OBB Documentation

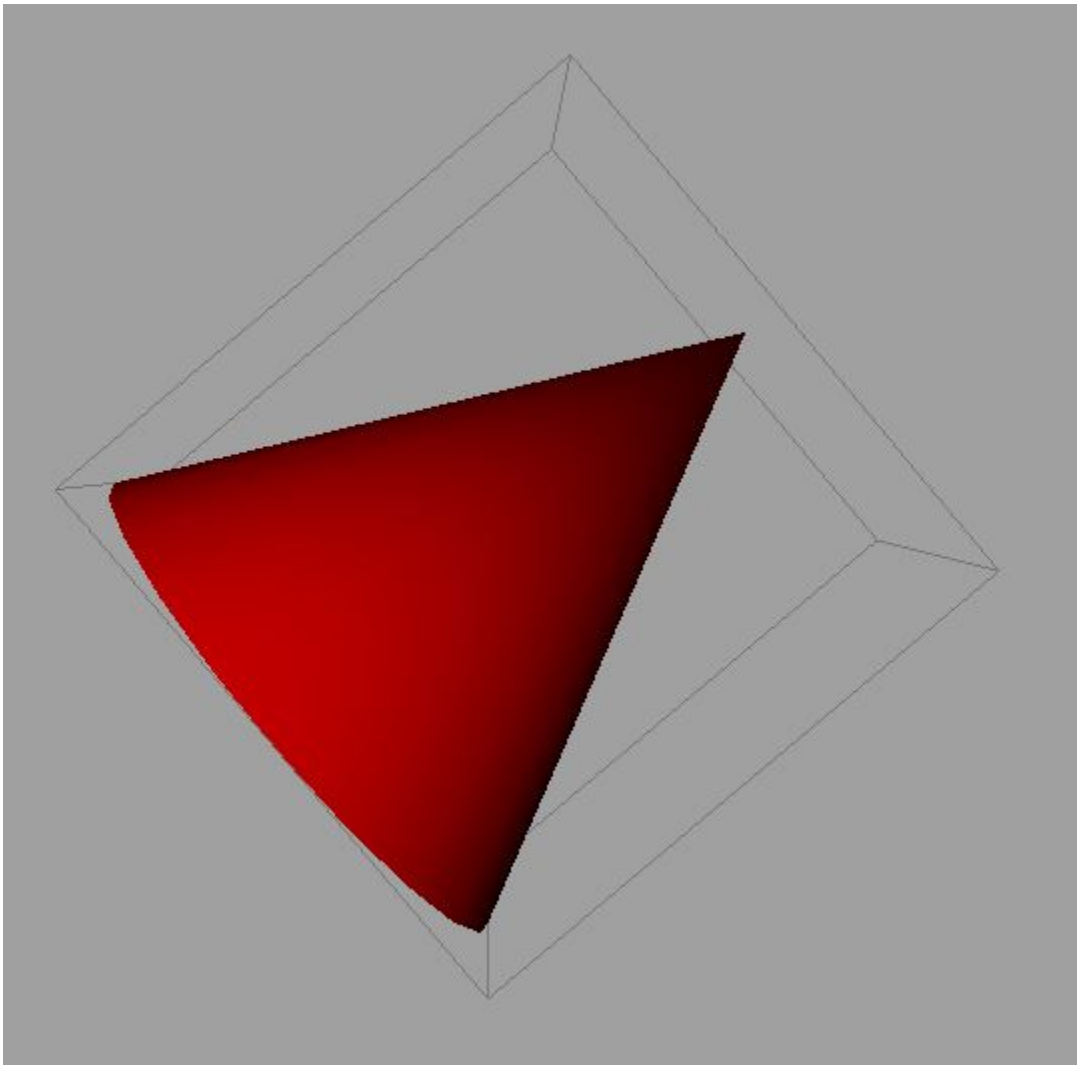
Release 0.1.6

Christopher DeVito

August 21, 2016

Contents

A simple python class that is based off code from here.



CHAPTER 1

Features

- 3 different solve methods (from points, triangles, and hull).
- Has a matrix attribute that can be applied to any transform in Maya (deformers, meshes, etc...).

CHAPTER 2

Planned Features

- Snap object to object.
- Translate, rotate, and scale attributes.
- Increase speed by making it a C++ plugin.

CHAPTER 3

Requirements

- Autodesk Maya 2015 (http://www.autodesk.com/products/maya/overview)

CHAPTER 4

Optional Requirements

- Scipy 0.16.0 (https://www.scipy.org/)
- Numpy 1.9.2 (http://www.numpy.org/)

CHAPTER 5

Table of Contents

5.1 Installation

5.1.1 Get OBB for Maya

Using the MEL setup script

- Download the package from the github repo http://github.com/chrisdevito/OBB.git and click Download Zip.
- After extraction, drag and drop the setup.mel (found in the OBB directory) into any open maya window.
- This will install it into your maya/scripts directory and add the OBB Shelf.

Using Pip

```
$ pip install OBB_Maya
```

Git

```
$ git clone https://github.com/chrisdevito/OBB
$ cd OBB
$ python setup.py install
```

5.1.2 Installing the shelf

After installation through whatever means, just type in the python script editor:

```
import OBB.shelf
```

5.1.3 Installing numpy/scipy (Optional)

You do not have to install numpy/scipy to get this to work. It currently just doesn't allow you to use the from_hull method in the api.

Using Pip

Windows (Maya requires libraries compiled against MSVC2010:

```
$ pip install -i https://pypi.anaconda.org/carlkl/simple numpy
$ pip install -i https://pypi.anaconda.org/carlkl/simple scipy
```

Non-Windows:

```
$ pip install numpy
$ pip install scipy
```

5.2 Usage

Here's a simple api usage example with the 3 methods.

```
from maya import cmds
from OBB.api import OBB

if __name__ == '__main__':

    mesh = cmds.ls(selection=True)

    if len(mesh) == 0:
        raise RuntimeError("Nothing selected!")

    obbBoundBoxPnts = OBB.from_points(mesh)
    obbCube = cmds.polyCube(
        constructionHistory=False, name="pointMethod_GEO")[0]
    cmds.xform(obbCube, matrix=obbBoundBoxPnts.matrix)
    print(obbBoundBoxPnts.volume)

    obbBoundBoxTris = OBB.from_triangles(mesh)
    obbCube = cmds.polyCube(
        constructionHistory=False, name="triangleMethod_GEO")[0]
    cmds.xform(obbCube, matrix=obbBoundBoxTris.matrix)
    print(obbBoundBoxTris.volume)

    obbBoundBoxHull = OBB.from_hull(mesh)
    obbCube = cmds.polyCube(
        constructionHistory=False, name="hullMethod_GEO")[0]
    cmds.xform(obbCube, matrix=obbBoundBoxHull.matrix)
    print(obbBoundBoxHull.volume)
```

5.3 API

class OBB.api.**OBB**(*meshName=None*, *method=0*)

OBB Oriented Bounding Box Class.

Requires an input meshName.

build_from_covariance_matrix(*cvMatrix=None*)

Build eigen vectors from covariance matrix.

Parameters of lists) (*matrix(list*) – covariance matrix

Raises: None

Returns: None

build_from_hull()
Test oriented bounding box algorithm using convex hull points.

Raises: None

Returns: EigenVectors(OpenMaya.MVector) CenterPoint(OpenMaya.MVector) BoundingExtents(OpenMaya.MVector)

build_from_points()
Bounding box algorithm using vertex points.

Raises: None

Returns: EigenVectors(OpenMaya.MVector) CenterPoint(OpenMaya.MVector) BoundingExtents(OpenMaya.MVector)

build_from_triangles(*points=None*, *triangles=None*)
Test oriented bounding box algorithm using triangles.

Parameters

- **points(OpenMaya.MVectorArray)** – points to represent geometry.
- **triangles(OpenMaya.MIntArray)** – points to represent geometry.

Raises: None

Returns: EigenVectors(OpenMaya.MVector) CenterPoint(OpenMaya.MVector) BoundingExtents(OpenMaya.MVector)

center
Property center of the bounding box.

Returns: (OpenMaya.MVector)

create_bounding_box(*meshName='bounding_GEO'*)
Create the bounding box mesh.

Parameters **meshName(string)** – Name of created mesh.

Raises: None

Returns: (string) Cube Transform

depth
Property depth of the bounding box.

classmethod from_hull(*meshName=None*)
Bounding box algorithm using triangles points.

Raises: None

Returns: (OBB Instance)

classmethod from_points(*meshName=None*)
Bounding box algorithm using vertex points.

Raises: None

Returns: (OBB Instance)

classmethod **from_triangles**(*meshName=None*)
Bounding box algorithm using triangles points.

Raises: None

Returns: (OBB Instance)

getMFnMesh(*mesh*)
Gets the MFnMesh of the input mesh.

Parameters **(str)** (*mesh*) – string name of input mesh.

Raises: *RuntimeError* if not a mesh.

Returns: (OpenMaya.MFnMesh) MFnMesh mesh object.

getMatrix()
Gets the matrix representing the transformation of the bounding box.

Raises: None

Returns: (list of floats) Matrix

getPoints(*fnMesh*)
Get the points of each vertex.

Parameters **(OpenMaya.MFnMesh)** (*fnMesh*) – mesh function set.

Raises: None

Returns: (OpenMaya.MVectorArray) list of points.

getShape(*node*)
Gets the shape node from the input node.

Parameters **(str)** (*node*) – string name of input node

Raises: *RuntimeError* if no shape node.

Returns: (str) shape node name

getTriangles(*fnMesh*)
Get the triangles indices.

Parameters **(OpenMaya.MFnMesh)** (*fnMesh*) – mesh function set.

Raises: None

Returns: (OpenMaya.MIntArray) indices of triangles.

get_bounding_points()
Gets the bounding box points from the build.

Raises: None

Returns: (list of MVectors) Bounding box points.

height
Property height of the bounding box.

matrix
Property matrix of the bounding box.

volume
Property volume of bounding box.

width
Property width of the bounding box.

OBB.api.**timeit**(*method*)
Decorator to time function evaluation. Prints "method (args, kwargs) time.sec"

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

O

B

C

D

F

G

H

M

O

T

V

W