# Finite Elemente Analyse auf Android

**Dipl.-Ing. Daniel Goller, Christian Glenk, M.Sc., Prof. Dr.-Ing. Frank Rieg**

Lehrstuhl für Konstruktionslehre und CAD, Universität Bayreuth
Universitätsstraße 30, 95447 Bayreuth
E-Mail: daniel.goller@uni-bayreuth.de; christian.glenk@uni-bayreuth.de
Internet: http://www.lscad.de

***Inhalt:*** *Im folgenden Beitrag wird das Konzept und die Entwicklung einer Finiten Elemente App für Android beschrieben, sowie die dafür verwendete Softwareumgebung.*

***Abstract:*** *This paper describes the concept and development of a finite element app for android, as well as the software environment that was used.*

***Stichwörter:*** *Finite Elemente Analyse, Android, App, C#, .NET, Z88Aurora*

***Keywords:*** *Finite Elemente Analysis, Android, App, C#, .NET, Z88Aurora*

## 1 Einleitung

Die Finite Elemente Analyse hat sich seit vielen Jahren in den Ingenieurwissenschaften bewährt und ist heute oft als fester Bestandteil in simulationsbasierten Entwicklungsprozessen anzutreffen. Eine Simulation besteht meist aus dem Präprozessing, in dem die Ausgangparameter der Studie, wie beispielsweise die Geometrie und die Randbedingungen, festgelegt werden. Anschließend wird die Aufgabenstellung vom Solver berechnet und die Ergebnisse im Postprozessor dargestellt. Dieses Vorgehen liegt auch der am Lehrstuhl für Konstruktionslehre und CAD entwickelten, freien Finite Elemente Analyse Software Z88Aurora zugrunde.

Die schnelle Entwicklung der Computertechnik und der wachsende verfügbare Speicher erlauben es, immer detailliertere und komplexere Probleme mit diesem Verfahren zu lösen. Im Folgenden wird die Entwicklung einer Android App beschrieben, die es ermög-

licht, einfache strukturmechanische Aufgabenstellungen zu lösen, um in diesem Bereich erste Schritte auf einer neuen Plattform zu gehen.

## 2 Android OS

Android ist ein weit verbreitetes Betriebssystem, das hauptsächlich auf mobilen Endgeräten eingesetzt wird, aber auch für PCs oder Notebooks verfügbar ist. Es wird offiziell seit 2007 von Google entwickelt, das erste Handy mit Android Betriebssystem wurde 2008 ausgeliefert [1]. Seitdem setzt es sich immer mehr am Markt durch und ist heute das verbreitetste Betriebssystem für Mobiltelefone. Im zweiten Quartal 2014 liefen von allen ausgelieferten mobilen Geräten 84,7 % mit Android [2]. Der hohe Marktanteil liegt zum einen daran, dass Android das erste mobile Betriebssystem war, das für Mobiltelefonhersteller kostenlos angeboten wurde, zum anderen an der quelloffenen Entwicklung. Dies bringt nicht nur Transparenz, sondern auch einen technologischen Innovationsschub. Dadurch, dass viele Hersteller die gleiche Softwarebasis verwenden, spielt sich der größte Teil des Wettbewerbs im Bereich der Hardware ab, wodurch die Hersteller gezwungen sind, sich durch stetige Verbesserungen und Innovationen am Markt zu behaupten [1].

Applikationen sind auf einem Android Betriebssystem modular aufgebaut und relativ lose gekoppelt. Einzelne Komponenten wie z. B. die Tastatur oder die Kamerasoftware können sehr einfach ausgetauscht oder erweitert werden. Für Entwickler stellt die Plattform zahlreiche Funktionen zur Verfügung, um die in den mobilen Endgeräten vorhandenen Sensoren wie z. B. das GPS, die Kamera, das Mikrofon oder den Beschleunigungssensor anzusteuern, sowie Signale der Touch-Oberfläche auszuwerten. Jede Android App läuft in einer eigenen VM (virtuellen Maschine), ähnlich der Java VM, um den Prozess und die Anwendungsdaten zu schützen und die Ausführbarkeit anderer Anwendungen nicht zu beeinträchtigen.

Die grafische Oberfläche, die der Nutzer bedient, wird durch sog. Activities erzeugt. Jede Ansicht ist eine eigene Activity. Nachdem eine Activity vom System aufgebaut wurde, wird sie dem Nutzer am Display angezeigt und dieser kann mit ihr interagieren. Nach dem Start einer neuen Activity gerät die vorhergehende in den Hintergrund, liegt aber trotzdem weiterhin im Speicher. Das System beendet die Activities und damit auch die Anwendungen selbst, wenn der Speicher oder die Rechenressourcen benötigt werden. Der Entwickler kann durch sogenannte Intents auch Activities anderer Applikationen aufrufen, Daten

übergeben und Rückgabewerte empfangen. Dies ist z. B. sinnvoll wenn in der App eine Datei geladen werden soll. Statt den Pfad per Hand einzugeben kann auch die Activity eines installierten Dateimanagers gestartet werden. So kann der Nutzer die Datei bequem mit Hilfe der grafischen Anzeige des Dateimanagers suchen und auswählen und die eigene Applikation erhält den Pfad als Rückgabewert.

# 3 Entwicklung einer Finiten Elemente App

Um eine Finite Elemente Analyse auf einem Android Gerät durchzuführen und damit die Möglichkeiten und Grenzen dieses Betriebssystems in diesem Bereich zu ermitteln, wird eine App entwickelt, die einfache strukturmechanische Probleme mit Hilfe der FEA lösen kann.

Die Zielgruppe einer solchen App sind z. B. Studenten, die sich mit den Grundlagen der FEA beschäftigen und dies in der Vorlesung oder Zuhause auf Android Geräten testen wollen. Durch das Erstellen und Berechnen einfacher 2D Balkenstrukturen werden grundlegende Effekte wie Spannungsverteilung oder Verschiebungen im Lastfall analysiert und erlernt.

## 3.1 Werkzeuge

Zum Entwickeln von Android Anwendungen wird das Android SDK (Software Development Kit) und eine Entwicklungsumgebung benötigt. Hier gibt es viele Möglichkeiten, die sich jeweils für unterschiedliche Anwendungsfälle eignen. Im Folgenden werden die Tools beschrieben, die für die FEA App verwendet wurden.

### Android SDK

Das Android SDK wird kostenlos von Google zur Verfügung gestellt. Darin enthalten ist die API (Application Programming Interface), um auf die Funktionen des Betriebssystems zuzugreifen. Zudem sind im SDK Tools enthalten, die einige Aufgaben des Entwicklungsprozesses unterstützen, wie z. B. das Testen oder Debuggen des Codes. [3]

### .NET-Framework

Das 2002 von Microsoft veröffentlichte .NET-Framework ermöglicht objektorientiertes Programmieren mit unterschiedlichen Sprachen (z. B. C++, C#, VB oder JScript) unter

Windows. Der Compiler erzeugt dabei Bytecode der nur mit der .NET CLR (Common Language Runtime), dem Interpreter, ausführbar ist. In .NET werden Konzepte aus der Programmiersprache Java aufgegriffen, in der native Android Applikationen geschrieben werden.

Im Framework ist eine umfangreiche Funktionsbibliothek in Form einer Klassenbibliothek enthalten, um die Entwicklung von Anwendungen zu beschleunigen. Die Installation beinhaltet über 10.000 Typen, unter anderem Klassen für den Umgang mit Zeitfunktionen, mathematischen Operationen, Oberflächenprogrammierung und Web-Anwendungen. Damit ist das .NET-Framework ein effizientes Werkzeug der Softwareentwicklung. [4]

**Xamarin**

Da die für .NET Anwendungen zum Ausführen nötige CLR nur auf Windows verfügbar war, gab es bereits 2001 Bestrebungen, diese auch auf Unix Systeme sowie iOS zu portieren. 2004 wurde Mono veröffentlicht, ein open source Projekt, das einen C# Compiler und die CLR für nicht-Windows Systeme bietet. Dies ermöglicht die Nutzung eines großen Teils der .NET Bibliothek auf Betriebssystemen wie Unix, iOS oder auch Android. Eine Firma namens Xamarin leitet die Verwaltung und Weiterentwicklung des Frameworks.

Mit Hilfe von Mono lassen sich unter Verwendung des .NET-Frameworks in C# plattformübergreifend mobile Anwendungen entwickeln. Dafür stellt Xamarin auch die Entwicklungsumgebung Xamarin Studio zur Verfügung. Durch „shared code“ können die logischen Vorgänge für Android genauso wie für iOS verwendet werden, lediglich die Oberfläche muss für jedes System gesondert erstellt werden.

Neben Xamarin gibt es auch Alternativen zur plattformübergreifenden Entwicklung von Apps, die auf anderen Verfahren basieren. Beispiele hierfür sind PhoneGap (HTML 5, jQuery Mobile, CSS), Appcelerator Titanium (ähnlicher Ansatz wie PhoneGap) oder RhoMobile Rhodes (Ruby-Framework). Bei diesem Projekt fiel die Entscheidung vor allem wegen der Programmiersprache C# und die damit verbundene Möglichkeit, bereits existierenden Code weiterzuverwenden, auf Xamarin.

**OpenGL**

Die Grafikbibliothek OpenGL ist auf Android als OpenGL ES (Open Graphics Library for embedded Systems) verfügbar. Dies ist eine vereinfachte Form von OpenGL, in der

z. B. redundante Befehle entfernt wurden. Bei der Entwicklung von Anwendungen mit Mono bzw. dem .NET Framework kann mit Hilfe des OpenGL Wrappers OpenTK (Open Toolkit Library) darauf zugegriffen werden. Dieses, in C# geschriebene Modul, ist für die verbreitetsten Plattformen verfügbar und ermöglicht den Zugriff auf OpenGL Funktionen aus allen Sprachen, die das .NET Framework unterstützt.

## 3.2 Vorgehen

Um das Projekt umzusetzen, muss ein Backend mit dem Solver und der Dateiverwaltung erstellt werden, sowie eine GUI (Graphical User Interface) als Frontend, die den Nutzer durch das Präprozessing führt und im Postprozessing die Ergebnisse aufbereitet darstellt.

### Solver

Als Solver wird ein Modul verwendet, das von Prof. Dr.-Ing. Frank Rieg geschrieben wurde und das Gleichungssystem nach der Methode von Cholesky mit in-situ Speicherung löst. Die Strukturdaten sowie Materialparameter und Randbedingungen werden in einer Dateistruktur gespeichert, die der von Z88Aurora gleicht. Ergebnisse wie Spannungen, Kräfte und Verschiebungen werden jeweils in ähnlichen Dateien im Android Dateisystem abgelegt. [5]

### Oberfläche

Die Oberfläche stellt eine Herausforderung dar, da sie die Schnittstelle zum Nutzer bildet. So kann eine ineffiziente Ansteuerung die entwickelte Software unbrauchbar machen, selbst wenn der dahinterliegende Rechenkern sehr gut ist. Deshalb ist es wichtig, dass alle Funktionen und Optionen so in der GUI integriert sind, dass der Nutzer sich möglichst schnell in das Programm einarbeiten und es anschließend mit möglichst geringem Zeitaufwand verwenden kann. Um diese Ansprüche für Einsteiger und erfahrenere Nutzer gleichermaßen umzusetzen, wird ein entwickeltes Konzept umgesetzt, das es ermöglicht beiden Nutzergruppen gerecht zu werden. Zudem ist in die Oberfläche eine OpenGL Zeichenebene integriert, in der der Nutzer durch Gestensteuerung Balkenstrukturen, bestehend aus Knoten und Elementen, zeichnen kann.

**Konzept**

In jeder Activity stehen zwei Modi zur Verfügung. Im Basis Modus ist der Nutzer in seinen Möglichkeiten eingeschränkt, kann jedoch schon durch wenige einfache Eingaben eine 2D-Balkenstruktur erstellen und rechnen. Im Detail Modus ist es möglich, jeden einzelnen Parameter der Analyse selbst zu definieren. So können beispielsweise 3D Raumkoordinaten erstellt oder die Biegeträgheitsmomente von Balkenstrukturen explizit eingegeben werden. Es kann jederzeit zwischen den beiden Modi gewechselt werden, allerdings wird für jeden Modus eine eigene Struktur angelegt.


Abbildung 1: Symbole für die verfügbaren Modi

Um den Nutzer durch das Präprozessing zu leiten, werden verschiedenen Activities nach der Reihe aufgerufen, ohne die Möglichkeit wichtige Einstellungen zu überspringen.

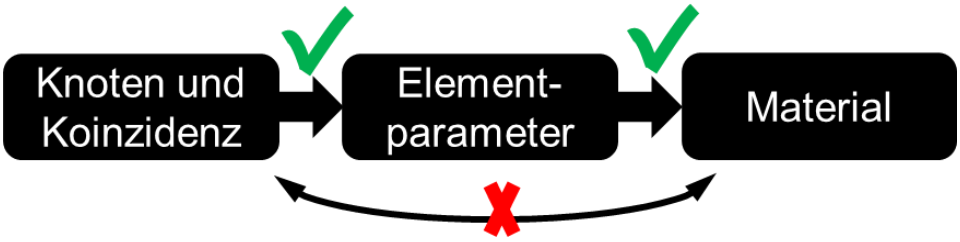


Abbildung 2: Vorgegebener Workflow

Der Nutzer kann sich nur entlang des vorgegebenen Workflows bewegen, damit sichergestellt ist, dass der Solver alle notwendigen Daten erhält um das FE-Problem zu lösen.

Die App sollte möglichst viele Vorteile des Android Systems wie z. B. die Möglichkeit der Gestensteuerung durch das Touch Display (Balkenstrukturen zeichnen) nutzen. Zudem ist die App komplett in Englisch gehalten, um sie einem breiteren Publikum zugänglich zu machen.

# 4 Nutzung der App

Nach dem Starten der Anwendung findet sich der Nutzer im Hauptmenü mit folgenden Funktionen wieder:

- „Enter Structure“: Erstellen einer neuen Struktur oder Bearbeiten einer bereits erstellten/ geladenen.
- „Load Structure“: Laden eines Projektes mit Struktur und Randbedingungen aus dem Android Dateisystem.
- „About“: Informationen über die Herausgeber.
- „Help“: Ausführliche Hilfsdokumente, die über das Internet zur Verfügung gestellt werden.

Der Workflow vom Erstellen einer Struktur bis zur Darstellung der Ergebnisse im Postprozessor ist in Abbildung 3 dargestellt. Dabei sind die vier Activities „Nodes and Elements“, „Element Geometry“, „Material Data“ und „Boundary Conditions“ zu durchlaufen. Wird ein Projekt geladen, können auch eine oder mehrere dieser Activities übersprungen werden, je nachdem wie viele Informationen nach dem Laden zur Verfügung stehen.

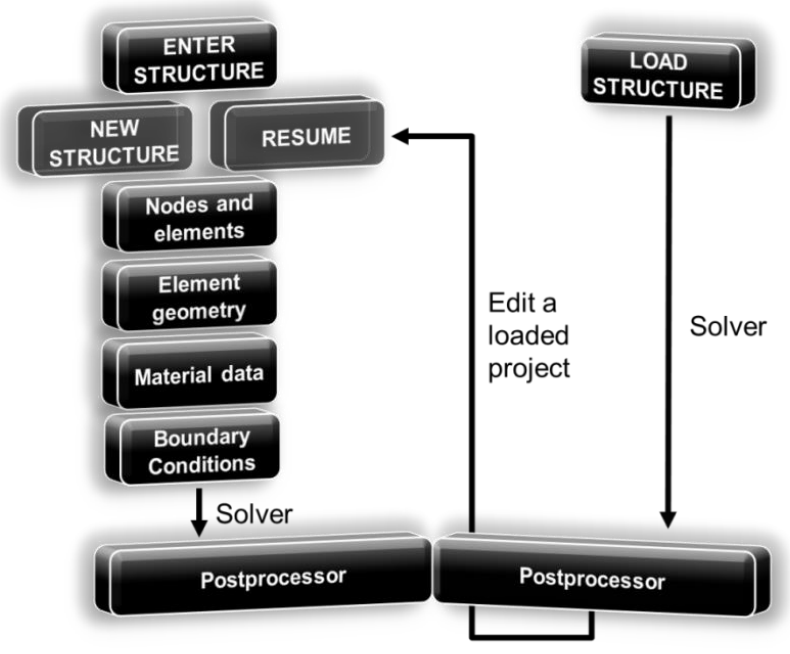


Abbildung 3: Prozess vom Erstellen/ Laden einer Struktur bis zum Postprozessor

Als Beispiel für eine Activity ist in Abbildung 4 die Basisansicht von „Nodes and Elements“ dargestellt.

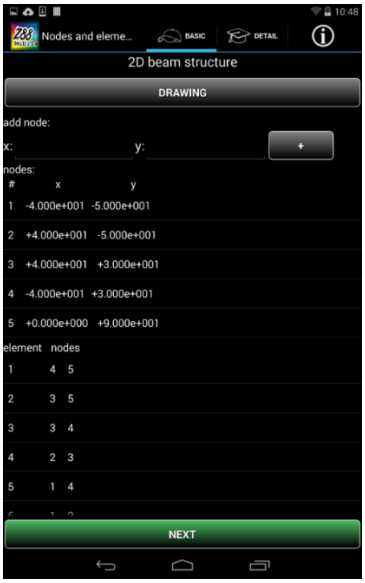


Abbildung 4: Ansicht der Activity „Nodes and Elements"

## 5 Ausblick

Nach dem Abschluss der Entwicklung und ausführlicher Tests, soll die App kostenlos im Google Playstore und ähnlichen Plattformen zur Verfügung gestellt werden. Zudem kann dann die Leistungsfähigkeit der verwendeten Hard- und Softwareumgebung auf die Probe gestellt werden, indem größere Projekte importiert und gerechnet werden, um so Grenzen und Chancen, die dieses relativ junge Betriebssystem bringt, besser einschätzen zu können.

## Literatur

[1] McClure, Wallace; Blevins, Nathan; Croft, John; Dick, Jonathan; Hardy, Chris: Professional Android Programming, Indianapolis, Ind.: Wiley Pub., 2012

[2] IDC Corporate USA: Worldwide Smartphone Shipments Edge Past 300 Million Units in the Second Quarter. http://www.idc.com/getdoc.jsp?containerId=prUS25037214 (15.09.2014)

[3] Google Inc.: Android Developers. http://developer.android.com (15.09.2014)

[4] Wenger, Rolf: Handbuch der .NET-Programmierung, Unterschleißheim: Microsoft Press, 2007

[5] Rieg, Frank; Alber-Laukant, Bettina; Hackenschmidt, Reinhard: Finite Elemente Analyse für Ingenieure, 4. Aufl. München: Carl Hanser Verlag, 2012