Mesの言語仕様策定について

iranika（@happy_packet）

v0.3.0
改訂
2022.06.25　バージョン0.2仕様の策定
2022.08.22　バージョン0.3仕様の策定
2022.12.01　バージョン0.4仕様の策定
2023.02.17　バージョン1.0仕様の策定
2023.03.28　以降のバージョンは2023.xxxxと記載するように変更(xxxxは日付)
2023.04.05　現時点での最新仕様にアップデート

■目次

■注釈

＊(TBD)と表記されているものは検討段階であり決定事項ではないことを指します。
＊ご意見等があれば、下記のフォームにお寄せください。
　https://forms.gle/YxArhEPkj46snpQa9

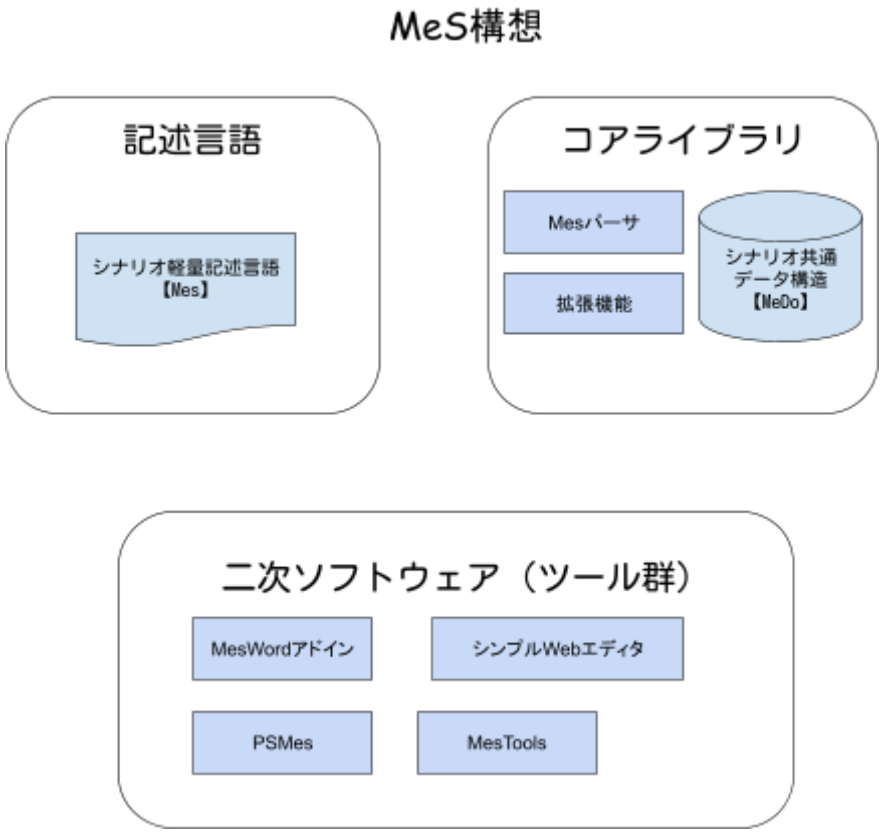
はじめに～Mesプロジェクトの概要～

※この章は読み飛ばしても問題ありません。

Mesプロジェクトは「シナリオ（脚本）の価値と可能性を高める」ことを目的としたプロジェクトです。

現在、Mesプロジェクトには以下のミッションがあります。

- シナリオライターが記述しやすい軽量記述言語の策定（Mes）
- アプリケーションが利用しやすい共通データ構造の策定（MeDo）
- Mesを構文解析してMeDoに変換するパーサライブラリ（Mesコアライブラリ）
- MeDoから台本を生成するビュアーソフト（二次ソフトウェア）



Mesプロジェクトでは、小さなプロジェクトが複合体として機能することができるように、責任を「記述言語」「コアライブラリ」「二次ソフトウェア」の3領域に分割することで、変更柔軟性を高めています。この3領域に分割された構想をMes構想と呼んでいます。

例えば現時点で記述言語はMesのみですが、今後新たな記述言語が生まれるかもしれません。新言語はパーサーが必要になりますが、パーサーはMeDo形式のデータを出力すれば良いだけで、MeDoのみに依存している他のエコシステム(Viewerソフト等)は新言語向けに書き直す必要はほぼありません。

Mes構想では責任分割のおかげで、新しい取り組みの追加、古い取り組みの削除に柔軟に対応できるよう意識されています。

このMes構想において非常に重要なのがデータ構造です。
シナリオをアプリケーションが利用しやすいデータ構造に変換できれば、シナリオの利用価値が高まります。以下に例をあげます。

＊シナリオ中のセリフ文字数をキャラクター毎に集計
＊オリジナルの組版を適用した台本の生成
＊ノベルゲームのスクリプトへ変換
＊タイピングゲームのテキストとして利用
＊etc…。

共通の汎用データ構造があれば、アプリケーションの開発者はデータ加工に注力できるので開発がしやすくなります。もしかしたら、おもしろいエコシステムのツールをあなたが自作するかもしれません。
Mesはソフトウェアが解析可能な記述ルールを整備し、脚本テキストを構造化されたデータ形式で扱えるようにすることで、『キャラ毎のセリフ文字数の集計』『Excelに貼り付け可能なCSVとして出力』などのソフトウェアの恩恵を最大限に受けられるようにすることを目標のひとつにしています。
Mesでシナリオとシナリオに関わる人たちがより快適に過ごしやすいアクアリウムになることを目指しています。

Mesについて

　Mesはシナリオに意味を付加するための軽量記述言語です。Mesはシナリオを字コンテとしても活用できるようにすることを目標のひとつにしています。
　Mesの特徴はセリフをデコレーションするようにト書きを記述する点です。

Mesの一例

```
== 場面１　　//==でシーンが区切られる

@ニカ　　//@はキャラ名を指す
＄駅前の音　　//＄は音響コメントを指す
！正面　　//！はサウンドポジションを指す
＃待たされた感じで言う　　//＃は汎用コメント（ト書き）
＆00:00:05.000 --> 00:00:10.039　　//＆は字幕用のタイミング情報（WebVTT推奨）
あ、キタキタ。女の子二人を待たせるなんて、失礼だぞ。　//セリフ

@花子
！正面
そういうニカちゃんも、ついさっき来たばかりじゃないですか。

@ニカ
＄少し離れてヒソヒソ声になる
こういう時は待たせた弱みに漬け込んで、ランチを奢らせるのがだな…

@花子
前回もそうやって奢らせてたじゃないですか。今日はだめです。
＃ちょっと叱る感じで（デコレーターは後置もOK）

@花子
それにしても久しぶりですね。この三人で会うのは何年ぶりでしたっけ？

@ニカ
確か、最後に会ったのは富山に海鮮丼食べに行ったときだから、二年ぶりだね。
```

　シナリオライティングは、キャラクターたちの会話をセリフとト書きでアウトプットしていく作業です。そして現代において多くの場合、キーボードを叩いて入力していく作業です。
　この時、脳は「会話の創造」と「入力作業」というタスクでコンテキストスイッチが頻繁に発生します。入力作業が流れるように行えなければ、脳のリソースは会話の創造に集中することができません。集中できたとしても、制作遅延に繋がりかねません。
　思考する速度に置いていかれないようにするためには、キータイピングの速度だけでなく記述フォーマットにも工夫が必要になるはずです。
　これらを念頭に「流れるような記述」を意識した軽量記述言語として、Mesの仕様策定が始まりました。

Mesのデコレーターによる記法は従来のシナリオ記述から少しかけ離れているため、読みにくいと感じる人が居るかもしれません。それは至極当然で、Mesにおける読みやすさの責任は二次ソフトウェア(ビューアー等)に移譲されており、Mesは読みやすさよりも書きやすさを重視するよう作られています。**書くための「原稿」、読むための「台本」という役割の分割を行い、原稿と台本の変換は外部アプリケーションに委ねることで、可読性における問題をMesから分離しています。**

Mesによる原稿が読みにくいと感じるなら台本プレビューアプリを使うか、外部アプリを自作してしまう。

それがMesの設計思想です。

※とはいえ、無闇に可読性を下げる仕様は記述にも混乱をもたらすため、Mesは可読性も意識しています。あくまで可読性よりも記述性を優先しているということです。

セクション表記について

セクションは==＜セクション名＞で記述します。
セクション名を省略して==だけで無名のセクションとして区切ることも可能です。
セクション自体の記述を省略すると１つの無名セクションとして扱われます。
例）==第一章　旅立ち

デコレーターの表記について

現バージョンでのデコレーター(修飾子)の表記について説明します。
デコレーターはあくまで付加情報であり、シナリオ記述において必須ではありません。

キャラクター名（@）

セリフを発言するキャラクター名として利用されます。
修飾子は@です。全角文字の＠も同様に使用できます。
例：@アリス

一般コメント（#）

一般コメントは普遍的なト書きとして利用されます。
声優への注釈という位置づけとして解釈しても構いません。
修飾子は#です。全角文字の＃も同様に使用できます。
例：＃アリスは笑っている

音響コメント（$）

音響コメントは、音響に関する指示として利用されます。
音響監督への注釈という位置づけとして解釈しても構いません。
修飾子は$です。全角文字の＄も同様に使用できます。
例：＄雨音

サウンドポジション（！）

サウンドポジションは、セリフがどの方向から聞こえるかの定位を明示できます。
現状はコメントと同じ注釈レベルでの指示しかサポートしていません。
後継のバージョンで、方向と距離が数値として指定できるよう改良される予定です。
修飾子は!です。全角文字の！も同様に使用できます。
例：！右斜め前

字幕用タイミング情報（&）

字幕用タイミング情報は、主に字幕データ生成のために用いられます。
音声または動画で、セリフが出現するタイミング(時間情報)を明示します。
現時点で、特定の字幕データフォーマットに該当するか構文チェックは行われません。
現時点で、データはシンプルな文字列として扱われます。
現時点で、時間の記述方式はWebVTTを推奨しています。
例：&00:00:05.000 --> 00:00:10.039

拡張フィールド（？）

拡張フィールドはMesと連携するアプリケーションが求める情報を付与するための任意デコレーターです。Mesパーサはこの情報をそのままテキストとして出力します。テキストをパースして情報をどのように利用するかは、外部アプリケーションに委ねられます。

例：？content-type:header,text-color:red
　　？text
　　？work

コメントアウトについて

Mesのデータとして必要ない行は//でコメントアウトすることができます。
//の開始位置から改行までの文字列は事前に削除されます。

糖衣構文(フラットレイヤー)

Mesには一般的なシナリオ記述形式の一部をサポートしたり、記述体験を向上するための糖衣構文（略式記法）があります。この糖衣構文はシナリオテキストをパースする前にMes記法へと変換されて、変換後のテキストがMeDo形式にパースされます。

これにより、一般的なシナリオ記述で使われる一部の表現形式がそのまま記述できます。

糖衣構文には処理優先度があり、優先度が高い構文から先に展開されます。優先度は3桁の数字で表現され、数字が小さいほど優先度が高くなり先に展開されます。

(デコレータ){{ テキスト }}

処理優先度は011です。{{}}内のテキストの各行にデコレータが付与されます。

{{}}内のテキストにこの構文を再度埋め込む（ネストさせる）ことはできません。

下記は変換の例です。

```
(@太郎){{
これはサンプルのセリフだ

こんなふうに囲った範囲のテキスト各行にデコレータを付与するぞ
}}
↓(変換後)
@太郎
これはサンプルのセリフだ

@太郎
こんなふうに囲った範囲のテキスト各行にデコレータを付与するぞ
```

名前「セリフ」

処理優先度は100です。

下記は変換の例です。

```
太郎「これはサンプルのセリフだ」
↓(変換後)
@太郎
これはサンプルのセリフだ
```

名前<TAB or 半角スペース4つ>セリフ

処理優先度は101です。デリミタとしてTAB（¥t）または半角スペース４つが使われます。
下記は変換の例です。

TAB版

```
太郎　これはサンプルのセリフだ
↓(変換後)
@太郎
これはサンプルのセリフだ
```

半角スペース４つ版

```
太郎　　これはサンプルのセリフだ
↓(変換後)
@太郎
これはサンプルのセリフだ
```

(TBD)ブロックの区切りを表す;

処理優先度は010です。
セミコロンは改行を挿入します。全角の；も同様に扱われます。
下記は変換例です。

```
これはセリフ1行目です。;
これはセリフ2行目です。;
↓(変換後)
これはせりふ１行目です。

これはセリフ2行目です。
```

Mesの構文基礎

※Mesのサンプルテキストを一読しておくと理解が早まると思います。
※記述するだけならサンプルテキストを真似て書くほうが早く習得できます。
Mesのサンプルテキスト

Mesは始めにheaderとbodyに区分されます。
headerとbodyはヘッダセパレーター「----」によって仕切られ、上がheader、下がbodyのフィールドになります。

headerはシナリオに関するメタデータを記述するためのフィールドです。メタデータの例として、タイトル、デフォルトキャラネームなどがあります。headerはコンフィグのような情報を格納するフィールドと解釈しても構いません。
headerは省略することが可能です。その場合はセパレーターも省略することが出来ます。
(TBD)headerで記述可能な情報はHeaderで使用可能な記述の章を参照してください。

bodyはコンテンツ本体を格納するフィールドです。主にセリフとデコレーターが記述されます。Mesの本体と言っても良いでしょう。
bodyは空白行で分割された複数のピース(MeDoPiece)で構成されます。
ピースにはセリフとデコレーターの情報が記述されます。セリフとデコレーターは、省略すると値は空白として扱われます。セリフの記載がなくデコレーターのみ記載されたピースは、空白のセリフをデコレーションしていると見なされるため構文としてエラーにはなりません。これによりセリフと連動しないようなト書きを表現することが出来ます。これは現時点で章と節の表現としての利用が想定されています。

Headerで使用可能な記述

デコレーター記号の設定変数

Headerでは各種デコレータの記号を任意の１文字に変更することができます。
デフォルト設定から変更することは基本的に推奨されていないことに注意してください。
バグの原因になります。
また、デフォルト設定と異なり、大文字と小文字は区別されます。

各種デコレーターの変数名は以下の通りです。

```
$deco_charactor=#
$deco_comments=!
$deco_sound_position=$
$deco_sound_note=&
$deco_timing=?
$deco_ext_field=@
```

カウント除外記号の設定変数

$ignore_char設定変数でカウント除外文字（記号）の指定ができます。
デフォルト設定では除外する文字はありません。
この機能では文字列を指定することはできません。
右辺に記述された文字列は一文字ごとに分解されて解釈されます。
右辺で重複する文字は一意(unique)になるように重複削除されます。
＝の後ろに空白を入れるのは非推奨です。
以下は『、』『。』を除外するケースの例です。

```
$ignore_char=、。
```

デフォルトキャラクター名の設定変数

$default_charactor_name設定変数でキャラクター名のデフォルト値を設定できます。
キャラクター名の記述を省略した場合、このデフォルト値が使われます。
＝の後ろに空白を入れるのは非推奨です。

```
$default_charactor_name=メロス
```

(TBD)展開定数の定義

%%<定数名>=<文字列>で展開定数を定義できます。
展開定数はBody部分において利用でき、%%[<定数名>]で記述された箇所は<文字列>に置換されます。[]内はトリミングされないことに注意してください。
下記は例です。

```
%%主人公=メロス
----
@ナレーション
%%[主人公]は激怒した。
```

(TBD)タイトル

シナリオのタイトルを == タイトル と記述します。

(TBD)メタコメントブロック

@```から```で囲まれた部分はメタコメントブロックとして処理されます。
メタコメントブロックはキャラクター情報やあらすじ等をメモとして記述するためのエリアとして使うことが想定されています。

Mesのサンプルテキスト

※サンプルテキストはGoogleDriveにもあります。
https://drive.google.com/drive/folders/1AdWCIsvHBumlAp56kwwPW-cj8pJB1SEa?usp=share_link
※脚本のサンプルをご提供頂ける方はiranika(@happy_packet)までご連絡ください。
Mesで記述されていなくても構いません(iranikaが修正します)

// 餅月ゆなの_01■すごい豪雨は突然に（導入）

// 餅月ゆなの_前日譚

== ニカ先生のMes講座１

ADV向け分岐のサンプル