Rubik Cube

1.0

Generated by Doxygen 1.8.20

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ApplicationContext

RubikApp 6

InputListener

RubikApp 6

RBox 2

RBox::RFace 4

Rotator 4

RubikCube 7

TrayListener

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

3 Class Documentation

3.1 RBox Class Reference

Represents cube inside Rubiks cube.

`#include <box.h>`

Classes

- struct RFace

  This structure represents one face of the cube.

Public Member Functions

- RBox (SceneManager *mgr, SceneNode *parent, double x, double y, double z, double size)

  Constructs the cube (without initializing any faces)
- SceneNode * getNode ()

  Getter of the SceneNode pointer.
- void addFace (const RBOX_FACE face, const String &color)

  Adds new face to the cube.
- void fillFaces ()

  Sets or faces that were not initialized to black.
- RFace & getFace (const RBOX_FACE face)

  Returns reference to the face.
- const AxisAlignedBox & getBoundingboxFromFace (const RBOX_FACE face)

  Return bounding box of some face. Useful for Raycasting.
- void pitchFaces (bool alternative=false)

  Box face rotation. Required in order to make data inside RBox usable.
- void **yawFaces** (bool alternative=false)
- void **rollFaces** (bool alternative=false)

Static Public Member Functions

- static void generateAllMaterials ()

  Generates all materials required for the cube to adopt all avaible colors.

Private Member Functions

- ManualObject * **generateFace** (ManualObject *o, const String &color, const RBOX_FACE face)
- void **swapFaces** (const RBOX_FACE f1, const RBOX_FACE f2)

Static Private Member Functions

- static void **generateMaterial** (String name, Vector3 colors)

Private Attributes

- Real **size**
- Real **x**
- Real **y**
- Real **z**
- SceneNode * **node**
- SceneManager * **scnMgr**
- std::map< RBOX_FACE, RFace > **faces**

3.1.1 Detailed Description

Represents cube inside Rubiks cube.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 RBox() `RBox::RBox (`

```
SceneManager * mgr,
SceneNode * parent,
double x,
double y,
double z,
double size )
```

Constructs the cube (without initializing any faces)

Parameters

mgr	Pointer to SceneManager
parent	Parent of the cube
x	X coordinate of the cube
y	Y coordinate of the cube
z	Z coordinate of the cube
size	Size of the cube

3.1.3 Member Function Documentation

3.1.3.1 addFace() `void RBox::addFace ( const RBOX_FACE face, const String & color )`

Adds new face to the cube.

Parameters

face	Location of the face (FRONT, BACK, etc)
color	Color of the face

The documentation for this class was generated from the following files:

- box.h
- box.cpp

3.2 RBox::RFace Struct Reference

This structure represents one face of the cube.

`#include <box.h>`

Public Attributes

- MovableObject * plane
  Pointer to the "mesh".
- String color
  Color of the face.

3.2.1 Detailed Description

This structure represents one face of the cube.

The documentation for this struct was generated from the following file:

- box.h

3.3 Rotator Class Reference

This class rotates by 90 degrees group of Nodes around some axis over time.

`#include <rotator.h>`

Public Member Functions

- Rotator (std::vector< Node * > nodes, Vector3 axis)

  Constructor that takes Nodes to be rotated along axis.
- Rotator (const Rotator &copy)

  Simple copy contructor.
- bool update (Real dt)

  Function that should be called every frame to make rotation happen.

Private Attributes

- const Real MaxDuration = 0.4

  Time that each rotation takes.
- std::vector< Node * > **nodes**
- Real **duration** = 0
- Vector3 **axis**

3.3.1 Detailed Description

This class rotates by 90 degrees group of Nodes around some axis over time.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Rotator()

```
Rotator::Rotator (
        std::vector< Node * > nodes,
        Vector3 axis )
```

Constructor that takes Nodes to be rotated along axis.

Parameters

nodes	List of pointers of Nodes to be rotated
axis	Axis that nodes will be rotated along

3.3.3 Member Function Documentation

3.3.3.1 update()

```
bool Rotator::update (
        Real dt )
```

Function that should be called every frame to make rotation happen.

Parameters

dt	Time interval

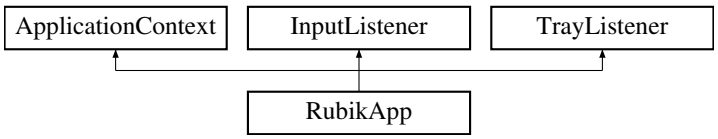
The documentation for this class was generated from the following files:

- rotator.h
- rotator.cpp

3.4 RubikApp Class Reference

This class inherits Ogre classes to show everything.

Inheritance diagram for RubikApp:



Public Member Functions

- RubikApp ()

  Constructs class.
- void **setup** () override
- bool **keyPressed** (const OgreBites::KeyboardEvent &evt) override
- bool **keyReleased** (const OgreBites::KeyboardEvent &evt) override
- bool **mouseMoved** (const OgreBites::MouseMotionEvent &evt) override
- bool **mousePressed** (const OgreBites::MouseButtonEvent &evt) override
- bool **mouseReleased** (const OgreBites::MouseButtonEvent &evt) override
- bool **mouseWheelRolled** (const OgreBites::MouseWheelEvent &evt) override
- bool **frameStarted** (const Ogre::FrameEvent &evt) override
- void **buttonHit** (OgreBites::Button ∗b) override

Private Member Functions

- bool isCubeRotating ()

  Returns true if rotation animation is ON.
- void updateRotation (float dt)

  Updates animation of the rotation.
- void rotateCube (const RubiksRotation rotation, bool alternative, bool saveMove=true)

  Rotates cube (internal structure) and starts the animation.

Private Attributes

- Ogre::SceneNode * **cam**
- Ogre::SceneNode * **lightNode**
- Ogre::SceneNode * **sky**
- OgreBites::CameraMan * **cameraMan**
- OgreBites::TrayManager * **mTrayMgr**
- std::optional< RubikCube > **cube**
- std::optional< Rotator > **rotator**
- std::stack< std::tuple< RubiksRotation, bool > > **rotations**
- bool **alt** = false

3.4.1 Detailed Description

This class inherits Ogre classes to show everything.

3.4.2 Member Function Documentation

3.4.2.1 rotateCube()

```
void RubikApp::rotateCube (
        const RubiksRotation rotation,
        bool alternative,
        bool saveMove = true )  [private]
```

Rotates cube (internal structure) and starts the animation.

Parameters

rotation	Rotation type
alternative	If alternative rotation should be applied
saveMove	if it is set to true then the move will be added to the undo stack

The documentation for this class was generated from the following file:

- main.cpp

3.5 RubikCube Class Reference

Class that represents Rubik Cube.

```
#include <rubik.h>
```

Public Member Functions

- RubikCube (SceneManager *scnMgr)

  Default contructor, it takes SceneManager and creates rubik cube at the origin of the scene.
- Rotator rotate (const RubiksRotation rotation, bool alternative=false)

  This function does two things. First of all it applies rotation to internal matrix of boxes. For example for L rotation, where x = 0, rotation works as follows.
- std::unique_ptr< RBox > & cubeAt (const int x, const int y, const int z)

  Gets reference to pointer that points to cube at the location (x, y, z), according to RubikData type.

Public Attributes

- const Vector3 **Origin** = Vector3(0, 0, 0)

Private Attributes

- const int BoxSize = 1

  "Radius" of each box
- std::unique_ptr< RubikData > data

  Pointer to rubik cube data.
- SceneNode * cube

  Scene object that represents rubik cube.

3.5.1 Detailed Description

Class that represents Rubik Cube.

3.5.2 Member Function Documentation

3.5.2.1 rotate()

```
Rotator RubikCube::rotate (
        const RubiksRotation rotation,
        bool alternative = false )
```

This function does two things. First of all it applies rotation to internal matrix of boxes. For example for L rotation, where x = 0, rotation works as follows.

```
(OZ - Z axis, OY - Y axis, Numbers are some faces):

      OZ ->                 <- OY
OY    1  2  3    ROT_L      7  4  1   OZ
|     4  5  6   -------->    8  5  2    |
v     7  8  9   (not alt)    9  6  3    v
```

Above operation is applied with respect to proper rotation type.

Parameters

rotation	Rotation to applied to the cube
alternative	If set to true, makes inverted rotation to the default one

Returns

Rotator object, that represents proper animation

The documentation for this class was generated from the following files:

- rubik.h
- rubik.cpp

Index