TMP Notation Engine (TextMeshPro)

A rule-based notation engine for rendering math & science expressions in TextMeshPro.

A lightweight **TextMeshPro (TMP) formatting SDK** for Unity that automatically converts plain-text mathematical, scientific, and chemical expressions into properly formatted TMP-rich text.

This SDK is designed for **educational games, quizzes, simulations, and math/chemistry content**, where expressions like `x^2`, `H2O`, or `1/2` must render correctly without manual TMP tags.

✨ Key Features

- Automatic **superscript & subscript** handling
- Supports **caret (^) and underscore (_)** notation
- Unicode superscript/subscript conversion (^{2}, $_{2}$, etc.)
- Smart **fraction formatting**
- Automatic **chemical formula formatting** (H_2O, CO_2, $Ca(OH)_2$)
- Fully configurable formatting rules
- Extension methods for clean, fluent usage
- Zero TMP dependency at runtime (string-based)

📦 Namespace

using AbS;

🧠 Architecture Overview

Core Components

Component	Responsibility
`TMPNotationEngine`	Main static formatter engine

`FormatConfig`	Controls formatting behavior
Unicode Maps	Converts Unicode superscript/subscript
Regex Converters	Parse math, fractions & chemistry
Extension Methods	Fluent TMP string helpers

⚙️ FormatConfig (Configuration)

Customize how formatting behaves using `FormatConfig`.

```
var config = new TMPNotationEngine.FormatConfig
{
   SuperscriptSize = 60f,
   SubscriptSize = 60f,
   FractionSize = 70f,
   EnableCaretNotation = true,
   EnableUnicodeConversion = true,
   EnableChemicalFormulas = true,
   EnableFractions = true,
   EnableUnderscoreSubscript = true
};
```

Configuration Options

Property	Description
`SuperscriptSize`	TMP size percentage for superscripts
`SubscriptSize`	TMP size percentage for subscripts
`FractionSize`	TMP size percentage used in fractions
`EnableCaretNotation`	Enables `x^2`, `x^(n+1)`
`EnableUnderscoreSubscript`	Enables `x_1`, `x_{n+1}`
`EnableUnicodeConversion`	Converts ^{2}, $_3$, etc

`EnableFractions`	Enables `1/2`, `(a+b)/(c+d)`
`EnableChemicalFormulas`	Converts `H2O`, `CO2`

🚀 Basic Usage

Format with Default Settings

```
tmpText.text = TMPNotationEngine.Format("x^2 + H2O → CO2");
```

Using Extension Method (Recommended)

```
tmpText.text = "E = mc^2".FormatForTMP();
```

🧪 Supported Syntax & Examples

Superscripts (Caret Notation)

Input	Output
`x^2`	x^{2}
`x^-3`	x^{-3}
`x^(n+1)`	x^{n+1}
`e^{2x}`	e^{2x}

Subscripts (Underscore Notation)

Input	Output
`x_1`	x_{1}
`x_{n+1}`	$x_{□+1}$
`a_(i,j)`	$a_{i□}$

Unicode Conversion

Automatically converts Unicode characters:

$x^2 + H_2O$

➡ Converted to TMP `<sup>` / `<sub>` tags internally

Fractions

Input	Output
`1/2`	$\frac{1}{2}$
`(a+b)/(c+d)`	$(a+b)/(c+d)$
`{x+1}/{y-1}`	fraction formatted

Implementation:

tmpText.text = "(a+b)/(c+d)".FormatForTMP();

Chemical Formulas

Automatically subscripts numbers after elements:

Input	Output
`H2O`	H_2O
`CO2`	CO_2
`Ca(OH)2`	$Ca(OH)_2$

Already formatted TMP text is safely ignored.

✍️ Manual Formatting API

Superscript

```
string sup = TMPNotationEngine.Superscript("2");
```

Subscript

```
string sub = TMPNotationEngine.Subscript("n");
```

Fraction

```
string frac = TMPNotationEngine.Fraction("a+b", "c+d");
```

🧰 Utility Methods

Strip TMP Tags

```
string plain = TMPNotationEngine.StripTags(formattedText);
```

Check Formatting

```
bool hasFormatting = TMPNotationEngine.HasFormatting(text);
```

🔗 Extension Methods (Fluent API)

```
"x^2".FormatForTMP();
"2".ToSuperscript();
"n".ToSubscript();
"a+b".ToFraction("c+d");
formattedText.StripTMPTags();
text.HasTMPFormatting();
```

🧪 Full Example

```
using AbS;

void ShowEquation()
{
    tmpText.text = "E = mc^2 + H2O".FormatForTMP();
}
```

⚠️ Notes & Best Practices

- Use **TMP fonts that support math symbols** (e.g. Liberation Sans, Noto Sans, STIX)
- Avoid mixing manual TMP tags with auto-formatting
- Order of formatting is handled internally for safety
- Ideal for **World Space Canvas** math rendering

🎯 Ideal Use Cases

- Educational & assessment games
- Math / physics / chemistry quizzes
- Scientific simulations
- Formula-heavy UI text
- TextMeshPro-based world-space equations

📄 License & Customization

This SDK is fully extensible:

- Add more regex rules
- Extend Unicode mappings
- Disable individual features via `FormatConfig`

Target: TextMeshPro (Unity)

License: MIT
Author: **Abhishek Sahu**