Introduction

Welcome to RNGNeeds, your key to unlocking a world of possibility within Unity. We're thrilled to have you on board and deeply appreciate the trust you've placed in us by choosing our product. You've made an excellent choice and we're confident you'll find RNGNeeds to be an indispensable tool for your Unity projects.

Get the Plugin from Asset Store → | Getting Started | Join our Discord

Preface

Every game thrives on an element of surprise, that tantalizing unpredictability that hooks players in. But managing this randomness, these 'RNG needs', is often more art than science. With RNGNeeds, you have a Unity plugin that redefines randomness, transforming it from a wild variable into a finely-tuned instrument of creativity.

Beyond the dice rolls and item drops, RNGNeeds offers you a canvas of opportunity. Imagine dynamically adjusting the frequency of in-game storms as your player progresses through the storyline, modulating NPC behavior based on time of day, or making the battle more intense if the music volume is loud.

RNGNeeds puts you in the director's seat, allowing you to craft unique, immersive experiences that respond and adapt to your game world. Because RNG isn't just about luck - it's about creating living, breathing worlds where anything is possible.

What is RNGNeeds?

RNGNeeds is a powerful plugin that enables you to design and manage probability lists for any value, type, or custom object directly within the Unity Inspector. With this plugin, you can take control of your dice rolls, monster spawns, card decks, item drops, damage modifiers, or even organic animations using probability distribution with unparalleled simplicity and ease-of-use. Whether you want to design your lists, variable pick counts, and seeding right in the inspector, or harness the powerful API to control everything from code, RNGNeeds has you covered.

With RNGNeeds, you're not just getting a powerful tool, but also a user-friendly and adaptable solution. It's designed to cater to both coders and designers, with all features accessible via the Inspector and an easy-to-use API. The plugin supports all LTS versions of Unity and offers a range of powerful features. You can pick multiple items at once using various probability methods, including pure random selection. It also offers an easy way to retain a preferred seed, set a custom one, or implement your own Seed Provider. Furthermore, you can easily implement and register your custom Selection Methods or seeding options.

RNGNeeds also helps to make your workflow more efficient and adaptable. It enables you to design Influence Providers to dynamically modify probabilities based on external factors. You can choose a fixed pick count, or select a random range with an adjustable curve to control bias. Furthermore, you can avoid consecutive item selections with multiple repeat prevention techniques and track previous item picks for each list separately, employing history to mimic deterministic probability behavior.

Our commitment to you extends beyond the current version of RNGNeeds. We're excited to share that we have a range of new features on the roadmap that we're actively working on. These include dynamic modification of probabilities, card deck extensions, and the ability to assign units to items in the list and allow picks to deplete them, among others.

Thank you once again for choosing RNGNeeds. We're excited to see the amazing creations you'll bring to life with our plugin. Your trust in us fuels our passion to continue developing and refining RNGNeeds, to ensure it remains - probably - the best solution for all your random needs!

Getting Started

This section will guide you through the process of installing RNGNeeds from the Unity Asset Store and quickly setting up your first probability list.

> You can install RNGNeeds either by visiting the plugin page on the Asset Store or by opening the store inside the Unity Editor (Toolbar Menu » Window » Asset Store).

Installation

1. Open the Unity Editor

Start by opening your Unity Editor. Make sure you have a project open, or create a new one if necessary.

2. Access the Unity Asset Store

Navigate to the Unity Asset Store from within the Unity Editor. You can do this by clicking on **Window** from the menu bar and selecting **Asset Store** from the dropdown menu.

3. Search for RNGNeeds

In the search bar of the Asset Store, type in "RNGNeeds" and hit Enter. The RNGNeeds plugin should appear in the search results.

4. Download and Import the Plugin

Click on the RNGNeeds plugin from the search results to open its page. Here, click on the **Download** or **Import** button (the button's label will depend on whether you have downloaded the plugin before). If a window appears asking you to upgrade, you can just click Continue.

Once the download is complete, Unity will automatically prompt you to import the plugin into your project. Make sure all the files are checked in the Import Unity Package window and click Import.

> The RNGNeeds Plugin installs as a **Package**. This means that its contents will not appear in your Assets folder as usual, but instead in the **Packages/** folder. This approach helps reduce clutter in your Assets folder and allows RNGNeeds to operate through the Package Manager.
>
> Open the **Package Manager » In Project » Packages - Starphase Lab**. You will see two packages listed in this section:
>
> - **RNGNeeds** - *This is the main package for this plugin.*
> - **Starphase Core** - *This is a dependency of RNGNeeds, and it's a small collection of common utilities.*

5. Verify the Installation

After the import is complete, verify the installation by opening RNGNeeds preferences by going to **Edit » Preferences » RNGNeeds**. After opening the RNGNeeds preferences window, a message about loading default preferences and color palettes should appear in the console.

Quick Start

To start using probability distribution in your project, add the generic `ProbabilityList<T>` class to one of your classes, or create a new script by right-clicking in the Project panel and choosing **Create » C# Script**

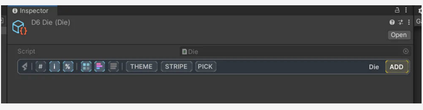
> You can use multiple **ProbabilityList** in any of your `MonoBehaviour` or `ScriptableObject` classes and design probabilities using the Inspector. Alternatively you can also use it in any other class and control the list and it's probability distribution using the API.

For this example, we will create a biased die that rolls the six a bit more often. Name the script `Die` and open it. Inherit from `ScriptableObject` and add a Probability List of type `int`.

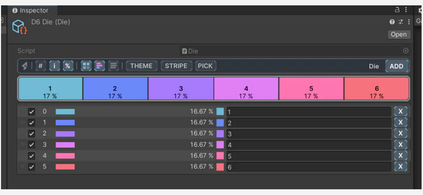
Adding the ProbabilityList

```
[CreateAssetMenu(fileName = "dXX", menuName = "My Die")]
public class Die : ScriptableObject
{
    public ProbabilityList<int> die;
}
```

In Unity Editor, right-click on the **Assets** folder and choose **Create » Die**. This will create a new Scriptable Object with the suggested name `dXX`. We will create a 6-sided die, so name the object `D6 Die`. Clicking on the object will display its properties in the Inspector, where you will see the Probability List Drawer.



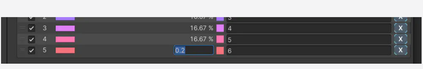
To add items into the list, click on the `Add` button on the right. These items will represent the sides of our die. Add six sides and specify their values.



Currently, the probabilities of items are even. We want to make the die roll the six more often, but retain the even chances among the other sides.

> You can adjust item probabilities by dragging their rectangles or edges on the stripe, or by typing in the value directly. Direct input will recalculate other items with respect to their distribution ratio.

Click on the 'side 6' probability `16.67%` and type in `0.2` (for 20%) and hit `Enter`. Probabilities of other items will drop to 16%.



Now it's time to roll the die. In RNGNeeds, you can select a value from the ProbabilityList easily by calling `.PickValue()`. Let's make our die object useful by adding a function that returns the roll.

Selecting values from list

```
[CreateAssetMenu(fileName = "DXX", menuName = "My Die")]
public class Die : ScriptableObject
{
    public ProbabilityList<int> die;

    public int Roll()
    {
        return die.PickValue();
    }
}
```

Now, whenever you call Roll() on one of your dice, you will get a single `int` value based on probability distribution.

> Using this approach, you can now create multiple dice for your project, such as `d10` or `d20` and roll them easily. Each of the Die ScriptableObjects can have its own number of items (sides of a die) and probability distributions. Speaking of which, now would be a good idea to rename our six sided die to `D6 Die (biased)`.

What's Next?

Congratulations on setting up RNGNeeds and taking your first steps into the world of advanced probability control! However, there's still a lot more to discover. To help you navigate and make the most out of RNGNeeds, we recommend diving into the following resources:

Exploring the Basics

Start by getting a strong grasp of the fundamentals. These articles will guide you through the user interface, introduce you to key terminology, and help you customize your setup via the Preferences Window:

- Understanding the User Interface
- Customizing RNGNeeds with the Preferences Window
- Key Terminology in RNGNeeds

Explore the RNGNeeds Example Usage →

Unleashing the Full Power

Once you're comfortable with the basics, it's time to explore some real-world applications and learn how to harness the full potential of RNGNeeds. These example tutorials will not only deepen your understanding but also inspire you with possibilities:

- Creating a Treasure Chest: A Basic RNGNeeds Example
- Building a Monster Spawner: Intermediate RNGNeeds Usage
- Designing a Card Deck: Advanced RNGNeeds Project
- Mastering Probability Influence: A Deep Dive

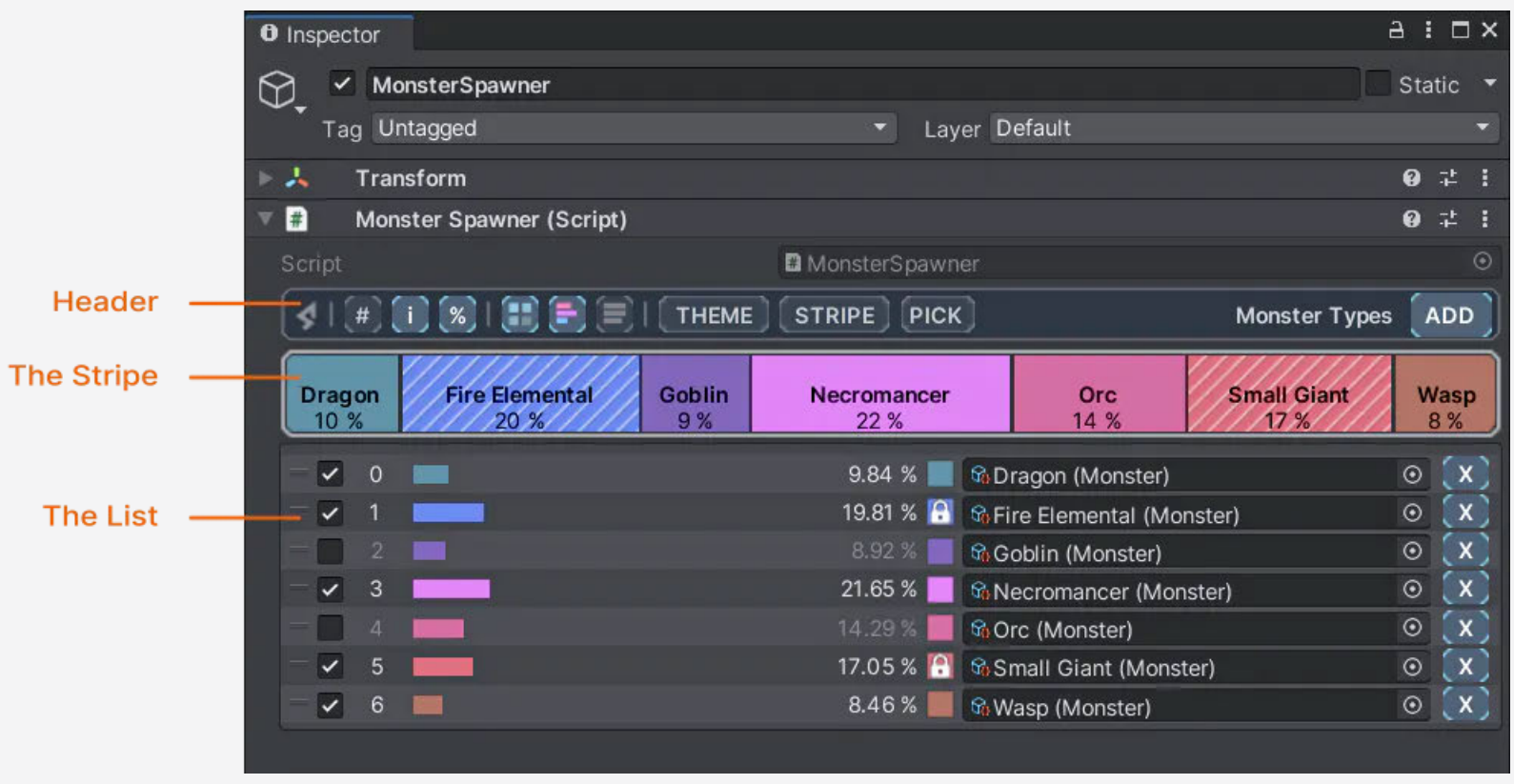
Remember, RNGNeeds is a powerful tool designed to cater to your unique needs. So dive in, explore, and let your creativity reign!

User Interface

Welcome to the User Interface guide for RNGNeeds! As a Unity plugin designed with ease-of-use in mind, we've made sure our UI is intuitive and user-friendly. This guide will help you navigate through the interface and make the most out of RNGNeeds' powerful features. From understanding the Probability List Drawer to using our dynamic Stripe widget, we'll cover all you need to know to confidently manage and design your probability lists within the Unity Inspector. Let's dive in and explore the user interface of RNGNeeds, your essential tool for managing randomness in your game design.

Basic Anatomy

The RNGNeeds Inspector drawer is comprised of three distinct sections - the **Header Bar**, the **Probability Stripe**, and the **Item List**. Each of these sections has a unique function in the process of creating and managing your Probability Lists.



Header Bar

The Header Bar is your control panel in the RNGNeeds Inspector drawer. It allows you to customize the drawer, adjust options for your Probability List, and add new items to the list.

The Stripe

A visual tool in the RNGNeeds Inspector that represents your items as colored blocks, with their width proportional to their probability. It allows you to visually adjust the probabilities of each item by dragging their edges or the blocks themselves.

The List

This is where your items are displayed in the RNGNeeds Inspector. Each entry provides information such as enable/disable toggle, index, probability preview bar, exact probability value, color code, and the option to remove the item from the list.

Header Bar

The Header Bar sits at the top of the RNGNeeds Inspector drawer. It's where you make global adjustments and customizations. Here, you can see the name of your property displayed next to the Add button, which is used for adding new items to your Probability List.

There are also quick toggles, represented by small rectangular buttons, that allow you to control various display options. These include showing or hiding item information on the Stripe, dimming colors, and managing preview bars in the List.

The Stripe

The Stripe is a powerful visual tool that lets you design and adjust your probabilities intuitively. Each item in your Probability List is represented as a colored block (let's call them 'probability blocks') on the Stripe. The width of these blocks is proportional to their associated probabilities, and together they fill up the whole Stripe, representing a total probability value of 1.

By dragging the edges between these blocks or the blocks themselves, you can adjust the probabilities of your items. For fine-tuning, you can hold Ctrl while scrolling to adjust probabilities by 0.1%, or Ctrl+Shift for a precision of 0.01%. The colors of the blocks correspond to those of the items in the List, providing a visually consistent reference across the interface.

> If the scroll direction feels unintuitive (for example on Mac OS with 'natural' setting on), you can reverse it. Visit the Preferences Section to learn more about additional and advanced options.

The List

The List is where your items are neatly laid out. Each item in the List has several properties displayed. From left to right, these are: an enable/disable toggle, index, a preview bar that visually represents the item's probability, the exact probability value, a color code, and a remove item button.

The color code square not only provides a visual reference for the item's color, but also acts as a lock to protect the item's probability from accidental changes or recalculation. By clicking the color code square, you can lock or unlock the item. The remaining space of the item entry is reserved for the value field, where you can interact with your item's value directly, be it a primitive type or a complex Unity object type.

That's the basic rundown of the three main sections of the RNGNeeds Inspector drawer. Dive in and explore these features further to truly make the most of RNGNeeds.

Preferences

In the "Preferences" section of the RNGNeeds plugin, you have the freedom to fine-tune how you interact with the tool. These options allow you to choose between compact and full button displays, invert scroll direction for a more natural feel, and more. Plus, you have the ability to set visual preferences based on your editor's theme, whether it's dark or light. It's all about making your experience as user-friendly as possible, so you can focus on crafting perfect probability lists for your game elements.

> To open RNGNeeds Preferences Window, click on the **Editor Menu Bar » Edit » Preferences**. In the window that appears, click on **RNGNeeds** in the left section.

Understanding RNGNeeds Preferences

RNGNeeds has three types of preferences.

- **Global Preferences** - these options are shared among all drawers and define their high-level behavior, such as the level of options, scroll direction, colorize gradients or logger settings.
- **Default Drawer Settings** - these preferences are drawer-specific and applied to each new drawer when it is first initialized (viewed). This is handy if you find yourself changing the same settings for each new drawer to suite your needs. Simply change the defaults in the Preferences Window under **Drawer Options » Default Drawer Settings**
- **Color Palettes** - a list of color themes shared by all drawers. Palettes are not applied to drawers. Drawers only keep reference to the palette name. This way you can customize existing palettes and drawers will reflect the change.

Preferences Window

The RNGNeeds preferences are located next to other Editor preferences in your Unity Project. To open the preferences window, go to **Editor Menu Bar » Edit » Preferences » RNGNeeds**

Global Preferences

Drawer Options Level - Choose between 'Basic' and 'Advanced' options in the drawer. Advanced will show additional options in the drawer.

Drawer Option Buttons - Show Full or Compact buttons in drawer Header. Use Compact if you have limited inspector space.

Inspector Refresh Mode - To achieve responsive visual-authoring drawer, the inspector has to repaint frequently. If you are experiencing editor slowdowns, try 'Optimized' option, which will repaint only during mouse drag. However, other actions, such as changing settings might feel less responsive.

Invert Scroll Direction - Use this to compensate system settings. For example on MacOS, the 'natural' scroll direction might feel inversed when using scroll to adjust probabilities.

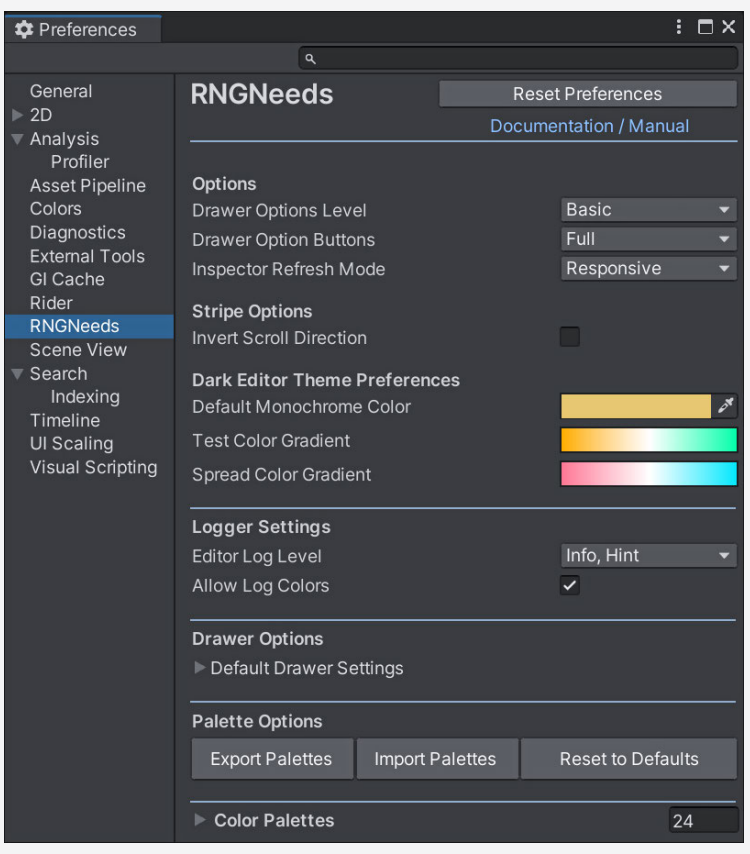
Default Monochrome Color - This will become the default Monochrome color for every new initialized drawer.

Test Color Gradient - This gradient will be used to colorize test result variation from desired probability.

Spread Color Gradient - This gradient will be used to colorize probability spread in an influenced list.

Editor Log Level - Select the message types to be displayed in the editor console.

Allow Log Colors - Enable or disable color-coded messages in the editor console.



Default Drawer Settings

Default Drawer Settings - New drawers will be initialized with these settings. Expanding this will reveal the default settings for new drawers.

Palette Options

Export Palettes - Current list of palettes can be exported as XML.

Import Palettes - Import list of palettes from XML. Note that the current list will be overwritten.

Reset to Defaults - Clears the current list of palettes and imports the factory-defaults.

Color Palettes - Expand this list to modify the palettes. You can change existing or add new ones.

Persistent Drawer Settings

RNGNeeds silently tracks preferences for each drawer individually, allowing you to stay organized when dealing with large number of objects you design. With this approach, you can easily change appearance of the drawers as you design your Probability Lists and come back to them later with settings unchanged. All options should persist with your commits and version control.

> The preferences window offers options to reset all preferences to defaults, which also resets the default drawer settings.
> The preferences are stored in your project folder under **/ProjectSettings/RNGNeedsSettings.asset**.
> Changes to this file will frequently show up in your version control. This is by design and is to ensure all your settings are carried over as you work.
> In case you don't want to commit preferences of RNGNeeds, add these lines to your `.gitignore` file:

```
# Ignore RNGNeedsSettings Asset
/[Pp]roject[Ss]ettings/RNGNeedsSettings.*
```

Support

Your satisfaction is a top priority here at Starphase Lab. RNGNeeds is designed with a user-first approach, prioritizing ease of use and a polished user experience. However, questions and issues can sometimes come up, and when they do, we're here to help.

Before reaching out for direct support, you might find it useful to check out the various sections of this documentation and our FAQ. Many common questions and issues are covered there, and you might find the solution you're looking for more quickly.

- FAQ
- Understanding the User Interface
- Customizing RNGNeeds with the Preferences Window
- Key Terminology in RNGNeeds

Email Support

If you're still having difficulties, feel free to reach out directly via email. We will make every effort to respond to your inquiry as quickly as possible.

Support Email

Join Our Discord Community

You're also invited to join our growing Discord community. It's a great place to chat with other users, find answers to common questions, and share or find inspiration for your projects. Please understand that while we strive to monitor and respond to inquiries on Discord, email remains the most reliable channel for urgent support requests.

Discord is a vibrant platform that allows for real-time discussion and sharing. We look forward to seeing the incredible creations you bring to life with RNGNeeds!

Join our Discord

Your feedback is invaluable. If you have suggestions for improvements or features you'd like to see in RNGNeeds, don't hesitate to share them with us. Together, we can make RNGNeeds even better!