2D Pixel-Scrolling Tilemap

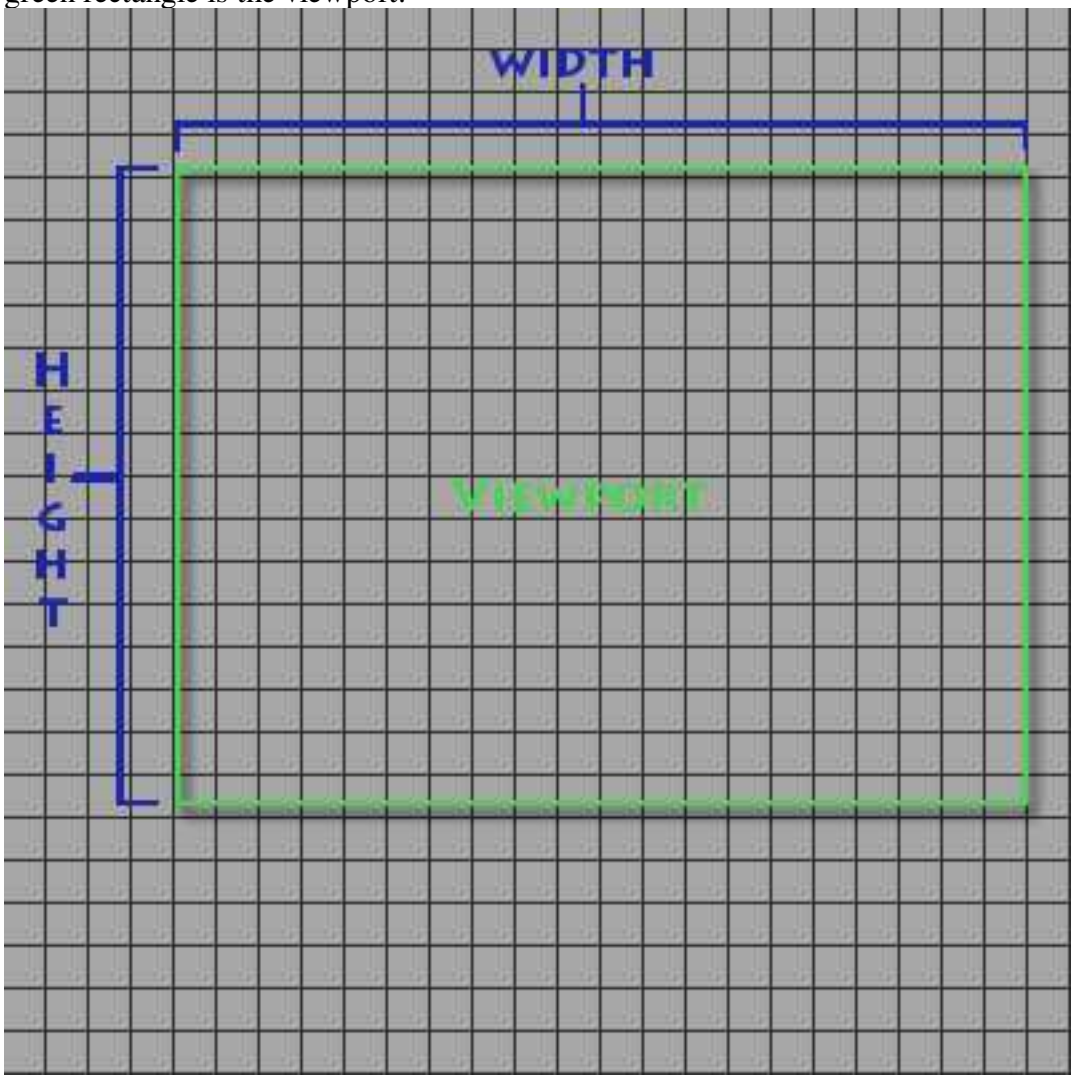
Phaelax (phaelax@hotmail.com)

Introduction

So you want to make a 2D scrolling map? What I'm about to explain can be used for a large variety of 2D games, from Mario-style platformers and sidescrollers to top-down shooters/RPGs/RTS or whatever style of game uses a scrolling map. This tutorial does assume working knowledge of DarkBasic Professional, so the reader should already know arrays, loop structures, etc…

Goal

First we need to define the goal or, in other words, what we want to accomplish. And to do that, we must define a problem. The question here is how to represent a tile map. In the picture below is a grid, imagine this grid represents our map broken into tiles. The green rectangle is the viewport.



The viewport is essentially just the screen on your monitor and typically never changes size. Simple put, the only part of the map we can see is the part that's within the viewport's dimensions. So now we know what we must accomplish, and that is to figure out the following:

1. How many tiles are in the viewport
2. Given the position of the viewport in relation to the map, which tiles should be drawn

Let's get started

First, we need to define our variables.

```
Type Tile
     img as integer
Endtype

Type SystemVariables
     mapWidth as integer
     mapHeight as integer
     viewWidth as integer
     viewHeight as integer
Endtype

Global System as SystemVariables
rem number of tiles across and down of the map
System.mapWidth = 16
System.mapHeight = 16
rem dimensions of our viewport in number of tiles
System.viewWidth = screen width()/ System.size
System.viewHeight = screen height()/ System.size

rem define the map array
Dim map( System.mapWidth ,  System.mapHeight) as Tile

rem load some map tiles
load image "tiles\t.bmp", 1
load image "tiles\t2.bmp", 2
```

The Tile UDT defines the properties for each tile in the map. In this case all we need to know is the image number that corresponds to each tile. The next UDT is the SystemVariables. These variables define the size of the map in terms of number of tiles and the size of the viewport also in number of tiles. For now, we can just set the viewport dimensions to match the maximum number of tiles that can fit within the screen's resolution. In an actual game, you would probably want some kind of HUD or interface

on part of the screen and would adjust the viewport dimensions accordingly. And finally, we create the actual map array.

For the tile images you would normally have a single image file and break it apart into individual tiles, but for simplicity we'll just load 2 separate images. Make sure your images are all the same size, and square, for this map. The 2 images I'll be using are 128x128 pixels and loaded as image values 1 and 2.

Build a map

Now that everything we need is defined, how do we actually create a map? Again for simplicity, the answer is DATA.
The DATA statement is a way to hard-code values into a program in a structured fashion.

```
Level1:
DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
```

We defined our map above as having 16 tiles across by 16 tiles down. The code above defines DATA with 16 numbers across and has 16 lines down. Each digit represents a tile, and the value represents an image number.

To load the DATA values into the array is simple:

```
restore Map
for x = 1 to System.mapWidth
     for y = 1 to System.mapHeight
          read Map(x,y)
     next y
next x
```

Of course this assumes the amount of DATA is the same as the dimensions of our map. You could put another DATA line at the beginning of the block which defines two values, width and height of the map. Then read those two values first, define your array

based on those numbers instead, then read the rest of the DATA. That would be the preferred method anyway, but instead of reading your map data from the DATA statements you would be reading it from a file you created.

Draw the map

Now the fun part, let's draw our map on the screen. Create a function called "drawMap" that takes two float parameters. The parameters will be the viewport's position. First thing we need to find out is which tile in the map to start drawing from. To do this, we simply take the viewport's position (in pixels) and divide it by the size of the tiles, which are 128 in our example. Since we can't draw part of a tile and only whole tiles, we want the integer part of this division, so drop the decimal. The fractional part will be drawn off screen, so you'll never see it anyway. Before we draw the map, we need one more piece of information, the decimal part we dropped from the previous calculation. You can get this value by taking the modulus of the viewport position and the tile size. (for those that don't know, modulus [mod] returns the remainder of a division)

```
Function drawMap(vx#, vy#)
     rem first tile to draw
     sx = int(vx#/System.size)
     sy = int(vy#/System.size)
     rem tile offset, this tiny bit is what give
     rem real pixel scrolling
     ox = vx# mod System.size
     oy = vy# mod System.size
```

Now we can actually start drawing the tiles. Set up a nest FOR loop which will start from sx,sy and continues for the dimensions of the viewport plus 1. Because we're doing per-pixel scrolling and the map can draw its first tiles partially off-screen, the edge of the map would appear choppy because it would only draw whole tiles and no fractional pieces if we didn't draw 1 more tile beyond the size. For example, if we could view 4 tiles across, the range might actually be from 1.5 to 5.5. It still only shows 4 tiles, but if we only drew from 1.5 to 5, then we have a blank space in part of our viewport. Leave out the "+1" to see what I'm talking about and it should become more clear.
Get the image of the tile that corresponds to the array coordinates from your FOR loops. Keep in mind that the math we did above results in array indices assumed to begin at 0. Since DarkBasic starts array elements at 1, we add 1 to the indices we pass into the array to retrieve the image number. Below you can see what the rest of the function looks like.

```
for x = sx to sx+System.viewWidth+1
     for y = sy to sy+System.viewHeight+1
          img = getTileImage(x+1,y+1)
          if img > 0
               ix = (x - sx)*System.size - ox
               iy = (y - sy)*System.size - oy
               paste image img, ix, iy
          endif
     next y
next x

endfunction
```

Once you got the image number, determine its position on the screen.
The (x-sx) makes sure we start at 0 on the screen, the upper left corner. Since these are map coordinates and not screen coordinates, we have to offset each value by the size of the tiles, so multiple the (x-sx) by the tile size. And finally, we subtract our offset values. This is what helps get the smooth scrolling effect by drawing partially displayed images by positioning them slightly off-screen. Without the offset, you only see the map move by whole tiles at a time, and not by each pixel.

Now you're probably wondering what that "getTileImage" function is for. All it does is it gives us a safe way to retrieve data from the array.

```
function getTileImage(x, y)
     if x > 0 and x <= System.mapWidth and y > 0 and y <=
System.mapHeight
          i = map(x,y).img
     endif
endfunction i
```

This function ensures that we will never try to access an array element outside of its bounds.

Conclusion

That's basically all there is to it. The example may not look like much, but it can be extended to so many different styles of games. Add collision and gravity and you got a side-scrolling platform game. You can also make the map auto-scroll and make an overhead space shooter. See the Appendix for the tile images I used in the example or provide your own. Just make sure you keep the dimensions the same for each image and you change the code accordingly to whatever dimensions you choose for your tile images.

Complete Code

Appendix

```
set display mode 800,600, 32

sync on
sync rate 60
backdrop on
color backdrop 0

rem properties of each tile
Type Tile
     img as integer
Endtype

rem properties of the map system
Type SystemVariables
     mapWidth as integer
     mapHeight as integer
     viewWidth as integer
     viewHeight as integer
Endtype

Global System as SystemVariables
rem how big each tile is (tiles are square so we
rem only need to define 1 number)
System.size = 128
rem make sure we're reading from the correct chunk of DATA
restore Level1
rem number of tiles across and down of the map
read System.mapWidth
read System.mapHeight
rem dimensions of our viewport in number of tiles
System.viewWidth = screen width()/ System.size
System.viewHeight = screen height()/ System.size

rem define the map array
Dim map( System.mapWidth ,  System.mapHeight) as Tile

rem load some map tiles
load image "tiles\t.bmp", 1
load image "tiles\t2.bmp", 2

rem load map data
for y = 1 to System.mapHeight
   for x = 1 to System.mapWidth
      read Map(x,y).img
   next x
next y

DO
```