**The Board Logic**

**The coordinate system**

The hexagons in the board are positioned using a coordinate system. Because the individual position is a hexagon, there are three axes (shown in figure 1), x, y and z.

\\cabinet\work$\tup916\cmpt\cswin\Downloads\coordinates.png

The origin of the axis is in the middle of the board. The hexagon in the middle of the board is (0, 0, 0).

\\cabinet\work$\tup916\cmpt\cswin\Downloads\0coordinate.png

Therefore, the coordinate for each hexagon is relative to the origin at the centre of the board. The format is (x-coordiate, y-coordinate, z-coordinate).

Notice that 1 unit in the x-direction and 1 unit in the z-direction = 1 unit in the y-direction.

The system exploits this idea to have a consistent and unique coordinate for every hexagon in the board. Every hexagon in the board can solely be written in the form of x and z coordinate, without a y coordinate. A form of the coordinae where the y-coordinate is 0 is called reduced form, and there exists a method in the HexCoord class, reduce(), which converts a coordinate into its reduced form.

It is very tempting to throw away the y-coordinate and work with just the x and z coordinates. However, the y-coordinate is needed for other purposes such as calcualating the shortest distance, which is why we are allowing the y-coordinate to exist.

The following are the unique coordinates in the reduced form, given to each hexagon.

\\cabinet\work$\tup916\cmpt\cswin\Downloads\allcoordinates.png

Although the reduced form saves us from having inconsistent names for a hexagon, it is not quite useful for measuring the shortest distance between two hexagons. It is required to find the smallest x, y and z to get to the hexagon from the origin. This is why there is an un-reduce function called toVector() function which rearranges the coordinate for the purposes of calculating shortest relative displacement between the origin and the coordinate. The unreduce() function makes use of the same fact that 1 unit in +x, and 1 unit in +z results in 1 unit in +y direction, and 1 unit in -x, and 1 unit in -z results in 1 unit in -y direction.

**Calculating the shortest distance**

To calculate the shortest distance between two hexagonal coordinates, the two essential methods required are shortestDistance() and displacement().

The shortestDistance method

The shortestDistance method accepts two hexCoords. It first converts them to the reduced state.

To make the common case fast, it compares whether the two coordinates are aligned in x or y axis. If this is the case, then it just returns the distance between the two coordinates.

If however, this is not the case, the displacement method is called to calculate the shortest x, y and z lengths needed to go from the first coordinate to the second. The x, y, and z lengths are then added together and returned as the shortest distance.

The displacement method

The displacement method accepts two hexCoords. It first calculates the smalest x,y, and z needed to get to each coordinate individually from the origin using the toVector() method. It then calculates the x,y and z distance between the two coordinates.

Now, because of the fact that (1,0,1) 🡸🡺 (0, 1, 0) and ( -1,0, -1) 🡸🡺 (0, -1, 0), the method calculates the minimum x, y and z displacement between the two coordinates and returns the vector between the two coordinates. This vector stores information about how many minimum units of x, y and z are needed to get from the first coordinate to the second coordinate.