# **S**ystem **E**ntwicklungs **P**rojekt

-

# Visualisierung von bakteriellen Genomen

(Visualization algorithms for whole bacterial genomes)

von Daniel Koitzsch und Christian Becker
SEP-Betreuer: Dr. Peter Kipfer
Lehrstuhl: Computergrafik und Visualisierung,
Prof. Dr. Rüdiger Westermann

Ein Genom bezeichnet die Gesamtheit der Erbanlagen eines Lebewesens. Die Informationen, die für die Vererbung von Eigenschaften erforderlich sind, sind in den Desoxyribonukleinsäuren (DNS) enthalten. Dort liegen sie in sequentieller Form von DNS-Basen vor. Die Basen heißen Adenin, Guanin, Cytosin und Thymin. Die Abschnitte der DNS-Moleküle können in kodierende und nicht-kodierende unterteilt werden. Von diesen enthalten die kodierenden Teile, die Gene, Informationen für bestimmte Proteine. Der andere Teil des DNS-Moleküls hat regulierende Aufgaben, die hier nicht weiter erwähnt werden sollen.
Die Genomgröße wird in Basenpaaren gemessen, die von folgender Art sein können: A-T bzw. T-A und C-G bzw. G-C. Das Darmbakterium „E.coli“ z.B. hat eine Genomgröße von $4x10^6$, das menschliche Erbmaterial eine Größe von $3x10^9$ (die DNA einer einzelnen menschlichen Zelle ist ca. 1,80m lang) und ein Molch hat eine Genomgröße von etwa $4x10^{10}$. Die Genomgröße hat offenbar keine Beziehung zur Komplexität des Organimus. Der Molch hat zwar ein größeres Genom, aber ist deswegen nicht zehnmal komplexer aufgebaut als ein Mensch. Der Grund dafür ist, daß eben nicht die gesamte DNS kodierende Teile enthält, sondern nur ein Bruchteil.

Um, trotz der Komplexität von Genomen, mit (Mikro-)Organismen hantieren und forschen zu können, wird das Anwendungsprogramm ARB seit 1993 entwickelt. Den Benutzern ist es möglich sich die Information grafisch stark vereinfacht und 2-dimensional darstellen zu lassen.

Im Rahmen des SEPs „Visualization algorithms for whole bacterial genomes“ ist es einerseits unsere Aufgabe eine hardware-unterstützende Version dieser Darstellung mittels OpenGL zu implementieren, andererseits verschiedene Visualisierungen durch Fragment-Shader Programme zu realisieren.

Die Gene der Genome können auf drei Arten betrachtet werden. Zum einen in der radialen Darstellung (die Gene werden auf einem Kreis angeordnet), zum anderen in der vertikalen (die Gene werden untereinander gezeichnet), und drittens in der „book“ Darstellung (die Gene werden sequentiell und am Fensterrand umgebrochen angeordnet). In den Darstellungsformen wird die Leserichtung durch einen „Haken“ bzw. Pfeil am Anfang der das Genom repräsentierenden Linie verdeutlicht. Bisher sind dabei folgende visuelle mit der Maus ausführbare Operationen möglich:

- zoomen
- markieren

In der „neuen“ OpenGL-Version wurde die Interaktion mit dem Programm durch die Maus um folgende Punkte erweitert:

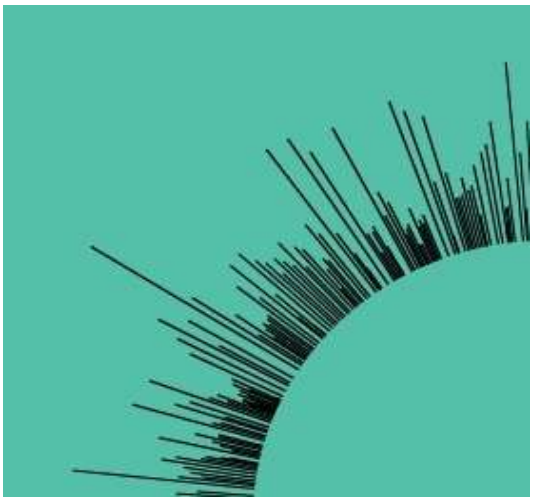
- bewegen
- drehen

Mit der Tastatur sind folgende neue Steuerungen möglich:

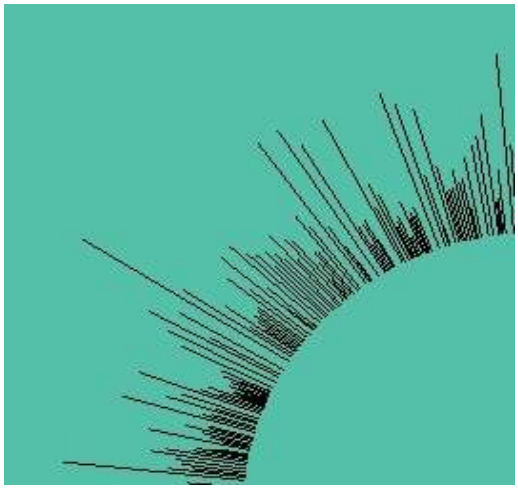
- Linesmooth („weicheres“ Zeichnen der Linien): l
- Richtungsanzeige der Gene: s
- Variieren der Linien- und Knubbeldicke: e,r bzw. q,w
- Anzeige für die Längeninformation der Gene switchen: c (Farbkodierung: weiß für die längsten und rot für die kürzesten Gene; Knubbel: markieren der längsten/kürzesten Gene)
- Drehen: 7,4,8,5,9,6 auf dem Nummerblock
- Ausgangsstellung: Pos1 bzw. Home

- Texturen anschalten: t
- Shader- Programm switchen: p

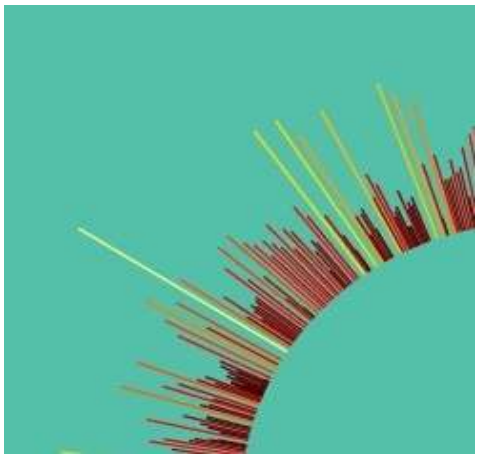
Beispiele:



*Mit Linesmooth*

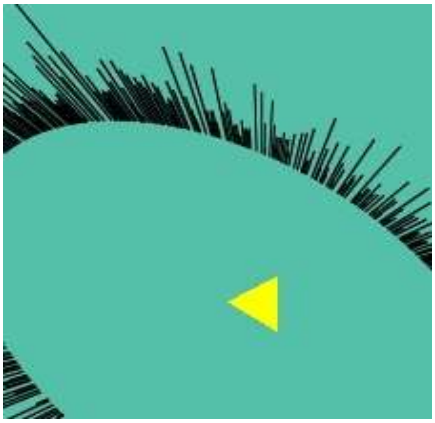


*Ohne Linesmooth*



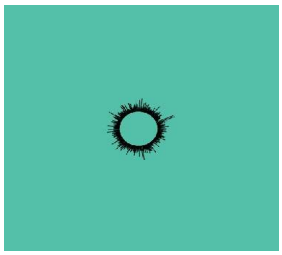
*Farbkodierung entsprechend der Genlänge*

*Markierung der längsten und kürzesten Gene*



*Drehen des Genoms (Pfeil zeigt die ursprüngliche Ausrichtung)*

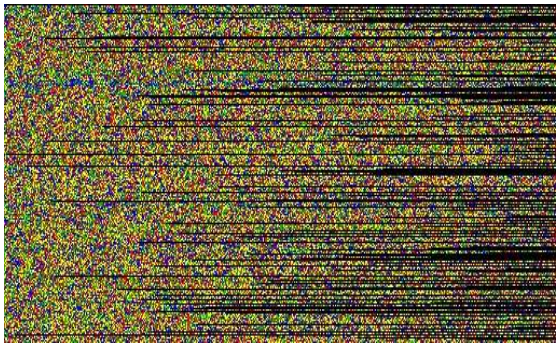
*Reinzoomen*



*Rauszoomen*

Die neue Darstellung wird erst dann angezeigt, wenn im ARB Homeverzeichnis eine Datei „showOpenGL.cfg“ existiert, deren Inhalt eine 1 ist. Ist der Inhalt eine 0 oder ist diese Datei nicht vorhanden, wird der konventionelle Darstellungsmodus gewählt.

Sollte die Grafikkarte auf der das ARB Programm ausgeführt wird, keine Shaderfunktionen unterstützten, dann kann der User die Informationen der Basensequenzen als einfache Texture anzeigen lassen. Dabei wird jedem möglichen Basenpaar (siehe oben) eine Farbe zugeordnet.
Dazu werden die gesamten Daten eines jeden Organismus geparst und Base für Base verarbeitet. Aus Performancegründen wird nicht für jedes Gen eine eigene Textur erstellt. Sondern es wird eine maximale (abhängig von der Leistung der Grafikkarte) 2-dimensionale Textur erzeugt, in der die Texturdaten der Gene jeweils eine oder mehr Zeilen einnehmen können.

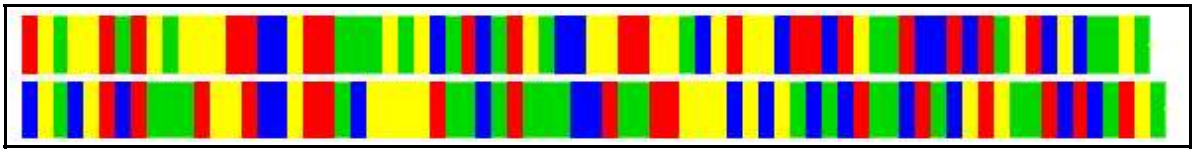


*Basensequenzinformation: Ein Teil einer Textur als Bitmap gespeichert*

Zur optisch besseren Visualisierung und Analysemöglichkeit haben wir drei verschiedene Shader implementiert. Der erste ist recht einfach und hat lediglich die Funktion die Basen voneinander abzugrenzen, damit man sie besser abzählen kann. Der zweite Shader ersetzt die Farbe einer Base durch eine entsprechende Textur. Der dritte Shader ist für die Anzeige der Codon-Textur verantwortlich.
Um die Shaderprogramme verwenden zu können werden die Gene als Rechtecke (GL_QUADS) und nicht als Linien gezeichnet.

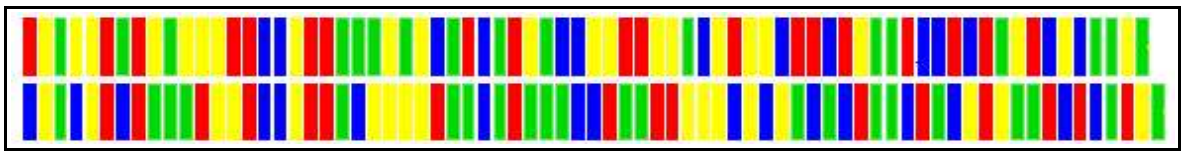
Der erste Shader trennt die einzelnen Basen von einander ab. Dies dient dazu dass man die Basen besser zählen kann.

Bild ohne Shader:



*Eine Kurze Gensequenz ohne diesen shader*

Bild mit Shader:



*Und diesselbe Sequenz mit angeschaltetem Shader*

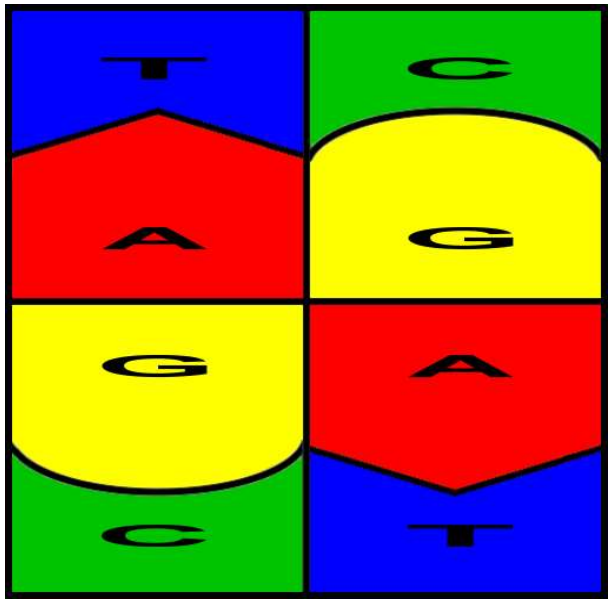
Die Realisierung ist relativ einfach:
bereits beim Aufruf wird in der dritten Texturkoordinate die Anzahl der Basen mitgegeben, damit diese im Shader direkt zur Verfügung stehen. Die Vertices links unten und links oben bekommen eine 0, die beiden Vertices auf der rechten Seite die Anzahl der Basen die auf diesem Strang liegen. Die dritte Texturkoordinate geht somit z.B. für die zweite Base von 2.000 bis 2.999. Im Shader erfolgt nun ein standardmäßiges Textur- Lookup das als Rückgabewert dient. Dieser Rückgabewert wird verändert, wenn die Nachkommastelle der dritten Texturkoordinate kleiner als 0.15 ist. Dadurch werden von jeder Base 15% abgeschnitten und durch einen schwarzen Strich ersetzt, bzw. durch einen reinen alpha- wert. Der Wert „15%“ ist ein reiner Erfahrungswert und könnte selbstverständlich geändert werden. Somit entsteht eine optische Trennung der Basen durch eine kurze Ausparung. Der Code des Shaders ist dementpsrechend gering:

```
fragout FPmain(vertout IN, uniform sampler2D tex0) {
  fragout OUT;
  OUT.col = tex2D(tex0, IN.TexCoord0.xy).rgba;
  if( frac(IN.TexCoord0.z) <= 0.15){
    OUT.col = float4(0,0,0,0);
  }
  return OUT;
}
```

Zweiter Shader:
dieses Fragment- Programm ersetzt die normale Farbe einer Base durch ein Bild der Base. Dazu wird dem Programm die ganz normale Textur- Information(siehe Bild oben: „Basensequenz- information“) mitgegeben und zusätzlich, in einer zweiten Textur, ein Referenzbild von den Basen(siehe Bild „Grundbild für die Basenpaare“).
Auf diesem Referenzbild sind die vier möglichen Basenkombinationen dargestellt. Unser Gen- Rechteck bekommt nun im Shader die ganz normalen xy- Texturkoordinaten für die Basensequenzinformationen mit. Die z- Texturkoordinate enthält wie im ersten Shader- Program die Anzahl der Basen. Zusätzlich bekommen die unteren Vertices des Quads eine 0.0 und die oberen eine 0.5 als w- Koordinate mit.

Im Shader passiert nun zunächst ein ganz normales Texture-Lookup, so dass die Farbinformation für jedes Fragment vorhanden ist. Anhand dieser Farbe wird nun bestimmt, welcher Ausschnitt des Referenzbildes auf das Fragment gelegt wird. Die Y-Koordinate für den Zugriff auf die Textur des Referenzbildes steht bereits in der w-Texturkoordinate zur Verfügung. Da diese automatisch inter-poliert wird kann sie für die Basen C und T ungeändert übernommen werden, und für die Basen A und G wird 0.5 addiert. Die X-Koordinate muss nun noch berechnet werden:

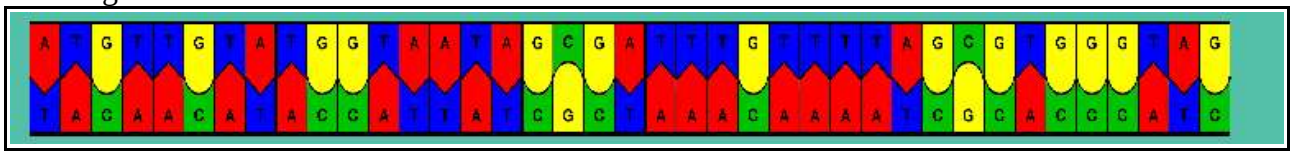
*Grundbild für die Basenpaare*




es wird wieder die Nachkommastelle der Basennummer genommen, die immer zwischen 0 und 1 ist. Diese wird halbiert und wenn es sich um eine G oder T Base handelt wird noch eine 0.5 addiert. Damit erhält man immer Texturkoordinaten zwischen 0 und 0.5, sowohl für x als auch für y. Für die entsprechenden Basen wird dann noch ein passender Offset addiert, damit man im richtigen Viertel der Textur des Referenzbildes landet. Als Shaderprogramm sieht das folgendermaßen aus:

```
fragout FPmain(vertout IN, uniform sampler2D tex0, uniform sampler2D tex1)
{
  fragout OUT;
  float4 textureValue = tex2D(tex0, IN.TexCoord0.xy);
  float2 coord = frac(IN.TexCoord0.z) / 2.0f;
  coord.y = IN.TexCoord0.w;
  if(textureValue.r == 1.0 && textureValue.g == 0.0){
    coord.y += 0.5f;
  }
  else if(textureValue.b == 1.0){
    coord.x += 0.5f;
  }
  else if(textureValue.r == 1.0 && textureValue.g > 0.0){
    coord += 0.5f;
  }
  OUT.col = tex2D(tex1, coord);
  return OUT;
}
```

Das Ergebnis sieht dann so aus:



Hier erkennt man nun noch deutlicher als beim obigen Shader die Basen selber und auch die Abgrenzungen zwischen ihnen. Kleiner Nachteil: dieser Shader wirkt erst bei näherer Betrachtung gut, für weitere Entfernungen ist die einfache

Farbdarstellung vorzuziehen.

Der Codon Shader:
der Shader hat die Aufgabe bestimmte Pixel der normalen Textur durch ein Bild zu ersetzen. Doch diesmal werden keine einzelne Basen betrachtet, sondern diese werden nach einer Codon- Tabelle ersetzt. Dabei ergeben immer drei einzelne Basen eine Aminosäure, was für den genetischen Code interessanter ist als einzelne Basen.

| T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | C | C | C | C | C | C | C | C | C | C | C | C | C | C | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T | T | T | T | C | C | C | C | A | A | A | A | G | G | G | G | T | T | T | T | C | C | C | C | A | A | A | A | G | G | G | G |
| T | C | A | G | T | C | A | G | T | C | A | G | T | C | A | G | T | C | A | G | T | C | A | G | T | C | A | G | T | C | A | G |

*Ausschnitt aus der Codontabelle für bakterielle Genome*

Theoretisch gibt es 4*4*4 mögliche Trippel(und damit Aminosäuren), in der Realität jedoch gibt es nur ungefähr 20 verschiedene.
Für den Shader wird zunächst eine spezielle 1- farbige Codontextur mit 24 unterschiedlichen Farben erzeugt. Im Fragment- Programm wird dann anhand der Farbindizes der entsprechende Ausschnitt aus dem Codonbild genommen.

*Codonbild*

Das Fragmentprogramm sieht folgendermaßen aus:

```
fragout FPmain(vertout IN, uniform sampler2D tex0, uniform sampler2D tex1)
{
  fragout OUT;
  //Get Codon Index from texture
  float textureValue = tex2D(tex0, IN.TexCoord0.xy).r;
  //Scale coord.x to Range [0...1/24]
  float2 coord = frac(IN.TexCoord0.z) /  (24.0f);
  //Scale coord.y to Range [0...1]
  coord.y = IN.TexCoord0.w;
  //Add offset
  coord.x += (textureValue.r * (255.0f / 24.0f));
  //Get right Texture Value
  OUT.col = tex2D(tex1, coord);
  return OUT;
}
```

Das Ergebnis sieht sehr ansprechend aus:

Mit Hilfe dieses Shaders ist es nun möglich ein Genom sehr schnell bezüglich der Aminosäuren zu untersuchen.

## Schlußwort

Dieses SEP hat uns sehr herausgefordert und wir lernten viel im Umgang mit der Motif- und X-Programmierung. Vielen Dank an die Betreuer Dr. Peter Kipfer und Dr. Harald Meier der Lehrstühle „Computergrafik und Visualisierung“ und Bioinformatik für die hilfreiche Unterstützung.