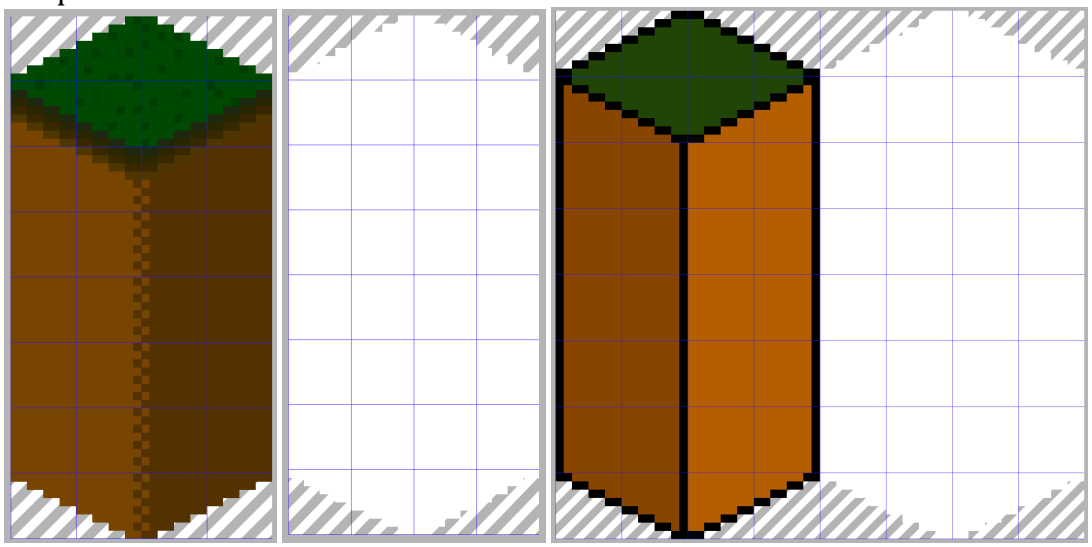
Iso – Tutorial

In diesem Tutorial werde ich nur eine „Art“ von Iso-maps behandeln, und zwar rechteckige (nicht rautenförmige) Maps.
Zu beginn brauchen wir ein kleines Tileset (in diesem fall 32x64 Pixel groß).
Das könnte dann ungefähr so aussehen. Wir brauchen zusätzlich aber noch ein Komplett weißes Tile.

Beispiel:



Wozu wir das weiße Tile brauchen, dazu später.
Schreiben wir erstmal unseren Type TTile und ein paar Methoden und Funktionen .

Code:

```
Type TTile

	Global _width:Int   = 32
	Global _height:Int= 16

	Field _xPos:Int
	Field _yPos:Int
	Field _zPos:Int

	Field _material:Int

	'seter

	Method setXPos( xPos:Int )

		_xPos = xPos
```

```
    EndMethod

    Method setYPos( yPos:Int )

        _yPos = yPos

    EndMethod

    Method setZPos( zPos:Int )

        _zPos = zPos

    EndMethod

    Method setMaterial( material:Int )

        _material = material

    EndMethod

    'geter

    Method getXPos:Int()

        Return _xPos

    EndMethod

    Method getYPos:Int()

        Return _yPos

    EndMethod

    Method getZPos:Int()

        Return _zPos

    EndMethod

    Method getMaterial:Int()

        Return _material
```

```
	EndMethod

	'functions / methods

	Function Create:TTile(
xPos:Int,yPos:Int,zPos:Int,material:Int )

		Local n:TTile = New TTile

			n.setXPos( xPos )
			n.setYPos( yPos-zPos )
			n.setMaterial( material )

		Return n

	EndFunction

	Method draw()

		DrawImage tileset,_xPos,_yPos - _zPos,_material

	EndMethod

EndType
```

Bis hier müsste eigentlich alles klar sein. Wir legen geter und seter Methoden und eine Erstell-Funktion für den Type TTile an.
Die beiden Felder _width und _height speichern breite/höhe eines Tiles.
Das Feld _zPos speichert die Höhe unseres Tiles.

Als nächste Brauchen wir einen Array der unsere viele Tiles enthält, wir nennen den Array schlicht und einfach Map.

Code:

```
Global map:TTile [ 256,256 ]
```

In dem Array werden beim einlesen der Map alle Erstellten TTiles gespeichert und beim Zeichnen der Map wird aus dem Array gelesen.

Zunächst brauchen wir noch eine Funktion um eine neue Map zu erstellen oder aus einer Datei zu laden.

Code:

```
Function readMap()
```

```
     For Local y:Int = 0 To 255
          For Local x:Int = 0 To 255

               Local xpos:Int = ((x*TTile._width)+(y Mod
2)*(TTile._width/2))
               Local ypos:Int = (y*(TTile._height/2))

               map[ x,y ] = TTile.Create( xPos,yPos,0,0 )

          Next
     Next

EndFunction
```

Wir berechnen für jedes Tile die x-Position und die Y-Position (xpos,ypos)

Berechnung der TilePosition X:

```
xpos = ((x*TTile._width) + (y Mod 2)*(TTile._width/2))
```

Die Berechnung ist gar nicht kompliziert auch wenn sie vielleicht so aussieht.
Als erstes wird jedes Tile 32 Pixel weiter nach rechts „geschoben“,in der jeder ungeraden reihen (wir beginnen mit 0 daher ist 1 die erste ungrade reihe) wird jedes Tile um die hälfte seiner breite nach rechts verschoben, damit die Tiles „ineinander greifen“.

Berechnung der TilePosition Y:

```
ypos = (y*(TTile._height/2))
```

Hier dürfte eig. Alles klar sein. Jede reihe wird um die hälfte ihrer höhle nach unten verschoben, damit die Tiles ineinander greifen.

Und übergeben die werte dann an die Funktion TTile.Create() als Parameter.
(Wir könnten die Berechnung der Koordinate natürlich auch in der draw Funktion machen, ich glaube aber das es ein klitzekleines bisschen schneller ist, so wie in diesem Tutorial beschrieben).

Da jedes Tile seine Position kennt (_xPos,_yPos) müssen wir nur noch alle Tles Durchgehen und die Methode „Draw()“ aufrufen.

Code:

```
Function drawMap()

     For Local y:Int = 0 To 255
          For Local x:Int = 0 To 255

               map[ x,y ].draw()

          Next
```

```
    Next

EndFunction
```

So, das war's dann auch schon, denn viel fehlt nun nicht mehr.
Was fehlt ist nur noch der Main-Loop, das laden der Grafik und das aufrufen von Graphics()

Hier noch mal der Ganze Code (Mit Main-Loop):

```
SuperStrict

Type TTile

    Global _width:Int   = 32
    Global _height:Int= 16

    Field _xPos:Int
    Field _yPos:Int
    Field _zPos:Int

    Field _material:Int

    'seter

    Method setXPos( xPos:Int )

        _xPos = xPos

    EndMethod

    Method setYPos( yPos:Int )

        _yPos = yPos

    EndMethod

    Method setZPos( zPos:Int )

        _zPos = zPos

    EndMethod

    Method setMaterial( material:Int )
```

```
          _material = material

     EndMethod

     'geter

     Method getXPos:Int()

          Return _xPos

     EndMethod

     Method getYPos:Int()

          Return _yPos

     EndMethod

     Method getZPos:Int()

          Return _zPos

     EndMethod

     Method getMaterial:Int()

          Return _material

     EndMethod

     'functions / methods

     Function Create:TTile(
xPos:Int,yPos:Int,zPos:Int,material:Int )

          Local n:TTile = New TTile

               n.setXPos( xPos )
               n.setYPos( yPos-zPos )
               n.setMaterial( material )

          Return n

     EndFunction
```

```
	Method draw()

		DrawImage tileset,_xPos,_yPos - _zPos,_material

	EndMethod

EndType

Global map:TTile [ 256,256 ]

Graphics 800,600

Global tileset:TImage = LoadAnimImage(
"gfx/tileset.png",32,64,0,2 )
readMap()

Repeat
Cls

	drawMap()

Flip 0
Until AppTerminate()

Function readMap()

	For Local y:Int = 0 To 255
		For Local x:Int = 0 To 255

			Local xpos:Int = ((x*TTile._width)+(y Mod
2)*(TTile._width/2))
			Local ypos:Int = (y*(TTile._height/2))

			map[ x,y ] = TTile.Create( xPos,yPos,0,0 )

		Next
	Next

EndFunction

Function drawMap()

	For Local y:Int = 0 To 255
```

```
            For Local x:Int = 0 To 255

                map[ x,y ].draw()

            Next
        Next

EndFunction
```

Das war's dann fürs erste.
Im zweiten Tutorial wird behandelt: Kollision der Tiles und Verbesserung(Performance und Technik)

Danke fürs Lesen.
J. W. , 25.12.2009