Interactive visualization of multi-dimensional data in R using OpenGL

Chapter 1

Introduction

The visualization of empirical and simulated data in conjunction with function graphs is a common technique for scientists to identify correlations in multivariate data, to compare distributions with known distribution functions and to illustrate state facts.
関数グラフと（組み合わせた）実験(経験)データやシミュレーションデータの可視化は多変数データの相関性を確認したり、分布を既知の分布関数と比較したり、状態事実を図解したりするための科学者の共通のテクニックです。

2D visualizations such as scatterplots, distribution and density function plots, histograms and pair plots are useful graphical techniques for interactive data analysis.
散布図、密度関数プロット、ヒストグラム、ペアプロットとしての2D可視化はインタラクティブなデータ解析のための役に立つグラフィカルテクニックです。

3D visualizations provide an additional coordinate axis.
Due to the fact that 3D visualizations are projected on 2D screen displays, interactive viewpoint navigation is required to get an adequate overview by exploring the 3D world.
In contrast, 2D visualizations do not require special navigation facilities.
3D可視化は追加の座標軸を提供する。
2Dスクリーンディスプレイに映し出される3D可視化の事実のために、インタラクティブな視点ナビゲーションは3D世界を探索することによって十分な概観を得るために要求される。
対照的に、2D可視化は特別なナビゲーション機能を必要としない。

Three variable visualizations and the usage of modern graphics drawing techniques like transparency give scientists abilities at hand to analyse multivariate data.
３変数の可視化や透かしのような現代的なグラフィック描画技術の使用は科学者に多変数データの解析を手近にする能力を与える。

The current R graphics capability lacks the ability to interactively visualize data in 3D and has limited support for 3D graphics plots.
現在のRのグラフィック能力は3Dデータをインタラクティブに可視化する能力を欠き、3Dグラフィックプロットのためのサポートは限定的です。

Interactive visualization requires certain degree of graphics rendering speed which optimized engines should deliver.
インタラクティブな可視化は最適化したエンジンが提供するべき ある程度のグラフィックレン

ダリングスピードを必要とする。

This work documents the development of a R package named "RGL".
ここでは"RGL"と名付けたRのパッケージの開発を記録する。

It contains an interactive visualization device system with an R programming interface.
それはRプログラミングインターフェースと一緒にインタラクティブな可視化デバイスシステムを含みます。

RGL uses the OpenGL library as the rendering backend providing an interface to graphics hardware.
RGLはグラフィックハードウェアにインターフェースを提供しているレンダリングバックエンドとしてオープンGLライブラリを利用します。

Most graphics hardware vendors provide an OpenGL driver for their hardware systems and even software-only implementations exist.
ほとんどのグラフィックハードウェア販売者はそれらのハードウェアシステム用のオープンGLドライバーを提供し、ソフトウェアのみの実装さえ存在します。

Mesa

1. Approach

The R system is widely used throughout the statistical community.
Rシステムは統計学コミュニティの至る所で広く利用されている。

RGL extends R with a graphics device system that includes new graphical capabilities.
RGLは新たなグラフィカル能力を含むグラフィックデバイスシステムでRを拡張している。

By adapting some common programming interface designs from the current R graphics system, users need less time to study the usage of RGL.
現在のRグラフィックシステムからいくつかの一般的なプログラミングインターフェースデザインを適合することによって、利用者はRGLの利用方法を学習する時間を減らすことができる。

This leads to the approach of analyzing R graphics capabilities and its programming interface to derive requirements and guidelines for the software design.
これにより、Rグラフィックス性能とそのプログラミングインターフェイスを分析したり、ソフトウェア設計の要件とガイドラインを導き出したりするアプローチにつながります。

The software concept evolves using a scene database oriented approach to design the functionality.
ソフトウェアの概念は、機能を設計するためのシーンデータベース指向のアプローチを使用して進化しています。

The development has been done using the methodology of object-oriented software development and an object-oriented programming language.
開発はオブジェクト指向ソフトウェア開発の方法論とオブジェクト指向のプログラミング言語

を使用しています。

The advantage is a solid transition from the software concept to design, and finally to implementation.
利点は、ソフトウェアコンセプトから設計、そして最終的には実装への堅実な移行です。

The object-orientation provides techniques that structure software systems logically using modularization, abstraction, data encapsulation and polymorphism, which leads to a reduction of complexity.
オブジェクト指向は、モジュール化、抽象化、データカプセル化、および多形性を論理的に使用してソフトウェアシステムを構造化する技術を提供し、複雑さの低減につながる。

The methodology of Software Patterns has been applied to design a solid architecture.
ソフトウェアパターンの方法論は、強固なアーキテクチャを設計するために適用されています。

1.2 Further reading

Chapter 2 gives an overview of the R environment and a brief introduction into key features of the R language. A short overview of the graphics capabilities and programming interface is presented. Futhermore, the extension mechanism of R is discussed.
第2章では、R環境の概要と、R言語の主要機能の概要について説明します。 グラフィックス機能とプログラミングインタフェースの概要を示します。 さらに、Rの拡張機構について議論する。

Chapter 3 gives an introduction into the field of interactive visualization. Real-time rendering will be outlined together with an introduction to OpenGL.
第3章では、インタラクティブな可視化の分野について紹介します。 リアルタイムレンダリングについては、OpenGLの紹介とともに概説します。

Chapter 4 describes the functionality of the device, the graphics rendering model and explains the application programming interface using a top-down approach.
第4章では、デバイスの機能、グラフィックスレンダリングモデル、およびトップダウンアプローチを使用したアプリケーションプログラミングインターフェイスについて説明します。

In Chapter 5 the development process is presented. The chapter starts with an introduction to the object-oriented methodology including the C++ lan- guage, UML notations and Software Patterns. The R extension mechanism via shared libraries and a discussion about windowing systems give the out- line for the architecture design. A detailed bottom-up description on a module- and class-level with details on method implementations round up the C++ implementation. The RGL API implemention in R completes the software system documentation.
第5章では開発プロセスを紹介します。 この章では、C ++言語、UML表記、ソフトウェアパターンなどのオブジェクト指向メソドロジについて説明します。 共有ライブラリによるR拡張

メカニズムとウィンドウシステムに関する議論は、アーキテクチャ設計のアウトラインを示しています。 メソッドの実装に関する詳細を含むモジュールおよびクラスレベルの詳細なボトムアップの説明は、C ++実装をラウンドアップします。 RのRGL APIの実装は、ソフトウェアシステムのドキュメントを完成させます。

Chapter 6 gives some examples to illustrate features of RGL package.
第6章では、RGLパッケージの機能を説明するいくつかの例を示します。

Chapter 7 gives a summary and a future outlook of further development plans.
第7章では、今後の開発計画の概要と今後の展望について述べる。

Chapter 3
Interactive visualization

Interactive visualization is common in the field of computer aided design, scientific visualization and computer games. A computer graphics system is the core providing computer-generated images.
インタラクティブな可視化は設計、科学の可視化、コンピュータゲームを助けるコンピュータの共通分野です。コンピュータグラフィックシステムはコンピュータが作り出す画像を提供する核となっています。

Angel[2] describes five major components that can be found in a computer graphics system: Processor, Memory, Frame buffer, Output device and Input devices. At present, almost all graphics systems are raster based. A picture is produced as an array – the raster – of picture elements, or pixels, within the graphics system.
エンジェルはコンピュータグラフィックシステム内で見つけられる五つの有名な要素（プロセッサ,メモリー,フレームバッファ,出力デバイス,入力デバイス）を述べる。現在、ほとんど全てのグラフィックシステムはラスターが元になっている。写真はグラフィックシステム内部で、画像素子もしくは画素の行列(ラスター)として提供される。

3.1 Geometry-based Computer graphics

Geometry-based computer graphics use three-dimensional primitives such as points, lines, triangles, quadrilaterals and polygons as basic building blocks to construct complex objects.
幾何学ベースのコンピュータグラフィックは複雑なオブジェクトを構築するための基礎構成要素として点、線、三角形、四角形、多角形のような3次元の原始関数を使います。

The primitives are described using vertices in a local coordinate space.

その原始関数は局所的な座標空間の頂点を利用して述べられている。

The object is transformed using translation, rotation and scaling to map it into the eye coordinate space, where it actually gets rendered.
オブジェクトは実際にレンダリングされる視線座標空間に対応させるために平行移動、回転、スケーリングを使った変換が行われる

Attributes such as color, material properties, normal vectors, edge flags and texture coordinates are associated with the vertex.
色、材料特性、法線ベクトル、エッジフラグ、テクスチャ座標などの属性は、頂点に関連付けられます。

This section focuses on geometry-based computer graphics used for high-performance rendering.
このセクションではハイパフォーマンスレンダリングのために使用される幾何学ベースのコンピュータグラフィックに焦点を当てる。

For detailed information on 3D computer graphics, including ray-tracing and ray-casting, see[21].
レイトレーシングとレイキャスティングを含む3Dコンピュータグラフィックの詳細情報は[21]をご覧ください。

Chapter 1

Introduction

The visualization of empirical and simulated data in conjunction with function graphs is a common technique for scientists to identify correlations in multivariate data, to compare distributions with known distribution functions and to illustrate state facts.

2D visualizations such as scatterplots, distribution and density function plots, histograms and pair plots are useful graphical techniques for interactive data analysis.

3D visualizations provide an additional coordinate axis. Due to the fact that 3D visualizations are projected on 2D screen displays, *interactive viewpoint navigation* is required to get an adequate overview by exploring the 3D world. In contrast, 2D visualizations do not require special navigation facilities.

Three variable visualizations and the usage of modern graphics drawing techniques like transparency give scientists abilities at hand to analyse multivariate data.

The current R graphics capability lacks the ability to interactively visualize data in 3D and has limited support for 3D graphics plots. Interactive visualization requires a certain degree of graphics rendering speed which optimized engines should deliver.

This work documents the development of a R package named "RGL". It contains an interactive visualization device system with an R programming interface. RGL uses the OpenGL library as the rendering backend providing an interface to graphics hardware. Most graphics hardware vendors pro-

vide an OpenGL driver for their hardware systems and even software-only implementations exist.

1.1 Approach

The R system is widely used throughout the statistical community. RGL extends R with a graphics device system that includes new graphical capabilities. By adapting some common programming interface designs from the current R graphics system, users need less time to study the usage of RGL.

This leads to the approach of analysing R graphics capabilities and its programming interface to derive requirements and guidelines for the software design.

The software concept evolves using a scene database oriented approach to design the functionality.

The development has been done using the methodology of object-oriented software development and an object-oriented programming language. The advantage is a solid transition from the software concept to design, and finally to implementation. The object-orientation provides techniques that structure software systems logically using modularization, abstraction, data encapsulation and polymorphism, which leads to a reduction of complexity. The methodology of *Software Patterns* has been applied to design a solid architecture.

1.2 Further reading

Chapter 2 gives an overview of the R environment and a brief introduction into key features of the R language. A short overview of the graphics capabilities and programming interface is presented. Futhermore, the extension mechanism of R is discussed.

Chapter 3 gives an introduction into the field of interactive visualization. Real-time rendering will be outlined together with an introduction to OpenGL.

Chapter 4 describes the functionality of the device, the graphics rendering model and explains the application programming interface using a top-down

approach.

In Chapter 5 the development process is presented. The chapter starts with an introduction to the object-oriented methodology including the C++ language, UML notations and *Software Patterns*. The R extension mechanism via shared libraries and a discussion about windowing systems give the outline for the architecture design. A detailed bottom-up description on a module- and class-level with details on method implementations round up the C++ implementation. The RGL API implemention in R completes the software system documentation.

Chapter 6 gives some examples to illustrate features of RGL package.

Chapter 7 gives a summary and a future outlook of further development plans.

Chapter 3

Interactive visualization

Interactive visualization is common in the fields of computer aided design, scientific visualization and computer games. A computer graphics system is the core providing computer-generated images.

Angel[2] describes five major components that can be found in a computer graphics system: Processor, Memory, Frame buffer, Output devices and Input devices. At present, almost all graphics systems are raster based. A picture is produced as an array - the raster - of picture elements, or pixels, within the graphics system.

3.1 Geometry-based Computer graphics

Geometry-based computer graphics use three-dimensional primitives such as points, lines, triangles, quadrilaterals and polygons as basic building blocks to contruct complex objects. The primitives are described using vertices in a local coordinate space. The object is transformed using translation, rotation and scaling to map it into the eye coordinate space, where it actually gets rendered. Attributes such as color, material properties, normal vectors, edge flags and texture coordinates are associated with the vertex. This section focuses on geometry-based computer graphics used for high-performance rendering. For detailed information on 3D computer graphics, including ray-tracing and ray-casting, see [21].