Prozedurale Generierung von Autos - Principal Component Analysis

Vitalij Kagadij

26 Dezember 2015

1 Einleitung

Das Ziel meiner Masterarbeit ist es, einen neuen Ansatz für die prozedurale Modellierung der Autos zu erarbeiten. Prozedurale Synthese ist ein mächtiger Mechanismus in Computer Grafik, welcher die investierte Zeit und den Aufwand bei der Modellierung eines 3D-Objektes mäßig verringert. Anstatt alles manuell zu kreieren, braucht man nur ein Paar vom wichtigen Parameter, welche man manipulieren kann, um eine Menge von verschiedenen Objekten zu erzeugen. Das ist etwa für die 3D-Künstler relevant, die ihre Aufgaben effizienter machen können. 3D-Modellierung von Autos ist ein weiteres Bereich, wo man prozedurale Synthese anwenden kann.

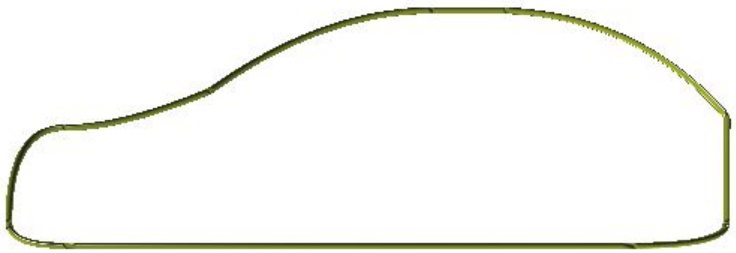

Figure 1: Grafische Darstellung eines Autos

Bis jetzt habe ich schon ein Konzept vorgestellt (Aufgabe von Projekt 1), wie ein formales Modell des Autos in meiner Masterarbeit aussehen soll. Dafür wurde ein 2D-Prototyp eines abstrakten Autos von mir entworfen, welcher das Auto aus vier Blöcken "zusammenbastelt", als ein Array von Bezier-Kurven speichert und grafisch darstellt. Das "Erzeugen"eines Autos hat allerdings manuell passiert.

In diesem Projekt stelle ich eine Technik für die prozedurale Modellierung von Autos in zwei Dimensionen. Ein Auto-Modell (grafisch im 2D dargestellt als eine Kontur) lässt man auf Basis von vorher analysierten typischen Auto-Beispielen automatisch generieren. Gewünscht ist, dass dieser Ansatz es erlaubt, eine riesige Menge von verschiedenen Autos sehr schnell und mit wenig Aufwand erzeugen.

Zuerst wird ein Set von typischen Autos als zweidimensionales Modell dargestellt, als Vektor repräsentiert und analysiert. Die neuen Autos kann man modellieren, indem man lineare Kombinationen aus diesen Prototypen bildet. Die Idee ist relativ einfach: Angenommen, wir haben zwei typischen Autos, der Einfachheit halber bestehen die aus nur drei Punkten – A, B und C. Die Punkten A und B von beiden Autos sind identisch, der Punkt C hat jeweils verschiedenen Koordinaten. Um ein neues Auto zu kriegen, bilden wir die Summe von diesen zwei Autos, multipliziert mit einem Skalar-Koeffizient. Das Ergebnis ist ein neues Auto mit den gleichen A und B, und einem neuen C (siehe Abbildung 2).

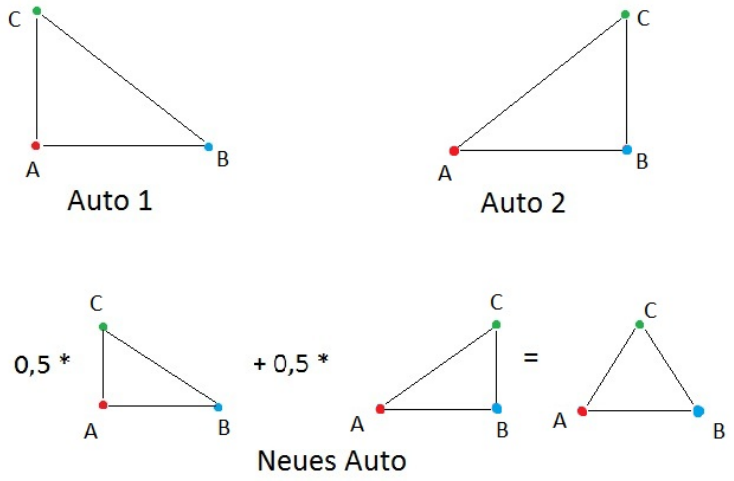

Figure 2: Das Erzeugen neues Autos durch lineare Kombination

1.1 Aufbau der Arbeit

Nach dieser Einführung werden im zweiten Kapitel zunächst verwandte Arbeiten vorgestellt. Im dritten Kapitel beschreibt kurz ein formales Modell des Autos vorgestellt, welches in meiner Masterarbeit verwendet wird. Im viertem Kapitel wird auf den Datenbank der zur Analyse benutzten Fahrzeuge eingegangen. Das fünfte Kapitel erläutert das Konzept meines Verfahrens, insbesondere wird hier auch gezeigt, wie man mit Hilfe von Principal Component Analysis (PCA) die wichtigsten Merkmale

eines Modells extrahieren kann, um die Dimensionierung des Problems zu reduzieren. Ergebnisse finden sich im sechsten, eine abschließende Zusammenfassung und ein Ausblick auf zukünftige Arbeit in siebtem Kapitel.

2 Related Work

PCA, als mathematische Verfahren, ist schon seit längere Zeit bekannt. Die verschiedenste Papers beschreiben sowohl das Prinzip von PCA selbst ([Jolliffe]) als auch die einzelne Probleme, welche mit PCA gelöst werden können ([Aguilera], [Aitchison], [Ali, Clarke u. Trustrum]). In diesen Papers werden die Grundlagen von Hauptkomponentenanalyse vorgestellt, was kritisch für das Verständnis ist, wie und für welche Zwecke man PCA überhaupt verwenden kann.

Bei meiner Arbeit habe ich den Rücksicht auf das Paper von [Blanz u. Vetter] genommen, welches sich mit Synthese von 3D-Gesichter beschäftigt. Ich verwende das ähnliche Verfahren. In ihrem Paper wird es ein Prinzip erläutert, wie man eine Reihe von typischen Elementen (in diesem Fall – Gesichter) kombinieren kann, um ein neues Element zu generieren.

Das Gesichtsmodell von [Blanz u. Vetter] basiert sich auf Daten Set von 3D-Gesichter (die Laser scnans von 200 menschlichen Köpfen, repräsentiert als 70000 Eckpunkten und genau so viel Farbwerten). Die Geometrie von einem Gesicht wird durch einen Form-Vektor S = $(X_1, Y_1, Z_1, X_2, \ldots, Y_n, Z_n)^T$ aus R^{3n} repräsentiert, welcher die X, Y, Z-Koordinaten von seinen n Eckpunkten enthält. Genauso werden die Farbwerten dargestellt (die Farbwerte sind für meine Arbeit nicht relevant, deswegen werde ich die weiter nicht erwähnen). Ein morphable Gesichtsmodell hat man aus einem Set von m Beispielgesichter konstruiert, repräsentiert durch Formvektor S_i. Die neuen Formen S_model kann man in baryzentrischen Koordinaten als eine lineare Kombination der Formen von m Beispielgesichter ausdrücken:

$$S_{mod} = \sum_{i=1}^{m} a_i S_i,$$

wobei

$$\sum_{i=1}^{m} a_i = 1;$$

Ein Morphable Modell ist als Set von Gesichter ($S_{mod}(\vec{a})$) definiert, welche mit durch Koeffizienten $\vec{a} = (a_1, a_2, \ldots, a_m)^T$ parametrisiert sind. Die neuen Gesichter kann man dementsprechend durch Variieren von Parameter a generieren. Denselben Verfahren verwende ich für die Erstellung der neuen Autos bei meinem Ansatz. Ein Set von Beispielautos wird analysiert, um ein Durchschnittsauto und die sinnvolle Parameterwerte für $\vec{a}$ zu finden.

3 Modell des Autos

Es werden hier die Fahrzeuge betrachtet, die durch Bezier-Kurven gegeben sind. Die Kurven werden jeweils durch $n + 1$ Kontrollpunkte repräsentiert (hier meist drei pro Bezierkurve). Der Grad der Kurve ist also n. Jede Bezierkurve hat die Form:

$$p(t) = \sum_{i=0}^{n} p_i B_i^n(t).$$

Es werden k Bezier-Kurven zusammengefügt (in unserer Realisierung besteht jedes Auto aus k = 10 Kurven). Damit ergibt sich ein Vektor x von Kontrollpunkten. Der Vektor beinhaltet jeweils 3D-Vektoren (die Kontrollpunkte) als Koordinaten. Der Vektor hat die Dimension $m = k \cdot (n + 1)$. Ein solcher Vektor definiert eindeutig ein Fahrzeug.

Für die Realisierung meines Ansatzes und für die grafische Darstellung des Fahrzeugmodells verwende ich die Computer Graphics Lab (CGL), ein von Professor Philipp Jenke entwickeltes Projekt, welches die Entwicklung von Graik-basierten Anwendungen erleichtert. CGL beinhaltet eine Menge von Klassen, welche die wichtigsten geometrischen Datenstrukturen und mathematische Operationen, zusammenfasst. Die benutzte Software erlaubt es so einen Vektor grafisch als zweidimensionales Automodell darstellen. Es ist auch möglich ein 2D-Modell als Vektor zu speichern. So einen R^{28} Vektor enthält die Kontrollpunkte aller Bezier-Kurven, welche das Auto bestimmen. Die Kontrollpunkte sind in einer bestimmte Reihenfolge sortiert. Kennt man die Reihenfolge, kann man die Kurven wiederherstellen, um ein Auto anzuzeigen.

4 Fahrzeug-Datenbank

Um es zu analysieren, wie das prozedural synthetisierte Auto tatsächlich aussehen soll, habe ich ein Set von typischen Autos gesammelt [www.carlook.net]. Das sind die 30 Fahrzeuge verschiedener Größe und Karosserietyp. Alle Beispielautos haben den gleichen Maßstab.

Alle 30 Autos aus dem Datenbank werden manuell mit Hilfe von einem Editor als zweidimensionale Modelle nachgebildet. Als nächstes wird jedes Auto als R^{28} Vektor mit Kontrollpunkten gespeichert. So entstehen 30 solchen Vektoren und das sind zu analysierende Daten.

5 Analyse und Synthese

Es werden nun Wege gesucht, neue Fahrzeuge zu synthetisieren, also neue (sinnvolle) Vektoren von Kontrollpunkten zu finden. Der Einfachheit halber trennen wir dazu die Vektoren für die drei Achsen x, y und z auf. Es ergeben sich also drei Vektoren v^x, v^y und v^z jeweils der Dimension m (in unserem Fall die Dimension von jedem Vektor ist m = 28).

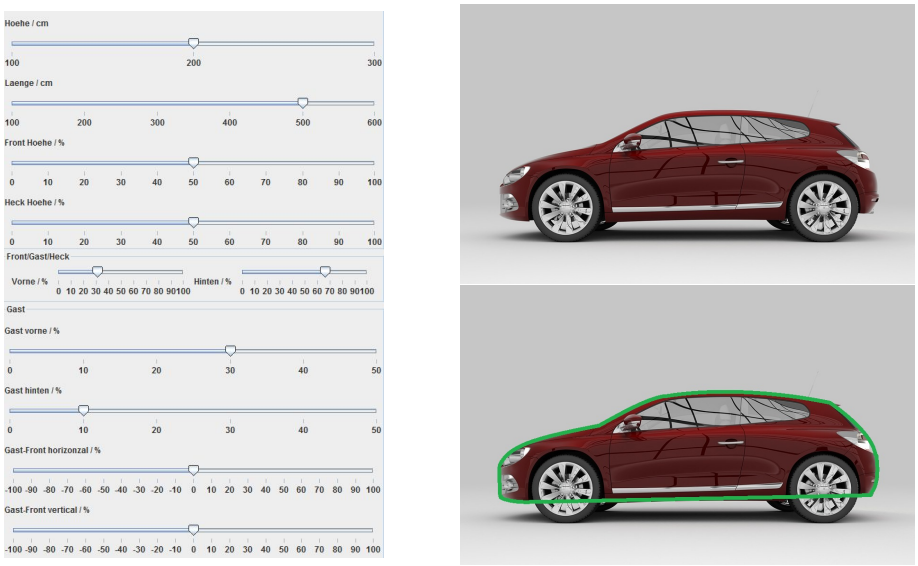

Figure 3: Mit dem Editor manuell nachgebildetes Fahrzeugmodell [www.pixabay.com]

Im Folgenden wird der Bezeichner v als Repräsentant für einen beliebigen der drei Teilvektoren v^x, v^y und v^z verwendet. Dabei handelt es sich also um einen m-dimensionalen Vektor mit Skalaren als Komponenten.

5.1 PCA

Ein Mittel, wichtige Hauptrichtungen aus einer Menge von Vektoren zu extrahieren, ist die Principal Component Analyse (PCA). Um damit zu arbeiten, verwende ich die selbst erzeugten Fahrzeuge aus dem Fahrzeug-Datenbank. Es steht also eine Liste $V = v_i$ von Vektoren zur Verfügung, die jeweils ein Fahrzeug repräsentieren (genau genommen repräsentieren sie jeweils eine Komponente der Kontrollpunkte der verwendeten Bezier-Kurven).

Zunächst wird für die Vektoren den Schwerpunkt berechnet:

$$\bar{v} = \frac{1}{|V|} \sum_{i=1}^{m} v_i.$$

Mit Hilfe des Schwerpunktes wird die Kovarianzmatrix bestimmt als

$$C = (v_i - \bar{v}) \cdot (v_i - \bar{v})^t.$$

Bei C handelt es sich also um eine $m \times m$-Matrix. Für diese Matrix sind weiterhin die Eigenvektoren e_i und die zugehörigen Eigenwerte λ_i zu berechnen. Die Eigenvektoren e_i haben Dimension m, die Eigenwerte sind Skalare.

Von hier an nehmen wir an, dass die Eigenwerte und damit die zugehörigen Eigenvektoren der Größe nach absteigend sortiert sind. Der größte Eigenwert und damit sein zugehöriger Eigenvektor werden als λ_0 und e_0 bezeichnet. Diese repräsentieren die wichtigste Komponente im Eigenraum zur Generierung von Fahrzeugen.

5.2 Transformation

Nun ergeben sich zwei Räume. Der Raum, in dem die Vektoren v zur Repräsentation der Fahrzeuge existieren, ist der Euklidische Raum. Durch die Eigenvektoren ergibt sich ein zweiter Raum, der Eigenraum. Die Eigenvektoren können verwendet werden, um einen Vektor vom Euklidischen Raum in den Eigenraum zu transformieren. Dazu werden die Eigenvektoren jeweils als Spalten einer Matrix B verwendet. Bei B handelt es sich also um eine $m \times m$-Matrix. Um einen Vektor v^{Euklid} in den Eigenraum zu transformieren (also v^{Eigen} zu berechnen) ergibt sich:

$$v^{Eigen} = B^t(v^{Euklid} - \bar{v}).$$

Analog ergibt sich die Rücktransformation vom Eigenraum in den Euklidischen Raum als

$$v^{Euklid} = B \cdot v^{Eigen} + \bar{v}. \tag{1}$$

Was ist damit erreicht? Man kann die Eigenkoordinaten a_i eines Fahrzeugs berechnen, indem man den Euklidischen Fahrzeugvektor in den Eigenraum transformiert (für je ein Fahrzeug ist also $a_i = v^{Euklid}$). Diese Eigenkoordinaten lassen sich als Gewichtungen der entsprechenden Eigenvektoren interpretieren, um das Fahrzeug im Eigenraum zu repräsentieren. Insbesondere ist es möglich alle manuell erstellten Fahrzeuge in den Eigenraum zu transformieren und beobachten, welche Eigenkoordinaten typisch sind. Hat man l Fahrzeuge, so erhält man beispielsweise l Werte für a_0 und damit eine Verteilung der Gewichtungen für den ersten Eigenwert. Beispielsweise kann man den Mittelwert der verschiedenen a_0 verwenden, um den *typischen* Gewichtungsfaktor für den ersten Eigenwert zu bestimmen (gegeben unsere Testfahrzeuge). Macht man das für alle Eigenkoordinaten a_i so kann man im Eigenraum das *Durchschnittsfahrzeug* generieren und dann wieder in den Euklidischen Raum zurücktransformieren (z.B. um es anzuzeigen). Für eine gegebene Belegung der Eigenkoordinaten a_i ergibt sich das Fahrzeug im Eigenraum natürlich wieder als

$$v^{Eigen} = \sum_{i=1}^{m} a_i e_i \tag{2}$$

Außerdem lässt sich ein Fahrzeug, das im Eigenraum repräsentiert ist, zurück in den Euklidischen Raum transformieren.

5.3 Dimensionsreduzierung

Mit Hilfe der Gewichtung durch die Eigenwerte lässt sich eine Dimensionsreduktion erreichen. So kann man die Repräsentation eines Fahrzeugs im Eigenraum auf $o < m$

reduzieren. Dazu verwendet man natürlich die o (in unserem Fall $o = 10$) größten Eigenwerte und deren Eigenvektoren. Die Matrix B wird damit zur Matrix B^o, also einer $m \times o$-Matrix. Es werden zur Erstellung von B^o nur die ersten o Eigenvektoren verwendet und als Spalten in die Matrix geschrieben. Entsprechend ergeben sich nach der Transformation in den Eigenraum auch nur noch o Eigenkoordinaten: $v^{Eigen} \in R^o$. Bei der Rücktransformation in den Euklidischen Raum kommen die fehlenden Koordinaten aber wieder dazu und es ergeben sich wieder m-dimensionale Vektoren v^{Euklid}.

Will man nun Fahrzeuge bei reduzierter Dimension im Eigenraum erzeugen, dann verwendet man entsprechend auch nur die Eigenkoordinaten der verwendeten Eigenvektoren. Gleichung 2 reduziert sich also zu:

$$v^{Eigen} = \sum_{i=1}^{o} a_i e_i$$

5.4 Synthese neuer Fahrzeuge

Ein neues Fahrzeug wird also synthetisiert, indem man im Eigenraum geeignete Eigenkoeffizienten a_i findet, die dann zusammen einen Vektor a bilden. Dieser Vektor hat im Normalfall die Dimension m. Verwendet man aber die Dimensionsreduktion aus Abschnitt 5.3, hat er zunächst nur die Dimension $o < m$. Die fehlenden $m-o$ Koordinaten werden dann einfach mit Nullen aufgefüllt. Den zugehörigen Fahrzeugvektor berechnet man wieder nach Gleichung 1.

6 Evaluation

In diesem Kapitel möchte ich die Ergebnisse meiner Analyse darstellen. Zuerst habe ich für die 30 Vektoren (welche die Beispielautos repräsentieren) die PCA angewendet. Danach habe ich jedes einzelnen Testauto zuerst in Eigenraum umgewandelt (ohne Dimensionsreduzierung). Nach der Rücktransformation in den Euklidischen Raum war das resultierende Auto dem Testauto identisch.

Als nächstes habe ich dieselbe Prozedur wiederholt, diesmal aber mit Dimensionsreduzierung. Ich habe 10 Hauptkomponenten genommen (aus insgesamt 28). Die Abbildung 4 zeigt einige Ergebnisse der Transformation des Autos in den Eigenraum und zurück. Es ist immer die Kontur des Testautos zusammen mit der Kontur des hin- und zurücktransformierten Autos angezeigt.

Man sieht hier, dass die Ergebnisse nicht perfekt sind: Die Konturen stimmen nicht komplett überein, es gibt "Umbrüche"und nicht "glatte Übergänge". Man muss aber bedenken, dass mehr als die Hälfte der Informationen bei Dimensionsreduzierung verworfen wurde. Das heißt nämlich, dass bei der Rekonstruktion des Autos wurde nicht 28, sondern nur 10 Parameter verwendet.

Nichtsdestotrotz kann man in dem Ergebnis definitiv ein Auto erkennen. Manche Problemstellen lassen sich korrigieren, zum Beispiel die "Umbrüche", indem wir wissen,

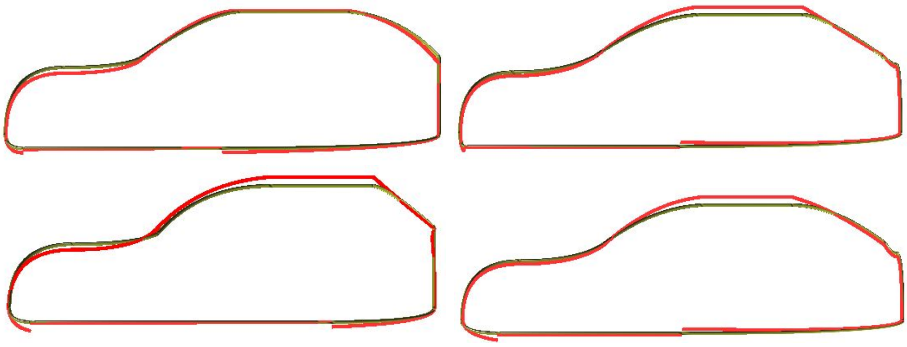

Figure 4: Die Autos nach der Transformation in den Eigenraum und zurück (vor der Transformation grün, nach der Transformation rot)

dass die aufeinander folgenden Kurven den gleichen Start-/Endpunkt haben. Die weiteren "Unschicklichkeiten"lassen sich verbessern, indem man die sinnvollen Eigenkoeffizienten findet.

Wie ich schon erwähnt habe, man braucht nur sehr wenig Parameter (in diesem Fall 10), um eine zahlreiche Menge von verschiedenen Autos zu generieren. Diese sind einfach zu manipulieren, und die kleinere Änderungen bei der Eingabe ergeben schnelle und vielfältige Ausgabe. Das ist der Vorteil meines Verfahrens.

Der Nachteil ist eine geringe Datenbank von Testautos. Es gibt eine riesige Menge von verschiedensten Autos, und man muss eine viel größere Menge als 30 Beispielautos analysieren, um alle Varianten von Fahrzeuge generieren zu können.

7 Fazit und Schluss

Die Aufgabe des Projektes war es eine Technik vorzustellen, wie man ein Auto in 2D automatisch modellieren kann. Dafür wurden die typischen Autos mit Hilfe von PCA analysiert. Mittels Dimensionsreduzierung habe ich die Hauptkomponenten herausbekommen, welche für die Generierung des neuen Autos relevant sind.

Der nächste Schritt wäre es zu prüfen, welche Zahlenwerte für diese Komponenten sinnvoll sind und diese als GUI-Element darzustellen, so dass man die manipulieren und dadurch verschiedene neue Fahrzeuge erzeugen kann. Dafür analysiere ich die einzelne Eigenkoeffizienten a_i von allen Testautos, die ich für PCA verwendet habe. Für jeden Koeffizient berechne ich den Mittelwert sowie das Minimum und das Maximum. Dann ist es die Aufgabe zu analysieren, wie beeinflusst jeder Koeffizient das Ergebnisauto, um für jeder daraus entstehender Parameter den sinnvollen Wertebereich zu finden.

References

[Blanz u. Vetter] Volker Blynz; Thomas Vetter: A Morphable Model For The Synthesis Of 3D Faces. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA 1999

[Jolliffe] I.T.Jolliffe: Principal Component Analysis,Second Edition. Springer-Verlag, New York, 1986

[Maier] Benjamin Maier: Ein Morphable Model für Fische. Diplomarbeit. Tübingen, Mai 2007

[Aguilera] A.M. Aguilera; R. Guti´errez; F.A. Ocana and M.J. Valderrama. Computational approaches to estimation in the principal component analysis of a stochastic process. Appl. Stoch. Models Data Anal. 1995

[Aitchison] J. Aitchison. Principal component analysis of compositional data. Biometrika, 70, 1983

[Ali, Clarke u. Trustrum] A. Ali; G.M. Clarke and K. Trustrum. Principal component analysis applied to some data from fruit nutrition experiments. Statistician, 34, 1985

[Al-Kandari] N. Al-Kandari; Variable Selection and Interpretation in Principal Component Analysis. Unpublished Ph.D. thesis, University of Aberdeen, 1998

[Hecker, Lischinski, Szeliski u. Salesin] F. Pighin, J. Hecker, D. Lischinski, Szeliski R, and D. Salesin. Synthesizing realistic facial expressions from photographs. In Computer Graphics Proceedings SIGGRAPH'98, pages 75–84, 1998.

[Prusinkiewicz und Lindenmayer 1990] Prusinkiewicz, Przemyslaw; Lindenmayer, Aristid: The algorithmic beauty of plants. New York: Springer-Verlag, 1990 (The Virtual laboratory).– ISBN 0387946764.

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

[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.

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

[www.carlook.net] http://carlook.net/de/db/ (14. Dezember 2015).

[www.pixabay.com] https://pixabay.com (26. Dezember 2015)., Bilder verwendet unter Creative Commons CC0 Lizenz.