DEVELOPER DOCUMENTATION

PeerLo

Version 1.0
21. Dezember 2006

ein Projekt von
Andreas Augustin, Christian Becker

http://peerlo.sourceforge.net

Inhaltsverzeichnis

1.0 Einleitung 3
  1.1 Was ist PeerLo? 3
  1.2 Warum wurde PeerLo erstellt? 3
  1.3 Aus welchen Komponenten besteht PeerLo? 4
  1.4 Software Lizenz 4
    1.4.1 Weitere Information zur Entwicklung 5

2.0 Anforderungen 5
  2.1 Hardwareanforderungen 5
  2.2 Softwareanforderungen 5

3.0 Installation 6

4.0 Kommentierung von Quellcode 7

5.0 Struktur des Plugins 7
  5.2 Pakete 7
    5.2.1 org.gudy.azureus2.peerlo 7
    5.2.2 com.maxmind.geoip 8
  5.3 Grafiken & Bilder 8
    5.3.1 Kartenbeispiele 8
    5.3.2 Maskenbeispiele 10
  5.4 Die plugin.properties Datei 11
  5.5 Die Datenbank 11
  5.6 Integrierte JAR-Archive 11

6.0 Klassenbeschreibungen 12
  6.1 AzureusPlug 12
  6.2 CityBarDataset 12
  6.3 CountryBarDataset 12
  6.4 CountryBarRatioDataset 13
  6.5 CountryBarRatioLeecherDataset 13
  6.6 CountryPieDataset 13
  6.7 EarthViewListener 13
  6.8 StatisticsListener 14
  6.9 IpLookup 14
  6.10 MapsStatsTable 14
  6.11 MyUIManagerListener & MapUIManagerListener 14
  6.12 PeerLo 14
  6.11 PeerloPeer 15
  6.12 PeerloPoint 15
  6.13 Statistics 15
  6.14 MapView 16
  6.15 StatisticView 16
  6.16 ZoomedMap 16
  6.17 LookupService 16

7.0 Kontakt 17

8.0 Ihre Notizen 18

1.0 Einleitung

1.1 Was ist PeerLo?

PeerLo ist ein Plugin für den weit verbreiteten BitTorrent-Client „Azureus“. PeerLo visualisiert den Schwarm auf einer Weltkarte damit derjenige, der einen Torrent herunterläd, eine Vorstellung davon bekommt, wo die anderen Personen sitzen, welche sich auch für diese Datei interessieren.
PeerLo unterscheidet dabei hauptsächlich zwischen „Seedern“ und „Leechern“. „Seeder“ sind die Personen, die die Datei bereits komplett auf ihrem Rechner haben und diese den anderen Peers (allgemein für eine Person im BitTorrent-Netzwerk) zur Verfügung stellt.
Desweiteren bietet PeerLo die Möglichkeit, Statistiken bezüglich der ländlichen Verteilung des Schwarms (die Gesamtheit der Peers) einzusehen.
Auch das Download-/Uploadverhalten der vereinzelten Länder wird in einer seperaten Tabelle angezeigt.
Durch diese Features ist es möglich eine spezifische Analyse des Filesharing-Verhaltens der einzelnen Länder zu erstellen.

1.2 Warum wurde PeerLo erstellt?

PeerLo wurde im Rahmen des Kurses „Medienkonzeption- und Produktion“ an der Fachhochschule Kaiserslautern Standort Zweibrücken erstellt. Der betreuende Professor war Prof. Hendrik Speck. Die Aufgabe bestand in der Visualisierung eines File-Sharing Netzwerks. Da wir bereits Kontakt mit der Programmiersprache Java hatten, entschlossen wir uns ein Plugin für den in Java geschriebenen BitTorrent-Client „Azureus“ zu entwickeln und somit das BitTorrent-Netzwerk zu visualisieren. Da es in diesem Bereich noch sehr wenige und desweitern zu undetailierte Plugins gab, entschlossen wir uns die ländliche Verteilung des Netzwerks zu untersuchen und zu visualisieren.

Fachhochschule Kaiserslautern
University of Applied Sciences

Fachhochschule Kaiserslautern
Morlauterer Straße 31
Telefon: +49 (0)631 / 37 24 - 0
Telefax: +49 (0)631 / 37 24 - 105
E-Mail: presse (at) fh-kl.de
Internet: www.fh-kl.de

1.3 Aus welchen Komponenten besteht PeerLo?

PeerLo besteht aus zwei Hauptkomponenten:

- Der Datenbankkomponente, GeoLite City, welche von Maxmind frei zur Verfügung gestellt wird, und einer entsprechenden Java API welche es ermöglicht, das Land, die Stadt, Längen- & Breitengrad einer zugehörigen IP-Adresse zu bestimmen. Diese Datenbank ist im DAT-Dateiformat im Plugin-Ordner zu finden.
  Wollte man weiter Informationen wie z.B. die Bandbreite, den ISP-Provider und die Organisation erfahren ist dafür eine weitere Datenbank notwendig, welche für ein gewisses Entgelt bei Maxmind erstanden werden kann. Die Internetpräsens ist unter http://www.maxmind.com zu finden.
- Dem Plugin PeerLo selbst, ein JAR-Archiv, welches alle Klassen und weitere JAR-Archive beinhaltet.

1.4 Software Lizenz

PeerLo ist ein Programm, das bei SourceForge.net, dem größten OpenSource Anbieter im Internet, zum freien Download zur Verfügung gestellt wird. Das Programm kann nicht nur einfach benutzt werden, sondern der Programmcode ist durch die OpenSource Lizenz auch für jeden zugänglich, kann erweitert und ins eigene OpenSource-Programm eingebaut werden.
PeerLo wird unter der OpenSource-Lizenz GNU General Public License (GPL) entwickelt. Mehr Informationen hierzu unter http://www.opensource.org. Die Lizenz ist auch im Programmpaket von PeerLo enthalten.

1.4.1 Weitere Information zur Entwicklung

Wenn Sie diese Software in Ihre Software integrieren, dann ergeben sich hieraus gewisse Rechte und Pflichten! Für genauere Informationen werfen Sie bitte einen Blick in die Lizenzbestimmungen unter:

http://www.gnu.org/licenses/gpl.txt
http://www.gnu.de/gpl-ger.html (deutsche Übersetzung)

Desweiteren müssen Sie in Ihrem Produkt bekanntgeben, dass Sie die Geolite City Datenbank von Maxmind verwenden. Für genauere Informationen werfen Sie auch hier einen Blick in die Lizenzbestimmungen unter:

http://www.maxmind.com

2.0 Anforderungen

2.1 Hardwareanforderungen

Die Hardwareanforderungen entsprechen denen von Azureus:

- Pentium oder ähnliches mit mindestens 1GHz
- 256 MB RAM (empfehlenswert 512MB, da Azureus mit der Zeit sehr viel RAM und Ressourcen verbraucht)
- DSL-Internetzugang (damit Azureus genügend Verbindungen aufbauen kann)
- Port-Freigabe (empfehlenswert TCP/UDP auf 49152)
- Eine ausreichend schnelle Grafikkarte mit einer Mindestauflösung von 1024x768 Pixel

2.2 Softwareanforderungen

Damit das Plugin genutzt und auch weiterentwickelt werden kann, wird folgende Software benötigt:

- Betriebssystem: Windows, Linux, MacOS oder ähnliches
- Java Software Developer Kit (Java SDK), davon Java Runtime Environment (JRE) am besten in der Version 1.5, da diese von Azureus empfohlen wird
- Azureus BitTorrent-Client ab der Version 2.2, die derzeit aktuelle Version ist 2.5 und ist unter http://azureus.sourceforge.net erhältlich.

- Ein ZIP-Programm, mit dem das Plugin-Archiv entpackt werden kann.
- Eine Entwicklungsumgebung, am besten Eclipse, aktuelle Version 3.2 erhältlich unter www.eclipse.org

3.0 Installation

Zur Installation gibt es zwei Möglichkeiten:

1.Möglichkeit (Installation der letzten stabilen Version):

1. Laden Sie die PeerLo-Plugin Quellcode-Dateien von https://sourceforge.net/project/showfiles.php?group_id=179936 herunter. Sollte der Link nicht mehr zur Verfügung stehen, besuchen Sie die Projekt-Homepage: http://peerlo.sourceforge.net um weitere Informationen zu erhalten.
2. Entpacken Sie das ZIP-Archiv in einen neuen Ordner und importieren Sie diesen als neues JAVA-Projekt
3. Damit die importierten Klassen gefunden werden können benötigen Sie noch weitere JAR-Archive:
   1) SWT (swt.jar):
      Im Installationspfad von Eclipse oder im Internet z.B. im Azureus Plugin Developement Wiki zu finden
   2) JFreeChart (jfreechart-1-x-x.jar):
      Erhältlich unter Sourceforge.net, Projekt JFreeChart
   3) JCommon (jcommon-1-x-x.jar):
      Ist bei JFreeChart enthalten
4. Registrieren Sie diese im CLASSPATH des Projekts

2.Möglichkeit (Installation der aktuellsten Version):

1. Benutzen Sie einen CVS-Client, wie z.B. den der in Eclipse integriert ist und erstellen Sie ein neues Reposistory mit folgenden Kennungsdaten:

   Connection Type: extssh
   User: anonymous
   Password: anonymous
   Host: peerlo.cvs.sourceforge.net
   Repository Path: /cvsroot/peerlo

2. Checken Sie dieses Repository nun als neues Projekt aus
3. Unter den Dateien befindet sich im Root-Verzeichnis 3 Jar-Archive, die Sie im CLASSPATH des Projekts registrieren müssen

Bemerkung: Um neue Dateien ins Repository von PeerLo zu committen, müssen Sie einer der Projektadministratoren sein. Ansonsten wenden Sie sich an einen, und schicken Sie Ihm die Quelldateien oder Fragen Sie ihn, ob Sie auch Administrator werden können.

4.0 Kommentierung von Quellcode

Bitte legen Sie viel Wert auf die Kommentierung ihres Quellcodes und die eindeutige Benennung von Variablen und Methoden, damit auch andere Entwickler ihren Code verstehen und weiterentwickeln können. Auch wir haben sehr viel Wert auf einen gut kommentierten Quelltext gelegt, um Ihnen den Einstieg zu erleichtern. Eine weitere Hilfe sind Dokumentationen mit denen es Softwareentwicklern neben dem kommentierten Quelltext wesentlich einfach fällt das Programm zu verstehen.

5.0 Struktur des Plugins

5.2 Pakete

5.2.1 org.gudy.azureus2.peerlo

In diesem Paket finden Sie alle Klassen, die die visuelle Komponente und den Algorithmus darstellen. Es sind derzeit folgende Klassen vorhanden:

- org.gudy.azureus2.peerlo.AzureusPlug.java
- org.gudy.azureus2.peerlo.CityBarDataset.java
- org.gudy.azureus2.peerlo.CountryBarDataset.java
- org.gudy.azureus2.peerlo.CountryBarRatioDataset.java
- org.gudy.azureus2.peerlo.CountryBarRatioLeecherDataset.java
- org.gudy.azureus2.peerlo.CountryPieDataset.java
- org.gudy.azureus2.peerlo.EarthViewListener.java
- org.gudy.azureus2.peerlo.Geo.java
- org.gudy.azureus2.peerlo.Graphics.java
- org.gudy.azureus2.peerlo.HookUtils.java
- org.gudy.azureus2.peerlo.IpLookup.java
- org.gudy.azureus2.peerlo.MapStatsTable.java
- org.gudy.azureus2.peerlo.MapUIManagerListener.java
- org.gudy.azureus2.peerlo.MapView.java
- org.gudy.azureus2.peerlo.MyUIManagerListener.java
- org.gudy.azureus2.peerlo.PeerLo.java
- org.gudy.azureus2.peerlo.PeerloPeer.java
- org.gudy.azureus2.peerlo.PeerloPoint.java
- org.gudy.azureus2.peerlo.Statistics.java
- org.gudy.azureus2.peerlo.StatisticsListener.java
- org.gudy.azureus2.peerlo.StatisticsView.java
- org.gudy.azureus2.peerlo.ZoomedMap.java

Eine detailliertere Klassenbeschreibung finden Sie im Kapitel 6.

5.2.2 com.maxmind.geoip

In diesem Paket finden Sie die komplette Java-API, die von Maxmind zum Ansprechen Ihrer Datenbank gestellt wird. Sollten Sie fragen zur Benutzung der API haben, werfen Sie einen Blick auf die zahlreichen Beispiele, die man unter www.maxmind.com neben den Datenbanken herunterladen kann. Es sind derzeit folgende Klassen in dem Paket enthalten:

- com.maxmind.geoip.Country.java
- com.maxmind.geoip.DatabaseInfo.java
- com.maxmind.geoip.Location.java
- com.maxmind.geoip.LookupService.java
- com.maxmind.geoip.Region.java

Eine detailliertere Klassenbeschreibung finden Sie im Kapitel 6.

5.3 Grafiken & Bilder

Im Paket org.gudy.azureus2.peerlo sind weitere Ordner mit folgendem Inhalt vorhanden:

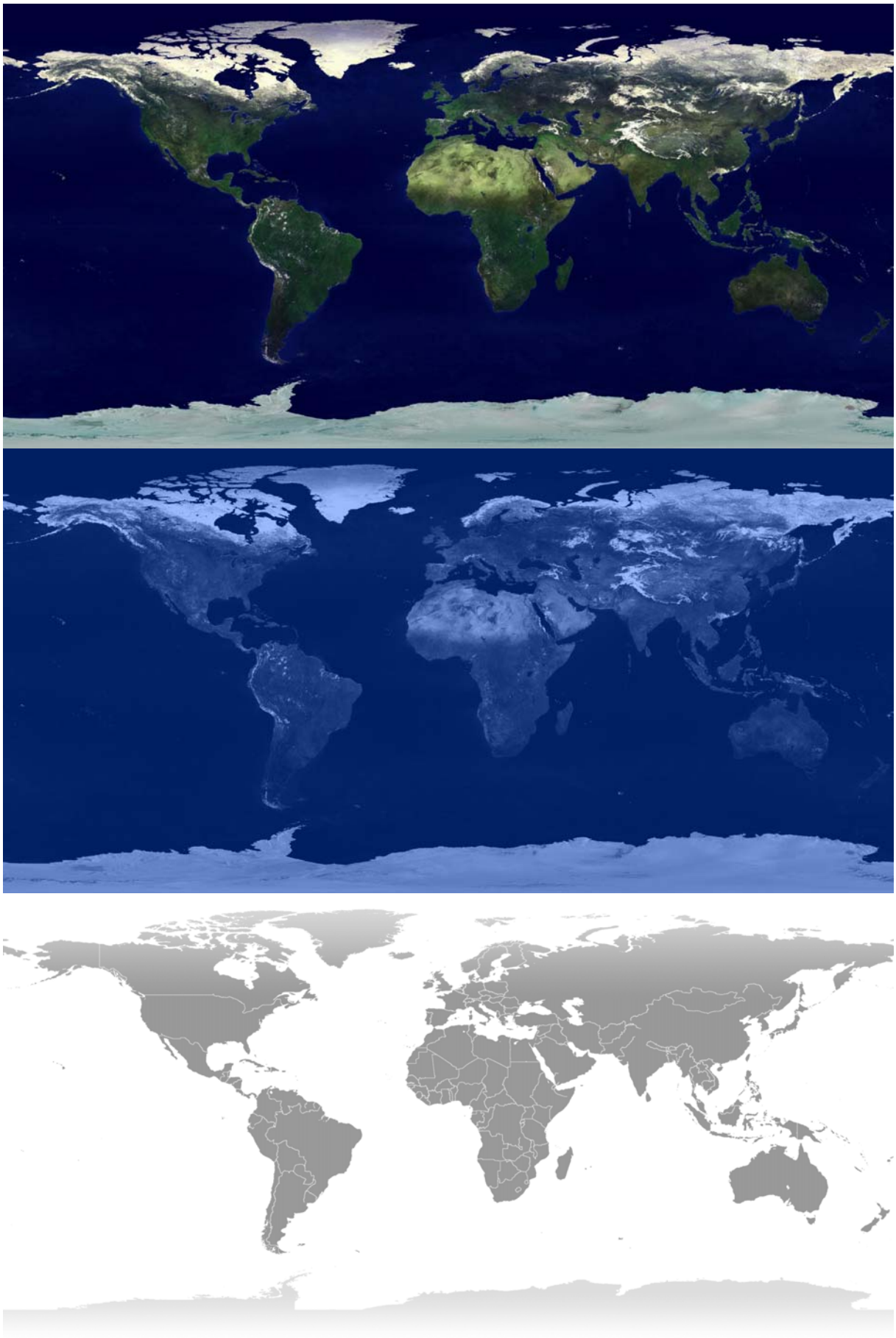
- flags: alle Flaggenbilder im PNG-Format. Diese werden für die Tabelle in der PeerLo-Worldmap-View benötigt, um sie den Herkunftsländern zuzuordnen.

- pictures: unser Logo in verschiedenen Formaten und Größen. Diese werden für die Standardanzeige in der PeerLo-Statistics-View verwendet.

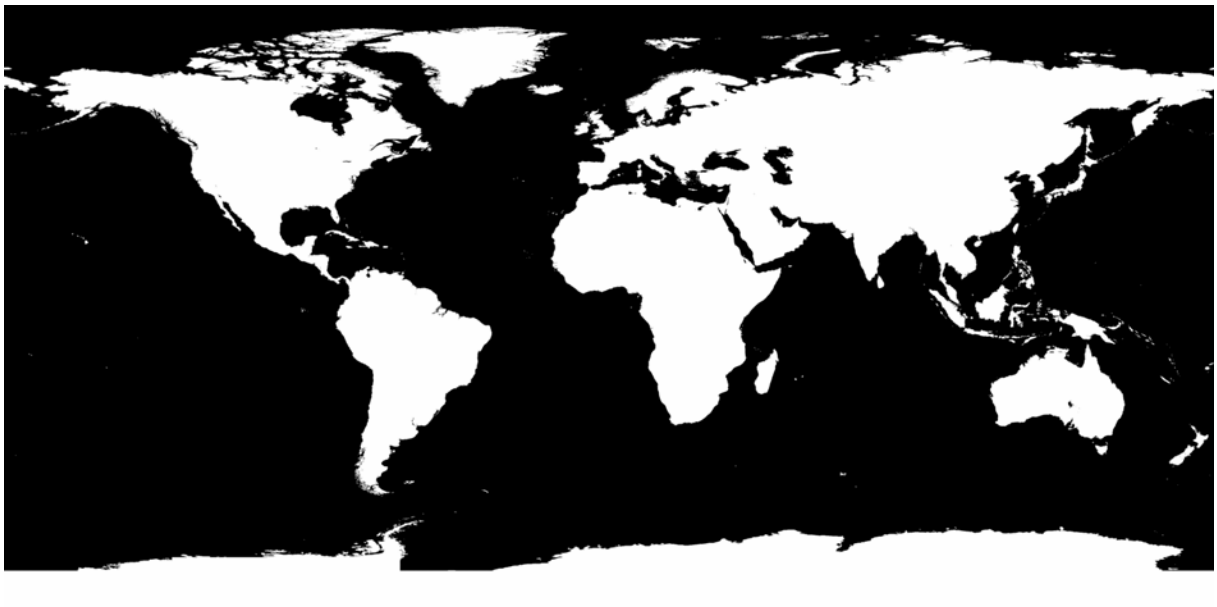
- maps: hier ist zurzeit die bearbeitete earthmap, welche man von der NASA bekommt in der Größe 2048x1024 enthalten. Man kann jederzeit eine andere Karte verwenden solange diese im Größenverhältnis gleich ist, da nur mit diesem Verhältnis eine Umrechnung von Längen- und Breitengrad auf Pixelkoordinaten möglich ist. Es steht Ihnen auch frei unsere Karte nach Wunsch zu bearbeiten.

5.3.1 Kartenbeispiele

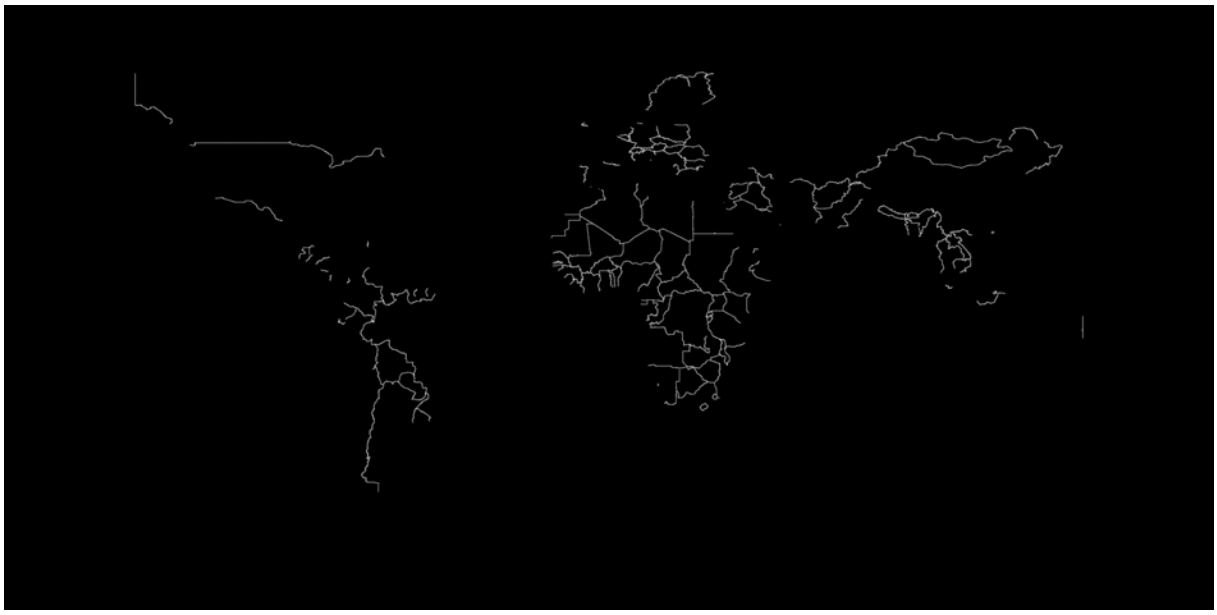
Nutzen Sie alle Möglichkeiten, die Ihnen z.B. „Gimp“ bietet. Von der NASA erhalten Sie nicht nur die Karte sondern auch Masken, mit denen Sie Flüsse, Kontinent- und Ländergrenzen auf die verschiedenen Karten projiezieren können.



5.3.2 Maskenbeispiele



Eine Landmaske



Eine Ländergrenzenmaske

Mit einem geeigneten Bildbearbeitungsprogramm ist es dann möglich, die Masken aufeinander zu legen, da diese das gleiche Größenverhältnis besitzen.

5.4 Die plugin.properties Datei

Neben den Paketen „org.gudy.azureus2.peerlo“ und „com.maxmind.geoip“ finden sie eine einzelne Datei mit dem Namen „plugin.properties“. Diese Datei beinhaltet die Informationen die Azureus braucht, um das Plugin zu identifizieren und in das Programm einzubinden. In dieser Datei wird unter anderem die Sprachdatei angegeben, falls Sie das Plugin in mehrere Sprachen veröffentlichen wollen. Der Haupteintrag jedoch ist die Angabe der der Hauptklasse mit der das Plugin gestartet wird. In unserem Fall ist dies die Klasse „AzureusPlug“. Desweiteren wird hier die aktuelle Version, sowie der Plugin-Name und dessen ID festgelegt.

5.5 Die Datenbank

Die Datenbank wird von der Firma Maxmind unter der GPL zur Verfügung gestellt. Man erhält diese unter www.maxmind.com.
Wir verwenden die Geolite City Datenbank, diese ist kostenlos. Man muss lediglich bekanntgeben, dass die Datenbank von Maxmind stammt.
Wollen Sie weitere Informationen wie z.B. den Provider, die Bandbreite und die Organisation der vereinzelten IP-Adressen erfahren, müssen Sie weitere Datenbanken für ein gewisses Entgelt einbinden. Auch diese bekommen Sie unter www.maxmind.com.
Die freie Datenbank Geo City Lite, welche wir verwenden können Sie in verschieden Formaten bekommen. Sie haben die Möglichkeit diese als Datenbankskript selbst zu erstelln, wovon wir Ihnen aber abraten, solange Sie keine zusätzliche Datenbank verwenden. Eine weitere Möglichkeit ist, die Datenbank als DAT-Format zu bekommen. Diese wird dann über eine Java-API angesprochen. Wir haben uns für Letzteres entschieden, da diese Methode laut Maxmind die effizientere ist. Die Datenbankdatei muss bei der Installation neben der „peerlo.jar“ im Pluginordner enthalten sein.

5.6 Integrierte JAR-Archive

Damit PeerLo funktioniert mussten verschiedene JAR-Archive integriert werden. Das Archiv „swt.jar“ wird bereits von Azureus bereitgestellt. Es muss also noch „jfreechart-1-x-x.jar“ und „jcommon-1-x-x.jar“ eingebunden werden. JFreeChart ist eine OpenSource API welches es ermöglicht mit relativ leichten Mitteln Diagramme der verschiedensten Formen zu erstellen. Eine äußerliche Anpassung dieser ist auch sehr einfach. Um ein Diagramm zu erstellen muss zunächt ein Dataset mit den darzustellenden Daten erstellt werden. Anschließend wird daraus ein Chart Objekt erstellt welches mittels eines Renderers auf verschiedene Kanäle ausgegeben werden kann. Ausgabemöglichkeiten sind z.B. als Frame, integriert in eine Swing oder AWT Anwendung, als experimentelles SWT oder als Bilder.

Wir mussten die letzte Möglichkeit wählen, da das experimentelle SWT mit der aktuellen SWT Version, welche Azureus verwendet nicht harmonierte. So werden nun die Charts als Bilder gerendert und als PNG Dateien im PeerLo Ordner ausgegeben. Wird das Plugin geschlossen, werden alle Bilder automatisch gelöscht. Man könnte dies also als eine temporäre Ausgabe betrachten, um die Darstellung in einem integriertem Reiter (PeeLo-Statistics) zu ermöglichen. Das „jcommon-1-x-x.jar“ wird von JFreeChart benötigt und muss somit auch eingegliedert werden.

6.0 Klassenbeschreibungen

Es folgen nun kurze Klassenbeschreibungen, um Ihnen den Einstieg zu erleichtern. Es wird allerdings nicht zu detailliert auf die verschiedenen Klassen eingegangen, um die Übersicht zu wahren. Wollen Sie mehr über bestimmte Klassen erfahren, öffnen Sie diese und werfen Sie einen Blick auf die Kommentierung. Wir werfen auch nur einen Blick auf die wichtigsten Klassen.

6.1 AzureusPlug

Diese Klasse wird als erstes von Azureus aufgerufen, da diese die *initialize()*-Methode enthält und vom Interface *Plugin* abgeleitet ist. Dies sind die Vorraussetzungen die Azureus benötigt um ein Plugin zu erkennen. Diese Klasse muss natürlich auch in der *plugin.properties* eingetragen werden.

6.2 CityBarDataset

Diese Klasse erstellt ein oder mehrere *Dataset(s)*, welche JFreeChart benötigt um ein oder mehrere *Chart* Objekte zu erstellen. Die Charts werden in der Klasse *Statistics* erstellt. Bei diesem *Dataset* handelt es sich um das *Dataset*, welches die vereinzelten Länder und deren interne Verteilung untersucht. Es werden zwei Methoden zur Verfügung gestellt.
Bei der Ersten werden alle *Datasets* für alle Länder auf einmal erstellt. Bei der Zweiten ist es möglich nur gezielte *Datasets* zu erstellen, um somit die Ressourcen zu schonen. Es wird wie immer eine *Collection* aus *PeerloPeer´*s als Übergabeparameter erwartet.

6.3 CountryBarDataset

Diese Klasse dient als Objektschablone für ein Country Objekt, das für die Erstellung dieses *Datasets* benötigt wird. Die

wichtigste Methode allerdings ist *creatCountryBarDataset()*. Welche als Übergabeparameter eine *Collection* von *PeerloPeer´*s benötigt um daraus ein *Dataset* zu erstellt. Anschließend wird dieses zurückgegeben. Dieses *Dataset* wird dann in *Statistics* (von wo auch die Methode aufgerufen wird) zu einem *Chart* umgebaut und als Datei ausgegeben. Bei diesem Balkendiagramm handelt es sich um die globale Verteilung von Peers, Seedern und Leechern.

6.4 CountryBarRatioDataset

In dieser Klasse wird ähnlich wie bei *CountryBarDataset* ein *BarDataset* Object erstellt, welches dann in der Klasse *Statistics* als *Chart* ausgegeben wird. Die Klasse dient auch als Objektschablone für die Country Objekte, welche für das *Dataset* benötigt werden. Bei diesem Balkendiagramm handelt es sich um den globalen Anteil von Seedern des jeweiligen Landes.

6.5 CountryBarRatioLeecherDataset

Es handelt sich hierbei um eine analoge Klasse zu *CountryBarRatioDataset*. Allerdings wird hier der globale Anteil von Leechern betrachtet und auf die einzelnen Länder verteilt. Der Quelltext ist bis auf 1-2 Zeilen identisch mit dem von *CountryBarRatioDataset*.

6.6 CountryPieDataset

In dieser Klasse wird ein *PieDataset* Objekt erstellt, um die globale Verteilung von Peers zu veranschaulichen. Die Methode *createCountryPieDataset()* spielt hier die hauptsächliche Rolle. Es wird ein *Collection* aus *PeerloPeer´*s erwartet und daraus wird das *PieDataset* erstellt und an die Klasse Statistics (von der aus die Methode aufgerufen werden sollte) zurückgegeben. Das Ergebnis ist ein Tortendiagramm mit dem Namen „Peers worldwide“, das man in der Statistics-View begutachten kann.

6.7 EarthViewListener

Hier handelt es sich um eine Klasse, die vom Interface *UISWTViewEventListener* abgeleitet ist und dessen Methoden implementiert. Bei dieser Klasse handelt es sich um einen *Listener* der auf der Klasse *MapView* horcht und dessen Events abfängt. Typische Events sind TYPE_CREATE, TYPE_INITIALIZE, TYPE_REFRESH und TYPE_DESTROY. Diese beziehen sich auf eine View, bei diesem Beispiel auf MapView (also den Reiter „PeerLo“).

6.8 StatisticsListener

Äquivalente Klasse zu EarthViewListener. Bei dieser Klasse handelt es sich um einen Listener der auf die Klasse *StatisticsView* horcht und dessen Events abfängt. Beim Event TYPE_DESTROY werden zusätzlich die erstellten Statistiken gelöscht.

6.9 IpLookup

Diese Klasse handhabt die Umsetzung von IP-Adressen-Strings zu einer *Collection* aus *IpLookup*-Objekten, welche die benötigten Attribute wie z.B. Längengrad, Breitengrad, Stadt, Land usw. enthält. Dafür sind zwei statische Methoden implementiert worden: *lookupIp()* und *lookupIps()*.
Diese gibt es in zwei Varianten, einmal mit Übergabe des *User-Interfaces* und einmal ohne. Wenn Sie die Methoden ohne den Parameter *User-Interface* nutzen wollen, müssen Sie einmal die Methode *getDatabaseLocation()* ausführen, damit das Plugin weiß, wo sich die Datenbank befindet und sie somit ansprechen kann. Zusätzlich sind natürlich alle *getter*-Methoden implementiert.

6.10 MapsStatsTable

Diese Klasse wird benötigt um die Tabelle unter der Karte zu erstellen. Sie beinhaltet nur eine statische Methode, die wieder eine *Collection* von *PeerloPoint*`s benötigt um die Tabelle zu füllen: *createMapStatsTable()*. Die Ausgabe dieser Tabelle wird auch in dieser Klasse erledigt. Deswegen muss die *MapView* der Methode auch übergeben werden.

6.11 MyUIManagerListener & MapUIManagerListener

Diese Klassen regeln die Erstellung der zwei *Views* und das Zuteilen von *Listeners* auf diese.

6.12 PeerLo

Diese Klasse intialisiert das Plugin. Sie enthält die intialise()-Methode, die den Anschlag für alle anderen Klassen darstellt.

6.11 PeerloPeer

Diese Klasse ist von *IpLookup* abgeleitet und erbt somit alle Attribute und Methoden. Desweiteren enthält sie die zusätzlichen Attribute *client*, *percentDoneInThousandNotation*, *port*, *isSeed*, *date*, *alive* und *pStats*. Ein PeerloPeer Objekt beinhaltet also alle Daten, die für die Statistiken und für die Anzeige notwendig sind. Eine ausschlaggebende Methode hierbei ist *getPeerloPeers()* welche eine *Collection* von PeerloPeer´s zurück gibt. Diese *Collection* wird überall im Programm verwendet. Eine weitere Methode ist *updatePeerloPeers()*, dieser Methode wird die aktuelle *Collection* übergeben und die neuen *PeerloPeer*-Objekte werden hinzugefügt. Diese Methode ist wichtig, um sich Peers, mit denen man nicht mehr verbunden ist zu merken.

6.12 PeerloPoint

Die Klasse *PeerloPoint* ähnelt sehr *PeerloPeer*. Allerdings beinhaltet ein *PeerloPoint*-Objekt alle Attribute, die für die Darstellung eines Punktes auf der Karte notwendig sind. Man könnte ein Objekt praktisch gesehen als Container von *PeerloPeer*´s sehen. Die Klasse beinhaltet zwei sehr wichtige Methoden. Erstens *getPeerloPoints()*: diese liefert eine *Collection* von *PeerloPoint*-Objekten zurück. Diese Objekte besitzen alle Attribute, die für das zeichnen der Punkte nötig sind (Größe, Farbe, Längengrad, Breitengrad, Stadt, Land uvm…).Die zweitens Methode ist *orderPeerloPoints()*, welche die *Collection*, die man *getPeerloPoints()* erhalten hat sortiert, damit kleinere Punkte nicht von größeren überdeckt werden. Diese Methode liefert dann Array zurück, welches das ständige Durchlaufen wesentlich erleichtert.

6.13 Statistics

Diese Klasse ist für die Erstellung der verschiedenen Diagramme zuständig und beinhaltet somit nur statische Methoden. In diesen Methoden werden die *Datasets* erzeugt und anschließend daraus ein *Chart*-Objekt erstellt, welches der *Renderer* dann in eine Grafik-Datei schreibt. Zusätzlich beinhaltet jede Methode noch Code, der die Darstellung der einzelnen Diagramme anpasst (Farbe, Abstand…). Eine weitere wichtige statische Methode ist *setPeerloLocation()* welches das statische Attribut dieser Klasse setzt, damit diese weiss, auf welchem Pfad PeerLo installiert ist und wohin die Grafiken abgespeichert werden können. Die Methode sollte vor der Erstellung von Diagrammen ausgeführt werden.

6.14 MapView

Die Klasse regelt die Anzeige des Reiters „PeerLo“, baut diesen Reiter also in der *initialize()*-Methode zusammen. Die Methode *refresh()* regelt die Programmaufrufe, die bei der Aktualisierung ausgeführt werden sollen, wie z.B. das erneute Zeichnen der Karte und der Tabelle.

6.15 StatisticView

Diese Klasse regelt die Anzeige des Reiters „PeerLo-Statistics“. Sie funktioniert analog zur Klasse MapView und benötigt keine weitere Betrachtung. Hier wird anstatt eine Tabelle und einer Karte nur ein Drop-Down-Menü und ein Canvas für die Diagrammanzeige initialisiert. Die *refresh()*-Methode wird mittels eines Counters nicht immer ausgeführt, um unnötigen Ressourcenverbrauch zu verhindern. Die Klasse regelt auch wann welche Diagramme neu erstellt werden sollen und wann welche angezeigt werden.

6.16 ZoomedMap

In dieser Klasse wird die Karte erstellt und die Punkte darauf gezeichnet. Auch die Umrechnung von Längen- und Breitengrad auf Pixelkoordinaten geschieht hier. Auch das Handhaben des Zooms und die Anzeige der Scrollbars wird hier geregelt.

6.17 LookupService

Diese Klasse ist im Paket von Maxmind enthalten und bildet die Schnittstelle zur Datenbank. Für nähere Informationen laden Sie sich die Beispiele unter www.maxmind.com herunter.

7.0 Kontakt

Die Entwickler von PeerLo sind:

Andreas Augustin
Email: anau0001@users.sourceforge.net
Web: http://www.thingo.de

Christian Becker
Email: schenes@users.sourceforge.net
Web: http://www.chris-becker.de

Supervisor:

Prof. Hendrik Speck
Kurs: Medienkonzeption und Produktion im Wintersemester 2006/2007
Web: http://www.egs.edu/faculty/speck.html

Fachhochschule Kaiserslautern
Standort Zweibrücken
Amerikastr. 1
66482 Zweibrücken
Web: http://www.fh-kl.de

Fachhochschule
Kaiserslautern
University of
Applied Sciences

8.0 Ihre Notizen