

Tutorium für einen simplen Space-Shooter by Jasper111

Teil 1

Angeregt durch Gernot und Schranz0r und andere aus dem GLBasic - Forum.

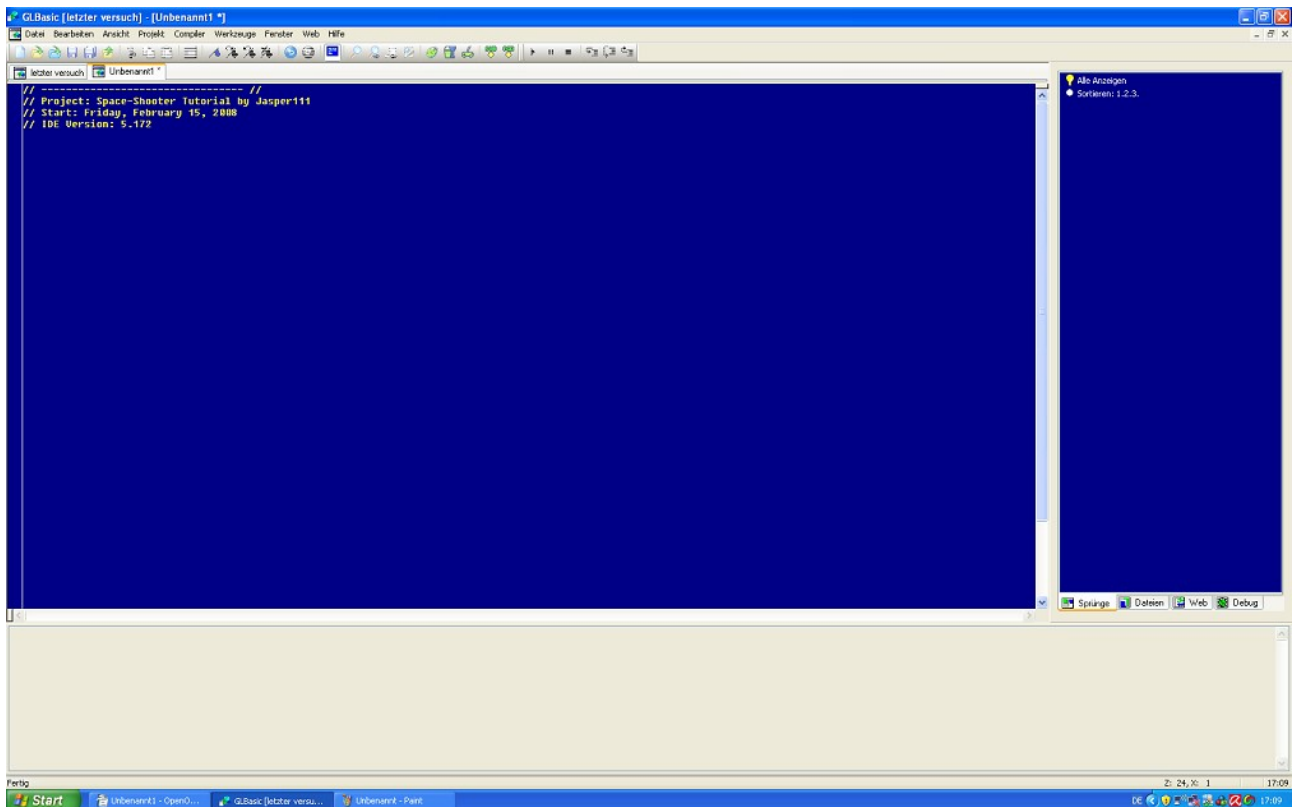
Der Anfang

Unser Space-Shooter soll einen Spieler, Gegner (hier in Form von Asteroiden), sowie einen Schuss enthalten. Hierfür brauchen wir dann auch die drei genannten Sprites(Bilder).Diese legen wir im Projekt - Ordner ab. Am besten machen wir einen GFX – Ordner für unsere Grafiken.



Hinweis: Die Grafiken hab ich aus einem Forum, wo sie bereitgestellt wurden und somit frei benutzt werden können.

So, nach dem Start und den Einstellungen für unser neues Projekt haben wir folgenden Bildschirm. Bis auf auf die Projektdaten wie Name,Datum u.s.w sehen wir nichts.



Hauptschleife erstellen

Ich baue als nächstes immer erst die Hauptschleife ein (könnt ihr später machen wie es euch beliebt).

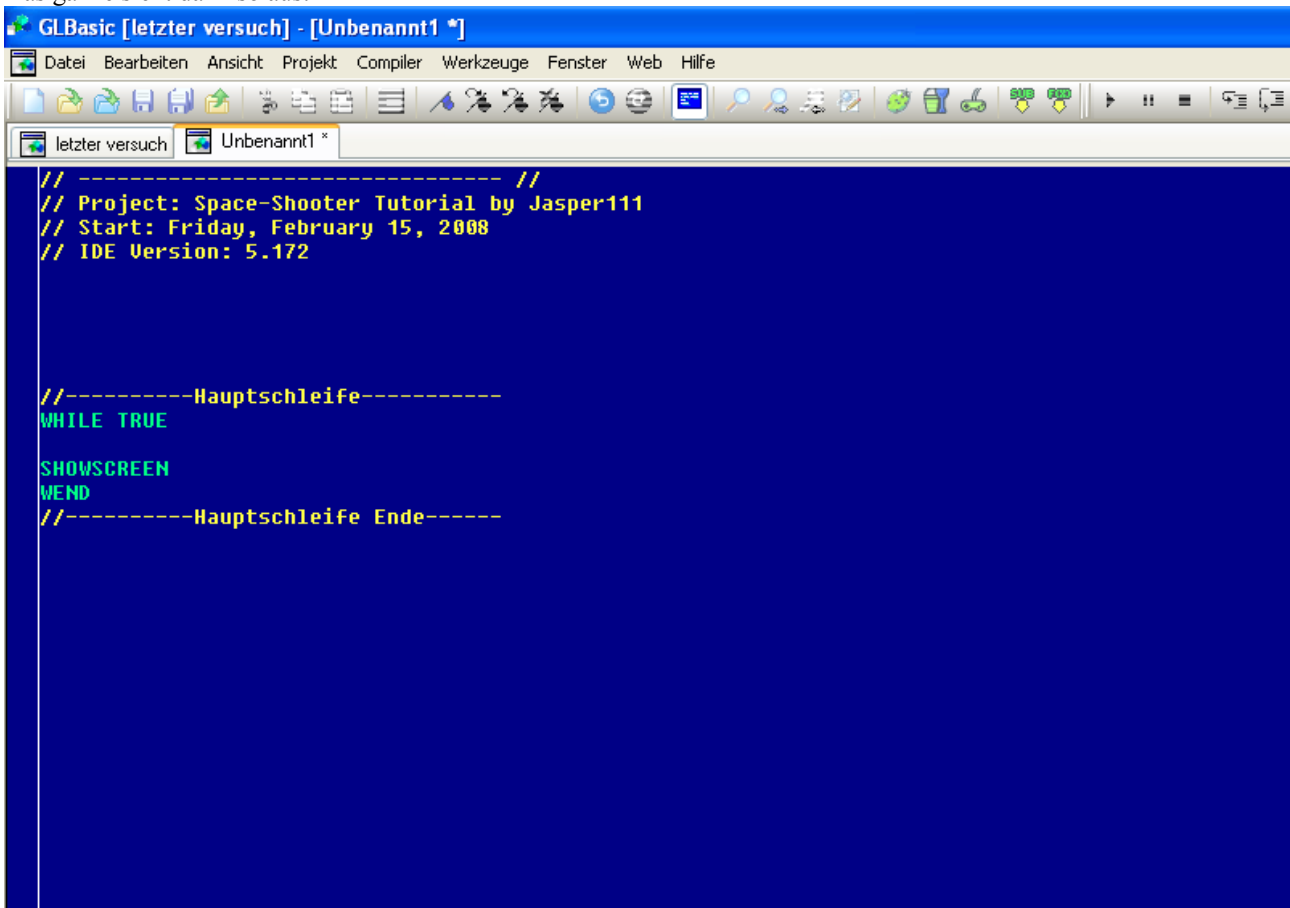
Diese beinhaltet drei Befehle.

- 1.) **WHILE TRUE** = Wenn wahr
- 2.) **SHOWSCREEN** = Zeige Bildschirm (ANZEIGE)
- 3.) **WEND** = Wenden(zurück)

Die Hauptschleife wird immer wieder durchlaufen und das ist für unser Programm wichtig, weil das Programm sonst einmal von oben nach unten durchlaufen und beendet würde.

Durch **WHILE TRUE** und **WEND** verhindern wir das Beenden, wenn das Programm auf **WEND** stößt geht es zurück zu **WHILE TRUE** und läuft wieder bis **WEND** und wieder und wieder.

Das ganze sieht dann so aus:



```
// ----- //
// Project: Space-Shooter Tutorial by Jasper111
// Start: Friday, February 15, 2008
// IDE Version: 5.172

//-----Hauptschleife-----
WHILE TRUE
SHOWSCREEN
WEND
//-----Hauptschleife Ende-----
```

Spielfigur einbauen und bewegen

So, und nun zur Spielfigur von uns, das Raumschiff.



Die Grafik muss erst einmal geladen werden, dafür brauchen wir den Befehl '**LOADSPRITE**'.

Der Befehl **LOADSPRITE** lädt unser Bild in den Speicher.

Um ein das Bild zu laden benötigt der Befehl noch einige Daten wie den Namen und das Format des Bildes, sowie eine **ID** mit der das Bild dann angesprochen wird. Die **ID** besteht aus einer ganz Zahl wie 0,1,2,3....u.s.w.

Der ganze Befehl sieht dann so aus >> **LOADSPRITE**"PFAD auf der Festplatte/**raumschiff.png**", (**ID**) **0** <<
Um das Raumschiff auf dem Bildschirm auch zu sehen brauchen wir noch einen weiteren Befehl und zwar **DRAWSPRITE**, der Befehl benötigt unsere vergebene **ID** und die Position auf der x-Achse sowie y-Achse des Bildschirms.

Also hier der Ganze Code:

```
LOADSPRITE "../thomas/raumschiff.png", 0 // <<< Bild in den Speicher laden und eine ID vergeben
// (Hier ist die ID = 0), der Pfad ist der Weg zum Bild auf
// der Festplatte, wo auch immer es gerade liegt.

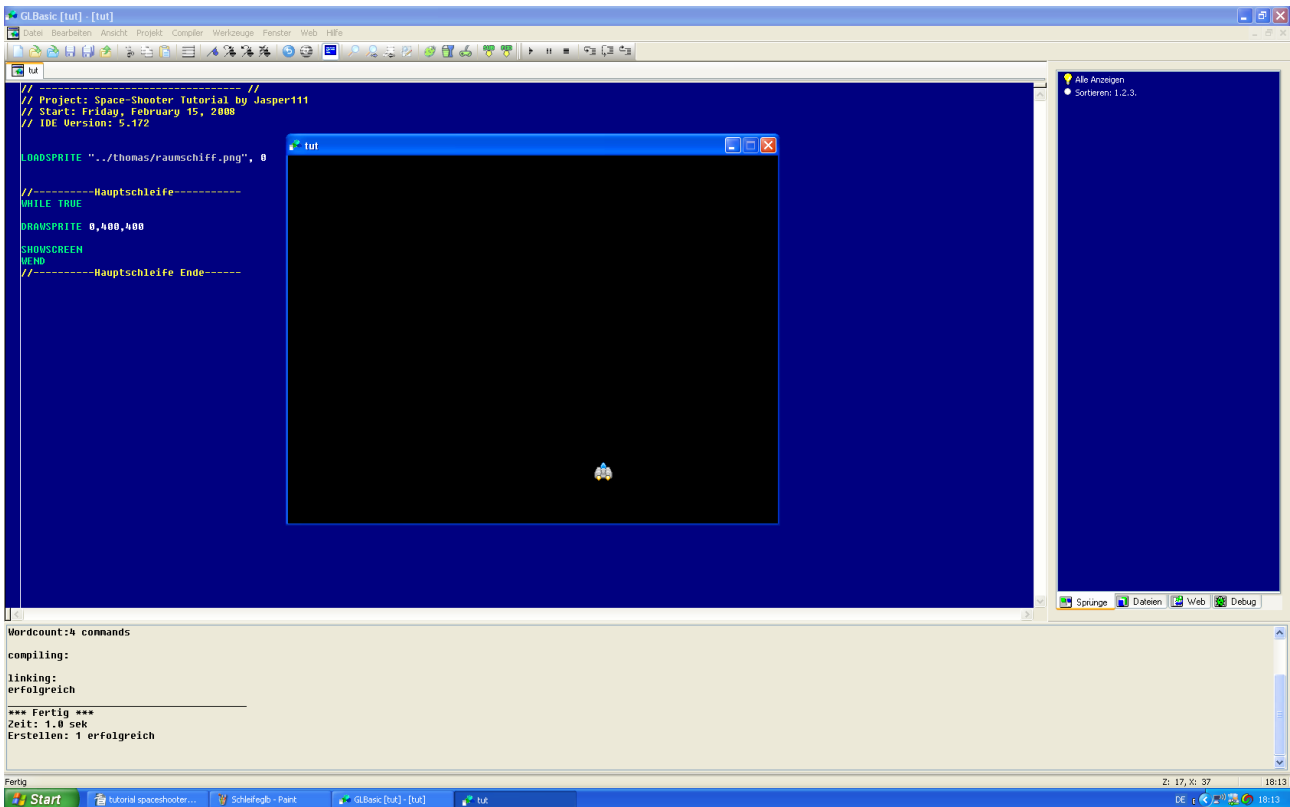
//-----Hauptschleife-----
WHILE TRUE

DRAWSPRITE 0,400,400 // <<< Bild auf den Bildschirm zeichnen, ID = 0 und dann
// das Bild auf 400 der x-Achse und 400 der y-Achse
// zeichnen.

SHOWSCREEN
WEND
//-----Hauptschleife Ende-----
```

(So sieht das ganze nach Betätigung der F5 Taste aus.)

Prima soweit. Aber wir wollen ja auch das Raumschiff nach links und rechts bewegen.



Nun gut, bringen wir unser Programm dazu, das Raumschiff zu bewegen.

Wir werden die Richtungstasten zur Steuerung benutzen.

Die Tasten sind **203** und **205**. Um die Codes zu ermitteln brauchen wir **WERKZEUGE** aus der oberen Leiste der **IDE** und gehen dann auf **KEYCODES**. Bei Betätigung einer Taste wird der Code angezeigt. **Richtungstaste links 203** und **Richtungstaste rechts 205**.

Um die Bewegung hin zu bekommen brauchen wir den Befehl **'IF KEY(x) THEN**.
x = KEYCODE .

IF = Wenn **KEY (203)** (Keycode = 203 = linke Richtungstaste)**THEN** **x-Achse = x-Achse - Wert**(Wert = Felder auf der x-Achse, um die das Raumschiff nach links bewegt wird).

Hier sollte auch die Möglichkeit mit **DEC** und **INC** erklärt werden.

Anstelle von **IF KEY(203) THEN playerx = playerx - 2** kann man auch mit **DEC** arbeiten.

DEC (heißt soviel wie minus) versetzt den **playerx** um **WERT** nach links, **INC** (heißt soviel wie plus) macht das selbe nach rechts.

IF KEY(203) THEN DEC **playerx, WERT** = die Zahl der Felder zum versetzen des Players nach links auf der **x-Achse**.

Dann würde der Code so aussehen:

```
IF KEY(203) THEN DEC playerx, 2
```

Das selbe gilt für **INC**, hier der Code:

```
IF KEY(205) THEN INC playerx, 2
```

Um das darstellen zu können, müssen wir die Positionen **x-Achse** sowie **y-Achse** von **DRAWSPRITE** in Variablen packen.

Für die **x-Achse** nehmen wir **playerx** und **y-Achse** dann **playery** (der Name der Variable ist jedem frei überlassen).

Die Variablen schreiben wir vor die **Hauptschleife**.

Die **If** Anweisung in die Hauptschleife, wir werden das Raumschiff jedes mal um 2 Pixel nach links oder rechts bewegen, je nach gedrückter Taste.

Hier der Code:

```
LOADSPRITE "../thomas/raumschiff.png", 0 //<<< Bild in den Speicher laden und eine ID vergeben
// (Hier habe ich die (ID =) 0 vergeben)

playerx = 400
playery = 400

//-----Hauptschleife-----
WHILE TRUE

IF KEY(203) THEN playerx = playerx -2
IF KEY(205) THEN playerx = playerx +2

DRAWSPRITE 0, playerx, playery // <<< Bild auf den Bildschirm zeichnen, ID = 0 und dann
// das Bild auf 400 x-Achse und 400 y-Achse
//zeichnen.

SHOWSCREEN
WEND
//-----Hauptschleife Ende-----
```

Jetzt können wir das Raumschiff bewegen, aber wenn wir zu weit in eine Richtung gehen fährt das Schiff aus dem Fenster. Das werden wir im nächsten Schritt verhindern.

Hier gibt es mehrere Möglichkeiten. Einmal mit einer **If**- Anweisung.

```
IF playerx > 600 THEN playerx = 600 //Hier wird geprüft ob das Raumschiff einen WERT von mehr als 600
//hat, wenn ja dann Raumschiff auf 600 setzen.
IF playerx < 0 THEN playerx = 0 // Hier der umgekehrte Fall, WERT kleiner 0 dann Raumschiff auf 0
// setzen.
```

Es geht aber auch einfacher, dazu benötigen wir neue Befehle.

Und hier sind die neuen Befehle **MAX** und **MIN**. **MAX** liefert den größeren Wert und **MIN** den kleineren Wert.

So sollte es dann aussehen:

```
      x-----x
>>> playerx = MAX(MIN(playerx, 600), 0) <<<
      x-----x
```

Bei diesen Befehlen setzt **MAX** den playerx auf **600** und **MIN** den playerx auf **0**.

Eingebaut wir der Code in die Hauptschleife vor **DRAWSPRITE**.

Hier wird das Raumschiff zwischen **0** und **600** auf der **x-Achse** in der Bewegung beschränkt.

So jetzt die F5 - Taste gedrückt und staunen. Hiermit haben wir diesen Teil abgearbeitet und gehen nun in die nächste Phase, der SCHUSS.

Schüsse erzeugen und bewegen

Als erstes laden wir die Grafik, direkt nach dem Raumschiff.

Erzeugen:

In diesem Teil werden wir **Type** kennen lernen. Ich bin fast daran verzweifelt, aber letztendlich hat mir dieser Teil den Ansporn gegeben diese Tutorium zu erstellen.

Nun gut, hier der Anfang:

```
//-----  
TYPE SCHUSS //SCHUSS mit Feld >>x<< und Feld >>y<< vormerken  
x //stellt eine Variable für die x-Achse bereit  
y //stellt eine Variable für die y-Achse bereit.  
ENDTYPE  
//-----
```

Hinweis: Das Ganze kommt vor die Hauptschleife

Mit **TYPE SCHUSS** (<< **Kann jeder benennen wie er will**)legen wir den Grundstein für die Schüsse. Mit X und Y merken wir 2 FELDER für Einträge vor.
ENDTYPE schließt die ganze Sache ab.

Das war noch der leichte Teil des Ganzen.

Jetzt aber weiter zum nächsten Schritt der TYPE's.

```
//-----  
LOCAL schuesse[] AS SCHUSS //schuesse[] als "Container des Types" (veränderbar) AS(als)  
//SCHUSS,schuesse[] ist ein Array und da da alles drin gespeichert wird,  
//nennen wir es einfach Container  
LOCAL schuss AS SCHUSS //(wir erinnern uns,SCHUSS wurde als TYPE SCHUSS hinterlegt,vor der  
Schleife)  
//-----
```

LOCAL bedeutet hier das die Variable nur hier zur Verfügung steht, im Gegensatz zu **GLOBAL** auf die von überall aus zugegriffen werden kann.

Weiter, jetzt treffen wir auf **AND(und)** und das sollte keiner Erklärung bedürfen.

Zur Erklärung:

GETTIMERALL() dagegen schon. **GET**=nehmen **TIMER**=Zeitnehmer **ALL**=ganz(im ganzen)()

Damit bekommen wir die absolute Zeit vom System seit Programmstart in 1/1000 Sekunden.

DIMPUSH fügt die Einträge in das Feld ein.

WICHTIG ist die Verschachtelung der Variablen zu durchblicken. Das versuch ich euch am Ende des Codes zu erklären.

Vorab der Code, die Erklärung findet ihr nach dem Code.

```
//-----  
IF KEY(57) AND warten<GETTIMERALL() //warten = eine Variable für die Zeit  
    schuss.y= 700 //Wenn Space(Leertaste) gedrückt dann den schuss.y  
                  //<-auf der y-Achse malen hier 400  
  
    schuss.x = playerx+10 //hier wird der schuss auf der x-Achse gemalt  
                          //Position +10 nach rechts  
                          //damit er in der Mitte der Grafik erscheint  
    DIMPUSH schuesse[], schuss //fügt ein neuen schuss ein in die Variable schuesse[]  
                                  //steht schuss als Wert der eingetragen wird in ->
```

```
//schuesse[] ( schuesse[] wurden ja "LOCAL"  
// als "SCHUSS" bezeichnet)
```

```
//Also zum Verständnis >>schuss<< wird in die Variable schuesse[] "+einen schuss" eingetragen, die wiederum  
//in den Type SCHUSS hinterlegt wird.
```

```
warten = GETTIMERALL()+200
```

```
ENDIF
```

```
//-----
```

Erklärung:

Diese Erklärung entstand ,dank Schranz0r der mir die Type – Geschichte mit Ordnern erklärt hat.

[So hat Schanz0r mir das erklärt, zu finden hier im Forum.](#)

Types kannst du dir so vorstellen:

Beim Arzt gibt es ein Formular, das man immer ausfüllen muss wenn man das erste Mal bei ihm ist, ich nenne es jetzt mal Standardformular, in GLBasic ist es so, dass das Standardformular die Declaration des Types und der Felder ist.

```
TYPE Patient  
  Name$  
  Alter  
  Geburtsjahr  
ENDTYPE
```

Jetzt braucht der Arzt aber noch einen Ordner, wo er die Standardformulare ablegen kann, die schon ausgefüllt worden sind.

(Ordner ist hier ein offenes Array)

```
LOCAL Patienten[] AS Patient
```

So, nun kommt ein Patient zum ersten mal zum Arzt, und die nette Arzthelferin(mit ordentlich Holz vor der hütt'n) fragt den Patienten:

Waren sie denn schon mal hier

NEIN, natürlich nicht, und deshalb muss man das Formular ausfüllen!

Da der Arzt alles schön sortiert haben will geht er nach Krankenkassen:

```
LOCAL AOK AS Patient  
AOK.Name$ = "Max Musterman"  
AOK.Alter = 30  
AOK.Geburtsjahr = 1977  
DIMPUSH Patienten[], AOK // Fügt das Formular in den Ordner ein
```

So jetzt will der Arzt natürlich mal alle durchgehen, um was zu prüfen:

```
LOCAL Anzahl  
FOREACH Check IN Patienten[]  
  IF Check.Alter = 30  
    INC Anzahl,1  
  ENDIF  
  PRINT "Es gibt "+Anzahl+" Patienten die 30 Jahre alt sind",10,10  
NEXT
```

Also nochmal zusammengefasst:

Alles bis auf die FOREACH-Schleife kommt vor der Hauptschleife, die FOREACH kommt in die Hauptschleife rein!

Vor der Hauptschleife haben wir mit:

```
TYPE SCHUSS
x
y
ENDTYPE
```

ein Formular erstellt was uns Felder vorgibt die wir ausfüllen müssen.

Beispiel:

FORMULAR SCHUSS

```
x :..... Hier schreiben wir den WERT der x-Achse rein
y :..... Hier schreiben wir den WERT der y-Achse rein
```

FORMULAR FERTIG

Um die **FORMULARE** nach dem Ausfüllen wieder zu finden legen wir die **FORMULARE** in einem **ORDNER** ab.

Den **ORDNER** nennen wir dann `schuesse[] >>>LOCAL schuesse[] AS schuss <<<`
`schuss` = Ausgefülltes **FORMULAR** im **ORDNER SCHUSS** mit den **WERTEN** `schuesse[] <<=` Eintragung abgeheftet.

`schuesse[]` = Eintragungen der offenen **FELDER** (welche wir eintragen) im **FORMULAR**

Der Befehl `DIMPUSH schuesse[]` , bedeutet quasi lochen und abheften ;-) im **Register schuss**

`schuesse[]` ist dann das ausgefüllte **FORMULAR** mit den **WERTEN**

`schuss` ist dann die Sortierung im **Register**

```
x :..... Hier stehen dann die eingetragenen WERTE
y :..... Hier stehen dann die eingetragenen WERTE
```

Damit haben wir jetzt ein **FORMULAR** mit **WERTEN** gefüllt und in einem **ORDNER** abgelegt.
Somit haben wir den Grundstein .

Mit **LOCAL schuss AS SCHUSS** haben wir jetzt die Möglichkeit geschaffen auch auf die abgehefteten ausgefüllten **FORMULARE** zu zugreifen.

Bewegen:

Hier überprüfen wir ob die Leertaste gedrückt wurde. **Überprüfung =IF KEY(57)** , wenn ja dann füllen wir das **FORMULAR** mit den aktuellen **WERTEN** aus.

```
schuss.y= 700           //<-auf der y-Achse malen hier 400

schuss.x = playerx+10   //hier wird der schuss auf der x-Achse gemalt
                        //Position +10 nach rechts
                        //damit er in der Mitte der Grafik erscheint
```

`DIMPUSH schuesse[], schuss` //bedeutet quasi lochen und abheften ;-) im **Register schuss**

Das ausgefüllte **FORMULAR** nennen wir dann `schuesse[] >>>LOCAL schuesse[] AS schuss <<<`
`schuss` = **Register** (Sortierung) der ausgefüllten **FORMULARE** im **ORDNER SCHUSS** mit den **WERTEN** `schuesse[] <<=` Eintragung abgeheftet.)

schuesse[] = Eintragung der **FELDER** (welche wir eintragen haben mit drücken der Leertaste) im **FORMULAR**

Der Befehl **DIMPUSH schuesse[]** , bedeutet quasi lochen und abheften ;-) im **Register schuss**

schuesse[] ist dann das ausgefüllte **FORMULAR** mit den **WERTEN**

schuss ist dann die Sortierung im **Register**

Zugriff auf die im **ORDNER** abgehefteten **FORMULARE**.

```
FOREACH laser IN schuesse[] // Jetzt nehmen wir ein ausgefülltes Formular schuesse[] und übergeben die
//WERTE an unseren laser <<=Bild
```

```
DEC laser.y, 2 // hier wird der laser.y, also die y-Achse pro durchlauf um 2 verringert (DEC =
//verringern),das heißt der laser geht 2 Felder auf y-Achse hoch
```

```
IF laser.y < 0 THEN DELETE laser //wenn der laser kleiner 0 auf der y-achse ist, bitte löschen
```

```
DRAWSPRITE 1,laser.x,laser.y //Naja und hier zeichnen wir den laser(BILD) auf den Bildschirm mit den
//WERTEN des FORMULARES.
```

```
NEXT // das nächste // Bei jedem durchlaufen wir unser laser durch DEC um 2 Felder nach oben
//versetzt.
```

Hier der gesamte Code:

```
LOADSPRITE "../thomas/raumschiff.png", 0
```

```
LOADSPRITE "../..../Desktop/spaceshooter/space shooter/gfx/rakete.png", 1
```

```
playerx = 400
playery = 400
```

```
TYPE SCHUSS //SCHUSS mit feld >>x<< und Feld >>y<< vormerken
x
y
ENDTYPE
```

```
LOCAL schuesse[] AS SCHUSS //schuesse[] als variable(veränderbar) AS(als) SCHUSS
```

```
//-----Hauptschleife-----
WHILE TRUE
```

```
IF KEY(203) THEN playerx = playerx -2
IF KEY(205) THEN playerx = playerx +2
```

```
LOCAL schuss AS SCHUSS //nach dem Abschuss wird der schuss als Neuer SCHUSS der wiederum als variable
schuesse[]
```

```
IF KEY(57) AND warten<GETTIMERALL()
```

```
schuss.y= 400
schuss.x = playerx+5
```

```
DIMPUSH schuesse[], schuss
```

```

        warten = GETTIMERALL()+200
ENDIF

FOREACH laser IN schuesse[]

    DEC laser.y, 2
    IF laser.y < 0 THEN DELETE laser
    DRAWSPRITE 1,laser.x,laser.y
NEXT
DRAWSPRITE 0, playerx, playery
SHOWSCREEN
WEND
//-----Hauptschleife Ende-----

```

Nach Betätigung der F5 – Taste können wir jetzt verschiedene Schüsse erstellen.

Gegner:

In diesem Teil bauen wir den Gegner ein und machen am Schluss eine Treffer-Abfrage.

```
LOADSPRITE "../thomas/asteroid.png", 2
```

mit dem oberen Code laden wir die Grafik für unseren Gegner (hier ein Asteroid) in den Speicher. Wichtig ist das wir nicht vergessen der Grafik eine ID zugeben.

Vor der Hauptschleife bauen wir, wie vorher schon mit den Schüssen den Type für die Gegner ein.

```

TYPE GEGNER
x
y
ENDTYPE

```

Da wir einen einfachen Shooter bauen reichen uns hier die x-Achse und die y-Achse, da unser Gegner nicht schießen kann.

```
LOCAL gegner[] AS GEGNER //gegner[] als variable("Container des Type's") veränderbar AS(als) GEGNER
```

Das ganze funktioniert genau wie die Schuss Erstellung. Lest euch einfach nochmal den Teil der Type's durch.

Mit **LOCAL gegnerneu AS GEGNER**, haben wir jetzt die Möglichkeit geschaffen auch auf die abgehefteten und ausgefüllten **FORMULARE**, für den **GEGNER** zu zugreifen. Die wir mit **gegnerneu** im weiteren Verlauf noch mit Daten füllen.

```

LOCAL gegnerneu AS GEGNER

    zeitgegner = zeitgegner +1
    IF zeitgegner = 25
    gegnerneu.x = 0
    gegnerneu.y = RND(100)
    DIMPUSH gegner[], gegnerneu

    zeitgegner = 0
ENDIF

```

Gegnerneu ist unser Register mit den gefüllten **Formularen**, auf das wir jetzt jederzeit wieder zugreifen können.

`Zeitgegner = zeitgegner + 1` ist eine Warteschleife die bei jedem Durchlauf um einen Zähler erhöht wird und wenn der Zähler 25 ist, dann erstellen wir einen neuen **GEGNER**.

`gegnerneu.x = 0` << gibt die **x-Achse** für unseren neuen **Gegner** an, die wir hier mit den **WERT 0** füllen.

`Gegnerneu.y = RND(100)` << gibt die **y-Achse** von unserem **Gegner** an. Die wir per Zufall (`RND(100)`) erzeugen.

`RND (100)` << **RND** = Random und der **WERT** in den Klammern (**100**) gibt uns die Möglichkeit eine Zahl die zufällig zwischen 0 und 100 erzeugt wurde unserer **y-Achse** zu zuweisen.

Um die Werte dann zu übergeben brauchen wir wieder `DIMPUSH gegner[]`, `gegnerneu` somit ist der **GEGNER** entstanden.

Um einen weiteren **GEGNER** zu erstellen müssen wir noch den **Zähler auf 0** setzen um dann wieder bis **25** zählen zu können.

Das machen wir mit `zeitgegner = 0`

Am Ende schließen wir die **If – Anweisung** mit **ENDIF** ab.

Mit der **FOREACH** (Für alle/jeden) schreiben wir bei jedem Durchlauf die neuen WERTE der **y-Achse** und **x-Achse** in unser **Formular gegner[]**.

```
FOREACH aster IN gegner[] //Heißt soviel wie, "Für jeden aster in gegner[]
  INC aster.x,2
  aster.y = aster.y
  IF aster.x > xx THEN DELETE aster
```

Hinweis: Mit FOREACH wird jeder Gegner der schon erzeugt ist angesprochen und um die Werte erhöht.

`INC` versetzt unsere **x-Achse** bei jedem Durchlauf um **2** nach rechts.

```
          x-----der Wert um den wir nach rechts versetzen
INC aster.x, 2
          x-----x-Achse des Asteroiden
```

```
IF aster.x > xx THEN DELETE aster
```

Mit der **IF – Anweisung** prüfen wir ob der Asteroid (**GEGNER**) den Bildschirmrand erreicht hat und löschen ihn dann.

```
IF aster.x > xx THEN DELETE aster
```

(Wenn) `IF aster.x (größer) > (Bildschirmrand) xx` Then(dann) `DELETE(lösche) aster`

Jetzt können wir den **GEGNER** auf dem Bildschirm zeichnen und wenn der Zähler 25 erreicht hat dann den nächsten **GEGNER** und so weiter. Dafür brauchen wir wieder `DRAWSPRITE`.

```
DRAWSPRITE 2, aster.x, aster.y
          X die 2 gibt unsere SPRITE ID an, die wir beim laden der Grafik vergeben haben.
```

Hier der ganze **Code** (**Neuer Code ist braun gefärbt**) :

```
LOADSPRITE "../thomas/raumschiff.png", 0
```

```
LOADSPRITE "../././Desktop/spaceshooter/space shooter/gfx/rakete.png", 1
```

```
LOADSPRITE "../thomas/asteroid.png", 2 // unsere ID ist hier die 2
```

```
playerx = 400
```

```
playery = 400
```

```

TYPE SCHUSS //SCHUSS mit feld >>x<< und Feld >>y<< vormerken
x
y
ENDTYPE

TYPE GEGNER
x
y
ENDTYPE

LOCAL schuesse[] AS SCHUSS //schuesse[] als variable(veränderbar) AS(als) SCHUSS

LOCAL gegner[] AS GEGNER

//-----Hauptschleife-----
WHILE TRUE

IF KEY(203) THEN playerx = playerx -2
IF KEY(205) THEN playerx = playerx +2

LOCAL schuss AS SCHUSS //nach dem Abschuss wird der schuss als Neuer SCHUSS der wiederum als variable
schuesse[]

IF KEY(57) AND warten<GETTIMERALL()

    schuss.y= 400
    schuss.x = playerx+5

    DIMPUSH schuesse[], schuss

    warten = GETTIMERALL()+200
ENDIF
FOREACH laser IN schuesse[]

    DEC laser.y, 2
    IF laser.y < 0 THEN DELETE laser
    DRAWSPRITE 1,laser.x,laser.y
NEXT

LOCAL gegnerneu AS GEGNER
    zeitgegner = zeitgegner +1
    IF zeitgegner = 25
    gegnerneu.x = 0
    gegnerneu.y = RND(100)
    DIMPUSH gegner[], gegnerneu

    zeitgegner = 0
ENDIF

FOREACH aster IN gegner[]
    INC aster.x,2
    aster.y = aster.y
    IF aster.x > 660 THEN DELETE aster

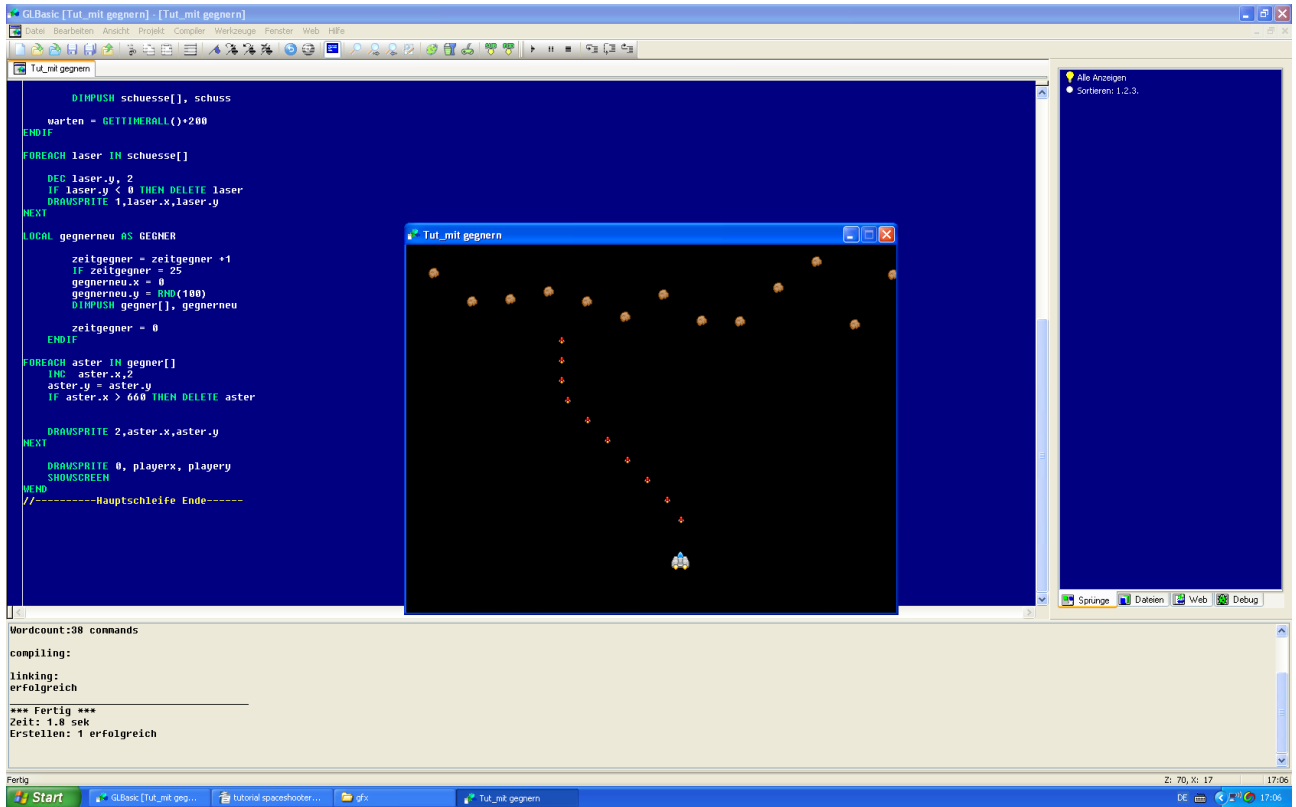
    DRAWSPRITE 2,aster.x,aster.y
NEXT

    DRAWSPRITE 0, playerx, playery
    SHOWSCREEN

WEND
//-----Hauptschleife Ende-----

```

Jetzt mal schnell F5 – Taste drücken und sehen was passiert.



Jetzt sehen wir schon unser Raumschiff, Schuesse und Gegner.

Jetzt noch Treffer abfragen und wir haben einen kleinen Space-Shooter.