Coordinate Transformations Between Square and Hex Grids

John Lusk

August 26, 2011

In a square grid, cell coordinates are simple. The origin square is $(0,0)$ (labelling the *center* of the square), the square immediately to the right of the origin cell is $(1,0)$. the square immediately above the origin cell is $(0,1)$, and so on.

In a hex grid, we can do the same, but the axes must be tilted. The origin hex is $(0,0)$ (again, labelling the center of the hex), the hex immediately to the right of the origin hex is $(1,0)$, the hex immediately to the "northeast" of the origin hex is $(0,1)$, and so on.

We want a mathematical transformation between the two coordinate systems, to be used for the following purposes:

- Hit-testing. We want to transform a mouse click (in magenta), in square-space coordinates (shown in maroon in Figure 1), to hex-space coordinates (shown in blue).
- Rendering. We want to know where to draw a particular hex, given in hex-space coordinates, in square-space coordinates.

See this paper for basic change-of-basis matrices: `http://www.math.ucsd.edu/~nslingle/bases.pdf`.

1 Change-of-basis matrix

If the distance between hex centers is 1 unit (in GURPS, say, 1 yard), then r, the distance from a hex center to the center of any of its edges is

$$r = 0.5 \tag{1}$$

And s, which is both the distance from the hex's center to any of its vertices *and* the length of a side (since a hex is composed of equilateral triangles) is

$$\begin{aligned} s &= \frac{0.5}{\cos\left(\frac{\pi}{6}\right)} \\ &\approx 0.577 \end{aligned} \tag{2}$$

(See the maroon calculation in the lower-left corner of Figure 1.)

The basis vectors in what I will call "hex space" ($\mathbb{H}^2$), in blue at the lower-left corner of Figure 1, are given (in terms of the traditional Cartesian space $\mathbb{R}^2$) as $\mathbb{H}^2$

$$(1,0)$$

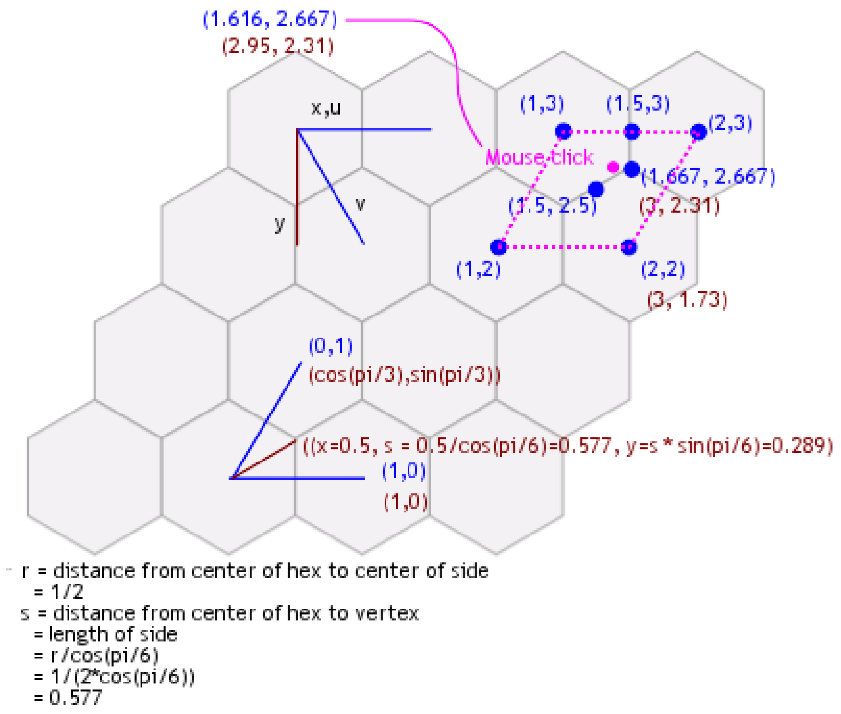

Figure 1: Hex Grid Elements To Be Represented In Software

and

$$\begin{aligned}\left(\cos\left(\frac{\pi}{3}\right), \sin\left(\frac{\pi}{3}\right)\right) &= \left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right) \\ &\approx (0.5, 0.866)\end{aligned}$$

We write this as a matrix and multiply it by a vector in $\mathbb{H}^2$ to get a vector in $\mathbb{R}^2$:

$$\begin{aligned} M &= \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} \end{bmatrix} \\ \vec{x}_{\mathbb{R}^2} &= M\vec{x}_{\mathbb{H}^2} \end{aligned} \tag{3}$$

As an example, we transform the base vectors, given in $\mathbb{H}^2$, into $\mathbb{R}^2$, and we see that we get the $\mathbb{R}^2$ versions we worked out above:

$$\begin{aligned} \begin{bmatrix} 1 \\ 0 \end{bmatrix} &= M \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix} &= M \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{aligned}$$

Inverting the matrix (see http://mathworld.wolfram.com/MatrixInverse.html) should get us a matrix capable of performing the opposite transformation (from $\mathbb{R}^2$ to $\mathbb{H}^2$):

$$M^{-1} = \begin{bmatrix} 1 & \frac{-1}{\sqrt{3}} \\ 0 & \frac{2}{\sqrt{3}} \end{bmatrix} \tag{4}$$

2 Hit-testing

The magenta point in Figure 1 represents a mouse-click, $\vec{m}_{\mathbb{R}^2}$, whose coordinates arrive to us in $\mathbb{R}^2$. Our objective is to figure out which hex this corresponds to, without a lot of trigonometry and conditional logic dependent on edge slopes and x-y coordinates. We transform $\vec{m}_{\mathbb{R}^2}$ to $\mathbb{H}^2$ by multiplying by M^{-1}:

$$\vec{m}_{\mathbb{H}^2} = M^{-1}\vec{m}_{\mathbb{R}^2}$$

We then round each coordinate of $\vec{m}_{\mathbb{H}^2}$ both up and down to give four candidate "closest hex centers"' (using $\vec{h}$ to represent $\vec{m}_{\mathbb{H}^2}$, u to represent the first basis vector of $\mathbb{H}^2$, v to represent the second basis vector of $\mathbb{H}^2$, to simplify the notation):

$$\begin{aligned} (p_1)_{\mathbb{H}^2} &= (\lfloor h_u \rfloor, \lfloor h_v \rfloor) \\ (p_2)_{\mathbb{H}^2} &= (\lceil h_u \rceil, \lceil h_v \rceil) \\ (p_3)_{\mathbb{H}^2} &= (\lfloor h_u \rfloor, \lceil h_v \rceil) \\ (p_4)_{\mathbb{H}^2} &= (\lceil h_u \rceil, \lfloor h_v \rfloor) \end{aligned}$$

We then multiply each of the four hex-center points by M, to transform them back to $\mathbb{R}^2$ (I'm not sure this is necessary – will Pythagoras work in $\mathbb{H}^2$?), and use Pythagoras to measure the square of the distance of the mouse-click point to each of the four hex centers. For the first point, p_1, in $\mathbb{R}^2$:

$$\begin{aligned}(\vec{p}_1)_{\mathbb{R}^2} &= M \cdot (\vec{p}_1)_{\mathbb{H}^2} \\ r_1 &= |\vec{m}_{\mathbb{R}^2} - (\vec{p}_1)_{\mathbb{R}^2}|\end{aligned}$$

And so on. Of the $r_1 \ldots r_4$, the smallest one corresponds to the closest hex center. (Note that in the software, we don't need to pay the cost of square root. Since we're just interested in the smallest value, we can use the square of the distance, since that requires fewer compute cycles.)

3 Some values

$$\begin{aligned}\frac{1}{2\cos\left(\frac{\pi}{6}\right)} &= \frac{1}{\sqrt{3}} \\ &\approx 0.577 \\ \frac{1}{4\cos\left(\frac{\pi}{6}\right)} &\approx 0.289\end{aligned}$$

4 Origin at top left

Most computer graphics systems seem to have their origins at the top left corner of the window, with increasing y values appearing lower on the screen. (See the axes drawn in the top left corner of Figure 1. $\mathbb{R}^2$ axes are x and y, while $\mathbb{H}^2$ axes are u and v.) I don't think this really changes anything, since we'll be dealing with mouse clicks and path-finding, so I'm not going to worry about it in my code. One thing I may need to take into account is icon bitmap orientation, when placed on the screen. I may need to reverse "up" and "down" and "clockwise" and "counterclockwise" in cases like that.

5 "Winged edge" data structure

The "winged edge" data structure is illustrated in the top right corner of Figure 2. Pick any edge (say, the black one). The two vertices are v_0 and v_1. Travelling from v_0 to v_1, face f_0 is on the left, while face f_1 is on the right. Considering vertex v_i, the "more counterclockwise" (farther in the counterclockwise direction from v_i) neighboring edge is e_i^+, and the "less counterclockwise" edge is e_i^-. Each face (hex) points to an arbitrary edge bordering it, and each vertex points to arbitrary incident edge.

```
class Edge {
        Vertex  vStart, vEnd;
        Face    fLeft, fRight;
        Edge    e0ccw, e0cw,        // ccw = +, cw = -
                e1ccw, e1cw;
}
```

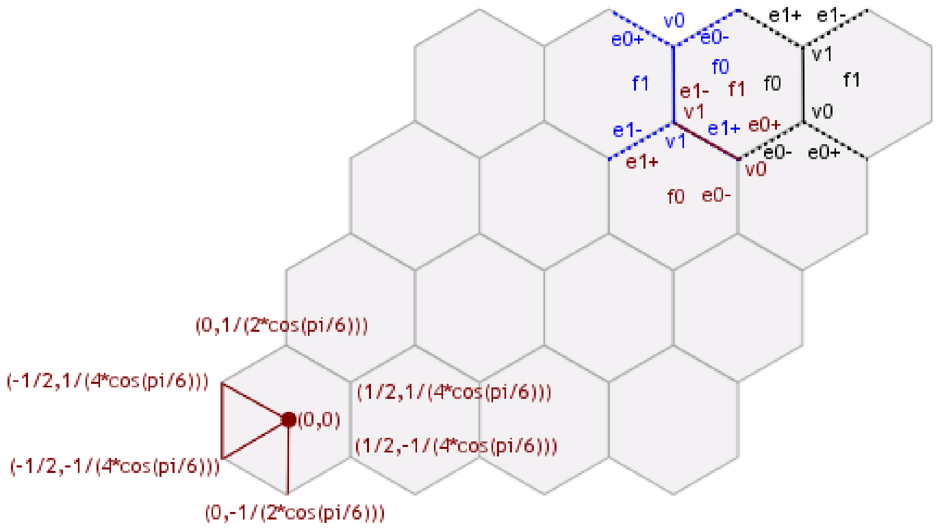

Figure 2: Hex Grid Vertexes and Edges

A complete circuit around a hex (for example) is taken by traversing the `ccw` edges six times. You'll have to figure out whether the current vertex is v_0 or v_1 whenever you enter a new edge, but that shouldn't be hard.

Each face (hex) will have attributes such as background and foreground colors, icon (e.g., game piece). If rivers flow along edges, then the edge will have a "river" attribute (e.g., a long, thin icon to be laid on top of the edge). (Not yet sure how to easily do hit-testing against edges and vertices.)

6 Data Structure Set-Up and Initial Rendering

6.1 Hex numbering in $\mathbb{H}^2$

- Origin hex is $(0, 0)$.
- Rightmost hex of bottom row is $(\texttt{numHorizHexes} - 1, 0)$.
- Leftmost hex of next row up (row 1) is $(0, 1)$.
- Rightmost hex of row 1 is $(\texttt{numHorizHexes} - 1, 1)$.
- Leftmost hex of row 2 is $(-1, 2)$.
- Rightmost hex of row 2 is $(\texttt{numHorizHexes} - 2, 2)$.
- Leftmost hex of row 3 is $(-1, 3)$.
- Rightmost hex of row 3 is $(\texttt{numHorizHexes} - 2, 3)$.
- Leftmost hexes of rows i and $i+1$, where $i \bmod 2 = 0$, are $\left(-\frac{i}{2}, i\right)$ and $\left(-\frac{i}{2}, i+1\right)$. Rightmost hexes are $\left(\texttt{numHorizHexes} - 1 - \frac{i}{2}, i\right)$ and $\left(\texttt{numHorizHexes} - 1 - \frac{i}{2}, i+1\right)$.

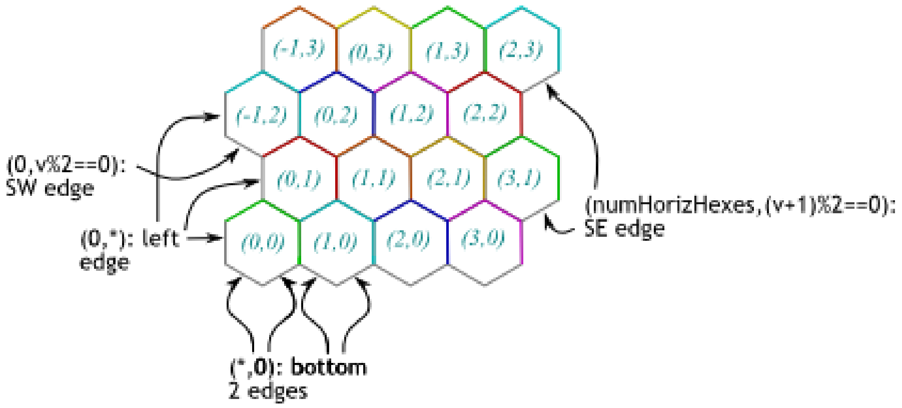

Figure 3: Hex-Edge Rendering

Define $u_{\min}$, $u_{\max}$

$$\begin{aligned} u_{\min} &= -\frac{i}{2} \\ u_{\max} &= \texttt{numHorizHexes} - 1 - \frac{i}{2} \end{aligned}$$

for rows i and $i+1$, where $i \bmod 2 = 0$.

6.2 Set-Up

Hex $(0,0)$ (in $\mathbb{H}^2$) is initialized with all six of its edges, starting with the one to the direct "east" and proceeding clockwise.

Vertex v_0 of the first edge is the lower of the two vertices (see Figure 2), $\left(\frac{1}{2}, \frac{-1}{4\cos\left(\frac{\pi}{6}\right)}\right) = (0.5, -0.289)$ (in $\mathbb{R}^2$). We initialize the first three edges (shown in Figure 3 in green), then the left edge (gray, with vertex v_0 being the lower vertex of that edge, as if it were the eastern edge of a hex to the west), then the bottom two edges (also gray). Note that all three gray edges run "backwards", meaning, as one traverses the edges of the hex counterclockwise, one encounters vertex v_1 of edge before v_0. This is the same for all hexes, due to the fact that a hex shares edges with its neighbors, and if they were all spinning counterclockwise, they would "grind" against one another.

Then we proceed to the next hex, $(1,0)$ in $\mathbb{H}^2$, beginning with same edge of that hex, the "eastern" vertical edge, and proceeding counterclockwise through the three cyan edges. The last cyan edge links to the first edge of hex $(0,0)$. Then, the bottom two edges, as for hex $(0,0)$.

Having finished the first row, we proceed to the second row. Hex $(0,1)$ in $\mathbb{H}^2$ has its red edges created as usual. Then, its left (western, gray) edge is created in the same way as hex $(0,0)$'s. The southwestern edge (green) is the second in hex $(0,0)$'s linked list of edges (proceeding clockwise), and the southeastern edge (cyan) is the third in hex $(1,0)$'s list of edges.

Hex $(1,1)$ is the first of the "normal" hexes, no gray edges. The orange edges are initialized in the usual order, the red (western) edge is hex $(0,1)$'s first edge, the cyan (southwestern) edge is hex $(1,0)$'s second edge, and the blue (southeastern) edge is hex $(2,0)$'s third edge.

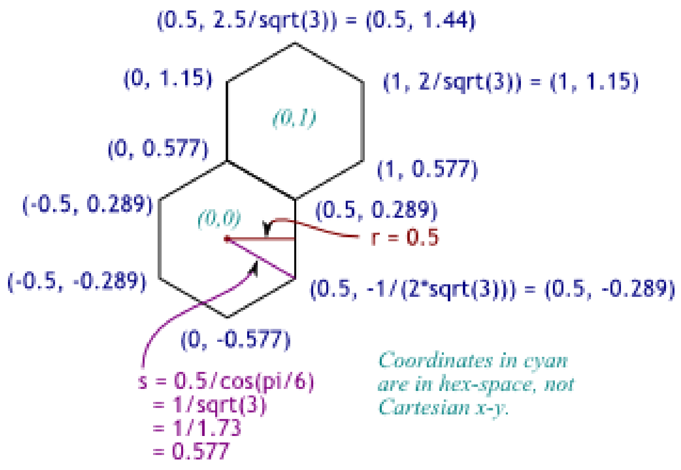
(0.5, 2.5/sqrt(3)) = (0.5, 1.44)
(0, 1.15)
(1, 2/sqrt(3)) = (1, 1.15)
(0,1)
(0, 0.577)
(1, 0.577)
(-0.5, 0.289)
(0,0)
(0.5, 0.289)
r = 0.5
(-0.5, -0.289)
(0.5, -1/(2*sqrt(3))) = (0.5, -0.289)
(0, -0.577)
s = 0.5/cos(pi/6)
= 1/sqrt(3)
= 1/1.73
= 0.577
Coordinates in cyan are in hex-space, not Cartesian x-y.

Figure 4: Hex Vertex Coordinates in $\mathbb{R}^2$

At the end of the row, hex (`numHorizHexes`, 1)'s southeastern edge (gray) must be created, running "backwards", as usual.

So, for each hex (u, v), the first three edges are instantiated, but the next three edges are:

- Western edge: hex $(u-1, v)$'s first edge (unless $u = u_{\min}$, in which case we instantiate a new edge running backwards);
- Southwestern edge: hex $(u, v-1)$'s second edge (unless $v = 0$ *or* $(u = u_{\min}$ and $v \bmod 2 = 0)$, in which case we instantiate a new edge running backwards);
- Southeastern edge: hex $(u+1, v-1)$'s third edge (unless $v = 0$ *or* $(u = u_{\max}$ and $v \bmod 2 = 1)$, in which case we instantiate a new edge running backwards).

6.3 Render

Rendering is done the same way as setting up, except that we substitute the word "draw" whereever "instantiate" appears. For the sake of rendering speed, we *may* want to implement the list of edges for a hex as an array rather than a linked list, although this will only have beneficial effect for edge hexes (for interior hexes, we simply draw three edges (first, second, third) and move on to the next hex).

6.4 Vertex Coordinates in $\mathbb{R}^2$

In the following discussion, r, the distance from a hex center to the center of an edge is half the intra-hex distance, and s, the length of a hex side (and also the distance from the center of a hex to a vertex), are r, s

$$\begin{aligned} r &= \frac{1}{2} \\ s &= \frac{1}{\sqrt{3}} \end{aligned}$$

For integers $i \in [0, \texttt{numHorizHexes} - 1]$, $j \in [0, \texttt{numVertHexes} - 1]$, the coordinates (u, v) of the i-th hex of the j-th row are

$$\begin{aligned} u &= i - (j \bmod 2)\,\frac{j}{2} \\ v &= j \end{aligned}$$

and the coordinates of hex (u, v) are:

$$\begin{aligned} v_0 &= \left(((j \bmod 2) + 2i + 1)\,r, \frac{(3j-1)\,s}{2} \right) \\ v_1 &= \left(((j \bmod 2) + 2i + 1)\,r, \frac{(3j+1)\,s}{2} \right) \\ v_2 &= \left(((j \bmod 2) + 2i)\,r, \frac{(3j+2)\,s}{2} \right) \\ v_3 &= \left(((j \bmod 2) + 2i - 1)\,r, \frac{(3j+1)\,s}{2} \right) \\ v_4 &= \left(((j \bmod 2) + 2i - 1)\,r, \frac{(3j-1)\,s}{2} \right) \\ v_5 &= \left(((j \bmod 2) + 2i)\,r, \frac{(3j-2)\,s}{2} \right) \end{aligned}$$

For most hexes, only v_0, v_1, and v_2 need to be calculated.

(old, obsolete text:)
For integers $u \in [0, \texttt{numHorizHexes} - 1]$, $v \in [0, \texttt{numVertHexes} - 1]$, when $v \bmod 2 = 0$,

$$\begin{aligned} v_0 &= \left(\frac{1}{2} + u - \frac{v}{2}, -\frac{1}{2}s + 2vs \right) \\ &= \left(\frac{1}{2} + u - \frac{v}{2}, \frac{s\,(4v-1)}{2} \right) \\ v_1 &= \left(\frac{1}{2} + u - \frac{v}{2}, \frac{1}{2}s + 2vs \right) \\ &= \left(\frac{1}{2} + u - \frac{v}{2}, \frac{s\,(4v+1)}{2} \right) \\ v_2 &= \left(u - \frac{v}{2}, s + \frac{3s}{2} \right) \end{aligned}$$