Scopic Test Task

Scopic

sales@scopicsoftware.com
+1 (508) 886-3240,
+1 (855) 717-5586 (toll free)
12 Carly Circle,
Rutland, MA, 01543, US
scopicsoftware.com

General Requirements

Place your solution of the task into a single Visual Studio solution. Make it build into command line executable.

Your solution should be a self-contained package that can be opened in Visual Studio, built and run in one click. It should include all of the dependencies.

You are allowed to use 3rd party libraries for code that is not part of your core task description. Such 3rd parties must be included in your resulting package.
For 3d/2d math you may use 3rd party library of your liking, or you may develop your own set of classes. For the tasks you receive, you won't need a full rich set of operations. So, developing own types and operations is a reasonable choice. Choose for yourself.

Your resulting package will be evaluated by (listed not necessarily in order of importance):

1. Package organization (we expect it to be organized close to the way production project would be organized)

2. Usage of safe and modern C++

3. The quality of function and class interfaces. Design decisions you made.

4. The clarity and readability of code

5. Code efficiency (within reasonable limits). We value code clarity over efficiency. We don't expect fully optimized code. But we also expect that there is no unreasonable waste.

6. Application of unit tests when meaningful.

Scopic

sales@scopicsoftware.com
+1 (508) 886-3240,
+1 (855) 717-5586 (toll free)
12 Carly Circle,
Rutland, MA, 01543, US
scopicsoftware.com

Task-Contour Class

1. Develop a class Contour that can store a series of segments. Segment can be a line segment or an arc. Contour class only need to support 2D elements.

2. It's up to you to decide how line and arc segments are represented.

3. Contour should anticipate support more segment types (as a future improvement). The list of all possible/supported types will always be known at compile time.

4. Contour should be copyable and moveable.

5. It should be possible to iterate over Contour segments. To read and modify contour segments.

6. It should be possible to add, insert (in the middle) and remove contour segments.

7. Segments are considered directional. Contour is valid if all of its segments are sequentially connected end-to-begin (within epsilon). Implement isValid() function that returns if contour is valid. This should be calculated on demand and cached, such that consequent calls to isValid() doesn't calculate it again.

8. Implement a routine (could be in a form of unit test) that:

a. Construct vector of 4 hardcoded Contours. Few of them are valid, few invalid.
b. Start asynchronous task to search and return all pointers to valid contour objects.
c. At the same time start another asynchronous search and return all pointers to invalid contour objects.
d. Get results from both jobs.
e. Validate/test that each set of resulting contours is unique and combination of both sets contain all the original contours.

9. Implement utility function that constructs Contour from the series of points that are interpreted as polyline with straight edges. Make this function as flexible/reusable as possible.