SmartPolygonOptimizer
API Reference

Version 1.3

January 29, 2010

3D Incorporated

3D Incorporated LICENSE AGREEMENT

1. Software
As used herein, the term, “Software” means the software accompanying this Agreement, including: (i) the object code form of 3D Incorporated's library of function calls and the ASCII form of 3D Incorporated’s header files for function calls (the “API”); and (ii) the executable form of certain 3D Incorporated's software tools (“Tools”).

2. Evaluation
If you received the Software for the purpose of internal evaluation, as expressed by 3D Incorporated, or if you received the Software without conditions of payment, then the Software is to be used for evaluation purposes only, and this Agreement is effective for a fixed period of time to be determined by 3D Incorporated. If no explicit period of time is given by 3D Incorporated, then this Agreement will terminate in 90 days from receipt of the Evaluation Software, with no written notice of termination required. Upon termination of this agreement, see 9. Term.

3. License Grant
Subject to the terms and conditions of this Agreement, 3D Incorporated grants you a non-exclusive, non-transferable, limited license to: (a) use the copy of the Software and accompanying materials, including a dongle (as applicable to the licensed Software), enclosed in this package (collectively the “Product”) on the Designated System; (b) develop separate software applications derived from the API (the “Applications”); and (c) use, copy, and distribute the Applications; provided, however, that you obtain written approval from 3D Incorporated prior to any sale, license, lease or other distribution of the Applications. You may transfer the Software to the Designated System provided you keep the original Software solely for backup or archival purposes. “Designated Systems” for any Software means a computer system that is: (i) owned or controlled and operated by you; (ii) designated as the computer system on which the Software will be used; and (iii) included a dongle (solely for Software that does not require and unlock code from 3D Incorporated). All rights not expressly granted to you herein are retained by 3D Incorporated. You acknowledge that the Software is copy protected and requires either a key code furnished by 3D Incorporated or an appropriate dongle for continuing operation, as applicable.

4. Software Media and Dongle
You may receive the Product on media which contain various executables or in multiple forms of media. Regardless of the number or types of executables or media you receive, you may use only the media and executables specified in the applicable purchase order or loan agreement. The media may contain executables which have not been licensed; and such unlicensed executables may not be used unless a license is acquired by you from 3D Incorporated. In the event that a dongle that is included as part of the Product you receive with this Agreement is lost or damaged it cannot be replaced by 3D Incorporated, and such loss or damage will require that you purchase another copy of the Software.

5. Ownership
All rights, title and interest to the Product, and any proprietary information contained on the media, or in the associated materials or dongle, are owned by 3D Incorporated and are protected by copyright, trademark and trade secret law and international treaties. You acquire only the right to use the Product during the term of this Agreement. You agree not to develop separate software applications of any kind derived from the Tools or from any other proprietary information of 3D Incorporated, except for the Applications. Any rights, express or implied, in the Product, and any proprietary information contained in the media or dongle other than those specified in this Agreement are reserved by 3D Incorporated. You must treat the Product like any other copyrighted material except as otherwise provided under this Agreement. You agree not to remove, deface or obscure 3D Incorporated’s copyright or trademark notices or legends, or any other proprietary notices in or on the Product or media.

6. Copies and Modifications
You may make one (1) copy of the Software solely for back-up purpose; provided, however, that you reproduce and include all copyright, trademark, and other proprietary rights notices on the copy. You may not make copies of any of the written documentation included in the Product without prior permission, in writing, from 3D Incorporated. You may not nor may you assist another to modify, translate, convert to another programming language, decompile, reverse engineering or disassemble any portions of the Product. Except as otherwise expressly provided by this Agreement, you may not copy the Software. You agree to notify your employees and agents who may have access to the Product of the restrictions contained in this Agreement and to ensure their compliance with such restrictions.

7. Taxes
You shall be liable for and shall pay all charges and taxes, including all sales and use taxes, which may now or hereafter be imposed or levied upon the license or possession or use of the Product, except taxes based on 3D Incorporated’s income.

8. Confidentiality
By accepting this license, you acknowledge that the Product, and any proprietary information contained in the associated media and dongle, are proprietary in nature to 3D Incorporated and contain valuable trade secrets and other proprietary information developed or acquired at

great expense to 3D Incorporated. You agree not to disclose to others or to utilize such trade secrets or proprietary information except as expressly provided herein.

9. Term
This Agreement is effective from the date you use the Software, until the earlier of: (i) the Agreement is terminated; or (ii) if applicable, the dongle is lost or damaged. 3D Incorporated or you may terminate this Agreement at any time by giving thirty (30) days written notice of termination to the other party. Notwithstanding the above, if you fail to comply with any term of this Agreement, or if you become the subject of a voluntary or involuntary petition in bankruptcy or any proceeding relating to insolvency, receivership, liquidation, or composition for the benefit of creditors, if that petition or proceeding is not dismissed with prejudice within thirty (30) days after filling, 3D Incorporated may terminate this Agreement immediately upon notice to you. Promptly upon termination of this Agreement, you agree to cease all use of the Product, and to either destroy or promptly return to 3D Incorporated the Product, together with all copies you made thereof. Notwithstanding the remedies provided above, 3D Incorporated may enforce all of its other legal rights. Sections 4 – 12 and 14 – 18 will survive termination of this Agreement.

10. Assignment
You may not assign, sublicense, rent, loan, lease, convey or otherwise transfer this Agreement, any applicable unlock code, or the Product without prior written permission from 3D Incorporated. Any unauthorized assignment, sublicense, rental, loan, lease, conveyance or other transfer of any copy of the Product or the unlock code shall be void and shall automatically terminate this Agreement.

11. Limited Warranty
3D Incorporated warrants that the Software provided to you shall operate as described in the accompanying documentation under normal use consistent with the terms of this Agreement, for a period of ninety (90) days from the date of your receipt thereof. For the purposes of this Section 10, “Defective Software” means Software which does not operate as described in the accompanying documentation under normal use during the warranty period. 3D Incorporated’s warranty as set forth above shall not be enlarged, diminished or affected by, and no liability shall arise out of, 3D Incorporated’s rendering of technical advice or service in connection with the Product. 3D Incorporated does not warrant that the Software will meet your requirement, operate without interruption or be error free. Your sole remedy under this Section 10 shall be, upon return of the Defective Software to 3D Incorporated, at 3D Incorporeated’s sole discretion: (i) repair or replacement of any Defective Software within the warranty period; or (ii) within the warranty period, return of the amount, if any, paid by you to 3D Incorporated for the Defective Software. Any replacement Software will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer.

12. Warranty Exceptions
Except for the warranty expressly provided in Section 10, the Software is provided “as is”. To the maximum extent permitted by applicable law, 3D Incorporated disclaims all other warranties of any kind, express or implied, including, but not limited to, implied warranties of performance, merchantability, and fitness for a particular purpose. You bear all risk relating to quality and performance of the Software, and assume the entire cost of all necessary servicing, repair or correction.

Some jurisdictions do not allow limitations on implied warranties, so the above limitation may not apply to you. In that event, such warranties are limited to the warranty period. This warranty gives you specific legal rights. You may also have other rights which vary from jurisdiction to jurisdiction.

13. Limitation of Remedies
3D Incorporated’s maximum liability for any claim by you or anyone claiming through or on behalf of you arising out of this Agreement shall not in any event exceed the actual amount paid by you for the license to the Product. To the maximum extent permitted by applicable law, 3D Incorporated shall not be liable for the loss of revenue or profits, expense or inconvenience, or for any other direct, indirect, special, incidental, exemplary or consequential damages, arising out of this Agreement or caused by the use, misuse or inability of use the Product, even if 3D Incorporated has been advised of the possibility of such damages. This limited warranty shall not extend to anyone other than the original user of the Product. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

14. Support
3D Incorporated is not responsible for maintaining or helping you to use the Product, and is not required to make available to you any updates, fixes or support for the Product (an “Upgrade”), except pursuant to a separate written Software Maintenance Agreement, except that if any license is included by 3D Incorporated with the upgrade which contains terms additional to or inconsistent with this Agreement, then such additional or inconsistent terms shall supersede the applicable portions of this Agreement when applied to the Upgrade.

15. Governing Law
This Agreement shall be governed by the laws of Japan, exclusive of its choice of law principles.

16. General provisions
If any provision of this Agreement is held to be void, invalid, unenforceable or illegal, the other provisions shall continue in full force and effect. Failure of a party to enforce any provision of this Agreement shall not constitute or be construed as a waiver of such provision or of the right to enforce such provision. If any legal action, including arbitration, arises under this Agreement or by any reason of any asserted breach of this Agreement, the

prevailing party shall be entitled to recover all costs and expenses, including reasonable attomeys' fees, incurred as a result of such legal action.

17. Export

You agree to comply fully with all laws and regulations of Japan and other countries ("Export Laws") to assure that the Product is not: (i) exported, directly or indirectly, in violation of Export Laws; or (ii) used for any purpose prohibited by Export Laws.

18. Acknowledgment

This Agreement is the complete and exclusive statement of agreement between the parties and supersedes all proposals or prior agreements, verbal or written, and any other communications between the parties relating to the subject matter of this Agreement. No amendment to this Agreement shall be effective unless signed by an officer of 3D Incorporated.

SmartPolygonOptimizer™ API

version 1.3

Copyright

Trademarks

Web Information

English:
http://www.ddd.co.jp/tech_info/eng_tech_smartcollision.htm

Japanese:
http://www.ddd.co.jp/tech_info/tech_smartcollision.htm

Support

mailto:haptics@ddd.co.jp

Corporate Headquarters

3D Incorporated
http://www.ddd.co.jp/
Urban Square Yokohama 2F,
1-1 Sakae-cho, Kanagawa-ku, Yokohama, 221-0052, Japan
tel:+81-45-450-1330, fax:+81-45-450-1331
mailto:haptics@ddd.co.jp

Contents

Figures

1. Preface

This reference manual describes interfaces of SmartPolygonOptimizerAPI.

2. Interface

The SmartPolygonOptimizer API consists of a SPOObject class and a SPOPiece struct.

SPOPiece is a struct which stores an indexed triangle set in two dynamically allocated arrays; one array for the vertex data, and one for the index.

SPOObject is a class which stores the SPOPiece objects that together form one object model, and can be directly used by SmartCollision. SPOObject has methods which diagnose various specific features or properties of the triangle set given to it, and can modify the triangle data to fix flaws such as holes, or to optimize for some specific parameter, without changing the model's essential shape.

2.1 Definition of SPOPieces

The definition of SCObject is as follows.

```
struct SPOPiece{
        SPOdouble*vertices;
        SPOint vertexSize;
        SPOint*triangles;
        SPOint triangleSize;
};
```

Figure 2-1 shows the format of indexed triangle set. Indices of vertices start at 0.

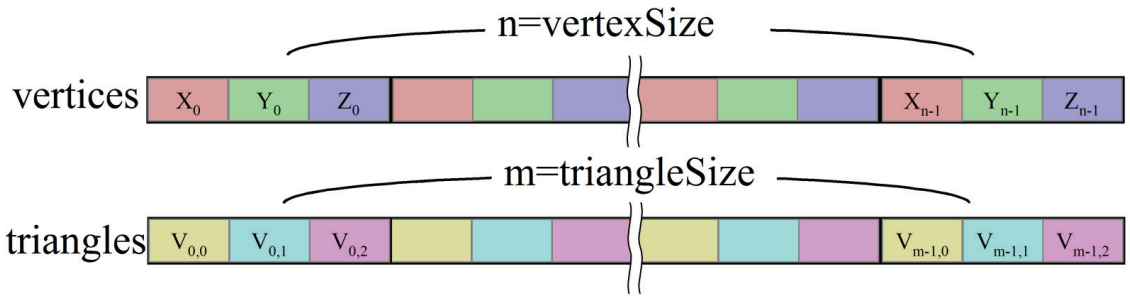

Figure 2-1: Indexed triangle set

2.2 The methods of SPOObject

The methods of SPOObject are as follows.

2.2.1 SPOObject ()

【Syntax】

SPOObject (void);

【Description】

The constructor of SPOObject.

【Arguments】

〈INPUT〉

〈OUTPUT〉

【Return】

2.2.2 ～SPOObject ()

【Syntax】

～SPOObject (void);

【Description】

The distructor of SPOObject.

【Arguments】

【Return】

2.2.3 AddTriangles()

【Syntax】

int AddTriangles(const SPOfloat*vertices, SPOint vertexNum, const SPOint triangles, SPOint triangleNum);

int AddTriangles (const SPOdouble*vertices, SPOint vertexNum, const SPOint triangles, SPOint triangleNum);

【Description】

Adds triangles to the object. Triangles to be added at one time are treated as one piece.

It is possible to call AddTriangles at multiple times.

【Arguments】

〈INPUT〉

vertices	The array of vertices. The array has the 3*vertexNum elements.
vertexNum	The number of vertices.
triangles	The array of index of vertices of triangles. Index starts from 0. The array has the 3*triangleNum elements.
triangleNum	The number of triangles.

〈OUTPUT〉

【Return】

SPO_NO_ERROR: There has been no error.

SC_ERROR_FAILED: Failed to execution.

SC_ERROR_INVALID_DATA: The data specified is invalid.

2.2.4 ChangeTriangulationPattern()

【Syntax】

int RemoveConvexPieces(SPOenum type ,SPOdouble tolerance,SPOint iteration=1);

difference

【Description】

Change triangulation pattern.

【Arguments】

〈INPUT〉

type	Type of triangulation pattern ■ SPO_TRIANGULATION_TYPE_REDUCE_EDGE_LENGTH: triangulation such that total edge lengths are reduced. ■ SPO_TRIANGULATION_TYPE_REDUCE_AREA_DIFFERENCE: triangulation such that area differences are reduced ■ SPO_TRIANGULATION_TYPE_REDUCE_WIDTH_DIFFERENCE: triangulation such that width differences of triangles are reduced
tolerance	The angle between normals of adjacent triangles. If the angle is less equal than tolerance, triangles are treated as being on same plane. The angle is given in degrees.
Iteration	Iteration of process.

〈OUTPUT〉

【Return】

SPO_NO_ERROR: There has been no error.

SPO_ERROR_FAILED: Failed to execution.

2.2.5 ConnectVertices()

【Syntax】

SPOint ConnectVertices(SPOdouble tolerance,SPObool removeVerticesFlag=true);

【Description】

This method connects vertices of which the difference between each coordinate is under a given tolerance. This method also removes degenerated triangles.

【Arguments】

〈INPUT〉

tolerance	The number of vertices.
removeVerticesFlag	If this is true, the connected vertices are removed from the vertex list.

〈OUTPUT〉

【Return】

SPO_NO_ERROR: There has been no error.

SPO_ERROR_FAILED: Failed to execution.

2.2.6 DecomposeIntoSingleBoundaryPieces ()

【Syntax】

SPOint DecomposeIntoSingleBoundaryPieces(void);

【Description】

Decomposes each SPOPiece in SPOObject into single boundary pieces. All triangles in single boundary piece are connected each other.

【Arguments】

〈INPUT〉

Object SPOObject to be modified

〈OUTPUT〉

【Return】

SPO_NO_ERROR: There has been no error.

SPO_ERROR_FAILED: Failed to execution.

2.2.7 GetEdgeCount()

【Syntax】

SPOint GetEdgeCount(SPOenum type,SPOdouble tolerance);

SPOint GetEdgeCount(SPOenum type SPOint index,,SPOdouble tolerance);

【Description】

Gets the count of specified type of edge in SPOObject/ SPOPiece.

【Arguments】

〈INPUT〉

index	Index of piece.
type	Type of edge to be counted. ■ SPO_EDGE_ TYPE_BRANCHED: branched edges. ■ SPO_EDGE_ TYPE_DUPLICATE: duplicate edges. ■ SPO_EDGE_TYPE_UNLINKED: unlinked edges. ■ SPO_EDGE_TYPE_FOLDING: folding edges.
tolerance	The angle between adjacent triangles. The angle is given in degrees.

〈OUTPUT〉

【Return】

the count of specified type of edges

2.2.8 GetPieceCount()

【Syntax】

SPOint GetPieceCount(void);

【Description】

Gets the count of pieces in SPOObject.

【Arguments】

〈INPUT〉

〈OUTPUT〉

【Return】

The count of pieces in SPOObject.

2.2.9 GetPiece ()

【Syntax】

const SPOPiece*GetPiece (SPOint index);

【Description】

Gets the pointer of SPOPiece.

【Arguments】

〈INPUT〉

index	Index of piece to get

〈OUTPUT〉

【Return】

The point of SPOPiece

2.2.10 IsClosedPolyhedr*()

【Syntax】

SPObool IsClosedPolyhedra(void);

SPObool IsClosedPolyhedron(int index);

【Description】

Checks whether the object/piece is a closed polyhedra/polyhedron for SmartCollisionSDK. Closed polyhedra/polyhedron for SmartCollision must be closed and each piece must be single boundary and has no duplicate or branched edges.

【Arguments】

〈INPUT〉

Index Index of piece.

〈OUTPUT〉

【Return】

true: The object consists of only closed polyhedra or the piece is a closed polyhedron.

false: The object consists of not only closed polyhedra or the piece is not a closed polyhedron.

2.2.11 IsClosed()

【Syntax】

SPObool IsClosed(void);

SPObool IsClosed(int index);

【Description】

Checks whether the object/piece is closed or not.

【Arguments】

〈INPUT〉

index Index of piece.

〈OUTPUT〉

【Return】

true: If the object is closed.

false: If the object is not closed.

2.2.12 IsConvex()

【Syntax】

SPObool IsConvex(SPOdouble tolerance=0);

SPObool IsConvex(int index, SPOdouble tolerance=0);

【Description】

Checks whether the object/piece is convex or not.

【Arguments】

〈INPUT〉

index	Index of piece.
tolerance	The angle between normals of adjacent triangles. If the angle is less equal than tolerance, triangles are treated as being on same plane. The angle is given in degrees.

〈OUTPUT〉

【Return】

true: If the object is convex.

false: If the object is not convex.

2.2.13 IsSingleBoundary ()

【Syntax】

SPObool IsSingleBoundary(void);

SPObool IsSingleBoundary(SPOint index);

【Description】

Checks whether the object/pieces consists of single boundary pieces or not.

【Arguments】

〈INPUT〉

index	Index of piece.

〈OUTPUT〉

【Return】

true: If the object consists of single piece.

false: If the object does not consists of single piece.

2.2.14 MergePieces()

【Syntax】

int MergePieces(void);

【Description】

Merge pieces of SPOObject.

【Arguments】

〈INPUT〉

〈OUTPUT〉

【Return】

SPO_NO_ERROR: There has been no error.

SPO_ERROR_FAILED: Failed to execution.

2.2.15 RemoveClosedPieces()

【Syntax】

int RemoveClosedPieces(void);

【Description】

Removes closed pieces.

【Arguments】

〈INPUT〉

〈OUTPUT〉

【Return】

SPO_NO_ERROR: There has been no error.

SPO_ERROR_FAILED: Failed to execution.

2.2.16 RemoveConvexPieces()

【Syntax】

int RemoveConvexPieces(SPOdouble tolerance);

【Description】

Removes convex pieces.

【Arguments】

〈INPUT〉

tolerance	The angle between normals of adjacent triangles. If the angle is less equal than tolerance, triangles are treated as being on same plane. The angle is given in degrees.

〈OUTPUT〉

【Return】

SPO_NO_ERROR: There has been no error.

SPO_ERROR_FAILED: Failed to execution.

2.2.17 RemoveNonconvexPieces()

【Syntax】

int RemoveConvexPieces(SPOdouble tolerance);

【Description】

Removes non-convex pieces.

【Arguments】

〈INPUT〉

tolerance	The angle between normals of adjacent triangles. If the angle is less equal than tolerance, triangles are treated as being on same plane. The angle is given in degrees.

〈OUTPUT〉

【Return】

SPO_NO_ERROR: There has been no error.

SPO_ERROR_FAILED: Failed to execution.

2.2.18 RemoveRedundantVertices()

【Syntax】

int RemoveRedundantVertices (SPOdouble tolerance, SPOint iteration=1,SPObool moveVerticesFlag=true);

【Description】

Remove redundant vertices.

【Arguments】

〈INPUT〉

tolerance	The angle between normals of adjacent triangles. If the angle is less equal than tolerance, triangles are treated as being on same plane. The angle is given in degrees.
iteration	Iteration of process
removeVerticesFlag	If this is true, the redundant vertices are removed from the vertex list. If not, the redundant vertices remain in the vertex list.

〈OUTPUT〉

【Return】

SPO_NO_ERROR: There has been no error.

SPO_ERROR_FAILED: Failed to execution.

2.2.19 RemoveSmallVolumePieces()

【Syntax】

int RemoveSmallVolumePieces(SPOdouble tolerance);

【Description】

Removes small volume pieces, if it is closed.

【Arguments】

〈INPUT〉

tolerance	The tolerance of volume. If the volume of closed piece is less equal than tolerance, it is removed.

〈OUTPUT〉

【Return】

SPO_NO_ERROR: There has been no error.

SPO_ERROR_FAILED: Failed to execution.

2.2.20 RemoveThinTriangles()

【Syntax】

int RemoveThinTriangles(SPOdouble tolerance, SPOint iteration=1, SPObool moveVerticesFlag=false);

【Description】

Remove redundant vertices.

【Arguments】

〈INPUT〉

tolerance	Tolerance of width of triangle.
iteration	Iteration of process
moveVerticesFlag	If this is true, vertieces move on the edges.

〈OUTPUT〉

【Return】

SPO_NO_ERROR: There has been no error.

SPO_ERROR_FAILED: Failed to execution.

2.2.21 RemoveUnclosedPieces()

【Syntax】

int RemoveClosedPieces(void);

【Description】

Removes unclosed pieces.

【Arguments】

〈INPUT〉

object SPOObject to be modified

〈OUTPUT〉

【Return】

SPO_NO_ERROR: There has been no error.

SPO_ERROR_FAILED: Failed to execution.

2.2.22 SplitEdges()

【Syntax】

int SplitEdges(SPOdouble tolerance, SPOint iteration=1, SPObool moveVerticesFlag =false);

【Description】

Splits edges, if there are verteice between the edges.

【Arguments】

〈INPUT〉

tolerance	Tolerance of distance from edge to the vertex. If the distance is less than the tolerance, the vertex is considered as between the edge.
iteration	Iteration of process
moveVerticesFlag	If this is true, vertieces move on the edges.

〈OUTPUT〉

【Return】

SPO_NO_ERROR: There has been no error.

SPO_ERROR_FAILED: Failed to execution.