

Table of contents

- Overview
- Discover InspectMe
- Who Benefits from InspectMe?
- Redefining Development
- Step 1: Acquiring InspectMe
- Step 2: Opening InspectMe
- Step 3: Inspecting Your First Object
- Step 4: Exploring Features
- Tree-View
 - Key Functionalities
 - Integration with InspectMe Features
 - Features
 - Intuitive Interface
 - Comprehensive Inspection
 - Advanced Search
 - Key Features
 - Unveiling the Unseen: Invisible Members
 - Quick Search and Filter Techniques
 - Practical Applications
- Inspect Viewer
 - Core Functionality
 - Usage
- Bookmarks
 - Core Functionality
 - Bookmarks Menu
 - Using Bookmarks
 - How to Add a Bookmark
- Watchers
 - Quick Overview
 - Setting Up a Watcher
 - Key Implementation Guideline

- Supported Types
- Watcher States
- Notifications
- Managing and Customizing
- Frequently Asked Questions (FAQs)
- Contact Support
- Bug Reporting
- Feature Requests

Welcome to InspectMe

Overview

InspectMe is a transformative Unity plugin, engineered to elevate the debugging and inspection processes for Unity developers to unprecedented levels. This potent tool refines your interactions with game objects and scripts, offering profound insights into the mechanics of your Unity projects.

Tailored for game developers, Unity aficionados, and technical artists alike, InspectMe is your companion in crafting a seamless development experience. With its intuitive interface and robust features, InspectMe doesn't just streamline debugging—it redefines it.

Discover InspectMe

We invite you to explore the array of features InspectMe has to offer. Each function is a building block towards a more refined development process, ensuring your focus remains on creativity and innovation.

Dive into the features that make InspectMe an indispensable part of Unity development. From real-time value inspection to advanced search capabilities, InspectMe is your guide through the complexities of game development.

Who Benefits from InspectMe?

InspectMe resonates with a broad spectrum of Unity enthusiasts:

- **Game Developers:** Discover a more fluid creative process.

- **Technical Artists:** See your art come to life with greater control.
- **Educators and Students:** Educate with, and learn through, an interactive lens.
- **Indie Developers:** Small teams can now achieve big dreams with tools that streamline development.

Redefining Development

With InspectMe, your journey through Unity development becomes not just manageable but enjoyable. It's an invitation to not only enhance your projects but also to join a movement towards intuitive and accessible game development.

Getting Started with InspectMe

Welcome to the beginning of a streamlined development experience with **InspectMe**! Let's get you up and running with minimal fuss.

Step 1: Acquiring InspectMe

- **Get InspectMe:** Discover InspectMe on the [Unity Asset Store](#). Access the full suite of features designed to enhance your Unity development workflow. Once you have InspectMe, you'll find it ready to be downloaded within the Unity Editor under `Window > Asset Store`. After downloading, Unity will prompt you to import InspectMe into your project.

Step 2: Opening InspectMe

- Access InspectMe from the Unity Editor by navigating to `Window > InspectMe`.
- Open the InspectMe window to dock it in your workspace or let it float so it's always in view while you work.

Step 3: Inspecting Your First Object

Initiate your first inspection with ease:

- **Via Inspect Viewer:** Drag and drop a gameObject from the hierarchy into the InspectMe window or right-click on a gameObject in the hierarchy and select 'Inspect Me' from the context menu for immediate inspection. [See more..](#)
- **Via Direct Component Drag:** Enhance your inspection scope by directly dragging

components into the **InspectMe** tree view. This feature provides an intuitive way to examine specific parts of your **GameObjects**.

- **Via Script:** Invoke `this.InspectMe()`; within any class to automatically inspect all its members, or specify an instance name for a targeted inspection.
- **Via Attribute:** Add the `[InspectMe]` attribute to classes or members, and use the generated 'Inspect' button in the Unity Inspector for quick access.

Step 4: Exploring Features

- Begin by expanding member nodes in the Tree View to see the structure and details of your objects.
- Use the Search Field to quickly find specific elements within your project.
- Experiment with Watchers to monitor real-time changes in your game's variables.

Features

Tree-View

The Tree-View is a core feature of **InspectMe**, presenting a hierarchical structure of inspected objects and their properties in real-time. It is designed to offer an intuitive and interactive visualization, making the debugging and inspection process in Unity clear and straightforward.

Key Functionalities

1. **Node Structure:** The Tree View effectively organizes data into nodes. Each node represents either a member information (`memberInfo`), a collection entry, or a virtual node. Virtual nodes are particularly useful for grouping elements or representing key-value pairs in dictionaries, enhancing the clarity and organization of data.
2. **Dynamic Columns:** The Tree View's structure includes three primary columns:
 - **Name Column:** Displays the name of the member or property, allowing easy identification.
 - **Value Column:** Shows the current value of the property, reflecting real-time changes.
 - **Type Column:** Indicates the member's data type, providing insight into the nature of the data handled.
3. **Node Interaction:** Enhance your workflow with flexible node interactions. Nodes can be expanded or collapsed for detailed or summarized views. For frequently accessed nodes, the

pinning feature allows quick and easy access.

4. **Keyboard Navigation:** This feature enables rapid selection, and expansion or collapse of nodes, streamlining your debugging process.
5. **ContextMenu:** Access a variety of actions through the context menu by right-clicking on a node. Depending on the node type, options like bookmarking, deletion, and change watching are available. For added convenience, copy options such as Copy Name, Copy Value, Copy Type, and Copy All are included.
6. **Supported and Not Supported Types:** The Tree View supports a range of types such as classes, structs, Unity components, primitives, arrays and generic collections like lists, dictionaries and SortedList etc... It does not support static members, events, delegates, and other specific types. For detailed insights into what is and isn't supported, please refer to our guide [here](#).
7. **Grouping of Collection Entries:** To optimize performance, Tree View groups large collections into manageable chunks. This prevents performance issues that could arise from attempting to render a vast number of nodes all at once.
8. **Tree View Menu:** The top menu of the Tree View offering immediate access to actions like clearing the view, collapsing nodes, and managing bookmarks or watchers. Additionally, it features a dropdown menu listing predefined groups like Prefabs, GameObjects, Classes, Structs, and visibility modifiers (Private, Public, etc.), to quickly pinpoint and highlight relevant members in the Tree View.

For an in-depth tutorial on using the Tree View, including tips and tricks for effective debugging, visit our [Usage](#) section.

Integration with InspectMe Features

The Tree View's powerful functionality is further enhanced when used in conjunction with other InspectMe features such as:

- **Inspector:** For an in-depth look at the selected member.
- **Bookmarks:** Quick access to frequently referenced member.
- **Watchers:** Real-time monitoring of value changes.

The Inspector is a versatile tool in **InspectMe**, designed to provide in-depth insights into the properties of your Unity objects. It enhances your ability to observe and analyze elements within your Unity project.

Features

Intuitive Interface

- **Quick Selection:** Easily select any member within the Unity Hierarchy or Project window directly from the Inspector.
- **Visibility Toggles:** Check the status of components, including enabled/disabled states and visibility (public/private) for classes.

Comprehensive Inspection

- Analyze a wide range of member types, from simple primitives to complex Unity objects.
- View detailed attributes and interfaces for classes and structs.
- Access customized displays for specific value types, like DateTime or Color, for more intuitive understanding.

Advanced Search

This powerful tool is designed to significantly streamline your debugging process in Unity, making it both efficient and intuitive. Let's dive into how you can leverage this feature to its fullest potential.

Key Features

Unveiling the Unseen: Invisible Members

- The Tree View primarily shows direct children of an inspected member. However, there's more beneath the surface. The "Invisible Members" from parent classes or superclasses are initially hidden in the Tree View.
- To reveal these Invisible Members, simply select them from the search dropdown. They will then be appended to the Tree View for detailed inspection.

Quick Search and Filter Techniques

Maximize your search efficiency in Inspector Search with advanced filtering options:

- **Effortless Searching:** Type in the search bar for instant feedback based on your input.
- **Filter Tags:**
 - `<n>` to focus on Names.
 - `<v>` for Value-based searches.

- <t> to target Type.
- Combine tags like <n,v> for a comprehensive Name and Value search.

Practical Applications

- Quickly access specific elements in complex hierarchies.
- Navigate and understand the properties of your classes and components with ease.

Inspect Viewer

The Inspect Viewer is an innovative feature within **InspectMe** designed to enhance real-time debugging and inspection capabilities for Unity developers. It provides a seamless interface for analyzing game objects and their associated components within the Unity Editor.

Core Functionality

1. **GameObject Inspection:** You can effortlessly drag game objects, either individually or multiple at a time, from the Unity Hierarchy or the Project window into the Inspect Viewer's feature area. The Inspect Viewer is capable of handling both single and multiple game objects, organizing them into structured groups for a clear and detailed view. The highlighting zones within the viewer change color to indicate the correct drop area, streamlining the inspection initiation process.
2. **Component Inspection:** Drag a single component to the tree view. The tree view area will be highlighted during this action, indicating the appropriate drop spot for an in-depth analysis of the component.
3. **Contextual Interaction:** Right-clicking a game object within Unity Hierarchy reveals an "Inspect Me" option that triggers the same inspection action as drag-and-drop.

Usage

1. **Initiating Scan:** Click and drag a game object from the Hierarchy or Project window to the Inspect Viewer area. A red zone indicates where to drop the object, turning green upon correct placement.
2. **Group Inspection:** Once dropped, the Inspect Viewer displays the game object and its components as a group for detailed analysis.
3. **Contextual Scan:** Alternatively, right-click a game object in the Hierarchy and select "Inspect Me" to achieve the same result without dragging.

!!! info Whenever you drop a component into the tree view or select a component from a gameObject within the viewer list, a convenient popup dialog will prompt you. This dialog is

designed for you to easily assign a custom name to the inspected component. Should you prefer a more streamlined approach, this feature can be deactivated in the preferences panel. In such cases, inspected components will automatically adopt a default naming format, which combines the gameObject's name with the component type, like so: `gameObjectName.ComponentType` .

Bookmarks

The Bookmarks feature in InspectMe is designed to streamline your workflow as a Unity developer. It offers quick and easy access to frequently inspected elements, significantly enhancing the efficiency of your development and debugging processes.

Core Functionality

1. **Organized Bookmark Groups:** Your bookmarks are intuitively grouped based on the root inspected member, simplifying the location and management of related elements.
2. **Interactive Bookmark Management:** You have the flexibility to interact with each bookmark, allowing you to select or remove bookmarked members as needed.
3. **Search and Display Options:** Utilize the powerful search function to either highlight your search results or filter to show only those bookmarks that match your criteria.
4. **Group Management:** Manage your bookmark groups effectively with options to expand, collapse, or delete entire groups, streamlining your organizational process.

Bookmarks Menu

- **Search Types:** Choose your preferred search mode with two toggle buttons:
 - **Highlight:** This option highlights the bookmarks matching your search while shading the others.
 - **Filter:** Select this to display only the bookmarks that match your search terms.
- **Control Buttons:** The menu includes 'Expand All', 'Collapse All', and 'Delete All' buttons for comprehensive bookmark group management.

Using Bookmarks

1. **Accessing Bookmarks:** To view all your bookmarked inspected members, simply open the Bookmarks feature, where they are organized into groups.
2. **Managing Bookmarks:** Use the search function for quick identification of specific bookmarks. Clicking on a bookmark navigates you directly to the inspected member within the Unity environment.
3. **Bookmark Group Controls:** Within the menu, you can expand, collapse, or delete groups of

bookmarks or individual bookmarks, tailoring your bookmark management to your project's needs.

How to Add a Bookmark

Here's how you can bookmark an inspected member for easy access:

1. **Context Menu Method:** Right-click on a member in the Tree View and select 'Add Bookmark' from the context menu.
2. **Bookmark Button:** Alternatively, use the bookmark button in the Tree View menu for quick bookmarking.

Watchers

The Watchers feature in InspectMe is a dynamic tool designed to enhance your debugging process in Unity. It allows you to monitor changes in real-time for various properties and fields of Unity objects.

Quick Overview

- **Functionality:** Watchers observe and alert you to changes in values of Unity objects during runtime.
- **One Watcher per Members:** Each inspected member within InspectMe's Tree View can have one Watcher.
- **Notification Options:** Customize how you receive alerts (e.g., logs, beep sound, editor pauses, or pop-ups).

Setting Up a Watcher

1. **Attach a Watcher:** Select a member from the Tree View and attach a Watcher. To attach a Watcher to a member:
 - **Via Context Menu:** Right-click on a member in the Tree View and select 'Add Watcher' from the context menu.
 - **Via Menu Button:** Use the menu button at the top of the Tree View to add a Watcher to the selected member.
 - **Scripted Attachment:** Directly add a Watcher in your script following an `InspectMe()` call. Customize settings to suit your project's needs. [Learn More](#)
2. **Configure Settings:** Choose your preferred notification method and other settings like continuous monitoring or history tracking.

3. Monitor Changes: Receive notifications based on your configured settings when values change.

!!! Info For a detailed guide on adding and using Watchers via code, check out the [Usage](#) Page for practical examples and tips.

Key Implementation Guideline




Watchers serve as essential debugging tools in the Unity Editor environment. They are not suitable for managing game logic in your production releases. Please ensure to use Watchers appropriately for debugging purposes only.

Supported Types

Watchers in InspectMe can be utilized with various member types, each with specific trigger conditions:

- **Value Types:** Inspect basic data types such as int, float, string, etc., for direct value changes.
- **Collections:** Monitor all generic collections, including custom classes derived from these. The watcher triggers when there's a change in the collection size.
- **Interfaces:** Track members representing interface implementations. The watcher activates upon changes in the assigned reference.
- **Special Types:** Includes DateTime, TimeSpan, and Unity-specific value types like Vector3, Color, Curve, etc., catering to a wide range of use cases.

Watcher States

State	Description
 Idle	The Watcher is set up but not actively monitoring the inspected member.
 Active	Actively monitoring the inspected member's value for changes.
 Invoked	A change has been detected in the inspected member and reported.

Notifications

Customize how you want to receive notifications when a watched value changes:

Notification	Description
Console Logs	Outputs messages in the Unity console for observed changes.

Notification	Description
Beep	Plays a beep sound to alert you of changes.
Pause Editor	Automatically pauses the Unity Editor when a change is detected.
Pop-Up	Displays a pop-up window with details about the change.

Managing and Customizing

Easily manage your watchers through InspectMe's intuitive interface. Activate, pause, or delete watchers as needed, and customize their settings to suit your workflow.

Frequently asked questions

???- Question "How does InspectMe enhance the debugging of Unity scripts?" InspectMe enhances script debugging by allowing developers to visually inspect and monitor variables and object states, identify issues quickly, and understand the behavior of their scripts more effectively.

???- Question "Can InspectMe display real-time updates of property values?" Yes, InspectMe can display real-time updates of property values, particularly useful for tracking changes during gameplay or other dynamic scenarios.

???- Question "Can InspectMe be used in conjunction with Unity's profiler for performance analysis?" Yes, InspectMe can complement Unity's profiler by providing detailed insights into object states and values, which can be valuable when analyzing performance data from the profiler.

???- Question "What types of Unity objects can I inspect with InspectMe?" InspectMe allows you to inspect a wide range of Unity objects, including classes, structs, Unity components, primitive types, and specific Unity/C# value types like `Vector2`, `Color`, etc. It also supports all generic collection types like lists, arrays, hashSets and dictionaries. [See more..](#)

???- Question "Can InspectMe be used for editing values during runtime?" In its current version, InspectMe is primarily designed for observing values, not editing them. It provides a comprehensive view of object properties and states for debugging purposes.

???- Question "Is it possible to monitor changes in a variable in real-time using InspectMe?" Yes, InspectMe's Watchers feature allows you to monitor changes in variables in real-time. You can set up watchers on specific nodes to receive notifications when their values change.

???- Question "Can I customize the notifications I receive from Watchers?" Yes, InspectMe allows

you to customize how you receive notifications from Watchers, including options like logs, beep sounds, pausing the Unity Editor, or pop-up messages.

???- Question "Can I use InspectMe to inspect interface implementations?" Yes, InspectMe allows you to inspect interface implementations. It shows the instance that implements the inspected interface and its members.

???- Question "How does InspectMe handle large collections or arrays?" For large collections or arrays, InspectMe uses techniques like entry grouping to optimize performance and display. It only draws entries that are visible in the Tree View.

???- Question "What kind of search capabilities does the Search Field in Inspector Values offer?" The Search Field allows you to search by name, value, or type within the selected node. It also reveals invisible members and supports inline filtering for more refined searches.

???- Question "Does InspectMe support keyboard navigation?" Yes, InspectMe supports keyboard navigation in the Tree View, allowing you to navigate through nodes using arrow keys and expand/collapse nodes efficiently.

???- Question "Is it possible to inspect private fields or properties with InspectMe?" Yes, InspectMe can inspect private fields and properties, offering a more in-depth view of an object's state than what is typically accessible through the standard Unity inspector.

???- Question "Does InspectMe support graphical types for inspection?" Yes, InspectMe supports the inspection of graphical types like sprites, textures, and materials, offering detailed views and previews where applicable.

???- Question "Does InspectMe support nested collections and arrays?" InspectMe supports nested collections and arrays, allowing you to inspect complex data structures like multidimensional arrays or nested lists in an organized and accessible manner.

???- Question "Can InspectMe be integrated into automated testing frameworks within Unity?" InspectMe is primarily a manual inspection tool and might not directly integrate with automated testing frameworks. However, its insights can inform and enhance automated testing strategies.

Support for InspectMe

Encountering challenges or have questions about InspectMe? We're here to help! Our support resources are designed to provide you with the assistance you need to make the most out of InspectMe.

Frequently Asked Questions (FAQs)

Before reaching out, we recommend checking our [FAQ](#). Your question might already have an answer there!

Contact Support

For inquiries specifically related to support, technical issues, or assistance with InspectMe:

- **Email Support:** Send us your queries at support@divinitycodes.de, and we'll get back to you as soon as possible.

For all other topics not related to support, including general inquiries, feedback, or collaboration:

- **General Contact:** Reach out to us at contact@divinitycodes.de.

Bug Reporting

If you encounter a bug or an issue with InspectMe, we encourage you to report it promptly so we can work on addressing it. Here are the ways you can report a bug:

- **GitHub Issues:** Report bugs directly through our [Issue-Tracker](#) on GitHub. This helps us track and manage issues efficiently.
- **Email Bug Report:** If you prefer, you can also send a detailed description of the issue to support@divinitycodes.de. Please include as much information as possible, such as what you were doing when the bug occurred, steps to reproduce it, and any relevant screenshots.
- **Community Reporting:** Share bug details on our [Discord](#) or [Reddit](#) for community-supported solutions and direct team interaction.

Prompt reporting of bugs and issues is crucial for us to maintain the quality and reliability of InspectMe. Your cooperation in this regard is greatly appreciated :pray:

Feature Requests

Got an idea that can make InspectMe even better? We love hearing from our users and welcome your suggestions:

- **Community Ideas:** Share your innovative ideas and engage in discussions on [Discord](#) and [Reddit](#). Here, you can propose new features and participate in community voting.

- **GitHub Issues:** For a more formal feature request, consider using our [Issue-Tracker](#) on GitHub. This allows us to track and manage your suggestions effectively.
- **Feedback Form:** Prefer to submit your ideas anonymously? Use our [Feature Request Form](#). We regularly review submissions from this form for inclusion in future updates.

Your experience with InspectMe matters to us. Whether you need help, want to report a bug, or have suggestions, our team is here to support you every step of the way.