JAM

Simulationsumgebung zur Steuerung einer Lichtsignalanlage mit animiertem Verkehrsfluss

Version 0.9.1

vom 7.02.2009

by Marco Haase

kontakt@viktorianer.de

Inhalt

1. Einführung

1.1 Was ist JAM?

JAM ist eine Simulationsumgebung, die es erlaubt, eine vierteilige Lichtsignalanlage an einer Verkehrskreuzung durch selbst zu entwickelnde Skripte (Programme) zu steuern. Zusätzlich können die Fahrspuren einzeln mit einem bestimmten Verkehrsfluss versehen werden, der animiert dargestellt wird, so dass unmittelbar erkennbar wird, wie sich ein bestimmtes Steuerprogramm auf die Verkehrsentwicklung auswirkt.

Zwar sind verschiedene Ergebnisse wissenschaftlicher Studien zur Verkehrsentwicklung in die Entwicklung von JAM eingeflossen, um eine einigermaßen realitätsnahe Simulation zu erreichen, dennoch sind die Ergebnisse (bestenfalls) realitäts*nah*. JAM hat nicht den Anspruch, ein reales Verkehrsgeschehen zu simulieren. Im Vordergrund steht die Möglichkeit der Steuerung einer Lichtsignalanlage in Echtzeit.

Entwickelt wurde JAM speziell für den Einsatz im Informatikunterricht der Sekundarstufe I. Der Name ist angelehnt an die englische Bezeichnung „traffic jam" für „Verkehrsstau" – diesen zu vermeiden ist ja unter anderem das Ziel eines gelungenen Steuerungsprogramms für eine Lichtsignalanlage.

1.2 Systemvoraussetzungen und Lizenz

JAM ist als ausführbare exe-Datei für **Windows**-Systeme frei und kostenlos verfügbar. Einfach die Datei `jam.exe` starten und fertig. Eine Installation ist nicht erforderlich, Veränderungen am System werden nicht vorgenommen. Die jeweils aktuelle Fassung kann man unter *www.viktorianer.de/info/info-ampel.html* herunterladen.

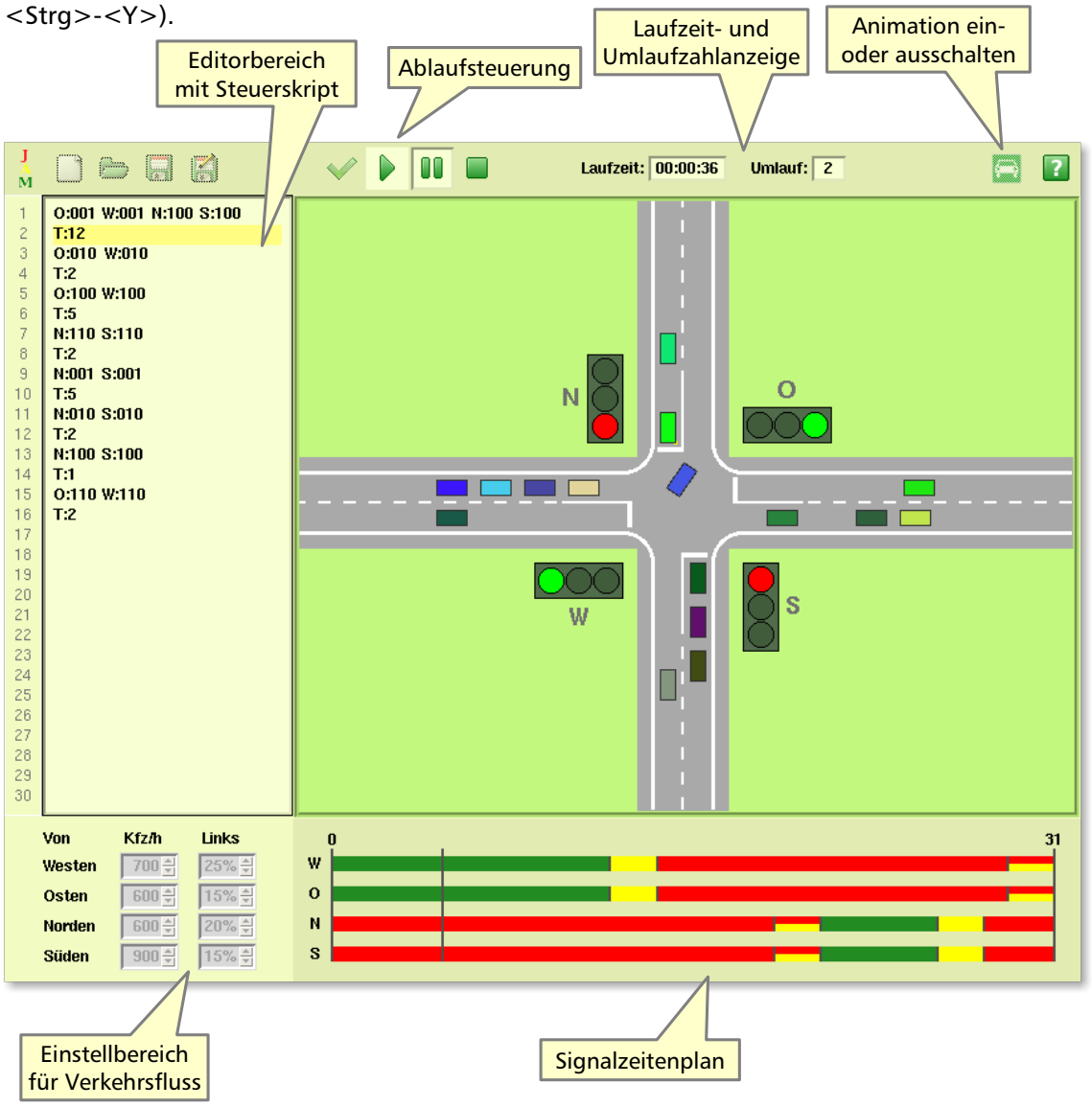
Da JAM in der Programmiersprache Python entwickelt wurde, die kostenlos auch für **Linux** und **MacOS** verfügbar ist, steht einem Einsatz auf diesen Betriebssystemen grundsätzlich nichts im Wege. Dazu benötigt man dann eine funktionsfähige Python-Installation (Python 2.x ab 2.5 oder 3.x) samt Tkinter und das Python-Skript (zumindest im Bytecode). Bei begründetem Interesse bitte per E-Mail Kontakt mit mir aufnehmen.

Die Anforderungen an die Hardware hängen von den vom Anwender eingestellten Eckdaten für die Simulation des Verkehrsflusses ab. Bei hoher Verkehrsdichte aus allen Richtungen und einem hohen Anteil an abbiegenden Fahrzeugen erhöht sich der Rechenaufwand beträchtlich, so dass bei schwacher Hardware die Animationsgeschwindigkeit in einem Maße nachlassen kann, dass eine realitätsnahe Entwicklung nicht mehr simuliert werden kann. Der Anwender erhält in diesem Fall einen entsprechenden Hinweis vom Programm und hat die Möglichkeit, die Simulation ohne animierten (aber dennoch simulierten!) Verkehrsfluss durchzuführen, um auch bei leistungsschwächerer Hardware noch eine realitätsnahe Verkehrsentwicklung zu erreichen.

2. Grundlagen der Bedienung

Die Benutzeroberfläche von JAM ist intuitiv erfassbar. Über eine **Symbolleiste** am oberen Rand sind alle Funktionen steuerbar – Tooltipps helfen bei Unsicherheit weiter.

Der integrierte **Editor** verfügt über die typischen Funktionen eines einfachen Texteditors. Steuerskripte können geladen und gespeichert, editiert und mittels „copy/cut & paste" über die Zwischenablage kopiert, ausgeschnitten und eingefügt werden. Dazu verwendet man die üblichen Shortcuts <Strg>-<C>, <Strg>-<X> und <Strg>-<V> (bzw. auf Linux-Systemen <Strg>-<Y>).



Bevor ein Steuerskript ausgeführt werden kann, wird es auf syntaktische und semantische Korrektheit geprüft – dies geschieht beim Klick auf das Symbol ✓. Erst nach erfolgreicher Überprüfung wird der **Einstellbereich** für die Festlegung der Verkehrsparameter und die Ablaufsteuerung freigeschaltet und der **Signalzeitenplan** eingeblendet. Die Ausführung des Steuerprogramms startet man dann durch einen Klick auf ▶. Der Programmablauf kann jederzeit und beliebig oft durch Betätigung der Pause-Taste ⏸ angehalten und fortgesetzt werden. Das endgültige Beenden des Ablaufs erfolgt durch einen Klick auf ■.

3. Steuerung der Lichtzeichenanlage

3.1 Grundlegendes

Die vierteilige Lichtsignalanlage an der Kreuzung wird gesteuert über ein Skript, das für jede der Ampeln getrennt über **Schaltsignale** festlegt, welchen Signalzustand die jeweilige Ampel einnehmen soll. Weiterhin wird über **Zeitsignale** festgelegt, wie lange dieser Zustand anhalten soll, bis das folgende Schaltsignal umgesetzt wird. Schaltsignale und Zeitsignale werden durch entsprechende **Schaltbefehle** bzw. **Zeitbefehle** des Steuerskripts an die Lichtsignalanlage gesendet.

Das Steuerskript im Editor läuft dabei in einer Endlosschleife – diese muss nicht ausdrücklich programmiert werden. Ist das Skript am Ende angekommen, fängt es wieder von vorne an. In der Symbolleiste befindet sich – neben einer sekundengenauen **Laufzeitanzeige** in Echtzeit – ein **Umlaufzähler**, der einem das Selberzählen erspart.

Mit Hilfe der oben schon beschriebenen **Funktionen zur Ablaufsteuerung** in der Symbolleiste kann ein solches Steuerprogramm zu jedem beliebigen Zeitpunkt angehalten und später an genau dieser Stelle fortgesetzt werden.

3.2 Syntax und Semantik

Der Aufbau eines gültigen Steuerskripts lässt sich am einfachsten anhand eines konkreten Beispiels erläutern. Hier zunächst das Beispiel, auf der nächsten Seite folgen Erläuterungen.

```
1   O:001 N:100     # Ost auf grün, Nord auf rot
2   T:8             # 8 Sekunden so halten
3   O:010           # Ost auf gelb
4   T:2             # 2 Sekunden so halten
5   O:100           # Ost auf rot
6   T:5             # 5 Sekunden so halten
7   N:110           # Nord auf rot-gelb
8   T:2             # 2 Sekunden so halten
9   N:001           # Nord auf grün
10  T:5             # 5 Sekunden so halten
11  N:010           # Nord auf gelb
12  T:2             # 2 Sekunden so halten
13  N:100           # Nord auf rot
14  T:1             # 1 Sekunde so halten
15  O:110           # Ost auf rot-gelb
16  T:2             # 2 Sekunden so halten
17  # jetzt geht es in Zeile 1 weiter
```

Erläuterungen zum Beispiel:

Ein **Schaltbefehl** besteht aus der Kennung für die anzusteuernde Ampel entsprechend der jeweiligen Himmelsrichtung (siehe Kreuzungsdarstellung) und der Angabe des Schaltsignals für diese Ampel, getrennt durch einen Doppelpunkt. Schaltsignale sind eine Folge von 3 Bits, von denen das höchstwertige (linke) die rote Lampe, das mittlere die gelbe Lampe und das niederwertige (rechte) die grüne Lampe schaltet – 1 ist an, 0 ist aus.

Wie man in Zeile 1 sehen kann, ist es zulässig, mehrere Schaltbefehle in einer Zeile unterzubringen. Dann werden die einzelnen Befehle durch Leerzeichen oder Tabulator getrennt. Pro Zeile darf eine Ampel jedoch nur höchstens einmal geschaltet werden.

Ein **Zeitbefehl** beginnt mit der Kennung T, es folgt ein Doppelpunkt und die gewünschte Schaltzeit in Sekunden. Zulässig sind nur ganzzahlige Werte, die allgemeine Mindestschaltdauer beträgt 1 Sekunde, die Höchstschaltdauer für die Grün- oder Rotphase 300 Sekunden, für die Rot-Gelb-Phase 2 Sekunden und für die Gelb-Phase 4 Sekunden. Im Unterschied zu den Schaltbefehlen müssen Zeitbefehle grundsätzlich allein in einer Zeile stehen.

Von Schalt- und Zeitbefehlen abgesehen können an beliebigen Stellen **Leerzeilen** und **Kommentare** eingefügt werden. Alles, was nach einer Raute „#“ steht, wird nicht ausgewertet. Damit ist die Syntax auch bereits vollständig erklärt.

Außer der syntaktischen Richtigkeit gibt es weitere Aspekte, die bei der Erstellung eines Steuerskripts zu beachten sind:

- Auf jeden Schaltbefehl (bezogen auf *eine* Ampel) muss ein Zeitbefehl folgen, bevor ein weiterer Schaltbefehl an diese Ampel geschickt werden darf.
- Ein Steuerskript darf nicht mit einem Zeitbefehl beginnen – zunächst muss ein Schaltbefehl gesendet werden.
- Es ist zulässig, in einem Steuerprogramm nicht alle Ampeln zu schalten. In solchen Fällen erhalten diese Ampeln ein „Dauer-Rot“ und die Einstellung des Verkehrsflusses für die entsprechende Spur bleibt inaktiv.
- Für jede Ampel gilt, dass die auf ein Schaltsignal folgenden Zeitsignale aufaddiert werden, bis für diese Ampel ein neues Schaltsignal gesendet wird. Im obigen Beispiel bedeutet das z.B., dass die Ampel Ost insgesamt 15 Sekunden rot zeigt – auf den Schaltbefehl in Zeile 5 folgt der nächste Schaltbefehl für diese Ampel ja erst in Zeile 15.
- Steuerskripte, die dazu führen, dass Ampeln sich kreuzender Richtungen gleichzeitig die Weiterfahrt erlauben, sind ungültig.

4. Der Verkehrsfluss

Das eigentliche Anliegen von JAM ist es, eine Umgebung zur Entwicklung von Steuerskripten für eine Lichtsignalanlage zu schaffen, die dann auch ausgeführt werden können. Ein grundlegendes Qualitätsmerkmal eines solchen Steuerprogramms ist zunächst seine Gültigkeit, d.h. seine Fähigkeit den Ablauf der Signale so zu steuern, dass für alle Richtungen ein Verkehrsfluss ermöglicht wird und gleichzeitig Kollisionen vermieden werden. Ist das geschafft, dann beginnt die Phase der Optimierung, d.h. der Anpassung der Schaltzeiten und der Schaltphasen an die vorhandenen Verkehrsverhältnisse. Damit dies gelingt, muss das Verkehrsgeschehen ebenfalls simuliert werden – mindestens dadurch, dass die Anzahl der auf einer Fahrspur (vor der Ampel) befindlichen Fahrzeuge eingeblendet wird. Eine Animation des Verkehrsflusses erhöht zwar nicht den funktionalen Nutzen gegenüber einer reinen Simulation mit Fahrzeugzahlen, macht das Programm und das Arbeiten damit aber fraglos attraktiver und darum bietet JAM diese Möglichkeit.

Im Einstellbereich der Bedienoberfläche lässt sich die Verkehrsdichte in Fahrzeugen pro Stunde pro Fahrtrichtung im Bereich von 0 bis 1500 festlegen. Der Höchstwert liegt etwas oberhalb dessen, was im innerstädtischen Bereich als maximale Durchflusskapazität angesehen wird (*www.upi-institut.de/_handschuhsheim/neuenheimer_feld.htm*). Der Einstellbereich wird erst zur Bedienung freigegeben, nachdem ein als gültig identifiziertes Steuerprogramm vorliegt und kann während des Ablaufs nicht mehr verändert werden. Erst nach Abbruch des Steuerprogramms sind Änderungen am Verkehrsfluss möglich. Für Fahrspuren, die im Steuerprogramm gar nicht vorkommen, ist ebenfalls keine Einstellung möglich.

Eine vom Programm zufällig ermittelte Zahl an Fahrzeugen biegt an der Kreuzung jeweils rechts ab, der Anteil der Linksabbieger am Verkehrsaufkommen pro Spur kann vom Anwender im Bereich von 0% bis 95% eingestellt werden.

Die Geschwindigkeit der Fahrzeuge entspricht – bezogen auf den Maßstab – der innerorts üblichen Geschwindigkeit, sofern die dem Programm zur Verfügung stehenden Ressourcen ausreichen, um den internen Systemtakt von 55 Hz in etwa zu erreichen. Sinkt der Systemtakt unter 35 Hz, erhält der Anwender einen Hinweis darauf und die Empfehlung, die Animation abzuschalten und die Simulation ohne animierten Verkehrsfluss durchzuführen, da ansonsten die Verkehrsentwicklung keine brauchbaren Ergebnisse mehr liefert.

5. Sonstiges

5.1 Fenstergröße anpassen

Nach dem Start hat die Arbeitsoberfläche von JAM etwa die Maße 860 x 680 Pixel – ein weiteres Verkleinern ist zwar theoretisch möglich, allerdings werden dadurch Elemente der Oberfläche verdeckt und/oder abgeschnitten. Eine Vergrößerung bis zur Maximierung bewirkt – je nach Auflösung des Monitors – eine Verlängerung des Editorbereichs und des Bereichs für den Signalzeitenplan, jedoch aus Gründen der Performance keine Vergrößerung des Kreuzungsbereichs.

5.2 Log-Dateien

Beginnend mit dem Start von JAM werden bis zum Beenden des Programms wichtige Eckdaten der Bedienung und des Ablaufs im Hintergrund protokolliert und – sofern die Zugriffsrechte des Anwenders dafür ausreichen – in einer log-Datei `jam-yyyymmtthhmmss.log` gespeichert, die im gleichen Verzeichnis liegt wie das Programm `jam.exe`. Diese log-Dateien dienen in erster Linie dazu, bei aufgetretenen – tatsächlichen oder angeblichen – Fehlfunktionen anhand des aufgezeichneten Protokolls der Sache auf den Grund gehen zu können. In solchen Fällen bitte die entsprechende log-Datei per E-Mail an mich schicken und die beobachtete Fehlfunktion beschreiben: *kontakt@viktorianer.de*

5.3 Bekannte Fehler

Ganz gelegentlich kommt es anscheinend zu „Kollisionen“ auf der Kreuzung, bei denen ein Linksabbieger einem Entgegenkommenden die Vorfahrt nimmt und diesen einfach „überfährt“. Das sollte zwar eigentlich nicht passieren, aufgrund der Eigenart des Programms sind diese seltenen Zustände aber nur sehr schwer reproduzierbar und infolgedessen auch nur mit größerem (zeitlichen) Aufwand eine Lösung dafür zu entwickeln. Da der Animation des Verkehrsflusses aber nur eine ungeordnete Bedeutung zukommt, halte ich das für verschmerzlich (möglicherweise erhöht es sogar die Motivation der Schüler, wenn nicht auszuschließen ist, dass ein „Unfall“ zu beobachten sein könnte ...)