SmartCollision™ SDK
API Reference

Version 2.3

January 29, 2010

3D Inc.

3D Incorporated

3D Incorporated LICENSE AGREEMENT

3D Incorporated

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.

SmartCollision™ SDK

version 2.3

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

Tables

Lists

1. Preface

This reference manual describe class interface of SmartCollisionSDK.

2. Class interface

SmartCollisionSDK consists of two classes, namely SCSceneManager, SCObject. Class interface of each class is described below.

2.1 The methods of SCSceneManager

The methods of SCSceneManager are as follows.

Table 2-1: Methods of SCSceneManager

Categories	Methods
Constructor	SCSceneManager ()
Destructor	～SCSceneManager ()
Setting/Getting attributes	SetAttribute () GetAttribute ()

Add/Delete object	AddObject()
	DeleteObject()

Grouping	AddObjectToGroup()
	DeleteObjectFromGroup()
	DeleteGroup()
	ResetGroup()

Setting activity of collision detection	SetActivity()

Setting/Getting transformation	SetTransformation()
	GetTransformation()

Updating status (Execute collision detection)	UpdateStatus()
Getting status (Getting results of collision detection)	GetStatus()
Resetting status (Resetting results of collision detection)	ResetStatus()

2.1.1 SCSceneManager ()

【Syntax】

SCSceneManager (SCenum mode);

【Description】

The constructor of SCSceneManager.

【Arguments】

〈INPUT〉

mode	Sets mode of model data. Default value is SC_MODE_TRIANGLE_SOUP. ■ SC_SCENE_MANAGER_TRIANGLE_SOUP: Arbitrary set of triangles or triangle soup. ■ SC_SCENE_MANAGER_CLOSED_POLYHEDRA: Closed polyhedra. This type is more efficient than SC_MODE_TRIANGLE_SOUP.

〈OUTPUT〉

【Return】

【Examples】

List 2-1: How to construct SCSceneManager for triangle soup

```
SCSceneManager scene(SC_SCENE_MANAGER_TRIANGLE_SOUP);
```

List 2-2: How to construct SCSceneManager for closed polyhedra

```
SCSceneManager scene(SC_SCENE_MANAGER_CLOSED_POLYHEDRA);
```

2.1.2 ～SCSceneManager ()

【Syntax】

```
～SCSceneManager (void);
```

【Description】

The destructor of SCSceneManager. All the SCObjects added in the scene are deleted from the scene by DeleteObject. Please note that SCSceneManager does not call the destructor of SCObject.

【Arguments】

【Return】

2.1.3 SetAttribute ()

【Syntax】

SCint SetAttributeDouble(SCenum attribute,SCdouble value);

SCint SetAttributeFloat(SCenum attribute,SCfloat value);

SCint SetAttributeInteger(SCenum attribute,SCint value);

SCint SetAttributeEnum(SCenum attribute,SCenum value);

【Description】

Sets attributes of SCSceneManager.

【Arguments】

〈INPUT〉

Attribute Attribute to set. List of attributes are shown in Table A- 1

Value Value to set.

〈OUTPUT〉

【Return】

SC_NO_ERROR: There has been no error.

SC_INVALID_ATTRIBUTE: The attribute to set is invalid.

SC_INVALID_VALUE: The value to set is invalid.

【Examples】

List 2-3: How to set the tolerance value and maximum iteration of penetration depth computation

```
SCdouble tolerance=0.1;// the tolerance value of calculation
SCdouble safetyCoefficient=0.49;// the safety coefficient
SCint maxIteration=10; // maximum iteration
SCSceneManager scene(SC_SCENE_MANAGER_CLOSED_POLYHEDRA);
// Setting of transformation and attributes of SCSceneManager
…
scene.SetAttributeDouble(SC_SCENE_MANAGER_TOLERANCE,tolerance);
scene.SetAttributeInteger(SC_SCENE_MANAGER_SAFETY_COEFFICIENT,safetyCoefficient);
scene.SetAttributeInteger(SC_SCENE_MANAGER_MAX_ITERATION,maxIteration);
```

2.1.4 GetAttribute ()

【Syntax】

```
SCint GetAttributeDouble(SCenum attribute, SCdouble&value);
SCint GetAttributeFloat(SCenum attribute, SCfloat&value);
SCint GetAttributeInt(SCenum attribute, SCint&value);
SCint GetAttributeEnum(SCenum attribute, SCenum&value);
SCint GetAttributeString(SCenum attribute, const SCchar*&value);
```

【Description】

Gets attributes of SCSceneManager.

【Arguments】

〈INPUT〉

Attribute	Attribute to set. List of attributes are shown in Table A- 1

〈OUTPUT〉

Value	Value to get.

【Return】

SC_NO_ERROR: There has been no error.

SC_INVALID_ATTRIBUTE: The attribute to get is invalid.

SC_INVALID_VALUE: The value to get is invalid.

【Examples】

List 2-4: How to get attributes of SCSceneManager

```
SCdouble tolerance;// the tolerance value of calculation
SCdouble safetyCoefficient;// the safety coefficient
SCint maxIteration; // maximum iteration
SCSceneManager scene(SC_SCENE_MANAGER_CLOSED_POLYHEDRA);
…
scene.GetAttributeDouble(SC_SCENE_MANAGER_TOLERANCE,tolerance);
scene.GetAttributeInteger(SC_SCENE_MANAGER_SAFETY_COEFFICIENT,safetyCoefficient);
scene.GetAttributeInteger(SC_SCENE_MANAGER_MAX_ITERATION,maxIteration);
```

2.1.5 AddObject()

【Syntax】

SCint AddObject (SCint id, SCObject*object);

【Description】

Adds SCObject in the scene. The information of the shape, position and orientation is obtained from SCObject. By default, the objects are automatically added to the static group. It is not possible to add a group in multiple scenes simultaneously.

【Arguments】

〈INPUT〉

id The ID to set to the object. The value of ID is chosen by the user and must be a positive integer.

object The address of SCObject to add.

〈OUTPUT〉

【Return】

SC_NO_ERROR: There has been no error.

SC_ERROR_DUPLICATE_ID: The *id* specified has already been registered.

SC_ERROR_DUPLICATE_ENTRY: *object* specified has already been added in the scene.

SC_ERROR_INVALID_TYPE_COMBINATION: The type of SCObject does not match the type of SCSceneManager.

SC_ERROR_NO_GEOMETRY: The object has no geometry.

SC_ERROR_INVALID_LICENSE: The license is invalid.

SC_ERROR_BAD_ALLOCATION: Bad allocation has happened during the execution.

SC_ERROR_RUNTIME: Runtime error has happened during the execution.

【Examples】

List 2-5: How to add objects to the scene.

```
SCSceneManager scene(SC_SCENE_MANAGER_CLOSED_POLYHEDRA);
SCObject object1(SC_OBJECT_TYPE_CLOSED_POLYHEDRA);
SCObject object2(SC_OBJECT_TYPE_CLOSED_POLYHEDRA);
SCObject object3(SC_OBJECT_TYPE_CLOSED_POLYHEDRA);
SCObject object4(SC_OBJECT_TYPE_CLOSED_POLYHEDRA);
SCObject object5(SC_OBJECT_TYPE_CLOSED_POLYHEDRA);
SCObject object6(SC_OBJECT_TYPE_CLOSED_POLYHEDRA);
// Add triangles for each SCObject
```

```
…
scene.AddObject(0,&object1);
scene.AddObject(1,&object2);
scene.AddObject(2,&object3);
scene.AddObject(3,&object4);
scene.AddObject(4,&object5);
scene.AddObject(5,&object6);
```

2.1.6 DeleteObject()

【Syntax】

SCint DeleteObject (SCint id);

【Description】

Deletes the object in the scene. The object specified by *id* must have been registered by AddObject.

【Arguments】

〈INPUT〉

id The ID of the object to be deleted.

〈OUTPUT〉

【Return】

SC_NO_ERROR: There has been no error.

SC_ERROR_INVALID_ID: The *id* specified is invalid.

【Examples】

List 2-6: How to delete objects from the scene.

```
…
scene.DeleteObject(4);
scene.DeleteObject(5);
```

2.1.7 AddObjectToGroup()

【Syntax】

SCint AddObjectToGroup(SCint gid,SCint id);

SCint AddControlledObject(SCint id, SCint gid=SC_DEFAULT_GROUP_ID);

【Description】

Adds the object specified by *id* to the group specified by *gid*. The object specified by *id* must have been registered by AddObject. If the group specified by *gid* does not exist, the new group is created.

A group of objects is treated as one object. The center of rotation of the object which is added first to the group is adopted as the center of rotation of the the group.

AddControlledObject is the obsolete form of AddObjectToGroup.

【Arguments】

〈INPUT〉

id	ID of the object to be added to the group.
gid	ID of the group to be added to. SC_STATIC_GROUP_ID is a negative integer and is reserved as the static group. SC_STATIC_GROUP_ID must not be specified for *gid*.

〈OUTPUT〉

【Return】

SC_NO_ERROR: There has been no error.

SC_ERROR_INVALID_ID: The *id* specified is invalid.

SC_ERROR_INVALID_GROUP_ID: The *gid* specified is invalid.

【Examples】

List 2-7: How to add objects to groups.

```
…
scene.AddObjectToGroup(0,0);
scene.AddObjectToGroup(0,1);
scene.AddObjectToGroup(1,2);
scene.AddObjectToGroup(1,3);
```

2.1.8 DeleteObjectFromGroup()

【Syntax】

SCint DeleteObjectFromGroup(SCint gid,SCint id);

SCint DeleteControlledObject (SCint id, SCint gid=SC_DEFAULT_GROUP_ID);

【Description】

Deletes the object specified by id from the group. The object specified by *id* must have been registered by AddObject and added to the group by AddObjectToGroup. Objects deleted from their groups are automatically returned to the static group.

DeleteControlledObject is the obsolete form of DeleteObjectToGroup.

【Arguments】

〈INPUT〉

id ID of the object to be deleted from the group.

gid ID of the group to be deleted from.

SC_STATIC_GROUP_ID is a negative integer and is reserved as the static group. SC_STATIC_GROUP_ID must not be specified for *gid*.

〈OUTPUT〉

【Return】

SC_NO_ERROR: There has been no error.

SC_ERROR_INVALID_ID: The *id* specified is invalid.

SC_ERROR_INVALID_GROUP_ID: The *gid* specified is invalid.

【Examples】

List 2-8: How to delete objects from groups.

```
…
scene.DeleteFromGroup(0,1);
scene.DeleteFromGroup(1,3);
```

2.1.9 DeleteGroup()

【Syntax】

SCint DeleteGroup(SCint gid);

【Description】

Deletes a group specified by *gid*.

【Arguments】

〈INPUT〉

gid	ID of the group to be deleted. SC_STATIC_GROUP_ID is a negative integer and is reserved as the static group. SC_STATIC_GROUP_ID must not be specified for *gid*.

〈OUTPUT〉

【Return】

SC_NO_ERROR: There has been no error.

SC_ERROR_INVALID_GROUP_ID: The *gid* specified is invalid.

【Examples】

List 2-9: How to delete a group.

```
…
scene.DeleteGroup(0);
```

2.1.10 ResetGroup()

【Syntax】

SCint ResetGroup(void);

【Description】

Deletes all existing groups and adds all objects to the static group whose *gid* is SC_STATIC_GROUP_ID.

【Arguments】

〈INPUT〉

〈OUTPUT〉

【Return】

SC_NO_ERROR: There has been no error.

【Examples】

List 2-10: How to reset groups

```
…
scene.ResetGroup();
```

2.1.11 SetActivity()

【Syntax】

```
SCint SetActivityGroup (SCint gid,SCenum type);
SCint SetActivityGroupPair (SCint gid1,SCint gid2, SCenum type);
SCint SetActivityObject (SCint oid,SCenum type);
SCint ActivateObject (SCint oid);
SCint DeactivateObject (SCint oid);
```

【Description】

Sets activity of the object, the group or the pair of groups specified by ID or IDs.

ActivateObject and DeactivateObject are the obsolete forms of SetActivity.

ActivateObject(oid) is equivalent to SetActivityObject (oid , SC_ACTIVITY_ACTIVE).

DeactivateObject(oid) is equivalent to SetActivityObject (oid , SC_ACTIVITY_INACTIVE).

【Arguments】

〈INPUT〉

oid The ID of the object to set activity of.

gid The ID of the group to set activity of.

gid1,gid2 The IDs of the pair of groups to set activity of.

type Possible types of activities are as follows.

- ■ SC_ACTIVITY_ACTIVE: The object, the group or the pair of groups is taken into account of collision detection.
- ■ SC_ACTIVITY_INACTIVE: The object, the group or the pair of groups is not taken into account of collision detection. Statuses of collision detection are discarded.
- ■ SC_ACTIVITY_PASSIVE: The group is taken into account of collision detection only if the other group is active. The penetration depth computation is performed only in the direction from the active group to the passive group.
- ■ SC_ACTIVITY_SLEEPING: The group is taken into account of collision detection only if the other group is active. However, statuses of collision detection are kept in memory, even if the other group is passive.
- ■ SC_ACTIVITY_ONE_WAY_ACTIVE: The collision detection from *gid1* to *gid2* is performed, ignoring the other configurations.
- ■ SC_ACTIVITY_ONE_WAY_INACTIVE: The collision detection from *gid1* to *gid2* is not performed, ignoring the other configurations.

Activities of collision detection according to the activities of two groups are shown in Appendix A- 2.

〈OUTPUT〉

【Return】

SC_NO_ERROR: There has been no error.

SC_ERROR_INVALID_TYPE: The type specified is invalid.

SC_ERROR_INVALID_ID: The *id* specified is invalid.

SC_ERROR_INVALID_GROUP_ID: The *gid* specified is invalid.

【Examples】

List 2-11: How to set activities of objects

```
…
scene.SetActivityObject(0,SC_ACTIVITY_ACTIVE);
scene.SetActivityObject(1,SC_ACTIVITY_INACTIVE);
```

List 2-12: How to set activities of groups

```
…
scene.SetActivityGroup(0,SC_ACTIVITY_ACTIVE);
scene.SetActivityGroup(1,SC_ACTIVITY_PASSIVE);
scene.SetActivityGroup(2,SC_ACTIVITY_INACTIVE);
scene.SetActivityGroup(3,SC_ACTIVITY_SLEEPING);
scene.SetActivityGroup(4,SC_ACTIVITY_ACTIVE);
scene.SetActivityGroup(5,SC_ACTIVITY_SLEEPING);
scene.SetActivityGroup(6,SC_ACTIVITY_PASSIVE);
```

List 2-13: How to set activities of group pairs

```
…
scene.SetActivityGroup(0,SC_ACTIVITY_ACTIVE);
scene.SetActivityGroup(1,SC_ACTIVITY_PASSIVE);
scene.SetActivityGroup(2,SC_ACTIVITY_INACTIVE);
scene.SetActivityGroup(3,SC_ACTIVITY_ACTIVE);
scene.SetActivityGroup(4,SC_ACTIVITY_ACTIVE);
scene.SetActivityGroup(5,SC_ACTIVITY_SLEEPING);
scene.SetActivityGroup(6,SC_ACTIVITY_PASSIVE);
// Before set activities of group pairs
scene.SetActivityGroupPair(2,6,SC_ACTIVITY_ACTIVE);
scene.SetActivityGroupPair(0,4,SC_ACTIVITY_INACTIVE);
scene.SetActivityGroupPair(1,5,SC_ACTIVITY_ONE_WAY_ACTIVE);
scene.SetActivityGroupPair(0,3,SC_ACTIVITY_ONE_WAY_INACTIVE);
// After set activities of group pairs
```

2.1.12 SetTransformation()

【Syntax】

SCint SetTransformation(SCint gid,SCenum type, const Float*trans);

SCint SetTransformation(SCint gid,SCenum type, const SCdouble*trans);

SCint SetTransformation(SCenum type, const Float*trans, SCint gid=0);

SCint SetTransformation(SCenum type, const SCdouble*trans, SCint gid=0);

【Description】

Sets transformation of the group specified by *gid*.

【Arguments】

〈INPUT〉

gid ID of a group. SC_STATIC_GROUP_ID is a negative integer and is reserved as static object group.

ID of the group to be set transformation of.

SC_STATIC_GROUP_ID is a negative integer and is reserved as the static group. SC_STATIC_GROUP_ID must not be specified for *gid*.

trans Transformation to set

type The type of transformation. Possible types of transformation are shown in Appendix A- 3.

〈OUTPUT〉

【Return】

SC_NO_ERROR: There has been no error.

SC_ERROR_INVALID_TYPE: The type specified is invalid.

SC_ERROR_INVALID_GROUP_ID: The *gid* specified is invalid.

【Examples】

List 2-14: How to set transformations for groups.

```
SCdouble position[3]={100.0, 200.0, -150.0};
SCdouble center1[3]={50.0, 100.0, -75.0};
SCdouble center2[3]={100.0, 100.0, 100.0};
SCdouble orientation[4]={1.0, 0.0, 0.0, 0.0};
SCdouble matrix[16]={1.0, 0.0, 0.0, 0.0,
                     0.0, 1.0, 0.0, 0.0,
                     0.0, 0.0, 1.0, 0.0,
                     50.0, -30.0, 100.0, 1.0};

scene.SetTransformation(0,SC_NEW_WORLD_CENTER,center1);
scene.SetTransformation(0,SC_POSITION_WORLD_CENTER,position);
scene.SetTransformation(0,SC_ORIENTATION_QUATERNION,orientation);

scene.SetTransforamtion(1,SC_TRANSFORMATION_MATRIX,matrix);
scene.SetTransformation(1,SC_NEW_WORLD_CENTER,center2);
```

2.1.13 GetTransformation()

【Syntax】

SCint GetTransformation(SCint gid ,SCenum type,SCdouble position[3]) const;

SCint GetTransformation(SCenum type,SCfloat position[3], SCint gid=0) const;

【Description】

Gets transformation of the group specified by *gid*.

【Arguments】

〈INPUT〉

gid	ID of a group. SC_STATIC_GROUP_ID is a negative integer and is reserved as static object group.
type	The type of transformation. Possible types of transformation are shown in Appendix A- 3.

〈OUTPUT〉

position	Position to get

【Return】

SC_NO_ERROR: There has been no error.

SC_ERROR_INVALID_TYPE: The type specified is invalid.

SC_INVALID_ID: The ID specified is invalid.

SC_ERROR_INVALID_GROUP_ID: The *gid* specified is invalid.

【Examples】

List 2-15: How to get transformations of groups.

```
SCdouble position[3];
SCdouble orientation[4];
SCdouble matrix[16];

scene.GetTransformation(0,SC_POSITION_WORLD_CENTER,position);
scene.GetTransformation(0,SC_ORIENTATION_QUATERNION,orientation);
scene.GetTransforamtion(1,SC_TRANSFORMATION_MATRIX,matrix);
```

2.1.14 UpdateStatus()

【Syntax】

SCint UpdateStatus(void);

【Description】

Updates the statuses of distance computation or penetration depth computation with respect to current transformations.

【Arguments】

〈INPUT〉

〈OUTPUT〉

【Return】

SC_NO_ERROR: If there is no error, otherwise as follows.

SC_ERROR_BAD_ALLOCATION: Bad allocation has happened during the execution.

SC_ERROR_RUNTIME: Runtime error has happened during the execution.

In the former version (Ver. 2.01 or older), this method returns the result of the collision detection between the first pair of the objects, such as SC_ERROR_INVALID_INITIAL_TRANSFORMATION, SC_ERROR_UNKNOWN_DISTANCE, SC_ERROR_NO_RESULT, SC_ERROR_FAILED. However, in latter version, this function returns the status of the execution of collision detection.

【Examples】

List 2-16: How to execute collision detection

```
SCSceneManager scene(SC_SCENE_MANAGER_CLOSED_POLYHEDRA);
// Setting of transformation and attributes of SCSceneManager
…
scene.UpdateStatus();
```

2.1.15 GetStatus()

【Syntax】

```
SCint GetStatus(SCenum type, SCint*status);
SCint GetStatus(SCenum type,SCint*status, SCint index, SCbool reverseFlag=false);
SCint GetStatus(SCenum type,SCfloat*status, SCint index, SCbool reverseFlag=false);
SCint GetStatus(SCenum type,SCdouble*status, SCint index, SCbool reverseFlag=false);
```

【Description】

Gets the statuses of minimum distance/penetration depth computation between groups of objects. If there are more than two groups of objects, the number of pairs is at most the number of combination of the two groups. Only the statues between the pairs of two groups whose distance are within less equal than SC_MAX_DISTANCE can be obtained. The number of pairs can be obtained by GetStatus(SC_PAIR_COUNT,pairCount). If the *reverseFlag* is true, the status in which the order of the groups is reversed can be obtained. Although, the group ID which comes first is not determined generally, if one of the IDs is SC_STATIC_GROUP_ID, the other ID comes first, in the case of which *reverseFlag* is false. *index* must be specified except for SC_PAIR_COUNT. Types except for SC_PAIR_COUNT need *index* and *reverseFlag* for parameters. However, GetStatus(SCenum type, SCint*status) can be used for any other type than SC_PAIR_COUNT by assuming *index*=0, *reverseFlag*=false for compatibility with older versions.

【Arguments】

〈INPUT〉

index	Index specifies one of the results. Index starts at 0 and must be smaller than the number of the status. The number of status can be obtained by SC_PAIR_COUNT.
type	Type of status to get. Possible types of status are shown in Appendix A- 4.
reverseFlag	If the *reverseFlag* is true, the status in which the roles of target and opponent are reversed can be obtained.

〈OUTPUT〉

status	Status to get.

【Return】

SC_NO_ERROR: There has been no error. Distance computation or penetration depth computation has been performed normally.

SC_ERROR_NO_RESULT: There is no result.

SC_ERROR_INVALID_TYPE: The type specified is invalid.

SC_ERROR_INVALID_INDEX: The index specified is invalid.

SC_ERROR_FAILED: Failed to get status.

【Examples】

List 2-17: How to get the number of pairs

```
SCSceneManager scene(SC_SCENE_MANAGER_CLOSED_POLYHEDRA);
// Setting of transformation and attributes of SCSceneManager
// Execution of collision detection of current configurations
…
SCint count;
scene.GetStatus(SC_PAIR_COUNT,&count);//get the number of pairs

for(int i=0;i<count;i++){
    // Get status about each pair
}
```

List 2-18: How to get status information.

```
SCint result;
SCint gids[2];// group IDs
SCint oids[2];// object IDs
SCint pids[2];// piece IDs
SCdouble distance;
SCdouble normal[3];
SCdouble point1[3],point2[3];
SCdouble tpdv[3],rpdv[3];
SCdouble contactPosition[3],contactOrientation[4];
SCint featureTypes[2];
SCint featureIndices1[3],featureIndices1[3];

scene.GetStatus(SC_GROUP_ID,gids,i,false);// Get the group IDs
                                          // Target: gids[0], Opponent: gids[1]
scene.GetStatus(SC_STATUS_RESULT,&result,i,false); // Get the result
switch(result){
case SC_NO_ERROR:
    // Minimum distance compultation or penetration depth computation has succeeded
    scene.GetStatus(SC_OBJECT_ID,oids,i,false);// Get the object IDs
    scene.GetStatus(SC_PIECE_ID,pids,i,false);// Get the piece IDs
    scene.GetStatus(SC_DISTANCE,&distance,i,false);// Get the distance
    scene.GetStatus(SC_CONTACT_NORMAL,normal,i,false);// Get the contact normal
    scene.GetStatus(SC_POINT_ON_TARGET,point1,i,false);// Get the end point on the target
    scene.GetStatus(SC_POINT_ON_OPPONENT,point2,i,false);// Get the end point on the opponent
    scene.GetStatus(SC_FEATURE_TYPE,featureTypes,i,false);// Get the feature types
    scene.GetStatus(SC_FEATURE_ON_TARGET,
                    featureIndices1,i,false);// Get the features on the target
    scene.GetStatus(SC_FEATURE_ON_OPPONENT,
                    featureIndices2,i,false);// Get the features on opponent
    if(distance<=0){
        // Penetration depth computation was performed
        scene.GetStatus(SC_TPD_VECOTR,tpdv,i,false);// Get the TPDV
        scene.GetStatus(SC_RPD_VECOTR,rpdv,i,false);// Get the RPDV
        scene.GetStatus(SC_CONTACT_POSITION,
                        contactPosition,i,false);   // Get the contact position
        scene.GetStatus(SC_CONTACT_ORIENTATION,
                        contactOrientation,i,false);// Get the contact orientation
    }else{
        // Minimum distance compultation was performed
```

```
   }
   break;
case SC_ERROR_INVALID_INTIAL_TRANSFORMATION:
   // There is intersection, but penetration depth computation could not be performed.
   break;
case SC_ERROR_NO_RESULT:
   // There is no result in this direction
   break;
default:
   // Fatal error
   break;
}
```

List 2-19: How to get status information focusing on a particular group.

```
#define MOVING_GROUP_ID 100
…
scene.GetStatus(SC_GROUP_ID,gids,i,false);// Get the group IDs
                                          // Target: gids[0], Opponent: gids[1]
bool reverseFlag;
if(gids[0]==MOVING_GROUP_ID){
     reverseFlag=false;
}else if(gids[1]==MOVING_GROUP_ID){
     reverseFlag=true;
}else{
   continue;
}
scene.GetStatus(SC_STATUS_RESULT,&result,i, reverseFlag); // Get the result
switch(result){
case SC_NO_ERROR:
   // Minimum distance compultation or penetration depth computation has succeeded
   scene.GetStatus(SC_OBJECT_ID,oids,i,reverseFlag);// Get the object IDs
   scene.GetStatus(SC_PIECE_ID,pids,i,reverseFlag);// Get the piece IDs
   scene.GetStatus(SC_DISTANCE,&distance,i,reverseFlag);// Get the distance
   scene.GetStatus(SC_CONTACT_NORMAL,normal,i,reverseFlag);// Get the contact normal
   …
   break;
case SC_ERROR_INVALID_INTIAL_TRANSFORMATION:
   // There is intersection, but penetration depth computation could not be performed.
   break;
case SC_ERROR_NO_RESULT:
   // There is no result in this direction
   break;
default:
   // Fatal error
   break;
}
```

2.1.16 ResetStatus()

【Syntax】

SCint ResetStatus(void);

【Description】

Resets the statuses of distance computation or penetration depth computation.

【Arguments】

〈INPUT〉

〈OUTPUT〉

【Return】

SC_NO_ERROR: There has been no error.

【Examples】

List 2-20: How to reset statuses

```
…
scene.ResetStatus();
```

2.2 The methods of SCObject

The methods of SCObject are as follows.

Table 2-2: Methods of SCObject

Categories	Methods
Constructor	SCObject ()
Destructor	～SCObject ()
Setting geometry	AddTriangles()

Setting/Getting transformation	SetTransformation()
	GetTransformation()

2.2.1 SCObject ()

【Syntax】

SCObject (SCenum type);

SCObject (void);

【Description】

The constructor of SCObject.

【Arguments】

〈INPUT〉

type

Type of triangles to set.

- ■ SC_OBJECT_TYPE_CLOSED_POLYHEDRA / SC_OBJECT_TYPE_CLOSED_POLYHEDRON: Closed polyhedra. This type of object can be add to SCSceneManager in SC_MODE_CLOSED_POLYHEDRA mode.
- ■ SC_OBJECT_TYPE_TRIANGLE_SOUP: Arbitrary set of triangles. So-called triangle soup. This type of object can be add to SCSceneManager in SC_MODE_TRIANGLE_SOUP mode.

〈OUTPUT〉

【Return】

【Examples】

List 2-21: How to make SCObject for triangle soup

```
SCObject object(SC_OBJECT_TYPE_TRIANGLE_SOUP);
```

List 2-22: How to make SCObject for closed polyhedra

```
SCObject object(SC_OBJECT_TYPE_CLOSED_POLYHEDRA);
```

2.2.2 ～SCObject ()

【Syntax】

～SCObject (void);

【Description】

The distructor of SCObject. If a SCObject is added to a SCSceneManager, it deletes itself by calling SCSceneManager::DeleteObject.

【Arguments】

【Return】

【Examples】

List 2-23: How to delete SCObject

```
SCObject*object=new SCObject(SC_OBJECT_TYPE_CLOSED_POLYHEDRA);
…
delete object;
```

2.2.3 AddTriangles()

【Syntax】

SCint AddTriangles (const SCfloat*vertices, SCint vertexNum, const SCint triangles, SCint triangleNum, const SCchar*bvhFile=NULL);

SCint AddTriangles (const SCdouble*vertices, SCint vertexNum, const SCint triangles, SCint triangleNum, const SCchar*bvhFile=NULL);

SCint AddTriangles (SCenum type, const SCfloat*vertices, SCint vertexNum, const SCint triangles, SCint triangleNum, const SCchar*bvhFile=NULL);

SCint AddTriangles (SCenum type, const SCdouble*vertices, SCint vertexNum, const SCint triangles, SCint triangleNum, const SCchar*bvhFile=NULL);

【Description】

Adds a set of triangles to the object. A set of triangles added by this method is called a piece. The conditions for each piece for closed polyhedron are as follows. (1)All edges in each pieces are shared by only two triangles. (2)This means that there are no duplicate or branched edges in each piece.(3)There is no degeneration in each triangle.

Each piece must be single boundary. It is possible to call AddTriangles at multiple times, but combinations of different type of model are not allowed.

【Arguments】

〈INPUT〉

type Type of triangles to set.

■ SC_OBJECT_TYPE_CLOSED_POLYHEDRA / SC_OB JECT_TYPE_CLOSED_POLYHEDRON: Closed polyhedra. This type of object can be add to SCSceneManager in SC_MODE_CLOSED_POLYHEDRA mode.

■ SC_OBJECT_TYPE_TRIANGLE_SOUP: Arbitrary set of triangles. So-called triangle soup. This type of object can be add to SCSceneManager in SC_MODE_TRIANGLE_SOUP mode.

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.

bvhFile File name of BVH file to set, if the type of triangles is SC_OBJECT_TYPE_CLOSED_POLYHEDRA. Otherwise, ignores this argument. If BVH file exits, reads the file. If BVH file does not exist, creates the file. If BVH file is NULL or not specified, creates a temporary BVH when the object is added in the scene.

〈OUTPUT〉

【Return】

SC_NO_ERROR: There has been no error.

SC_ERROR_INVALID_TYPE: The type specified is invalid.

SC_ERROR_FAILED: Failed to execution.

SC_ERROR_INVALID_TYPE_COMBINATION: The combination of type of data is invalid.

SC_ERROR_INVALID_DATA: The data specified is invalid.

SC_ERROR_INVALID_BVH_FILE: The BVH file specified is invalid.

SC_ERROR_INVALID_LICENSE: The license is invalid.

SC_ERROR_BAD_ALLOCATION: Bad allocation has happened during the execution.

SC_ERROR_RUNTIME: Runtime error has happened during the execution.

【Examples】

List 2-24: How to set goemetry

```
SCdouble vertices[3*4]={
    0.0,0.0,0.0, // vertex 0
    1.0,0.0,0.0, // vertex 1
    0.0,1.0,0.0, // vertex 2
    0.0,0.0,1.0  // vertex 3
};
SCint triangles[3*4]={
    0,2,1, // triangle 0
    1,3,0, // triangle 1
    0,3,2, // triangle 2
    1,2,3  // triangle 3
};

SCObject object(SC_OBJECT_TYPE_CLOSED_POLYHEDRON);
If(object.AddTriangles(vertex,4,triangles,4)!=SC_NO_ERROR){
    // Input geometry is invalid
}
```

List 2-25: How to make the object consisting of multiple pieces.

```
SCObject object(SC_OBJECT_TYPE_CLOSED_POLYHEDRON);

If(object.AddTriangles(vertex1,vertexCount1,triangles1,triangleCount1)!=SC_NO_ERROR){
    // Input geometry is invalid
}
If(object.AddTriangles(vertex2,vertexCount2,triangles2,triangleCount2)!=SC_NO_ERROR){
    // Input geometry is invalid
}
If(object.AddTriangles(vertex3,vertexCount3,triangles3,triangleCount3)!=SC_NO_ERROR){
    // Input geometry is invalid
}
If(object.AddTriangles(vertex4,vertexCount4,triangles4,triangleCount4)!=SC_NO_ERROR){
    // Input geometry is invalid
}
```

List 2-26: How to make and reuse BVH

```
SCObject object(SC_OBJECT_TYPE_CLOSED_POLYHEDRON);

char bvhFile[]=”test.bvh”;

If(object.SetTriangles(vertex,vertexCount,triangles,triangleCount,bvhFile)!=SC_NO_ERROR){
   // Input geometry is invalid
}
```

2.2.4 SetTransformation()

【Syntax】

SCint SetTransformation(SCenum type,const SCfloat*transformation);

SCint SetTransformation(SCenum type,const SCdouble*transformation);

【Description】

Sets position of the object. After a SCObject has been added to SCSceneManager, it is required that you use the method SCSceneManger::SetTransformation to set the transformations of the SCObject (2.1.12).

【Arguments】

〈INPUT〉

transformation	Transformation to set.
type	The type of transformation. Possible types of transformation are shown in Appendix A- 3.

〈OUTPUT〉

【Return】

SC_NO_ERROR: There has been no error.

SC_ERROR_INVALID_TYPE: The type specified is invalid.

【Examples】

List 2-27: How to set transformation(1)

```
SCdouble local_center[3]={13,6,11};
SCdouble world_center[3]={10,50,35};
SCdouble orientation[4]={0.707107,0.707107,0,0};// 90 degree rotation around x axis

SCObject object(SC_OBJECT_TYPE_CLOSED_POLYHEDRON);                       // (1)
object.SetTransformation(SC_POSITION_NEW_LOCAL_CENTER, local_center); // (2)
object.SetTransformation(SC_POSITION_WORLD_CENTER,world_center);      // (3)
object.SetTransformation(SC_ORIENTATION_QUATERNION,orientation);      // (4)
```

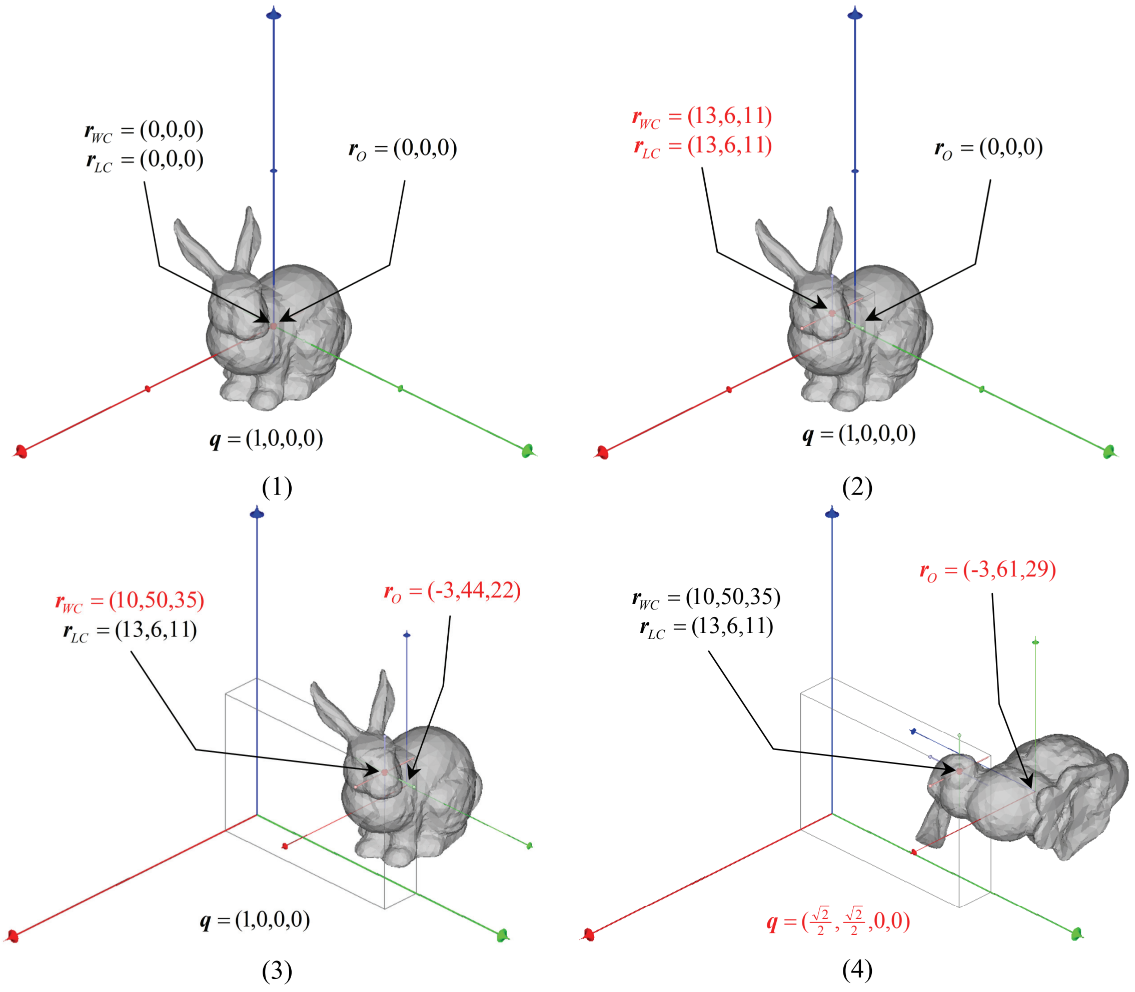
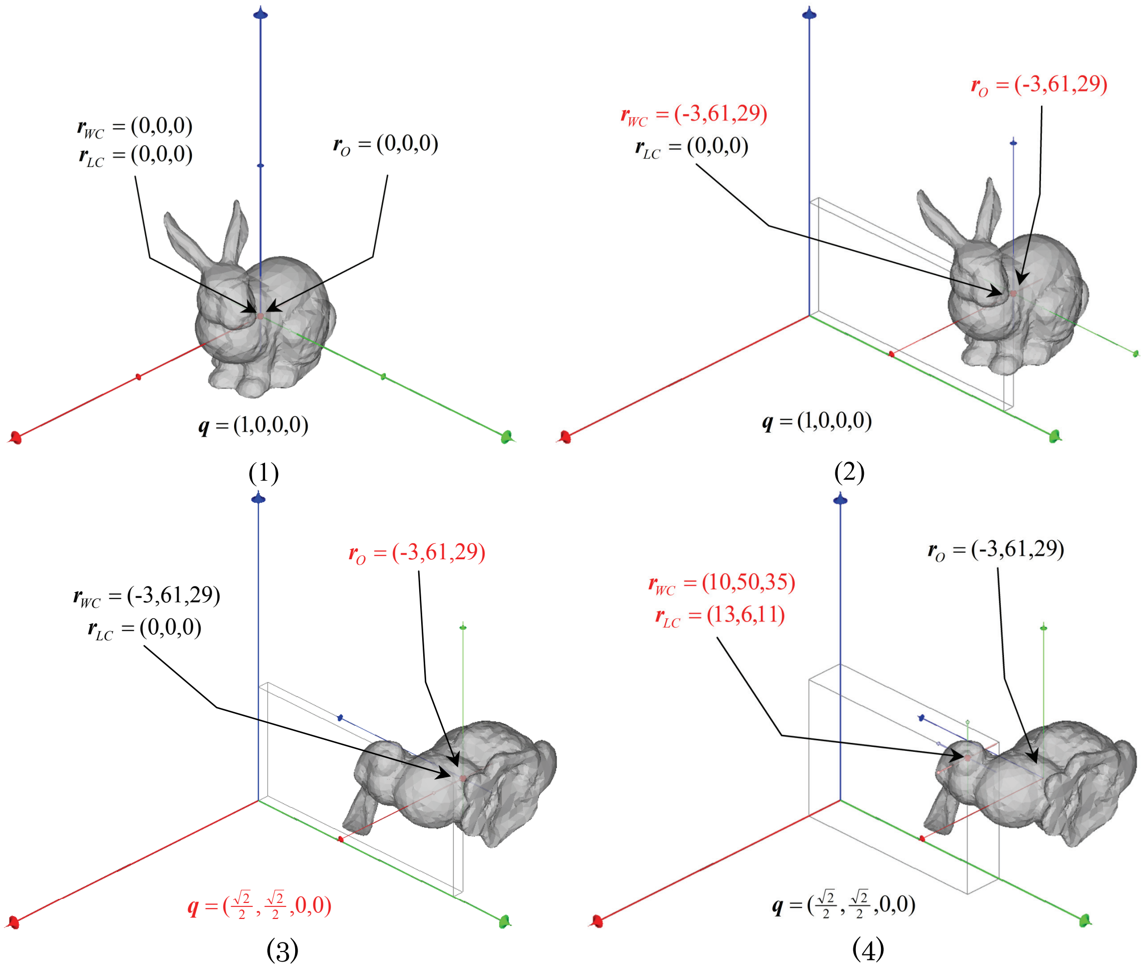


Figure 2-1: Transition of transformation(1)

List 2-28: How to set transformation(2)

```
SCdouble origin[3]={-3,61,29};
SCdouble world_center[3]={10,50,35};
SCdouble orientation[4]={0.707107,0.707107,0,0};// 90 degree rotation around x axis

SCObject object(SC_OBJECT_TYPE_CLOSED_POLYHEDRON);                    // (1)
object.SetTransformation(SC_POSITION_ORIGIN,origin);                 // (2)
object.SetTransformation(SC_ORIENTATION_QUATERNION,orientation);      // (3)
object.SetTransformation(SC_POSITION_NEW_WORLD_CENTER,world_center);  // (4)
```



(1)

(2)

(3)

(4)

Figure 2-2: Transition of transformation(2)

2.2.5 GetTransformation()

【Syntax】

SCint GetTransformation(SCenum type,SCfloat position[3]) const;

SCint GetTransformation(SCenum type,SCdouble position[3]) const;

【Description】

Gets position of the object. Even after a SCObject has been added to SCSceneManager, it is possible to use SCObject::GetTransformation to get transformations for each object.

【Arguments】

〈INPUT〉

type	The type of transformation. Possible types of transformation are shown in Appendix A- 3.

〈OUTPUT〉

transformation	Transformation to get

【Return】

SC_NO_ERROR: There has been no error.

SC_ERROR_INVALID_TYPE: The type specified is invalid.

【Examples】

List 2-29: How to get transformation

```
SCdouble world_center[3];
SCdouble orientation[4]
SCdouble matrix[16];

…
object.GetTransformation(SC_POSITION_WORLD_CENTER,world_center);
object.GetTransformation(SC_ORIENTATION_QUATERNION,orientation);
object.GetTransformation(SC_POSITION_LOCAL_CENTER, matrix);
```

Appendix A

Appendix A- 1 Attributes of SCSceneManager

Table A- 1 shows attributes of SCSceneManager.

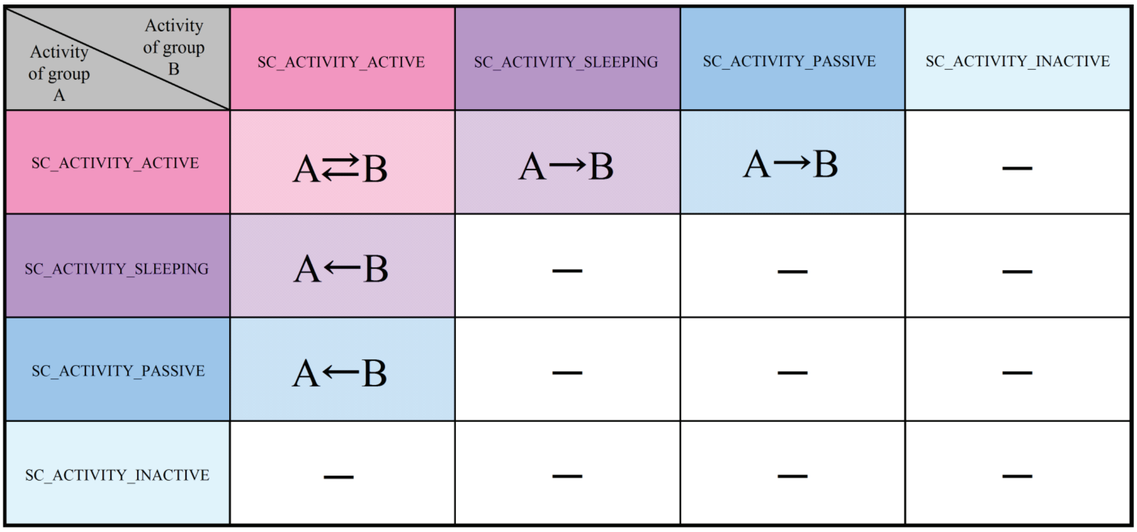
Table A- 1:Attributes of SCSceneManager

Name of attributes	Type	Units	Description
SC_SCENE_MANAGER_TOLERANCE	SCdouble SCfloat	[Length]	Tolerance for penetration depth computation. Default value is 0.2.
SC_SCENE_MANAGER_ROTATION_MODE	SCenum		Rotation mode. ■ SC_ROTATION_MODE_NONE: Contact orientation keeps the value at the time when the penetration happened. This is the default. ■ SC_ROTATION_MODE_INPUT: The combination of the penetration depth vector and the penetration rotation vector is determined such that the norm of the penetration rotation vector has the minimum value. ■ SC_ROTATION_MODE_FREE: The combination of the penetration depth vector and the penetration rotation vector is determined such that the norm of the penetration depth has the minimum value. ■ SC_ROTATION_MODE_MIX: The combination of the penetration depth vector and the penetration rotation vector is determined such that potential has the minimum value.
SC_SCENE_MANAGER_MAX_ITERATION	SCint		Maximum iteration of penetration depth computation. Default value is 5.
SC_SCENE_MANAGER_MAX_DISTANCE	SCdouble SCfloat	[Length]	Maximum distance for distance computation. If the distance between the controlled object and the static object beyond the value, the results of distance computation may be unknown. Default value is 0.5.
SC_SCENE_MANAGER_FORCE_STIFFNESS	SCdouble SCfloat	[Force/Length]	Stiffness of force to calculate potential. Defalut value is 0.4.
SC_SCENE_MANAGER_TORQUE_STIFFNESS	SCdouble SCfloat	[Force*Length]	Stiffness of torque to calculate potential. Default value is 100.
SC_SCENE_MANAGER_SAFETY_COEFFICIENT	SCdouble SCfloat		Safety coefficient of penetration depth computation. The value must be larger than 0 and less than 0.5. Default value is 0.49.
SC_SCENE_MANAGER_PENETRATION_DEPTH_COMPUTATION	SCenum		Execution of penetration depth computation. ■ SC_TRUE : Penetration depth computation is enabled. This is the default. ■ SC_FALSE: Penetration depth computation is disabled.
SC_ VERSION	const SCchar*		Version of SmartCollision.

Appendix A- 2 Activities of minimum distance computation

Table A- 2 shows activities of collision detection according to the activities of two groups. In Table A- 2 , the group at the origin of the arrow plays the role of the **target**, and the group at the head of the arrow plays the role of the **opponent**.

Table A- 2: Activities of collision detection according to the activities of two groups



Activity of group B Activity of group A	SC_ACTIVITY_ACTIVE	SC_ACTIVITY_SLEEPING	SC_ACTIVITY_PASSIVE	SC_ACTIVITY_INACTIVE
SC_ACTIVITY_ACTIVE	A⇄B	A→B	A→B	—
SC_ACTIVITY_SLEEPING	A←B	—	—	—
SC_ACTIVITY_PASSIVE	A←B	—	—	—
SC_ACTIVITY_INACTIVE	—	—	—	—

Appendix A- 3 Types of transformation

Table A- 3 shows types of transformation.

Table A- 3:Types of transformation

Type of transformation	Type of array	Size	Description
SC_POSITION_ORIGIN	SCfloat SCdouble	3	The origin of local coordinates system of the object in world coordinates system.
SC_POSITION_WORLD_CENTER	SCfloat SCdouble	3	The center of rotation of the object in world coordinates system.
SC_POSITION_LOCAL_CENTER	SCfloat SCdouble	3	The center of rotation of the object in local coordinates system.
SC_ORIENTATION_QUATERNION	SCfloat SCdouble	4	The orientation of the object specified by quaternion.
SC_ORIENTATION_MATRIX	SCfloat SCdouble	16	The orientation of the object specified by matrix. Transformation is expressed by 4x4 matrix.
SC_TRANSFORMATION_MATRIX	SCfloat SCdouble	16	Transformation of the object specityed by 4x4 matrix.
SC_POSITION_NEW_WORLD_CENTER	SCfloat SCdouble	3	The new center of rotation of the object in world coordinates system.
SC_POSITION_NEW_LOCAL_CENTER	SCfloat SCdouble	3	The new center of rotation of the object in local coordinates system.

Appendix A- 4 Types of status

Table A- 4 shows types of status of SCSceneManager.

Table A- 4: Types of status of SCSceneManager

Type of transformation	Type of array	Size	Description
SC_PAIR_COUNT SC_STATUS_COUNT	SCint	1	The number of pair.
SC_ STATUS_ RESULT	SCint	1	The status of result. The possible values are as follows. ■ SC_NO_ERROR: Either minimum distance computation or penetration depth computation was executed normally. ■ SC_ERROR_INVALID_INITIAL_TRANSFORMATION : There were intersections, before penetration depth computation started. In this case, only the results of SC_GROUP_ID, SC_OBJECT_ID, SC_TARGET_OBJECT_ID , SC_OPPONENT_OBJECT_ID, SC_PIECE_ID can be obtained. ■ SC_ERROR_NO_RESULT: There was no result between the pair in this direction. In this case, no other result can be obtained. ■ SC_ERROR_UNKNOWN_DISTANCE: In version 2.01 or former versions, if the distance is larger than the maximum distance of minimum distance computation, the status information can be SC_ERROR_UNKNOWN_DISTANCE or SC_ERROR_NO_RESULT. Here, SC_ERROR_UNKNOWN_DISTANCE means the distance is larger than the maximum distance. However, in version 2.1 or later, the status information for this case is SC_ERROR_NO_RESULT. Although SC_ERROR_UNKNOWN_DISTANCE still exists in version 2.1 or later, its value is equal to SC_ERROR_NO_RESULT.
SC_GROUP_ID	SCint	2	The IDs of the pair of the groups, which have end points of minimum distance / penetration depth computation. The group specified by the first ID in the array is the target, and the group specified by the next ID is the opponent. Although the group ID which comes first is not determined, if one of the IDs is SC_STATIC_GROUP_ID, the other ID comes first, in the case of which reverseFlag is false. There is only this rule about the order of group IDs. There is no other rule about the order of group. Therefore, the code assuming a rule of the order of group IDs may be affected by the situation and the version of the library.
SC_OBJECT_ID	SCint	2	The IDs of the pair of the objects, which have end points of minimum distance /penetration depth computation. The ID of the target comes first, and the ID of the opponent comes next in the array.
SC_TARGET_OBJECT_ID SC _CONTROLLED_OBJECT_ID	SCint	1	The ID of the target object, which has the end point of distance/penetration depth computation.
SC_OPPONENT_OBJECT_ID SC _STATIC_OBJECT_ID	SCint	1	The ID of the opponent object which has the end point of distance/penetration depth computation.
SC_PIECE_ID	SCint	2	The IDs of the pair of the pieces, which have end points of minimum distance /penetration depth computation. The ID of the target comes first, and the ID of the opponent comes in the array. Each piece corresponds to an indexed triangle set which is added by a call of SCObject::AddTriangles. The ID of piece is the order of calls of SCObject::AddTriangles and starts at 0.
SC_DISTANCE	SCfloat SCdouble	1	Positive value means the minimum distance between the pair. If the value is zero or negative, there is penetration between the pair. If penetration depth

			computation is enabled, the magnitude means magnitude of translational penetration depth vector. Otherwise,the magnitude has no meaning.
SC_CONTACT_NORMAL	SCfloat SCdouble	3	The normal between the target in contact state and opponent
SC_MD_VECTOR	SCfloat SCdouble	3	Minimum distance vector. If there are penetrations, this is zero vector.
SC_TPD_VECTOR	SCfloat SCdouble	3	Translational penetration depth vector. If there is no penetration, this is zero vector.
SC_RPD_VECTOR	SCfloat SCdouble	3	Rotational penetration depth vector. If there is no penetration, this is zero vector.
SC_RPD_QUATERNION	SCfloat SCdouble	4	Quaternion representation of rotational penetration depth vector. If there is no penetration, this is identical quaternion.
SC_PD_MATRIX	SCfloat SCdouble	16	Matrix representation of penetration depth.
SC_PD_INVERSE_MATRIX	SCfloat SCdouble	16	Inverse matrix of penetration depth.
SC_CONTACT_POSITION	SCfloat SCdouble	3	The contact position. The position is expressed by the center of rotation of the object in world coordinates system.
SC_CONTACT_ORIENTATION	SCfloat SCdouble	4	The contact orientation. The orientation is expressed by quaternion representation.
SC_CONTACT_TRANSFORMATION_MATRIX	SCfloat SCdouble	16	Contact transformation. Transformation is specified by 4x4 matrix.
SC_POINT_ON_TARGET SC_POINT_ON_CONTROLLED_OBJECT	SCfloat SCdouble	3	The end point on the target of the result of minimum distance/penetration depth computation.
SC_POINT_ON_TARGET_IN_CONTACT SC_POINT_ON_CONTACT_OBJECT	SCfloat SCdouble	3	The end point on the target in contact of the result of penetration depth computation.
SC_POINT_ON_OPPONENT SC_POINT_ON_STATIC_OBJECT	SCfloat SCdouble	3	The end point on the opponent of the result of distance/penetration depth computation.
SC_FEATURE_TYPE	SCint	2	The pair of feature types which have end points of minimum distance /penetration depth computation. The feature type of the target comes first, and the feature type of the opponent comes in the array. The possible values are as follows. ■ SC_FEATURE_VERTEX: The feature is the original vertex. ■ SC_FEATURE_FACE: The feature is the original face. ■ SC_FEATURE_TWO_VERTICES: The features are two vertices. ■ SC_FEATURE_THREE_VERTICES: The features are three vertices.
SC_FEATURE_ON_TARGET	SCint	1-3	The indices of feature on the target which have end points of minimum distance /penetration depth computation. If the feature type is SC_FEATURE_VERTEX/SC_FEATURE_FACE, then only one index of original vertex/face can be obtained. If the feature type is SC_FEATURE_TWO_VERTEX /SC_FEATURE_THREE_VERTEX, then 2/3 indices of vertices.
SC_FEATURE_ON_OPPONENT	SCint	1-3	The indices of feature on the opponent which have end points of minimum distance /penetration depth computation. If the feature type is SC_FEATURE_VERTEX/SC_FEATURE_FACE, then only one index of original vertex/face can be obtained. If the feature type is SC_FEATURE_TWO_VERTEX /SC_FEATURE_THREE_VERTEX, then 2/3 indices of vertices.