2016

TBZ

Stoop Adrian

VEKTORGRAFIK

3D-Grafik auf dem MCB32

Inhaltsverzeichnis

Projektbeschreibung

Lastenheft(Was & Warum)

Ich werde auf dem Mikrocontroller-Display ein 3D-Würfel rotieren lassen in der X,Y und Z Achse.

Der Würfel soll im als auch gegen den Uhrzeiger drehen.

Kein Schalter Bewirk eine Veränderung des Würfels.

Der Code soll auskommentiert sein und nachvollziehbar.

Der Würfel soll nicht immer gleich drehen.

Pflichtenheft(Wie & Womit)

Einführung

Auftraggeber: RMA

Auftragnehmer: AST

Es soll ein 3D Würfel auf einem 2D Display mit Vektorgrafik dargestellt werden.

Auftrag

- Programmiert werden muss:

    Ein 3D-Würfel, der sich in X,Y und Z Achse drehen kann.

    Kein Schalter oder Berührung soll den Prozess ändern.

    Der Würfel soll im und gegen den Uhrzeiger drehen.

    Der Würfel soll den Bildschirm so gut wies geht Füllen mit seiner Grösse.

Auftragszeit: 10.09.15 – 07.01.2015

Beschreibung der Schnittstellen

Das Programm hat keine Schnittstellen und läuft ohne Benutzter eingaben.

Analyse des Problems

Das Display kann evt. Zu langsam sein um den Würfel „fliessend“ darzustellen.

Der Code kann max. 32Kb gross sein, sonst muss Keil gekauft werden(Lite Version).

Externe Einflüsse

Wenn der Mikrocontroller angeschlossen wird, soll das Programm starten.

Wenn der Mikrocontroller ausgeschaltet wird und wieder ein, soll das Programm von vorne beginnen.

Beurteilung der Macharbeit des Auftrages

a) Organisatorisch / Betriebswirtschaftlich
   Die Aufgabe kann mit den verfügbaren Mittel bewältigt werden.

b) Technisch
   Die Hardware ist dieser Aufgabe gewachsen, ausser das refreshen des Displays könnte ein flimmern bewirken.

Hard & Software

Ich Arbeite alleine mit dem ARM STM32 F107VC von ST.(MCB32)

Ich benutzte als Compiler Keil 5 (http://www2.keil.com/mdk5/install)

Projekt wurde erstellt gemäss Dokument: MCB32_uv5_erstes_projekt_V323Beta.pdf auf Db.

Tagebuch

10.09.1015	Heute hab ich den Projektplan erstellt. Ich denke ich werde keine grosse Schwierigkeiten haben bei diesem Projekt.
17.09.2015	Ich habe im Projektplan alles unterteilt und auf das ganze Semester verteilt.
24.09.2015	In dem Dossier, das ich bekommen habe, sind sehr nützliche links gewesen. Ich habe im Internet viele Beispiele gesehen, aber leider nicht immer in C :/
01.10.2015	Ich habe herausgefunden, dass die x und y Achse immer gleich verschoben werden muss, sonst entsteht aus dem Würfel ein Rechteck.
08.10.2015	Das erste Testprogramm ist geschrieben aber läuft noch nicht, da die Vektoren falsch berechnet werden.
15.10.2015	Nicht weiter gekommen >> immer noch selben Fehler :/
22.10.2015	Fehler wurde gefunden >> es wurde 2x x-achse genommen anstatt 1x x-achse und 1x y-achse
29.10.2015	Programm läuft ohne Probleme ausser das der Display ein wenig flackert.
05.11.2015	Habe angefangen die PPP vorzubereiten.
12.11.2015	Weiterarbeit an PPP.
19.11.2015	PPP beendet und ich habe mir vorgenommen eine weitere Grafik einzubinden (Pfeil)
26.11.2015	Habe jetzt Würfel und Pfeil, die sich drehen in x,y und z-Achse.
02.12.2015	Code wurde erweitert, sodass man max. 40 Vektorpunkte benutzten kann (Würfel hat 8)
10.12.2015	Code wurde Auskommentiert
17.12.2015	Abschliessung des Projekt
24.12.2015	Ferien
31.12.2015	Ferien
07.01.2016	Vortrag

Projektplannung

3D Vektor würfel auf 2D Display	10.09.2015	17.09.2015	24.09.2015	01.10.2015	08.10.2015	15.10.2015	22.10.2015	29.10.2015	05.11.2015	12.11.2015	19.11.2015	26.11.2015	02.12.2015	10.12.1015	17.12.2015	24.12.2015	31.12.2015	07.01.2016	14.01.2015
Ferien					x	x													
keine Schule										x									
Projektbeschreibung	x																		
Projektplan	x																		
Vektoren Information suche		x	x																
Formelbuch erstellen			x	x															
erste Idden aufstellen				x															
Würfel programieren				x	x														
x achse drehen					x	x	x	x											
y achse drehen								x	x										
z achse drehen									x	x									
2 achsen drehen										x	x	x							
3 achsen drehen												x	x	x	x				
Präsentation															x	x	x		
Überarbeitung, auskommentieren																	x	x	x
Stunden aufwand Geschätzt:	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Stunden Aufwand:	4	4	3	4	4	3	3	4	4	4	3	4	3	3	4	3	3	7	0

Dokumentation

Was ist ein Vektor

Ein Vektor(lat. *Vector „Träger, Fahrer“*) ist ein mathematisches Objekt mit einer Länge und Richtung.

Es beschreibt eine Parallelverschiebung in der Ebene oder im Raum.

Variablen, die für einen Vektor stehen, werden häufig mit einem Pfeil beschriftet: $\vec{a}, \vec{v}$

Pfeile, die parallel, gleich lang und gleich gerichtet sind, beschreiben dieselbe Verschiebung und stellen somit denselben Vektor dar.

Wenn ein Vektor mehr als eine Variable hat, werden die zahlen untereinander geschrieben:

$$\vec{v} = \begin{pmatrix} 7 \\ 3 \end{pmatrix}$$

7 = x-Achse

3 = y-Achse

Wo werden Vektoren eingesetzt

Vektoren werden z.B. für Vektorräume gebraucht.

Ein Vektorraum oder Linearer Raum ist eine algebraische Struktur, die in vielen Teilgebieten der Mathematik verwendet wird.

Vektorräume in der Physik bringen zu Beginn des 20. Jahrhunderts den großen Durchbruch zum Verständnis der Welt, in der wir leben. Vektoren und Tensoren in der Riemann'schen Geometrie führen zur Relativitätstheorie. (Universum, Kosmologie, ..., die Welt im Großen.) Vektoren in Hilberträumen braucht man für die Quantentheorie. (Atome, Elementarteilchen, Protonen, Neutronen, Elektronen, Neutrinos, Quarks, ..., die Welt im Kleinen).

In der Mathematik fällt mir spontan keine Theorie ein, in der man Vektorräume NICHT brauchen könnte:

1. Geometrie: ist mit analytischer Geometrie auf Vektorräumen aufgebaut.
2. Analysis: Funktionen bilden Vektorräume. Differentiale und Integrale sind lineare Operatoren.
3. Funktionentheorie: komplexe Funktionen genauso
4. Funktionalanalysis, die Theorie von Operatoren, baut auf unendlichdimensionale Vektorräume.
5. (lineare) Optimierung, die heute am weitesten verbreitete mathematische Anwendung, nutzt Matrixalgebra, also Vektorräume.
6. Approximation, Interpolation von Funktionen: Newton, Leibniz, Fourier und neuerdings Spline-Funktionen : alles Vektorräume
7. Zahlentheorie: Polynomringe sind Vektorräume, Körpererweiterungen sind Vektorräume, ...

Was ist eine Vektorgrafik

Vektorgrafik basieren auf der Rastergrafik, nicht auf einem Pixelraster. Eine Vektorgrafik ist eine Computergrafik, die nicht jedes Pixel abspeichern, sondern grafische Primitiven wie Linien, Kreisse, Polygonen oder allgemein Kurven Speichert. Meinst versteht man unter Vektorgrafik die Darstellung in der zweidimensionaler Ebene. Eine Bildbeschreibung, die sich auf dreidimensionale Primitiven stützt, wird eher 3D-Modell oder Szene genannt.

Wenn ein Kreiss abgespeichert werden soll, benötigt eine Vektorgrafik mindestens zwei Werte: die Lage des Kreismittelpunkts und der Kreisdurchmesser. Neben der Form und Position der Primitiven werden eventuell auch die Farbe, Strichstärke, diverse Füllmuster und weitere, das Aussehen bestimmende Daten angegeben.

Vektorgrafiken lassen sich ohne Qualitätsverlust beliebig skalieren.

Formeln

Um einen Würfel zu Programmieren der um seine Achsen mit Vektorfunktionen drehen soll muss man nur eine Formel **verstehen:**

$$\begin{aligned} x'_1 &= \cos\theta \cdot x_1 - \sin\theta \cdot x_2 \\ x'_2 &= \sin\theta \cdot x_1 + \cos\theta \cdot x_2 \\ x'_3 &= x_3 \end{aligned}$$

Damit man jetzt ein Vektorpunkt mit 3 Koordinaten auf 2 Koordinaten umrechnen kann, muss man diese Formel mehrmals anwenden(mehrere Winkel):

Vorgaben:

Vektorpunkte: X,Y,Z

Zwischenvariable: Xt,Yt,Zt

Winkel Xan,Yan,Zan

Yt = Y * cos(Xan) - Z * sin(Xan)

Zt = Y * sin(Xan) + Z * cos(Xan)

Y = Yt

Z = Zt

Xt = X * cos(Yan) - Z * sin(Yan)

Zt = X * sin(Yan) + Z * cos(Yan)

X = Xt

Z = Zt

Xt = X * cos(Zan) - Y * sin(Zan)

Yt = X * sin(Zan) + Y * cos(Zan)

X = Xt (resultat X-Pixel)

Y = Yt (resultat Y-Pixel)

C-Code

```
/*includes*/
#include <stm32f10x.h>
#include "TouchP0P1.h"
#include "math.h"
/*Variablen*/
double Figur[125] = {
//    x   y   z
   -20,-80, 10,  // 1. Vektor
    20,-80, 10,  // 2. Vektor
    20,-20, 10,  // 3. Vektor
    60,-20, 10,  // 4. Vektor
    60, 20, 10,  // 5. Vektor
    20, 80, 10,  // 6. Vektor
   -20, 80, 10,  // 7. Vektor
   -60, 20, 10,  // 8. Vektor
   -60,-20, 10,  // 9. Vektor
   -20,-20, 10,  // 10. Vektor
```

Math.h braucht man für die sin/cos Funktionen.

Es wird ein Array erstellt das immer abwechselnd x,y,z Vektor-Koordinaten hat.

```
double  Vektor_1_x, Vektor_1_y,
        Vektor_2_x, Vektor_2_y,
        Vektor_3_x, Vektor_3_y,
        Vektor_4_x, Vektor_4_y,
        Vektor_5_x, Vektor_5_y,
        Vektor_6_x, Vektor_6_y,
        Vektor_7_x, Vektor_7_y,
        Vektor_8_x, Vektor_8_y,
        Vektor_9_x, Vektor_9_y,
        Vektor_10_x, Vektor_10_y,
```

Da meine Formel aus 3 Werten 2 macht, speichere ich sie separat ab.

```
/*Funktionen*/
int sgn(int x){return (x > 0) ? 1 : (x < 0) ? -1 : 0;}
void Linie(int xstart,int ystart,int xend,int yend, int farbe)
```

Diese Zwei Funktionen hab ich aus dem Wiki und sie verbinden 2 Vektorpunkte miteinander auf dem Display (Bresenham-Algorithmus).

```
void pixel(int farbe)
    {
        Linie(Vektor_1_x,Vektor_1_y,Vektor_2_x,Vektor_2_y,farbe);
        Linie(Vektor_2_x,Vektor_2_y,Vektor_3_x,Vektor_3_y,farbe);
        Linie(Vektor_3_x,Vektor_3_y,Vektor_4_x,Vektor_4_y,farbe);
        Linie(Vektor_4_x,Vektor_4_y,Vektor_5_x,Vektor_5_y,farbe);
        Linie(Vektor_5_x,Vektor_5_y,Vektor_6_x,Vektor_6_y,farbe);
        Linie(Vektor_6_x,Vektor_6_y,Vektor_7_x,Vektor_7_y,farbe);
        Linie(Vektor_7_x,Vektor_7_y,Vektor_8_x,Vektor_8_y,farbe);
        Linie(Vektor_8_x,Vektor_8_y,Vektor_9_x,Vektor_9_y,farbe);
        Linie(Vektor_9_x,Vektor_9_y,Vektor_10_x,Vektor_10_y,farbe);
        Linie(Vektor_10_x,Vektor_10_y,Vektor_1_x,Vektor_1_y,farbe);
```

In dieser Funktion wird bestimmt welche Punkte auf dem Display verbunden werden sollen.

Bei einem Würfel sind es 12 Linien die Gezeichnet werden müssen.

```
void delay(){Long t;for(t = 0;t<100000;t++);}
```

Delay Rutine.

```
void Vektor(double X,double Y, double Z, int Vektorenanzahl)
    {
        int increment = 0;
        double Xt,Yt,Zt;
        double Zan =  Z;
        double Yan =  Y;
        double Xan =  X;

        int Counter = 1;

        while(Counter <= (Vektorenanzahl + 1))
            {
                Yt = (Figur[increment+1] * cos(Xan)) - (Figur[increment+2] * sin(Xan));
                Zt = (Figur[increment+1] * sin(Xan)) + (Figur[increment+2] * cos(Xan));

                Figur[increment+1] = Yt;
                Figur[increment+2] = Zt;

                Xt = (Figur[increment] * cos(Yan)) - (Figur[increment+2] * sin(Yan));
                Zt = (Figur[increment] * sin(Yan)) + (Figur[increment+2] * cos(Yan));

                Figur[increment] = Xt;
                Figur[increment+2] = Zt;

                Xt = (Figur[increment] * cos(Zan)) - (Figur[increment+1] * sin(Zan));
                Yt = (Figur[increment] * sin(Zan)) + (Figur[increment+1] * cos(Zan));

                Figur[increment] = Xt;   // plot x
                Figur[increment+1] = Yt; // plot y

                switch(Counter)
                    {
                        case(1):{Vektor_1_x = Xt;Vektor_1_y = Yt;break;}
                        /**/
                        case(40):{Vektor_40_x = Xt;Vektor_40_y = Yt;break;}
                    }

                increment = increment + 3;
                Counter++;
            }
    }
```

Funktion von der Formel oben. Ich habe die Funktion so ausgeschrieben, das man sie auf egal wie viele Vektoren anwenden kann.

```
int main(void)
    {

        InitTouchScreen();
        clearScreen(WHITE);

        while(1)
            {
                Vektor(0.0125,0.0125,0.025,20);
                pixel(BLACK);
                delay();
                pixel(WHITE);

            }
    }
```

Die Main Funktion.

Schlusswort

Ich fand das Thema sehr spannend und werde Privat selber noch etwas daran arbeiten. Ich habe mir schon den Source Codes von DOOM runtergeladen, weil ich durch das viel mehr Interesse bekommen hab was Grafik oder Spielbeginne angeht. Der Einstieg war etwas schwierig, weil ich nicht immer wusste ob ich diese Fakten, die ich lese überhaupt brauche.