Calculating Cartesian Movement to Isometric Movement via Multiple Orientations

Written on August 27, 2017

Introduction

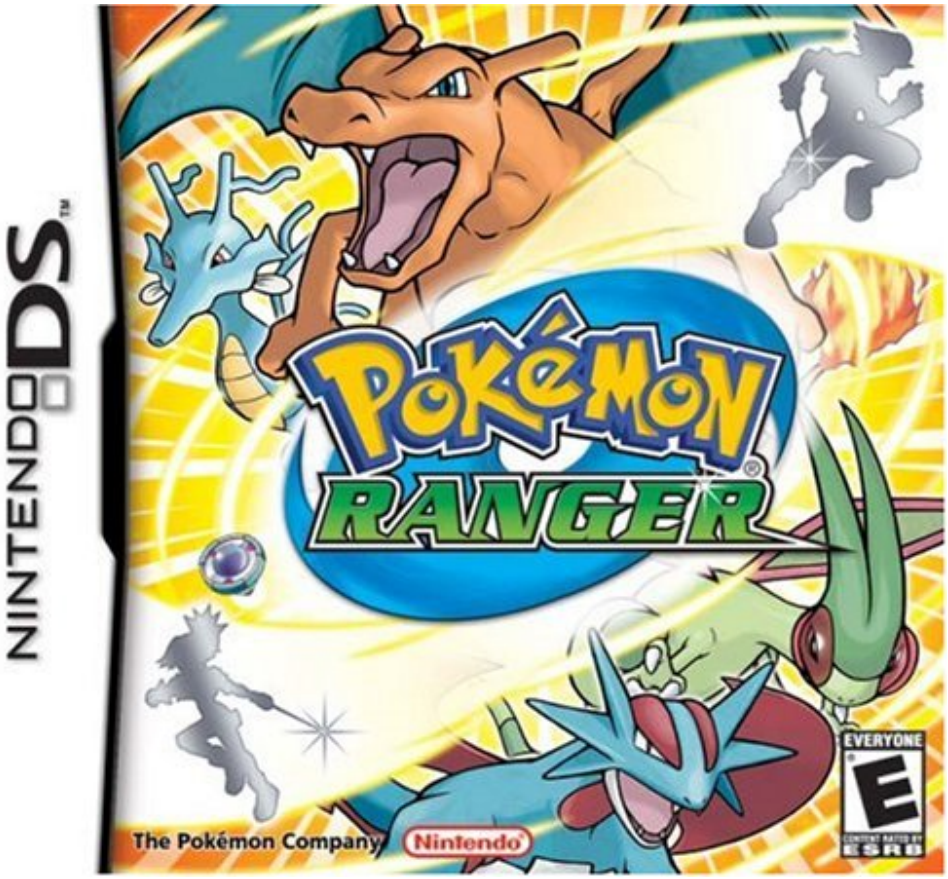
I finally was able to tackle this problem and have succeeded in solving the long-sought answer on making a simp isometric movement.

One might say this is easy to do, but when you think about the following context, you'll understand just how important it is to understand where I'm coming from in regards to the difficulty of solving this answer.

It all began with a favorite game of mine during my childhood, MegaMan Battle Network, as shown:

Walking around in the game world, I noticed a peculiar thing about the way the main character moves around in t game. When pressing UP, I was able to move "diagonally" in the world, and when pressing UP + RIGHT, I was ab to move "up" in the world. As far as I can tell, the game doesn't distinguish which way is the true up. But nonetheless, the game is playable.

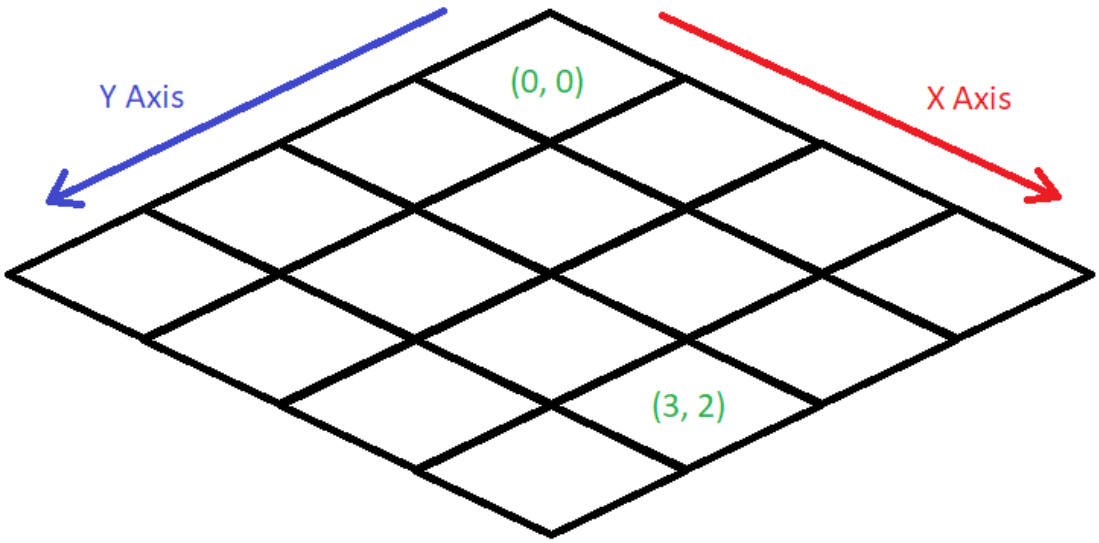
Then I came across another instance of the game, which uses isometric movement is Pokemon Ranger.

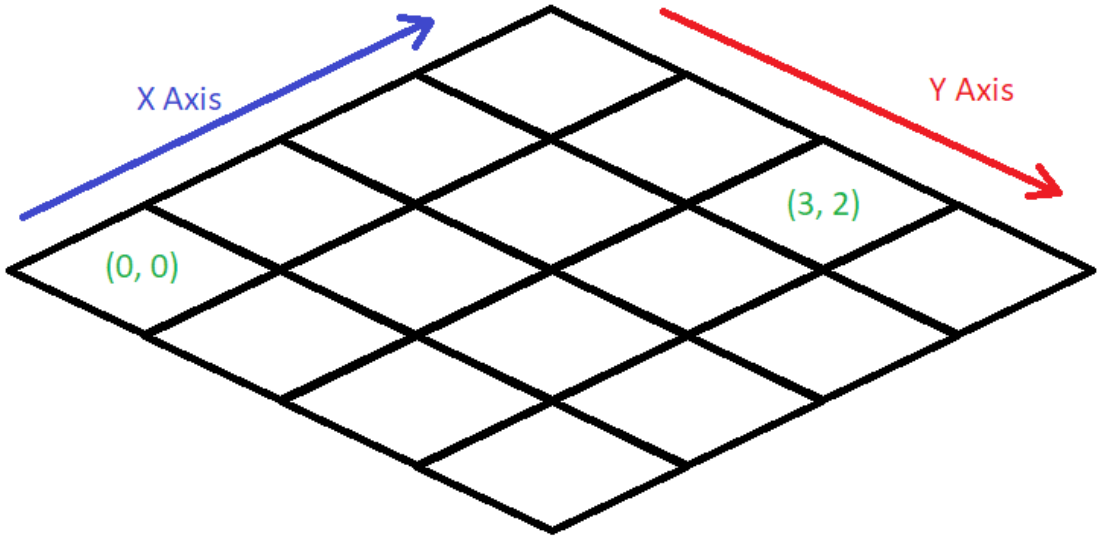


Now granted, somewhere along those lines, and all those games that I have played, there was a moment where was found the Settings menu, in which you can choose where the Up direction should be. The game would allow me to choose whether Northwest, North, or Northeast, should be my primary Up direction. I may have forgotten which game it is, but that was the spark of this motivation for me to understand how the isometric movements wo and in such a way that it can be interchangeable across multiple orientations.

Isometric Grids

Below shows the two common orientations for isometric movements. The first one shows the Up direction oriente towards the Northeast direction, and the other one shows the Up direction oriented towards the Northwest directi





We will first look at the first picture and create the isometric coordinates here for moving around, and then look to see how we can apply the same isometric calculations to the second picture. We will be using the most common isometric aspect ratio, where the width is longer than the height.

Understanding the Isometric Grids and Cartesian Grids

All of this section uses the first picture.

The Cartesian coordinates are rotated in such a way where the positive X is going southeast, and positive Y is going southwest.

Using a sample of 4x4 perpendicular grid, we can turn the Cartesian grid, so that:

- X axis is rotated -30 degrees.
- Y axis is rotated -60 degrees.

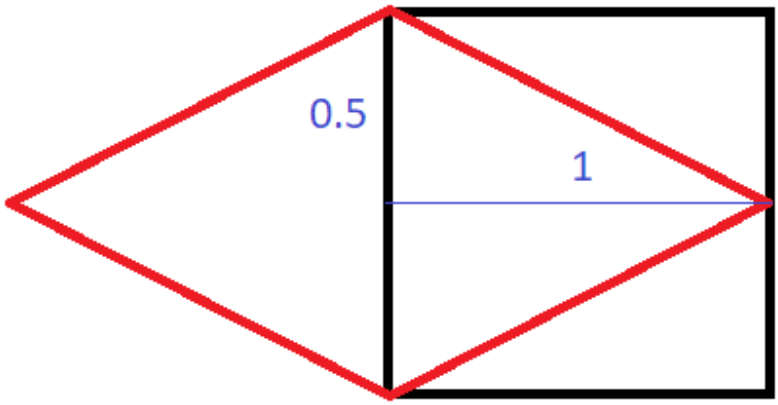
In order to get the illusion that negative Y (up) is facing to the northeast bound direction and positive X (right) is facing southeast bound direction.

By turning the grid in such a way, we can see that width to height ratio has changed from:

- 1:1 to 2:1 (width:height)

When a unit moves in 1 direction on the Cartesian grid, that unit would move both on the screen X and screen Y axes.

Imagine a rhombus (diamond shape laid flat horizontally). When compared to a square grid, we can see that by using the ratio given above, the diagonals of an isometric grid cross the half of the Cartesian square grid. This ca easily be seen if we imagine the rhombus overlaid on top of the square.



Therefore, we can see that if we move by +1 on the X axis, it is as follows:

- Cartesian: (+1, 0)
- Isometric: (+1, +0.5)

Expanding this to moving by +1 on the Y axis, it is as follows:

- Cartesian: (0, +1)
- Isometric: (-1, +0.5)

Therefore, we do the following calculations:

1. First, we consider moving by (+1, 0) on the Cartesian grid. We move by +1 on the X axis, and we move b +0.5 on the Y axis.
2. Second, we consider moving by (0, +1) on the Cartesian grid. We move by -1 on the X axis, and we mov by +0.5 on the Y axis.

First () belongs to the X axis. Second () belongs to the Y axis. (newX, newY) is the target isometric coordinates. T coordinates are based on the isometric rhombus movement of 2:1, ****NOT**** the original Cartesian grid movement

Target Coordinates	=	Cartesian X Axis	+	Cartesian Y Axis
newX	=	x	+	(-y)
newY	=	x/2	+	y/2

Of which, newY can be simplified as (x+y)/2.

Thus, we have derived our Cartesian coordinates to convert into Isometric coordinates.

Applying to Another Orientation

Normally, when moving downwards, we are moving in the northeast direction.

To go right, we need to move in the northeast direction. Thus:

- Cartesian: (+1, 0)
- Isometric: (+1, -0.5)

Same with moving downwards, we need to move in the southeast direction. Thus:

- Cartesian: (0, +1)
- Isometric: (+1, +0.5)

Again, using the isometric 2:1 rhombus as our reference coordinates, not the original Cartesian square grid coordinates. Therefore, we get:

Target Coordinates	=	Cartesian X Axis	+	Cartesian Y Axis
newX	=	x	+	y
newY	=	(-x)/2	+	y/2

Of which, newY can be simplified into (-x+y)/2.

Conclusion

We now have the ability to convert one isometric orientation to the other orientation. With a simple `switch...case` statement, we can choose whether to use the Isometric Movement 1, Isometric Movement 2, o Cartesian Movement by using the following code:

`flipUp` is an enumeration variable. Vector2D is a struct containing the X and Y values.

```
Vector2D CartesianToIsometric(Vector2D position, int flipUp){
    float x = position.x;
    float y = position.y;
    float newX, newY;
    switch (flipUp){
        case 0:
            newX = (x - y);
            newY = (x + y) / 2.0f;
            break;
        case 1:
        default:
            newX = x;
            newY = y;
            break;
        case 2:
            newX = (x + y);
            newY = (-x + y) / 2.0f;
```

```
                break;
            }
            return Vector2D(newX, newY);
        }
```

Back