拡張性の高い Java マルチユーザ・フレームワーク Mug

5M-05　根山 亮† 後藤 真孝†† 園田 智也† 村岡 洋一†
†早稲田大学理工学部 ††電子技術総合研究所

1. はじめに

本研究では, マルチプレイヤー型ゲームのような, ネットワークを介したマルチユーザ環境を必要とするさまざまなアプリケーション (以下, 単にアプリケーションと呼ぶ) で, 1つのマルチユーザ環境を共用するためのフレームワーク (**マルチユーザ・フレームワーク**) **Mug** を **Java** 言語で実装した. ユーザ情報の管理やチャット等の機能を, 様々なアプリケーションに持たせることを容易にしただけでなく, 公開/非公開グループやアプリケーションごとに拡張可能なユーザ属性も導入した. 実際に, 不特定多数のユーザによる遠隔音楽セッションシステムを **Mug** 上で実装することにより, その有効性を実証した.

2. 従来のマルチユーザ環境の構築環境

従来のアプリケーションの多くは, それぞれに特化したマルチユーザ環境を独自に実装していた. そのため, そのマルチユーザ環境の機能を他のアプリケーションで再利用することは難しかった.

一方, 従来の汎用的なマルチユーザ環境の多くは, 仮想空間の中でユーザ間のコミュニケーションや知識共有を重視しており [1], 拡張性はあまり考慮していなかった. そのため, その環境の上でさまざまなアプリケーションをシームレスに組み込むことは難しかった.

ユーザにとって使いやすいマルチユーザ環境を実現するためには, 長い時間をかけてより多くのユーザからの要望を採り入れることが望ましい. そのためには, マルチユーザ環境にさまざまなアプリケーションを組み込むためのフレームワークを用意し, より汎用的で使いやすいマルチユーザ環境を構築するための基盤にできるとよい.

従来研究ではこのようなフレームワークが提案されていなかったため, アプリケーションが個々にマルチユーザ環境の開発を行なわなければならなかった. また, 複数のアプリケーションが統一した GUI (*Graphical User Interface*) を提供することも難しかったため, ユーザにとって使い勝手が悪かった.

3. フレームワークに必要な機能と拡張性

拡張性の高いマルチユーザ・フレームワークを実現するため, 我々は以下の **(1)** ～ **(4)** の機能を持つフレームワークを提案する.

(1) ユーザ構成を自由に指定できる仮想空間の作成

ユーザの好みやアプリケーションの種類によっては, すべてのユーザで情報を共有する開かれた仮想空間だけでなく, 限られた何人かだけで情報を共有する閉じた仮想空間も必要となる. マルチユーザ環境で実現される仮想空間のユーザ構成を, ユーザやアプリケーション開発者が自由に指定できるとよい.

例えば, 将棋対局の場合, 2人の対局者の他に複数の観戦者が存在する縁台将棋のような対局には開かれた仮想空間が必要となり, 第3者から観戦されたくない対局には閉じた仮想空間が必要となる. チャットについても, サーバがあらかじめ用意した話題にユーザが集まって来る形態には開かれた仮想空間を適用し, ユーザ同士が個人的に集まる形態には閉じた仮想空間を適用できる.

(2) アプリケーションによる Mug の機能のカスタマイズ

マルチユーザ・フレームワークは, それを共用するアプリケーションに合わせて, カスタマイズできるとよい. アプリケーション固有の情報 (例えば, マルチプレイヤー型ゲームでの得点など) を定義し, それを GUI で他のユーザ情報 (例えば, ユーザ名やクライアントのホスト名など) とともにシームレスに表示できるとよい. また, ユーザ情報の変更に合わせてアプリケーションに固有の処理をさせるための機構が必要となる.

(3) アプリケーションで共有可能な GUI の提供

ユーザ一覧の表示とその整列, チャット等の他のユーザとのコミュニケーションのための GUI が必要となる. 複数のアプリケーションで GUI を共有できることが, ユーザの使い勝手の向上と機能の再利用の意味で重要である.

(4) ユーザ情報の管理

サーバへのユーザのログイン・ログアウトや現在ログイン中のユーザの情報の管理し, ユーザ情報をすべてのクライアントで共有する機能が必要となる.

4. システムの概要

Mug は, 公開グループ (後述) を表す **PublicGroup**, ユーザ情報を表す **MugUserInfo**, ユーザ情報を管理しユーザ間の通信を中継する **MugServer** と GUI である **MugClient** で構成される (図 1). それぞれのクラスは, 再利用が容易なコンポーネントとして設計されている.

○ **PublicGroup**　ユーザの公開/ 非公開グループは, 開かれた/閉じた仮想空間をそれぞれ実現する (前述の機能 **(1)**). 公開グループは, 所属するユーザの一覧をサーバが管理し, すべてのクライアントでその情報を共有できるグループである. **PublicGroup** は, 公開グループに所属するユーザの一覧を保持する. 一方, 非公開グループは, 所属するユーザの一覧をクライアントが管理し, 他のクライアントとは無関係に、そのクライアントのユーザが自由にカスタマイズできるグ

Mug: An Expandable Multiuser Framework for Java
Ryo Neyama†, Masataka Goto††,
Tomonari Sonoda†, Yoichi Muraoka†
†School of Science and Engineering, Waseda University
††Electrotechnical Laboratory

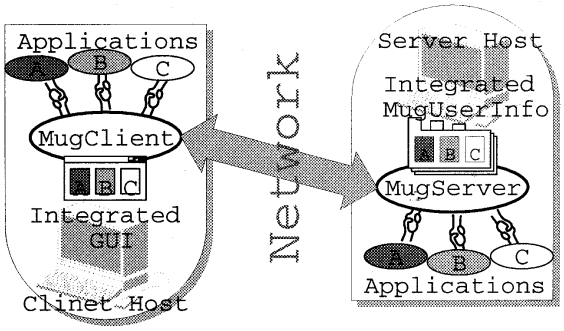

図 1: マルチユーザ・フレームワーク Mug

ループである.

○ **MugUserInfo**　アプリケーションはそれ固有のユーザ情報を, ユーザ情報を表す MugUserInfo のサブクラスとして定義できる (前述の機能 **(2)**). MugUserInfo はユーザ情報の属性の名前とその型 (クラス) の表を持っている. MugUserInfo のインスタンスは属性表で定められた型のインスタンスを値としてを保持する. MugUserInfo は, ユーザ ID 番号, ユーザ名, 所属する公開グループ名, ユーザのホスト名を標準の属性として持っている. アプリケーションは, 固有の属性を新たに追加したり, 削除できる.

アプリケーションがユーザ情報を利用する場合, 能動的に取得する機構の他に, ユーザ情報の変更をイベントとして受け取る機構も必要となる. Mug では Java の AWT (*Abstract Window Toolkit*) のイベント処理と同じ委譲型のイベント処理ができる. MugUserInfo の属性の追加・削除, 各インスタンスに対する値の変更のイベントを, アプリケーションの登録したイベントリスナ (*Event Listener*) が処理できる.

○ **MugClient**　Mug 上のアプリケーションは MugClient (図 2) を共通の GUI として利用できる (前述の機能 **(3)(4)**). ユーザは, MugServer へのログイン・ログアウトをここで行なう. ユーザがログインすると, 右側の「すべてのユーザ」のサブウィンドウに現在ログイン中のユーザの一覧が表示される.

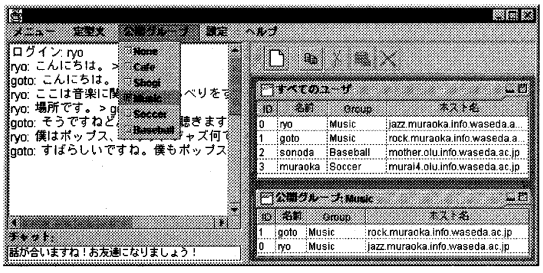

図 2: アプリケーションを組み込んでいない MugClient (ユーザ "ryo" のウィンドウ)

「公開グループ」メニューで, 公開グループへの参加, 脱退ができる. この時, アプリケーションは, イベントリスナに「公開グループに参加しているユーザの顔写真を表示する」などの特別な処理を行わせることも容易である. また, 右側のサブウィンドウの上部にあるツールバーを操作することにより, 非公開グループを新たに作成したり, 削除できる.

各サブウィンドウをアクティブにして, 左下のチャット入力欄に文字列を入力ですることで, サブウィンドウに表示されたユーザとチャットを行なう. サブウィンドウ内の一覧からユーザを選択する (複数も可) ことにより, 指定したユーザだけとのチャットも可能である.

各サブウィンドウのユーザ属性名をマウスでクリックすることにより, 属性の値を鍵としたユーザ一覧の整列ができる. また「定型文」メニューで, 定型文を編集したり, その定型文をチャット入力欄に張り付けられる.

○ **MugServer**　MugServer はユーザのログイン・ログアウトの管理や, ログイン中のユーザ情報の管理を行なう (前述の機能 **(4)**). また, ある MugClient のユーザ情報の更新やチャットのメッセージなどを他の MugClient にマルチキャストする. アプリケーションは, MugServer のサブクラスを定義し, MugServer のメソッドをオーバーライド (*override*) することにより, アプリケーション固有の処理を行なえる. 例えば, ユーザがログインした時に「そのユーザを待っている他のユーザにそれを知らせる」などの特別な処理をしたい場合は, `login` メソッドをオーバーライドすればよい.

5. 実装とアプリケーション

Mug は, JDK (*Java Development Kit*) 1.1, Swing 1.1, Java 分散オブジェクト・プログラミング環境 RyORB[2] を用いて実装した.

Mug に組み込むアプリケーションとして, インターネットのような広域ネットワーク上の不特定多数のユーザ同士が遠隔地間で音楽セッションを行なえるシステム **Open RemoteGIG**[3] を開発した. Mug により, 演奏相手を発見する環境や打合せ用のチャット機能を容易に実現でき, ユーザが使用する楽器の種類のような Open RemoteGIG 固有の情報を MugClient の中でシームレスに表示できた. また、その GUI は, 他のアプリケーションに特化して開発されたマルチユーザ環境と同等の使いやすさが得られた.

6. まとめと今後の課題

本稿では, マルチユーザ環境を必要とするアプリケーションのためのマルチユーザ・フレームワーク Mug について述べた. 今後はより多くのアプリケーションを Mug に組み込む予定である.

参考文献

[1] Toru Ishida, et al.: *Community Computing, Collaboration over Global Information Networks, JOHN WILEY & SONS (1998).*

[2] 根山 亮: *RyORB* ホームページ, **http://www.info.waseda.ac.jp/muraoka/members/ryo/RyORB/index-j.html**

[3] 後藤 真孝, 根山 亮: 不特定多数による遅延を考慮した遠隔セッションシステム, 情処研報, *Vol.98, No.74,* 音楽情報科学 *98-MUS-26-14 (1998).*