Worldwind - Test Plan

Capabilities Tested

Identify the capabilities tested and briefly describe them. Each capability is a small, independently tested requirement and can cover one or more closely-related methods/functions tested. Each capability ideally relies on a single, compact input space model.

#	Capability Name	Description
Sample	fixLeadingBracketSugar()	Given a dotNotation style outputPath like "data[2].&(1,1)", this method fixes the syntactic sugar of "data[2]" --> "data.[2]"
1	makeDefaultDetailLevels()	-
2	getLocations()	-
3	setLocations(LatLon beginLocation, LatLon endLocation)	-
4	getWidths()	-
5	setWidths(double leftWidth, double rightWidth)	-
6	getCornerAzimuths()	-
7	setCornerAzimuths(Angle beginLeftAzimuth, Angle beginRightAzimuth, Angle endLeftAzimuth,Angle endRightAzimuth)	-
8	isEnableCaps()	-
9	setEnableCaps(boolean enableStartCap, boolean enableEndCap)	-
10	getReferencePosition()	-
11	invalidateGeometry()	-

12	computeExtent(Globe globe, double verticalExaggeration)	-
13	computeMinimalGeometry (Globe globe, double verticalExaggeration)	-
14	regenerateSurfaceShape(DrawContext dc, SurfaceShape shape)	-
15	doMoveTo(Globe globe, Position oldRef, Position newRef)	-
16	setPillars(int pillars)	-
17	getStacks()	-
18	setStacks(int stacks)	-
19	getHeightStacks()	-
//********** Geometry Rendering **********//		
20	doRenderGeometry(Draw Context dc, String drawStyle)	Rendering a geometry from a draw context
21	applyCenterLineState(Dra wContext dc)	-
22	drawBox(DrawContext dc, double[] altitudes, boolean[] terrainConformant, int lengthSegments, int widthSegments)	-
23	drawBoxOutline(DrawCont ext dc, double[] altitudes, boolean[] terrainConformant, int lengthSegments, int widthSegments)	-
24	drawBoxCenterLine(Draw Context dc, double[]	-

	altitudes, boolean[] terrainConformant, int lengthSegments, int widthSegments)	
25	getBoxGeometry(DrawContext dc, double[] altitudes, boolean[] terrainConformant, int lengthSegments, int widthSegments)	-
26	makeBoxGeometry(DrawContext dc, double[] altitudes, boolean[] terrainConformant, int lengthSegments, int widthSegments, BoxGeometry geom)	-
27	makeSideGeometry(Terrain terrain, double[] altitudes, boolean[] terrainConformant, int lengthSegments, int widthSegments, BoxGeometry geom)	Creating sides for a geometry from terrain and geo information
28	makeCapGeometry(Terrain terrain, double[] altitudes, boolean[] terrainConformant, int lengthSegments, int widthSegments, BoxGeometry geom)	-
29	makeSideLocations(Globe globe, int lengthSegments, int widthSegments)	-
30	makeCapLocations(Globe globe, int lengthSegments, int widthSegments)	-
31	computeBoxCorners(Globe globe)	-

32	appendLocations(LatLon begin, LatLon middle, LatLon end, int numSegments, List<LatLon> result)	Appending locations from the current location of the box
33	doGetRestorableState(RestorableSupport rs, RestorableSupport.StateObject context)	-
34	doRestoreState(RestorableSupport rs, RestorableSupport.StateObject context)	-

Attributes

List attributes for tested capabilities. Give the source of the attribute: where did it originate from and how did you identify it (e.g. parameter of function/method, state variable, pre-condition, etc.)?

Attribute Name	**Definition and Source**	**Related Capabilities**
(Sample) firstInputString	Representing an input String for String manipulation	1 - 3, 6 - 11
drawStyle	Representing an input draw style ENUM of Airspace e.g. { DRAW_STYLE_FILL, DRAW_STYLE_OUTLINE }	20
drawContextInfo	Representing an object input relating to drawing information e.g. a list of all the sectors rendered so far this frame or the current GL information	14, 20 - 26
terrainInfo	Representing an object input for terrain information	27 - 28
geoAltitudes	An input representing an altitude of the geometry that is going to be drawn	22 - 28

terrainConformant	An input representing if the terrain is conformant	22 - 28
lengthSegments	An input representing the length of the box	22 - 30
widthSegments	An input representing the length of the box	22 - 30
boxGeometry	An input representing an information about the geometry of the box	25 - 28
beginLatLon	An input representing the beginning of latitude and longitude of the box	32
middleLatLon	An input representing the middle point of latitude and longitude of the box	32
endLatLon	An input representing the ending of latitude and longitude of the box	32
numSegment	An input representing the number of segments that will be divided of the box	32

Input Space Models

Create one or more input space models based on the attributes. Define characteristics. Give the source of the characteristic: where did it originate from and how did you identify it (e.g. directly based on an attribute, output property, post-condition, input validity, etc.)?

Input Space Model 1

Q#	**Characteristic Name (Chr)**	**Definition (Def)**	**Source or Participating Attributes (Atr)**	**Related Capabilities (RC)**
	Partition (Prt)		**Notes & Constraints (N&C)**	
Q1	**Chr:** isNullDrawStyle	**Def:** Whether the draw context received is null or not	**Atr:** drawStyle	**RC**: 20
	Prt: T, F		**N&C:** no constraints	

Q2	**Chr**: validDrawStyle	**Def**: The possible draw style of the box	**Atr**: drawContextInfo	**RC:** 20
	Ptr: DRAW_STYLE_OUTLINE, DRAW_STYLE_FILL, DRAW_STYLE_INVALID		**N&C:** Muss pass Q1	
Q3	**Chr**: isNullDrawContext	**Def**: Whether the draw context received is null or not	**Atr**: drawContextInfo	**RC:** 14, 20 - 26
	Prt: T, F		**N&C**: no constraints	
Q4	**Chr**: widthSegmentValue	**Def**: The possible value of width of the box	**Atr**: widthSegments	**RC:** 22 - 30
	Ptr: widthSegmentValue > 0, widthSegmentValue == 0, widthSegmentValue < 0, widthSegmentValue > Integer.MAX_VALUE/2, widthSegmentValue < Integer.MIN_VALUE/2		**N&C:** no constraints	
Q5	**Chr**: isNullTerrain	**Def**: Whether the terrain object received is null or not	**Atr**: terrainInfo	**RC:** 27 - 28
	Prt: T, F		**N&C**: no constraints	
Q6	**Chr**: altitudeLength	**Def**: The altitude is valid only when altitudeLength >= 1	**Atr**: altitude	**RC:** 14, 20 - 26
	Ptr: altitudeLength < 1, altitudeLength > 1, altitudeLength == 1		**N&C:** no constraints	
Q7	**Chr**: altitudeValue	**Def**: The possible value of an altitude	**Atr**: altitude	**RC:** 14, 20 - 26
	Prt: altitudeValue > 0, altitudeValue == 0, altitudeValue < 0, altitudeValue > Double.MAX_VALUE, altitudeValue < Double.MIN_VALUE		**N&C**: Must pass Q6 before checking this	
Q8	**Chr**: terrainConformantLength	**Def**: The terrain conformant array is valid only when terrainConformantLength >= 1	**Atr**: terrainConformant	**RC:** 22 - 28
	Ptr: terrainConformantLength < 1, terrainConformantLength > 1, terrainConformantLength == 1		**N&C:** no constraints	
Q9	**Chr**:	**Def**: The possible value of length of	**Atr**:	**RC:** 22 - 30

	lengthSegmentValue	the box	lengthSegments	
	Prt: lengthSegmentValue > 0, lengthSegmentValue == 0, lengthSegmentValue < 0, lengthSegmentValue > Integer.MAX_VALUE, lengthSegmentValue < Integer.MIN_VALUE		**N&C**: no constraints	
Q10	**Chr**: widthSegmentValue	**Def**: The possible value of width of the box	**Atr**: widthSegments	**RC:** 22 - 30
	Ptr: widthSegmentValue > 0, widthSegmentValue == 0, widthSegmentValue < 0, widthSegmentValue > Integer.MAX_VALUE/2, widthSegmentValue < Integer.MIN_VALUE/2		**N&C:** no constraints	

Test Design Strategy

List steps of your test design strategy here. Remember behavior-first, then structure if applicable.
Step A. *Do Render Validity: All Combination (AC) for capabilities doRenderGeometry because most of the inputs constrain will be true or false which is possible to apply All Combination.*
Step B. *Possible sides geometry: All Pairs (AP) for capability makeSideGeometry. There are 6 characteristics for this capability, which would make 240 total test cases for All Combination. All pairs testing prevents combinatorial explosion by covering more than one interaction at a time.*
Step C. *…..*

Step A - Test Case Specs

All Combinations (AC): πq\|Bq\| -> 24 Cases - 12 constrain cases = 12 cases
Blocks:
***isNullDrawStyle**: {T, F}*
***validDrawStyle**: {DRAW_STYLE_OUTLINE, DRAW_STYLE_FILL, DRAW_STYLE_INVALID}*
***isNullDrawContext**: {T, F}*
***isValidDrawContext**: {T, F}*

Case Id	isNullDrawStyle	validDrawStyle	isNullDrawContext	isValidDrawContext
A1	*T*	*-*	*T*	*-*
A2	*T*	*-*	*F*	*T*

A3	*T*	*-*	*F*	*F*
A4	*F*	*DRAW_STYLE_OUTLINE*	*T*	*-*
A5	*F*	*DRAW_STYLE_OUTLINE*	*F*	*T*
A6	*F*	*DRAW_STYLE_OUTLINE*	*F*	*F*
A7	*F*	*DRAW_STYLE_FILL*	*T*	*-*
A8	*F*	*DRAW_STYLE_FILL*	*F*	*T*
A9	*F*	*DRAW_STYLE_FILL*	*F*	*F*
A10	*F*	*DRAW_STYLE_INVALID*	*T*	*-*
A11	*F*	*DRAW_STYLE_INVALID*	*F*	*T*
A12	*F*	*DRAW_STYLE_INVALID*	*F*	*F*

Step B - Test Case Specs

All pairs: Maxq|Bq| x Maxq,2|Bq| -> minimum 25 Cases
Blocks:
***isNullTerrain**: {T, F}*
***altitudeLength**: {altitudeLength < 1, altitudeLength > 1, altitudeLength == 1}*
***altitudeValue**: {altitudeValue > 0, altitudeValue == 0, altitudeValue < 0, altitudeValue >*

Double.MAX_VALUE, altitudeValue < Double.MIN_VALUE}
terrainConformantLength*: {terrainConformantLength < 1, terrainConformantLength > 1, terrainConformantLength == 1}*
lengthSegmentValue*: {lengthSegmentValue > 0, lengthSegmentValue == 0, lengthSegmentValue < 0, lengthSegmentValue > Integer.MAX_VALUE, lengthSegmentValue < Integer.MIN_VALUE}*
widthSegmentValue*: {*widthSegmentValue > *0,* widthSegmentValue == *0,* widthSegmentValue < *0,* widthSegmentValue > *Integer.MAX_VALUE/2,* widthSegmentValue < *Integer.MIN_VALUE/2}*

Case Id	isNullTerrain	altitudeLength	altitudeValue	terrainConformantLength	lengthSegmentValue	widthSegmentValue
B1	*T*	-	*altitudeValue > 0,*	*terrainConformantLength > 1*	lengthSegmentValue > 0	widthSegmentValue > 0
B2	*F*	*altitudeLength < 1*	*altitudeValue > 0,*	*terrainConformantLength > 1*	lengthSegmentValue > 0	widthSegmentValue > 0
B3	*F*	*altitudeLength == 1*	*altitudeValue > 0,*	*terrainConformantLength > 1*	lengthSegmentValue > 0	widthSegmentValue > 0
B4	*F*	*altitudeLength > 1*	*altitudeValue > 0,*	*terrainConformantLength > 1*	lengthSegmentValue > 0	widthSegmentValue > 0
B5	*F*	*altitudeLength > 1*	*altitudeValue == 0*	*terrainConformantLength > 1*	lengthSegmentValue > 0	widthSegmentValue > 0
B6	*F*	*altitudeLength > 1*	*altitudeValue < 0*	*terrainConformantLength > 1*	lengthSegmentValue > 0	widthSegmentValue > 0
B7	*F*	*altitudeLength > 1*	*altitudeValue > Double.MAX_*	*terrainConformantLengt*	lengthSegmentValue > 0	widthSegmentValue > 0

			VALUE	*h > 1*		
B8	*F*	*altitudeLength > 1*	*altitudeValue > Double.MIN_VALUE*	*terrainConformantLength > 1*	lengthSegmentValue > 0	widthSegmentValue > 0
B9	*F*	*altitudeLength == 1*	*altitudeValue > 0*	*terrainConformantLength > 1*	*lengthSegmentValue == 0*	widthSegmentValue > 0
B10	*F*	*altitudeLength == 1*	*altitudeValue >*	*terrainConformantLength > 1*	*lengthSegmentValue < 0*	widthSegmentValue > 0
B11	*F*	*altitudeLength == 1*	*altitudeValue > 0*	*terrainConformantLength > 1*	*lengthSegmentValue > Integer.MAX_VALUE*	widthSegmentValue > 0
B12	*F*	*altitudeLength == 1*	*altitudeValue > 0*	*terrainConformantLength > 1*	*lengthSegmentValue < Integer.MIN_VALUE*	widthSegmentValue > 0
B13	*F*	*altitudeLength == 1*	*altitudeValue > 0*	*terrainConformantLength > 1*	engthSegmentValue > 0	widthSegmentValue == *0*
B14	*F*	*altitudeLength == 1*	*altitudeValue > 0*	*terrainConformantLength > 1*	engthSegmentValue > 0	widthSegmentValue < *0*
B15	*F*	*altitudeLength == 1*	*altitudeValue > 0*	*terrainConformantLength > 1*	engthSegmentValue > 0	widthSegmentValue > *Integer.MAX_VALUE/2*
B16	*F*	*altitudeLength == 1*	*altitudeValue > 0*	*terrainConformantLength > 1*	engthSegmentValue > 0	widthSegmentValue < *Integer.MIN_VALUE/2*
B17	*F*	*altitudeLength*	*altitudeValue*	*terrainC*	lengthSegmentV	widthSegme

		> 1	*> 0*	*onformantLength < 1*	alue > 0	ntValue > 0
B19	*F*	*altitudeLength > 1*	*altitudeValue > 0*	*terrainConformantLength == 1*	lengthSegmentValue > 0	widthSegmentValue > 0
B20	*F*	*altitudeLength > 1*	*altitudeValue > 0*	*terrainConformantLength == 1*	lengthSegmentValue == 0	widthSegmentValue == 0
B21	*F*	*altitudeLength > 1*	*altitudeValue > 0*	*terrainConformantLength == 1*	lengthSegmentValue == 0	widthSegmentValue > 0
B22	*F*	*altitudeLength > 1*	*altitudeValue > 0*	*terrainConformantLength == 1*	lengthSegmentValue > 0	widthSegmentValue == 0
B23	*F*	*altitudeLength > 1*	*altitudeValue > 0*	*terrainConformantLength == 1*	lengthSegmentValue < 0	widthSegmentValue == 0
B24	*F*	*altitudeLength > 1*	*altitudeValue > 0*	*terrainConformantLength == 1*	lengthSegmentValue == 0	widthSegmentValue < 0
B25	*T*	-	*altitudeValue > 0*	*terrainConformantLength == 1*	lengthSegmentValue == 0	widthSegmentValue == 0