RoomKit

Hypar Inc.

Version 0.3.0

Tue Mar 5 2019

# Namespace Documentation

## RoomKit Namespace Reference

### Classes

class **ArcEx**

*Extends Elements.Geometry.Arc with utility methods.*

class [CoordGrid](#AAAAAAAAAB)

*Maintains a list of available and allocated points in a grid of the specified interval within the orthogonal bounding box of a Polygon.*

class **LineEx**

*Extends Elements.Geometry.Line with utility methods.*

class **Messages**

*Common exception messages.*

class **Palette**

*Commonly used colors for Space rendering. These colors are translucent to allow viewing of representions several layers deep.*

class **Place**

*Rooms 2D Polygons in various spatial relationships to each other.*

class **PolygonEx**

class [Room](#AAAAAAAAAC)

*A data structure recording room characteristics.*

class [RoomGroup](#AAAAAAAAAD)

*Creates and manages Rooms within a perimeter.*

class [RoomRow](#AAAAAAAAAE)

*Creates and manages Rooms placed along a line.*

class [Scope](#AAAAAAAAAF)

*Data structure recording space program characteristics and the status of a* [*Room*](#AAAAAAAAAC) *placing process.*

class **Shaper**

*Utilities for creating and editing Polygons.*

class [Story](#AAAAAAAAAG)

*Creates and manages the geometry of a slab and Rooms representing corridors, occupied rooms, and services.*

class [TopoBox](#AAAAAAAAAH)

*Maintains a set of points on the orthogonal bounding box of a supplied Polygon corresponding to four divisions of each side.*

class [Tower](#AAAAAAAAAI)

class **Vector3Ex**

*Extends Elements.Geometry.Vector3 with utility methods.*

### Enumerations

enum [Corner](#AAAAAAAAAJ) { **NE**, **SE**, **SW**, **NW** }

*A list of box corners as compass designations. NE = maximum X and Y corner. SE = maximum X and minimum Y corner. SW = minimum X and Y corner. NW = minimum X and maximum Y corner.* enum [Orient](#AAAAAAAAAK) { **C**, **N**, **NNE**, **NE**, **ENE**, **E**, **ESE**, **SE**, **SSE**, **S**, **SSW**, **SW**, **WSW**, **W**, **WNW**, **NW**, **NNW** }

*A list of compass orientations used to designate locations on a 2D box. N, S, E, and W define middle points on each orthogonal side of the box. NE, NW, SE, and SW correspond to the corners of the box. C corresponds to the center of the box. Other compass points define locations along the relevant side between the cardinal and corner points. See documentation of corresponding properties of the TopoBox class for full documentation.*

### Enumeration Type Documentation

#### enum [RoomKit.Corner](#AAAAAAAAAJ)[strong]

A list of box corners as compass designations. NE = maximum X and Y corner. SE = maximum X and minimum Y corner. SW = minimum X and Y corner. NW = minimum X and maximum Y corner.

#### enum [RoomKit.Orient](#AAAAAAAAAK)[strong]

A list of compass orientations used to designate locations on a 2D box. N, S, E, and W define middle points on each orthogonal side of the box. NE, NW, SE, and SW correspond to the corners of the box. C corresponds to the center of the box. Other compass points define locations along the relevant side between the cardinal and corner points. See documentation of corresponding properties of the [TopoBox](#AAAAAAAAAH) class for full documentation.

# Class Documentation

## RoomKit.CoordGrid Class Reference

Maintains a list of available and allocated points in a grid of the specified interval within the orthogonal bounding box of a Polygon.

### Public Member Functions

[CoordGrid](#AAAAAAAAAL) (Polygon polygon, double xInterval=1, double yInterval=1, int randomSeed=1)

*Creates an orthogonal 2D grid of Vector3 points from the supplied Polygon and axis intervals.*

void [Allocate](#AAAAAAAAAM) (Polygon polygon)

*Allocates the points in the grid falling within or on the supplied Polygon.*

void [Allocate](#AAAAAAAAAN) (IList< Polygon > polygons)

*Allocates points in the grid falling within the supplied Polygons.*

Vector3 [AllocatedNearTo](#AAAAAAAAAO) (Vector3 point)

*Returns the allocated grid point nearest to the supplied point.*

Vector3 [AllocatedRandom](#AAAAAAAAAP) ()

*Returns a random allocated point.*

Vector3 [AvailableMax](#AAAAAAAAAQ) ()

*Returns the maximum available grid point.*

Vector3 [AvailableMin](#AAAAAAAAAR) ()

*Returns the minimum available grid point.*

Vector3 [AvailableNearTo](#AAAAAAAAAS) (Vector3 point)

*Returns the available grid point nearest to the supplied Vector3 point.*

Vector3 [AvailableRandom](#AAAAAAAAAT) ()

*Returns a random available grid point.*

### Properties

List< Vector3 > [Allocated](#AAAAAAAAAU) [get]

*The list of vector3 allocated points.*

List< Vector3 > [Available](#AAAAAAAAAV) [get]

*The list of Vector3 points available for allocation.*

Polygon **Perimeter** [get, set]

### Detailed Description

Maintains a list of available and allocated points in a grid of the specified interval within the orthogonal bounding box of a Polygon.

### Constructor & Destructor Documentation

#### RoomKit.CoordGrid.CoordGrid (Polygon *polygon*, double *xInterval* = 1, double *yInterval* = 1, int *randomSeed* = 1)

Creates an orthogonal 2D grid of Vector3 points from the supplied Polygon and axis intervals.

##### Parameters:

|  |  |
| --- | --- |
| *perimeter* | The Polygon boundary of the point grid. |
| *xInterval* | The spacing of the grid along the x-axis. |
| *yInterval* | The spacing of the grid along the y-axis. |

##### Returns:

A new [CoordGrid](#AAAAAAAAAB).

### Member Function Documentation

#### void RoomKit.CoordGrid.Allocate (Polygon *polygon*)

Allocates the points in the grid falling within or on the supplied Polygon.

##### Parameters:

|  |  |
| --- | --- |
| *polygon* | The Polygon bounding the points to be allocated. |

##### Returns:

None.

#### void RoomKit.CoordGrid.Allocate (IList< Polygon > *polygons*)

Allocates points in the grid falling within the supplied Polygons.

##### Parameters:

|  |  |
| --- | --- |
| *polygon* | The Polygon bounding the points to be allocated. |

##### Returns:

None.

#### Vector3 RoomKit.CoordGrid.AllocatedNearTo (Vector3 *point*)

Returns the allocated grid point nearest to the supplied point.

##### Parameters:

|  |  |
| --- | --- |
| *point* | The Vector3 point to compare. |

##### Returns:

A Vector3 point.

#### Vector3 RoomKit.CoordGrid.AllocatedRandom ()

Returns a random allocated point.

##### Returns:

A Vector3 point.

#### Vector3 RoomKit.CoordGrid.AvailableMax ()

Returns the maximum available grid point.

##### Returns:

A Vector3 point.

#### Vector3 RoomKit.CoordGrid.AvailableMin ()

Returns the minimum available grid point.

##### Returns:

A Vector3 point.

#### Vector3 RoomKit.CoordGrid.AvailableNearTo (Vector3 *point*)

Returns the available grid point nearest to the supplied Vector3 point.

##### Parameters:

|  |  |
| --- | --- |
| *point* | The Vector3 point to compare. |

##### Returns:

A Vector3 point.

#### Vector3 RoomKit.CoordGrid.AvailableRandom ()

Returns a random available grid point.

##### Returns:

A Vector3 point.

### Property Documentation

#### List<Vector3> RoomKit.CoordGrid.Allocated[get]

The list of vector3 allocated points.

#### List<Vector3> RoomKit.CoordGrid.Available[get]

The list of Vector3 points available for allocation.

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

RoomKit/CoordGrid.cs

## RoomKit.Room Class Reference

A data structure recording room characteristics.

### Public Member Functions

[Room](#AAAAAAAAAX) ()

*Constructor setting all internal variables to default values to create a 1.0 x 1.0 x 1.0 white cube with no required adjacencies placed on the zero plane with an empty string, null perimeter, and an integer TypeID of -1.*

Polygon [MoveFromTo](#AAAAAAAAAY) (Vector3 from, Vector3 to)

*Moves the* [*Room*](#AAAAAAAAAC) *along a 3D vector calculated between the supplied Vector3 points.*

bool [Rotate](#AAAAAAAAAZ) (Vector3 pivot, double angle)

*Rotates the* [*Room*](#AAAAAAAAAC) *Perimeter in the horizontal plane around the supplied pivot point.*

bool [SetDimensions](#AAAAAAAABA) (Vector3 xyz, Vector3 moveTo=null)

*Creates and sets a rectangular* [*Room*](#AAAAAAAAAC) *Perimeter, Height, and southwest corner location with a supplied vectors. Sets the DesignX and DesignY properties.*

bool [SetPerimeter](#AAAAAAAABB) (Vector3 moveTo=null)

*Creates and sets a rectangular* [*Room*](#AAAAAAAAAC) *Perimeter with dimensions derived from* [*Room*](#AAAAAAAAAC) *characteristics with its southwest corner at the origin or at the 2D location implied by the supplied Vector3.*

bool [SetPerimeter](#AAAAAAAABC) (double area, double ratio=1.5, Vector3 moveTo=null)

*Creates and sets a rectangular* [*Room*](#AAAAAAAAAC) *Perimeter with dimensions derived from* [*Room*](#AAAAAAAAAC) *characteristics with its southwest corner at the supplied Vector3 point. If no point is supplied, the southwest corner is placed at the origin.*

bool [SetPerimeter](#AAAAAAAABD) (Line axis, double width)

*Creates and sets a rectangular* [*Room*](#AAAAAAAAAC) *perimeter with dimensions derived from a supplied Line and a width. Intended for creating corridors.*

bool [SetPerimeter](#AAAAAAAABE) (Vector3 start, Vector3 end, double width)

*Creates and sets a rectangular* [*Room*](#AAAAAAAAAC) *perimeter with dimensions derived from two points and a width. Intended for creating corridors.*

### Properties

int [] [AdjacentTo](#AAAAAAAABF) [get, set]

*A list of Resource ID integers indicating the desired adjacencies of this* [*Room*](#AAAAAAAAAC) *type to other* [*Room*](#AAAAAAAAAC) *types.*

double [Area](#AAAAAAAABG) [get]

*The area of the room's perimeter Polygon. Returns -1.0 if the* [*Room*](#AAAAAAAAAC)*'s Perimeter is null.*

double [AreaVariance](#AAAAAAAABH) [get]

*The ratio between the intended area and the actual area of the* [*Room*](#AAAAAAAAAC)*. Returns a negative value if the* [*Room*](#AAAAAAAAAC) *has no Perimeter value.*

Space [AsSpace](#AAAAAAAABI) [get]

*A Space created from* [*Room*](#AAAAAAAAAC) *characteristics. Adds properties to the Space recording Name TypeID as Type DesignArea as Design Area DesignX as Design Length DesignY as Design Width Perimeter.Area as Area Elevation Height*

Color **Color** [get, set]

double **DesignArea** [get, set]

double [DesignLength](#AAAAAAAABL) [get, set]

*Desired x-axis dimension of this* [*Room*](#AAAAAAAAAC)*.*

double [DesignWidth](#AAAAAAAABM) [get, set]

*Desired y-axis dimension of this* [*Room*](#AAAAAAAAAC)*.*

double **DesignRatio** [get, set]

bool [DesignSet](#AAAAAAAABO) [get]

*Returns true if both DesignLength and DesignWidth are positive values.*

Vector3 **DesignXYZ** [get, set]

double [Elevation](#AAAAAAAABQ) [get, set]

*The vertical position of the* [*Room*](#AAAAAAAAAC)*'s lowest plane, parallel to the ground plane.*

double [Height](#AAAAAAAABR) [get, set]

*Height of the* [*Room*](#AAAAAAAAAC) *prism. Set ignores non-positive values.*

string [Name](#AAAAAAAABS) [get, set]

*Arbitrary string identifier for this* [*Room*](#AAAAAAAAAC) *instance.*

Polygon **Perimeter** [get, set]

bool [Placed](#AAAAAAAABU) [get, set]

*Manual flag to record if the* [*Room*](#AAAAAAAAAC) *has been placed in its final position.*

double [SizeX](#AAAAAAAABV) [get]

*X dimensions of the* [*Room*](#AAAAAAAAAC) *Perimeter orthogonal bounding box.*

double [SizeY](#AAAAAAAABW) [get]

*X dimensions of the* [*Room*](#AAAAAAAAAC) *Perimeter orthogonal bounding box.*

int [TypeID](#AAAAAAAABX) [get, set]

*Arbitrary integer identifier of this* [*Room*](#AAAAAAAAAC) *type. Can be used to identify desired adjacencies.*

string [UniqueID](#AAAAAAAABY) [get]

*UUID for this* [*Room*](#AAAAAAAAAC) *instance, set on initialization.*

### Detailed Description

A data structure recording room characteristics.

### Constructor & Destructor Documentation

#### RoomKit.Room.Room ()

Constructor setting all internal variables to default values to create a 1.0 x 1.0 x 1.0 white cube with no required adjacencies placed on the zero plane with an empty string, null perimeter, and an integer TypeID of -1.

### Member Function Documentation

#### Polygon RoomKit.Room.MoveFromTo (Vector3 *from*, Vector3 *to*)

Moves the [Room](#AAAAAAAAAC) along a 3D vector calculated between the supplied Vector3 points.

##### Parameters:

|  |  |
| --- | --- |
| *from* | Vector3 base point of the move. |
| *to* | Vector3 target point of the move. |

##### Returns:

A Polygon represeting the [Room](#AAAAAAAAAC)'s new Perimeter.

#### bool RoomKit.Room.Rotate (Vector3 *pivot*, double *angle*)

Rotates the [Room](#AAAAAAAAAC) Perimeter in the horizontal plane around the supplied pivot point.

##### Parameters:

|  |  |
| --- | --- |
| *pivot* | Vector3 point around which the [Room](#AAAAAAAAAC) Perimeter will be rotated. |
| *angle* | Angle in degrees to rotate the Perimeter. |

##### Returns:

True if the Perimeter is successfully rotated.

#### bool RoomKit.Room.SetDimensions (Vector3 *xyz*, Vector3 *moveTo* = null)

Creates and sets a rectangular [Room](#AAAAAAAAAC) Perimeter, Height, and southwest corner location with a supplied vectors. Sets the DesignX and DesignY properties.

##### Parameters:

|  |  |
| --- | --- |
| *xyz* | Vector3 dimensions of a new Polygon Perimeter. If xy.Z is > 0.0, sets the height of the [Room](#AAAAAAAAAC). |
| *moveTo* | Vector3 location of the new Polygon's southwest corner. |

##### Returns:

True if the Perimeter is successfully set.

#### bool RoomKit.Room.SetPerimeter (Vector3 *moveTo* = null)

Creates and sets a rectangular [Room](#AAAAAAAAAC) Perimeter with dimensions derived from [Room](#AAAAAAAAAC) characteristics with its southwest corner at the origin or at the 2D location implied by the supplied Vector3.

##### Returns:

True if the Perimeter is successfully set.

#### bool RoomKit.Room.SetPerimeter (double *area*, double *ratio* = 1.5, Vector3 *moveTo* = null)

Creates and sets a rectangular [Room](#AAAAAAAAAC) Perimeter with dimensions derived from [Room](#AAAAAAAAAC) characteristics with its southwest corner at the supplied Vector3 point. If no point is supplied, the southwest corner is placed at the origin.

##### Parameters:

|  |  |
| --- | --- |
| *area* | Area override for the new [Room](#AAAAAAAAAC) Perimeter. If zero, defaults to the value of DesignArea. |
| *ratio* | Desired ratio of X to Y [Room](#AAAAAAAAAC) dimensions. |
| *moveTo* | Vector3 location of the new Polygon's southwest corner. |

##### Returns:

True if the Perimeter is successfully set.

#### bool RoomKit.Room.SetPerimeter (Line *axis*, double *width*)

Creates and sets a rectangular [Room](#AAAAAAAAAC) perimeter with dimensions derived from a supplied Line and a width. Intended for creating corridors.

##### Parameters:

|  |  |
| --- | --- |
| *axis* | The Line defining the centerline of the perimeter. |
| *width* | The width of the perimeter along the axis Line. |

##### Returns:

True if the Perimeter is successfully set.

#### bool RoomKit.Room.SetPerimeter (Vector3 *start*, Vector3 *end*, double *width*)

Creates and sets a rectangular [Room](#AAAAAAAAAC) perimeter with dimensions derived from two points and a width. Intended for creating corridors.

##### Parameters:

|  |  |
| --- | --- |
| *start* | The start point of an axis defining centerline of the perimeter. |
| *end* | The end point of an axis defining centerline of the perimeter. |
| *width* | The width of the perimeter along the axis Line. |

##### Returns:

True if the Perimeter is successfully set.

### Property Documentation

#### int [] RoomKit.Room.AdjacentTo[get], [set]

A list of Resource ID integers indicating the desired adjacencies of this [Room](#AAAAAAAAAC) type to other [Room](#AAAAAAAAAC) types.

#### double RoomKit.Room.Area[get]

The area of the room's perimeter Polygon. Returns -1.0 if the [Room](#AAAAAAAAAC)'s Perimeter is null.

#### double RoomKit.Room.AreaVariance[get]

The ratio between the intended area and the actual area of the [Room](#AAAAAAAAAC). Returns a negative value if the [Room](#AAAAAAAAAC) has no Perimeter value.

#### Space RoomKit.Room.AsSpace[get]

A Space created from [Room](#AAAAAAAAAC) characteristics. Adds properties to the Space recording Name TypeID as Type DesignArea as Design Area DesignX as Design Length DesignY as Design Width Perimeter.Area as Area Elevation Height

#### double RoomKit.Room.DesignLength[get], [set]

Desired x-axis dimension of this [Room](#AAAAAAAAAC).

#### bool RoomKit.Room.DesignSet[get]

Returns true if both DesignLength and DesignWidth are positive values.

#### double RoomKit.Room.DesignWidth[get], [set]

Desired y-axis dimension of this [Room](#AAAAAAAAAC).

#### double RoomKit.Room.Elevation[get], [set]

The vertical position of the [Room](#AAAAAAAAAC)'s lowest plane, parallel to the ground plane.

#### double RoomKit.Room.Height[get], [set]

Height of the [Room](#AAAAAAAAAC) prism. Set ignores non-positive values.

#### string RoomKit.Room.Name[get], [set]

Arbitrary string identifier for this [Room](#AAAAAAAAAC) instance.

#### bool RoomKit.Room.Placed[get], [set]

Manual flag to record if the [Room](#AAAAAAAAAC) has been placed in its final position.

#### double RoomKit.Room.SizeX[get]

X dimensions of the [Room](#AAAAAAAAAC) Perimeter orthogonal bounding box.

#### double RoomKit.Room.SizeY[get]

X dimensions of the [Room](#AAAAAAAAAC) Perimeter orthogonal bounding box.

#### int RoomKit.Room.TypeID[get], [set]

Arbitrary integer identifier of this [Room](#AAAAAAAAAC) type. Can be used to identify desired adjacencies.

#### string RoomKit.Room.UniqueID[get]

UUID for this [Room](#AAAAAAAAAC) instance, set on initialization.

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

RoomKit/Room.cs

## RoomKit.RoomGroup Class Reference

Creates and manages Rooms within a perimeter.

### Public Member Functions

[RoomGroup](#AAAAAAAABZ) ()

*Creates an empty group of Rooms.*

void [MoveFromTo](#AAAAAAAACA) (Vector3 from, Vector3 to)

*Moves all Rooms in the* [*RoomGroup*](#AAAAAAAAAD) *and the* [*RoomGroup*](#AAAAAAAAAD) *Perimeter along a 3D vector calculated between the supplied Vector3 points.*

void [Rotate](#AAAAAAAACB) (Vector3 pivot, double angle)

*Rotates the* [*RoomGroup*](#AAAAAAAAAD) *Perimeter and Rooms in the horizontal plane around the supplied pivot point.*

void [SetColor](#AAAAAAAACC) (Color color)

*Uniformly sets the color of all Rooms in the* [*RoomGroup*](#AAAAAAAAAD)*.*

void [SetHeight](#AAAAAAAACD) (double height)

*Uniformly sets the height of all Rooms in the* [*RoomGroup*](#AAAAAAAAAD)*.*

bool [RoomsByDivision](#AAAAAAAACE) (int xRooms=1, int yRooms=1, double height=3.0)

*Clears the current Rooms list and creates new Rooms defined by orthogonal x- and y-axis divisions of the* [*RoomGroup*](#AAAAAAAAAD) *Perimeter.*

### Properties

double [AreaAvailable](#AAAAAAAACF) [get]

*Unallocated area of the* [*RoomGroup*](#AAAAAAAAAD) *perimeter.*

double [AreaPlaced](#AAAAAAAACG) [get]

*Area allocated within the* [*RoomGroup*](#AAAAAAAAAD)*.*

double **Elevation** [get, set]

string [Name](#AAAAAAAACI) [get, set]

*Arbitrary string identifier for this* [*RoomGroup*](#AAAAAAAAAD)*.*

Polygon **Perimeter** [get, set]

List< [Room](#AAAAAAAAAC) > [Rooms](#AAAAAAAACK) [get]

*List of Rooms placed within the Perimeter.*

List< Polygon > [RoomsAsPolygons](#AAAAAAAACL) [get]

*List of all* [*Room*](#AAAAAAAAAC) *perimeters as Polygons.*

List< Space > [RoomsAsSpaces](#AAAAAAAACM) [get]

*List of all Rooms as Spaces.*

double [SizeX](#AAAAAAAACN) [get]

*X dimension of the Perimeter orthogonal bounding box.*

double [SizeY](#AAAAAAAACO) [get]

*Y dimension of the Perimeter orthogonal bounding box.*

string [UniqueID](#AAAAAAAACP) [get]

*UUID for this* [*RoomGroup*](#AAAAAAAAAD) *instance, set on initialization.*

### Detailed Description

Creates and manages Rooms within a perimeter.

### Constructor & Destructor Documentation

#### RoomKit.RoomGroup.RoomGroup ()

Creates an empty group of Rooms.

##### Returns:

A new [RoomGroup](#AAAAAAAAAD).

### Member Function Documentation

#### void RoomKit.RoomGroup.MoveFromTo (Vector3 *from*, Vector3 *to*)

Moves all Rooms in the [RoomGroup](#AAAAAAAAAD) and the [RoomGroup](#AAAAAAAAAD) Perimeter along a 3D vector calculated between the supplied Vector3 points.

##### Parameters:

|  |  |
| --- | --- |
| *from* | Vector3 base point of the move. |
| *to* | Vector3 target point of the move. |

##### Returns:

None.

#### bool RoomKit.RoomGroup.RoomsByDivision (int *xRooms* = 1, int *yRooms* = 1, double *height* = 3.0)

Clears the current Rooms list and creates new Rooms defined by orthogonal x- and y-axis divisions of the [RoomGroup](#AAAAAAAAAD) Perimeter.

##### Parameters:

|  |  |
| --- | --- |
| *xRooms* | The quantity of Rooms along orthogonal x-axis. Must be positive. |
| *yRooms* | The quantity of Rooms along orthogonal y-axis. Must be positive. |

##### Returns:

True if the Rooms are created.

#### void RoomKit.RoomGroup.Rotate (Vector3 *pivot*, double *angle*)

Rotates the [RoomGroup](#AAAAAAAAAD) Perimeter and Rooms in the horizontal plane around the supplied pivot point.

##### Parameters:

|  |  |
| --- | --- |
| *pivot* | Vector3 point around which the [Room](#AAAAAAAAAC) Perimeter will be rotated. |
| *angle* | Angle in degrees to rotate the Perimeter. |

##### Returns:

None.

#### void RoomKit.RoomGroup.SetColor (Color *color*)

Uniformly sets the color of all Rooms in the [RoomGroup](#AAAAAAAAAD).

##### Parameters:

|  |  |
| --- | --- |
| *color* | The new color of the Rooms. |

##### Returns:

None.

#### void RoomKit.RoomGroup.SetHeight (double *height*)

Uniformly sets the height of all Rooms in the [RoomGroup](#AAAAAAAAAD).

##### Parameters:

|  |  |
| --- | --- |
| *elevation* | The new height of the Rooms. |

##### Returns:

None.

### Property Documentation

#### double RoomKit.RoomGroup.AreaAvailable[get]

Unallocated area of the [RoomGroup](#AAAAAAAAAD) perimeter.

#### double RoomKit.RoomGroup.AreaPlaced[get]

Area allocated within the [RoomGroup](#AAAAAAAAAD).

#### string RoomKit.RoomGroup.Name[get], [set]

Arbitrary string identifier for this [RoomGroup](#AAAAAAAAAD).

#### List<[Room](#AAAAAAAAAC)> RoomKit.RoomGroup.Rooms[get]

List of Rooms placed within the Perimeter.

#### List<Polygon> RoomKit.RoomGroup.RoomsAsPolygons[get]

List of all [Room](#AAAAAAAAAC) perimeters as Polygons.

#### List<Space> RoomKit.RoomGroup.RoomsAsSpaces[get]

List of all Rooms as Spaces.

#### double RoomKit.RoomGroup.SizeX[get]

X dimension of the Perimeter orthogonal bounding box.

#### double RoomKit.RoomGroup.SizeY[get]

Y dimension of the Perimeter orthogonal bounding box.

#### string RoomKit.RoomGroup.UniqueID[get]

UUID for this [RoomGroup](#AAAAAAAAAD) instance, set on initialization.

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

RoomKit/RoomGroup.cs

## RoomKit.RoomRow Class Reference

Creates and manages Rooms placed along a line.

### Public Member Functions

[RoomRow](#AAAAAAAACQ) (Line row)

*Constructor initializes the* [*RoomRow*](#AAAAAAAAAE) *with a new Line.*

[RoomRow](#AAAAAAAACR) (Vector3 start, Vector3 end)

*Constructor initializes the* [*RoomRow*](#AAAAAAAAAE) *with line endpoints.*

bool [AddRoom](#AAAAAAAACS) ([Room](#AAAAAAAAAC) room, Polygon within=null, IList< Polygon > among=null)

*Attempts to place a* [*Room*](#AAAAAAAAAC) *perimeter on the next open segment of the Row, with optional restrictions of a perimeter within which the* [*Room*](#AAAAAAAAAC)*'s perimeter must fit and a list of Polygons with which it cannot intersect.*

void [MoveFromTo](#AAAAAAAACT) (Vector3 from, Vector3 to)

*Moves all Rooms in the* [*RoomRow*](#AAAAAAAAAE) *and the* [*RoomRow*](#AAAAAAAAAE) *Row along a 3D vector calculated between the supplied Vector3 points.*

void [Rotate](#AAAAAAAACU) (Vector3 pivot, double angle)

*Rotates the* [*RoomRow*](#AAAAAAAAAE) *Row and Rooms in the horizontal plane around the supplied pivot point.*

void [SetColor](#AAAAAAAACV) (Color color)

*Uniformly sets the color of all Rooms in the* [*RoomRow*](#AAAAAAAAAE)*.*

void [SetHeight](#AAAAAAAACW) (double height)

*Uniformly sets the height of all Rooms in the* [*RoomRow*](#AAAAAAAAAE)*.*

### Properties

double [AreaPlaced](#AAAAAAAACX) [get]

*Aggregate area of the Rooms placed on this Row.*

double [AvailableLength](#AAAAAAAACY) [get]

*Unallocated length of the* [*RoomRow*](#AAAAAAAAAE)*.*

Polygon [Circulation](#AAAAAAAACZ) [get]

*Circulation envelope around the row.*

double **CirculationWidth** [get, set]

double [Depth](#AAAAAAAADB) = 0.0 [get]

*Depth of the deepest room along the Row.*

double **Elevation** [get, set]

string [Name](#AAAAAAAADD) [get, set]

*Arbitrary string identifier for this* [*RoomRow*](#AAAAAAAAAE)*.*

IList< [Room](#AAAAAAAAAC) > [Rooms](#AAAAAAAADE) [get]

*List of Rooms placed along the Row.*

List< Polygon > [RoomsAsPolygons](#AAAAAAAADF) [get]

*List of all* [*Room*](#AAAAAAAAAC) *perimeters as Polygons.*

List< Space > [RoomsAsSpaces](#AAAAAAAADG) [get]

*List of all Rooms as Spaces.*

Line [Row](#AAAAAAAADH) [get]

*Line along which Rooms can be placed.*

double [SizeX](#AAAAAAAADI) [get]

*X dimension of the Circulation orthogonal bounding box.*

double [SizeY](#AAAAAAAADJ) [get]

*Y dimension of the Circulation orthogonal bounding box.*

string [UniqueID](#AAAAAAAADK) [get]

*UUID for this* [*RoomRow*](#AAAAAAAAAE) *instance, set on initialization.*

### Detailed Description

Creates and manages Rooms placed along a line.

### Constructor & Destructor Documentation

#### RoomKit.RoomRow.RoomRow (Line *row*)

Constructor initializes the [RoomRow](#AAAAAAAAAE) with a new Line.

#### RoomKit.RoomRow.RoomRow (Vector3 *start*, Vector3 *end*)

Constructor initializes the [RoomRow](#AAAAAAAAAE) with line endpoints.

### Member Function Documentation

#### bool RoomKit.RoomRow.AddRoom ([Room](#AAAAAAAAAC) *room*, Polygon *within* = null, IList< Polygon > *among* = null)

Attempts to place a [Room](#AAAAAAAAAC) perimeter on the next open segment of the Row, with optional restrictions of a perimeter within which the [Room](#AAAAAAAAAC)'s perimeter must fit and a list of Polygons with which it cannot intersect.

##### Parameters:

|  |  |
| --- | --- |
| *room* | [Room](#AAAAAAAAAC) from which to derive the Polygon to place. |
| *within* | Polygon perimeter within which a new [Room](#AAAAAAAAAC) must fit. |
| *among* | List of Polygon perimeters the new [Room](#AAAAAAAAAC) cannot intersect. |

##### Returns:

True if the room was successfully placed.

#### void RoomKit.RoomRow.MoveFromTo (Vector3 *from*, Vector3 *to*)

Moves all Rooms in the [RoomRow](#AAAAAAAAAE) and the [RoomRow](#AAAAAAAAAE) Row along a 3D vector calculated between the supplied Vector3 points.

##### Parameters:

|  |  |
| --- | --- |
| *from* | Vector3 base point of the move. |
| *to* | Vector3 target point of the move. |

##### Returns:

None.

#### void RoomKit.RoomRow.Rotate (Vector3 *pivot*, double *angle*)

Rotates the [RoomRow](#AAAAAAAAAE) Row and Rooms in the horizontal plane around the supplied pivot point.

##### Parameters:

|  |  |
| --- | --- |
| *pivot* | Vector3 point around which the [Room](#AAAAAAAAAC) Perimeter will be rotated. |
| *angle* | Angle in degrees to rotate the Perimeter. |

##### Returns:

None.

#### void RoomKit.RoomRow.SetColor (Color *color*)

Uniformly sets the color of all Rooms in the [RoomRow](#AAAAAAAAAE).

##### Parameters:

|  |  |
| --- | --- |
| *color* | New color of the Rooms. |

##### Returns:

None.

#### void RoomKit.RoomRow.SetHeight (double *height*)

Uniformly sets the height of all Rooms in the [RoomRow](#AAAAAAAAAE).

##### Parameters:

|  |  |
| --- | --- |
| *elevation* | New height of the Rooms. |

##### Returns:

None.

### Property Documentation

#### double RoomKit.RoomRow.AreaPlaced[get]

Aggregate area of the Rooms placed on this Row.

#### double RoomKit.RoomRow.AvailableLength[get]

Unallocated length of the [RoomRow](#AAAAAAAAAE).

#### Polygon RoomKit.RoomRow.Circulation[get]

Circulation envelope around the row.

#### double RoomKit.RoomRow.Depth = 0.0[get]

Depth of the deepest room along the Row.

#### string RoomKit.RoomRow.Name[get], [set]

Arbitrary string identifier for this [RoomRow](#AAAAAAAAAE).

#### IList<[Room](#AAAAAAAAAC)> RoomKit.RoomRow.Rooms[get]

List of Rooms placed along the Row.

#### List<Polygon> RoomKit.RoomRow.RoomsAsPolygons[get]

List of all [Room](#AAAAAAAAAC) perimeters as Polygons.

#### List<Space> RoomKit.RoomRow.RoomsAsSpaces[get]

List of all Rooms as Spaces.

#### Line RoomKit.RoomRow.Row[get]

Line along which Rooms can be placed.

#### double RoomKit.RoomRow.SizeX[get]

X dimension of the Circulation orthogonal bounding box.

#### double RoomKit.RoomRow.SizeY[get]

Y dimension of the Circulation orthogonal bounding box.

#### string RoomKit.RoomRow.UniqueID[get]

UUID for this [RoomRow](#AAAAAAAAAE) instance, set on initialization.

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

RoomKit/RoomRow.cs

## RoomKit.Scope Class Reference

Data structure recording space program characteristics and the status of a [Room](#AAAAAAAAAC) placing process.

### Public Member Functions

[Scope](#AAAAAAAADL) ()

*Contructor creates empty* [*Room*](#AAAAAAAAAC) *lists for Circulation, Occupation, Service, and Tenant.*

[Room](#AAAAAAAAAC) [FindByDesignArea](#AAAAAAAADM) (double area, bool unplaced=true)

*Finds the first Occupant* [*Room*](#AAAAAAAAAC) *with the DesignArea value closest to the supplied area. C*

[Room](#AAAAAAAAAC) [FindByDesignXY](#AAAAAAAADN) (double designLength, double designWidth, bool unplaced=true)

*Finds the first Occupant* [*Room*](#AAAAAAAAAC) *with the designed x and y dimensions closest to the supplied values.*

[Room](#AAAAAAAAAC) [FindByTypeID](#AAAAAAAADO) (int typeID, bool unplaced=true)

*Finds the first unplaced* [*Room*](#AAAAAAAAAC) *with the specifed TypeID.*

### Properties

List< [Room](#AAAAAAAAAC) > [Circulation](#AAAAAAAADP) [get]

*List of Rooms designated as circulation.*

List< [Room](#AAAAAAAAAC) > [Occupant](#AAAAAAAADQ) [get]

*List of Rooms designated for occupation, rather than circulation.*

List< [Room](#AAAAAAAAAC) > [Service](#AAAAAAAADR) [get]

*List of Rooms designated for building services.*

List< [Room](#AAAAAAAAAC) > [Tenant](#AAAAAAAADS) [get]

*List of Rooms intended as a series of tenant space containers of other Rooms.*

List< Polygon > [AllocatedAsPolygons](#AAAAAAAADT) [get]

*List of allocated Circulation, Occupant, and Service* [*Room*](#AAAAAAAAAC) *Perimeters as Polygons.*

double [AreaDesignAvailable](#AAAAAAAADU) [get]

*Area available for horizontal circulation.*

double [AreaDesignCirculation](#AAAAAAAADV) [get]

*Intended aggregate area of all Occupant Rooms.*

double [AreaDesignOccupant](#AAAAAAAADW) [get]

*Intended aggregate area of all Occupant Rooms.*

double [AreaCirculation](#AAAAAAAADX) [get]

*Allocated aggregate area of all placed Circulation Rooms.*

double [AreaOccupant](#AAAAAAAADY) [get]

*Allocated aggregate area of all placed Occupant Rooms.*

double [AreaService](#AAAAAAAADZ) [get]

*Aggregate area of all Services Rooms.*

double [AreaTenant](#AAAAAAAAEA) [get]

*Aggregate area of all occupiable Tenant Rooms.*

List< Polygon > [CirculationAsPolygons](#AAAAAAAAEB) [get]

*List of all Circulation* [*Room*](#AAAAAAAAAC) *Perimeters as Polygons.*

List< Polygon > [OccupantAsPolygons](#AAAAAAAAEC) [get]

*List of all Occupant* [*Room*](#AAAAAAAAAC) *Perimeters as Polygons.*

List< Polygon > [ServiceAsPolygons](#AAAAAAAAED) [get]

*List of all Service* [*Room*](#AAAAAAAAAC) *Perimeters as Polygons.*

List< Polygon > [TenantAsPolygons](#AAAAAAAAEE) [get]

*List of all Tenant* [*Room*](#AAAAAAAAAC) *Perimeter Polygons.*

List< [Room](#AAAAAAAAAC) > [Placed](#AAAAAAAAEF) [get]

*List of all Rooms marked as Placed.*

bool [PlacedAll](#AAAAAAAAEG) [get]

*Returns whether all Occupant Rooms have been Placed.*

double [PlacedQuantity](#AAAAAAAAEH) [get]

*The quantity of placed Rooms.*

double [RatioCirculation](#AAAAAAAAEI) [get]

*Returns the ratio of the aggregate area of all Occupant Rooms against the Circulation area.*

double [RatioDesignCirculation](#AAAAAAAAEJ) [get]

*Returns the ratio of the aggregate area of all designed Occupant Rooms against the designed Circulation area.*

List< [Room](#AAAAAAAAAC) > [Unplaced](#AAAAAAAAEK) [get]

*Returns all Rooms with Placed = false.*

double [UnplacedQuantity](#AAAAAAAAEL) [get]

*The quantity of unplaced Rooms.*

### Detailed Description

Data structure recording space program characteristics and the status of a [Room](#AAAAAAAAAC) placing process.

### Constructor & Destructor Documentation

#### RoomKit.Scope.Scope ()

Contructor creates empty [Room](#AAAAAAAAAC) lists for Circulation, Occupation, Service, and Tenant.

##### Returns:

A new [Scope](#AAAAAAAAAF).

### Member Function Documentation

#### [Room](#AAAAAAAAAC) RoomKit.Scope.FindByDesignArea (double *area*, bool *unplaced* = true)

Finds the first Occupant [Room](#AAAAAAAAAC) with the DesignArea value closest to the supplied area. C

##### Parameters:

|  |  |
| --- | --- |
| *area* | Area to match from the list of all Occupant [Room](#AAAAAAAAAC) definitions. |

##### Returns:

A [Room](#AAAAAAAAAC).

#### [Room](#AAAAAAAAAC) RoomKit.Scope.FindByDesignXY (double *designLength*, double *designWidth*, bool *unplaced* = true)

Finds the first Occupant [Room](#AAAAAAAAAC) with the designed x and y dimensions closest to the supplied values.

##### Parameters:

|  |  |
| --- | --- |
| *designX* | The x-axis dimension to match. |
| *designY* | The y-axis dimension to match. |

##### Returns:

A [Room](#AAAAAAAAAC).

#### [Room](#AAAAAAAAAC) RoomKit.Scope.FindByTypeID (int *typeID*, bool *unplaced* = true)

Finds the first unplaced [Room](#AAAAAAAAAC) with the specifed TypeID.

##### Parameters:

|  |  |
| --- | --- |
| *typeID* | The integer ID of a [Room](#AAAAAAAAAC) type. |

##### Returns:

A [Room](#AAAAAAAAAC).

### Property Documentation

#### List<Polygon> RoomKit.Scope.AllocatedAsPolygons[get]

List of allocated Circulation, Occupant, and Service [Room](#AAAAAAAAAC) Perimeters as Polygons.

#### double RoomKit.Scope.AreaCirculation[get]

Allocated aggregate area of all placed Circulation Rooms.

#### double RoomKit.Scope.AreaDesignAvailable[get]

Area available for horizontal circulation.

#### double RoomKit.Scope.AreaDesignCirculation[get]

Intended aggregate area of all Occupant Rooms.

#### double RoomKit.Scope.AreaDesignOccupant[get]

Intended aggregate area of all Occupant Rooms.

#### double RoomKit.Scope.AreaOccupant[get]

Allocated aggregate area of all placed Occupant Rooms.

#### double RoomKit.Scope.AreaService[get]

Aggregate area of all Services Rooms.

#### double RoomKit.Scope.AreaTenant[get]

Aggregate area of all occupiable Tenant Rooms.

#### List<[Room](#AAAAAAAAAC)> RoomKit.Scope.Circulation[get]

List of Rooms designated as circulation.

#### List<Polygon> RoomKit.Scope.CirculationAsPolygons[get]

List of all Circulation [Room](#AAAAAAAAAC) Perimeters as Polygons.

##### Returns:

A list of Polygons.

#### List<[Room](#AAAAAAAAAC)> RoomKit.Scope.Occupant[get]

List of Rooms designated for occupation, rather than circulation.

#### List<Polygon> RoomKit.Scope.OccupantAsPolygons[get]

List of all Occupant [Room](#AAAAAAAAAC) Perimeters as Polygons.

##### Returns:

A list of Polygons.

#### List<[Room](#AAAAAAAAAC)> RoomKit.Scope.Placed[get]

List of all Rooms marked as Placed.

##### Returns:

A list of Rooms.

#### bool RoomKit.Scope.PlacedAll[get]

Returns whether all Occupant Rooms have been Placed.

##### Returns:

Returns true if each [Room](#AAAAAAAAAC) in Occupant has been marked with [Room.Placed](#AAAAAAAABU) = true.

#### double RoomKit.Scope.PlacedQuantity[get]

The quantity of placed Rooms.

#### double RoomKit.Scope.RatioCirculation[get]

Returns the ratio of the aggregate area of all Occupant Rooms against the Circulation area.

##### Returns:

A list of Rooms.

#### double RoomKit.Scope.RatioDesignCirculation[get]

Returns the ratio of the aggregate area of all designed Occupant Rooms against the designed Circulation area.

##### Returns:

A list of Rooms.

#### List<[Room](#AAAAAAAAAC)> RoomKit.Scope.Service[get]

List of Rooms designated for building services.

#### List<Polygon> RoomKit.Scope.ServiceAsPolygons[get]

List of all Service [Room](#AAAAAAAAAC) Perimeters as Polygons.

##### Returns:

A list of Polygons.

#### List<[Room](#AAAAAAAAAC)> RoomKit.Scope.Tenant[get]

List of Rooms intended as a series of tenant space containers of other Rooms.

#### List<Polygon> RoomKit.Scope.TenantAsPolygons[get]

List of all Tenant [Room](#AAAAAAAAAC) Perimeter Polygons.

##### Returns:

A list of Polygons.

#### List<[Room](#AAAAAAAAAC)> RoomKit.Scope.Unplaced[get]

Returns all Rooms with Placed = false.

##### Returns:

A list of Rooms.

#### double RoomKit.Scope.UnplacedQuantity[get]

The quantity of unplaced Rooms.

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

RoomKit/Scope.cs

## RoomKit.Story Class Reference

Creates and manages the geometry of a slab and Rooms representing corridors, occupied rooms, and services.

### Public Member Functions

[Story](#AAAAAAAAEM) ()

*Creates a* [*Story*](#AAAAAAAAAG) *at a 1.0 Height on the zero plane with new lists for Corridors, Rooms, and Services. Perimeter is set to null, Name is blank, and SlabThickness is s0.1.*

bool [AddCorridor](#AAAAAAAAEN) ([Room](#AAAAAAAAAC) room, bool fit=true)

*Adds a* [*Room*](#AAAAAAAAAC) *to the Corridors list.*

bool [AddExclusion](#AAAAAAAAEO) ([Room](#AAAAAAAAAC) room, bool fit=true)

*Adds a* [*Room*](#AAAAAAAAAC) *to the Exclusions list.*

bool [AddRoom](#AAAAAAAAEP) ([Room](#AAAAAAAAAC) room, bool fit=true)

*Adds a* [*Room*](#AAAAAAAAAC) *to the Rooms list.*

bool [AddService](#AAAAAAAAEQ) ([Room](#AAAAAAAAAC) room, bool fit=true)

*Adds a* [*Room*](#AAAAAAAAAC) *to the Services list.*

void [MoveFromTo](#AAAAAAAAER) (Vector3 from, Vector3 to)

*Moves all Rooms in the* [*Story*](#AAAAAAAAAG) *and the* [*Story*](#AAAAAAAAAG) *Envelope along a 3D vector calculated between the supplied Vector3 points.*

bool [RoomsByDivision](#AAAAAAAAES) (int xRooms=1, int yRooms=1, double height=3.0, double setback=0.0, string name="", Color color=null, bool fit=true)

*Creates Rooms by orthogonally dividing the interior of the* [*Story*](#AAAAAAAAAG) *perimeter by a quantity of x-axis and y-axis intervals. Adds the new Rooms to the Rooms list. New Rooms conform to Corridor and Service perimeters.*

void [Rotate](#AAAAAAAAET) (Vector3 pivot, double angle)

*Rotates the* [*Story*](#AAAAAAAAAG) *Perimeter and Rooms in the horizontal plane around the supplied pivot point.*

### Properties

double [Area](#AAAAAAAAEU) [get]

*Area of the perimeter.*

double [AreaAvailable](#AAAAAAAAEV) [get]

*Unallocated area within the* [*Story*](#AAAAAAAAAG)*.*

double [AreaPlaced](#AAAAAAAAEW) [get]

*Area allocated to Corridors, Rooms, and Services.*

Color **Color** [get, set]

List< [Room](#AAAAAAAAAC) > [Corridors](#AAAAAAAAEY) [get]

*List of Rooms designated as cooridors.*

List< Polygon > [CorridorsAsPolygons](#AAAAAAAAEZ) [get]

*Polygons representing Corridors. Rooms Perimeters in the* [*Story*](#AAAAAAAAAG) *conform to Corridor Perimeters.*

List< Space > [CorridorsAsSpaces](#AAAAAAAAFA) [get]

*List of Spaces created from* [*Room*](#AAAAAAAAAC) *characteristics within the Corridors list.*

Color [CorridorsColor](#AAAAAAAAFB) [set]

*Sets the Corridors color.*

double **Elevation** [get, set]

[Room](#AAAAAAAAAC) [Envelope](#AAAAAAAAFD) [get]

[*Room*](#AAAAAAAAAC) *representing the* [*Story*](#AAAAAAAAAG) *envelope.*

Polygon [EnvelopeAsPolygon](#AAAAAAAAFE) [get]

*Polygon representation of the* [*Story*](#AAAAAAAAAG) *Perimeter.*

Space [EnvelopeAsSpace](#AAAAAAAAFF) [get]

*Space created from* [*Story*](#AAAAAAAAAG) *characteristics.*

List< [Room](#AAAAAAAAAC) > [Exclusions](#AAAAAAAAFG) [get]

*Rooms representing areas that must not be intersected, but which will not be available as Spaces. All other* [*Room*](#AAAAAAAAAC) *Perimeters in the* [*Story*](#AAAAAAAAAG) *conform to Exclusion* [*Room*](#AAAAAAAAAC) *Perimeters.*

List< Polygon > [ExclusionsAsPolygons](#AAAAAAAAFH) [get]

*Polygons representing areas that must not be intersected. All other* [*Room*](#AAAAAAAAAC) *Perimeters in the* [*Story*](#AAAAAAAAAG) *conform to Exclusion* [*Room*](#AAAAAAAAAC) *Perimeters.*

double **Height** [get, set]

double [HeightInteriors](#AAAAAAAAFJ) [set]

*Sets the height of all Corridors, Rooms, and Services.*

IList< Polygon > [InteriorsAsPolygons](#AAAAAAAAFK) [get]

*Returns all Corridors, Exclusions, Rooms, and Services as Polygons.*

IList< Space > [InteriorsAsSpaces](#AAAAAAAAFL) [get]

*Returns all Corridors, Rooms, and Services as Spaces.*

string [Name](#AAAAAAAAFM) [get, set]

*Arbitrary string identifier.*

Polygon **Perimeter** [get, set]

List< [Room](#AAAAAAAAAC) > [Rooms](#AAAAAAAAFO) [get]

*List of Rooms designated as occupiable rooms.*

List< Polygon > [RoomsAsPolygons](#AAAAAAAAFP) [get]

*Polygons representing Services. Corridors and Rooms Perimeters in the* [*Story*](#AAAAAAAAAG) *conform to Service* [*Room*](#AAAAAAAAAC) *Perimeters.*

List< Space > [RoomsAsSpaces](#AAAAAAAAFQ) [get]

*List of Spaces created from* [*Room*](#AAAAAAAAAC) *characteristics within the Rooms list.*

Color [RoomsColor](#AAAAAAAAFR) [set]

*Sets the Rooms rendering color.*

List< [Room](#AAAAAAAAAC) > [Services](#AAAAAAAAFS) [get]

*A list of Rooms designated as building services.*

List< Polygon > [ServicesAsPolygons](#AAAAAAAAFT) [get]

*Polygons representing Services. Corridors and Rooms Perimeters in the* [*Story*](#AAAAAAAAAG) *conform to Service* [*Room*](#AAAAAAAAAC) *Perimeters.*

List< Space > [ServicesAsSpaces](#AAAAAAAAFU) [get]

*List of Spaces created from* [*Room*](#AAAAAAAAAC) *characteristics within the Services list.*

Color [ServicesColor](#AAAAAAAAFV) [set]

*Sets the Services Space rendering color.*

Floor [Slab](#AAAAAAAAFW) [get]

*Concrete Floor created from* [*Story*](#AAAAAAAAAG) *and Slab characteristics.*

double **SlabThickness** [get, set]

### Detailed Description

Creates and manages the geometry of a slab and Rooms representing corridors, occupied rooms, and services.

### Constructor & Destructor Documentation

#### RoomKit.Story.Story ()

Creates a [Story](#AAAAAAAAAG) at a 1.0 Height on the zero plane with new lists for Corridors, Rooms, and Services. Perimeter is set to null, Name is blank, and SlabThickness is s0.1.

##### Returns:

A new [Story](#AAAAAAAAAG).

### Member Function Documentation

#### bool RoomKit.Story.AddCorridor ([Room](#AAAAAAAAAC) *room*, bool *fit* = true)

Adds a [Room](#AAAAAAAAAC) to the Corridors list.

##### Parameters:

|  |  |
| --- | --- |
| *room* | [Room](#AAAAAAAAAC) to add. |
| *fit* | Indicates whether the new room should mutually fit to other [Story](#AAAAAAAAAG) features. Default is true. |

##### Returns:

True if one or more rooms were added to the [Story](#AAAAAAAAAG).

#### bool RoomKit.Story.AddExclusion ([Room](#AAAAAAAAAC) *room*, bool *fit* = true)

Adds a [Room](#AAAAAAAAAC) to the Exclusions list.

##### Parameters:

|  |  |
| --- | --- |
| *room* | [Room](#AAAAAAAAAC) to add. |
| *fit* | Indicates whether the new room should mutually fit to other [Story](#AAAAAAAAAG) features. Default is true. |

##### Returns:

True if one or more rooms were added to the [Story](#AAAAAAAAAG).

#### bool RoomKit.Story.AddRoom ([Room](#AAAAAAAAAC) *room*, bool *fit* = true)

Adds a [Room](#AAAAAAAAAC) to the Rooms list.

##### Parameters:

|  |  |
| --- | --- |
| *room* | [Room](#AAAAAAAAAC) to add. |
| *fit* | Indicates whether the new room should mutually fit to other [Story](#AAAAAAAAAG) features. Default is true. |

##### Returns:

True if one or more rooms were added to the [Story](#AAAAAAAAAG).

#### bool RoomKit.Story.AddService ([Room](#AAAAAAAAAC) *room*, bool *fit* = true)

Adds a [Room](#AAAAAAAAAC) to the Services list.

##### Parameters:

|  |  |
| --- | --- |
| *room* | [Room](#AAAAAAAAAC) to add. |
| *fit* | Indicates whether the new room should mutually fit to other [Story](#AAAAAAAAAG) features. Default is true. |

##### Returns:

True if one or more rooms were added to the [Story](#AAAAAAAAAG).

#### void RoomKit.Story.MoveFromTo (Vector3 *from*, Vector3 *to*)

Moves all Rooms in the [Story](#AAAAAAAAAG) and the [Story](#AAAAAAAAAG) Envelope along a 3D vector calculated between the supplied Vector3 points.

##### Parameters:

|  |  |
| --- | --- |
| *from* | Vector3 base point of the move. |
| *to* | Vector3 target point of the move. |

##### Returns:

None.

#### bool RoomKit.Story.RoomsByDivision (int *xRooms* = 1, int *yRooms* = 1, double *height* = 3.0, double *setback* = 0.0, string *name* = "", Color *color* = null, bool *fit* = true)

Creates Rooms by orthogonally dividing the interior of the [Story](#AAAAAAAAAG) perimeter by a quantity of x-axis and y-axis intervals. Adds the new Rooms to the Rooms list. New Rooms conform to Corridor and Service perimeters.

##### Parameters:

|  |  |
| --- | --- |
| *xRooms* | Quantity Rooms along the orthogonal x-axis. |
| *yRooms* | Quantity Rooms along the orthogonal y-axis. |
| *height* | Height of the new Rooms. |
| *setback* | Offset from the [Story](#AAAAAAAAAG) perimeter. |
| *name* | String identifier applied to every new [Room](#AAAAAAAAAC). |
| *color* | Rendering color of the [Room](#AAAAAAAAAC) as a Space. |

##### Returns:

None.

#### void RoomKit.Story.Rotate (Vector3 *pivot*, double *angle*)

Rotates the [Story](#AAAAAAAAAG) Perimeter and Rooms in the horizontal plane around the supplied pivot point.

##### Parameters:

|  |  |
| --- | --- |
| *pivot* | Vector3 point around which the [Room](#AAAAAAAAAC) Perimeter will be rotated. |
| *angle* | Angle in degrees to rotate the Perimeter. |

##### Returns:

None.

### Property Documentation

#### double RoomKit.Story.Area[get]

Area of the perimeter.

#### double RoomKit.Story.AreaAvailable[get]

Unallocated area within the [Story](#AAAAAAAAAG).

#### double RoomKit.Story.AreaPlaced[get]

Area allocated to Corridors, Rooms, and Services.

#### List<[Room](#AAAAAAAAAC)> RoomKit.Story.Corridors[get]

List of Rooms designated as cooridors.

#### List<Polygon> RoomKit.Story.CorridorsAsPolygons[get]

Polygons representing Corridors. Rooms Perimeters in the [Story](#AAAAAAAAAG) conform to Corridor Perimeters.

#### List<Space> RoomKit.Story.CorridorsAsSpaces[get]

List of Spaces created from [Room](#AAAAAAAAAC) characteristics within the Corridors list.

#### Color RoomKit.Story.CorridorsColor[set]

Sets the Corridors color.

#### [Room](#AAAAAAAAAC) RoomKit.Story.Envelope[get]

[Room](#AAAAAAAAAC) representing the [Story](#AAAAAAAAAG) envelope.

#### Polygon RoomKit.Story.EnvelopeAsPolygon[get]

Polygon representation of the [Story](#AAAAAAAAAG) Perimeter.

#### Space RoomKit.Story.EnvelopeAsSpace[get]

Space created from [Story](#AAAAAAAAAG) characteristics.

#### List<[Room](#AAAAAAAAAC)> RoomKit.Story.Exclusions[get]

Rooms representing areas that must not be intersected, but which will not be available as Spaces. All other [Room](#AAAAAAAAAC) Perimeters in the [Story](#AAAAAAAAAG) conform to Exclusion [Room](#AAAAAAAAAC) Perimeters.

#### List<Polygon> RoomKit.Story.ExclusionsAsPolygons[get]

Polygons representing areas that must not be intersected. All other [Room](#AAAAAAAAAC) Perimeters in the [Story](#AAAAAAAAAG) conform to Exclusion [Room](#AAAAAAAAAC) Perimeters.

#### double RoomKit.Story.HeightInteriors[set]

Sets the height of all Corridors, Rooms, and Services.

#### IList<Polygon> RoomKit.Story.InteriorsAsPolygons[get]

Returns all Corridors, Exclusions, Rooms, and Services as Polygons.

#### IList<Space> RoomKit.Story.InteriorsAsSpaces[get]

Returns all Corridors, Rooms, and Services as Spaces.

#### string RoomKit.Story.Name[get], [set]

Arbitrary string identifier.

#### List<[Room](#AAAAAAAAAC)> RoomKit.Story.Rooms[get]

List of Rooms designated as occupiable rooms.

#### List<Polygon> RoomKit.Story.RoomsAsPolygons[get]

Polygons representing Services. Corridors and Rooms Perimeters in the [Story](#AAAAAAAAAG) conform to Service [Room](#AAAAAAAAAC) Perimeters.

#### List<Space> RoomKit.Story.RoomsAsSpaces[get]

List of Spaces created from [Room](#AAAAAAAAAC) characteristics within the Rooms list.

#### Color RoomKit.Story.RoomsColor[set]

Sets the Rooms rendering color.

#### List<[Room](#AAAAAAAAAC)> RoomKit.Story.Services[get]

A list of Rooms designated as building services.

#### List<Polygon> RoomKit.Story.ServicesAsPolygons[get]

Polygons representing Services. Corridors and Rooms Perimeters in the [Story](#AAAAAAAAAG) conform to Service [Room](#AAAAAAAAAC) Perimeters.

#### List<Space> RoomKit.Story.ServicesAsSpaces[get]

List of Spaces created from [Room](#AAAAAAAAAC) characteristics within the Services list.

#### Color RoomKit.Story.ServicesColor[set]

Sets the Services Space rendering color.

#### Floor RoomKit.Story.Slab[get]

Concrete Floor created from [Story](#AAAAAAAAAG) and Slab characteristics.

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

RoomKit/Story.cs

## RoomKit.TopoBox Class Reference

Maintains a set of points on the orthogonal bounding box of a supplied Polygon corresponding to four divisions of each side.

### Public Member Functions

[TopoBox](#AAAAAAAAFY) (Polygon polygon)

*Constructor creates a new mathematical bounding box from the supplied Polygon and populates all orientation points.*

Vector3 [PointBy](#AAAAAAAAFZ) ([Orient](#AAAAAAAAAK) orient)

*Returns the requested bounding box location by orientation.*

Vector3 [PointOpposite](#AAAAAAAAGA) ([Orient](#AAAAAAAAAK) orient)

*Returns the reciprocal bounding box location by orientation.*

### Properties

Vector3 [C](#AAAAAAAAGB) [get]

*Vector3 location identifier corresponding to the center of the box perimeter.*

Vector3 [N](#AAAAAAAAGC) [get]

*Vector3 location identifier corresponding to the midpoint of the maximum Y side of the box perimeter.*

Vector3 [NNW](#AAAAAAAAGD) [get]

*Vector3 location identifier corresponding to the midpoint between the NW and N points of the box perimeter.*

Vector3 [NW](#AAAAAAAAGE) [get]

*Vector3 location identifier corresponding to the mimimum X and maximum Y corner of the box perimeter.*

Vector3 [WNW](#AAAAAAAAGF) [get]

*Vector3 location identifier corresponding to the midpoint between the NW and W points of the box perimeter.*

Vector3 [W](#AAAAAAAAGG) [get]

*Vector3 location identifier corresponding to the midpoint of the minimum X side of the box perimeter.*

Vector3 [WSW](#AAAAAAAAGH) [get]

*Vector3 location identifier corresponding to the midpoint between the SW and W points of the box perimeter.*

Vector3 [SW](#AAAAAAAAGI) [get]

*Vector3 location identifier corresponding to the mimimum X and Y corner of the box perimeter.*

Vector3 [SSW](#AAAAAAAAGJ) [get]

*Vector3 location identifier corresponding to the midpoint between the SW and S points of the box perimeter.*

Vector3 [S](#AAAAAAAAGK) [get]

*Vector3 location identifier corresponding to the midpoint of the minimum Y side of the box perimeter.*

Vector3 [SSE](#AAAAAAAAGL) [get]

*Vector3 location identifier corresponding to the midpoint between the SE and S points of the box perimeter.*

Vector3 [SE](#AAAAAAAAGM) [get]

*Vector3 location identifier corresponding to the maximum X and minimum Y corner of the box perimeter.*

Vector3 [ESE](#AAAAAAAAGN) [get]

*Vector3 location identifier corresponding to the midpoint between the SE and E points of the box perimeter.*

Vector3 [E](#AAAAAAAAGO) [get]

*Vector3 location identifier corresponding to the midpoint of the maximum X side of the box perimeter.*

Vector3 [ENE](#AAAAAAAAGP) [get]

*Vector3 location identifier corresponding to the midpoint between the NE and E points of the box perimeter.*

Vector3 [NE](#AAAAAAAAGQ) [get]

*Vector3 location identifier corresponding to the maximum X and Y corner of the box perimeter.*

Vector3 [NNE](#AAAAAAAAGR) [get]

*Vector3 location identifier corresponding to the midpoint between the NE and N points of the box perimeter.*

double [SizeX](#AAAAAAAAGS) [get]

*X and Y dimensions of the* [*TopoBox*](#AAAAAAAAAH) *perimeter.*

double **SizeY** [get]

### Detailed Description

Maintains a set of points on the orthogonal bounding box of a supplied Polygon corresponding to four divisions of each side.

### Constructor & Destructor Documentation

#### RoomKit.TopoBox.TopoBox (Polygon *polygon*)

Constructor creates a new mathematical bounding box from the supplied Polygon and populates all orientation points.

##### Returns:

A new [TopoBox](#AAAAAAAAAH).

### Member Function Documentation

#### Vector3 RoomKit.TopoBox.PointBy ([Orient](#AAAAAAAAAK) *orient*)

Returns the requested bounding box location by orientation.

##### Parameters:

|  |  |
| --- | --- |
| *orient* | The Orient value to index point. |

##### Returns:

A Vector3 point.

#### Vector3 RoomKit.TopoBox.PointOpposite ([Orient](#AAAAAAAAAK) *orient*)

Returns the reciprocal bounding box location by orientation.

##### Parameters:

|  |  |
| --- | --- |
| *orient* | The Orient value to find the reciprocal point. |

##### Returns:

A Vector3 point.

### Property Documentation

#### Vector3 RoomKit.TopoBox.C[get]

Vector3 location identifier corresponding to the center of the box perimeter.

#### Vector3 RoomKit.TopoBox.E[get]

Vector3 location identifier corresponding to the midpoint of the maximum X side of the box perimeter.

#### Vector3 RoomKit.TopoBox.ENE[get]

Vector3 location identifier corresponding to the midpoint between the NE and E points of the box perimeter.

#### Vector3 RoomKit.TopoBox.ESE[get]

Vector3 location identifier corresponding to the midpoint between the SE and E points of the box perimeter.

#### Vector3 RoomKit.TopoBox.N[get]

Vector3 location identifier corresponding to the midpoint of the maximum Y side of the box perimeter.

#### Vector3 RoomKit.TopoBox.NE[get]

Vector3 location identifier corresponding to the maximum X and Y corner of the box perimeter.

#### Vector3 RoomKit.TopoBox.NNE[get]

Vector3 location identifier corresponding to the midpoint between the NE and N points of the box perimeter.

#### Vector3 RoomKit.TopoBox.NNW[get]

Vector3 location identifier corresponding to the midpoint between the NW and N points of the box perimeter.

#### Vector3 RoomKit.TopoBox.NW[get]

Vector3 location identifier corresponding to the mimimum X and maximum Y corner of the box perimeter.

#### Vector3 RoomKit.TopoBox.S[get]

Vector3 location identifier corresponding to the midpoint of the minimum Y side of the box perimeter.

#### Vector3 RoomKit.TopoBox.SE[get]

Vector3 location identifier corresponding to the maximum X and minimum Y corner of the box perimeter.

#### double RoomKit.TopoBox.SizeX[get]

X and Y dimensions of the [TopoBox](#AAAAAAAAAH) perimeter.

#### Vector3 RoomKit.TopoBox.SSE[get]

Vector3 location identifier corresponding to the midpoint between the SE and S points of the box perimeter.

#### Vector3 RoomKit.TopoBox.SSW[get]

Vector3 location identifier corresponding to the midpoint between the SW and S points of the box perimeter.

#### Vector3 RoomKit.TopoBox.SW[get]

Vector3 location identifier corresponding to the mimimum X and Y corner of the box perimeter.

#### Vector3 RoomKit.TopoBox.W[get]

Vector3 location identifier corresponding to the midpoint of the minimum X side of the box perimeter.

#### Vector3 RoomKit.TopoBox.WNW[get]

Vector3 location identifier corresponding to the midpoint between the NW and W points of the box perimeter.

#### Vector3 RoomKit.TopoBox.WSW[get]

Vector3 location identifier corresponding to the midpoint between the SW and W points of the box perimeter.

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

RoomKit/TopoBox.cs

## RoomKit.Tower Class Reference

### Public Member Functions

bool [AddServiceCore](#AAAAAAAAGU) (Polygon perimeter, int baseStory=0, double addHeight=0.0, Color color=null)

*Adds a new service Core to the* [*Tower*](#AAAAAAAAAI)*.*

void [MoveFromTo](#AAAAAAAAGV) (Vector3 from, Vector3 to)

*Moves all Cores and Stories in the* [*Tower*](#AAAAAAAAAI) *along a 3D vector calculated between the supplied Vector3 points.*

void [Rotate](#AAAAAAAAGW) (Vector3 pivot, double angle)

*Rotates the* [*Tower*](#AAAAAAAAAI) *Perimeter and Stories in the horizontal plane around the supplied pivot point.*

bool [Stack](#AAAAAAAAGX) (int floors=0, double storyHeight=0.0)

*Creates the* [*Tower*](#AAAAAAAAAI) *by stacking a series of* [*Story*](#AAAAAAAAAG) *instances from the* [*Tower*](#AAAAAAAAAI) *Elevation.*

bool [SetStoryHeight](#AAAAAAAAGY) (int story, double height, bool interiors=true)

*Sets the height of an index-specified* [*Story*](#AAAAAAAAAG) *and relocates Stories above to accommodate the* [*Story*](#AAAAAAAAAG)*'s new height.*

### Public Attributes

List< [Story](#AAAAAAAAAG) > [Stories](#AAAAAAAAGZ) = null

*List of all Stories in the* [*Tower*](#AAAAAAAAAI)*.*

### Properties

Color **Color** [get, set]

List< [Room](#AAAAAAAAAC) > [Cores](#AAAAAAAAHB) [get]

*List of all service Cores in the* [*Tower*](#AAAAAAAAAI)*.*

double **Elevation** [get, set]

int **Floors** [get, set]

double [Height](#AAAAAAAAHE) [get]

*Total height of all Stories in the* [*Tower*](#AAAAAAAAAI)*.*

Polygon **Perimeter** [get, set]

List< Floor > [Slabs](#AAAAAAAAHG) [get]

*List of all Slabs from every* [*Story*](#AAAAAAAAAG) *in the* [*Tower*](#AAAAAAAAAI)*.*

List< Space > [Spaces](#AAAAAAAAHH) [get]

*List of all Spaces from every* [*Story*](#AAAAAAAAAG) *in the* [*Tower*](#AAAAAAAAAI)*.*

double **StoryHeight** [get, set]

### Member Function Documentation

#### bool RoomKit.Tower.AddServiceCore (Polygon *perimeter*, int *baseStory* = 0, double *addHeight* = 0.0, Color *color* = null)

Adds a new service Core to the [Tower](#AAAAAAAAAI).

##### Parameters:

|  |  |
| --- | --- |
| *perimeter* | Polygon perimeter defining the footprint of the service Core. |
| *baseStory* | Index of the lowest [Story](#AAAAAAAAAG) whose elevation will serve as the lowest level of the Core. |
| *addHeight* | Additional height of the Core above the highest [Story](#AAAAAAAAAG). |
| *color* | Color of the Core when it it is accessed as a Space. |

##### Returns:

True if the Core is successfully added.

#### void RoomKit.Tower.MoveFromTo (Vector3 *from*, Vector3 *to*)

Moves all Cores and Stories in the [Tower](#AAAAAAAAAI) along a 3D vector calculated between the supplied Vector3 points.

##### Parameters:

|  |  |
| --- | --- |
| *from* | Vector3 base point of the move. |
| *to* | Vector3 target point of the move. |

##### Returns:

None.

#### void RoomKit.Tower.Rotate (Vector3 *pivot*, double *angle*)

Rotates the [Tower](#AAAAAAAAAI) Perimeter and Stories in the horizontal plane around the supplied pivot point.

##### Parameters:

|  |  |
| --- | --- |
| *pivot* | Vector3 point around which the [Room](#AAAAAAAAAC) Perimeter will be rotated. |
| *angle* | Angle in degrees to rotate the Perimeter. |

##### Returns:

None.

#### bool RoomKit.Tower.SetStoryHeight (int *story*, double *height*, bool *interiors* = true)

Sets the height of an index-specified [Story](#AAAAAAAAAG) and relocates Stories above to accommodate the [Story](#AAAAAAAAAG)'s new height.

##### Parameters:

|  |  |
| --- | --- |
| *story* | Index of the [Story](#AAAAAAAAAG) to affect. |
| *height* | Desired new height of the specified [Story](#AAAAAAAAAG). |
| *interiors* | If true also sets any Corridors and Rooms in the [Story](#AAAAAAAAAG) to the new Height. |

##### Returns:

True if the [Tower](#AAAAAAAAAI) is successfully stacked.

#### bool RoomKit.Tower.Stack (int *floors* = 0, double *storyHeight* = 0.0)

Creates the [Tower](#AAAAAAAAAI) by stacking a series of [Story](#AAAAAAAAAG) instances from the [Tower](#AAAAAAAAAI) Elevation.

##### Parameters:

|  |  |
| --- | --- |
| *floors* | Desired quantity of stacked Stories to form the [Tower](#AAAAAAAAAI). If greater than zero, overrides and resets the current Floors property. |
| *storyHeight* | Desired typical [Story](#AAAAAAAAAG) height of the [Tower](#AAAAAAAAAI). If greater than zero, overrides and resets the current StoryHeight property. |

##### Returns:

True if the [Tower](#AAAAAAAAAI) is successfully stacked.

### Member Data Documentation

#### List<[Story](#AAAAAAAAAG)> RoomKit.Tower.Stories = null

List of all Stories in the [Tower](#AAAAAAAAAI).

### Property Documentation

#### List<[Room](#AAAAAAAAAC)> RoomKit.Tower.Cores[get]

List of all service Cores in the [Tower](#AAAAAAAAAI).

#### double RoomKit.Tower.Height[get]

Total height of all Stories in the [Tower](#AAAAAAAAAI).

#### List<Floor> RoomKit.Tower.Slabs[get]

List of all Slabs from every [Story](#AAAAAAAAAG) in the [Tower](#AAAAAAAAAI).

#### List<Space> RoomKit.Tower.Spaces[get]

List of all Spaces from every [Story](#AAAAAAAAAG) in the [Tower](#AAAAAAAAAI).

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

RoomKit/Tower.cs

# Index

INDEX