2D Layout

Description

In a typical user interface users need to distribute screen space to different elements. If the size of elements is dynamic, it's desired that the screen space distribution is done automatically based on the contents's natural size. To enable easy arrangement of elements, Candera2D supports layout functionalities.

Example Solution

- Layout2DSolution in folder *cgi_studio_player/content/Tutorials/Layout2DSolution.zip*

Candera::GridLayouter

Introduction

Basically, in SceneComposer for each Candera::Group2D a Candera::Layouter can be attached. The attached layouter defines the arrangements of the group's children.

Cell Spanning

Candera::GridLayouter provides a feature that allows joining of adjacent columns and rows into a larger single cell.

Cells can be joined horizontally, vertically or both. In order to achieve this, you must specify **Row Span** and **Column Span** properties for elements inside a Candera::GridLayouter:

- Row Span: Specifies the number of rows that the element should span across.
- Column Span: Specifies the number of columns that the element should span across.

> Default value for row and column span properties is 1.

Example

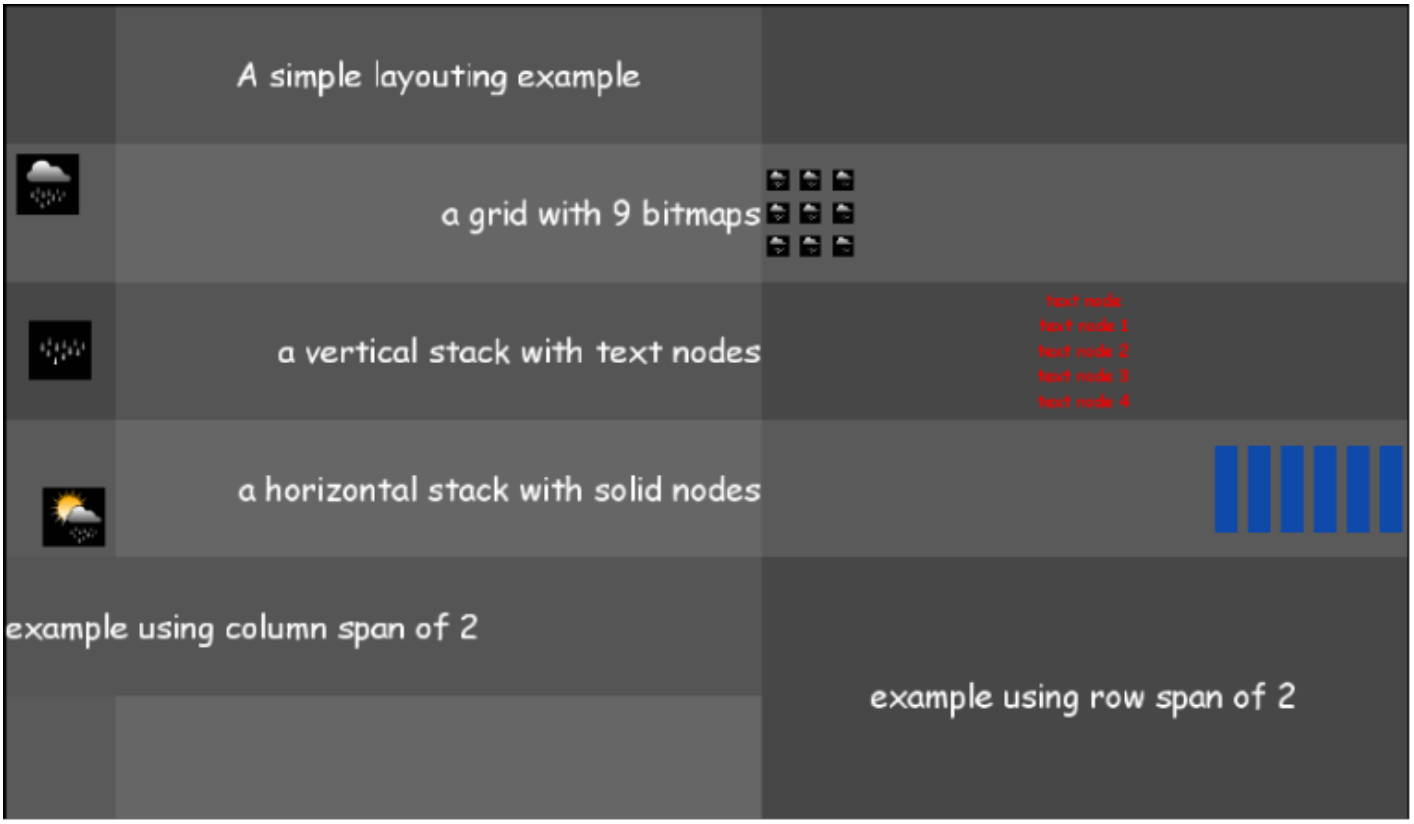
The **Layout2DSolution** provided in the content folder of *cgi_studio_player* gives an example how the 2D layout can be applied in SceneComposer. This section gives just a brief overview on layout

features and shows how it looks like.

In the example on the outermost group, a Candera::GridLayouter with 6 rows and 3 columns is defined.

All child nodes, here a bunch of bitmap-, text- and solid-nodes, are arranged along the grid. Each inner grid element can set various dynamic properties which define its size and alignment inside a grid cell.

Layouters can be nested. The outer grid contains a inner grid and two inner stack layouters (Candera::StackLayouter).



The example consists of a grid layouter with fixed size 1280 x 756 px.

- The **3 column's width** values are: 100 / 0,5 / 0,5.
  - The first column has a absolute width of 100 px,
  - the second and third column share the rest of the space (1180 px) relatively to each other (indicated by 0.0 <= value <= 1.0).
  - In fact the effective width column 2 and 3 is calculated by the layouter as follows: (1180 * 0,5) / (0,5 + 0,5) = 590 px.
- The **row heights** are defined as 126 / 0,1 / 0,1 / 0,1 / 0,1 / 0,1..
  - The first row has a absolute height of 126 px,
  - and the remaining height (630 px) is partitioned relatively to the other rows. Row height = (630 * 0,1) / (0,1 + 0,1 + 0,1 + 0,1 + 0,1) = 126 px.

> If a size value is set to -1 the width or height is set to the preferred size of the inner elements.

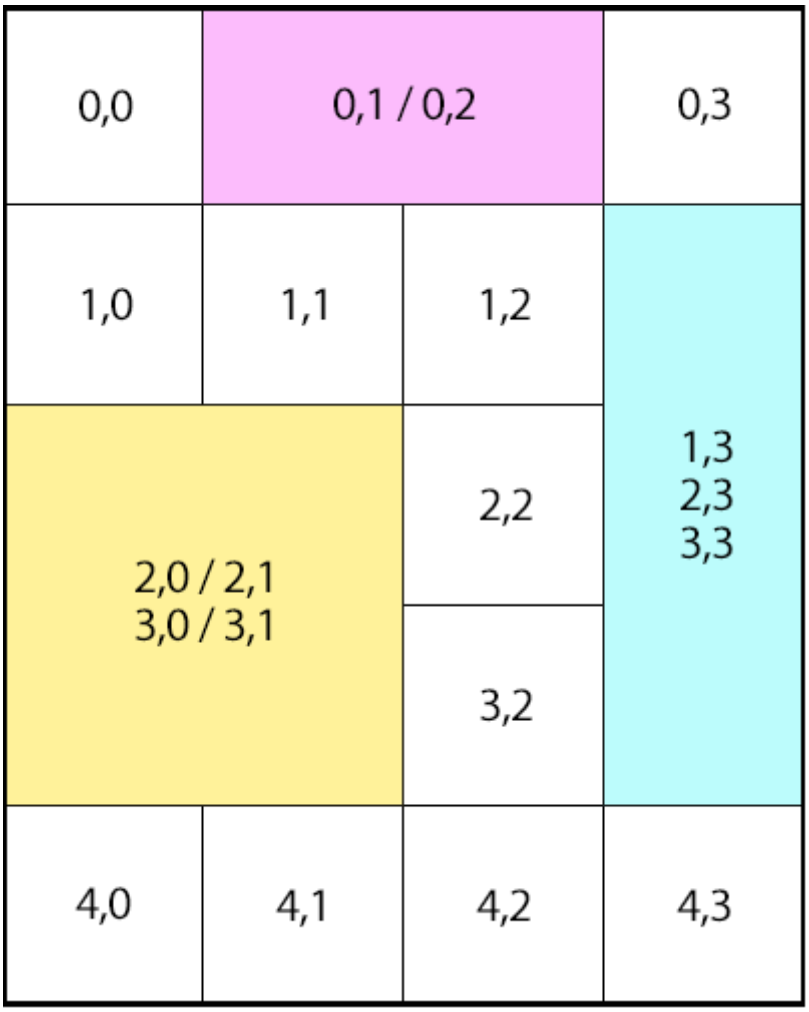
The bitmaps in the left column have different **alignments** (Top-Left, Centered, Bottom-Right). In the inner grid 9 bitmaps are scaled down (by the stretching property) to a fixed element size.

> The grey grid raster in the background is set up manually by solid nodes to illustrate the inner grid borders.

Cell Spanning Example

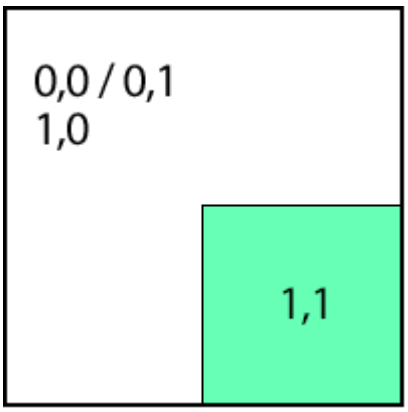
In the example, below a Candera::GridLayouter with 5 rows and 4 columns defined is used to exemplify how cell spanning is working:

- **Horizontal Spanning**: Element in Row(0) and Column(1) has a Column Span of 2.
- **Vertical Spanning**: Element in Row(1) and Column(3) has a Row Span of 3.
- **Horizontal and Vertical Spanning**: Element in Row(2) and Column(0) has a Row Span of 2 and Column Span of 2.



> Please note that single cells might still be used for other elements if needed, like in the following example:

> A Candera::GridLayouter with 2 rows and 2 columns is defined. Element in (0,0) has a row and column span of 2, but element at (1,1) is still accessible.



See also:

- Grid Layouter in SceneComposer User Manual

Candera::BaseLineLayouter

Introduction

Candera::BaseLineLayouter behaves similar to a Horizontal StackLayouter but instead of aligning the objects to the client area of the Layouter, it aligns them to a line.

The idea is to be able to align text with different sizes in different ways.

- For instance it can align all the text by the top of its bounding box (VTop).
- It can align the text by taking into consideration the bottom of the bounding box (VBottom).
- Or, the most interesting feature it can align the items with different size by their **base line** (VCenter).

Example

The **Layout2DSolution** provided in the content folder of *cgi_studio_player* gives an example how the 2D BaseLineLayouter can be applied in SceneComposer.

- In the Scene2DBaseLineLayouter scene, on three groups, a Candera::BaseLineLayouter is defined.
- The base line offset configured on the groups is visualized by an orange line.
- The text parts of each group are defined with "VTop", "VBottom", or "VCenter", for the Vertical Alignment attribute.

The baseline can be either fixed or automatic. The automatic baseline is configured such that all the children fit from the top of the client area.

See also:

- Baseline Layouter in SceneComposer User Manual

Candera::OverlayLayouter

Introduction

A Candera::OverlayLayouter is a container that layouts enfolded nodes by overlaying them in same place in a sorted sequence. With this layouter, elements can be arranged that they fulfill a given area and lay on top of one another.

Example

The **Layout2DSolution** provided in the content folder of *cgi_studio_player* gives an example how the OverlayLayouter can be applied in SceneComposer. Some important use cases are:

- The outer group represents the given area, in which the nodes are arranged. You can set the size of the group to define the height and the width of the area. If you don't set any size, it will automatically be set to the size of the biggest inner child element.
- The sequence of the nodes in the overlay group is important. In the example the Inner_Overlay_Group is displayed in front of the Outer_Overlay_Group.
- Three very small images (mainBmp, gradient_verticalBmp, gradient_horizontalBmp, see in ExampleSolution) are stretched and blended to create a big image. They fulfill the given area.
- The text nodes are aligned to the center and in the corners. You can set several vertical and horizontal alignments for each node.

See also:

- Overlay Layouter in SceneComposer User Manual

Candera::DockPanelLayouter

Introduction

The Candera::DockPanelLayouter provides an easy docking of elements to the left, right, top, bottom or center of the panel. To dock an element to the center of the panel, it must be the last child of the panel.

Example

The **Layout2DSolution** provided in the content folder of *cgi_studio_player* gives an example how the DockPanelLayouter can be applied in SceneComposer.

See also:

- Candera::DockPanelLayouter
- DockPanel Layouter in SceneComposer User Manual

Automatic Resize Depending on Content

Introduction

This tutorial will show how to create resizable dialog popups using a hierarchy of groups and different kind of nodes (bitmap, solid color and text) arranged specifically using layouters.

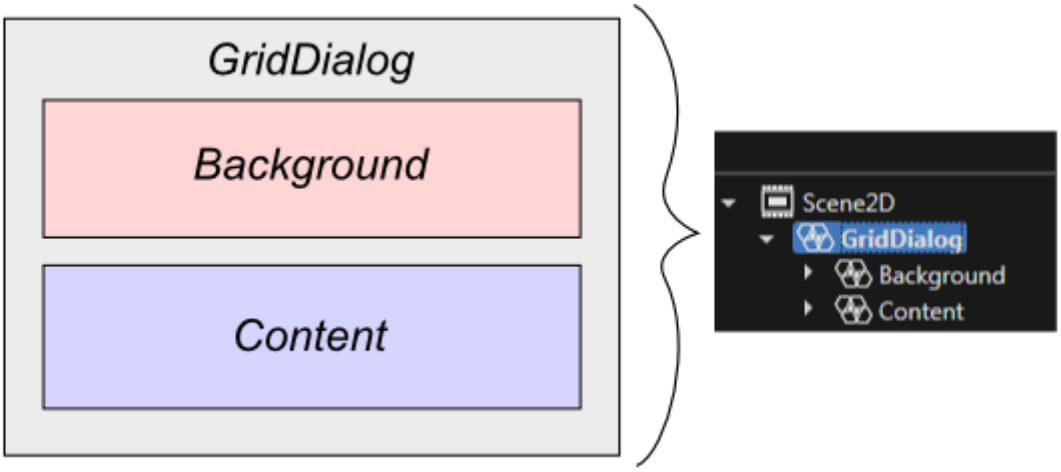
Examples

Resizable Dialog Example Using Grid Layouter

For this example the dialog elements are arranged using Grid and Overlay layouts.

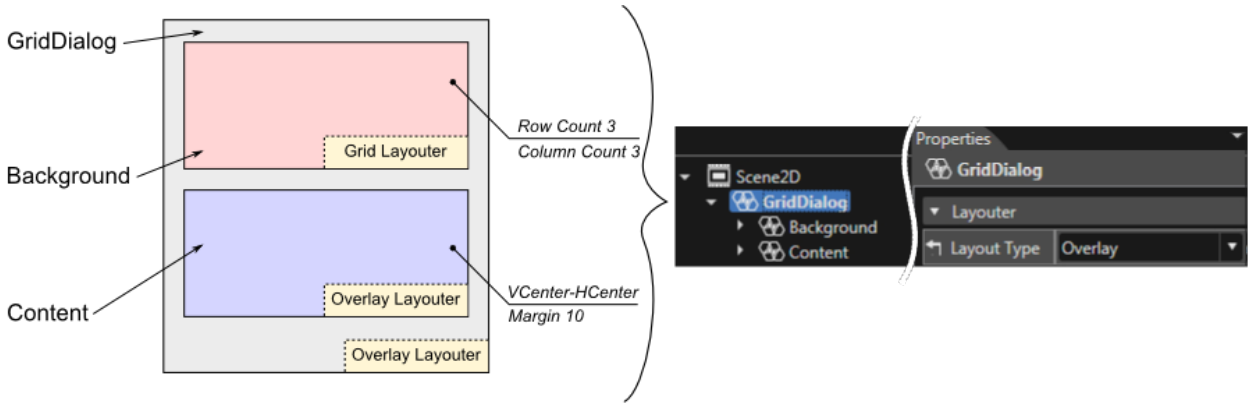
Group Hierarchy

Create the parent group for the pop-up and name it GridDialog. Create another two groups (Background and Content) and add them as children of GridDialog group.

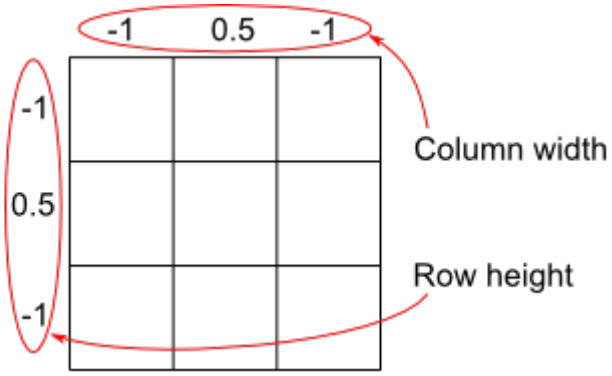


Layout Configuration

Set the layout for each group as shown in the image below:



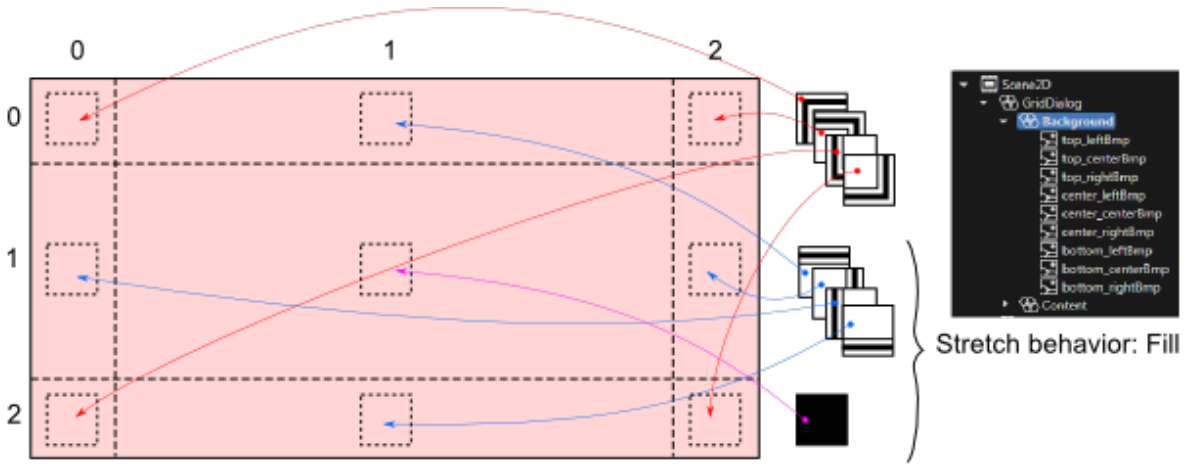
Configure the column width and row height:



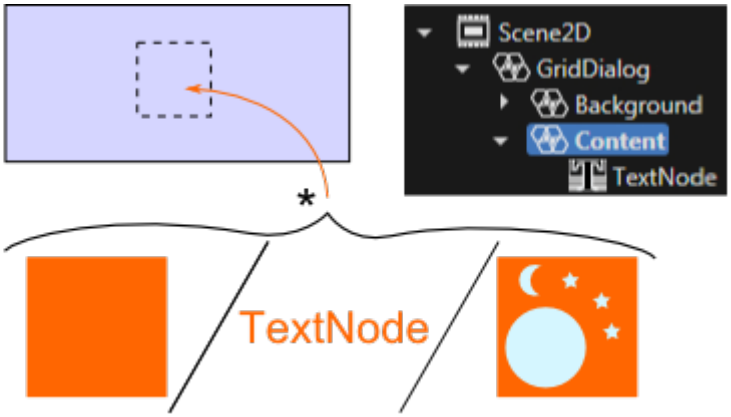
In SceneComposer this can be achieved in the "Grid Layout Editor" (right click on the Background group and select "Configure Layout").

Adding Nodes

The Background group should be populated with bitmap or solid color nodes with the following settings:



The Content group might be filled with one or more nodes of any type, as desired:



Result

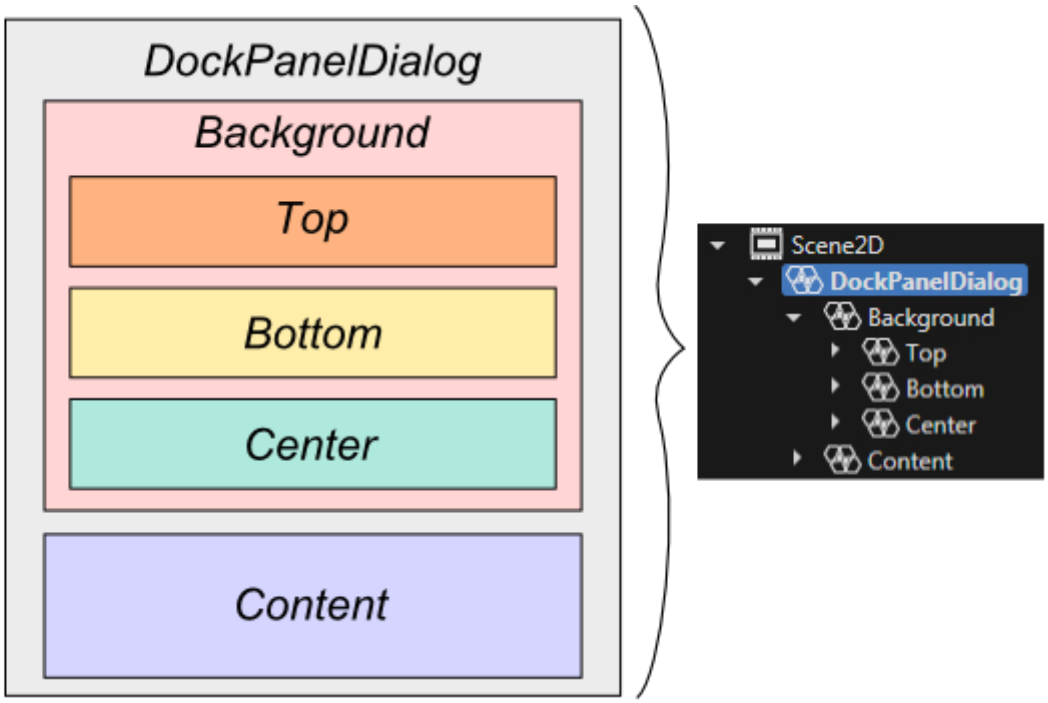
After putting everything together the dialog will look as above:

Resizable Dialog Example Using DockPanel Layouter

For this example the dialog elements are arranged using DockPanel and Overlay layouts.

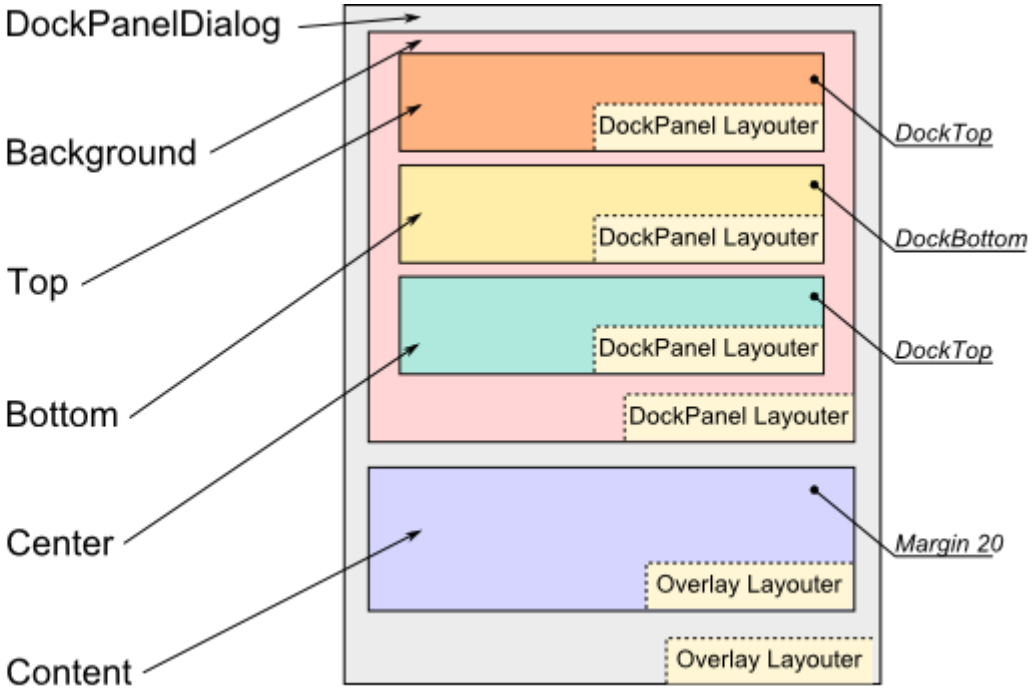
Group Hierarchy

Create the following structure of groups:



Layout Configuration

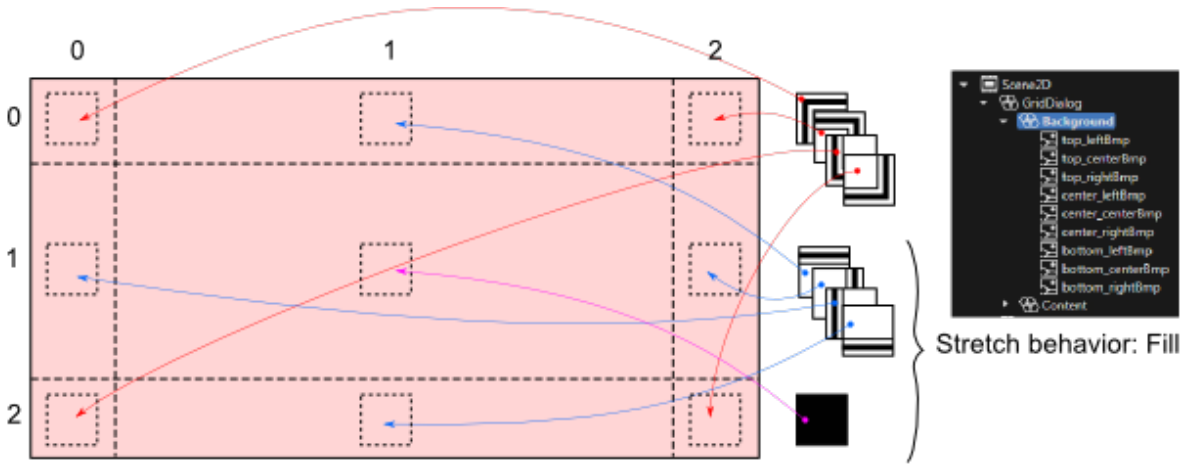
Set the layout for each group as shown in the image below:



> Be sure the order of adding the children groups in the Background group is Top - Bottom - Center otherwise the elements will not be arranged correctly.

Adding Nodes

The children groups of Background should be populated with bitmap or solid color nodes with the following settings:



> Be sure the order of nodes in the groups is Left - Right - Center otherwise the elements will not be arranged correctly.

The Content group should be configured as explained in the previous chapter for the dialog based on grid layout with the only difference that the text node will be replaced with a bitmap. Please see the Content group part from the Adding Nodes section.

Result

After putting everything together the dialog will look as above:

Revision #9
Created 2023-02-21 06:00:36 UTC by Tsuyoshi.Kato
Updated 2025-01-10 10:34:00 UTC by Elisabeth Zauner