

Von Anfang an...

LAMA NETWORK



V.1.04

LamaNet By Chrise 2010

Hallo lieber LamaNet Benutzer,

Damit ich dir einiges bei dem Einstieg in das Programmieren mit der umfangreichen Funktionssammlung erleichtern kann, findest du in dieser Dokumentation alle nützlichen Tipps, die du dazu brauchst um dein eigenes Netzwerkspiel zu erstellen.

Ich hoffe du findest hier alles was du brauchst.
Sollte ich dennoch vergessen haben eine Erklärung oder einen Befehl hinzuzufügen, dann lass es mich doch bitte wissen.
Ich werde so schnell wie möglich versuchen die Dokumentation oder die Funktionssammlung um die Befehle zu ergänzen und zur

Verfügung zu stellen. Erreichen kannst du mich unter

mylama@yahoo.de

ich freue mich über jedes Feedback :)

So genug geredet, du bist schließlich hier, um zu lernen wie LamaNet funktioniert!



Was benötige ich?

Zuerst einmal benötigst du die LamaNet-Funktionssammlung und alle zugehörigen Dateien. In der Version 1.04 sind es bisher nur folgende Dateien die du benötigst:

```
LamaNet.bb  
LamaNet.decls
```

Für Einsteiger und Neulinge

Die LamaNet.decls ist dringend notwendig, um das Netzwerkspiel im Blitzbasic-Debugger zu kompilieren. Ohne decls, funktioniert es nicht. Also schnell in den richtigen userlib Ordner im Blitzbasic-Verzeichnis. Wenn Blitz3D richtig installiert wurde, liegen die .decls-Dateien in folgendem Verzeichnis:



Blitz2D
Blitz3D

C:\Programme\Blitz2D\userlibs
C:\Programme\Blitz3D\userlibs

Die LAmalNet.bb kopierst du am besten dorthin, wo du sie bequem in den Projekt einbinden kannst. Das solltest du dringend tun, denn davor wirst du keine der Funktionen in der Funktionssammlung benutzen können.

Von Anfang an...

IDEal

Für all diejenigen, die den Editor „IDEal“ verwenden, aber noch nie eine userlib eingebunden haben, hier noch eine kurze Anleitung, wie man die userlibs richtig verwendet.
Öffne das Menü

Compile>Edit Userlibs>

Sofern die decls-Datei in das richtige Verzeichnis kopiert wurde, findest du nun im Userlib-menü die durch ein Buch gekennzeichnete Funktionssammlung „LamaNet“. Aktiviere die Userlib links neben dem Buch. Ein Häkchen erscheint. Drücke anschließend:

Refresh Definitions

Nun dauert es einen kurzen Augenblick, bis folgende Nachricht erscheint:

Keyword definitions for this compiler successfully created!

Drücke auf okay. Die Userlib sollte nun korrekt installiert und verwendbar sein. Anschließend kannst du das Userlib-Menü wieder schließen.

So, nun werde ich nacheinander beginnen dich in die Welt von LamaNet einzuführen. Ich werde Schritt für Schritt die notwendigen Funktionen erklären und in Beispielen mit einbinden. Aber keine Angst. Am Ende der .PDF-Datei findest du eine große Liste von allen Funktionen die es in LamaNet insgesamt gibt. Verwende am besten die Suchfunktion des Acrobatreaders um die gewünschte Funktion schnell zu finden.

Von Servern und Netzwerken...



Zuerst sollte man natürlich einen Plan davon haben, was man verwirklichen will und vor allem auch kann. Zweiteres ist meist sogar Projekt-entscheidender als der erste Punkt. Ohne wenigstens einen groben Plan im Kopf zu haben, kann man kein Projekt erstellen. Darum gebe ich in diesem Tutorial ein Beispiel vor, in dem alle Schritte ausführlich erklärt werden.

Mein Beispiel wird klassischerweise „Pong“ lauten. Es beinhaltet einige Eigenschaften, mit deren Hilfe ich die Funktionen einführen kann. Außerdem werden keine externen Grafiken oder Skripte benötigt. Die Technik kann bei Bedarf auch einfach gehalten werden.

Nun gut, erstelle für das Tutorial am besten einen eigenen Projektordner. Wie du diesen nennst bleibt völlig dir überlassen, Hauptsache du findest ihn wieder, wenn du eine Pause einlegst ;)

Natürlich müssen wir zwischendurch immer wieder kleine klassische Befehle einfügen, damit das Programm nach unseren Wünschen arbeitet. So bietet sich an erst einmal den Grafikmodus zu initialisieren. Zusätzlich können wir gleich den Backbuffer setzen. Ich nehme bewusst ein Quadrat, da später jeder Spieler ja gleiche Chancen bekommen soll:

```
Graphics 480,480,0,2  
SetBuffer BackBuffer()
```

LamaNet.bb

Und jetzt kommt eigentlich der allerwichtigste Teil unseres Netzwerkspiels.

Sofern du es nicht sowieso schon erledigt hast, kopiere nun die LamaNet.bb in den Projektordner und füge folgende Zeile deinem Projekt hinzu:

```
Include "LamaNet.bb"
```

Dein Programm hat somit eigentlich alles auf einem Schlag, was für das spielen im Netzwerk nötig ist. Ab sofort werden alle LAMA_ Befehle wirksam!

Server oder Client?

Server und Client benutzen zwar im groben und ganzen die selben LAMA_ Funktionen, dennoch muss man sich die Frage stellen, welchen Part das Programm vertreten muss. Am einfachsten ist es, zu Beginn eines Netzwerkspiels (egal ob Erstellung oder Beitritt) den Spieler entscheiden zu lassen, welchen Part das Programm vertritt.

Am einfachsten geht das natürlich mit einem Input-Befehl. Erfahrene Programmierer werden aber zu einer schöneren Variante greifen. Für unser Beispiel genügt ein Input allerdings völlig.

```
Local ServerOrClient=Input("Server erstellen(1) - Server suchen(2)")
```



Das Programm wird den Spieler fragen, ob er vorhat einem bestehenden Spiel im Netzwerk beizutreten oder selbst eines zu erstellen.

Nun kommen wir zur ersten Selektion. Das Programm entscheidet nun anhand der Variable „ServerOrClient“, ob er ein Spiel startet oder nicht.

```
Select ServerOrClient
  Case 1
    LAMA_HostServer("Pongserver",4,"Lama")
  Default
    End
End Select
```

Wie ihr seht, habe ich die Funktion LAMA_HostServer schon ausgefüllt. Natürlich kann man auch Variablen oder andere Werte übergeben.

```
1. Servername$:      Mein Server wird PongServer heißen.
2. MaxPlayers$:      Es Server werden maximal 4 Spieler spielen können.
3. Name$:            Der Spieler, der das Spiel erstellt hat, wird „Lama“
heißen.
```

Wenn der Spieler scherzhafterweise etwas völlig anderes eingibt, als erlaubt, wird das Spiel einfach geschlossen ;)

Zunächst findest du den Code, so wie er bisher aussehen sollte. Stimmt er nicht mit meinem Beispiel überein, hast du womöglich einen Fehler gemacht, oder baust schon längst an Extraeingaben für Namen, Spieleranzahl, usw. :P

Dein Programm sollte bisher so aussehen
und leider bisher noch nicht all zu viel können:



```
Graphics 640,480,0,2
SetBuffer BackBuffer()

Include "LamaNet.bb"

Local ServerOrClient=Input("Server erstellen(1) - Server suchen(2)")
Select ServerOrClient
  Case 1
    LAMA_HostServer("Pongserver",4,"Lama")
  Default
    End
End Select
```

Serverlist



Wir werden eine dynamische Serverlist erstellen. D.h. Das Spiel des Clients läuft weiter und die Serverlist erweitert sich jenachdem wann und wie viele Server auf dessen Anfrage antworten. Dazu benötigen wir erst einmal die allgemeine Grundlage für ein BB-Programm mit Mainloop, Doublebuffering und Frame-begrenzung:

```
Local Timer=CreateTimer(60)
While Not KeyHit(1)
    Cls

    WaitTimer Timer
    Flip 0
Wend
End
```

Updating

Nun kommt ein weiterer wichtiger Punkt. Ich möchte mal behaupten, dies sei der zweit-wichtigste Punkt nach „Include“. Die Funktion LAMA_Update() aktualisiert alles, was in der LamaNet-Engine passiert. „LAMA_Update“ verschickt Nachrichten, Pings, Dateien. Es kickt und bannt Spieler und sorgt dafür, dass die Nachrichten beim richtigen Empfänger ankommen. Natürlich geschieht einiges nur im Auftrag, also wenn ein Spieler eine andere Funktion ausführt. Aber LAMA_Update() ist letzten Endes die Funktion die die Arbeit erledigt. Diese Funktion muss im Main-loop ausgeführt werden, damit ein Netzwerkspiel funktionieren kann.

```
LAMA_Update()
```

Füge diesen Befehl in die Hauptschleife ein und dann kann es eigentlich schon mit allen Spielinternen Funktionen beginnen! :)

LAMA_StartCountServers()

... diese Funktion wird benötigt, wenn du eine Serverlist erstellen willst. Diese Funktion ermöglicht dir das Suchen von Servern im eigenen Netzwerk. Wie diese Funktion schon darauf deuten lässt, benötigt man sie, um die Suche von Servern zu beginnen.

Das Gegenstück zu dieser Funktion lautet LAMA_StopCountServers(). Sie stoppt die Suche nach Servern wieder. LAMA_SendCountRequest() startet die Suche von neuem. Sie wird benötigt, um die Serverlist zu aktualisieren, wird aber erst wirksam, wenn man zuvor LAMA_StartCountServers() verwendet hat.

Sicherlich erinnerst du dich noch daran, dass der Spieler zuvor entscheiden konnte, ob er ein Spiel erstellen wollte, oder einem bestehenden Spiel beitreten wollte. Auf diesen Teil kommen wir nun wieder zurück:



```
Select ServerOrClient
  Case 1
    LAMA_HostServer("Pongserver",4,"Lama")
  Case 2
    LAMA_StartCountServers()
  Default
    End
End Select
```

Wie du siehst, habe ich einfach einen neuen Fall eingefügt und lasse bei der Eingabe „2“ das Programm starten, die Server zu suchen, die sich im Netzwerk aufhalten.

Online oder Offline?

Woher weiß das Programm nun eigentlich, ob es in einem bestehenden Netzwerk eingeloggt ist, oder nicht? Dazu gibt es eine Abfragemöglichkeit mittels einer globalen Variablen namens „LAMA_Connection“. Sobald LAMA_Connection den Wert „1“ annimmt, besteht eine Verbindung zu einem Server im Netzwerk. Solange sie allerdings „0“ ist, befindet sich das Programm im Offline-modus.

Wofür das ganze nun für die Serverlist wichtig ist?

Ganz einfach: Solange das Programm offline ist, wird es eine Serverlist erstellen und darstellen.

```
If LAMA_Connection Then
  ;Spielinternes
Else
  ;Serverlist
EndIf
```

Natürlich muss dieser Code auch in den Main-loop. Schließlich sollte das Programm zu jeder Zeit wissen, ob es verbunden ist oder nicht.

Nach „Else“ wird nun die Serverlist entstehen.

Bei Erfüllung der Bedingung wird alles stehen, was mit dem Spiel selbst zu tun hat ;)

LAMA_CountServers

Oh, schon wieder so eine globale Variable! Welchen Wert diese Variable enthält ist wohl offensichtlich. Sie speichert immer die aktuelle Anzahl an Servern die im Netzwerk gefunden wurden. Sie ist notwendig um eine schöne Liste zu erstellen, in welcher man die Server auswählen kann.

Wenn ich hier schon von einer Variable spreche, die eine Anzahl speichert, könnt ihr euch sicher denken, dass das ganze auf eine „For/Next“ Schleife hinauslaufen wird ;)

Und das wird es auch! Diese Schleife kommt jetzt nach Else in die Bedingung.



```
For i=1 To LAMA_CountServers  
Next
```

Das Programm wird nun für jeden gefundenen Server die Schleife abarbeiten.
Nun gibt es bei LamaNet einige Funktionen die Informationen zu den Servern zurückgeben.
Hier zeige ich dir einmal die Liste aller Serverinformationen, die zurückgegeben werden können:

```
LAMA_ServerName$ ([ServerID%])  
LAMA_ServerPing% ([ServerID%])  
LAMA_ServerPlayers% ([ServerID%])  
LAMA_ServerMaxPlayers% ([ServerID%])  
LAMA_ServerLocked% ([ServerID%])  
LAMA_ServerPassword% ([ServerID%])  
LAMA_ServerBannedFrom% ([ServerID%])  
LAMA_ServerPublisher$ ([ServerID%])  
LAMA_ServerInformation$ ([ServerID%] [,InfoID% von 1-10])
```

Wird die Angabe der ServerID weggelassen, so wird der Wert des Servers zurückgegeben, mit dem das Programm verbunden ist. Außerdem kann jeder Server insgesamt 10 zusätzliche Informationen speichern, welche mit InfoID% jeweils abgefragt werden können.

Da wir in unserem Beispiel „Serverlist“ aber die ID des Servers wissen, können wir ihn auch Anhand von „i“ angeben.

Um beispielsweise nur die Namen der gefundenen Server untereinander aufzulisten, könnte unser Code so aussehen:

```
For i=1 To LAMA_CountServers  
Text 0, (i*15)-15, LAMA_ServerName$(i)  
Next
```

Da wir aber nicht nur die Namen anzeigen lassen wollen, sondern auch die Spieleranzahl und die maximale Anzahl an Spielern, die sich in dem Spiel aufhalten dürfen, bauen wir den Inhalt der kleinen Schleife noch etwas um:

```
For i=1 To LAMA_CountServers  
Text 0, (i*15)-15, LAMA_ServerName(i)+"- (" + LAMA_ServerPlayers(i)  
+" / "+ LAMA_ServerMaxPlayers(i) + ") "  
Next
```

Hier in der PDF Datei passt der Code leider nicht mehr in eine Zeile. Solltest du das ganze allerdings in den Editor kopieren, dürfte es wieder stimmen ;)

Überblick



Hier schreibe ich nochmal den ganzen Code, so wie er bisher aussehen sollte:



```
Graphics 480,480,0,2
SetBuffer BackBuffer()

Include "LamaNet.bb"

Local ServerOrClient=Input("Server erstellen(1) - Server suchen(2)")
Select ServerOrClient
    Case 1
        LAMA_HostServer("Pongserver",4,"Lama")
    Case 2
        LAMA_StartCountServers()
    Default
        End
End Select

Local Timer=CreateTimer(60)
While Not KeyHit(1)
    LAMA_Update()
    Cls

    If LAMA_Connection Then
    Else
        For i=1 To LAMA_CountServers
            Text 0,(i*15)-15,LAMA_ServerName(i)+" -
("+LAMA_ServerPlayers(i)+"/"+LAMA_ServerMaxPlayers(i)+") "
        Next
    Endif

    WaitTimer Timer
    Flip 0
Wend
End
```

Nun haben wir schon: Eine Entscheidung des Spielers, ob Server oder Client, eine vereinfachte Serverlist im laufenden Spiel und eine Unterscheidung, ob das Programm in einem Server eingeloggt ist oder nicht. Das ist doch schon einmal ein guter Anfang!

Wenn du das Programm als .Exe-Datei abspeicherst, öffnest, einen Server erstellst und das selbe nur diesmal mit Serverbeitreten machst, wirst du beim Client-Fenster feststellen, dass nun in der linken oberen Ecke „**PongServer – (1/4)**“ steht. Wenn das da Fall ist, hast du bisher alles richtig gemacht!

Wir werden unsere vereinfachte Serverlist noch um ein kleines bisschen erweitern. Schließlich wollen wir den Server auswählen können, dem wir beitreten wollen.

Dazu bauen wir uns am besten mit Rect einen Rahmen, dessen Farbe sich bei MouseOver verändert. Das kommt mit in die Schleife:

```
Rect 0,(i*15)-15,GraphicsWidth()/2,15,0
```

Wenn man den Versuch wiederholt, sieht man nun ein nettes weißes Rechteck. Nett. :P Das lassen wir nun ersteinmal so. Nun brauchen wir noch die Farbänderung bei Mouseover.



Wow! Der Mouseover – Effekt!

Naja, so großartig, wie ich hier gerade tue, ist das ganze eigentlich gar nicht ;)

Schreibt über den „**Text und Rect**“ Befehlen den Befehl „**Rectsoverlap**“, kopiert die Daten (ausser Füllung) aus dem Rect-Befehl in die Klammern und fügt noch „**MouseX(),MouseY(),1,1**“ hinzu, dann noch der „**Color**“ Befehl und fertig!

```

    If Rectsoverlap(0, (i*15)-15,GraphicsWidth()/2,15,MouseX(),MouseY(),1,1)
Then
    Color 172,189,237
Else
    Color 255,255,255
Endif

```

Schon wird Text und Rechteck bei Mausberührung in einer anderen Farbe angezeigt (in meinem Beispiel das typische Lama-Blau ;))

Jetzt müssen wir nur noch entscheiden, ob die Maustaste gedrückt wurde, oder nicht, dann können wir einbauen, dass wir einem bestehenden Spiel auch beitreten können.

MouseHit

Natürlich könnten wir einfach Mousehit(1) machen. Wenn du willst kannst du es einfach nehmen, jedoch kann es vorkommen, dass dann ein Mouseklick registriert wird, obwohl du nur auf die leere Fläche neben den Button geklickt hast und dann mit der Maus über den Button gekommen bist. Ich verwende deshalb immer eine extra Variable, in der im Main-loop Mousehit() angewandt wird. Ich nenne sie einfach „Mhit“ und führe am Anfang des Mainloops folgende Zeile aus:

```
Hit=Mousehit(1)
```

Damit können wir nun in unserer Rectoverlap Abfrage weiterarbeiten. Jetzt wollen wir dem Server beitreten:



```

    If Rectsoverlap(0, (i*15)-15,GraphicsWidth()/2,15,MouseX(),MouseY(),1,1)
Then
    Color 172,189,237
    If MHit Then
        LAMA_JoinServer(i,"JoinLama")
    EndIf
Else
    Color 255,255,255
Endif

```

Ich habe ersteinmal den Namen „JoinLama“ gewählt, da ich später sowieso noch vor habe, eine



Namenseingabe zu machen. Für den Test sollte dieser Name allerdings reichen.
„i“ ist wieder die Index-Nummer des aktuellen Servers in der For/Next-Schleife.

Wenn du jetzt das Programm wieder zweimal in unterschiedlichem Modus ausführst, wirst du sehen, dass die „Serverlist“ (bestehend aus einem/deinem eigenen Server ;)) verschwindet, sobald du den Server anklickt.



Das ganze hat folgenden Grund: Sobald du dem Server beitriffst, ändert sich die Variable LAMA_Connection von „0“ auf „1“ und das Programm ist so mit dem Server verbunden.
→ **die Serverlist wird nicht mehr gezeichnet.**

Spielersystem 1.0

Jetzt wird es aber endlich mal Zeit, um mit dem Spielersystem anzufangen. Am einfachsten löst man das System, indem man jedem Spieler einen Typeeintrag gibt und jeden „Handle“-Wert dieser Typeeinträge in einem „Dim“-Feld abspeichert.

Somit kann man die Typeeinträge per Index-Nummer aufrufen und die Variablen verändern.

Erstelle ein Typefeld, welches folgende Variablen enthält:

```
ID%, X%, Y%, Score%
```

Davon ist „ID“ die wohl wahrscheinlich wichtigste Variable (Praline) des Programms. ;) Mit ihr kann man später auf Variablen und andere Spielinterne Dateien zurückgreifen. Besonders für komplexere Spiele ist das wichtig.

„X,Y und Score%“ habe ich nur für das Pong Beispiel gewählt. Für andere Spiele sind sie nicht notwendig. X,Y werden später nur die Schläger-position beinhalten. Score% wird die Spielstände mitspeichern.

In Ausführung sieht das schließlich so aus und lässt sich bereits in den Code einfügen.



```
Type Player  
    Field ID%, X%, Y%, Score%  
End Type
```

Handles

Jetzt müssen wir eigentlich nur noch die Type-felder erstellen und die Handles in ein Dim-Feld abspeichern.

Gut ist es, wenn wir das kurz vor oder nach der Frage erledigen, ob der Spieler ein Spiel erstellen oder einem beitreten will. Ich lasse bei dem Dim-Feld bewusst den 0-ten Platz unbelegt, weil dadurch einiges vereinfacht wird. Schließlich gehen bei diesem Feld ohnehin nur 4 Bytes insgesamt verloren.





```
Dim Player(4);für maximal 4 Spieler ausgelegt
Local P.Player
For i=1 to 4
    P.Player=New Player
    P\ID=i
    Player(i)=Handle(P.Player)
Next
```

Für eingefahrene Programmierer sollte dieser Code-fetzen selbstverständlich sein. Für Neulinge noch einmal kurz erklärt.

Anfangs wird ein Dim-Feld mit eigentlich 5 Plätzen erzeugt. Wir werden einfachheitshalber nur 4 davon benutzen. Schließlich erstellen wir für jeden Spieler einen Type-Eintrag und speichern auch gleich die ID mit. Danach speichern wir den Eintrag per Handle in dem Dim-Feld ab und können den Typeeintrag schnell per ID wieder aufrufen.

Umgekehrt geht es genauso: Solange wir einen aktiven Type-eintrag haben, wissen wir anhand der im Type gespeicherten ID, um welchen Spieler es sich gerade handelt.

Spieler 1-4

Zuerst wollen wir uns ein wenig mit dem Spielersystem ansich beschäftigen. Es gibt darin 2 wichtige Variablen.

Einmal „LAMA_MasterID“ und einmal „LAMA_From“.

Die MasterID ist die eigene ID des Programmes. So weiß jedes Programm automatisch welche SpielerID es besitzt und kann somit auch seinen eigenen Type-Eintrag verwalten.

Die ID die in LAMA_From eingespeichert wird ist die ID des Spielers, die dem aktuellen Programm eine Nachricht gesendet hat. Erhält ein Programm also eine Nachricht, ist in LAMA_From immer die ID des Absenders gespeichert.

Das Programm soll nun aber zunächst den Schläger des eigenen Spielers platzieren.

Dazu kommt in die Connection-Abfrage (die bis jetzt übrigens noch leer sein sollte) folgendes:



```
P.Player=Object.Player(Player(LAMA_MasterID))
```

Der eigene Type-Eintrag des Programms wird per eigener ID aufgerufen und kann ab nun bearbeitet werden.

Spielerkoordinaten Unterscheidung

Da sich die Spieler nicht alle am selben Platz befinden sollen, bauen wir an dieser Stelle noch eine Unterscheidung aus. Dabei verwenden wir wieder den „Select“ befehl und arbeiten die 4 Fälle ab:

```
Select P\ID
```



```

Case 1
    P\X=10
    P\Y=MouseY()
Case 2
    P\X=Graphicswidth()-10
    P\Y=MouseY()
Case 3
    P\X=MouseX()
    P\Y=10
Case 4
    P\X=MouseX()
    P\Y=Graphicsheight()-10
End Select

```

Sobald nun 4 Spieler gleichzeitig im Spiel online sind, wird sich nun an jeder Kante des Fensters ein Spieler befinden. Bisher eben nur unsichtbar, aber das wird sich nun ändern.

Spielerdarstellung

Um die Spieler nun richtig darstellen zu können, müssen wir nun für jeden Spieler eine Zeichenroutine durchführen. Das geht am besten mit einer For/Each-Schleife.

Ich hoffe du merkst allmählich warum ich in den Netzwerk-spielen lieber auf Types zurückgreife, als auf Dim-Felder. Sie sind einfach viel flexibler und in solchen Beispielen um einiges leichter zu bedienen.

Die Spielerdarstellung wird auch in die Connection-Abfrage gesteckt, da sie ja nur passieren soll, wenn man mit einem Server verbunden ist oder eben aktuell einen Server geöffnet hat.

Wir werden wieder eine „Select“ Abfrage benötigen, da die Schläger einmal Waagrecht und einmal Senkrecht stehen werden. Das geht aber nicht schwieriger bzw. leichter als die Selektierung zuvor ;)

```

Color 255,255,255
For P.Player=Each Player
    Select P\ID
        Case 1
            Rect P\X-1,P\Y-50,3,101
        Case 2
            Rect P\X-1,P\Y-50,3,101
        Case 3
            Rect P\X-50,P\Y-1,101,3
        Case 4
            Rect P\X-50,P\Y-1,101,3
    End select
Next

```



Spieler online?!



Woher soll man nun wissen, ob ein Spieler online ist und überhaupt dargestellt werden soll?

Hier kommt wieder so eine wichtige Funktion ins Spiel. Sie heißt:

LAMA_Online(PlayerID%) und gibt auf Wunsch die Information zurück, ob der Spieler nun online oder offline ist. Wir werden die Select-Abfrage von gerade eben mit dieser Variable erweitern und blenden somit alle nicht mit spielenden Spieler einfach aus ;)

```
Color 255,255,255
For P.Player=Each Player
    If LAMA_Online(P\ID) Then
        Select P\ID
            Case 1
                Rect P\X-1,P\Y-50,3,101
            Case 2
                Rect P\X-1,P\Y-50,3,101
            Case 3
                Rect P\X-50,P\Y-1,101,3
            Case 4
                Rect P\X-50,P\Y-1,101,3
        End select
    Endif
Next
```

Wenn du willst, kannst du das Programm ja einmal auf Modus 1 laufen lassen. Du wirst sehen, du bist der einzige Balken, der auf dem Bildschirm zu sehen ist. Und sobald du mit einem zweiten Programm dem Spiel beitriffst, wird aus einem Balken zwei.

... doch Leider werden sich die Balken des jeweils anderen Spieler noch nicht bewegen...

Synchronizität – oder Senden/Empfangen

Doch dazu kommen wir jetzt!

Jetzt wird es spannend, denn die Basis, auf dem das Netzwerkspiel funktioniert, ist bereits fertiggestellt. Jetzt muss nur noch fleißig gesendet bzw. empfangen werden und zwar so, dass sich jedes Programm auskennt, um welche Nachricht und vorallem von wem es sich handelt.

Wir werden diesmal eine „While“-Schleife benötigen, mit der wir den Empfang kontrollieren. Aber zuerst werden wir uns darum kümmern etwas zu senden und das indem ich dir zuerst erkläre, wie das Senden im LamaNet funktioniert.

Zuerst die Funktion mit ihren Parametern:



```
LAMA_Send(PlayerID%,MsgID%,Value$, [Secure%])
```



Allgemein

Zuerst sei gesagt. Ich habe versucht LamaNet ein wenig effizienter zu machen. D.h. Es werden Variablen differenziert und Datenblöcke gebildet.

Nachrichten die direkt nacheinander gesendet werden, den selben Empfänger und die selbe NachrichtenID enthalten, werden automatisch in den selben Block zusammengefasst.

PlayerID

Die PlayerID ist kurz gesagt der Empfänger der Nachricht. Bleibt der Wert auf „0“, wird die Nachricht an alle Spieler geschickt, außer dem Programm selbst. D.h. Man muss keine Überprüfung einbauen, ob die Nachricht von sich selbst stammt oder nicht. Das wird sie nicht.

MsgID

Das ist die Nachrichtenennung. Natürlich muss jedes Programm wissen, um was es sich handelt. Wurden ihm gerade Positionen geschickt? Wurde ihm eine normale Nachricht geschickt? Mit der MsgID kann man das Senden und Empfangen um einiges effizienter machen, denn mit ihr muss nicht alles immer mitgeschickt werden, sondern nur zum entscheidenden Zeitpunkt.

Value

In Value kann jede Art von Variablen gespeichert werden. Ein kleiner Kniff ist: LamaNet erkennt, ob es sich um einen Byte,Short,Int,Float oder String-Wert handelt und kann somit den Wert mit so wenig Streamplatz wie möglich versenden.

Secure

Die Sicherheit der Nachricht ist sehr entscheidend.

LamaNet basiert auf UDP und UDP ist bekannterweise nicht die sicherste Variante Daten zu versenden. Dafür ist es um einiges flotter als die TCP Verbindung. LamaNet versucht daher die Sicherheit von Nachrichten zu gewährleisten und schickt unbeantwortete Nachrichten in regelmäßigen Abständen dem betreffenden Spieler nach. Diese Nachrichten verfallen natürlich spätestens wenn der Spieler offline ist, oder eine festgelegte Versuchsanzahl überschritten wurde.

Positionen versenden

Damit unser Programm nun fleißig Kund-tut, wo sich sein Schläger momentan befindet, schreiben wir unter der Positionsfestlegung, noch vor der Spielerdarstellung folgende Zeilen:

```
LAMA_Send(0,1,P\X)
LAMA_Send(0,1,P\Y)
```

Sicher müssen diese zwei Nachrichten diesmal nicht sein, denn sie werden ja ohnehin jedes Frame übermittelt.

Ich habe die MsgID diesmal auf 1 gesetzt. D.h. Alle anderen Programme können später registrieren, dass es sich um Positionsvariablen handeln muss.

Nun sendet jedes Programm sobald es eingeloggt ist, schonmal seine eigenen Positionsdaten!



Positionen Empfangen

Damit unser Programm nun auch weiß, von wem und welche Positionen er von einem Spieler erhalten hat, kommt nun die Empfangsschleife mit „While“

Sie lautet wie folgt und wird idealerweise unter dem Sendevorgang platziert:

```
While LAMA_Recv()  
  
Wend
```

Aber das ist noch nicht ganz alles. Grundsätzlich soll das Programm nämlich nun sofort den Spieler auswählen, von dem er die Nachricht erhalten hat:

```
While LAMA_Recv()  
    P.Player=Object.Player(Player(LAMA_From))  
Wend
```

Wie ich vorher bereits sagte, speichert LAMA_From die SpielerID ab, von dem die aktuelle Nachricht empfangen wurde.

Und nun müssen wir zwischen den verschiedenen NachrichtenIDs unterscheiden:

```
While LAMA_Recv()  
    P.Player=Object.Player(Player(LAMA_From))  
    Select LAMA_MsgID  
        Case 1  
    End Select  
Wend
```

Und fertig ist die einfache Empfangsschleife. Ich habe auch gleich den Fall LAMA_MsgID=1 hinzugefügt, da wir diesen Fall gleich brauchen.

Spieler bearbeiten

Da wir unseren Absender-spieler schon längst ausgewählt haben, können wir nun auch gleich bequem seine Daten bearbeiten. Das nützen wir doch gleich aus und speichern auch sofort seine gesendeten Daten in den Type-Eintrag:

```
While LAMA_Recv()  
    P.Player=Object.Player(Player(LAMA_From))  
    Select LAMA_MsgID  
        Case 1  
            P\X=LAMA_Read()  
            P\Y=LAMA_Read()  
    End Select  
Wend
```



Es wird genau in der gleichen Reihenfolge ausgelesen, wie das Absenderprogramm die Daten abgesendet hat und das erleichtert uns das Auslesen um einiges, sodass nur ein lockeres LAMA_Read() notwendig ist und wir haben die Daten.

1.....2....3....Test (meins!)

Wenn du das Programm nun viermal öffnest, einmal als Server und 3 mal als Client, wirst du erfreut feststellen können, dass jeder Spieler den korrekt platzierten Balken des anderen sehen kann: Das erste eigene Netzwerk-spiel steht auf den Beinen!

... nein, es funktioniert nicht bei dir?

Vielleicht hat sich ja ein Käfer in deinem Programm eingeschlichen und verursacht Getriebschäden im Programm. Schau dir einfach den folgenden Code noch einmal an und vergleiche ihn mit deinen (nicht erschrecken, der Code ist kürzer als er aussieht ;)) :



```
Graphics 480,480,0,2
SetBuffer BackBuffer()
Include "LamaNet.bb"
Local ServerOrClient=Input("Server erstellen(1) - Server suchen(2)")
Select ServerOrClient
    Case 1
        LAMA_HostServer("Pongserver",4,"Lama")
    Case 2
        LAMA_StartCountServers()
    Default
        End
End Select
Dim Player(4);für maximal 4 Spieler ausgelegt
Local P.Player
For i=1 To 4
    P.Player=New Player
    P\ID=i
    Player(i)=Handle(P.Player)
Next
Local Timer=CreateTimer(60)
While Not KeyHit(1)
    LAMA_Update()
    Cls
    MHit=MouseHit(1)
    If LAMA_Connection Then
        P.Player=Object.Player(Player(LAMA_MasterID))
        Select P\ID
            Case 1
                P\X=10
                P\Y=MouseY()
            Case 2
                P\X=GraphicsWidth()-10
                P\Y=MouseY()
            Case 3
                P\X=MouseX()
                P\Y=10
            Case 4
```




```

        P\X=MouseX()
        P\Y=GraphicsHeight()-10
    End Select
    LAMA_Send(0,1,P\X)
    LAMA_Send(0,1,P\Y)
    While LAMA_Recv()
        P.Player=Object.Player(Player(LAMA_From))
        Select LAMA_MsgID
            Case 1
                P\X=LAMA_Read()
                P\Y=LAMA_Read()
            End Select
        Wend
        Color 255,255,255
        For P.Player=Each Player
            If LAMA_Online(P\ID) Then
                Select P\ID
                    Case 1
                        Rect P\X-1,P\Y-50,3,101
                    Case 2
                        Rect P\X-1,P\Y-50,3,101
                    Case 3
                        Rect P\X-50,P\Y-1,101,3
                    Case 4
                        Rect P\X-50,P\Y-1,101,3
                End Select
            EndIf
        Next
    Else
        For i=1 To LAMA_CountServers
            If RectsOverlap(0,(i*15)-
15,GraphicsWidth()/2,15,MouseX(),MouseY(),1,1) Then
                Color 172,189,237
                If MHit Then
                    LAMA_JoinServer(i,"JoinLama")
                EndIf
            Else
                Color 255,255,255
            EndIf
            Rect 0,(i*15)-15,GraphicsWidth()/2,15,0
            Text 0,(i*15)-15,LAMA_ServerName(i)+" -
("+LAMA_ServerPlayers(i)+"/"+LAMA_ServerMaxPlayers(i)+") "
        Next
    EndIf
    WaitTimer Timer
    Flip 0
Wend
End
Type Player
    Field ID%,X%,Y%,Score%
End Type

```

Ball bitte!

Gut, gut! Nicht ungeduldig werden ;)

Man muss sich schließlich auch noch überlegen, welches Programm nun eigentlich die ganze Arbeit



macht. Diese Frage sollte man sich immer wieder stellen, denn sie ist sehr wichtig. Soll der Server die ganze Arbeit machen, oder nur einen Teil. Wenn ja, welchen Teil? Usw.

Man sollte sich im Kopf einen kleinen Plan ausdenken, wie man das Spielsystem umsetzen will, vorallem wenn es darum geht Boni, Waffen oder Sonstiges zum einsammeln einzubauen. Dann sollte jedes Programm wissen, wann es ein Objekt aufnehmen darf oder viel wichtiger, ob es überhaupt noch in dieser Form existiert.

Im Pong Beispiel allerdings wird der Server den größten Teil der Arbeit machen, denn viel ist da sowieso nicht zu machen.

Woher weiß das Programm nun eigentlich, ob es ein Server ist, oder ein Client?

Natürlich könnte man die Frage mit unseren eigenen Variable „**ServerOrClient**“ beantworten. Zusätzlich und vorallem global gibt es dafür aber noch eine andere Variable und zwar:

```
LAMA_Mode
```

Solange diese Variable den Wert „1“ angenommen hat, vertritt das Programm einen Server.

Solange sie den Wert „2“ angenommen hat, ist das Programm ein Client.

Mithilfe dieser Variable veranlassen wir also, dass nur allein der Server den Ball berechnet und sendet.

Der Ball

Definieren wir die Eigenschaften des Balls erst einmal mit ein paar Variablen:

```
Global BallX=GraphicsWidth()/2  
Global BallY=GraphicsHeight()/2  
SeedRnd Millisecs()  
Global BallXSpeed=Rand(-5,+5)  
Global BallYSpeed=Rand(-5,+5)
```

BallX und BallY habe ich bereits vordefiniert, da der Ball anfangs in der Spielfeldmitte erscheinen und starten soll.

Schließlich benötigen wir vor oder nach der Darstellung der Spieler die Darstellung des Balles. Das geht einfach:

```
Rect BallX-1,BallY-1,3,3,1
```

So nun zur Balltechnik selbst.

Ich habe mich dazu entschieden, die Technik dafür vor den Einstellungen des eigenen Spielers vorzunehmen. Nicht vergessen: Wir benötigen eine Modus-Abfrage!

Zusätzlich können wir gleich das Senden dazuschreiben, verwenden aber diesmal die MsgID „2“.

```
If LAMA_Mode=1 then  
    BallX=BallX+BallXSpeed  
    BallY=BallY+BallYSpeed
```



```
LAMA_Send(0,2,BallX)
LAMA_Send(0,2,BallY)
Endif
```

Halt!

Bevor du nun testest, solltest du unbedingt beim Client die Empfangsschleife erneuern, da sonst die Schleife ununterbrochen wiederholt werden würde. → es würde zu Programmabstürzen führen können!

```
While LAMA_Recv()
    P.Player=Object.Player(Player(LAMA_From))
    Select LAMA_MsgID
        Case 1
            P\X=LAMA_Read()
            P\Y=LAMA_Read()
        Case 2
            BallX=LAMA_Read()
            BallY=LAMA_Read()
    End Select
Wend
```

Keine Angst. Der Server wird nicht davon irritiert, dass er die selbe Empfangsschleife hat, wie die Clients. Er ist eben nur der einzige, der die Ball-Koordinaten nicht empfangen wird, sondern nur senden ,). „Case 2“ wird auf dem Server also einfach nicht ausgeführt.

Reflektion

Der Ball, soll jetzt noch möglichst abprallen. Bevor die Schläger drankommen, sollten wir nun ersteinmal auf Wände achten. Solange der Spieler an einer bestimmten Kante offline ist, sollte der Ball an der Kante abprallen und zurückkommen. Später, wenn der Spieler dann online ist, sollte der Ball schließlich ins Aus gehen und Punkte kosten.



```
If BallX>GraphicsWidth() Then
    If LAMA_Online(2)=0 Then
        BallXSpeed=-BallXSpeed
        BallX=GraphicsWidth()
    Else
        BallX=GraphicsWidth()/2
        BallY=GraphicsHeight()/2
        BallXSpeed=Rand(-5,+5)
        BallYSpeed=Rand(-5,+5)
    EndIf
EndIf
If BallX<0 Then
    BallX=GraphicsWidth()/2
    BallY=GraphicsHeight()/2
```



```

        BallXSpeed=Rand(-5,+5)
        BallYSpeed=Rand(-5,+5)
    EndIf
    If BallY>GraphicsHeight() Then
        If LAMA_Online(4)=0 Then
            BallYSpeed=-BallYSpeed
            BallY=GraphicsHeight()
        Else
            BallX=GraphicsWidth()/2
            BallY=GraphicsHeight()/2
            BallXSpeed=Rand(-5,+5)
            BallYSpeed=Rand(-5,+5)
        EndIf
    EndIf
    If BallY<0 Then
        If LAMA_Online(3)=0 Then
            BallYSpeed=-BallYSpeed
            BallY=0
        Else
            BallX=GraphicsWidth()/2
            BallY=GraphicsHeight()/2
            BallXSpeed=Rand(-5,+5)
            BallYSpeed=Rand(-5,+5)
        EndIf
    EndIf
EndIf

```

Diese Zeilen kommen ebenfalls in den Bereich, den nur der Server erledigt. Am besten bevor er die Ball-koordinaten sendet, damit alle möglichst aktuelle Daten besitzen!

Warum bei „BallX<0“ keine zusätzliche If-Bedingung steht?

Ganz einfach: Sobald der Server nicht mehr online ist, existiert das Spiel auch nicht mehr. Deshalb muss auch nicht gefragt werden, ob der Spieler 1 on- oder offline ist ;)

Sobald der Ball nun aus dem Spielfeld schießt, sobald der Spieler an der jeweiligen Kante online ist, Startet der Ball mit zufälliger Geschwindigkeit neu von der Mitte aus.

Teste das Programm doch mal wieder! ;) Du wirst sehen, wie synchron das Spiel abläuft!

Schlägerabprall

So, nun noch der letzte Punkt in meinem Tutorial. Den Rest solltest du ab nun von selbst schaffen, denn dir Grundlagen sind erklärt. Ich habe noch keinen kleinen Skript geschrieben, der den Abprall des Schlägers berechnet (kein realistischer Abprall):

```

    For P.Player=Each Player
        If LAMA_Online(P\ID) Then
            Select P\ID
                Case 1
                    If RectsOverlap(P\X-1,P\Y-50,3,101,BallX-1,BallY-
1,3,3) Then
                        BallXSpeed=-BallXSpeed
                    EndIf
                Case 2
                    If RectsOverlap(P\X-1,P\Y-50,3,101,BallX-1,BallY-
1,3,3) Then
                        BallXSpeed=-BallXSpeed
                    EndIf
                Case 3
                    If RectsOverlap(P\X-50,P\Y-1,101,3,BallX-1,BallY-
1,3,3) Then
                        BallYSpeed=-BallYSpeed

```



```
EndIf
Case 4
  If RectsOverlap(P\X-50,P\Y-1,101,3,BallX-1,BallY-
1,3,3) Then
    BallYSpeed=-BallYSpeed
  EndIf
End Select
EndIf
Next
```

Wenn du diesen Code noch nach den Abprallcode an den Fensterkanten kopierst, sollte der grobe Teil des Spiels funktionieren :)

Natürlich könnte ich jetzt den Score noch fertigbauen, aber diese Aufgabe überlasse ich, aus Übungszwecken, dir. Die Art und Weise des Herangehens an das Problem, wäre ohnehin nicht viel anders, als der bisherige Teil und würde dich womöglich noch langweilen ;)

Ich hoffe das kleine Tutorial hat dir ein wenig geholfen, dich in der Welt von LamaNetwork zurecht zu finden. So schwer ist das alles gar nicht. Sofern man sich bereits ein wenig mit Blitzbasic auskennt, kann man damit schnell ganz ordentliche Netzwerkspiele gestalten.

Bald gibt es hoffentlich noch mehr von mir!

Aber das war jetzt schon ein Haufen Arbeit, dieses Tutorial so zu präsentieren. Ich hoffe, ihr freut euch über diese kleine Hilfe und noch mehr, dass sie euch hilft!

Schreibt mir einfach, wie ihr LamaNet findet und was man eventuell noch verbessern könnte.

Ich werde euch dann antworten, inwiefern das realisierbar ist, oder nicht. :)

mylama@yahoo.de

liebe Grüße,
Chrise

**Und nun folgt noch wie versprochen,
die Befehlsübersicht**



A-Z

Globale Variablen

LAMA_Connection%

1=online, 0=offline

LAMA_CountServers%

Anzahl im Netzwerk gefundener Server

LAMA_CountTransfers%

Anzahl laufender Datei-transfers

LAMA_EventGadgetID%

Identifikationsnummer eines Event-Objektes

LAMA_EventID%

Identifikationsnummer eines Events

LAMA_EventInfo\$

Zusätzliche Info bei bestimmten Events

LAMA_EventPlayerID%

SpielerID eines vom Event betroffenen Spielers

LAMA_FileEnding\$

Dateiendung (ohne .) der aktuellen Datei

LAMA_FileName\$

Dateiname (ohne Endung) der aktuellen Datei

LAMA_FilePath\$

Dateipfad der aktuellen Datei

LAMA_FileSize%

Dateigröße (in Bytes) der aktuellen Datei

LAMA_From%

Absender Identifikationsnummer eines Nachricht

LAMA_GadgetID%

Identifikationsnummer eines Objekts

LAMA_MasterID%

Spieler-/Programmeigene Identifikationsnummer

LAMA_Mode%

Programmmodus (1=Host, 2=Client)

LAMA_MsgID%

Identifikationsnummer einer Nachricht

LAMA_ServerID%

Identifikationsnummer des Servers

Felder

LAMA_TransferEnding\$(0)

Endung einer übertragenen Datei

LAMA_TransferFileName\$(0)

Name einer übertragenen Datei (mit Endung)

LAMA_TransferMode(0)

Modus einer Dateiübertragung (1=Senden, 2=Empfangen)

LAMA_TransferName\$(0)

Name einer übertragenen Datei (ohne Endung)

LAMA_TransferPath\$(0)

Pfad einer übertragenen Datei

LAMA_TransferProgress#(0)

Fortschritt einer Dateiübertragung

LAMA_TransferSize%(0)

Größe einer übertragenen Datei (in Bytes)

Events

LAMA_Banned%

Programm wurde vom Server gebannt

LAMA_CanceledTransfer%

Dateiübertragung wurde abgebrochen

LAMA_Connected%

Verbindung wurde hergestellt (Host und Join)

LAMA_CountRequest%

Server wurde von einer Suchanfrage erreicht

LAMA_Hosted%

Verbindung wurde hergestellt (nur Host)

LAMA_JoinDenied%

Joinanfrage wurde verweigert

LAMA_Joined%

Verbindung wurde hergestellt (nur Join)

LAMA_JoinRequest%

Server wurde von einer Joinanfrage erreicht

LAMA_JoinTimeOut%

Join Zeitüberschreitung

LAMA_Kicked%

Programm wurde vom Server gekickt

LAMA_NameExists%

Join verweigerung - Name existiert bereits

LAMA_NewPlayer%

Neuer Spieler tritt dem Spiel bei

LAMA_PlayerBanned%

Ein Spieler wurde vom Server gebannt

LAMA_PlayerBlocked%

Ein Spieler wurde vom Server geblockt

LAMA_PlayerCanceledTransfer%

Ein Spieler hat Datei-transfer abgebrochen

LAMA_PlayerKicked%

Ein Spieler wurde vom Server gekicked

LAMA_PlayerLeft%

Ein Spieler hat das Spiel verlassen

LAMA_PlayerRenamed%

Ein Spieler wurde umbenannt

LAMA_ReceivedBank%

Datenbank wurde erhalten

LAMA_ReceivedFile%

Datei wurde erhalten

LAMA_ServerChangedInformation%

Server hat Informationen geändert

LAMA_ServerFull%

Server ist voll

LAMA_ServerLeft%

Server ist offline gegangen

LAMA_ServerRenamed%

Server wurde umbenannt

LAMA_ServerStillLocked%

Server ist weiterhin gesperrt

LAMA_StillBanned%

Programm ist weiterhin vom Server gebannt

LAMA_TransferFailed%

Datei-transfer fehlgeschlagen

LAMA_TransmittedFile%

Datei wurde verschickt

LAMA_WrongPassword%

falsches Passwort wurde eingegeben



Funktionen

LAMA_BanPlayer%(PlayerID%,BanReason\$,BanSeconds%,BanMinutes%,BanHours%)

Diese Funktion bannt einen bestimmten Spieler vom Server. Dazu wird einfach die Identifikationsnummer des Spielers und die Dauer des Banns in Sekunden, Minuten und/oder Stunden angegeben. Werden alle 3 Werte nicht übergeben, so findet ein dauerhafter Bann per Banlist statt.

Parameter

PlayerID%	SpielerID des zu bannenden Spielers
BanReason\$	Grund für den Spielerbann
Banseconds%	Bannzeit in Sekunden
BanMinutes%	Bannzeit in Minuten
BanHours%	Bannzeit in Stunden

LAMA_CancelTransfer%(TransferID%)

Diese Funktion bricht einen laufenden Datentransfer ab. Dazu wird einfach die TransferID angegeben. Verwende LAMA_CountTransfers um die Anzahl aller Dateitransfers zu erhalten.

LAMA_ClearEvents%()

Der Befehl LAMA_ClearEvents löscht alle sich im Speicher befindenden Events. Diese Funktion kann besonders dann nützlich sein, wenn man erneut ein Spiel startet.

LAMA_CloseServer%()

Diese Funktion unterbricht die Verbindung zum aktuellen Server. Sie ist besonders am Ende eines Programmes oder beim verlassen eines Spieles nützlich. Mit dieser Funktion werden Informationen zum Verlassen des Spiels direkt an die anderen Spieler weitergeleitet und diese müssen nicht auf eine Zeitüberschreitung warten, bis sie das Verlassen des Spielers registrieren.

LAMA_Event%()

Diese Funktion registriert, ob ein Event vorliegt oder nicht.

LAMA_HostServer%(ServerName\$,MaxPlayers%,Name\$,ServerPassword\$)

Diese Funktion erstellt einen Server im Netzwerk.

Parameter

ServerName\$	Name des zu erstellenden Servers
MaxPlayers%	Maximale Anzahl zugelassener Spieler
Name\$	Spielernamen
[Passwort\$]	Festlegung des Serverpasswortes

LAMA_JoinServer%(Server%,Name\$,Password\$,MaxPing%)

Diese Funktion verbindet sich mit einem bestehenden Server. Dazu wird einfach die ServerID angegeben. Verwende LAMA_CountServers um die Anzahl aller Server im Netzwerk zu erhalten.

Parameter

Server%	Identifikationsnummer des Servers
Name\$	Spielernamen
[Passwort\$]	Fall benötigt, Passwort
[MaxPing%]	Maximale Wartezeit um auf Serverantwort zu warten

LAMA_KickPlayer%(PlayerID%)

Der Befehl LAMA_KickPlayer kickt einen bestimmten Spieler vom Server. Dazu wird einfach die ID des Spielers angegeben.

Parameter

PlayerID%	SpielerID
-----------	-----------

LAMA_LockServer()

Diese Funktion sperrt den Server, sodass niemand mehr beitreten kann

LAMA_Name\$(PlayerID%)

Diese Funktion liefert anhand der SpielerID den Namen eines bestimmten Spielers zurück

Parameter

PlayerID%	SpielerID
-----------	-----------

LAMA_NewBlock%()

Diese Funktion legt einen neuen Sendeblock an. Nützlich wird diese Funktion, wenn man mehrere Type-Einträge nacheinander verschickt. Wende den Befehl beispielsweise in der For/Each-Schleife an, um die Übertragung zu schonen.

LAMA_Online%(PlayerID%)

Diese Funktion liefert anhand der SpielerID den Onlinestatus eines bestimmten Spielers zurück

Parameter

PlayerID%	SpielerID
-----------	-----------

LAMA_Ping%(PlayerID%)

Diese Funktion liefert anhand der SpielerID den Pingwert eines bestimmten Spielers zurück

Parameter

PlayerID%	SpielerID
-----------	-----------

LAMA_Read\$()

Diese Funktion liest empfangene Daten aus

LAMA_ReadFile%()

Diese Funktion liest Informationen einer übertragenen Datei aus

LAMA_Recv%()

Der Befehl LAMA_Recv registriert, ob Nachrichten empfangen wurden, oder nicht.

LAMA_RenamePlayer%(PlayerID%,Name\$)

Diese Funktion benennt anhand der SpielerID und dem Namen einen bestimmten Spieler um.



Parameter

PlayerID%	SpielerID
Name\$	Neuer Spielername

LAMA_RenameServer(ServerName\$)

dies Funktion benennt den Server um (kann nur vom Server ausgeführt werden).

LAMA_Send%(PlayerID%,MsgID%,Value\$,Secure%)

Diese Funktion bereitet Werte zum senden vor.

Parameter

PlayerID%	Empfänger SpielerID (Standard=0 an alle Spieler)
MsgID%	Nachrichten Identifikationsnummer
Value\$	Zu übermittelnder Wert
Secure%	Sicherheit der Nachricht (0=nicht sicher, 1=sicher) (Standard=0)

LAMA_SendCountRequest%()

Diese Funktion sendet eine Suchanfrage an alle Server, die sich im gleichen Netzwerk befinden.

LAMA_SendFile%(PlayerID%,GadgetID%,Path\$)

Diese Funktion bereitet eine Nachricht zum senden vor.

Parameter

PlayerID%	Empfänger SpielerID (Standard=0 an alle Spieler)
GadgetID%	Identifikationsnummer der Dateiübertragung
Path\$	Dateipfad

LAMA_ServerBannedFrom%(ServerID%)

Diese Funktion ermittelt, ob das Programm von einem bestimmten Server gebannt ist, oder nicht. (0=nicht gebannt, 1=gebannt)

Parameter

ServerID%	Identifikationsnummer des Servers
-----------	-----------------------------------

LAMA_ServerInformation\$(ServerID%,InfoID%)

Diese Funktion liefert ServerInformationen anhand der ServerID und der InformationsID zurück.

Parameter

ServerID%	Identifikationsnummer des Servers (ermittelt durch LAMA_CountServers, bei 0=aktueller Server)
InfoID%	Indexnummer der Information (1-10)

LAMA_ServerLocked%(ServerID%)

Diese Funktion ermittelt, ob ein bestimmter Server gesperrt ist, oder nicht. (0=nicht gesperrt, 1=gesperrt)

Parameter

ServerID%	Identifikationsnummer des Servers (Standard=0 aktueller Server)
-----------	---

LAMA_ServerMaxPlayers%(ServerID%)

Diese Funktion ermittelt, die maximale Anzahl zugelassener Spieler auf dem Server

Parameter

ServerID%	Identifikationsnummer des Servers (Standard=0 aktueller Server)
-----------	---

LAMA_ServerName\$(ServerID%)

Diese Funktion ermittelt, den Namen des Servers

Parameter

ServerID%	Identifikationsnummer des Servers (Standard=0 aktueller Server)
-----------	---

LAMA_ServerPassword%(ServerID%)

Diese Funktion ermittelt, ob ein Passwort zum Beitritt nötig ist, oder nicht. (1=Passwort geschützt, 0=nicht Passwort geschützt)

Parameter

ServerID%	Identifikationsnummer des Servers (Standard=0 aktueller Server)
-----------	---

LAMA_ServerPing%(ServerID%)

Diese Funktion ermittelt, den Ping zum Server (gemessen zum Zeitpunkt der Suchanfrage)

Parameter

ServerID%	Identifikationsnummer des Servers (Standard=0 aktueller Server)
-----------	---

LAMA_ServerPlayers%(ServerID%)

Diese Funktion ermittelt, die Anzahl aktuell verbundener Spieler auf dem Server

Parameter

ServerID%	Identifikationsnummer des Servers (Standard=0 aktueller Server)
-----------	---

LAMA_ServerPublisher\$(ServerID%)

Diese Funktion ermittelt, den Namen des Veröffentlichers des Servers

Parameter

ServerID%	Identifikationsnummer des Servers (Standard=0 aktueller Server)
-----------	---

LAMA_SetPort(Port%)

Diese Funktion legt den Port für die Übertragung fest. Diese Funktion muss vor LAMA_HostServer oder LAMA_JoinServer bzw. LAMA_StartCountServers und LAMA_SendCountRequest ausgeführt werden, damit die Änderung eine Wirkung erzielt.

Parameter

Port%	Neuer Port
-------	------------

LAMA_SetServerInformation(InfoID%,Information\$)

Diese Funktion speichert eine definierte Information auf dem Server. Diese Informationen können dann mit LAMA_StartCountServers und LAMA_SendCountRequest abgerufen werden.

Parameter

InfoID%	Indexnummer der Information (1-10)
Information\$	Information die auf dem Server gespeichert werden soll

LAMA_StartCountServers%(MaxServers%)

Diese Funktion beginnt die Suche nach Servern im Netzwerk

Parameter

MaxServers%	Maximale Anzahl an Servern, nach denen gesucht werden soll
-------------	--



LAMA_StopCountServers%()

Diese Funktion beendet die Suche nach Servern im Netzwerk

LAMA_UnLockServer()

Diese Funktion entsperrt den Server

LAMA_Update%()

Diese Funktion aktualisiert die komplette LamaNet-Engine und ist notwendig für alle Funktionen in der Funktionssammlung, damit diese korrekt laufen.

