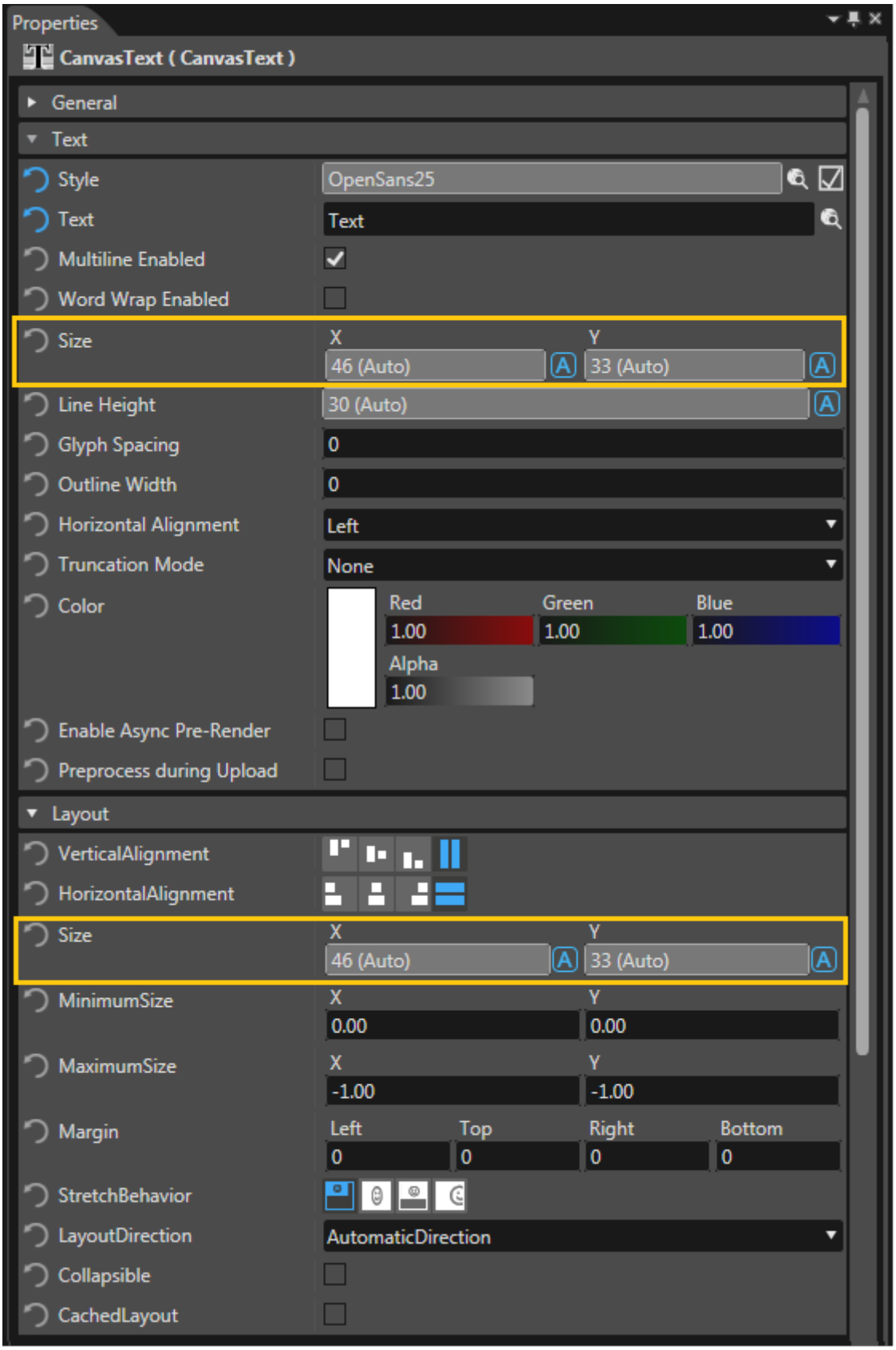
Canvas Text - Size Property

The Canvas Text provides a huge range of configuration possibilities. Some of them offer a great opportunity for customization but also increase the complexity level.
This section will look into one of the more complex features of the text: The ‘Size’ property and gives a short insight into the combination of size and other properties as the behavior will adjust accordingly.
In total there are up to three different size properties on each text:

1. The Text::Size property
2. The Layout::Size property
3. And an additional layout size (e.g. the column width or row height when using a grid layout)



The Text::Size (1.) property itself is mainly used to layout the text (text measuring) and provides text alignment, word wrapping and other text features to applications which have not set layout to enabled via CMake (CANDERA_LAYOUT_ENABLED is not set).
Aside from that, this is the most basic way to define the dimensions in which text is rendered.

For the two layout properties (2. and 3.) it is possible to set only one of them which will result in the same output.

E.g. setting the grid column width (3.) to 100 px is equal to setting the Layout::Size (2.) to 100 px. However, in the first variant the layout size is bound to the underlying layout, whereas in the second variant it is bound to the text.
If both are set, both will have impact on the layout. The grid column/row size (3.) will have higher priority when calculating the complete layout but the Layout::Size (2.) will still have some impact.

The Text::Size (1.) is the normal way to define a text's dimensions. This is comparable to setting the size of a SolidColorBrush effect. The reference image here shows exactly that: A SolidColorBrush effect (green) which has a greater width than the layout size of the grid (grey). Therefore, the second element overlaps the first one as it is laid out based on the layout dimensions and not the size dimensions. This effect is only visible when clipping is disabled. Otherwise, the SolidColorBrush will be clipped at the layout's size.

For the text, it will be rendered similarly into this area and then this block will be laid out. So overlapping might occur. Use cases:

1. Text is rendered ignoring layout settings – only using its own settings
2. Text should be within the layout boundaries as first requirement. And if Text::Size smaller than Layout::Size, it should be placed within Text::Size
3. Text should be positioned within the Layout::Size.

SceneComposer UI

The SceneComposer supports the user who is adjusting the size of the Layouter by providing a visual helper - an orange rectangle that makes it possible to visualize the configured dimensions.

Impact of size on different properties:

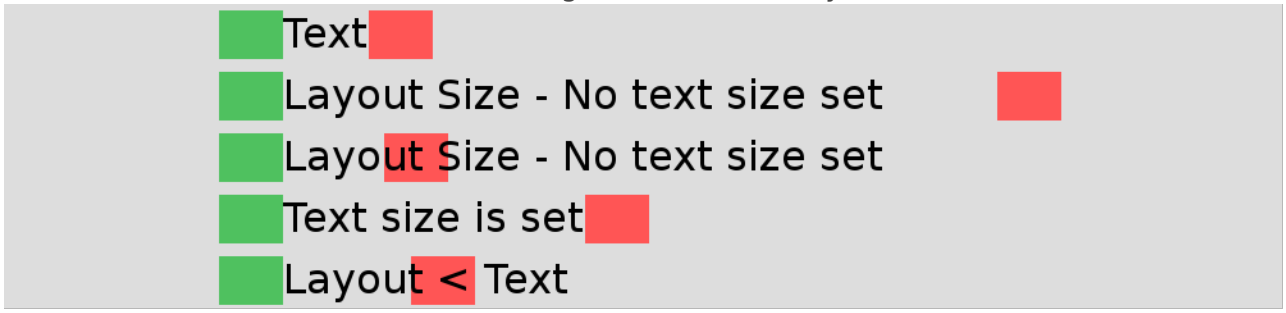
Alignment: The text has a property for horizontal alignment. This property aligns the text within Text::Size. If no Text::Size is set, the Layout::Size is used. The layout has a horizontal and vertical layout alignment property which is used to align the 'block' of the measured text within the Layout::Size. The defined layout alignment can be propagated to the text alignment when the text alignment is set to 'Auto'.
Otherwise, it is possible to right align the text within Text::Size but left align this block of right aligned text within Layout::Size. The properties will be inverted if the layout direction is set to 'right to left'.

Word wrap and truncation: Both properties prepare a text for use cases in which they do not fit into a size. Therefore, enabling one of them forces the text to fit into the provided smallest size (Layout::Size or Text::Size).

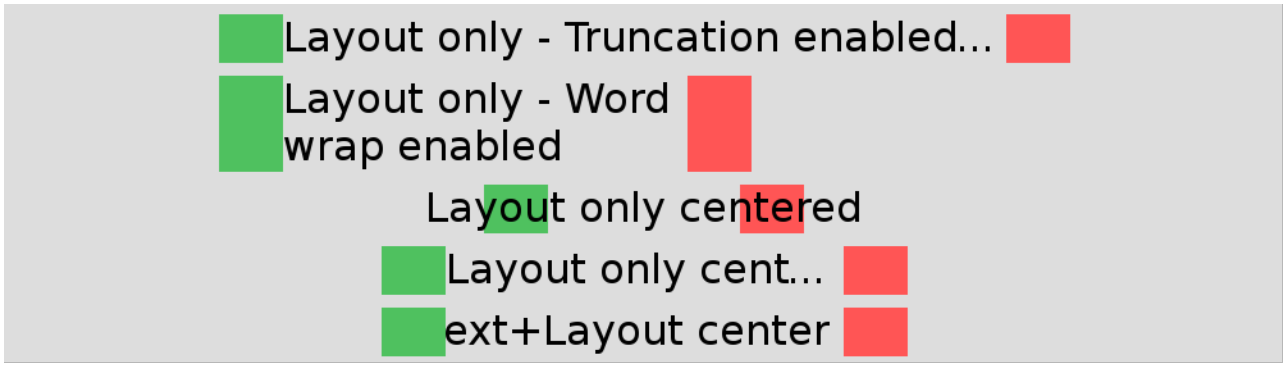
Examples

The following image shows some use cases to provide a better understanding of the layout behavior with different sizes. Every use case is based on a StackPanel with a CanvasSprite, a CanvasText and another CanvasSprite.

1. Default Text: No sizes set. The text is measured and the result is used for layout.
2. The layout size is set to a huger value than the text actually requires. This result is expected when the text is smaller than the layout size or the text size is limited to the shown size. The layout size defines the size for the layout in the end.
3. Same case as 2. But the layout size is smaller. There is no explicit restriction for the text itself.
   Therefore, the layout area is used for the layout process but the text will be drawn overlapping the other nodes of the layout as it requires more space.
4. Layout size is not set but text size is set to the actual size of the text. The layout uses this size to position the text.
5. Both sizes are set, but the Text::Size is greater than the layout size.



6. The layout size is set and Text::Size is not. The text would normally write over other elements. However, truncation is enabled.
7. Same as case as 6. but word wrap is enabled.
8. The Layout::Size is set and the layout alignment is center. It overlaps other elements but the text keeps being centered.
9. Same as 8. but truncation is enabled. Now the text is not centered anymore, as it requires more space than available. The beginning of the text is shown until the end of space is reached.
10. Layout::Size as well as Text::Size is set. Normally, the beginning of the text would be shown. However, the text alignment is also centered. Therefore, the center part of the text is visible.



Revision #2
Created 2023-02-27 08:00:20 UTC by Tsuyoshi.Kato
Updated 2023-02-28 01:05:53 UTC by Tsuyoshi.Kato