平成 23 年度卒業論文

論文題目

クラウドファイルシステム

神奈川大学 工学部 電子情報フロンティア学科
学籍番号　200802949
磯村　淳

指導担当者　木下宏揚　教授

目 次

図 目 次

第1章　序論

インターネットが普及し、近年の社会ではクラウド・コンピューティングが注目されている。中でも世の中はデータセンターとサービスとしてのクラウドソフトウェア（SaaS）を組み合わせたビジネスモデルである。我々はクラウドを多様で不確実にインターネット上に存在する情報リソースと人間の「関係性」、そしてその「振る舞い」とみなしている。そこで、情報リソースをPC内のファイルとみなし、そのファイルの振る舞いを群知能によって実現することによって、他者との連携により人間の創発的活動（学術的活動・ビジネス・地域ボランティアなど）を刺激・支援するシステムを研究の目的としている。

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。私は、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、このファイル振る舞いを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。

キュレーションという言葉がある。キュレーションとは、博物館や図書館の学芸員を「キュレーター（curator）」という。「情報を収集し、選別し、意味付けを与えて、それをみんなと共有すること」という定義。情報の洪水の中で、それぞれのユーザーにとって必要なものをコンテクスト（意味の文脈）に沿ってピックアップし、整理し、新しい付加価値を与えることである。クラウド内の大量にあるファイルの種類を自分なりにキュレーションし、「ファイルの見える化」を行い、ユーザーにとって見やすいファイルシステムを作ることがこの研究の目的である。

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（National Institute of Standards and Technology：NIST）によってクラウド・コンピューティングの定義を定めている。
クラウドコンピューティングとは、コンフィグレーションが可能なコンピューティングリソース（ネットワーク／サーバー／ストレージ／アプリケーション／サービス）で構成され、分散されたコンピューティングリソースへのオンデマンドのネットワークアクセスを可能にする。利便性の高いモデルのことである。そして、それらのリソースは、最小の管理手順もしくは、サービスプロバイダーとのやりとりにより、迅速に供給され、また、解消されるものとなる。このクラウドモデルは、5つの本質的な特徴と、3つのサービスモデル、そして4つのディプロイメントモデルで構成され、可用性を促進するものとなる。

2.1.1　本質的な特徴

- オンデマンドのセルフサービス

それぞれのサービスプロバイダーとの人的な対話に依存することなく、消費者は必要に応じて自動的かつ一方的に、サーバーやネットワーク、ストレージの利用時間といった、コンピューティングの能力を供給する。

- 幅広いネットワーク・アクセス

このコンピューティング能力は、ネットワーク上で利用でき、また、標準的なメカニズムを介してアクセスできる。それにより、異種のシン／シッククライアントプラットフォーム（携帯電話／ラップトップ／ PDA）からの利用が促進される。

- 資源の置き場所

プロバイダーのコンピューティングリソースは共有され、マルチテナントモデルを利用する多数の消費者に提供される。そこでは、消費者からの需要にしたがって動的に割当／解消される、物理的あるいは仮想的なリソースを用いられる。一般的に、そこで供給されるリソースの正確な位置を、顧客が制御／知覚すること

はない。そのため、ロケーションから独立した感覚で、より高い抽象レベル（国／州／DC）においてロケーションが特定されてもよい。こうしたリソースの例としては、ストレージ／プロセッサ／メモリ／ネットワーク帯域幅／仮想マシンなどが含まれる。

- 弾力性に富む

コンピューティング能力の配給は、弾力性に富む。すなわち、いくつかのケースでは自動的に、スケールアウトの際に拡大し、また、スケールインの際に縮小する。消費者にとって、この供給能力は無限に追加できるものになり、また、従量制で購入できるものとなる。

- 最適化サービス

サービスの種類（ストレージ／プロセッサ／バンド幅／アクティブユーザーアカウント）に適した抽象レベルにおける測定機能を高めることで、クラウドシステムは自動的にリソース利用を制御し、最適化する。こうしたリソースの使用量については、利用されたサービスのプロバイダーと消費者から、透過的にモニター／コントロール／レポートされる。

2.1.2 サービスモデル

- サービスとしてのクラウドソフトウェア（SaaS）

このコンピューティングの能力は、クラウドインフラストラクチャ上で実行されるプロバイダーのアプリケーションを用いて、消費者に提供される。 そのアプリケーションは、 Web ブラウザ（Web メールなど）といったシンクライアントインターフェイスを介して、各種のクライアントデバイスからアクセスできる。消費者はネットワーク、サーバー、オペレーティング・システム、ストレージや、個別のアプリケーション機能さえも含めて、基礎となるクラウドインフラストラクチャの管理／制御は行わないが、個々のユーザーに特定されるアプリケーションコンフィグレーションは例外となる。

- サービスとしてのクラウドプラットフォーム（PaaS）

プロバイダーがサポートするプログラム言語とツールで作成したクラウドインフラストラクチャに、消費者が作成もしくは取得したアプリケーションを配置することが、消費者に提供される機能となる。消費者はネットワーク、サーバー、オペレーティング・システム、ストレージなどの、基礎となるクラウドインフラストラクチャの管理／制御は行わないが、配置されたアプリケーションを制御し、また、そのホスティング環境をコンフィグレーションすることがある。

- サービスとしてのクラウド基礎（IaaS）

オペレーティングシステムやアプリケーションを含む任意のソフトウェアを、消費者が配置／実行することができる場所で、プロセッサ／ストレージ／ネットワーク／重要なコンピューティングリソースなどを供給するための機能が、消費者に対して提供される。消費者は、基礎となるクラウドインフラストラクチャの管理／制御は行わないが、オペレーティングシステム／ストレージ／配置されたアプリケーションを制御し、また、選択された（ホストファイアウォールなどの）ネットワークコンポーネントを限定的に制御する。

2.1.3 ディプロイメントモデル

- プライベートクラウド

このクラウドインフラストラクチャは、特定の組織のために単独で運用される。そして、当該組織あるいは第三者団体により管理され、敷地内あるいは敷地外で運用されるだろう。

- コミュニティクラウド

このクラウドインフラストラクチャは、いくつかの組織により共有され、また、関心事（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）を共有する特定のコミュニティをサポートする。そして、当該組織あるいは第三者団体により管理され、敷地内あるいは敷地外で運用されるだろう。

- パブリッククラウド

このクラウドインフラストラクチャは、不特定多数の人々や大規模な業界団体などに提供され、対象となるクラウドサービスを販売する組織により所有される。

- ハイブリットクラウド

このクラウドインフラストラクチャは、複数のクラウド定義（プライベート／コミュニティ／パブリック）から、2 つ以上を組み合わせたものとなる。それぞれに固有の実体は保持するが、標準あるいや個別のテクノロジーによりバインドされ、データとアプリケーションのポータビリティ（クラウド間でのロードバランシングのためのクラウドバーストなど）を実現する。

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。

- Separation（衝突の回避）：近くにいる仲間と衝突しないようにする
- Alignment（速度を合わせる）：近くの仲間と速度を一致させようとする
- Cohesion（群れの中心（重心）に向かう）：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。

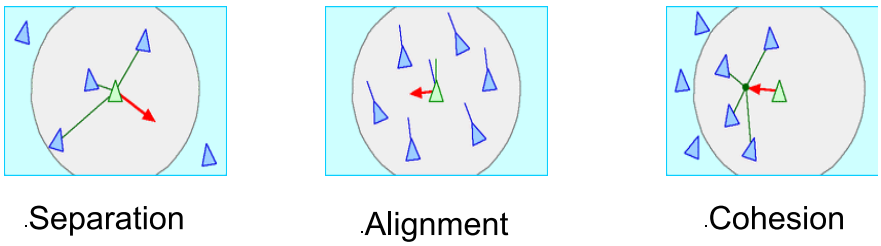

図 2.1: Separation・Alignment・Cohesion

2.3 ACO（Ant Colony Optimization）

アリの摂食行動から着想を得たアルゴリズム。
フェロモンという揮発性物質を模したパラメータを最適化する。
組合せ最適化やネットワークルーティングなどに応用。

実世界では、アリは始めランダムにうろつき、食物を見つけるとフェロモンの跡を付けながらコロニーへ戻る。他のアリがその経路を見つけると、アリはランダムな彷徨を止めてその跡を辿り始め、食物を見つけると経路を補強しながら戻る。しかし、時間とともにフェロモンの痕跡は蒸発しはじめ、その吸引力がなくなっていく。その経路が長いほどフェロモンは蒸発しやすい。それに対して、経路が短ければ行進にも時間がかからず、フェロモンが蒸発するよりも早く補強されるため、フェロモン濃度は高いまま保たれる。従って、あるアリがコロニーから食料源までの良い（すなわち短い）経路を見つけると、他のアリもその経路を辿る可能性が高くなり、正のフィードバック効果によって結局すべてのアリが 1 つの経路を辿ることになる。

2.4 家族的類似

ルートヴィヒ・ウィトゲンシュタインの言語ゲームで振る舞う particle は、家族的類似性によって群れを作ると定義した。家族的類似性は言語と行為の類似性を表現するものである。家族的類似性は同値関係でもないし、等価関係でもない。常に変動しつつ、少しづつ似ているエンティティの集まりである。しかし、それは自己から見れば、同値関係であってもよいし等価関係であってもよい。そのような個人個人の意味論を統合して世界を記述する意味論が無いということである。家族的類似性は不確実な世界の中の同類の定義である。家族的類似性は公的言語世界の観察視点の同類の定義である。
群知能において群れる振る舞いは家族的類似に基づいている。

集まる力の源は、「家族的類似」であり、群知能のパラメータとして表現される。
群れる正の力

- 群れの中心に向かう力： Cohesion
- 隣人と家族的類似行為をする力： Alignment
- 行為濃度： Pheromone

Pheromone は行為の軌跡の重要性を表現する．Pheromone は揮発性である．濃度が濃い pheromone は重要な行為を表す.

群れる負の力

- 群れる負の力は，群れから排除する力，群れから離れる（解除する）力である． Separation

2.5 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。

2.5.1 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 2.2: Android のアーキテクチャ

2.5.2 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。

2.5.3 アプリケーション開発の流れ

開発作業の流れは、 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。

2.5.4 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）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

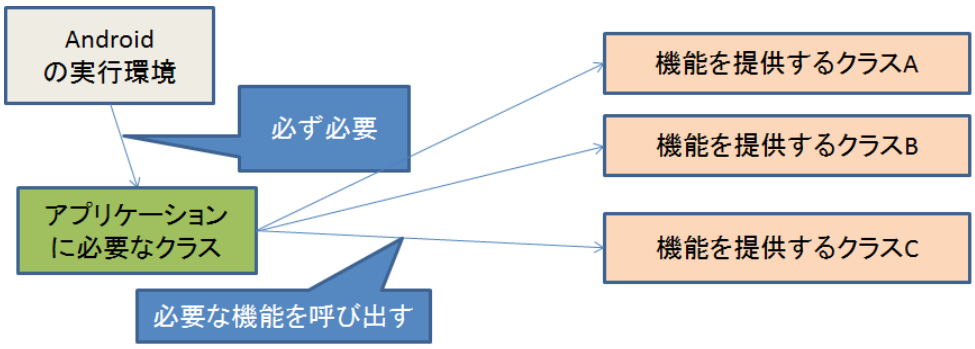

図 2.3: Android アプリケーションを構成するクラス。２つのクラスの関係性

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。

第3章　提案

3.1　目的

従来のファイルシステム（木構造）では、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、クラウドでは、ファイルが大量なので整理・分類するのが大変である。クラウド内の大量なファイルを自動的に整理・分類できるようにし、重要なファイル・関連するファイルを視覚的に見やすいファイルシステムを開発することが目的である。そこで以下のような特徴を持つ新しいクラウドのファイルシステムの一環として、PC 上のローカルなシステムを対象としたファイルシステムを提案する。

- 多様で不確実なクラウド内のファイルを一種の群れとして扱う。
- 類似的なファイルを群れの中心に集める。
- 重要なファイルを群れの中心に集める。

このクラウドファイルシステムを使用すると、例えばPC のデスクトップに様々のファイルがぐちゃぐちゃに散らばってしても、探しているファイルや関連するファイルを容易に見つけることができる。また、自分ではあまり重要ではないとファイル・関連性のないファイルだと思っているものが以外と重要である場合に、クラウドファイルシステムによって提示しユーザーの支援を行う。

3.2　Boid をベースとする群知能

クラウド内の大量なファイルを自動的に整理・分類したい。また探しているファイル・そのファイルに似たようなファイル・関連するファイルを簡単に見つかるように「ファイルの見える化」を行いたい。そこでこのようなファイルの振る舞いを Boid の Alignment、Cohesion、Sparation をベースとする群知能によって実現する。この群知能によって、似たようなファイル・関連するファイル同士が群れを作り集まる。

- Alignment,Cohesion(引力)：家族的類似なファイル同士に引力が働く。
- Sparation（斥力)：異なるファイル同士に斥力が働く。

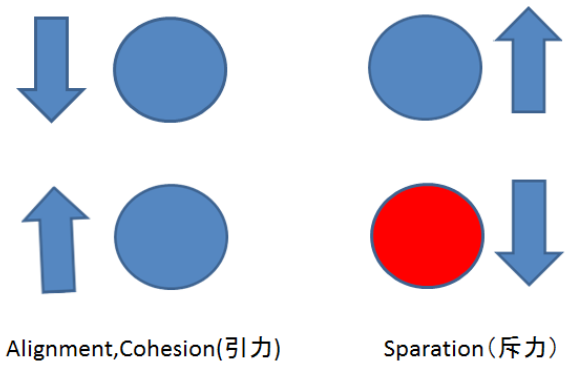

図 3.1: 引力・斥力の振る舞い

3.3 ACO の概念的特徴の「フェロモン」

群れの中心に重要なファイルが集まるようにするとそのファイルを見つけやすくなる。そこで ACO（Ant Colony Optimization）の概念的特徴「フェロモン」を用いる。群れの中心にフェロモンの強いオブジェクトが集まる、なので、フェロモン＝重要度と定義し、群れの中心に重要なファイルを集める。フェロモンは揮発性なので、あまり使用されないファイルは重要度が薄れていく。なので、使用頻度の多いファイルは重要度が高いファイルとして群れの中心に集まる。逆に使用頻度の少ないファイルは重要度の低いファイルとして群れの外側に移動する。

3.4 Boid とフェロモン

クラウドファイルシステムでのファイルの群れの振る舞いは Boid の Alignment, Cohesion,Sparation とフェロモンによって創発される。
まず、Boid の Alignment,Cohesion,Sparation で家族的類似しているファイル同士が群れを作る。その群れの中での位置関係としてフェロモンを使用する。群れの中心にフェロモンの濃度が高いファイルが集まる。フェロモン＝重要度としているので重要度が高いファイルが群れの中心に集まる。ファイルの重要度は 10 段階にに設定する。そして、図に示すようにあらかじめファイルの重要度に対して群れの中での群れる範囲・位置を定めておく。そうして、各ファイルの群れの中でも位置を定めていく。

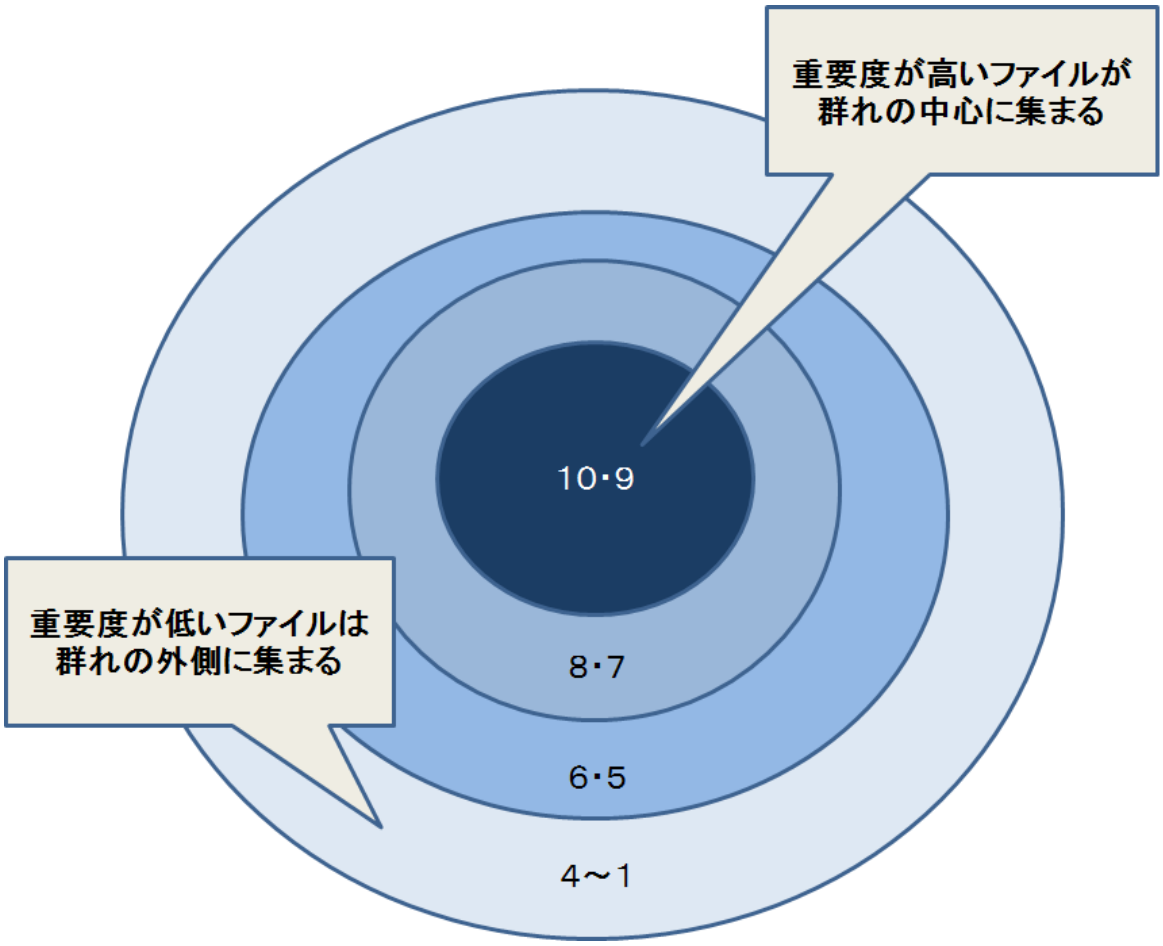

図 3.2: 重要度による位置関係

3.5 クラウドファイルシステム

3.5.1 重要度の定義

すべてのファイルに重要度を設定し、群れの中でのファイルの位置を決める。重要度は 10 段階に設定する。どのようなファイルが重要度が高いか、以下のように定める。

- 使用頻度

使用頻度が多いファイルが重要度が高いとする。

- 時系列

仕事や学校での課題、研究等の締切を設定する。締切が近いファイルは重要度が高いとする。また締切が近づいてくると自動的に重要度が高くなる。

- 依存関係

ファイル同士の重要度の依存関係のこと。A の仕事の前に B の仕事をやらなくてはならない。そのような場合、重要度は A ＜ B となる。

- ノルマ

ノルマが設定されてる仕事などに対して、ノルマが達成されてない場合は重要度が高いとする。達成されて時点で重要度は低くなる。

- 任意の重要度

今まで述べた重要度の定義に当てはまらなくても重要なファイルはある。そこでユーザーが重要度を指定できるようにする。

3.5.2 タグ

ファイルに対してあらかじめタグというものを付けておく。タグの種類を以下のようにする。

- 色

ユーザーがそのファイルに対して色を指定する。色の付け方はそのファイルを使う目的別に分けると良い。卒業研究に必要なファイルであれば青色、就職活動に必要なファイルであれば赤といったように色を付けると、色ごとに群れを作り関係するファイルが集まる。

- 文字列

そのファイルに対するキーワードを付ける。付ける数は多ければ多いほど良い。理由としては、「人の創発的活動を刺激・支援」するためである。群れは自分が指定した色同士でしか集まらない。異なる色でも関係しているファイルはたくさんあるだろう。いろんなキーワードで検索をかけると自分が関係ないと思っていたファイルが以外なところで関係していて良いアイディアが得られることがある。このタグのおかげでインターネットクラウドの中での「人の創発的活動を刺激・支援」につながる。タグ付けは「整理・管理」の為だけでなく、「予想外の物を得る」ためでもある。

- 時系列

締切がある仕事や学校での課題、研究等のファイルに対して設定する。

- ノルマ

ノルマがある仕事や学校での課題、研究等のファイルに対して設定する。

- 任意の重要度

重要度の定義で当てはまらないが重要なファイルに対して、ユーザーが重要度を指定できる。基本的に指定した重要度は固定される。

3.5.3 重要度やタグの設定

クラウドファイルシステムを使用するにあたり、各ファイルに対して重要度やタグを設定し反映させる必要がある。ファイルの重要度に対してはそのファイルが今までに何回開かれているか調べるカウンターをクラウドファイルシステムに設定しておき、開かれた回数によって重要度を定める。全体のファイルに対して相対的に多く開かれているファイルを重要度が高いファイルとする。

タグに対しては 3.5.3「タグ」で述べた各タグを設定してクラウドファイルシステムで使用する。ファイルにタグの設定画面を設け設定する。ファイルは普段は色を基準として群れを作るため、色の設定は必須条件とする。色はあらかじめ用意されているカラーテーブルから選択する。また文字列の「キーワード」も「人の創発的活動を刺激・支援」のために最低でも 1 つ設定することを必須条件とする。キーワードはできるだけ多く設定するほうが望ましい。「時系列」、「ノルマ」、「任意の重要度」は必要であれば設定する。

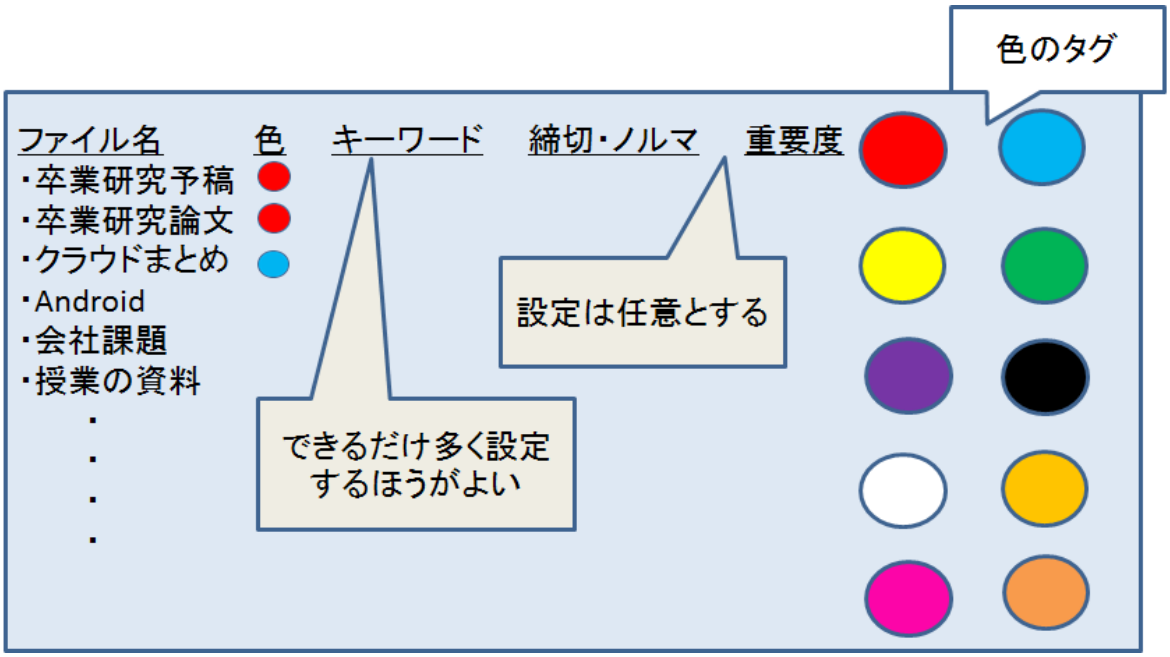

図 3.3: タグの設定画面イメージ

3.5.4 見せ方

- 群れになっている中から目的のファイルをタップ/クリックするとそのファイルの詳細画面（ファイル名、作成者、サイズ、更新日時など）のウィンドが開く。さらにタップ/クリックするとそのファイルが開く。
- そのファイルに指定した色の濃さで重要度を示す。重要度が高いと濃く、低いと薄くする。グラデーションを付けることによって視覚的に重要度の高いファイルを見つけやすくする。
- 重要度の高いファイルのアイコンは大きく、低いファイルのアイコンは小さく。

第4章　開発

4.1　開発概要

Android 端末を使ってクラウドファイルシステムを開発する。

4.1.1　開発環境

・Java
・JDK
・Eclipse
・Android SDK
・ADT Plugin for Eclipse
・Android 端末：「SONY Tablet」

仕様		
モデル		SGPT111JP/S
OS	名称	Android3.1
プロセッサー	名称	NVIDIA Tegra2 モバイル プロセッサ
	動作周波数	1GHz
液晶表示装置	サイズ/解像度	9.4型 WXGA（1280×800）
	パネル種別	TFTカラー液晶
ストレージ	容量	16GB
Wi-Fi	仕様	IEEE 802.11b/g/n準拠
GPS機能	仕様	搭載
Bluetooth機能	仕様	Bluetooth2.1 + EDR準拠
赤外線通信	赤外線リモコン機能	搭載
スピーカー	仕様	内臓ステレオスピーカー
マイク	仕様	内臓モノラルマイク
カメラ（フロント）	仕様	Webカメラ
	有効画素数	30万画素
カメラ（リア）	仕様	“Exmor for mobile” CMOSセンサー搭載 HDカメラ
	有効画素数	511万画素
外形寸法	本体（幅×高さ×奥行）	約 幅241.2mm×高さ10.1mm×奥行174.3mm
質量	本体	約598g

図 4.1: SONY Tablet の仕様

4.2 開発手順

1. 開発環境のセットアップ

開発に必要な JDK、Android SDK、Eclipse、ADT Plugin for Eclipse をインストールする。すべて無料でインストールできる。

・JDK（Java SE Development Kit）とは Java のソフトフェア開発キット

・Android　SDK（Software Development Kit ）とは Android のソフトフェア開発キット

・Eclipse とはオープンソースの統合ソフトフェア開発環境

・ADT Plugin for Eclipse とは、Android アプリケーションを Eclipse 上で開発するためのプラグイン

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 4.2: エミュレーションの起動画面

3. サンプルプログラムのダウンロード
クラウドファイルシステムのアプリケーションを開発するにあたって、画面上のボールが動き回り、画面の境界をバウンドしたり、ボール同士が衝突するとバウンドするサンプルアプリケーションを元に開発を行う。Web サイト”アスキー書籍用ダウンロードサイト”（http://books.ascii.jp）よりダウンロードする。

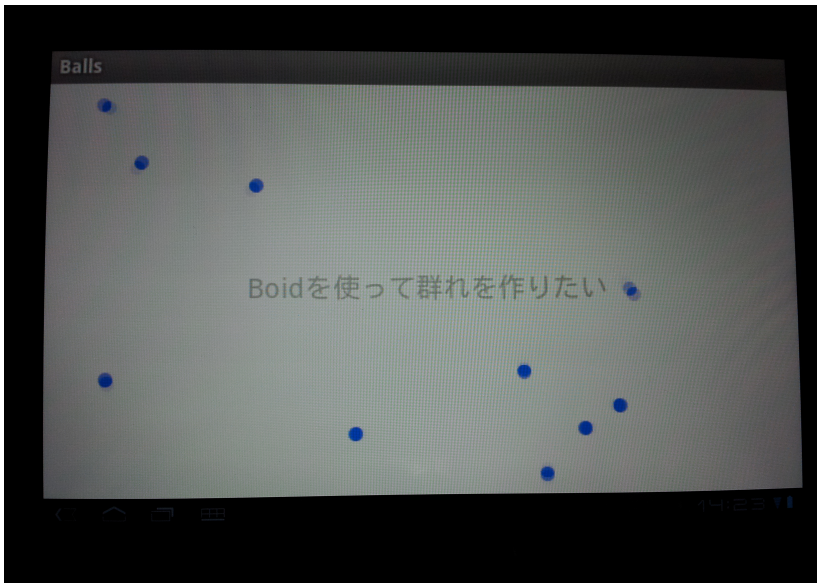

図 4.3: サンプルプログラム実行画面

4. 新規プロジェクトの作成
Eclipse を開き新規プロジェクトを作成する。

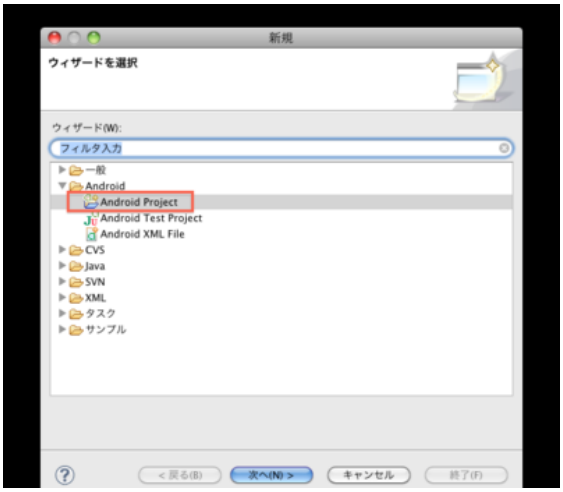

図 4.4: プロジェクト選択画面。「Android」から「Android　Project」を選択する。

5. ファイルやクラスの定義
それぞれのファイルで必要事項をプログラミングする。

6. 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 4.5: 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。

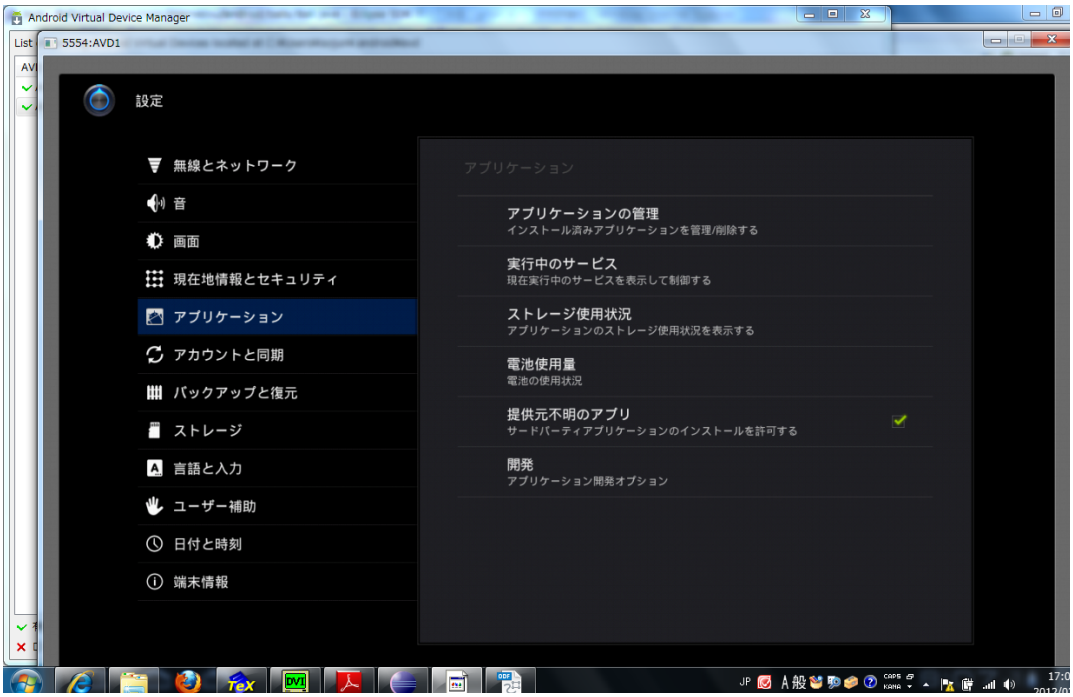

図 4.6: 「アプリケーション」の設定画面

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 使用クラス

- Ball.java

ファイルの動作（動く方向、速度）やファイルが動く領域などを決めている。

- BallsActivity.java

レイアウトや表示させる文字などファイルを呼ぶためのクラス

- BallsSurfaceView.java

描画性能を向上するためのクラス（描画処理専用）

- BallsView.java

タッチイベントやファイルの数、背景の色などを決めているクラス

- CollisionCombination.java

ファイルが壁やファイルと衝突した際の処理を決めている。

4.4 結果

- サンプルプログラムのインストールは成功した。
- ファイルの群れの表示の基本的な機能として、Android 端末上で実装を行い、ファイルの群れを表示し、移動できることを確認した。
- Boid の Alignment,Cohesion,Sparation の要素を Ball,java に組み込むことができなかった。一つのクラスだけに Alignment,Cohesion,Sparation のプログラミングを書こうとしていたが、実際は複数のクラスを変更しなければならないと思われ、とても複雑であった。

図 4.7: 「Android 端末での実行画面

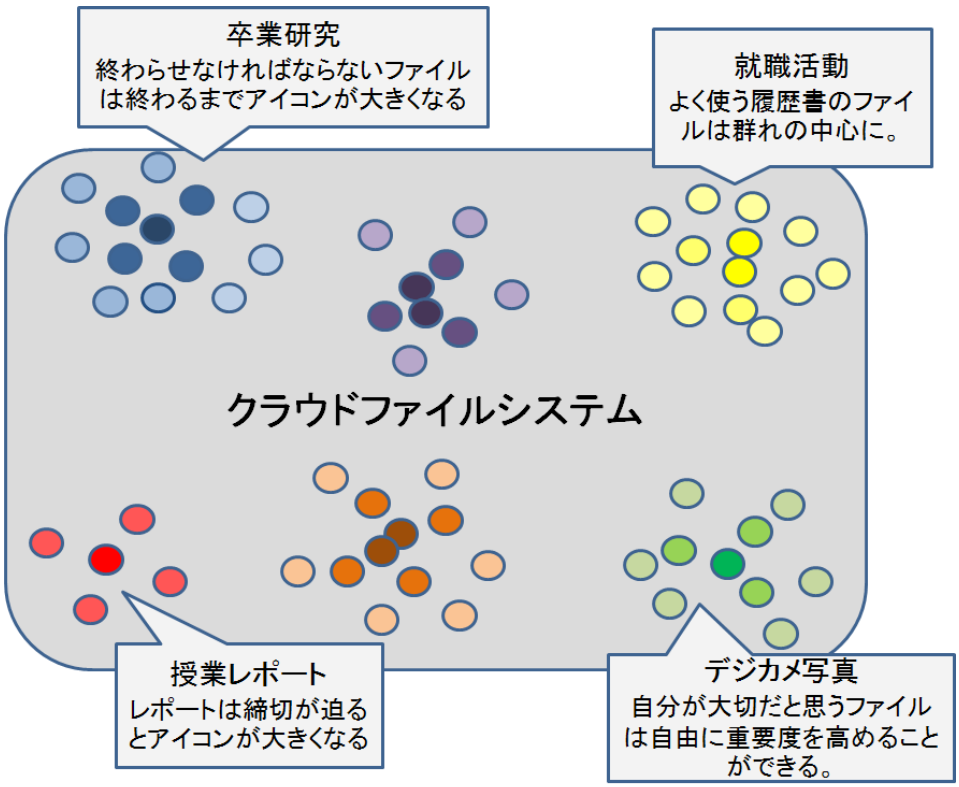

図 4.8: クラウドファイルシステムイメージ図

第5章　まとめ

本研究では「人の創発的活動の刺激・支援」するような新しいファイルシステムを提案した。そのファイルシステムは関連するファイル同士が群れを作ること目標とし群知能を用いて開発を行った。Android 端末の画面上でボールが動き回り単純なアプリケーションに Boid の Alignment,Cohesion,Sparation のアルゴリズムと ACO の「フェロモン」という概念を組み込むことを目標とした。今後の課題として、組み込むことができなかった Boid の Alignment,Cohesion,Sparation のアルゴリズムと ACO の「フェロモン」という概念を組み込み、ファイルが群れるようにすることが第一の課題である。また、群れの生成に必要なアクセス頻度や締切期限などのファイルのメタ情報の選定とメタ情報付与のインターフェースを構築すること。これらを今後の課題とする。

第6章　謝辞

本研究を行なうにあたり，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御助言をしていただいた東洋ネットワークシステムズ株式会社の森住哲也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．

2012年　2月
磯村　淳

関連図書

[1] 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 徹郎： ”Google Android プログラミング入門”株式会社豆蔵 （2009）

[2] 岩谷 宏：”JAVA の哲学”ソフトバンクパブリッシング (2001)

[3] 伊庭 斉志：”複雑系のシミュレーション Swarm によるマルチエージェント・システム”コロナ社 (2007)

[4] ”Boid とは”
http://members.jcom.home.ne.jp/ibot/boid.html

[5] ”National Institute of Standards and Technology”
http://www.nist.gov/index.html

[6] ”Java 初心者入門講座” http://sunjava.seesaa.net/

[7] ”鰯の群れ ライブ壁紙”QSDN http://www.qsdn.co.jp/service/android+apps/

[8] ”Boids もどき” http://hp.vector.co.jp/authors/VA009508/Java/Boids/boids.html

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

[10] ”mdellavo / boids” https://github.com/mdellavo/boids

[11] ”家族的類似” Wikipedia http://ja.wikipedia.org/wiki/

[12] ”Google USB Driver”Android developers
http://developer.android.com/intl/ja/sdk/win-usb.html

[13] ”Xperia Android 開発環境 SDK インストール ”
http://start-android-sdk.blogspot.com/

[14] ”Android 入門.”テックファーム
http://www.techfirm.co.jp/lab/android.html

[15] ”Sony Tablet”Sony
http://www.sony.jp/tablet/

[16] ”オブジェクト思考
http://think-on-object.blogspot.com/2011/11/is-ahas-is-ahas-top-is-a-is-b.html

[17] ”インスタンス化”
http://sunjava.up.seesaa.net/image/java-195.gif

[18] ”ボリューム-意味・説明・解説”ASCII.jp
http://yougo.ascii.jp/caltar/

[19] ”Flockers”MASON
http://cs.gmu.edu/ eclab/projects/mason/

[20] ”Boid”TM’s Workspace
http://termat.sakura.ne.jp/actionscript/boid/extended

[21] ”Android”Wikipedia
http://ja.wikipedia.org/wiki/Android

第7章　質疑応答

- Q. ファイルの色が途中で変わることがあるか。

A. ファイルに対して、色だけでなくメータデータ（文字列など）のタグを付け、キーワードで検索をかけたときに色が違っても群れが作れるようにしたいと思っている。

- Q.Boid と ACO の関係性はどのようなものなのか

A.Boid は、家族的類似のファイル同士の群れを作るためのアルゴリズム。ACO は Boid によりできた群れの中のファイルの重要度によって群れの中での位置を示すための使用する。フェロモン＝重要度としているので重要度の高いファイルが群れの中心集まる。
Boid によって群れを作り、ACO によって群れの中の中の位置を定める。