TMPNotationEngine Documentation

Namespace: `AbS`

Purpose:
TMPNotationEngine is an advanced TextMeshPro (TMP) formatting SDK for Unity that enables clean rendering of superscripts, subscripts, fractions, chemical formulas, and Unicode math/scientific notation using TMP-compatible rich text tags.

✨ Key Features

- Caret-based superscripts (`x^2`, `x^(n+1)`)
- Underscore-based subscripts (`x_1`, `x_{n+1}`)
- Automatic Unicode conversion (`x²`, `H₂O`)
- Fraction formatting (`(a+b)/(c+d)`)
- Chemical formula detection (`H2O`, `CO2`, `Ca(OH)2`)
- Reversible conversion (TMP ⇄ Unicode)
- Extension methods for fluent usage
- Fully configurable formatting rules

📦 Installation

1. Add the SDK source file to your Unity project.
2. Ensure TextMeshPro is installed.
3. Add the namespace where needed:

```
using AbS;
```

🚀 Basic Usage

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

Using Extension Methods

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

⚙️ Configuration

You can control formatting behavior using `FormatConfig`.

```
var config = new TMPNotationEngine.FormatConfig
{
   SuperscriptSize = 50f,
   SubscriptSize = 55f,
   FractionSize = 70f,
   EnableChemicalFormulas = true
};

string result = TMPNotationEngine.Format("H2SO4 + x^2", config);
tmpText.text = result;
```

Available Options

Property	Description	Default
SuperscriptSize	Size percentage for superscripts	60%
SubscriptSize	Size percentage for subscripts	60%
FractionSize	Size for numerator/denominator	70%
EnableCaretNotation	Enables `^` notation	true
EnableUnderscoreSubscript	Enables `_` notation	true
EnableUnicodeConversion	Converts Unicode scripts	true
EnableFractions	Enables fraction formatting	true
EnableChemicalFormulas	Enables chemical parsing	true

✍️ Manual Formatting API

```
string super = TMPNotationEngine.Superscript("2");
string sub = TMPNotationEngine.Subscript("0");
string fraction = TMPNotationEngine.Fraction("a+b", "c+d");
```

🧠 Supported Patterns

Mathematical

- `x^2`, `x^-3`, `x^(n+1)`
- `x_1`, `x_{n+1}`
- `(a+b)/(c+d)`

Unicode

- `x²`, `x₂` (auto-detected)

Chemistry

- `H2O`
- `CO2`
- `Ca(OH)2`

🔧 Utility Methods

Formatting Detection

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

Unicode Detection

```
bool hasSuper = TMPNotationEngine.HasUnicodeSuperscript("x²y³");
bool hasSub = TMPNotationEngine.HasUnicodeSubscript("H₂O");
bool hasAny = TMPNotationEngine.HasUnicodeScripts("x²₁");
```

Extract Unicode Characters

```
List<char> supers = TMPNotationEngine.GetUnicodeSuperscripts("x²y³z⁴");
List<char> subs = TMPNotationEngine.GetUnicodeSubscripts("H₂O + CO₂");
```

🔄 Conversion Methods

TMP → Unicode

```
string tmp = "A<sub><size=60%>0</size></sub>";
string unicode = TMPNotationEngine.Unicode(tmp); // A₀
```

TMP → Plain Text

```
string plain = TMPNotationEngine.PlainText(tmp); // A0
```

Round Trip Example

```
string original = "H₂O";
string formatted = TMPNotationEngine.Format(original);
string back = TMPNotationEngine.Unicode(formatted);
```

🧩 Extension Methods

```
"2".ToSuperscriptTMP();
"n".ToSubscriptTMP();
"a+b".ToFractionTMP("c+d");

"x^2 + H2O".ToFormatTMP();
```

Unicode Checks

```
"x²y³".HasUnicodeSuperscriptTMP();
"H₂O".HasUnicodeSubscriptTMP();
"x²₁".HasUnicodeScriptsTMP();
```

🧪 Advanced Examples

Mathematics

```
string math = "∫(x^2 + 2x + 1)dx = x^3/3 + x^2 + x + C".ToFormatTMP();
```

Chemistry

```
string chem = "2H2 + O2 → 2H2O".ToFormatTMP();
```

Physics

```
string physics = "F = G(m_1*m_2)/r^2".ToFormatTMP();
```

Mixed Notation

```
string mixed = "x₁² + y^2 = r²".ToFormatTMP();
```

✅ Best Practices

- Use world-space canvases with stable scaling for TMP text.
- Prefer `ToFormatTMP()` for readability and chaining.
- Disable unused features in `FormatConfig` for performance-critical paths.

📄 License & Usage

License: MIT

Author: Abhishek Sahu
SDK: TMPNotationEngine
Target: Unity + TextMeshPro
