CardiacModelGenerator Detailed Design 0.0.1

Generated by Doxygen 1.12.0

Chapter 1

Namespace Index

1.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 CardiacModelGenerator Namespace Reference

Classes

- class CardiacMeshalyzer
- class CleanTetraMeshOptions
- class HomePage
- class PointCloudOptions
- class StartPage

Functions

- generate_point_cloud (coords1=None, masks1=None, coords2=None, masks2=None, coords3=None, masks3=None, whichmask=1, tol=0.1, colormap_name="viridis", point_size=5)
- generate_tetra_mesh (point_cloud_cleaned)
- clean_tetra_mesh (grid, subdivisions=2, poisson_iterations=10, clean_tolerance=0.001, quality_↩ threshold=1e-5)
- get_cell_quality (final_volumetric_mesh)

Variables

- app = wx.App(False)
- frame = CardiacMeshalyzer(None)

5.1.1 Detailed Description

```
Created on Tue Dec  3 17:35:28 2024

@author: vinayjani
```

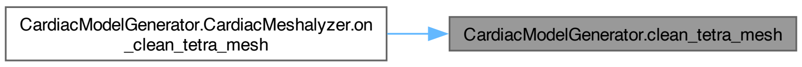
5.1.2 Function Documentation

5.1.2.1 clean_tetra_mesh()

```
CardiacModelGenerator.clean_tetra_mesh (
              grid,
              subdivisions = 2,
              poisson_iterations = 10,
              clean_tolerance = 0.001,
              quality_threshold = 1e-5)
```

```
@brief Cleans and smooths a tetrahedral mesh to improve quality and remove artifacts.
@param grid A PyVista UnstructuredGrid object representing the input volumetric mesh.
@param subdivisions Number of subdivisions for mesh refinement (default: 2).
@param poisson_iterations Number of smoothing iterations (default: 10).
@param clean_tolerance Tolerance for cleaning the mesh (default: 0.001).
@param quality_threshold Minimum acceptable quality for mesh cells (default: 1e-5).
@return A PyVista PolyData object representing the cleaned and smoothed surface mesh.
```

Here is the caller graph for this function:


CardiacModelGenerator.CardiacMeshalyzer.on_clean_tetra_mesh
CardiacModelGenerator.clean_tetra_mesh

5.1.2.2 generate_point_cloud()

```
CardiacModelGenerator.generate_point_cloud (
              coords1 = None,
              masks1 = None,
              coords2 = None,
              masks2 = None,
              coords3 = None,
              masks3 = None,
              whichmask = 1,
              tol = 0.1,
              colormap_name = "viridis",
              point_size = 5)
```

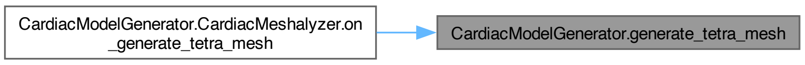
```
@brief Generates and visualizes a point cloud from up to three coordinate-mask pairs.
@param coords1 Optional. First set of coordinates (3D array).
@param masks1 Optional. First set of masks corresponding to coords1.
@param coords2 Optional. Second set of coordinates (3D array).
@param masks2 Optional. Second set of masks corresponding to coords2.
@param coords3 Optional. Third set of coordinates (3D array).
@param masks3 Optional. Third set of masks corresponding to coords3.
@param whichmask Mask value to extract points (default: 1).
@param tol Tolerance for cleaning the point cloud (default: 0.1).
@param colormap_name Colormap used for coloring the point cloud (default: "viridis").
@param point_size Size of the points in the visualization (default: 5).
@return A PyVista PolyData object representing the cleaned point cloud.
```

5.1.2.3 generate_tetra_mesh()

```
CardiacModelGenerator.generate_tetra_mesh (
              point_cloud_cleaned)
```

```
@brief Generates a tetrahedral mesh from a cleaned point cloud.
@param point_cloud_cleaned A PyVista PolyData object representing the cleaned point cloud.
@return A PyVista UnstructuredGrid object representing the generated tetrahedral mesh.
```

Here is the caller graph for this function:

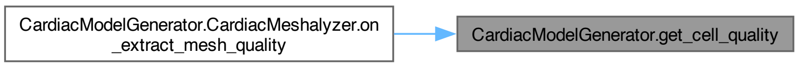

CardiacModelGenerator.CardiacMeshalyzer.on_generate_tetra_mesh
CardiacModelGenerator.generate_tetra_mesh

5.1.2.4 get_cell_quality()

```
CardiacModelGenerator.get_cell_quality (
              final_volumetric_mesh)
```

```
@brief Computes and visualizes cell quality for a tetrahedral mesh.
@param final_volumetric_mesh A PyVista UnstructuredGrid object representing the volumetric mesh.
@return A PyVista UnstructuredGrid object with cell quality values added as a scalar field.
```

Here is the caller graph for this function:


CardiacModelGenerator.CardiacMeshalyzer.on_extract_mesh_quality
CardiacModelGenerator.get_cell_quality

5.1.3 Variable Documentation

5.1.3.1 app

```
CardiacModelGenerator.app = wx.App(False)
```

5.1.3.2 frame

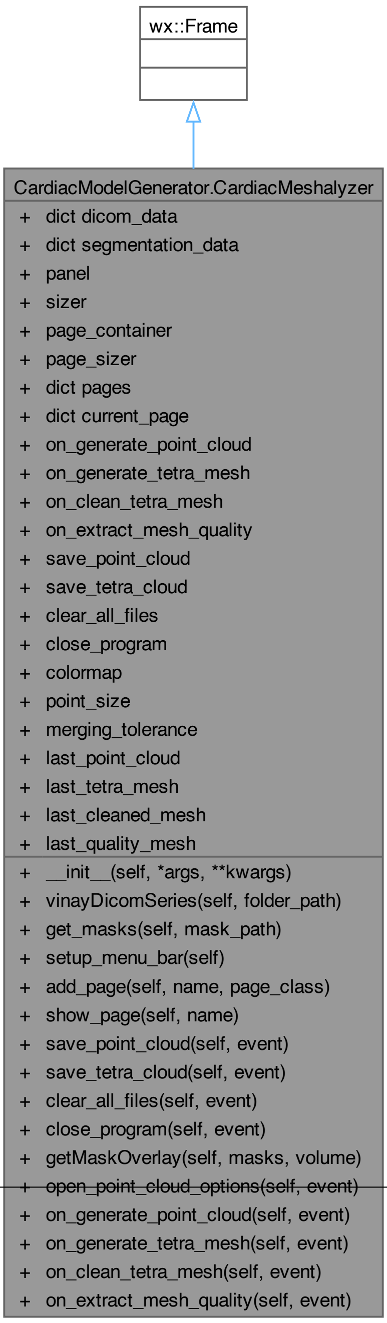
```
CardiacModelGenerator.frame = CardiacMeshalyzer(None)
```

Chapter 6

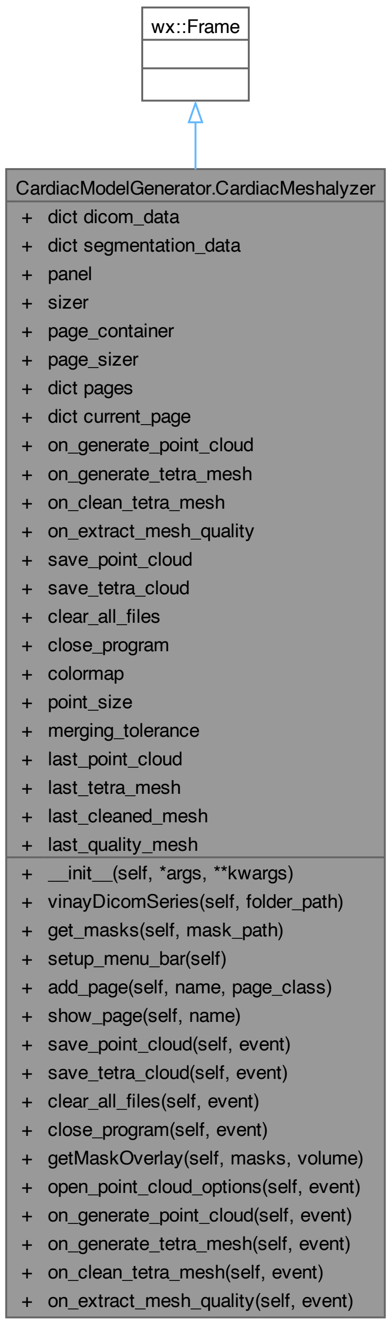
Class Documentation

6.1 CardiacModelGenerator.CardiacMeshalyzer Class Reference

Inheritance diagram for CardiacModelGenerator.CardiacMeshalyzer:


wx::Frame
CardiacModelGenerator.CardiacMeshalyzer
+ dict dicom_data
+ dict segmentation_data
+ panel
+ sizer
+ page_container
+ page_sizer
+ dict pages
+ dict current_page
+ on_generate_point_cloud
+ on_generate_tetra_mesh
+ on_clean_tetra_mesh
+ on_extract_mesh_quality
+ save_point_cloud
+ save_tetra_cloud
+ clear_all_files
+ close_program
+ colormap
+ point_size
+ merging_tolerance
+ last_point_cloud
+ last_tetra_mesh
+ last_cleaned_mesh
+ last_quality_mesh
+ __init__(self, *args, **kwargs)
+ vinayDicomSeries(self, folder_path)
+ get_masks(self, mask_path)
+ setup_menu_bar(self)
+ add_page(self, name, page_class)
+ show_page(self, name)
+ save_point_cloud(self, event)
+ save_tetra_cloud(self, event)
+ clear_all_files(self, event)
+ close_program(self, event)
+ getMaskOverlay(self, masks, volume)
+ open_point_cloud_options(self, event)
+ on_generate_point_cloud(self, event)
+ on_generate_tetra_mesh(self, event)
+ on_clean_tetra_mesh(self, event)
+ on_extract_mesh_quality(self, event)

Collaboration diagram for CardiacModelGenerator.CardiacMeshalyzer:


wx::Frame
CardiacModelGenerator.CardiacMeshalyzer
+ dict dicom_data
+ dict segmentation_data
+ panel
+ sizer
+ page_container
+ page_sizer
+ dict pages
+ dict current_page
+ on_generate_point_cloud
+ on_generate_tetra_mesh
+ on_clean_tetra_mesh
+ on_extract_mesh_quality
+ save_point_cloud
+ save_tetra_cloud
+ clear_all_files
+ close_program
+ colormap
+ point_size
+ merging_tolerance
+ last_point_cloud
+ last_tetra_mesh
+ last_cleaned_mesh
+ last_quality_mesh
+ __init__(self, *args, **kwargs)
+ vinayDicomSeries(self, folder_path)
+ get_masks(self, mask_path)
+ setup_menu_bar(self)
+ add_page(self, name, page_class)
+ show_page(self, name)
+ save_point_cloud(self, event)
+ save_tetra_cloud(self, event)
+ clear_all_files(self, event)
+ close_program(self, event)
+ getMaskOverlay(self, masks, volume)
+ open_point_cloud_options(self, event)
+ on_generate_point_cloud(self, event)
+ on_generate_tetra_mesh(self, event)
+ on_clean_tetra_mesh(self, event)
+ on_extract_mesh_quality(self, event)

Public Member Functions

- __init__ (self, ∗args, ∗∗kwargs)
- vinayDicomSeries (self, folder_path)
- get_masks (self, mask_path)
- setup_menu_bar (self)
- add_page (self, name, page_class)

- show_page (self, name)
- save_point_cloud (self, event)
- save_tetra_cloud (self, event)
- clear_all_files (self, event)
- close_program (self, event)
- getMaskOverlay (self, masks, volume)
- open_point_cloud_options (self, event)
- on_generate_point_cloud (self, event)
- on_generate_tetra_mesh (self, event)
- on_clean_tetra_mesh (self, event)
- on_extract_mesh_quality (self, event)

Public Attributes

- dict dicom_data = {}
- dict segmentation_data = {}
- panel = wx.Panel(self)
- sizer = wx.BoxSizer(wx.VERTICAL)
- page_container = wx.Panel(self.panel)
- page_sizer = wx.BoxSizer(wx.VERTICAL)
- dict pages = {}
- dict current_page = None
- on_generate_point_cloud
- on_generate_tetra_mesh
- on_clean_tetra_mesh
- on_extract_mesh_quality
- save_point_cloud
- save_tetra_cloud
- clear_all_files
- close_program
- colormap = dialog.colormap
- point_size = dialog.point_size
- merging_tolerance = dialog.merging_tolerance
- last_point_cloud
- last_tetra_mesh = tetra_mesh
- last_cleaned_mesh = cleaned_mesh
- last_quality_mesh = quality_mesh

6.1.1 Detailed Description

```
@class CardiacMeshalyzer
@brief GUI application for managing and processing cardiac imaging data.
@details This class provides a graphical user interface (GUI) for handling DICOM series, generating point clou
         creating tetrahedral meshes, and performing mesh cleaning and quality assessment.

@uml
@startuml
class CardiacMeshalyzer {
    - dicom_data : dict
    - segmentation_data : dict
    - panel : wx.Panel
    - sizer : wx.BoxSizer
    - page_container : wx.Panel
    - page_sizer : wx.BoxSizer
    - pages : dict
    - current_page : object
    + __init__(*args, **kwargs)
    + vinayDicomSeries(folder_path : str) : tuple
    + get_masks(mask_path : str) : numpy.ndarray
    + setup_menu_bar()
    + add_page(name : str, page_class : type)
    + show_page(name : str)
    + save_point_cloud(event : wx.Event)
    + save_tetra_cloud(event : wx.Event)
    + clear_all_files(event : wx.Event)
    + close_program(event : wx.Event)
    + getMaskOverlay(masks : numpy.ndarray, volume : numpy.ndarray) : numpy.ndarray
    + open_point_cloud_options(event : wx.Event)
    + on_generate_point_cloud(event : wx.Event)
    + on_generate_tetra_mesh(event : wx.Event)
    + on_clean_tetra_mesh(event : wx.Event)
    + on_extract_mesh_quality(event : wx.Event)
}
@enduml
```

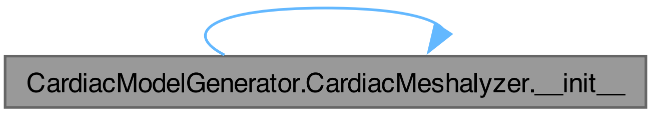
6.1.2 Constructor & Destructor Documentation

6.1.2.1 __init__()

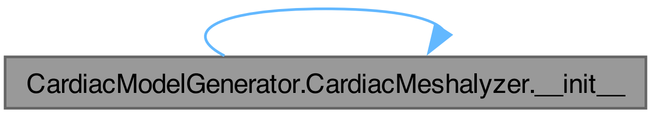
```
CardiacModelGenerator.CardiacMeshalyzer.__init__ (
              self,
             * args,
             ** kwargs)
```

```
@brief Initializes the CardiacMeshalyzer GUI application.
@param *args Positional arguments for the wx.Frame superclass.
@param **kwargs Keyword arguments for the wx.Frame superclass.
@details Sets up the GUI components, initializes global variables, and adds the Start Page and Home Page to th
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3 Member Function Documentation

6.1.3.1 add_page()

```
CardiacModelGenerator.CardiacMeshalyzer.add_page (
              self,
              name,
              page_class)
```

```
@brief Adds a new page to the GUI.
@param name The name of the page to be added.
@param page_class The class representing the page to add.
@details Creates an instance of the specified page class, adds it to the page container, and hides it initiall
```

6.1.3.2 clear_all_files()

```
CardiacModelGenerator.CardiacMeshalyzer.clear_all_files (
              self,
              event)
```

```
@brief Clears all loaded DICOM data and segmentation files.
@param event The wxPython event triggering this action.
@details Resets the dictionaries holding DICOM and segmentation data and displays a confirmation message.
```

6.1.3.3 close_program()

```
CardiacModelGenerator.CardiacMeshalyzer.close_program (
              self,
              event)
```

```
@brief Closes the application.
@param event The wxPython event triggering this action.
```

6.1.3.4 get_masks()

```
CardiacModelGenerator.CardiacMeshalyzer.get_masks (
              self,
              mask_path)
```

```
@brief Loads segmentation masks from a specified file.
@param mask_path The file path to the segmentation mask in NIfTI format.
@return A NumPy array containing the segmentation mask data.
@details Reads the segmentation mask from the provided NIfTI file and converts it to a NumPy array for further
```

6.1.3.5 getMaskOverlay()

```
CardiacModelGenerator.CardiacMeshalyzer.getMaskOverlay (
              self,
              masks,
              volume)
```

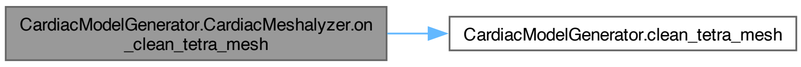
```
@brief Generates an overlay of the segmentation mask on the volume.
@param masks A NumPy array containing the segmentation masks.
@param volume A NumPy array representing the image volume.
@return A NumPy array containing the overlay, where the segmentation masks are blended with the volume.
@details This method normalizes the volume data, assigns colors to the segmentation masks, and blends them wit
         to create a visual overlay. Handles up to 4 predefined segmentation classes and generates random colo
@throws ValueError If the volume data is not numeric or not a NumPy array.
```

6.1.3.6 on_clean_tetra_mesh()

```
CardiacModelGenerator.CardiacMeshalyzer.on_clean_tetra_mesh (
              self,
              event)
```

```
@brief Handles the "Clean Tetra Mesh" menu option.
@param event The wxPython event triggering this action.
@details Opens a dialog to configure cleaning options and applies these settings to clean the tetrahedral mesh
         Visualizes the cleaned mesh upon successful completion. Displays an error message if no tetrahedral m
```

Here is the call graph for this function:

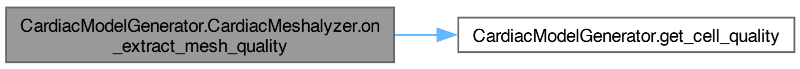

CardiacModelGenerator.CardiacMeshalyzer.on_clean_tetra_mesh
CardiacModelGenerator.clean_tetra_mesh

6.1.3.7 on_extract_mesh_quality()

```
CardiacModelGenerator.CardiacMeshalyzer.on_extract_mesh_quality (
              self,
              event)
```

```
@brief Handles the "Extract Mesh Quality" menu option.
@param event The wxPython event triggering this action.
@details Computes the quality of cells in the tetrahedral mesh. Stores the quality mesh for future use
         and displays a success message upon completion. Displays an error message if no cleaned mesh exists.
```

Here is the call graph for this function:


CardiacModelGenerator.CardiacMeshalyzer.on_extract_mesh_quality
CardiacModelGenerator.get_cell_quality

6.1.3.8 on_generate_point_cloud()

```
CardiacModelGenerator.CardiacMeshalyzer.on_generate_point_cloud (
              self,
              event)
```

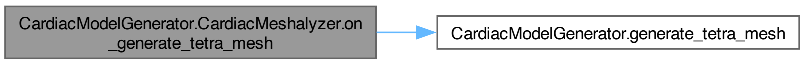
```
@brief Handles the "Generate Point Cloud" menu option.
@param event The wxPython event triggering this action.
@details Collects DICOM coordinate and mask data, opens the Point Cloud Options dialog for user input, and
         generates a point cloud using the specified options. If an error occurs during point cloud generatior
         an error message is displayed.
```

6.1.3.9 on_generate_tetra_mesh()

```
CardiacModelGenerator.CardiacMeshalyzer.on_generate_tetra_mesh (
              self,
              event)
```

```
@brief Handles the "Generate Tetra Mesh" menu option.
@param event The wxPython event triggering this action.
@details Generates a tetrahedral mesh from the last generated point cloud.
         Displays an error message if no point cloud exists.
```

Here is the call graph for this function:


CardiacModelGenerator.CardiacMeshalyzer.on_generate_tetra_mesh
CardiacModelGenerator.generate_tetra_mesh

6.1.3.10 open_point_cloud_options()

```
CardiacModelGenerator.CardiacMeshalyzer.open_point_cloud_options (
              self,
              event)
```

```
@brief Opens the Point Cloud Options dialog.
@param event The wxPython event triggering this action.
@details Displays a dialog to configure point cloud generation options, including colormap, point size, and me
         If the user confirms their selection, the `generate_point_cloud` method is called with the selected c
```

6.1.3.11 save_point_cloud()

```
CardiacModelGenerator.CardiacMeshalyzer.save_point_cloud (
              self,
              event)
```

```
@brief Saves the generated point cloud to a file.
@param event The wxPython event triggering this action.
@details Displays an informational message. The save functionality is not yet implemented.
```

6.1.3.12 save_tetra_cloud()

```
CardiacModelGenerator.CardiacMeshalyzer.save_tetra_cloud (
              self,
              event)
```

```
@brief Saves the generated tetrahedral mesh to a file.
@param event The wxPython event triggering this action.
@details Displays an informational message. The save functionality is not yet implemented.
```

6.1.3.13 setup_menu_bar()

```
CardiacModelGenerator.CardiacMeshalyzer.setup_menu_bar (
              self)
```

```
@brief Configures and initializes the menu bar for the application.
@details Creates a menu bar with options for loading images, processing models, and saving/clearing data.
         Each menu item is bound to the corresponding event handler method.
```

6.1.3.14 show_page()

```
CardiacModelGenerator.CardiacMeshalyzer.show_page (
              self,
              name)
```

```
@brief Displays the specified page in the GUI.
@param name The name of the page to display.
@details Hides the currently visible page (if any) and shows the specified page.
```

6.1.3.15 vinayDicomSeries()

```
CardiacModelGenerator.CardiacMeshalyzer.vinayDicomSeries (
              self,
              folder_path)
```

```
@brief Processes a series of DICOM files in a given folder.
@param folder_path The path to the folder containing DICOM files.
@return A tuple containing:
    - Volume: A 3D NumPy array representing the reconstructed image volume.
    - Coords: A 4D NumPy array with the absolute (x, y, z) coordinates for each pixel.
    - out_images: A NumPy array containing the original DICOM objects.
@details Reads DICOM files from the specified folder, reconstructs the image volume, and computes the absolute
         coordinates for each pixel using the image’s orientation, spacing, and position metadata.
```

6.1.4 Member Data Documentation

6.1.4.1 clear_all_files

```
CardiacModelGenerator.CardiacMeshalyzer.clear_all_files
```

6.1.4.2 close_program

```
CardiacModelGenerator.CardiacMeshalyzer.close_program
```

6.1.4.3 colormap

```
CardiacModelGenerator.CardiacMeshalyzer.colormap = dialog.colormap
```

6.1.4.4 current_page

```
dict CardiacModelGenerator.CardiacMeshalyzer.current_page = None
```

6.1.4.5 dicom_data

```
dict CardiacModelGenerator.CardiacMeshalyzer.dicom_data = {}
```

6.1.4.6 last_cleaned_mesh

```
CardiacModelGenerator.CardiacMeshalyzer.last_cleaned_mesh = cleaned_mesh
```

6.1.4.7 last_point_cloud

```
CardiacModelGenerator.CardiacMeshalyzer.last_point_cloud
```

Initial value:

```
=  generate_point_cloud(
                      coords1=coords1, masks1=masks1,
                      coords2=coords2, masks2=masks2,
                      coords3=coords3, masks3=masks3,
                      whichmask=whichmask,
                      tol=merging_tolerance,
                      colormap_name=colormap_name,
                      point_size=point_size
                )
```

6.1.4.8 last_quality_mesh

```
CardiacModelGenerator.CardiacMeshalyzer.last_quality_mesh = quality_mesh
```

6.1.4.9 last_tetra_mesh

```
CardiacModelGenerator.CardiacMeshalyzer.last_tetra_mesh = tetra_mesh
```

6.1.4.10 merging_tolerance

```
CardiacModelGenerator.CardiacMeshalyzer.merging_tolerance = dialog.merging_tolerance
```

6.1.4.11 on_clean_tetra_mesh

```
CardiacModelGenerator.CardiacMeshalyzer.on_clean_tetra_mesh
```

6.1.4.12 on_extract_mesh_quality

```
CardiacModelGenerator.CardiacMeshalyzer.on_extract_mesh_quality
```

6.1.4.13 on_generate_point_cloud

```
CardiacModelGenerator.CardiacMeshalyzer.on_generate_point_cloud
```

6.1.4.14 on_generate_tetra_mesh

```
CardiacModelGenerator.CardiacMeshalyzer.on_generate_tetra_mesh
```

6.1.4.15 page_container

```
CardiacModelGenerator.CardiacMeshalyzer.page_container = wx.Panel(self.panel)
```

6.1.4.16 page_sizer

```
CardiacModelGenerator.CardiacMeshalyzer.page_sizer = wx.BoxSizer(wx.VERTICAL)
```

6.1.4.17 pages

```
dict CardiacModelGenerator.CardiacMeshalyzer.pages = {}
```

6.1.4.18 panel

```
CardiacModelGenerator.CardiacMeshalyzer.panel = wx.Panel(self)
```

6.1.4.19 point_size

```
CardiacModelGenerator.CardiacMeshalyzer.point_size = dialog.point_size
```

6.1.4.20 save_point_cloud

```
CardiacModelGenerator.CardiacMeshalyzer.save_point_cloud
```

6.1.4.21 save_tetra_cloud

```
CardiacModelGenerator.CardiacMeshalyzer.save_tetra_cloud
```

6.1.4.22 segmentation_data

```
dict CardiacModelGenerator.CardiacMeshalyzer.segmentation_data = {}
```

6.1.4.23 sizer

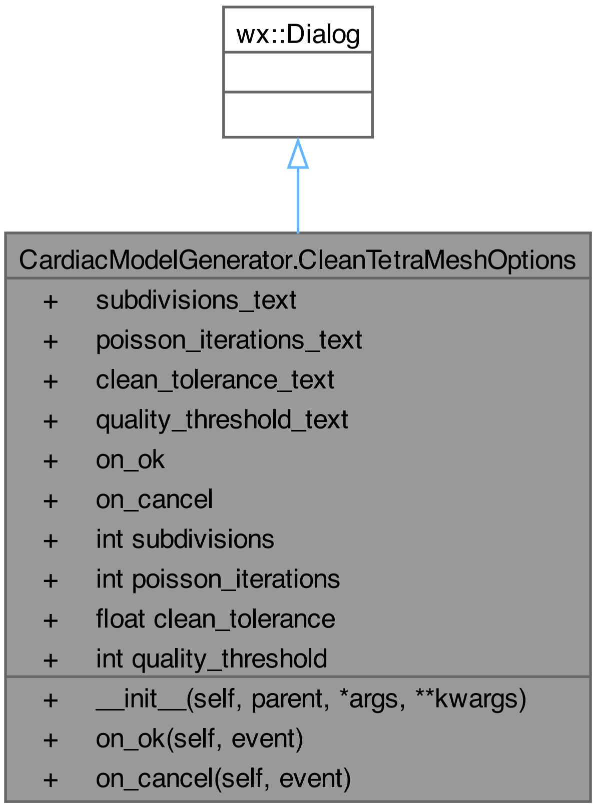
```
CardiacModelGenerator.CardiacMeshalyzer.sizer = wx.BoxSizer(wx.VERTICAL)
```

The documentation for this class was generated from the following file:

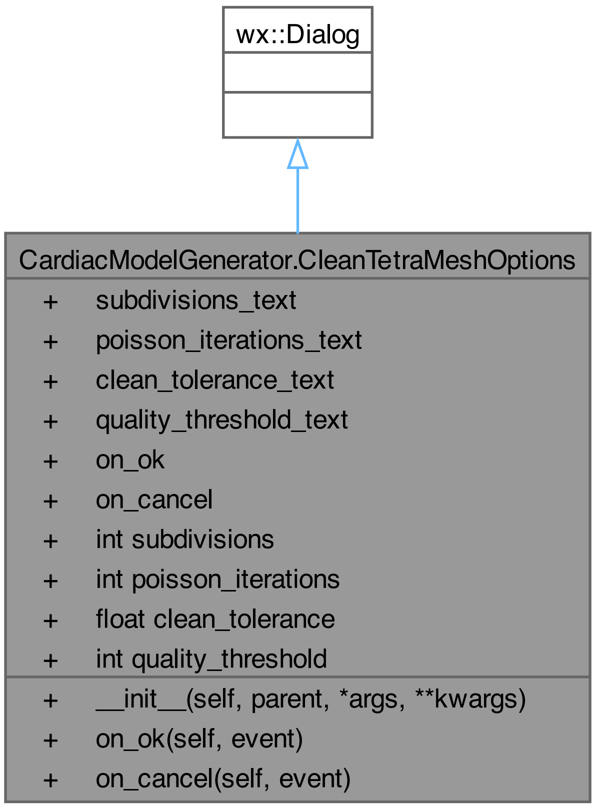
- CardiacModelGenerator.py

6.2 CardiacModelGenerator.CleanTetraMeshOptions Class Reference

Inheritance diagram for CardiacModelGenerator.CleanTetraMeshOptions:



Collaboration diagram for CardiacModelGenerator.CleanTetraMeshOptions:



Public Member Functions

- __init__ (self, parent, ∗args, ∗∗kwargs)
- on_ok (self, event)
- on_cancel (self, event)

Public Attributes

- subdivisions_text = wx.TextCtrl(self, value="2")
- poisson_iterations_text = wx.TextCtrl(self, value="10")
- clean_tolerance_text = wx.TextCtrl(self, value="0.001")
- quality_threshold_text = wx.TextCtrl(self, value="1e-5")
- on_ok
- on_cancel
- int subdivisions = 2
- int poisson_iterations = 10
- float clean_tolerance = 0.001
- int quality_threshold = 1e-5

6.2.1 Detailed Description

```
@class CleanTetraMeshOptions
@brief Dialog for configuring tetrahedral mesh cleaning options.
@details Provides controls for setting parameters such as subdivisions, Poisson iterations, cleaning tolerance
         and quality threshold for cleaning a tetrahedral mesh.

@uml
@startuml
class CleanTetraMeshOptions {
    - subdivisions_text : wx.TextCtrl
    - poisson_iterations_text : wx.TextCtrl
    - clean_tolerance_text : wx.TextCtrl
    - quality_threshold_text : wx.TextCtrl
    - subdivisions : int
    - poisson_iterations : int
    - clean_tolerance : float
    - quality_threshold : float

    + __init__(parent : wx.Window, *args, **kwargs)
    + on_ok(event : wx.Event)
    + on_cancel(event : wx.Event)
}

CleanTetraMeshOptions *-- wx.Dialog : inherits
CleanTetraMeshOptions o-- wx.TextCtrl : "User input fields"
CleanTetraMeshOptions o-- wx.Button : "OK and Cancel buttons"
CleanTetraMeshOptions --> MeshCleaning : "Configures parameters for mesh cleaning"

' Notes for context
note top of CleanTetraMeshOptions
    CleanTetraMeshOptions allows users to define parameters for
    cleaning a tetrahedral mesh. It ensures proper numerical inputs
    and saves these configurations for further processing.
end note
@enduml
```

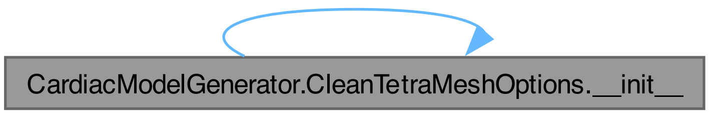
6.2.2 Constructor & Destructor Documentation

6.2.2.1 __init__()

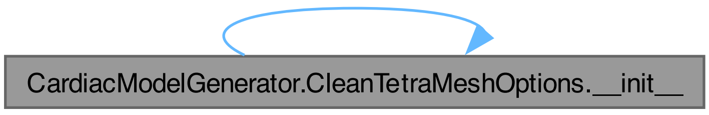
```
CardiacModelGenerator.CleanTetraMeshOptions.__init__ (
             self,
             parent,
            * args,
            ** kwargs)
```

```
@brief Initializes the CleanTetraMeshOptions dialog.
@param parent The parent window that contains this dialog.
@param *args Additional positional arguments for wx.Dialog.
@param **kwargs Additional keyword arguments for wx.Dialog.
@details Creates a dialog with input fields for setting parameters related to tetrahedral mesh cleaning.
         These parameters include:
         - Subdivisions.
         - Poisson Iterations.
         - Cleaning Tolerance.
         - Quality Threshold.
         The dialog also provides OK and Cancel buttons for user interaction.
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.3 Member Function Documentation

6.2.3.1 on_cancel()

```
CardiacModelGenerator.CleanTetraMeshOptions.on_cancel (
              self,
              event)
```

```
@brief Closes the dialog without saving user inputs.
@param event The wxPython event triggering this action.
@details Simply closes the dialog and returns a cancellation state without processing or saving any user input
```

6.2.3.2 on_ok()

```
CardiacModelGenerator.CleanTetraMeshOptions.on_ok (
              self,
              event)
```

```
@brief Saves the user inputs and closes the dialog.
@param event The wxPython event triggering this action.
@details Reads and validates user inputs from the text fields for subdivisions, Poisson iterations,
         cleaning tolerance, and quality threshold. If the inputs are valid, they are saved as instance
         variables and the dialog is closed with a success state.
@throws ValueError If any input is not a valid numerical value.
```

6.2.4 Member Data Documentation

6.2.4.1 clean_tolerance

```
float CardiacModelGenerator.CleanTetraMeshOptions.clean_tolerance = 0.001
```

6.2.4.2 clean_tolerance_text

```
CardiacModelGenerator.CleanTetraMeshOptions.clean_tolerance_text = wx.TextCtrl(self, value="0.↩
001")
```

6.2.4.3 on_cancel

```
CardiacModelGenerator.CleanTetraMeshOptions.on_cancel
```

6.2.4.4 on_ok

```
CardiacModelGenerator.CleanTetraMeshOptions.on_ok
```

6.2.4.5 poisson_iterations

```
int CardiacModelGenerator.CleanTetraMeshOptions.poisson_iterations = 10
```

6.2.4.6 poisson_iterations_text

```
CardiacModelGenerator.CleanTetraMeshOptions.poisson_iterations_text = wx.TextCtrl(self, value="10")
```

6.2.4.7 quality_threshold

```
int CardiacModelGenerator.CleanTetraMeshOptions.quality_threshold = 1e-5
```

6.2.4.8 quality_threshold_text

```
CardiacModelGenerator.CleanTetraMeshOptions.quality_threshold_text = wx.TextCtrl(self, value="1e-5")
```

6.2.4.9 subdivisions

```
int CardiacModelGenerator.CleanTetraMeshOptions.subdivisions = 2
```

6.2.4.10 subdivisions_text

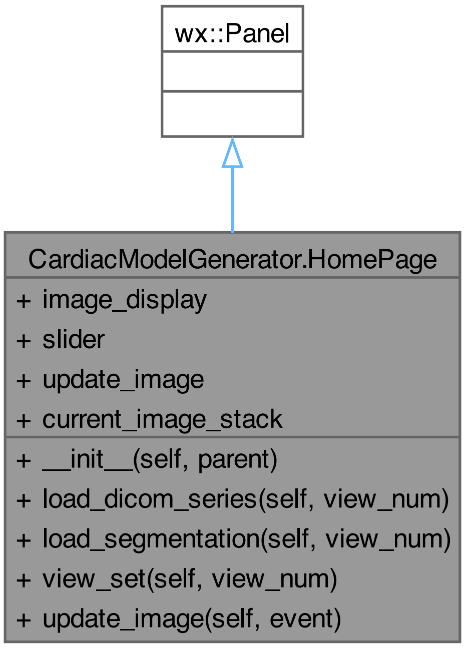
```
CardiacModelGenerator.CleanTetraMeshOptions.subdivisions_text = wx.TextCtrl(self, value="2")
```

The documentation for this class was generated from the following file:

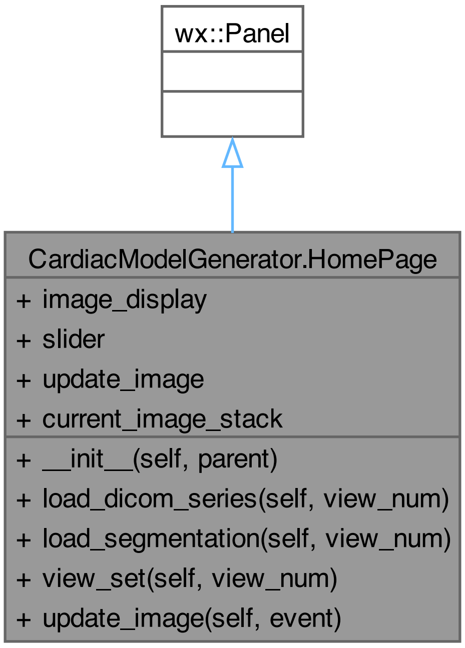
- CardiacModelGenerator.py

6.3 CardiacModelGenerator.HomePage Class Reference

Inheritance diagram for CardiacModelGenerator.HomePage:


wx::Panel
CardiacModelGenerator.HomePage
+ image_display
+ slider
+ update_image
+ current_image_stack
+ __init__(self, parent)
+ load_dicom_series(self, view_num)
+ load_segmentation(self, view_num)
+ view_set(self, view_num)
+ update_image(self, event)

Collaboration diagram for CardiacModelGenerator.HomePage:


wx::Panel
CardiacModelGenerator.HomePage
+ image_display
+ slider
+ update_image
+ current_image_stack
+ __init__(self, parent)
+ load_dicom_series(self, view_num)
+ load_segmentation(self, view_num)
+ view_set(self, view_num)
+ update_image(self, event)

Public Member Functions

- __init__ (self, parent)
- load_dicom_series (self, view_num)
- load_segmentation (self, view_num)
- view_set (self, view_num)
- update_image (self, event)

Public Attributes

- image_display = wx.StaticBitmap(right_panel)
- slider = wx.Slider(right_panel, minValue=0, maxValue=1, value=0, style=wx.SL_HORIZONTAL)
- update_image
- current_image_stack = None

6.3.1 Detailed Description

```
@class HomePage
@brief Main page for interacting with DICOM images and segmentations.
@details This class provides functionality to load and display DICOM images and segmentations,
         manage image viewing with sliders, and bind buttons for different views and actions.

@uml
@startuml
class HomePage {
    - current_image_stack : np.ndarray

    + __init__(parent : wx.Window)
    + load_dicom_series(view_num : int)
    + load_segmentation(view_num : int)
    + view_set(view_num : int)
    + update_image(event : wx.Event)
}

HomePage *-- wx.Panel : inherits
HomePage o-- wx.Button : "Handles action buttons"
HomePage o-- wx.Slider : "Controls image stack navigation"
HomePage o-- wx.StaticBitmap : "Displays images"
HomePage --> DICOM : "Interacts with DICOM data"
HomePage --> Segmentation : "Handles segmentation data"

' Notes for additional context
note top of HomePage
    The HomePage class allows users to load, view, and interact
    with DICOM images and corresponding segmentation masks.
    It provides tools for navigating through image stacks and
    visualizing segmented regions overlaid on images.
end note
@enduml
```

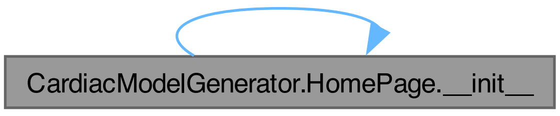
6.3.2 Constructor & Destructor Documentation

6.3.2.1 __init__()

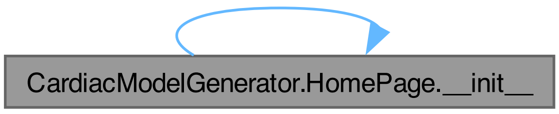
```
CardiacModelGenerator.HomePage.__init__ (
              self,
              parent)
```

```
@brief Initializes the HomePage class and its UI components.
@param parent The parent wx.Window to which this HomePage belongs.
@details This method sets up the main layout of the HomePage, including:
         - A dark-themed background.
         - Buttons for loading image views and segmentations.
         - Buttons for viewing specific segmentations.
         - An image display area and a slider for navigating through image stacks.
         The layout is divided into a left panel for action buttons and a right panel for viewing and interact
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.3.3 Member Function Documentation

6.3.3.1 load_dicom_series()

```
CardiacModelGenerator.HomePage.load_dicom_series (
              self,
              view_num)
```

```
@brief Loads a DICOM series for a specified view.
@param view_num The view number (1, 2, or 3) to associate with the loaded DICOM series.
@details Opens a directory dialog to select a folder containing the DICOM series. The series is processed to e
         image volume, coordinates, and individual image objects. These are stored in the parent window's 'dic
```

6.3.3.2 load_segmentation()

```
CardiacModelGenerator.HomePage.load_segmentation (
              self,
              view_num)
```

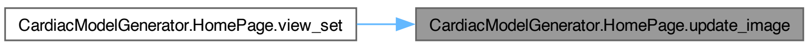
```
@brief Loads a segmentation mask for a specified view.
@param view_num The view number (1, 2, or 3) to associate with the loaded segmentation mask.
@details Opens a file dialog to select a segmentation file. The mask is processed and stored in the parent wir
         'segmentation_data' dictionary.
```

6.3.3.3 update_image()

```
CardiacModelGenerator.HomePage.update_image (
              self,
              event)
```

```
@brief Updates the displayed image based on the slider value.
@param event The wxPython event triggering this action.
@details Extracts the selected slice from the image stack, resizes it to maintain aspect ratio, and updates th
         in the StaticBitmap widget. The image is optionally rotated for better visualization.
```

Here is the caller graph for this function:

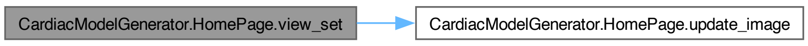

CardiacModelGenerator.HomePage.view_set
CardiacModelGenerator.HomePage.update_image

6.3.3.4 view_set()

```
CardiacModelGenerator.HomePage.view_set (
              self,
              view_num)
```

```
@brief Displays a DICOM image stack with its corresponding segmentation overlay for a specified view.
@param view_num The view number (1, 2, or 3) to display.
@details Checks if both DICOM images and segmentation masks are loaded for the specified view. If they are, th
         segmentation is overlaid on the images, and the stack is displayed. The slider is configured to navig
         the slices. Displays an error message if the data is missing or invalid.
```

Here is the call graph for this function:


CardiacModelGenerator.HomePage.view_set
CardiacModelGenerator.HomePage.update_image

6.3.4 Member Data Documentation

6.3.4.1 current_image_stack

```
CardiacModelGenerator.HomePage.current_image_stack = None
```

6.3.4.2 image_display

```
CardiacModelGenerator.HomePage.image_display = wx.StaticBitmap(right_panel)
```

6.3.4.3 slider

```
CardiacModelGenerator.HomePage.slider = wx.Slider(right_panel, minValue=0, maxValue=1, value=0,
style=wx.SL_HORIZONTAL)
```

6.3.4.4 update_image

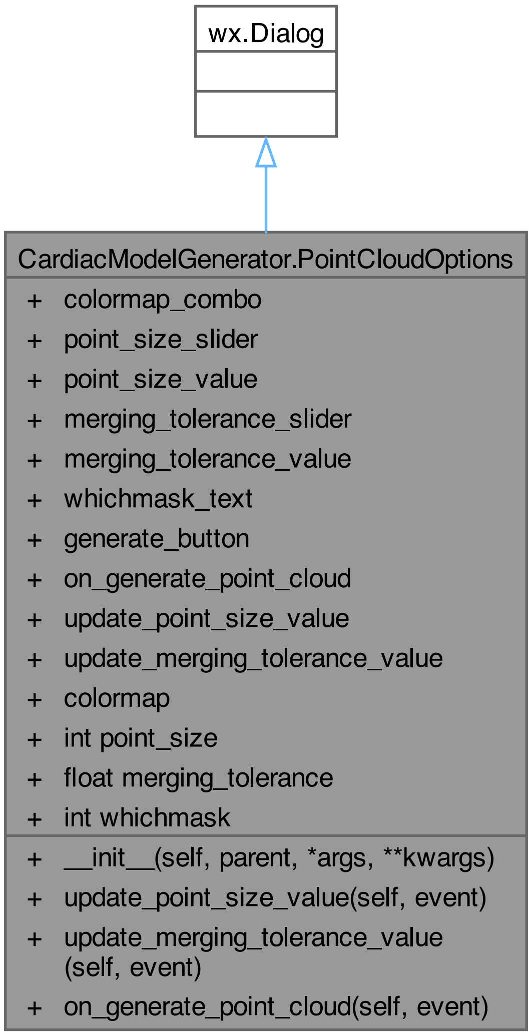
```
CardiacModelGenerator.HomePage.update_image
```

The documentation for this class was generated from the following file:

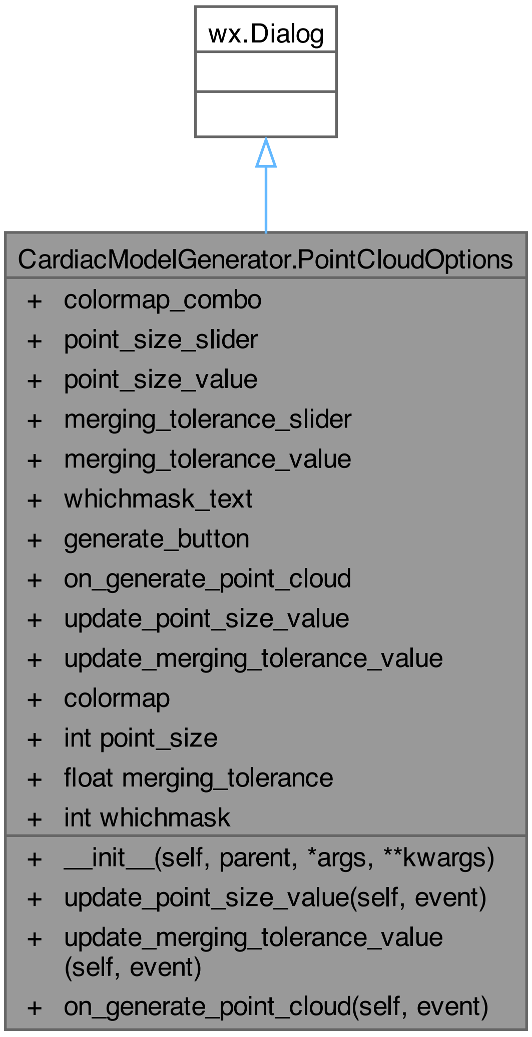
- CardiacModelGenerator.py

6.4 CardiacModelGenerator.PointCloudOptions Class Reference

Inheritance diagram for CardiacModelGenerator.PointCloudOptions:


wx.Dialog
CardiacModelGenerator.PointCloudOptions
+ colormap_combo
+ point_size_slider
+ point_size_value
+ merging_tolerance_slider
+ merging_tolerance_value
+ whichmask_text
+ generate_button
+ on_generate_point_cloud
+ update_point_size_value
+ update_merging_tolerance_value
+ colormap
+ int point_size
+ float merging_tolerance
+ int whichmask
+ __init__(self, parent, *args, **kwargs)
+ update_point_size_value(self, event)
+ update_merging_tolerance_value (self, event)
+ on_generate_point_cloud(self, event)

Collaboration diagram for CardiacModelGenerator.PointCloudOptions:


wx.Dialog
CardiacModelGenerator.PointCloudOptions
+ colormap_combo
+ point_size_slider
+ point_size_value
+ merging_tolerance_slider
+ merging_tolerance_value
+ whichmask_text
+ generate_button
+ on_generate_point_cloud
+ update_point_size_value
+ update_merging_tolerance_value
+ colormap
+ int point_size
+ float merging_tolerance
+ int whichmask
+ __init__(self, parent, *args, **kwargs)
+ update_point_size_value(self, event)
+ update_merging_tolerance_value (self, event)
+ on_generate_point_cloud(self, event)

Public Member Functions

- __init__ (self, parent, ∗args, ∗∗kwargs)
- update_point_size_value (self, event)
- update_merging_tolerance_value (self, event)
- on_generate_point_cloud (self, event)

Public Attributes

- colormap_combo = wx.ComboBox(self, choices=colormap_choices, style=wx.CB_READONLY)
- point_size_slider = wx.Slider(self, minValue=3, maxValue=20, value=3, style=wx.SL_HORIZONTAL)
- point_size_value = wx.StaticText(self, label=str(self.point_size_slider.GetValue()))
- merging_tolerance_slider

- merging_tolerance_value = wx.StaticText(self, label=f"{self.merging_tolerance_slider.GetValue() / 100:.2f}")
- whichmask_text = wx.TextCtrl(self, value="1")
- generate_button = wx.Button(self, label="Generate Point Cloud")
- on_generate_point_cloud
- update_point_size_value
- update_merging_tolerance_value
- colormap = None
- int point_size = 3
- float merging_tolerance = 0.0
- int whichmask = 1

6.4.1 Detailed Description

```
@class PointCloudOptions
@brief Dialog for configuring point cloud generation options.
@details Provides controls for selecting a colormap, adjusting point size, setting merging tolerance,
         and specifying the mask to use for generating the point cloud.

@uml
@startuml
class PointCloudOptions {
    - colormap_combo : wx.ComboBox
    - point_size_slider : wx.Slider
    - point_size_value : wx.StaticText
    - merging_tolerance_slider : wx.Slider
    - merging_tolerance_value : wx.StaticText
    - whichmask_text : wx.TextCtrl
    - generate_button : wx.Button
    - colormap : str
    - point_size : int
    - merging_tolerance : float
    - whichmask : int

    + __init__(parent : wx.Window, *args, **kwargs)
    + update_point_size_value(event : wx.Event)
    + update_merging_tolerance_value(event : wx.Event)
    + on_generate_point_cloud(event : wx.Event)
}

PointCloudOptions *-- wx.Dialog : inherits
PointCloudOptions o-- wx.ComboBox : "Colormap selection"
PointCloudOptions o-- wx.Slider : "Adjust point size and merging tolerance"
PointCloudOptions o-- wx.TextCtrl : "Mask selection"
PointCloudOptions o-- wx.Button : "Generate point cloud"
PointCloudOptions --> Colormap : "Uses matplotlib colormap options"
PointCloudOptions --> PointCloud : "Generates point cloud with configured options"

' Notes for context
note top of PointCloudOptions
    PointCloudOptions allows the user to configure parameters for generating a point cloud.
    It includes widgets for selecting colormap, adjusting point size and merging tolerance,
    and specifying the segmentation mask to use.
end note
@enduml
```

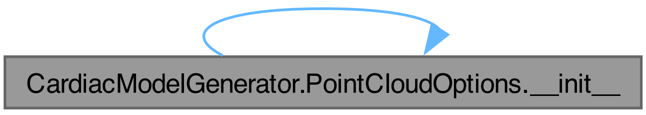
6.4.2 Constructor & Destructor Documentation

6.4.2.1 __init__()

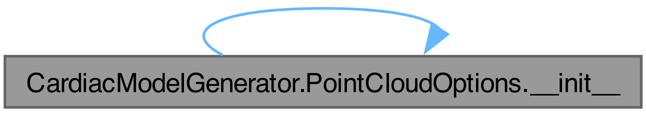
```
CardiacModelGenerator.PointCloudOptions.__init__ (
              self,
              parent,
             * args,
             ** kwargs)
```

```
@brief Initializes the PointCloudOptions dialog.
@param parent The parent window that contains this dialog.
@param *args Additional positional arguments for the wx.Dialog.
@param **kwargs Additional keyword arguments for the wx.Dialog.
@details Sets up the layout, widgets, and event bindings for configuring point cloud generation options.
         Includes controls for colormap selection, point size adjustment, merging tolerance, and mask selectio
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.3 Member Function Documentation

6.4.3.1 on_generate_point_cloud()

```
CardiacModelGenerator.PointCloudOptions.on_generate_point_cloud (
              self,
              event)
```

```
@brief Retrieves user inputs and closes the dialog.
@param event The wxPython button event triggering this action.
@details Captures user-selected values from the colormap dropdown, point size slider, merging tolerance slider
         and mask text input. Validates the mask input as an integer and closes the dialog with a success stat
@throws ValueError If the mask input is not a valid integer.
```

6.4.3.2 update_merging_tolerance_value()

```
CardiacModelGenerator.PointCloudOptions.update_merging_tolerance_value (
              self,
              event)
```

```
@brief Updates the displayed value for the merging tolerance slider.
@param event The wxPython slider event triggering this action.
@details Converts the slider value to a floating-point representation (0.00 to 5.00) and updates the
         `merging_tolerance_value` label for user feedback.
```

6.4.3.3 update_point_size_value()

```
CardiacModelGenerator.PointCloudOptions.update_point_size_value (
              self,
              event)
```

```
@brief Updates the displayed value for the point size slider.
@param event The wxPython slider event triggering this action.
@details Reflects the current slider value in the ‘point_size_value‘ label to provide immediate feedback to th
```

6.4.4 Member Data Documentation

6.4.4.1 colormap

```
CardiacModelGenerator.PointCloudOptions.colormap = None
```

6.4.4.2 colormap_combo

```
CardiacModelGenerator.PointCloudOptions.colormap_combo = wx.ComboBox(self, choices=colormap_↵
choices, style=wx.CB_READONLY)
```

6.4.4.3 generate_button

```
CardiacModelGenerator.PointCloudOptions.generate_button = wx.Button(self, label="Generate
Point Cloud")
```

6.4.4.4 merging_tolerance

```
float CardiacModelGenerator.PointCloudOptions.merging_tolerance = 0.0
```

6.4.4.5 merging_tolerance_slider

```
CardiacModelGenerator.PointCloudOptions.merging_tolerance_slider
```

Initial value:

```
=  wx.Slider(
            self, minValue=0, maxValue=500, value=0, style=wx.SL_HORIZONTAL
        )
```

6.4.4.6 merging_tolerance_value

```
CardiacModelGenerator.PointCloudOptions.merging_tolerance_value = wx.StaticText(self, label=f"{self.↵
merging_tolerance_slider.GetValue() / 100:.2f}")
```

6.4.4.7 on_generate_point_cloud

```
CardiacModelGenerator.PointCloudOptions.on_generate_point_cloud
```

6.4.4.8 point_size

```
int CardiacModelGenerator.PointCloudOptions.point_size = 3
```

6.4.4.9 point_size_slider

```
CardiacModelGenerator.PointCloudOptions.point_size_slider = wx.Slider(self, minValue=3, max↩
Value=20, value=3, style=wx.SL_HORIZONTAL)
```

6.4.4.10 point_size_value

```
CardiacModelGenerator.PointCloudOptions.point_size_value = wx.StaticText(self, label=str(self.↩
point_size_slider.GetValue()))
```

6.4.4.11 update_merging_tolerance_value

```
CardiacModelGenerator.PointCloudOptions.update_merging_tolerance_value
```

6.4.4.12 update_point_size_value

```
CardiacModelGenerator.PointCloudOptions.update_point_size_value
```

6.4.4.13 whichmask

```
int CardiacModelGenerator.PointCloudOptions.whichmask = 1
```

6.4.4.14 whichmask_text

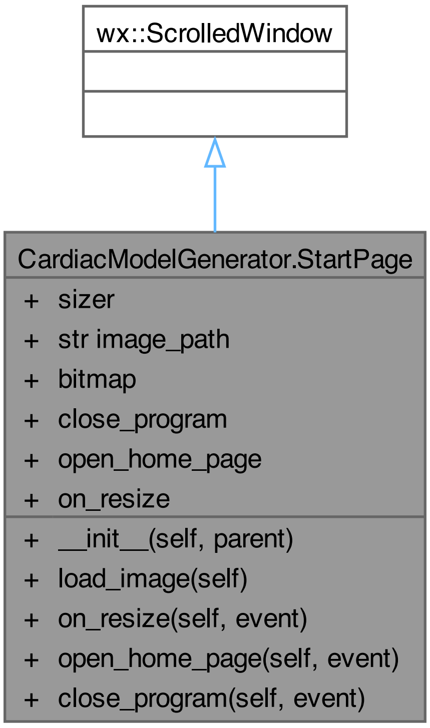
```
CardiacModelGenerator.PointCloudOptions.whichmask_text = wx.TextCtrl(self, value="1")
```

The documentation for this class was generated from the following file:

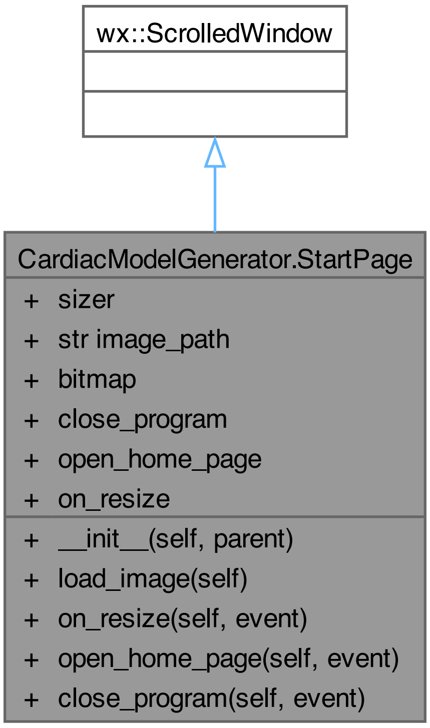
- CardiacModelGenerator.py

6.5 CardiacModelGenerator.StartPage Class Reference

Inheritance diagram for CardiacModelGenerator.StartPage:



Collaboration diagram for CardiacModelGenerator.StartPage:



Public Member Functions

- __init__ (self, parent)
- load_image (self)
- on_resize (self, event)
- open_home_page (self, event)
- close_program (self, event)

Public Attributes

- sizer = wx.BoxSizer(wx.VERTICAL)
- str image_path = "mesh_intro_pic.png"
- bitmap = wx.StaticBitmap(self)
- close_program
- open_home_page
- on_resize

6.5.1 Detailed Description

```
@class StartPage
@brief Introductory page for the Cardiac Meshalyzer application.
@details This class represents the starting page of the GUI, which includes a title, an introductory image,
         a warning message, and navigation buttons for proceeding or exiting the application.

@uml
@startuml
class StartPage {
    - sizer : wx.BoxSizer
    - image_path : str
    - bitmap : wx.StaticBitmap

    + __init__(parent : wx.Window)
    + load_image()
    + on_resize(event : wx.Event)
    + open_home_page(event : wx.Event)
    + close_program(event : wx.Event)
}

' Associations to other elements in the GUI
StartPage *-- wx.ScrolledWindow : inherits
StartPage o-- wx.BoxSizer : "Main vertical sizer"
StartPage o-- wx.StaticBitmap : "Image placeholder"
StartPage --> wx.Button : "Handles Close and Continue buttons"

' Notes for additional context
note top of StartPage
    StartPage serves as the introductory page for the Cardiac Meshalyzer GUI.
    It includes a title, image placeholder, warning text, and navigation buttons.
    The buttons allow users to navigate to the Home Page or close the application.
end note

@enduml
```

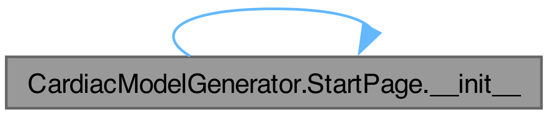
6.5.2 Constructor & Destructor Documentation

6.5.2.1 __init__()

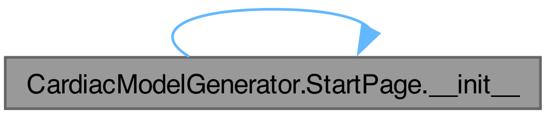
```
CardiacModelGenerator.StartPage.__init__ (
              self,
              parent)
```

```
@brief Initializes the StartPage class and its UI components.
@param parent The parent wx.Window to which this StartPage belongs.
@details This method sets up the introductory page of the Cardiac Meshalyzer application, including:
         - A black background with a title.
         - A placeholder for an introductory image.
         - A warning message about using the application.
         - Two navigation buttons ("Close" and "Continue").
         Enables scrolling and binds events for resize, navigation, and program termination.
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.3 Member Function Documentation

6.5.3.1 close_program()

```
CardiacModelGenerator.StartPage.close_program (
              self,
              event)
```

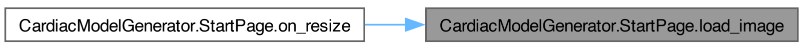
```
@brief Closes the application.
@param event The wxPython event triggering this action.
@details Calls the `Close` method of the top-level window to terminate the application.
```

6.5.3.2 load_image()

```
CardiacModelGenerator.StartPage.load_image (
              self)
```

```
@brief Loads and displays the introductory image.
@details Attempts to load the image from the specified path and scales it to fit the current window size.
         If the image cannot be loaded, a placeholder message is displayed.
@throws Exception If the image file cannot be read or scaled.
```

Here is the caller graph for this function:

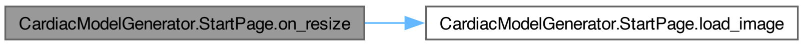

CardiacModelGenerator.StartPage.on_resize
CardiacModelGenerator.StartPage.load_image

6.5.3.3 on_resize()

```
CardiacModelGenerator.StartPage.on_resize (
              self,
              event)
```

```
@brief Handles window resize events.
@param event The wxPython resize event triggering this action.
@details Reloads and rescales the introductory image to fit the updated window size.
```

Here is the call graph for this function:


CardiacModelGenerator.StartPage.on_resize
CardiacModelGenerator.StartPage.load_image

6.5.3.4 open_home_page()

```
CardiacModelGenerator.StartPage.open_home_page (
              self,
              event)
```

```
@brief Navigates to the Home Page of the application.
@param event The wxPython event triggering this action.
@details Calls the `show_page` method of the top-level window to display the "Home Page".
```

6.5.4 Member Data Documentation

6.5.4.1 bitmap

```
CardiacModelGenerator.StartPage.bitmap = wx.StaticBitmap(self)
```

6.5.4.2 close_program

```
CardiacModelGenerator.StartPage.close_program
```

6.5.4.3 image_path

```
str CardiacModelGenerator.StartPage.image_path = "mesh_intro_pic.png"
```

6.5.4.4 on_resize

```
CardiacModelGenerator.StartPage.on_resize
```

6.5.4.5 open_home_page

```
CardiacModelGenerator.StartPage.open_home_page
```

6.5.4.6 sizer

```
CardiacModelGenerator.StartPage.sizer = wx.BoxSizer(wx.VERTICAL)
```

The documentation for this class was generated from the following file:

- CardiacModelGenerator.py

Chapter 7

File Documentation

7.1 CardiacModelGenerator.py File Reference

Classes

- class CardiacModelGenerator.CardiacMeshalyzer
- class CardiacModelGenerator.StartPage
- class CardiacModelGenerator.HomePage
- class CardiacModelGenerator.PointCloudOptions
- class CardiacModelGenerator.CleanTetraMeshOptions

Namespaces

- namespace CardiacModelGenerator

Functions

- CardiacModelGenerator.generate_point_cloud (coords1=None, masks1=None, coords2=None, masks2=None, coords3=None, masks3=None, whichmask=1, tol=0.1, colormap_name="viridis", point_size=5)
- CardiacModelGenerator.generate_tetra_mesh (point_cloud_cleaned)
- CardiacModelGenerator.clean_tetra_mesh (grid, subdivisions=2, poisson_iterations=10, clean_↩ tolerance=0.001, quality_threshold=1e-5)
- CardiacModelGenerator.get_cell_quality (final_volumetric_mesh)

Variables

- CardiacModelGenerator.app = wx.App(False)
- CardiacModelGenerator.frame = CardiacMeshalyzer(None)

Index