â ï¸ **UNDER DEVELOPMENT ã USE AT YOUR OWN RISK.**

Howl

Howl is a symbolic notation engine for C# programming. Have a look:

```
ã  IsEscapedDoubleQuoteInString(ã ¹ x, á  i){ â¤´ (suffix â  "\"" â ¨
x[i] â  '"') â®  â  ; ã  esc = â  ; â ² (--i > 0){ â¤´ (x[i] â  '\\')
Â¦ â¤µ esc = !esc; } â®  esc; }
```

Here is the C# translation:

```
cs bool IsEscapedDoubleQuoteInString(string x, int i){ if (suffix != "\""
|| x[i] != '"') return false; bool esc = false; while (--i > 0){ if (x[i]
!= '\\') break; else esc = !esc; } return esc; }
```

- Howl is a superset of C#: configure which notations are applied, be it on first import, or later.
- Bi-directional translation imports your legacy (C#) sources.
- Input Howl source comfortably (VS Code and Atom snippets). As you type C#, Howl source is generated; therefore, learning the notation is easy and fun.

In Atom, beautiful syntax highlighting is also available.

Unity 3D: The UPM package provides seamless build integration:

Howl, build and publish either C#, Howl scripts, or both.

Why Howl?

Notations are commonplace. They are used in music, mathematics, dance and many other places. The case for any notation is that once you know it, information processing is faster.

Muting language level semantics increases focus. Also, modifiers and keywords take up space (up to 20% of your program source). Compressing this helps with writing delightfully concise, readable and expressive programs.

Getting started

The UPM package requires Unity 3D (the game engine). After adding the package to an existing project go to `Window > Activ > Howl`. You will be guided through simple steps:

- Install a supported IDE (Atom or VS Code)
- Language support package (for Atom)
- Ensure your project uses Git. Technically not a requirement but we'd like to know your files are safe.
- Optionally, import your legacy code. If you need more control, skip this.

After setup you are presented with the Howl main window/tab. Options are explained therein, but here is a summary anyway:

- **Refresh** - to import and build out of date Howl sources. Building is automated so not needed unless the asset database is taking a nap.
- **Rebuild** - to clean and rebuild everything. Useful if you somehow ended with conflicting/duplicate files in your `~build` directory (read along to learn what this is).
- **Import all** - to import C# sources lying around, if any.
- **Export all** - restores your C# sources (if you no longer wish to use Howl).
- **Configure and apply your symset** - select/deselect symbols then choose `Make Snippets` or `Apply` to update sources to the currently selected symbols.

TIP: Import, export and symset config may also be applied on a per file/directory basis via the Unity project window.

In normal use you do not need any of the above. Close the Howl window/tab, focus on your code, and be happy.

Good to know

Use the Unity project window to move and delete Howl scripts. This is similar to how Unity does not like you modifying files outside the editor.

Unity needs to see the C# output in order to build your project. We keep the C# output under `Assets/~build`. It is okay to move/rename this but if you do so, close Unity first and do it manually.

If you are using **assemblies**:

- Howl supports **.asmdef* on import. Your assembly definitions are placed in the `~build` directory.
- Edit assembly definitions as normal, except they now live inside the `~build` directory.
- Do not mix Howl and C# sources in the same assembly; this is not supported and you will get errors.
- Do not delete the `~build` directory (or lose your assembly definitions).
- Currently you may only have ONE `~build` directory.

Should you wish to create a new assembly for your Howl scripts:

1) Create the assembly normally. The same way that you create assemblies for C# sources. 2) Right click on the assembly and select `Use Howl`.

Howl generates **.asmdt* token files which are just house-keeping so we know where to find the C# assemblies.

Where next

- If you are still on the fence, read *Should I use Howl?*
- Learn about exciting features being worked on.
- Have a peak at the Howl source code (written in Howl).
- View the Cosmo [LINK] specification

Think Howl is obnoxious but still kinda cool? Fuel this rocket ð and feed the beast â ï¸

In C#, nobody will hear you Howl ð ð ¼