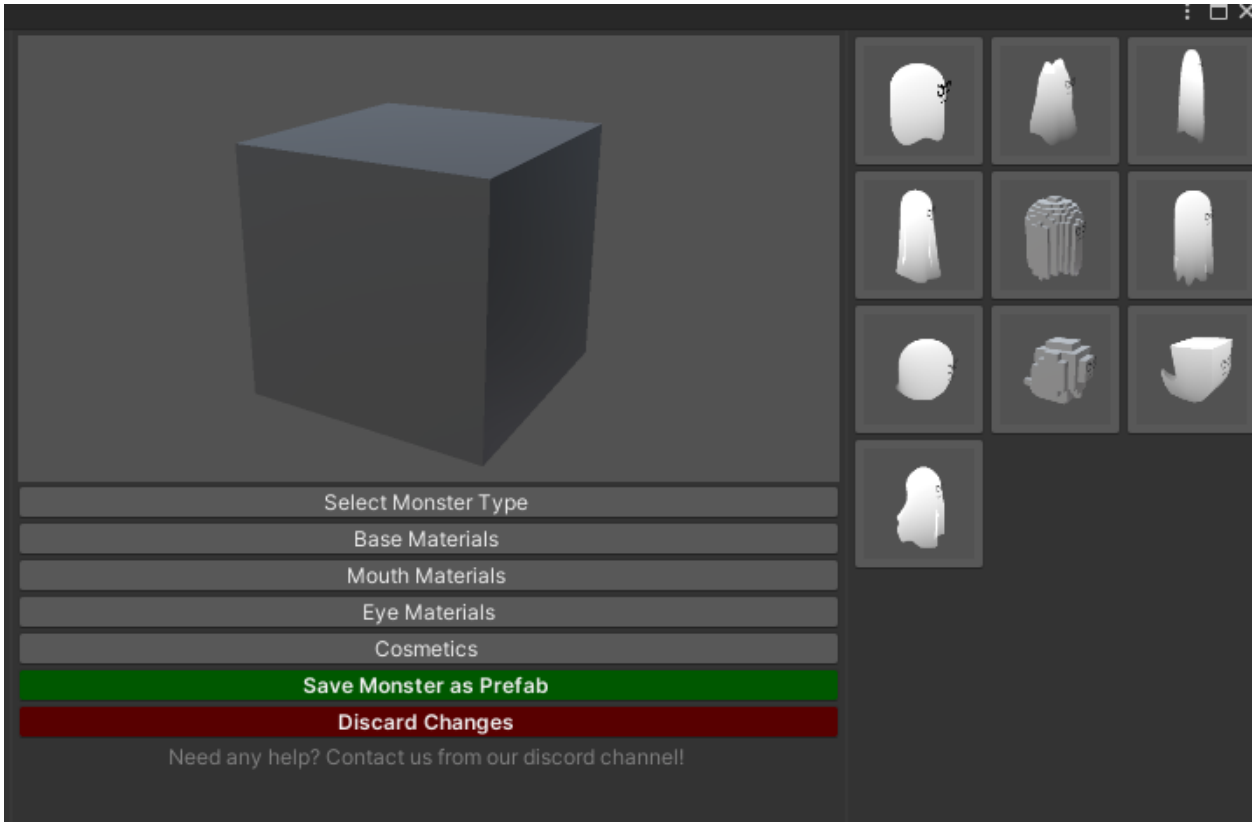


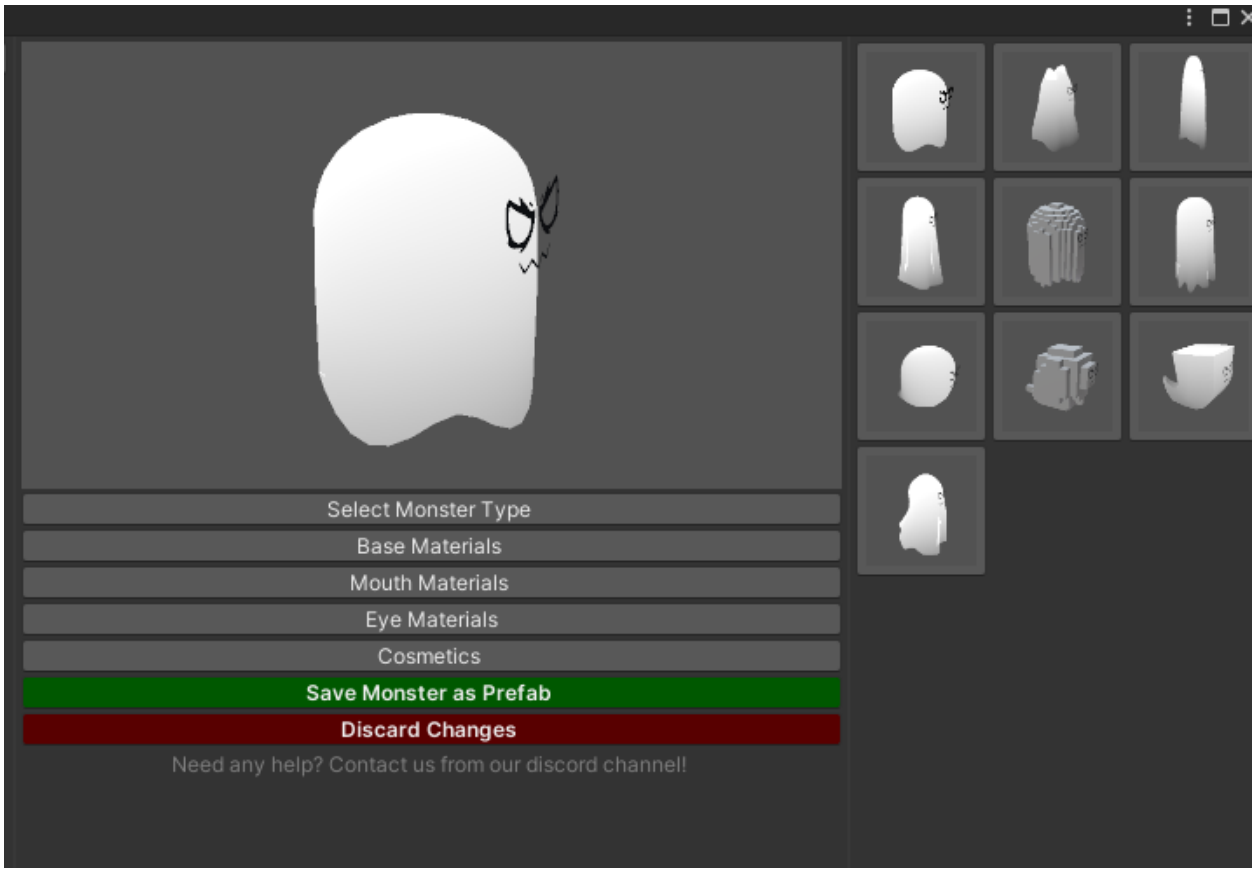
2. You will face a window separated into three sections. Left pane will display the packets which are compatible with MonsterCreatorTool. Click one of those buttons to start customizing a monster from the selected packet.



3. As default, the creator tool will preview a standard cube at the middle pane. First, select monster type as the base of the customization


MekaruStudios.MonsterC...
Customizable Ghost
Select Monster Type
Base Materials
Mouth Materials
Eye Materials
Cosmetics
Save Monster as Prefab
Discard Changes
Need any help? Contact us from our discord channel!

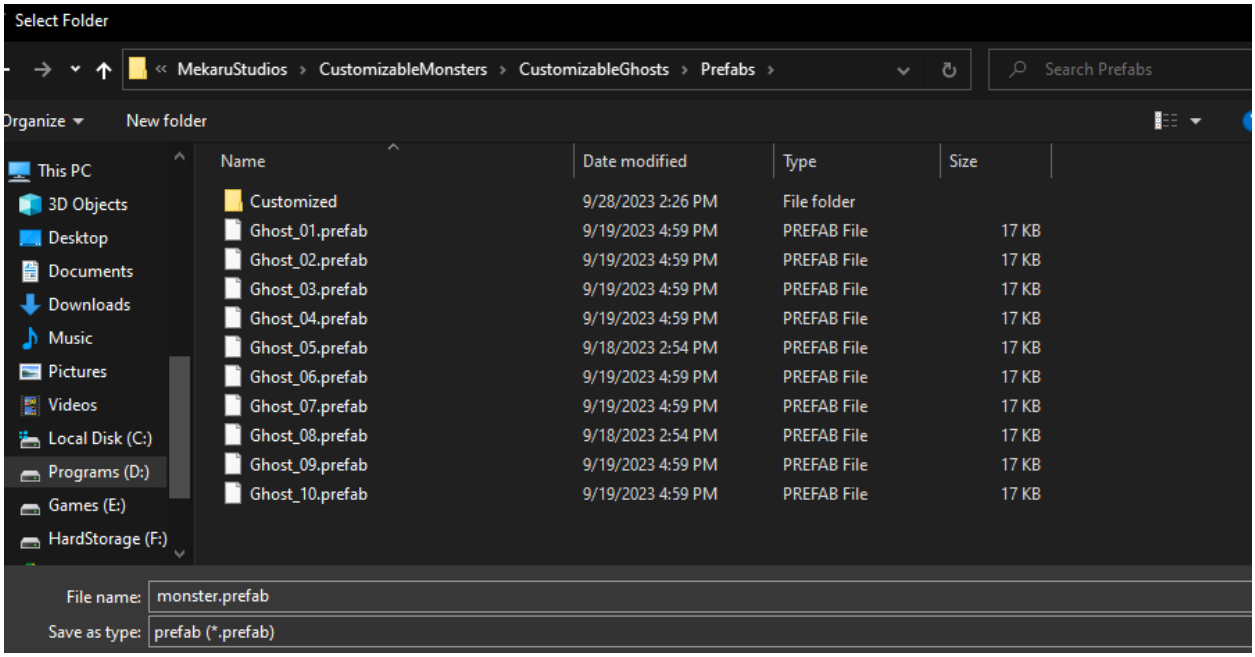

Select Monster Type
Base Materials
Mouth Materials
Eye Materials
Cosmetics
Save Monster as Prefab
Discard Changes
Need any help? Contact us from our discord channel!


Select Monster Type
Base Materials
Mouth Materials
Eye Materials
Cosmetics
Save Monster as Prefab
Discard Changes
Need any help? Contact us from our discord channel!

4. After selecting the monster type, now you modify the base material, face texture and cosmetics of the monster.

5. Once you've selected the customizations you desire, you can save the monster as a prefab by simply clicking the "Save Monster as Prefab" button.

6. Then select a **relative path** to your project, for saving your new monster.



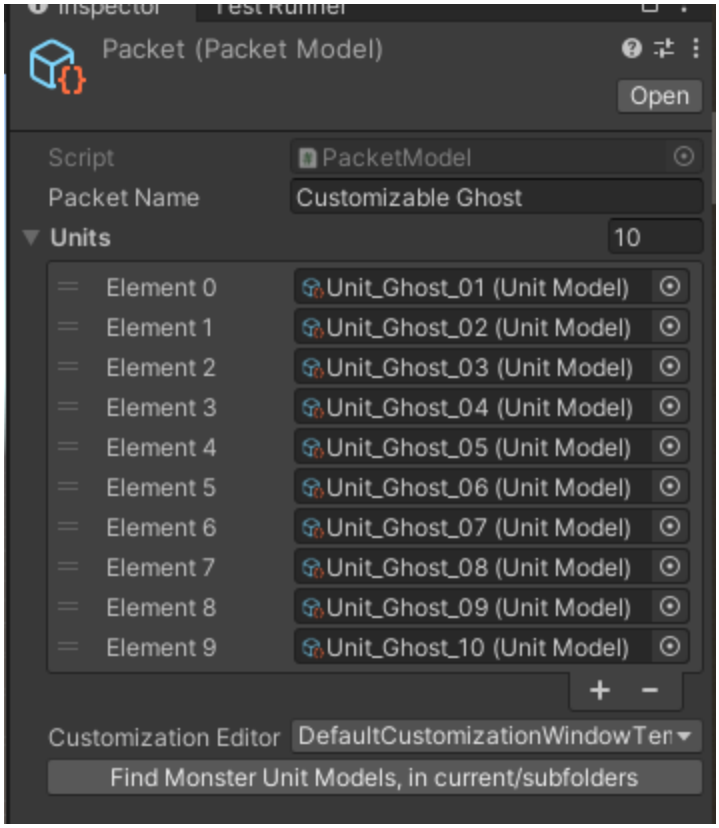
Congratulations, now you can drag and drop your monster into the scene and start to use it!

CUSTOMIZING THE TOOL

PACKET MODEL

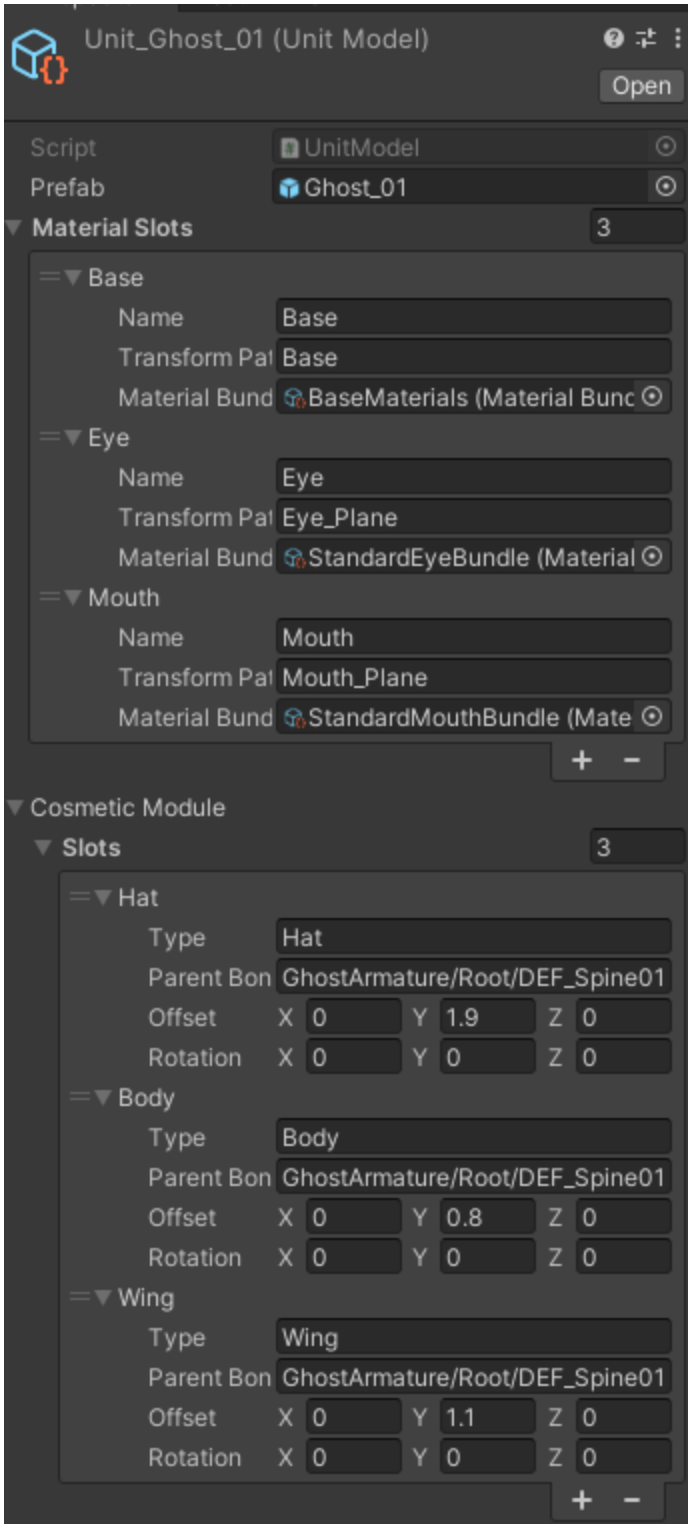
The MonsterCreatorTool primarily stores and interprets its data as scriptable objects. Our initial data set is encapsulated within the "Packet Model" scriptable object. All scriptable objects related to monster packets are neatly organized within the MonsterCreatorData folder. It's important to note that every packet compatible with the MonsterCreatorTool will include this MonsterCreatorData folder for seamless integration and customization.

Within the PacketModel scriptable object, you will find crucial information including the monster types (referred to as monster units), the packet's name, and its associated editor type. This data collectively forms the foundation for customizing and managing your monster packets effectively.



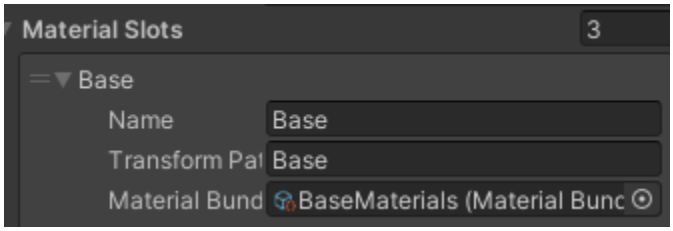
UNIT

We've chosen to refer to this data as "unit" rather than "monster type" because some of our packages offer the flexibility of interchangeable parts. Each unit has the potential to represent either the entire monster model or a distinct component of the monster, accommodating a wide range of customization possibilities.


Unit_Ghost_01 (Unit Model)
Open
Script
UnitModel
Prefab
Ghost_01
Material Slots
3
Base
Name
Base
Transform Pat
Base
Material Bund
BaseMaterials (Material Bunc
Eye
Name
Eye
Transform Pat
Eye_Plane
Material Bund
StandardEyeBundle (Material
Mouth
Name
Mouth
Transform Pat
Mouth_Plane
Material Bund
StandardMouthBundle (Mate
Cosmetic Module
Slots
3
Hat
Type
Hat
Parent Bon
GhostArmature/Root/DEF_Spine01
Offset
X 0
Y 1.9
Z 0
Rotation
X 0
Y 0
Z 0
Body
Type
Body
Parent Bon
GhostArmature/Root/DEF_Spine01
Offset
X 0
Y 0.8
Z 0
Rotation
X 0
Y 0
Z 0
Wing
Type
Wing
Parent Bon
GhostArmature/Root/DEF_Spine01
Offset
X 0
Y 1.1
Z 0
Rotation
X 0
Y 0
Z 0

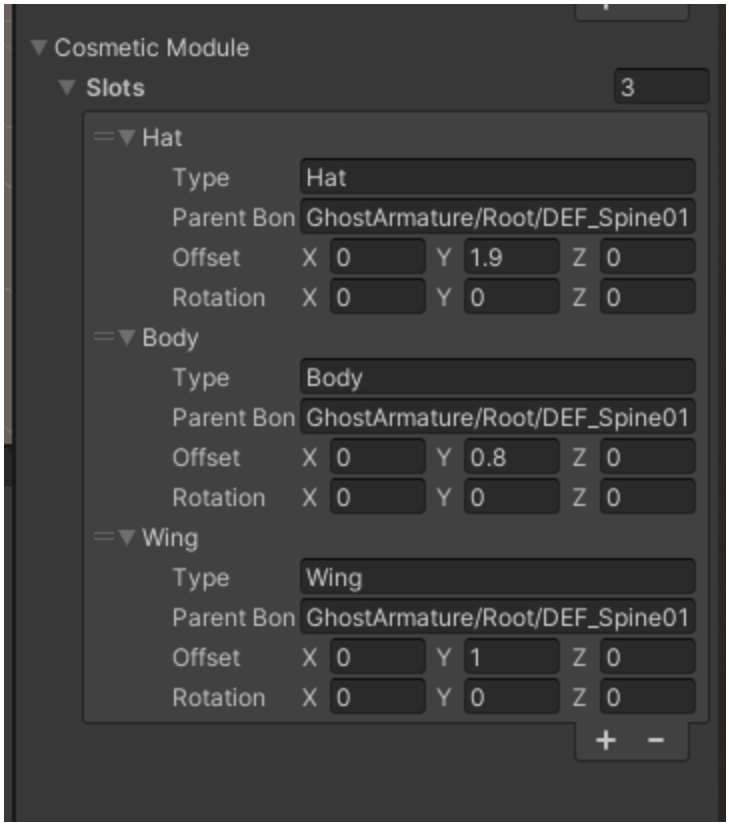
Unit data contains prefab of the unit, material slots, and cosmetic slots.

For adding a new material, add a new material slot.



A Material slot comprises three essential properties:
Name: This property signifies the name of the material slot type.
Transform Path: It should point to the path of an object containing a mesh renderer within the prefab. If the object you wish to reference is a child object, you can simply specify the path using '/' similar to folder paths. This allows for precise mapping and configuration of material slots in your project. *Example: Face/Eye_Plane*
Material Bundle: The Material bundle serves as a straightforward container for materials, and the choice to employ a scriptable object for this purpose is deliberate. In packets with multiple units, making modifications to a material within one unit would traditionally necessitate replicating those changes across all units, resulting in a laborious task. However, by utilizing a material bundle scriptable object, we streamline this process, alleviating the need for redundant work and ensuring consistent material management across all units.

The Cosmetic Module is the essential component responsible for managing all cosmetic-related tasks within the model unit. Within the Cosmetic Module, you'll find a collection of cosmetic types specifically designed to work seamlessly with the associated model unit.
For instance, let's consider a model unit that embodies the classic ghost shape. In its corresponding Cosmetic Module, there may be two designated slots, such as "Hat" and "Body." These slots enable the attachment of hat and body-type cosmetics to this particular model unit.
It's worth noting that model type names are predefined, and it's crucial to ensure that the type names match these predefined designations. However, the flexibility of the system also allows for the effortless addition of different types of cosmetics, making it a versatile and user-friendly tool for customization.

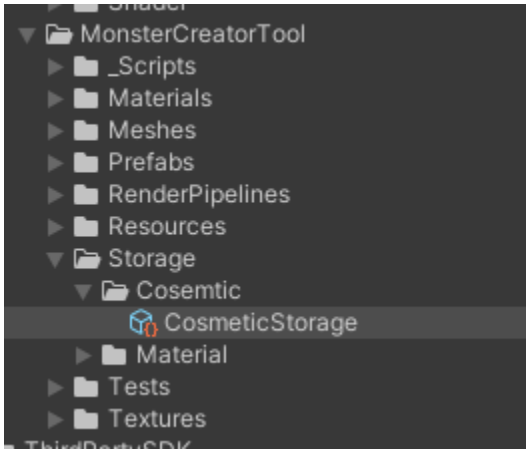


Once you've added a cosmetic type, the next step is to specify the parent bone path. When the tool places a cosmetic, it will search for this bone as a transform and designate it as the parent for the cosmetic.
Additionally, you can define an offset, which represents the positional adjustment for the cosmetic. The Rotation property allows you to set the cosmetic's orientation using vector3 values.
We value your feedback and customization needs. If you find that these options are insufficient for your project, don't hesitate to reach out to us via Discord or our email address. We are committed to providing ongoing updates and are open to expanding customization options based on your valuable input. Your satisfaction is our priority.

ADDING CUSTOM COSMETICS AND TYPES

All cosmetics assets stored in one scriptable object. We can add new cosmetics and cosmetic types into this Cosmetic Storage scriptable object.

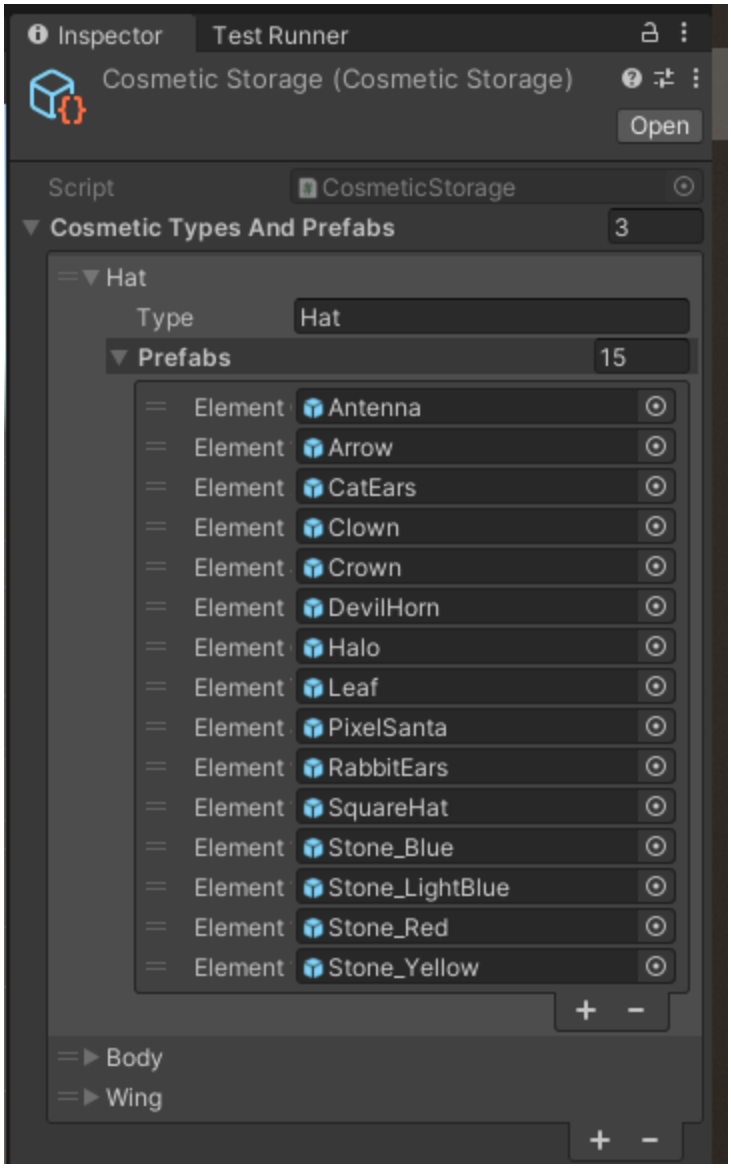
Cosmetic Storage, scriptable object located at *MonsterCreatorTool>Storage>Cosmetic>CosmeticStorage*



To effortlessly incorporate new cosmetics, follow these straightforward steps:

1. Access the Cosmetic Storage in the Inspector.
2. You have two options: a. Add a cosmetic prefab for an existing type. b. Create a new type, then add your cosmetic prefab under this newly defined type name.

This streamlined process empowers you to expand your cosmetic offerings with ease, providing you with the flexibility to continually enrich your project's customization options.



Furthermore, it's important to bear in mind that, following the addition of a custom cosmetic type, you must also incorporate this new type into the corresponding monster unit file. This step ensures that your custom cosmetic type is seamlessly integrated and accessible for use within the monster unit, allowing for comprehensive and personalized cosmetic customization.

ADVANCED

CUSTOM EDITORS

Certain packets may demand unique customization logic and specialized editors. Moreover, as you delve into future projects, you might find the need for a distinct style of editor. To accommodate these variations and foster adaptability, our system allows for the addition of new editors, giving developers the freedom to craft custom editors as needed.

By default, the tool relies on "DefaultCustomizationWindowTemplate.cs" to construct the central pane of the editor. This template serves as a starting point, enabling developers to tailor the editor interface to suit their specific requirements and design preferences.

The tool uses reflection to find editors which have [CustomizationEditor] attribute.

```
[CustomizationEditor]
yanard18
public class DefaultCustomizationWindowTemplate : IWindowTemplate
{
    Frequently called  0+1 usages  yanard18
    public IWindow Build()
    {
        IGUIComponent gui = new GUIComponent();
        gui = new PreviewEntityGUI(gui);
        gui = new MonsterBtn(gui);
        gui = new WindowLoaderBtn(gui, windowTypeName: "base-material", label: "Base Materials");
        gui = new WindowLoaderBtn(gui, windowTypeName: "mouth-material", label: "Mouth Materials");
        gui = new WindowLoaderBtn(gui, windowTypeName: "eye-material", label: "Eye Materials");
        gui = new WindowLoaderBtn(gui, windowTypeName: "cosmetic", label: "Cosmetics");
        gui = new SaveBtn(gui, new FileSaver());
        gui = new DiscardBtn(gui);
        gui = new FooterGUI(gui);

        return (Window)A.Window()
            .WithWidthPercentage(.50f)
            .WithComponent(gui); // WindowBuilder
    }

}
```

To guarantee that your template aligns with the necessary structure, it's crucial to integrate the "IWindowTemplate" interface into your class. This integration ensures that your custom editor template harmonizes with the tool's specifications, enabling seamless integration and robust functionality.

Within the Build method, the key step is to return an instance of IWindow. This means you will construct a new window in this method, allowing you to create a functional and customized editor experience tailored to your specific needs.

WINDOW

The Window class incorporates IGUIComponents within its structure, employing the decorator pattern instead of relying on a conventional list. This design choice provides a flexible and extensible approach to constructing windows. While we provide standard GUI components to represent common functionality, you also have the freedom to develop your custom GUIComponents to cater to your unique needs.
As demonstrated in the previous image, the process of building a window involves defining these GUI components and arranging them to wrap around each other. This modular approach empowers you to create versatile and tailored windows that suit your specific requirements.

SUPPORT AND COMMUNITY

➡️ Discord | We are always ready to help with any problem.
We value your feedback and are committed to continuously improving and expanding our product. With plans to introduce new animations, cosmetics, and special shaders, we aim to provide you with endless customization options to suit your unique needs. So you can join our discord channel, and help us to release new updates which fit your needs!

Also you can always contact us from: support@mekarustudios.com

Thanks For Choosing Us

Thank you for choosing Mekaru Studios for your recent purchase. We're thrilled to have you as our valued customer and want to express our gratitude for your trust in our products.

Thanks for Your Purchase: Your support means the world to us, and we hope our document exceeds your expectations.

Reach Out Anytime: If you have any questions, encounter issues, or need assistance with anything related to your purchase, please don't hesitate to reach out. Our dedicated support team is here to help you around the clock.

Exclusive Offers on Discord: Be sure to join our Discord community to stay updated on exclusive offers, promotions, and connect with other customers. It's a great way to be part of our growing community.
➡️ Discord

Visit Our Website: Explore our website for more information about our products and services. You might find other resources that can be helpful for your needs.
www.mekarustudios.com

Check Out Our Other Assets: In addition to the document you've purchased, we offer a range of other assets and resources that could be valuable to you. Feel free to explore our catalog and discover more.

- Slime Packet
- Pixel Forest

Thank you again for choosing Mekaru Studios. Your satisfaction is our top priority, and we look forward to serving you in the future.