平成 20 年 1 月 31 日

卒業研究論文

ユーザ参加型地図アプリケーションの開発

大阪工業大学情報科学部

ヒューマンインタフェース研究室

B04-093　高橋　博史

目次

１．はじめに

近年、地理空間情報に関連する技術が大きな注目を浴びており、地図や位置情報、GIS（地理情報システム：Geographic Information System）、GPS に関連した記事が頻繁に報じられている。ネイチャー誌では 2004 年 1 月に Mapping Opportunities という記事を掲載し、ジオテクノロジーはナノテクノロジーとバイオテクノロジーと並んで、21 世紀の３大ビジネス分野の１つになるとされている[1]。同誌は 2005 年 5 月号に Google Earth [2]の地球画像で表紙をかざり、The Web-Wide World という記事の中でグーグル社の Google Earth、マイクロソフトの Virtual Earth[3]、NASA の World Wind[4] など、代表的なデジタルアース製品をレビューし、科学研究や市民参加をはじめとする多くの分野において、Web マップサービスは今後、力が発揮されるという[5][6]。

上記のような記事が登場している背景には、GIS 技術がインターネット、GPS 等の急速に発展している情報技術と結びつくことにより、Web 技術を融合させた GIS である WebGIS といった、より高度なものに変わってきたからだといえる。また、Google Maps API などに挙げられる API を公開した地図サービスの登場によって、カーナビゲーションやインターネット地図サービスといった、地図帳としての使用方法以外にも、既存のサービス（例えば SNS サービス）に地図を組み合わせたサービスが生まれている。これらのサービスの登場によって、ユーザ自身が地図を用いて情報発信を行う機会も増えている。しかしながら、発信される情報の量が膨大な数になってしまっており、１つの情報が他の大量の情報の中に埋まってしまっている。そのような環境の中において、ユーザが欲しい情報を探すときに、その情報数の多さから欲しい情報を的確に取得することができないといった問題点が生じている。例えば、ある飲食店についての情報を検索し、地図上に投稿されている情報をたどって行ったとしてもユーザが求めた飲食店の情報にたどり着かない状況が発生したりする。また、個人によって地図上に投稿された情報がコミュニティ単位で扱われてしまい、コミュニティの消滅とともに、投稿した個人に関係なくコミュニティ内のに投稿された全ての情報が削除されてしまっている。また、コミュニティ内の情報からある個人の投稿した情報を抽出できない。例えば、映画館に関しての情報に定評のある A さんの情報を取り出そうとしても、書き込まれた情報コミュニティ単位で扱われているためにその情報が A さんのものであるか知ることが出来ない。また、ユーザ自身で情報の投稿ができる地図サービスにおいては会員制のサービスであることが多く、情報の投稿や削除が会員に限定されてしまっている。

そこで、本研究では Gooogle Maps API[7]を用いて地図上に投稿された情報からカテゴリや投稿者別で情報を取り出すことが可能なコミュニティ地図サービスを提供するアプリケーションの設計と開発を行った。開発アプリケーションにおいて、ユーザが地図上に情報の投稿や編集は制限なしに自由に行えるようにしている。また、地図上に投稿された情報に対し、特定のカテゴリに対するデータを取り出すための情報（「カテゴリ種別」）と、特定ユーザが投稿したデータを識別するための情報（「ユーザ種別」）の２つを付加した。そ

して付加された 2 つの情報を用いて、地図上に書き込まれている情報から指定された「カテゴリ種別」と「ユーザ種別」に見合った情報のみを抽出してユーザに提示する機能を持たせた。前述の例を用いると、「カテゴリ種別」を付加する目的は、飲食店についての情報を検索し、リンクをたどって行ったとしてもユーザが求めた飲食店の情報にたどり着かない状況が発生することを防ぐためである。また、「ユーザ種別」を付加する目的は、映画館に関しての情報に定評のある A さんの情報を取り出そうとしたときに、書き込まれた情報がコミュニティ単位で扱われており、その情報が本当に A さんのものであるかどうかを知ることができるようにするためである。

第 2 章では GIS 技術の変遷について述べる。次に、現在の地図サービスについて、そしてそれらが抱えている問題点について述べる。第 3 章では本研究で開発するユーザ参加型地図アプリケーション「M・L・M（Multi Layered Map)」の概要と特長について説明する。第 4 章では開発するアプリケーションの設計と開発について説明する。第 5 章では開発したアプリケーションの画面や操作について述べる。第 6 章に考察を述べた後に第 7 章で本研究のまとめを行う。

２．地図情報サービスの現状

本章では、現在までの GIS や地図サービスの変遷について説明する。現在のインターネット地図サービスがどのようなものであるかを述べた後に、抱えている問題点について述べる。

２．１．GIS の変遷

GIS（地理情報システム：Geographic Information System）は、地理的位置を手がかりに、位置に関する情報を持ったデータ（空間データ）を総合的に管理・加工し、視覚的に表示し、高度な分析や迅速な判断を可能にする技術である[7]。GIS 技術は 1960 年代に登場しており、歴史の長い技術である。また、時代の流れに伴って、GIS 技術の利用形態も変化してきている[9]。GIS の登場時はコンピュータマッピング手法の技術開発が中心であった。しかしながら、高機能な汎用 GIS ソフトウェアの登場とワークステーション（WS）やパーソナルコンピュータ（PC）といった身近で使いやすいコンピュータ環境が登場したことで、GIS が我々にとって手軽に使えるものとなった。さらに、GIS 技術の進歩だけではなく、インターネット、GPS、といった情報技術の急速な発展に伴って、さまざまな分野において GIS 技術が応用されている。また、近年においては、ブロードバンド環境を用いた大容量のデータ通信が普及してきている。これに伴って Web 技術を融合させた GIS である WebGIS を用いたサービスの研究開発や運用が行われるようになった。このように、GIS 技術は今なお大きな注目を浴び続けている。

表 1. GIS の利用形態の変化[9]

分類	年代	利用形態
初期のＧＩＳ	1960 年代～	・コンピュータマッピング ・専門家が操作する大型コンピュータシステム上で稼動 ・各分野・実務への応用は限定的
現代のＧＩＳ	1980 年代～	・ＧＩＳベンダーによる汎用ＧＩＳソフトウェアの開発と販売 ・個人単位で操作可能なワークステーション（ＷＳ）やパーソナルコンピュータ（ＰＣ）上で稼動
	1990 年代後半～	・インターネット、リモートセンシング、ＧＰＳ、モバイル通信などの急速な情報技術の進歩により、これらの様々な技術と融合化 ・様々な分野での応用化、実務への普及

２．２．一般的なオンライン地図情報サービス

WebGIS の普及にともなって、インターネット上で地図情報を扱ったサービスが提供されており、国内では MapFan[10]や Mapion[11]などは早くからこのサービスに取り組んでいる。また、2005 年には Google 社が「Google Maps」[12]の提供を開始している。これらの地図サービスは、ユーザが運営会社のサーバ上におかれてある地図データベースに Web ブラウザでアクセスし情報を得ることができる。

また、具体的に提供されているサービスの種類は多岐にわたっており、基本的なサービスとして「地図画像の表示」が挙げられる。ブラウザ上に表示された地図は、マウス操作で地図の表示場所を移動させることや縮尺の切り替えなどを行うことができる。また、キーワードから場所を検索して地図の表示を行ったりなど、各サイトごとにさまざまな機能が提供されている。また、API（Application Program Interface）を公開している地図サービス、例えば Google 社の「Google Maps API」[7]や Yahoo! Maps API[13]、Virtual Earth API[14]といったものが WebGIS をさらに発展させているといえる。

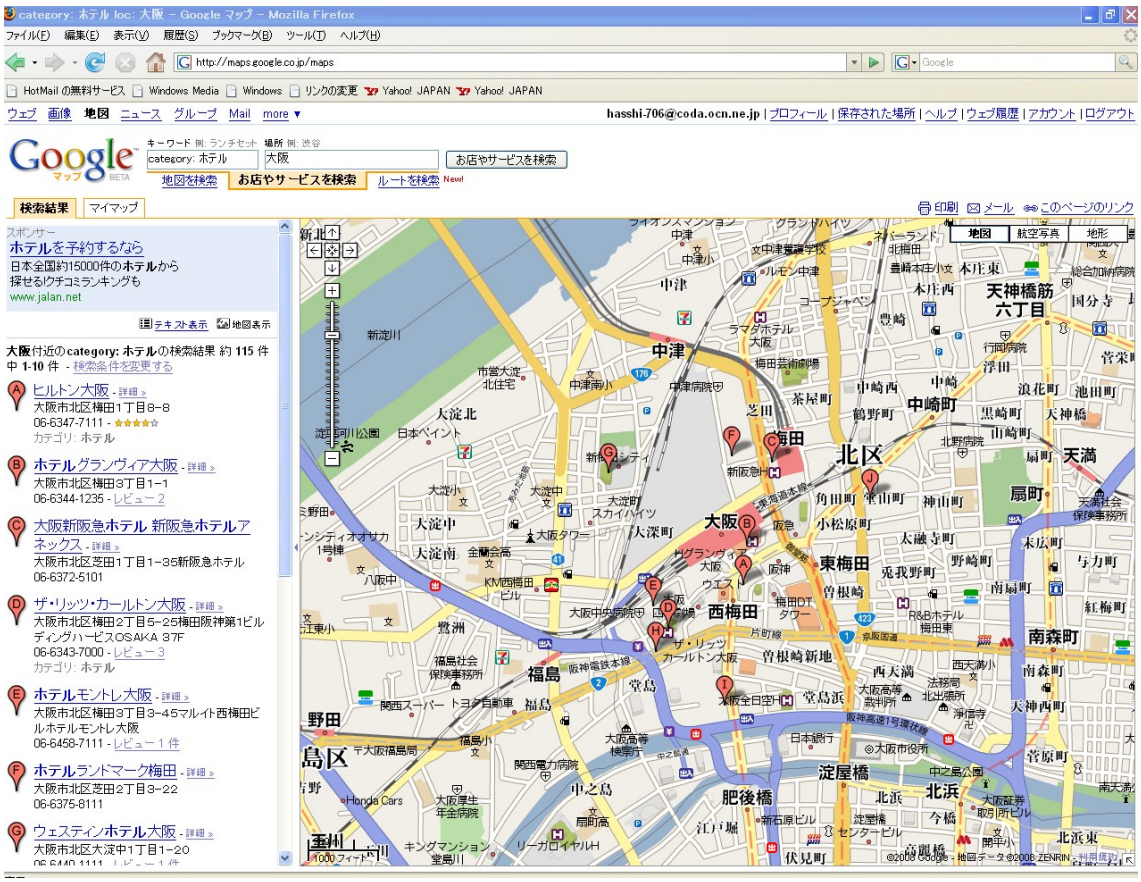

図 1. Google Maps の画面[12]

２．３．WebGIS を用いた新しいサービス

前節で述べた地図情報サービスは、「万人向け」「公共」といった一般的な情報を扱っているのに対し、最近では WebGIS 技術を使った新しい取り組みとして、既存のサービスに地図を組み合わせたものが登場している。その代表例として SNS（Social Network Service）と組み合わせたサービスが挙げられる。人と人とのつながりを促進・サポートする、コミュニティ型の Web サイトで、友人・知人間のコミュニケーションを円滑にする手段や場を提供する。趣味や嗜好、居住地域、出身校、あるいは「友人の友人」といったつながりを通じて新たな人間関係を構築する場を提供する、会員制のサービスのである SNS と地図サービスを結びつけたものが見られる。図 2 に見られるように、SNS で扱われるような「口コミ」「仲間と情報交換の場」といったユーザに密着した情報を地図上に表現しているのである。その例として SNS 地図サービス「ちずる」[15]を示す。

図 2. SNS 地図サービス「ちずる」[15]

他にも、コミュニティとしての機能や、情報の分類を行ったりする機能は存在していないものの、全世界のユーザが 1 枚のインターネット地図上に投稿された情報の共有ができる「Wikimapia」[16]というサービスも新たに登場している。図 3 に Wikimapia の画面を示す。

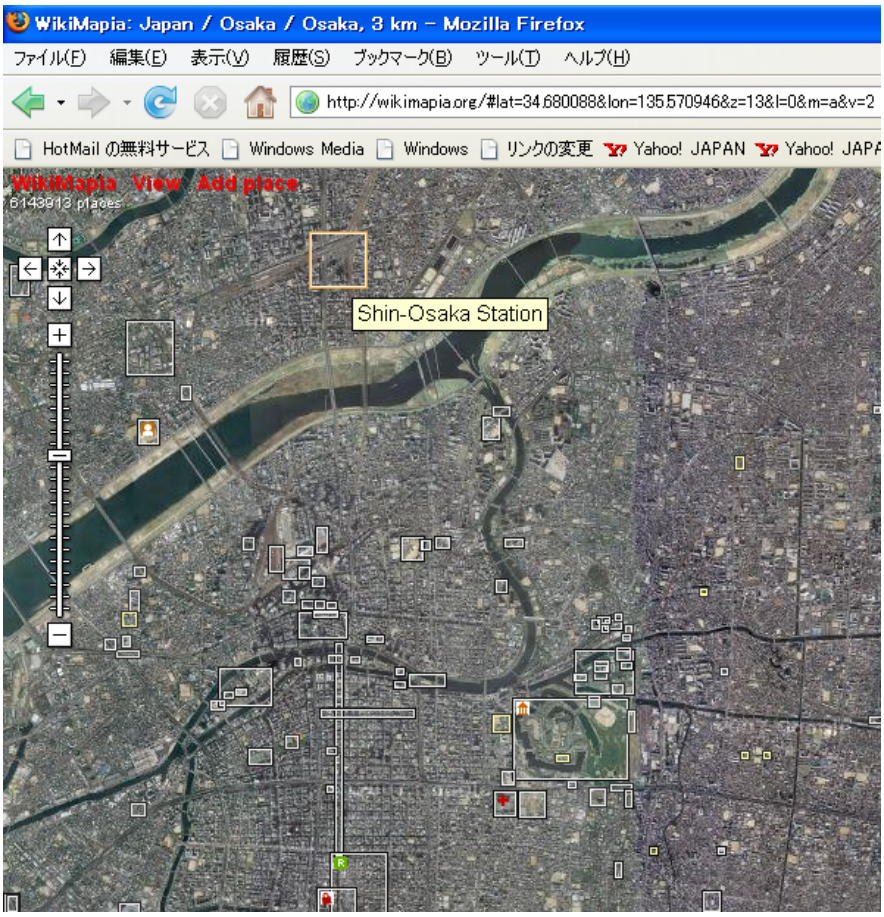

図 3. 地図サービス「Wikimapia」[16]

２．４．地図サービスの問題点

地図サービスは、本来、ローカルで一般的な情報を提供するものであり、住所や道路、公共施設、交通機関などといった情報を扱っていればよいものであった。しかしながら、GIS 技術、IT 技術の発展に伴って手軽に地図情報を扱えるようになると、地図から情報を得るだけではなく、ユーザ自身が地図を用いて情報発信を行うようになった。このように、地図サービスにも双方向性が求められる必要がでてきたのである。しかしながら、既存の SNS に地図サービスを結び付けた新しいサービスで情報を得る際、地図上に投稿された情報が多すぎて、本当に自分が知りたい情報がほかの情報に埋もれてしまったり、情報量の多さから、欲しい情報を的確に得られないという問題点がある。また、投稿された情報がコミュニティ単位で扱われるようになってしまい、誰が投稿した情報であったのか取り出すことが困難になっている。以下に本項で挙げた問題点を箇条書きで示す。

問題点 1：SNS に地図サービスを結びつけたサービスでは、投稿されている情報の数の多さから、欲しい情報を的確に得ることが困難になっている。例えば、ある飲食店についての情報のみを取り出したくても、数多くの飲食店の情報が同時に投稿されていたりすることがある。

問題点２：個人によって投稿された情報がコミュニティ単位で扱われてしまい、誰が投稿した情報であったのか取り出すことが困難になっている。例えば、映画館に関しての情報に定評のある A さんの情報を取り出そうとしても、書き込まれた情報コミュニティ単位で扱われているためにその情報が A さんのものであるか知ることが出来ない。

これらの問題点を解決するために、3 章以降では、本研究で設計・開発を行ったユーザ参加型地図アプリケーションの概要や設計・開発、評価について述べる。

３．ユーザ参加型地図アプリケーション「M・L・M（Multi Layered Map)」の概要

本章ではユーザ参加型地図アプリケーション「M・L・M（Multi Layered Map)」の概要について述べる。初期的なユーザヒアリングを行い、その結果も踏まえ、アプリケーションの機能要求定義を定めた。そして、設計するシステムの特長についても述べる。

３．１．本システムに対する機能要求

ユーザ参加型地図アプリケーション「M・L・M」に要求される機能を決定するために、既存の一般的なオンライン地図サービスを使用した経験のある大学生 22 名（男性 20 名・女性 2 名）を対象に初期的なユーザヒアリングを行った。その結果、得られたコメントについて以下に述べる。

１）地図アプリケーションの操作に関するもの

・クリックなどの操作を行わずに、マウスポインタを動かすだけで様々な操作が可能な地図アプリケーションが欲しい。
・クリック、ドラッグ＆ドロップ操作だけでなく、ホイール操作を用いて地図を動かせる機能があればよいと考えられる。

２）地図アプリケーションの機能に関するもの

・ニュースやテレビ番組などで取り上げられている地域をリアルタイムで地図上に表示し、その内容と共にユーザに提供してくれるシステムがあれば便利だ。
・地図上に表示されている情報が細かすぎて、探すのが大変だ。
・指定した地点から半径〇km といった範囲を指示したときに、範囲内に存在するスポット情報などを検索して出力する機能が欲しい
・１つの地図を仲間同士で持てるようにして、遊びや旅行の計画を立てるのに使用できればよいと思う。
・自分がアクセスした地図上のスポット情報に対して、順番にそのスポット情報の場所を自動で追尾するような機能があれば面白いものになるのではないか。

これらの得られたコメントなども参考に、ユーザ参加型地図アプリケーションである本システムに必要とされる機能要求の選定を行って、表 2 のように決定した。

表 2. システムへの機能要求事項

要求番号	要求項目
1	インターネット地図上に書き込まれた情報について、複数ユーザの相互間で共有できるようにすること。
2	複数ユーザ間で共有されているインターネット地図上にユーザ自身で情報の追加を行うこと。
3	ユーザがインターネット地図上に追加した情報を蓄積できるようにすること。
4	インターネット地図上に追加・蓄積されている情報から、その情報の種類を分類するなどして画面上で閲覧ができること。
5	既にインターネット地図上に追加されている情報に関して、その情報をユーザ自身で編集や削除することが容易に可能であること。

表 2 における決定されたそれぞれの機能要求の詳細について以下に述べる。

１．インターネット地図上に書き込まれた情報について、複数ユーザの相互間で共有できるようにすること。

システムを構築する前提条件として、１つのインターネット地図上に複数ユーザがアクセス可能であることが必要とされる。したがって、本システムにおいては１つのＷｅｂサーバ上で地図アプリケーションを稼動させ、そのＷｅｂサーバに複数のクライアントからアクセスができれば本要求を実現できると予想できる。

２．複数ユーザ間で共有されているインターネット地図上にユーザ自身で情報の追加を行えるようにすること。

要求番号１の機能が満たされたと仮定した上で、そのインターネット地図上においてユーザが個人単位で口コミやスポット情報などの書き込みができる仕組みを構築し、組み込むことで実現する。

３．ユーザがインターネット地図上に追加した情報を蓄積できるようにすること。

情報の追加をする機能があったとしても、アプリケーションを終了した時点でサーバ上にファイル（もしくはデータベース上のデータ）としてその情報が蓄積されなければ、他のユーザがアクセスしたときに追加された情報を取得することができない。その問題を解決し、ユーザがインターネット地図上に追加した情報を蓄積できるようにする機能を実装することが必要であると考えられる。

４．インターネット地図上に追加・蓄積されている情報から、その情報の種類を分類するなどして画面上で閲覧ができること。

地図アプリケーション実行時にサーバ上に蓄積されている情報に対して、個々のユーザが本当に欲しい情報のみを取り出して提示ための機能で、本システムの核となる部分だといえる。スポット情報を保存する際、インターネット地図上の 2 次元座標や情報の中身のほかに、スポット情報に対して数種類の属性を与えることによって差別化をはかる。そのように情報の種類を分けることで本要求を実現させる。

５．既にインターネット地図上に追加されている情報に関して、その情報をユーザ自身で編集や削除することが容易に可能であること。

サーバ上に蓄積されている情報に対して、他のユーザがその情報にアクセスした際に「編集したい」といった欲求が生まれる。また、情報が蓄積されていくだけでは双方向性がなく、本来、意図していた「個人に特化した情報」を生かすためのユーザ参加型地図アプリケーションとはいえない。そこで本システムにおいても双方向性を持たせるために、アプリケーションの管理者以外にも、ユーザによって投稿された情報の編集や削除が可能な仕組みを実現することが必要であるといえる。いわば、地図を媒体としたＷｉｋｉのような機能である。

３．２．本システムの特長

本システムは、現在の地図サービスには実現されていない「個人単位によって地図上に投稿された情報の中からユーザが知りたいと思ったジャンル（投稿者)の情報を抽出して提示する機能」(以下、レイヤー機能とする) をアプリケーションの機能に組み込んだ。このレイヤー機能は、地図上に投稿された情報に対し、特定のカテゴリに対するデータを取り出すための情報（「カテゴリ種別」）と、特定ユーザが投稿したデータを識別するための情報（「ユーザ種別」）を付加するものである。付加された 2 つの情報を用いて、地図上に書き込まれている情報から指定された「カテゴリ種別」と「ユーザ種別」に見合った情報のみを抽出してユーザに提示することを目的としている。「カテゴリ種別」を付加する目的は、飲食店についての情報を検索し、リンクをたどって行ったとしてもユーザが求めた飲食店の情報にたどり着かない状況が続いてしまうケースが生じないようにするためである。また、「ユーザ種別」を付加する目的は、映画館に関しての情報に定

評のある A さんの情報を取り出そうとしても、書き込まれた情報コミュニティ単位で扱われているためにその情報が A さんのものであるか知るためである。「カテゴリ種別」と「ユーザ種別」にそれぞれ関連付けられているデータをアプリケーション本体に渡して、書き込まれている情報の中から選択された「カテゴリ種別」と「ユーザ種別」に見合った情報のみを抽出するようにしている。このような機能を組み込むことで、誰が投稿した情報であったのか取り出すことを可能にし、地図の個人化（パーソナリゼーション）を図っている。そして、SNS を結びつけた地図サービスで発生していた、地図上に投稿された情報が多すぎて埋もれてしまっている情報の中から本当にユーザが欲している情報を取り出して提供することが可能になるのではないかと考えて設計を行った。

また、一般公開されている無償の Google Maps API を用いて設計を行ったことなど、オープンな技術のみを用いて実現したことから、将来的に本システムをひとつのパッケージとすることも可能である。それにより、個人に特化した地図アプリケーションをユーザ自身で自由にカスタマイズして使用できるといった点も特長のひとつであるといえる。このような特長を本アプリケーションに持たせることで、現在提供されている一般的な地図サービスや SNS を結びつけた地図サービスには無い機能をユーザに提供することができるのではないかと考える。

４．ユーザ参加型地図アプリケーション「M・L・M」の設計・開発

前章において、ユーザ参加型地図アプリケーション「M・L・M」の概要を述べた。そこで本章では設計に用いる Google Maps API の技術的な面を説明する。また、定められた機能要求定義に従ってシステム概念図をつくった。その後、システムに必要となる各機能の設計を行う。また、開発環境を定め、設計されたものに従って実際にシステムの開発を行った。

４．１．Google Maps API の機能

Google Maps は、Google 社が 2004 年に公開したオンライン地図情報サービスのひとつである。Ajax（Asynchronous JavaScript and XML）[17]と呼ばれる技術が用いられており、Ajax の非同期通信によって、クライアント側で特殊なプラグインなどを用いることなく、Web ブラウザのインターフェースの中でシームレスなマウスドラッグによる移動や、地図の縮尺変更を行うことが可能になっている[18]。また、地図を表示する機能だけでなく、実写の衛星写真を表示する「航空写真」、地図と衛星写真を重ね合わせて表示する「地図＋写真」といった機能を持っている。そして、2005 年 6 月に「Google Maps」の基本的な機能を誰でも Web ブラウザ上に埋め込んで使用できるようにする「Google Maps API」の公開が行われた。この機能は、オンライン登録を行えば無料で使用することができるようになっており、ユーザ自身で地図機能のカスタマイズも可能となっている。プログラミング言語 JavaScript によるライブラリとして提供されており、標準の機

能として地図上にマーカーを立てることや、情報ウインドウを用いて地図上に情報の表示を行うことができる。本研究では、このような機能を使用して、ユーザ参加型地図アプリケーション「M・L・M（Multi Layered Map)」の設計を行った。以下、本アプリケーションを構築する上で重要な Google Maps API の内部機能について説明する。

・GLatLng(緯度，経度)

緯度（lat）と経度（lng）による、地図上の２次元座標を生成するための関数で、緯度は-90 度から+90 度の間、経度は-180 度から+180 度の間の値をとる。

・GDownloadUrl(ＵＲＬ，処理内容)

ＵＲＬで指定されたスクリプトの実行や、ファイルの読み込みを行うための関数である。ＵＲＬのリンク先は実際のソースファイルと同じサーバ内に存在している必要がある。

・GEvent

地図上でのマウス操作などに対するイベントを生成するための機能である。また、イベントの検知などの関数も含んでいる。

・GMarker

地図上に表示するマーカーを生成するための関数である。マーカーを生成するための要素として地図上の２次元座標とマーカーのアイコン情報の要素を持っており、アイコン情報がない場合は標準設定のアイコンが使用されるようになっている。マーカーは、マウスや情報ウインドウのイベントによって操作でき、マーカー用の情報ウインドウの利用ができる。

・GInfoWindowTab(label，content)

タブつきの情報ウインドウを生成する関数である。タブの構成要素としてタブ名（label）とタブの内容（content）が必要となっている。

４．２．本システム全体の設計

ユーザ参加型地図アプリケーション「M・L・M」に要求される機能は前節で決定されたので、本システムの全体像について設計を行った。図３にその概念図を示す。

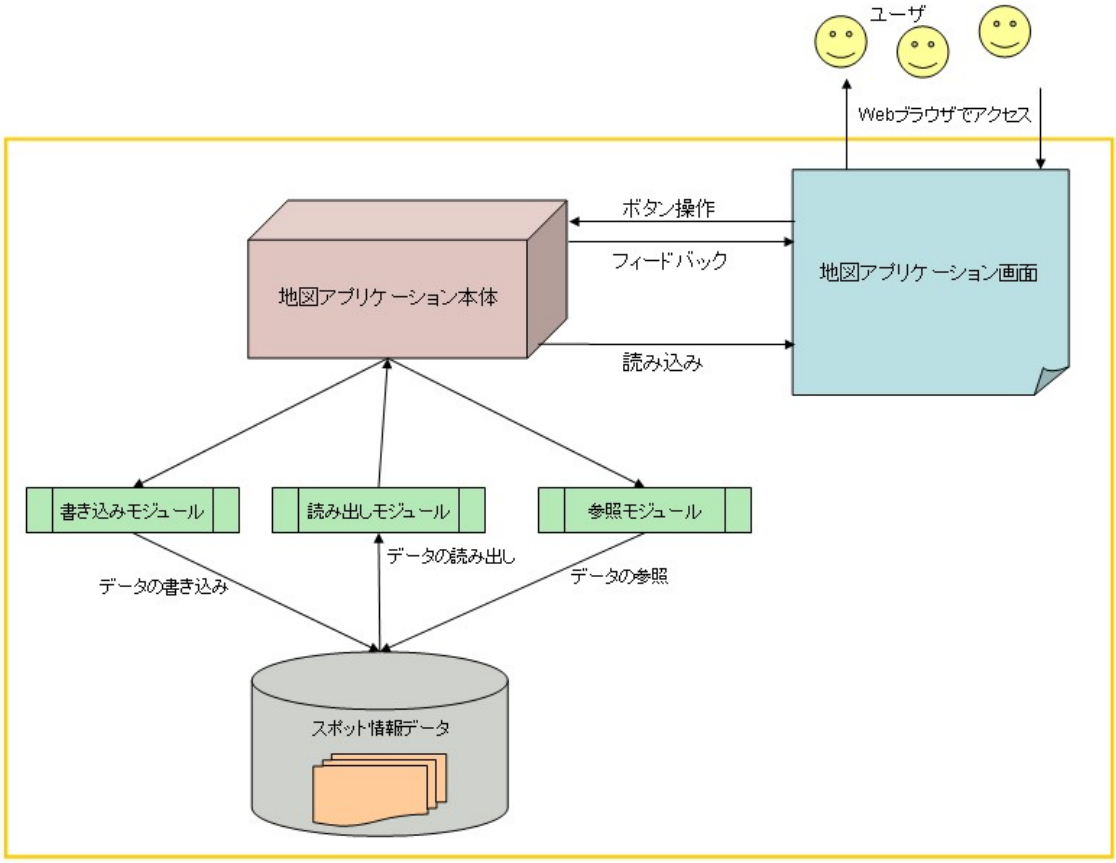

図 4. 「M・L・M」のシステム概念図

また、本システムを構成しているそれぞれの要素がどのような機能を持っているのか明確にしなければならない。各要素が保有している機能をまとめた表を表 3 に示す。

表 3.「M・L・M」のシステムの構成要素とその機能

構成要素	構成要素が持つ機能
地図アプリケーション画面	地図アプリケーションの表示 地図アプリケーションに必要なフォームやボタンといったＧＵＩ部品 画面デザイン（スタイルシート）
地図アプリケーション本体	メインモジュール POST 関数(書き込みモジュールと中継) 地図上のマーカーに関する操作 スポット情報の条件つき出力機能 ユーザ(アイコン)種別切り替え機能
書き込みモジュール	地図アプリケーション本体から送られてきた情報を Web サーバに保存する機能。また、既に保存されている情報の更新を行う機能。
読み出しモジュール	サーバ上に保存されているスポット情報の内容を地図アプリケーションに渡す機能。
参照モジュール	地図アプリケーション本体がスポット情報にアクセスするための機能
スポット情報データ	地図アプリケーション本体から送られてきたスポット情報をファイルとして格納しておくスペース。

以上のように、本システムの実現において必要な機能やデータ構造については、①地図アプリケーション画面、②地図アプリケーション本体、③書き込みモジュール、④読み出しモジュール、⑤参照モジュール、⑥スポット情報データ、の６つの構成要素が必要になると考えられる。

４．３．本システムの構成要素の設計

地図アプリケーション本体に必要とされる機能は先に示した通りであるが、本項ではそれらの機能の中身を掘り下げて説明する。また、それぞれの設計した機能の処理の流れをフローチャートで示す。

４．３．１．地図アプリケーション画面の設計

Web ブラウザ上で表示する地図アプリケーションの出力画面を構成する際にいくつかの要素が存在する。1 つ目に画面のメインとなる地図、2 つ目にスポット情報の条件

付出力機能や、ユーザ（アイコン）種別切り替え機能に用いるプルダウンメニューとボタンが最低限必要であると考えられる。また、地図の移動をキーワード検索によって実現するためのフォームや、画面に出力されている地図の座標、アプリケーションのタイトルといったコンテンツも実装することにした。また、地図の描画領域をブラウザのサイズに連動させることとした。しかしながら、地図画面の部分以外にもタイトルやフォームなどといったアプリケーションの要素が存在することから、画面のサイズとして幅 600 ピクセル×高さ 800 ピクセル程度は必要になると考えられる。図 4 に本アプリケーションのコンテンツレイアウトを示す。

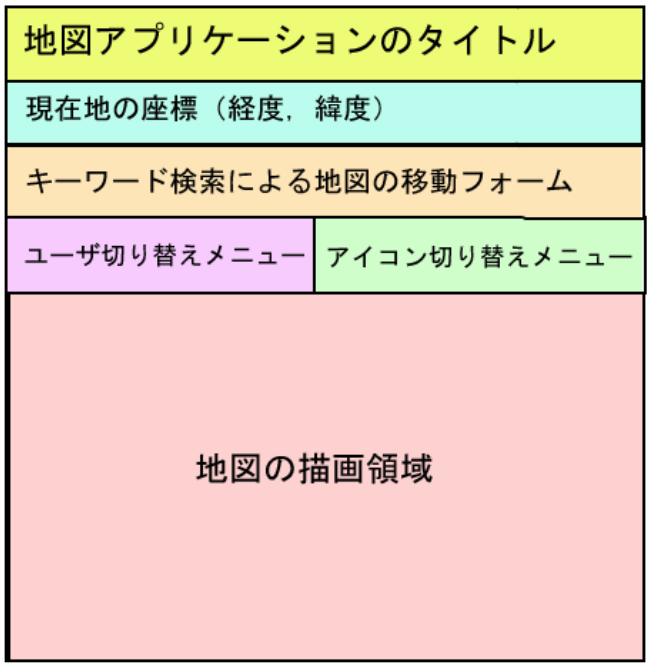

図 5.　コンテンツレイアウト

４．３．２．地図アプリケーション本体の設計（mapwiki2.js の設計）

アプリケーション本体の機能で、まず初めに地図の生成、中心座標の指定、API によって定義されている地図をコントロールする機能（拡大・縮小・地図と衛星写真の切り替え）の呼び出しを行う。次にサーバに保存されているスポット情報データを参照し、読み出しモジュールを用いて生成された地図にスポット情報のマッピングを行う。これで、地図アプリケーションの起動時の画面が出力されたことになる。地図上のマーカーに関する操作は、マーカーや情報ウインドウの生成、スポット情報に関する処理の流れをあらわしたものである。POST 関数は、生成された情報ウインドウに書

き込まれたスポット情報をサーバに保存する操作を行ったときに、書き込みモジュールに渡すためのデータ形式に変換するための機能を有する。また、正しく変換されたデータをひとまとめにして書き込みモジュールに渡すようにしている。スポット情報の条件付出力機能は、地図アプリケーション画面のプルダウンメニューと連動して、サーバに保存されているスポット情報データの中から画面上のプルダウンメニューより指定された条件を満たすスポット情報を画面に出力するための機能である。ユーザ（アイコン）種別切り替え機能は、地図アプリケーション画面のプルダウンメニューと連動して、書き込み時のユーザ種別やカテゴリ種別の選択を行うための機能である。図 5 に地図アプリケーション本体の処理の手順をあらわしたフローチャートを、図6に情報ウインドウの操作の手順を示したフローチャートを、図 7 にスポット情報の条件付出力機能の処理を示したフローチャートを、図 8 に種別切り替え機能の処理を示したフローチャートをそれぞれ示す。

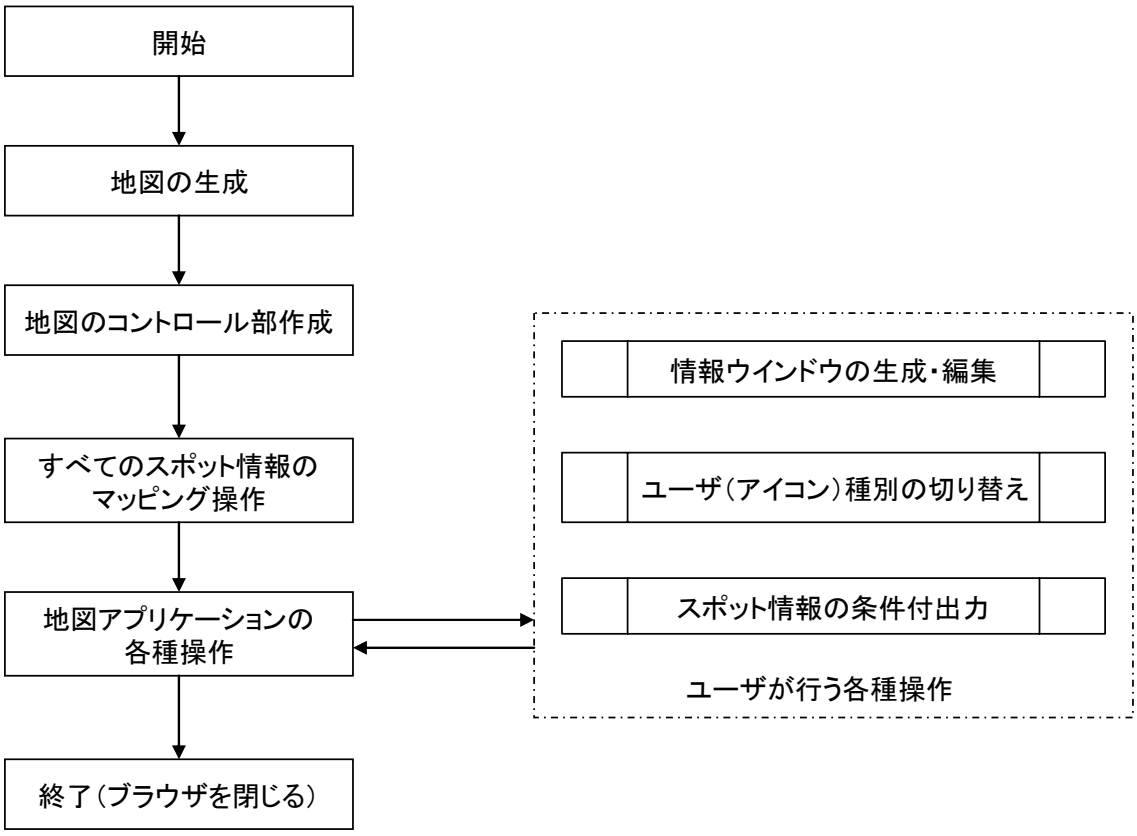

図 6　地図アプリケーション本体のフローチャート

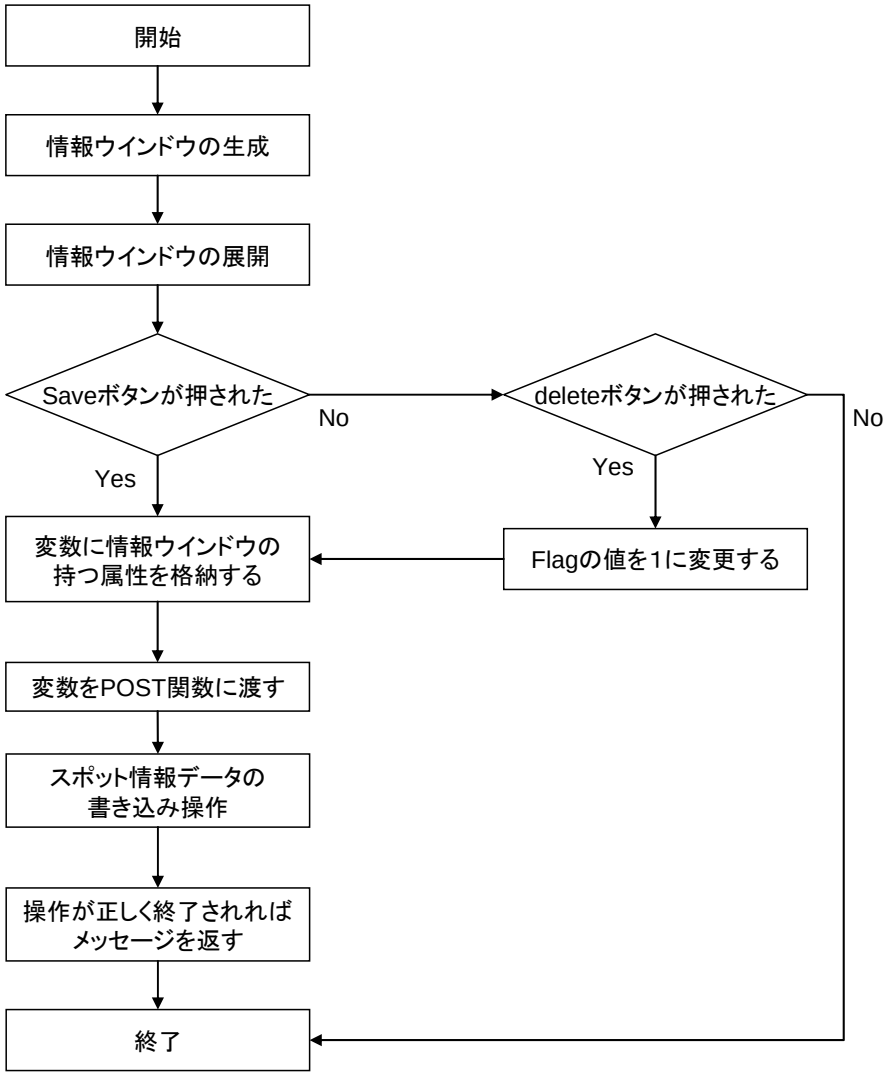

図 7　情報ウインドウの操作を示したフローチャート

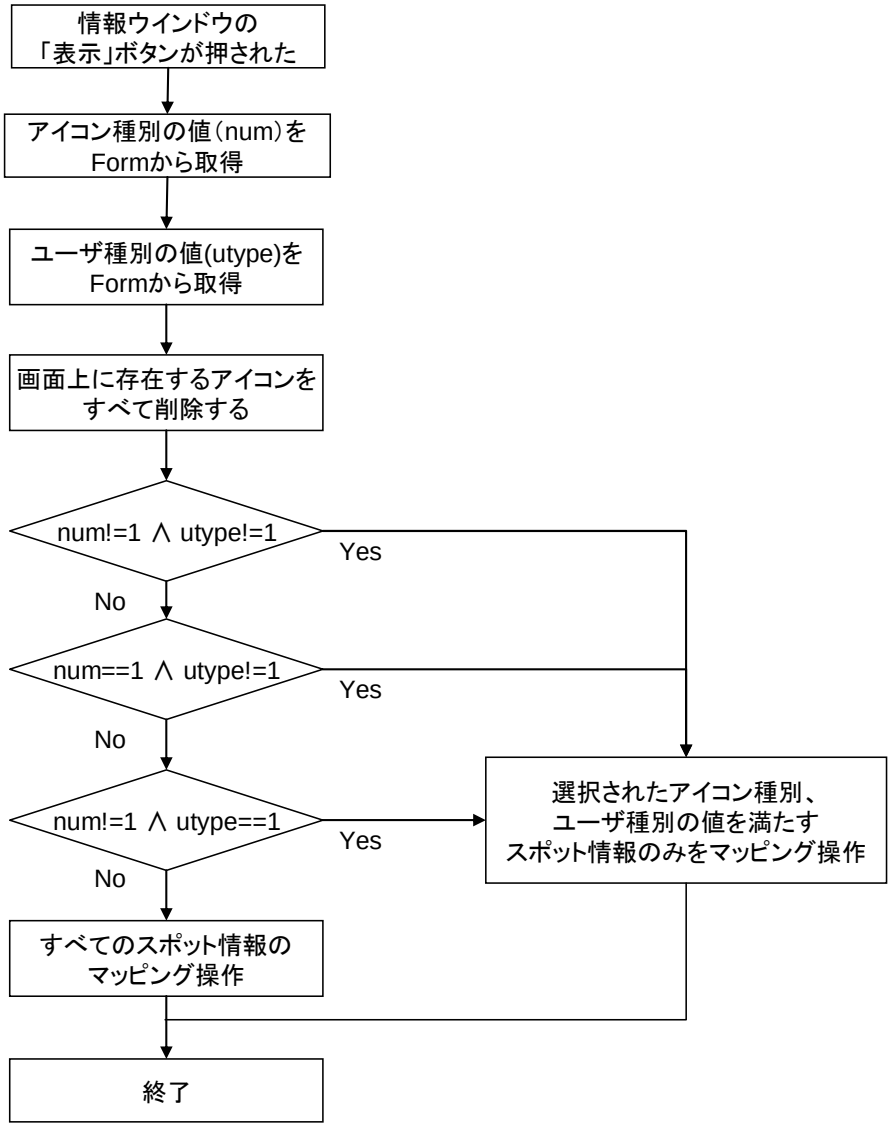

図 8. スポット情報の条件付出力機能のフローチャート

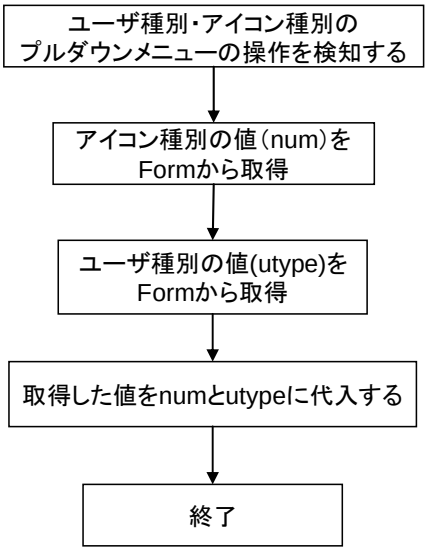

図 9. 種別切り替え機能のフローチャート

４．３．３．書き込みモジュールの設計（write.php の設計）

地図アプリケーション本体の POST 関数から渡されたスポット情報データをサーバのスポット情報データを格納しておくディレクトリに保存するための機能を有するモジュールである。このモジュールは本体の情報ウインドウにある「save」ボタンが押された際に呼び出され、スポット情報ファイルの存在するディレクトリに移動する。次にファイルの一覧を取得し、受け渡された情報が正しいものであるかを確認した後にスポット情報ファイルの作成・更新を行う。書き込みモジュールの処理の流れを示したフローチャートを図 9 に示す。

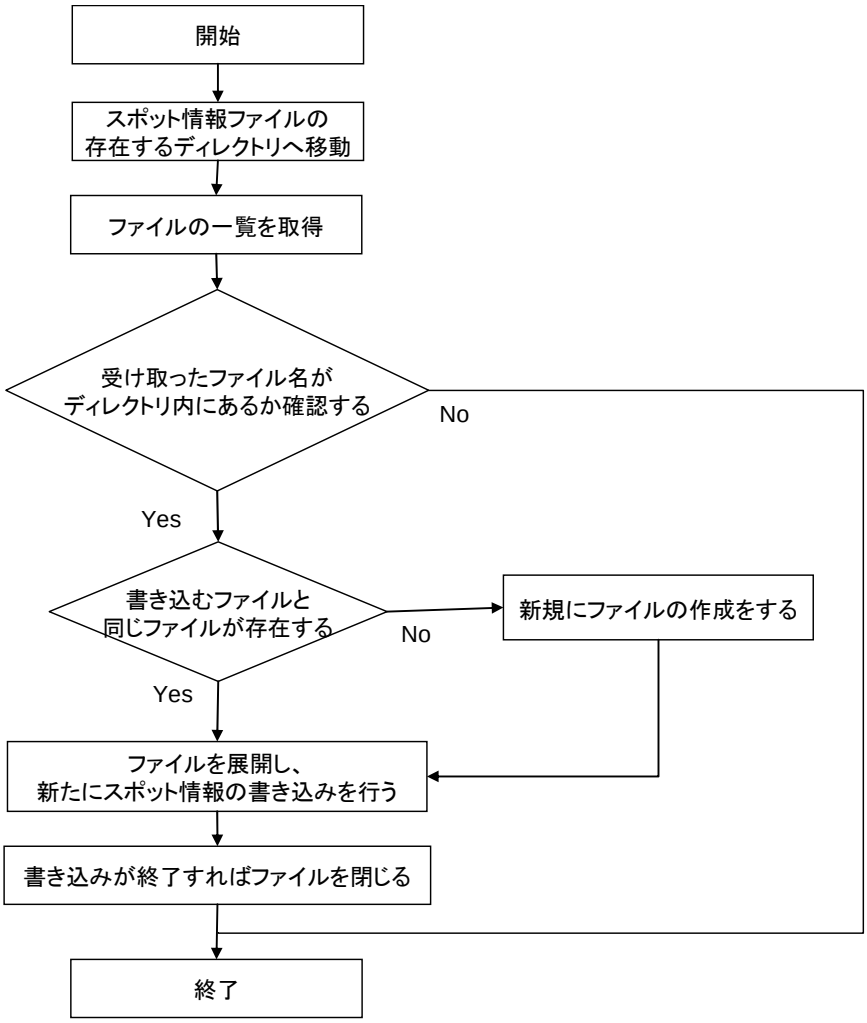

図 10.　書き込みモジュールのフローチャート

４．３．４．読み込みモジュールの設計（read.php の設計）

地図アプリケーション上にスポット情報をマッピングするために、スポット情報データを格納しているディレクトリからデータを読み出し、地図アプリケーション本体にそのデータを渡すための機能を有するモジュールである。このモジュールは地図アプリケーション本体の初回起動時に必ず実行される。また、スポット情報の条件付出力機能が実行される際にも実行される。処理の中身としては、初めにスポット情報ファイルの存在するディレクトリに移動し、次にファイルの一覧を取得する。読み込むファイルの情報が正しいものであるかを確認した後にスポット情報ファイル読み込み、そのファイルの中身を展開し、地図上に出力する。モジュールの処理の流れを示したフローチャートを図 10 に示す。

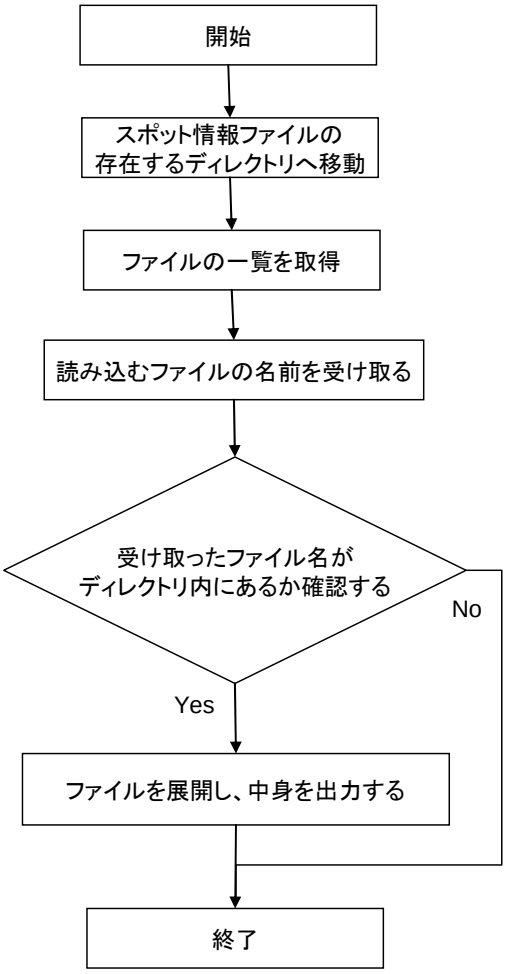

図 11.　読み込みモジュールのフローチャート

４．３．５．スポット情報データのファイル形式

本アプリケーションにおけるスポット情報のファイルは JSON(JavaScript Object Notation)[19]形式とする。1 つのファイルに対して構成している要素の属性と値は表 4 に示される 7 つである。

表 4.　スポット情報ファイルのデータ構造

属性名	値
id	データを一意識別するための個別ID
lat	緯度の座標
lng	経度の座標
data	スポット情報のコメントの中身
icontype	アイコン種別
userid	ユーザ種別
flag	フラグ

４．４．開発環境

本研究の開発環境について述べる。図 4 の各構成要素は FedoraCore6 を OS とするサーバ上にシステムの実装を行った。IDE（統合開発環境）はフリーソフトの Aptana を用い、言語は地図アプリケーション本体では Google Maps API を用いるために JavaScript、書き込みモジュールと読み出しモジュールは PHP を用いた。アプリケーションを使用するための Web ブラウザは Firefox での動作を想定して開発を行った。

５．ユーザ参加型地図アプリケーション「M・L・M」の機能

本章では、前章で開発された、ユーザ参加型地図アプリケーション「M・L・M」について述べる。また、使用方法について簡潔に説明するさらに、スポット情報の抽出について使用にあたってのシナリオを示す。

本研究で開発したアプリケーションはすべての機能モジュールをひとつの Web サーバに置くことで動作させることとした。また、アクセスした時点で地図アプリケーション画面が展開されるのを防ぐために、トップページの作成を行った。そこには簡単な使用方法や本アプリケーションの各機能の概要を表示し、本アプリケーションへのリンクを張ることとした。そのリンクをクリックすると本アプリケーションにアクセスでき、開始される。画面の構成は設計の項で述べた画面設計に基づいて作成されており、設計・開発を行った各機能が使用できるようになっている。そして、開発を行って実装した本アプリケーションの実画面を図１１に示した。

図 12.　完成アプリケーションの実画面と操作手順

例えば、スポット情報を作成して Web サーバに保存するには以下の手順で行えばよい。

①スポット情報を作成したい地図上の場所をクリックして情報ウインドウを開く。

②入力フォームにスポット情報の中身を入力する。

③入力フォームの下にある「Save」ボタンをクリックする。

③'また、図 12 の画面において、表示されているスポット情報を削除するためには、入力フォームの「delete」ボタンをクリックすればよい。また、情報の内容を変更して新たに上書き処理を行うときは、スポット情報を作成したときと同じように、入力フ

ォームの中身の変更を行った後で「save」ボタンを押せばよい。

また、地図の移動を入力フォームにキーワードの入力を行って「移動」ボタンを押せば検索にヒットした場所を中心として地図が描画される機能の実装も行った。例えば、ユーザが大阪駅周辺の地図を見たいと思ったときに、入力フォームに「大阪駅」と入力し、「移動」ボタンをクリックする。すると、検索機能がバックグラウンドで起動し、検索がヒットすれば地図上の大阪駅の上に「↓」が表示される仕組みになっている。その図を図 13 に示す。

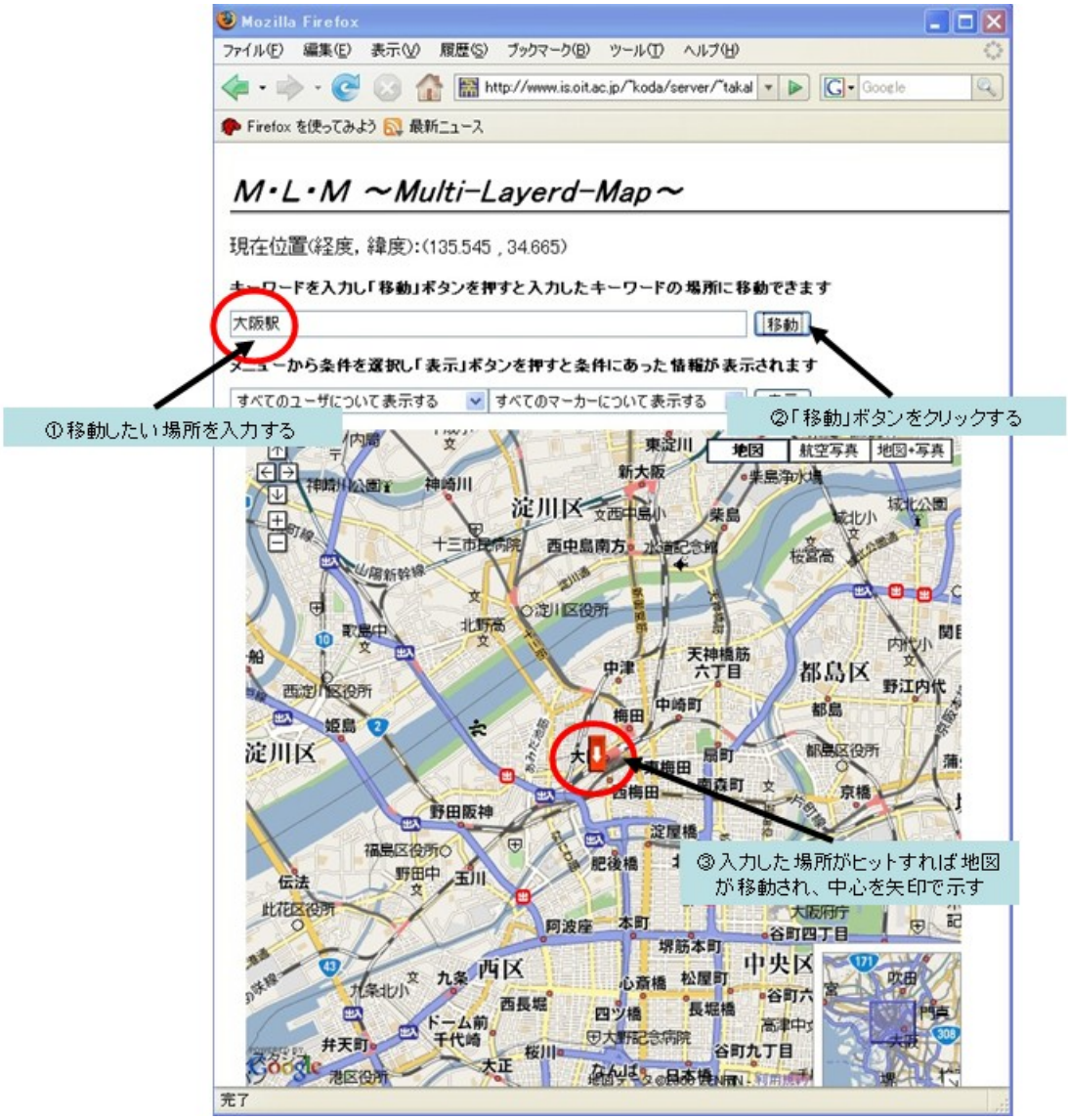

図 13. 検索機能を用いて「大阪駅」で検索した結果の画面と操作手順

次に、図 12 の状態から本アプリケーションの核となるレイヤー機能を用いて上記の画面からある特定の条件を満たすスポット情報のみを抽出した画面を図 14 に示す。その一連の作業の例として、以下に示すシナリオが考えられる。

シナリオ例

①Aさんは地域のたこ焼き屋の情報について知りたいと思っている。

②M・L・Mにアクセスしたものの、地図画面を見たらさまざまなジャンルの情報が地図上に山のようにあったため困ってしまった

③しかしながら、Aさんは、Bさんがおいしいたこ焼き屋の情報に定評があることを知っていたので、Bさんが投稿した情報について知りたいと考えた。

④そこでAさんは、「たこ焼き屋」というジャンル情報と「Bさん」という投稿者情報の２つを用いて地図上に投稿されている多くの情報から欲しい情報のみを取り出すことに成功した。

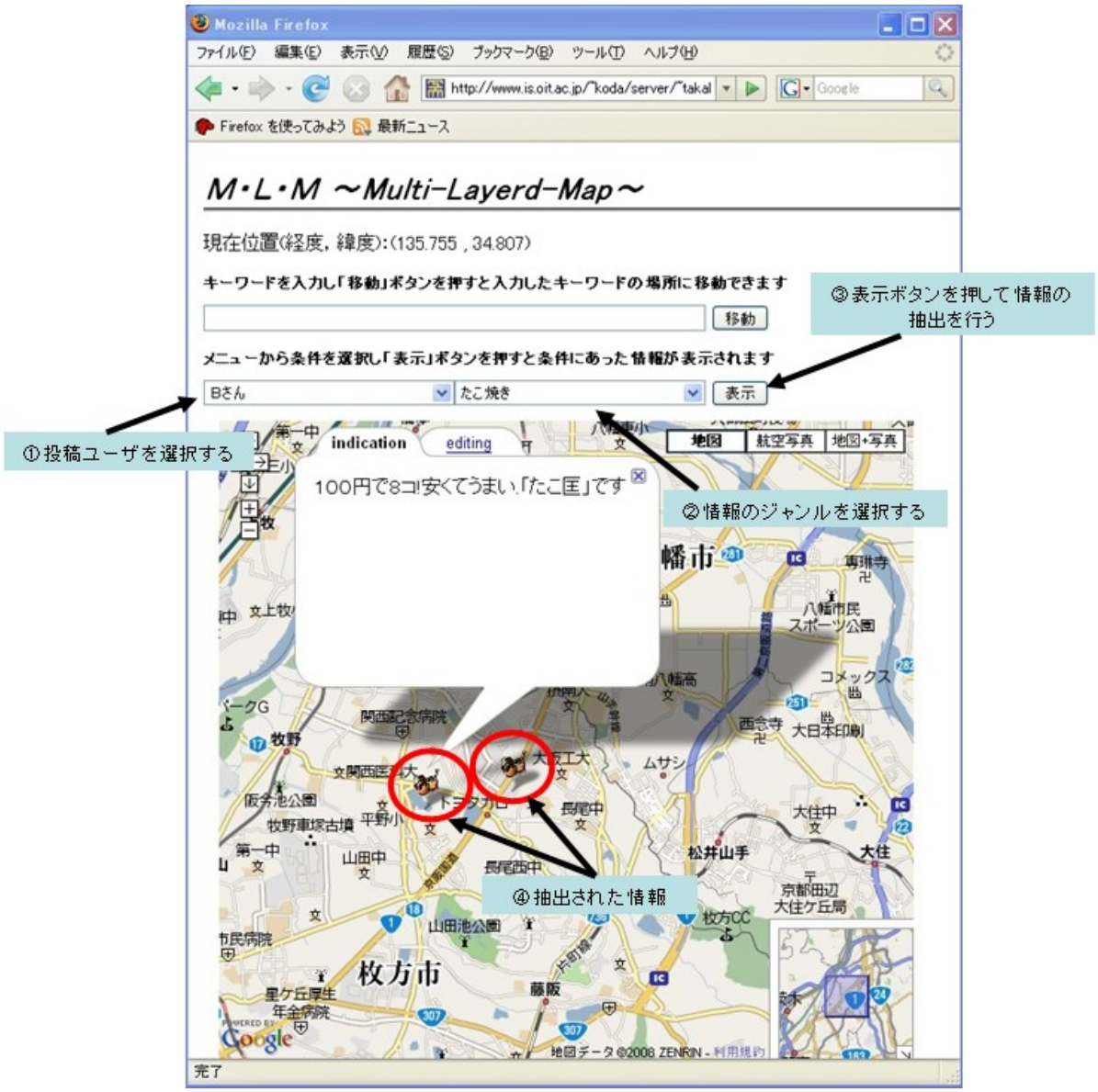

図 14.　抽出されたスポット情報と操作手順

まず初めに投稿ユーザの種類を選択する必要があることから、1つめのメニュー（左側）から投稿ユーザの種類を選択する。(今回のシナリオでは「Bさん」を選択する) 2番目に、知りたい情報のジャンルを決める必要があるので、2つ目のメニュー（右側）から投稿されている情報のジャンルの選択を行う（今回のシナリオでは「たこ焼き」を選択する)。そして表示ボタンが押されると、選択されている投稿ユーザの種類と情報のジャンルの2つの条件を満たすスポット情報（今回のシナリオでは、「Bさん」が投稿した「たこ焼き屋」についての情報)のみが画面上にプロットされるという仕組みである。確かに図 14 を見れば、条件を満たす情報だけが画面上にプロットされているのがわかる。

6．評価

本アプリケーションのデモンストレーションを大阪工業大学情報科学部「第 11 回北山祭」で行った。そこで実際に来場者に使用していただき、ユーザヒアリングを実施した。そこで得られたコメントを以下に示す。

1）運用について

・防災マップとして使ってみたい。近隣の自治会に紙で一軒ずつ配布するより Web 上において住民同士でひとつのホームページとして作ったほうが効率よく情報の収集ができ、面白いのではないか。また、防災マップを作成する人の負担も減るのではないか。

・このアプリケーションのパッケージを得られるのなら、自分たちでオリジナルの地図を媒体としたコミュニティを簡単に作成できるのでやってみたい。

・グループ単位で共有できるインターネット上の地図というものが、今までになかったので便利だ。

2）レイヤー機能について

・レイヤー機能におけるスポット情報の抽出に関して、キーワード検索といったより高度な抽出機能を実装して欲しい。

以上のようなコメントを得られた。そこで、これらの意見をふまえた上で、地域の防災マップとしての利用を想定したものを図 15 に示す。

図 15. 防災マップとしての利用例

図 15 ではユーザが自治体の職員が追加した公共施設の情報についての知りたいという要求から「ユーザ種別」を自治体職員に、「カテゴリ種別」を公共施設に設定して情報の出力をした画面である。また、数名に使用してもらい、いくつかのコメントを頂いた。そのコメントを以下に示す。

1）防災マップとしての機能について

・一般的な防災や地域の安全としての情報以外にも、ちょっとした情報を載せられる点では優れているのではないかと思う。

2）使用感について

長所

・ユーザ自身で地図を作っていると実感できるのは良いと思う。

・ユーザ自身でスポット情報データの属性の追加できる機能など、ユーザがアプリケーション本体部分に参加できるようにすれば面白いと考えられる。

短所

・防災マップとしてある程度完成されているものを与えられないと、ユーザとしては自分たちで地図を作っていこうという意欲がわかない。

・書き込み手法をもっと簡略化するか、画面上にマニュアルがあったほうがよい。

7．考察

本研究では、2 章で紹介した現在の地図サービスには存在しなかったレイヤー機能を実現している。特長にも挙げていたように、一般公開されているオープンな技術のみを用いて実現することもできた。また、開発した本アプリケーションは一般的な SNS サービスにある認証技術による会員制といったものを持っていないために、誰でも地図上に情報を書き込むことができ、編集や削除も投稿者に限定されることなく行うことができるようになっている。いわば、WebGIS の Wiki 化を実現したといえる。Wiki とは Web で見ているページを、Web ブラウザ経由で書き換えることができるシステムである。Cunningham によって開発され、簡単な記法によって構造を持った文章を書くことができる。また新しいページを作り、ページからページへのリンクを張ることができる、非常にシンプルなシステムであり、気軽に新しい情報を加え、編集することができる、編集可能な Web システムである。情報を蓄積・編集し、不定形の情報を柔軟にまとめていくことができるという点で、Wiki は非常に使いやすいシステムである。全てのページを誰でも自由に書き換えできるため、管理者を必要としない、といった利点がある[20][21]。

これらを総合すると、開発した本アプリケーションは、既存のサービスとして提供されている「ちずる」[15]、「Wikimapia」と比較して、それらになかった、「個人単位によって地図上に投稿された情報の中からユーザが知りたいと思ったジャンル（投稿者)の情報を抽出して提示する機能」（レイヤー機能）を実現することで、WebGIS の個人化（パーソナリゼーション）を実現している。また、WebGIS の Wiki 化を実現した。それらを実現したことで、ユーザに近い情報を埋もれさせることなくユーザに提供できていることがいえるのではないか。

本研究では、WebGIS の Wiki 化やレイヤー機能の実装について重点的に扱ってきており、そこで今後の拡張として考えられることを以下に述べる。ユーザアンケートにもあったよう、スポット情報を抽出する際にキーワード検索を可能にしたりすれば、さらに良いものになるのではないかと考える。Web2.0 のサービスでは、「利用者にとって価値が付加されることにより全体の価値が高まり、そのサービスの成功に結びつくことが多

く、利用者からの貢献が受けられることは重要なポイントなのである[22]」と示されている。本アプリケーションは、現時点においては情報ウインドウにテキストしか書き込むことができない仕様となっている。そこで、画像やリンクが書き込めるようになれば、さらにユーザの自由度が増し、より大きな価値が付加されると期待でき、ユーザの書き込み意欲も向上するのではないかと考えられる。また、スポット情報を書き込むための手順を示してあるページのリンクをアプリケーション画面に表示させておくことも望まれている。

８．おわりに

本研究では、SNS などと地図を結び付けたサービスで情報を得る際に、ユーザから地図上に投稿された「個人に特化した情報」を生かすために、ユーザ参加型地図アプリケーションの設計と開発を行った。開発したアプリケーションでは、地図上に投稿された情報に対して、特定のカテゴリに対するデータを取り出すための情報と、特定ユーザが投稿したデータを識別するための情報の２つを付加して地図上に情報を投稿することができる。

上記の機能を実現することで、１つの情報が他の大量の情報の中に埋まってしまうのを防ぐとともに、それを元にしてユーザから投稿された情報の中から特定のカテゴリに対するデータや特定のユーザが投稿したデータを抽出できる機能を持っている。例えば、映画館の情報に関して定評のある A さんの投稿した情報のみを取り出すことが可能になる。また、誰でも地図上に情報を書き込むことができ、編集や削除も投稿者に限定されることなく行うことができる仕様としていることで、WebGIS のパーソナリゼーション化、WebGIS の Wiki 化を実現した。今後は、ユーザアンケートにあったスポット情報を抽出する際にキーワード検索を可能にする機能や、書き込む情報の要素をテキスト文書だけでなく、画像やリンクの投稿を可能とすることについて検討することで、投稿時の自由度を増やす必要がある。

謝辞

本研究を進めるにあたって適切な御指導、多大なアドバイスをいただいた大阪工業大学情報メディア学科神田智子准教授に心より感謝いたします。大阪工業大学情報科学部北山祭実行委員会には「第 11 回北山祭」におきまして、研究室発表の機会をいただきました。また、日頃から有益なご意見を頂き、議論を交わしていただきました大阪工業大学情報科学部情報メディア学科ヒューマンインタフェース研究室の関係者各位に心よりお礼申し上げます。

参考文献

[1] Gevin V. ,”Mapping Opportunities” , Nature Vol.427, 2005,Jan 22,pp.376-378.

[2] Google Earth ,http://earth.google.co.jp/

[3] Virtual Earth,http://maps.live.com/
[4] World Wind,http://worldwind.arc.nasa.gov/
[5] Butler D., “The Web-wide world”, Nature Vol.439, 2006,Feb 16,pp.776-778.
[6] 厳網林, 2006, デジタルアースで拓く GIS の新しい展開,
電気学会 H18 年度産業応用部門大会
[7] Google Maps API, http://code.google.com/apis/maps/index.html
[8] 国土地理院ホームページ, http://www.gsi.go.jp/GIS/
[9] 東善広・藤田友丈,”参加型環境ＧＩＳの課題と展望”, 琵琶湖環境科学研究センター,
平成 16 年度 試験研究報告, P17~26
[10] MapFan , http://www.mapfan.com/
[11] Mapion, , http://www.mapion.co.jp/
[12] Google Maps, http://maps.google.co.jp/
[13] Yahoo! Maps API ,http://developer.yahoo.co.jp/map/
[14] Virtual Earth API,http://dev.live.com/virtualearth/
[15] ちずるー地図でつながるクチコミ情報コミュニティ, http://chizu-ru.jp/portaltoppre/
[16] Wikimapia, http://wikimapia.org
[17] Ajax：Web アプリケーション開発の新しいアプローチ,
http://antipope.zapto.org/docs/translations/ajax.html
[18] 高橋登史朗,”入門 Ajax” ソフトバンククリエイティブ,2005
[19] JSON, http://json.org/json-ja.html
[20] Leuf, B. and Cunningham, W.”The WikiWay Quick Collaboration on the Web”,Addison-Wesley ,2001.
[21] 江渡浩一郎, 高林哲, 増井俊之 “qwikWeb：メーリングリストと Wiki を統合したコミュニケーション・システム” 情報処理学会研究報告. ヒューマンインタフェース研究会報告, Vol.2004, No.115(20041111) pp. 5-11 ,2004
[22] 遠藤智代, ”Web2.0 - 利用者参加による情報の共有と価値の付加”,情報知識学会誌 Vol.16 , No.4 pp.4_1-4_10 ,2006

付録

地図アプリケーション　メインプログラム（mapwiki2.js）

```
var map;
var count=1;
var num=10;  //アイコン画像の個別 ID（ジャンル判別）

var utype=10; //ユーザ判別用の変数
var flag=0; //削除されたものか判定するフラグ

function startUp() {
  //クラス生成
  map = new GMap2( document.getElementById("mymap") );

  //中心を外部ファイルから読み込み
  GDownloadUrl( "center.json", mySetCenter );
    function mySetCenter(data){
    var obj = data.parseJSON();
    map.setCenter( new GLatLng( obj.lat, obj.lng), 13 ); //obj.zoom );
    }
  //map.setCenter( new GLatLng(35.6984, 139.7732), 13 );
  //map.addControl( new GLargeMapControl() );
  map.addControl( new GOverviewMapControl() );
  map.addControl(new GSmallMapControl() );
  map.addControl(new GMapTypeControl() );

  //list.php でマーカーの一覧を取得(GET)
  GDownloadUrl( "list.php", function(listdata){
    var markers = listdata.parseJSON();
    if(markers)
      for(var i=0; i<markers.length; ++i) { //  それぞれについて…
        var markerID = markers[i];
        // read.php でマーカー情報を取得(GET)
        GDownloadUrl("read.php?id="  +  encodeURIComponent(markerID),
function(mkdata){
          //地図にマーカーを追加
```

```
            var mk = mkdata.parseJSON();
                if( mk.flag == 0 ){
                    num = mk.icontype;
                addEditableMarker( mk.id, new GLatLng(mk.lat, mk.lng), mk.data );
                    num=10;
                }
            });
        }
    });

    // クリック時もマーカー追加処理
    GEvent.addListener( map, "click", function(ov,latlng){
      if( ov )
        return;
      addEditableMarker( undefined, latlng, "Please add Infomation" );
    })

    //現在位置の座標を出力する
    GEvent.addListener( map, "mousemove", function(latlng){
        var p = document.getElementById("postext");
        p.innerHTML = "("+ shortForm(latlng.lng()) + " , " + shortForm(latlng.lat()) + ")";
    })

}

function shortForm(x){
    return Math.floor(x*1000)/1000;

}

function PostUrl( url, data, handler ) {
    // POST 用の XMLHttpRequest オブジェクトを作成
    var req = GXmlHttp.create();
    req.open( "POST", url, true );
    // 通常のフォームからの POST とヘッダを揃える
    req.setRequestHeader( "content-type",
```

```
        "application/x-www-form-urlencoded;charset=utf-8" );
    // 結果受信時の処理
    req.onreadystatechange = function() {
      if( req.readyState == 4 )
          handler( req.responseText, req.status );
    }
    // 送信
    req.send( data );
}

function addEditableMarker(id, latlng, initData) {
     //オリジナルマーカーの生成
     var icon = new GIcon();
     icon.image="marker/marker"+num+".png"; // +count+".png"; マーカー画像の読み込み
     icon.iconSize = new GSize(23, 25);                                    // マーカーサイズの指定
     icon.shadow = "http://www.google.com/mapfiles/shadow50.png"; // 影画像の読み込み
     icon.shadowSize = new GSize(37, 34);                                  // 影サイズの指定
     icon.iconAnchor = new GPoint(10, 34); // 画像のどの位置を指定座標にもってくるか
     icon.infoWindowAnchor = new GPoint(10,3);
     count ++;
     //num=1;

     var mk = new GMarker(latlng, icon);
     map.addOverlay(mk);

    GEvent.addListener( mk, "click", function(){
      var tab1 = new GInfoWindowTab( "indication", view );
      var tab2 = new GInfoWindowTab( "editing", form );
       var tab3 = new GInfoWindowTab( "del",form2 );
      mk.openInfoWindowTabs( [tab1, tab2 ], {"maxWidth":250} );
    });

    // "表示" タブの内容
    var view = document.createElement("div");
    view.innerHTML = initData;
```

```
// "編集" タブの内容
var form = document.createElement("form");
   var textarea = document.createElement("textarea");
      textarea.cols = "30";
      textarea.rows = "6";
      textarea.value = initData;
   var button = document.createElement("input");
      button.type       = "button";
      button.value      = "save";
      button.onclick = function(){ //ボタンクリック時の処理関数
        // view の内容切り替え
        view.innerHTML = textarea.value;
        map.getInfoWindow().selectTab(0);

              flag = 0;
        //  サーバーに保存
        var postdata = "data=" + encodeURIComponent(textarea.value);
        postdata += "&lat=" + encodeURIComponent(latlng.lat());
        postdata += "&lng=" + encodeURIComponent(latlng.lng());
              if( id !== undefined )
          postdata += "&id=" + encodeURIComponent(id);
              postdata += "&icontype=" + encodeURIComponent(num);
              postdata += "&userid=" + encodeURIComponent(utype);
              postdata += "&flag=" + encodeURIComponent(flag);


        PostUrl( "write.php", postdata, function(newID){id=newID;} );

    //alert("保存しました！")
    // alert("save compleate !");
      }

var form2 = document.createElement("form2");
   var textarea2 = document.createElement("textarea");
      textarea2.cols = "30";
      textarea2.rows = "6";
```

```
  textarea2.value = initData;

 var button2       = document.createElement("input");
 button2.type       = "button";
button2.value     = "delete";
 button2.onclick = function(){

            //ここに削除機能の中身を書く。(12/3 追記)

            // view の内容切り替え
      view.innerHTML = textarea.value;
      map.getInfoWindow().selectTab(0);

            flag = 1;
      //  サーバーに保存
      var postdata = "data=" + encodeURIComponent(textarea.value);
      postdata += "&lat=" + encodeURIComponent(latlng.lat());
      postdata += "&lng=" + encodeURIComponent(latlng.lng());
            if( id !== undefined )
        postdata += "&id=" + encodeURIComponent(id);
            postdata += "&icontype=" + encodeURIComponent(num);
            postdata += "&userid=" + encodeURIComponent(utype);
            postdata += "&flag=" +encodeURIComponent(flag);

      PostUrl( "write.php", postdata, function(newID){id=newID;} );

       //alert("削除しました！")
       alert("delete compleate ! ");

            pulld1(); //再描画する関数

              }

form.appendChild(textarea);
form.appendChild(button);
 form.appendChild(button2);
```

```
  if( id === undefined )              // 新規作成マーカーは
    GEvent.trigger(mk, "click"); // クリックされたらすぐ開く
}

//場所の簡易移動（検索で全文 Hit 時に移動、でもって精度の低い検索・・・・）
function moveTo(place) {
  var gls = new GlocalSearch();
  gls.setSearchCompleteCallback( null, moveToThePlace );
  gls.setCenterPoint(map); // 現在地図で表示されている付近を重点的に検索
  gls.execute(place);

  function moveToThePlace() {
    if( !gls.results || gls.results.length==0 )
      alert( "[ " + place + " ]    is    Not   Found ! "); //は見つかりません!" );
    else {
      var p = gls.results[0];
      map.setCenter( new GLatLng(p.lat, p.lng) );

       //検索で Hit した地点を↓アイコンで表示する
      var icon = new GIcon();
      icon.image      = "marker/arrows-rectangle_010.gif";   // マーカー画像の読み込み
      icon.iconSize    = new GSize(14, 30);                  // マーカーサイズの指定
      icon.shadow="http://www.google.com/mapfiles/shadow50.png"; //影画像の読み込み
      icon.shadowSize = new GSize(37, 34);                   // 影サイズの指定
      icon.iconAnchor = new GPoint(13, 20); // 画像の座標位置を指定
      icon.infoWindowAnchor = new GPoint(10,3);

       var mrk = new GMarker( new GLatLng(p.lat, p.lng) ,icon );
       map.addOverlay(mrk);
    }
  }
}

//抽出ボタンの処理
function pulld1(){
```

```
//①アイコンの種類を Form より取得
var sel1 = document.getElementById("pointtype");
num = parseInt(document.getElementById("pointtype").value);
//alert(sel.value+" set up !");

//②ユーザの種類を Form から取得
var sel2 = document.getElementById("usertype");
utype = parseInt(document.getElementById("usertype").value);
//alert(sel.value+" change compleate !");

//いったん画面をクリアーする
map.clearOverlays();

if(num!=1 && utype!=1){
  //「特定のユーザ」と「特定の種類のアイコン」を出力する処理
  GDownloadUrl( "list.php", function(listdata){
 var markers = listdata.parseJSON();
 if(markers)
   for(var i=0; i<markers.length; ++i) { // それぞれについて…
     var markerID = markers[i];
     // read.php でマーカー情報を取得(GET)
     GDownloadUrl( "read.php?id=" + encodeURIComponent(markerID),
function(mkdata){
       // 地図にマーカーを追加
       var mk = mkdata.parseJSON();

                  if( mk.flag == 0 && (mk.icontype == num &&
mk.userid==utype) ){
                num = mk.icontype;
                    utype = mk.userid;
            addEditableMarker( mk.id, new GLatLng(mk.lat, mk.lng), mk.data );
                  }
     });
  }
});
```

```
        }else if(num==1 && utype!=1){
          //特定ユーザにおけるすべての種類のアイコンを出力する処理
          GDownloadUrl( "list.php", function(listdata){
         var markers = listdata.parseJSON();
         if(markers)
           for(var i=0; i<markers.length; ++i) { // それぞれについて…
             var markerID = markers[i];
             // read.php でマーカー情報を取得(GET)
             GDownloadUrl(    "read.php?id="    +    encodeURIComponent(markerID),
function(mkdata){
               // 地図にマーカーを追加
               var mk = mkdata.parseJSON();
                    if( mk.flag == 0 && mk.userid == utype ){
                        num = mk.icontype;
                  addEditableMarker( mk.id, new GLatLng(mk.lat, mk.lng), mk.data );
                        }
                });
           }
         });

        }else if(num!=1 && utype==1){
          //特定のジャンル(icontype)を満たすものだけを出力する処理
          GDownloadUrl( "list.php", function(listdata){
         var markers = listdata.parseJSON();
         if(markers)
           for(var i=0; i<markers.length; ++i) { // それぞれについて…
             var markerID = markers[i];
             // read.php でマーカー情報を取得(GET)
             GDownloadUrl(    "read.php?id="    +    encodeURIComponent(markerID),
function(mkdata){
               // 地図にマーカーを追加
               var mk = mkdata.parseJSON();
                    if(mk.flag == 0 && mk.icontype == num ){
                          num = mk.icontype;
                       utype = mk.userid;
```

```
                    addEditableMarker( mk.id, new GLatLng(mk.lat, mk.lng), mk.data );
                        }
                });
              }
            });

          }else{
                //「すべてのユーザ」と「すべての種類のアイコン」を出力する処理
            GDownloadUrl( "list.php", function(listdata){
            var markers = listdata.parseJSON();
            if(markers)
              for(var i=0; i<markers.length; ++i) { // それぞれについて…
                var markerID = markers[i];
                // read.php でマーカー情報を取得(GET)
                GDownloadUrl(  "read.php?id="  +  encodeURIComponent(markerID),
function(mkdata){
                  // 地図にマーカーを追加
                  var mk = mkdata.parseJSON();
                        if( mk.flag == 0 ){
                      num = mk.icontype;
                    addEditableMarker( mk.id, new GLatLng(mk.lat, mk.lng), mk.data );
                        }
                  });
              }
            });
          }

    }

    function pulld3(){
        //アイコン種別の切り替え処理
        var sel = document.getElementById("pointtype");
        num = parseInt(document.getElementById("pointtype").value);
       if(num==1)
         num=10;
    }
```

```
function pulld4(){
        //ユーザ種別の切り替え処理
        var sel = document.getElementById("usertype");
        utype = parseInt(document.getElementById("usertype").value);
       if(utype==1)
          utype=10;
}

onload     = startUp;
onunload = GUnload;
onresize = function(){map.checkResize();}
```

書き込みモジュール（write.php）

```
<?php
  ini_set( "mbstring.script_encoding", "utf-8" );
  ini_set( "default_charset", "utf-8" );
  require_once "Jsphon.php";

  // markerdata ディレクトリのファイル名一覧を取得
  chdir("markerdata");
  $markerlist = glob( "*" );

  // マーカー名を、id という名前の変数でうけとる
  $id = $_REQUEST["id"];

  // $id がディレクトリ内のファイルでなければ、
  // 新しいマーカーと判断して、新規 ID を割り当てる
  $cnt = array_count_values($markerlist);
  if( $cnt[$id] != 1 )
    $id = count($markerlist); // ID は 0 からの連番とし、まだ使われてない値

  // ファイルを開き、JSON 形式で書き込み
  $wp = fopen($id, "w");
    $s = Jsphon::encode( array(
      "id"     => $id,
      "lat"    => $_REQUEST["lat"],
      "lng"    => $_REQUEST["lng"],
      "data" => $_REQUEST["data"],
       "icontype" => $_REQUEST["icontype"],
       "userid" => $_REQUEST["userid"],
       "flag" => $_REQUEST["flag"]
    ));
  fputs($wp, $s);
  fclose($wp);

  // 書き込んだマーカーの ID を返す
  print $id;
?>
```

読み込みモジュール（read.php）

```
<?php
  ini_set( "mbstring.script_encoding", "utf-8" );
  ini_set( "default_charset", "utf-8" );

  // markerdata ディレクトリのファイル名一覧を取得
  chdir( "markerdata" );
  $markerlist = glob("*");

  // マーカー名を、id という名前の変数でうけとる
  $id = $_REQUEST["id"];

  // $id がディレクトリ内のファイルであることを確認して、中身を出力
  $cnt = array_count_values($markerlist);
  if( $cnt[$id] == 1 )
    print file_get_contents( $id );
?>
```

参照モジュール（list.php）

```
<?php
  ini_set( "mbstring.script_encoding", "utf-8" );
  ini_set( "default_charset", "utf-8" );
  require_once "Jsphon.php";

  // markerdata ディレクトリのファイル名一覧を取得
  chdir( "markerdata" );
  $markerlist = glob("*");

  // JSON 形式にして返す
  print Jsphon::encode( $markerlist );
?>
```