**Verkehrssimulation**

**Beschreibung der Architektur**

Die Verkehrssimulation teilt sich in folgende 6 Teile auf:

1. Design
2. Verkehrsteilnehmer
3. Ampelsteuerung
4. Straßennetz
5. Straßenverkehrsregeln
6. Datenhaltung

Mit Ausnahme des Designs wird jeder Teilbereich in einer separaten DLL bearbeitet. Das Design selbst wird mit … realisiert.

In den folgenden Abschnitten wird beschrieben, welches DLL für welche Funktion verantwortlich sein wird und wie die DLLs miteinander kommunizieren.

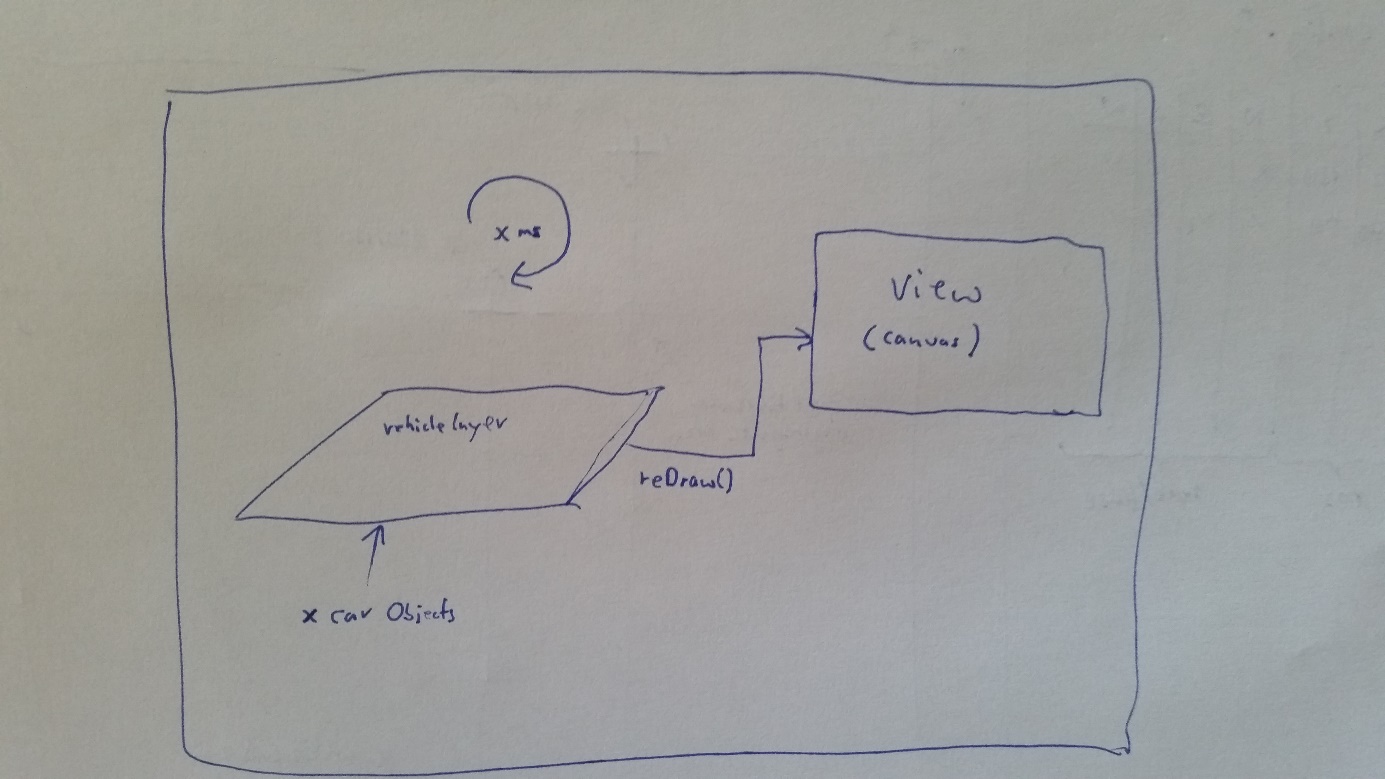
**Design:**

Der Abschnitt Design behandelt die graphische Darstellung der Elemente. Die folgenden Elemente werden abgebildet: PKW, LKW, Ampel, Straße und Straßenschilder. Mit … wird die Beweglichkeit der Elemente realisiert.

Weiters wird hier der Zugriff der Konfigurationsdateien der Verkehrsteilnehmer, der Ampelsteuerung und des Straßennetzes bereitgestellt.

Die Informationen, welche von den DLLs bereitgestellt wird, holt sich das Design aus der Datenhaltung.

Eine Möglichkeit die Bewegung zu realisieren wäre die Verkehrsteilnehmer in einem Layer zu halten und alle Elemente in einem eingestellten Zeitraum zu aktualisieren (vehiclelayer.reDraw()).



**Verkehrsteilnehmer:**

In diesem Teil werden die Objekte der Verkehrsteilnehmer festgelegt. Es wird PKW sowie LKW realisiert. Weiters sollte der Teilnehmer standardmäßig als „gemäßigter Fahrer“ gesetzt werden. Es sollte trotzdem die Möglichkeit bestehen, alle Teilnehmer in Bezug auf Verhalten zu ändern (Auffahrverhalten, aggressiver Fahrstil). Die Fahrtrichtung der Verkehrsteilnehmer soll zufällig stattfinden.

**Ampelsteuerung:**

Die Ampelsteuerung soll automatisch die Straßen und Kreuzungen Regeln. Hierbei detailliert wird es folgendes Verhalten geben:

Rot, Gelb, Gelb blinken, Grün, Grün blinken

Weiters soll die Dauer der Ampelphasen geregelt werden können. Eine logische Verknüpfung zwischen den Ampeln soll realisiert werden.

**Straßennetz:**

In diesem Abschnitt wird festgelegt, welche Elemente die Umgebung ausmacht. In der Verkehrssimulation wird es Kreuzungselemente geben, welche geregelt und ungeregelt sind sowie auch ein Straßenelement. Welches die Kreuzungen miteinander verbindet. Dies soll automatisch geschehen. Steigungen sollten ebenfalls umgesetzt werden.

Elemente:

* Kreuzungselement - Geregelt
* Kreuzungselement - Ungeregelt
* Straßenelement
* Straßenelement - Entrypoint

Eine **Möglichkeit** zum Erstellen des Straßennetzes ist die Implementierung mit einem Koordinatensystem X/Y mit einem [Canvas](https://msdn.microsoft.com/en-us/library/system.windows.controls.canvas(v=vs.110).aspx) panel.

Konfiguration:

Die Konfiguration eines solchen Straßennetzes könnte z.b in einem JSON, oder TSV File beschrieben sein, das beim Initialisieren das Straßennetz aufbaut.

Ein Beispiel hierfür:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| type | x\_Pos | y\_Pos | north | east | south | west |
| GK | 250 | 120 | false | true | true | true |
| GK | 170 | 50 | true | true | true | true |

Bei diesem Beispiel sind zwei Geregelte Kreuzungen definiert. Beim Initialisieren würde dann daraus eine Ungeregelte Kreuzung, sowie 7 Entrypoints für Fahrzeuge resultieren (siehe folgende Abb.).

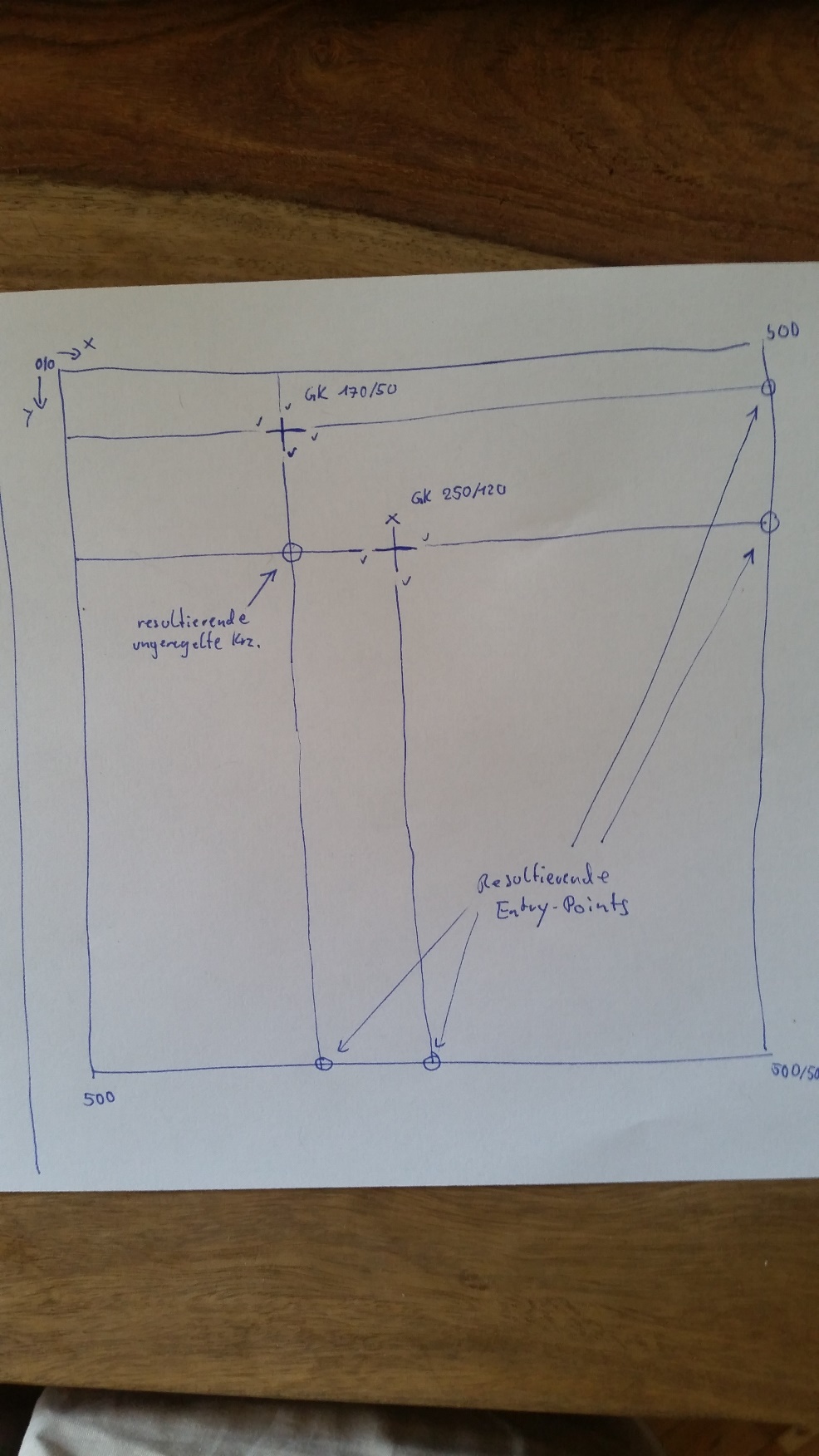


Figure : 500x500 Canvas

**Straßenverkehrsregeln:**

Eine weitere Unterteilung werden die Straßenverkehrsregeln sein. Hierbei wird es die Regeln „Rechts vor Links Regel“ sowie die „Gegenverkehrsregel“ geben. Ebenfalls sollten Straßenschilder wie Stopp, Vorrang geben und die Vorrangstraße für eine geregelte Kreuzung und Straße sorgen.

**Datenhaltung:**

Damit die Informationen zwischen den Abschnitten ausgetauscht werden können, wird eine zentrale Datenhaltung realisiert. Mit Files, welche beschrieben und ausgelesen werden, werden die Informationen zwischen den DLLs ausgetauscht.

Hilfe:

[Traffic-Simulation Github](https://github.com/movsim/traffic-simulation-de)