There are two main coordinate systems used in the game.

The global coordinate system is shown below as the three blue axes X,Y and Z. Their purpose is to allow the designer of the game to easily input coordinates. For this reason they can be set up however the designer chooses.

In my Pygame, to make things easier, the Z axis points to the zenith and the X and Y axes lie on the horizontal plane to make a right hand coordinate system. The origin is also at the centre of the room.

The local coordinate system is shown below as the three solid yellow axes x, y and z. Their purpose is to make it easier for the global points to be plotted to canvas when the player turns. The y axis is the direction the player is looking and the x-z plane is parallel to the canvas plane which points will be projected on later.

In my Pygame, is the angle the player spins on the horizontal plane and is the angle of elevation that the player looks down.



To get any global point P=[X,Y,Z] into local coordinates p=[x,y,z], a rotation matrix is needed. To form a rotation matrix, R, so that p=RP, the rows of R will be the unit vectors of the local coordinate system i.e. .

The x axis will always stay on the XY plane so has no component in Z. Since it is just rotated on the horizontal unit circle its vector is

The y axis will be in the same plane as z and Z. This plane also contains a new vector s, such that s lies on the horizontal plane perpendicular to x.

Consider the diagram of the triangle above which has as a side. The horizontal component of is a factor of . The vertical component of is . In that triangle:

By scaling to

You can get as

Similarly, consider the diagram of the triangle above which has as a side. The horizontal component of is a factor of . The vertical component of is . In that triangle:

By scaling to

You can get as

The rotational matrix then has rows R = [, , ]

In my Pygame the function to make the rotational matrix is called every frame. It is named ‘updateRotationMatrix’ and appears in the code like below

# derived from the placement of the local unit vectors depending on t and a

# see ‘Rotation matix.docx’ for explanation

def updateRotationMatrix(t, a):

return [

[cos(t), sin(t), 0],

[-cos(a)\*sin(t), cos(a)\*cos(t), -sin(a)],

[-sin(a)\*sin(t), sin(a)\*cos(t), cos(a)]

]