minecraftstuff Documentation

Release 1.0.1

Martin O’Hanlon

Jul 12, 2017

Contents

An extension library of useful 'stuff' (aka classes) I have created for Minecraft: Pi Edition's API.

It provides functions for drawing lines, creating, moving and rotating shapes and a pretty cool turtle. Check out the documentation and examples.

Created by Martin O'Hanlon, @martinohanlon, stuffaboutco.de.

CHAPTER 1

Install

Open a terminal and run the following commands:

```
sudo pip install minecraftstuff
sudo pip3 install minecraftstuff
```

CHAPTER 2

Code

See the documentation for more information, but below is a summary.

MinecraftDrawing

```python
from minecraftstuff import MinecraftDrawing
from mcpi.minecraft import Minecraft
from mcpi import block

#Connect to minecraft
mc = Minecraft.create()
# get the players position
pos = mc.player.getTilePos()

#Using the Minecraft Drawing API
mcdrawing = MinecraftDrawing(mc)

# draw a circle with a radius of 10 blocks
mcdrawing.drawCircle(pos.x, pos.y + 15, pos.z, 10, block.WOOD.id)
```

MinecraftShape

```python
from minecraftstuff import MinecraftShape
from mcpi.minecraft import Minecraft
from mcpi import block
from time import sleep

#Connect to minecraft
mc = Minecraft.create()
# get the players position
pos = mc.player.getTilePos()
```

```
#Using the Minecraft Shape API
mcshape = MinecraftShape(mc, pos)

# create a stone cube
mcshape.setBlocks(-5, -5, -5, 5, 5, 5, block.STONE.id)

# move it around
for i in range(0,10):
    mcshape.moveBy(1,0,1)
    sleep(0.5)
```

MinecraftTurtle

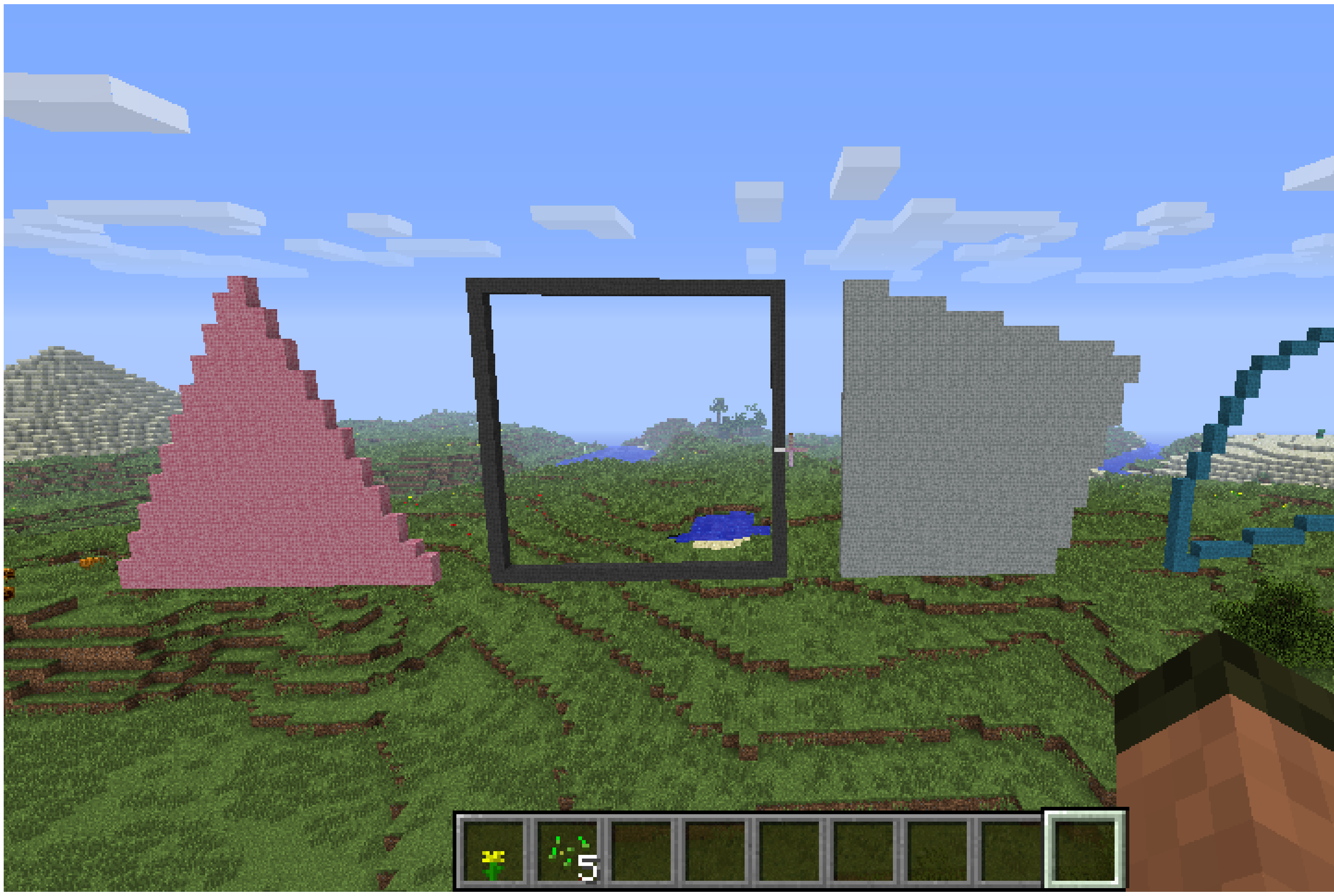
```
from minecraftstuff import MinecraftTurtle
from mcpi.minecraft import Minecraft

#Connect to minecraft
mc = Minecraft.create()
# get the players position
pos = mc.player.getTilePos()

#Using the Minecraft Turtle
steve = MinecraftTurtle(mc, pos)

# draw a square
steve.forward(5)
steve.right(90)
steve.forward(5)
steve.right(90)
steve.forward(5)
steve.right(90)
steve.forward(5)
```

CHAPTER 3

Table of Contents

MinecraftDrawing API

The MinecraftDrawing API can be used to create lines, circles, spheres, and faces.



Example:

```
from mcpi.minecraft import Minecraft
from mcpi import block
from minecraftstuff import MinecraftDrawing

mc = Minecraft.create()

mcdraw = MinecraftDrawing(mc)

# draw a diagonal line
mcdraw.drawLine(0,0,0,10,10,10,block.STONE.id)
```

MinecraftDrawing

class `minecraftstuff.`**`MinecraftDrawing`**(*mc*)

MinecraftDrawing - a class of useful drawing functions

Parameters **`mc`** (*`mcpi.minecraft.Minecraft`*) – A Minecraft object which is connected to a world.

`drawCircle`(*x0*, *y0*, *z*, *radius*, *blockType*, *blockData=0*)

draws a circle in the Y plane (i.e. vertically)

Parameters

- **x0** (*int*) – The x position of the centre of the circle.
- **y0** (*int*) – The y position of the centre of the circle.
- **z** (*int*) – The z position of the centre of the circle.
- **radius** (*int*) – The radius of the sphere.
- **blockType** (*int*) – The block id.
- **blockData** (*int*) – The block data value, defaults to `0`.

drawFace (*vertices*, *filled*, *blockType*, *blockData=0*)
draws a face, when passed a collection of vertices which make up a polyhedron

Parameters

- **vertices** (*list*) – The a list of points, passed as either a `minecraftstuff.Points` object or as a list of `mcpi.minecraft.Vec3` objects.
- **filled** (*boolean*) – If `True` fills the face with blocks.
- **blockType** (*int*) – The block id.
- **blockData** (*int*) – The block data value, defaults to `0`.

drawHollowSphere (*x1*, *y1*, *z1*, *radius*, *blockType*, *blockData=0*)
draws a hollow sphere around a point to a radius, sphere has to big enough to be hollow!

Parameters

- **x1** (*int*) – The x position of the centre of the sphere.
- **y1** (*int*) – The y position of the centre of the sphere.
- **z1** (*int*) – The z position of the centre of the sphere.
- **radius** (*int*) – The radius of the sphere.
- **blockType** (*int*) – The block id.
- **blockData** (*int*) – The block data value, defaults to `0`.

drawHorizontalCircle (*x0*, *y*, *z0*, *radius*, *blockType*, *blockData=0*)
draws a circle in the X plane (i.e. horizontally)

Parameters

- **x0** (*int*) – The x position of the centre of the circle.
- **y** (*int*) – The y position of the centre of the circle.
- **z0** (*int*) – The z position of the centre of the circle.
- **radius** (*int*) – The radius of the circle.
- **blockType** (*int*) – The block id.
- **blockData** (*int*) – The block data value, defaults to `0`.

drawLine (*x1*, *y1*, *z1*, *x2*, *y2*, *z2*, *blockType*, *blockData=0*)
draws a line between 2 points

Parameters

- **x1** (*int*) – The x position of the first point.
- **y1** (*int*) – The y position of the first point.

- **z1** (*int*) – The z position of the first point.
- **x2** (*int*) – The x position of the second point.
- **y2** (*int*) – The y position of the second point.
- **z2** (*int*) – The z position of the second point.
- **blockType** (*int*) – The block id.
- **blockData** (*int*) – The block data value, defaults to `0`.

`drawPoint3d`(*x*, *y*, *z*, *blockType*, *blockData=0*)
draws a single point in Minecraft, i.e. 1 block

Parameters

- **x** (*int*) – The x position.
- **y** (*int*) – The y position.
- **z** (*int*) – The z position.
- **blockType** (*int*) – The block id.
- **blockData** (*int*) – The block data value, defaults to `0`.

`drawSphere`(*x1*, *y1*, *z1*, *radius*, *blockType*, *blockData=0*)
draws a sphere around a point to a radius

Parameters

- **x1** (*int*) – The x position of the centre of the sphere.
- **y1** (*int*) – The y position of the centre of the sphere.
- **z1** (*int*) – The z position of the centre of the sphere.
- **radius** (*int*) – The radius of the sphere.
- **blockType** (*int*) – The block id.
- **blockData** (*int*) – The block data value, defaults to `0`.

`drawVertices`(*vertices*, *blockType*, *blockData=0*)
draws all the points in a collection of vertices with a block

Parameters

- **vertices** (*list*) – A list of `mcpi.minecraft.Vec3` objects.
- **blockType** (*int*) – The block id.
- **blockData** (*int*) – The block data value, defaults to `0`.

`getLine`(*x1*, *y1*, *z1*, *x2*, *y2*, *z2*)
Returns all the points which would make up a line between 2 points as a list

3d implementation of bresenham line algorithm

Parameters

- **x1** (*int*) – The x position of the first point.
- **y1** (*int*) – The y position of the first point.
- **z1** (*int*) – The z position of the first point.
- **x2** (*int*) – The x position of the second point.

- **y2** (*int*) – The y position of the second point.
- **z2** (*int*) – The z position of the second point.

Points

class `minecraftstuff.`**`Points`**

Points - a collection of minecraft positions or Vec3's. Used when drawing faces `MinecraftDrawing.drawFace()`.

`add`(*x*, *y*, *z*)

add a single position to the list of points.

Parameters

- **x** (*int*) – The x position.
- **y** (*int*) – The y position.
- **z** (*int*) – The z position.

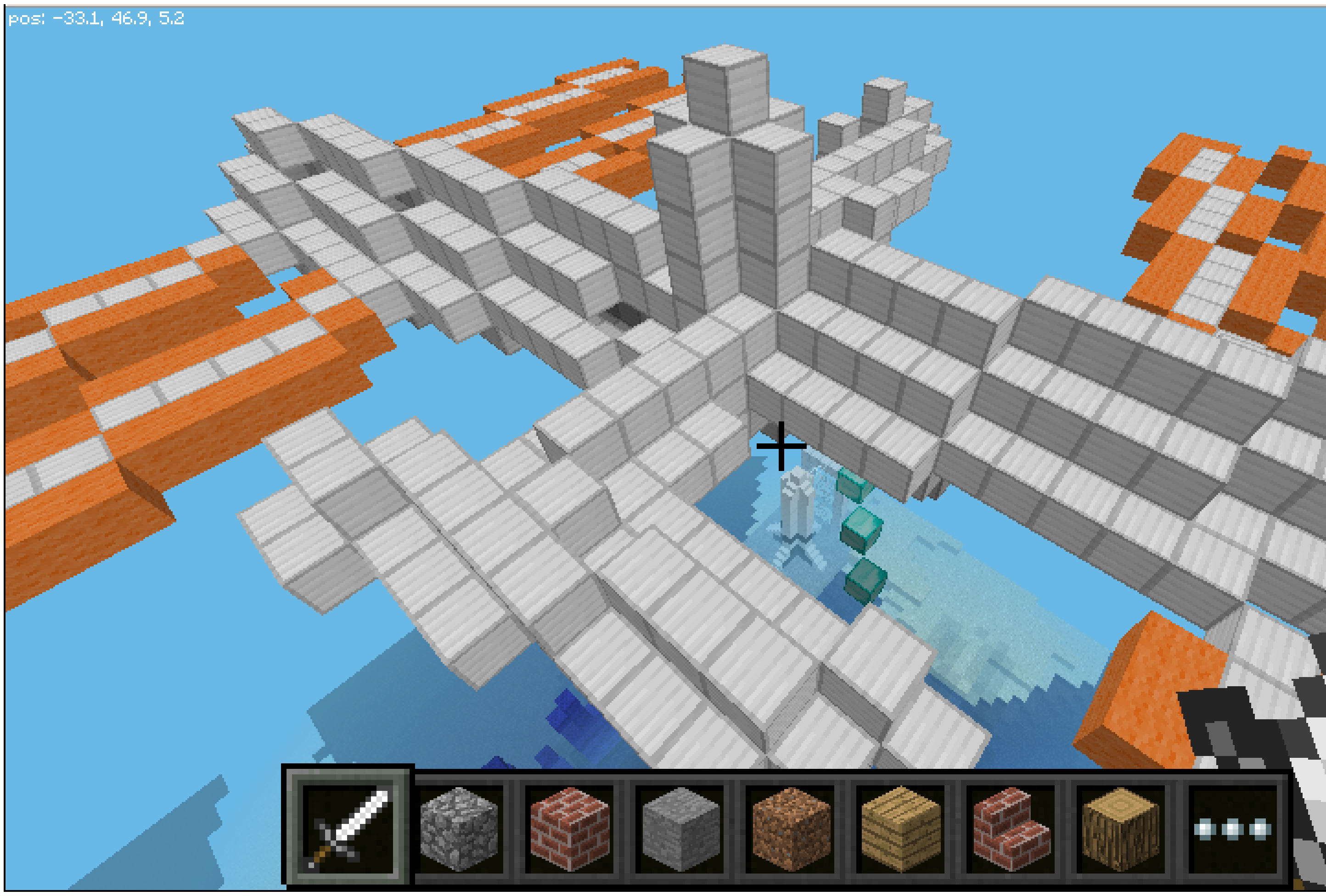
`getVec3s`()

returns a list of Vec3 positions

MinecraftShape API

The MinecraftShape API allows you create and manipulate shapes (collections of blocks) in Minecraft.

The shapes can be small or big and the API helps by efficiently moving them and only changing the blocks which have actually changed.



Example:

```
from mcpi.minecraft import Minecraft
from mcpi import block
from minecraftstuff import MinecraftShape

mc = minecraft.Minecraft.create()

pos = mc.player.getTilePos()

myShape = MinecraftShape(mc, pos)

# create a big cube
myShape.setBlocks(-5, -5, -5, 5, 5, 5, block.WOOL.id, 5)

#move it 10 blocks up
myshape.moveBy(0, 10, 0)

#rotate it 45 degrees
myshape.rotate(45, 0, 0)
```

MinecraftShape

class `minecraftstuff.`**`MinecraftShape`**(*mc*, *position*, *shapeBlocks=None*, *visible=True*)

MinecraftShape - the implementation of a 'shape' in Minecraft.

Each shape consists of one or many blocks with a position relative to each other.

Shapes can be transformed by movement and rotation.

When a shape is changed and redrawn in Minecraft only the blocks which have changed are updated.

Parameters

- **`mc`** (*`mcpi.minecraft.Minecraft`*) – A Minecraft object which is connected to a world.
- **`position`** (*`mcpi.minecraft.Vec3`*) – The position where the shape should be created
- **`shapeBlocks`** (*`list`*) – A list of ShapeBlocks which make up the shape. This defaults to `None`.
- **`visible`** (*`bool`*) – Where the shape should be visible. This defaults to `True`.

`clear`()

clears the shape in Minecraft

`draw`()

draws the shape in Minecraft, taking into account where it was last drawn, only updating the blocks which have changed

`getShapeBlock`(*x*, *y*, *z*)

returns the ShapeBlock for an 'actual position'

Parameters

- **`x`** (*`int`*) – The x position.
- **`y`** (*`int`*) – The y position.
- **`z`** (*`int`*) – The z position.

`move`(*x*, *y*, *z*)

moves the position of the shape to x,y,z

Parameters

- **`x`** (*`int`*) – The x position.
- **`y`** (*`int`*) – The y position.
- **`z`** (*`int`*) – The z position.

`moveBy`(*x*, *y*, *z*)

moves the position of the shape by x,y,z

Parameters

- **`x`** (*`int`*) – The number of blocks to move in x.
- **`y`** (*`int`*) – The number of blocks to move in y.
- **`z`** (*`int`*) – The number of blocks to move in z.

`redraw`()

redraws the shape in Minecraft, by clearing all the blocks and redrawing them

reset()
resets the shape back to its original position

rotate(*yaw*, *pitch*, *roll*)
sets the rotation of a shape by yaw, pitch and roll

Parameters

- **yaw** (*float*) – The yaw rotation in degrees.
- **pitch** (*float*) – The pitch rotation in degrees.
- **roll** (*float*) – The roll rotation in degrees.

rotateBy(*yaw*, *pitch*, *roll*)
increments the rotation of a shape by yaw, pitch and roll

Parameters

- **yaw** (*float*) – The yaw rotation in degrees.
- **pitch** (*float*) – The pitch rotation in degrees.
- **roll** (*float*) – The roll rotation in degrees.

setBlock(*x*, *y*, *z*, *blockType*, *blockData=0*, *tag=''*)
sets one block in the shape and redraws it

draws a single point in Minecraft, i.e. 1 block

Parameters

- **x** (*int*) – The x position.
- **y** (*int*) – The y position.
- **z** (*int*) – The z position.
- **blockType** (*int*) – The block id.
- **blockData** (*int*) – The block data value, defaults to `0`.
- **tag** (*string*) – A tag for the block, this is useful for grouping blocks together and keeping track of them as the position of blocks can change, defaults to `""`.

setBlocks(*x1*, *y1*, *z1*, *x2*, *y2*, *z2*, *blockType*, *blockData=0*, *tag=''*)
creates a cuboid of blocks in the shape and redraws it

Parameters

- **x1** (*int*) – The x position of the first point.
- **y1** (*int*) – The y position of the first point.
- **z1** (*int*) – The z position of the first point.
- **x2** (*int*) – The x position of the second point.
- **y2** (*int*) – The y position of the second point.
- **z2** (*int*) – The z position of the second point.
- **blockType** (*int*) – The block id.
- **blockData** (*int*) – The block data value, defaults to `0`.
- **tag** (*string*) – A tag for the block, this is useful for grouping blocks together and keeping track of them as the position of blocks can change, defaults to `""`.

ShapeBlock

class `minecraftstuff.`**`ShapeBlock`**(*x*, *y*, *z*, *blockType*, *blockData=0*, *tag=''*)

ShapeBlock - a class to hold one block within a shape

Parameters

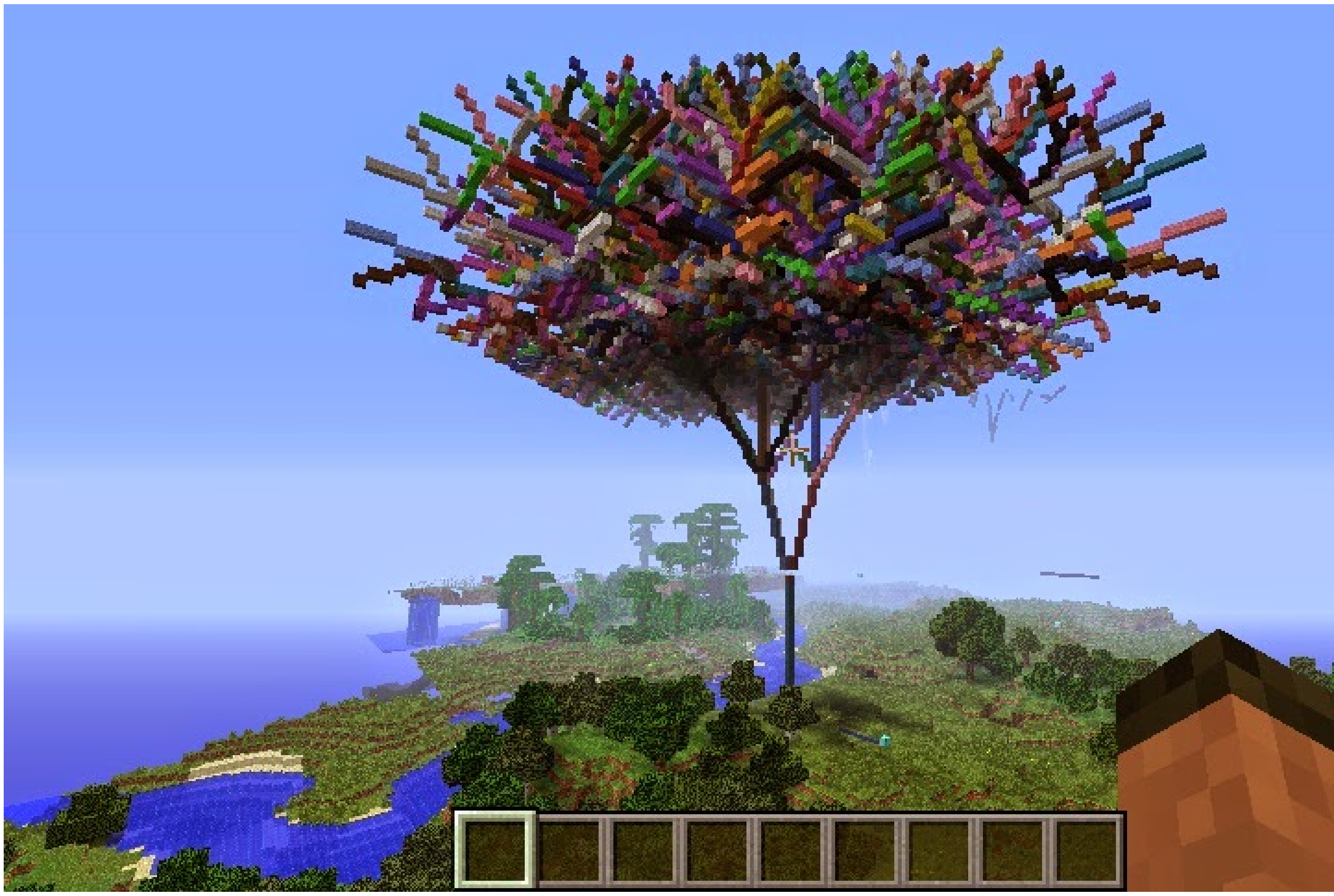
- **`x`** (*`int`*) – The x position.
- **`y`** (*`int`*) – The y position.
- **`z`** (*`int`*) – The z position.
- **`blockType`** (*`int`*) – The block id.
- **`blockData`** (*`int`*) – The block data value, defaults to `0`.
- **`tag`** (*`string`*) – A tag for the block, this is useful for grouping blocks together and keeping track of them as the position of blocks can change, defaults to `""`.

`resetRelativePos`()

resets the relative position of the block back to its original position

MinecraftTurtle

The Minecraft Turtle is a recreation of the classic graphics turtle for Minecraft. The key difference is that you can draw in 3 dimensions not just 2.



Example:

```
from mcpi.minecraft import Minecraft
from mcpi import block
from minecraftstuff import MinecraftTurtle

mc = minecraft.Minecraft.create()
pos = mc.player.getTilePos()

# create minecraft turtle
steve = MinecraftTurtle(mc, pos)

# draw a pentagon
steve.forward(5)
steve.right(72)
steve.forward(5)
steve.right(72)
steve.forward(5)
steve.right(72)
steve.forward(5)
steve.right(72)
steve.forward(5)
```

MinecraftTurtle

class `minecraftstuff.`**`MinecraftTurtle`**(*mc*, *position=<MagicMock name='mock()' id='140651776724944'>*)

MinecraftTurle - a graphics turtle, which can be used to create 'things' in Minecraft by controlling its position, angles and direction

Parameters

- **`mc`** (*`mcpi.minecraft.Minecraft`*) – A Minecraft object which is connected to a world.
- **`position`** (*`mcpi.minecraft.Vec3`*) – The position where the shape should be created, defaults to `0,0,0`.

`backward`(*distance*)
move the turtle backward

Parameters **`distance`** (*`int`*) – the number of blocks to move.

`down`(*angle*)
rotate the turtle down

Parameters **`angle`** (*`float`*) – the angle in degrees to rotate.

`fly`()
sets the turtle to 'fly', i.e. not have to move along the ground.

`forward`(*distance*)
move the turtle forward

Parameters **`distance`** (*`int`*) – the number of blocks to move.

`home`()
reset the turtle's position

`isdown`()
returns `True` if the pen is down

`left`(*angle*)
rotate the turtle left

Parameters **`angle`** (*`float`*) – the angle in degrees to rotate.

`penblock`(*blockId*, *blockData=0*)
set the block the turtle uses as its pen.

Parameters

- **`blockType`** (*`int`*) – The block id.
- **`blockData`** (*`int`*) – The block data value, defaults to `0`.

`pendown`()
put the turtles pen down, show it will draw

`penup`()
put the turtles pen up, show it wont draw

`right`(*angle*)
rotate the turtle right

Parameters **`angle`** (*`float`*) – the angle in degrees to rotate.

`setheading`(*angle*)
set the turtle's horizontal heading

Parameters **angle** (*float*) – the angle in degrees.

setposition (*x*, *y*, *z*)
set the turtle's position

Parameters

- **x** (*int*) – the x position.
- **y** (*int*) – the y position.
- **z** (*int*) – the z position.

setverticalheading (*angle*)
set the turtle's verticle heading

Parameters **angle** (*float*) – the angle in degrees.

setx (*x*)
set the turtle's x position

Parameters **x** (*int*) – the x position.

sety (*y*)
set the turtle's y position

Parameters **y** (*int*) – the y position.

setz (*z*)
set the turtle's z position

Parameters **z** (*int*) – the z position.

speed (*turtlespeed*)
set the turtle's speed.

Parameters **turtlespeed** (*int*) – `1 - 10`, 1 being the slowest, 10 being the fastest, defaults to `6`. When set to `0` the turtle draws instantaneously.

up (*angle*)
rotate the turtle up

Parameters **angle** (*float*) – the angle in degrees to rotate.

walk ()
sets the turtle to 'walk', i.e. it has to move along the ground.

Change history

- 1.0.1 - MinecraftShape performance improvement
- 1.0 - added docs
- 0.10 - added Points class to simplify drawFace
- 0.9 - added MinecraftTurtle
- 0.8 - bug fixes
- 0.7 - minor changes
- 0.6 - python 3 compatibility changes
- 0.5 - replace sets.Set with set

- 0.4 - rotating shapes & code tidy up
- 0.3 - included DrawHollowSphere function
- 0.2 - extended with new drawing functions and MinecraftShapes class
- 0.1 - first beta release, MinecraftDrawing class

Python Module Index

Index