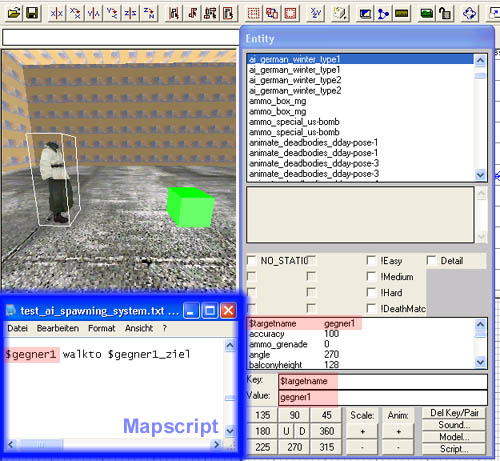
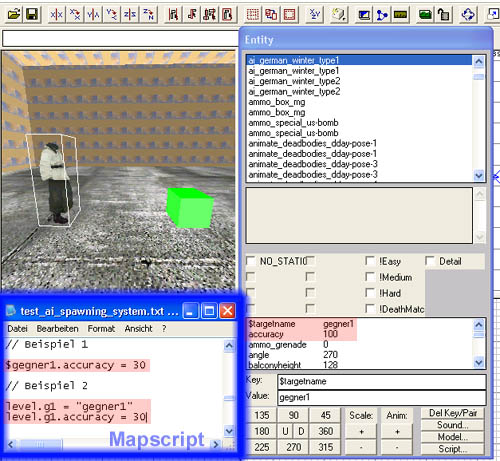
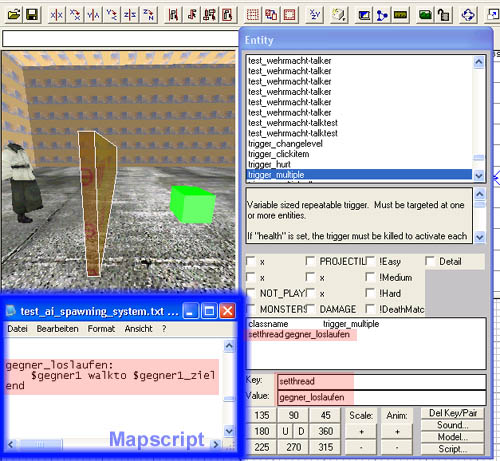
**Scripting Einführung (SP/MP)**

© 3/2003 T.Reim | Dieses Tutorial ist für Singleplayer und Multiplayer Maps.

So, nun gehts ans Eingemachte. Ich will mal versuchen euch das Scripten zu erklären, dazu ist es ratsam wenn ihr schon ein paar Tutorials die Scripte enthalten durchgenommen habt. Der Grund dafür ist, das ihr schon ein paar Ausdrücke kennt, da ich nicht genau weiß wo ich beim Erklären anfangen soll. (Am besten ihr lest euch die Erklärung 2 – 3 mal durch)

**Allgemeines:**  
Sicher habt ihr schon in den Scripten Zeilen gesehen die mit // anfangen. Dies bedeutet das diese Zeile vom Spiel nicht verarbeitet wird wenn ihr eure Map spielt. Ihr könnt die Zeichen sehr gut nutzen um Notizen in eurem Script unterzubringen und um Befehlszeilen zu deaktivieren beim Testen eurer Maps/Scripte.

1. “targetname” und “$targetname”  
   Der targetname sollte euch ja schon aus dem Radiant und den anderen Tutorials bekannt sein. Ich erkläre hier nochmal kurz den Zusammenhang zwischen der Map (Radiant) und dem Mapscript (.scr).  
   Damit das Spiel nachher weiß welches Objekt (Explosionswall, Zünder, Türen, Panzer, Kanonen, Trigger, script\_object usw..) in der Map was machen soll, benötigt jedes Objekt ein targetname. (Genauso wie jeder Mensch einen Namen hat damit man weiß wie man ihn ansprechen soll) Der targetname wird im Radiant im Entity Menü eingegeben, und bezieht sich mit der Eingabe von $ vor dem targetname, auf Objekte die über das Script gesteuert werden. Ohne das Zeichen $ vor dem targename bezieht sich das Objekt auf ein anderes Objekt in der Map oder ein Global Script. Es ist aber möglich den targetname mit $ und ohne $ für Befehle im Mapscript zu verwenden. Also wenn ihr feststellt das ihr das $ vergessen habt und eure Map schon compiliert ist, macht das nichts. Ihr solltet aber darauf achten keine targetname mit $ und ohne $ auf verschiedene Objekte doppelt zu verwenden.  
   Der $targetname für das Mapscript:  
   Der targetname eines Objektes aus einer Map wird immer mit dem $ vorangestellt im Mapscript geschrieben. Wenn ihr also einen AI in der Map habt der den Entity-Eintrag $targetname = gegner1 hat, dann wird der AI im Mapscript mit $gegner1 angesprochen.  
   Hier nochmal ein Bild um das deutlicher zu machen:  
   
2. Variablen  
   Damit wir gewisse Bedingungen abfragen und Scenarien Scripten können benötigen wir Variablen.  
   Variablen werden immer mit dem $ vorangestellt geschrieben, mit Ausnahme der vordefinierten Variablen. Eine Variable kann ein targetname sein oder nur einen bestimmten Wert enthalten.  
   **Vordefinierte Variablen:**  
   Die vordefinierten Variablen werden mit dem Stammnamen geschrieben, gefolgt von einem Namen eurer Wahl. (ihr müsst aber darauf achten keine Namen zu verwenden die im Spiel vordefiniert sind, z.b. health, nehmt Namen die aus einem Buchstaben und einer Zahl bestehen)  
   Es gibt verschiedene Arten von Variablen die im Mapscript eingesetzt werden können:  
   **level.**  
   Die Variable level. steht für das gesamte Mapscript zur Verfügung, und kann überall im Mapscript gesetzt werden. Beispiel:  
   **level.dg1 = 1**  
   Dies bedeutet das die Levelvariable level.dg1 mit dem von mir gewählten Namen dg1 den Wert 1 hat.  
   **local.**  
   Die Variable local. steht nur für den thread zur Verfügung in dem sie auch gesetzt wird. Das heisst ihr müsst die Variable in dem thread definieren in dem sie auch steht. Beispiel:  
   **…  
   gegner\_loslaufen:  
   local.dg2 = 2  
   end  
   …**  
   Dies bedeutet das die Localvariable local.dg2 mit dem von mir gewählten Namen dg2 den Wert 2 hat, aber nur für Abfragen in dem thread zur Verfügung steht in der die Variable auch steht. local bezogen auf den einen thread.  
   **Objektbezogene Variablen**  
   Wie ich am Anfang schon erwähnt habe, solltet ihr darauf achten das ihr den Namen einer vordefinierten Variable für das Spiel nicht verwendet, wie z.B. health  
   Diese Variablen sind im Spiel vordefiniert für Objekt Parameter. Beispiel 1: ihr habt in eure Map einen Gegner plaziert mit dem targetname $gegner1. Alle Parameter des AI stehen über das Verknüpfungszeichen . im Script zur Verfügung. Wenn ihr jetzt den Entity Wert accuracy 100 nachträglich ändern möchtet auf accuracy 30, wird im Mapscript das wie folgt geschrieben:  
   **…  
   $gegner1.accuracy = 30  
   …**  
   Das gleiche gilt auch für die Levelvariable level.  
   Beispiel 2: ihr habt einen AI in eurer Map der aus irgendeinen Grund über die Levelvariable angeseteuert werden soll. Dann wird eine neue Levelvariable erstellt und mit dem targetname des Gegners belegt, somit könnt ihr wieder die Objektbezogenen Variablen des AI ändern:  
   **…  
   level.g1 = “gegner1″  
   level.g1.accuracy = 30  
   …**  
   
3. “thread”  
   Den Begriff thread kennt ihr ja schon aus den Foren im Internet. Im Prinzip ist es hier die selbe Funktion, ein thread grenzt einen gewissen Bereich im Mapscript ein, mit einem thread Namen . Jeder thread beginnt mit einem Namen: und wird mit einem end abgeschlossen. Hier ein kleines Beispiel:  
   **…  
   gegner\_loslaufen:  
   $gegner1 walkto $gegner1\_ziel  
   end  
   …**  
   Der Name des threads ist gegner\_loslaufen Der thread endet beim Befehl end. Die in dem thread auszuführenden Befehle sind $gegner1 walkto $gegner1\_ziel , das heisst der AI $gegner1 geht zum Ziel $gegner1\_ziel.  
   **Merke:** die Zeile des thread Name wird mit einem : abgeschlossen, der komplette thread wird mit einem end abgeschlossen  
   **thread aufrufen über das Mapscript:**  
   um nun die Befehle für den $gegner1 ausführen zu lassen, müssen wir den thread auch aufrufen. Das wird mit dem Befehl thread gefolgt von dem Namen des threads, ins Mapscript geschrieben. In unserem Fall sieht es dann an irgendeiner Stelle im Mapscript so aus:  
   **…  
   thread gegner\_loslaufen  
   …**  
   **thread aufrufen über einen Trigger in der Map:**  
   eine weitere Möglichkeit die Befehle in einem thread ausführen zu lassen, ist in der Map einen Trigger zu setzen und diesem den folgenden Entity Eintrag zu geben:  
   **setthread = gegner\_loslaufen**  
     
   Hierdurch wird im Spiel beim auslösen des Triggers der thread gegner\_loslaufen aufgerufen und die dort enthaltenen Befehle bis zum end ausgeführt.  
   **thread auf ein bestimmtes Objekt aufrufen:**  
   um eine Befehlsfolge auf ein bestimmtes Objekt ausführen zu lassen, muss man den targetname des Objektes vor den Befehl thread schreiben. Dann wird in dem aufgerufenen thread der targetname als self weiter verarbeitet. Der Befehl self ist ein Platzhalter für den targetname beim aufrufen eines thread bezogen auf den targetname. Mit self kann man eine einheitliche Befehlsfolge für mehrere Objekte realisieren.  
   **…  
   $gegner1 thread gegner\_loslaufen  
   $gegner2 thread gegner\_loslaufen  
   …  
   gegner\_loslaufen:  
   self walkto $gegner1\_ziel  
   end  
   …**  
   In diesem Beispiel wird das Ziel wo der AI hinlaufen soll nicht übergeben. Gegner 1 und 2 würden hier zum selben Ziel laufen. Wie das Ziel mit übergeben wird erkläre ich im nächsten Abschnitt.
4. Variablen / Werte auf ein thread übergeben  
   In manchen Fällen ist es notwendig Variablen oder Werte zusätzlich zur weiteren Verarbeitung auf einen thread zu übergeben. Die definition der zu übergebenden Werte erfolgt am thread anfang, als Localvariable, und steht somit nur für den thread zur Verfügung. Als Beispiel nehmen wir das der Befehl zum Ziellauf eines AI für mehrere AI´s zur Verfügung stehen soll.  
   **…  
   $gegner1 thread gegner\_loslaufen $gegner1\_ziel  
   …  
   …  
   gegner\_loslaufen local.t1:  
   self walkto local.t1  
   end  
   …**  
   Die Localvariable local.t1 nimmt hier eine Platzhalterstellung ein, überall wo loacal.t1 in diesem thread steht wird der Wert $gegner1\_ziel verarbeitet.
5. Rechenoperationen  
   Damit wir auch bestimmte Werte errechnen lassen können stehen die Rechenoperationen zur Verfügung:  
   + Addition  
   - Subtraktion  
   / Division  
   \* Multiplikation  
   Hinzu kommen noch ein paar Befehle um dem Scripter das Leben leichter zu machen:  
   ++ Werte um 1 erhöhen  
   – Werte um 1 verringern
6. Vergleichsoperatoren  
   Um unsere Abfragen zu realisieren brauchen wir natürlich auch Vergleichsoperatoren:  
   == ist gleich  
   >= ist grösser gleich  
   <= ist kleiner gleich  
   != ist ungleich  
   und natürlich auch Logische Operatoren  
   && und  
   || oder
7. Schleife “while”  
   Eine Schleife wird so lange ausgeführt bis die Bedingung erfüllt ist. Das heisst wenn z.B. die Levelvariable level.dg1 grösser als 5 ist, wird die Schleife beendet. Solange level.dg1 kleiner als 5 ist werden die Befehle innerhalb der Schleife ausgeführt.  
   while (Bedingung)  
   {  
   auszuführende Befehle  
   }  
   **Merke:** die auszuführenden Befehle werden immer mit { } eingeklammert.  
   Beispiel:  
   **…  
   level.dg1 = 0  
   while (level.dg1 < 5)  
   {  
   inprintlnbold “Levelvariable dg1 ist kleiner als 5″  
   level.dg1++  
   wait 5  
   }  
   …**  
   Die Bedingung der Schleife ist in diesem Fall die Abfrage level.dg1 < 5  
   iprintlnbold hat zur Folge, das der Text innerhalb der “” als Consolentext im Spiel angezeigt wird.  
   level.dg1++ erhöht bei jeder Befehlsausführung der Schleife den Wert level.dg1 um 1  
   wait 5 hat zur Folge das an dieser Stelle im Script 5 Sekunden gewartet wird bis die nächsten Befehle ausgeführt werden
8. Fallunterscheidung mit “if”  
   Damit wir verschiedene Bedingungen unterschiedliche Befehle zuweisen können, gibt es den if Befehl. Mit if kann man eine Bedingung stellen, und bei nicht erfüllung der Bedingung oder bei erfüllung der Bedingung bestimmte Befehle ausführen lassen. Es gibt verschiedene Arten diesen Befehl anzuwenden, schaut euch erstmal diese kurze Erklärung an:  
   a)  
   if (Bedingung)  
   {  
   auszuführende Befehle wenn die Bedingung erfüllt ist  
   }  
   …  
   b)  
   if (Bedingung)  
   {  
   auszuführende Befehle wenn die Bedingung erfüllt ist  
   }  
   else  
   {  
   auszuführende Befehle wenn die Bedingung nicht erfüllt ist  
   }  
   …  
   c)  
   if (1. Bedingung)  
   {  
   auszuführende Befehle wenn die 1. Bedingung erfüllt ist  
   }  
   else if (2. Bedingung)  
   {  
   auszuführende Befehle wenn die 1. Bedingung nicht erfüllt ist, aber die 2. Bedingung erfüllt ist  
   }  
   else  
   {  
   auszuführende Befehle wenn die 1. und 2. Bedingung nicht erfüllt ist  
   }  
   …  
   Machen wir gleich mit einem Beispiel weiter. Wenn die Levelvariable level.dg1 den Wert 2 hat soll ein AI zum zweiten Ziel laufen, wenn der Wert nicht 2 ist, soll der AI zum ersten Ziel laufen.  
   …  
   **level.dg1 = 2  
   if (level.dg1 == 2)  
   {  
   $gegner1 walkto $gegner1\_ziel2  
   }  
   else  
   {  
   $gegner1 walkto $gegner1\_ziel1  
   }  
   …**  
   Ein weiteres Beispiel:  
   Wenn die Levelvariable level.dg1 grösser als 2 ist soll der AI zum dritten Ziel laufen, wenn der Wert 2 ist soll der AI zum zweiten Ziel laufen und wenn der Wert eben kleiner als 2 ist soll der AI zum ersten Ziel laufen.  
   **…  
   level.dg1 = 1  
   if (level.dg1 == 2)  
   {  
   $gegner1 walkto $gegner1\_ziel2  
   }  
   else if (level.dg1 > 2)  
   {  
   $gegner1 walkto $gegner1\_ziel3  
   }  
   else  
   {  
   $gegner1 walkto $gegner1\_ziel1  
   }  
   …**
9. Fallunterscheidung mit “switch”  
   Der Befehl switch ist im Prinzip genauso einzusetzen wie der if Befehl, mit dem Unterschied das der Wert der Bedingung fest definiert sein muss, und nicht z.B. grösser oder kleiner als eine Zahl sein kann. Die verwendung von switch kann unter Umständen einiges an Arbeit und Zeilen im Script ersparen.  
   …  
   switch (Variable)  
   {  
   case “Variablenwert 1″:  
   auszuführende Befehle wenn die Variable den Variablenwert 1 hat  
   break  
   case “Variablenwert 2″:  
   auszuführende Befehle wenn die Variable den Variablenwert 2 hat  
   break  
   default:  
   auszuführende Befehle wenn die Variable keinen der anderen beiden Variablenwerte hat  
   break  
   }  
   …  
   Der Befehl break hat zur Folge das die Befehlsfolge dort unterbrochen wird und nicht alle nachfolgenden Befehle die zu den anderen case und default gehören ausgeführt werden. Der Befehl default muss gesetzt werden und beinhaltet die Befehle die ausgeführt werden wenn keiner der case zutrifft.  
   Beispiel:  
   Wenn die Levelvariable level.dg1 1 ist, soll der AI zum ersten Ziel laufen, wenn der Wert 2 ist soll der AI zum zweiten Ziel laufen und wenn der Wert nicht gesetzt ist soll er zum dritten Ziel laufen.  
   **…  
   level.dg1 = 2  
   switch (level.dg1)  
   {  
   case 1:  
   $gegner1 walkto $gegner1\_ziel1  
   break  
   case 2:  
   $gegner1 walkto $gegner1\_ziel2  
   break  
   default:  
   $gegner1 walkto $gegner1\_ziel3  
   break  
   }  
   …**
10. MOHAA Map-Basisscript  
    Da manche immer noch fragen wo denn das Mapscript hinkommt hier nochmal die Erklärung. Das Mapscript hat immer den selben Dateinamen wie eure Map nur mit der Dateiendung .scr und wird dort gespeichert wo eure Map gespeichert wird. Das scr bedeutet bei MOHAA Script und nicht wie allgemein bekannt ist Bildschirmschoner für Windows. Dateien für MOHAA die die Dateiendung .scr haben, sind Textdateien und können mit der Dateiendung .txt versehen werden, um sie zu editieren. Ihr dürft nur nicht vergessen es wieder in .scr umzubenennen wenn ihr eure Map spielt.  
    Jetzt zum Inhalt des Mapscriptes:  
    Grundsätzlich fängt jedes MOHAA Script mit dem main: thread an. Dieser ist für das Spiel vordefiniert und brauch nicht extra aufgerufen werden, da er automatisch beim Mapstart ausgeführt wird. Der main thread beinhaltet für die Map relevante Befehle die zum Ausführen bestimmter Animationen usw. wichtig sind.  
    **main:**  
    Befehle die beim Laden der Map ausgeführt werden, z.B. Global Scripte für AI  
    **level waittill prespawn**  
    Befehle die nach dem Laden der Map ausgeführt werden, z.B. Ambientsound  
    **level waittill spawn**  
    Befehle die nach dem spawnen der Map ausgeführt werden, z.B. Waffenzuweisung für den Spieler  
    **end**  
    Die Befehle level waittill prespawn und level waittill spawn gehören zu jedem Mapscript und stehen im main thread. Nach dem main thread folgen dann alle anderen threads die benötigt werden.
11. Anwendungsbeispiel  
    Siehe Tutorial: [AI Spawning System](http://www.steinhuder-meer-portal.de/mohaa/tutorials/ai-spawning-system).

Soooo,  
das wars von meiner Seite, jetzt seit ihr dran. :-)  
Ich hoffe das euch dieses Tutorial als Einstieg in das Scripten und als Glossar für die wichtigsten Befehle ausreicht.

Viel Spass und Erfolg  
Cheech