Prozedurale Generierung des Autos

Vitalij Kagadij

19 Oktober 2015

1 Einleitung

Prozedurale Generierung spielt heutzutage eine große Rolle in der Welt von der Computergrafik. Mit einem Klick ist es möglich ein 3D-Modell zur Verfügung zu kriegen, das man sonst mehrere Stunden, oder sogar Tage, modellieren würde. Ein so erstelltes 3D-Modell kann man, bei Bedarf, mit Parametern anpassen und sofort in einer 3D-Szene verwenden. Ein vernünftig entwickelter Generierungsalgorithmus spart die Zeit und den Aufwand für den Grafik-Künstler, um so eine Szene zu bauen. Besonders bei Computerspielen finden prozedurale Verfahren Anwendung, um den Inhalt schnell und platzsparend zu produzieren. 3D-Modellierung von Autos ist ein weiteres Bereich, wo die prozedurale Generierung den 3D-Künstler helfen könnte, ihre Aufgaben schneller und effizienter zu machen.

2 Stand der Technik

In diesem Kapitel erzähle ich über die wissenschaftlichen Arbeiten, die es schon für diese Thematik gibt, und welche ich für meine Ziele verwenden könnte. Außerdem erläutere ich die Grundlagen der prozeduralen Generierung und Kurvenlehre, welche die zwei wichtigen Themen meiner Ausarbeitung sind.

2.1 Related Work

Prozedurale Verfahren sind sehr verbreitet in der Informatik. Für viele komplexe Probleme gibt es mindestens eine (und meistens auch mehrere) prozedurale Lösungen dafür. Wie ich schon oben erwähnt habe, ist die prozedurale Generierung in Computer Grafik besonders beliebt. Es gibt die verschiedenste Algorithmen für die Generierung von: Vegetation, Landschaften, Texturen, Gesichter, Städte, Häuser, Straßennetze...

Das Ziel meiner Masterarbeit ist es, einen neuen Ansatz zu finden, Autos prozedural zu generieren. Zu diesem Thema gibt es noch kein fertiges Verfahren oder veröffentlichten Papers. Deswegen könnte ich nur die Papers zur meine Hilfe nehmen, in denen es allgemein um prozedurale Generierung ging.

Das Paper von [Kelly u. McCabe] beschreibt die grundlegenden prozeduralen Verfahren und Algorithmen. Mit Hilfe von hier gezeigten Techniken kann man fast alle möglichen Probleme prozedural lösen. Die Papers von [Dettmers] und [Handler] detaillieren einige Verfahren und bieten eigene Ansätze an. Diese Papers geben mir eine Grundlage für meine eigene Realisierung.

Da ich kein Automechaniker bin, brauche ich natürlich bestimmtes Wissen im Bereich “Autos”. Um ein Auto zu erzeugen, brauche ich erstmal zumindest wissen, aus welchen Bestandteilen ein Auto besteht, welche Parametern es besitzen kann, die durchschnittlichen Abmessungen usw. Das Paper von [Schuster, Sattler u. Hoffmann] gibt mir die wichtigen Daten.

2.2 Prozedurale Synthese

Prozedurale Synthese oder prozedurale Generierung bezeichnet in der Informatik die Verfahren, welche in Echtzeit und zur Laufzeit des Computerprogrammes die Programminhalten (wie Texturen, virtuellen Welten, 3D-Objekten und sogar Musik) erzeugen, ohne dass diese Inhalte vorher fest angelegt oder vorprogrammiert sind. Diese Inhalte werden nicht zufällig erzeugt, sondern die Generierung folgt deterministischen Algorithmen, damit die Inhalte immer gleich sind, wenn die Ausgangsbedingungen auch gleich waren.

Die Schlüsseleigenschaft der prozeduralen Generierung ist die Tatsache, dass sie das Objekt (Geometrie, Textur oder Effekt) als eine Sequenz von Anweisungen beschreibt, und nicht als ein statischer Datenblock. Diese Anweisungen können dann aufgerufen werden, um eine Instanz von diesem Objekt zu erstellen, und die Beschreibung kann parametrisiert werden, damit die Instanzen variierte Charakteristiken haben. Ein typisches Beispiel solches Verfahrens ist die Bevölkerung von einem Wald mit prozedural erstellten einzigartigen Bäume [SpeedTree].

Solche prozedurale Verfahren kann man dafür verwenden, um abwechslungsreiche Objekte zu erzeugen. Eine der Grundtechniken, die verwendet werden können, ist die Generierung von 3D-Primitiven mit zufälligen Parametern, zum Beispiel ein Quader mit zufälliger Höhe. Die einfachen Algorithmen, welche die pseudo-zufälligen Funktionen verwenden, können benutzt werden, um "rauschen"zu generieren, welches zum Beispiel in Texturierung gebraucht wird. Die komplexeren rekursiven Algorithmen, wie z. B. Fraktale oder L-Systeme, kann man verwenden, um organische Strukturen aus der Natur wie die Schneeflocken und Bäume nachzubauen. Ebert et al. [David S. Ebert] identifizieren die folgenden wichtigen Eigenschaften von prozeduralen Verfahren:

- **Abstraktion:** die Daten sind nicht im herkömmlichen Sinn angegeben, stattdessen sind die Details in einen Algorithmus oder eine Reihe von Prozeduren abstrahiert. Diese werden von Computer bearbeitet und aufgerufen, falls einen Bedarf entsteht. So kann der Operator mit dem Minimum von Daten das Model sehr einfach manipulieren, ohne dass er viel über die Implementierung weiß.
- **Parametrisierte Steuerung:** Parameter sind definiert und angepasst, was einem spezifischen Verhalten in prozeduralem Generierung genau entspricht. Der Entwickler kann so viele nützliche Steuerelemente definieren, dass der Anwender effektiv arbeiten kann. Beispiel für einen Parameter: Die Höhe der Berge in einem Gelände-Algorithmus oder die Nummer der Segmente in einer prozeduralen Kugel.
- **Flexibilität:** ein Objekt muss nicht unbedingt in Rahmen von realer Welt begrenzt werden. Das heißt, Parameter können so verändert werden, um eine Menge von Ergebnissen zu produzieren, welche unseren Weltvorstellungen oder gewöhnlichen Physikgesetzen widersprechen.

Die präzise Beschreibung für die generierten Objekte ist also möglich und kann durch wenige einfache Parameter ausgedrückt werden. Diese formalere Beschreibung kann verwendet werden, um die große Anzahl von detaillierte Texturen und Geometrien zu kreieren. Dieser Effekt ist als Datenverstärkung (data amplification) bekannt und erlaubt es den Entwicklern die ganze Spielwelt zu erschaffen, die sehr einfach über das Netzwerk verteilt werden kann.

Die Flexibilität und Kontrolle der prozeduralen Generierung geben dem Designer eine Plattform für künstlerische und Experimenten Freiheit. Neue visuelle Effekte und Originalobjekte können durch das Experimentieren mit Parameterwerten, die normalen Grenzen überschreiten, erstellt werden.

2.3 Kurvenlehre

Bézier-Kurven, welche ich in meiner Realisierung verwende, sind die komplexe parametrisierte Kurven, welche den Vorteil haben, dass mit einer sehr geringen Datenmenge komplizierte Kurven beschrieben werden können. Statt jeden einzelnen Punkt der Kurve betrachtet man im Fall der Bézier-Kurven nur die vorgegebenen Bézier-Punkte im Zusammenspiel mit der Berechnungsvorschrift, die sich aus der Benutzung der Bernsteinpolynome ergibt [Holldack].

Eine Bézierkurve n-ten Grades zu gegebenen n + 1 Kontroll- oder Bézier-Punkten b_0, b_1,...,b_n, die das so genannte Kontrollpolygon bilden, ist für λ aus [0, 1] definiert als

$$x(\lambda) = \sum_{i=0}^{n} b_i * B_i^n(\lambda)$$

wobei

$$B_i^n(\lambda) = \binom{n}{i}\lambda^i(1 - \lambda)^{n-1}$$

Das Tupel (b_0,b_1, ..., b_n) heißt Bézier-Polygon, und die Bézier-Kurve kann nur innerhalb von diesem Polygon verlaufen.

Für die Randpunkte x(0) und x(1) einer Bézier-Kurve, die durch die Bézier-Punkte b_0, . . . , b_n gegeben ist, gilt[Mario Holldack]:

x(0) = b_0;
x(1) = b_n;

Die Tangenten an den Randpunkten sind nur durch die benachbarten Bézier-Punkte bestimmt [Holldack].

x‘(0) = n*(b_1 – b_0);
x‘(1) = n*(b_n – b_{n-1});

Mit diesem Satz wissen wir nun zusätzlich, dass die Bézier-Kurve im ersten Punkt b_0 des Polygons beginnt und im letzten Punkt b_n endet. Das bedeutet insbesondere, dass durch die Aneinanderreihung mehrerer Bézier-Polygone stetige und sogar glatte (differenzierbare) Kurven dargestellt werden können. Zur Veranschaulichung sollen diese beiden Bézier-Kurven, die durch die beiden Bézier-Polygone (b_0, b_1, b_2) und (b_2, b_3, b_4) gegeben sind. Für einen differenzierbaren Übergang der beiden Kurvenstücke müssen die beiden Nachbarpunkte und der gemeinsame Bézier-Punkt auf einer Gerade liegen.

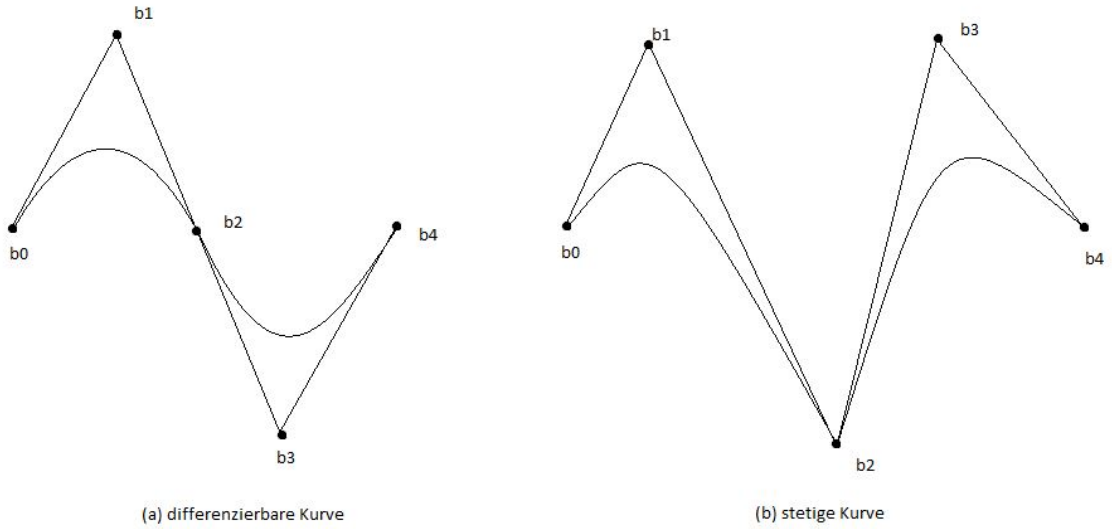

Figure 1: Darstellung komplizierterer Kurven durch Aneinanderreihung der Bézierpolygone

Die Punkte einer Bézier-Kurve können aus gegebenen Bézier-Punkten b_i für i = 0..n schrittweise berechnet werden. Um das geometrisch vorzustellen, muss folgender Algorithmus verwendet werden [Holldack]:

- **Schritt 1** Zeichne das n-eckige Bézier-Polygon (b0, . . . , bn). Dies entspricht den Konvexkombinationen (1 - λ) * b_i + λ * b_i für i = 0..(n - 1).
- **Schritt 2** Teile nun die Seiten des Bézier-Polygons in einem gewählten Verhältnis λ und zeichne nun Strecken zwischen jeweils zwei benachbarten “Mittelpunkten”, sodass ein (n-1)-Eck entsteht. Dies entspricht der Konvexkombination der in Schritt 1 entstandenen Hilfspunkte mit je zwei linearen Polynomen als Koeffizienten.
- **Schritt k=2..n** Wiederhole den vorherigen Schritt mit dem neu entstandenen Polygon.
- **Schritt n+1** Es ist nur noch eine Strecke zwischen zwei Punkten übrig. Teile diese wieder in dem anfangs gewählten Verhältnis λ, um den Punkt x(λ) der Bézier-Kurve zu erhalten. x(λ) ist somit die Konvexkombination der letzten entstandenen Hilfspunkte mit Polynomen vom Grad n - 1.

Für den Fall n=3 und $\lambda = 1/2$ ergeben sich somit die folgenden Zeichnungen (Abbildung 2).

3 Design

Wie eingeführt basiert mein Konzept nicht auf Zufälligkeit, sondern es folgt bestimmten Regeln. Das Ziel hier ist es, einen Prototyp in 2D zu bauen.

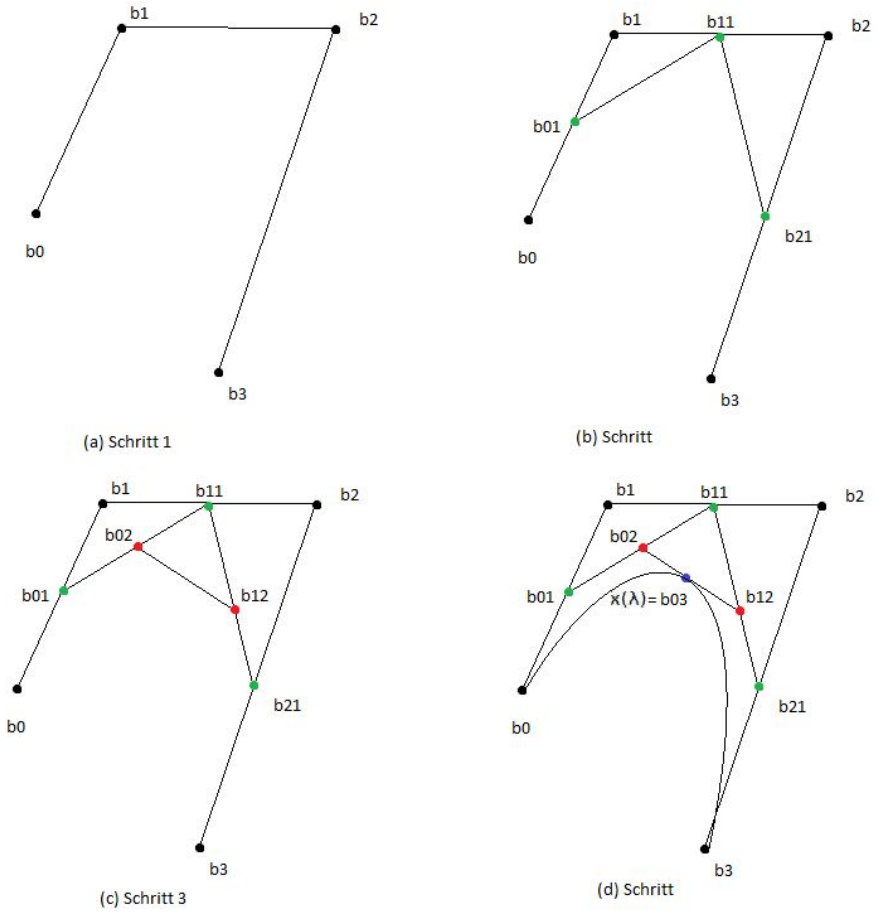

Figure 2: Schritte des De-Casteljau-Algorithmus

3.1 Box styling

Das 2D-Modell eines Autos gehört zu dem "Box styling"-Design. Wenn man das Auto seitlich anguckt, kann man dies in drei Blöcke teilen – Motorraum, Fahrgastraum und Kofferraum. Diese Verteilung nennt man "Box styling", und man unterscheidet drei verschiedene Designs.

One-Box Design (Monospace or Monovolume)
Dieses Design löst alle Unterschiede zwischen separaten auf und umschließt der gesamte Innenraum eines Fahrzeugs in eine einzige Form.

Two-box Design
Dieses Design betont den Motorraum und setzt den Fahrgast- und Kofferraum zusammen.

Three-box Design
Dies ist ein Design, bei dem alle drei Räume(Motor-, Fahrgast- und Kofferraum) auffällig sind und man sie sehr genau unterscheiden kann.

Das three-box Design betrachte ich als Grunddesign für meine Realisierung, da man mit diesen drei Räumen fast alle Autos beschreiben kann (Jedes Auto hat einen Motor, einen Fahrgastraum und einen Kofferraum).

3.2 Bestandteile eines Autos

Die erste wichtige Aufgabe ist es ein hierarchisches Modell eines Autos zu entwickeln. Der erste Schritt wäre es zu bestimmen, aus welchen Bestandteilen ein Auto überhaupt besteht. Natürlich ist Auto ein sehr komplexer Mechanismus, der sehr viele kleinere und größere Teile hat. In meiner Arbeit beschränke ich mich auf das Äußere. Auf der Abbildung kann man die Bestandteile von Karosserie sehen.

Die Anzahl von kleineren Details finde ich hier zu viel für meine Zwecken. So könnte ich versuchen einiges zusammenzufassen. Die zwei Türen z. B. (vorne und hinten) gehören zusammen, da

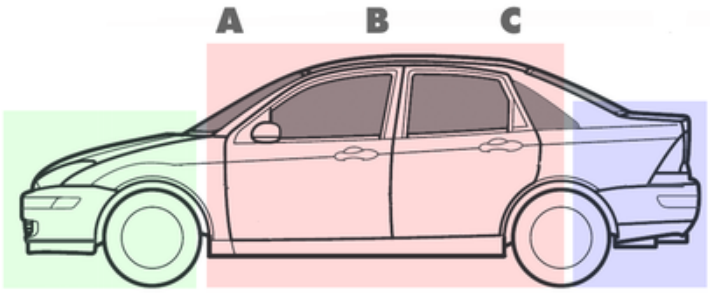

Figure 3: Typische Fahrzeugsäule-Konfiguration von Sedan(three-box). [Wiki]

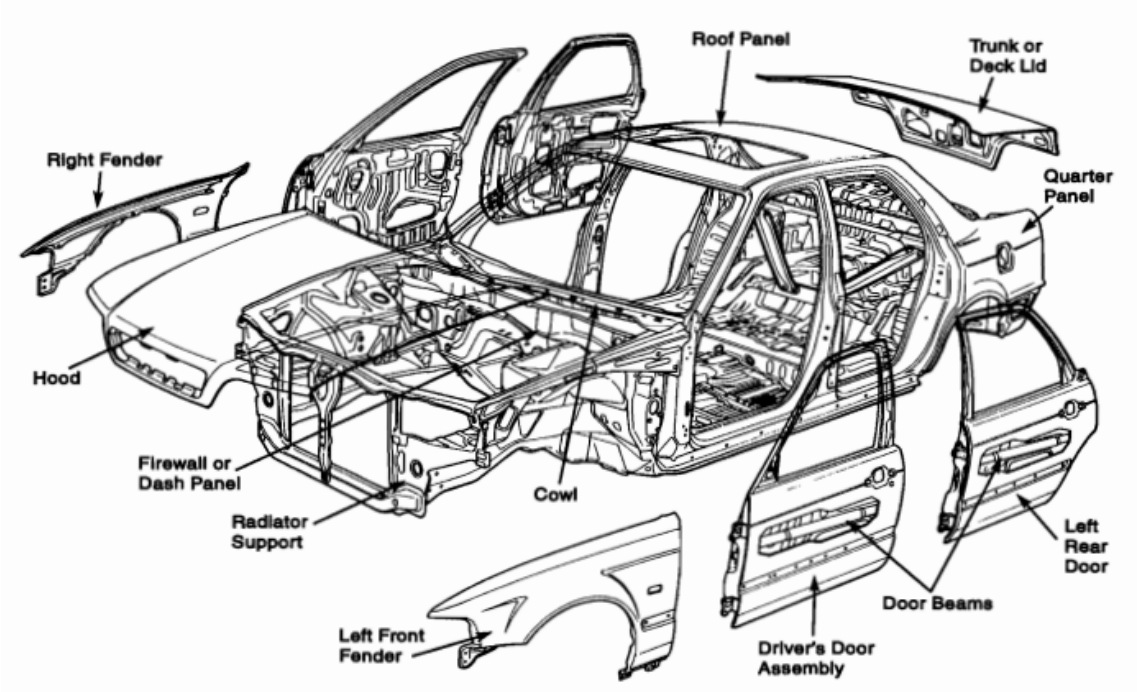

Figure 4: Bestandteile eines Autos.

die Hintertür ist immer von der Vordertür abhängig(kann nicht in einem anderem "Stil"gebaut werden).

Das Dach gehört auch dazu. Es wird desto breiter, je breiter die Türen sind. Die linken und rechten Kotflügel gehören mit dem Haube und dem Stoßstange zu einem "Front"eines Autos zusammen. Genauso besteht der hintere Teil des Autos aus Kofferraum, hinteren Kotflügel und Stoßstange. Schematisch könnte man die Bestandteile eines Autos (nach three-box Modell) wie folgend gruppieren:

Zu dem Frontteil gehören die Kotflügel vorne, die Haube und die Stoßstange. Der mittlere Teil (Fahrgastraum) besteht aus Türen (zwei oder vier), Dach und Windschutzscheibe. Der hintere Teil umschließt die Kotflügel hinten, die Stoßstange, der Kofferraum und die Heckscheibe.

3.3 Konzept

Wie ich schon erwähnt habe, betrachte ich mein Modell des Autos als three-box Design. Es gibt also drei Hauptteile: Motorraum, Gastraum und Kofferraum. Ich füge noch ein Chassis hinzu, das als Basis für das ganze Auto da ist.

So sieht das primitivste Modell eines Autos. Alle Blöcke (Motor, Gast, Koffer und Chassis) sind die Rechtecke. Das kann ich als Skelett eines Autos betrachten. Ein rechteckiger Block dient als ein "Behälter"für den tatsächlichen Umriss eines Auto-Bestandteiles. Die Rechtecke sind deswegen gut, weil ich mit deren Hilfe sehr einfach die wichtigsten Daten speichern kann: die Höhe und die Länge eines Blockes, die Position im Raum (Zentrum eines Blocks), Verhältnis zu anderen Blöcke.

Wie schon gesagt ist, betrachte (darstelle) ich ein Auto als Modell, das aus vier Blöcken besteht

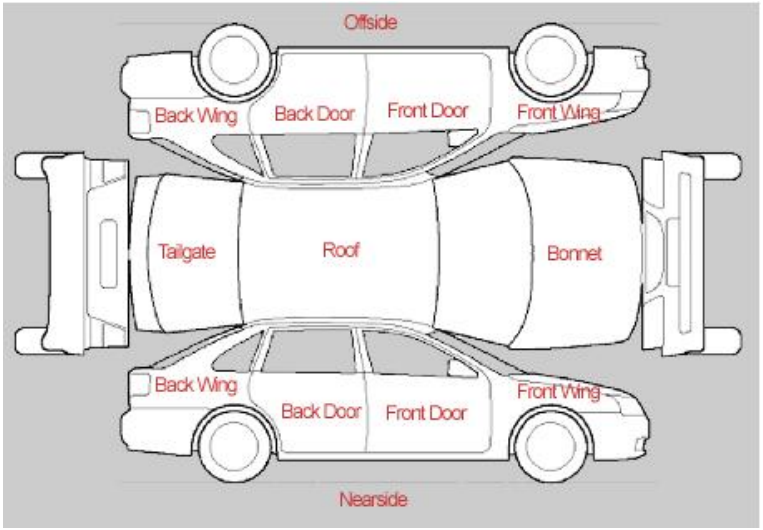

Figure 5: Bestandteile eines Autos(gruppiert). [Autoteile]

– Chassis, Motorraum, Fahrgastraum und Kofferraum. Diese Blöcke werden später als "Container"für die tatsächliche 3D-Modells (Meshes) der Auto-Bestandteile benutzt werden. Das Zentrum von Chassis-Block wird immer im Nullpunkt gerendert, damit das Finden von anderen Koordinaten erleichtert wird.

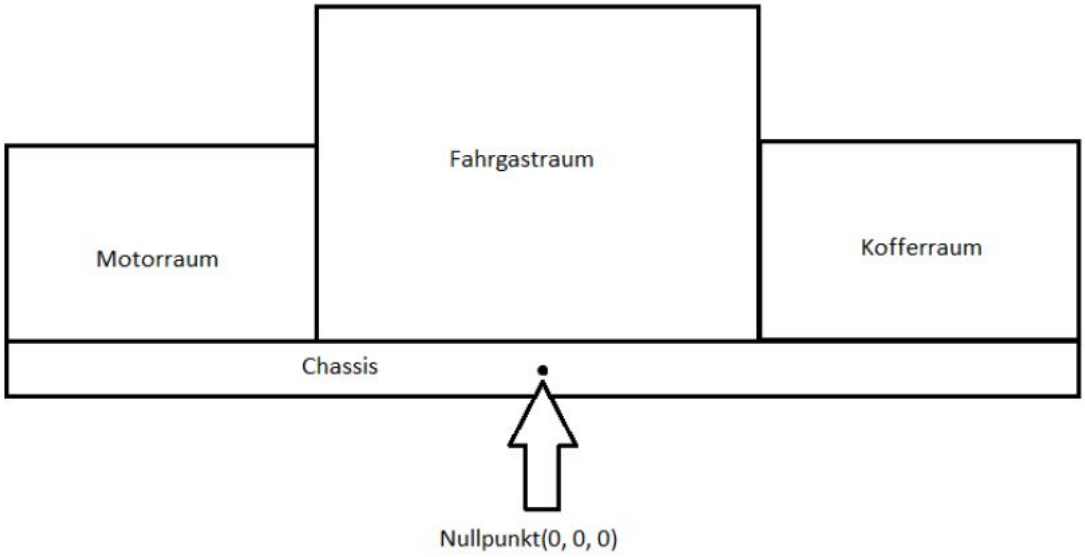

Figure 6: Block-Modell des Autos

Allgemein, um das Modell einfacher manipulieren zu können, brauche ich ein globales Koordinatensystem (in 2D-Darstellung). Die Linien, wo sich die verschiedene Blöcke treffen/schneiden sind meine Achsen. Dann kann ich die wichtigsten Punkte so definieren:

- Wo sich das Chassis mit dem Motorraum treffen
- Wo sich das Chassis, der Motorraum und der Fahrgastraum treffen
- Wo sich das Chassis, der Fahrgastraum und der Kofferraum treffen
- Wo sich das Chassis mit dem Kofferraum treffen

Zusätzlich, kann ich die Treffpunkte des Fahrgastraums mit dem Motorraum und dem Kofferraum nehmen. Diese helfen mir später die Höhe der entsprechenden Blöcke zu bestimmen.

Außerdem, weiß ich so, wo sich die Linie, der ein Block repräsentiert endet und die Linie, die den Nachbarn-Block repräsentiert anfängt. Also, den Übergangspunkt wird bekannt. Die Punkte A, B und C (Abbildung), zum Beispiel, werden dafür benutzt, um eine Kurve zu berechnen, die den Motorraum darstellt (Abbildung 7).

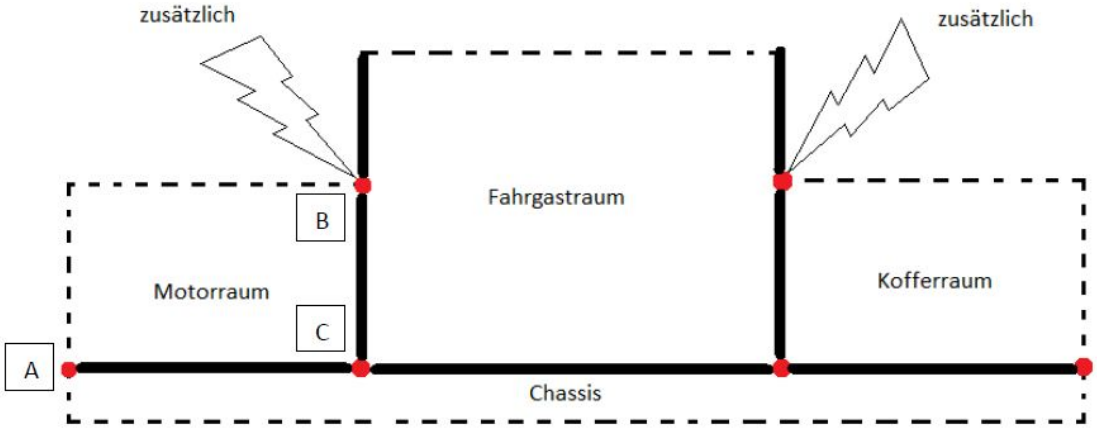

Figure 7: Lokales Koordinatensystem

Die Abbildung 8 zeigt, wie eine Kurve für die Darstellung des Motorraums benutzt werden kann. Die größeren Punkte des globalen Koordinatensystems sind die Grenzpunkte der Kurve, und die kleineren Punkte, die auf der Kurve liegen, sind die Kontrollpunkte, die für das Konstruieren der Kurve gebraucht werden. Genauso kann man die andere Komponenten des Autos erzeugen. Die Kurven, die den Grundriss des Autos darstellen, werden später als Basis für die Erstellung des Dreiecksnetzes benutzt.

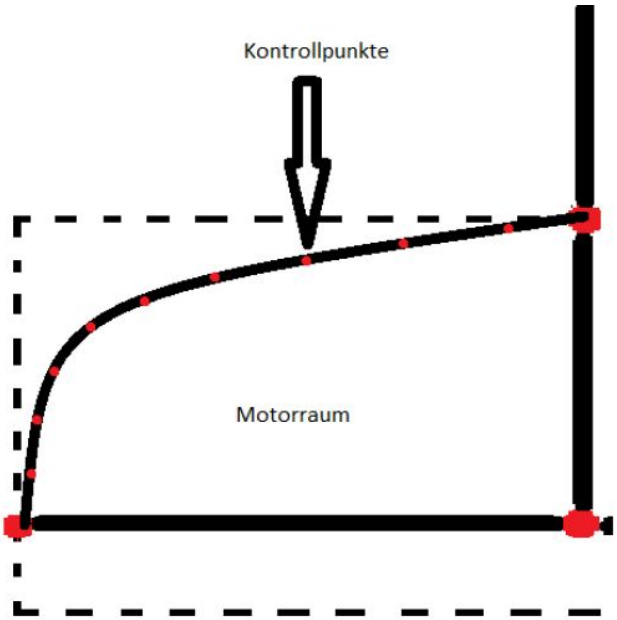

Figure 8: Kurvendarstellung

4 Umsetzung

4.1 GUI

Bedienoberfläche ist ein sehr wichtiges Element meines Prototyps, weil die Parametrisierung des Autos genau an dieser Stelle stattfindet. Allerdings dient die GUI zurzeit nur als Hilfsmechanismus. Später sollte das Auto nicht "manuell"erzeugt werden kann, sondern automatisch. Es ist hier eine Menge von Sliders zu finden, die für die Parameter des Autos zuständig sind. Man kann, zum Beispiel, die Länge und die Höhe von dem Auto ändern; oder die Höhe und die Länge von einzelnen Blöcken; oder sogar die Form von Motorhaube und den Umriss von Heckraum. Ändert man die Position von Sliders, wird das Auto in Echtzeit neugeneriert.

Die Parameter, die zurzeit relevant sind, kann man in folgende Gruppen teilen:

- **Abmessungen des Autos:** das sind die Höhe, die Länge und die Breite des Autos.
- **Art des Autos:** das sind die Parameter, die das Karosserieart des Autos bestimmen. Dazu zählt man die Höhe von dem Haube und die Höhe von dem Kofferraum, sowohl auch die Länge des einzelnen Blöcken relativ zueinander.
- **Umriss des Autos:** diese Parameter beschreiben die Kurven, die das Umriss des Autos bestimmen.

4.2 Klassenmodell

Der Prototyp wird mit Java Programmiert. Als Entwicklungsplattform werden Eclipse und "ComputerGraphics"-Framework von Prof. Dr. Philipp Jenke benutzt.

Da die vier Blöcke, die mein Auto-Modell beschreiben, für das Konzept sehr wichtig sind, hat jeder Block seine eigene Java-Klasse: *Chassis2D*, *Front2D*, *Gats2D* und *Heck2D*. Außerdem gibt es eine Klasse *Auto2D*, die ein komplettes Auto-Modell beschreibt und die verschiedenen Blöcke verwaltet.

Jede Block-Klasse hat mehrere Variablen (Felder), die das Manipulieren und Erstellen von Instanzen vereinfachen. Da jeder Block im Prinzip ein Rechteck ist ("Behälter"), habe ich den mit vier Punkten (*IVector3*) definiert. Die Punkte sind bei jedem Block gleich verteilt: Punkt A (links unten), Punkt B (links oben), Punkt C (rechts oben), Punkt D (rechts unten). Demnächst habe ich die Punkten mit entsprechenden Kanten verbunden (*Linie3D*): AB, BC, CD und DA. Außerdem hat jeder Block zwei Variablen, die seine Höhe und seine Länge speichern. Mit Hilfe von diesen Daten ist es uns viel einfacher den Block zu manipulieren und rendern.

Die Klasse *Auto2D* nimmt eine Menge von Parametern an und generiert auf Basis von denen vier Bestandsblöcke, die ein Auto repräsentieren. Die Klasse hat die Methoden *generateChassis()*, *generateFront()*, *generateGast()* und *generateHeck()*, welche aus Parametern die konkreten Zahlen berechnen und dementsprechend die Böcke generieren.

4.3 Kurven

Wie früher schon eingeführt ist, verwende ich für meine Realisierung die Bézier-Kurven.

Die rechteckigen Blöcke sind nur die "Behälter"für die tatsächliche Realisierung von Umrissen eines Autos. Diese werden mit Hilfe von Kurven erstellt. Ich benutze die Bézier-Kurve in meiner Realisierung. Die Bézier-Kurve hat einen Grad und eine Reihe von Kontrollpunkte (Grad+1). Die Einfachste Bézier-Kurve erstes Grades hat nur zwei Kontrollpunkte: Start- und Endpunkt. Das ist eigentlich keine Kurve, sondern eine Gerade. Wenn man mehrere Kontrollpunkte hinzugefügt, kann man den Verlauf der Kurve kontrollieren.

Jede Block-Klasse besitzt vier Bézier-Kurven (*left*, *top*, *right* und *bottom*), die man zur Darstellung seines Umrisses benutzt werden können. Ich verwende die Kurven zweites Grades. Das bedeutet, dass die zwei Kontrollpunkte definieren den Start und das Ende der Kurve, und die Krümmungswert von noch einem Kontrollpunkt steuert, wie diese Kurve verläuft. Alle Kontrollpunkte sind an bestimmte Parameter gebunden, sodass man mit Hilfe von GUI ihre Position ändern kann. Damit wird der Umriss des ganzen Autos geändert.

Zum Beispiel für die Windschutzscheibe ist die Bézier-Kurve *left* des Gast-Blockes zuständig. Ihr Startpunkt ist der Treffpunkt von dem Front-Block und Gast-Block und der Endpunkt ist der Treffpunkt der Kurve *top* von Gast-Block (Abbildung). Der dritte Kontrollpunkt, der die Krümmung bestimmt, ist standardmäßig der Punkt B dieses Blockes, ist aber mit zuständigem Slider änderbar.

5 Evaluation

In diesem Kapitel möchte ich die Ergebnisse meines Programms darstellen. Die Abbildungen 9, 10 und 11 zeigen die Beispiele von Karosseriearten, welche man in 2D manuell erstellen kann.

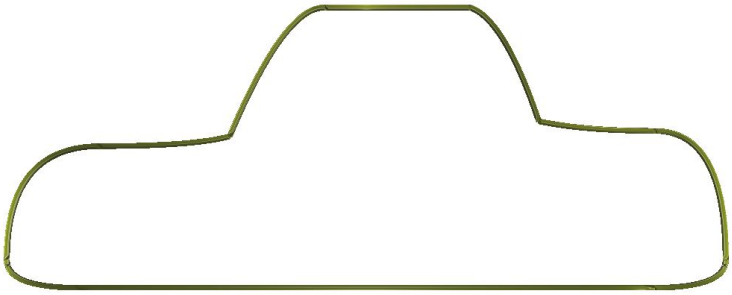

Figure 9: Ergebnis 1 - Karosserietyp: Sedan

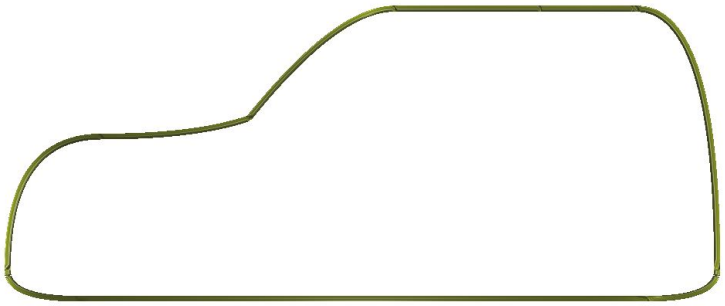

Figure 10: Ergebnis 2 - Karosserietyp: Kombi

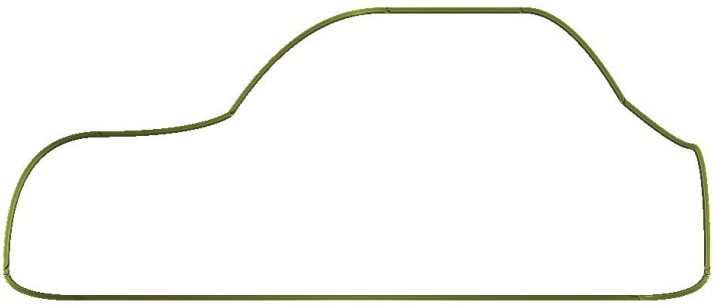

Figure 11: Ergebnis 3 - Karosserietyp: Coupe

Diese Ergebnisse zeigen uns, dass die Hauptaufgabe des Projekts erfüllt ist - man kann die 2D-Umrisse eines Autos generieren und grafisch darstellen.

Die "Generierung"passiert zur Zeit, leider, erst nur manuell was dem Begriff "prozedurale Synthese"nicht entspricht. Auch die "glatten Konturen"sind noch nicht variabel genug, das heißt sie sehen ziemlich gleich aus. Es bedeutet, dass die Kurvendarstellung (insbesondere die Kontrollpunkten) noch verbessert werden muss.

6 Fazit und Schluss

Die Aufgabe des Projektes war es ein Konzept vorzustellen, wie ein formales Modell eines Autos aussehen könnte. Dafür habe ich ein 2D-Prototyp entworfen, der ein abstraktes Auto repräsentiert. Ein Konzept wurde von mir entwickelt, welches das Auto als ein Gegenstand darstellt, der aus vier Blöcke besteht. Die Kernpunkte dieser Darstellung waren:

- globales Koordinatensystem, welches die Relation von Blöcken zueinander beschreibt
- Realisierung von “glatten Konturen”
- umfangreiche Parametrisierung

Diese drei Aufgaben habe ich in meinem Projekt realisiert, und das Ergebnis – ein formales Modell eines Autos in 2D Darstellung – kann man als Basis für die weitere Entwicklung verwenden.

Der nächste Schritt für mich wäre, an “Automatismus”weiter zu arbeiten, sodass die Autos nicht mehr manuell erzeugt werden sollen, sondern tatsächlich prozedural generiert. Dafür muss ich insbesondere die Parametrisierung erweitern, das heißt die zusätzliche Parameter einfügen und die existierende erweitern (zum Beispiel mehr Kontrollpunkten für die Kurven). Das Ganze führt dazu, dass die resultierende Autos vielfältig sind. Eine weiter Aufgabe wäre es, mein 2D Modell in die dritte Dimension zu bringen. Es ist den bestehenden Prototyp zu erweitern, sodass mein Modell des Autos in 3D generiert wird. Dabei ist zu überlegen, wie ich die Kurven in kreisförmige Flächen übersetzen kann. Wichtig ist, dass alle für das 2D Modell relevante Parameter beibehalten sind. Es ist auch die neuen Parameter zu überlegen, welche die Breite des Autos beschreiben. Im Endeffekt soll ich das 3D Modell des Autos haben, die genauso wie das 2D Modell generiert werden kann.

References

[Kelly u. McCabe] George Kelly; Hugh McCabe: A Survey of Procedural Techniques for City Generation. ITB Journal, Issue 14, December 2006

[Schuster, Sattler u. Hoffmann] Andreas Schuster; Josef Sattler; Stephan Hoffmann: Bestimmen der aktuellen Abmessungen differenzierter Personen-Bemessungsfahrzeuge. Westsächsische Hochschule Zwickau, Institut für Verkehrssystemtechnik i. G., April 2011

[David S. Ebert] David S. Ebert; F Kenton Musgrave; Darwyn Peachy; Ken Perlin; Steven Worley; Texturing and Modelling - A Procedural Approach. Morgan Kaufmann 2003.

[Lindenmayer] A. Lindenmayer; Mathematical models for cellular interaction in development, Parts I and II. 1968.

[Holldack] Mario Holldack: Bézierkurven. Ausarbeitung im Proseminar Bernsteinpolynome und ihre Anwendungen, SoSe 2011

[Dettmers] Sven Dettmers: Prozedurale Generierung von Siedlungen in einer mutable cube world. Fakultät Technik und Informatik, Studiendepartment Informatik. 12. Juni 2014

[Handler] Bernhard Handler: Prozedurale Levelgenerierung für 2D Plattformspiele. Fachhochschul-Masterstudiengang Interactive Media ,Hagenberg, Oktober 2012

[Schwarz u. Köchler] Hans Rudolf Schwarz; Norbert Köchler: Numerische Mathematik. 7. Auflage. In: Vieweg+Teubner; GWV Fachverlage GmbH, Wiesbaden 2009

[Wiki] $https://en.wikipedia.org/wiki/Three-box_styling$

[SpeedTree] Interactive Data Visualization Inc. SpeedTree RT. $http://www.speedtree.com$2006

[Autoteile] $http://aceautomotive.biz/car/car-diagram-exterior-parts-names.html$