Kapitel 6: Scripting

In diesem Teil des Tutorials werden wir uns im Allgemeinen mit den Scripts befassen. Es werden keine Programmierkenntnisse benötigt, um Scripts anzupassen, jedoch sollte man sich die Kommentare hinter den Code-Zeilen genau durchlesen, um richtige Änderungen vorzunehmen. Wie bereits unter dem Punkt "Das erste Konzept" erwähnt, werden wir ein **Startscript**, ein einfaches **Alarmierungsscript** und ein **Parkscript** in die Modifikation einbauen.

Inhaltsverzeichnis

Das Start-Script

Das Startscript wird zu Beginn des Freeplay, also unmittelbar nach Beendigung des Ladevorgangs ausgeführt. Da wir uns dafür entschieden haben, die Fahrzeuge direkt auf der Map zu platzieren, fällt hier ein automatisches Positionieren der Fahrzeuge weg.In unserem Script wird den Fahrzeugen nur eine Geschwindigkeit zugeordnet und einer Zahl des Codeblocks zugewiesen.Hier erst mal das Script:

Quellcode

```
1. void Start()
2. {
3. //Block//Fahrzeuge////////////////////////////////////////////////////////////////////////////
4. VehicleList vl("Fahrzeug_1"); //Name des Fahrzeugs auf der Map if(vl.GetNumVehicles() == 0)
5. System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
6. else
7. {
8. Vehicle v(vl.GetVehicle(0));
9. v.SetSpeed(10.0f); //Wert der Geschwindigkeit anpassen
10. }
11. //Block//Fahrzeuge////////////////////////////////////////////////////////////////////////////
12. //Bei weiteren Fahrzeugen Block kopieren und Werte auf das neue Fahrzeug anpassen
13. //Block//Disponenten//////////////////////////////////////////////////////////////////////////
14. PersonList pl("Disponent_1");
15. if(pl.GetNumPersons() == 0)
16. System::Log("Disponent nicht gefunden: %s", pl.GetPerson(0)->GetName());
17. else
18. Game::AddToGroup(pl.GetPerson(0), 0); //Letzte Zahl gibt die Zahl im Codeblock an
19. //0 entspricht Codeblock 1
20. //1 entspricht Codeblock 2
21. //2 entspricht Codeblock 3
22. //...
23. //Block//Disponenten//////////////////////////////////////////////////////////////////////////
24. //Bei weiteren Disponenten Block kopieren und Werte auf den neuen Disponenten anpassen
25. };
26. bool OnLoad()
27. {
28. Start();
29. Process::Kill();
30. return true;
31. }
```

Alles anzeigen

Wir ersparen uns an dieser Stelle das genauere Eingehen auf die Funktionsweise dieses relativ kurzen Scripts. Viel interessanter dagegen ist das Erweitern um weitere Fahrzeuge. Beginnen wir mit den drei Fahrzeugen der hauptamtlichen Wache: Dem ELW, der DLK und dem HLF.Im Editor wurde den Prototypen bereits die Information über die Anzahl der Sitzplätze gegeben und benötigte Commands zugewiesen. Jedoch kann im Editor nicht die Geschwindigkeit definiert

werden, was aus diesem Grund das Startscript übernehmen wird.

Fahrzeuge im Start-Script

Hauptamtliche Wache

Quellcode

```
1. VehicleList vl("ELW_Haupt"); //Name des ELW der hauptamtlichen Wache auf der Map
2. if(vl.GetNumVehicles() == 0)
3. System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
4. else
5. {
6. Vehicle v(vl.GetVehicle(0));
7. v.SetSpeed(12.0f); //Wert der Geschwindigkeit des ELW der hauptamtlichen Wache
8. }
9. VehicleList vl("DLK_Haupt"); //Name der DLK der hauptamtlichen Wache auf der Map
10. if(vl.GetNumVehicles() == 0)
11. System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
12. else
13. {
14. Vehicle v(vl.GetVehicle(0));
15. v.SetSpeed(9.0f); //Wert der Geschwindigkeit der DLK der hauptamtlichen Wache
16. }
17. VehicleList vl("HLF_Haupt"); //Name des HLF der hauptamtlichen Wache auf der Map
18. if(vl.GetNumVehicles() == 0)
19. System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
20. else
21. {
22. Vehicle v(vl.GetVehicle(0));
23. v.SetSpeed(10.0f); //Wert der Geschwindigkeit des HLF der hauptamtlichen Wache
24. }
```

Alles anzeigen

An dieser Stelle haben wir uns für die Geschwindigkeiten 12 für ELW, 9 für die DLK und 10 für das HLF entschieden. Natürlich kann an den Werten später noch rumgespielt werden. Viel wichtiger ist aber, dass die Namen im Script mit dem tatsächlichen Namen der Fahrzeuge übereinstimmen.

Wache der Freiwilligen Feuerwehr

Hier wenden wir das gleiche Vorgehen an, wie bei der Hauptamtlichen Feuerwache.

Quellcode

```
1. VehicleList vl("MTW_FFW"); //Name des MTW der Freiwilligen Feuerwehr auf der Map
2. if(vl.GetNumVehicles() == 0)
3. System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
4. else
5. {
6. Vehicle v(vl.GetVehicle(0));
7. v.SetSpeed(12.0f); //Wert der Geschwindigkeit des MTW der Freiwilligen Feuerwehr
8. }
9. VehicleList vl("LF16_FFW"); //Name des LF der Freiwilligen Feuerwehr auf der Map
10. if(vl.GetNumVehicles() == 0)
11. System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
12. else
13. {
14. Vehicle v(vl.GetVehicle(0));
15. v.SetSpeed(8.0f); //Wert der Geschwindigkeit des LF der Freiwilligen Feuerwehr
16. }
```

Alles anzeigen

Die Rettungswache

Auch hier greifen wir auf die gleiche Systematik zurück.

Quellcode

```
VehicleList vl("RTW1"); //Name des ersten RTW auf der Map
if(vl.GetNumVehicles() == 0)
System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
else
{
Vehicle v(vl.GetVehicle(0));
v.SetSpeed(11.0f); //Wert der Geschwindigkeit des LF der Freiwilligen Feuerwehr
}
VehicleList vl("RTW2"); //Name des zweiten RTW auf der Map
if(vl.GetNumVehicles() == 0)
System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
else
{
Vehicle v(vl.GetVehicle(0));
v.SetSpeed(11.0f); //Wert der Geschwindigkeit des LF der Freiwilligen Feuerwehr
}
VehicleList vl("NEF"); //Name des NEF auf der Map
if(vl.GetNumVehicles() == 0)
System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
else
{
Vehicle v(vl.GetVehicle(0));
v.SetSpeed(13.0f); //Wert der Geschwindigkeit des LF der Freiwilligen Feuerwehr
}
```

Alles anzeigen

Disponenten im Start-Script

Die Disponenten sind Personen, denen wir später im Editor die Alarmierungscommands (wird zu einem späteren Zeitpunkt noch behandelt) zuweisen werden. Wir werden drei verschiedene Disponenten verwenden. Der erste wird alle **Zugalarmierungen** beinhalten, der Zweite nur **Feuerwehr** und der Dritte nur **Rettungsdienst**.

Quellcode

```
PersonList pl("Disponent_1"); //Name des ersten Disponenten
if(pl.GetNumPersons() == 0)
System::Log("Disponent nicht gefunden: %s", pl.GetPerson(0)->GetName());
else
Game::AddToGroup(pl.GetPerson(0), 0); //Disponent auf Codeblock 1
PersonList pl("Disponent_2"); //Name des zweiten Disponenten
if(pl.GetNumPersons() == 0)
System::Log("Disponent nicht gefunden: %s", pl.GetPerson(0)->GetName());
else
Game::AddToGroup(pl.GetPerson(0), 1); //Disponent auf Codeblock 2
PersonList pl("Disponent_3"); //Name des dritten Disponenten
if(pl.GetNumPersons() == 0)
System::Log("Disponent nicht gefunden: %s", pl.GetPerson(0)->GetName());
else
Game::AddToGroup(pl.GetPerson(0), 2); //Disponent auf Codeblock 3
```

Alles anzeigen

Die Namen der Fahrzeuge und Personen im Script müssen mit den Namen der Objekte auf der Map übereinstimmen. Ansonsten kann es zu Fehlfunktionen kommen!

Das fertige Start-Script
Quellcode

```
1. void Start()
2. {
3. VehicleList vl("ELW_Haupt"); //Name des ELW der hauptamtlichen Wache auf der Map
4. if(vl.GetNumVehicles() == 0)
5. System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
6. else
7. {
8. Vehicle v(vl.GetVehicle(0));
9. v.SetSpeed(12.0f); //Wert der Geschwindigkeit des ELW der hauptamtlichen Wache
10. }
11. VehicleList vl("DLK_Haupt"); //Name der DLK der hauptamtlichen Wache auf der Map
12. if(vl.GetNumVehicles() == 0)
13. System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
14. else
15. {
16. Vehicle v(vl.GetVehicle(0));
17. v.SetSpeed(9.0f); //Wert der Geschwindigkeit der DLK der hauptamtlichen Wache
18. }
19. VehicleList vl("HLF_Haupt"); //Name des HLF der hauptamtlichen Wache auf der Map
20. if(vl.GetNumVehicles() == 0)
21. System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
22. else
23. {
24. Vehicle v(vl.GetVehicle(0));
25. v.SetSpeed(10.0f); //Wert der Geschwindigkeit des HLF der hauptamtlichen Wache
26. }
27. VehicleList vl("MTW_FFW"); //Name des MTW der Freiwilligen Feuerwehr auf der Map
28. if(vl.GetNumVehicles() == 0)
29. System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
30. else
31. {
32. Vehicle v(vl.GetVehicle(0));
33. v.SetSpeed(12.0f); //Wert der Geschwindigkeit des MTW der Freiwilligen Feuerwehr
34. }
35. VehicleList vl("LF16_FFW"); //Name des LF der Freiwilligen Feuerwehr auf der Map
36. if(vl.GetNumVehicles() == 0)
37. System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
38. else
39. {
40. Vehicle v(vl.GetVehicle(0));
41. v.SetSpeed(8.0f); //Wert der Geschwindigkeit des LF der Freiwilligen Feuerwehr
42. }
43. VehicleList vl("RTW1"); //Name des ersten RTW auf der Map
44. if(vl.GetNumVehicles() == 0)
45. System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
46. else
47. {
48. Vehicle v(vl.GetVehicle(0));
49. v.SetSpeed(11.0f); //Wert der Geschwindigkeit des LF der Freiwilligen Feuerwehr
50. }
51. VehicleList vl("RTW2"); //Name des zweiten RTW auf der Map
52. if(vl.GetNumVehicles() == 0)
53. System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
```

```
58. else
59. {
60. Vehicle v(vl.GetVehicle(0));
61. v.SetSpeed(11.0f); //Wert der Geschwindigkeit des LF der Freiwilligen Feuerwehr
62. }
63. VehicleList vl("NEF"); //Name des NEF auf der Map
64. if(vl.GetNumVehicles() == 0)
65. System::Log("Fahrzeug nicht gefunden: %s", vl.GetVehicle(0)->GetName());
66. else
67. {
68. Vehicle v(vl.GetVehicle(0));
69. v.SetSpeed(13.0f); //Wert der Geschwindigkeit des LF der Freiwilligen Feuerwehr
70. }
71. PersonList pl("Disponent_1"); //Name des ersten Disponenten
72. if(pl.GetNumPersons() == 0)
73. System::Log("Disponent nicht gefunden: %s", pl.GetPerson(0)->GetName());
74. else
75. Game::AddToGroup(pl.GetPerson(0), 0); //Disponent auf Codeblock 1
76. PersonList pl("Disponent_2"); //Name des zweiten Disponenten
77. if(pl.GetNumPersons() == 0)
78. System::Log("Disponent nicht gefunden: %s", pl.GetPerson(0)->GetName());
79. else
80. Game::AddToGroup(pl.GetPerson(0), 1); //Disponent auf Codeblock 2
81. PersonList pl("Disponent_3"); //Name des dritten Disponenten
82. if(pl.GetNumPersons() == 0)
83. System::Log("Disponent nicht gefunden: %s", pl.GetPerson(0)->GetName());
84. else
85. Game::AddToGroup(pl.GetPerson(0), 2); //Disponent auf Codeblock 3
86. };
87. bool OnLoad()
88. {
89. Start();
90. Process::Kill();
91. return true;
92. }
```

Alles anzeigen

Sollte es bei Scripts zu einer Fehlermeldung oder sogar zu einem Absturz kommen und eine eigenständige Fehlerbehebung nicht möglich sein, besteht die Möglichkeit ein Thema im Bereich Scripting im EMERGENCY-Forum zu eröffnen. Um schnelle Hilfe zu bekommen, sollte jedoch das komplette Script und die Logfile, die sich im Hauptverzeichnis von EMERGENCY 4 befindet, in dem Thema zur Verfügung stellt werden.

Diese Information betrifft alle Probleme im Bereich Scripting, nicht nur Probleme mit dem Startscript.

Die Alarm-Scripts

Ersten Disponenten erstellen

Wie bereits erwähnt wurde, werden wir drei Disponenten verwenden. Der erste Disponent wird alle Arten von Zügen alarmieren können (**Löschzug mit Personenschaden**, **Löschzug ohne Personenschade**, **dringende Türöffnung usw.**).Über den zweiten Disponent werden alle Feuerwehrfahrzeuge gesondert alarmiert. Jedoch muss hier eine Unterscheidung zwischen **hauptamtlichen Kräften** und **freiwilligen Kräften** getroffen werden. Berufsfeuerwehrleute verbringen ihren Tag auf der Wache und müssen somit nicht von zu Hause an das Gerätehaus laufen.Der dritte Disponent wird nur die Alarm-Commands der **Rettungsdienstfahrzeuge** beinhalten. Da einsatzbereite Fahrzeuge des Rettungsdienst eigentlich immer mit Kräften besetzt sind, die sich vor Ort befinden, wird hier das gleiche System wie bei der hauptamtlichen Wache zum Einsatz kommen. Nachfolgend der Grundaufbau des Scripts - hier werden später noch Änderungen vorgenommen.

Quellcode

```
const char ICON[] = "";
const char CURSOR[] = "";
const char VEHICLE_NAME[] = "";
const char VIRTUAL_OBJECT[] = "";
const char SOUND[] = "";
const char MESSAGE[] = "";
const char NUM_PERSON[] = "";
const char PROTO_PERSON[] = "";
float WAIT_PERSON = 5;
float WAIT_VEHICLE = 20;
bool vehiclefull = false;
object Alarm_ELW : CommandScript
{
Alarm_ELW()
{
SetIcon(ICON);
SetCursor(CURSOR);
SetValidTargets(TARGET_STREET);
SetValidTargets(TARGET_FLOOR);
}
bool CheckTarget(GameObject *Caller, Actor *Target, int childID)
{
return true;
}
void PushActions(GameObject *Caller, Actor *Target, int childID)
{
Vector APos = Game::GetCommandPos();
VehicleList vl(VEHICLE_NAME);
if(gol.GetNumObjects()==0)
{
System::Log("%s nicht gefunden", VEHICLE_NAME);
return;
}
Vehicle v(vl.GetVehicle(0));
ActorList al(VIRTUAL_OBJECT);
if(al.GetNumActors()==0)
{
System::Log("%s nicht gefunden", VIRTUAL_OBJECT);
return;
}
Vector SPos = al.GetActor(0)->GetPosition();
v.SetUserData(1);
Audio::PlaySample(SOUND);
Mission::PlayHint(MESSAGE);
for(int i = 1; i <= NUM_PERSON; i++)
{
Person p = Game::CreatePerson(PROTO_PERSON, "Unnamed");
p.SetRole(ROLE_SQUAD);
p.SetUpgradeLevel(3);
Game::FindFreePosition(&p, spawnp, 100.f);
p.SetPosition(SPos);
p.PushActionShowHide(ACTION_NEWLIST, true);
p.PushActionWait(ACTION_APPEND, WAIT_PERSON);
p.PushActionShowHide(ACTION_APPEND, false);
p.PushActionMove(ACTION_APPEND, v.GetPosition());
```

```
65. p.PushActionEnterCar(ACTION_APPEND, &v);
66. if(i == NUM_PERSON)
67. vehiclefull = true;
68. }
69. if(vehiclefull)
70. {
71. v.PushActionWait(ACTION_NEWLIST, WAIT_VEHICLE);
72. v.PushActionMove(ACTION_APPEND, Apos);
73. }
74. }
75. }
76. };
```

Alles anzeigen

Eigentlich muss an diesem Script nicht viel verändert werden. Änderungen sind nur im oberen Teil nötig. Das Script ist so aufgebaut, dass alle wichtigen Variablen, wie Ausrückedauer, Prototyp der Person, Fahrzeugname usw. in den ersten 8 Zeilen einfach abgeändert werden können.

Weitere Disponenten erstellen

Der dritte Disponent ist für die Kräfte des Rettungsdienstes zuständig. Wir beginnen also exemplarisch mit einem Rettungswagen.

Quellcode

```
1. const char ICON[] = "RTW1";
2. const char CURSOR[] = "RTW1";
3. const char VEHICLE_NAME[] = "";
4. const char VIRTUAL_OBJECT[] = "";
5. const char SOUND[] = "";
6. const char MESSAGE[] = "";
7. const char NUM_PERSON[] = "";
8. const char PROTO_PERSON[] = "";
```

Die Variablen *ICON* und *CURSOR* sind die Namen derGrafiken , die in dem Bedienfeld der Personen oder Fahrzeuge auftauchen. Hier muss nur ein Name angegeben werden, da das Spiel die passenden Bilddateien automatisch aus dem Icon- und Cursor-Ordner bezieht.

```
const char VEHICLE_NAME[] = "RTW1";
```

Ähnlich wie be beim Script-Script, wird hier der Name des Fahrzeugs auf der Map festgelegt. Somit weiß das Spiel, zu welchem Fahrzeug die Personen bei einer Alarmierung laufen sollen.

```
const char VIRTUAL_OBJECT[] = "Spawn_RTW1";
```

Die Variable *VIRTUAL_OBJECT* ist der Name des virtuellen Objekts, an dem die Personen generiert (gespawnt) werden.

Quellcode

```
1. const char SOUND[] = "mod:Audio/Alarm/Piepser.wav";
2. const char MESSAGE[] = "Alarm für den Rettungswagen 1";
```

An dieser Stelle hat man die Möglichkeit eine Sounddatei -zum Beispiel von einem Alarmgong oder einen Melder - abspielen zu lassen. Hierzu muss der Pfad zu einer .wav-Datei angegeben werden. Normalerweise werden Audiodateien im Ordner *Audio* hinterlegt, jedoch ist das ziemlich egal, solange der angegebene Pfad im Script korrekt angegeben ist.Bei der Variable *Message* kann ein Text eingegeben werden, der unmittelbar nach der Alarmierung in dem blauen Fenster am oberen Bildschirmrand als Laufschrift eingeblendet wird.

Quellcode

```
1. const char NUM_PERSON[] = "2";
```

2. const char PROTO_PERSON[] = "mod:Prototypes/Persons/Ambulance/paramedic.e4p";

Hier wird die Anzahl der zu erstellenden Personen eingetragen. Bei einem RTW sind das im Normalfall zwei Personen. Darunter wird der Pfad zum Prototyp zur Personen, die erstellt werden sollen, angegeben.
Dieser Prozess des Anpassens muss nun sowohl für den zweiten RTW, das NEF, den ELW, die DLK und das HLF der ständig besetzten Wache durchgeführt werden.

Sollte bei einem späteren Test des Scripts auffallen, dass keine Personen gespawnt werden, ist zuerst die Richtigkeit des Pfades zu prüfen.