Isometric Coordinates Documentation

Release 1.0

Paul Vincent Craven

May 15, 2018

Contents

Traditional top-down or side-view games normally work in a traditional grid. Graphics are placed in these grid locations. Each graphic is a rectangle, and sometimes they are referred to as "tiles."

Converting between grid locations and the screen's pixel coordinates are reasonably straight-forward.

Another type of 2D game uses "Isometric Tiles." Here, we can fake a 3D view with 2D graphics. We do that by tilting the grid 45 degrees. Each tile then becomes a diamond.

Unfortunately the math to go from pixels to grid locations is no longer straight forward.

CHAPTER 1

Equations For Tiles To Pixels

Equations to go from tile coordinates to screen coordinates.

1.1 Variable definitions

Given:

- tilewidth = width of each tile in pixels
- tileheight = height of each tile in pixels
- tilex = x-coordinate of the tile, in tiles
- tiley = y-coordinate of the tile, in tiles
- width = width of the map, in tiles
- height = hieght of the map, in tiles

Result:

- screenx = x-coordinate of the screen in pixels
- screeny = y-coordinate of the screen in pixels

1.2 Equations

$$screenx = \frac{tilewidth \cdot tilex}{2} + \frac{height \cdot tilewidth}{2} - \frac{tiley \cdot tilewidth}{2}$$

$$screeny = \frac{(height - tiley - 1) \cdot tileheight}{2} + \frac{width \cdot tileheight}{2} - \frac{tilex \cdot tileheight}{2}$$

CHAPTER 2

Equations For Pixels To Tiles

Equations to go from screen pixel coordinates to tile coordinates.

This needs to work for any coordinate inside the diamond, not just the center.

2.1 Variable definitions

Given:

- screenx = x-coordinate of the screen in pixels
- screeny = y-coordinate of the screen in pixels
- tilewidth = width of each tile in pixels
- tileheight = height of each tile in pixels
- width = width of the map, in tiles
- height = hieght of the map, in tiles

Result:

- tilex = x-coordinate of the tile, in tiles
- tiley = y-coordinate of the tile, in tiles

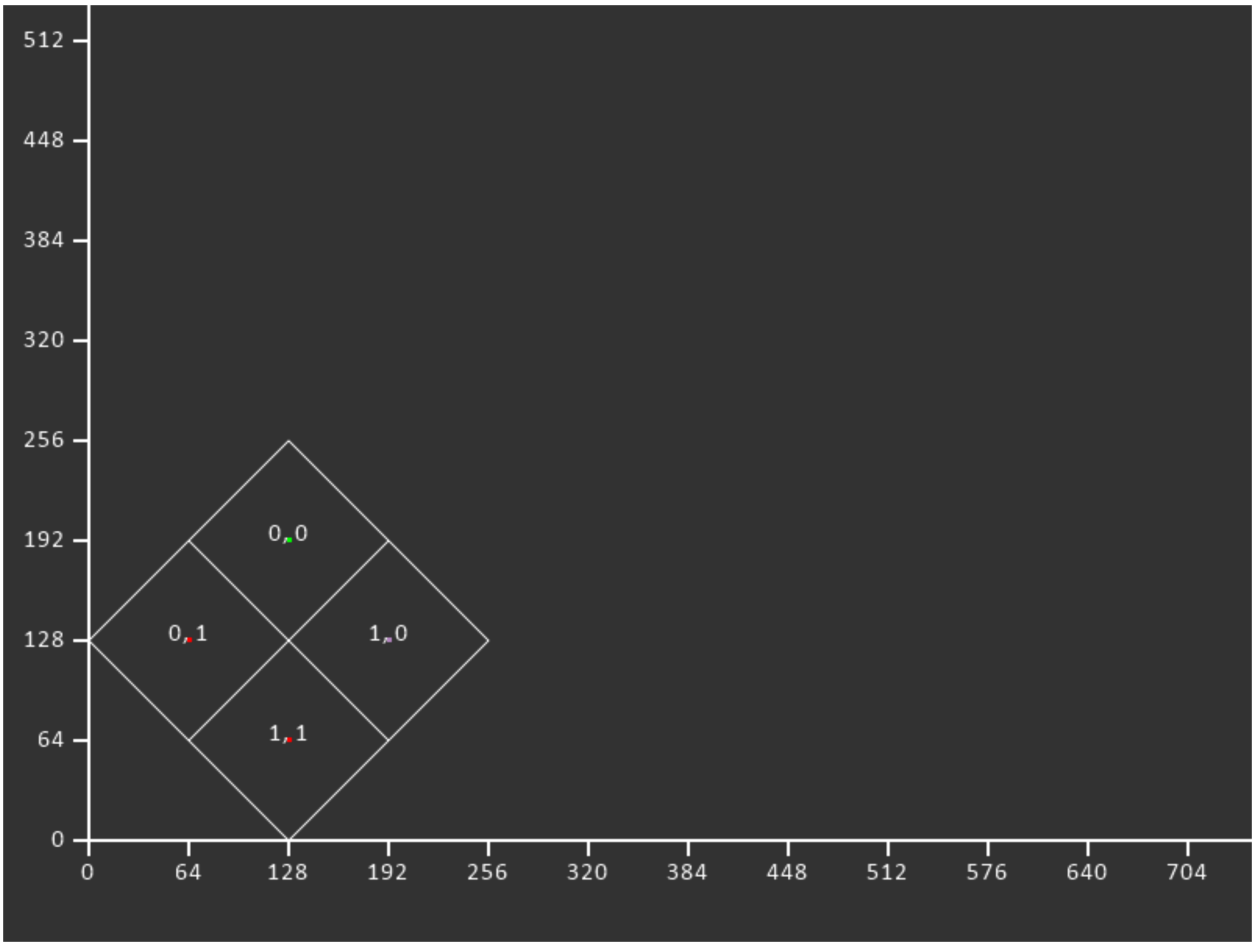
2.2 Equations

<Insert magic math stuff here.>

CHAPTER 3

Examples

3.1 2x2 Grid



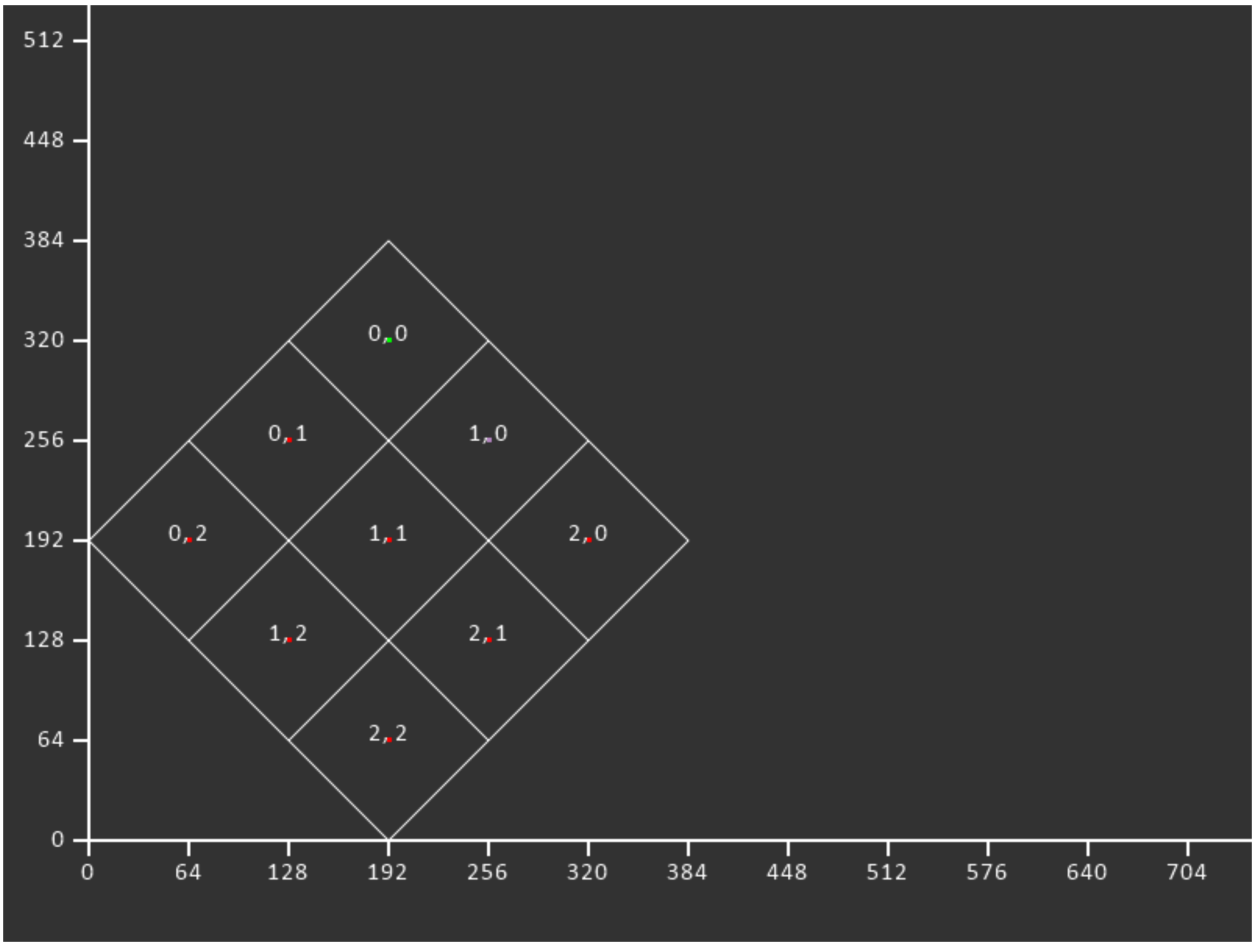
Listing 1: Parameters

```
MAP_WIDTH = 2
MAP_HEIGHT = 2
TILE_WIDTH = 128
TILE_HEIGHT = 128
```

Listing 2: Tile coordinates to screen coordinates (center of tile)

```
0, 0 => 128, 192
0, 1 =>  64, 128
1, 0 => 192, 128
1, 1 => 128,  64
```

3.2 3x3 Grid



Listing 3: Parameters

```
MAP_WIDTH = 3
MAP_HEIGHT = 3
TILE_WIDTH = 128
```

(continues on next page)

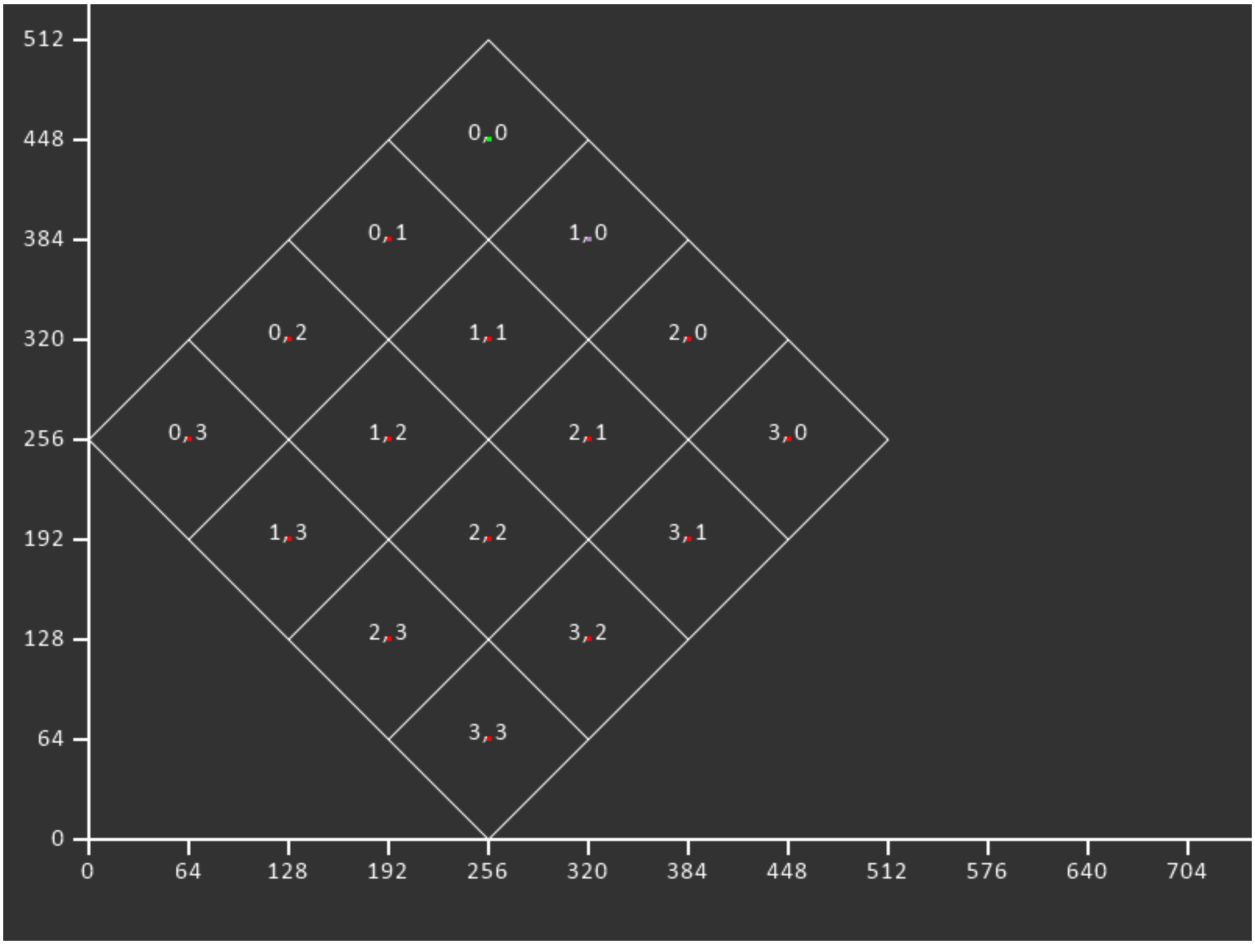
(continued from previous page)

```
TILE_HEIGHT = 128
```

Listing 4: Tile coordinates to screen coordinates (center of tile)

```
0, 0 => 192, 320
0, 1 => 128, 256
0, 2 =>  64, 192
1, 0 => 256, 256
1, 1 => 192, 192
1, 2 => 128, 128
2, 0 => 320, 192
2, 1 => 256, 128
2, 2 => 192,  64
```

3.3 4x4 Grid



Listing 5: Parameters

```
MAP_WIDTH = 4
MAP_HEIGHT = 4
```

(continues on next page)

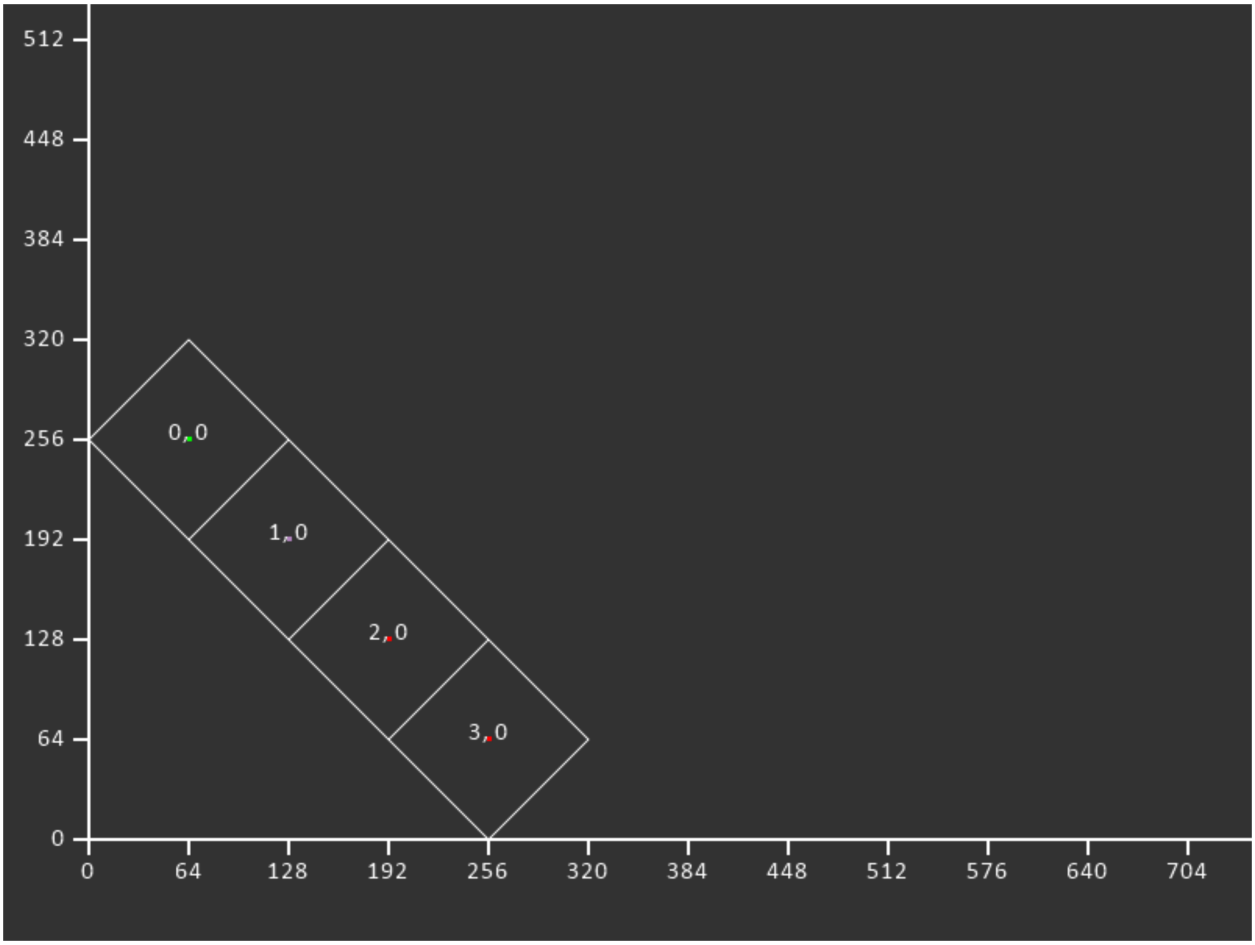
(continued from previous page)

```
TILE_WIDTH = 128
TILE_HEIGHT = 128
```

Listing 6: Tile coordinates to screen coordinates (center of tile)

```
0, 0 => 256, 448
0, 1 => 192, 384
0, 2 => 128, 320
0, 3 =>  64, 256
1, 0 => 320, 384
1, 1 => 256, 320
1, 2 => 192, 256
1, 3 => 128, 192
2, 0 => 384, 320
2, 1 => 320, 256
2, 2 => 256, 192
2, 3 => 192, 128
3, 0 => 448, 256
3, 1 => 384, 192
3, 2 => 320, 128
3, 3 => 256,  64
```

3.4 4x1 Grid



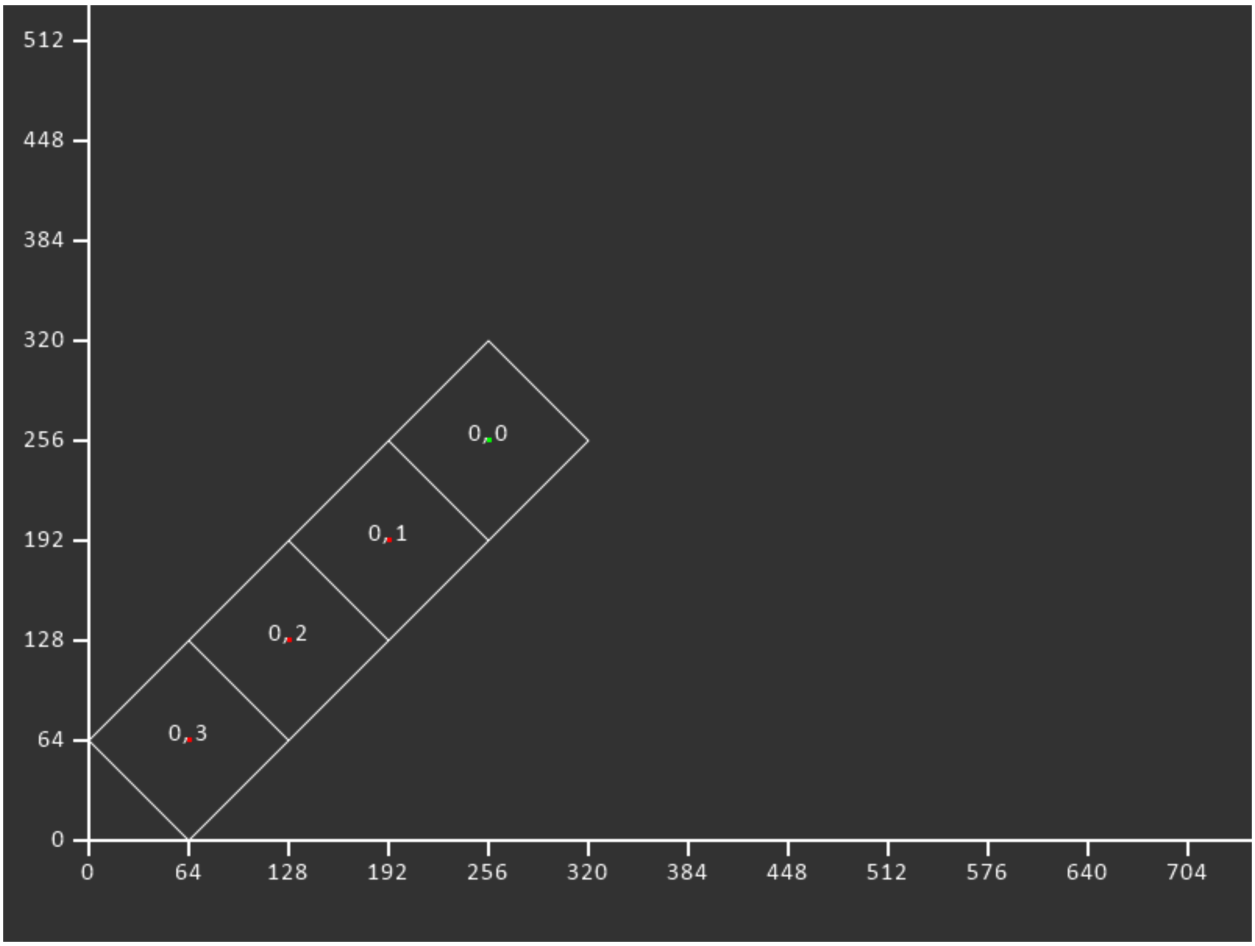
Listing 7: Parameters

```
MAP_WIDTH = 4
MAP_HEIGHT = 1
TILE_WIDTH = 128
TILE_HEIGHT = 128
```

Listing 8: Tile coordinates to screen coordinates (center of tile)

```
0, 0 =>  64, 256
1, 0 => 128, 192
2, 0 => 192, 128
3, 0 => 256,  64
```

3.5 1x4 Grid



Listing 9: Parameters

```
MAP_WIDTH = 1
MAP_HEIGHT = 4
TILE_WIDTH = 128
TILE_HEIGHT = 128
```

Listing 10: Tile coordinates to screen coordinates (center of tile)

```
0, 0 => 256, 256
0, 1 => 192, 192
0, 2 => 128, 128
```

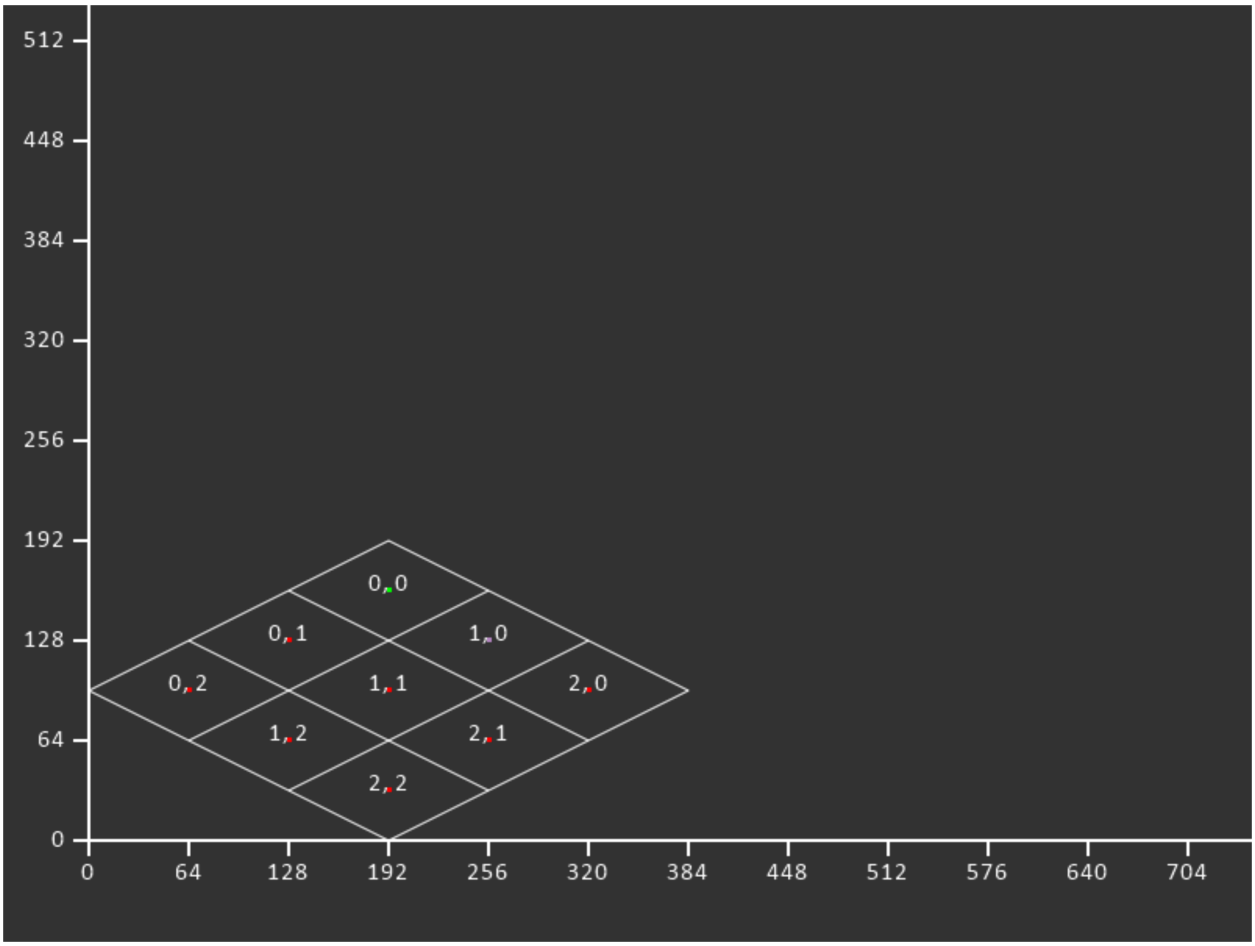
(continues on next page)

(continued from previous page)

```
0, 3 =>  64,  64
```

3.6 3x3 Squished Grid

The height and width don't have to equal each other. In fact, they often don't. Here's an example where they are different.



Listing 11: Parameters

```
MAP_WIDTH = 3
MAP_HEIGHT = 3
TILE_WIDTH = 128
TILE_HEIGHT = 64
```

Listing 12: Tile coordinates to screen coordinates (center of tile)

```
0, 0 => 192, 160
0, 1 => 128, 128
0, 2 =>  64,  96
1, 0 => 256, 128
1, 1 => 192,  96
```

(continues on next page)

(continued from previous page)

```
1, 2 => 128,  64
2, 0 => 320,  96
2, 1 => 256,  64
2, 2 => 192,  32
```

CHAPTER 4

Code Example

Listing 1: isometric_example.py

```
"""
Example code showing Isometric Grid coordinates
"""

import arcade
import os

SCREEN_WIDTH = 700
SCREEN_HEIGHT = 700

MAP_WIDTH = 5
MAP_HEIGHT = 4
TILE_WIDTH = 128
TILE_HEIGHT = 128


def get_screen_coordinates(tile_x, tile_y, width, height, tilewidth, tileheight):
    screen_x = tilewidth * tile_x // 2 + height * tilewidth // 2 - tile_y * tilewidth // 2
    screen_y = (height - tile_y - 1) * tileheight // 2 + width * tileheight // 2 - tile_x * tileheight // 2
    return screen_x, screen_y


class MyGame(arcade.Window):
    """ Main application class. """

    def __init__(self, width, height):
        super().__init__(width, height)

        self.axis_shape_list = None
        self.isometric_grid_shape_list = None
```

(continues on next page)

(continued from previous page)

```
31
32      def setup(self):
33          """ Set up the game and initialize the variables. """
34
35          # Set the background color
36          arcade.set_background_color((50, 50, 50))
37
38          self.axis_shape_list = arcade.ShapeElementList()
39
40          # Axis
41          start_x = 0
42          start_y = 0
43          end_x = 0
44          end_y = SCREEN_HEIGHT
45          line = arcade.create_line(start_x, start_y, end_x, end_y, arcade.color.WHITE,␣
    ↪2)
46          self.axis_shape_list.append(line)
47
48          # Axis
49          start_x = 0
50          start_y = 0
51          end_x = SCREEN_WIDTH
52          end_y = 0
53          line = arcade.create_line(start_x, start_y, end_x, end_y, arcade.color.WHITE,␣
    ↪2)
54          self.axis_shape_list.append(line)
55
56          # x Tic Marks
57          for x in range(0, SCREEN_WIDTH, 64):
58              start_y = -10
59              end_y = 0
60              line = arcade.create_line(x, start_y, x, end_y, arcade.color.WHITE, 2)
61              self.axis_shape_list.append(line)
62
63          # y Tic Marks
64          for y in range(0, SCREEN_HEIGHT, 64):
65              start_x = -10
66              end_x = 0
67
68              line = arcade.create_line(start_x, y, end_x, y, arcade.color.WHITE, 2)
69              self.axis_shape_list.append(line)
70
71          tilewidth = TILE_WIDTH
72          tileheight = TILE_HEIGHT
73          width = MAP_WIDTH
74          height = MAP_HEIGHT
75
76          # Gridlines 1
77          for tile_row in range(-1, height):
78              tile_x = 0
79              start_x, start_y = get_screen_coordinates(tile_x, tile_row, width, height,
    ↪ tilewidth, tileheight)
80              tile_x = width - 1
81              end_x, end_y = get_screen_coordinates(tile_x, tile_row, width, height,␣
    ↪tilewidth, tileheight)
82
83              start_x -= tilewidth // 2
```

(continues on next page)

(continued from previous page)

```
            end_y -= tileheight // 2

            line = arcade.create_line(start_x, start_y, end_x, end_y, arcade.color.
↪WHITE)
            self.axis_shape_list.append(line)

        # Gridlines 2
        for tile_column in range(-1, width):
            tile_y = 0
            start_x, start_y = get_screen_coordinates(tile_column, tile_y, width,␣
↪height, tilewidth, tileheight)
            tile_y = height - 1
            end_x, end_y = get_screen_coordinates(tile_column, tile_y, width, height,␣
↪tilewidth, tileheight)

            start_x += tilewidth // 2
            end_y -= tileheight // 2

            line = arcade.create_line(start_x, start_y, end_x, end_y, arcade.color.
↪WHITE)
            self.axis_shape_list.append(line)

        for tile_x in range(width):
            for tile_y in range(height):
                screen_x, screen_y = get_screen_coordinates(tile_x, tile_y, width,␣
↪height, tilewidth, tileheight)
                point_width = 3
                point_height = 3
                point = arcade.create_rectangle_filled(screen_x, screen_y, point_
↪width, point_height, arcade.color.LIGHT_CORNFLOWER_BLUE, 3)
                self.axis_shape_list.append(point)
                print(f"{tile_x}, {tile_y} => {screen_x:3}, {screen_y:3}")


    def on_draw(self):
        """
        Render the screen.
        """

        # This command has to happen before we start drawing
        arcade.start_render()


        self.axis_shape_list.draw()

        # x Labels
        for x in range(0, SCREEN_WIDTH, 64):
            text_y = -25
            arcade.draw_text(f"{x}", x, text_y, arcade.color.WHITE, 12, width=200,␣
↪align="center",
                             anchor_x="center")

        # y Labels
        for y in range(0, SCREEN_HEIGHT, 64):
            text_x = -50
            arcade.draw_text(f"{y}", text_x, y - 4, arcade.color.WHITE, 12, width=70,␣
↪align="right",
```

(continues on next page)

(continued from previous page)

```
                          anchor_x="center")


        tilewidth = TILE_WIDTH
        tileheight = TILE_HEIGHT
        width = MAP_WIDTH
        height = MAP_HEIGHT

        for tile_x in range(width):
            for tile_y in range(height):
                screen_x, screen_y = get_screen_coordinates(tile_x, tile_y,
                                                            width, height,
                                                            tilewidth, tileheight)
                arcade.draw_text(f"{tile_x}, {tile_y}",
                                 screen_x, screen_y + 6,
                                 arcade.color.WHITE, 12,
                                 width=200, align="center", anchor_x="center")

    def update(self, delta_time):
        view_left = -50
        view_bottom = -50
        arcade.set_viewport(view_left,
                            SCREEN_WIDTH + view_left,
                            view_bottom,
                            SCREEN_HEIGHT + view_bottom)

    def on_mouse_press(self, x: float, y: float):
        screen_x = x + self.view_left
        screen_y = y + self.view_bottom

        grid_x = screen_x // TILE_WIDTH
        grid_y = screen_y // TILE_HEIGHT
        point_x = (screen_x % TILE_WIDTH) - (TILE_WIDTH / 2)
        point_y = (screen_y % TILE_HEIGHT) - (TILE_HEIGHT / 2)


        # print(f"({screen_x}, {screen_y}) -> ({map_x:.2}, {map_y:.2})")


def main():
    """ Main method """
    window = MyGame(SCREEN_WIDTH, SCREEN_HEIGHT)
    window.setup()
    arcade.run()


if __name__ == "__main__":
    main()
```