Substation Beta (SSB) v0.1

A typesetting-oriented subtitle format

File Format & Syntax

File Encoding: Text-only, UTF-8
Line Breaks: Windows (CRLF) & Unix (LF)
Data Layout: Line-based
Ignore empty lines: Yes
Comments: Lines beginning with //
Syntax error warnings: Depends on renderer implementation

Sections

SBB scripts are separated in sections for different properties.
Sections begin with a header. It's a name, starting with character **#**.
They can be in any order but this is the recommended order:
#INFO
Meta information.
#TARGET
Target frame dimensions and other properties.
#MACRO
Base-Macros to complement render data. Use this to create base styling for your lines.
#EVENT
Render data & condition.
#RESOURCE
Describes all resources like textures and fonts.

#INFO-Section

The meta section contains some side informations and has nothing to do what's rendered at the end. It's just interesting for editors.
Fields begin with a name, followed by ": " and the value.
There are no constraints but here are some examples:
Title
Script title.
Author
Script author.
Description
Script description.
Version
Script version.

#TARGET-Section

Contains size information for the canvas that will be rendered on. On difference to the actual material, will be scaled to fit. Fields begin with a name, followed by ": " and the value. Valid fields for frame are...
Width
Target frame width.
Height
Target frame height.
Depth
Used to decide what to render before what. Whole positive number.
View
Perspective of the view. Can be `orthogonal` or `perspective`. Default is perspective.

#MACRO-Section

In the macros section you can define collections of tags to use them later. Use these macros to generate base stylings for your lines.
Macros begin with an unique identifier name, followed by ": " and the associated text.
Example:

default: font=Open Sans;font-size=12;

#RESOURCE-Section

Usually defined at the end of the file due to possibly being very long. This section contains links to the file system or base64 encoded resources like fonts or textures.
You can define Texture- and Font-Resources in the following manner:

Texture
Texture: TEXTURE_ID, data | url, string
Fonts
Font: FONT_FAMILY, style, base64_endoded_string

FONT_FAMILY is just the name you choose, it must not have a `,`in the name though.
Style must be either `regular`,`bold`,`italic` or `bold-italic`

#EVENT-Section

The events section contains rendering data and is the core of this format.
Each line is structured this way:
<START_TIME> - **<END_TIME>|<MACRO>|<NOTE>|<TEXT>**
<START_TIME> and **<END_TIME>** are timestamps for a time range when to
render.

Timestamp structure from right to left:
milliseconds -> point -> seconds -> colon -> minutes -> colon -> hours
Units don't have to be written completely, so following two examples are valid:
12:3:4.56
0
Hours limit is 99, but that should be enough for nearly all purposes.

Instead of **<START_TIME>** - **<END_TIME>** you could also define a single **<EVENT>**.

Event tags are just strings with single quotation-marks around them: 'event-tag'

<MACRO> is a style identifier name, mentioned in **#MACRO-Section**. The content of the chosen macro
will be prepended to **<TEXT>**. This is the base-macro for this line and can be seen as the base styling of the line.
<NOTE> is just a note for editors, nothing more.
<TEXT> is the description what to render. It's a combination of styling tags and geometries.
Tags are enclosed by brackets [], single ones parted by ;, everything else are geometries.
Additionally, content of macros can be inserted by \\<MACRO_NAME>\\. Insertions are limited by the renderer implementation.

Geometries

Geometries are the render source. Their appearance is influenced by styling tags. Different geometry types are usable:
Text
Plain text. Line breaks are to write as \n, [are to escape with \.
Example: Hello world!\n\[Hallo Welt!\]
Points
Floating point number pairs as center coordinates for pixels and circles. Example: 0 100 -50 -.125
Path
Description for a 2D graphics path. Segments of one type begin with a specifier, followed by necessary values.

Type	Specifier	Values per segment
Movement	m	1 target point
Line	l	1 end point
Bezier Curve	b	2 control points + 1 end point
Arc	a	1 control point + 1 number in degree

Example: *m 0 0 l 100 0 100 100.5 b 50 200 0 100 0 20 a 30 30 -45.5 c*

Tags