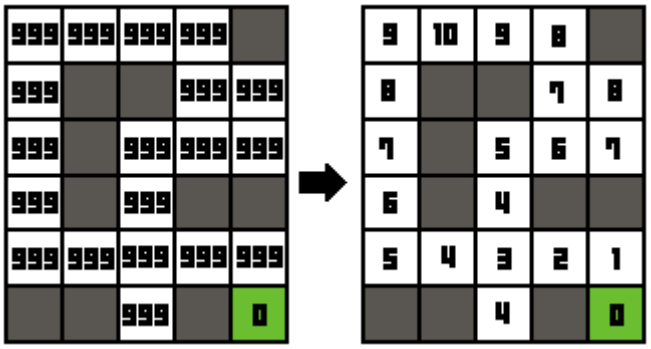
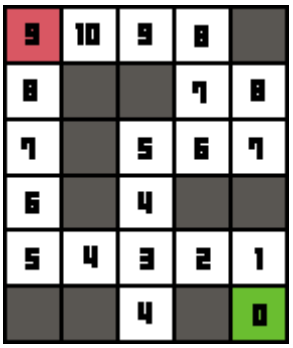
Introduction

This pathfinding will be an implementation of Dijkstra Maps used in the roguelike Brogue. The gist of it is that we will build an array with the cost of each tile. We do this by setting a goal tile to the cost of zero and all the other tiles to some high number. Then each tile check their neighbours cost, if the tile has a cost that is at least 2 higher than the cheapest neighbour we change the cost to 1 above the neighbouring tile.
We keep doing this until no more changes have been made. In the end we will have something like this:



Then the entity simply follows the cheapest tiles until it reaches the goal.



Limitations

The system requires two variable for each tile; one to store the actual cost and one to store a temporary cost while we do the calculation. This can quickly get out of hand for larger maps. In this case we will use a 10x10 map and 200 variables to store the Dijkstra map.
If you want more than one entity pathing at the same time you would need a dijkstra map for each of them, quickly resulting in an unreasonable amount of variables being used.

This system wouldn't work for a real time action game with ten zombies that stalks the player on each map. But is instead a better fit for something like a Fire Emblem or Final Fantasy Tactics combat where only one unit will navigate the map at time.

Terrain ID will also be used to track which tiles are passable or not. So if you place an impassable rock from the upper layer on a passable grass tile the system will still consider

that tile walkable. Most likely ending up with everything freezing while the entity is trying to walk into a rock.

Implementing Common Events

First let's set up our Terrain ID and Tileset. So go into the database and go to the "Terrain" tab. Create a new terrain id and name it "impassable", you can ignore all the settings.

Next go to the "Tileset" tab and then select the tileset you want use. Go through the lower layer and set all the impassable tiles to the newly created impassable terrain id.

Here is all the switches we will be using:

```
0001: initializedGame
0002: startRandomlyWalking
```

And here is all the variables:

```
0002: temp 0
0003: temp 1
0004: temp 2
0005: temp 3
0006: temp 4
0007: temp 5
0008: temp 6
0009: temp 7
0010: temp 8
0011: temp 9
0012: tempX
0012: tempY

0021: return
0025: map offset X
0026: map offset X
0027: impassable cost
0028: impassable ID
0032: goal X
0033: goal Y
0034: entity X
0035: entity X

0501:  tile 0 0
0502:  tile 0 1
0503:  tile 0 2
0504:  tile 0 3
0505:  tile 0 4
0506:  tile 0 5
0507:  tile 0 6
0508:  tile 0 7
0509:  tile 0 8
0510:  tile 0 9
0511:  tile 1 0
0512:  tile 1 1
0513:  tile 1 2
0514:  tile 1 3
0515:  tile 1 4
0516:  tile 1 5
```

```
0517:  tile 1 6
0518:  tile 1 7
0519:  tile 1 8
0520:  tile 1 9

0521:  tile 2 0
0522:  tile 2 1
0523:  tile 2 2
0524:  tile 2 3
0525:  tile 2 4
0526:  tile 2 5
0527:  tile 2 6
0528:  tile 2 7
0529:  tile 2 8
0530:  tile 2 9
0531:  tile 3 0
0532:  tile 3 1
0533:  tile 3 2
0534:  tile 3 3
0535:  tile 3 4
0536:  tile 3 5
0537:  tile 3 6
0538:  tile 3 7
0539:  tile 3 8
0540:  tile 3 9

0541:  tile 4 0
0542:  tile 4 1
0543:  tile 4 2
0544:  tile 4 3
0545:  tile 4 4
0546:  tile 4 5
0547:  tile 4 6
0548:  tile 4 7
0549:  tile 4 8
0550:  tile 4 9
0551:  tile 5 0
0552:  tile 5 1
0553:  tile 5 2
0554:  tile 5 3
0555:  tile 5 4
0556:  tile 5 5
0557:  tile 5 6
0558:  tile 5 7
0559:  tile 5 8
0560:  tile 5 9

0561:  tile 6 0
0562:  tile 6 1
0563:  tile 6 2
0564:  tile 6 3
0565:  tile 6 4
0566:  tile 6 5
0567:  tile 6 6
0568:  tile 6 7
0569:  tile 6 8
0570:  tile 6 9
0571:  tile 7 0
0572:  tile 7 1
0573:  tile 7 2
0574:  tile 7 3
0575:  tile 7 4
0576:  tile 7 5
0577:  tile 7 6
0578:  tile 7 7
```

0579: tile 7 8
0580: tile 7 9

0581: tile 8 0
0582: tile 8 1
0583: tile 8 2
0584: tile 8 3
0585: tile 8 4
0586: tile 8 5
0587: tile 8 6
0588: tile 8 7
0589: tile 8 8
0590: tile 8 9
0591: tile 9 0
0592: tile 9 1
0593: tile 9 2
0594: tile 9 3
0595: tile 9 4
0596: tile 9 5
0597: tile 9 6
0598: tile 9 7
0599: tile 9 8
0600: tile 9 9

0601: temp 0 0
0602: temp 0 1
0603: temp 0 2
0604: temp 0 3
0605: temp 0 4
0606: temp 0 5
0607: temp 0 6
0608: temp 0 7
0609: temp 0 8
0610: temp 0 9
0611: temp 1 0
0612: temp 1 1
0613: temp 1 2
0614: temp 1 3
0615: temp 1 4
0616: temp 1 5
0617: temp 1 6
0618: temp 1 7
0619: temp 1 8
0620: temp 1 9

0621: temp 2 0
0622: temp 2 1
0623: temp 2 2
0624: temp 2 3
0625: temp 2 4
0626: temp 2 5
0627: temp 2 6
0628: temp 2 7
0629: temp 2 8
0630: temp 2 9
0631: temp 3 0
0632: temp 3 1
0633: temp 3 2
0634: temp 3 3
0635: temp 3 4
0636: temp 3 5
0637: temp 3 6
0638: temp 3 7
0639: temp 3 8
0640: temp 3 9

0641: temp 4 0
0642: temp 4 1
0643: temp 4 2
0644: temp 4 3
0645: temp 4 4
0646: temp 4 5
0647: temp 4 6
0648: temp 4 7
0649: temp 4 8
0650: temp 4 9
0651: temp 5 0
0652: temp 5 1
0653: temp 5 2
0654: temp 5 3
0655: temp 5 4
0656: temp 5 5
0657: temp 5 6
0658: temp 5 7
0659: temp 5 8
0660: temp 5 9

0661: temp 6 0
0662: temp 6 1
0663: temp 6 2
0664: temp 6 3
0665: temp 6 4
0666: temp 6 5
0667: temp 6 6
0668: temp 6 7
0669: temp 6 8
0670: temp 6 9
0671: temp 7 0
0672: temp 7 1
0673: temp 7 2
0674: temp 7 3
0675: temp 7 4
0676: temp 7 5
0677: temp 7 6
0678: temp 7 7
0679: temp 7 8
0680: temp 7 9

0681: temp 8 0
0682: temp 8 1
0683: temp 8 2
0684: temp 8 3
0685: temp 8 4
0686: temp 8 5
0687: temp 8 6
0688: temp 8 7
0689: temp 8 8
0690: temp 8 9
0691: temp 9 0
0692: temp 9 1
0693: temp 9 2
0694: temp 9 3
0695: temp 9 4
0696: temp 9 5
0697: temp 9 6
0698: temp 9 7
0699: temp 9 8
0700: temp 9 9

Some notes about the variables. All the temp variables is just to temporarily store the results of calculations and are never used outside the event. Goal X and Goal Y is used to store the position our entity wants to go to and entity X and entity Y stores it's current position.
There are 100 tile variables and 100 temp tile variables because of our map size. If you want a larger or smaller map you will need to add or remove the tile and temp variables.
The other variables are explained as they come up in the code.

Now it's time to get started with implementing the code. We'll be using three common events; one to just set up some variables that will be use, another to build the Dijkstra Map and one last to find the cheapest neighbouring tile while we're pathing.

We'll start with the Set Up event, so go to the "Common Event" tab. Create an event called "Initialize Game" and add the following code:

```
@> Comment:
 :          : ===
 :          : Initializes the variables used for building the Dijkstra Map
 :          : ===
@> Control Variables: [0027:impassable cost] = 9999999
@> Comment: Should be the same as the terrain id for impassable tiles
@> Control Variables: [0028:impassable ID] = 2
@> Comment:
 :          : The Map Offset variables is used to calculate where our map
 :          : starts. Should be the top left corner of the pathfinding
 :          : area.
@> Control Variables: [0025:map offset X] = 5
@> Control Variables: [0026:map offset Y] = 2
@> Control Switches: [0001:initializedGame] = ON
```

This piece of code should be pretty straightforward but I'll do a quick run down of it. The 'impassable cost' variable will be the cost of our impassable tiles and should just be some high number. 'Impassable ID' is the index of our impassable terrain, in this case we use index 2. The map offset is the beginning of our pathfinding area. Since we'll only be using a 10x10 map in this demonstration we'll have to offset it to where it begins.
Finally we set a switch to ON so we can let the Build Dijkstra Map event know that we already initialized and that there is no need to do it again.

Next up it's time to build the map. So create a new event and call it "Build Dijkstra Map". I'll split this code up in several parts as it is a bit longer.

```
@> Comment:
 :          : ===
 :          : Builds the Dijkstra map used for pathfinding
 :          : ===
@> Comment:
@> Comment: Initialize the variables used for the dijkstra map if
 :          : haven't already
@> Conditional Branch: Switch [0001:initializedGame] is OFF
```

```
 @> Call Event: [Initialize Game]
 @>
: Branch End
```

First we'll initialize the game if we haven't already.

```
@> Comment: Find all the impassable tiles and set them to a ridiculous
 :             : high number
@> Comment: temp 0 is used to track the current variable index of the
 :             : tiles we're on in the loop
@> Control Variables: [0002:temp 0] = 501
@> Comment: Set the temp Y to the map offset Y  and temp X to 0
@> Control Variables: [0012:tempX] = 0
@> Control Variables: [0013:tempY] = Variable [0026]
@> Loop
 @> Loop
  @> Comment: Offset the X position with map offset
  @> Control Variables: [0004:temp 2] = Variable [0012]
  @> Control Variables: [0004:temp 2] += Variable [0025]
  @> Comment: Check if tile is impassable
  @> Get Terrain ID: [0003:temp 1], Variable [0004][0013]
  @> Conditional Branch: Variable [0003:temp 1] == Variable [0028:impassable ID]
   @> Comment: impassable tile found, set the variable of the index stored
    :             : in temp 0 to impassable cost
   @> Control Variables: Variable [0002] = Variable [0027]
   @>
  : Branch End
  @> Comment: Continue loop through x
  @> Control Variables: [0012:tempX] += 1
  @> Control Variables: [0002:temp 0] += 1
  @> Conditional Branch: Variable [0012:tempX] > 9
   @> Control Variables: [0012:tempX] = 0
   @> Break Loop
   @>
  : Branch End
  @>
 : Repeat Above
 @> Control Variables: [0013:tempY] += 1
 @> Comment: Check if we reached the end of our tiles. In this case
  :             : variable index 600.
 @> Conditional Branch: Variable [0002:temp 0] > 600
  @> Break Loop
  @>
 : Branch End
 @>
: Repeat Above
```

Here we look for all the impassable tiles and set them to the cost stored away in the impassable cost variable. Keep in mind that you will need to offset the temp X variable with the map offset variable before checking terrain id.

```
@> Comment: Find and set the goal tile to a cost of 0 and set the
 :            : rest to a cost of 1000
@> Control Variables: [0002:temp 0] = 501
@> Comment: calculate the index of the goal tile and store it in temp 1
@> Control Variables: [0003:temp 1] = Variable [0033]
@> Control Variables: [0003:temp 1] *= 10
@> Control Variables: [0003:temp 1] += Variable [0032]
@> Control Variables: [0003:temp 1] += Variable [0002]
@> Loop
  @> Conditional Branch: Variable [0002:temp 0] == Variable [0003:temp 1]
    @> Control Variables: Variable [0002] = 0
    @>
  : Else
    @> Control Variables: [0004:temp 2] = Variable ID [V[0002]]
    @> Conditional Branch: Variable [0004:temp 2]  != Variable [0027:impassable cost]
      @> Control Variables: Variable [0002] = 1000
      @>
    : Branch End
    @>
  : Branch End
  @> Control Variables: [0002:temp 0] += 1
  @> Conditional Branch: Variable [0002:temp 0] > 600
    @> Break Loop
    @>
  : Branch End
  @>
 : Repeat Above
```

We loop through all our tiles once again and set the goal tile to a cost of 0 and the rest that aren't impassable to a cost of 1000.
To calculate the index of the goal tile we use the formula: (goal Y * map height + goal X) + an offset to the beginning of the tile variables

```
@> Comment: Set up temp tiles cost
 :            : temp 0 is used to store the current index of the tile we're
 :            : on in the loop.
 :            : temp 1 is used to store the current index of the temp tile
@> Control Variables: [0002:temp 0] = 501
@> Control Variables: [0003:temp 1] = 601
@> Loop
  @> Control Variables: Variable [0003] = Variable ID [V[0002]]
  @> Control Variables: [0002:temp 0] += 1
  @> Control Variables: [0003:temp 1] += 1
  @> Conditional Branch: Variable [0002:temp 0] > 600
    @> Break Loop
    @>
  : Branch End
  @>
 : Repeat Above
```

Here we copy the current cost of all our tile variables into the temp tile variables.

```
@> Comment: Build the Dijkstra Map
@> Label: 1
@> Comment:
 :              : temp 0 is used to track the current tile index
 :              : temp 1 is used to track if any changes was made in the map
@> Control Variables: [0002:temp 0] = 501
@> Control Variables: [0003:temp 1] = 0
@> Loop
  @> Comment: Check if the tile isn't impassable
  @> Control Variables: [0004:temp 2] = Variable ID [V[0002]]
  @> Conditional Branch: Variable [0004:temp 2]  != Variable [0027:impassable cost]
    @> Comment:
     :              : Get the cost from all the neighbours. If we don't have a
     :              : neighbour set the cost to impassable
    @> Comment: Get top
    @> Control Variables: [0011:temp 9] = Variable [0002]
    @> Control Variables: [0011:temp 9] -= 10
    @> Conditional Branch: Variable [0011:temp 9] >= 501
      @> Control Variables: [0004:temp 2] = Variable ID [V[0011]]
      @>
    : Else
      @> Control Variables: [0004:temp 2] = Variable [0027]
      @>
    : Branch End
    @> Comment: Get bottom
    @> Control Variables: [0011:temp 9] = Variable [0002]
    @> Control Variables: [0011:temp 9] += 10
    @> Conditional Branch: Variable [0011:temp 9] <= 600
      @> Control Variables: [0005:temp 3] = Variable ID [V[0011]]
      @>
    : Else
      @> Control Variables: [0005:temp 3] = Variable [0027]
      @>
    : Branch End
    @> Comment: Get right
    @> Control Variables: [0011:temp 9] = Variable [0002]
    @> Control Variables: [0011:temp 9] %= 10
    @> Conditional Branch: Variable [0011:temp 9] == 0
      @> Control Variables: [0006:temp 4] = Variable [0027]
      @>
    : Else
      @> Control Variables: [0011:temp 9] = Variable [0002]
      @> Control Variables: [0011:temp 9] += 1
      @> Control Variables: [0006:temp 4] = Variable ID [V[0011]]
      @>
    : Branch End
    @> Comment: Get left
    @> Control Variables: [0011:temp 9] = Variable [0002]
    @> Control Variables: [0011:temp 9] -= 1
    @> Control Variables: [0010:temp 8] = Variable [0011]
    @> Control Variables: [0010:temp 8] %= 10
    @> Conditional Branch: Variable [0010:temp 8] == 0
      @> Control Variables: [0007:temp 5] = Variable [0027]
      @>
    : Else
      @> Control Variables: [0007:temp 5] = Variable ID [V[0011]]
      @>
    : Branch End
```

Here we start building the Dijkstra map. It's a lot going on so let's break it down. First off we set the label 1 before starting this loop so that we later can jump back and start the loop

over. After that we reset the temp 0 variable that is used to store the current tile index we're on. Temp 1 is used to store whatever we changed the cost of a tile.
After that we begin the loop. If the current tile isn't an impassable tile we begin to check and store away the cost of each neighbouring tile. If we don't find a neighbouring tile we store away the cost for an impassable tile. The cost of the top tile is stored away in temp 2, down in temp 3, right in temp 4 and left in temp 5.
Note that all the code to get neighbours are currently hard coded for a 10x10 map and would need to be changed if you want to change the map size.

```
@> Comment:
  :            : Check which neighbour is cheapest. If it's cheaper by two
  :            : store away the new cost in the temp tiles
  @> Comment: Check if up is lowest
  @> Conditional Branch: Variable [0004:temp 2] <= Variable [0005:temp 3]
   @> Conditional Branch: Variable [0004:temp 2] <= Variable [0006:temp 4]
    @> Conditional Branch: Variable [0004:temp 2] <= Variable [0007:temp 5]
     @> Control Variables: [0008:temp 6] = Variable ID [V[0002]]
     @> Control Variables: [0008:temp 6] -= Variable [0004]
     @> Conditional Branch: Variable [0008:temp 6] >= 2
      @> Comment: Set temp 1 to 1 to mark that we did a change
      @> Control Variables: [0003:temp 1] = 1
      @> Control Variables: [0008:temp 6] = Variable [0002]
      @> Control Variables: [0008:temp 6] += 100
      @> Control Variables: Variable [0008] = Variable [0004]
      @> Control Variables: Variable [0008] += 1
      @>
     : Branch End
    @>
   : Branch End
  @>
 : Branch End
@>
: Branch End
@> Comment: Check if down is lowest
@> Conditional Branch: Variable [0005:temp 3] <= Variable [0004:temp 2]
 @> Conditional Branch: Variable [0005:temp 3] <= Variable [0006:temp 4]
  @> Conditional Branch: Variable [0005:temp 3] <= Variable [0007:temp 5]
   @> Control Variables: [0008:temp 6] = Variable ID [V[0002]]
   @> Control Variables: [0008:temp 6] -= Variable [0005]
   @> Conditional Branch: Variable [0008:temp 6] >= 2
    @> Comment: Set temp 1 to 1 to mark that we did a change
    @> Control Variables: [0003:temp 1] = 1
    @> Control Variables: [0008:temp 6] = Variable [0002]
    @> Control Variables: [0008:temp 6] += 100
    @> Control Variables: Variable [0008] = Variable [0005]
    @> Control Variables: Variable [0008] += 1
    @>
   : Branch End
  @>
 : Branch End
@>
: Branch End
@> Comment: Check if right is lowest
@> Conditional Branch: Variable [0006:temp 4] <= Variable [0004:temp 2]
 @> Conditional Branch: Variable [0006:temp 4] <= Variable [0005:temp 3]
  @> Conditional Branch: Variable [0006:temp 4] <= Variable [0007:temp 5]
   @> Control Variables: [0008:temp 6] = Variable ID [V[0002]]
   @> Control Variables: [0008:temp 6] -= Variable [0006]
```

```
        @> Conditional Branch: Variable [0008:temp 6] >= 2
          @> Comment: Set temp 1 to 1 to mark that we did a change
          @> Control Variables: [0003:temp 1] = 1
          @> Control Variables: [0008:temp 6] = Variable [0002]
          @> Control Variables: [0008:temp 6] += 100
          @> Control Variables: Variable [0008] = Variable [0006]
          @> Control Variables: Variable [0008] += 1
          @>
         : Branch End
        @>
       : Branch End
      @>
     : Branch End
    @>
   : Branch End
  @> Comment: Check if left is lowest
  @> Conditional Branch: Variable [0007:temp 5] <= Variable [0004:temp 2]
    @> Conditional Branch: Variable [0007:temp 5] <= Variable [0005:temp 3]
      @> Conditional Branch: Variable [0007:temp 5] <= Variable [0006:temp 4]
        @> Control Variables: [0008:temp 6] = Variable ID [V[0002]]
        @> Control Variables: [0008:temp 6] -= Variable [0007]
        @> Conditional Branch: Variable [0008:temp 6] >= 2
          @> Comment: Set temp 1 to 1 to mark that we did a change
          @> Control Variables: [0003:temp 1] = 1
          @> Control Variables: [0008:temp 6] = Variable [0002]
          @> Control Variables: [0008:temp 6] += 100
          @> Control Variables: Variable [0008] = Variable [0007]
          @> Control Variables: Variable [0008] += 1
          @>
         : Branch End
        @>
       : Branch End
      @>
     : Branch End
    @>
   : Branch End
  @>
 : Branch End
```

Next up we need to compare the cost of all the neighbours to find the cheapest one. When we found one we check if it's cheaper by at least 2. If it is we set temp 1 to one to mark that we made a change to the map costs. After that we calculate the new cost (that should be one above the neighbour cost) and store it in the temp tile.
To get the correct temp tile index we have to offset temp 0 variable by 100 because our map is 100 tiles big. If you had a larger or smaller map you would need to change the offset to your map widths * map height.

```
@> Control Variables: [0002:temp 0] += 1
  @> Conditional Branch: Variable [0002:temp 0] > 600
    @> Comment:
     :            : If we have made changes to the map then repeat the
     :            : loop
    @> Conditional Branch: Variable [0003:temp 1]  != 0
      @> Comment: Apply all the changes from temp tiles
```

```
    @> Control Variables: [0002:temp 0] = 501
    @> Control Variables: [0003:temp 1] = 601
    @> Loop
     @> Control Variables: Variable [0002] = Variable ID [V[0003]]
     @> Control Variables: [0002:temp 0] += 1
     @> Control Variables: [0003:temp 1] += 1
     @> Conditional Branch: Variable [0002:temp 0] > 600
      @> Comment: Start over again
      @> Jump to Label: 1
      @>
     : Branch End
     @>
    : Repeat Above
    @>
   : Else
    @> Comment: No changes was made, stop building map
    @> Break Loop
    @>
   : Branch End
  @>
 : Branch End
@>
: Repeat Above
```

If we finished looping through all the tiles we check if we made a change to the map by checking if temp 1 is not equal to zero.
In case changes have been made we loop through all the temp tiles and apply the cost to the tiles variable and jump back to label 1 and start the whole thing over again.
If no changes was made we simply break out of the loop and our Dijkstra map is finished.

Next up we need one last common event to retrieve the cheapest neighbour while navigating the map. So create a new common event and call it “Cheapest Neighbour”.

```
@> Comment:
:          : ===
:          : Returns the direction of the cheapest neighbour tile
:          : Return  - the direction of cheapest neighbour
@> Comment:
:          : 1 = down
:          : 2 = left
:          : 3 = right
@> Comment:
:          : 4 - up
:          : 5 - none
:          : ===
@> Comment: Calculate current tile index from entity X and entity Y
@> Control Variables: [0002:temp 0] = Variable [0035]
@> Control Variables: [0002:temp 0] *= 10
@> Control Variables: [0002:temp 0] += Variable [0034]
@> Control Variables: [0002:temp 0] += 501
```

We’ll start by calculating our current tile index using the same formula we used to calculate the goal index before but this time we use entity X and entity Y instead.

```
@> Comment: Get the cost of each neighbour
@> Comment: Get top
@> Control Variables: [0003:temp 1] = Variable [0002]
@> Control Variables: [0003:temp 1] -= 10
@> Conditional Branch: Variable [0003:temp 1] >= 501
  @> Control Variables: [0004:temp 2] = Variable ID [V[0003]]
  @>
 : Else
  @> Control Variables: [0004:temp 2] = Variable [0027]
  @>
 : Branch End
@> Comment: Get bottom
@> Control Variables: [0003:temp 1] = Variable [0002]
@> Control Variables: [0003:temp 1] += 10
@> Conditional Branch: Variable [0003:temp 1] <= 600
  @> Control Variables: [0005:temp 3] = Variable ID [V[0003]]
  @>
 : Else
  @> Control Variables: [0005:temp 3] = Variable [0027]
  @>
 : Branch End
@> Comment: Get right
@> Control Variables: [0003:temp 1] = Variable [0002]
@> Control Variables: [0003:temp 1] %= 10
@> Conditional Branch: Variable [0003:temp 1] == 0
  @> Control Variables: [0006:temp 4] = Variable [0027]
  @>
 : Else
  @> Control Variables: [0003:temp 1] = Variable [0002]
  @> Control Variables: [0003:temp 1] += 1
  @> Control Variables: [0006:temp 4] = Variable ID [V[0003]]
  @>
 : Branch End
@> Comment: Get left
@> Control Variables: [0003:temp 1] = Variable [0002]
@> Control Variables: [0003:temp 1] -= 1
@> Control Variables: [0011:temp 9] = Variable [0003]
@> Control Variables: [0011:temp 9] %= 10
@> Conditional Branch: Variable [0011:temp 9] == 0
  @> Control Variables: [0007:temp 5] = Variable [0027]
  @>
 : Else
  @> Control Variables: [0007:temp 5] = Variable ID [V[0003]]
  @>
 : Branch End
```

Here we get the cost of our neighbours and store away in temp variables. We check the neighbours in the same way we did in when we built the Dijkstra map. If we have no neighbour we get the cost for impassable tiles instead.

```
@> Comment:
 :          : Compare cost and return the cheapest direction
 :          : temp 9 is used to track if we found a direction
@> Control Variables: [0011:temp 9] = 0
@> Comment: If we are surrounded by impassable tiles return direction
 :          : none (5)
```

```
@> Conditional Branch: Variable [0004:temp 2] == Variable [0027:impassable cost]
 @> Conditional Branch: Variable [0005:temp 3] == Variable [0027:impassable cost]
  @> Conditional Branch: Variable [0006:temp 4] == Variable [0027:impassable cost]
   @> Conditional Branch: Variable [0007:temp 5] == Variable [0027:impassable cost]
    @> Control Variables: [0011:temp 9] = 1
    @> Control Variables: [0021:return] = 5
    @>
   : Branch End
   @>
  : Branch End
  @>
 : Branch End
 @>
: Branch End
```

Then we must compare all the neighbours to each other to find the cheapest one. We start off by checking if all the neighbours are equal to impassable tiles. In that case we set the return variable to 5.
Temp 9 is just use to track if we found the cheapest neighbour and used to skip all the other checks.

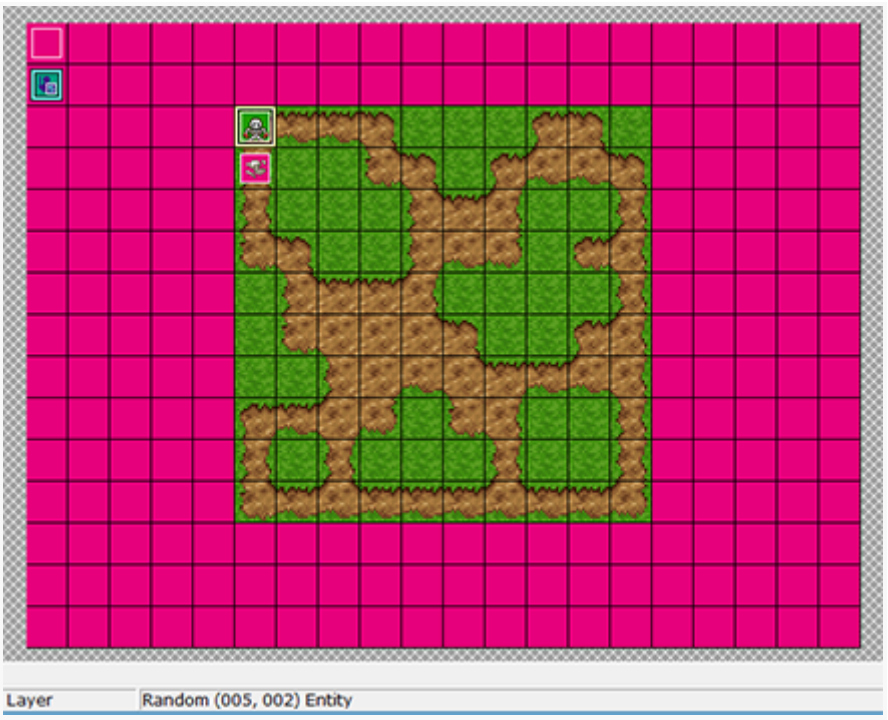
```
@> Comment: Check if top is cheapest
@> Conditional Branch: Variable [0011:temp 9] == 0
 @> Conditional Branch: Variable [0004:temp 2] <= Variable [0005:temp 3]
  @> Conditional Branch: Variable [0004:temp 2] <= Variable [0006:temp 4]
   @> Conditional Branch: Variable [0004:temp 2] <= Variable [0007:temp 5]
    @> Control Variables: [0011:temp 9] = 1
    @> Control Variables: [0021:return] = 4
    @>
   : Branch End
   @>
  : Branch End
  @>
 : Branch End
 @>
: Branch End
@> Comment: Check if bottom is cheapest
@> Conditional Branch: Variable [0011:temp 9] == 0
 @> Conditional Branch: Variable [0005:temp 3] <= Variable [0004:temp 2]
  @> Conditional Branch: Variable [0005:temp 3] <= Variable [0006:temp 4]
   @> Conditional Branch: Variable [0005:temp 3] <= Variable [0007:temp 5]
    @> Control Variables: [0011:temp 9] = 1
    @> Control Variables: [0021:return] = 1
    @>
   : Branch End
   @>
  : Branch End
  @>
 : Branch End
 @>
: Branch End
@> Comment: Check if right is cheapest
@> Conditional Branch: Variable [0011:temp 9] == 0
 @> Conditional Branch: Variable [0006:temp 4] <= Variable [0004:temp 2]
  @> Conditional Branch: Variable [0006:temp 4] <= Variable [0005:temp 3]
   @> Conditional Branch: Variable [0006:temp 4] <= Variable [0007:temp 5]
    @> Control Variables: [0011:temp 9] = 1
    @> Control Variables: [0021:return] = 3
```

```
      @>
     : Branch End
    @>
   : Branch End
  @>
 : Branch End
@>
: Branch End
@> Comment: Check if left is cheapest
@> Conditional Branch: Variable [0011:temp 9] == 0
 @> Conditional Branch: Variable [0007:temp 5] <= Variable [0004:temp 2]
  @> Conditional Branch: Variable [0007:temp 5] <= Variable [0005:temp 3]
   @> Conditional Branch: Variable [0007:temp 5] <= Variable [0006:temp 4]
    @> Control Variables: [0011:temp 9] = 1
    @> Control Variables: [0021:return] = 2
    @>
   : Branch End
  @>
 : Branch End
@>
: Branch End
```

We compare all the directions and set the return variable to the cheapest one. With that we got all the common events we need. Next up we move on the the specific code needed on the maps to get an entity to move around.

Implementing Map Specific Code

Create a new map and name it something cool. Keep the rest of the default setting. Start from x005 y002 and draw a 10x10 rectangle from there. Make some paths for the entity to walk around on.



Next up create a new event and name it “Entity”, select some cool looking character (I went for a skeleton). Place it on some passable tile in the map.
Create another event and name it “Goal Marker”, it will be showing the position the entity is pathing to. Set some fitting graphic to it and make sure it’s priority is set to be below characters. It doesn’t matter where you place the event.

It’s time to move on to coding again. Create a new event and name it “Game Loop” and make sure it’s trigger is Autorun. Add this piece of code to it:

```
@> Call Event: [Initialize Game]
@> Control Variables: [0034:entity X] = 0
@> Control Variables: [0035:entity Y] = 0
@> Control Switches: [0002:startRandomlyWalking] = ON
```

First we call the Initialize Game event because we want to use some of the variables it initializes. Then we need to initialize the entities position, I put mine in the top left corner so it’s position is 0, 0 if you put yours somewhere else make sure you got the right position for it. Last we set the switch to ON to indicate that we should start pathing.
Now create a second page in the event and set it’s trigger to Parallel Processing. Add the following code:

```
@> Comment: Pick a random position
@> Loop
  @> Control Variables: [0032:goal X] = Random No. (0...9)
  @> Control Variables: [0033:goal Y] = Random No. (0...9)
  @> Comment: Check if it's a valid position
```

```
 @> Conditional Branch: Variable [0032:goal X]  != Variable [0034:entity X]
  @> Conditional Branch: Variable [0033:goal Y]  != Variable [0035:entity Y]
   @> Control Variables: [0012:tempX] = Variable [0032]
   @> Control Variables: [0013:tempY] = Variable [0033]
   @> Control Variables: [0012:tempX] += Variable [0025]
   @> Control Variables: [0013:tempY] += Variable [0026]
   @> Get Terrain ID: [0002:temp 0], Variable [0012][0013]
   @> Conditional Branch: Variable [0002:temp 0]  != Variable [0028:impassable ID]
    @> Break Loop
    @>
   : Branch End
   @>
  : Branch End
  @>
 : Branch End
 @>
: Repeat Above
@> Comment: Build map to position
@> Set Event Location: [Goal Marker], Variable [0012][0013]
@> Call Event: [Build Dijkstra Map]
```

We start with picking a random position on the map by randomizing goal X and goal Y between 0 and 9 (if you have a different map size you should change it to reflect that). Then we check if it's a valid position by checking so it's not equal to the entity's current position and that the position isn't impassable. Before getting terrain id we need to offset the randomized position by the map offset variables, we do this by storing the new offset position in temp X and temp Y.
If the position checks out we move the "Goal Marker" to the temp X and temp Y position and proceed to build a map to it.
After this we need to add some more code to handle entity movement.

```
@> Comment: Follow path
@> Loop
 @> Call Event: [Cheapest Neighbour]
 @> Conditional Branch: Variable [0021:return] == 1
  @> Set Move Route: [Entity], Move Down
  @> Wait for All Movement
  @> Control Variables: [0035:entity Y] += 1
  @>
 : Else
  @> Conditional Branch: Variable [0021:return] == 2
   @> Set Move Route: [Entity], Move Left
   @> Wait for All Movement
   @> Control Variables: [0034:entity X] -= 1
   @>
  : Else
   @> Conditional Branch: Variable [0021:return] == 3
    @> Set Move Route: [Entity], Move Right
    @> Wait for All Movement
    @> Control Variables: [0034:entity X] += 1
    @>
   : Else
    @> Conditional Branch: Variable [0021:return] == 4
     @> Set Move Route: [Entity], Move Up
     @> Wait for All Movement
     @> Control Variables: [0035:entity Y] -= 1
     @>
    : Else
```

```
        @> Text: Stuck
        @> Break Loop
        @>
       : Branch End
      @>
     : Branch End
    @>
   : Branch End
  @>
 : Branch End
@> Comment: Check if we reached goal
@> Conditional Branch: Variable [0034:entity X] == Variable [0032:goal X]
  @> Conditional Branch: Variable [0035:entity Y] == Variable [0033:goal Y]
    @> Break Loop
    @>
   : Branch End
  @>
 : Branch End
@>
: Repeat Above
@> Wait: 0.0 seconds
```

We set up a loop that we will spend all our time in until we reaches the goal tile. The first thing we do in the loop is to call the Cheapest Neighbour event. As you might remember that event checks the cost of all the neighbours and store the direction in the return variable.
We check the return variable for the direction, set our entity to move in it, wait for it finish moving and update it's position.
Last thing we do in the loop is to check if the entity reached the goal position. If so break out from the loop.

If you hit play now you should see something like this:

Epilogue

I've made a demo project containing more examples on how to move characters, controlling the player and as well the example from this tutorial.

You can download the project [here]