平成 24 年度卒業論文
論文題目

Boid を用いた
クラウド内のファイル検索システム

神奈川大学　工学部　電子情報フロンティア学科
学籍番号　200902854
鈴木　健人

指導担当者　木下宏揚　教授

目 次

図 目 次

第1章　序論

1.1　研究の背景

近年、クラウドコンピューティングが普及し自宅のパソコンや会社のネットワーク上にあるサーバではなく、インターネット上のサーバを利用して顧客管理のような企業の業務アプリケーションから、Gmail に代表されるメール・サービスやファイルを保存するストレージ・サービスのような個人向けのものまで様々なサービスがありその中でもストレージ・サービスではクラウド内のデータ量が大量になってきており、これによって大量にあるファイルの中から重要なファイルが見つけづらくなっている。また計算機やインターネットの進化により、巨大なデータを扱う機会が増えてきたが、大量のデータの中から必要な情報を得るのは簡単なことではない。必要なデータを自動的に抽出するアルゴリズムはいろいろあるが、データを自分の目で見て確かめつつ情報を取り出す方法は、直感的であるだけでなく、自動的な手法よりも有効な場合も多いようである。

1980 年代後半、Xerox PARC の Stuart Card らは、大量の情報を人間にわかりやすい形で提示して人間に判断を促す情報視覚化 (Information Visualization) という考え方を提唱した [1][2][3][4][5]。天気の予測に代表されるような、科学的シミュレーションの結果を 3 次元画像として表示する視覚化手法はサイエンティフィックビジュアライゼーションと呼ばれ、現在でも研究が行われていて、地図や位置に関連する情報を視覚化して利用する地理情報システム (Geographic Information System : GIS) のようなシステムも近年広く利用されつつある。サイエンティフィックビジュアライゼーションや GIS のような視覚化システム、実世界の 2 次元/3 次元の対象を扱っているため、計算対象の座標をそのまま拡大/縮小して表示すればよいのに対し、一般的な情報視覚化システムは、Web のような大規模なネットワーク構造や、巨大な木構造のファイルシステムのような、現実の 3 次元空間上に存在しないデータ構造を扱うため、データ構造を図形として表現する手法/コンピュータの画面上にマッピングする手法/視点や表示方法を対話的に変える手法などを駆使した視覚化手法が必要になる。このような手法を駆使することによって詳細にデータを吟味したり全体的構造を把握したりすることができる様々な情報視覚化システムが研究されてきた。

従来はファイルの振る舞いを記述するシステムはなかった。このようなシステムに対応するのがファイルシステム [6] である。ファイルシステムとは、コンピュータのリソースを操作するためのオペレーティングシステムが持つ機能の一つで、抽

象データ型の集まりであり、ストレージ、階層構造、データの操作・アクセス・検索のために実装されたものであるため、ファイルの振る舞いを表現できない。この振る舞いの概念を Boid の Separation・Alignment・Cohesion をベースとする群知能とみなす。そして、このファイルの振る舞いに、ある場所へ集まるという行動を加え群れを動かす。また、その時に集まりやすいファイルと集まりにくいファイルが存在するようにすると、群れの中で優先となるファイルができるので、ファイルシステムにおいて検索しやすい新たなシステムになるものと考え、Android を用いて実装することを試みた。Andoid とはスマートフォン・タブレット PC などのモバイルデバイス向けのオープンでフリーなソフトウェアプラットフォームであり、クラウドの端末としてとても重要なシステムである。

1.2　研究の目的

本研究は、大量にあるファイルを視覚化することによってより見やすく、検索しやすいファイル検索システムを提案することを目的としている。また、情報を視覚化してファイル検索を行い易くするための方法として、以下の 2 つを組み込み研究を行う。

- 大量にあるファイルを一つの群れとして考え、なおかつ群れの中での位置としてファイルに重要性を出し重要なファイルほど群れの中心に集まるようにする。
- ファイルの量が膨大過ぎると視覚化しても見づらい可能性があるので、ファイルを表示する際にフィルタをかけファイルの表示を限定させる。

1.3　本論文の構成

本論文は、全 5 章から構成される。
第 1 章では、本研究の背景、目的と本論文の構成を述べる。第 2 章では、クラウド・コンピューティング、Boid、Android、Processing についての基礎知識を述べる。第 3 章では、Boid を用いて群れを作り、目的のファイルに行動を取らせるプロセスとフィルタについて述べ、第 4 章では、開発の流れと結果を述べ、第 5 章では、結論を述べている。

第2章　基礎知識

2.1　クラウド・コンピューティング

アメリカ国立標準技術研究所によるクラウドの定義は、クラウド・コンピューティングとはモデルであり、構成変更が可能なコンピューティング資源（例：ネットワーク・サービス、ストレージ、アプリケーション・サービス）の共有プールをオンデマンドなネットワークアクセスで可能にする。それはわずかな管理の手間、もしくはサービスプロバイダとのやりとりによって迅速に準備され、提供される。このクラウドモデルは可用性を促進するものであり、5つの本質的な性質と3つのサービスモデル、そして4つの配備モデルから構成される。

2.1.1　本質的な特徴

- オンデマンドセルフサービス
  利用者は自身で、コンピューティングの能力、すなわちサーバの利用時間やネットワークストレージといったものをサービスプロバイダの人手を介することなく必要に応じて準備できる。

- 幅広いネットワーク経由のアクセス
  クラウドの能力はネットワーク経由で利用でき、標準的な機構を持つさまざまなシンクライアントやそうでないクライアント（携帯電話やラップトップPC、PDAなど）からアクセスされる。

- リソースプール
  クラウドを提供するプロバイダのコンピューティングリソースは、マルチテナントモデルを用いて複数の利用者へ提供すべくプールされている。それは利用者の要求に従って物理的にあるいは論理的に区別されたリソースをダイナミックに割り当て、あるいは再割り当てする。利用者は提供されているリソースが物理的にどこにあるかは一般に認識せず、関知も管理もしないが（国や州あるいはデータセンターなどの）抽象化の高いレベルにおいてその位置を特定することができるものもある。リソースの例としては、ストレージ、計算処理、メモリ、ネットワーク帯域幅、そして仮想マシンなどがある。

- 迅速な伸縮性
  クラウドの能力は迅速に伸縮する。あるときにはそれは自動的に行われ、す

ばやくスケールアウトし、あるいはスケールインのためにすぐに解放される。利用者にとって、その能力はほとんど無制限であり、いつでも、いくらでも購入できるように準備されているように見える。

- 計測されるサービス
  クラウドのシステムは自動的に管理され、リソース利用の最適化が行われる。それはいくつかのサービスの種類（ストレージ、計算処理、帯域幅、アクティブなユーザーアカウントなど）に応じた適切な抽象レベルの計測能力による。リソースの利用は監視、管理され、利用されたサービスは顧客とプロバイダの双方に対して透過的にレポートが提供される。

2.1.2　サービスモデル

- SaaS（Software as a Service）
  クラウドの能力は、クラウドインフラストラクチャの上で実行されるアプリケーションとして利用者に提供される。アプリケーションはWebブラウザのようなシンクライアントインターフェイスを通じて、さまざまなクライアントデバイスからアクセスできる。利用者は、ユーザーに特化したアプリケーションの設定以外は、クラウドインフラストラクチャであるネットワークやサーバ、OS、ストレージなどを管理しないが、アプリケーションのデプロイもしくはアプリケーションをホスティングする環境の構成を制御することはできる。

- PaaS（Platform as a Service）
  クラウドの能力として提供されるのは、プログラミング言語やツールによって開発、もしくは取得したアプリケーションをクラウド上でデプロイすることである。利用者はクラウドインフラストラクチャとしてのネットワークやサーバ、OS、ストレージなどを管理しないが、アプリケーションのデプロイもしくはアプリケーションをホスティングする環境の構成を制御することはできる。

- IaaS（Infrastructure as a Service）
  クラウドの能力として提供されるのは、利用者がデプロイし適切にソフトウェアを実行するための処理能力、ストレージ、ネットワークそのほかの基本的なコンピューティングリソースの事前準備などでありOSやアプリケーションなども含むことがある。利用者はクラウドインフラストラクチャを管理しないが、OS、ストレージ、デプロイされたアプリケーションなどの制御や限定された範囲のネットワークコンポーネント（ファイアウォールなど）の制御が可能。

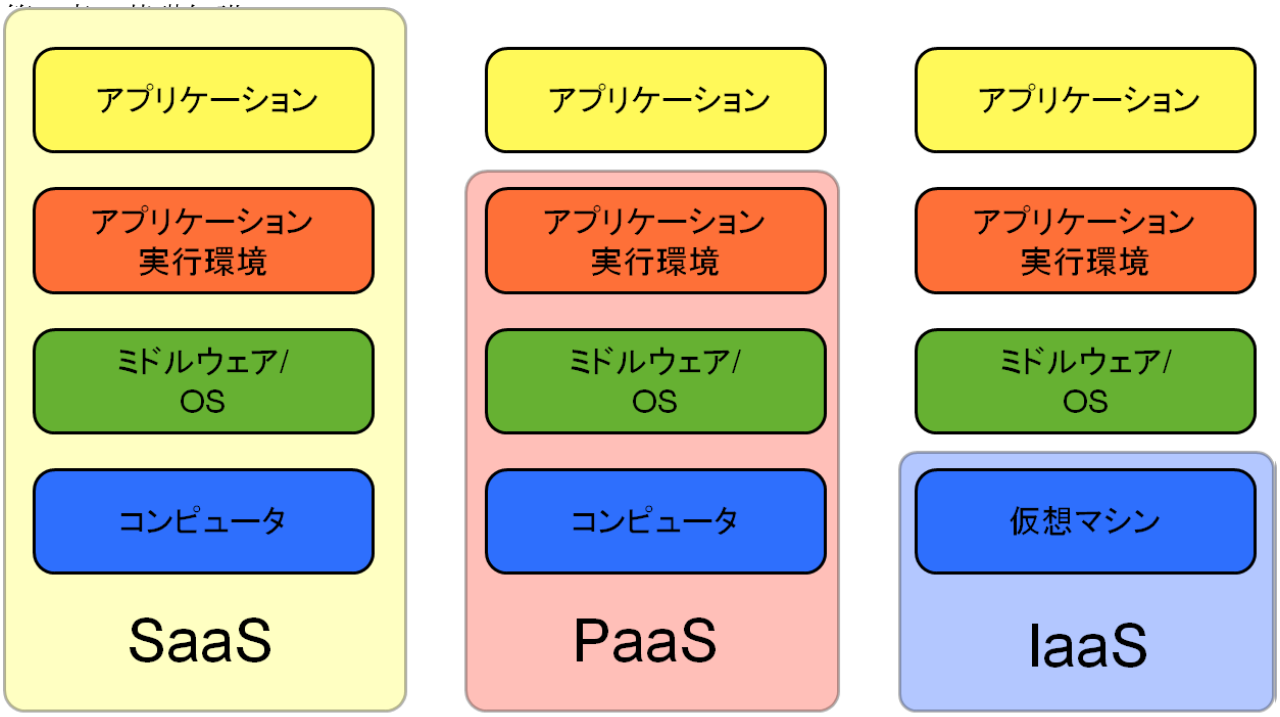
アプリケーション
アプリケーション
実行環境
ミドルウェア/
OS
コンピュータ
SaaS
アプリケーション
アプリケーション
実行環境
ミドルウェア/
OS
コンピュータ
PaaS
アプリケーション
アプリケーション
実行環境
ミドルウェア/
OS
仮想マシン
IaaS

図 2.1: サービスモデル

2.1.3　デプロイメントモデル

- プライベートクラウド
  クラウドインフラストラクチャーは単独の組織によって運用される。その組織、あるいはサードパーティによって管理され、オンプレミスもしくはオフプレミスとして設置される。

- コミュニティクラウド
  クラウドインフラストラクチャーは幾つかの組織によって共有され、同じ意識（ミッション、セキュリティ、要件、ポリシー、コンプライアンス）を持つコミュニティをサポートする。その組織、あるいはサードパーティによって管理されオンプレミスもしくはオフプレミスとして設置される。

- パブリッククラウド
  クラウドインフラストラクチャーは誰でも、あるいは広い範囲の業種に対して利用可能となる。クラウドサービスを販売する組織によって所有される。

- ハイブリットクラウド
  クラウドインフラストラクチャーが、2 つかそれ以上のクラウド（プライベート、コミュニティ、あるいはパブリック）から構成されるもの。個別の実体を残しつつも、標準もしくは独自技術によるデータやアプリケーションの可搬性（クラウド間のロードバランスなど）によって結合されている。

2.2　Boid

Boid とは 1987 年にアメリカのアニメーション・プログラマであるクレイグ・レイノルズによって考案・作製された人工生命シミュレーションプログラムである。Boid というモデル名は、鳥もどきという意味の言葉である“ bird - android（バード・アンドロイド）”が短くなったことに由来している。Boid の群れを実現させる振る舞いは、３つの要素からなり、「衝突の回避」、「速度を合わせる」、「群れの中心に向かう」といった３つのルールを規定するだけで鳥の群れをシミュレーションすることができる。[7]

- Separation（分離）: 近くにいる仲間と衝突しないようにする。

- Alignment（整列）: 近くの仲間と速度を一致させようとする。

- Cohesion（結合）: 近くにいる仲間に周りを囲まれた状態になろうとする。

群れの一連の動きは、自分自身と仲間との間の距離を最適に保とうとするルールを重要視している。このような 3 つの単純な行動規範をそれぞれの個体が持ち、全体として複雑な群れの行動が創発する。衝突回避のために、boid はそれぞれ自分にとっての「最適距離」を持っている。自分の最も近くにいる仲間との間で、この距離を保とうと振る舞う。また、速度を合わせるために、最も近くにいる仲間と平行に（同じベクトルで）移動する。これによるスピードの変化はない。さらに、群れの中心（boid 全体の集合の重心）に向かうようにも速度を常に変更している。

図 2.2: Separation・Alignment・Cohesion

2.3　Android

Android とは、スマートフォン・タブレット PC などのモバイルデバイス向けのオープンでフリーなソフトウェアプラットフォームである。ここで言うソフトウェアプラットフォームとは、アプリケーションやアプリケーションフレームワークのみではなく OS やミドルウェアなども含んだ広範にわたるソフトウェアの集合体を指す。このことから、Android ではソフトウェア開発は大きく 2 つの視点に分けて考えることができる。

- ある端末の上で動作する Android プラットフォームを作る
  Android はソフトウェアの集合体であるため、最終的にはあるハードウェアの上に載せて動かす必要がある。Android とハードウェアを結び付けていく作業のことをポーティング（porting）と呼ぶ。この領域の開発を行うには、特に Linux カーネルやデバイスドライバに関する知識が必要不可欠。アプリ

ケーションの開発者は、Android プラットフォームを直接操作することはなく、アプリケーションフレームワークの機能を通じて利用する形になる。

- Android プラットフォームの上で動作するアプリケーションを作る
  開発者は「Android SDK」として提供されているアプリケーションフレームワークの機能や開発ツールを用いてアプリケーションを開発する。Android でアプリケーションを開発するためには、Java やアプリケーションフレームワークに関する知識が必要になる。

2.3.1　Android のアーキテクチャ

アーキテクチャは大きく 4 つの層、5 つの領域に分かれている。

- Linux カーネル層
  最下層となる「Linux カーネル」は、バージョン 2.6 を元にモバイルデバイス向けに変更が加えられている。システムサービスをはじめとしたアプリケーションを実行するために必要な基本的な機能を提供する、いわば基礎となる層。他の 3 つの層で動作する機能は、この Linux カーネル上で動作する。

- ライブラリ層
  「Linux カーネル」層の上位層は、「ライブラリ」層となる。Android は、さまざまなコンポーネントから使用されるライブラリを提供する。ライブラリは、C や C++言語で作成されたライブラリを含む。これらの機能は、基本的にアプリケーション開発者が直接使用することはなく、上位層であるアプリケーションフレームワーク層の機能を通じて使用されることになる。

- Android ランタイム
  「ライブラリ」層と同レベルの機能として「Android ランタイム」がある。Android ランタイムは Java 言語に準拠するコアライブラリと Android アプリケーションを実行する Dalvik 仮想マシンにより構成されている。

- アプリケーションフレームワーク層
  「ライブラリ」層の上位層は、「アプリケーションフレームワーク」層になる。アプリケーション開発者は、SDK にあらかじめ含まれているアプリケーションで使用されているものと同じクラスを使用することができる。アプリケーションを構築する際に使用するアーキテクチャは、コンポーネントの再利用が簡単にできるように設計されている。

- アプリケーション層
  最上位の層が「アプリケーション」層。SDK には、電話や Web ブラウザなどの実行可能なアプリケーションがあらかじめ含まれている。アプリケーション開発者が作成したさまざまなアプリケーションもこの層に位置づけられる。

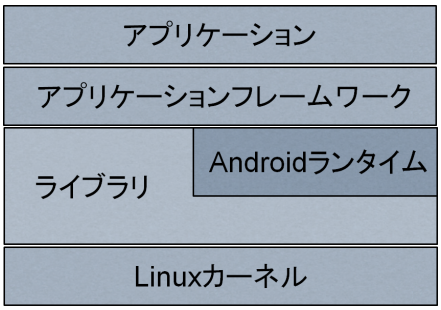

図 2.3: Android のアーキテクチャ

2.3.2　アプリケーション開発者の視点から見た Android の特徴

- Java 言語を使用してアプリケーションを作成する
- サーバアプリケーション開発で使用されているものに近いアプリケーションフレームワークが提供されている。
  Web アプリケーションを構築する際によく使用される MVC2 パターンの要素（Model、View、Controllor の 3 つの要素）を Android アプリケーションの構成要素に対応づけることができる。
- Java 言語と他の言語の使い分けをしている。
  ライブラリ層には、モバイルデバイスに含まれるハードウェアを制御するためのソフトウェアが含まれている。この部分については C/C++言語などの Java 以外の言語で記述されている。Java 言語は、C/C++言語と比較した場合、実行速度やタイミング制御、メモリの管理などを不得意としている。なので、Android はそうした不得意な箇所には適切な言語で実装を行い、アプリケーション開発者には意識させないクラスのインターフェイスを提供している。

2.3.3　アプリケーション開発の流れ

開発作業の流れ [8] は、・プロジェクトの作成・ソースコードの作成・アプリケーションの実行の 3 つに分けることができる。
・プロジェクトの作成 ··· プロジェクトはアプリケーションの単位で作成する。
・ソースコードの作成
・アプリケーションの実行 ··· アプリケーションが完成したら、Android エミュレータ上でアプリケーションを実行する。その手順は以下の通り。

1. Java プログラムのコンパイル
2. Java バイトコード（.class）を Android バイトコード（.dex）に変換
3. Android アプリケーションをパッケージング（.apk ファイルの作成）

4. Android エミュレータの起動
5. アプリケーションを Android エミュレータにインストール
6. アプリケーションを Android エミュレータ上で起動

上記の作業が正常に完了したら、Android エミュレータ上でアプリケーションが動作する。

2.3.4　アプリケーションを構成する要素

Android では、クラスとして提供されるコンポーネント要素（API）[9][10][11]を開発者が組み合わせることで 1 つのアプリケーションを作成する。このクラスには 2 つのタイプがある。

- Android アプリケーションとして動作するために必要なクラス

- Android アプリケーションの機能を提供するクラス

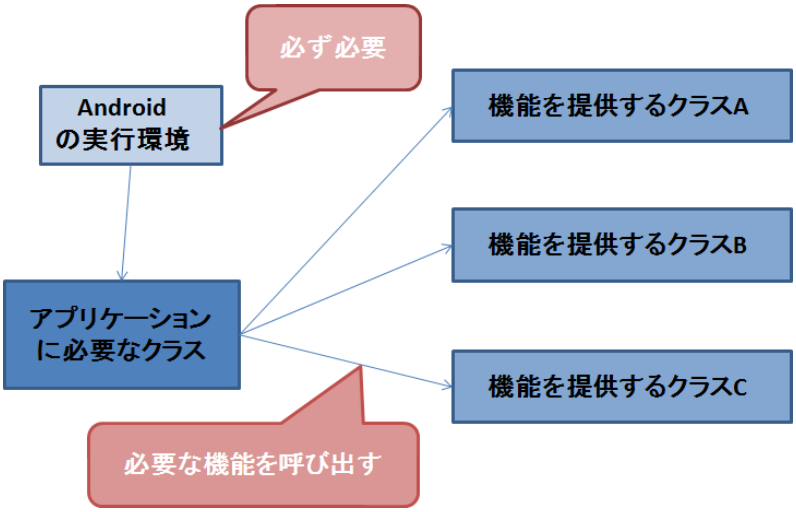

図 2.4: Android アプリケーションを構成するクラス

Android の実行環境は、システム上にアプリケーションが必要となるクラスに紐づけたプロセスを作り、クラスの呼び出しを通じてアプリケーションの実行を制御する。この特別な役割を持ったクラスには 4 つの要素がある。

- アクティビティ
  ユーザーとアプリケーションとの間で行われるやり取り全般を仲介する。
- インテント
  アプリケーションの実行時に各要素を紐づける。
- サービス
  ユーザーの画面に対する操作に依存することなく処理を実行する。(バックグラウンド)
- コンテントプロバイダ
  アプリケーション間で情報を共有する。

アプリケーションの開発において、これらの要素が一度にすべて必要になるわけではない。たとえば、ユーザーが画面に対して操作を行うような標準的なモバイルアプリケーションでは、アクティビティを利用してアプリケーションを構築する。また、画面上に行う必要のない処理、いわゆるバックグラウンド処理と呼ばれるようなタイプのアプリケーションでは、サービスのみを使用するといった具合である。

Android アプリケーションとして機能を提供するためのクラスには、モバイルデバイスに装備されたハードウェアを利用するためのクラスを含む。また、アプリケーションを作成するための基本的な機能、たとえば画面制御やデータ制御などの機能も提供させる。さらには、Java 言語のレベルで提供されるクラスも広い意味ではこのタイプに含めてよい。

2.4　Java

2.4.1　Java とは

Java は、1995 年に Sun Microsystems から初めてリリースされたプログラミング言語およびコンピューティングプラットフォームであり、ユーティリティ、ゲーム、ビジネスアプリケーションなど、最先端のプログラムの基礎となっているテクノロジーである。Java は、世界中の 8 億 5000 万台を超える個人用コンピュータや、世界中の何十億台ものデバイス (モバイルデバイスや TV デバイスなど) で動作している。[12]

2.5　Processing

2.5.1　Processing とは

Processing とは、Casey Reas と Ben Fry によるオープンソースプロジェクトであり、かつては MIT メディアラボで開発されていた。電子アートとビジュアルデザインのためのプログラミング言語であり、統合開発環境 (IDE) である。視覚的なフィードバックが即座に得られるため、初心者がプログラミングを学習するのに適しており、電子スケッチブックの基盤としても利用できる。Java を単純化し、グラフィック機能を特化した言語 [13][14] といえる。

Java をベースにした実行環境とエディタが用意されていて、Windows、MacOSX、Linux 版ソフトが無償で提供されている。アプリケーションとしては、Android や Web 上で動作させることもできる。

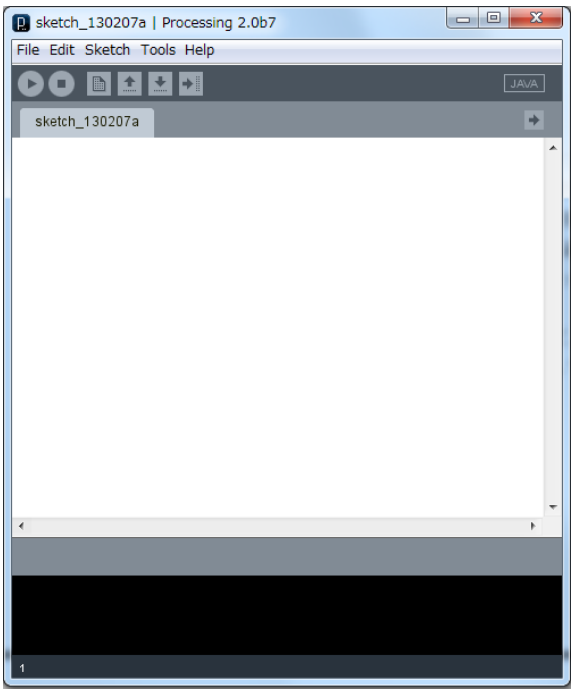

図 2.5: Processing の IDE

第3章　ファイル検索システムの要点

3.1　目的

従来のファイルシステム (木構造) では、ファイルをあるフォルダ内のフォルダ内のさらにフォルダ内に保存をしている場合など、探しているファイルがどのフォルダに入っているか忘れてしまうことがある。また、クラウド上でもファイルが大量となっているこれを情報を視覚化することによって見やすく、検索しやすいファイルシステムを新たに提案することを目的とする。そこで以下のような特徴をもつファイルシステムを提案する。

- クラウド内のファイルを一種の群れとして扱う。
- ファイルに重要性を持たせ、重要度が高いファイルを群れの中心に集める。
- フィルタ機能を持ち、ファイルの表示を限定させる。

3.2　Boid を用いた群知能

クラウド内に存在する大量のファイルを視覚的に見やすくするためにファイルを群れとして扱う。そのために Boid の Separation、Alignment、Cohesion をベースとする群知能によって実現する。
実際に Boid を組み込んだものを Boid のアプリケーションとして、図 3.1 に示す。

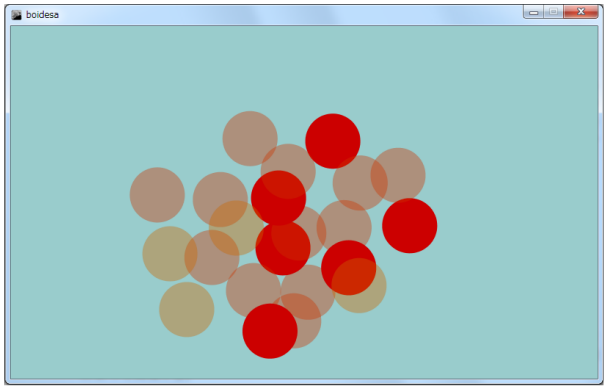

図 3.1: Boid のアプリケーション

3.2.1　群れを作るアルゴリズム

群れを作るアルゴリズムとして、ファイルごとに、Separation、Alignment、Cohesion の視野範囲（どのファイルも同じ範囲）を持つ。このファイルが持つ Separation・Alignment、Cohesion の視野範囲を図 3.2 に示す。そして自分に最も近いファイルとの距離を出し、その距離が Separation、Alignment、Cohesion の視野範囲内であったらいずれかの行動をして群れを作り出す。[15]

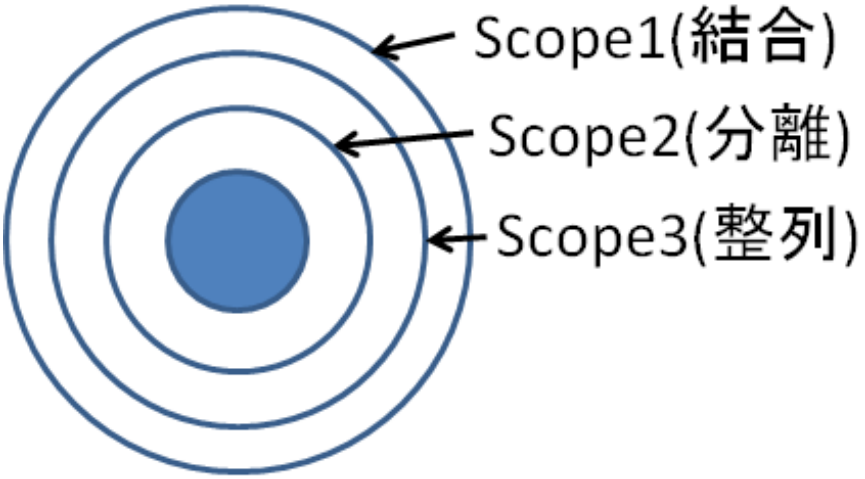

図 3.2: Boid の視野範囲

3.2.2　餌追いアルゴリズム

Boid を用いれば群れを作ることはできるが、それだけではただ群れるだけになってしまう。そこで群れに行動を取らせるアルゴリズムとして餌をまき、餌の所へ群れが集まってくるようにすることで群れが動くという行動をとらせることができる。餌を追うアルゴリズムを図 3.3 に示す。

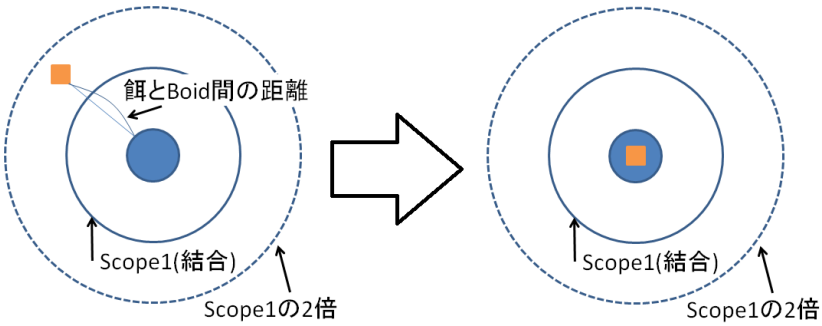

図 3.3: 餌を追うアルゴリズム

これは、Boid と餌の間の距離を出し、scope1 の 2 倍の範囲内に餌があれば、Boid と餌の距離が 0 になるまで処理を繰り返すことによって餌を追っている。

3.3　重要度

ファイルに重要度を与える。これによって、ファイルごとに重要度が高いファイルと低いファイルができ、重要度が高いファイルを群れの中心に来るように行動させることで一目で重要なファイルを検索することができる。

3.3.1　重要度の定義

複数の重要度を定義する。

- 使用頻度
  使用回数が多く、かつ閲覧日が新しいファイルほど重要度が高いとする。
- ノルマ
  ノルマが設定されている仕事に対して、ノルマが達成されていない場合は重要度が高いとする。
- 時系列
  仕事や学校での、課題などの締め切りを設定する。締め切りが近いファイルが重要度が高いとする。

3.3.2　重要度の選択

何を重要としているかユーザーによって異なるので、ファイル 1 つ 1 つに重要度をいくつか持たせそれを選択できるようにする。

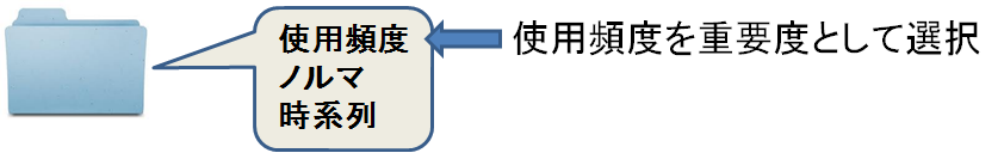

図 3.4: 重要度の選択

3.3.3　フィルタ

大量のファイルをすべて表示させると見づらいので表示させるファイルを制限する。そのためにフィルタを作り、条件以外のファイルは表示しないようにする。

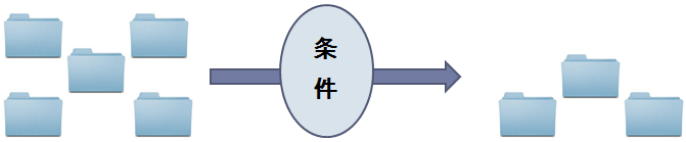

図 3.5: フィルタ

3.4　Boid と重要度と餌追いアルゴリズムの関係

これまで Boid を用いて群れを作り出したが、これだけでは、ファイルが群れるだけであり、これに餌追いアルゴリズムで群れに餌の元へ行くという行動を取らせることで、群れに動きをさせることができるようになる。しかし、これでもまだ群れ全体を動かすことができるだけであり、ファイルの重要性によって群れの中での位置を決めることはできない。そこで、餌を重要度が高いファイルの集合地点として、重要度が高いファイルはそのまま餌追いアルゴリズムに従い、餌の元へ行き、重要度が低いファイルは、餌を検知する範囲を 0 にすることで、餌を検知しないが Boid により群れから離れて分裂することはなく、重要度が高いファイルは群れの中心に集まるものと考えられる。

重要度が高いファイルの場合は、

$$\text{ファイルと餌との距離 } dr = \sqrt{dx^2 + dy^2} \tag{3.1}$$

を出し、その距離が scope1(Cohesion) の 2 倍の距離より小さい場合 (dr<scope1*2) は、

$$\cos\left(\tan^{-1}\frac{dy}{dx}\right) - \tan^{-1}\frac{y}{x} \tag{3.2}$$

式 (3.2) で角度を計算して近づき、if 文で餌との距離が 0 になるまで処理を続ける。重要度が低いファイルの場合 (dr<0) は、検知する範囲を 0 にして式 (3.2) の計算を行わないようにして餌を追わないようにする。
Boid と重要度と餌追いアルゴリズムの関係を図 3.6 に示した。

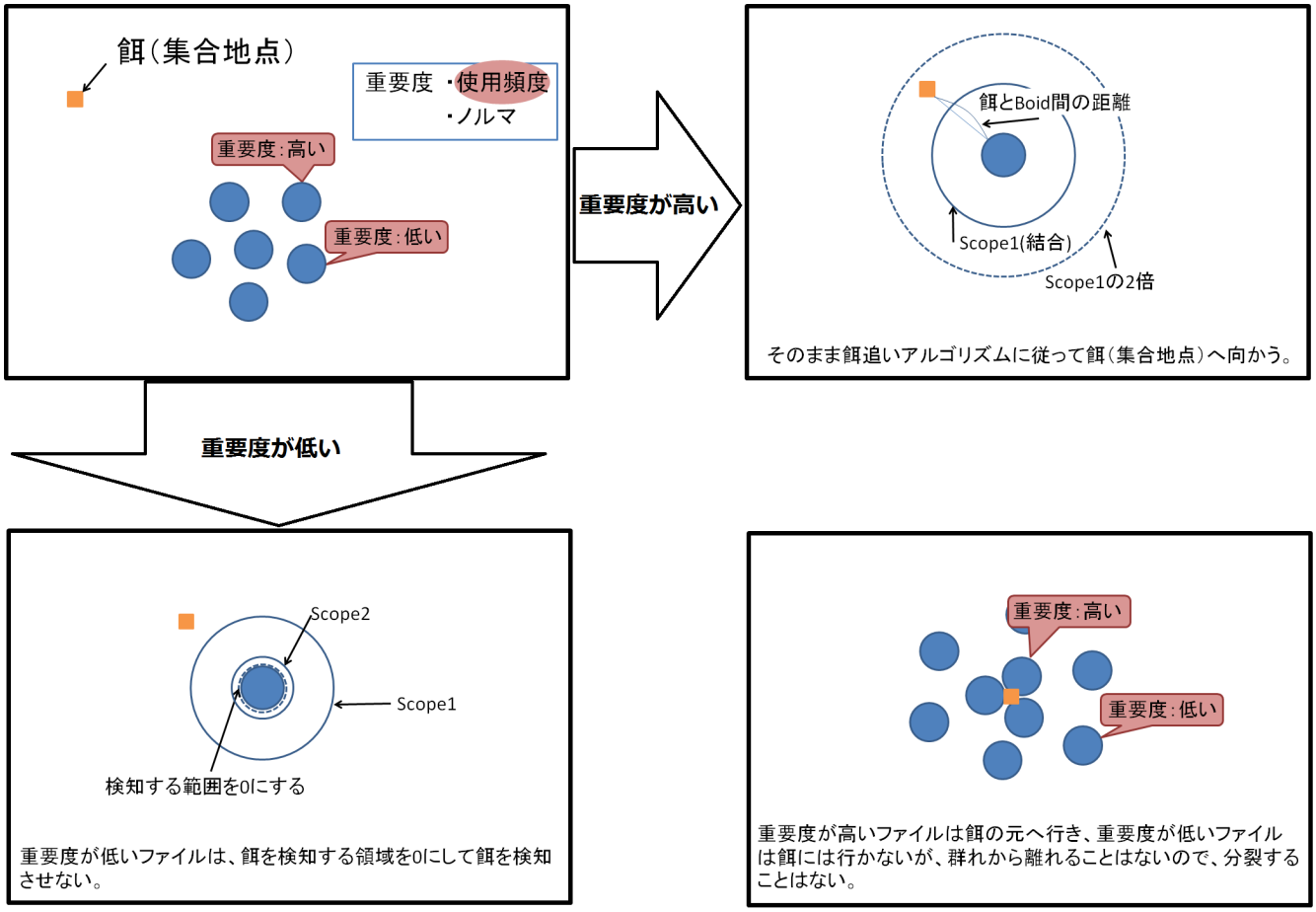

図 3.6: Boid と重要度と餌追いアルゴリズムの関係

第4章　開発

4.1　開発概要

Android 端末を使って開発する。

4.1.1　開発環境

- Java
- Android SDK
- Processing
- Android 端末:「GALAXY S SC-02B」

表 4.1: GALAXY S SC-02B の仕様 [16]

名称	GALAXY S(SC-02B)
OS	Android 2.3
CPU	Samsung S5PC110 1GHz
GPU	PowertVR SGX540
RAM	512MB
ROM	16GB
ディスプレイ	4 インチ Super AMOLED
解像度	VGA(480 × 800)
カメラ	500 万画素
通信	IEEE802.11b/g/n
バッテリー	1500mAh
サイズ (mm)	高さ 122 ×幅 64 ×厚さ 9.9〜12.0
重量	118g

4.2　開発手順

1. 開発環境のセットアップ

開発に必要な Android SDK、Processing を公式サイトから Download をクリックし、Windows 版をインストールする。全て無料でインストールできる。

- Android SDK(Software Development Kit) とは Android のソフトウェア開発キット

2.AVD(Android 仮想デバイス) の作成

AVD とは、エミュレータに設定する端末情報を指定するもので、Android プラットフォームのターゲット情報などによって構成せれている。このエミュレータを使って開発したアプリケーションや端末の振る舞いをエミュレーションさせる。

図 4.1: エミュレーションの起動画面

3. 新規スケッチの作成

Processing を開くとエディタが起動し、ここにプログラムを記述していく。

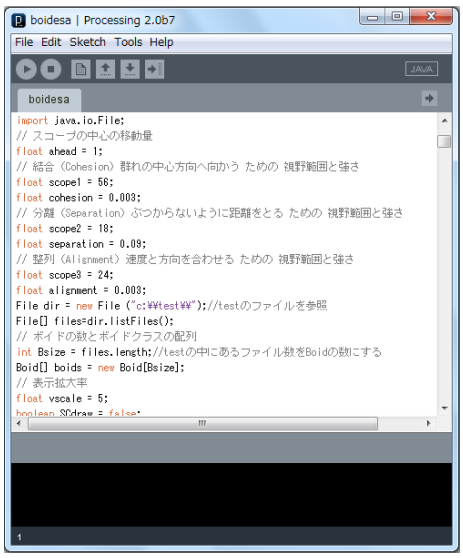

図 4.2: エディタ

4. プログラムを記述

Boid、餌追い、重要度、フィルタをプログラムする。

5. エミュレータでの実行

AVD Manager よりエミュレータを起動させ、アプリケーションを実行する。

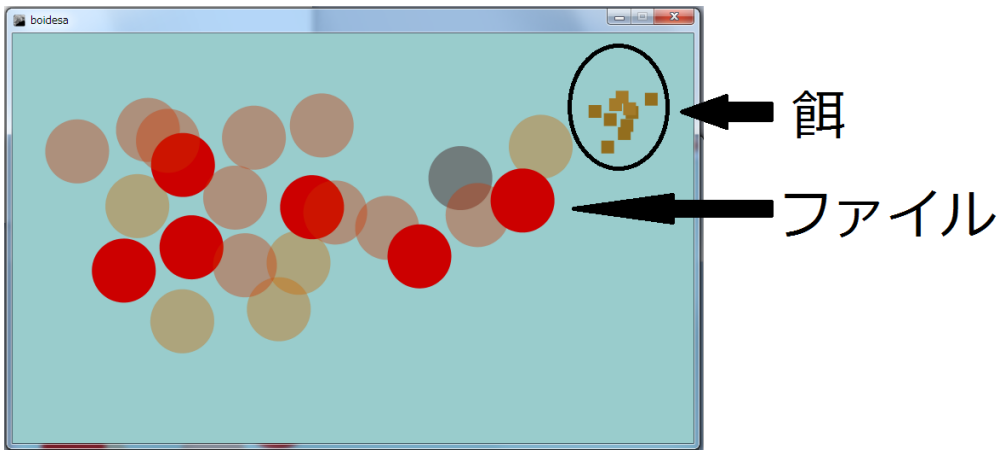

図 4.3: エミュレータでの実行画面

7.Android 端末での設定
Android 端末にアプリケーションをインストールする前に Android マーケット以外からダウンロードしたアプリケーションのインストールを許可する設定を行う。手順は、ホーム画面でメニューボタンを押して「設定」を選択する。「設定」から「アプリケーション」を選択する。「アプリケーション」の「提供元不明のアプリ」をチェックする。

図 4.4:「アプリケーション」の設定画面

8.Android 端末での実行
PC に Android 端末を接続し、アプリケーションを実行する。

4.3　プログラム内容

- setup
  画面のサイズやフレームレートを設定する。Boid の数は test というフォルダを作り、その中のファイル数に応じて変化する。

- Boid
  ファイルの動作（動く方向、速度）や Separation、Alignment、Cohesion の範囲などを決めている。

- Esa
  餌を撒く、餌を追うプログラム。

- display
  ファイルの形や色を表示する（今回は円形）。

4.4　結果

- Boid の Separation、Alignment、Cohesion をプログラムして群れを作り、それに餌を追うアルゴリズムを組み込むことができ、Android 端末上で動かすことができた。

- 餌を撒くことで群れに行動を取らせることはできたが、重要度とフィルタまで組み込むことができなかった。

図 4.5: Android 端末上での実行画面

第5章　結論

本研究では、大量にあるファイルを情報視覚化することによって見やすく、検索しやすいファイル検索システムを新たに提案した。そのシステムはファイルを群れとして扱い、重要なファイルほど群れの中心へ集まる行動を取らせることで、重要であるファイルが一目で分かるような群れを作ることを目標としてBoidを用いて開発を行った。今後の課題として、組み込むことができなかった重要度とフィルタを組み込み、重要度が高いファイルを群れの中心へと集めたり、ファイルの表示を限定させるようにすることが課題である。重要度の選択次第では、ノルマがある仕事がある時などやるべきことが群れの中心に集まってくるので検索しやすくなるだけでなく、作業がより円滑になるものと考える。また、今回の研究ではファイルの色はランダムで決めていたので、色によって群れに新たな行動を取らせることができると考えられ、これらを今後の課題とする。

第6章　謝辞

本研究を行うにあたり、終始熱心に御指導していただいた木下宏揚教授に心から感謝致します。また、様々な面で数多くの有益な助言をしていただいた宮田純子氏に深く感謝致します。さらに、研究活動一般に様々な助言をいただきました南出和宏氏をはじめ公私にわたり良き研究生活を送らせていただいた木下研究室の方々に感謝致します。

2013年　2月
鈴木　健人

参考文献

[1] Ben Fry :” ビジュアライジング・データ　—Processing による情報視覚化手法”オライリー・ジャパン (2008)

[2] Maydene Fisher,Jon Ellis,Jonathan Bruce, ” JDBC API TUtorial and Reference,3rd ed”,Prentice Hall(2003)

[3] Greg Wilson, ” Data Crunching:Solve Everyday Problems Using Java,Python,and More”,The Pragmatic Bookshelf(2005)

[4] Usama Fayyad,Georges G. Grinstein,Andreas Wierse, ” Information Visualization in Data Mining and Knowledge Discovery”,Morgan Kaufmann(2002)

[5] ”Processing によるデータ視覚化” http://www.ibm.com/developerworks/jp/opensource/library/os-datavis2/index.html

[6] 磯村　淳 :” クラウドファイルシステム”(2011)

[7] ” Processing boid 金　魚”http://www.tiu.ac.jp/zohzemi/processing/index.html

[8] ” Android 入門”http://www.javadrive.jp/android/

[9] ” Google Android アプリケーション” http://www.hino.meisei-u.ac.jp/is/saishu/androidjikken5x.htm

[10] 松岡　宣 :” 作りながら覚える Android プログラミング” ソフトバンク　クリエイティブ株式会社 (2012)

[11] 江川　崇、竹端　進、山田　暁通、麻野　耕一、山岡　敏夫、藤井　大助、藤田　泰介、佐野　敏郎 :” Google Android プログラミング入門”株式会社豆蔵 (2009)

[12] 今川　美保 :” かんたん Java”技術者評論社 (2010)

[13] ” Processing によるデータの可視化-時系列のデータを可視化する” http://yoppa.org/bma10/1250.html

[14] ”Processing　ボタンをクリック　1”　http://happy-arduino.blogspot.jp/2012/11/processing.html

[15] 科学シミュレーション :”パソコンで見る複雑系・カオス・量子―シミュレーションで一目瞭然！”講談社 (1997)

[16] ”GALAXY S SC-02B” http://www.samsung.com/jp/latest-home

第7章　質疑応答