Arduino Cube

Generated by Doxygen 1.8.9.1

Wed Aug 19 2015 01:06:16

Contents

1 Hierarchical Index

1.1 Class Hierarchy

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

2 Class Index

2.1 Class List

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

3 File Index

3.1 File List

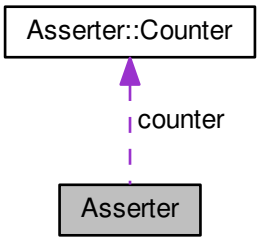
Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 Asserter Class Reference

```
#include <Asserter.h>
```

Collaboration diagram for Asserter:



Classes

- struct Counter

Static Public Member Functions

- static void reset ()
- static bool assert (bool assertion, const char ∗msg)
- static bool assertEqual (unsigned char a, unsigned char b, const char ∗msg)
- static bool assertNotEqual (unsigned char a, unsigned char b, const char ∗msg)

Static Public Attributes

- static Counter counter = {0, 0}

Static Private Attributes

- static const char ASSERT_PASSED_OUTPUT [ ] = "\e[32m(∗) passed: %s\e[0m\n"
- static const char ASSERT_FAILED_OUTPUT [ ] = "\e[31m(∗) failed: %s\e[0m\n"
- static const char ASSERT_EQUAL_FAILED_OUTPUT [ ] = "\e[31m(F) failed: %s (expected %d to be equal %d)\e[0m\n"
- static const char ASSERT_NOT_EQUAL_FAILED_OUTPUT [ ] = "\e[31m(F) failed: %s (expected %d to not be equal %d)\e[0m\n"

4.1.1 Detailed Description

Definition at line 7 of file Asserter.h.

4.1.2 Member Function Documentation

4.1.2.1 bool Asserter::assert (bool *assertion*, const char ∗ *msg*) [static]

Definition at line 16 of file Asserter.cpp.

4.1.2.2 bool Asserter::assertEqual (unsigned char *a*, unsigned char *b*, const char ∗ *msg*) [static]

Definition at line 29 of file Asserter.cpp.

4.1.2.3 bool Asserter::assertNotEqual (unsigned char *a,* unsigned char *b,* const char * *msg*) [static]

Definition at line 42 of file Asserter.cpp.

4.1.2.4 void Asserter::reset ( ) [static]

Definition at line 11 of file Asserter.cpp.

4.1.3 Member Data Documentation

4.1.3.1 const char Asserter::ASSERT_EQUAL_FAILED_OUTPUT = "\e[31m(F) failed: %s (expected %d to be equal %d)\e[0m\n" [static], [private]

Definition at line 11 of file Asserter.h.

4.1.3.2 const char Asserter::ASSERT_FAILED_OUTPUT = "\e[31m(*) failed: %s\e[0m\n" [static], [private]

Definition at line 10 of file Asserter.h.

4.1.3.3 const char Asserter::ASSERT_NOT_EQUAL_FAILED_OUTPUT = "\e[31m(F) failed: %s (expected %d to not be equal %d)\e[0m\n" [static], [private]

Definition at line 12 of file Asserter.h.

4.1.3.4 const char Asserter::ASSERT_PASSED_OUTPUT = "\e[32m(*) passed: %s\e[0m\n" [static], [private]

Definition at line 9 of file Asserter.h.

4.1.3.5 Asserter::Counter Asserter::counter = {0, 0} [static]

Definition at line 21 of file Asserter.h.

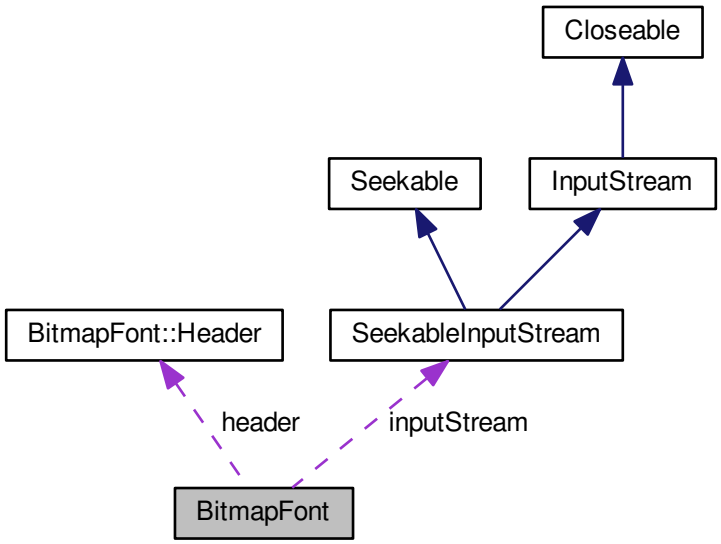
The documentation for this class was generated from the following files:

- Asserter.h
- Asserter.cpp

4.2 BitmapFont Class Reference

```
#include <BitmapFont.h>
```

Collaboration diagram for BitmapFont:



Classes

- struct Header

Public Member Functions

- BitmapFont (SeekableInputStream *inputStream)
- unsigned char getInfo ()
- unsigned char getCharacterWidth ()
- unsigned char getCharacterHeight ()
- unsigned char getSequenceCount ()
- unsigned char getGlyphLength ()
- virtual unsigned char readGlyphData (unsigned char *buf, char c)

Protected Member Functions

- virtual unsigned int getGlyphOffset (char c)

Protected Attributes

- Header header
- unsigned char glyphLength
- SeekableInputStream * inputStream
- unsigned int dataOffset

4.2.1 Detailed Description

Arduino Cube Library.

BitmapFont.h

The representation of a font.

Bitmap font is an array which represents the font glyph as a bitmap.

This font has fixed glyph size.

The first bytes specify the font's information and glyph sequences;

Header example:

```
unsigned char info;
unsigned char characterWidth;
unsigned char characterHeight;
unsigned char sequenceCount;
```

```
0x00, 0xww, 0xhh, 0xnn
 |     |     |     |___ Sequence count
 |     |     |_________ Character height (in bits)
 |     |_______________ Character width (in bits)
 |_____________________ Into
```

The next bytes, after the head, specify the sequences information, each sequence have 3 information:

```
unsigned char first;
unsigned char last;
unsigned char offset[msb];
unsigned char offset[lsb];
```

The sequence information are followed one by another. Example: Considering a font with 2 sequences, this could be the sequence bytes:

```
0x20, 0x22, 0x00, 0xff,
0x40, 0x43, 0x0d, 0xff,
```

which means we have a sequence that starts with the 0x20 char and goes to the 0x22, and the glyph data are stored at the 0x00ff offset. Another sequence starts with the 0x40 char and goes to the 0x43, and the glyph data are stored at the 0x0dff offset.

File structure

Header organization:

```
+-------------------------+
|      Font info          | \
+-------------------------+  \
|   Character width       |   \
+-------------------------+    } Header
|   Character height      |   /
+-------------------------+  /
|   Sequence count        | /
+-------------------------+
|   First character       | \
+-------------------------+  \
|   Last character        |   \
+-------------------------+    } Sequence #0
|   Offset MSB            |   /
+-------------------------+  /
```

```
|   Offset LSB            | /
+-------------------------+
|   First character       | \
+-------------------------+  \
|   Last character        |   \
+-------------------------+    } Sequence #n
|   Offset MSB            |   /
+-------------------------+  /
|   Offset LSB            | /
+-------------------------+
|                         | 0b00000000
|                         | 0b00000000
|                         | 0b00000000
|        FONT DATA        | 0b00000000
|     ARRAY OF BYTES      | 0b00000000
|                         | 0b00000000
|                         | 0b00000000
|                         | 0b00000000
+-------------------------+
```

Font data

Each character could have any multiple by 8 height. For characters which is 8 bits height, they are just made by bytes in sequence, as follows:

```
  L    L    L    L    L
  i    i    i    i    i
  n    n    n    n    n
  e    e    e    e    e
  0    1    2    3    4    ...
+-------------------------...
| b7 | b7 | b7 | b7 | b7 |
| b6 | b6 | b6 | b6 | b6 |
| b5 | b5 | b5 | b5 | b5 |
| b4 | b4 | b4 | b4 | b4 |
| b3 | b3 | b3 | b3 | b3 |
| b2 | b2 | b2 | b2 | b2 |
| b1 | b1 | b1 | b1 | b1 |
| b0 | b0 | b0 | b0 | b0 |
+-------------------------...
```

Or, this is a character with width equals 5 and height equals 8: (can be the "T" letter)

```
+-+-+-+-+-+
|1|1|1|1|1| -> the MSB
|1|1|1|1|1|
|0|0|1|1|0|
|0|0|1|1|0|
|0|0|1|1|0|
|0|0|1|1|0|
|0|0|1|1|0|
|0|0|0|0|0| -> the LSB
+-+-+-+-+-+
```

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 141 of file BitmapFont.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 BitmapFont::BitmapFont (SeekableInputStream * *inputStream*)

Public constructor.

Parameters

inputStream	The associated input stream.

Definition at line 16 of file BitmapFont.cpp.

4.2.3 Member Function Documentation

4.2.3.1 unsigned char BitmapFont::getCharacterHeight ( )

Gets the character height.

Returns

The height of a char.

Definition at line 34 of file BitmapFont.cpp.

4.2.3.2 unsigned char BitmapFont::getCharacterWidth ( )

Gets the character width.

Returns

The width of a char.

Definition at line 30 of file BitmapFont.cpp.

4.2.3.3 unsigned char BitmapFont::getGlyphLength ( )

Gets the glyph length.

Returns

The length of the glyph.

Definition at line 42 of file BitmapFont.cpp.

4.2.3.4 unsigned int BitmapFont::getGlyphOffset (char *c*) [protected], [virtual]

Gets the offset to the given character.

Parameters

c	The character to be used.

Returns

The offset.

Definition at line 55 of file BitmapFont.cpp.

4.2.3.5 unsigned char BitmapFont::getInfo ( )

Gets the font info.

Returns

Font info entry.

Definition at line 26 of file BitmapFont.cpp.

4.2.3.6 unsigned char BitmapFont::getSequenceCount ( )

Gets the sequence count.

Returns

The number of sequences.

Definition at line 38 of file BitmapFont.cpp.

4.2.3.7 unsigned char BitmapFont::readGlyphData (unsigned char * *buf*, char *c*) [virtual]

Gets the array of bytes representing the given character.

Parameters

buf	The buffer.
c	The character.

Returns

The number of bytes read.

Definition at line 46 of file BitmapFont.cpp.

4.2.4 Member Data Documentation

4.2.4.1 unsigned int BitmapFont::dataOffset [protected]

Data offset.

It is the point when the header ends.

Definition at line 169 of file BitmapFont.h.

4.2.4.2 unsigned char BitmapFont::glyphLength [protected]

Glyph length.

Definition at line 159 of file BitmapFont.h.

4.2.4.3 Header BitmapFont::header [protected]

Definition at line 154 of file BitmapFont.h.

4.2.4.4 SeekableInputStream* BitmapFont::inputStream [protected]

Input stream which font data comes from.

Definition at line 164 of file BitmapFont.h.

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

- BitmapFont.h
- BitmapFont.cpp

4.3 BitmapFontSpec Class Reference

```
#include <BitmapFontSpec.h>
```

Public Member Functions

- BitmapFontSpec ()
- void run ()
- void getInfoSpec ()
- void getCharacterWidthSpec ()
- void getCharacterHeightSpec ()
- void getSequenceCountSpec ()
- void getGlyphLengthSpec ()
- void readGlyphDataSpec ()
- void getGlyphOffsetSpec ()

4.3.1 Detailed Description

Definition at line 7 of file BitmapFontSpec.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 BitmapFontSpec::BitmapFontSpec ( )

Definition at line 11 of file BitmapFontSpec.cpp.

4.3.3 Member Function Documentation

4.3.3.1 void BitmapFontSpec::getCharacterHeightSpec ( )

Definition at line 32 of file BitmapFontSpec.cpp.

4.3.3.2 void BitmapFontSpec::getCharacterWidthSpec ( )

Definition at line 28 of file BitmapFontSpec.cpp.

4.3.3.3 void BitmapFontSpec::getGlyphLengthSpec ( )

Definition at line 40 of file BitmapFontSpec.cpp.

4.3.3.4 void BitmapFontSpec::getGlyphOffsetSpec ( )

Definition at line 48 of file BitmapFontSpec.cpp.

4.3.3.5 void BitmapFontSpec::getInfoSpec ( )

Definition at line 24 of file BitmapFontSpec.cpp.

4.3.3.6 void BitmapFontSpec::getSequenceCountSpec ( )

Definition at line 36 of file BitmapFontSpec.cpp.

4.3.3.7 void BitmapFontSpec::readGlyphDataSpec ( )

Definition at line 44 of file BitmapFontSpec.cpp.

4.3.3.8 void BitmapFontSpec::run ( )

Definition at line 14 of file BitmapFontSpec.cpp.

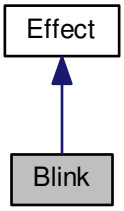
The documentation for this class was generated from the following files:

- BitmapFontSpec.h
- BitmapFontSpec.cpp

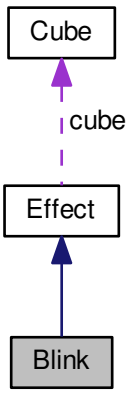
4.4 Blink Class Reference

`#include <Blink.h>`

Inheritance diagram for Blink:



Collaboration diagram for Blink:



Public Member Functions

- Blink (Cube *cube, unsigned int iterations, unsigned int iterationDelay)
- virtual void run ()

Additional Inherited Members

4.4.1 Detailed Description

Definition at line 10 of file Blink.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Blink::Blink (Cube * *cube,* unsigned int *iterations,* unsigned int *iterationDelay*)

Definition at line 11 of file Blink.cpp.

4.4.3 Member Function Documentation

4.4.3.1 void Blink::run ( ) [virtual]

Reimplemented from Effect.

Definition at line 15 of file Blink.cpp.

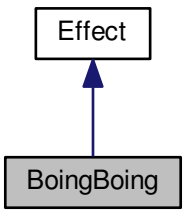
The documentation for this class was generated from the following files:

- Blink.h
- Blink.cpp

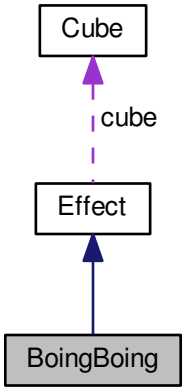
4.5 BoingBoing Class Reference

`#include <BoingBoing.h>`

Inheritance diagram for BoingBoing:



Collaboration diagram for BoingBoing:



Public Member Functions

- BoingBoing (Cube *cube, unsigned int iterations, unsigned int iterationDelay)
- virtual void run ()

Additional Inherited Members

4.5.1 Detailed Description

Definition at line 10 of file BoingBoing.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 BoingBoing::BoingBoing (Cube * *cube,* unsigned int *iterations,* unsigned int *iterationDelay*)

Definition at line 9 of file BoingBoing.cpp.

4.5.3 Member Function Documentation

4.5.3.1 void BoingBoing::run ( ) [virtual]

Reimplemented from Effect.

Definition at line 13 of file BoingBoing.cpp.

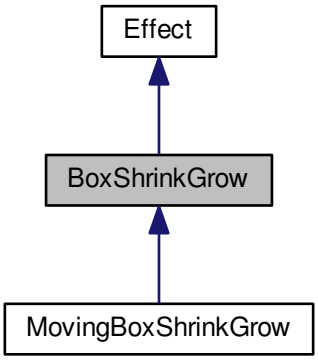
The documentation for this class was generated from the following files:

- BoingBoing.h
- BoingBoing.cpp

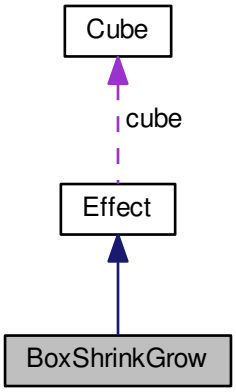
4.6 BoxShrinkGrow Class Reference

```
#include <BoxShrinkGrow.h>
```

Inheritance diagram for BoxShrinkGrow:



Collaboration diagram for BoxShrinkGrow:



Public Types

- enum BoxType { WIREFRAME = 0x00, FILLED = 0x01, WALL = 0x02 }

Public Member Functions

- BoxShrinkGrow (Cube *cube, unsigned int iterations, unsigned int iterationDelay, BoxType boxType)
- virtual void run ()
- void shrink ()
- void grow ()
- void drawFrame (char size)
- virtual void draw (char size)

Protected Attributes

- BoxType boxType

Additional Inherited Members

4.6.1 Detailed Description

Definition at line 10 of file BoxShrinkGrow.h.

4.6.2 Member Enumeration Documentation

4.6.2.1 enum BoxShrinkGrow::BoxType

Enumerator

WIREFRAME

FILLED

WALL

Definition at line 14 of file BoxShrinkGrow.h.

4.6.3 Constructor & Destructor Documentation

4.6.3.1 BoxShrinkGrow::BoxShrinkGrow (Cube * *cube*, unsigned int *iterations*, unsigned int *iterationDelay*, BoxType *boxType*)

Definition at line 10 of file BoxShrinkGrow.cpp.

4.6.4 Member Function Documentation

4.6.4.1 void BoxShrinkGrow::draw (char *size*) [virtual]

Reimplemented in MovingBoxShrinkGrow.

Definition at line 44 of file BoxShrinkGrow.cpp.

4.6.4.2 void BoxShrinkGrow::drawFrame (char *size*)

Definition at line 38 of file BoxShrinkGrow.cpp.

4.6.4.3 void BoxShrinkGrow::grow ( )

Definition at line 31 of file BoxShrinkGrow.cpp.

4.6.4.4 void BoxShrinkGrow::run ( ) [virtual]

Reimplemented from Effect.

Reimplemented in MovingBoxShrinkGrow.

Definition at line 14 of file BoxShrinkGrow.cpp.

4.6.4.5 void BoxShrinkGrow::shrink ( )

Definition at line 24 of file BoxShrinkGrow.cpp.

4.6.5 Member Data Documentation

4.6.5.1 BoxType BoxShrinkGrow::boxType [protected]

Definition at line 34 of file BoxShrinkGrow.h.

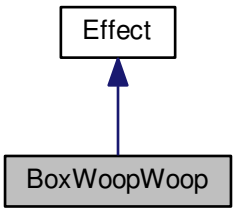
The documentation for this class was generated from the following files:

- BoxShrinkGrow.h
- BoxShrinkGrow.cpp

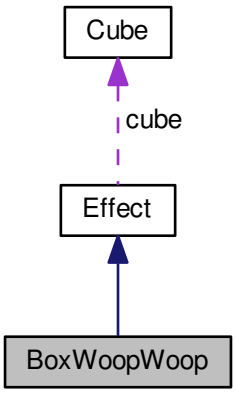
4.7 BoxWoopWoop Class Reference

`#include <BoxWoopWoop.h>`

Inheritance diagram for BoxWoopWoop:



Collaboration diagram for BoxWoopWoop:



Public Member Functions

- BoxWoopWoop (Cube *cube, unsigned int iterations, unsigned int iterationDelay)

- virtual void run ()

Additional Inherited Members

4.7.1 Detailed Description

Definition at line 10 of file BoxWoopWoop.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 BoxWoopWoop::BoxWoopWoop (Cube * *cube*, unsigned int *iterations*, unsigned int *iterationDelay*)

Definition at line 9 of file BoxWoopWoop.cpp.

4.7.3 Member Function Documentation

4.7.3.1 void BoxWoopWoop::run ( ) [virtual]

Reimplemented from Effect.

Definition at line 13 of file BoxWoopWoop.cpp.

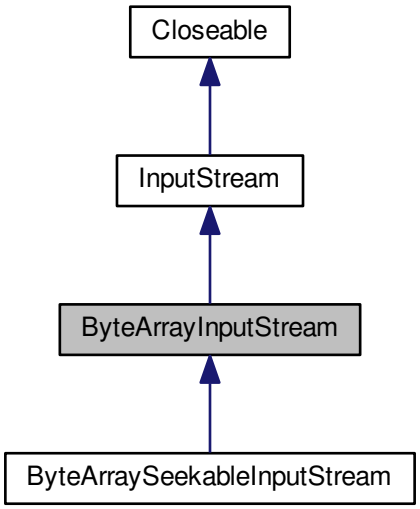
The documentation for this class was generated from the following files:

- BoxWoopWoop.h
- BoxWoopWoop.cpp

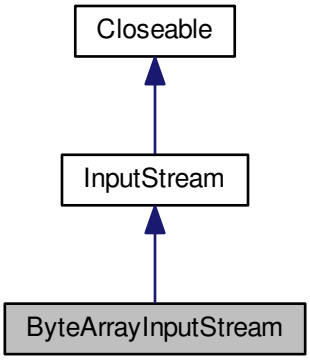
4.8 ByteArrayInputStream Class Reference

`#include <ByteArrayInputStream.h>`

Inheritance diagram for ByteArrayInputStream:



Collaboration diagram for ByteArrayInputStream:



Public Member Functions

- ByteArrayInputStream (unsigned char *buf, unsigned int count)
- virtual int available ()
- virtual void mark ()
- virtual bool markSupported ()
- virtual int read ()
- virtual void reset ()

Protected Attributes

- unsigned char * buf
- unsigned int count
- unsigned int pos
- unsigned int markpos

4.8.1 Detailed Description

Arduino IO.

ByteArrayInputStream

A ByteArrayInputStream contains an internal buffer that contains bytes that may be read from the stream.

Definition at line 15 of file ByteArrayInputStream.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 ByteArrayInputStream::ByteArrayInputStream (unsigned char * *buf*, unsigned int *count*)

Definition at line 15 of file ByteArrayInputStream.cpp.

4.8.3 Member Function Documentation

4.8.3.1 int ByteArrayInputStream::available ( ) [virtual]

Returns the number of bytes that can be read(or skipped over) from this input stream without blocking by the next caller of a method for this input stream.

NOTE: This implementation return 1 or 0. It is because the size of the array is unsigned int, and this method returns a signed int, which means there is no way to return the difference between the current position (can be 0) and the size of the array without possible overflow.

Returns

Reimplemented from InputStream.

Definition at line 22 of file ByteArrayInputStream.cpp.

4.8.3.2 void ByteArrayInputStream::mark ( ) [virtual]

Marks the current position in this input stream.

Reimplemented from InputStream.

Definition at line 29 of file ByteArrayInputStream.cpp.

4.8.3.3 bool ByteArrayInputStream::markSupported ( ) [virtual]

Tests if this input stream supports the mark and reset methods.

Returns

Reimplemented from InputStream.

Definition at line 33 of file ByteArrayInputStream.cpp.

4.8.3.4 int ByteArrayInputStream::read ( ) [virtual]

Reads the next unsigned char of data from the input stream.

Returns

Implements InputStream.

Definition at line 37 of file ByteArrayInputStream.cpp.

4.8.3.5 void ByteArrayInputStream::reset ( ) [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

Reimplemented from InputStream.

Definition at line 44 of file ByteArrayInputStream.cpp.

4.8.4 Member Data Documentation

4.8.4.1 unsigned char* ByteArrayInputStream::buf [protected]

Definition at line 21 of file ByteArrayInputStream.h.

4.8.4.2 unsigned int ByteArrayInputStream::count [protected]

Definition at line 26 of file ByteArrayInputStream.h.

4.8.4.3 unsigned int ByteArrayInputStream::markpos [protected]

Definition at line 36 of file ByteArrayInputStream.h.

4.8.4.4 unsigned int ByteArrayInputStream::pos [protected]

Definition at line 31 of file ByteArrayInputStream.h.

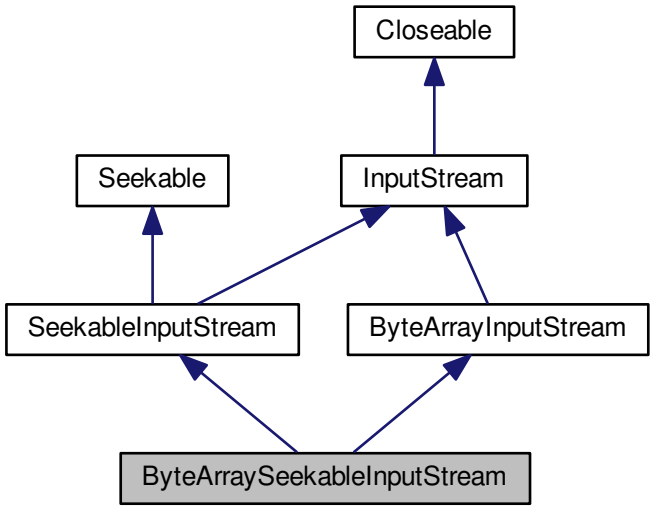
The documentation for this class was generated from the following files:

- ByteArrayInputStream.h
- ByteArrayInputStream.cpp

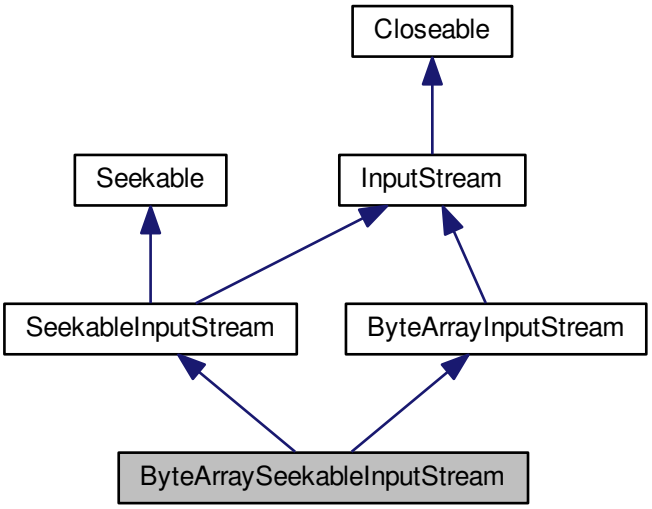
4.9 ByteArraySeekableInputStream Class Reference

`#include <ByteArraySeekableInputStream.h>`

Inheritance diagram for ByteArraySeekableInputStream:



Collaboration diagram for ByteArraySeekableInputStream:



Public Member Functions

- ByteArraySeekableInputStream (unsigned char *buf, unsigned int count)
- virtual void seek (unsigned int pos)

Additional Inherited Members

4.9.1 Detailed Description

Arduino IO.

ByteArraySeekableInputStream

A ByteArraySeekableInputStream obtains input bytes from a resource in a file system that implements Seekable↩ InputStream interface.

Definition at line 16 of file ByteArraySeekableInputStream.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 ByteArraySeekableInputStream::ByteArraySeekableInputStream (unsigned char * *buf*, unsigned int *count*)

Definition at line 15 of file ByteArraySeekableInputStream.cpp.

4.9.3 Member Function Documentation

4.9.3.1 void ByteArraySeekableInputStream::seek (unsigned int *pos*) [virtual]

Implements Seekable.

Definition at line 20 of file ByteArraySeekableInputStream.cpp.

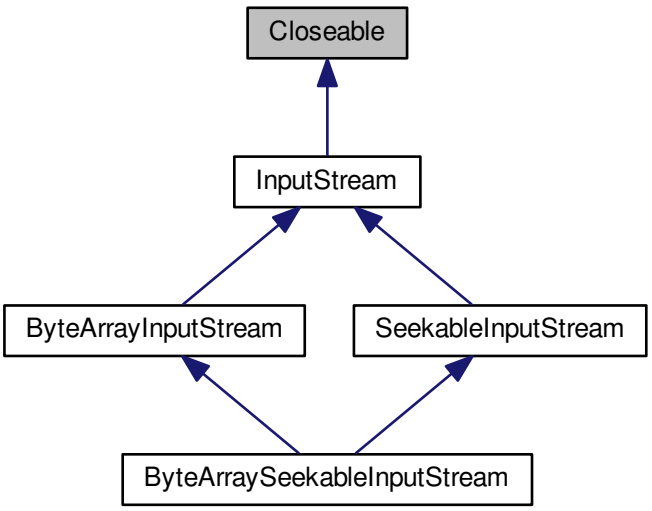
The documentation for this class was generated from the following files:

- ByteArraySeekableInputStream.h
- ByteArraySeekableInputStream.cpp

4.10 Closeable Class Reference

`#include <Closeable.h>`

Inheritance diagram for Closeable:



Public Member Functions

- virtual void close ()=0

4.10.1 Detailed Description

Arduino IO.

Closeable

A Closeable is a source or destination of data that can be closed.

Definition at line 12 of file Closeable.h.

4.10.2 Member Function Documentation

4.10.2.1 virtual void Closeable::close ( ) [pure virtual]

Implemented in InputStream.

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

- Closeable.h

4.11 Asserter::Counter Struct Reference

`#include <Asserter.h>`

Public Attributes

- unsigned int error
- unsigned int success

4.11.1 Detailed Description

Definition at line 16 of file Asserter.h.

4.11.2 Member Data Documentation

4.11.2.1 unsigned int Asserter::Counter::error

Definition at line 17 of file Asserter.h.

4.11.2.2 unsigned int Asserter::Counter::success

Definition at line 18 of file Asserter.h.

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

- Asserter.h

4.12 Cube Class Reference

`#include <Cube.h>`

Public Types

- enum Buffer { FRONT_BUFFER = 0x00, BACK_BUFFER = 0x01 }

Public Member Functions

- Cube ()
- void selectBuffer (Buffer buffer)
- void useBackBuffer ()
- void useFrontBuffer ()
- bool isInRange (Point *p) const
- void fitInRange (Point *p)
- void fill ()
- void clear ()
- void fill (unsigned char pattern)
- void writeVoxel (Point *p, Voxel v)
- void writeVoxel (unsigned char x, unsigned char y, unsigned char z, unsigned char state)
- void turnVoxelOn (Point *p)
- void readVoxel (Point *p, Voxel *v)
- void turnVoxelOff (Point *p)
- void invertVoxel (Point *p)
- void turnPlaneZOff (unsigned char z)

- void turnPlaneZOn (unsigned char z)
- void writePlaneZ (unsigned char z, Voxel v)
- void turnPlaneYOff (unsigned char y)
- void turnPlaneYOn (unsigned char y)
- void writePlaneY (unsigned char y, Voxel v)
- void turnPlaneXOff (unsigned char x)
- void turnPlaneXOn (unsigned char x)
- void writePlaneX (unsigned char x, Voxel v)
- void writePlane (Axis axis, unsigned char pos, Voxel v)
- void mirrorX ()
- void mirrorY ()
- void mirrorZ ()
- void line (Point *from, Point *to)
- void filledBox (Point *from, Point *to)
- void wallBox (Point *from, Point *to)
- void wireframeBox (Point *from, Point *to)
- void shift (Axis axis, Direction direction)
- void shiftOnX (Direction direction)
- void shiftOnY (Direction direction)
- void shiftOnZ (Direction direction)
- void writeSubCube (Point *p, Voxel v, unsigned char size)
- void swapBuffers ()

Public Attributes

- unsigned char * frontBuffer
- unsigned char * backBuffer

Static Public Attributes

- static const unsigned char SIZE = CUBE_SIZE
- static const unsigned char BYTE_SIZE = CUBE_BYTE_SIZE

Private Attributes

- unsigned char ** bufferToWrite

Static Private Attributes

- static unsigned char buffer0 [CUBE_SIZE][CUBE_SIZE] = {}
- static unsigned char buffer1 [CUBE_SIZE][CUBE_SIZE] = {}

4.12.1 Detailed Description

Definition at line 16 of file Cube.h.

4.12.2 Member Enumeration Documentation

4.12.2.1 enum Cube::Buffer

Enumerator

FRONT_BUFFER

BACK_BUFFER

Definition at line 29 of file Cube.h.

4.12.3 Constructor & Destructor Documentation

4.12.3.1 Cube::Cube ( ) [inline]

Definition at line 34 of file Cube.h.

4.12.4 Member Function Documentation

4.12.4.1 void Cube::clear ( ) [inline]

Cleans the current buffer.

Definition at line 79 of file Cube.h.

4.12.4.2 void Cube::fill ( ) [inline]

Fills the current buffer.

Definition at line 72 of file Cube.h.

4.12.4.3 void Cube::fill (unsigned char *pattern*)

Fill a value into all 64 bytes of the cube buffer.

Mostly used for clearing. fill(0x00) or setting all on. fill(0xff)

Definition at line 49 of file Cube.cpp.

4.12.4.4 void Cube::filledBox (Point * *from,* Point * *to*)

Draws a filled cube.

Definition at line 207 of file Cube.cpp.

4.12.4.5 void Cube::fitInRange (Point * *p*)

Fits the point in cube range.

Definition at line 43 of file Cube.cpp.

4.12.4.6 void Cube::invertVoxel (Point * *p*)

Switch volex state.

Definition at line 74 of file Cube.cpp.

4.12.4.7 bool Cube::isInRange (Point * *p*) const

Validates if we the p Point is inside the cube.

Definition at line 36 of file Cube.cpp.

4.12.4.8 void Cube::line (Point * *from,* Point * *to*)

Draw a 3d line.

Definition at line 149 of file Cube.cpp.

4.12.4.9 void Cube::mirrorX ( )

Flip the cube 180 degrees along the x axis.

Definition at line 174 of file Cube.cpp.

4.12.4.10 void Cube::mirrorY ( )

Flip the cube 180 degrees along the y axis.

Definition at line 185 of file Cube.cpp.

4.12.4.11 void Cube::mirrorZ ( )

Flip the cube 180 degrees along the z axis.

Definition at line 196 of file Cube.cpp.

4.12.4.12 void Cube::readVoxel (Point * *p*, Voxel * *v*)

Get Voxel.

Parameters

p	3D Point pointer

Definition at line 59 of file Cube.cpp.

4.12.4.13 void Cube::selectBuffer (Buffer *buffer*)

Set if will use back or front buffer to write to.

Definition at line 17 of file Cube.cpp.

4.12.4.14 void Cube::shift (Axis *axis*, Direction *direction*)

Definition at line 316 of file Cube.cpp.

4.12.4.15 void Cube::shiftOnX (Direction *direction*)

Definition at line 269 of file Cube.cpp.

4.12.4.16 void Cube::shiftOnY (Direction *direction*)

Definition at line 285 of file Cube.cpp.

4.12.4.17 void Cube::shiftOnZ (Direction *direction*)

Definition at line 301 of file Cube.cpp.

4.12.4.18 void Cube::swapBuffers ( )

Swap the buffers.

Current buffer becomes backed buffer and vice-versa.

Definition at line 27 of file Cube.cpp.

4.12.4.19 void Cube::turnPlaneXOff (unsigned char *x*)

Turn off plane x.

Definition at line 114 of file Cube.cpp.

4.12.4.20 void Cube::turnPlaneXOn (unsigned char *x*)

Turn on plane x.

Definition at line 119 of file Cube.cpp.

4.12.4.21 void Cube::turnPlaneYOff (unsigned char *y*)

Turn off plane y.

Definition at line 97 of file Cube.cpp.

4.12.4.22 void Cube::turnPlaneYOn (unsigned char *y*)

Turn on plane y.

Definition at line 102 of file Cube.cpp.

4.12.4.23 void Cube::turnPlaneZOff (unsigned char *z*)

Turn off plane z.

Definition at line 80 of file Cube.cpp.

4.12.4.24 void Cube::turnPlaneZOn (unsigned char *z*)

Turn on plane z.

Definition at line 85 of file Cube.cpp.

4.12.4.25 void Cube::turnVoxelOff (Point * *p*)

Set voxel to OFF.

Parameters

p	3D Point pointer

Definition at line 69 of file Cube.cpp.

4.12.4.26 void Cube::turnVoxelOn (Point * *p*)

Set voxel to ON.

Parameters

p	3D Point pointer

Definition at line 64 of file Cube.cpp.

4.12.4.27 void Cube::useBackBuffer ( ) [inline]

Set the buffer to use back buffer.

Definition at line 48 of file Cube.h.

4.12.4.28 void Cube::useFrontBuffer ( ) [inline]

Set the buffer to use front buffer.

Definition at line 55 of file Cube.h.

4.12.4.29 void Cube::wallBox (Point * *from,* Point * *to*)

Definition at line 229 of file Cube.cpp.

4.12.4.30 void Cube::wireframeBox (Point * *from,* Point * *to*)

Definition at line 246 of file Cube.cpp.

4.12.4.31 void Cube::writePlane (Axis *axis,* unsigned char *pos,* Voxel *v*)

Turn the p plane.

Definition at line 135 of file Cube.cpp.

4.12.4.32 void Cube::writePlaneX (unsigned char *x*, Voxel *v*)

Write plane x.

Definition at line 124 of file Cube.cpp.

4.12.4.33 void Cube::writePlaneY (unsigned char *y*, Voxel *v*)

Write plane y.

Definition at line 107 of file Cube.cpp.

4.12.4.34 void Cube::writePlaneZ (unsigned char *z*, Voxel *v*)

Write plane z.

Definition at line 90 of file Cube.cpp.

4.12.4.35 void Cube::writeSubCube (Point * *p*, Voxel *v*, unsigned char *size*)

Definition at line 219 of file Cube.cpp.

4.12.4.36 void Cube::writeVoxel (Point * *p*, Voxel *v*) [inline]

Writes the v Voxel to the cube at p Point.

If b == true writes to the backed buffer instead.

Parameters

**p*	3D Point pointer
v	Voxel
b	Write it to the buffer/cube

Definition at line 99 of file Cube.h.

4.12.4.37 void Cube::writeVoxel (unsigned char *x*, unsigned char *y*, unsigned char *z*, unsigned char *state*)

Definition at line 53 of file Cube.cpp.

4.12.5 Member Data Documentation

4.12.5.1 unsigned char* Cube::backBuffer

Definition at line 27 of file Cube.h.

4.12.5.2 unsigned char Cube::buffer0 = {} [static], [private]

Definition at line 18 of file Cube.h.

4.12.5.3 unsigned char Cube::buffer1 = {} [static], [private]

Definition at line 19 of file Cube.h.

4.12.5.4 unsigned char** Cube::bufferToWrite [private]

Definition at line 20 of file Cube.h.

4.12.5.5 const unsigned char Cube::BYTE_SIZE = CUBE_BYTE_SIZE [static]

Definition at line 25 of file Cube.h.

4.12.5.6 unsigned char* Cube::frontBuffer

Definition at line 26 of file Cube.h.

4.12.5.7 const unsigned char Cube::SIZE = CUBE_SIZE [static]

Definition at line 24 of file Cube.h.

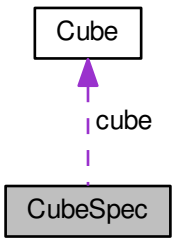
The documentation for this class was generated from the following files:

- Cube.h
- Cube.cpp

4.13 CubeSpec Class Reference

`#include <CubeSpec.h>`

Collaboration diagram for CubeSpec:



Public Member Functions

- CubeSpec (Cube *cube)
- void run ()
- void isInRangeSpec ()
- void writeVoxelSpec ()
- void invertVoxelSpec ()
- void writePlaneZSpec ()
- void writePlaneYSpec ()
- void writePlaneXSpec ()
- void flipByteSpec ()
- void mirrorXSpec ()
- void mirrorYSpec ()
- void mirrorZSpec ()
- void filledBoxSpec ()
- void lineSpec ()
- void shiftOnXSpec ()
- void shiftOnYSpec ()
- void shiftOnZSpec ()

Public Attributes

- Cube * cube

4.13.1 Detailed Description

Definition at line 9 of file CubeSpec.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 CubeSpec::CubeSpec (Cube * *cube*)

Definition at line 9 of file CubeSpec.cpp.

4.13.3 Member Function Documentation

4.13.3.1 void CubeSpec::filledBoxSpec ( )

Definition at line 155 of file CubeSpec.cpp.

4.13.3.2 void CubeSpec::flipByteSpec ( )

Definition at line 110 of file CubeSpec.cpp.

4.13.3.3 void CubeSpec::invertVoxelSpec ( )

Definition at line 56 of file CubeSpec.cpp.

4.13.3.4 void CubeSpec::isInRangeSpec ( )

Definition at line 30 of file CubeSpec.cpp.

4.13.3.5 void CubeSpec::lineSpec ( )

Definition at line 181 of file CubeSpec.cpp.

4.13.3.6 void CubeSpec::mirrorXSpec ( )

Definition at line 116 of file CubeSpec.cpp.

4.13.3.7 void CubeSpec::mirrorYSpec ( )

Definition at line 130 of file CubeSpec.cpp.

4.13.3.8 void CubeSpec::mirrorZSpec ( )

Definition at line 139 of file CubeSpec.cpp.

4.13.3.9 void CubeSpec::run ( )

Definition at line 12 of file CubeSpec.cpp.

4.13.3.10 void CubeSpec::shiftOnXSpec ( )

Definition at line 196 of file CubeSpec.cpp.

4.13.3.11 void CubeSpec::shiftOnYSpec ( )

Definition at line 210 of file CubeSpec.cpp.

4.13.3.12 void CubeSpec::shiftOnZSpec ( )

Definition at line 222 of file CubeSpec.cpp.

4.13.3.13 void CubeSpec::writePlaneXSpec ( )

Definition at line 95 of file CubeSpec.cpp.

4.13.3.14 void CubeSpec::writePlaneYSpec ( )

Definition at line 82 of file CubeSpec.cpp.

4.13.3.15 void CubeSpec::writePlaneZSpec ( )

Definition at line 69 of file CubeSpec.cpp.

4.13.3.16 void CubeSpec::writeVoxelSpec ( )

Definition at line 40 of file CubeSpec.cpp.

4.13.4 Member Data Documentation

4.13.4.1 Cube* CubeSpec::cube

Definition at line 13 of file CubeSpec.h.

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

- CubeSpec.h
- CubeSpec.cpp

4.14 Dumper Class Reference

`#include <Dumper.h>`

Static Public Member Functions

- static void dumpPoint (Point *point)
- static void dumpCube (Cube *cube)

4.14.1 Detailed Description

Definition at line 11 of file Dumper.h.

4.14.2 Member Function Documentation

4.14.2.1 void Dumper::dumpCube (Cube * *cube*) [static]

Definition at line 13 of file Dumper.cpp.

4.14.2.2 void Dumper::dumpPoint (Point * *point*) [static]

Definition at line 9 of file Dumper.cpp.

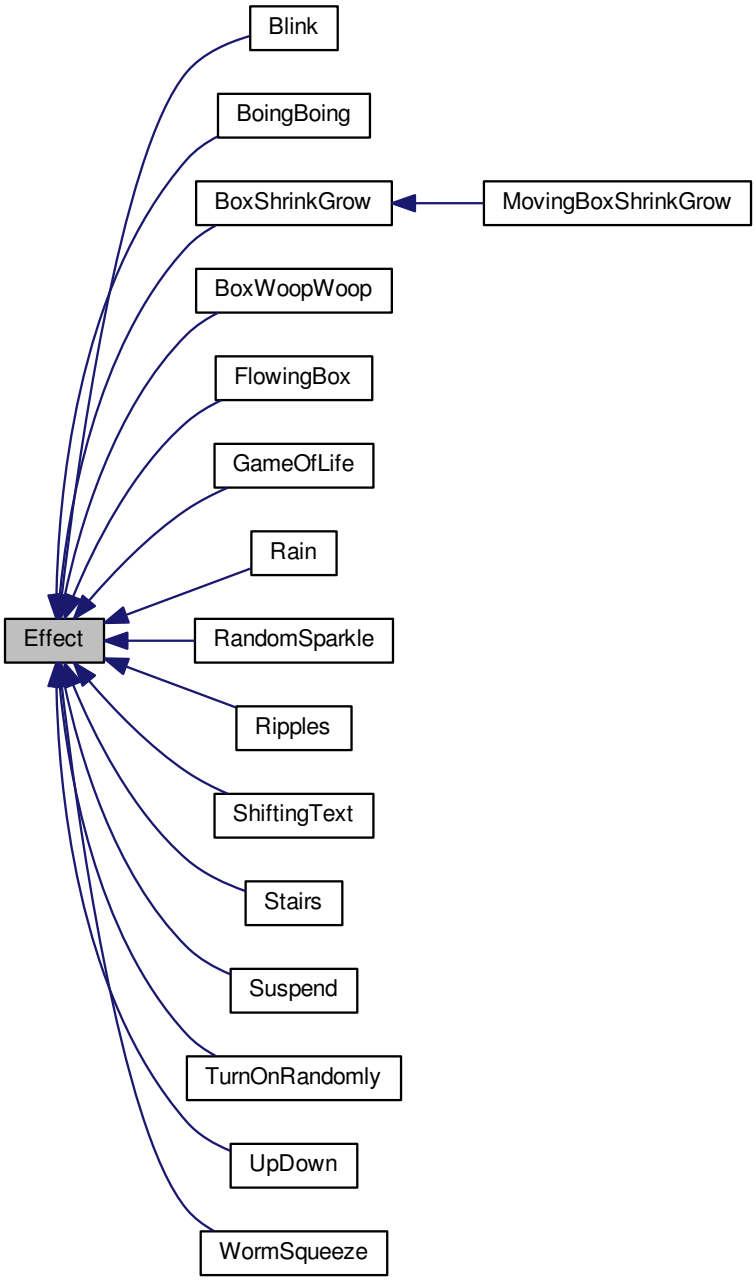
The documentation for this class was generated from the following files:

- Dumper.h
- Dumper.cpp

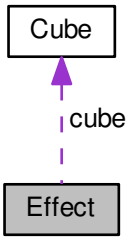
4.15 Effect Class Reference

`#include <Effect.h>`

Inheritance diagram for Effect:



Collaboration diagram for Effect:



Public Member Functions

- Effect (Cube ∗cube, unsigned int iterations, unsigned int iterationDelay)
- virtual void run ()
- void sendVoxel (Point ∗origin, Direction direction, unsigned int stepDelay)

Public Attributes

- unsigned int iterations
- unsigned int iterationDelay
- Cube ∗ cube

4.15.1 Detailed Description

Definition at line 11 of file Effect.h.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 Effect::Effect (Cube ∗ *cube,* unsigned int *iterations,* unsigned int *iterationDelay*)

Definition at line 11 of file Effect.cpp.

4.15.3 Member Function Documentation

4.15.3.1 void Effect::run ( ) [virtual]

Reimplemented in ShiftingText, UpDown, GameOfLife, BoxShrinkGrow, MovingBoxShrinkGrow, Rain, Turn↩ OnRandomly, Blink, BoingBoing, BoxWoopWoop, FlowingBox, RandomSparkle, Ripples, Stairs, Suspend, and WormSqueeze.

Definition at line 15 of file Effect.cpp.

4.15.3.2 void Effect::sendVoxel (Point ∗ *origin,* Direction *direction,* unsigned int *stepDelay*)

Definition at line 18 of file Effect.cpp.

4.15.4 Member Data Documentation

4.15.4.1 Cube* Effect::cube

Definition at line 17 of file Effect.h.

4.15.4.2 unsigned int Effect::iterationDelay

Definition at line 16 of file Effect.h.

4.15.4.3 unsigned int Effect::iterations

Definition at line 15 of file Effect.h.

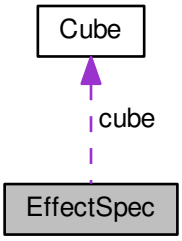
The documentation for this class was generated from the following files:

- Effect.h
- Effect.cpp

4.16 EffectSpec Class Reference

`#include <EffectSpec.h>`

Collaboration diagram for EffectSpec:



Public Member Functions

- EffectSpec (Cube *cube)
- void run ()
- void selfSpec ()
- void rainSpec ()
- void blinkSpec ()
- void boingBoingSpec ()
- void boxShrinkGrowSpec ()
- void boxWoopWoopSpec ()
- void flowingBoxSpec ()
- void gameOfLifeSpec ()
- void movingBoxShrinkGrowSpec ()
- void randomSparkleSpec ()
- void ripplesSpec ()
- void stairsSpec ()

- void suspendSpec ()
- void upDownSpec ()
- void wormSqueezeSpec ()
- void turnOnRandomlySpec ()
- void shiftingTextSpec ()

Public Attributes

- Cube * cube

4.16.1 Detailed Description

Definition at line 9 of file EffectSpec.h.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 EffectSpec::EffectSpec (Cube * *cube*)

Definition at line 28 of file EffectSpec.cpp.

4.16.3 Member Function Documentation

4.16.3.1 void EffectSpec::blinkSpec ( )

Definition at line 59 of file EffectSpec.cpp.

4.16.3.2 void EffectSpec::boingBoingSpec ( )

Definition at line 81 of file EffectSpec.cpp.

4.16.3.3 void EffectSpec::boxShrinkGrowSpec ( )

Definition at line 87 of file EffectSpec.cpp.

4.16.3.4 void EffectSpec::boxWoopWoopSpec ( )

Definition at line 93 of file EffectSpec.cpp.

4.16.3.5 void EffectSpec::flowingBoxSpec ( )

Definition at line 99 of file EffectSpec.cpp.

4.16.3.6 void EffectSpec::gameOfLifeSpec ( )

Definition at line 105 of file EffectSpec.cpp.

4.16.3.7 void EffectSpec::movingBoxShrinkGrowSpec ( )

Definition at line 111 of file EffectSpec.cpp.

4.16.3.8 void EffectSpec::rainSpec ( )

Definition at line 65 of file EffectSpec.cpp.

4.16.3.9 void EffectSpec::randomSparkleSpec ( )

Definition at line 117 of file EffectSpec.cpp.

4.16.3.10 void EffectSpec::ripplesSpec ( )

Definition at line 123 of file EffectSpec.cpp.

4.16.3.11 void EffectSpec::run ( )

Definition at line 31 of file EffectSpec.cpp.

4.16.3.12 void EffectSpec::selfSpec ( )

Definition at line 51 of file EffectSpec.cpp.

4.16.3.13 void EffectSpec::shiftingTextSpec ( )

Definition at line 159 of file EffectSpec.cpp.

4.16.3.14 void EffectSpec::stairsSpec ( )

Definition at line 129 of file EffectSpec.cpp.

4.16.3.15 void EffectSpec::suspendSpec ( )

Definition at line 135 of file EffectSpec.cpp.

4.16.3.16 void EffectSpec::turnOnRandomlySpec ( )

Definition at line 153 of file EffectSpec.cpp.

4.16.3.17 void EffectSpec::upDownSpec ( )

Definition at line 141 of file EffectSpec.cpp.

4.16.3.18 void EffectSpec::wormSqueezeSpec ( )

Definition at line 147 of file EffectSpec.cpp.

4.16.4 Member Data Documentation

4.16.4.1 Cube* EffectSpec::cube

Definition at line 13 of file EffectSpec.h.

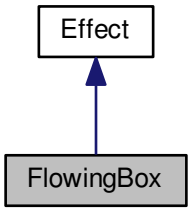
The documentation for this class was generated from the following files:

- EffectSpec.h
- EffectSpec.cpp

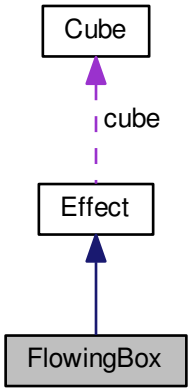
4.17 FlowingBox Class Reference

```
#include <FlowingBox.h>
```

Inheritance diagram for FlowingBox:



Collaboration diagram for FlowingBox:



Public Member Functions

- FlowingBox (Cube *cube, unsigned int iterations, unsigned int iterationDelay)
- virtual void run ()

Additional Inherited Members

4.17.1 Detailed Description

Definition at line 10 of file FlowingBox.h.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 FlowingBox::FlowingBox (Cube * *cube*, unsigned int *iterations*, unsigned int *iterationDelay*)

Definition at line 9 of file FlowingBox.cpp.

4.17.3 Member Function Documentation

4.17.3.1 void FlowingBox::run ( ) [virtual]

Reimplemented from Effect.

Definition at line 13 of file FlowingBox.cpp.

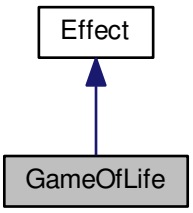
The documentation for this class was generated from the following files:

- FlowingBox.h
- FlowingBox.cpp

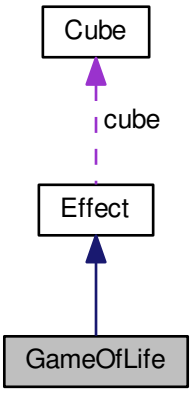
4.18 GameOfLife Class Reference

`#include <GameOfLife.h>`

Inheritance diagram for GameOfLife:



Collaboration diagram for GameOfLife:



Public Member Functions

- GameOfLife (Cube *cube, unsigned int iterations, unsigned int iterationDelay, unsigned char firstGenerationSize)
- virtual void run ()
- void genesis ()
- unsigned char getNeighbors (Point *p)
- void nextGeneration ()
- bool hasChanges ()

Static Public Attributes

- static const unsigned char LONELY_DEATH = 1
- static const unsigned char CROWDED_DEATH = 4
- static const unsigned char CREATE_MIN = 3
- static const unsigned char CREATE_MAX = 3

Private Attributes

- unsigned char firstGenerationSize

Additional Inherited Members

4.18.1 Detailed Description

Definition at line 10 of file GameOfLife.h.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 GameOfLife::GameOfLife (Cube * *cube*, unsigned int *iterations*, unsigned int *iterationDelay*, unsigned char *firstGenerationSize*)

Definition at line 10 of file GameOfLife.cpp.

4.18.3 Member Function Documentation

4.18.3.1 void GameOfLife::genesis ( )

Definition at line 26 of file GameOfLife.cpp.

4.18.3.2 unsigned char GameOfLife::getNeighbors (Point * *p*)

Definition at line 59 of file GameOfLife.cpp.

4.18.3.3 bool GameOfLife::hasChanges ( )

Definition at line 79 of file GameOfLife.cpp.

4.18.3.4 void GameOfLife::nextGeneration ( )

Definition at line 36 of file GameOfLife.cpp.

4.18.3.5 void GameOfLife::run ( ) [virtual]

Reimplemented from Effect.

Definition at line 14 of file GameOfLife.cpp.

4.18.4 Member Data Documentation

4.18.4.1 const unsigned char GameOfLife::CREATE_MAX = 3 [static]

Definition at line 19 of file GameOfLife.h.

4.18.4.2 const unsigned char GameOfLife::CREATE_MIN = 3 [static]

Definition at line 18 of file GameOfLife.h.

4.18.4.3 const unsigned char GameOfLife::CROWDED_DEATH = 4 [static]

Definition at line 17 of file GameOfLife.h.

4.18.4.4 unsigned char GameOfLife::firstGenerationSize [private]

Definition at line 12 of file GameOfLife.h.

4.18.4.5 const unsigned char GameOfLife::LONELY_DEATH = 1 [static]

Definition at line 16 of file GameOfLife.h.

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

- GameOfLife.h
- GameOfLife.cpp

4.19 BitmapFont::Header Struct Reference

`#include <BitmapFont.h>`

Public Attributes

- unsigned char info
- unsigned char characterWidth
- unsigned char characterHeight
- unsigned char sequenceCount

4.19.1 Detailed Description

Font header.

Definition at line 148 of file BitmapFont.h.

4.19.2 Member Data Documentation

4.19.2.1 unsigned char BitmapFont::Header::characterHeight

Definition at line 151 of file BitmapFont.h.

4.19.2.2 unsigned char BitmapFont::Header::characterWidth

Definition at line 150 of file BitmapFont.h.

4.19.2.3 unsigned char BitmapFont::Header::info

Definition at line 149 of file BitmapFont.h.

4.19.2.4 unsigned char BitmapFont::Header::sequenceCount

Definition at line 152 of file BitmapFont.h.

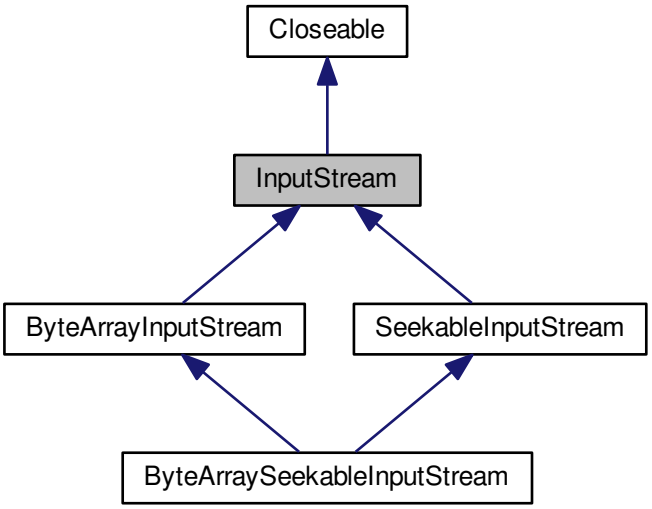
The documentation for this struct was generated from the following file:

- BitmapFont.h

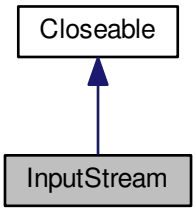
4.20 InputStream Class Reference

`#include <InputStream.h>`

Inheritance diagram for InputStream:



Collaboration diagram for InputStream:



Public Member Functions

- virtual int available ()

- virtual void close ()
- virtual void mark ()
- virtual bool markSupported ()
- virtual int read ()=0
- virtual int read (unsigned char *b, int len)
- virtual int read (unsigned char *b, int off, int len)
- virtual void reset ()
- virtual unsigned int skip (unsigned int n)

4.20.1 Detailed Description

Arduino IO.

InputStream

This abstract class is the superclass of all classes representing an input stream of bytes.

Applications that need to define a subclass of InputStream must always provide a method that returns the next unsigned char of input.

Definition at line 18 of file InputStream.h.

4.20.2 Member Function Documentation

4.20.2.1 int InputStream::available ( ) [virtual]

Returns the number of bytes that can be read(or skipped over) from this input stream without blocking by the next caller of a method for this input stream.

Reimplemented in ByteArrayInputStream.

Definition at line 18 of file InputStream.cpp.

4.20.2.2 void InputStream::close ( ) [virtual]

Closes this input stream and releases any system resources associated with the stream.

Implements Closeable.

Definition at line 22 of file InputStream.cpp.

4.20.2.3 void InputStream::mark ( ) [virtual]

Marks the current position in this input stream.

Reimplemented in ByteArrayInputStream.

Definition at line 25 of file InputStream.cpp.

4.20.2.4 bool InputStream::markSupported ( ) [virtual]

Tests if this input stream supports the mark and reset methods.

Reimplemented in ByteArrayInputStream.

Definition at line 28 of file InputStream.cpp.

4.20.2.5 virtual int InputStream::read ( ) [pure virtual]

Reads the next unsigned char of data from the input stream.

Implemented in ByteArrayInputStream.

4.20.2.6 int InputStream::read (unsigned char * *b*, int *len*) [virtual]

Reads some number of bytes from the input stream and stores them into the buffer array b.

Definition at line 32 of file InputStream.cpp.

4.20.2.7 int InputStream::read (unsigned char * *b*, int *off*, int *len*) [virtual]

Writes len of bytes into the stream.

Parameters

b	
off	
len	

Returns

Definition at line 36 of file InputStream.cpp.

4.20.2.8 void InputStream::reset ( ) [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

Reimplemented in ByteArrayInputStream.

Definition at line 56 of file InputStream.cpp.

4.20.2.9 unsigned int InputStream::skip (unsigned int *n*) [virtual]

Skips over and discards n bytes of data from this input stream.

Definition at line 59 of file InputStream.cpp.

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

- InputStream.h
- InputStream.cpp

4.21 UpDown::Location Struct Reference

Public Attributes

- unsigned char position: 4
- unsigned char destination: 4

4.21.1 Detailed Description

Definition at line 12 of file UpDown.h.

4.21.2 Member Data Documentation

4.21.2.1 unsigned char UpDown::Location::destination

Definition at line 14 of file UpDown.h.

4.21.2.2 unsigned char UpDown::Location::position

Definition at line 13 of file UpDown.h.

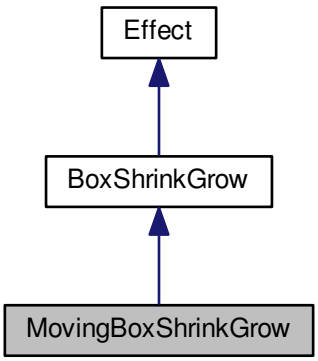
The documentation for this struct was generated from the following file:

- UpDown.h

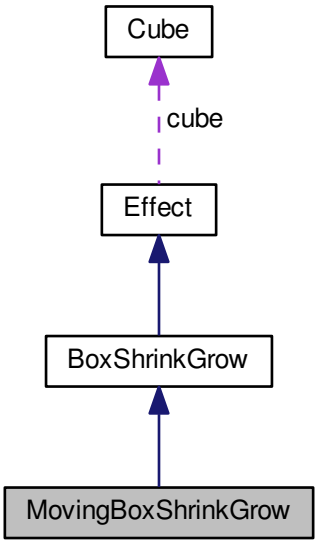
4.22 MovingBoxShrinkGrow Class Reference

`#include <MovingBoxShrinkGrow.h>`

Inheritance diagram for MovingBoxShrinkGrow:



Collaboration diagram for MovingBoxShrinkGrow:



Public Member Functions

- MovingBoxShrinkGrow (Cube *cube, unsigned int iterations, unsigned int iterationDelay, BoxType boxType)
- virtual void run ()
- void draw (char size)

Static Public Attributes

- static const unsigned char MAX_DIFF_MOVEMENTS = 0x03

Private Attributes

- unsigned char state

Additional Inherited Members

4.22.1 Detailed Description

Definition at line 10 of file MovingBoxShrinkGrow.h.

4.22.2 Constructor & Destructor Documentation

4.22.2.1 MovingBoxShrinkGrow::MovingBoxShrinkGrow (Cube * *cube*, unsigned int *iterations*, unsigned int *iterationDelay*, BoxType *boxType*)

Definition at line 11 of file MovingBoxShrinkGrow.cpp.

4.22.3 Member Function Documentation

4.22.3.1 void MovingBoxShrinkGrow::draw (char *size*) [virtual]

Reimplemented from BoxShrinkGrow.

Definition at line 26 of file MovingBoxShrinkGrow.cpp.

4.22.3.2 void MovingBoxShrinkGrow::run ( ) [virtual]

Reimplemented from BoxShrinkGrow.

Definition at line 15 of file MovingBoxShrinkGrow.cpp.

4.22.4 Member Data Documentation

4.22.4.1 const unsigned char MovingBoxShrinkGrow::MAX_DIFF_MOVEMENTS = 0x03 [static]

Definition at line 16 of file MovingBoxShrinkGrow.h.

4.22.4.2 unsigned char MovingBoxShrinkGrow::state [private]

Definition at line 12 of file MovingBoxShrinkGrow.h.

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

- MovingBoxShrinkGrow.h
- MovingBoxShrinkGrow.cpp

4.23 Point Class Reference

`#include <Point.h>`

Public Member Functions

- Point ()
- Point (unsigned char x, unsigned char y, unsigned char z)
- void randomize (unsigned char maxRange)
- unsigned char distanceOnXTo (Point *p)
- unsigned char distanceOnYTo (Point *p)
- unsigned char distanceOnZTo (Point *p)
- float distance2DTo (Point *p)
- float distance3DTo (Point *p)

Public Attributes

- unsigned char x
- unsigned char y
- unsigned char z

Private Member Functions

- void init (unsigned char x, unsigned char y, unsigned char z)

4.23.1 Detailed Description

Definition at line 7 of file Point.h.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 Point::Point ( )

Definition at line 12 of file Point.cpp.

4.23.2.2 Point::Point (unsigned char *x*, unsigned char *y*, unsigned char *z*)

Definition at line 16 of file Point.cpp.

4.23.3 Member Function Documentation

4.23.3.1 float Point::distance2DTo (Point * *p*)

Definition at line 38 of file Point.cpp.

4.23.3.2 float Point::distance3DTo (Point * *p*)

Definition at line 44 of file Point.cpp.

4.23.3.3 unsigned char Point::distanceOnXTo (Point * *p*)

Definition at line 26 of file Point.cpp.

4.23.3.4 unsigned char Point::distanceOnYTo (Point * *p*)

Definition at line 30 of file Point.cpp.

4.23.3.5 unsigned char Point::distanceOnZTo (Point * *p*)

Definition at line 34 of file Point.cpp.

4.23.3.6 void Point::init (unsigned char *x*, unsigned char *y*, unsigned char *z*) [inline], [private]

Definition at line 33 of file Point.h.

4.23.3.7 void Point::randomize (unsigned char *maxRange*)

Definition at line 20 of file Point.cpp.

4.23.4 Member Data Documentation

4.23.4.1 unsigned char Point::x

Definition at line 11 of file Point.h.

4.23.4.2 unsigned char Point::y

Definition at line 12 of file Point.h.

4.23.4.3 unsigned char Point::z

Definition at line 13 of file Point.h.

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

- Point.h
- Point.cpp

4.24 PointSpec Class Reference

`#include <PointSpec.h>`

Public Member Functions

- PointSpec ()
- void run ()
- void randomizeSpec ()
- void distanceOnXToSpec ()
- void distanceOnYToSpec ()
- void distanceOnZToSpec ()
- void distance2DToSpec ()
- void distance3DSpec ()

4.24.1 Detailed Description

Definition at line 7 of file PointSpec.h.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 PointSpec::PointSpec ( )

Definition at line 6 of file PointSpec.cpp.

4.24.3 Member Function Documentation

4.24.3.1 void PointSpec::distance2DToSpec ( )

Definition at line 56 of file PointSpec.cpp.

4.24.3.2 void PointSpec::distance3DSpec ( )

Definition at line 66 of file PointSpec.cpp.

4.24.3.3 void PointSpec::distanceOnXToSpec ( )

Definition at line 26 of file PointSpec.cpp.

4.24.3.4 void PointSpec::distanceOnYToSpec ( )

Definition at line 36 of file PointSpec.cpp.

4.24.3.5 void PointSpec::distanceOnZToSpec ( )

Definition at line 46 of file PointSpec.cpp.

4.24.3.6 void PointSpec::randomizeSpec ( )

Definition at line 18 of file PointSpec.cpp.

4.24.3.7 void PointSpec::run ( )

Definition at line 9 of file PointSpec.cpp.

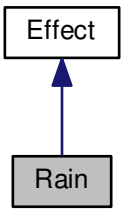
The documentation for this class was generated from the following files:

- PointSpec.h
- PointSpec.cpp

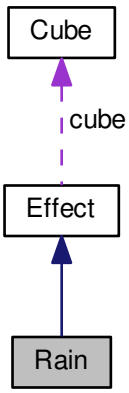
4.25 Rain Class Reference

`#include <Rain.h>`

Inheritance diagram for Rain:



Collaboration diagram for Rain:



Public Member Functions

- Rain (Cube *cube, unsigned int iterations, unsigned int iterationDelay, unsigned char minDrops, unsigned char maxDrops)
- virtual void run ()

Private Attributes

- unsigned char minDrops
- unsigned char maxDrops

Additional Inherited Members

4.25.1 Detailed Description

Definition at line 10 of file Rain.h.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 Rain::Rain (Cube * *cube,* unsigned int *iterations,* unsigned int *iterationDelay,* unsigned char *minDrops,* unsigned char *maxDrops*)

Definition at line 11 of file Rain.cpp.

4.25.3 Member Function Documentation

4.25.3.1 void Rain::run ( ) [virtual]

Reimplemented from Effect.

Definition at line 15 of file Rain.cpp.

4.25.4 Member Data Documentation

4.25.4.1 unsigned char Rain::maxDrops [private]

Definition at line 13 of file Rain.h.

4.25.4.2 unsigned char Rain::minDrops [private]

Definition at line 12 of file Rain.h.

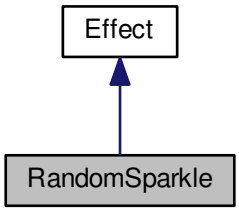
The documentation for this class was generated from the following files:

- Rain.h
- Rain.cpp

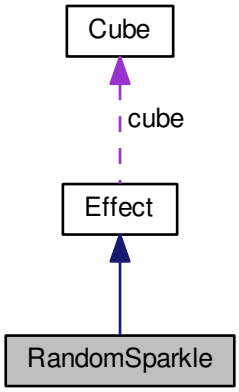
4.26 RandomSparkle Class Reference

```
#include <RandomSparkle.h>
```

Inheritance diagram for RandomSparkle:



Collaboration diagram for RandomSparkle:



Public Member Functions

- RandomSparkle (Cube *cube, unsigned int iterations, unsigned int iterationDelay)
- virtual void run ()

Additional Inherited Members

4.26.1 Detailed Description

Definition at line 10 of file RandomSparkle.h.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 RandomSparkle::RandomSparkle (Cube * *cube*, unsigned int *iterations*, unsigned int *iterationDelay*)

Definition at line 9 of file RandomSparkle.cpp.

4.26.3 Member Function Documentation

4.26.3.1 void RandomSparkle::run ( ) [virtual]

Reimplemented from Effect.

Definition at line 13 of file RandomSparkle.cpp.

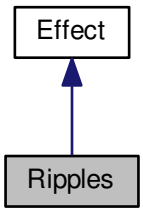
The documentation for this class was generated from the following files:

- RandomSparkle.h
- RandomSparkle.cpp

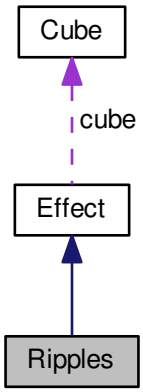
4.27 Ripples Class Reference

`#include <Ripples.h>`

Inheritance diagram for Ripples:



Collaboration diagram for Ripples:



Public Member Functions

- Ripples (Cube *cube, unsigned int iterations, unsigned int iterationDelay)
- virtual void run ()

Additional Inherited Members

4.27.1 Detailed Description

Definition at line 10 of file Ripples.h.

4.27.2 Constructor & Destructor Documentation

4.27.2.1 Ripples::Ripples (Cube * *cube*, unsigned int *iterations*, unsigned int *iterationDelay*)

Definition at line 9 of file Ripples.cpp.

4.27.3 Member Function Documentation

4.27.3.1 void Ripples::run ( ) [virtual]

Reimplemented from Effect.

Definition at line 13 of file Ripples.cpp.

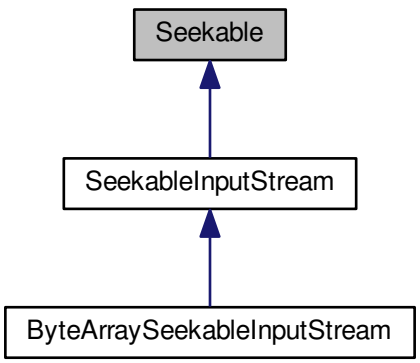
The documentation for this class was generated from the following files:

- Ripples.h
- Ripples.cpp

4.28 Seekable Class Reference

`#include <Seekable.h>`

Inheritance diagram for Seekable:



Public Member Functions

- virtual void seek (unsigned int pos)=0

4.28.1 Detailed Description

Arduino IO.

Seekable

Definition at line 10 of file Seekable.h.

4.28.2 Member Function Documentation

4.28.2.1 virtual void Seekable::seek (unsigned int *pos*) [pure virtual]

Implemented in ByteArraySeekableInputStream.

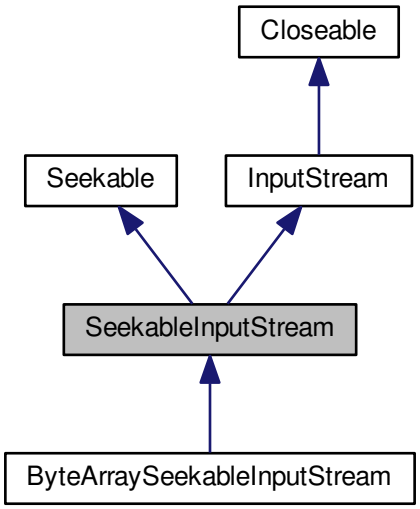
The documentation for this class was generated from the following file:

- Seekable.h

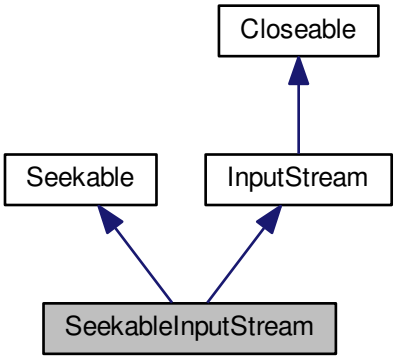
4.29 SeekableInputStream Class Reference

`#include <SeekableInputStream.h>`

Inheritance diagram for SeekableInputStream:



Collaboration diagram for SeekableInputStream:



Additional Inherited Members

4.29.1 Detailed Description

Arduino IO.

SeekableInputStream

Definition at line 13 of file SeekableInputStream.h.

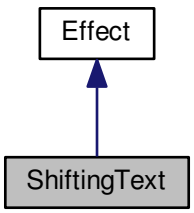
The documentation for this class was generated from the following file:

- SeekableInputStream.h

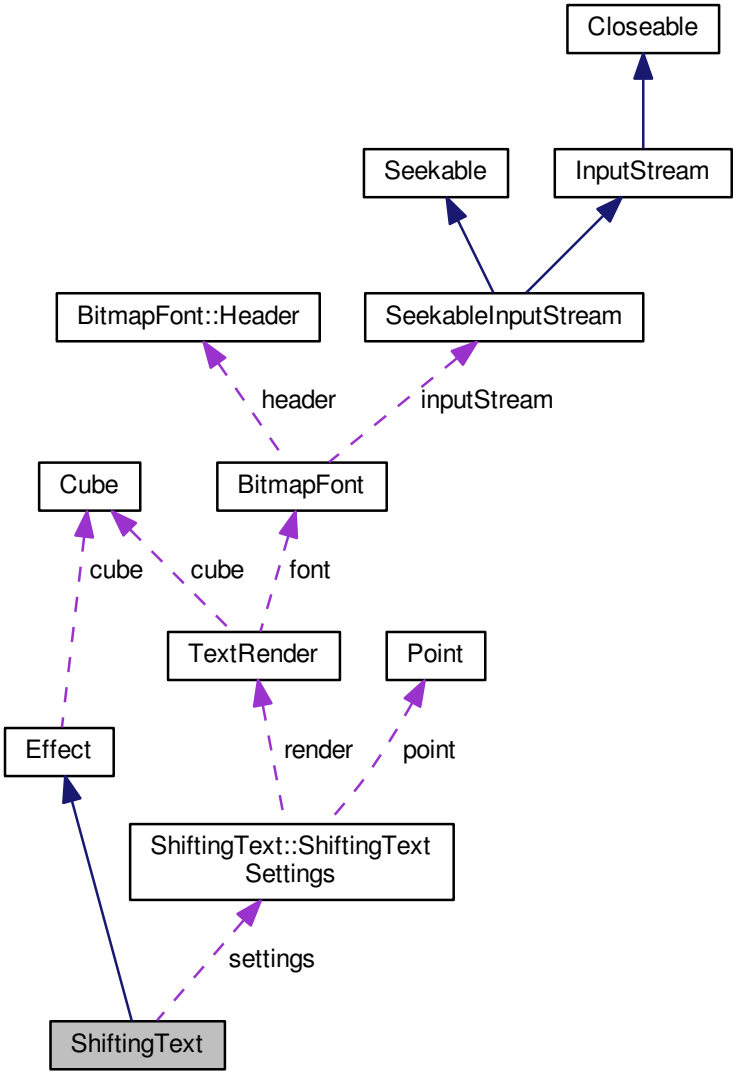
4.30 ShiftingText Class Reference

`#include <ShiftingText.h>`

Inheritance diagram for ShiftingText:



Collaboration diagram for ShiftingText:



Classes

- struct ShiftingTextSettings

Public Member Functions

- ShiftingText (Cube *cube, unsigned int iterations, unsigned int iterationDelay, ShiftingTextSettings *settings)
- virtual void run ()
- void displayCharacter (const char c)
- void shiftCharacter ()

Public Attributes

- ShiftingTextSettings * settings

4.30.1 Detailed Description

Definition at line 11 of file ShiftingText.h.

4.30.2 Constructor & Destructor Documentation

4.30.2.1 ShiftingText::ShiftingText (Cube * *cube*, unsigned int *iterations*, unsigned int *iterationDelay*, ShiftingTextSettings * *settings*)

Definition at line 11 of file ShiftingText.cpp.

4.30.3 Member Function Documentation

4.30.3.1 void ShiftingText::displayCharacter (const char *c*)

Definition at line 28 of file ShiftingText.cpp.

4.30.3.2 void ShiftingText::run ( ) [virtual]

Reimplemented from Effect.

Definition at line 15 of file ShiftingText.cpp.

4.30.3.3 void ShiftingText::shiftCharacter ( )

Definition at line 33 of file ShiftingText.cpp.

4.30.4 Member Data Documentation

4.30.4.1 ShiftingTextSettings* ShiftingText::settings

Definition at line 23 of file ShiftingText.h.

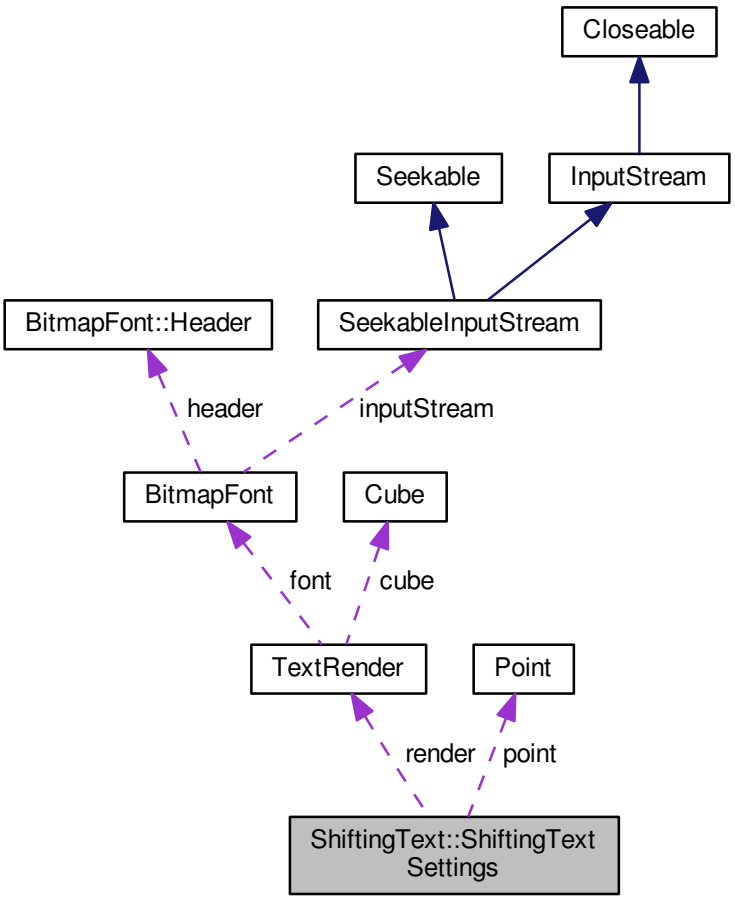
The documentation for this class was generated from the following files:

- ShiftingText.h
- ShiftingText.cpp

4.31 ShiftingText::ShiftingTextSettings Struct Reference

`#include <ShiftingText.h>`

Collaboration diagram for ShiftingText::ShiftingTextSettings:



Public Attributes

- TextRender * render
- const char * text
- unsigned char charDepth
- unsigned char orientation
- Point * point

4.31.1 Detailed Description

Definition at line 15 of file ShiftingText.h.

4.31.2 Member Data Documentation

4.31.2.1 unsigned char ShiftingText::ShiftingTextSettings::charDepth

Definition at line 18 of file ShiftingText.h.

4.31.2.2 unsigned char ShiftingText::ShiftingTextSettings::orientation

Definition at line 19 of file ShiftingText.h.

4.31.2.3 Point* ShiftingText::ShiftingTextSettings::point

Definition at line 20 of file ShiftingText.h.

4.31.2.4 TextRender* ShiftingText::ShiftingTextSettings::render

Definition at line 16 of file ShiftingText.h.

4.31.2.5 const char* ShiftingText::ShiftingTextSettings::text

Definition at line 17 of file ShiftingText.h.

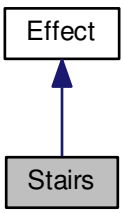
The documentation for this struct was generated from the following file:

- ShiftingText.h

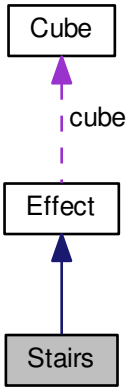
4.32 Stairs Class Reference

`#include <Stairs.h>`

Inheritance diagram for Stairs:



Collaboration diagram for Stairs:



Public Member Functions

- Stairs (Cube *cube, unsigned int iterations, unsigned int iterationDelay)
- virtual void run ()

Additional Inherited Members

4.32.1 Detailed Description

Definition at line 10 of file Stairs.h.

4.32.2 Constructor & Destructor Documentation

4.32.2.1 Stairs::Stairs (Cube * *cube*, unsigned int *iterations*, unsigned int *iterationDelay*)

Definition at line 9 of file Stairs.cpp.

4.32.3 Member Function Documentation

4.32.3.1 void Stairs::run ( ) [virtual]

Reimplemented from Effect.

Definition at line 13 of file Stairs.cpp.

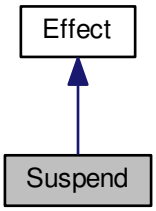
The documentation for this class was generated from the following files:

- Stairs.h
- Stairs.cpp

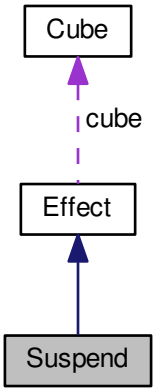
4.33 Suspend Class Reference

`#include <Suspend.h>`

Inheritance diagram for Suspend:



Collaboration diagram for Suspend:



Public Member Functions

- Suspend (Cube *cube, unsigned int iterations, unsigned int iterationDelay)
- virtual void run ()

Additional Inherited Members

4.33.1 Detailed Description

Definition at line 10 of file Suspend.h.

4.33.2 Constructor & Destructor Documentation

4.33.2.1 Suspend::Suspend (Cube * *cube*, unsigned int *iterations*, unsigned int *iterationDelay*)

Definition at line 9 of file Suspend.cpp.

4.33.3 Member Function Documentation

4.33.3.1 void Suspend::run ( ) [virtual]

Reimplemented from Effect.

Definition at line 13 of file Suspend.cpp.

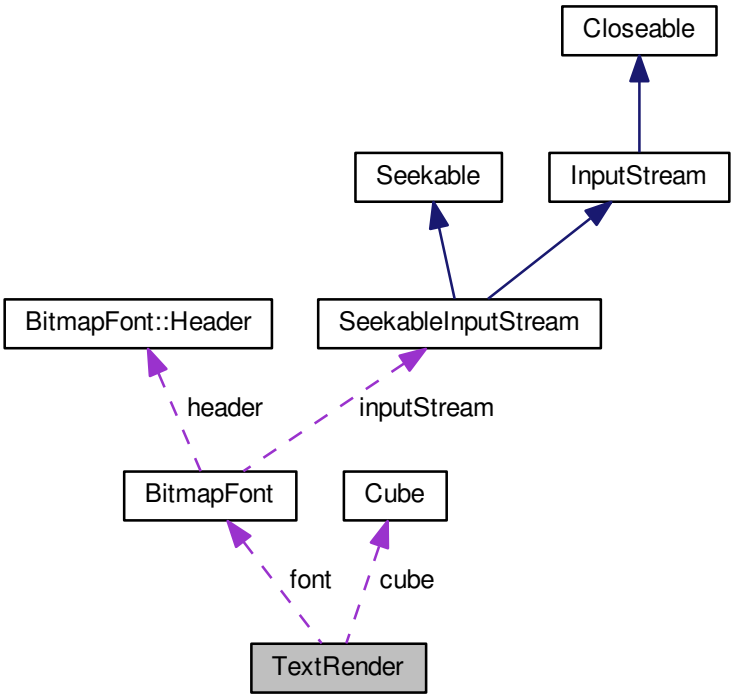
The documentation for this class was generated from the following files:

- Suspend.h
- Suspend.cpp

4.34 TextRender Class Reference

`#include <TextRender.h>`

Collaboration diagram for TextRender:



Public Types

- enum TextOrientation {
  XYZ, XZY, YXZ, YZX,
  ZXY, ZYX }

Public Member Functions

- TextRender (Cube *cube, BitmapFont *font)

- void printChar (Point *p, TextOrientation orientation, unsigned char depth, const char c)
- void adjustCoordinates (Point *p, TextOrientation orientation, unsigned char **x, unsigned char **y, unsigned char **z)

Private Attributes

- Cube * cube
- BitmapFont * font

4.34.1 Detailed Description

Arduino Cube Library.

TextRender.h

The functions to draw text in a glcd plane.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 18 of file TextRender.h.

4.34.2 Member Enumeration Documentation

4.34.2.1 enum TextRender::TextOrientation

Enumerator

XYZ

XZY

YXZ

YZX

ZXY

ZYX

Definition at line 32 of file TextRender.h.

4.34.3 Constructor & Destructor Documentation

4.34.3.1 TextRender::TextRender (Cube * *cube*, BitmapFont * *font*)

Public constructor.

Parameters

cube	The cube instance
font	The font to be used.

Definition at line 10 of file TextRender.cpp.

4.34.4 Member Function Documentation

4.34.4.1 void TextRender::adjustCoordinates (Point * *p*, TextOrientation *orientation*, unsigned char ** *x*, unsigned char ** *y*, unsigned char ** *z*)

Definition at line 36 of file TextRender.cpp.

4.34.4.2 void TextRender::printChar (Point * *p,* TextOrientation *orientation,* unsigned char *depth,* const char *c*)

Write a char on the cube.

Parameters

p	3D point
axis	Axis to print
c	The char.
size	The size.

Definition at line 13 of file TextRender.cpp.

4.34.5 Member Data Documentation

4.34.5.1 Cube* TextRender::cube [private]

The cube.

Definition at line 23 of file TextRender.h.

4.34.5.2 BitmapFont* TextRender::font [private]

The used font.

Definition at line 28 of file TextRender.h.

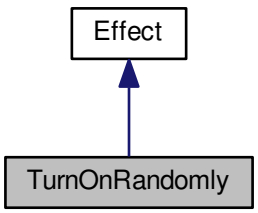
The documentation for this class was generated from the following files:

- TextRender.h
- TextRender.cpp

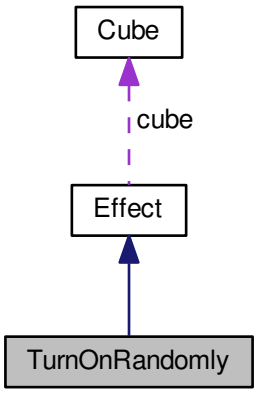
4.35 TurnOnRandomly Class Reference

`#include <TurnOnRandomly.h>`

Inheritance diagram for TurnOnRandomly:



Collaboration diagram for TurnOnRandomly:



Public Member Functions

- TurnOnRandomly (Cube *cube, unsigned int iterations, unsigned int iterationDelay, unsigned char maxOnVoxels)
- virtual void run ()

Private Attributes

- unsigned char maxOnVoxels

Additional Inherited Members

4.35.1 Detailed Description

Definition at line 10 of file TurnOnRandomly.h.

4.35.2 Constructor & Destructor Documentation

4.35.2.1 TurnOnRandomly::TurnOnRandomly (Cube * *cube*, unsigned int *iterations*, unsigned int *iterationDelay*, unsigned char *maxOnVoxels*)

Definition at line 11 of file TurnOnRandomly.cpp.

4.35.3 Member Function Documentation

4.35.3.1 void TurnOnRandomly::run ( ) [virtual]

Reimplemented from Effect.

Definition at line 15 of file TurnOnRandomly.cpp.

4.35.4 Member Data Documentation

4.35.4.1 unsigned char TurnOnRandomly::maxOnVoxels [private]

Definition at line 12 of file TurnOnRandomly.h.

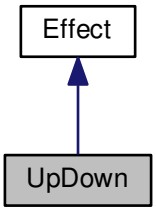
The documentation for this class was generated from the following files:

- TurnOnRandomly.h
- TurnOnRandomly.cpp

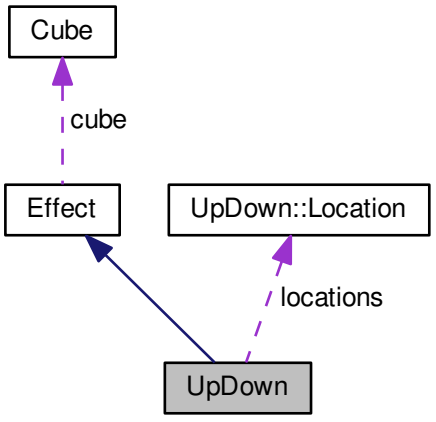
4.36 UpDown Class Reference

`#include <UpDown.h>`

Inheritance diagram for UpDown:



Collaboration diagram for UpDown:



Classes

- struct Location

Public Member Functions

- UpDown (Cube *cube, unsigned int iterations, unsigned int iterationDelay, Axis axis, unsigned char initialPosition)
- virtual void run ()
- void draw ()
- void move ()

Private Attributes

- Location locations [Cube::BYTE_SIZE]
- Axis axis
- unsigned char initialPosition

Additional Inherited Members

4.36.1 Detailed Description

Definition at line 10 of file UpDown.h.

4.36.2 Constructor & Destructor Documentation

4.36.2.1 UpDown::UpDown (Cube * *cube*, unsigned int *iterations*, unsigned int *iterationDelay*, Axis *axis*, unsigned char *initialPosition*)

Definition at line 11 of file UpDown.cpp.

4.36.3 Member Function Documentation

4.36.3.1 void UpDown::draw ( )

Draws current position to the cube.

Definition at line 33 of file UpDown.cpp.

4.36.3.2 void UpDown::move ( )

Move one step the LEDs to the destination.

Definition at line 57 of file UpDown.cpp.

4.36.3.3 void UpDown::run ( ) [virtual]

Reimplemented from Effect.

Definition at line 15 of file UpDown.cpp.

4.36.4 Member Data Documentation

4.36.4.1 Axis UpDown::axis [private]

Definition at line 18 of file UpDown.h.

4.36.4.2 unsigned char UpDown::initialPosition [private]

Definition at line 19 of file UpDown.h.

4.36.4.3 Location UpDown::locations[Cube::BYTE_SIZE] [private]

Definition at line 17 of file UpDown.h.

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

- UpDown.h
- UpDown.cpp

4.37 Util Class Reference

`#include <Util.h>`

Static Public Member Functions

- static unsigned char rotatingShift (unsigned char v, unsigned char isLeft)
- static void flipByte (unsigned char *p)
- static void orderArgs (unsigned char *a, unsigned char *b)
- static void swapArgs (unsigned char *a, unsigned char *b)
- static unsigned char byteLine (unsigned char start, unsigned char end)
- static void set (unsigned char *p, unsigned char mask)
- static void clr (unsigned char *p, unsigned char mask)

4.37.1 Detailed Description

Definition at line 11 of file Util.h.

4.37.2 Member Function Documentation

4.37.2.1 static unsigned char Util::byteLine (unsigned char *start,* unsigned char *end*) [inline], [static]

Returns a byte with a row of 1's drawn in it.

byteLine(2, 5) gives 0b00111100

Definition at line 39 of file Util.h.

4.37.2.2 static void Util::clr (unsigned char * *p,* unsigned char *mask*) [inline], [static]

Clears the ON bit on the mask.

Definition at line 53 of file Util.h.

4.37.2.3 void Util::flipByte (unsigned char * *p*) [static]

Flip a byte.

Definition at line 17 of file Util.cpp.

4.37.2.4 void Util::orderArgs (unsigned char * *a,* unsigned char * *b*) [static]

Order the args, lower first.

Definition at line 30 of file Util.cpp.

4.37.2.5 unsigned char Util::rotatingShift (unsigned char *v*, unsigned char *isLeft*) [static]

Shift the byte rotating it.

Definition at line 9 of file Util.cpp.

4.37.2.6 static void Util::set (unsigned char * *p*, unsigned char *mask*) [inline], [static]

Sets the ON bit on the mask.

Definition at line 46 of file Util.h.

4.37.2.7 void Util::swapArgs (unsigned char * *a*, unsigned char * *b*) [static]

Swap args.

Definition at line 36 of file Util.cpp.

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

- Util.h
- Util.cpp

4.38 Voxel Struct Reference

`#include <Voxel.h>`

Public Attributes

- unsigned char state

4.38.1 Detailed Description

Definition at line 27 of file Voxel.h.

4.38.2 Member Data Documentation

4.38.2.1 unsigned char Voxel::state

Definition at line 28 of file Voxel.h.

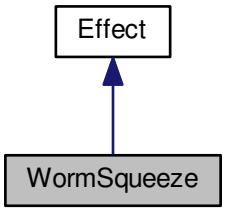
The documentation for this struct was generated from the following file:

- Voxel.h

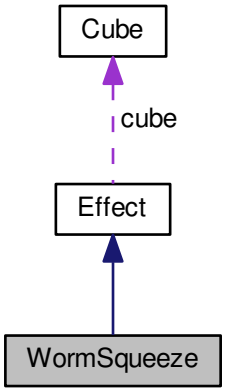
4.39 WormSqueeze Class Reference

`#include <WormSqueeze.h>`

Inheritance diagram for WormSqueeze:



Collaboration diagram for WormSqueeze:



Public Member Functions

- WormSqueeze (Cube *cube, unsigned int iterations, unsigned int iterationDelay)
- virtual void run ()

Additional Inherited Members

4.39.1 Detailed Description

Definition at line 10 of file WormSqueeze.h.

4.39.2 Constructor & Destructor Documentation

4.39.2.1 WormSqueeze::WormSqueeze (Cube * *cube*, unsigned int *iterations*, unsigned int *iterationDelay*)

Definition at line 9 of file WormSqueeze.cpp.

4.39.3 Member Function Documentation

4.39.3.1 void WormSqueeze::run ( ) [virtual]

Reimplemented from Effect.

Definition at line 13 of file WormSqueeze.cpp.

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

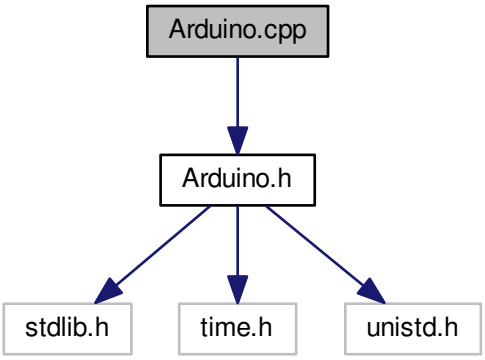
- WormSqueeze.h
- WormSqueeze.cpp

5 File Documentation

5.1 Arduino.cpp File Reference

```
#include <Arduino.h>
```

Include dependency graph for Arduino.cpp:



Functions

- void randomSeed (unsigned int seed)
- long random (long to)
- long random (long from, long to)
- long map (long x, long inMin, long inMax, long outMin, long outMax)
- long analogRead (unsigned char port)
- void delay (long millis)
- void interrupts ()
- void noInterrupts ()

Variables

- unsigned char arduinoCounter

5.1.1 Function Documentation

5.1.1.1 long analogRead (unsigned char *port*)

Definition at line 30 of file Arduino.cpp.

5.1.1.2 void delay (long *millis*)

Definition at line 34 of file Arduino.cpp.

5.1.1.3 void interrupts ( )

Definition at line 39 of file Arduino.cpp.

5.1.1.4 long map (long *x,* long *inMin,* long *inMax,* long *outMin,* long *outMax*)

Definition at line 26 of file Arduino.cpp.

5.1.1.5 void noInterrupts ( )

Definition at line 42 of file Arduino.cpp.

5.1.1.6 long random (long *to*)

Definition at line 11 of file Arduino.cpp.

5.1.1.7 long random (long *from,* long *to*)

Definition at line 18 of file Arduino.cpp.

5.1.1.8 void randomSeed (unsigned int *seed*)

Definition at line 5 of file Arduino.cpp.

5.1.2 Variable Documentation

5.1.2.1 unsigned char arduinoCounter

Definition at line 3 of file Arduino.cpp.

5.2 Arduino.cpp

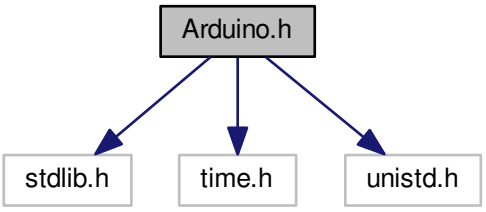
```
00001 #include <Arduino.h>
00002
00003 unsigned char arduinoCounter;
00004
00005 void randomSeed(unsigned int seed) {
00006   if (seed != 0) {
00007     srandom(seed);
00008   }
00009 }
00010
00011 long random(long to) {
00012   if (to == 0) {
00013     return 0;
00014   }
00015   return random() % to;
00016 }
00017
00018 long random(long from, long to) {
00019   if (from >= to) {
00020     return from;
00021   }
00022   long diff = to - from;
00023   return random(diff) + from;
00024 }
00025
```

```
00026 long map(long x, long inMin, long inMax, long outMin, long outMax) {
00027   return (x - inMin) * (outMax - outMin) / (inMax - inMin) + outMin;
00028 }
00029
00030 long analogRead(unsigned char port) {
00031   return time(NULL) + (port != 0) ? arduinoCounter++ : 0;
00032 }
00033
00034 void delay(long millis) {
00035   usleep(millis * 1000);
00036 }
00037
00038
00039 void interrupts() {
00040 }
00041
00042 void noInterrupts() {
00043 }
```

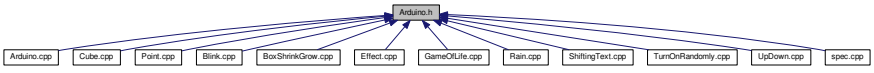
5.3 Arduino.h File Reference

```
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
```

Include dependency graph for Arduino.h:



This graph shows which files directly or indirectly include this file:



Functions

- void randomSeed (unsigned int seed)
- long random (long to)
- long random (long from, long to)
- long map (long x, long inMin, long inMax, long outMin, long outMax)
- long analogRead (unsigned char port)
- void delay (long millis)
- void interrupts ()
- void noInterrupts ()

5.3.1 Function Documentation

5.3.1.1 long analogRead (unsigned char *port*)

Definition at line 30 of file Arduino.cpp.

5.3.1.2 void delay (long *millis*)

Definition at line 34 of file Arduino.cpp.

5.3.1.3 void interrupts ( )

Definition at line 39 of file Arduino.cpp.

5.3.1.4 long map (long *x,* long *inMin,* long *inMax,* long *outMin,* long *outMax*)

Definition at line 26 of file Arduino.cpp.

5.3.1.5 void noInterrupts ( )

Definition at line 42 of file Arduino.cpp.

5.3.1.6 long random (long *to*)

Definition at line 11 of file Arduino.cpp.

5.3.1.7 long random (long *from,* long *to*)

Definition at line 18 of file Arduino.cpp.

5.3.1.8 void randomSeed (unsigned int *seed*)

Definition at line 5 of file Arduino.cpp.

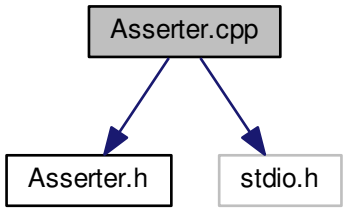
5.4 Arduino.h

```
00001
00004 #ifndef __ARDUINO_CUBE_ARDUINO_H__
00005 #define __ARDUINO_CUBE_ARDUINO_H__ 1
00006
00007 extern "C" {
00008   #include <stdlib.h>
00009   #include <time.h>
00010   #include <unistd.h>
00011 }
00012
00013 void randomSeed(unsigned int seed);
00014
00015 long random(long to);
00016
00017 long random(long from, long to);
00018
00019 long map(long x, long inMin, long inMax, long outMin, long outMax);
00020
00021 long analogRead(unsigned char port);
00022
00023 void delay(long millis);
00024
00025 void interrupts();
00026
00027 void noInterrupts();
00028
00029 #endif /* __ARDUINO_CUBE_ARDUINO_H__ */
```

5.5 Asserter.cpp File Reference

```
#include <Asserter.h>
#include <stdio.h>
```

Include dependency graph for Asserter.cpp:



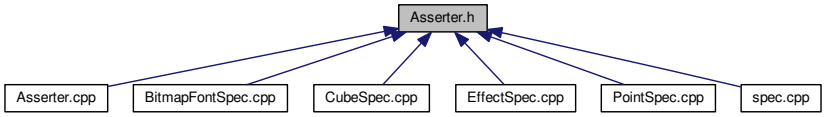
5.6 Asserter.cpp

```
00001 #include <Asserter.h>
00002 #include <stdio.h>
00003
00004 const char Asserter::ASSERT_PASSED_OUTPUT[] = "\e[32m(*) passed: %s\e[0m\n";
00005 const char Asserter::ASSERT_FAILED_OUTPUT[] = "\e[31m(*) failed: %s\e[0m\n";
00006 const char Asserter::ASSERT_EQUAL_FAILED_OUTPUT[] = "\e[31m(F) failed:
       %s (expected %d to be equal %d)\e[0m\n";
00007 const char Asserter::ASSERT_NOT_EQUAL_FAILED_OUTPUT[] = "\e[31m(F)
       failed: %s (expected %d to not be equal %d)\e[0m\n";
00008
00009 Asserter::Counter Asserter::counter = {0, 0};
00010
00011 void Asserter::reset() {
00012   counter.success = 0;
00013   counter.error = 0;
00014 }
00015
00016 bool Asserter::assert(bool assertion, const char *msg) {
00017   bool passed = true;
00018   if (assertion) {
00019     counter.success++;
00020     printf(ASSERT_PASSED_OUTPUT, msg);
00021   } else {
00022     passed = false;
00023     counter.error++;
00024     printf(ASSERT_FAILED_OUTPUT, msg);
00025   }
00026   return passed;
00027 }
00028
00029 bool Asserter::assertEqual(unsigned char a, unsigned char b, const char *msg) {
00030   bool passed = true;
00031   if(a == b) {
00032     counter.success++;
00033     printf(ASSERT_PASSED_OUTPUT, msg);
00034   } else {
00035     passed = false;
00036     counter.error++;
00037     printf(ASSERT_EQUAL_FAILED_OUTPUT, msg, a, b);
00038   }
00039   return passed;
00040 }
00041
00042 bool Asserter::assertNotEqual(unsigned char a, unsigned char b, const char *msg) {
00043   bool passed = true;
00044   if(a != b) {
00045     counter.success++;
00046     printf(ASSERT_PASSED_OUTPUT, msg);
00047   } else {
00048     passed = false;
00049     counter.error++;
00050     printf(ASSERT_NOT_EQUAL_FAILED_OUTPUT, msg, a, b);
```

```
00051   }
00052   return passed;
00053 }
```

5.7 Asserter.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class Asserter
- struct Asserter::Counter

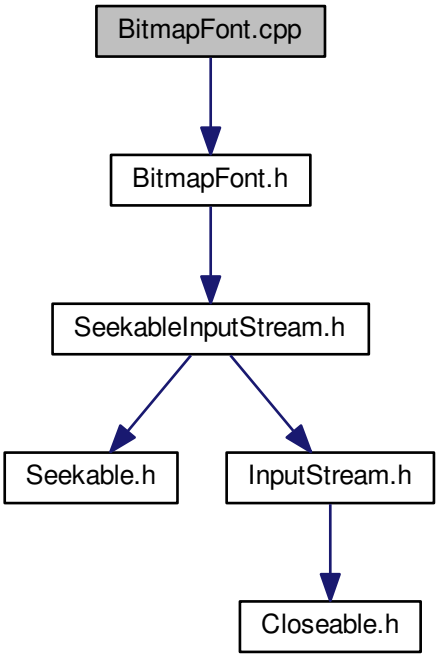
5.8 Asserter.h

```
00001
00004 #ifndef __ARDUINO_CUBE_ASSERTER_H__
00005 #define __ARDUINO_CUBE_ASSERTER_H__ 1
00006
00007 class Asserter {
00008
00009   static const char ASSERT_PASSED_OUTPUT[];
00010   static const char ASSERT_FAILED_OUTPUT[];
00011   static const char ASSERT_EQUAL_FAILED_OUTPUT[];
00012   static const char ASSERT_NOT_EQUAL_FAILED_OUTPUT[];
00013
00014 public:
00015
00016   typedef struct {
00017     unsigned int error;
00018     unsigned int success;
00019   } Counter;
00020
00021   static Counter counter;
00022
00023   static void reset();
00024
00025   static bool assert(bool assertion, const char *msg);
00026
00027   static bool assertEqual(unsigned char a, unsigned char b, const char *msg);
00028
00029   static bool assertNotEqual(unsigned char a, unsigned char b, const char *msg);
00030 };
00031
00032 #endif /* __ARDUINO_CUBE_ASSERTER_H__ */
00033
```

5.9 BitmapFont.cpp File Reference

```
#include "BitmapFont.h"
```

Include dependency graph for BitmapFont.cpp:



Macros

- #define __ARDUINO_CUBE_BITMAP_FONT_CPP__ 1

5.9.1 Macro Definition Documentation

5.9.1.1 #define __ARDUINO_CUBE_BITMAP_FONT_CPP__ 1

Arduino Cube Library.

BitmapFont.cpp

The representation of a glcd font.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 12 of file BitmapFont.cpp.

5.10 BitmapFont.cpp

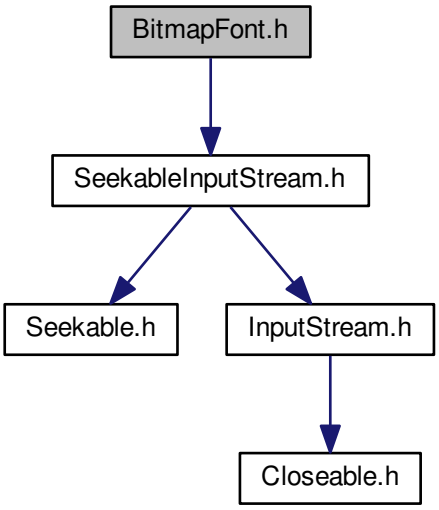
```
00001
00011 #ifndef __ARDUINO_CUBE_BITMAP_FONT_CPP__
```

```
00012 #define __ARDUINO_CUBE_BITMAP_FONT_CPP__ 1
00013
00014 #include "BitmapFont.h"
00015
00016 BitmapFont::BitmapFont(SeekableInputStream* inputStream) :
      inputStream(inputStream) {
00017   dataOffset = sizeof (Header);
00018   inputStream->seek(0);
00019   header.info = inputStream->read();
00020   header.characterWidth = inputStream->read();
00021   header.characterHeight = inputStream->read();
00022   header.sequenceCount = inputStream->read();
00023   glyphLength = header.characterWidth * (header.
      characterHeight / 8);
00024 }
00025
00026 unsigned char BitmapFont::getInfo() {
00027   return header.info;
00028 }
00029
00030 unsigned char BitmapFont::getCharacterWidth() {
00031   return header.characterWidth;
00032 }
00033
00034 unsigned char BitmapFont::getCharacterHeight() {
00035   return header.characterHeight;
00036 }
00037
00038 unsigned char BitmapFont::getSequenceCount() {
00039   return header.sequenceCount;
00040 }
00041
00042 unsigned char BitmapFont::getGlyphLength() {
00043   return glyphLength;
00044 }
00045
00046 unsigned char BitmapFont::readGlyphData(unsigned char *buf, char c) {
00047   unsigned int offset = getGlyphOffset(c);
00048   if (offset == 0) {
00049     return 0;
00050   }
00051   inputStream->seek(offset);
00052   return (unsigned char) inputStream->read(buf, 0, getGlyphLength());
00053 }
00054
00055 unsigned int BitmapFont::getGlyphOffset(char c) {
00056   unsigned char i, first, last;
00057   unsigned int offset = 0;
00058   inputStream->seek(dataOffset);
00059   for (i = 0; i < getSequenceCount(); i++) {
00060     first = inputStream->read();
00061     last = inputStream->read();
00062     if (c >= first && c <= last) {
00063       offset = inputStream->read();
00064       offset <<= 8;
00065       offset |= inputStream->read();
00066       offset += (c - first) * getGlyphLength();
00067       break;
00068     } else {
00069       inputStream->skip(2);
00070     }
00071   }
00072   return offset;
00073 }
00074
00075 #endif /* __ARDUINO_CUBE_BITMAP_FONT_CPP__ */
```

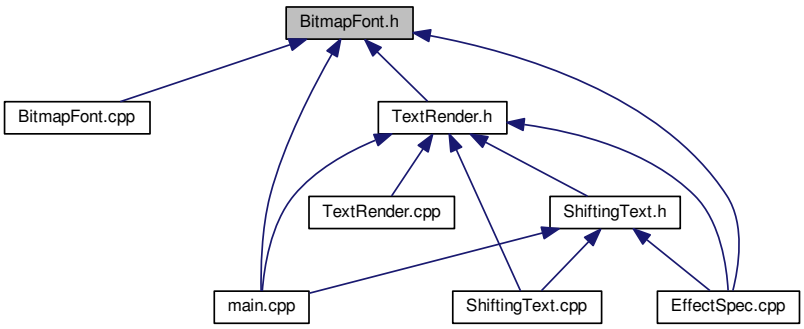
5.11 BitmapFont.h File Reference

```
#include <SeekableInputStream.h>
```

Include dependency graph for BitmapFont.h:



This graph shows which files directly or indirectly include this file:



Classes

- class BitmapFont
- struct BitmapFont::Header

5.12 BitmapFont.h

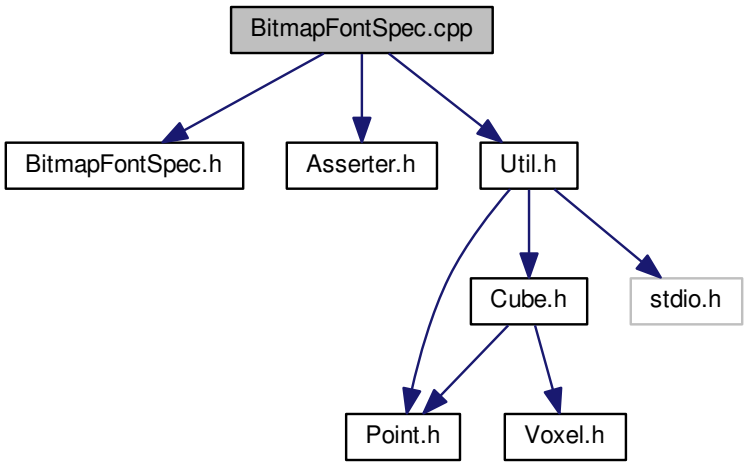
```
00001
00136 #ifndef __ARDUINO_CUBE_BITMAP_FONT_H__
00137 #define __ARDUINO_CUBE_BITMAP_FONT_H__ 1
00138
00139 #include <SeekableInputStream.h>
00140
00141 class BitmapFont {
```

```
00142
00143 protected:
00144
00148     struct Header {
00149         unsigned char info;
00150         unsigned char characterWidth;
00151         unsigned char characterHeight;
00152         unsigned char sequenceCount;
00153     };
00154     Header header;
00155
00159     unsigned char glyphLength;
00160
00164     SeekableInputStream* inputStream;
00165
00169     unsigned int dataOffset;
00170
00171 public:
00172
00178     BitmapFont(SeekableInputStream* inputStream);
00179
00185     unsigned char getInfo();
00186
00192     unsigned char getCharacterWidth();
00193
00199     unsigned char getCharacterHeight();
00200
00206     unsigned char getSequenceCount();
00207
00213     unsigned char getGlyphLength();
00214
00222     virtual unsigned char readGlyphData(unsigned char *buf, char c);
00223
00224 protected:
00225
00232     virtual unsigned int getGlyphOffset(char c);
00233 };
00234
00235 #endif /* __ARDUINO_CUBE_BITMAP_FONT_H__ */
```

5.13 BitmapFontSpec.cpp File Reference

```
#include <BitmapFontSpec.h>
#include <Asserter.h>
#include <Util.h>
```

Include dependency graph for BitmapFontSpec.cpp:



Macros

- #define __ARDUINO_CUBE_BITMAP_FONT_TEST_CPP__ 1

5.13.1 Macro Definition Documentation

5.13.1.1 #define __ARDUINO_CUBE_BITMAP_FONT_TEST_CPP__ 1

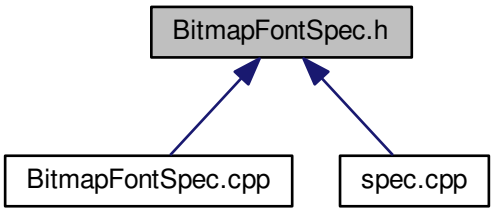
Definition at line 5 of file BitmapFontSpec.cpp.

5.14 BitmapFontSpec.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_BITMAP_FONT_TEST_CPP__
00005 #define __ARDUINO_CUBE_BITMAP_FONT_TEST_CPP__ 1
00006
00007 #include <BitmapFontSpec.h>
00008 #include <Asserter.h>
00009 #include <Util.h>
00010
00011 BitmapFontSpec::BitmapFontSpec() {
00012 }
00013
00014 void BitmapFontSpec::run() {
00015   getInfoSpec();
00016   getCharacterWidthSpec();
00017   getCharacterHeightSpec();
00018   getSequenceCountSpec();
00019   getGlyphLengthSpec();
00020   readGlyphDataSpec();
00021   getGlyphOffsetSpec();
00022 }
00023
00024 void BitmapFontSpec::getInfoSpec() {
00025   Asserter::assertEqual(0, 0, "getInfoSpec: Should pass.");
00026 }
00027
00028 void BitmapFontSpec::getCharacterWidthSpec() {
00029   Asserter::assertEqual(0, 0, "getCharacterWidthSpec: Should pass.");
00030 }
00031
00032 void BitmapFontSpec::getCharacterHeightSpec() {
00033   Asserter::assertEqual(0, 0, "getCharacterHeightSpec: Should pass.");
00034 }
00035
00036 void BitmapFontSpec::getSequenceCountSpec() {
00037   Asserter::assertEqual(0, 0, "getSequenceCountSpec: Should pass.");
00038 }
00039
00040 void BitmapFontSpec::getGlyphLengthSpec() {
00041   Asserter::assertEqual(0, 0, "getGlyphLengthSpec: Should pass.");
00042 }
00043
00044 void BitmapFontSpec::readGlyphDataSpec() {
00045   Asserter::assertEqual(0, 0, "readGlyphDataSpec: Should pass.");
00046 }
00047
00048 void BitmapFontSpec::getGlyphOffsetSpec() {
00049   Asserter::assertEqual(0, 0, "getGlyphOffsetSpec: Should pass.");
00050 }
00051
00052 #endif /* __ARDUINO_CUBE_BITMAP_FONT_TEST_CPP__ */
```

5.15 BitmapFontSpec.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class BitmapFontSpec

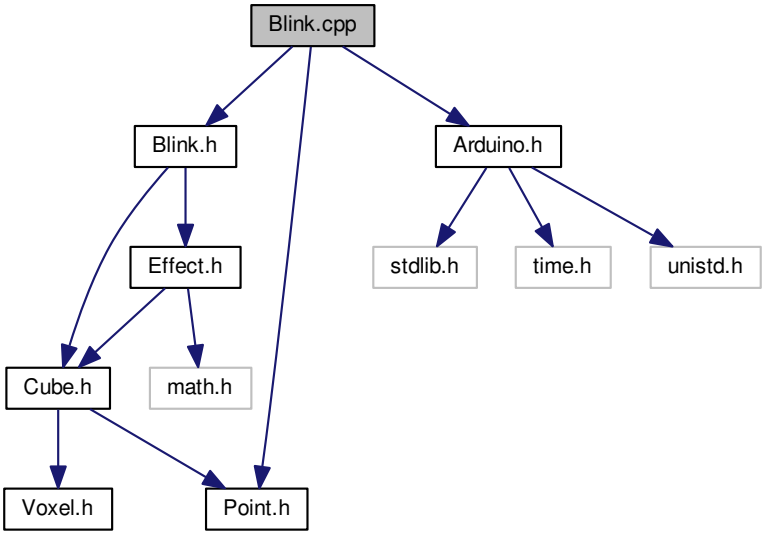
5.16 BitmapFontSpec.h

```
00001
00004 #ifndef __ARDUINO_CUBE_BITMAP_FONT_TEST_H__
00005 #define __ARDUINO_CUBE_BITMAP_FONT_TEST_H__ 1
00006
00007 class BitmapFontSpec {
00008
00009 public:
00010
00011   BitmapFontSpec();
00012
00013   void run();
00014
00015   void getInfoSpec();
00016
00017   void getCharacterWidthSpec();
00018
00019   void getCharacterHeightSpec();
00020
00021   void getSequenceCountSpec();
00022
00023   void getGlyphLengthSpec();
00024
00025   void readGlyphDataSpec();
00026
00027   void getGlyphOffsetSpec();
00028 };
00029
00030 #endif /* __ARDUINO_CUBE_BITMAP_FONT_TEST_H__ */
```

5.17 Blink.cpp File Reference

```
#include <Blink.h>
#include <Arduino.h>
#include <Point.h>
```

Include dependency graph for Blink.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_BLINK_CPP__ 1

5.17.1 Macro Definition Documentation

5.17.1.1 #define __ARDUINO_CUBE_EFFECTS_BLINK_CPP__ 1

Definition at line 5 of file Blink.cpp.

5.18 Blink.cpp

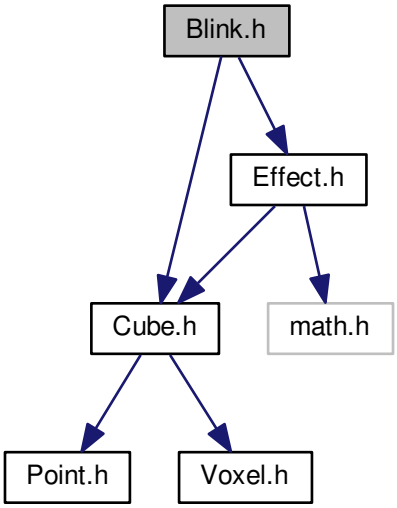
```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_BLINK_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_BLINK_CPP__ 1
00006
00007 #include <Blink.h>
00008 #include <Arduino.h>
00009 #include <Point.h>
00010
00011 Blink::Blink(Cube *cube, unsigned int iterations, unsigned int iterationDelay) :
00012     Effect(cube, iterations, iterationDelay) {
00013 }
00014
00015 void Blink::run() {
00016   unsigned int iteration;
00017   unsigned char step;
00018   int stepDelay;
00019   for (iteration = 0; iteration < iterations; iteration++) {
00020     for (step = 0; step < 2; step++) {
00021       stepDelay = iterationDelay;
00022       while (stepDelay > 0) {
00023         cube->clear();
00024         delay((step == 0) ? stepDelay : ((iterationDelay + 1) - stepDelay));
00025         cube->fill();
00026         delay(100);
00027         stepDelay -= (15 + (1000 / (stepDelay / 10)));
00028       }
00029       if (step == 0) {
```

```
00030         delay(3000);
00031       }
00032     }
00033   }
00034 }
00035
00036 #endif /* __ARDUINO_CUBE_EFFECTS_BLINK_CPP__ */
```

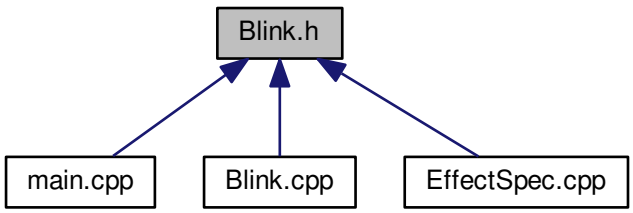
5.19 Blink.h File Reference

```
#include <Effect.h>
#include <Cube.h>
```

Include dependency graph for Blink.h:



This graph shows which files directly or indirectly include this file:



Classes

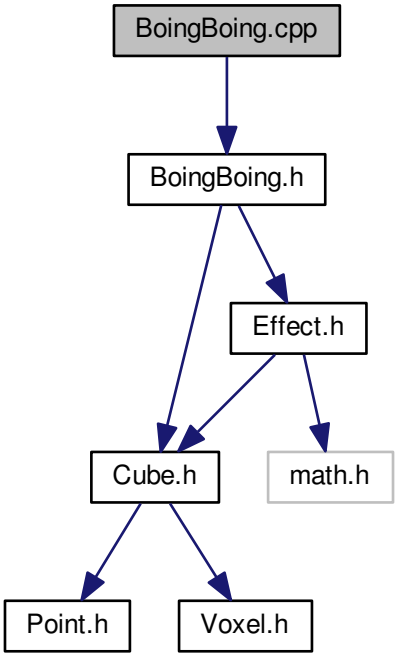
- class Blink

5.20 Blink.h

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_BLINK_H__
00005 #define __ARDUINO_CUBE_EFFECTS_BLINK_H__ 1
00006
00007 #include <Effect.h>
00008 #include <Cube.h>
00009
00010 class Blink : public Effect {
00011 public:
00012
00013   Blink(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay);
00014
00015   virtual void run();
00016 };
00017
00018 #endif /* __ARDUINO_CUBE_EFFECTS_BLINK_H__ */
```

5.21 BoingBoing.cpp File Reference

`#include <BoingBoing.h>`
Include dependency graph for BoingBoing.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_BOING_BOING_CPP__ 1

5.21.1 Macro Definition Documentation

5.21.1.1 #define __ARDUINO_CUBE_EFFECTS_BOING_BOING_CPP__ 1

Definition at line 5 of file BoingBoing.cpp.

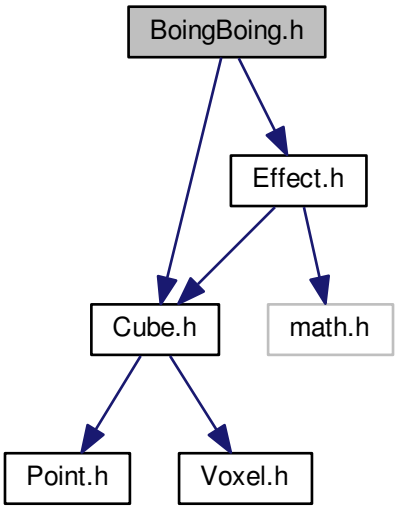
5.22 BoingBoing.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_BOING_BOING_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_BOING_BOING_CPP__ 1
00006
00007 #include <BoingBoing.h>
00008
00009 BoingBoing::BoingBoing(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay) :
00010     Effect(cube, iterations, iterationDelay) {
00011 }
00012
00013 void BoingBoing::run() {
00014   unsigned int iteration;
00015   for (iteration = 0; iteration < iterations; iteration++) {
00016   }
00017 }
00018
00019 #endif /* __ARDUINO_CUBE_EFFECTS_BOING_BOING_CPP__ */
```

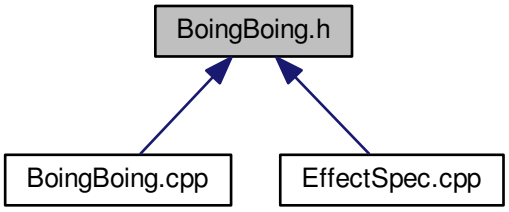
5.23 BoingBoing.h File Reference

```
#include <Effect.h>
#include <Cube.h>
```

Include dependency graph for BoingBoing.h:



This graph shows which files directly or indirectly include this file:



Classes

- class BoingBoing

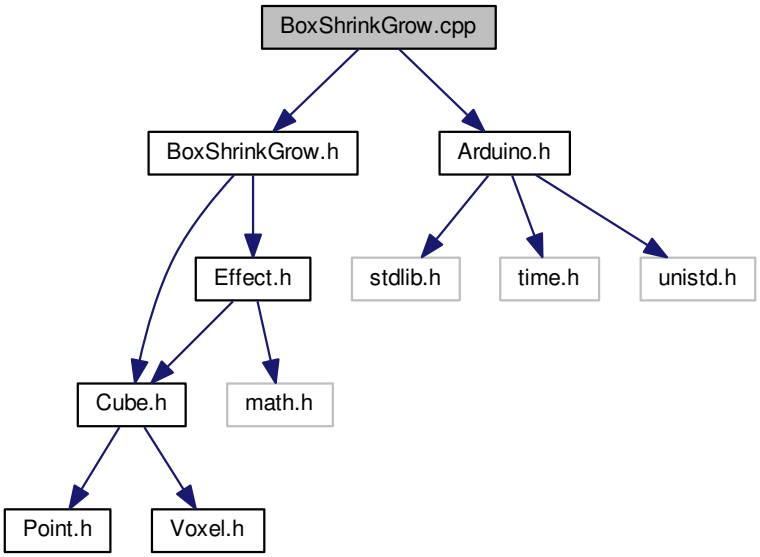
5.24 BoingBoing.h

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_BOING_BOING_H__
00005 #define __ARDUINO_CUBE_EFFECTS_BOING_BOING_H__ 1
00006
00007 #include <Effect.h>
00008 #include <Cube.h>
00009
00010 class BoingBoing : public Effect {
00011 public:
00012
00013   BoingBoing(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay);
00014
00015   virtual void run();
00016 };
00017
00018 #endif /* __ARDUINO_CUBE_EFFECTS_BOING_BOING_H__ */
```

5.25 BoxShrinkGrow.cpp File Reference

```
#include <BoxShrinkGrow.h>
#include <Arduino.h>
```

Include dependency graph for BoxShrinkGrow.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_BOX_SHRINK_GROW_CPP__ 1

5.25.1 Macro Definition Documentation

5.25.1.1 #define __ARDUINO_CUBE_EFFECTS_BOX_SHRINK_GROW_CPP__ 1

Definition at line 5 of file BoxShrinkGrow.cpp.

5.26 BoxShrinkGrow.cpp

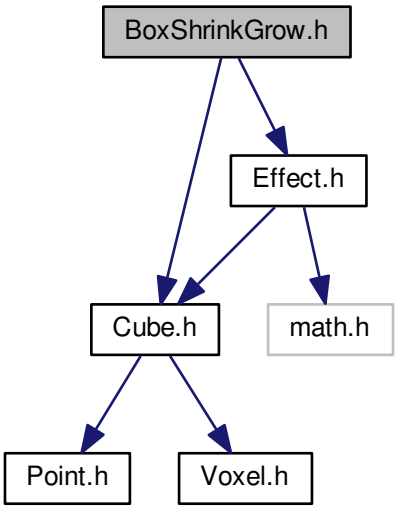
```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_BOX_SHRINK_GROW_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_BOX_SHRINK_GROW_CPP__ 1
00006
00007 #include <BoxShrinkGrow.h>
00008 #include <Arduino.h>
00009
00010 BoxShrinkGrow::BoxShrinkGrow(Cube *cube, unsigned int iterations,
      unsigned int iterationDelay, BoxType boxType) :
00011     Effect(cube, iterations, iterationDelay), boxType(boxType) {
00012 }
00013
00014 void BoxShrinkGrow::run() {
00015   unsigned int iteration;
00016   cube->useBackBuffer();
00017   for (iteration = 0; iteration < iterations; iteration++) {
00018     grow();
00019     shrink();
00020   }
00021   cube->useFrontBuffer();
00022 }
00023
00024 void BoxShrinkGrow::shrink() {
00025   char size;
00026   for (size = Cube::SIZE - 1; size >= 0; size--) {
00027     drawFrame(size);
```

```
00028   }
00029 }
00030
00031 void BoxShrinkGrow::grow() {
00032   char size;
00033   for (size = 0; size < Cube::SIZE; size++) {
00034     drawFrame(size);
00035   }
00036 }
00037
00038 void BoxShrinkGrow::drawFrame(char size) {
00039     draw(size);
00040     cube->swapBuffers();
00041     delay(iterationDelay);
00042 }
00043
00044 void BoxShrinkGrow::draw(char size) {
00045   Point from = Point();
00046   Point to = Point(size, size, size);
00047   cube->clear();
00048   switch(boxType) {
00049     case WIREFRAME:
00050       cube->wireframeBox(&from, &to);
00051       break;
00052     case FILLED:
00053       cube->filledBox(&from, &to);
00054       break;
00055     case WALL:
00056       cube->wallBox(&from, &to);
00057       break;
00058   }
00059 }
00060
00061 #endif /* __ARDUINO_CUBE_EFFECTS_BOX_SHRINK_GROW_CPP__ */
```

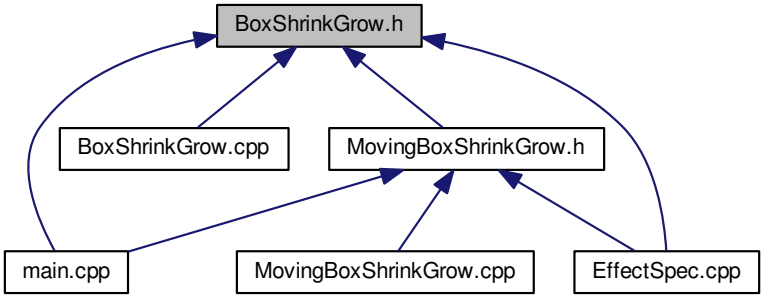
5.27 BoxShrinkGrow.h File Reference

```
#include <Effect.h>
#include <Cube.h>
```

Include dependency graph for BoxShrinkGrow.h:



This graph shows which files directly or indirectly include this file:



Classes

- class BoxShrinkGrow

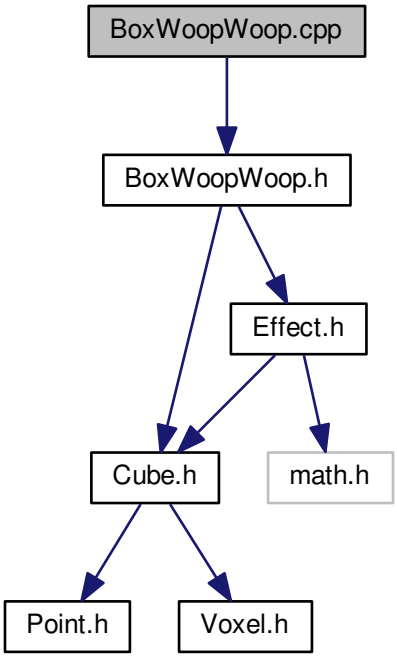
5.28 BoxShrinkGrow.h

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_BOX_SHRINK_GROW_H__
00005 #define __ARDUINO_CUBE_EFFECTS_BOX_SHRINK_GROW_H__ 1
00006
00007 #include <Effect.h>
00008 #include <Cube.h>
00009
00010 class BoxShrinkGrow : public Effect {
00011
00012 public:
00013
00014   typedef enum {
00015     WIREFRAME = 0x00,
00016     FILLED = 0x01,
00017     WALL = 0x02
00018   } BoxType;
00019
00020   BoxShrinkGrow(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay, BoxType boxType);
00021
00022   virtual void run();
00023
00024   void shrink();
00025
00026   void grow();
00027
00028   void drawFrame(char size);
00029
00030   virtual void draw(char size);
00031
00032 protected:
00033
00034   BoxType boxType;
00035 };
00036
00037 #endif /* __ARDUINO_CUBE_EFFECTS_BOX_SHRINK_GROW_H__ */
```

5.29 BoxWoopWoop.cpp File Reference

```
#include <BoxWoopWoop.h>
```

Include dependency graph for BoxWoopWoop.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_BOX_WOOP_WOOP_CPP__ 1

5.29.1 Macro Definition Documentation

5.29.1.1 #define __ARDUINO_CUBE_EFFECTS_BOX_WOOP_WOOP_CPP__ 1

Definition at line 5 of file BoxWoopWoop.cpp.

5.30 BoxWoopWoop.cpp

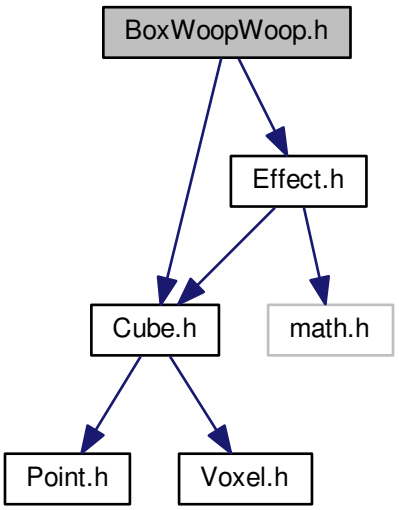
```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_BOX_WOOP_WOOP_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_BOX_WOOP_WOOP_CPP__ 1
00006
00007 #include <BoxWoopWoop.h>
00008
00009 BoxWoopWoop::BoxWoopWoop(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay) :
00010     Effect(cube, iterations, iterationDelay) {
00011 }
00012
00013 void BoxWoopWoop::run() {
00014   unsigned int iteration;
00015   for (iteration = 0; iteration < iterations; iteration++) {
00016       /*int i, ii;
00017       cube->clear();
00018       for (i = 0; i < 4; i++) {
00019           ii = i;
00020           if (grow > 0) {
00021               ii = 3 - i;
```

```
00022           }
00023           box_wireframe(4 + ii, 4 + ii, 4 + ii, 3 - ii, 3 - ii, 3 - ii);
00024           delay_ms(delay);
00025           fill(0x00);
00026       }*/
00027   }
00028 }
00029
00030 #endif /* __ARDUINO_CUBE_EFFECTS_BOX_WOOP_WOOP_CPP__ */
```

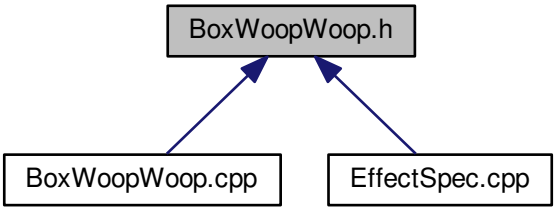
5.31 BoxWoopWoop.h File Reference

```
#include <Effect.h>
#include <Cube.h>
```

Include dependency graph for BoxWoopWoop.h:



This graph shows which files directly or indirectly include this file:



Classes

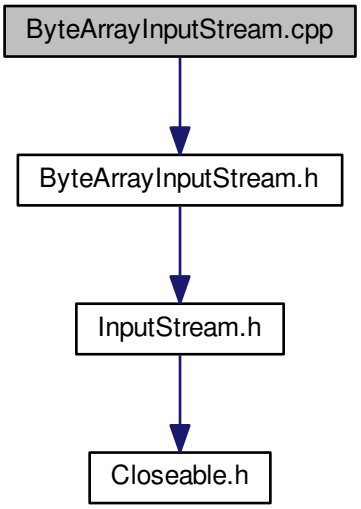
- class BoxWoopWoop

5.32 BoxWoopWoop.h

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_BOX_WOOP_WOOP_H__
00005 #define __ARDUINO_CUBE_EFFECTS_BOX_WOOP_WOOP_H__ 1
00006
00007 #include <Effect.h>
00008 #include <Cube.h>
00009
00010 class BoxWoopWoop : public Effect {
00011 public:
00012
00013   BoxWoopWoop(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay);
00014
00015   virtual void run();
00016 };
00017
00018 #endif /* __ARDUINO_CUBE_EFFECTS_BOX_WOOP_WOOP_H__ */
```

5.33 ByteArrayInputStream.cpp File Reference

`#include "ByteArrayInputStream.h"`
Include dependency graph for ByteArrayInputStream.cpp:



Macros

- #define __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_CPP__ 1

5.33.1 Macro Definition Documentation

5.33.1.1 #define __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_CPP__ 1

Arduino IO.

ByteArrayInputStream

A ByteArrayInputStream contains an internal buffer that contains bytes that may be read from the stream.

Definition at line 11 of file ByteArrayInputStream.cpp.

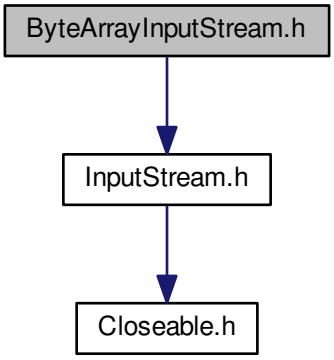
5.34 ByteArrayInputStream.cpp

```
00001
00010 #ifndef __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_CPP__
00011 #define __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_CPP__ 1
00012
00013 #include "ByteArrayInputStream.h"
00014
00015 ByteArrayInputStream::ByteArrayInputStream(unsigned char* buf,
00016         unsigned int count) :
00017         buf(buf), count(count) {
00018     markpos = 0;
00019     pos = 0;
00020 }
00021
00022 int ByteArrayInputStream::available() {
00023     if ((count - pos) > 0) {
00024         return 1;
00025     }
00026     return 0;
00027 }
00028
00029 void ByteArrayInputStream::mark() {
00030     markpos = pos;
00031 }
00032
00033 bool ByteArrayInputStream::markSupported() {
00034     return true;
00035 }
00036
00037 int ByteArrayInputStream::read() {
00038     if (pos >= count) {
00039         return -1;
00040     }
00041     return buf[pos++];
00042 }
00043
00044 void ByteArrayInputStream::reset() {
00045     pos = markpos;
00046 }
00047
00048 #endif /* __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_CPP__ */
```

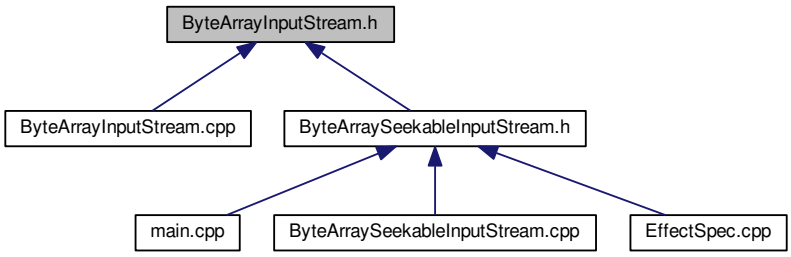
5.35 ByteArrayInputStream.h File Reference

```
#include <InputStream.h>
```

Include dependency graph for ByteArrayInputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ByteArrayInputStream

5.36 ByteArrayInputStream.h

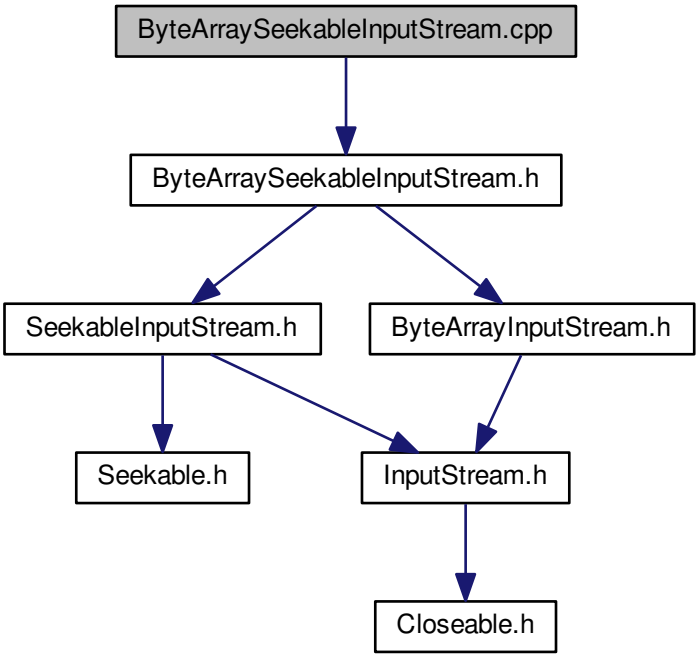
```
00001
00010 #ifndef __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_H__
00011 #define __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_H__ 1
00012
00013 #include <InputStream.h>
00014
00015 class ByteArrayInputStream : public virtual InputStream {
00016 protected:
00017
00018     /*
00019      * The buffer where data is stored.
00020      */
00021     unsigned char* buf;
00022
00023     /*
00024      * The number of valid bytes in the buffer.
00025      */
00026     unsigned int count;
```

```
00027
00028     /*
00029      * Current position
00030      */
00031     unsigned int pos;
00032
00033     /*
00034      * The currently marked position in the stream.
00035      */
00036     unsigned int markpos;
00037
00038 public:
00039
00040     ByteArrayInputStream(unsigned char* buf, unsigned int count);
00041
00054     virtual int available();
00055
00059     virtual void mark();
00060
00066     virtual bool markSupported();
00067
00071     using InputStream::read;
00072
00078     virtual int read();
00079
00084     virtual void reset();
00085 };
00086
00087 #endif /* __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_H__ */
```

5.37 ByteArraySeekableInputStream.cpp File Reference

`#include "ByteArraySeekableInputStream.h"`

Include dependency graph for ByteArraySeekableInputStream.cpp:



Macros

- #define __ARDUINO_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_CPP__ 1

5.37.1 Macro Definition Documentation

5.37.1.1 #define __ARDUINO_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_CPP__ 1

Arduino IO.

ByteArraySeekableInputStream

A ByteArraySeekableInputStream obtains input bytes from a resource in a file system that implements SeekableInputStream interface.

Definition at line 11 of file ByteArraySeekableInputStream.cpp.

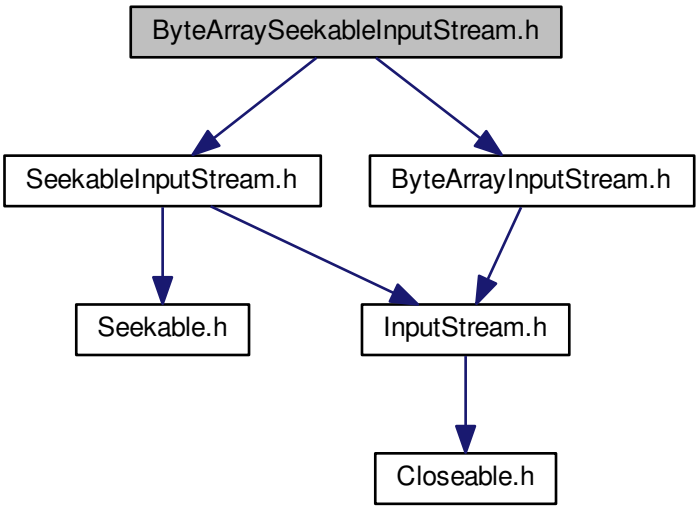
5.38 ByteArraySeekableInputStream.cpp

```
00001
00010 #ifndef __ARDUINO_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_CPP__
00011 #define __ARDUINO_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_CPP__ 1
00012
00013 #include "ByteArraySeekableInputStream.h"
00014
00015 ByteArraySeekableInputStream::ByteArraySeekableInputStream
      (unsigned char* buf,
00016         unsigned int count) :
00017         ByteArrayInputStream(buf, count) {
00018 }
00019
00020 void ByteArraySeekableInputStream::seek(unsigned int pos) {
00021     this->pos = pos;
00022 }
00023
00024 #endif /* __ARDUINO_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_CPP__ */
```

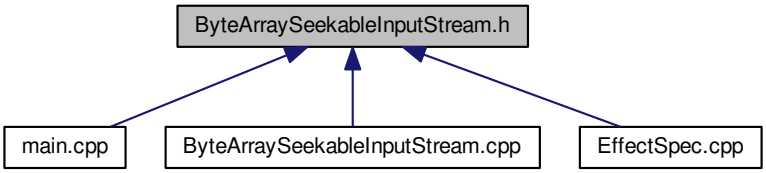
5.39 ByteArraySeekableInputStream.h File Reference

```
#include <SeekableInputStream.h>
#include <ByteArrayInputStream.h>
```

Include dependency graph for ByteArraySeekableInputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ByteArraySeekableInputStream

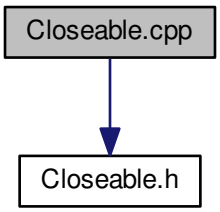
5.40 ByteArraySeekableInputStream.h

```
00001
00010 #ifndef __ARDUINO_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_H__
00011 #define __ARDUINO_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_H__ 1
00012
00013 #include <SeekableInputStream.h>
00014 #include <ByteArrayInputStream.h>
00015
00016 class ByteArraySeekableInputStream : public
      SeekableInputStream,
00017         public ByteArrayInputStream {
00018 public:
00019
00020     ByteArraySeekableInputStream(unsigned char* buf, unsigned int
      count);
```

```
00021
00022     virtual void seek(unsigned int pos);
00023 };
00024
00025 #endif /* __ARDUINO_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_H__ */
```

5.41 Closeable.cpp File Reference

`#include "Closeable.h"`
Include dependency graph for Closeable.cpp:



Macros

- #define __ARDUINO_IO_CLOSEABLE_CPP__ 1

5.41.1 Macro Definition Documentation

5.41.1.1 #define __ARDUINO_IO_CLOSEABLE_CPP__ 1

Arduino IO.

Closeable

A Closeable is a source or destination of data that can be closed.

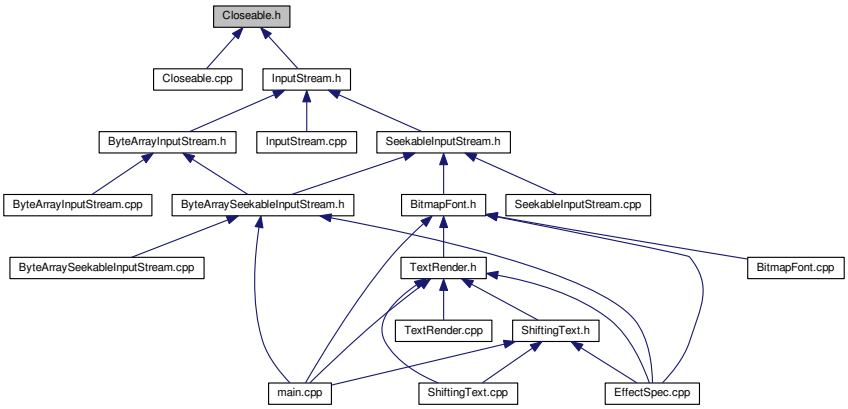
Definition at line 10 of file Closeable.cpp.

5.42 Closeable.cpp

```
00001
00009 #ifndef __ARDUINO_IO_CLOSEABLE_CPP__
00010 #define __ARDUINO_IO_CLOSEABLE_CPP__ 1
00011
00012 #include "Closeable.h"
00013
00014 #endif /* __ARDUINO_IO_CLOSEABLE_CPP__ */
```

5.43 Closeable.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class Closeable

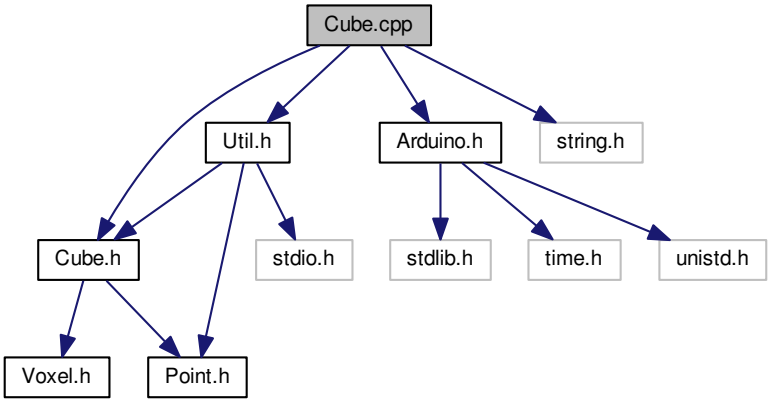
5.44 Closeable.h

```
00001
00009 #ifndef __ARDUINO_IO_CLOSEABLE_H__
00010 #define __ARDUINO_IO_CLOSEABLE_H__ 1
00011
00012 class Closeable {
00013 public:
00014
00015     virtual void close() = 0;
00016 };
00017
00018 #endif /* __ARDUINO_IO_CLOSEABLE_H__ */
```

5.45 Cube.cpp File Reference

```
#include <Cube.h>
#include <Util.h>
#include <Arduino.h>
#include <string.h>
```

Include dependency graph for Cube.cpp:



Macros

- #define __ARDUINO_CUBE_CUBE_CPP__ 1
- #define AT(y, z) *(*bufferToWrite + ((z * CUBE_SIZE + y) & CUBE_BYTE_SIZE_MASK))

5.45.1 Macro Definition Documentation

5.45.1.1 #define __ARDUINO_CUBE_CUBE_CPP__ 1

Definition at line 5 of file Cube.cpp.

5.45.1.2 #define AT(y, z) *(*bufferToWrite + ((z * CUBE_SIZE + y) & CUBE_BYTE_SIZE_MASK))

Definition at line 12 of file Cube.cpp.

5.46 Cube.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_CUBE_CPP__
00005 #define __ARDUINO_CUBE_CUBE_CPP__ 1
00006
00007 #include <Cube.h>
00008 #include <Util.h>
00009 #include <Arduino.h>
00010 #include <string.h>
00011
00012 #define AT(y, z) *(*bufferToWrite + ((z * CUBE_SIZE + y) & CUBE_BYTE_SIZE_MASK))
00013
00014 unsigned char Cube::buffer0[CUBE_SIZE][CUBE_SIZE] = {};
00015 unsigned char Cube::buffer1[CUBE_SIZE][CUBE_SIZE] = {};
00016
00017 void Cube::selectBuffer(Buffer buffer) {
00018   noInterrupts();
00019   if (buffer == BACK_BUFFER) {
00020     bufferToWrite = &backBuffer;
00021   } else {
00022     bufferToWrite = &frontBuffer;
00023   }
00024   interrupts();
00025 }
00026
00027 void Cube::swapBuffers() {
00028   unsigned char *buf;
00029   noInterrupts();
```

```
00030   buf = backBuffer;
00031   backBuffer = frontBuffer;
00032   frontBuffer = buf;
00033   interrupts();
00034 }
00035
00036 bool Cube::isInRange(Point *p) const {
00037   if (p->x < 0 || p->y < 0 || p->z < 0 || p->y >= Cube::SIZE || p->
      x >= Cube::SIZE || p->z >= Cube::SIZE) {
00038     return false;
00039   }
00040   return true;
00041 }
00042
00043 void Cube::fitInRange(Point *p) {
00044   p->x &= CUBE_SIZE_MASK;
00045   p->y &= CUBE_SIZE_MASK;
00046   p->z &= CUBE_SIZE_MASK;
00047 }
00048
00049 void Cube::fill(unsigned char pattern) {
00050   memset(*bufferToWrite, pattern, Cube::BYTE_SIZE);
00051 }
00052
00053 void Cube::writeVoxel(unsigned char x, unsigned char y, unsigned char z, unsigned char
      state) {
00054   unsigned char mask;
00055   mask = 1 << x;
00056   state == ON ? Util::set(&AT(y, z), mask) : Util::clr(&AT(y, z), mask);
00057 }
00058
00059 void Cube::readVoxel(Point *p, Voxel *v) {
00060   fitInRange(p);
00061   v->state =  (AT(p->y, p->z) & (0x01 << p->x)) != 0 ? ON : OFF;
00062 }
00063
00064 void Cube::turnVoxelOn(Point *p) {
00065   Voxel v = {ON};
00066   writeVoxel(p, v);
00067 }
00068
00069 void Cube::turnVoxelOff(Point *p) {
00070   Voxel v = {OFF};
00071   writeVoxel(p, v);
00072 }
00073
00074 void Cube::invertVoxel(Point *p) {
00075   unsigned char mask;
00076   mask = 0x01 << p->x;
00077   (AT(p->y, p->z) & mask) ? Util::clr(&AT(p->y, p->z), mask) :
      Util::set(&AT(p->y, p->z), mask);
00078 }
00079
00080 void Cube::turnPlaneZOff(unsigned char z) {
00081   Voxel v = {OFF};
00082   writePlaneZ(z, v);
00083 }
00084
00085 void Cube::turnPlaneZOn(unsigned char z) {
00086   Voxel v = {ON};
00087   writePlaneZ(z, v);
00088 }
00089
00090 void Cube::writePlaneZ(unsigned char z, Voxel v) {
00091   unsigned char pattern;
00092   z %= Cube::SIZE;
00093   pattern = (v.state == ON) ? 0xff : 0x00;
00094   memset(&AT(0, z), pattern, Cube::SIZE);
00095 }
00096
00097 void Cube::turnPlaneYOff(unsigned char y) {
00098   Voxel v = {OFF};
00099   writePlaneY(y, v);
00100 }
00101
00102 void Cube::turnPlaneYOn(unsigned char y) {
00103   Voxel v = {ON};
00104   writePlaneY(y, v);
00105 }
00106
00107 void Cube::writePlaneY(unsigned char y, Voxel v) {
00108   unsigned char z;
00109   y %= Cube::SIZE;
00110   for (z = 0; z < Cube::SIZE; z++)
00111     AT(y, z) = (v.state == ON) ? 0xff : 0x00;
00112 }
00113
```

```
00114 void Cube::turnPlaneXOff(unsigned char x) {
00115   Voxel v = {OFF};
00116   writePlaneX(x, v);
00117 }
00118
00119 void Cube::turnPlaneXOn(unsigned char x) {
00120   Voxel v = {ON};
00121   writePlaneX(x, v);
00122 }
00123
00124 void Cube::writePlaneX(unsigned char x, Voxel v) {
00125   unsigned char z, y, mask;
00126   mask = 1 << x;
00127   x %= Cube::SIZE;
00128   for (z = 0; z < Cube::SIZE; z++) {
00129     for (y = 0; y < Cube::SIZE; y++) {
00130       v.state == ON ? Util::set(&AT(y, z), mask) : Util::clr(&
      AT(y, z), mask);
00131     }
00132   }
00133 }
00134
00135 void Cube::writePlane(Axis axis, unsigned char pos, Voxel v) {
00136   switch(axis) {
00137     case AXIS_X:
00138       writePlaneX(pos, v);
00139       break;
00140     case AXIS_Y:
00141       writePlaneY(pos, v);
00142       break;
00143     case AXIS_Z:
00144       writePlaneZ(pos, v);
00145       break;
00146   }
00147 }
00148
00149 void Cube::line(Point *from, Point *to) {
00150   float ySteps, zSteps;
00151   Point p;
00152   if (from->x > to->x) {
00153       Point *aux = from;
00154       from = to;
00155       to = aux;
00156   }
00157   if (from->y > to->y) {
00158     ySteps = (float) (from->y - to->y) / (float) (to->x - from->x);
00159   } else {
00160     ySteps = (float) (to->y - from->y) / (float) (to->x - from->x);
00161   }
00162   if (from->z > to->z) {
00163     zSteps = (float) (from->z - to->z) / (float) (to->x - from->x);
00164   } else {
00165     zSteps = (float) (to->z - from->z) / (float) (to->x - from->x);
00166   }
00167   for (p.x = from->x; p.x <= to->x; p.x++) {
00168     p.y = (ySteps * (p.x - from->x)) + from->y;
00169     p.z = (zSteps * (p.x - from->x)) + from->z;
00170     turnVoxelOn(&p);
00171   }
00172 }
00173
00174 void Cube::mirrorX() {
00175   unsigned char y, z, buf[Cube::SIZE][Cube::SIZE];
00176   memcpy(buf, *bufferToWrite, Cube::BYTE_SIZE);
00177   for (z = 0; z < Cube::SIZE; z++) {
00178     for (y = 0; y < Cube::SIZE; y++) {
00179       Util::flipByte(&buf[z][y]);
00180     }
00181   }
00182   memcpy(*bufferToWrite, buf, Cube::BYTE_SIZE);
00183 }
00184
00185 void Cube::mirrorY() {
00186   unsigned char i, j, z, buf[Cube::SIZE][Cube::SIZE];
00187   memcpy(buf, *bufferToWrite, Cube::BYTE_SIZE);
00188   clear();
00189   for (z = 0; z < Cube::SIZE; z++) {
00190     for (i = 0, j = Cube::SIZE - 1; i < Cube::SIZE; i++, j--) {
00191       AT(i, z) = buf[z][j];
00192     }
00193   }
00194 }
00195
00196 void Cube::mirrorZ() {
00197   unsigned char i, j, y, buf[Cube::SIZE][Cube::SIZE];
00198   memcpy(buf, *bufferToWrite, Cube::BYTE_SIZE);
00199   clear();
```

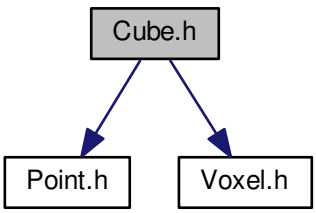
```
00200   for (i = 0, j = Cube::SIZE - 1; i < Cube::SIZE; i++, j--) {
00201     for (y = 0; y < Cube::SIZE; y++) {
00202       AT(y, i) = buf[j][y];
00203     }
00204   }
00205 }
00206
00207 void Cube::filledBox(Point *from, Point *to) {
00208   unsigned char z, y;
00209   Util::orderArgs(&from->x, &to->x);
00210   Util::orderArgs(&from->y, &to->y);
00211   Util::orderArgs(&from->z, &to->z);
00212   for (z = from->z; z <= to->z; z++) {
00213     for (y = from->y; y <= to->y; y++) {
00214       AT(y, z) |= Util::byteLine(from->x, to->x);
00215     }
00216   }
00217 }
00218
00219 void Cube::writeSubCube(Point *p, Voxel v, unsigned char size) {
00220   unsigned char x, y, z;
00221   x = p->x + size;
00222   for (z = p->z; z < p->z + size; z++) {
00223     for (y = p->y; y < p->y + size; y++) {
00224       AT(z, y) |= Util::byteLine(p->x, x);
00225     }
00226   }
00227 }
00228
00229 void Cube::wallBox(Point *from, Point *to) {
00230   unsigned char z, y, aux;
00231   Util::orderArgs(&(from->x), &(to->x));
00232   Util::orderArgs(&(from->y), &(to->y));
00233   Util::orderArgs(&(from->z), &(to->z));
00234   for (z = from->z; z <= to->z; z++) {
00235     for (y = from->y; y <= to->y; y++) {
00236       if (y == from->y || y == to->y || z == from->z || z == to->z) {
00237         aux = Util::byteLine(from->x, to->x);
00238       } else {
00239         aux |= ((0x01 << from->x) | (0x01 << to->x));
00240       }
00241       AT(y, z) = aux;
00242     }
00243   }
00244 }
00245
00246 void Cube::wireframeBox(Point *from, Point *to) {
00247   unsigned char z, y;
00248   Util::orderArgs(&(from->x), &(to->x));
00249   Util::orderArgs(&(from->y), &(to->y));
00250   Util::orderArgs(&(from->z), &(to->z));
00251   AT(from->y, from->z) = Util::byteLine(from->x, to->x);
00252   AT(to->y, from->z) = Util::byteLine(from->x, to->x);
00253   AT(from->y, to->z) = Util::byteLine(from->x, to->x);
00254   AT(to->y, to->z) = Util::byteLine(from->x, to->x);
00255   for (y = from->y; y <= to->y; y++) {
00256     writeVoxel(from->x, y, from->z, ON);
00257     writeVoxel(from->x, y, to->z, ON);
00258     writeVoxel(to->x, y, from->z, ON);
00259     writeVoxel(to->x, y, to->z, ON);
00260   }
00261   for (z = from->z; z <= to->z; z++) {
00262     writeVoxel(from->x, from->y, z, ON);
00263     writeVoxel(from->x, to->y, z, ON);
00264     writeVoxel(to->x, from->y, z, ON);
00265     writeVoxel(to->x, to->y, z, ON);
00266   }
00267 }
00268
00269 void Cube::shiftOnX(Direction direction) {
00270   unsigned char y, z, aux;
00271   for (z = 0; z < Cube::SIZE; z++) {
00272     for (y = 0; y < Cube::SIZE; y++) {
00273       aux = AT(z, y);
00274       if (direction == LEFT) {
00275         AT(z, y) <<= 1;
00276         AT(z, y) |= aux >> 7;
00277       } else {
00278         AT(z, y) >>= 1;
00279         AT(z, y) |= aux << 7;
00280       }
00281     }
00282   }
00283 }
00284
00285 void Cube::shiftOnY(Direction direction) {
00286   unsigned char z, *b, buf[Cube::SIZE][Cube::SIZE];
```

```
00287   bool isFront;
00288   isFront = (direction == FRONT);
00289   b = &(buf[0][0]);
00290   memcpy(b, *bufferToWrite, Cube::BYTE_SIZE);
00291   for (z = 0; z < Cube::SIZE; z++) {
00292     AT((isFront ? 0 : Cube::SIZE - 1), z) = buf[z][(isFront ? Cube::SIZE - 1 : 0)];
00293     memcpy(
00294       *bufferToWrite + (z * Cube::SIZE) + (isFront ? 1 : 0),
00295       b + (z * Cube::SIZE) + (isFront ? 0 : 1),
00296       Cube::SIZE - 1
00297     );
00298   }
00299 }
00300
00301 void Cube::shiftOnZ(Direction direction) {
00302   unsigned char z, k, *p, *first, *last, *src, *dst, aux[Cube::SIZE];
00303   first = *bufferToWrite;
00304   last = *bufferToWrite + (Cube::BYTE_SIZE -
      Cube::SIZE);
00305   p = (direction == UP) ? last : first;
00306   memcpy(aux, p, Cube::SIZE);
00307   for (z = 0, k = Cube::SIZE - 1; z < Cube::SIZE - 1; z++, k--) {
00308     dst = (direction == UP) ? (*bufferToWrite + Cube::SIZE * k) : (*
      bufferToWrite + Cube::SIZE * z);
00309     src = (direction == UP) ? dst - Cube::SIZE : dst + Cube::SIZE;
00310     memcpy(dst, src, Cube::SIZE);
00311   }
00312   p = (direction == UP) ? first : last;
00313   memcpy(p, aux, Cube::SIZE);
00314 }
00315
00316 void Cube::shift(Axis axis, Direction direction) {
00317   switch(axis) {
00318     case AXIS_X:
00319       shiftOnX(direction);
00320       break;
00321     case AXIS_Y:
00322       shiftOnY(direction);
00323       break;
00324     case AXIS_Z:
00325       shiftOnZ(direction);
00326       break;
00327   }
00328 }
00329
00330 #endif /* __ARDUINO_CUBE_CUBE_CPP__ */
```

5.47 Cube.h File Reference

```
#include <Point.h>
#include <Voxel.h>
```

Include dependency graph for Cube.h:



This graph shows which files directly or indirectly include this file:

Classes

- class Cube

Macros

- #define CUBE_SIZE 0x08
- #define CUBE_BYTE_SIZE CUBE_SIZE * CUBE_SIZE
- #define CUBE_SIZE_MASK 0x07
- #define CUBE_BYTE_SIZE_MASK 0x3f

5.47.1 Macro Definition Documentation

5.47.1.1 #define CUBE_BYTE_SIZE CUBE_SIZE * CUBE_SIZE

Definition at line 9 of file Cube.h.

5.47.1.2 #define CUBE_BYTE_SIZE_MASK 0x3f

Definition at line 11 of file Cube.h.

5.47.1.3 #define CUBE_SIZE 0x08

Cube class.

Definition at line 8 of file Cube.h.

5.47.1.4 #define CUBE_SIZE_MASK 0x07

Definition at line 10 of file Cube.h.

5.48 Cube.h

```
00001
00005 #ifndef __ARDUINO_CUBE_CUBE_H__
00006 #define __ARDUINO_CUBE_CUBE_H__ 1
00007
00008 #define CUBE_SIZE             0x08
00009 #define CUBE_BYTE_SIZE        CUBE_SIZE * CUBE_SIZE
00010 #define CUBE_SIZE_MASK        0x07
00011 #define CUBE_BYTE_SIZE_MASK   0x3f
00012
00013 #include <Point.h>
00014 #include <Voxel.h>
00015
00016 class Cube {
00017
00018   static unsigned char buffer0[CUBE_SIZE][CUBE_SIZE];
00019   static unsigned char buffer1[CUBE_SIZE][CUBE_SIZE];
00020   unsigned char **bufferToWrite;
00021
00022 public:
00023
00024   const static unsigned char SIZE = CUBE_SIZE;
00025   const static unsigned char BYTE_SIZE = CUBE_BYTE_SIZE;
00026   unsigned char *frontBuffer;
00027   unsigned char *backBuffer;
00028
```

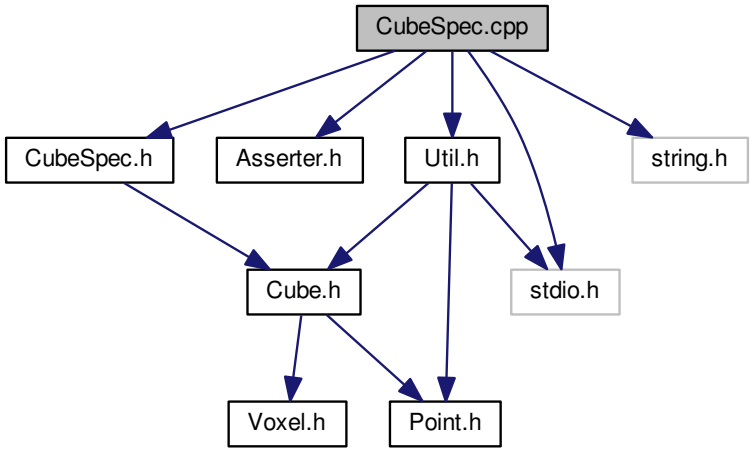
```
00029   typedef enum {
00030     FRONT_BUFFER = 0x00,
00031     BACK_BUFFER = 0x01
00032   } Buffer;
00033
00034   Cube() {
00035     frontBuffer = &buffer0[0][0];
00036     backBuffer = &buffer1[0][0];
00037     bufferToWrite = &frontBuffer;
00038   }
00039
00043   void selectBuffer(Buffer buffer);
00044
00048   void useBackBuffer() {
00049     selectBuffer(BACK_BUFFER);
00050   }
00051
00055   void useFrontBuffer() {
00056     selectBuffer(FRONT_BUFFER);
00057   }
00058
00062   bool isInRange(Point *p) const;
00063
00067   void fitInRange(Point *p);
00068
00072   void fill() {
00073     fill(0xff);
00074   }
00075
00079   void clear() {
00080     fill(0x00);
00081   }
00082
00089   void fill(unsigned char pattern);
00090
00099   void writeVoxel(Point *p, Voxel v) {
00100     writeVoxel(p->x, p->y, p->z, v.state);
00101   }
00102
00106   void writeVoxel(unsigned char x, unsigned char y, unsigned char z, unsigned char state);
00107
00113   void turnVoxelOn(Point *p);
00114
00120   void readVoxel(Point *p, Voxel *v);
00121
00127   void turnVoxelOff(Point *p);
00128
00132   void invertVoxel(Point *p);
00133
00137   void turnPlaneZOff(unsigned char z);
00138
00142   void turnPlaneZOn(unsigned char z);
00143
00147   void writePlaneZ(unsigned char z, Voxel v);
00148
00152   void turnPlaneYOff(unsigned char y);
00153
00157   void turnPlaneYOn(unsigned char y);
00158
00162   void writePlaneY(unsigned char y, Voxel v);
00163
00167   void turnPlaneXOff(unsigned char x);
00168
00172   void turnPlaneXOn(unsigned char x);
00173
00177   void writePlaneX(unsigned char x, Voxel v);
00178
00182   void writePlane(Axis axis, unsigned char pos, Voxel v);
00183
00187   void mirrorX();
00188
00192   void mirrorY();
00193
00197   void mirrorZ();
00198
00202  void line(Point *from, Point *to);
00203
00207   void filledBox(Point *from, Point *to);
00208
00212   void wallBox(Point *from, Point *to);
00213
00217   void wireframeBox(Point *from, Point *to);
00218
00222   void shift(Axis axis, Direction direction);
00223
00227   void shiftOnX(Direction direction);
00228
```

```
00232   void shiftOnY(Direction direction);
00233
00237   void shiftOnZ(Direction direction);
00238
00242   void writeSubCube(Point *p, Voxel v, unsigned char size);
00243
00247   void swapBuffers();
00248 };
00249
00250 #endif /* __ARDUINO_CUBE_CUBE_H__ */
```

5.49 CubeSpec.cpp File Reference

```
#include <CubeSpec.h>
#include <Asserter.h>
#include <Util.h>
#include <stdio.h>
#include <string.h>
```

Include dependency graph for CubeSpec.cpp:



Macros

- #define AT(b, y, z) *(b + ((((z) * CUBE_SIZE) + (y))))

5.49.1 Macro Definition Documentation

5.49.1.1 #define AT(*b*, *y*, *z*) *(b + ((((z) * CUBE_SIZE) + (y))))

Definition at line 7 of file CubeSpec.cpp.

5.50 CubeSpec.cpp

```
00001 #include <CubeSpec.h>
00002 #include <Asserter.h>
00003 #include <Util.h>
00004 #include <stdio.h>
00005 #include <string.h>
00006
00007 #define AT(b, y, z) *(b + ((((z) * CUBE_SIZE) + (y))))
```

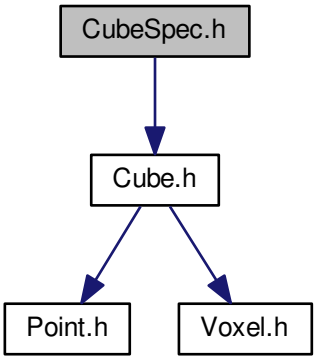
```
00008
00009 CubeSpec::CubeSpec(Cube *cube) : cube(cube) {
00010 }
00011
00012 void CubeSpec::run() {
00013   isInRangeSpec();
00014   writeVoxelSpec();
00015   invertVoxelSpec();
00016   writePlaneZSpec();
00017   writePlaneYSpec();
00018   writePlaneXSpec();
00019   flipByteSpec();
00020   mirrorXSpec();
00021   mirrorYSpec();
00022   mirrorZSpec();
00023   filledBoxSpec();
00024   lineSpec();
00025   shiftOnXSpec();
00026   shiftOnYSpec();
00027   shiftOnZSpec();
00028 }
00029
00030 void CubeSpec::isInRangeSpec() {
00031   Point p = Point();
00032   Asserter::assertEqual(cube->isInRange(&p), true, "isInRange: Point
       should be in range.");
00033   p.y = Cube::SIZE;
00034   Asserter::assertEqual(cube->isInRange(&p), false, "isInRange: Point
       with wrong y should not be in range.");
00035   p.y = 0;
00036   p.z = Cube::SIZE;
00037   Asserter::assertEqual(cube->isInRange(&p), false, "isInRange: Point
       with wrong z should not be in range.");
00038 }
00039
00040 void CubeSpec::writeVoxelSpec() {
00041   unsigned char x = 5, y = 1, z = 1;
00042   Point p = Point(x, y, z);
00043   Voxel v = {ON};
00044   AT(cube->frontBuffer, y, z) = 0x00;
00045   cube->useBackBuffer();
00046   cube->writeVoxel(&p, v);
00047   Asserter::assertEqual(AT(cube->backBuffer, y, z), (0x01 << x), "
      writeVoxel: Should write the correct voxel when ON.");
00048   v.state = OFF;
00049   cube->writeVoxel(&p, v);
00050   Asserter::assertEqual(AT(cube->backBuffer, y, z), 0x00, "writeVoxel:
       Should write the correct voxel when OFF.");
00051   cube->useFrontBuffer();
00052   cube->writeVoxel(&p, v);
00053   Asserter::assertEqual(AT(cube->frontBuffer, y, z), 0x00, "
      writeVoxel: Should write the correct voxel when writing to the front buffer.");
00054 }
00055
00056 void CubeSpec::invertVoxelSpec() {
00057   unsigned char x = 5, y = 3, z = 1;
00058   Point p = Point(x, y, z);
00059   AT(cube->frontBuffer, y, z) = 0xff;
00060   cube->useFrontBuffer();
00061   cube->invertVoxel(&p);
00062   Asserter::assertEqual(AT(cube->frontBuffer, y, z), ~(0x01 << x), "
      invertVoxel: should invert the voxel.");
00063   cube->useBackBuffer();
00064   AT(cube->backBuffer, y, z) = 0x00;
00065   cube->invertVoxel(&p);
00066   Asserter::assertEqual(AT(cube->backBuffer, y, z), 0x01 << x, "
      invertVoxel: should invert the voxel when writing to the back buffer.");
00067 }
00068
00069 void CubeSpec::writePlaneZSpec() {
00070   unsigned char i, aux, z = 2;
00071   Voxel v = {ON};
00072   cube->useFrontBuffer();
00073   cube->clear();
00074   cube->writePlaneZ(z, v);
00075   aux = 0xff;
00076   for (i = 0; i < Cube::SIZE; i++) {
00077     aux &= AT(cube->frontBuffer, i, z);
00078   }
00079   Asserter::assertEqual(aux, 0xff, "writePlaneZSpec: should write all z axis LEDs.");
00080 }
00081
00082 void CubeSpec::writePlaneYSpec() {
00083   unsigned char z, aux, y = 3;
00084   Voxel v = {ON};
00085   cube->useFrontBuffer();
00086   cube->clear();
```

```
00087   cube->writePlaneY(y, v);
00088   aux = 0xff;
00089   for (z = 0; z < Cube::SIZE; z++) {
00090     aux &= AT(cube->frontBuffer, y, z);
00091   }
00092   Asserter::assertEqual(aux, 0xff, "writePlaneXSpec: should write all y axis LEDs.");
00093 }
00094
00095 void CubeSpec::writePlaneXSpec() {
00096   unsigned char z, y, aux, x = 3;
00097   Voxel v = {ON};
00098   cube->useFrontBuffer();
00099   cube->clear();
00100   cube->writePlaneX(x, v);
00101   aux = 0x01 << x;
00102   for (z = 0; z < Cube::SIZE; z++) {
00103     for (y = 0; y < Cube::SIZE; y++) {
00104       aux |= AT(cube->frontBuffer, y, z) & (0x01 << x);
00105     }
00106   }
00107   Asserter::assertEqual(aux, (0x01 << x), "writePlaneXSpec: should write all x axis
       LEDs.");
00108 }
00109
00110 void CubeSpec::flipByteSpec() {
00111   unsigned char b = 0xab;
00112   Util::flipByte(&b);
00113   Asserter::assertEqual(b, 0xd5, "flipByteSpec: should flip the byte.");
00114 }
00115
00116 void CubeSpec::mirrorXSpec() {
00117   unsigned char aux, z, y;
00118   cube->useFrontBuffer();
00119   cube->fill(0xd5);
00120   cube->mirrorX();
00121   aux = 0xab;
00122   for (z = 0; z < Cube::SIZE; z++) {
00123     for (y = 0; y < Cube::SIZE; y++) {
00124       aux &= AT(cube->frontBuffer, y, z);
00125     }
00126   }
00127   Asserter::assertEqual(aux, 0xab, "mirrorX: should mirror X.");
00128 }
00129
00130 void CubeSpec::mirrorYSpec() {
00131   unsigned char aux = 0xaa;
00132   cube->useFrontBuffer();
00133   cube->clear();
00134   AT(cube->frontBuffer, 0, 0) = aux;
00135   cube->mirrorY();
00136   Asserter::assertEqual(AT(cube->frontBuffer,
      Cube::SIZE - 1, 0), aux, "mirrorY: should mirror Y.");
00137 }
00138
00139 void CubeSpec::mirrorZSpec() {
00140   unsigned char aux = 0xaa;
00141   cube->useFrontBuffer();
00142   cube->clear();
00143   AT(cube->frontBuffer, 0, 0) = aux;
00144   cube->mirrorZ();
00145   Asserter::assertEqual(AT(cube->frontBuffer, 0,
      Cube::SIZE - 1), aux, "mirrorZ: should mirror Z.");
00146   cube->clear();
00147   cube->useBackBuffer();
00148   cube->clear();
00149   AT(cube->backBuffer, 0, 0) = aux;
00150   cube->mirrorZ();
00151   Asserter::assertEqual(AT(cube->frontBuffer, 0,
      Cube::SIZE - 1), 0, "mirrorZ: should keep frontBuffer empty when using backBuffer when mirroring
       Z.");
00152   Asserter::assertEqual(AT(cube->backBuffer, 0,
      Cube::SIZE - 1), aux, "mirrorZ: should keep frontBuffer empty when using backBuffer when
       mirroring Z.");
00153 }
00154
00155 void CubeSpec::filledBoxSpec() {
00156   unsigned char z, y, aux = 0xff;
00157   Point p0 = Point(0, 0, 0);
00158   Point p1  = Point(2, 5, 5);
00159   cube->useFrontBuffer();
00160   cube->clear();
00161   cube->filledBox(&p0, &p1);
00162   for (z = 0; z < 6; z++) {
00163     for (y = 0; y < 6; y++) {
00164       aux &= AT(cube->frontBuffer, y, z);
00165     }
00166   }
```

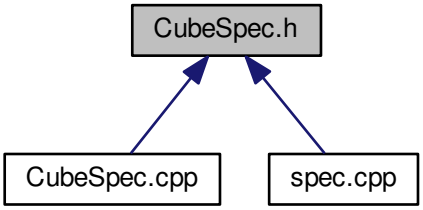
```
00167   Asserter::assertEqual(aux, 0x07, "filledBox: test#1 should draw a filled box.");
00168   p0.x = 1;
00169   p1.z = 2;
00170   aux = 0xff;
00171   cube->clear();
00172   cube->filledBox(&p0, &p1);
00173   for (z = 0; z < 3; z++) {
00174     for (y = 0; y < 6; y++) {
00175       aux &= AT(cube->frontBuffer, y, z);
00176     }
00177   }
00178   Asserter::assertEqual(aux, 0x06, "filledBox: test#2 should draw a filled box.");
00179 }
00180
00181 void CubeSpec::lineSpec() {
00182   unsigned char z, y, sum = 0;
00183   Point from = Point(0, 0, 0);
00184   Point to = Point(7, 7, 7);
00185   cube->useFrontBuffer();
00186   cube->clear();
00187   cube->line(&from, &to);
00188   for (z = 0; z < Cube::SIZE; z++) {
00189     for (y = 0; y < Cube::SIZE; y++) {
00190       sum += AT(cube->frontBuffer, y, z);
00191     }
00192   }
00193   Asserter::assertEqual(sum, 0xff, "line: should draw a line.");
00194 }
00195
00196 void CubeSpec::shiftOnXSpec() {
00197   unsigned char x = 0, y = 0, z = 7, aux;
00198   Point p = Point(x, y, z);
00199   cube->useFrontBuffer();
00200   cube->clear();
00201   cube->turnVoxelOn(&p);
00202   aux = AT(cube->frontBuffer, y, z);
00203   cube->shiftOnX(Direction::RIGHT);
00204   Asserter::assertEqual(AT(cube->frontBuffer, y, z),
      Util::rotatingShift(aux, false), "shiftOnXSpec: should shiftOnXSpec RIGHT.");
00205   aux = AT(cube->frontBuffer, y, z);
00206   cube->shiftOnX(Direction::LEFT);
00207   Asserter::assertEqual(AT(cube->frontBuffer, y, z),
      Util::rotatingShift(aux, true), "shiftOnXSpec: should shiftOnXSpec LEFT.");
00208 }
00209
00210 void CubeSpec::shiftOnYSpec() {
00211   cube->useFrontBuffer();
00212   cube->clear();
00213   AT(cube->frontBuffer, 7, 0) = 0xff;
00214   cube->shiftOnY(Direction::BACK);
00215   Asserter::assertEqual(AT(cube->frontBuffer, 7, 0), 0x00, "
      shiftOnYSpec: should shiftOnYSpec case 1.");
00216   Asserter::assertEqual(AT(cube->frontBuffer, 6, 0), 0xff, "
      shiftOnYSpec: should shiftOnYSpec case 2.");
00217   cube->shiftOnY(Direction::FRONT);
00218   Asserter::assertEqual(AT(cube->frontBuffer, 7, 0), 0xff, "
      shiftOnYSpec: should shiftOnYSpec case 3.");
00219   Asserter::assertEqual(AT(cube->frontBuffer, 6, 0), 0x00, "
      shiftOnYSpec: should shiftOnYSpec case 4.");
00220 }
00221
00222 void CubeSpec::shiftOnZSpec() {
00223   unsigned char x = 0, y = 0, z = 7;
00224   Point p = Point(x, y, z);
00225   cube->useFrontBuffer();
00226   cube->clear();
00227   cube->turnVoxelOn(&p);
00228   cube->shiftOnZ(Direction::UP);
00229   Asserter::assertEqual(AT(cube->frontBuffer, y, (z + 1) %
      Cube::SIZE), 0x01 << x, "shiftOnZSpec: should shiftOnZSpec UP.");
00230   cube->shiftOnZ(Direction::DOWN);
00231   Asserter::assertEqual(AT(cube->frontBuffer, y, z), 0x01 << x, "
      shiftOnZSpec: should shiftOnZSpec UP.");
00232 }
```

5.51 CubeSpec.h File Reference

`#include <Cube.h>`
Include dependency graph for CubeSpec.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CubeSpec

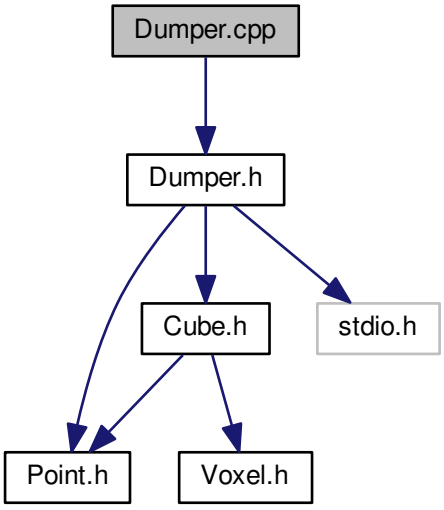
5.52 CubeSpec.h

```
00001
00004 #ifndef __ARDUINO_CUBE_CUBE_TEST_H__
00005 #define __ARDUINO_CUBE_CUBE_TEST_H__ 1
00006
00007 #include <Cube.h>
00008
00009 class CubeSpec {
00010
00011 public:
00012
00013   Cube *cube;
00014
00015   CubeSpec(Cube *cube);
```

```
00016
00017   void run();
00018
00019   void isInRangeSpec();
00020
00021   void writeVoxelSpec();
00022
00023   void invertVoxelSpec();
00024
00025   void writePlaneZSpec();
00026
00027   void writePlaneYSpec();
00028
00029   void writePlaneXSpec();
00030
00031   void flipByteSpec();
00032
00033   void mirrorXSpec();
00034
00035   void mirrorYSpec();
00036
00037   void mirrorZSpec();
00038
00039   void filledBoxSpec();
00040
00041   void lineSpec();
00042
00043   void shiftOnXSpec();
00044
00045   void shiftOnYSpec();
00046
00047   void shiftOnZSpec();
00048 };
00049
00050 #endif /* __ARDUINO_CUBE_CUBE_TEST_H__ */
```

5.53 Dumper.cpp File Reference

`#include <Dumper.h>`
Include dependency graph for Dumper.cpp:



Macros

- #define __ARDUINO_CUBE_DUMPER_CPP__ 1

5.53.1 Macro Definition Documentation

5.53.1.1 #define __ARDUINO_CUBE_DUMPER_CPP__ 1

Definition at line 5 of file Dumper.cpp.

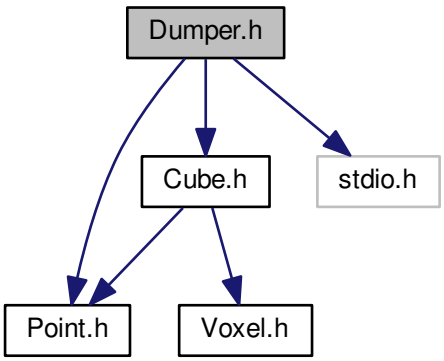
5.54 Dumper.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_DUMPER_CPP__
00005 #define __ARDUINO_CUBE_DUMPER_CPP__ 1
00006
00007 #include <Dumper.h>
00008
00009 void Dumper::dumpPoint(Point *point) {
00010   printf("Point {\n  x: %d,\n  y: %d,\n  z: %d\n}\n", point->x, point->y, point->
      z);
00011 }
00012
00013 void Dumper::dumpCube(Cube *cube) {
00014   unsigned char z, y, *buffer;
00015   buffer = cube->frontBuffer;
00016   for (z = 0; z < Cube::SIZE; z++) {
00017     for (y = 0; y < Cube::SIZE; y++) {
00018       printf("[%d,%d]%02x ", z, y, *(buffer + (Cube::SIZE * z) + y));
00019     }
00020     printf("\n");
00021   }
00022   printf("\n");
00023 }
00024
00025 #endif /* __ARDUINO_CUBE_DUMPER_CPP__ */
00026
```

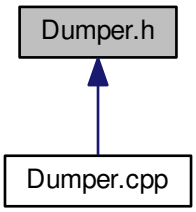
5.55 Dumper.h File Reference

```
#include <Point.h>
#include <Cube.h>
#include <stdio.h>
```

Include dependency graph for Dumper.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Dumper

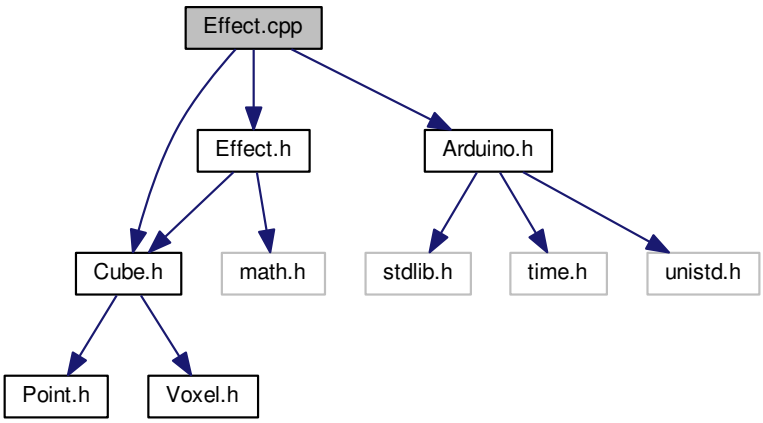
5.56 Dumper.h

```
00001
00004 #ifndef __ARDUINO_CUBE_DUMPER_H__
00005 #define __ARDUINO_CUBE_DUMPER_H__ 1
00006
00007 #include <Point.h>
00008 #include <Cube.h>
00009 #include <stdio.h>
00010
00011 class Dumper {
00012
00013 public:
00014
00015   static void dumpPoint(Point *point);
00016
00017   static void dumpCube(Cube *cube);
00018 };
00019
00020 #endif /* __ARDUINO_CUBE_DUMPER_H__ */
```

5.57 Effect.cpp File Reference

```
#include <Effect.h>
#include <Cube.h>
#include <Arduino.h>
```

Include dependency graph for Effect.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_EFFECT_CPP__ 1

5.57.1 Macro Definition Documentation

5.57.1.1 #define __ARDUINO_CUBE_EFFECTS_EFFECT_CPP__ 1

Definition at line 5 of file Effect.cpp.

5.58 Effect.cpp

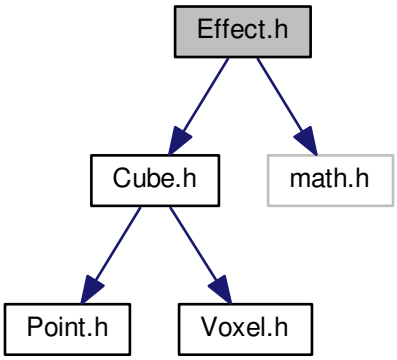
```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_EFFECT_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_EFFECT_CPP__ 1
00006
00007 #include <Effect.h>
00008 #include <Cube.h>
00009 #include <Arduino.h>
00010
00011 Effect::Effect(Cube *cube, unsigned int iterations, unsigned int iterationDelay) :
00012     cube(cube), iterations(iterations), iterationDelay(iterationDelay) {
00013 }
00014
00015 void Effect::run() {
00016 }
00017
00018 void Effect::sendVoxel(Point *p, Direction dir, unsigned int stepDelay) {
00019   Voxel origin, current;
00020   char *dim, inc = 1;
00021   cube->readVoxel(p, &origin);
00022   switch (dir) {
00023     case UP:
00024       inc = -1;
00025     case DOWN:
00026       dim = (char*)&(p->z);
00027       break;
00028     case FRONT:
00029       inc = -1;
00030     case BACK:
00031       dim = (char*)&(p->y);
00032       break;
00033     case RIGHT:
00034       inc = -1;
00035     case LEFT:
```

```
00036       dim = (char*)&(p->x);
00037       break;
00038   }
00039   for (; (inc > 0) ? *dim < Cube::SIZE : *dim >= 0; *dim += inc) {
00040     cube->readVoxel(p, &current);
00041     cube->writeVoxel(p, origin);
00042     delay(stepDelay);
00043     cube->writeVoxel(p, current);
00044   }
00045 }
00046
00047 #endif /* __ARDUINO_CUBE_EFFECTS_EFFECT_CPP__ */
```

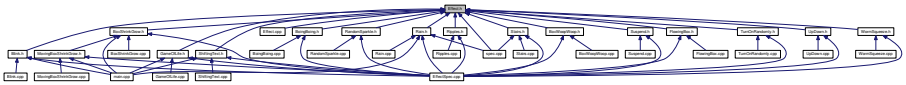
5.59 Effect.h File Reference

```
#include <Cube.h>
#include <math.h>
```

Include dependency graph for Effect.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Effect

5.60 Effect.h

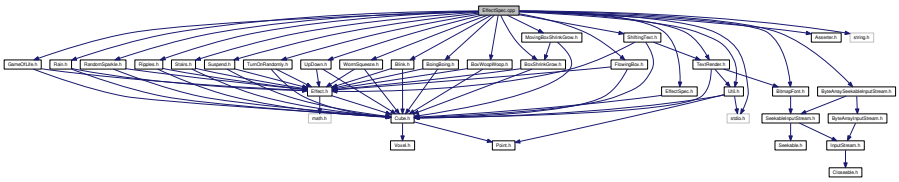
```
00001
00005 #ifndef __ARDUINO_CUBE_EFFECTS_EFFECT_H__
00006 #define __ARDUINO_CUBE_EFFECTS_EFFECT_H__ 1
00007
00008 #include <Cube.h>
00009 #include <math.h>
00010
00011 class Effect {
00012
00013 public:
```

```
00014
00015   unsigned int iterations;
00016   unsigned int iterationDelay;
00017   Cube *cube;
00018
00019   Effect(Cube *cube, unsigned int iterations, unsigned int iterationDelay);
00020
00021   virtual void run();
00022
00023   void sendVoxel(Point *origin, Direction direction, unsigned int stepDelay);
00024 };
00025
00026 #endif /* __ARDUINO_CUBE_EFFECTS_EFFECT_H__ */
```

5.61 EffectSpec.cpp File Reference

```
#include <EffectSpec.h>
#include <Asserter.h>
#include <Util.h>
#include <Blink.h>
#include <BoingBoing.h>
#include <BoxShrinkGrow.h>
#include <BoxWoopWoop.h>
#include <FlowingBox.h>
#include <GameOfLife.h>
#include <MovingBoxShrinkGrow.h>
#include <Rain.h>
#include <RandomSparkle.h>
#include <Ripples.h>
#include <Stairs.h>
#include <Suspend.h>
#include <TurnOnRandomly.h>
#include <UpDown.h>
#include <WormSqueeze.h>
#include <ShiftingText.h>
#include <stdio.h>
#include <string.h>
#include <ByteArraySeekableInputStream.h>
#include <BitmapFont.h>
#include <TextRender.h>
```

Include dependency graph for EffectSpec.cpp:



Macros

- #define AT(b, y, z) *(b + ((((z) * CUBE_SIZE) + (y))))

5.61.1 Macro Definition Documentation

5.61.1.1 #define AT(*b*, *y*, *z*) *(b + ((((z) * CUBE_SIZE) + (y))))

Definition at line 26 of file EffectSpec.cpp.

5.62 EffectSpec.cpp

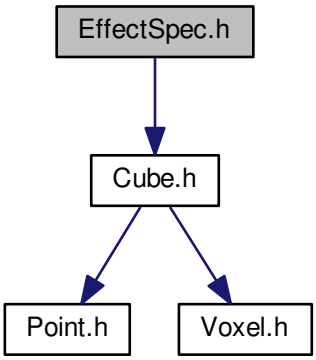
```
00001 #include <EffectSpec.h>
00002 #include <Asserter.h>
00003 #include <Util.h>
00004 #include <Blink.h>
00005 #include <BoingBoing.h>
00006 #include <BoxShrinkGrow.h>
00007 #include <BoxWoopWoop.h>
00008 #include <FlowingBox.h>
00009 #include <GameOfLife.h>
00010 #include <MovingBoxShrinkGrow.h>
00011 #include <Rain.h>
00012 #include <RandomSparkle.h>
00013 #include <Ripples.h>
00014 #include <Stairs.h>
00015 #include <Suspend.h>
00016 #include <TurnOnRandomly.h>
00017 #include <UpDown.h>
00018 #include <WormSqueeze.h>
00019 #include <ShiftingText.h>
00020 #include <stdio.h>
00021 #include <string.h>
00022 #include <ByteArraySeekableInputStream.h>
00023 #include <BitmapFont.h>
00024 #include <TextRender.h>
00025
00026 #define AT(b, y, z) *(b + ((((z) * CUBE_SIZE) + (y))))
00027
00028 EffectSpec::EffectSpec(Cube *cube) : cube(cube) {
00029 }
00030
00031 void EffectSpec::run() {
00032   selfSpec();
00033   rainSpec();
00034   blinkSpec();
00035   boingBoingSpec();
00036   boxShrinkGrowSpec();
00037   boxWoopWoopSpec();
00038   flowingBoxSpec();
00039   gameOfLifeSpec();
00040   movingBoxShrinkGrowSpec();
00041   randomSparkleSpec();
00042   ripplesSpec();
00043   stairsSpec();
00044   suspendSpec();
00045   upDownSpec();
00046   wormSqueezeSpec();
00047   turnOnRandomlySpec();
00048   shiftingTextSpec();
00049 }
00050
00051 void EffectSpec::selfSpec() {
00052   Point p = Point();
00053   Effect effect = Effect(cube, 1, 0);
00054   cube->clear();
00055   effect.sendVoxel(&p, FRONT, 0);
00056   Asserter::assertEqual(0, 0, "selfSpec: It should not break.");
00057 }
00058
00059 void EffectSpec::blinkSpec() {
00060   Blink effect = Blink(cube, 1, 0);
00061   effect.run();
00062   Asserter::assertEqual(0, 0, "blinkSpec: It should not break, we cannot test it.");
00063 }
00064
00065 void EffectSpec::rainSpec() {
00066   unsigned char i, aux = 0;
00067   Rain effect = Rain(cube, 1, 0, 1, 8);
00068   cube->clear();
00069   effect.run();
00070   for (i = 0; i < Cube::SIZE; i++) {
00071     aux += AT(cube->frontBuffer, i, Cube::SIZE - 2);
00072   }
00073   Asserter::assertNotEqual(aux, 0, "rainSpec: It should have turned on some LEDs.")
    ;
00074   aux = 0;
00075   for (i = 0; i < Cube::SIZE; i++) {
00076     aux += AT(cube->frontBuffer, i, Cube::SIZE - 1);
00077   }
00078   Asserter::assertEqual(aux, 0, "rainSpec: It should clear the top plane.");
00079 }
00080
00081 void EffectSpec::boingBoingSpec() {
00082   BoingBoing effect = BoingBoing(cube, 1, 0);
00083   effect.run();
00084   Asserter::assertEqual(0, 0, "BoingBoing: It should not break, we cannot test it.");
```

```
00085 }
00086
00087 void EffectSpec::boxShrinkGrowSpec() {
00088   BoxShrinkGrow effect = BoxShrinkGrow(cube, 1, 0,
      BoxShrinkGrow::WALL);
00089   effect.run();
00090   Asserter::assertEqual(0, 0, "BoxShrinkGrow: It should not break, we cannot test it."
      );
00091 }
00092
00093 void EffectSpec::boxWoopWoopSpec() {
00094   BoxWoopWoop effect = BoxWoopWoop(cube, 1, 0);
00095   effect.run();
00096   Asserter::assertEqual(0, 0, "BoxWoopWoop: It should not break, we cannot test it.");
00097 }
00098
00099 void EffectSpec::flowingBoxSpec() {
00100   FlowingBox effect = FlowingBox(cube, 1, 0);
00101   effect.run();
00102   Asserter::assertEqual(0, 0, "FlowingBox: It should not break, we cannot test it.");
00103 }
00104
00105 void EffectSpec::gameOfLifeSpec() {
00106   GameOfLife effect = GameOfLife(cube, 1, 0, 8);
00107   effect.run();
00108   Asserter::assertEqual(0, 0, "GameOfLife: It should not break, we cannot test it.");
00109 }
00110
00111 void EffectSpec::movingBoxShrinkGrowSpec() {
00112   MovingBoxShrinkGrow effect = MovingBoxShrinkGrow(
      cube, 1, 0, BoxShrinkGrow::WALL);
00113   effect.run();
00114   Asserter::assertEqual(0, 0, "MovingBoxShrinkGrow: It should not break, we cannot
       test it.");
00115 }
00116
00117 void EffectSpec::randomSparkleSpec() {
00118   RandomSparkle effect = RandomSparkle(cube, 1, 0);
00119   effect.run();
00120   Asserter::assertEqual(0, 0, "RandomSparkle: It should not break, we cannot test it."
      );
00121 }
00122
00123 void EffectSpec::ripplesSpec() {
00124   Ripples effect = Ripples(cube, 1, 0);
00125   effect.run();
00126   Asserter::assertEqual(0, 0, "Ripples: It should not break, we cannot test it.");
00127 }
00128
00129 void EffectSpec::stairsSpec() {
00130   Stairs effect = Stairs(cube, 1, 0);
00131   effect.run();
00132   Asserter::assertEqual(0, 0, "Stairs: It should not break, we cannot test it.");
00133 }
00134
00135 void EffectSpec::suspendSpec() {
00136   Suspend effect = Suspend(cube, 1, 0);
00137   effect.run();
00138   Asserter::assertEqual(0, 0, "Suspend: It should not break, we cannot test it.");
00139 }
00140
00141 void EffectSpec::upDownSpec() {
00142   UpDown effect = UpDown(cube, 1, 0, Axis::AXIS_X, 0);
00143   effect.run();
00144   Asserter::assertEqual(0, 0, "upDownSpec: It should not break, we cannot test it.");
00145 }
00146
00147 void EffectSpec::wormSqueezeSpec() {
00148   WormSqueeze effect = WormSqueeze(cube, 1, 0);
00149   effect.run();
00150   Asserter::assertEqual(0, 0, "WormSqueeze: It should not break, we cannot test it.");
00151 }
00152
00153 void EffectSpec::turnOnRandomlySpec() {
00154   TurnOnRandomly effect = TurnOnRandomly(cube, 1, 0, 8);
00155   effect.run();
00156   Asserter::assertEqual(0, 0, "turnOnRandomlySpec: It should not break, we cannot test
       it.");
00157 }
00158
00159 void EffectSpec::shiftingTextSpec() {
00160   unsigned char fontStream[] = {
00161     0x0, 0x5, 0x8, 0x2, 0x48, 0x48, 0x0, 0xc, 0x50, 0x50, 0x0, 0x11,
00162     0x7f, 0x8, 0x8, 0x8, 0x7f, 0x7f, 0x9, 0x9, 0x9, 0x6
00163   };
00164   ByteArraySeekableInputStream is =
      ByteArraySeekableInputStream(&fontStream[0], (unsigned int) sizeof(fontStream))
```

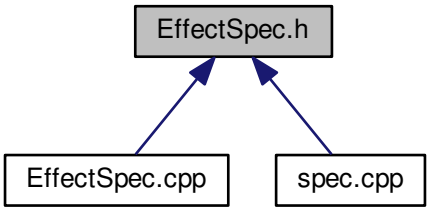
```
    ;
00165   BitmapFont font = BitmapFont(&is);
00166   TextRender render = TextRender(cube, &font);
00167   Point p = Point(0, 0, 0);
00168   ShiftingText::ShiftingTextSettings settings = {&
      render, (const char *) "HP", 2, TextRender::XYZ, &p};
00169   ShiftingText effect = ShiftingText(cube, 1, 0, &settings);
00170   effect.run();
00171   Asserter::assertEqual(0, 0, "shiftingTextSpec: It should not break, we cannot test
       it.");
00172 }
```

5.63 EffectSpec.h File Reference

`#include <Cube.h>`
Include dependency graph for EffectSpec.h:



This graph shows which files directly or indirectly include this file:



Classes

- class EffectSpec

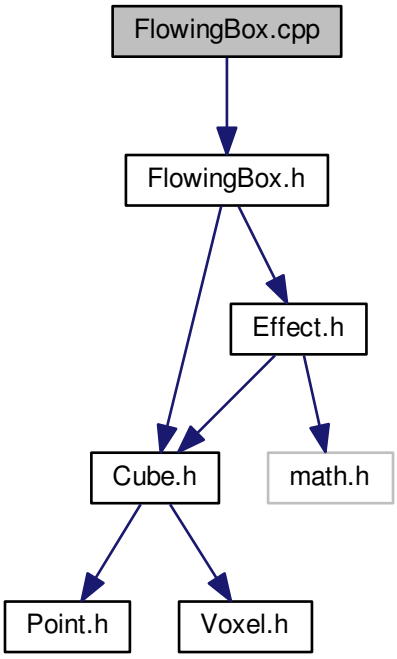
5.64 EffectSpec.h

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECT_TEST_H__
00005 #define __ARDUINO_CUBE_EFFECT_TEST_H__ 1
00006
00007 #include <Cube.h>
00008
00009 class EffectSpec {
00010
00011 public:
00012
00013   Cube *cube;
00014
00015   EffectSpec(Cube *cube);
00016
00017   void run();
00018
00019   void selfSpec();
00020
00021   void rainSpec();
00022
00023   void blinkSpec();
00024
00025   void boingBoingSpec();
00026
00027   void boxShrinkGrowSpec();
00028
00029   void boxWoopWoopSpec();
00030
00031   void flowingBoxSpec();
00032
00033   void gameOfLifeSpec();
00034
00035   void movingBoxShrinkGrowSpec();
00036
00037   void randomSparkleSpec();
00038
00039   void ripplesSpec();
00040
00041   void stairsSpec();
00042
00043   void suspendSpec();
00044
00045   void upDownSpec();
00046
00047   void wormSqueezeSpec();
00048
00049   void turnOnRandomlySpec();
00050
00051   void shiftingTextSpec();
00052 };
00053
00054 #endif /* __ARDUINO_CUBE_EFFECT_TEST_H__ */
```

5.65 FlowingBox.cpp File Reference

```
#include <FlowingBox.h>
```

Include dependency graph for FlowingBox.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_FLOWING_BOX_CPP__ 1

5.65.1 Macro Definition Documentation

5.65.1.1 #define __ARDUINO_CUBE_EFFECTS_FLOWING_BOX_CPP__ 1

Definition at line 5 of file FlowingBox.cpp.

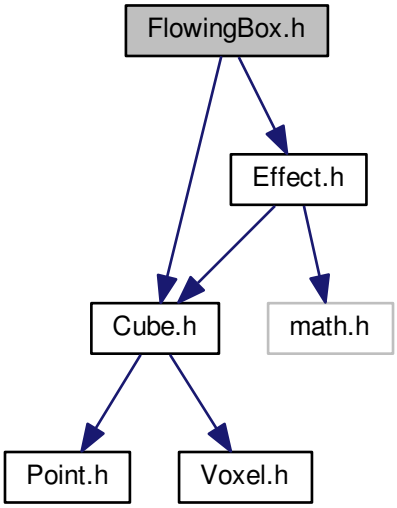
5.66 FlowingBox.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_FLOWING_BOX_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_FLOWING_BOX_CPP__ 1
00006
00007 #include <FlowingBox.h>
00008
00009 FlowingBox::FlowingBox(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay) :
00010     Effect(cube, iterations, iterationDelay) {
00011 }
00012
00013 void FlowingBox::run() {
00014   unsigned int iteration;
00015   for (iteration = 0; iteration < iterations; iteration++) {
00016   }
00017 }
00018
00019 #endif /* __ARDUINO_CUBE_EFFECTS_FLOWING_BOX_CPP__ */
```

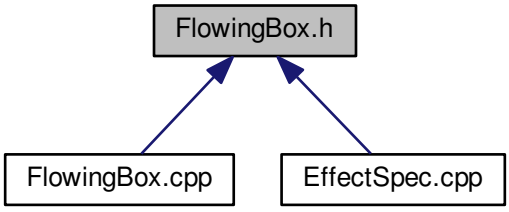
5.67 FlowingBox.h File Reference

```
#include <Effect.h>
#include <Cube.h>
```

Include dependency graph for FlowingBox.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FlowingBox

5.68 FlowingBox.h

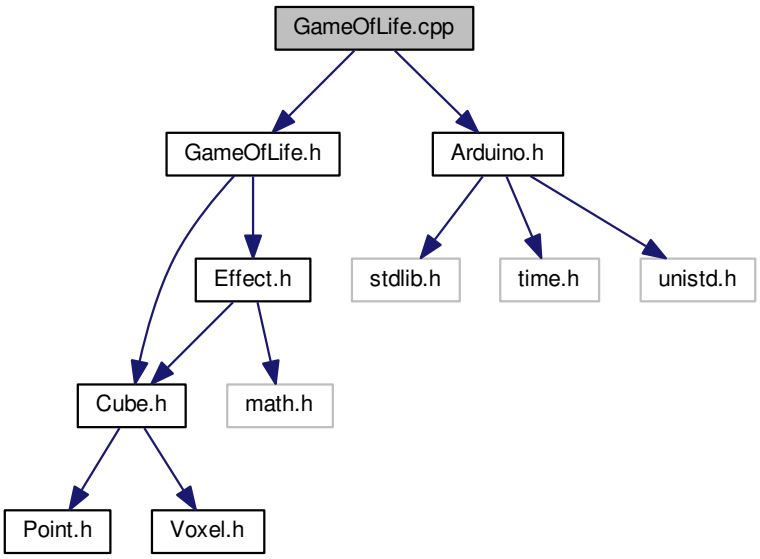
```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_FLOWING_BOX_H__
00005 #define __ARDUINO_CUBE_EFFECTS_FLOWING_BOX_H__ 1
00006
```

```
00007 #include <Effect.h>
00008 #include <Cube.h>
00009
00010 class FlowingBox : public Effect {
00011 public:
00012
00013   FlowingBox(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay);
00014
00015   virtual void run();
00016 };
00017
00018 #endif /* __ARDUINO_CUBE_EFFECTS_FLOWING_BOX_H__ */
```

5.69 GameOfLife.cpp File Reference

```
#include <GameOfLife.h>
#include <Arduino.h>
```

Include dependency graph for GameOfLife.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_GAME_OF_LIFE_CPP__ 1

5.69.1 Macro Definition Documentation

5.69.1.1 #define __ARDUINO_CUBE_EFFECTS_GAME_OF_LIFE_CPP__ 1

Definition at line 5 of file GameOfLife.cpp.

5.70 GameOfLife.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_GAME_OF_LIFE_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_GAME_OF_LIFE_CPP__ 1
```

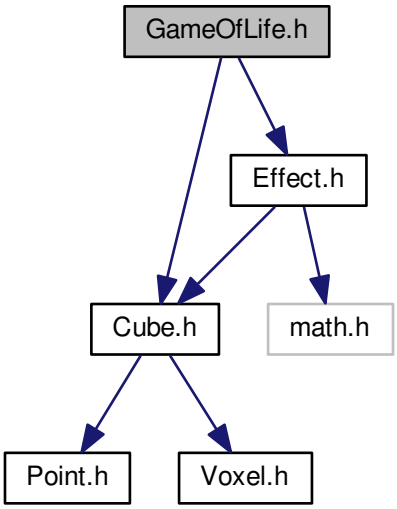
```
00006
00007 #include <GameOfLife.h>
00008 #include <Arduino.h>
00009
00010 GameOfLife::GameOfLife(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay, unsigned char firstGenerationSize) :
00011     Effect(cube, iterations, iterationDelay), firstGenerationSize(firstGenerationSize) {
00012 }
00013
00014 void GameOfLife::run() {
00015   unsigned int iteration;
00016   genesis();
00017   for (iteration = 0; iteration < iterations; iteration++) {
00018     nextGeneration();
00019     if (!hasChanges()) {
00020       return;
00021     }
00022         delay(iterationDelay);
00023   }
00024 }
00025
00026 void GameOfLife::genesis() {
00027   unsigned char i;
00028   Point p;
00029   cube->clear();
00030   for (i = 0; i < firstGenerationSize; i++) {
00031     p.randomize(Cube::SIZE);
00032     cube->turnVoxelOn(&p);
00033   }
00034 }
00035
00036 void GameOfLife::nextGeneration() {
00037     unsigned char neighbors;
00038     Point p;
00039     Voxel v;
00040     for (p.z = 0; p.z < Cube::SIZE; p.z++) {
00041     for (p.y = 0; p.y < Cube::SIZE; p.y++) {
00042       for (p.x = 0; p.x < Cube::SIZE; p.x++) {
00043                 neighbors = getNeighbors(&p);
00044         cube->readVoxel(&p, &v);
00045         if (v.state == ON) {
00046                     if (neighbors <= LONELY_DEATH || neighbors >= CROWDED_DEATH) {
00047                         cube->turnVoxelOff(&p);
00048                     }
00049                 } else {
00050                     if (neighbors >= CREATE_MIN && neighbors <=
      CREATE_MAX) {
00051                         cube->turnVoxelOn(&p);
00052           }
00053                 }
00054             }
00055         }
00056     }
00057 }
00058
00059 unsigned char GameOfLife::getNeighbors(Point *at) {
00060   unsigned char neighbors = 0;
00061   Point p;
00062   Voxel v;
00063   cube->fitInRange(&p);
00064   for (p.z = at->z - 1; p.z <= at->z + 1; p.z++) {
00065     for (p.y = at->y - 1; p.y <= at->y + 1; p.y++) {
00066       for (p.x = at->x - 1; p.x <= at->x + 1; p.x++) {
00067         if (cube->isInRange(&p)) {
00068           cube->readVoxel(&p, &v);
00069           if ((p.z != 0 || p.y != 0 || p.x != 0) && (v.state == ON)) {
00070             neighbors++;
00071           }
00072         }
00073       }
00074     }
00075   }
00076   return neighbors;
00077 }
00078
00079 bool GameOfLife::hasChanges() {
00080     unsigned char z, y;
00081     for (z = 0; z < Cube::SIZE; z++) {
00082         for (y = 0; y < Cube::SIZE; y++) {
00083           unsigned char pos = z * Cube::SIZE + y;
00084             if (*(cube->frontBuffer + pos) != *(cube->
      backBuffer + pos)) {
00085                 return true;
00086       }
00087         }
00088     }
00089     return false;
```

```
00090 }
00091
00092 #endif /* __ARDUINO_CUBE_EFFECTS_GAME_OF_LIFE_CPP__ */
```

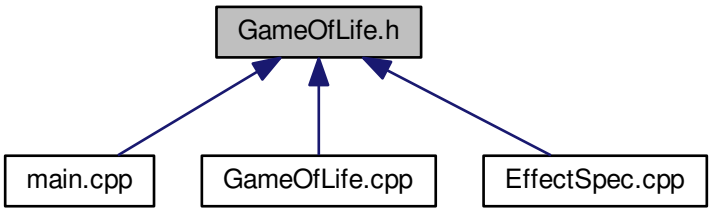
5.71 GameOfLife.h File Reference

```
#include <Effect.h>
#include <Cube.h>
```

Include dependency graph for GameOfLife.h:



This graph shows which files directly or indirectly include this file:



Classes

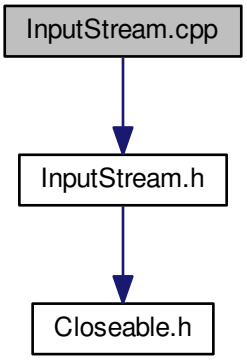
- class GameOfLife

5.72 GameOfLife.h

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_GAME_OF_LIFE_H__
00005 #define __ARDUINO_CUBE_EFFECTS_GAME_OF_LIFE_H__ 1
00006
00007 #include <Effect.h>
00008 #include <Cube.h>
00009
00010 class GameOfLife : public Effect {
00011
00012   unsigned char firstGenerationSize;
00013
00014 public:
00015
00016   static const unsigned char LONELY_DEATH = 1;
00017   static const unsigned char CROWDED_DEATH = 4;
00018   static const unsigned char CREATE_MIN = 3;
00019   static const unsigned char CREATE_MAX = 3;
00020
00021   GameOfLife(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay, unsigned char firstGenerationSize);
00022
00023   virtual void run();
00024
00025   void genesis();
00026
00027   unsigned char getNeighbors(Point *p);
00028
00029   void nextGeneration();
00030
00031   bool hasChanges();
00032 };
00033
00034 #endif /* __ARDUINO_CUBE_EFFECTS_GAME_OF_LIFE_H__ */
```

5.73 InputStream.cpp File Reference

`#include "InputStream.h"`
Include dependency graph for InputStream.cpp:



Macros

- #define __ARDUINO_IO_INPUT_STREAM_CPP__ 1

5.73.1 Macro Definition Documentation

5.73.1.1 #define __ARDUINO_IO_INPUT_STREAM_CPP__ 1

Arduino IO.

InputStream

This abstract class is the superclass of all classes representing an input stream of bytes.

Applications that need to define a subclass of InputStream must always provide a method that returns the next unsigned char of input.

Definition at line 14 of file InputStream.cpp.

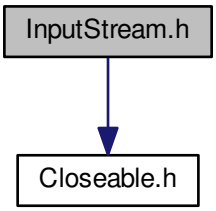
5.74 InputStream.cpp

```
00001
00013 #ifndef __ARDUINO_IO_INPUT_STREAM_CPP__
00014 #define __ARDUINO_IO_INPUT_STREAM_CPP__ 1
00015
00016 #include "InputStream.h"
00017
00018 int InputStream::available() {
00019     return 0;
00020 }
00021
00022 void InputStream::close() {
00023 }
00024
00025 void InputStream::mark() {
00026 }
00027
00028 bool InputStream::markSupported() {
00029     return false;
00030 }
00031
00032 int InputStream::read(unsigned char* b, int len) {
00033     return read(b, 0, len);
00034 }
00035
00036 int InputStream::read(unsigned char* b, int off, int len) {
00037     int i, c;
00038     if (b == (unsigned char*) 0) {
00039         return 0;
00040     }
00041     c = read();
00042     if (c == -1) {
00043         return -1;
00044     }
00045     b[off] = (unsigned char) c;
00046     for (i = 1; i < len; i++) {
00047         c = read();
00048         if (c == -1) {
00049             break;
00050         }
00051         b[off + i] = (unsigned char) c;
00052     }
00053     return i;
00054 }
00055
00056 void InputStream::reset() {
00057 }
00058
00059 unsigned int InputStream::skip(unsigned int n) {
00060     unsigned int i;
00061     for (i = 0; i < n && available() > 0; i++) {
00062         read();
00063     }
00064     return i;
00065 }
00066
00067 #endif /* __ARDUINO_IO_INPUT_STREAM_CPP__ */
```

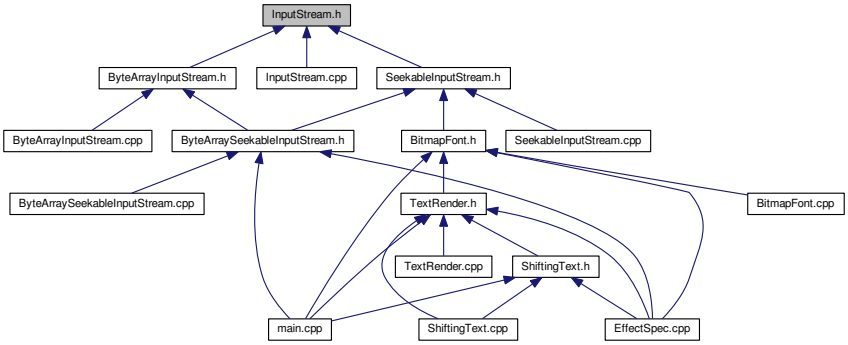
5.75 InputStream.h File Reference

```
#include <Closeable.h>
```

Include dependency graph for InputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class InputStream

5.76 InputStream.h

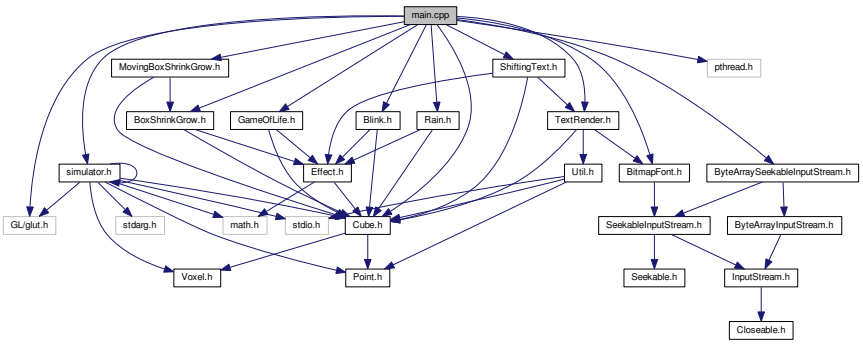
```
00001
00013 #ifndef __ARDUINO_IO_INPUT_STREAM_H__
00014 #define __ARDUINO_IO_INPUT_STREAM_H__ 1
00015
00016 #include <Closeable.h>
00017
00018 class InputStream : public Closeable {
00019 public:
00020
00026     virtual int available();
00027
00032     virtual void close();
00033
00037     virtual void mark();
00038
00042     virtual bool markSupported();
00043
00047     virtual int read() = 0;
00048
00053     virtual int read(unsigned char* b, int len);
00054
00063     virtual int read(unsigned char* b, int off, int len);
00064
00069     virtual void reset();
```

```
00070
00074     virtual unsigned int skip(unsigned int n);
00075 };
00076
00077 #endif /* __ARDUINO_IO_INPUT_STREAM_H__ */
```

5.77 main.cpp File Reference

```
#include <GL/glut.h>
#include <pthread.h>
#include <Cube.h>
#include <Blink.h>
#include <Rain.h>
#include <BoxShrinkGrow.h>
#include <MovingBoxShrinkGrow.h>
#include <GameOfLife.h>
#include <ByteArraySeekableInputStream.h>
#include <BitmapFont.h>
#include <TextRender.h>
#include <ShiftingText.h>
#include "simulator.h"
```

Include dependency graph for main.cpp:



Functions

- void ∗ effect_runner (void ∗arg)
- int main (int argc, char ∗argv[ ])

Variables

- Cube cube = Cube()

5.77.1 Function Documentation

5.77.1.1 void∗ effect_runner (void ∗ *arg*)

Definition at line 20 of file main.cpp.

5.77.1.2 int main (int *argc*, char ∗ *argv[ ]*)

Definition at line 42 of file main.cpp.

5.77.2 Variable Documentation

5.77.2.1 Cube cube = Cube()

Definition at line 18 of file main.cpp.

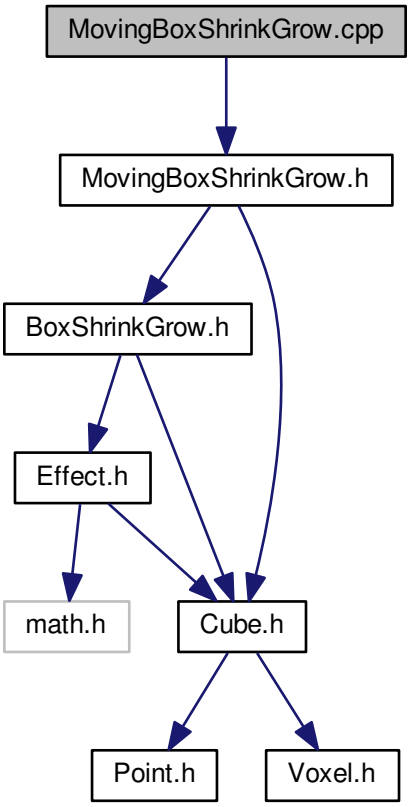
5.78 main.cpp

```
00001 #include <GL/glut.h>
00002 #include <pthread.h>
00003
00004 #include <Cube.h>
00005 #include <Blink.h>
00006 #include <Rain.h>
00007 #include <BoxShrinkGrow.h>
00008 #include <MovingBoxShrinkGrow.h>
00009 #include <GameOfLife.h>
00010
00011 #include <ByteArraySeekableInputStream.h>
00012 #include <BitmapFont.h>
00013 #include <TextRender.h>
00014 #include <ShiftingText.h>
00015
00016 #include "simulator.h"
00017
00018 Cube cube = Cube();
00019
00020 void *effect_runner(void *arg) {
00021   Blink blink = Blink(&cube, 10, 750);
00022   Rain rain = Rain(&cube, 10, 300, 4, 8);
00023   BoxShrinkGrow boxShrinkGrow = BoxShrinkGrow(&cube, 10, 750,
      BoxShrinkGrow::WIREFRAME);
00024   GameOfLife gameOfLife = GameOfLife(&cube, 1000, 750, 15);
00025   unsigned char fontStream[] = {
00026     0, 5, 8, 1, 32, 126, 0, 8,0, 0, 0, 0, 0, 0, 6, 95, 6, 0, 7, 3, 0, 7, 3, 36, 126, 36, 126, 36, 36, 43,
      106, 18, 0, 99, 19, 8, 100, 99, 54, 73, 86, 32, 80, 0, 7, 3, 0, 0, 0, 62, 65, 0, 0, 0, 65, 62, 0, 0, 8, 62,
      28, 62, 8, 8, 8, 62, 8, 8, 0, 224, 96, 0, 0, 8, 8, 8, 8, 8, 0, 96, 96, 0, 0, 32, 16, 8, 4, 2, 62, 81, 73, 69,
       62, 0, 66, 127, 64, 0, 98, 81, 73, 73, 70, 34, 73, 73, 73, 54, 24, 20, 18, 127, 16, 47, 73, 73, 73, 49, 60,
       74, 73, 73, 48, 1, 113, 9, 5, 3, 54, 73, 73, 73, 54, 6, 73, 73, 41, 30, 0, 108, 108, 0, 0, 0, 236, 108, 0,
      0, 8, 20, 34, 65, 0, 36, 36, 36, 36, 36, 0, 65, 34, 20, 8, 2, 1, 89, 9, 6, 62, 65, 93, 85, 30, 126, 9, 9, 9,
       126, 127, 73, 73, 73, 54, 62, 65, 65, 65, 34, 127, 65, 65, 65, 62, 127, 73, 73, 73, 65, 127, 9, 9, 9, 1, 62
      , 65, 73, 73, 122, 127, 127, 24, 127, 127, 0, 65, 127, 65, 0, 48, 64, 64, 64, 63, 127, 8, 20, 34, 65, 127,
      64, 64, 64, 64, 127, 2, 4, 2, 127, 127, 2, 4, 8, 127, 62, 65, 65, 65, 62, 127, 127, 27, 31, 14, 62, 65, 81,
      33, 94, 127, 9, 9, 25, 102, 38, 73, 73, 73, 50, 1, 1, 127, 1, 1, 63, 64, 64, 64, 63, 31, 32, 64, 32, 31, 63,
      64, 60, 64, 63, 99, 20, 8, 20, 99, 7, 8, 112, 8, 7, 113, 73, 69, 67, 0, 0, 127, 65, 65, 0, 2, 4, 8, 16, 32,
      0, 65, 65, 127, 0, 4, 2, 1, 2, 4, 128, 128, 128, 128, 128, 0, 3, 7, 0, 0, 32, 84, 84, 84, 120, 127, 68, 68,
      68, 56, 56, 68, 68, 68, 40, 56, 68, 68, 68, 127, 56, 84, 84, 84, 24, 8, 126, 9, 9, 0, 24, 164, 164, 164, 124
      , 127, 4, 4, 120, 0, 0, 0, 125, 0, 0, 64, 128, 132, 125, 0, 127, 16, 40, 68, 0, 0, 0, 127, 64, 0, 124, 4, 24
      , 4, 120, 124, 4, 4, 120, 0, 56, 68, 68, 68, 56, 252, 68, 68, 68, 56, 56, 68, 68, 68, 252, 68, 120, 68, 4, 8
      , 8, 84, 84, 84, 32, 4, 62, 68, 36, 0, 60, 64, 32, 124, 0, 28, 32, 64, 32, 28, 60, 96, 48, 96, 60, 108, 16,
      16, 108, 0, 156, 160, 96, 60, 0, 100, 84, 84, 76, 0, 8, 62, 65, 65, 0, 0, 0, 127, 0, 0, 0, 65, 65, 62, 8, 2,
       1, 2, 1, 0
00027   };
00028   ByteArraySeekableInputStream is =
      ByteArraySeekableInputStream(&fontStream[0], (unsigned int) sizeof(fontStream))
      ;
00029   BitmapFont font = BitmapFont(&is);
00030   TextRender render = TextRender(&cube, &font);
00031   Point p = Point(3, 0, 7);
00032   ShiftingText::ShiftingTextSettings settings = {&
      render, (const char *) "HP is the best company to work at", 1,
      TextRender::ZYX, &p};
00033   ShiftingText shiftingText = ShiftingText(&cube, 10, 1000, &settings);
00034   MovingBoxShrinkGrow movingBoxShrinkGrow =
      MovingBoxShrinkGrow(&cube, 100, 70, BoxShrinkGrow::WIREFRAME);
00035   GameOfLife gol = GameOfLife(&cube, 10, 750, 5);
00036   while (1) {
00037     shiftingText.run();
00038   }
00039   return NULL;
00040 }
00041
00042 int main(int argc, char* argv[]) {
00043   int arg = 0;
00044   void *exit_status;
00045
00046   glutInit(&argc,argv);
00047   pthread_t cube_thread, effect_runner_thread;
00048
00049   pthread_create(&cube_thread, NULL, cubeInit, &arg);
00050   pthread_create(&effect_runner_thread, NULL, effect_runner, &arg);
00051
```

```
00052   pthread_join(cube_thread, &exit_status);
00053   pthread_join(effect_runner_thread, &exit_status);
00054
00055   return 0;
00056 }
```

5.79 MovingBoxShrinkGrow.cpp File Reference

`#include <MovingBoxShrinkGrow.h>`
Include dependency graph for MovingBoxShrinkGrow.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_MOVING_BOX_SHRINK_GROW_CPP__ 1

5.79.1 Macro Definition Documentation

5.79.1.1 #define __ARDUINO_CUBE_EFFECTS_MOVING_BOX_SHRINK_GROW_CPP__ 1

Definition at line 5 of file MovingBoxShrinkGrow.cpp.

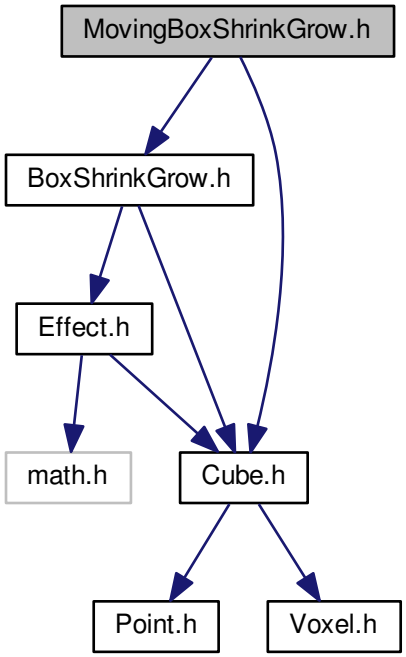
5.80 MovingBoxShrinkGrow.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_MOVING_BOX_SHRINK_GROW_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_MOVING_BOX_SHRINK_GROW_CPP__ 1
00006
00007 #include <MovingBoxShrinkGrow.h>
00008
00009 const unsigned char MovingBoxShrinkGrow::MAX_DIFF_MOVEMENTS = 0x03;
00010
00011 MovingBoxShrinkGrow::MovingBoxShrinkGrow(
      Cube *cube, unsigned int iterations, unsigned int iterationDelay,
      BoxType boxType) :
00012    BoxShrinkGrow(cube, iterations, iterationDelay, boxType) {
00013 }
00014
00015 void MovingBoxShrinkGrow::run() {
00016   unsigned int iteration;
00017   cube->useBackBuffer();
00018   for (iteration = 0; iteration < iterations; iteration++) {
00019     grow();
00020     state++;
00021     shrink();
00022   }
00023   cube->useFrontBuffer();
00024 }
00025
00026 void MovingBoxShrinkGrow::draw(char size) {
00027   this->BoxShrinkGrow::draw(size);
00028   switch(state % MAX_DIFF_MOVEMENTS) {
00029     case 0:
00030       cube->mirrorX();
00031       break;
00032     case 1:
00033       cube->mirrorY();
00034       break;
00035     case 2:
00036       cube->mirrorZ();
00037       break;
00038   }
00039 }
00040 #endif /* __ARDUINO_CUBE_EFFECTS_MOVING_BOX_SHRINK_GROW_CPP__ */
```

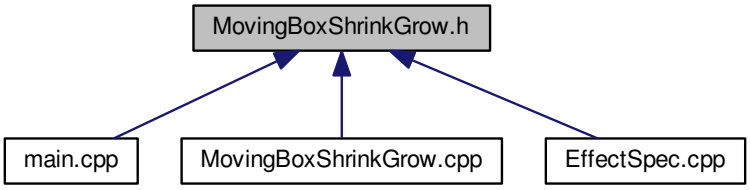
5.81 MovingBoxShrinkGrow.h File Reference

```
#include <BoxShrinkGrow.h>
#include <Cube.h>
```

Include dependency graph for MovingBoxShrinkGrow.h:



This graph shows which files directly or indirectly include this file:



Classes

- class MovingBoxShrinkGrow

5.82 MovingBoxShrinkGrow.h

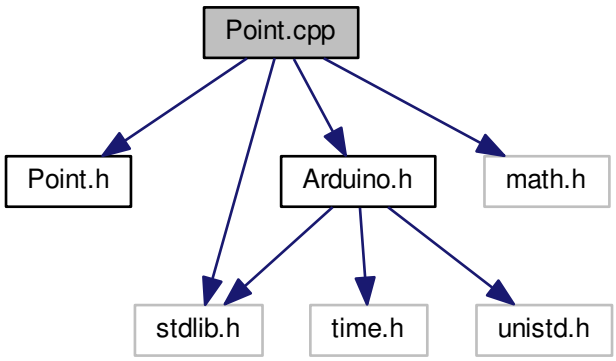
```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_MOVING_BOX_SHRINK_GROW_H__
00005 #define __ARDUINO_CUBE_EFFECTS_MOVING_BOX_SHRINK_GROW_H__ 1
00006
00007 #include <BoxShrinkGrow.h>
00008 #include <Cube.h>
```

```
00009
00010 class MovingBoxShrinkGrow : public BoxShrinkGrow {
00011
00012   unsigned char state;
00013
00014 public:
00015
00016   const static unsigned char MAX_DIFF_MOVEMENTS;
00017
00018   MovingBoxShrinkGrow(Cube *cube, unsigned int
      iterations, unsigned int iterationDelay, BoxType
      boxType);
00019
00020   virtual void run();
00021
00022   void draw(char size);
00023 };
00024
00025 #endif /* __ARDUINO_CUBE_EFFECTS_MOVING_BOX_SHRINK_GROW_H__ */
```

5.83 Point.cpp File Reference

```
#include <Point.h>
#include <Arduino.h>
#include <stdlib.h>
#include <math.h>
```

Include dependency graph for Point.cpp:



Macros

- #define __ARDUINO_CUBE_POINT_CPP__ 1

5.83.1 Macro Definition Documentation

5.83.1.1 #define __ARDUINO_CUBE_POINT_CPP__ 1

Definition at line 5 of file Point.cpp.

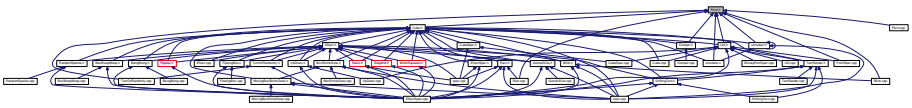
5.84 Point.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_POINT_CPP__
```

```
00005 #define __ARDUINO_CUBE_POINT_CPP__ 1
00006
00007 #include <Point.h>
00008 #include <Arduino.h>
00009 #include <stdlib.h>
00010 #include <math.h>
00011
00012 Point::Point() {
00013   init(0, 0, 0);
00014 }
00015
00016 Point::Point(unsigned char x, unsigned char y, unsigned char z) {
00017   init(x, y, z);
00018 }
00019
00020 void Point::randomize(unsigned char maxRange) {
00021   x = random(maxRange);
00022   y = random(maxRange);
00023   z = random(maxRange);
00024 }
00025
00026 unsigned char Point::distanceOnXTo(Point *p) {
00027   return (unsigned char) abs(x - p->x);
00028 }
00029
00030 unsigned char Point::distanceOnYTo(Point *p) {
00031   return (unsigned char) abs(y - p->y);
00032 }
00033
00034 unsigned char Point::distanceOnZTo(Point *p) {
00035   return (unsigned char) abs(z - p->z);
00036 }
00037
00038 float Point::distance2DTo(Point *p) {
00039   unsigned char dx = distanceOnXTo(p);
00040   unsigned char dy = distanceOnYTo(p);
00041   return sqrt(dx * dx + dy * dy);
00042 }
00043
00044 float Point::distance3DTo(Point *p) {
00045   unsigned char dx = distanceOnXTo(p);
00046   unsigned char dy = distanceOnYTo(p);
00047   unsigned char dz = distanceOnZTo(p);
00048   return sqrt(dx * dx + dy * dy + dz * dz);
00049 }
00050
00051 #endif /* __ARDUINO_CUBE_POINT_CPP__ */
```

5.85 Point.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class Point

5.86 Point.h

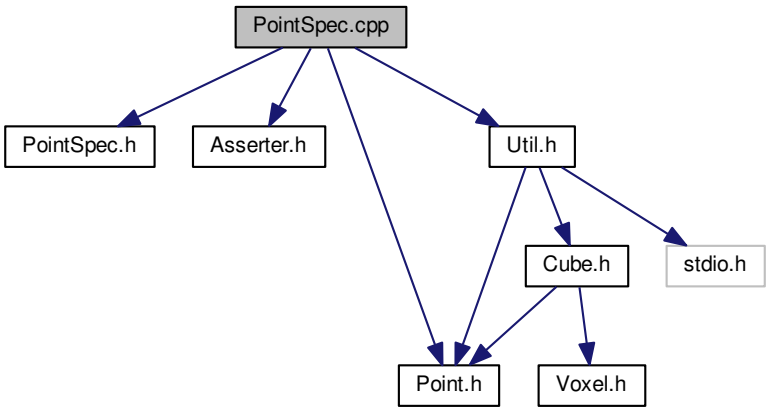
```
00001
00004 #ifndef __ARDUINO_CUBE_POINT_H__
00005 #define __ARDUINO_CUBE_POINT_H__ 1
00006
00007 class Point {
00008
00009 public:
00010
00011   unsigned char x;
00012   unsigned char y;
00013   unsigned char z;
```

```
00014
00015   Point();
00016
00017   Point(unsigned char x, unsigned char y, unsigned char z);
00018
00019   void randomize(unsigned char maxRange);
00020
00021   unsigned char distanceOnXTo(Point *p);
00022
00023   unsigned char distanceOnYTo(Point *p);
00024
00025   unsigned char distanceOnZTo(Point *p);
00026
00027   float distance2DTo(Point *p);
00028
00029   float distance3DTo(Point *p);
00030
00031 private:
00032
00033   void init(unsigned char x, unsigned char y, unsigned char z) {
00034     this->x = x;
00035     this->y = y;
00036     this->z = z;
00037   }
00038 };
00039
00040 #endif /* __ARDUINO_CUBE_POINT_H__ */
```

5.87 PointSpec.cpp File Reference

```
#include <PointSpec.h>
#include <Asserter.h>
#include <Point.h>
#include <Util.h>
```

Include dependency graph for PointSpec.cpp:



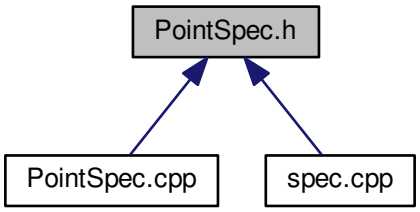
5.88 PointSpec.cpp

```
00001 #include <PointSpec.h>
00002 #include <Asserter.h>
00003 #include <Point.h>
00004 #include <Util.h>
00005
00006 PointSpec::PointSpec() {
00007 }
00008
00009 void PointSpec::run() {
00010   randomizeSpec();
00011   distanceOnXToSpec();
```

```
00012   distanceOnYToSpec();
00013   distanceOnZToSpec();
00014   distance2DToSpec();
00015   distance3DSpec();
00016 }
00017
00018 void PointSpec::randomizeSpec() {
00019   Point p;
00020   p.randomize(1);
00021   Asserter::assertEqual(true, (p.x == 0), "randomizeSpec: should randomize X dim.");
00022   Asserter::assertEqual(true, (p.y == 0), "randomizeSpec: should randomize Y dim.");
00023   Asserter::assertEqual(true, (p.z == 0), "randomizeSpec: should randomize Z dim.");
00024 }
00025
00026 void PointSpec::distanceOnXToSpec() {
00027   unsigned char distance;
00028   Point p0 = Point(0, 0, 0);
00029   Point p1 = Point(10, 20, 30);
00030   distance = p0.distanceOnXTo(&p1);
00031   Asserter::assertEqual(distance, 10, "distanceOnXToSpec: should get right distance
       when the second point is ahead.");
00032   distance = p1.distanceOnXTo(&p0);
00033   Asserter::assertEqual(distance, 10, "distanceOnXToSpec: should get right distance
       when the second point is behind.");
00034 }
00035
00036 void PointSpec::distanceOnYToSpec() {
00037   unsigned char distance;
00038   Point p0 = Point(0, 0, 0);
00039   Point p1 = Point(10, 20, 30);
00040   distance = p0.distanceOnYTo(&p1);
00041   Asserter::assertEqual(distance, 20, "distanceOnYToSpec: should get right distance
       when the second point is ahead.");
00042   distance = p1.distanceOnYTo(&p0);
00043   Asserter::assertEqual(distance, 20, "distanceOnYToSpec: should get right distance
       when the second point is behind.");
00044 }
00045
00046 void PointSpec::distanceOnZToSpec() {
00047   unsigned char distance;
00048   Point p0 = Point(0, 0, 0);
00049   Point p1 = Point(10, 20, 30);
00050   distance = p0.distanceOnZTo(&p1);
00051   Asserter::assertEqual(distance, 30, "distanceOnZToSpec: should get right distance
       when the second point is ahead.");
00052   distance = p1.distanceOnZTo(&p0);
00053   Asserter::assertEqual(distance, 30, "distanceOnZToSpec: should get right distance
       when the second point is behind.");
00054 }
00055
00056 void PointSpec::distance2DToSpec() {
00057   float distance;
00058   Point p0 = Point(0, 0, 0);
00059   Point p1 = Point(3, 4, 0);
00060   distance = p0.distance2DTo(&p1);
00061   Asserter::assertEqual(distance, 5, "distance2DToSpec: should get right 2D distance
       when the second point is ahead.");
00062   distance = p1.distance2DTo(&p0);
00063   Asserter::assertEqual(distance, 5, "distance2DToSpec: should get right 2D distance
       when the second point is behind.");
00064 }
00065
00066 void PointSpec::distance3DSpec() {
00067   float distance;
00068   Point p0 = Point(0, 0, 0);
00069   Point p1 = Point(3, 4, 5);
00070   distance = p0.distance3DTo(&p1);
00071   Asserter::assertEqual(distance, 7, "distance2DToSpec: should get right 3D distance
       when the second point is ahead.");
00072   distance = p1.distance3DTo(&p0);
00073   Asserter::assertEqual(distance, 7, "distance2DToSpec: should get right 3D distance
       when the second point is behind.");
00074 }
```

5.89 PointSpec.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class PointSpec

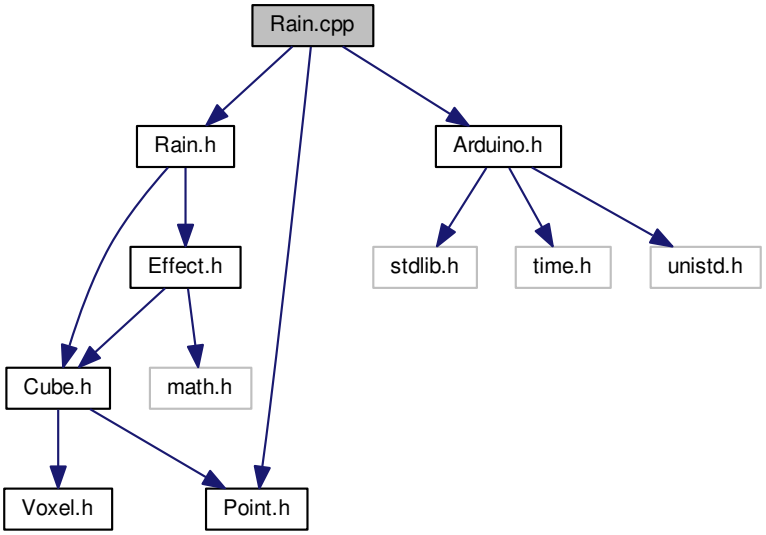
5.90 PointSpec.h

```
00001
00004 #ifndef __ARDUINO_CUBE_POINT_TEST_H__
00005 #define __ARDUINO_CUBE_POINT_TEST_H__ 1
00006
00007 class PointSpec {
00008
00009 public:
00010
00011   PointSpec();
00012
00013   void run();
00014
00015   void randomizeSpec();
00016
00017   void distanceOnXToSpec();
00018
00019   void distanceOnYToSpec();
00020
00021   void distanceOnZToSpec();
00022
00023   void distance2DToSpec();
00024
00025   void distance3DSpec();
00026 };
00027
00028 #endif /* __ARDUINO_CUBE_POINT_TEST_H__ */
```

5.91 Rain.cpp File Reference

```
#include <Rain.h>
#include <Arduino.h>
#include <Point.h>
```

Include dependency graph for Rain.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_RAIN_CPP__ 1

5.91.1 Macro Definition Documentation

5.91.1.1 #define __ARDUINO_CUBE_EFFECTS_RAIN_CPP__ 1

Definition at line 5 of file Rain.cpp.

5.92 Rain.cpp

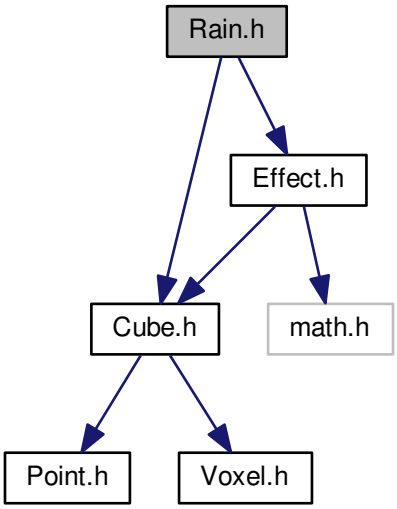
```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_RAIN_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_RAIN_CPP__ 1
00006
00007 #include <Rain.h>
00008 #include <Arduino.h>
00009 #include <Point.h>
00010
00011 Rain::Rain(Cube *cube, unsigned int iterations, unsigned int iterationDelay, unsigned
      char minDrops, unsigned char maxDrops) :
00012     Effect(cube, iterations, iterationDelay), minDrops(minDrops), maxDrops(maxDrops) {
00013 }
00014
00015 void Rain::run() {
00016   unsigned char k, n;
00017   unsigned int iteration;
00018   Point p = Point(0, 0, Cube::SIZE - 1);
00019   for (iteration = 0; iteration < iterations; iteration++) {
00020     n = random(minDrops, maxDrops);
00021     for (k = 0; k < n; k++) {
00022       p.x = random(0, Cube::SIZE);
00023       p.y = random(0, Cube::SIZE);
00024       cube->turnVoxelOn(&p);
00025     }
00026     delay(iterationDelay);
00027     cube->shiftOnZ(DOWN);
00028     cube->turnPlaneZOff(Cube::SIZE - 1);
```

```
00029   }
00030 }
00031
00032 #endif /* __ARDUINO_CUBE_EFFECTS_RAIN_CPP__ */
```

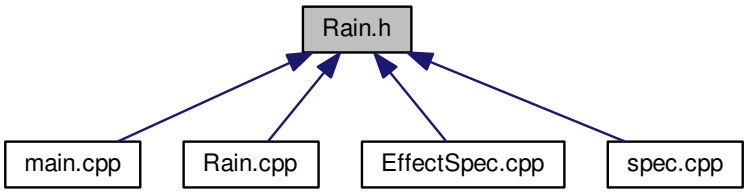
5.93 Rain.h File Reference

```
#include <Effect.h>
#include <Cube.h>
```

Include dependency graph for Rain.h:



This graph shows which files directly or indirectly include this file:



Classes

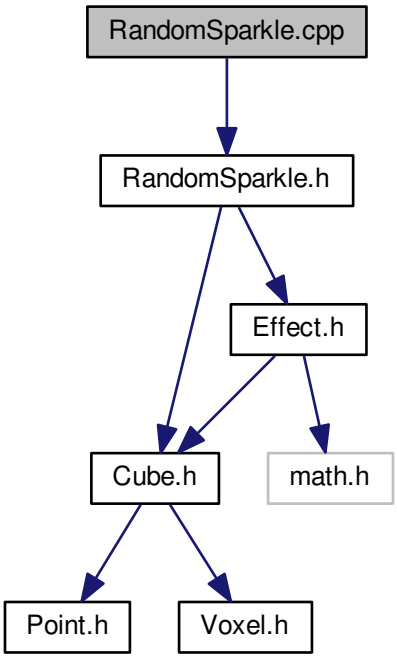
- class Rain

5.94 Rain.h

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_RAIN_H__
00005 #define __ARDUINO_CUBE_EFFECTS_RAIN_H__ 1
00006
00007 #include <Effect.h>
00008 #include <Cube.h>
00009
00010 class Rain : public Effect {
00011
00012   unsigned char minDrops;
00013   unsigned char maxDrops;
00014
00015 public:
00016
00017   Rain(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay, unsigned char minDrops, unsigned char maxDrops);
00018
00019   virtual void run();
00020 };
00021
00022 #endif /* __ARDUINO_CUBE_EFFECTS_RAIN_H__ */
```

5.95 RandomSparkle.cpp File Reference

`#include <RandomSparkle.h>`
Include dependency graph for RandomSparkle.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_RANDOM_SPARKLE_CPP__ 1

5.95.1 Macro Definition Documentation

5.95.1.1 #define __ARDUINO_CUBE_EFFECTS_RANDOM_SPARKLE_CPP__ 1

Definition at line 5 of file RandomSparkle.cpp.

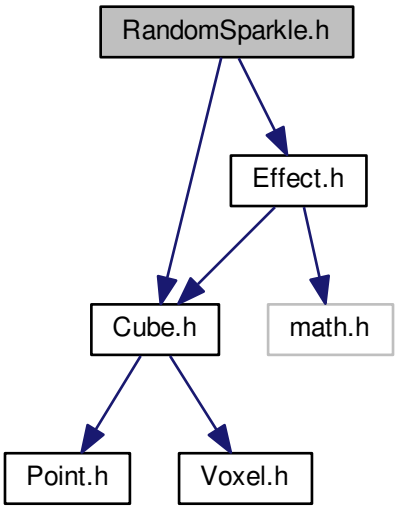
5.96 RandomSparkle.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_RANDOM_SPARKLE_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_RANDOM_SPARKLE_CPP__ 1
00006
00007 #include <RandomSparkle.h>
00008
00009 RandomSparkle::RandomSparkle(Cube *cube, unsigned int iterations,
      unsigned int iterationDelay) :
00010     Effect(cube, iterations, iterationDelay) {
00011 }
00012
00013 void RandomSparkle::run() {
00014   unsigned int iteration;
00015   for (iteration = 0; iteration < iterations; iteration++) {
00016   }
00017 }
00018
00019 #endif /* __ARDUINO_CUBE_EFFECTS_RANDOM_SPARKLE_CPP__ */
```

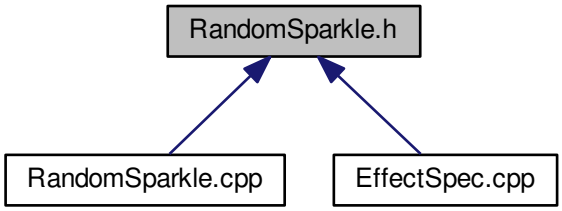
5.97 RandomSparkle.h File Reference

```
#include <Effect.h>
#include <Cube.h>
```

Include dependency graph for RandomSparkle.h:



This graph shows which files directly or indirectly include this file:



Classes

- class RandomSparkle

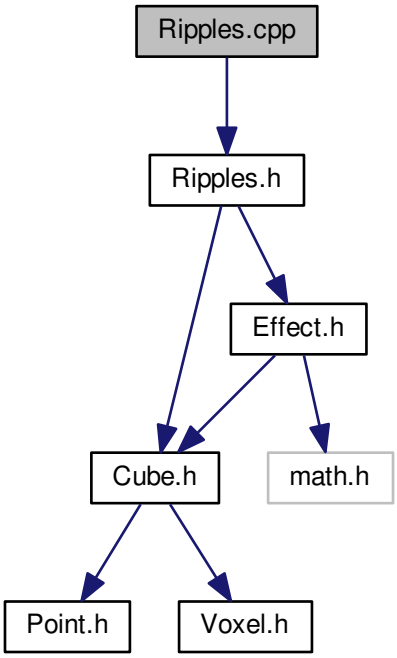
5.98 RandomSparkle.h

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_RANDOM_SPARKLE_H__
00005 #define __ARDUINO_CUBE_EFFECTS_RANDOM_SPARKLE_H__ 1
00006
00007 #include <Effect.h>
00008 #include <Cube.h>
00009
00010 class RandomSparkle : public Effect {
00011 public:
00012
00013   RandomSparkle(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay);
00014
00015   virtual void run();
00016 };
00017
00018 #endif /* __ARDUINO_CUBE_EFFECTS_RANDOM_SPARKLE_H__ */
```

5.99 Ripples.cpp File Reference

```
#include <Ripples.h>
```

Include dependency graph for Ripples.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_RIPPLES_CPP__ 1

5.99.1 Macro Definition Documentation

5.99.1.1 #define __ARDUINO_CUBE_EFFECTS_RIPPLES_CPP__ 1

Definition at line 5 of file Ripples.cpp.

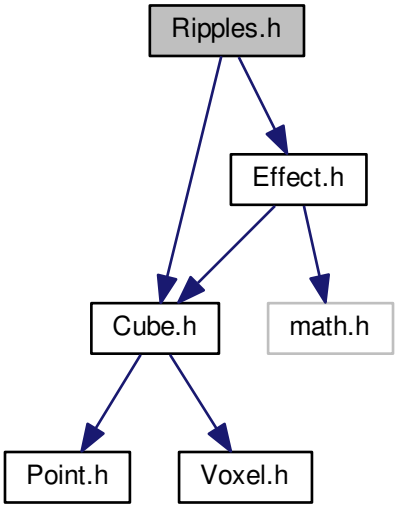
5.100 Ripples.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_RIPPLES_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_RIPPLES_CPP__ 1
00006
00007 #include <Ripples.h>
00008
00009 Ripples::Ripples(Cube *cube, unsigned int iterations, unsigned int iterationDelay)
    :
00010     Effect(cube, iterations, iterationDelay) {
00011 }
00012
00013 void Ripples::run() {
00014   unsigned int iteration;
00015   for (iteration = 0; iteration < iterations; iteration++) {
00016
00017   }
00018 }
00019
00020 #endif /* __ARDUINO_CUBE_EFFECTS_RIPPLES_CPP__ */
```

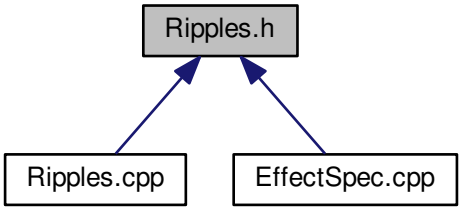
5.101 Ripples.h File Reference

```
#include <Effect.h>
#include <Cube.h>
```

Include dependency graph for Ripples.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Ripples

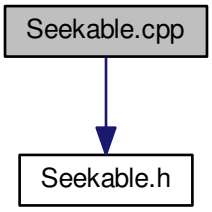
5.102 Ripples.h

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_RIPPLES_H__
00005 #define __ARDUINO_CUBE_EFFECTS_RIPPLES_H__ 1
00006
```

```
00007 #include <Effect.h>
00008 #include <Cube.h>
00009
00010 class Ripples : public Effect {
00011 public:
00012
00013   Ripples(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay);
00014
00015   virtual void run();
00016 };
00017
00018 #endif /* __ARDUINO_CUBE_EFFECTS_RIPPLES_H__ */
```

5.103 Seekable.cpp File Reference

`#include "Seekable.h"`
Include dependency graph for Seekable.cpp:



Macros

- #define __ARDUINO_IO_SEEKABLE_CPP__ 1

5.103.1 Macro Definition Documentation

5.103.1.1 #define __ARDUINO_IO_SEEKABLE_CPP__ 1

Arduino IO.

Seekable

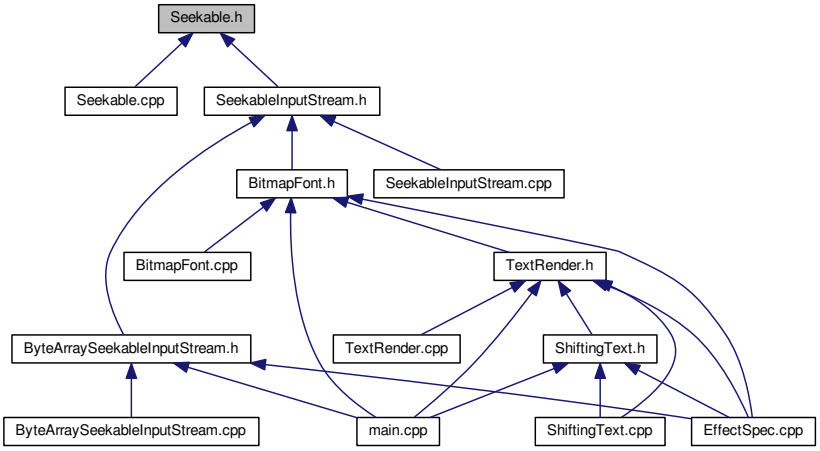
Definition at line 8 of file Seekable.cpp.

5.104 Seekable.cpp

```
00001
00007 #ifndef __ARDUINO_IO_SEEKABLE_CPP__
00008 #define __ARDUINO_IO_SEEKABLE_CPP__ 1
00009
00010 #include "Seekable.h"
00011
00012 #endif /* __ARDUINO_IO_SEEKABLE_CPP__ */
```

5.105 Seekable.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class Seekable

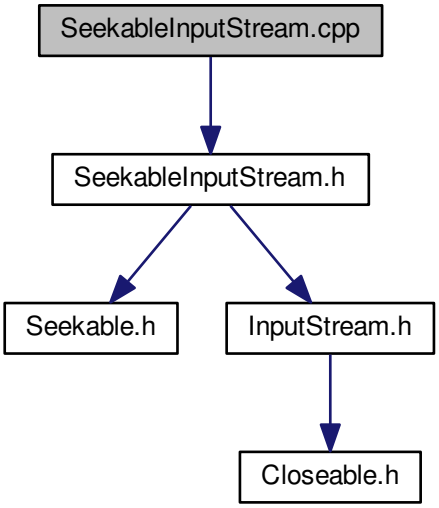
5.106 Seekable.h

```
00001
00007 #ifndef __ARDUINO_IO_SEEKABLE_H__
00008 #define __ARDUINO_IO_SEEKABLE_H__ 1
00009
00010 class Seekable {
00011 public:
00012
00013     virtual void seek(unsigned int pos) = 0;
00014 };
00015
00016 #endif /* __ARDUINO_IO_SEEKABLE_H__ */
```

5.107 SeekableInputStream.cpp File Reference

```
#include "SeekableInputStream.h"
```

Include dependency graph for SeekableInputStream.cpp:



Macros

- #define __ARDUINO_IO_SEEKABLE_INPUT_STREAM_CPP__ 1

5.107.1 Macro Definition Documentation

5.107.1.1 #define __ARDUINO_IO_SEEKABLE_INPUT_STREAM_CPP__ 1

Arduino IO.

SeekableInputStream

Definition at line 8 of file SeekableInputStream.cpp.

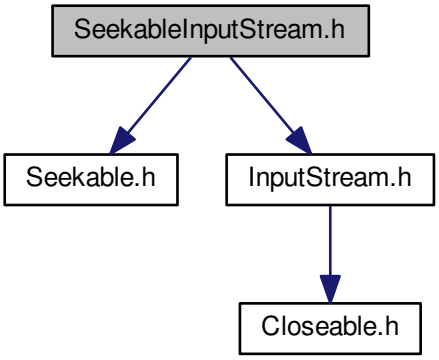
5.108 SeekableInputStream.cpp

```
00001
00007 #ifndef __ARDUINO_IO_SEEKABLE_INPUT_STREAM_CPP__
00008 #define __ARDUINO_IO_SEEKABLE_INPUT_STREAM_CPP__ 1
00009
00010 #include "SeekableInputStream.h"
00011
00012 #endif /* __ARDUINO_IO_SEEKABLE_INPUT_STREAM_CPP__ */
```

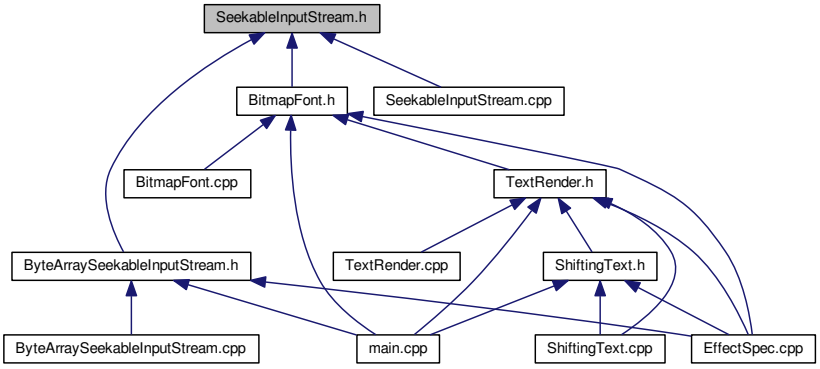
5.109 SeekableInputStream.h File Reference

```
#include <Seekable.h>
#include <InputStream.h>
```

Include dependency graph for SeekableInputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SeekableInputStream

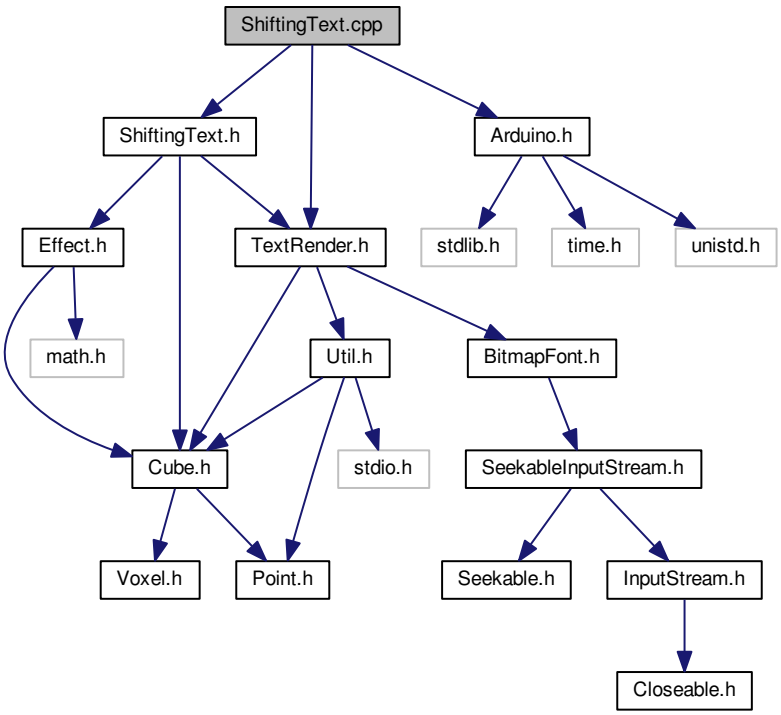
5.110 SeekableInputStream.h

```
00001
00007 #ifndef __ARDUINO_IO_SEEKABLE_INPUT_STREAM_H__
00008 #define __ARDUINO_IO_SEEKABLE_INPUT_STREAM_H__ 1
00009
00010 #include <Seekable.h>
00011 #include <InputStream.h>
00012
00013 class SeekableInputStream : public virtual Seekable, public virtual
      InputStream {
00014 public:
00015
00016 };
00017
00018 #endif /* __ARDUINO_IO_SEEKABLE_INPUT_STREAM_H__ */
```

5.111 ShiftingText.cpp File Reference

```
#include <ShiftingText.h>
#include <TextRender.h>
#include <Arduino.h>
```

Include dependency graph for ShiftingText.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_SHIFTING_TEXT_CPP__ 1

5.111.1 Macro Definition Documentation

5.111.1.1 #define __ARDUINO_CUBE_EFFECTS_SHIFTING_TEXT_CPP__ 1

Definition at line 5 of file ShiftingText.cpp.

5.112 ShiftingText.cpp

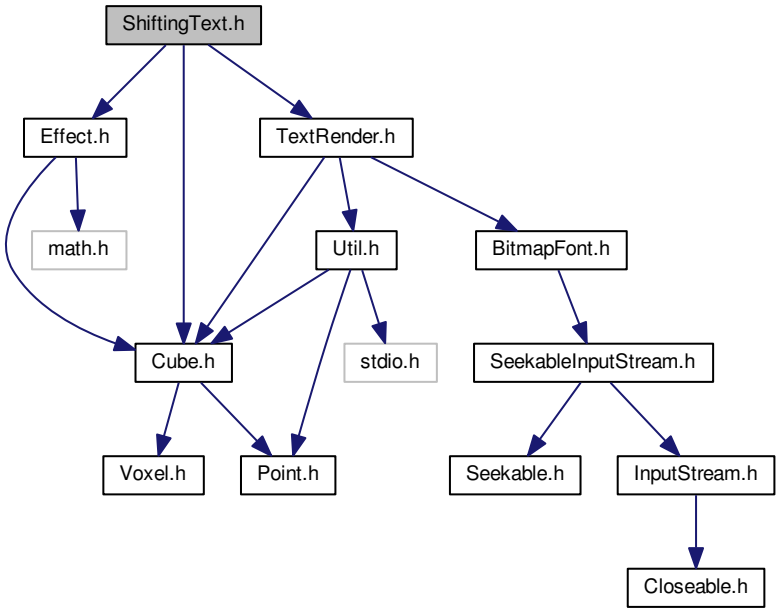
```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_SHIFTING_TEXT_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_SHIFTING_TEXT_CPP__ 1
00006
00007 #include <ShiftingText.h>
00008 #include <TextRender.h>
00009 #include <Arduino.h>
00010
00011 ShiftingText::ShiftingText(Cube *cube, unsigned int iterations, unsigned
      int iterationDelay, ShiftingTextSettings *settings) :
00012   Effect(cube, iterations, iterationDelay), settings(settings) {
00013 }
```

```
00014
00015 void ShiftingText::run() {
00016   unsigned int iteration;
00017   const char *c;
00018   for (iteration = 0; iteration < iterations; iteration++) {
00019     c = settings->text;
00020     while(*c != '\0') {
00021       displayCharacter(*c);
00022       shiftCharacter();
00023       c++;
00024     }
00025   }
00026 }
00027
00028 void ShiftingText::displayCharacter(const char c) {
00029   cube->clear();
00030   settings->render->printChar(settings->point, (
      TextRender::TextOrientation) settings->
      orientation, settings->charDepth, c);
00031 }
00032
00033 void ShiftingText::shiftCharacter() {
00034   unsigned char i;
00035   for (i = 1; i < Cube::SIZE; i++) {
00036     switch (settings->orientation) {
00037       case TextRender::ZYX:
00038         cube->shiftOnX(RIGHT);
00039       break;
00040       case TextRender::YZX:
00041         cube->shiftOnX(LEFT);
00042       break;
00043       case TextRender::XZY:
00044         cube->shiftOnY(FRONT);
00045       break;
00046       case TextRender::ZXY:
00047         cube->shiftOnY(BACK);
00048       break;
00049       case TextRender::XYZ:
00050         cube->shiftOnZ(DOWN);
00051       break;
00052       case TextRender::YXZ:
00053         cube->shiftOnZ(UP);
00054       break;
00055     }
00056     delay(iterationDelay / 8);
00057   }
00058 }
00059
00060 #endif /* __ARDUINO_CUBE_EFFECTS_SHIFTING_TEXT_CPP__ */
```

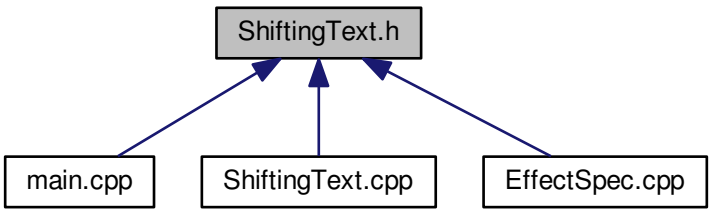
5.113 ShiftingText.h File Reference

```
#include <Effect.h>
#include <Cube.h>
#include <TextRender.h>
```

Include dependency graph for ShiftingText.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ShiftingText
- struct ShiftingText::ShiftingTextSettings

5.114 ShiftingText.h

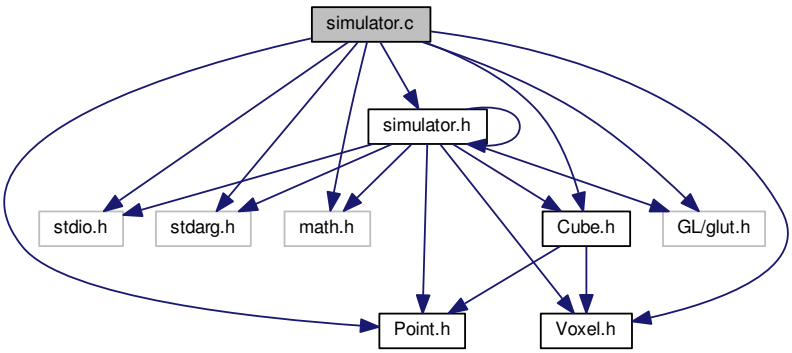
```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_SHIFTING_TEXT_H__
00005 #define __ARDUINO_CUBE_EFFECTS_SHIFTING_TEXT_H__ 1
00006
00007 #include <Effect.h>
00008 #include <Cube.h>
```

```
00009 #include <TextRender.h>
00010
00011 class ShiftingText : public Effect {
00012
00013 public:
00014
00015   typedef struct {
00016     TextRender *render;
00017     const char *text;
00018     unsigned char charDepth;
00019     unsigned char orientation;
00020     Point *point;
00021   } ShiftingTextSettings;
00022
00023   ShiftingTextSettings *settings;
00024
00025   ShiftingText(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay, ShiftingTextSettings *settings);
00026
00027   virtual void run();
00028
00029   void displayCharacter(const char c);
00030
00031   void shiftCharacter();
00032 };
00033
00034 #endif /* __ARDUINO_CUBE_EFFECTS_SHIFTING_TEXT_H__ */
```

5.115 simulator.c File Reference

```
#include <stdio.h>
#include <stdarg.h>
#include <math.h>
#include <GL/glut.h>
#include <Cube.h>
#include <Voxel.h>
#include <Point.h>
#include "simulator.h"
```

Include dependency graph for simulator.c:



Macros

- #define ROTATE_STEP 5
- #define SPACE 0.14

Functions

- void render ()

- void special (int key, int x, int y)
- void mouse (int button, int pressed, int x, int y)
- void mouseMotion (int x, int y)
- void ∗ cubeInit (void ∗arg)

Variables

- double rotateOnY
- double rotateOnX
- int isClicked
- int previousX
- int previousY
- int deltaX
- int deltaY

5.115.1 Macro Definition Documentation

5.115.1.1 #define ROTATE_STEP 5

Definition at line 12 of file simulator.c.

5.115.1.2 #define SPACE 0.14

Definition at line 13 of file simulator.c.

5.115.2 Function Documentation

5.115.2.1 void∗ cubeInit (void ∗ *arg*)

Definition at line 107 of file simulator.c.

5.115.2.2 void mouse (int *button,* int *pressed,* int *x,* int *y*)

Definition at line 79 of file simulator.c.

5.115.2.3 void mouseMotion (int *x,* int *y*)

Definition at line 91 of file simulator.c.

5.115.2.4 void render ( )

Definition at line 18 of file simulator.c.

5.115.2.5 void special (int *key,* int *x,* int *y*)

Definition at line 49 of file simulator.c.

5.115.3 Variable Documentation

5.115.3.1 int deltaX

Definition at line 16 of file simulator.c.

5.115.3.2 int deltaY

Definition at line 16 of file simulator.c.

5.115.3.3 int isClicked

Definition at line 16 of file simulator.c.

5.115.3.4 int previousX

Definition at line 16 of file simulator.c.

5.115.3.5 int previousY

Definition at line 16 of file simulator.c.

5.115.3.6 double rotateOnX

Definition at line 15 of file simulator.c.

5.115.3.7 double rotateOnY

Definition at line 15 of file simulator.c.

5.116 simulator.c

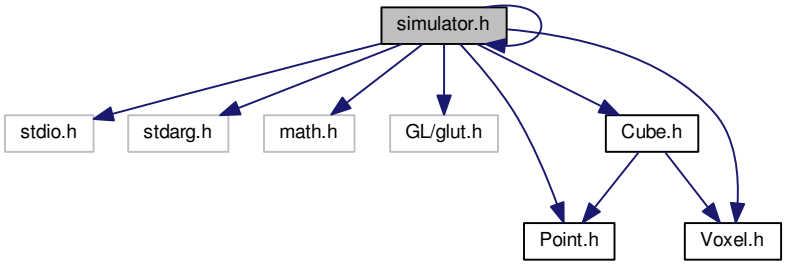
```
00001 #include <stdio.h>
00002 #include <stdarg.h>
00003 #include <math.h>
00004 #include <GL/glut.h>
00005
00006 #include <Cube.h>
00007 #include <Voxel.h>
00008 #include <Point.h>
00009
00010 #include "simulator.h"
00011
00012 #define ROTATE_STEP 5
00013 #define SPACE 0.14
00014
00015 double rotateOnY, rotateOnX;
00016 int isClicked, previousX, previousY, deltaX,
      deltaY;
00017
00018 void render() {
00019   Voxel v;
00020   Point p;
00021   glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
00022   glEnable(GL_BLEND);
00023   glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
00024   glLoadIdentity();
00025   glRotatef(rotateOnX, 1.0, 0.0, 0.0);
00026   glRotatef(rotateOnY, 0.0, 1.0, 0.0);
00027   glMatrixMode(GL_PROJECTION);
00028   for(p.x = 0; p.x < 8; p.x++) {
00029     for(p.y = 0; p.y < 8; p.y++) {
00030       for(p.z = 0; p.z < 8; p.z++) {
00031         cube.readVoxel(&p, &v);
00032         glPushMatrix();
00033         glTranslatef(SPACE * (p.x - Cube::SIZE / 2), SPACE * (p.
      y - Cube::SIZE / 2), -SPACE * (p.z - Cube::SIZE / 2));
00034         if (v.state == ON) {
00035           glColor4f(1.0, 0.0, 0.0, 1.0f);
00036           glutSolidSphere(0.018, 25, 5);
00037         } else {
00038           glColor4f(1.0, 1.0, 1.0, 0.1f);
00039           glutSolidSphere(0.015, 25, 5);
00040         }
00041         glPopMatrix();
00042       }
00043     }
00044   }
00045   glFlush();
00046   glutSwapBuffers();
00047 }
00048
00049 void special(int key, int x, int y) {
00050   switch(key) {
00051     case GLUT_KEY_RIGHT:
00052       rotateOnY += ROTATE_STEP;
00053       break;
```

```
00054     case GLUT_KEY_LEFT:
00055       rotateOnY -= ROTATE_STEP;
00056       break;
00057     case GLUT_KEY_UP:
00058       rotateOnX += ROTATE_STEP;
00059       break;
00060     case GLUT_KEY_DOWN:
00061       rotateOnX -= ROTATE_STEP;
00062       break;
00063     case '8':
00064       cube.shiftOnY(UP);
00065       break;
00066     case '2':
00067       cube.shiftOnY(DOWN);
00068       break;
00069     case '6':
00070       cube.shiftOnX(RIGHT);
00071       break;
00072     case '4':
00073       cube.shiftOnX(LEFT);
00074       break;
00075   }
00076   glutPostRedisplay();
00077 }
00078
00079 void mouse(int button, int pressed, int x, int y) {
00080   if (button == 0 && pressed == 0) {
00081     previousX = x;
00082     previousY = y;
00083     isClicked = 1;
00084   } else if (button == 0 && pressed == 1) {
00085     if (isClicked) {
00086       isClicked = 0;
00087     }
00088   }
00089 }
00090
00091 void mouseMotion(int x, int y) {
00092   if (isClicked) {
00093     if (previousX || previousY) {
00094       deltaX = previousX - x;
00095       deltaY = previousY - y;
00096       rotateOnX += deltaY;
00097       rotateOnY += deltaX;
00098       previousX = x;
00099       previousY = y;
00100     } else{
00101       previousX = x;
00102       previousY = y;
00103     }
00104   }
00105 }
00106
00107 void *cubeInit(void *arg) {
00108   glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
00109   glutInitWindowSize(800, 800);
00110   glutCreateWindow("ArduinoCube");
00111   glEnable(GL_DEPTH_TEST);
00112   glutDisplayFunc(render);
00113   glutIdleFunc(render);
00114   glutSpecialFunc(special);
00115   glutMouseFunc(mouse);
00116   glutMotionFunc(mouseMotion);
00117   glutMainLoop();
00118   return NULL;
00119 }
```

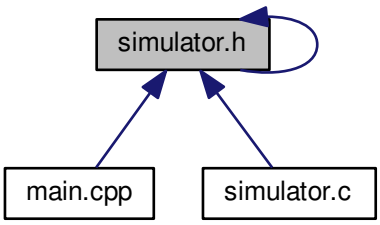
5.117 simulator.h File Reference

```
#include <stdio.h>
#include <stdarg.h>
#include <math.h>
#include <GL/glut.h>
#include <Cube.h>
#include <Voxel.h>
#include <Point.h>
#include "simulator.h"
```

Include dependency graph for simulator.h:



This graph shows which files directly or indirectly include this file:



Functions

- void render ()
- void special (int key, int x, int y)
- void mouse (int button, int pressed, int x, int y)
- void mouseMotion (int x, int y)
- void * cubeInit (void *arg)

Variables

- Cube cube

5.117.1 Function Documentation

5.117.1.1 void* cubeInit (void * *arg*)

Definition at line 107 of file simulator.c.

5.117.1.2 void mouse (int *button*, int *pressed*, int *x*, int *y*)

Definition at line 79 of file simulator.c.

5.117.1.3 void mouseMotion (int *x*, int *y*)

Definition at line 91 of file simulator.c.

5.117.1.4 void render ( )

Definition at line 18 of file simulator.c.

5.117.1.5 void special (int *key*, int *x*, int *y*)

Definition at line 49 of file simulator.c.

5.117.2 Variable Documentation

5.117.2.1 Cube cube

Definition at line 18 of file main.cpp.

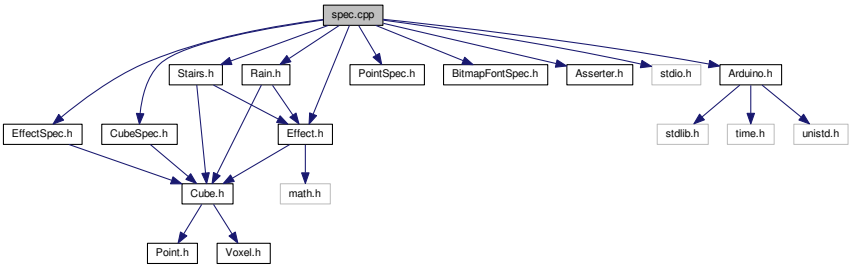
5.118 simulator.h

```
00001
00005 #ifndef __ARDUINO_CUBE_SIMULATOR_H__
00006 #define __ARDUINO_CUBE_SIMULATOR_H__ 1
00007
00008 extern Cube cube;
00009 #include <stdio.h>
00010 #include <stdarg.h>
00011 #include <math.h>
00012 #include <GL/glut.h>
00013
00014 #include <Cube.h>
00015 #include <Voxel.h>
00016 #include <Point.h>
00017 #include "simulator.h"
00018
00019 void render();
00020 void special(int key, int x, int y);
00021 void mouse(int button, int pressed, int x, int y);
00022 void mouseMotion(int x, int y);
00023 void *cubeInit(void *arg);
00024
00025 #endif /* __ARDUINO_CUBE_SIMULATOR_H__ */
```

5.119 spec.cpp File Reference

```
#include <CubeSpec.h>
#include <EffectSpec.h>
#include <PointSpec.h>
#include <BitmapFontSpec.h>
#include <Asserter.h>
#include <Effect.h>
#include <Rain.h>
#include <Stairs.h>
#include <stdio.h>
#include <Arduino.h>
```

Include dependency graph for spec.cpp:



Functions

- int main (int argc, char ∗argv[ ])

5.119.1 Function Documentation

5.119.1.1 int main (int *argc*, char ∗ *argv[ ]*)

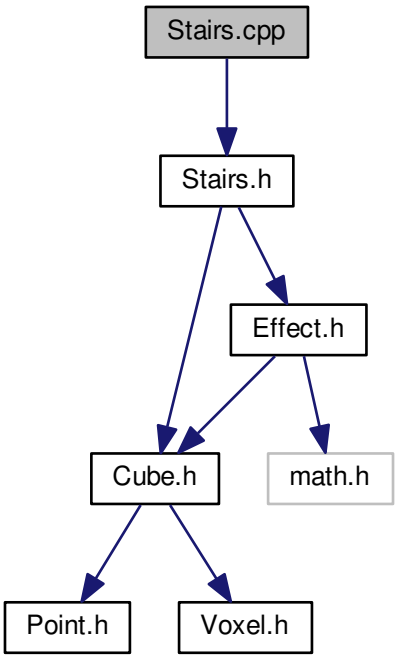
Definition at line 12 of file spec.cpp.

5.120 spec.cpp

```
00001 #include <CubeSpec.h>
00002 #include <EffectSpec.h>
00003 #include <PointSpec.h>
00004 #include <BitmapFontSpec.h>
00005 #include <Asserter.h>
00006 #include <Effect.h>
00007 #include <Rain.h>
00008 #include <Stairs.h>
00009 #include <stdio.h>
00010 #include <Arduino.h>
00011
00012 int main(int argc, char *argv[]) {
00013
00014   Cube cube = Cube();
00015
00016   CubeSpec cubeSpec = CubeSpec(&cube);
00017   PointSpec pointSpec = PointSpec();
00018   EffectSpec effectSpec = EffectSpec(&cube);
00019   BitmapFontSpec bitmapFontSpec = BitmapFontSpec();
00020
00021   randomSeed(time(NULL));
00022
00023   Asserter::reset();
00024   cubeSpec.run();
00025   printf("CubeSpec error: %d\n", Asserter::counter.error);
00026   printf("CubeSpec success: %d\n", Asserter::counter.success);
00027
00028   Asserter::reset();
00029   effectSpec.run();
00030   printf("EffectSpec error: %d\n", Asserter::counter.error);
00031   printf("EffectSpec success: %d\n", Asserter::counter.success);
00032
00033   Asserter::reset();
00034   pointSpec.run();
00035   printf("PointSpec error: %d\n", Asserter::counter.error);
00036   printf("PointSpec success: %d\n", Asserter::counter.success);
00037
00038   Asserter::reset();
00039   bitmapFontSpec.run();
00040   printf("BitmapFontSpec error: %d\n", Asserter::counter.error);
00041   printf("BitmapFontSpec success: %d\n", Asserter::counter.success);
00042
00043   return 0;
00044 }
```

5.121 Stairs.cpp File Reference

`#include <Stairs.h>`
Include dependency graph for Stairs.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_STAIRS_CPP__ 1

5.121.1 Macro Definition Documentation

5.121.1.1 #define __ARDUINO_CUBE_EFFECTS_STAIRS_CPP__ 1

Definition at line 5 of file Stairs.cpp.

5.122 Stairs.cpp

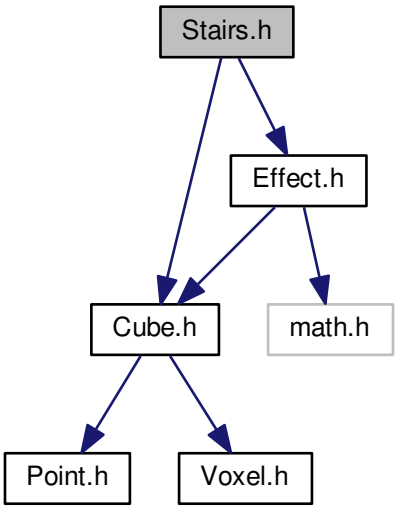
```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_STAIRS_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_STAIRS_CPP__ 1
00006
00007 #include <Stairs.h>
00008
00009 Stairs::Stairs(Cube *cube, unsigned int iterations, unsigned int iterationDelay) :
00010     Effect(cube, iterations, iterationDelay) {
00011 }
00012
00013 void Stairs::run() {
00014   unsigned int iteration;
00015   for (iteration = 0; iteration < iterations; iteration++) {
00016
00017   }
```

```
00018 }
00019 /*
00020 int effect_telcstairs_do(int x, int val, int delay) {
00021     int y, z;
00022     for (y = 0, z = x; y <= z; y++, x--) {
00023         if (x < CUBE_SIZE && y < CUBE_SIZE) {
00024             cube[x % CUBE_SIZE][y % CUBE_SIZE] = val;
00025         }
00026     }
00027     delay_ms(delay);
00028     return z;
00029 }
00030
00031 void effect_telcstairs(int invert, int delay, int val) {
00032     int x;
00033     if (invert) {
00034         for (x = CUBE_SIZE * 2; x >= 0; x--) {
00035             x = effect_telcstairs_do(x, val, delay);
00036         }
00037     } else {
00038         for (x = 0; x < CUBE_SIZE * 2; x++) {
00039             x = effect_telcstairs_do(x, val, delay);
00040         }
00041     }
00042 }*/
00043
00044 #endif /* __ARDUINO_CUBE_EFFECTS_STAIRS_CPP__ */
```

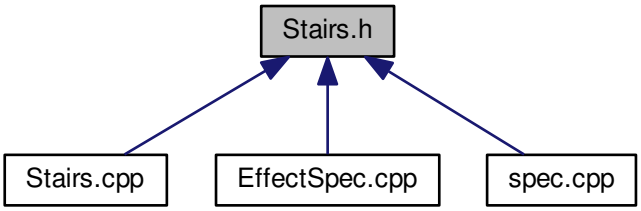
5.123 Stairs.h File Reference

```
#include <Effect.h>
#include <Cube.h>
```

Include dependency graph for Stairs.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Stairs

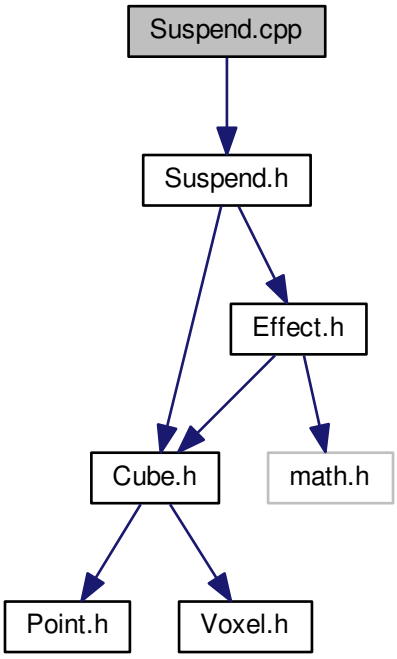
5.124 Stairs.h

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_STAIRS_H__
00005 #define __ARDUINO_CUBE_EFFECTS_STAIRS_H__ 1
00006
00007 #include <Effect.h>
00008 #include <Cube.h>
00009
00010 class Stairs : public Effect {
00011 public:
00012
00013   Stairs(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay);
00014
00015   virtual void run();
00016 };
00017
00018 #endif /* __ARDUINO_CUBE_EFFECTS_STAIRS_H__ */
```

5.125 Suspend.cpp File Reference

```
#include <Suspend.h>
```

Include dependency graph for Suspend.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_SUSPEND_CPP__ 1

5.125.1 Macro Definition Documentation

5.125.1.1 #define __ARDUINO_CUBE_EFFECTS_SUSPEND_CPP__ 1

Definition at line 5 of file Suspend.cpp.

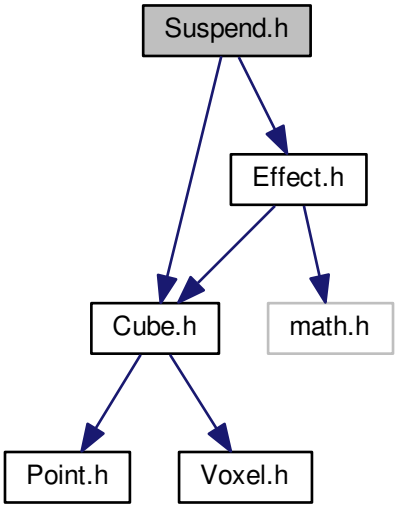
5.126 Suspend.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_SUSPEND_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_SUSPEND_CPP__ 1
00006
00007 #include <Suspend.h>
00008
00009 Suspend::Suspend(Cube *cube, unsigned int iterations, unsigned int iterationDelay)
      :
00010    Effect(cube, iterations, iterationDelay) {
00011 }
00012
00013 void Suspend::run() {
00014   unsigned int iteration;
00015   for (iteration = 0; iteration < iterations; iteration++) {
00016   }
00017 }
00018
00019 #endif /* __ARDUINO_CUBE_EFFECTS_SUSPEND_CPP__ */
```

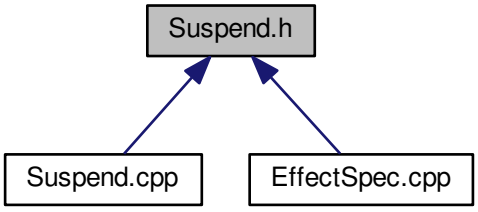
5.127 Suspend.h File Reference

```
#include <Effect.h>
#include <Cube.h>
```

Include dependency graph for Suspend.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Suspend

5.128 Suspend.h

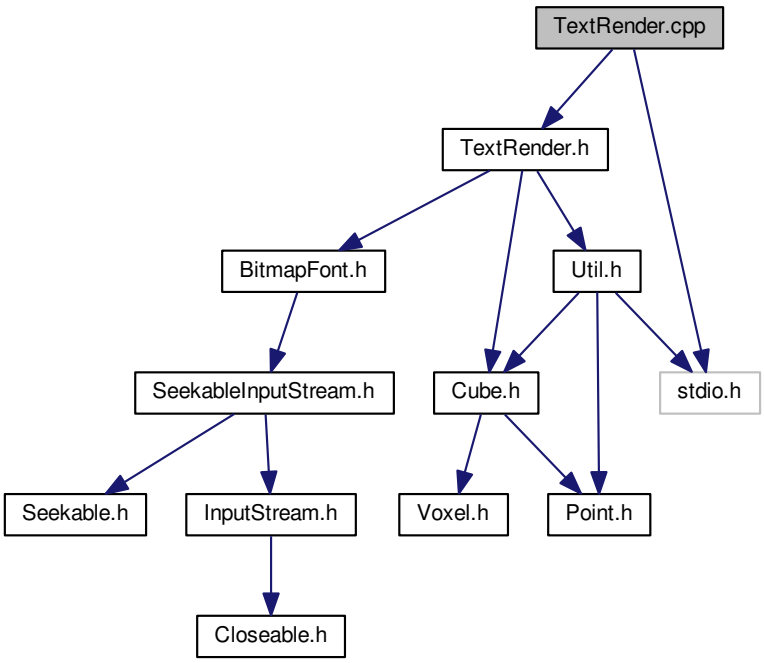
```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_SUSPEND_H__
00005 #define __ARDUINO_CUBE_EFFECTS_SUSPEND_H__ 1
00006
```

```
00007 #include <Effect.h>
00008 #include <Cube.h>
00009
00010 class Suspend : public Effect {
00011 public:
00012
00013   Suspend(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay);
00014
00015   virtual void run();
00016 };
00017
00018 #endif /* __ARDUINO_CUBE_EFFECTS_SUSPEND_H__ */
```

5.129 TextRender.cpp File Reference

```
#include <TextRender.h>
#include <stdio.h>
```

Include dependency graph for TextRender.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_TEXT_RENDER_CPP__ 1

5.129.1 Macro Definition Documentation

5.129.1.1 #define __ARDUINO_CUBE_EFFECTS_TEXT_RENDER_CPP__ 1

Definition at line 5 of file TextRender.cpp.

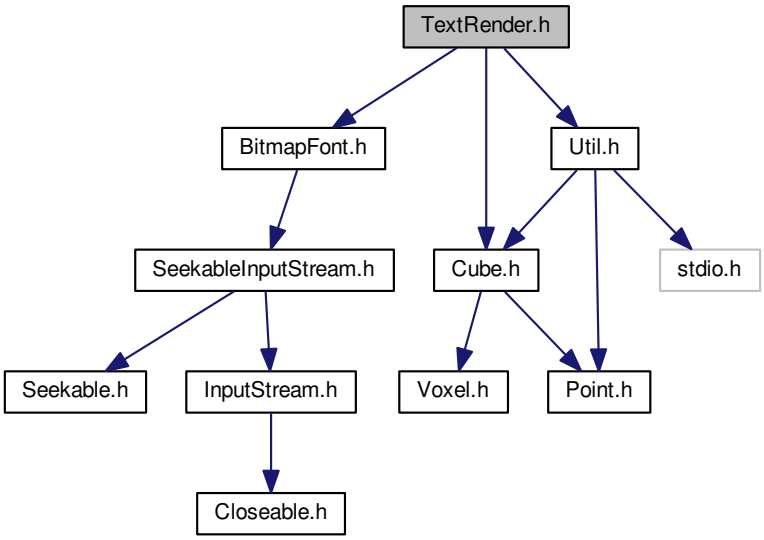
5.130 TextRender.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_TEXT_RENDER_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_TEXT_RENDER_CPP__ 1
00006
00007 #include <TextRender.h>
00008 #include <stdio.h>
00009
00010 TextRender::TextRender(Cube *cube, BitmapFont *font) : cube(cube),
      font(font) {
00011 }
00012
00013 void TextRender::printChar(Point *p, TextOrientation orientation,
      unsigned char depth, const char c) {
00014   Voxel v;
00015   unsigned char w, h, d, column, width, height, *x, *y, *z, glyphBuf[font->
      getGlyphLength()];
00016   x = y = z = 0;
00017   width = font->getCharacterWidth();
00018   height = font->getCharacterHeight();
00019   font->readGlyphData(glyphBuf, c);
00020   adjustCoordinates(p, orientation, &x, &y, &z);
00021   for (w = 0; w < width; w++) {
00022     column = glyphBuf[w];
00023     for (h = 0; h < height; h++) {
00024       if ((column & (0x01 << h)) != 0) {
00025         v.state = ON;
00026       } else {
00027         v.state = OFF;
00028       }
00029       for (d = 0; d < depth; d++) {
00030         cube->writeVoxel(*x + w, *y + h, *z + d, v.state);
00031       }
00032     }
00033   }
00034 }
00035
00036 void TextRender::adjustCoordinates(Point *p,
      TextOrientation orientation, unsigned char **x, unsigned char **y, unsigned char **z) {
00037   switch (orientation) {
00038     case XYZ:
00039       *x = &(p->x);
00040       *y = &(p->y);
00041       *z = &(p->z);
00042       break;
00043     case XZY:
00044       *x = &(p->x);
00045       *y = &(p->z);
00046       *z = &(p->y);
00047     break;
00048     case YZX:
00049       *x = &(p->y);
00050       *y = &(p->z);
00051       *z = &(p->x);
00052     break;
00053     case YXZ:
00054       *x = &(p->y);
00055       *y = &(p->x);
00056       *z = &(p->z);
00057     break;
00058     case ZYX:
00059       *x = &(p->z);
00060       *y = &(p->y);
00061       *z = &(p->x);
00062     break;
00063     case ZXY:
00064       *x = &(p->z);
00065       *y = &(p->x);
00066       *z = &(p->y);
00067      break;
00068   }
00069 }
00070
00071 #endif /* __ARDUINO_CUBE_TEXT_RENDER_CPP__ */
```

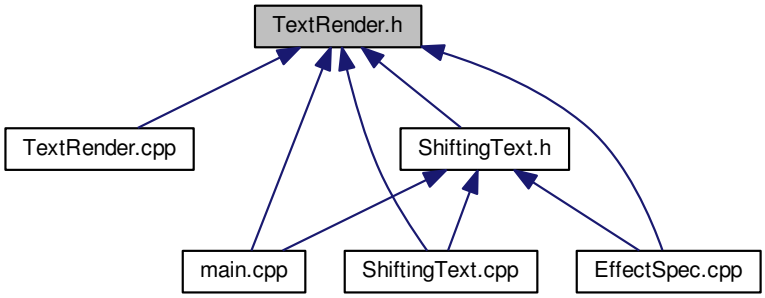
5.131 TextRender.h File Reference

```
#include <BitmapFont.h>
#include <Cube.h>
#include <Util.h>
```

Include dependency graph for TextRender.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TextRender

5.132 TextRender.h

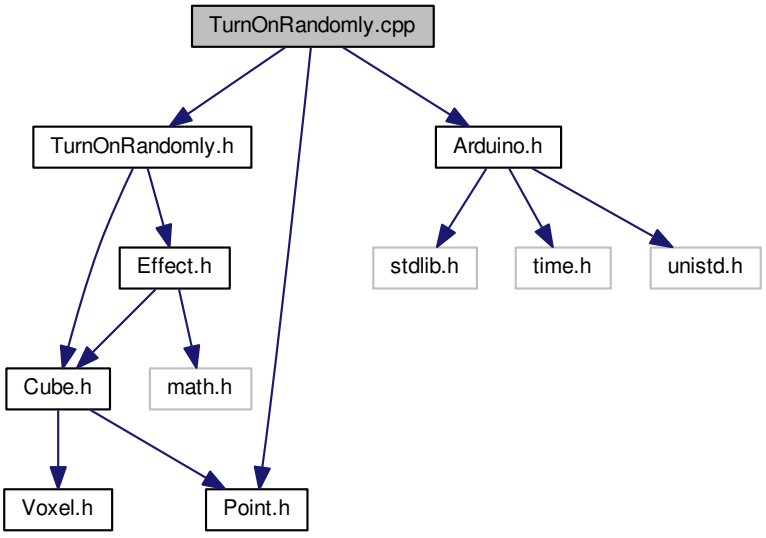
```
00001
00011 #ifndef __ARDUINO_CUBE_TEXT_RENDER_H__
00012 #define __ARDUINO_CUBE_TEXT_RENDER_H__ 1
00013
00014 #include <BitmapFont.h>
00015 #include <Cube.h>
00016 #include <Util.h>
00017
00018 class TextRender {
```

```
00019
00023   Cube *cube;
00024
00028   BitmapFont *font;
00029
00030 public:
00031
00032   typedef enum {
00033     XYZ,
00034     XZY,
00035     YXZ,
00036     YZX,
00037     ZXY,
00038     ZYX
00039   } TextOrientation;
00040
00047   TextRender(Cube *cube, BitmapFont *font);
00048
00057   void printChar(Point *p, TextOrientation orientation, unsigned char depth,
      const char c);
00058
00059
00060   void adjustCoordinates(Point *p, TextOrientation orientation,
      unsigned char **x, unsigned char **y, unsigned char **z);
00061 };
00062
00063 #endif /* __SDCC_CUBE_TEXT_RENDER_H__ */
```

5.133 TurnOnRandomly.cpp File Reference

```
#include <TurnOnRandomly.h>
#include <Arduino.h>
#include <Point.h>
```

Include dependency graph for TurnOnRandomly.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_TURN_ON_RANDOMLY_CPP__ 1

5.133.1 Macro Definition Documentation

5.133.1.1 #define __ARDUINO_CUBE_EFFECTS_TURN_ON_RANDOMLY_CPP__ 1

Definition at line 5 of file TurnOnRandomly.cpp.

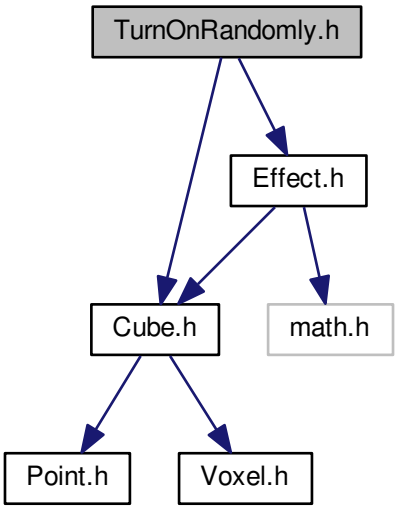
5.134 TurnOnRandomly.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_TURN_ON_RANDOMLY_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_TURN_ON_RANDOMLY_CPP__ 1
00006
00007 #include <TurnOnRandomly.h>
00008 #include <Arduino.h>
00009 #include <Point.h>
00010
00011 TurnOnRandomly::TurnOnRandomly(Cube *cube, unsigned int iterations,
      unsigned int iterationDelay, unsigned char maxOnVoxels) :
00012     Effect(cube, iterations, iterationDelay), maxOnVoxels(maxOnVoxels) {
00013 }
00014
00015 void TurnOnRandomly::run() {
00016   unsigned int iteration;
00017   unsigned char i;
00018   Point p;
00019   for (iteration = 0; iteration < iterations; iteration++) {
00020     cube->clear();
00021     for (i = 0; i < maxOnVoxels; i++) {
00022       p.randomize(Cube::SIZE);
00023       cube->turnVoxelOn(&p);
00024     }
00025     delay(iterationDelay);
00026   }
00027 }
00028
00029 #endif /* __ARDUINO_CUBE_EFFECTS_TURN_ON_RANDOMLY_CPP__ */
```

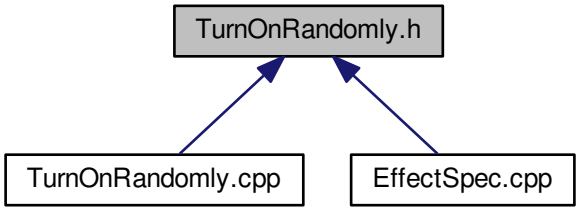
5.135 TurnOnRandomly.h File Reference

```
#include <Effect.h>
#include <Cube.h>
```

Include dependency graph for TurnOnRandomly.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TurnOnRandomly

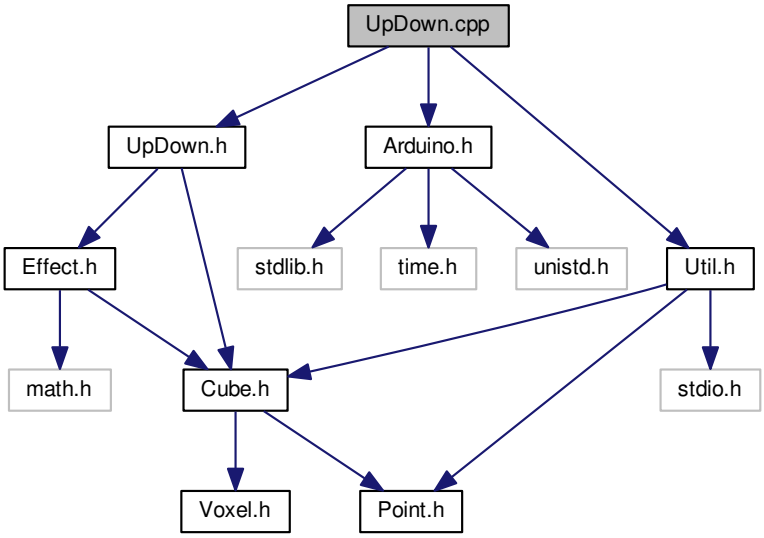
5.136 TurnOnRandomly.h

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_TURN_ON_RANDOMLY_H__
00005 #define __ARDUINO_CUBE_EFFECTS_TURN_ON_RANDOMLY_H__ 1
00006
00007 #include <Effect.h>
00008 #include <Cube.h>
00009
00010 class TurnOnRandomly : public Effect {
00011
00012   unsigned char maxOnVoxels;
00013
00014 public:
00015
00016   TurnOnRandomly(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay, unsigned char maxOnVoxels);
00017
00018   virtual void run();
00019 };
00020
00021 #endif /* __ARDUINO_CUBE_EFFECTS_TURN_ON_RANDOMLY_H__ */
```

5.137 UpDown.cpp File Reference

```
#include <UpDown.h>
#include <Arduino.h>
#include <Util.h>
```

Include dependency graph for UpDown.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_UP_DOWN_CPP__ 1

5.137.1 Macro Definition Documentation

5.137.1.1 #define __ARDUINO_CUBE_EFFECTS_UP_DOWN_CPP__ 1

Definition at line 5 of file UpDown.cpp.

5.138 UpDown.cpp

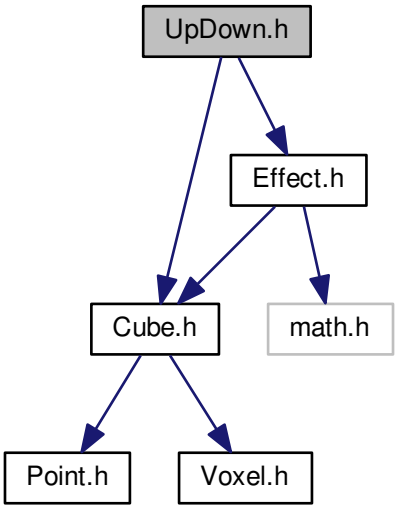
```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_UP_DOWN_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_UP_DOWN_CPP__ 1
00006
00007 #include <UpDown.h>
00008 #include <Arduino.h>
00009 #include <Util.h>
00010
00011 UpDown::UpDown(Cube *cube, unsigned int iterations, unsigned int iterationDelay,
      Axis axis, unsigned char initialPosition) :
00012     Effect(cube, iterations, iterationDelay), axis(axis), initialPosition(initialPosition) {
00013 }
00014
00015 void UpDown::run() {
00016   unsigned char i;
00017   unsigned int iteration;
00018   cube->clear();
00019   for (i = 0; i < Cube::BYTE_SIZE; i++) {
00020     locations[i].position = initialPosition;
00021   }
00022   for (iteration = 0; iteration < iterations; iteration++) {
00023     for (i = 0; i < Cube::BYTE_SIZE; i++) {
00024       locations[i].destination = random(Cube::SIZE);
00025     }
00026     for (i = 0; i < Cube::SIZE; i++) {
00027       move(); draw();
00028       delay(iterationDelay);
```

```
00029     }
00030   }
00031 }
00032
00033 void UpDown::draw() {
00034   unsigned char i, j, p;
00035   cube->useBackBuffer();
00036   cube->clear();
00037   for (i = 0; i < Cube::SIZE; i++) {
00038     for (j = 0; j < Cube::SIZE; j++) {
00039       p = locations[i * Cube::SIZE + j].position;
00040       switch(axis) {
00041         case AXIS_Z:
00042           cube->writeVoxel(i, j, p, ON);
00043           break;
00044         case AXIS_Y:
00045           cube->writeVoxel(i, p, j, ON);
00046           break;
00047         case AXIS_X:
00048           cube->writeVoxel(p, j, i, ON);
00049           break;
00050       }
00051     }
00052   }
00053   cube->swapBuffers();
00054   cube->useFrontBuffer();
00055 }
00056
00057 void UpDown::move() {
00058   unsigned char i;
00059   for (i = 0; i < Cube::BYTE_SIZE; i++) {
00060     Location *location = &(locations[i]);
00061     if (location->position < location->destination) {
00062       location->position++;
00063     }
00064     if (location->position > location->destination) {
00065       location->position--;
00066     }
00067   }
00068 }
00069
00070 #endif /* __ARDUINO_CUBE_EFFECTS_UP_DOWN_CPP__ */
```

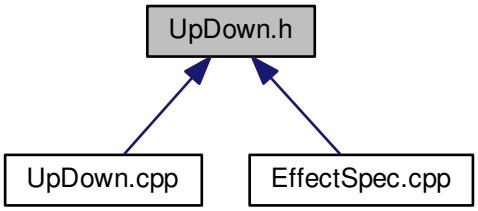
5.139 UpDown.h File Reference

```
#include <Effect.h>
#include <Cube.h>
```

Include dependency graph for UpDown.h:



This graph shows which files directly or indirectly include this file:



Classes

- class UpDown
- struct UpDown::Location

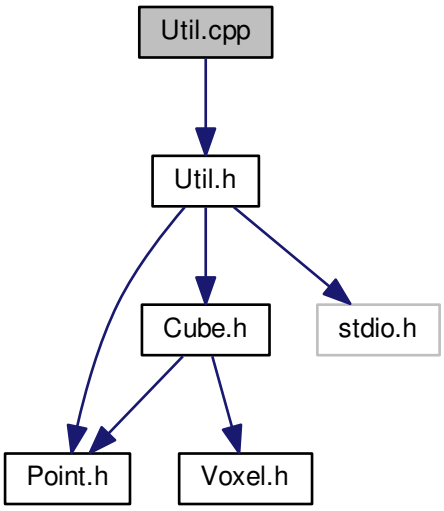
5.140 UpDown.h

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_UP_DOWN_H__
00005 #define __ARDUINO_CUBE_EFFECTS_UP_DOWN_H__ 1
00006
00007 #include <Effect.h>
00008 #include <Cube.h>
00009
00010 class UpDown : public Effect {
00011
```

```
00012   typedef struct {
00013     unsigned char position : 4;
00014     unsigned char destination : 4;
00015   } Location;
00016
00017   Location locations[Cube::BYTE_SIZE];
00018   Axis axis;
00019   unsigned char initialPosition;
00020
00021 public:
00022
00023   UpDown(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay, Axis axis, unsigned char initialPosition);
00024
00025   virtual void run();
00026
00030   void draw();
00031
00035   void move();
00036 };
00037
00038 #endif /* __ARDUINO_CUBE_EFFECTS_UP_DOWN_H__ */
00039
```

5.141 Util.cpp File Reference

`#include <Util.h>`
Include dependency graph for Util.cpp:



Macros

- #define __ARDUINO_CUBE_UTIL_CPP__ 1

5.141.1 Macro Definition Documentation

5.141.1.1 #define __ARDUINO_CUBE_UTIL_CPP__ 1

Definition at line 5 of file Util.cpp.

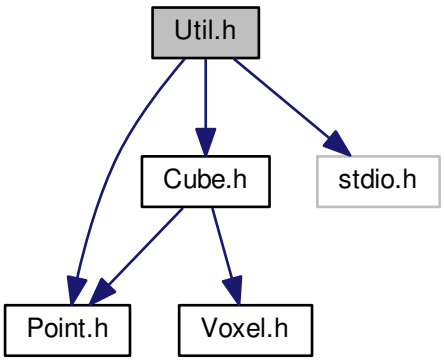
5.142 Util.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_UTIL_CPP__
00005 #define __ARDUINO_CUBE_UTIL_CPP__ 1
00006
00007 #include <Util.h>
00008
00009 unsigned char Util::rotatingShift(unsigned char v, unsigned char isLeft) {
00010   if (isLeft) {
00011     return (v << 1) | (v >> 7 & 0x01);
00012   } else {
00013     return (v >> 1) | (v << 7 & 0x80);
00014   }
00015 }
00016
00017 void Util::flipByte(unsigned char *p) {
00018   unsigned char flop = 0x00;
00019   flop = (flop & 0xfe) | (0x01 & (*p >> 7));
00020   flop = (flop & 0xfd) | (0x02 & (*p >> 5));
00021   flop = (flop & 0xfb) | (0x04 & (*p >> 3));
00022   flop = (flop & 0xf7) | (0x08 & (*p >> 1));
00023   flop = (flop & 0xef) | (0x10 & (*p << 1));
00024   flop = (flop & 0xdf) | (0x20 & (*p << 3));
00025   flop = (flop & 0xbf) | (0x40 & (*p << 5));
00026   flop = (flop & 0x7f) | (0x80 & (*p << 7));
00027   *p = flop;
00028 }
00029
00030 void Util::orderArgs(unsigned char *a, unsigned char *b) {
00031   if (*a > *b) {
00032     swapArgs(a, b);
00033   }
00034 }
00035
00036 void Util::swapArgs(unsigned char *a, unsigned char *b) {
00037   unsigned char _ = *b;
00038   *b = *a;
00039   *a = _;
00040 }
00041
00042 #endif /* __ARDUINO_CUBE_UTIL_CPP__ */
00043
```

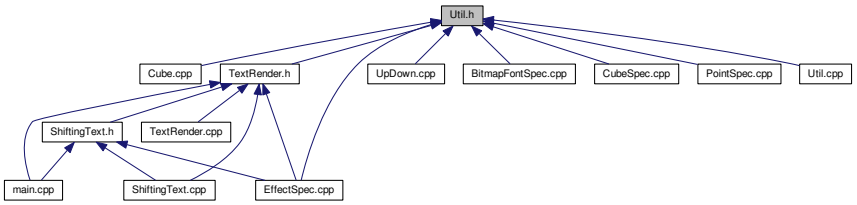
5.143 Util.h File Reference

```
#include <Point.h>
#include <Cube.h>
#include <stdio.h>
```

Include dependency graph for Util.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Util

5.144 Util.h

```
00001
00004 #ifndef __ARDUINO_CUBE_UTIL_H__
00005 #define __ARDUINO_CUBE_UTIL_H__ 1
00006
00007 #include <Point.h>
00008 #include <Cube.h>
00009 #include <stdio.h>
00010
00011 class Util {
00012
00013 public:
00014
00018   static unsigned char rotatingShift(unsigned char v, unsigned char isLeft);
00019
00023   static void flipByte(unsigned char *p);
00024
00028   static void orderArgs(unsigned char *a, unsigned char *b);
00029
00033   static void swapArgs(unsigned char *a, unsigned char *b);
00034
00039   static unsigned char byteLine(unsigned char start, unsigned char end) {
00040     return ((0xff << start) & ~(0xff << (end + 1)));
00041   }
00042
00046   static void set(unsigned char *p, unsigned char mask) {
00047     *p |= mask;
00048   }
00049
00053   static void clr(unsigned char *p, unsigned char mask) {
00054     *p &= ~mask;
00055   }
00056 };
00057
00058 #endif /* __ARDUINO_CUBE_UTIL_H__ */
```

5.145 Voxel.cpp File Reference

Macros

- #define __ARDUINO_CUBE_VOXEL_CPP__ 1

5.145.1 Macro Definition Documentation

5.145.1.1 #define __ARDUINO_CUBE_VOXEL_CPP__ 1

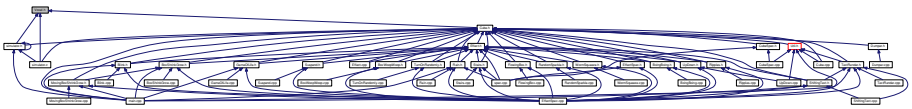
Definition at line 5 of file Voxel.cpp.

5.146 Voxel.cpp

```
00001
00004 #ifndef __ARDUINO_CUBE_VOXEL_CPP__
00005 #define __ARDUINO_CUBE_VOXEL_CPP__ 1
00006
00007 // Nothing
00008
00009 #endif /* __ARDUINO_CUBE_VOXEL_CPP__ */
```

5.147 Voxel.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct Voxel

Enumerations

- enum Axis { AXIS_X = 0x00, AXIS_Y = 0x01, AXIS_Z = 0x02 }
- enum Direction {
  UP = 0x00, DOWN = 0x01, LEFT = 0x02, RIGHT = 0x04,
  FRONT = 0x08, BACK = 0x10 }
- enum State { ON = 0xff, OFF = 0x00 }

5.147.1 Enumeration Type Documentation

5.147.1.1 enum Axis

Enumerator

AXIS_X

AXIS_Y

AXIS_Z

Definition at line 7 of file Voxel.h.

5.147.1.2 enum Direction

Enumerator

UP

DOWN

LEFT

RIGHT

FRONT

BACK

Definition at line 13 of file Voxel.h.

5.147.1.3 enum State

Enumerator

ON

OFF

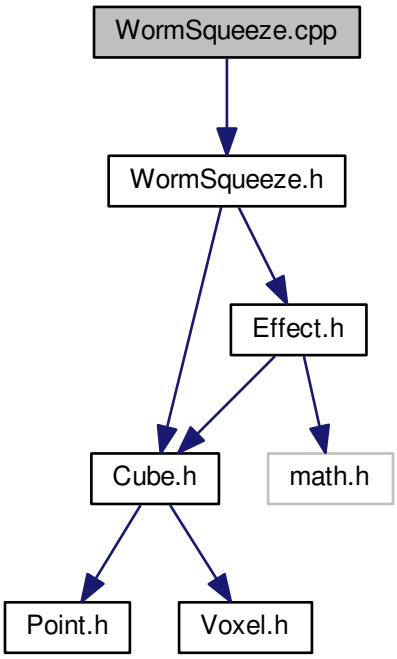
Definition at line 22 of file Voxel.h.

5.148 Voxel.h

```
00001
00004 #ifndef __ARDUINO_CUBE_VOXEL_H__
00005 #define __ARDUINO_CUBE_VOXEL_H__ 1
00006
00007 typedef enum {
00008   AXIS_X = 0x00,
00009   AXIS_Y = 0x01,
00010   AXIS_Z = 0x02
00011 } Axis;
00012
00013 typedef enum {
00014   UP = 0x00,
00015   DOWN = 0x01,
00016   LEFT = 0x02,
00017   RIGHT = 0x04,
00018   FRONT = 0x08,
00019   BACK = 0x10
00020 } Direction;
00021
00022 typedef enum {
00023   ON = 0xff,
00024   OFF = 0x00
00025 } State;
00026
00027 typedef struct {
00028   unsigned char state;
00029 } Voxel;
00030
00031
00032 #endif /* __ARDUINO_CUBE_VOXEL_H__ */
```

5.149 WormSqueeze.cpp File Reference

`#include <WormSqueeze.h>`
Include dependency graph for WormSqueeze.cpp:



Macros

- #define __ARDUINO_CUBE_EFFECTS_WORM_SQUEEZE_CPP__ 1

5.149.1 Macro Definition Documentation

5.149.1.1 #define __ARDUINO_CUBE_EFFECTS_WORM_SQUEEZE_CPP__ 1

Definition at line 5 of file WormSqueeze.cpp.

5.150 WormSqueeze.cpp

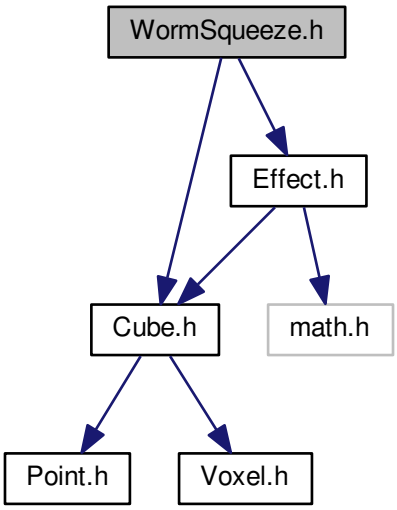
```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_WORM_SQUEEZE_CPP__
00005 #define __ARDUINO_CUBE_EFFECTS_WORM_SQUEEZE_CPP__ 1
00006
00007 #include <WormSqueeze.h>
00008
00009 WormSqueeze::WormSqueeze(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay) :
00010     Effect(cube, iterations, iterationDelay) {
00011 }
00012
00013 void WormSqueeze::run() {
00014   unsigned int iteration;
00015   for (iteration = 0; iteration < iterations; iteration++) {
00016   }
```

```
00017 }
00018
00019 #endif /* __ARDUINO_CUBE_EFFECTS_WORM_SQUEEZE_CPP__ */
```

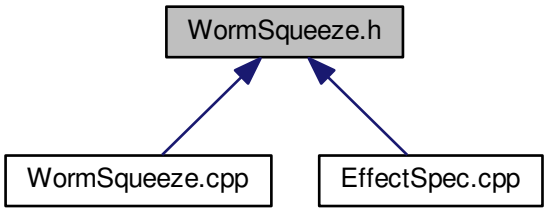
5.151 WormSqueeze.h File Reference

```
#include <Effect.h>
#include <Cube.h>
```

Include dependency graph for WormSqueeze.h:



This graph shows which files directly or indirectly include this file:



Classes

- class WormSqueeze

5.152 WormSqueeze.h

```
00001
00004 #ifndef __ARDUINO_CUBE_EFFECTS_WORM_SQUEEZE_H__
00005 #define __ARDUINO_CUBE_EFFECTS_WORM_SQUEEZE_H__ 1
00006
00007 #include <Effect.h>
00008 #include <Cube.h>
00009
00010 class WormSqueeze : public Effect {
00011 public:
00012
00013   WormSqueeze(Cube *cube, unsigned int iterations, unsigned int
      iterationDelay);
00014
00015   virtual void run();
00016 };
00017
00018 #endif /* __ARDUINO_CUBE_EFFECTS_WORM_SQUEEZE_H__ */
```

Index