Pemilihan Garis Pada Saat Runtime

Yan Friskantoni
aan43@yahoo.com

Pengantar

Untuk mendeteksi apakah user telah memilih objek garis pada saat aplikasi berjalan melalui posisi mouse adalah suatu permasalahan yang cukup menantang. Padahal akses untuk mendapatkan fungsi tersebut sangat penting untuk sebagian aplikasi, kususnya untuk aplikasi yang menyangkut dengan penggunaan garis atau grafik. Sayangnya, fungsi standart untuk hal tersebut, tidak di sediakan secara default oleh compiler kebanyakan. Sehingga, kita tampaknya harus menyediakan sendiri fungsi tersebut.

Dasar Teori

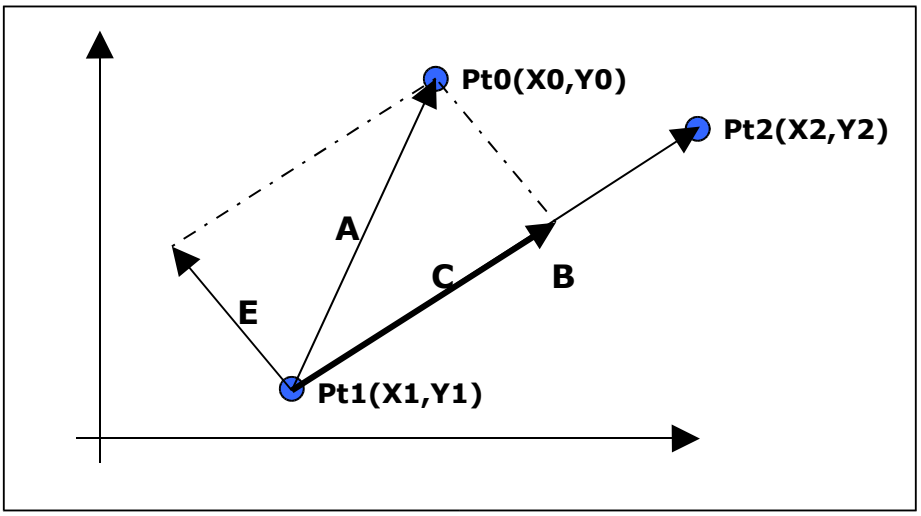
Pada kenyataannya ada banyak teori yang bisa digunakan dalam implementasinya. Namun yang paling mudah dan bisa diandalkan adalah dengan menggunakan teori proyeksi dari vector 2 dimensi.

Misalkan kita memiliki garis L, dimana garis tersebut memiliki koordinat pt1(X1,Y1) dan pt2(X2,Y2) di masing-masing ujungnya, maka kita bisa dapatkan vector **B** yang mewakili garis tersebut adalah :

$$\mathbf{B = (X1 - X2 , Y1 – Y2)}$$

Apabila terdapat titik pt0(X0,Y0) yang tidak berada dalam garis tersebut, dan kita ingin dapatkan jarak diantara garis dan titik tersebut, maka terlebih dahulu kita bisa menggunakan rumus proyeksi vektor.

Ambil garis dari titik pt1(X1,Y1) ke titik pt0(X0,Y0). Sehingga kita bisa mendapatkan vektor baru, kita namakan saja vektor **A**. Kemudian dengan menggunakan rumus proyeksi vektor **A** terhadap Vektor **B**, kita akan mendapatkan vektor **C** yang merupakan proyeksi vektor **A** terhadap vektor **B** (lebih jelasnya lihat gambar dibawah).



Tentunya tidak sulit untuk mendapatkan vektor **C**, dengan menggunakan rumus proyeksi vektor **A** terhadap vektor **B**.

$$C = ((A * B) / (|B| ^ 2)) * B$$

Setelah kita mendapatkan vektor **C**, maka dengan mudah kita bisa mendapatkan vektor **E**, yaitu dengan mengurangkan vektor **A** dengan vektor **C**.

$$E = A - C$$

Misalkan (3,3) dan (2,3) masing-masing adalah vektor **A** dan **B**. Maka vektor **E** adalah (3-2, 3-3)= (1, 0). Dari vektor **E** inilah, apabila kita menormalkannya, maka kita bisa mendapatkan jarak antara titik pt0 (X0,Y0) terhadap garis L.

Untuk menormalkan vektor **E**, kita menggunakan rumus :

$$Norm(E) = \sqrt{((a ^ 2) + (b ^ 2))}$$

Dimana a dan b masing-masing adalah elemen dari vektor **E**.

Implementasi

Untuk keperluan implementasi ini, saya menggunakan MS Visual Basic v 6.0. Sedangkan untuk source code dibawah ini, anda bisa mengetikkannya di **modul** atau bisa juga di **class module**. Bagi anda yang menggunakan bahasa pemrograman lainnya, saya rasa dengan menyimak dasar teori dan menganalisa source code dalam bahasa Basic beserta penjelasannya, anda bisa dengan mudah mengimplementasikannya di bahasa favorite anda.

Sebagai permulaan, kita akan membuat variabel dengan type sendiri yang akan mewakili vektor yang akan kita gunakan. Type ini kita namakan **VECTOR2D**.

```
Private Type VECTOR2D
      A As Single
      B As Single
End Type
Private Const Radius = 50
```

Konstanta Radius sendiri, kita gunakan untuk mengetahui seberapa besar jarak maksimal yang bisa kita golongkan sebagai aksi memilih garis. Semakin besar nilai Radius, semakin besar toleransi yang bisa diterima, atau semakin mudah user dalam memilih garis.

Apabila kita lihat kembali teori diatas, didalam implementasinya, nilai titik pt0(X0,Y0) adalah nilai dari posisi mouse yang saat itu di 'click' oleh user diatas form active. Sedangkan garis L, adalah object control Line standart yang dimiliki oleh Visual Basic. Kita bisa ambil property X1, X2, Y1, dan Y2 dari control Line yang akan kita uji, dan menjadikannya nilai dari pt1(X1,Y1) dan pt2(X2,Y2). Apabila nama object control Line kita namakan **currLine**, maka kita bisa dapatkan nilai vector A dan B adalah :

```
Dim AVec As VECTOR2D, BVec As VECTOR2D
 '//vector garis
BVec.A = currLINE.X1 - currLINE.X2
BVec.B = currLINE.Y1 - currLINE.Y2
 '//vector titik garis (X1,Y1) ke posisi mouse (dx, dy)
AVec.A = currLINE.X1 - dx
AVec.B = currLINE.Y1 – dy
```

Source code lengkapnya untuk itu adalah :

```
Public Function LineClick(currLine as Line,dx As Single, dy As Single) As
Boolean
'//start checking
 Dim AVec As VECTOR2D, BVec As VECTOR2D
 '//vector garis
 BVec.A = currLINE.X1 - currLINE.X2
 BVec.B = currLINE.Y1 - currLINE.Y2
 '//vector titik garis (X1,Y1) ke posisi mouse
 AVec.A = currLINE.X1 - dx
 AVec.B = currLINE.Y1 - dy
 '//now we proceed it
 If lengthOFNormal(AVec, BVec) <= Radius Then
    LineClick = True
 Else
    LineClick = False
 End If
End Function
```

Fungsi LineClick(currLine as Line ,dx As Single, dy As Single) akan menguji, apakah user telah memilih currLine. Fungsi akan mengembalikan nilai True apabila benar dan False apabila currLine tidak dipilih dengan benar oleh user.

Fungsi lengthOFNormal(AVec, BVec) sendiri, akan mencari tahu jarak titik terhadap garis, dengan menggunakan informasi berupa vektor Avec dan vektor Bvec (lihat teory diatas **pen.**). Apabila panjangnya melebihi nilai constanta Radius, maka kita simpulkan Line tidak terpilih oleh user. Line terpilih apabila sebaliknya.

```
Private Function lengthOFNormal(dA As VECTOR2D, dB As VECTOR2D) As
Single
 '//dapatkan vector proyeksi A terhadap B
 '// C = ((A * B)/(|B|^2)) * B
 Dim C As VECTOR2D, E As VECTOR2D
 C.A = dB.A * (dotProduct(dA, dB) / dotProduct(dB, dB))
 C.B = dB.B * (dotProduct(dA, dB) / dotProduct(dB, dB))
 '//dapatkan proyeksi dari E dimana E = A - C
 E = substractVector(dA, C)
 '//dapatkan panjang vector E
 lengthOFNormal = normalizeVector(E)
End Function
```

Fungsi dotProduct(vecA As VECTOR2D, vecB As VECTOR2D), akan melakukan perkalian titik (.) diantara vecA dan vecB. Hasilnya sendiri adalah penjumlahan dari hasil perkalian antar elemen matrik.

```
Private Function dotProduct(vecA As VECTOR2D, vecB As VECTOR2D) As Single
 dotProduct = vecA.A * vecB.A + vecA.B * vecB.B
End Function
```

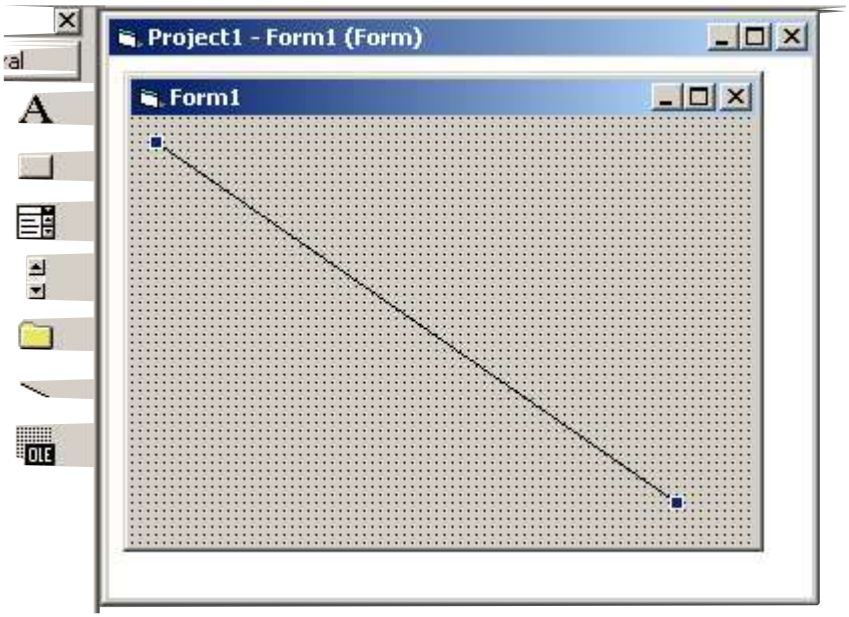
Sedangkan substractVector(vecA As VECTOR2D, vecB As VECTOR2D), akan mengurangkan vektor vecA dengan vektor vecB.

```
Private Function substractVector(vecA As VECTOR2D, vecB As VECTOR2D) As
VECTOR2D
 Dim tmpVec As VECTOR2D
 tmpVec.A = vecA.A - vecB.A
 tmpVec.B = vecA.B - vecB.B
 substractVector = tmpVec
End Function
```

Sedangkan fungsi terakhir yang kita gunakan adalah normalizeVector(thisVector As VECTOR2D). Fungsi ini akan mengembalikan hasil proses normalisasi atas thisVector.

```
Private Function normalizeVector(thisVector As VECTOR2D) As Single
 normalizeVector = Sqr((thisVector.A ^ 2) + (thisVector.B ^ 2))
End Function
```

Untuk mencoba fungsinya, anda bisa membuat project baru, dengan satu form standart. Kemudian tambahkan control Line di form.



Langkah selanjutnya hanya menambahkan code validasi inputan user di prosedur **Form_MouseDown**, dengan cara memanggil fungsi validasi yang telah kita buat tadi. Untuk lengkapnya, seperti di bawah ini.

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, x As
Single, y As Single)
 If Button = vbLeftButton Then
      If LineClick(Line1,x,y) Then
```

```
            MsgBox "Line Selected"
      Else
            MsgBox "Ehm …."
      End If
 End If
End Sub
```

Anda bisa bereksprimen dengan mengubah nilai constanta Radius.

Kesimpulan

Digunakannya vektor dalam aplikasi pemilihan objek garis, sangat memudahkan programmer. Karena dengan hanya berbekal teori yang sederhana, bisa memberikan efek yang bisa diandalkan dalam pengembangan aplikasi yang membutuhkan fasilitas pemilihan objek garis.

Saran Pengembangan

Dari hasil implementasi, aplikasi sudah bisa menentukan apakah user memilih objek garis atau tidak. Tetapi kekurangan dari implementasi diatas, apabila posisi mouse user berada di perpanjangan dari objek garis, maka program akan terus menganggap bahwa user telah memilih garis. Walaupun perpanjangan tersebut sebenarnya bukan bagian dari garis lagi. Sehingga perlu ditambahi proses validasi sederhana terhadap posisi mouse user.

Selain itu pula, contoh hanya untuk satu objek garis, bagaimana kalau banyak garis, bagaimana kalau kumpulan garis yang menjadi polyline ?

.

Lampiran A

Bikin project baru. Kemudian tambahkan control Line di form1.

Sekarang masuk ke **form1(code)**, ketikkan kode berikut.

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, x As
Single, y As Single)
 If Button = vbLeftButton Then
      If LineClick(Line1,x,y) Then
            MsgBox “Line Selected”
      Else
            MsgBox “Ehm ….”
      End If
 End If
End Sub
```

Langkah selanjutnya, lihat lampiran B.

Lampiran B

Tambahkan module ke project, kemudian ketikkan code ini di module.

```
Option Explicit

Private Type VECTOR2D
      A As Single
      B As Single
End Type
Private Const Radius = 50

Public Function LineClick(currLine as Line,dx As Single, dy As Single) As
Boolean
'//start checking
 Dim AVec As VECTOR2D, BVec As VECTOR2D
 '//vector garis
 BVec.A = currLINE.X1 - currLINE.X2
 BVec.B = currLINE.Y1 - currLINE.Y2
 '//vector titik garis (X1,Y1) ke posisi mouse
 AVec.A = currLINE.X1 - dx
 AVec.B = currLINE.Y1 - dy
 '//now we proceed it
 If lengthOFNormal(AVec, BVec) <= Radius Then
    LineClick = True
 Else
    LineClick = False
 End If
End Function


Private Function lengthOFNormal(dA As VECTOR2D, dB As VECTOR2D) As
Single
 '//dapatkan vector proyeksi A terhadap B
 '// C = ((A * B)/(|B|^2)) * B
 Dim C As VECTOR2D, E As VECTOR2D
 C.A = dB.A * (dotProduct(dA, dB) / dotProduct(dB, dB))
 C.B = dB.B * (dotProduct(dA, dB) / dotProduct(dB, dB))
 '//dapatkan proyeksi dari E dimana E = A - C
 E = substractVector(dA, C)
 '//dapatkan panjang vector E
 lengthOFNormal = normalizeVector(E)
End Function

Private Function dotProduct(vecA As VECTOR2D, vecB As VECTOR2D) As Single
 dotProduct = vecA.A * vecB.A + vecA.B * vecB.B
End Function

Private Function substractVector(vecA As VECTOR2D, vecB As VECTOR2D) As
VECTOR2D
 Dim tmpVec As VECTOR2D
 tmpVec.A = vecA.A - vecB.A
 tmpVec.B = vecA.B - vecB.B
 substractVector = tmpVec
End Function

Private Function normalizeVector(thisVector As VECTOR2D) As Single
 normalizeVector = Sqr((thisVector.A ^ 2) + (thisVector.B ^ 2))
End Function
```