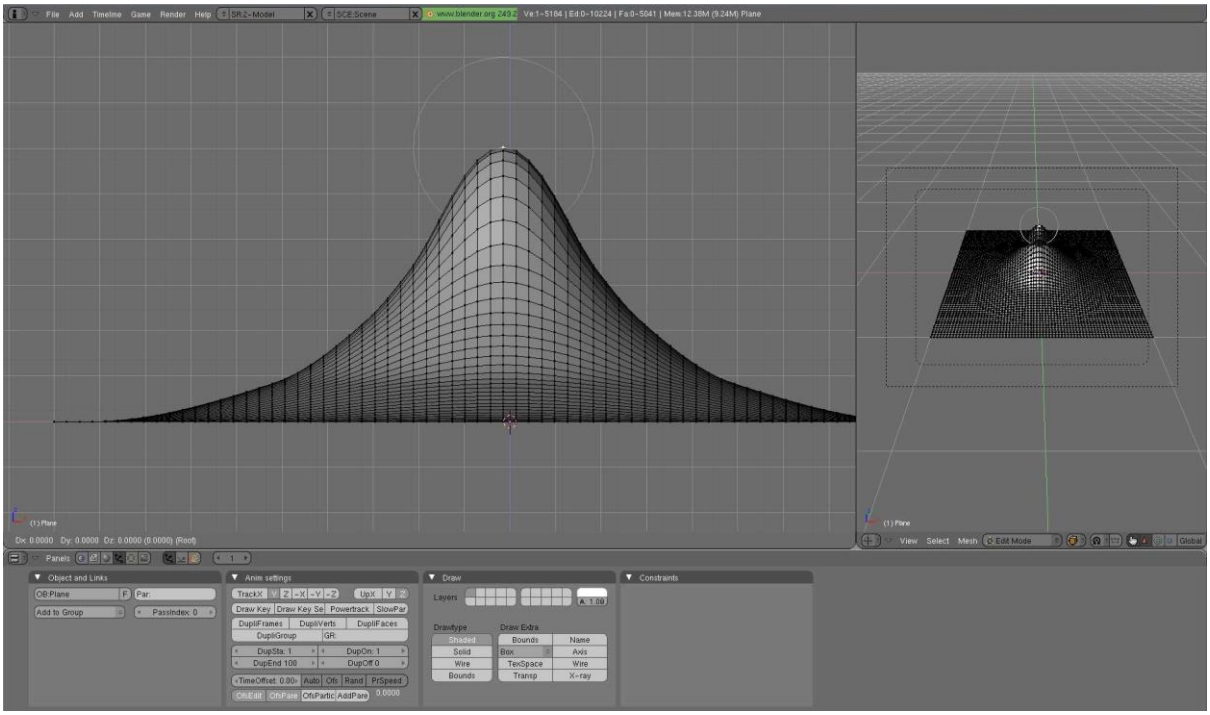
Wolfgang Müller
27. Februar 2022

3D Plotting auf andere Art



MI-Proj-M: Projekt zur Medieninformatik im Sommer 2022

Ziel des Praktkums ist es, für zwei bekannte Plot-Bibliotheken, nämlich **ggplot2** (R) und matplotlib (Python) Erweiterungen zu erstellen, die es ermöglichen, von diesen Bibliotheken erstellte Plots direkt in das Computergraphikpaket **Blender** zu exportieren. Wir wollen die folgenden Möglichkeiten schaffen:

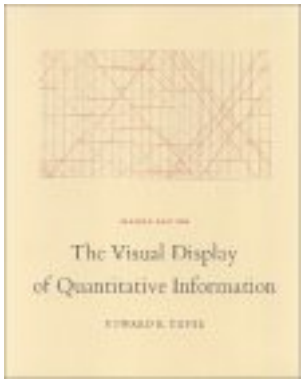
- Einen anderen Blick für Größenordnungen durch Perspektive
- Kombination von Plot und Grafik (Plot über Schneehöhe neben einem Schneemann)
- Compositing
- …

Blender wird als Werkzeug gewählt, weil es ein Open Source Werkzeug ist, mit großer Nutzergemeinde und Python API.

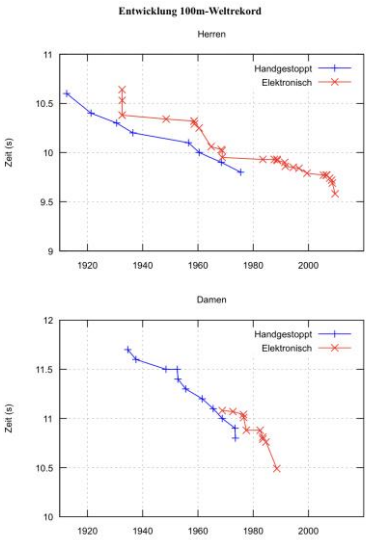
Warum ist das interessant?

Die Aufbereitung von Daten spielt eine nicht zu unterschätzende Rolle. Schwierige Vorgänge werden visuell aufbereitet besser verstanden. Und von einem Vortrag bleibt eventuell eine Folie, ein Plot hängen, wenn er besonders ist. Gute Visualisierungen können Sichtbarkeit weit über ihre Erstverwendung erhalten.

Es verwundert nicht, dass es viele **Plotting Tools** wie z.B. Gnuplot gibt, und generelle *Data Science* Werkzeuge wie Mathematica, Matlab und R, sowie Julia oder Python elaborate **Plotting-Bibliotheken** ihr Eigen nennen.



Was schon eher verwundert, ist, dass der Ansatz zur Visualisierung vieler Plotting-Werkzeuge oft “hausmacherisch” ist, trotz der Existenz von Büchern wie “The Viusual Display of Quantitative Information” von Edward R. Tufte. In diesem Buch wird nämlich klar gezeigt, wie die klassische Praxis des Plotting eher die Daten verschleiert anstatt die Daten klar herauszustellen.



Tufte empfiehlt die Minimierung der “non-Data-Ink”, also das “Drumherum” weniger fett darzustellen als die “Data Ink” also die Tinte, die den eigentlichen Plot darstellt. Demgegenüber ist das hier gezeigte klassische Beispiel von Gnuplot vor einigen Jahren recht “schwer”, denn die Achsen sind sichtbarer als die eigentlichen Plots.

Nun, die Position von Tufte ist recht extrem und wird nicht von Jedem geteilt. Es wird aber deutlich, dass moderne Plots anders aussehen.

Hier tritt **ggplot2** auf den Plan, eine Plotting-Bibliothek für R, die auf Basis von einem Buch (“A Visual Grammar of Graphics”) ermöglicht, Plots aus Bausteinen syntaktisch und visuell elegant zusammenzustellen. Diese Plot-Bibliothek hat eine Wirkung über die Sprachgrenzen von R hinaus. Die Bibliothek ist jedoch von der Anlage her auf 2D-Plots spezialisiert.

Innerhalb Python ist die Methode der Wahl Matplotlib, nicht ganz so elegant, aber auch mit der Möglichkeit, 3D zu plotten.

Allen diesen Methoden ist gemeinsam, dass sie zunächst bezwecken Daten für die Verwendung in *Schriftstücken* aufzubereiten. Dies ist nicht immer ein Endprodukt, ein

Beispiel ist **shiny**, die Möglichkeit R-Plots als HTML/Javascript Webseite aufzubereiten und so interaktiv werden zu lassen.

Die Sicht auf das Plotten von 3-D-Plots ist zwiegespalten. Paradoxerweise kann die dritte Dimension zwar *schön, ansprechend* scheinen, aber die Erfassung der Daten erschweren. Hiervon wollen wir uns aber nicht bremsen lassen. *Wir wollen die Erweiterungen erstellen, und dann diese Erweiterungen auch dokumentieren und demonstrieren. Die Erweiterungen sollen als FOSS (Free and Open Source Software) zur Verfügung stehen.*

Wie wollen wir vorgehen?

1. Wir müssen bereits existierende, ähnliche Pakete ansehen und auf Basis dessen unsere Alleinstellungsmerkmale definieren. https://github.com/HazilMohamed/plot-in-blender ist ein Beispiel, wie man in Blender plotten kann. Was kann es, was wir möchten, was kann es nicht?
2. Unser Programm wird mehrsprachig. Wir werden eine Basis-Architektur erstellen, die den Blender-nahen Python-Teil von Bibliotheks-spezifischen Teilen in R und Python trennt. Hierzu werden wir uns mit verschiedenen Arten des Interfacing zwischen R und Python auseinandersetzen. Machen wir einen Server, der JSON-Botschaften empfängt und die an Blender weiterleitet? Starten wir von R aus Skripte? Erstellen wir einen einzigen Prozess, der R und Python enthält? Hier müssen wir die Möglichkeiten untersuchen und zu einer Entscheidung kommen.
3. Wir müssen Matplotlib und ggplot2 analysieren. Wie kommt der Plot auf die (Bildschirm-)Seite, welche Low-Level Funktionen werden eingesetzt? Wo können wir "reingreifen", so dass Blender hinterher möglichst viel über die Graphikelemente weiß. Ich kann schon etwas spoilern, zumindest für ggplot2 habe ich schon einen Ort gefunden, wo Graphik-Primitive angesprochen werden. Linienzüge, Punkte und mehr.
4. Styles: Plotting-Tools stellen Styles zur Verfügung. Sinnvolle Styles in Blender werden anders aussehen. Wie lösen wir das Erstellen von Styles in Blender und die Nutzung in unserem Werkzeug?
5. Wir müssen dies dann umsetzen, so dass es funktioniert. Ich erwarte von jedem:r Teilnehmer:in, dass die Punkte von 1-3 nach dem Projekt verstanden sind.
6. Um diese Kerntätigkeit herum gibt es viele weitere Tätigkeiten, die während oder nach der Entwicklung durchzuführen sind. Wie intensiv sie durchgeführt werden, hängt davon ab, wie viele wir sein werden.

6.1. Dokumentation: Wir müssen für uns dokumentieren, welche Entscheidungen wir getroffen haben, worauf wir aufsetzen etc. Wenn möglich soll diese Dokumentation auf Englisch erfolgen.

6.2. Beispiele: FOSS-Programme leben davon, dass man ihren Wert schnell testen kann. Erstellen wir mit unserem Werkzeug Blender-Plots, die den Wert demonstrieren. Sieht es einfach unvergesslich aus? Werden vielleicht Animationen möglich, die zeigen, wie sich der Plot zusammensetzt? Oder sieht man von der Spitze einer auf einem Intervall geplotteten Exponentialfunktion, wie krass klein die Datenpunkte 3000Meter tiefer sind? Folgen wir der Funktion in einer Achterbahnfahrt? Nehmen wir einen 3-D-Plot und schneiden in einer abgeleiteten Animation eine Scheibe heraus? Hier sollten wir schon früh Möglichkeiten brainstormen, auf die wir dann hinarbeiten.

6.3. Code-Qualität und Code-Aufbereitung: Ebenso ist es wichtig, dass der Code lesbar ist. Dies sollten wir mit Code Reviews unterstützen. Wir brauchen für den ganzen Entwicklungsprozess eine funktionierende Continuous Integration Pipeline, die unsere Entwicklungen testet und uns schnelle Weiterentwicklung ermöglich.

6.4. Gruppiert wird dies in einer Projekt-Webseite, die auf GitHub gehostet werden soll und auf (mindestens) zweisprachigen GitHub-gehosteten Code verweist.

6.5. Installation und Package Management: Mittels PIP und CRAN sollte so viel wie möglich von unserer Software direkt und einfach installierbar sein. Ich halte für unwahrscheinlich, dass dies die Installation von Blender einschließen kann. Auch hier müssen wir uns gemeinsam Gedanken über sinnvolle Lösungen machen.

7. Schließlich ist ein Projektbericht zu verfassen.

Perspektiven: Wir sind mehrere Personen, die 180 Stunden arbeiten sollen. Nehmen wir an, der Bericht ist in einer Woche zu schreiben, so bleiben uns 140 Stunden pro Person für die Entwicklung. Da ist viel möglich. Ich fände spannend, wenn wir es in einem Semester schaffen, in der Ggplot2 und Matplotlib Community sichtbar zu werden. Entscheidend wird sein, Möglichkeiten zu erkennen, aber auch nicht zu viel zu wollen.