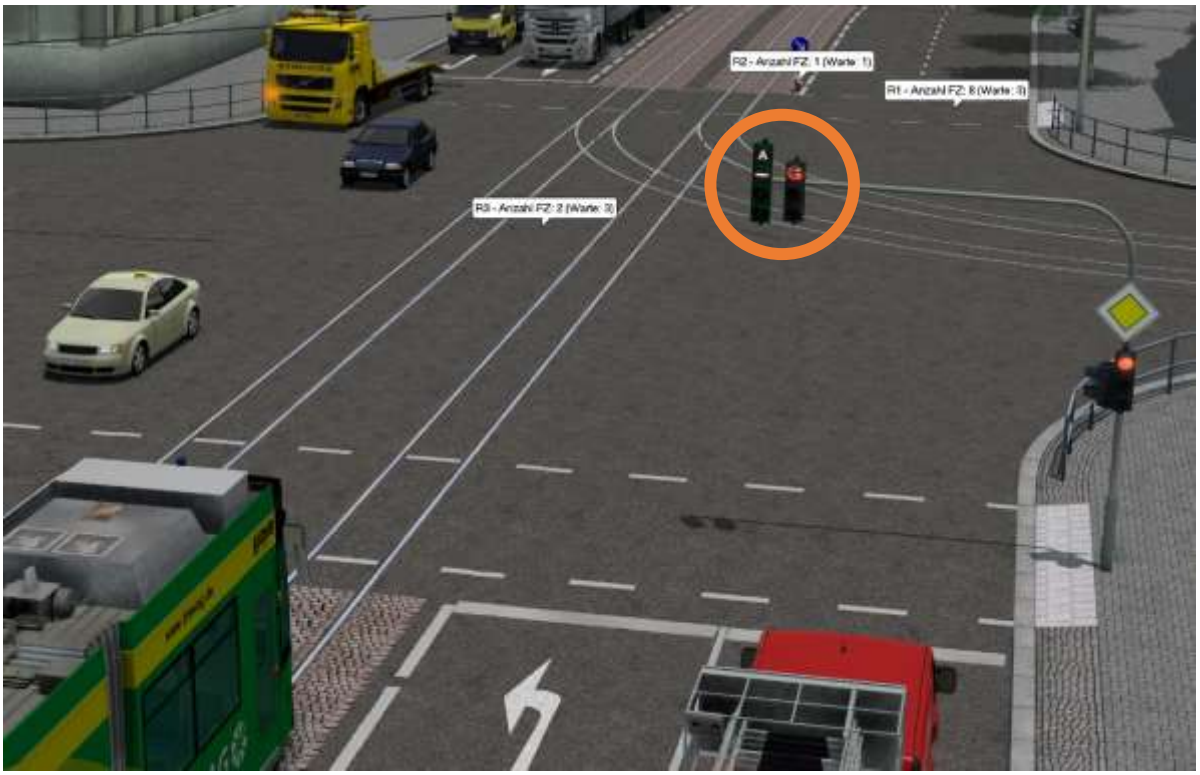
1 Kurzbeschreibung

Diese Anlage zeigt was mit LUA und EEP im Straßenverkehr alles möglich ist.

Das Herzstück der Anlage ist nicht die 3D-Ansicht, sondern ich möchte hiermit vorstellen, was ich in LUA ersonnen habe. Die Idee ist, andere zu inspirieren und aus ihrem Feedback zu lernen.



1.1 Ampeln werden idealerweise nach Anzahl der Fahrzeuge gesteuert

Die Anlage enthält zwei komplexe Kreuzungen, welche vollautomatisch so geschaltet werden, dass die Fahrtrichtungen mit den meisten Fahrzeugen auf grün geschaltet werden.

1.2 Schaltungen für Straßenbahnen sollen bei Bedarf nur nach Anforderung erfolgen

Somit müssen Straßenbahnsignale auf Wunsch nur dann geschaltet werden, wenn eine Anforderung durch eine Bahn vorliegt.

2 Installation

Die Dateien im Verzeichnis LUA der ZIP-Datei müssen in das bereits existierende Verzeichnis LUA im Installationsverzeichnis von EEP – z.B. nach C:\Spiele\EEP13\LUA

3 Notwendige Modelle

Damit die Funktion dieser Anlage sinnvoll angezeigt werden kann, müssen folgende Modelle zwingend vorhanden sein:

Großstadtstraßensystem AS3: https://eepshopping.de/

- 1Spur-Großstadtstraßen-System-Grundset (V10NAS30002)
- 1Spur-Ergänzungsset: https://www.eepforum.de/filebase/file/215-freeset-zu-meinem-1spur-strassensystem/

Ampeln als Signale: https://eepshopping.de/

- Ampel-Baukasten für mehrspurige Straßenkreuzungen (V80NJS20039)

Straßenbahnampeln als Immobilien: http://eep.euma.de/download/

- Straßenbahnsignale als Immobilien (V10MA1F011)
- Straßenbahnsignale als Immobilien (V80MA1F010)

4 Allgemeines zu den LUA-Skripten

4.1 Ziel der Programmierung

- **Die Skripte sollen so aufgeteilt werden, dass sie wiederverwendbar sind**
  Idealerweise sagt man in einer neuen Anlage nur noch, wie z.B. eine Kreuzung aussieht und muss sich nicht um den Rest kümmern.

- **Die Skripte sollen möglichst schnell Fehler melden**
  Die doppelte Verwendung von Speicher-IDs soll nicht möglich sein. Verwendet man falsche Parameter in LUA soll dies schnell aufgedeckt werden.

- **Die Skripte sollen ohne EEP testbar sein**
  Es soll ein Testskript geben, so dass die eigenen Skripte OHNE EEP laufen können – man soll z.B. selbst Kontaktpunktfunktionen aufrufen können und den Ablauf in EEP simulieren, damit die Skripte getestet werden, BEVOR sie in EEP importiert werden

5 Beschreibung der einzelnen LUA-Skripte

5.1 EEP-13-die-Moderne-07.lua (Anlagenskript)

Das Skript der Anlage enthält in erster Linie nur den Aufruf zum Laden des eigentlichen Skriptes und legt fest, welche Ausgaben zur Hilfestellung in LUA erfolgen sollen:

- **AkDebugInit = false**
  Hier kann „false“ auf „true“ geändert werden, wenn während der Initialisierung erweiterte Informationen angezeigt werden sollen (siehe die folgenden Einstellungen zur Ausgabe im LUA Log)

- **AkScheduler.debug = false**
  Hier kann „false“ auf „true“ geändert werden, wenn die Planung späterer Aktionen im LUA Log angezeigt werden sollen.

- **AkStorage.debug = false**
  Hier kann „false“ auf „true“ geändert werden, wenn Informationen zum Speichern und Laden im LUA Log angezeigt werden sollen.

- **AkAmpel.debug = false**
  Hier kann „false“ auf „true“ geändert werden, wenn Informationen zum Schalten von Ampeln im LUA Log angezeigt werden sollen.

- **AkKreuzung.debug = false**
  Hier kann „false“ auf „true“ geändert werden, wenn Informationen zum Schalten von Ampeln im LUA Log angezeigt werden sollen.

- **AkKreuzung.showAnforderungenAlsInfo = false**
  Hier kann „false“ auf „true“ geändert werden, wenn der Zähler der an einer Richtung wartenden Fahrzeuge (Anforderungen) und die Wartezyklen als Information am Signal angezeigt werden sollen. – Angezeigt werden dann die Richtung, die Anzahl der Fahrzeuge für diese Richtung und die Wartezeit

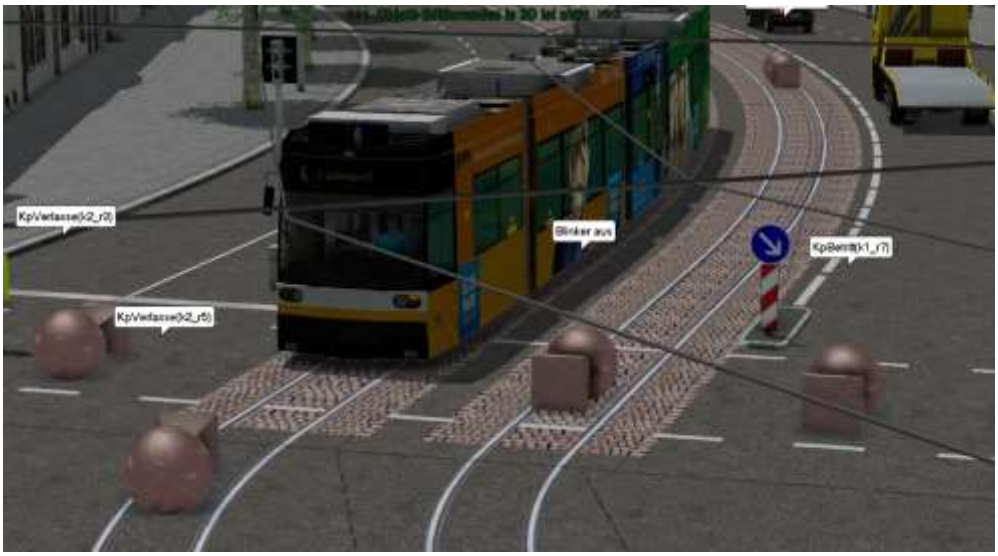
- **AkKreuzung.showSchaltungAlsInfo = false**
  Hier kann „false“ auf „true“ geändert werden, wenn am Signal die aktuell geschaltete Phase angezeigt werden soll (rot wird nicht angezeigt)

5.2 EEP-13-die-Moderne-07-main.lua

Dieses Skript besteht im groben aus 3 Teilen:

- Bereitstellen der Kontaktfunktionen
    - **KpBetritt(richtung)**
      Soll durch einen Kontaktpunkt immer dann aufgerufen werden, wenn ein Fahrzeug eine Richtung betritt, d.h. an eine bestimmte Ampel heranfährt.

    - **KpVerlasse(richtung, signalaufrot)**
      Soll durch einen Kontaktpunkt immer dann aufgerufen werden, wenn ein Fahrzeug eine Richtung verlässt, d.h. die Ampel passiert hat. Wenn signalaufrot „true“ ist, dann wird die Ampel für diese Richtung auf rot geschaltet. Dies ist wichtig für Ampeln, die z.B. vor Straßenbahnweichen stehen.



- **Definieren von Kreuzungen, Schaltungen und Richtungen** der Anlage (wie diese zusammenhängen, siehe Infos zu AkTrafficLightsFunctions.lua)

- **Die Hauptfunktion EEPMain()**
    - Aufruf von AkSchaltungStart() aus AkTrafficLightsFunctions.lua
    - Aufruf von AkScheduler:run() aus AkScheduler.lua

5.3 EEP-13-die-Moderne-07-test.lua

Diese Datei kann ohne EEP direkt in LUA gestartet werden, so dass die Skripte selbst schnell auf Korrektheit geprüft werden können.

Diese Datei besteht aus den folgenden Teilen:

1. **package.path = …**
   Setzen der Pfade, in denen nach LUA Skripten gesucht werden soll.
2. **require 'AkEepFunctions'**
   Einbinden der Datei AkEepFunctions – damit kann stehen dem eigentlichen Skript EEP-13-die-Moderne-07-main.lua die EEP-Funktionen zur Verfügung
3. **require 'EEP-13-die-Moderne-07-main'**
   Einbinden der Datei **'EEP-13-die-Moderne-07-main.lua'**
4. **local function run()**
   Diese Funktion kann innerhalb dieser Datei verwendet werden, um die Zeit vorzustellen und dann EEPMain() aufzurufen, also wie in EEP selbst, nur nicht so häufig
5. Aufrufen diverser Funktionen, um die Kreuzungen zu testen

5.4 AkEepFunctions.lua

Diese Datei stellt die Hauptfunktionen von EEP bereit, so dass man seine (Test-)Dateien auch ohne EEP laufen lassen kann. Sie merkt sich z.B. die Zustände von

- EEPSetSignal und EEPGetSignal
- EEPSetSwitch und EEPGetSwitch
- EEPSaveData und EEPLoadData

5.5 AkDebug.lua

Hier verstecken sich die Hilfsfunktionen zur Anzeige der Debug-Nachrichten

5.6 AkTrafficLightsFunctions.lua

Hier werden Kreuzungen wie folgt geschaltet:

- Jede AkKreuzung hat mehrere AkKreuzungsSchaltung
- Jede AkKreuzungsSchaltung hat mehrere AkRichtungen
    - Alle Richtungen einer Schaltung bekommen die gleiche Phase geschaltet
    - Die Richtungen einer Schaltung sollten sich nicht überschneiden, so dass die Fahrzeuge nicht ineinander fahren
- Jede AkRichtung hat eine oder mehrere AkAmpel
    - Alle AkAmpel für eine AkRichtung immer gleichzeitig geschaltet (Auf eine der Phasen rot, rotgelb, gruen, gelb, fussgaenger)
    - Jede Richtung hat einen Zähler für Fahrzeuge und einen für die Wartezeit. Richtungen mit vielen wartenden Fahrzeugen oder hoher Wartezeit werden bevorzugt behandelt.
- Die Schaltungen werden über den AkScheduler umgeschaltet, dieser startet z.B. die Rot-Phase erst wenn mind. 3 Sekunden nach dem Umschalten auf die Gelb-Phase und dem letzten LUA-Aufruf vergangen sind
- Wird eine Anlage geladen, werden alle Richtungen zunächst auf rot geschaltet, dann beginnt der reguläre Betrieb.

Diese Datei ist das Herzstück der Ampelschaltungen und besteht aus den folgenden Abschnitten:

- **require("AkScheduler")**
  lädt die Datei AkScheduler.lua
- **require("AkStorage")**
  lädt die Datei AkStorage.lua
- **AkPhase**
  hält die Phasen der Ampel: rot, rotgelb, gelb, grün, Fußgänger
- **AkAmpelModell**
  Hier können Ampelmodelle angegeben werden:
    - AkAmpelModell:neu() gibt den Namen des Modells und die Signalstellungen für die Ampelphasen an (kann man für eigene Ampelmodelle verwenden)
    - Danach folgen einige Ampelmodelle aus den Sets: V80NJS20039 und V10MA1F011
- **AkStrabWeiche**
    - Zeigt die Weichenstellung einer Straßenbahnweiche im Modell V10MA1F011 an – die Immobilien immo1, immo2 oder immo3 werden je nach Weichenstellung 1, 2 oder 3 der angegebenen Weiche weiche_id beleuchtet.
- **AkAmpel**
    - AkAmpel:new()
      Definition eines Ampelmodells mit signal_id und ampel_typ
      Werden hierbei die Parameter rotImmo, gruenImmo und gelbImmo angegeben, so wird beim Schalten dieser Ampel, das Licht für rot, grün und gelb eingeschaltet (siehe Doku für V10MA1F011)
      Werden hierbei der Parameter activeImmo angegeben, so wird das Licht für diese Immobilie angeschaltet, wenn ein Fahrzeug in der Richtung dieser Ampel wartet (z.B. Immobilie für „A“ in V10MA1F011)
    - Kann eine Ampel schalten
      AkAmpel:schalte(phase, grund)
    - Kann die Anzahl der Anforderungen setzen
      AkAmpel:setAnforderung(anforderung, richtung, anzahl)

- **AkKreuzungsSchaltung**
    - Fügt mehrere Richtungen zu einer Schaltung zusammen
- **AkRichtung**
    - Zählt die Fahrzeuge einer Richtung
    - Hält die Ampeln einer Richtung
    - Schaltet die Ampeln einer Richtung oder zählt deren Anforderungen hoch und runter
    - Eine Richtung benötigt einen Namen, eine Speicher-ID in EEP und Ampeln: AkRichtung:new(name, eepSaveId, ...)
- **AkKreuzung**
    - Hält Zustand über die aktuelle Schaltung
    - Kann die Schaltung mit der höchsten Priorität berechnen: AkKreuzung:getNextSchaltung()
    - Kann Schaltungen hinzufügen AkKreuzung:addSchaltung(schaltung)
- **AkSchaltungStart**
    - Startet die Schaltung in allen Kreuzungen, wenn die Kreuzung fertig ist:
        1. Wenn eine Kreuzung noch nicht fertig ist, dann weiter
        2. Mit AkKreuzung:getNextSchaltung() schauen, welche Schaltung als nächstes dran ist
        3. Dann wird berechnet, welche Richtungen auf rot und auf grün geschaltet werden müssen
        4. Dann wird eingeplant AkScheduler:addAction(sekunden, funktion):
           (+0 s) Aktuelle Schaltung der Kreuzung ihre Fußgänger auf Rot schalten,
           (+3 s) Aktuelle Schaltung der Kreuzung auf Gelb
           (+2 s) Aktuelle Schaltung der Kreuzung auf Rot
           (+3 s) Neue Schaltung auf RotGelb und Fussgänger auf Grün
           (+1 s) Neue Schaltung auf Grün
           (+X s) Kreuzung fertigschalten (X ist die GruenZeit der Kreuzung in Sekunden)

5.7 AkScheduler.lua

Diese Datei führt einmal geplante Aktionen in EEP als Funktionen aus. Die geplanten Funktionen überleben das Beenden von EEP nicht und müssen nach dem Programmstart erneut eingeplant werden.

Diese Datei besteht aus den folgenden Teilen:

1. **AkTime…()**
   Funktionen, die entweder die EEPZeit, oder die aktuelle Rechnerzeit zurückgeben
2. **function AkScheduler:run()**
   Sollte bei jedem Aufruf von EEPMain() aufgerufen werden. Diese Funktion prüft die Zeit und führt alle Aktionen aus, die bereits ausgeführt werden dürfen, da die Startzeit erreicht oder auch überschritten ist (letzteres ist wichtig für die 10fache Beschleunigung in EEP).
3. **function AkScheduler:addAction(offsetSeconds, newAction, previousAction)**
   Plant das Ausführen der Funktion newAction.
   a. Ist **nur offsetSeconds** angegeben, wird die Funktion nach dieser Anzahl von Sekunden eingeplant
   b. Sind **sowohl offsetSeconds als auch previousAction** angegeben, so wird versucht die neue Aktion nach dieser Anzahl von Sekunden NACH der Funktion previousAction eingeplant