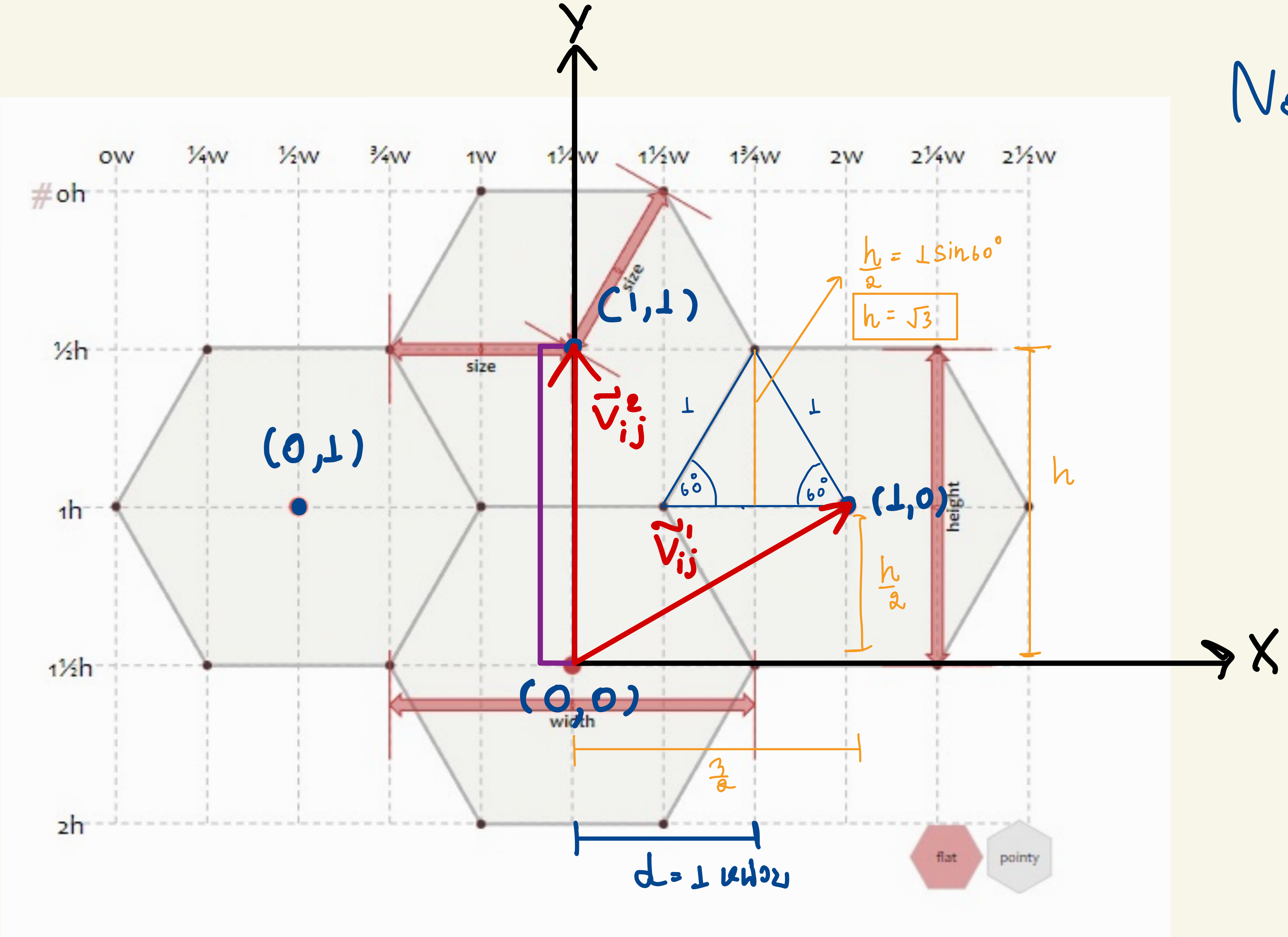
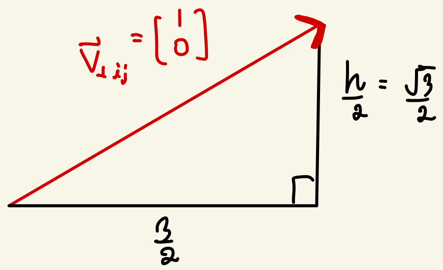
Note by 1. 6368
2. 6362



พิจารณา Vector $\vec{V}_1, \vec{V}_2$ ในพิกัด Hexagonal, Cartesian จะได้ว่า

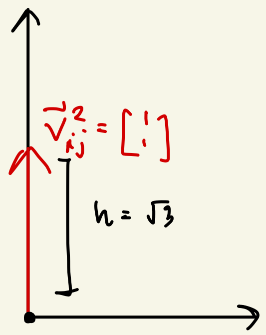


$\vec{V}^1_{ij} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ และ $\vec{V}^1_{xy} = \begin{bmatrix} \frac{3}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix}$

แสดงว่า linear tranformation eq ของพิกัด Hexagonal, Cartesian คือ

$$\vec{V}^1_{ij} = M\vec{V}^1_{xy}$$

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \frac{3}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix} \quad -①$$



$\vec{V}^2_{ij} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ และ $\vec{V}^2_{xy} = \begin{bmatrix} 0 \\ \sqrt{3} \end{bmatrix}$

$$\vec{V}^2_{ij} = M\vec{V}^2_{xy}$$

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 \\ \sqrt{3} \end{bmatrix} \quad -②$$

แก้สมการ ① และ ② จะได้

$$1 = \frac{3}{2}a + \frac{\sqrt{3}}{2}b \quad -③$$

$$0 = \frac{3}{2}c + \frac{\sqrt{3}}{2}d \quad -④$$

$$1 = 0a + \sqrt{3}b \quad -⑤$$

$$1 = 0c + \sqrt{3}d \quad -⑥$$

แก้สมการ ③,④,⑤,⑥ จะได้

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{\sqrt{3}} \\ -\frac{1}{3} & \frac{1}{\sqrt{3}} \end{bmatrix} \text{ หรือ } \begin{bmatrix} \frac{1}{3} & \frac{\sqrt{3}}{3} \\ -\frac{1}{3} & \frac{\sqrt{3}}{3} \end{bmatrix}$$

แสดงว่าเราสามารถแปลงพิกัด Hexagonal เป็น Cartesian ได้โดยใช้สมการ linear transformation

$$\boxed{\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{\sqrt{3}} \\ -\frac{1}{3} & \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}}$$

Method 1 : พิจารณา Matric Position $\begin{bmatrix} x \\ y \end{bmatrix}$ (2x1)

ในทำนองเดียวกัน เราก็สามารถแปลงพิกัด Cartesian เป็น Hexagonal ได้โดยใช้ Inv linear transformation ของการแปลง Hexagonal เป็น Cartesian

Display decimals

$$\begin{pmatrix} \frac{1}{3} & \frac{1}{3^{0.5}} \\ \frac{-1}{3} & \frac{1}{3^{0.5}} \end{pmatrix}^{(-1)} = \begin{pmatrix} \frac{3}{2} & \frac{-3}{2} \\ \frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \end{pmatrix}$$

▼ Details

Find 2×2 matrix inverse according to the formula: $A^{(-1)} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{(-1)} = \frac{1}{\det(A)} \cdot \begin{pmatrix} C_{11} & C_{21} \\ C_{12} & C_{22} \end{pmatrix} = \frac{1}{ad-bc} \cdot \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$

$$\begin{pmatrix} \frac{1}{3} & \frac{\sqrt{3}}{3} \\ \frac{-1}{3} & \frac{\sqrt{3}}{3} \end{pmatrix}^{(-1)} = \frac{1}{\frac{1}{3}\cdot\left(\frac{\sqrt{3}}{3}\right) - \frac{\sqrt{3}}{3}\cdot\left(\frac{-1}{3}\right)} \cdot \begin{pmatrix} \frac{\sqrt{3}}{3} & \frac{-\sqrt{3}}{3} \\ \frac{1}{3} & \frac{1}{3} \end{pmatrix} = \begin{pmatrix} \frac{3}{2} & \frac{-3}{2} \\ \frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \end{pmatrix}$$

จะได้

$$\boxed{\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{3}{2} & -\frac{3}{2} \\ \frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix}}$$

```python
def car2hex(self, posCartesian):
    # posCartesian is a 2x1 vector = [x,y]
    # posHexagonal is a 2x1 vector = [i,j]
    # linear transformation matrix is a 2x2 matrix = [[1/3, 1/np.sqrt(3],[-1/3, 1/np.sqrt(3]]
    linearTrans = np.array([[1/3, 1/np.sqrt(3)]
                            ,[-1/3, 1/np.sqrt(3)]])
    # posHexagonal = linear transformation matrix * posCartesian
    poshexagonal = np.matmul(linearTrans,posCartesian)
    # round to nearest integer and convert to int
    poshexagonal = np.rint(poshexagonal).astype(int)
    return poshexagonal

def hex2car(self, posHexagonal):
    # posHexagonal is a 2x1 vector = [i,j]
    # posCartesian is a 2x1 vector = [x,y]
    # linear transformation matrix is a 2x2 matrix = [[3/2, -3/2],[np.sqrt(3)/2,np.sqrt(3)/2]]
    linearTrans = np.array([[3/2,-3/2],
                            [np.sqrt(3)/2, np.sqrt(3)/2]])
    # posCartesian = linear transformation matrix * posHexagonal
    posCartesian = np.matmul(linearTrans,posHexagonal)
    return posCartesian
```

Method 2 : พิจารณาเป็น Homogenous tranformation Matrix

จากการคำนวณ จะได้ว่า

① การแปลง Frame จาก Cartesian เป็น Hexagonal

No translate

$$\left[\begin{array}{cc|c} \multicolumn{2}{c|}{R_{Car}} & P_{Car} \\ \hline 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{cc|c} \frac{3}{2} & -\frac{3}{2} & 0 \\ \frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} & 0 \\ \hline 0 & 0 & 1 \end{array}\right] \left[\begin{array}{cc|c} \multicolumn{2}{c|}{R_{Car}} & P_{Car} \\ \hline 0 & 0 & 1 \end{array}\right]$$

linear transform

```python
def car2hex(self, posCartesian):
    # convert the position of the BeeBot from Cartesian coordinate to Hexagonal coordinate
    # posCartesian is a 3x3 matrix
    # posHexagonal is a 3x3 matrix
    # linear transformation matrix is 3x3 matrix
    linearTrans = np.array([[1/3, 1/np.sqrt(3),0]
                            ,[-1/3, 1/np.sqrt(3),0],
                            [0,0,1]])
    # posHexagonal = linearTrans * posCartesian
    poshexagonal = np.matmul(linearTrans,posCartesian)
    # round to nearest integer and convert to int
    poshexagonal = np.rint(poshexagonal).astype(int)
    return poshexagonal
```

② การแปลง Frame จาก Hexagonal เป็น Cartesian

$$\left[\begin{array}{cc|c} \multicolumn{2}{c|}{R_{hex}} & P_{hex} \\ \hline 0 & 0 & 1 \end{array}\right] = \underbrace{\left[\begin{array}{cc|c} \frac{1}{3} & \frac{1}{\sqrt{3}} & 0 \\ -\frac{1}{3} & \frac{1}{\sqrt{3}} & 0 \\ \hline 0 & 0 & 1 \end{array}\right]}_{\text{linear transform}} \left[\begin{array}{cc|c} \multicolumn{2}{c|}{R_{car}} & P_{car} \\ \hline 0 & 0 & 1 \end{array}\right]$$

```python
def hex2car(self, posHexagonal):
    # convert the position of the BeeBot from Hexagonal coordinate to Cartesian coordinate
    # posCartesian is a 3x3 matrix
    # posHexagonal is a 3x3 matrix
    # linear transformation matrix is 3x3 matrix
    linearTrans = np.array([[3/2,-3/2,0],
                            [np.sqrt(3)/2, np.sqrt(3)/2,0],
                            [0,0,1]])
    # posCartesian =  linearTrans * posHexagonal
    posCartesian = np.matmul(linearTrans,posHexagonal)
    return posCartesian
```

③ เราเลือกที่จะพิจารณาการหมุน Beebot ในพิกัด Cartesian

$$\left[\begin{array}{cc|c} \multicolumn{2}{c|}{R_{new}} & P_{new} \\ \hline 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{cc|c} \multicolumn{2}{c|}{R_{old}} & P_{old} \\ \hline 0 & 0 & 1 \end{array}\right] \overset{\text{หมุน } 60^\circ\ [\text{comman } 3]}{\begin{bmatrix} \cos 60^\circ & -\sin 60^\circ & 0 \\ \sin 60^\circ & \cos 60^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}} \overset{\text{หมุน } -60^\circ\ [\text{comman4}]}{\begin{bmatrix} \cos 60^\circ & -\sin 60^\circ & 0 \\ \sin 60^\circ & \cos 60^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}} \cdots$$

องศา

คูณต่อไปเรื่อย หากมีการสั่งหมุน

```python
def RotationalMatrix(self,Matrix,rotation):
    # theta is the rotation angle around z-axis
    # first get sin and cos of theta in radian
    zC, zS = self.sinCos(rotation[0])
    # Rotation matrix around z-axis
    Rotate_Z_matrix = np.array([[zC, -zS, 0],
                                [zS, zC, 0],
                                [0, 0, 1]])
    return np.matmul(Matrix,Rotate_Z_matrix)
```

4 เราเลือกที่จะพิจารณาการเดิน,ถอย Bee bot ใน พิกัด Cartesian

$$
\begin{bmatrix} R_{new} & P_{new} \\ 0 \quad 0 & 1 \end{bmatrix}
=
\begin{bmatrix} R_{old} & P_{old} \\ 0 \quad 0 & 1 \end{bmatrix}
\overbrace{\begin{bmatrix} R_{All} & \begin{matrix} 0 \\ 0 \end{matrix} \\ 0 \quad 0 & 1 \end{bmatrix}}^{\text{Comman ที่เกบหมุนทั้งหมด}}
\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \sqrt{3} \\ 0 & 0 & 1 \end{bmatrix}
$$

เดินหน้า

หรือ

$$
\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -\sqrt{3} \\ 0 & 0 & 1 \end{bmatrix}
$$

ถวยหลัง

```python
# clear command '0' in com because it is not moving, does not effect the position and does not need to be plotted
com = com.replace('0','')
for c in com :
    if c == '1': # forward command
        buffer = np.matmul(self.posCar,self.transMatrixForward)
        if not self.checkCollision(W, self.car2hex(buffer)[0:2,2].reshape(2,1)):
            self.posCar = buffer
            self.posHex = self.car2hex(self.posCar)
            A = np.append(A, self.getPosition(self.posHex), axis=1)
            P = np.append(P, self.getPosition(self.posCar), axis=1)
    elif c == '2': # backward command
        buffer = np.matmul(self.posCar,self.transMatrixBackward)
        if not self.checkCollision(W, self.car2hex(buffer)[0:2,2].reshape(2,1)):
            self.posCar = buffer
            self.posHex = self.car2hex(self.posCar)
            A = np.append(A, self.getPosition(self.posHex), axis=1)
            P = np.append(P, self.getPosition(self.posCar), axis=1)
    elif c == '3': # turn left command (+60 degree)
        self.posCar = self.RotationalMatrix(self.posCar, [60])
    elif c == '4': # turn right command (-60 degree)
        self.posCar = self.RotationalMatrix(self.posCar, [-60])
return A , P
```