Substation Beta (SSB) v1.0

Modern subtitles

File format & syntax

File encoding: text-only; UTF-8
Line breaks: Windows (CRLF); Linux (LF)
Data layout: line-based
Ignore empty lines: yes
Comments: lines beginning with //
Syntax errors: Warnings display depends on implementation

Sections

SSB 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 following is recommend:

#INFO
Meta information.
#TARGET
Target frame dimensions and other properties.
# MACROS
Base macros to complement render data. Use this to create base styling for your events.
#EVENTS
Render data & conditions.
#RESOURCES
Describes all resources like textures and fonts.

Info section

The meta sections 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're no constraints but best practices:

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 differences 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
Influences geometries on 3D transformation. Positive Integer.
View
Perspective of the view. Can be 'orthogonal' or 'perspective'. Defaults is 'perspective'.

Macros section

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

An example:
default: [font=Open Sans;font-size=12]
green: [color=00FF00]

Usage:
5:0.0-2:5:0.0|default||${green}example text

Resources section

Usually defined at the end of the script due to possibly being very long. This sections 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,data|url,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'.

Events 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 -> DOT -> 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 an **<EVENT>** as an indicator when the line should be shown. Tags can be passed to the renderer to control which lines are shown at any given time.

<EVENT> tags are just strings with single quotation marks around them:
'event-id'

<MACRO> is a style identifier name, mentioned in **Macros section**. The content of the chosen macro will be prepended to **<TEXT>**. This is the base macro for this event and can be seen as the base styling of it.
<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 [], singles 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 controls points + 1 end point
Arc	a	1 control point + 1 number as degree
Close	c	

Example: *m 0 0 l 100 0 100 100.5 b 50 200 0 100 0 20 a 30 40 -45.5 c*

Tags

Tags define styling parameters for geometries or the type of themself. They come with a type name, followed by "=" and one or more values.

Font

font=Liberation Sans
  Name of font for text rendering.
size=20
  Geometry size (in pixels) depending on mode.
bold=n
  Font decoration.
  'y' for bold, 'n' for normal.
italic=n
underline=n
strikeout=n

Position

position=0,0 | 0,0,0
  Position on screen (2D or 3D).
alignment=20,30 | 5
  Alignment of geometry position.
  Can be either standard numpad alignments (1-9) or two numbers (0-100) in percentage.
margin=10 | 10,20,20,10
  Margin to screen edges. Either all edges in one number or each edge seperately (top, right, bottom, left).
margin-top=10
margin-right=20
margin-bottom=20
Margin-left=10
wrap-style=space
  Defines how the wrapper should handle breaking text into new lines. Can be 'space', 'character' or 'nowrap'.
direction=LTR
  Direction of text rendering.
  Can be LTR, RTL, TTB and BTT (left-to-right, right-to-left, top-to-bottom, bottom-to-top).
space=0 | 0,0
  Space between geometries/lines.
space-h=10
space-v=20
rotate=0,0,0
rotate-x=0
rotate-y=0
rotate-z=0
  Geometry rotation on target axis in degree.
scale=1 | 1,1 | 1,1,1
scale-x=1
scale-y=1
scale-z=1
  Geometry scale on target axis in percentage.
translate=0,0,0
translate-x=0
translate-y=0
translate-z=0
  Geometry translation on target in pixels.
shear=0,0
shear-x=0
shear-y=0
  Geometry shearing on target as weight.
matrix=1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1
  Matrix for transformation to be set.

Geometry

mode=text
  Mode can be 'text', 'shape' or 'points'.
border=2 | 2,2
  Border width of geometry.
border-h=2
border-v=2
join=round
  Border and line join style. Can be round, bevel or miter.
cap=round
  Border and line cap style. Can be round, butt or square.

Textures

texture=<RESOURCE_ID>
  Texture on geometry. Texturing can be enabled by valid RESOURCE_ID and disabled by an invalid one.
texfill=1,2,3,4 | 0,0,1,0,pad
Texture position (1,2) in percent, size (3,4) in percent and wrapping (pad).
  Wrapping can be pad, clamp, repeat and mirror.

Color

color=FFFFFF |
FFFFFF,FFFFFF |
FFFFFF,FFFFFF,FFFFFF |
FFFFFF,FFFFFF,FFFFFF,FFFFFF |
FFFFFF,FFFFFF,FFFFFF,FFFFFF,FFFFFF
  Color for geometry. Can be mono, left-to-right, left-to-mid-to-right, 4-corners gradient or 4-corners + center gradient.
bordercolor=FFFFFF |
FFFFFF,FFFFFF |
FFFFFF,FFFFFF,FFFFFF |
FFFFFF,FFFFFF,FFFFFF,FFFFFF |
FFFFFF,FFFFFF,FFFFFF,FFFFFF,FFFFFF
alpha=FF |
FF,FF |
FF,FF,FF |
FF,FF,FF,FF |
FF,FF,FF,FF,FF
  Alpha for geometry. Can be mono, left-to-right, left-to-mid-to-right, 4-corners gradient or 4-corners + center gradient.
borderalpha=FF |
FF,FF |
FF,FF,FF |
FF,FF,FF,FF |
FF,FF,FF,FF,FF
blur=0,0
  Gaussian blur on geometry and border.
blur-h=0
blur-v=0

Rastering

target=frame
  Swaps target on rendering.
  Can be frame or mask.
  Drawing on mask uses alpha value instead of color.
mask-mode=normal
  Defines how the mask is going to be applied on the current render process.
  Normal means every pixel with alpha=255 will not be visible, invert means every pixel with alpha=0 will not be visible.
mask-clear
  Resets all value in the mask with 0.
blend=overlay
  Blending mode.
  Can be 'add', 'subtract', 'multiply', 'invert', 'difference' or 'screen'.

Animation

animate=
[color=000000;translate-x=20]
animate=
[t,[color=000000;translate-x=20]]
animate=
[0,1000,sin(t*pi),[color=000000]]
  Interpolate between two values in a given timeframe with a specific acceleration function.

Karaoke

k=100
  Tag for how long a sylable is sung in milliseconds.
kset=0
  Reset karaoke time of event.
kcolor=FF00FF
  Color highlight of karaoke effect.

Example

```
#INFO
Title: My new project
Author: Youka
Version: 16.06.2012
Description: First concept of a new render format.

#TARGET
Width: 1280
Height: 720
Depth: 1000
View: perspective

#MACROS
Default: [bold=y]
Mine: [bold=n;color=FF0000]
Another: [Mine;position=100,200,-1;rotate-z=180]I'm a

#EVENTS
//0-2.0|||This line is a comment over 2 seconds!
2.0-5:0.0|Another|Hello, i'm a note!|red,    rotated\ntext over multiple lines.
5:0.0-2:5:0.0|Mine|Draw sth.|[mode=shape; texture=RAMEN]m 0 0 l 50.5 0 50.5 20.125 0 20.125
10:0.0-10:50:0.0||${Another}Lets scale some text to double its size!|[animate=[500, 1000, [scale=2]]This text is getting huge
20.0.0-21.0.0|||[font=MaterialIcon]some_circle_ligature
'show-something'|Default||This will only be shown when the event id is given

#RESOURCE
Texture: RAMEN,url,../ramen.tga
Font: MaterialIcon,regular,data,AAEAAAAKAIAAAwAgT1MvMnwMf9s...
```