溶け込むインタフェースの研究
～PC，Web，ユビキタスの環境を融合するインタフェースデザイン

A Study on Fusing Interface - Novel Interaction Design Integrating PC, Web, and Ubicomp

慶應義塾大学大学院 政策・メディア研究科
神原 啓介

論文概要

PC，Web，ユビキタスコンピューティングといったコンピュータ技術が普及する過程で，その多くの機能は「アプリケーション」という形でユーザに提供され，利用されるようになった．現在の主要なアプリケーションには,「デスクトップアプリケーション」「Web アプリケーション」「ユビキタスアプリケーション」があり，それぞれ利用環境や動作環境が異なっている．PC 環境，Web 環境，ユビキタス環境のそれぞれが持つ利点をうまく組み合わせることで，単一の環境ではできなかった新しいコンピュータの活用方法やコミュニケーションが生まれ，コンピュータの活用範囲がより一層広まると考えられる．しかし，現在のアプリケーションの多くは，他の環境と組み合わせて使おうとすると様々な不和が生じてしまう．この環境間の不和には，やりたいことを直接的にできない「操作の複雑化」や，操作方法や外観が環境に馴染まない「環境との非調和」，環境間で情報をやり取りできない「情報の非流動」といったものがある．

この環境間の不和を解決するため，本研究では「溶け込むインタフェース」を提案する．溶け込むインタフェースとは，PC，Web，ユビキタスといった環境の違いを越えて，アプリケーションを他の環境に馴染ませるインタフェースデザイン手法である．環境に馴染ませるための様々なインタフェースデザインの工夫により，素早く直接的に操作できる「平易性」や，外観や操作方法が環境に馴染んで違和感が無い「親和性」，環境間でスムーズに情報が流れる「流動性」を高める．そして，様々なアプリケーションが自然な形で互いに混ざり合うことで，PC，Web，ユビキタスがあたかも融合した状態を目指す．

本研究では，主要なアプリケーションの環境である「PC 環境」「Web 環境」「ユビキタス環境」において，それぞれ溶け込むインタフェースを提案，構築し，検証する．

PC 環境に溶け込むインタフェースとして，Web 環境の SNS（Social Networking Service）を PC 環境に溶け込ませる「ソーシャル顔アイコン」を提案する．ソーシャル顔アイコンは，SNS の中にいる人をアイコンとしてデスクトップに溶け込ませることで，複数のサイトを横断しながら，優先的に見たい人の情報が埋もれることなく，少ない操作で見やすく表示するインタフェースである．

Web 環境に溶け込むインタフェースとして，PC 環境のドローツールやペイントツールを，Web 環境のブログや SNS に溶け込ませる「Willustrator」「TwitPaint」という 2 つのシステムを提案する．これらは Web ブラウザ上で直接絵を編集できるようにすることで，他の人の描いた絵の再編集や再利用を可能にし，さらに Twitter などと連携した絵によるコミュニケーションを促進する．

ユビキタス環境に溶け込むインタフェースとして，PC 環境のビデオチャットを生活空間に溶け込ませる「なめらカーテン」を提案する．なめらカーテンは，カーテンメタファを用いた柔軟で分かりやすいプライバシー制御手法によって，生活空間でも違和感なく，常時利用可能なビデオチャットを実現する．また，このようなシステム開発を効率的に進める実践的な手法を紹介する．

さらに，溶け込むインタフェースに関連する研究を紹介し，本研究の位置づけについて整理する．最後に，溶け込むインタフェースの特徴や課題を明らかにし，今後の展望を示す．

キーワード：ユーザインタフェース，ユビキタスコンピューティング，Web，GUI

Abstract

Through the evolution of personal computing, Internet technologies and ubiquitous computing technologies, various application programs and service packages have been developed and used on these platforms. However, most applications and services are not integrated enough, and they should be used separately. For example, there are many useful applications for communication and applications for drawing, but they are not easily used together.

When a user tries to use more than one application at the same time for performing a single task, he might have to face various difficulties. First, different applications have different functions and they usually do not fit very well. Second, the conditions for running the applications sometimes do not fit the environment. Third, sufficient information does not flow between the applications and the environment because of various mismatches.

To solve these difficulties, we propose a novel concept of "Fusing Interface". The Fusing Interface extracts essential functions of computers from various applications and blends these functions flexibly. The Fusing Interface provides three key features: simplicity, adaptiveness and fluidity.

In this thesis, we describe three examples of the Fusing Interface in major platforms of computers: personal computing, internet technology, and ubiquitous computing.

First, we proposed the "SocialFaceIcon" system that blends SNS (Social Network Service) into PC environment. The SocialFaceIcon can visualize status of people in SNS as desktop icons. Users can quickly check status of their friends just by glancing at their desktop.

Second, we propose the "Willustrator" and "TwitPaint" system that blends drawing/painting tools into Web environment including the Blog and SNS. Using these systems, users can draw pictures on common Web browsers. They can also derive new pictures from existing ones drawn by other people and share them quickly via SNS like Twitter.

Third, we propose the "SmoothCurtain" system that blend the video chat into daily environment. The SmoothCurtain provides a novel video communication system suited for constant use in daily environment. Users can control privacy levels (e.g., screen blur and sound volume) quickly using curtain metaphor. Moreover, we introduce practical methods for developing systems in ubiquitous environment effectively.

In addition, we describe related work and discuss uniqueness of our approach.

Finally, we summarize features of the Fusing Interface, and discuss its future possibilities.

Keyword: User Interface, Ubiquitous Computing, Web, GUI

目 次

図 目 次

表 目 次

第 1 章

序論

概要

本章では本研究の背景と目的，本論文の構成について述べる．

1.1 研究の背景

PC，Web，ユビキタスコンピューティングといったコンピュータ技術が普及する過程で，その多くの機能は「アプリケーション」という形でユーザに提供され，利用されるようになった．現在の主要なアプリケーションには,「デスクトップアプリケーション」「Web アプリケーション」「ユビキタスアプリケーション」があり，それぞれ利用環境や動作環境が異なっている．本論文では，同種のアプリケーション群とその動作環境・利用環境をまとめて，アプリケーションの「環境」と呼ぶ．

PC 環境，Web 環境，ユビキタス環境はそれぞれ異なる利点を持ち，それらをうまく組み合わせることで，単一の環境ではできなかった新しいコンピュータの活用方法やコミュニケーションが生まれ，コンピュータの活用範囲がより一層広まると考えられる．しかし，現在のアプリケーションの多くは，他の環境と組み合わせて使おうとすると様々な不和が生じてしまう．この環境間の不和には，やりたいことを直接的にできない「操作の複雑化」や，操作方法や外観が環境に馴染まない「環境との非調和」，環境間で情報をやり取りできない「情報の非流動」といったものがある．

1.2 研究の目的

本研究の目的は，環境間の不和という問題に対して,「溶け込むインタフェース」を提案し，そのコンセプトに基づいたユーザインタフェースを提案，構築，評価することである．

本研究ではまず，コンピュータの普及に伴うアプリケーション環境の増加と，環境間の不和について考察し，問題点を明確にする．次に，環境間の不和を解決するための，溶け込むインタフェースというコンセプトを提案する．そして，主要なアプリケーションの環境である「PC 環境」「Web 環境」「ユビキタス環境」において，それぞれ溶け込むインタフェースを提案，構築し，検証する．

1.3 本論文の構成

本論文の構成を示す．

第 2 章 背景と問題意識

第 2 章では，本研究の背景と問題意識について述べる．まず背景として，コンピュータ技術が普及する過程で，コンピュータの様々な機能が「アプリケーション」という形でユーザに提供され，利用されるようになったことや，アプリケーションの「環境」について説明する．次に，問題意識として，複数のアプリケーションを組み合わせたときに生じる環境間の不和について説明する．

第 3 章 溶け込むインタフェースの提案

第 3 章では，環境間の不和という問題に対する「溶け込むインタフェース」を提

案する．そして，溶け込むインタフェースの有効性や課題を検証するシステムを試作するにあたり,「PC 環境」「Web 環境」「ユビキタス環境」という 3 つの環境について，それぞれアプリケーションを選定する．

第 4 章 PC 環境に溶け込む

第 4 章では，PC 環境に溶け込むインタフェースとして，Web 環境の SNS（Social Networking Service）を PC 環境に溶け込ませる「ソーシャル顔アイコン」を提案する．ソーシャル顔アイコンは，SNS の中にいる人をアイコンとしてデスクトップに溶け込ませることで，複数のサイトを横断しながら，優先的に見たい人の情報が埋もれることなく，少ない操作で見やすく表示するインタフェースである．

第 5 章 Web 環境に溶け込む

第 5 章では，Web 環境に溶け込むインタフェースとして，PC 環境のドローツールやペイントツールを，Web 環境のブログや SNS に溶け込ませる「Willustrator」「TwitPaint」という 2 つのシステムを提案する．これらは Web ブラウザ上で直接絵を編集できるようにすることで，他の人の描いた絵の再編集や再利用を可能にし，さらに Twitter などと連携した絵によるコミュニケーションを促進する．

第 6 章 ユビキタス環境に溶け込む

第 6 章では，ユビキタス環境に溶け込むインタフェースとして，PC 環境のビデオチャットを生活空間に溶け込ませる「なめらカーテン」を提案する．なめらカーテンは，カーテンメタファを用いた柔軟で分かりやすいプライバシー制御手法によって，生活空間でも違和感なく，常時利用可能なビデオチャットを実現する．

第 7 章 ユビキタス環境に溶け込むインタフェースの開発手法

第 7 章では，我々がこれまで開発してきたユビキタス環境に溶け込むインタフェースの中でも，開発事例の多い「GUI とデバイスを連携したシステム」の開発を効率的に進める実践的な手法を紹介する．

第 8 章 関連研究

第 8 章では，本研究に関連する研究領域について整理し，本研究の位置づけについて述べる．

第 9 章 溶け込むインタフェースの考察と展望

第 9 章では，溶け込むインタフェースに関する考察と展望を述べる．

第 10 章 結論

第 10 章では，本研究の成果についてまとめ，本論文を総括する．

第 2 章

背景と問題意識

概要

本章では，本研究の背景と問題意識について述べる．

2.1 はじめに

かつてコンピュータやインターネットが未成熟な時代，コンピュータを使う人や使う場所は限られていた．しかし，コンピュータ技術の進化やネットワークインフラの整備によって，徐々にコンピュータの利用範囲が拡大し，現在は日常生活や仕事など様々な場面でコンピュータが活用されている．

コンピュータの一般への普及は段階的に進んでおり，まずパーソナルコンピュータ（PC）の普及が進み，それにともなってインターネットとWebの普及が進んだ．そして現在，携帯電話をはじめとするモバイルコンピューティングが広く一般に普及し，ユビキタスコンピューティングの普及が徐々に進みつつある．

2.1.1 PCの普及

パーソナルコンピュータというコンセプトは1970年代にアラン・ケイによって提唱され，WIMP（Window, Icon, Menu, Pointer）のようなPCの基本的なユーザインタフェースは当時開発された．

PCが普及し始めたのは，1980年代にAppleのMacintoshが発売されてからである．この頃からワードプロセッサや表計算，ペイント・ドローツールといった様々なアプリケーションが利用されるようになった．その後，Microsoftの Windows95が発売された1990年代後半から一般家庭にも急速にPCが普及した．国内の世帯別PC普及率[1]は，1995年の16.3%から，2003年には78.2%に増え，2011年現在83.4%になっている．

2.1.2 インターネットとWebの普及

1969年にインターネットの前身となるARPANETが構築され，1980年代に入り，TCP/IPや，Ethernet，UNIXといったインターネットの基盤技術が登場した．

インターネットが爆発的に普及するきっかけとなったソフトウェアがWorld Wide Web（WWW）である．1990年代初頭にティム・バーナーズ＝リーがWWWの基礎となるソフトウェアを開発した後，1993年に登場したWebブラウザのNCSA Mosaicの流行によってWebの普及が始まった．その後，1994年にNetscape Navigator，1995年にInternet Explorerが登場して以来，PCの普及と併せて一般家庭にもインターネットとWebが広く普及した．

2000年代中頃になるとWebの利用方法が変化し，その変化はWeb2.0と呼ばれている．それまでのWebは，情報の送り手と受け手が固定しており一方的に情報が流れていたが，ブログやSNS，Wikiといったツールの登場によって誰でもWeb上で情報を発信できるようになった．その結果，ソーシャルブックマークや写真共有といった，多くのユーザが参加してWeb上で情報を共有しあうサービスが急増し，一般に広く浸透した．また，RSSのようなメタデータやWebサービスのAPIが整備されることで，

[1] 2011年 内閣府消費動向調査．単身世帯を含む．

これらのデータを組み合わせたマッシュアップと呼ばれる開発が行われるようになり，次々と新しいWebアプリケーションが生まれるようになった．

現在のWebは情報共有のソーシャル化，リアルタイム化が進んでいる．TwitterやFacebookといったSNSが世界的に流行し，「一言つぶやく」といったちょっとした情報であっても素早く知り合い同士で共有できる．さらにモバイルコンピューティングの普及によって，いつでもどこでも情報共有でき，また写真や動画などを気軽に用いる情報共有なども可能になった．

2.1.3　モバイル・ユビキタスコンピューティングの普及

コンピュータの小型化やネットワークインフラの充実にともない，モバイルコンピューティングやユビキタスコンピューティングと呼ばれるコンピュータ利用が普及した．

最も広く急速に普及したモバイル機器として携帯電話が挙げられる．国内では1990年代後半から急速に携帯電話が普及しており，世帯別の携帯電話普及率は，1993年の3.2%から，2003年には94.4%にまで増えている．普及当初の携帯電話は性能が低く，インターネット接続にも制限があり，できることは限られていた．しかし，2000年代初めには第三世代携帯電話の登場により高速通信が可能になり，携帯電話上でのWeb利用が一般化した．また，1997年にiPhoneが登場して以来，高機能なスマートフォンが急速に普及しつつある．スマートフォン上で動く様々なアプリケーションが現在も盛んに開発されており，モバイル機器でできることの幅が広がっている．

ユビキタスコンピューティングは1990年代初頭にMark Weiserによって提唱されたコンセプトで，「コンピュータが実世界のあらゆる場所に遍在し，協調しあうことで，人々がコンピュータの存在を意識することなく利用できる」というものである．ユビキタスコンピューティングは現在，インターネットにつながる情報家電や車載情報機器，デジタルサイネージといった形で一般に普及しつつある．

2.2　アプリケーションの「環境」

PC，Web，ユビキタスコンピューティングといったコンピュータ技術が普及する過程で，その多くの機能は「アプリケーション」という形でユーザに提供され，利用されるようになった．

現在の主要なアプリケーションには，「デスクトップアプリケーション」「Webアプリケーション」「ユビキタスアプリケーション」があり，それぞれ利用環境や動作環境が異なっている．文書作成ソフトやペイントソフトなどのデスクトップアプリケーションは，PCにインストールして利用される．また，ブログやWiki，検索エンジンなどのWebアプリケーションは，インターネット上のサーバで動作し，Webブラウザ経由で利用される．そして，位置情報系サービスやデジタルサイネージなどのユビキタスアプリケーションは，携帯電話や公共端末，情報家電の上で動作し，利用される．

本論文では，同種のアプリケーション群とその動作環境・利用環境をまとめて，アプリケーションの「環境」と呼ぶ．そして「デスクトップアプリケーション」「Web アプリケーション」「ユビキタスアプリケーション」のそれぞれが含まれる環境を，「PC 環境」「Web 環境」「ユビキタス環境」とする．

以下，それぞれの環境の特徴や利点について述べる．

2.2.1 PC 環境

Microsoft Word のようなワードプロセッサや PhotoShop のようなペイントツールなどのデスクトップアプリケーションと，それらをインストールして利用する PC やデスクトップ OS，さらにマウスやキーボードによる操作を前提とした CUI や GUI をまとめて「PC 環境」と呼ぶ．

PC 環境の利点は，高度な GUI により，比較的複雑な作業をしやすいことである．例えば，長い文章を書く，絵を描く，動画を編集する，プログラミングするといった複雑な編集作業をこなすことができる．

2.2.2 Web 環境

ブログ，Wiki，SNS，掲示板，検索エンジンといった Web アプリケーションや，それらが動作する Web サーバ，そしてユーザが利用する Web ブラウザなどを「Web 環境」と呼ぶ．

Web 環境の利点は，情報共有やコミュニケーションをしやすいことである．ブログや Wiki，写真，動画共有サービスによって，テキストや写真，動画といった様々な情報を，手軽に多くの人と共有できる．

2.2.3 ユビキタス環境

位置情報系サービスのようなモバイルアプリケーションや，デジタルサイネージなどのユビキタスアプリケーション，そしてそれらが動作するモバイル機器や情報家電，さらにそれらの機器を設置して利用する場所をまとめて「ユビキタス環境」と呼ぶ．

ユビキタス環境の利点は，様々な場所や状況に応じたサービスを受けられることや，身体性を活かした UI を利用できることである．モバイル機器は利用場所が固定されず，いつでもどこでも利用でき，GPS を用いて今いる場所付近の情報を見るといったことが簡単にできる．また，リビングや食卓，お店，駅といった，様々な場所に応じたユビキタスアプリケーションが存在する．

2.3 環境間の不和

それぞれ異なる環境の利点をうまく組み合わせることで，単一の環境ではできなかった新しいコンピュータの活用方法やコミュニケーションが生まれ，コンピュータ

の活用範囲がより一層広まると考えられる．

しかし，現在のアプリケーションの多くは，他の環境と組み合わせて使おうとすると様々な不和が生じてしまう．例えば,「Web 上の絵を描き変えようとすると，PC 上で絵を編集し，再度ファイルをアップロードするなどの手間がかかる」「台所では PC を操作しづらいため，デジタル化されたレシピを活用しにくい」「PC や Web から家電機器を操作できない」といった問題がある．

このような「環境間の不和」には，やりたいことを直接的にできない「操作の複雑化」や，操作方法や外観が環境に馴染まない「環境との非調和」，環境間で情報をやり取りできない「情報の非流動」などがある．

2.3.1 操作の複雑化

異なるアプリケーションを組み合わせると，その機能が咬み合わず，操作が複雑になってしまうことがある．すなわち，余計な手順が増えて遠回りなやり方になってしまい，本来やりたいことを素早く直接的にできなくなるということである．また，この余計な手順に意識が向かうことで，本来やりたいことに集中できなくなる．

例えば，Web 上にある写真をペイントツールで加工して自分のブログに載せたいとする．この際，Web 環境と PC のデスクトップ環境という 2 つの環境に分かれることで,「写真を PC にダウンロードする」「ペイントアプリケーションを起動して写真ファイルを開く」「加工した新しい写真ファイルを保存する」「写真をアップロードする」という余計な手間が増える．そのため，写真を加工してブログに載せるという，本来やりたいことを素早く直接的に行えない．

2.3.2 環境との非調和

異なる環境同士を組み合わせようとすると，お互いの外観や操作方法が調和せず，違和感を生じさせたり，直感的に使えなくなることがある．例えば，広めのリビングに置かれたテレビに，Web ブラウザの画面を映して使おうとした場合，そのままでは「文字が小さすぎて読みにくい」「ソファなどに座るとマウスでポインティング操作をしにくい」「テレビを見ながら同時に Web ブラウジングできない」といった問題がある．これはすなわち，リビングというユビキタス環境と，Web 環境がうまく調和していないために，直感的に使えなくなっているということである．

2.3.3 情報の非流動

環境と環境の間ではデータのやりとりをしにくい．例えば,「PC や Web から家電機器のデータを操作できない」「情報機器（テレビやカメラ）から PC や Web 上のデータを扱いにくい」といった問題がある．

また，特定の「機能」が一部の環境に集中していることも，データの流動性を下げている.「動画編集」や「絵を描く」といった機能は現在のところ PC 環境にほぼ限定されており，そのことが他の環境での動画や画像のスムーズな流通を阻害している．

2.4 まとめ

本章では，本研究の背景として，コンピュータ技術が普及する過程で，コンピュータの様々な機能が「アプリケーション」という形でユーザに提供され，利用されるようになったことや，アプリケーションの「環境」について述べた．さらに，本研究の問題意識として，複数のアプリケーションを組み合わせたときに生じる環境間の不和について述べた．

第 3 章

溶け込むインタフェースの提案

概要

本章では，環境間の不和という問題に対する「溶け込むインタフェース」を提案し，その特徴を示す．そして，溶け込むインタフェースの有効性や課題を検証するシステムを試作するにあたり,「PC 環境」「Web 環境」「ユビキタス環境」という 3 つの利用環境について，それぞれアプリケーションを選定する．

3.1 はじめに

2章では，コンピュータ技術が普及する過程で，様々な機能が「アプリケーション」という形で利用されるようになったことや,「PC 環境」「Web 環境」「ユビキタス環境」という 3 つのアプリケーションの「環境」とその特徴について述べた．さらに，異なる環境を組み合わせようとしたときに生じる，環境間の不和について述べた．環境間の不和とは「操作の複雑化」「環境との非調和」「情報の非流動」といった問題に分けられる．

3.2 溶け込むインタフェース

アプリケーションを他の環境と組み合わせることで生じる「環境間の不和」を解決し，コンピュータの活用範囲をより一層広げるアプローチとして，本研究では「溶け込むインタフェース」を提案する．

溶け込むインタフェースとは，PC，Web，ユビキタスといった環境の違いを越えて，アプリケーションを他の環境に馴染ませるインタフェースデザイン手法である．環境に馴染ませるための様々なインタフェースデザインの工夫により，素早く直接的に操作できる「平易性」や，外観や操作方法が環境に馴染んで違和感が無い「親和性」，環境間でスムーズに情報が流れる「流動性」を高める．そして，様々なアプリケーションが自然な形で互いに混ざり合うことで，PC，Web，ユビキタスがあたかも融合した状態を目指す（図 3.1）．

3.3 本研究のアプローチ

本研究では，2 章で挙げた主要なアプリケーションの環境である「PC 環境」「Web 環境」「ユビキタス環境」について，それぞれの環境に溶け込むアプリケーションを提案，試作し，溶け込むインタフェースの有効性，課題を検証する．

しかし，アプリケーションの種類は多く，あらゆるアプリケーションについて検証することは困難である．そこで本研究では，それぞれの環境において不和を起こすアプリケーション，すなわち「操作の複雑化」「環境との非調和」「情報の非流動」という 3 つの問題を抱えているアプリケーションを選び，検証することとした．

3.3.1 PC 環境におけるソーシャルウェブサイト

PC 環境に溶け込ませるアプリケーションとして，Twitter や Facebook などの SNS，ブログ，ソーシャルブックマークといったソーシャルウェブサイトを選んだ．

ソーシャルウェブサイトは Web 上で情報共有する際，特に頻繁に用いられる Web アプリケーションである．様々なソーシャルウェブサイトが利用されるようになり，それら複数のサイトに発言が分散することで確認の手間が増えるようになった（操作の複雑化）．

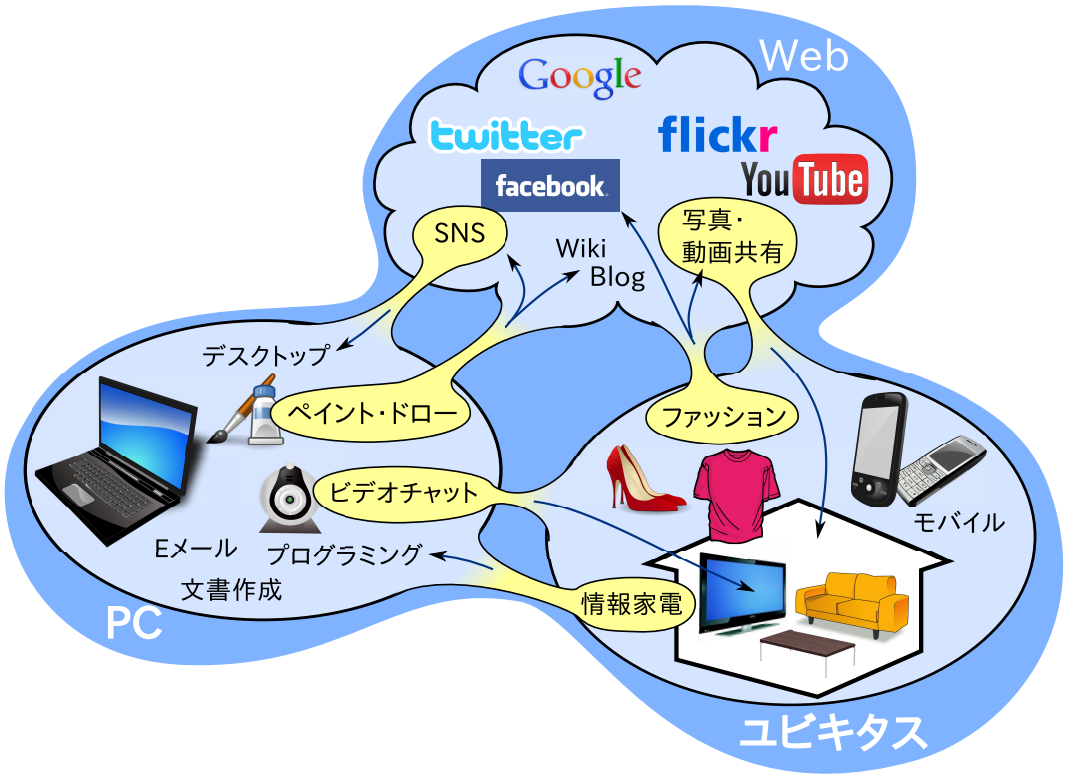

図 3.1: 溶け込むインタフェースのコンセプトイメージ

また，Twitter や Facebook のようにリアルタイム性の高いサイトでは，1 日に何度も情報確認するため，その度に Web ブラウザ上で意識的に更新情報を確認する必要がある．そのためデスクトップアプリケーションで作業している最中は，操作体系や見た目の異なる Web ブラウザとデスクトップアプリケーションを頻繁に切り替えることになる（環境との非調和）．

さらに，タイムラインをリストで表示するため，発言数の少ない人の発言が埋もれてしまう．PC 環境を利用している最中，すなわち Web ブラウザを開いていない状態ではこのような発言の有無に気づくのはさらに難しい（情報の非流動）．

本研究では，このようなソーシャルウェブサイトを PC 環境に溶け込ませる「ソーシャル顔アイコン」を提案する．

3.3.2 Web 環境におけるイラストツール

Web 環境に溶け込ませるアプリケーションとしてイラストツールを選んだ．

現在，コンピュータで絵を描くツールとしては，Adobe PhotoShop や Illustrator といった，PC 環境で動作するペイントツールやドローツールが一般的である．しかし，PC 環境で描いた絵をブログや SNS など Web 環境のアプリケーションに載せる場合，PC 上での画像ファイルの管理や，Web 上へのアップロードといった手間が発生する（操作の複雑化）．

また，デスクトップアプリケーションであるイラストツールの操作体系と，ブログや SNS などの Web アプリケーションの操作体系は大きく異なっている（環境との非調和）．

さらに一度 Web にアップロードされたイラストを他の人が改めて再利用/再編集するのは難しい（情報の非流動）．

本研究では，ブログや SNS といった Web アプリケーションにイラストツールを溶け込ませる「Willustrator」「TwitPaint」という 2 つのシステムを提案する．

3.3.3 ユビキタス環境におけるビデオチャット

ユビキタス環境に溶け込ませるアプリケーションとして，ビデオチャットを選んだ．

現在，Skype や iChat，Polycom などのビデオチャットアプリケーションが用いられている．これらのアプリケーションは，PC の側に座って会話し，アプリケーションの操作をするときは便利であるが，普段部屋の中を動きまわっている時など，PC から離れた状態で使おうとすると，操作や相手の状態の確認が面倒になる（操作の複雑化）．

また，生活空間でこれらのアプリケーションを利用しようとすると，PC の外観やマウス操作が生活空間での利用に馴染まないといった問題や，プライバシー保護が問題になる（環境との非調和）．

従来のビデオチャットは基本的に PC の前にいる時しか使用せず，一時的にしか通信は行われない．そのため部屋の様子や部屋の中での会話などは，お互いに伝わりにくい（情報の非流動）．

本研究では，このようなビデオチャットを生活空間に溶け込ませる「なめらカーテン」を提案する．

3.4 まとめ

本章では溶け込むインタフェースを提案した．溶け込むインタフェースでは，アプリケーションや利用環境の違いを越えて，コンピュータ技術が提供する本質的な機能に着目した．その上で，まるで溶け込むように自然な形で機能が混ざり合い，直接的でシームレスに情報を扱えるようにするインタフェースデザインのアプローチである．そして，溶け込むインタフェースは「平易性」「親和性」「流動性」の 3 つの特徴により，環境間の不和を解消する．

そして，溶け込むインタフェースの有効性や課題を検証するシステムを試作するにあたり,「PC 環境」「Web 環境」「ユビキタス環境」という 3 つの利用環境について，それぞれアプリケーションを選定した．

第 4 章

PC環境に溶け込む

概要

本章では，PC 環境に溶け込むインタフェースとして，PC 環境のデスクトップ上に Web 環境のソーシャルウェブサイトを溶け込ませる「ソーシャル顔アイコン」を提案する．ソーシャルウェブサイトは Web 上で情報共有する際，特に頻繁に用いられる Web アプリケーションである．様々なソーシャルウェブサイトが利用されるようになり，それら複数のサイトに発言が分散することで確認の手間が増えるようになった．また，Twitter や Facebook のようにリアルタイム性の高いサイトでは，1 日に何度も情報確認するため，その度に Web ブラウザ上で意識的に更新情報を確認する必要がある．さらに，タイムラインをリストで表示するため，発言数の少ない人の発言が埋もれてしまうといった問題が生じる．そこで，複数のソーシャルウェブサイトを横断しながら，優先的に見たい人がタイムラインに埋もれてしまうことなく，特定の人物を一目で見つけることのできるインタフェース「ソーシャル顔アイコン」を提案・試作した．ソーシャルウェブサイトの中にいる人をアイコンとしてデスクトップ上に溶け込ませることで人に対してより直接的にアクセスできるインタフェースを備えており，使い慣れたアイコンと同じように操作できる点や，発言の新鮮度や発言頻度といった時間軸情報を視覚的に分かりやすく表示する点が特徴である．

4.1 はじめに

近年の Web は人と人がつながりを持ちコミュニケーションをするソーシャルメディアとしての利用が進んでいる．Facebook[1] や mixi[2] などのソーシャルネットワークシステム（SNS）の国内会員数は延べ 7134 万人[3] となり，インターネット上の主要なメディアとして広く使われるようになった．またブログや写真・動画共有サービス，ソーシャルブックマーク，マイクロブログといった個人が情報を発信・共有するサービスの普及によって，ユーザ同士がネット上でつながりを持ち，コミュニケーションする機会が増えている．これらの SNS やブログ，コミュニティサイトを利用しているユーザは国内のインターネット利用者のうち 76.3%[4] に上る．

4.1.1 ソーシャルウェブサイト利用上の問題点

ソーシャルウェブサイトの利用者や利用頻度が増えた結果，Web ブラウザや Twitter/Facebook クライアントなどを用いた閲覧方法や，タイムラインをリストで表示するという UI に関して問題が生じるようになった．以下にその問題点の 4 つを挙げた後，それぞれについて詳細を述べる．

1. 発言数の少ない人が埋もれる
2. サイトごとに発言が分散する
3. 意識的に更新情報を確認する必要がある
4. 特定の人物を一目で見つけにくい

第一に，ソーシャルウェブサイトでは最新の発言を並べて表示するが，新着順に並べた場合，発言の多い人が目立ち，発言数の少ない人が埋もれてしまうことが多い．Twitter のリスト機能のように，人をグループ分けして一部の人たちの発言だけを表示することで，人が埋もれる問題を軽減できるが，発言を新着順に表示する限り，発言頻度によって一部の人が埋もれるという根本的な問題は解決していない．

第二に，複数のソーシャルウェブサイトに発言が分散すると，それを見る側は複数のウェブサイトを何度も訪れる必要がある．

第三に，Web ブラウザや Twitter/Facebook 専用クライアントなどを用いて情報（e.g., 特定ユーザの更新状況）を見る場合，ウィンドウやタブを開くなどの明示的な操作が必要になる．すなわち，これらのツールではユーザが意識的に情報にアクセスする必要があり，作業の合間に「ふと目に入る」というような無意識的な情報取得は難しい．

[1] http://facebook.com/
[2] http://mixi.jp/
[3] 総務省「ブログ・SNS の経済効果に関する調査研究」 (2009 年 7 月 13 日公表) http://www.soumu.go.jp/menu_news/s-news/16209.html
[4] インプレス R&D「インターネット白書 2009」

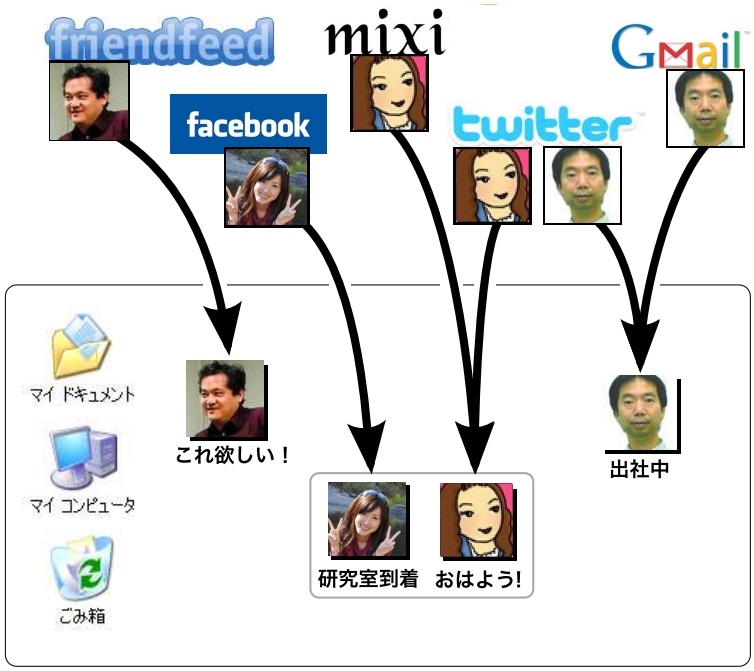

図 4.1: ソーシャル顔アイコンのコンセプトイメージ

第四に，ソーシャルウェブ上では全ての人の発言を一様に見たいとは限らず，親しい知人や有益な発言をする人といった特定の人物の様子や発言をより優先的に知りたいことも多いが，Web ブラウザや専用クライアントでは，そのような優先的に見たい人を一目で見つけにくい．デスクトップ上の時計やアイコンといった頻繁に見たい・アクセスしたい情報はディスプレイ上のほぼ同じ場所に置かれている（静的である）ため見つけやすいが，Web サイト内の情報やウィンドウ，タイムラインといった UI は，見たい情報の場所がたびたび移動してしまう（動的である）ため，頻繁に見たい情報にアクセスしづらい．

これらの問題は，既存の Web ブラウザに頼ってソーシャルウェブサイトを利用するために発生する問題と言える．そこで，ソーシャルウェブサイトをブラウザだけに頼らず，PC 環境に溶け込ませることにより上述した問題を解決する「ソーシャル顔アイコン」を提案する．

4.2 ソーシャル顔アイコン

ソーシャル顔アイコンとは SNS やブログなどのソーシャルウェブサイト上にいる人を顔アイコンとしてデスクトップに置くことで溶け込ませるというものである（図 4.1, 4.2）．顔アイコンごとにその人の最新記事やステータス，メッセージなどが表示され，アイコンをダブルクリックするとソーシャルウェブサイト内にあるその人のページを開いて見ることができる．また，これらの顔アイコンを動かしてくっつけることで手軽にアイコンを管理する．さらに，人のアイコンを二次元空間上に配置するだけでな

図 4.2: デスクトップに置かれたソーシャル顔アイコン

く，アイコンの透明度やエフェクトを変えることで，最近発言した人やあまり発言していない人，発言頻度といった時間軸情報を視覚的に分かりやすく表示する．

以降ではソーシャル顔アイコンのコンセプトおよび機能について述べる．

4.2.1 コンセプト

ソーシャル顔アイコンのコンセプトは，ソーシャルウェブサイトの中の「人」をデスクトップに溶け込ませることで「人を軸に見るアイコン型インタフェース」である．ソーシャルウェブサイトの中の「人」に着目し，普段使い慣れたデスクトップアイコンのメタファを取り入れることで，ウェブサイトや人に対してより直接的にアクセスできるインタフェースを考案した．

人を軸に見る

ソーシャルウェブサイトではいずれも個人ごとのページが存在するように,「人」が主要な構成要素となっているが，問題点 1 および 4 で述べたように「発言数の少ない人が埋もれる」「特定の人を一目で見つけにくい」といった，人を軸に見ることが難しいという問題がある．そこでソーシャル顔アイコンでは 1 人につき 1 つのアイコンを割り当て，知人などの優先的に発言を読みたい人を選んでデスクトップに並べておくことで，その人の最新の発言や状態は他の人の発言に埋もれることなく常時見ることができ，その人を一目で見つけられるようにした．また，ショートカットアイコンのようにデスクトップから Web 上の個人ページに対してより直接的にアクセスできるようにした．

さらに複数のソーシャルウェブサイトを横断して，同一人物や関連する人など「人」

を基準にまとめることで，問題点 2 の「サイトごとに発言が分散する」という問題を解決する．

アイコン型インタフェース

ソーシャル顔アイコンは「ドラッグ＆ドロップで配置」「ダブルクリックで開く」といった従来のファイルアイコンと同じような操作体系で使えるようにした．このようなアイコン型のインタフェースにすることで，多くの人が既に慣れた操作方法で使うことができ，既存のファイルアイコンなどに混じってデスクトップに置かれていても違和感が少ない．

問題点 3 で述べたように，ウィンドウやタブといった UI では「ユーザが意識的に」操作して更新情報を確認する必要があった．一方，ソーシャル顔アイコンではアイコンとして常にデスクトップ上に表示されることで，意識的に見ようとしなくても，ふとした作業の合間に目に入ってくるという特徴を持つ．

また，問題点 4 で述べたようにウィンドウやタイムラインを用いた UI では，発言がたびたび移動してしまうため，特定の人物の発言にアクセスしづらいという問題があった．ソーシャル顔アイコンは，アイコン画像の視認性に加えて，特定の人物の発言がディスプレイ内の絶対位置で固定されるため一目でその人を見つけることができる．

このようにアイコンの性質を活かした UI がソーシャル顔アイコンの特徴の一つとなっている．

時間軸情報の可視化

ソーシャルウェブサイト上の人を二次元空間上に配置・操作できるようにした場合，それだけでは発言の時間軸に関する情報が失われてしまう．そこで，発言の古さや発言の頻度といった時間軸の情報をアイコン表現により可視化することで，古い情報が埋もれないようにしつつ,「これはどれくらい前の発言なのか」「今誰が活発に発言しているか」「今イベントや議論で盛り上がっている」といった情報を分かりやすく表示するようにした．

4.2.2 機能と使い方

コンセプトを元に実装したソーシャル顔アイコンの主要な機能として,「顔アイコンを用いた発言表示」「くっつきアイコン」「廃れるアイコン」「ホットアイコン」を取り上げた後，細かな使い方の流れやその他の機能について述べる．

顔アイコンを用いた発言表示

ソーシャル顔アイコンの最も基本的な機能は，ソーシャルウェブサイトでつながりのある人をアイコンにしてデスクトップに置いて見る，というものである．

アイコンは図 4.3 のように，各個人ごとにアイコン画像と名前，Twitter や Facebook などのサービスを表す小さなアイコン，そして最新の発言や状態が表示される．通常

図 4.3: 顔アイコンを用いた発言の表示

状態では発言の一部だけ（20 文字程度）を読むことができ，発言に何か変化があれば分かるようになっている．

アイコンを選択またはマウスホバーすると全文を読むことができ，コンテキストメニューから過去の発言を見ることもできる．ただし，本文が画面内に収まりきらない場合はサマリーまたは本文の先頭部分だけが表示される．さらに続きを読みたいときはダブルクリックして Web ページを開く．

くっつきアイコン

アイコンをドラッグ＆ドロップして他のアイコンにくっつけることで，複数のアイコンをまとめることができる（図 4.4）．さらにまとめたアイコンは「重ねボタン」を押すことで，Mander ら [Mander, 1992] の提案する Pile Metaphor のようにアイコンを重ねて小さく表示できる．アイコンを重ねた状態では，それらのアイコンの中で最新の記事やメッセージが表示される．

くっつきアイコン機能は以下のような用途を想定している．

- 同一人物が複数のサービスを使っている場合，それらをまとめて 1 人 1 アイコンにする．
- 関連する人たちのアイコンをグループ化する．
- 優先度の低い人たちは重ねて小さく表示しておき，見たいときだけ展開する．

廃れるアイコン

各アイコンは最後の発言からしばらく時間が経つと，徐々に薄くなる．図 4.5 のように 1 日以内の発言ははっきりと表示されるが，数日・数週間・数ヶ月と古くなるごとにアイコンの透明度が増し，1 年以上前の発言は最低限文字が読める程度に薄くなる．

図 4.4: くっつきアイコン

情報の鮮度を視覚的に分かりやすく表示することで，その人の発言がだいたいいつ頃のものかをすぐに判別できるようになり，たくさんのアイコンを一覧したときに新しい発言を見つけやすくなる．

ホットアイコン

頻繁に発言している人のアイコンは，赤いグローエフェクトがかかることで活発（ホット）な様子を表す (図 4.6)．実際には直近 1 時間以内に一定回数（デフォルトでは 5 回）以上発言した場合，このエフェクトがかかるようになっている．

何かのイベントの最中や誰かと議論している時は，短時間に集中して発言することが多いことを利用して,「普段とは違う何か特別なことが起こっている」ことを知ることができる．

図 4.5: 廃れるアイコン

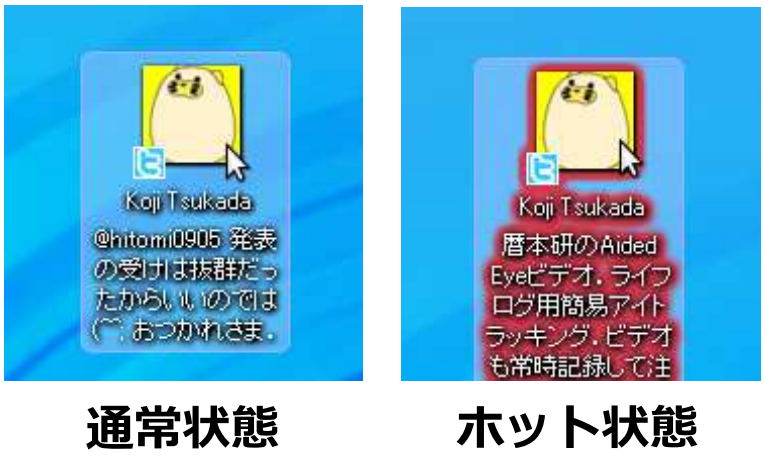

図 4.6: ホットアイコン

使い方の流れとその他の機能

ソーシャル顔アイコンの細かな使い方の流れを説明するとともに，周辺的な機能についても紹介する．

新しく顔アイコンを作成するには，後述する顔アイコン Dock 内のボタン（図 4.8 右のボタン）または通知領域のアプリケーションメニュー（Windows のみ）から友達一覧ウィンドウを開き，追加したい人をデスクトップにドラッグ＆ドロップする（図 4.7）．友達一覧ウィンドウには Facebook，Twitter[5] ，Friendfeed[6] などでつながりのある人が表示され，ブログなどの RSS を独自に追加することもできる．

デスクトップに配置した顔アイコンの発言はおよそ 2 分間隔で自動更新され，発言が無ければ徐々に廃れた状態になり，頻繁に発言すればホット状態になる．

アイコンの数が増えてきたときは，くっつきアイコン機能を使い，関連する人をひとまとめにするなどしてアイコンを整理する．またくっつけたアイコンはまとめてドラッグ＆ドロップして動かすこともできる．

くっつきアイコンの他にアイコンを管理するための機能として，Mac OS X の Dock のように画面の端にアイコンを並べられる領域「顔アイコン Dock」を設けた（図 4.8）．

[5] http://twitter.com/
[6] http://friendfeed.com/

図 4.7: 顔アイコンの新規作成

図 4.8: 顔アイコン Dock

デスクトップにアイコンを置くとウィンドウの下に隠れてしまうが，アイコンを Dock にドラッグ＆ドロップして置くことでウィンドウが開いた状態でも常時または素早く最前面にアイコンを表示することができる．頻繁に見たいアイコンは Dock に置くと良い．

顔アイコンを削除する時は，アイコンを右クリックしてコンテキストメニューから「Delete」を選択する．

4.3 実装

くっつきアイコンやホットアイコンのように通常のアイコンではできない表示方法や操作を実現するため，ソーシャル顔アイコンでは枠のない透過ウィンドウに仮想のアイコンを描画している．1 アイコンまたは 1 グループにまとめられたアイコンはそれぞれ 1 つの透過ウィンドウになっており，これらのウィンドウを他のウィンドウの背面に置くことで，デスクトップ上に顔アイコンが置かれているように見せている．この実装上の制約によりフォルダに入れるといった通常のアイコンと完全に同じ操作はできないものの，実用上はファイルアイコンやフォルダと混ざった状態でも問題なく使

うことができる．

顔アイコンに使用する画像や記事，ステータス，メッセージなどはTwitterやFacebookなど各サービスで提供されているAPIを通じて取得している．

これらはAdobe AIRで実装しており，Adobe AIRランタイムが動作する環境（Windows，Mac OS X，Linux）で利用できる．

4.4 議論

2009年12月3日より，ソーシャル顔アイコンはWebでオープンソースソフトウェアとして公開しており[7]，自由にダウンロードして利用できる．

ソフトウェア公開後は，窓の杜[8]やYahoo!ニュース[9]といったサイトや，ソフトウェアを紹介する雑誌などで取り上げられ，Twitter上でもソーシャル顔アイコンに関して200人以上のユーザから言及があった[10]．

利用者の意見を元に機能の追加や改良を行っているため，機能によって試した期間が異なるものの，我々を含む利用者が半年以上利用した上で得た知見や興味深い利用方法から有効性や課題などを議論する．

4.4.1 活用事例と考察

実際にソーシャル顔アイコンがどのように活用されたのか，いくつかの事例を元に考察を行う．

利用頻度の低いサイトの閲覧

ユーザの1人はTwitterを中心に利用していたが，Facebook上にはつながりのある友達が少なく投稿や閲覧をあまりしていなかった．ソーシャル顔アイコンを利用してFacebook上の友達をデスクトップ上に置いておくことで，Facebookにアクセスせずに Facebookにいる友達の近況も知ることができるようになった．ソーシャル顔アイコンのサイトを横断して人単位で見ることができるという特徴により，サイトの利用頻度に関係なく複数のサイトを見られるようになった効果と言える．

ただし，現状ではmixiのような日本国内でメジャーなサイトにソーシャル顔アイコンが対応していないなど利用できるサイトが限られており，さらに他のサービスに比べてTwitterの利用者の多く，ユーザが偏っていたということもあり，積極的にこの機能が利用されていたとは言えない．サイト横断の特徴を活かすためには対応サイトを増やすことが課題である．

[7] http://sappari.org/faceicon.html
[8] http://www.forest.impress.co.jp/docs/review/20100902_390982.html
[9] 2010年9月2日の記事．現在は削除されている．
[10] Topsy (http://topsy.com/) を用いて調査

知人の様子を知る

利用者の一人から「Twitter で今日会う予定の知人の様子を調べようとしたが，発言数の多すぎる人がいてなかなか見つからなかった．顔アイコンだとすぐ分かった」というコメントがあった．これは「見たい人が埋もれて見えない」という問題を解決した例と言える．

また，このように人の様子を確認する際，ほとんどの場合ユーザは顔アイコン上のテキストを見るだけであり，たまに右クリックをして過去の発言も見るというような使い方をしていた．発言の前後関係などを含めてさらに詳しく知りたい時，顔アイコンをダブルクリックしてブラウザを開いてみることもあったが頻度としては多くなかった．このことから，ブラウザを開くための単なるショートカットしての利用よりも，顔アイコンによって一目で情報（人）を見つけるためのツールとして主に利用されたと言える．これは「特定の人の発言を一目で見つけにくい」という問題の解決にあたる．

くっつきアイコンの多用

くっつきアイコンは多くのユーザが利用しており，主に「同じ職場の人をグループ化してまとめる」「頻繁に見る人をまとめる」「発言数の多い Bot（プログラムによる自動投稿）をまとめる」といった分類に用いられていた．このことからソーシャルウェブサイト上にいる人や Bot の分類表示には一定の需要があると考えられる．また，くっつけたアイコンはまとめて移動できるためアイコンが増えたときに整理しやすいことも理由として考えられる．これらの用途は，Twitter でフォローしている人を分類するためのリスト機能と用途が近いため，将来的にはリスト機能と連携することでより手軽に分類するといった実装も検討している．

逆に，機能の設計時に想定していた「同一人物をまとめる」という使い方はあまりされなかった．これは同種の複数のウェブサイトを使い分けるヘビーユーザの数は少ないということと，ソーシャル顔アイコンの対応しているウェブサイトがまだ少ないと言うことが原因として考えられる．

Twitter Bot の積極的な利用

興味深い使い方として，図 4.9 のように Twitter Bot をたくさんデスクトップに並べて見るという使い方をしているユーザがいた．Twitter Bot とは，Twitter の Web API を利用し様々な情報を機械的な自動投稿により擬人的に発言しているユーザアカウントであり，Twitter の UI を用いて人の発言以外の情報を通知できるという便利さや面白さにより，多くの Twitter Bot が作られている．図 4.9 左上のように Twitter Bot の一つであるプロ野球チームの試合状況を通知する Bot をデスクトップに置くことで試合の最新情報を見られるようにしたり，同様に天気やニュース，サイトの更新情報をアイコンにしてすぐにチェックするといった使い方をされていた．

このような Twitter Bot をいくつも置くという利用方法は「タイムラインが埋もれない」というソーシャル顔アイコンの特徴を活かしたものと言える．機械的に投稿する Bot の多くは人による投稿に比べて発言数が多く，タイムラインを不要に埋める原

図 4.9: Twitter Bot を並べた例

因の一つになっていた．上で述べたプロ野球チームの試合状況を通知する Bot も試合中は数分おきに発言するため，発言頻度の多い Bot の一つである．このような Bot をいくつも follow する[11] と，それだけ Bot の占める発言が増えるため，follow する Bot の数はある程度制限する必要があった．ソーシャル顔アイコンを利用することで発言数の少ない人が Bot を含む他の発言に埋もれる問題は解消されるため，Bot の数や発言頻度を気にすることなく follow することができ，より多くの Bot を活用できる UI だと言える．

廃れるアイコンの効用

廃れるアイコンは利用者の意見を元に追加した機能の一つであるが，機能の導入後，何日も発言が滞っていたり発言頻度が少ないために色が薄くなっていた人のアイコンがある日はっきりと表示されると自然と目にとまりやすくなり，「久々の発言を見つけやすくなった」という意見があった．

廃れるアイコン機能の設計時に意図していた「いつ頃の発言か素早く判別できる」「たくさんのアイコンを一覧したときに新しいものを素早く見つけられる」という効果に加えて，久々の発言をより見つけやすく・埋もれにくくする効果があると言える．

また，発言数が極端に少なくアイコンが消えかかっている人は，そのサイトをもうほとんど使わなくなってしまったと考え，デスクトップから削除するということがあった．このようにどの人をデスクトップに置く/置かないのかという判断基準になった．

[11] 発言を受け取れるように登録申請する

図 4.10: プロ野球の試合状況を通知する Bot

イベントの発生を知る

自分と関係の深い研究者らのアイコンが頻繁にホット状態になったことで，それまで自分の知らなかった学会イベントの開始を知り，それをきっかけに遠隔地からその議論に参加したという事例や，その会議中に面白い発表があると複数の人が同時にホット状態になり，そこにいる人たちが何に強い関心を持ったのかを知ることができるようになったという事例があった．同様に，Twitter Bot の項で挙げたプロ野球の試合状況を発言する Bot (図 4.10) がホット状態になることによって試合が始まったことを知るという例や，新製品発表のニュースで多くの人が盛り上がっているのをホット状態によって知るといった例があった．

これまで何かのイベントの発生を知るには，たまたまそのとき発言内容を読んで知るか，タイムラインを長時間見続けて発言頻度の増減から伺い知るしかなかったが，ホットアイコンを利用することにより，興味のありそうなイベントの発生をより高い確率で知ることができるようになったと言える．また，発言内容を読む前に，画面をパッと見たときの色の変化だけでイベントの有無を知ることができるようになり，そのとき読む価値のある情報を重みづけして提示する仕組みとしても有効と言える．

デスクトップに置けるアイコン数の問題

デスクトップ上に多くのファイル/フォルダ/ショートカットアイコンを置いているユーザの場合，ソーシャル顔アイコンを置くスペースがほとんど残されておらず，ソーシャル顔アイコンを活用できないという問題があった．デスクトップに置けるアイコンの数は画面解像度などに応じて限られるため，ソーシャル顔アイコンで置ける顔アイコンの数も必然的に限られてしまうことが原因である．これはソーシャル顔アイコンの問題点であると同時に，現在のデスクトップ環境がそもそも抱える問題であるため根本的な解決は難しいが，くっつきアイコンの重ねる機能によってよりコンパクトに表示したり，顔アイコン Dock に置けるようにすることで，ある程度の問題の軽減を図っている．Mac OS X のデスクトップ Widget などのように，マウスジェスチャやキーボード操作によって一時的にソーシャル顔アイコンを表示または消すことで対処するという方法も考えられるが，コンセプトで述べた「特別な操作無しに自然と目に

入る」という特徴が失われてしまうというトレードオフがある．

一方で，デスクトップにある程度余裕があれば，顔アイコンの数は実際はあまり問題にならなかった．デスクトップに置いて何度も繰り返し見たいという人の数はせいぜい数人-数十人程度であり，十分デスクトップ内に置ける数であることが理由として考えられる．

4.4.2 Web ブラウザや専用クライアントとの関係および比較

ソーシャル顔アイコンのユーザは，実際にはソーシャル顔アイコンだけを使っていたわけではなく，従来通り Web ブラウザや Twitter/Facebook 専用クライアントなどを併用してソーシャルウェブサイトを閲覧していた．ソーシャル顔アイコンは Web ブラウザなどを用いたソーシャルウェブサイトの閲覧手法を置き換えるものではなく，従来の Web ブラウザなどが苦手としていた閲覧手法を提供することで，お互いに苦手とする部分を補い合う関係にあると言える．

以下では，それぞれの閲覧手法の特徴を比較すると共に，利点や欠点，互いの補完関係について述べる．

時間軸と人物軸

Web ブラウザで見るソーシャルウェブサイトは基本的にタイムラインという時間軸で情報を見ていたのに対し，ソーシャル顔アイコンは主に人物を軸にして見る．

人物を軸に見た場合,「4.4.1 知人の様子を知る」で述べたように，ある特定の人の発言や様子をすぐに知ることができるが，一方で複数の人の発言の順序や対話関係が分かりにくくなるというトレードオフがある．

しかし，著者らの周囲のユーザの利用方法を見る限り，対話関係が分かりにくいという点はそれほど大きな問題にはなっていない．これは，ソーシャル顔アイコンを利用するユーザの主な関心･目的が，ユーザ間の対話よりも，特定の個人をアイコン化して見やすくする点にあるためだと思われる．すなわち，ユーザは SNS 上の身近な人物の変化を手軽に確認する目的で，一種の（内容込みの）更新チェッカーとしてソーシャル顔アイコンを利用しており，対話など SNS 上の全ての情報を見ようとはしていないと考えられる．あまり頻繁に利用はされなかったが，もし対話の内容を詳しく見たい場合でも，顔アイコンをダブルクリックしてブラウザを開けばすぐに確認できるため，従来のタイムラインと容易に共存できる．現状では実装に至っていないが，対話の状況もある程度見やすく表示する方法として,「対話があったときだけ，関係する人のアイコンを近くにポップアップ表示する」「右クリックして過去の発言を表示したとき，その人の発言だけでなく対話も同時に表示する」などの新機能を検討している．

複数のソーシャルウェブサイトの利用方法

日本国内では mixi や Twitter，GREE[12] といった SNS が多く利用されており，世界的には Facebook や Twitter，MySpace[13] などのユーザ数が多い．そして，これら複数のソーシャルウェブサイトのアカウントを持っている人も多くいる．ソーシャル顔アイコンでは「4.4.1 利用頻度の少ないサイト」で述べたように，これら別々のサイトに属している人をサイトの利用頻度に関係なく同時に閲覧できることが利点である．

しかし，全てのサイトを全く同じように閲覧できるわけではなく，実際にはサイトごとに機能やコンテンツといったサービス内容に差があるため，そのサービスを十分に利用する場合は Web ブラウザでサイトに訪れる必要がある．Web ブラウザだけでは使うサイトと使わないサイトがはっきり分かれてしまうため，ソーシャル顔アイコンを使うことによって Web ブラウザを併用しつつ緩やかに複数サイトを同時利用するという関係になる．

TweetDeck[14] や HootSuite[15] といったアプリケーションを利用することで，複数のソーシャルウェブサイトのタイムラインをリストでまとめて（または簡単に切り替えて）見ることができる．これらのアプリケーションと比較すると，ソーシャル顔アイコンは「タイムラインではなく人物を軸にして見る」「リストではなくアイコンを使った UI」といった点で，ツールの利用方法や目的が異なっている．

ウィンドウとアイコン型 UI の比較

ソーシャル顔アイコンは意識的な操作が必要なウィンドウなどの UI とは違い，「ふと目に入る」ような無意識的な見方ができるが，欲しい情報がいつも出てくるわけではなく，たまに面白そうな情報が現れることを期待して見るため，期待したような情報が見つかるまで時間がかかる．一方，Web ブラウザは意識的・能動的に操作する必要がある代わりに，欲しい情報に素早くアクセスできる．このように，ソーシャル顔アイコンのようなアイコン型インタフェースは，ウィンドウなどの UI と比べ，静的でゆっくりとしたタイムスパンで利用する UI と言える．

ただし，ソーシャル顔アイコンは常に無意識的な閲覧をするわけではなく，より詳しく情報を見たいときは 「4.2.2 顔アイコンを用いた発言表示」 で述べたような操作で発言内容を詳しく表示したり，Web ブラウザを起動することで能動的な閲覧への移行もサポートする．

ソーシャル顔アイコンはアイコン型の UI によって常に同じような情報を表示し続けたり，ドラッグ＆ドロップや右クリックなどのマウス操作が自然にできることが利点となっている．一方で，小さなアイコン上で表現できる情報は非常に限られる．ソーシャル顔アイコンではアイコンの透明度やグローエフェクトによって時間軸情報を可視化しているものの，さらに多くの情報を表示しようとすると，逆に情報が見づらく分かりにくいものになってしまうと考えられる．Web ブラウザであればウィンドウ内

[12] http://gree.jp/
[13] http://www.myspace.com/
[14] http://www.tweetdeck.com/
[15] http://hootsuite.com/

や複数ページ遷移によって多くの情報を表示することができ，HTML や Flash などを多用した多用な表現ができるため，アイコンで表示しきれない情報を Web ブラウザによって補うという関係になっている．

リストとアイコンの比較

ソーシャルウェブサイトや多くの Twitter クライアントはタイムラインをリストで表示する．一方，ソーシャル顔アイコンはリストではなくアイコンを利用する点で大きく異なっている．リストと比べたアイコンの長所と短所を以下に整理する．

- アイコンの長所
  - 画像の一覧に向いており，画像によって各項目を識別しやすい．
  - 二次元配置によって，関連する項目を近くにまとめたり，逆に関連しないもの同士を遠ざけて配置できる
- アイコンの短所
  - 長文テキストの一覧に向かない．
  - アイコンが増えすぎると一覧しにくい．
  - アイコンを整列配置しない場合，時系列のような順序情報が失われる．

このようにアイコン方式には長所と短所ともに存在するが，人を見つけやすくするというソーシャル顔アイコンの目的に対してはアイコン方式が適している．アイコンの長所により，リスト方式に比べて「顔アイコン画像によって人を識別しやすい」「二次元配置で関連する顔アイコンをまとめて見やすく表示できる」といった強みを持つ．特に，ソーシャルウェブサイト上においてもアイコンの画像で人を識別するように，アイコン画像は人を見つける手がかりとして重要な役割を果たしている．

一方でアイコンの短所により,「多くの発言を読むには向かない」「あまり多くの顔アイコンを置くと一覧しにくい」「発言の前後関係が分かりにくい」といった問題も起こる．これらの問題に対しては，Web ブラウザや他のクライアントと併用する/くっつきアイコンで整理する/ホットアイコンや廃れるアイコンで時間情報も表示するなどの方法で軽減できる．さらに発言の前後関係については,「4.4.2 時間軸と人間軸」で述べたような対話の視覚化手法を検討していく．

4.5 関連研究

顔アイコン [Takabayashi, 2003] では，顔の形をしたアイコンにファイルをドラッグ＆ドロップすることで，メールにファイルを添付する操作に比べて手軽で分かりやすいファイル転送を実現した．ソーシャル顔アイコンでは，人ごとにデスクトップアイコンを置くという顔アイコンの UI を参考にしながら，複数のソーシャルウェブサイトを横断して人単位でアクセスするという用途に応用した．

iPhone/iPod touch 用の Twitter クライアント Reportage[Reportage, 2011] では，ユーザのアイコンを並べておくことで，タイムラインの流れの早さに関係なく重点的に読みたい人の発言をすぐに読むことができる．人のアイコンを配置することで発言を埋もれないようにするというコンセプトや機能がソーシャル顔アイコンと近い．ソーシャル顔アイコンでは，くっつきアイコンや時間軸情報の可視化によって人の管理や閲覧を容易にし，複数のソーシャルウェブサイトを同じ UI/操作で見られるようにすることで，サイトの違いを意識させず，より人を中心に見やすくするための仕組みを提供している．

Web ブラウザを起動せずデスクトップ上で Web ページを見るソフトウェアとしては，Microsoft の ActiveDesktop や Apple の Web Clip ウィジェット [Apple, 2011] がある．また Windows Vista のサイドバーガジェットや Mac OS X の Dashboard といったデスクトップウィジェットを用いて Web の情報をデスクトップに表示することができる．ソーシャル顔アイコンはこれらの技術に近いが，Web の中でもソーシャルウェブサイトを対象にすることで顔アイコン型インタフェースを実現し，利便性を高めた．

Web アプリケーションをデスクトップアプリケーションのように起動して使えるようにする Web ブラウザの拡張機能として Mozilla Prism[Mozilla, 2011] がある．ソーシャル顔アイコンでは Web アプリケーションではなくその中にいる人のアイコンをデスクトップに置くことで直接その人のページを開くことができるほか，人のアイコンに最新記事などの更新情報を表示できる点が異なる．

ブログやソーシャルブックマークなどの記事を自動的に収集・一覧することで，Web 閲覧の負担を減らすツールとして RSS リーダーがある．RSS リーダーは最新記事をいち早く逃さず見たいという積極的・能動的に Web を見るユーザに向いているが，閲覧や管理に多くの操作が必要であり，できるだけ楽に使いたいユーザには敷居が高い．

永田ら [Nagata, 2007b] は，SNS やグループウェア，メーリングリストなどを統合したウェブサイト Enzin 内で，参加者のアバターアイコンをドラッグ＆ドロップして配置を変えることで情報の公開範囲を簡単に設定できるようにした．Enzin を利用しているユーザに限られるものの，人のアイコンをドラッグ＆ドロップでまとめることにより SNS 内の情報を管理する UI がソーシャル顔アイコンと似ている．

Time-Machine Computing[Rekimoto, 1999a] の提案で実装された時空間デスクトップ環境 TimeScape では，最終更新時刻の古いファイルほどアイコンを薄く表示することで，ファイルの鮮度を視覚的に表現した．また，塚田らの廃れるリンク [Tsukada, 2002] では，Web ページのリンク先情報の更新日時に応じてリンクに色あせる・掠れる・にじむといったエフェクトをかけることで鮮度を表現している．ソーシャル顔アイコンの廃れるリンクもこれらと同様に情報の鮮度を視覚的に表現しており，廃れるリンクが対象とした Web 上の情報の鮮度を，TimeScape のようにファイルのアイコンとして表現していると言える．

4.6 まとめと今後の課題

本章では，PC 環境に溶け込むインタフェースとして，PC 環境のデスクトップ上に Web 環境のソーシャルウェブサイトを溶け込ませる「ソーシャル顔アイコン」を提案した．ソーシャル顔アイコンにより，複数のソーシャルウェブサイトを横断しながら，優先的に見たい人がタイムラインに埋もれてしまうことなく，特定の人物を一目で見つけることのできる．

今後は利用できるソーシャルウェブサイトを増やすことや，TimeScape で実現されている，より詳細な時間軸情報の提示や時間移動を取り入れることで時空間をスムーズに切り替え，ソーシャルウェブサイトをより新しい視点で使える・楽しめるようにしたいと考えている．

第 5 章

Web環境に溶け込む

概要

本章では，Web 環境に溶け込むインタフェースとして，Web 環境のブログや SNS に PC 環境のイラストツールを溶け込ませる「Willustrator」「TwitPaint」という 2 つのシステムを提案する．現在，コンピュータで絵を描くツールとしては，PC 環境で動作するペイントツールやドローツールが一般的である．しかし，PC 環境で描いた絵をブログや SNS など Web 環境のアプリケーションに載せる場合，PC 上での画像ファイルの管理や，Web 上へのアップロードといった手間が発生する．また，デスクトップアプリケーションであるイラストツールの操作体系と，ブログや SNS などの Web アプリケーションの操作体系は大きく異なっている．さらに一度 Web にアップロードされたイラストを他の人が改めて再利用/再編集するのは難しいといった問題がある．そこで，Web 上にイラストツールを溶け込ませることで，Web ブラウザ上で絵を編集でき，他の人の描いた絵から「派生」させて新しい絵を作成することのできる 2 種類のシステム「Willustrator」と「TwitPaint」を提案，構築する．ドロー系である Willustrator は絵を再編集/再利用しやすいという特徴を持つ．一方，ペイント系の TwitPaint は Twitter と連携した絵によるコミュニケーションを特徴とする．両ツールを長期運用し，得られた派生データや事例，知見などを元に，相違や特徴，問題点を分析した．さらに今後の Web 上で編集/派生可能なイラストツールの課題や将来の可能性についても検討を行った．

5.1 はじめに

Web 上では人々によって描かれた多くの絵やクリップアート，図，イラストが公開/利用されている．中でもイラストの共有に特化したコミュニティサイトは昔から数多く運営されており，海外では deviantART[1] ，日本国内では Pixiv[2] といったサイトが特に人気を博している．このようなイラスト投稿サイトはアーティストやイラストレーター，漫画家といった専門家や絵を描くのが得意な人（熟達者）を中心に利用されているが，絵を描くスキルや専門知識をあまり持たない人（非熟達者）によるイラストの投稿や共有はあまり行われていない．

一方，非熟達者であっても仕事や趣味，コミュニケーションの中で絵を活用する機会は多い．ワープロの文書やブログの記事，プレゼンテーション用のスライド，ポスターやパンフレットを作る際には絵が頻繁に用いられる．そして Adobe Photoshop や Illustrator，Visio[3] ，OmniGraffle[4] のようなグラフィックスソフトウェアや，Microsoft Word や PowerPoint のようなドキュメント作成ソフトウェアの普及によって，非熟達者でも絵を利用することは珍しくなくなった．

5.1.1 Web 環境におけるイラスト利用の問題

PC 環境で描かれた絵は複数人での協同編集や，他の人の描いた絵を再利用するといったことが難しく，さらに自分の描いた絵を再利用しやすい形で共有することも面倒である．ブログや SNS，Wiki などの Web アプリケーションによって情報共有やコミュニケーションの敷居は下がったものの，Web 上での「非熟達者による絵の共有」や「絵の協同編集，再利用」はまだ発展途上と言える．

デスクトップアプリケーションを使って描かれた絵が共有されにくい理由の一つに，ローカル PC から Web 上へのファイルのアップロード作業が面倒なことが挙げられる．テキスト情報であればブログや Wiki などを利用してブラウザ上ですぐに編集し直すことができるが，絵は編集し直す度にファイルをアップロードする必要がある．また，ファイル名をつけて分類保存するといったファイル管理の手間も，ブラウザ上でのテキスト編集に比べて面倒な理由の一つである．

5.1.2 Web 環境に溶け込むイラストツール

もし，Web 上で公開/共有されている絵を手軽に再利用でき，さらに他の人が再利用しやすい形で簡単に共有することができれば，より便利で手軽に絵を活用できると考えられる．

そこで我々は，Web 環境にイラストツールを溶け込ませることで，「Web ブラウザ上で絵を直接編集でき，他の人の絵を元に再編集・再利用を可能にする」２種類のシス

[1] http://www.deviantart.com/
[2] http://www.pixiv.net/
[3] http://office.microsoft.com/ja-jp/visio/
[4] http://www.omnigroup.com/applications/omnigraffle/

テムを開発した．一つはドロー系の「Willustrator」で，もう一つはペイント系でかつ Twitter との連携を特徴とする「TwitPaint」である．この両ツールを一般に公開し長期運用することで，Web 上での絵の作成や再利用に関するデータや知見をまとめ，今後の課題や将来の可能性についても検討を行う．

5.1.3 絵の派生

Willustrator と TwitPaint はともに「他の絵から派生させて新しい絵を描ける」ことが大きな特徴となっている．ここでは以下の 2 つの条件を満たすことを「絵の派生」とする．

- ある絵を複製後，再編集して，元の絵とは別の新しい絵を作成する
- 再編集して作られた絵と，その元になった絵が相互にリンクされる

1 つめの条件により，他の絵を再編集する際に上書きしてしまうのではなく，一度複製してから編集することで，元の絵を残したままにすることができる．また 2 つめの条件により，再編集してできた絵と元になった絵が相互にリンクされることで，あとから元になった絵や，逆にその絵を元にして描かれた絵を辿って見ることができるようになる．

派生が次々と繰り返されることで，1 つの絵が様々に変化・分岐してゆき，複数の人がその一連の流れに関わることになる．本研究ではこの「絵の派生」を特徴とする両ツールを実際に運用することで，絵の変化・分岐の課程を楽しむという「新しい絵の楽しみ方」や，絵の派生を活用した「新しい表現の可能性」，絵と絵・作者と作者がつながることによる「新しいコミュニケーションの発生」といった，様々な可能性を探った．

5.2 Willustrator

Willustrator とは Web ブラウザ上で使えるドローツールである（図 5.1）．絵のベクターデータを Web 上で保存･共有することで，他の人の描いた絵を手軽に再利用して新しい絵を描くことができる．また，ドロー系ツールであるため，ペイント系ツールに比べて再編集しやすいことが特徴となっている．Willustrator は現在 http://willustrator.org/ にて運用している．

5.2.1 機能と使い方

Willustrator の描画機能やサイトの利用方法，再利用のための機能について述べる．

画像の一覧と新規作成

Willustrator にログインすると，そのユーザの個人ページ（図 5.2）が表示され，自分の描いた画像を一覧できるほか，新規作成ボタンを押すと新しい絵を描ける．

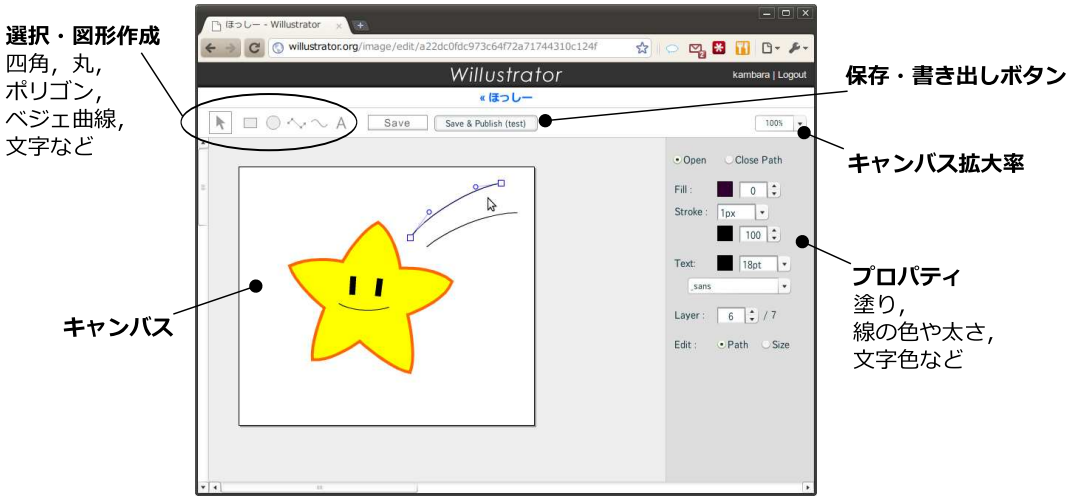

図 5.1: Willustrator の画像編集画面

描画機能

Willustrator の画像編集画面は図 5.1 のようになっている．丸や四角，ポリゴン，テキストといった基本的な図形のほかに，図 5.3 のようにベジェ曲線を編集することで複雑な図形も描くことができ，図 5.4 のように全ての図形の中にはテキストを書ける．図 5.5 のようにフリーハンドツールを使うことで，ベジェ曲線に不慣れなユーザでも簡単にきれいなベジェ曲線を描けるようになっている．

また図 5.6 のようなグリッド・スナッピング，コピー＆ペースト，拡大縮小といった，ドローツールとしての基本的な機能を備えている．

絵の派生による再利用

Willustrator の特徴の一つは「他の絵から派生させることで再利用できる」ことである．例えば図 5.7 上の絵を元にして描きたい場合，派生ボタンを押すとその絵が複製され，編集することができる．また，派生元になった絵や，派生によってできた絵をリンクで辿って見ることができる（図 5.7）．

5.2.2 実装

Willustrator の Web アプリケーション（サーバ）は Ruby on Rails で実装し，ユーザ認証にははてな認証 API[5] を用いた．

画像編集機能は Adobe Flash で実装した．ベクター画像の保存形式は独自の XML になっている．このベクター画像の PNG 形式へのラスタライズはクライアント側 (Flash) で行い，生成した PNG 画像をサーバへ送信・保存している．

[5] http://auth.hatena.ne.jp/

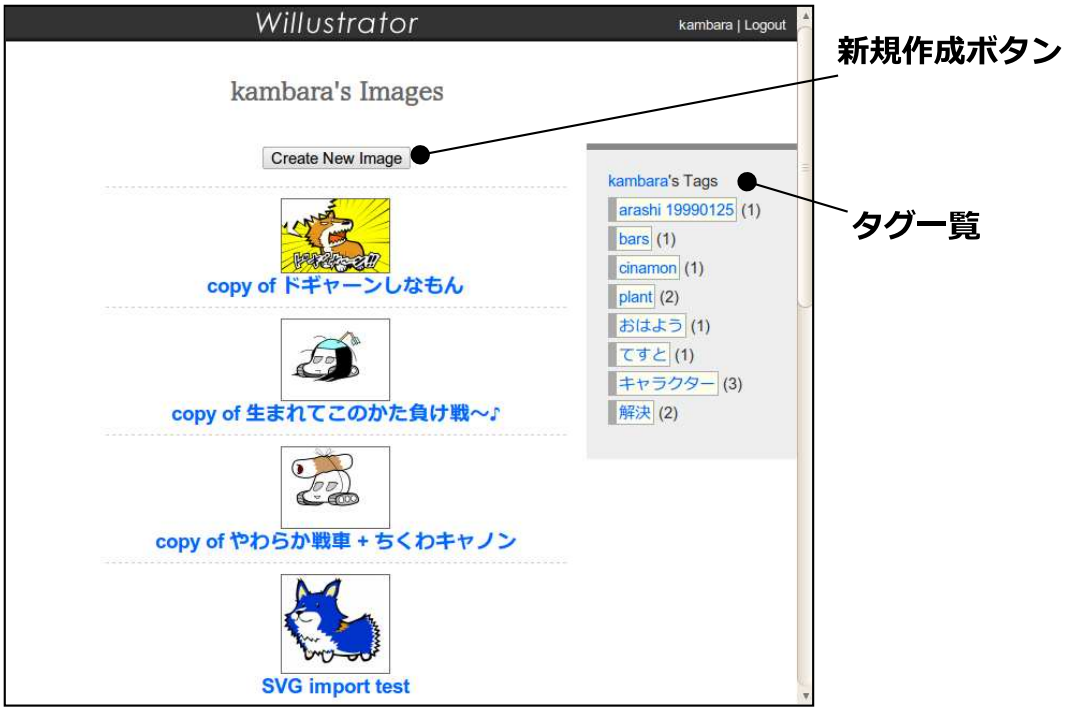

図 5.2: Willustrator のユーザ個人ページ

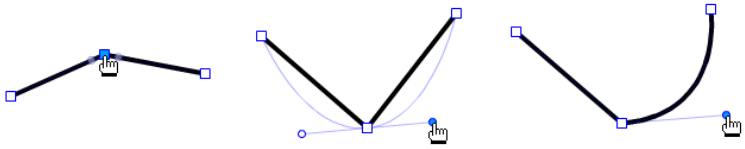

図 5.3: ベジェ曲線の編集（Willustrator）

5.3 TwitPaint

TwitPaint とは，Web ブラウザ上で絵を描けるペイントツールであり，描いた絵を自動的に Twitter に投稿することで，絵を通じてコミュニケーションすることができる．さらに，他の絵から派生させたのち再編集する（TwitPaint 内では Remix と呼ぶ）ことができ，この派生を活かすことで，複数のユーザが参加する絵の作成や，絵を使った遊び，Twitter には無いコミュニケーションを可能にする．

TwitPaint は現在 http://twitpaint.com/ で運用している．

5.3.1 機能と使い方

TwitPaint の描画機能や Twitter との連携や派生機能について述べる．

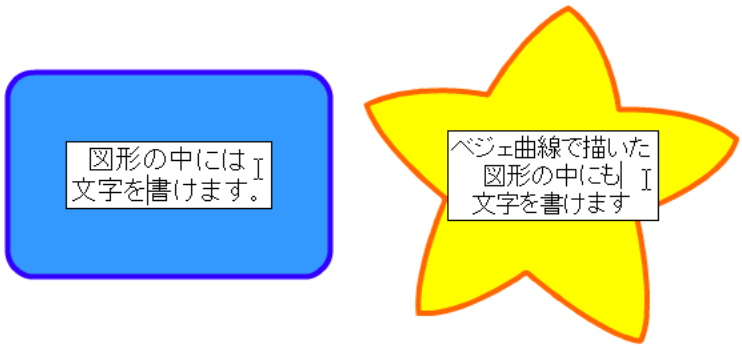

図 5.4: 図形内テキスト（Willustrator）

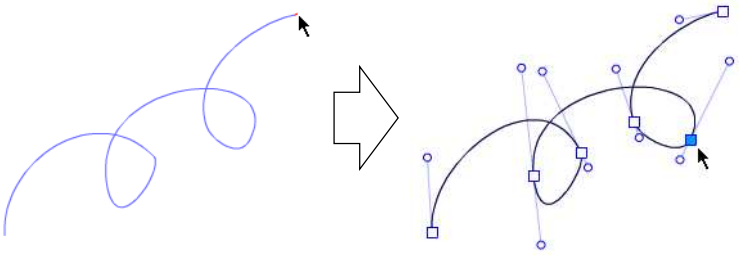

図 5.5: フリーハンドツール（Willustrator）

絵の描画と Twitter への投稿

TwitPaint には Twitter のアカウントでログインすることができ，ログインするとすぐに描き始められる．描画機能はシンプルで，ペイントブラシの太さと色，透明度を設定できるほか，アンドゥ機能を備えている（図 5.8）．

また，1 つの絵につき最大 140 ストロークだけ描けるようになっており，図 5.8 のようにキャンバスの右上には残りストローク数を表すカウンターが表示されている．140 ストロークという制限は，Twitter が 1 投稿につき 140 文字に制限することで短く気軽な投稿を促していることに倣っている．

絵を描き終わったあと，コメントを書いて投稿ボタンを押すと画像が保存され，さ

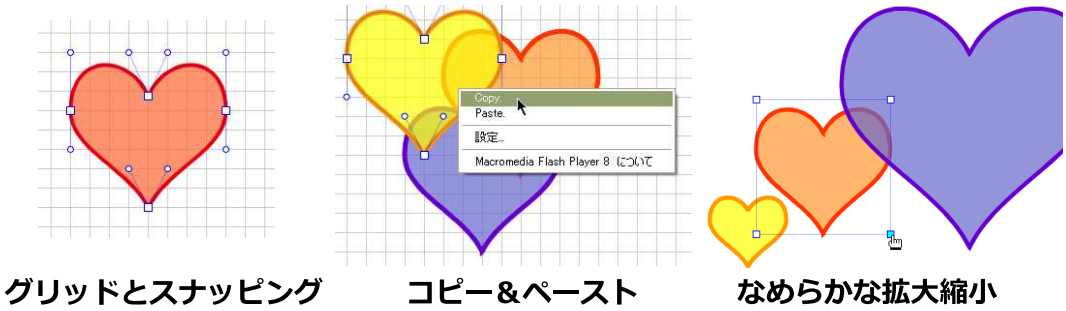

図 5.6: 基本的な図形編集機能（Willustrator）

図 5.7: 派生元および派生先の絵へのリンクと派生ボタン（Willustrator）

らに画像ページの URL とコメントが自動的に Twitter へ投稿される．

派生（Remix）機能

TwitPaint では，画像一覧画面などから他の人の描いた絵を選んで詳細表示するだけで，その画像の上に描きこむことができる（図 5.9）．そして他の絵に加筆・編集した後に投稿すると派生になる．派生元になった絵や，派生させてできた絵は図 5.9 の下部のように表示され，リンクを辿ってみることができる．

みんなの作品

「みんなの作品」というページでは TwitPaint のユーザが描いた絵を見ることができ，新着順の他に以下の 3 通りのランキングをそれぞれ月別・全期間で表示する（図 5.10）．

- 話題賞：コメントの多い順
- 元ネタ賞：その絵から派生した絵（子画像）が多い順
- 継続賞：派生の連鎖（世代）が長い順

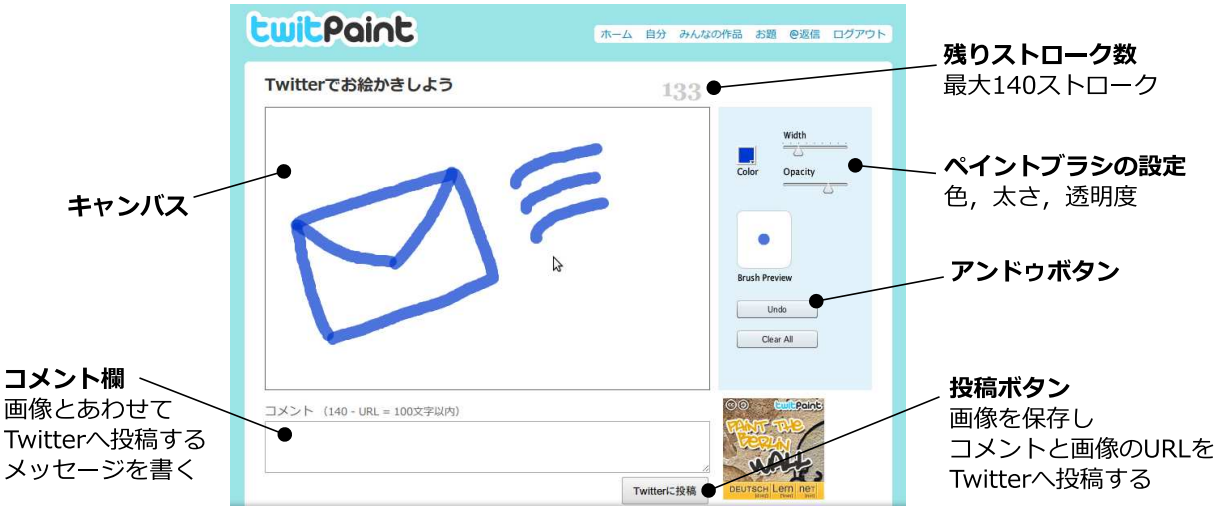

図 5.8: TwitPaint の画像編集およびコメント入力画面

お題

「お題」ページでは図 5.11 のように「あなたの大好物は？」「猫描いて！」といったお題を投稿することができ，他の TwitPaint ユーザがそのお題に従って絵を描いて投稿する．

5.3.2 実装

TwitPaint の Web アプリケーションは PHP で，ペイント部は Flash で実装した．ペイント部で描かれた絵（ビットマップ）は Flash 側で PNG 形式に圧縮後，サーバに送信・保存される．派生させて描く場合はその PNG 画像を読み込みビットマップとして編集する．

5.4 運用と結果

Willustrator は 2006 年 1 月から，TwitPaint は 2009 年 7 月からそれぞれ運用を開始した．2010 年 6 月時点での利用状況を表 5.1 に示す．また，図 5.12 のグラフは，各ユーザがそれぞれ何枚の画像を作成したかを示す．

表 5.1 の「派生ツリー」とは，ある初代画像（親画像を持たない画像）から派生した子画像と，さらにそこから派生した全ての子孫画像をまとめて 1 つのツリーとして呼び表したものである．1 回の派生ごとに世代が増え，親/子を 2 世代，親/子/孫を 3 世代という順に数える．また，表 5.1 の「派生ツリーの子孫画像数」および「派生ツリーの世代数」では 1 世代のみの画像，すなわち全く派生されなかった画像はツリーとして数えていない．

図 5.9: TwitPaint の派生機能

5.4.1 Willustrator の利用事例

Willustrator の特徴的な利用事例として「図解」「キャラクターと素材の合成」について述べる．

図解

Willustrator を用いて GUI(Graphical User Interface) のモックアップ作成や図解を作成した事例を紹介する．

図 5.13 では，ソフトウェアの GUI のモックアップ作成に使われた．この例では元となる画面から派生させることで 4 通りの画面を作成している．

表 5.1: Willustrator と TwitPaint の利用統計（2010 年 6 月時点）

	Willustrator	TwitPaint
登録ユーザ数（人）	768	15954
総画像数（枚）	1403	53999
派生により作られた画像数（枚）	326 (23%)	10820 (20%)
派生ツリーの子孫画像数の平均（枚）	2.54	2.19
派生ツリーの世代数の平均（世代）	2.12	2.86
最大世代数（世代）	5	84
子を持つ画像のうち 子の数の平均（枚）	1.28	1.14

図 5.10: みんなの作品（TwitPaint）

図 5.14 ではブログ記事に載せるため図を作成している．さらに他の人がこの図から派生させて描くことで，図の問題点を指摘するような，新しい意味を持つ図を作成した．

図 5.15 では四象限の元となる図から派生させることで，具体的な四象限の図を作成している．この例では一種のテンプレートとして派生機能を利用している．

キャラクター

Willustratow でキャラクターを描いた例が多く見られた．

図 5.16 では，ある人が描いた白黒のキャラクターから別の人が派生させて，そのキャラクターに色を塗っている．このようにキャラクターから派生させて色を塗る，また新しいパーツを付け加えるといった使い方が見られた．

図 5.17 では，素材となる 2 つの絵（犬と吹き出し文字）を組み合わせて新しい絵を作っている．さらにこの素材を描いたユーザは図 5.18 のように，一種の素材集として使えるようにキャラクターの画像をまとめて載せていた．

元となるキャラクター画像は直接 Willustrator で描かれた物よりも，Adobe Illustrator などで描かれた物をインポートしているものが多く見られた．

5.4.2 TwitPaint の利用事例

TwitPaint は利用目的ごとに派生の仕方にいくつかの傾向が見られた．

図 5.11: お題ページ（TwitPaint）

テンプレート

最も多くの子画像を持つ図5.19では，吹き出しの描かれた画像から派生させて，別のユーザが吹き出しの中に文字や絵などを書き足している．同様に吹き出しを書き込めるようなテンプレートとなる画像を作ることで，多くの人が書き足して遊ぶ例が見られた．

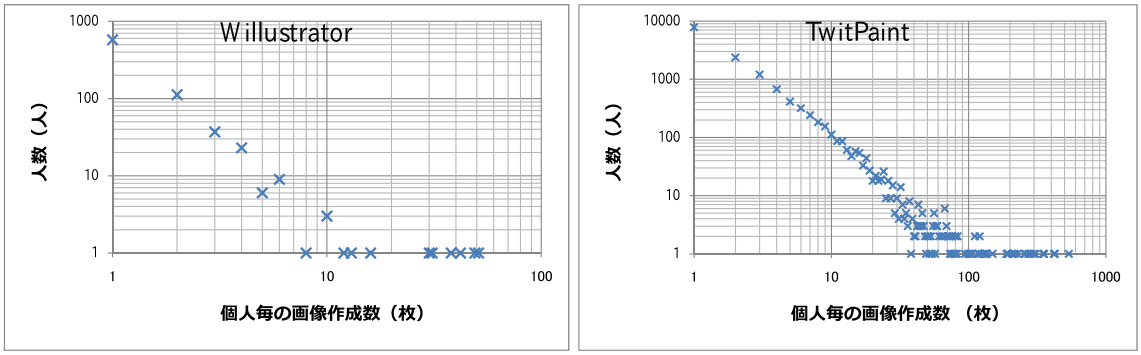

図 5.12: 個人毎の画像作成数とその人数の関係．各ユーザがそれぞれ何枚の画像を作成したかを表す．

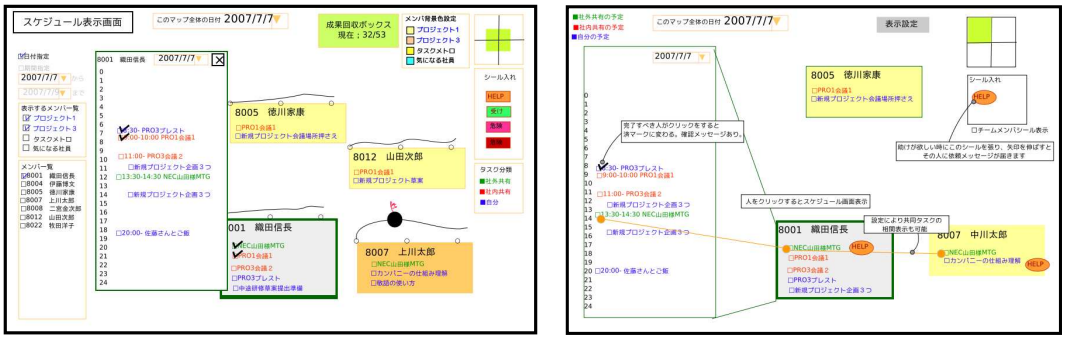

図 5.13: GUI のモックアップ作成（Willustrator）

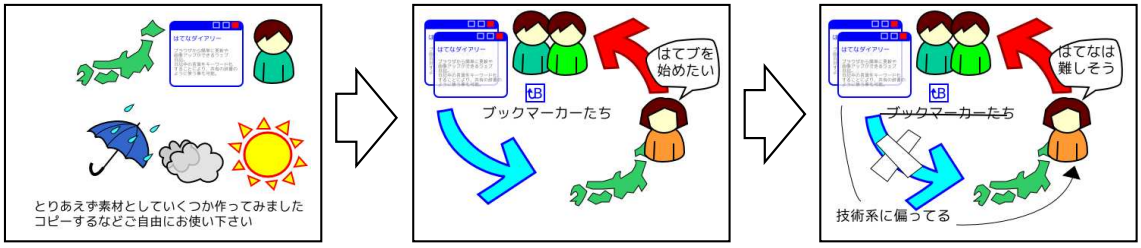

図 5.14: ブログの記事用の図解とその派生（Willustrator）

ゲーム

派生ツリー内の子孫画像数が 2 番目に多かった図 5.20 では，派生を利用して三目並べをしている．元となる「井」形の画像から別のユーザが派生させて〇や× を書き込むことでゲームが進行している．通常の三目並べのように 2 人のユーザが対戦するのではなく，途中で枝分かれして派生させることで一つのツリーの中で複数人が対戦しているのも特徴的である．

図 5.21 では絵を使ったしりとりをしている．最初のユーザが「リンゴ」の絵を描いたあと，別のユーザがそれから派生させて，しりとりとしてつながるように「ゴマ」「丸」というように次々と絵を足すことで派生が長く続いている．

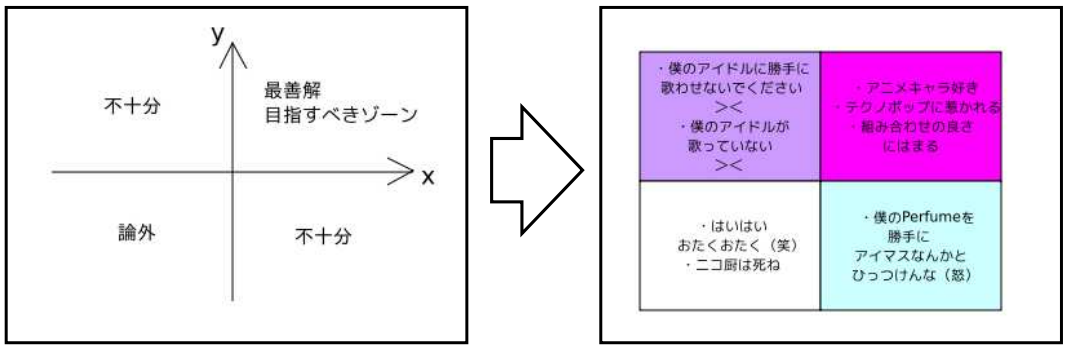

図 5.15: 四象限図とその派生（Willustrator）

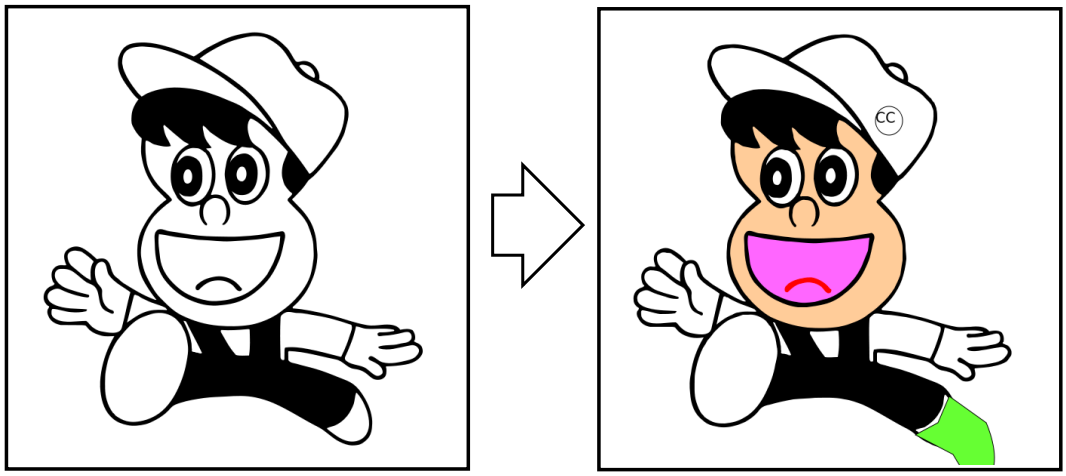

図 5.16: キャラクターに色を塗る（Willustrator）

力作絵画

時間をかけて書き込むことで一つの絵を仕上げていく，力作と言える絵を描くユーザが見られた．

派生ツリー内の子孫画像数が最も多く，最も世代数の多かった図 5.22 では，ある一人のユーザが画家（ゴヤ）の作品を模写していた．このような力作絵画は TwitPaint の 1 回の投稿につき 140 ストロークしか描けないという制約により，1 回では描ききれない．そこで 140 ストローク描く度に投稿して，またすぐに同じ人が派生させて書き足している．

一方，図 5.23 では有名なモナ・リザの絵を模写しているが，こちらは複数人で一つの絵を仕上げている．絵を仕上げようとする人もいれば，派生させた絵に落書きして遊ぶような人もおり，上記のゴヤの例に比べて枝分かれも多い．

お題

TwitPaint のお題機能を使った投稿では,「記憶だけでグリコのマークを描いてください」といった，記憶だけを頼りに描いて投稿させるものが特に人気であった．また「好きな歴史上の人物を描きましょう」といった，その人の好きな何かを描かせるものもお題や投稿数が多かった．上で述べた「しりとり」や「モナ・リザを描く」のように派生を利用してバトンタッチで描くものもお題機能を使って描かれている．

投稿されたお題の数は 113 件で，お題に答える形で投稿された絵の数は 2790 枚（全体の約 5%）であった．

5.5 議論

運用を通して得られたデータや事例，知見を元に Willustrator と TwitPaint を比較し，両者の相違や特徴，問題点を議論する．さらに Web 上のイラストツールの課題や

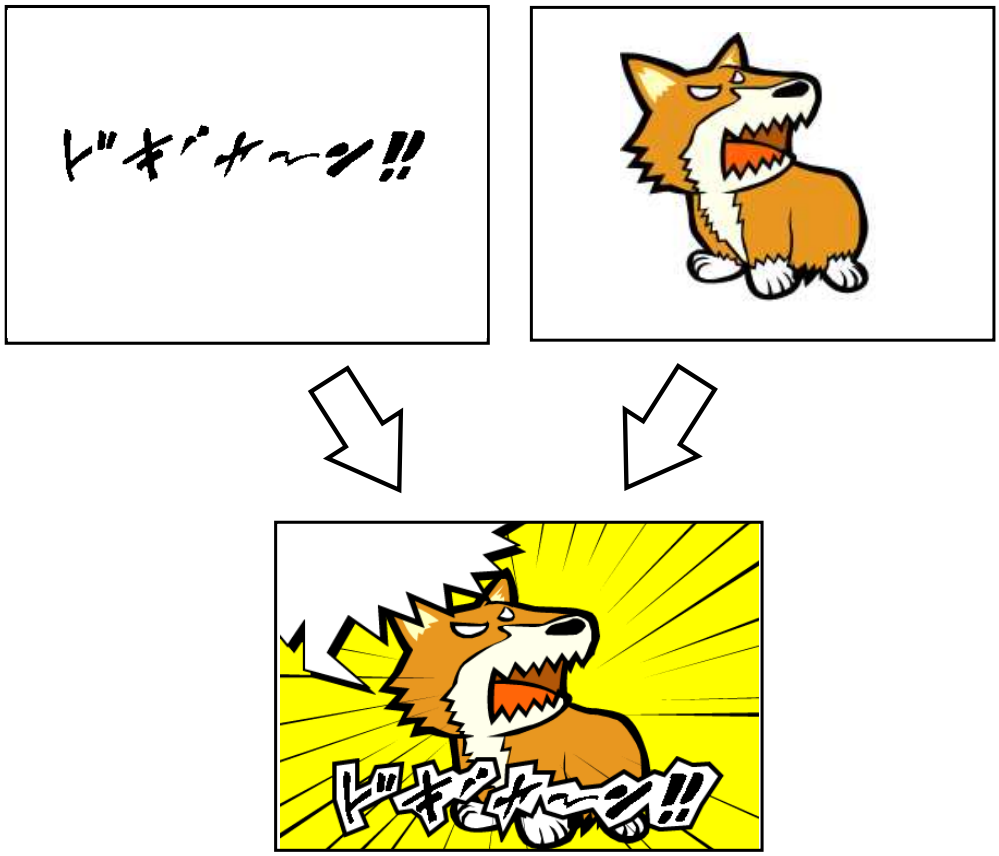

図 5.17: 図形の合成（Willustrator）

将来の方向性についても検討する．

5.5.1 機能の比較

表 5.2 に Willustrator と TwitPaint の機能や特徴を対比した．

ドロー系とペイント系

最も大きな違いは描画・保存方法の違いであり，Willustrator がベクター（ドロー系），TwitPaint がビットマップ（ペイント系）となっている．

ドローツールは一度描いた絵を「複製する」「拡大縮小する」「色や太さを変更する」ことを得意とするため，ペイントツールに比べて再編集しやすい．他の人が絵を再利用する場合も全く同様に再編集できるため，派生によって多くの人が絵に手を加えたあとでも，絵の一部の色や大きさを変更するといったことができる．一方，ペイントツールでは他の人が派生させて編集する場合，元の絵に上書きする形になるため，多くの人が絵を変更し複雑な絵になると，徐々に変更が難しくなる．

また，ドローツールは直線や幾何学的な線や図形を描くのに適しているため，図のような絵を描きやすい．そして，そのような図の元となるパーツを共有し，一部だけ再

図 5.18: キャラクター素材（Willustrator）

表 5.2: 描画機能や特徴の対比

	Willustrator	TwitPaint
描画・保存方式	ベクター（ドロー系）	ビットマップ（ペイント系）
絵の特徴	図のような幾何学的で整った絵	絵画のような手描き感のある絵
描画機能の数	複雑・多機能	シンプル・低機能
編集にかかる時間	比較的長時間	短時間（派生により長時間）
派生画像の編集方法	調整/複製/合成	描き足す

利用するといったこともできる．実際に「運用と結果」で述べたように，Willustrator では作図やキャラクター素材と合成を使った絵が描かれており，TwitPaint には無い絵を生んでいる．

編集にかかる時間

ペンの太さや色を選ぶだけですぐに絵を描けるシンプルなペイントツールに比べて，ドローツールは入力画面や操作方法が複雑になり，細かな値の調整をすることが多いため編集時間が比較的長くなる．TwitPaint は操作のシンプルさに加えて，1 投稿 140 ストローク制限により，1 つの絵を描くのにかかる時間が短い（平均 5 分 43 秒）．Twitter のようなリアルタイムに近いコミュニケーションでは，多少ラフであっても短時間で絵を描いて投稿できることが求められるため，短時間で描けるというペイントツールの特徴が Twitter を通じたコミュニケーションに適しており，TwitPaint の参加ユーザを増やした一因になったと考えられる．

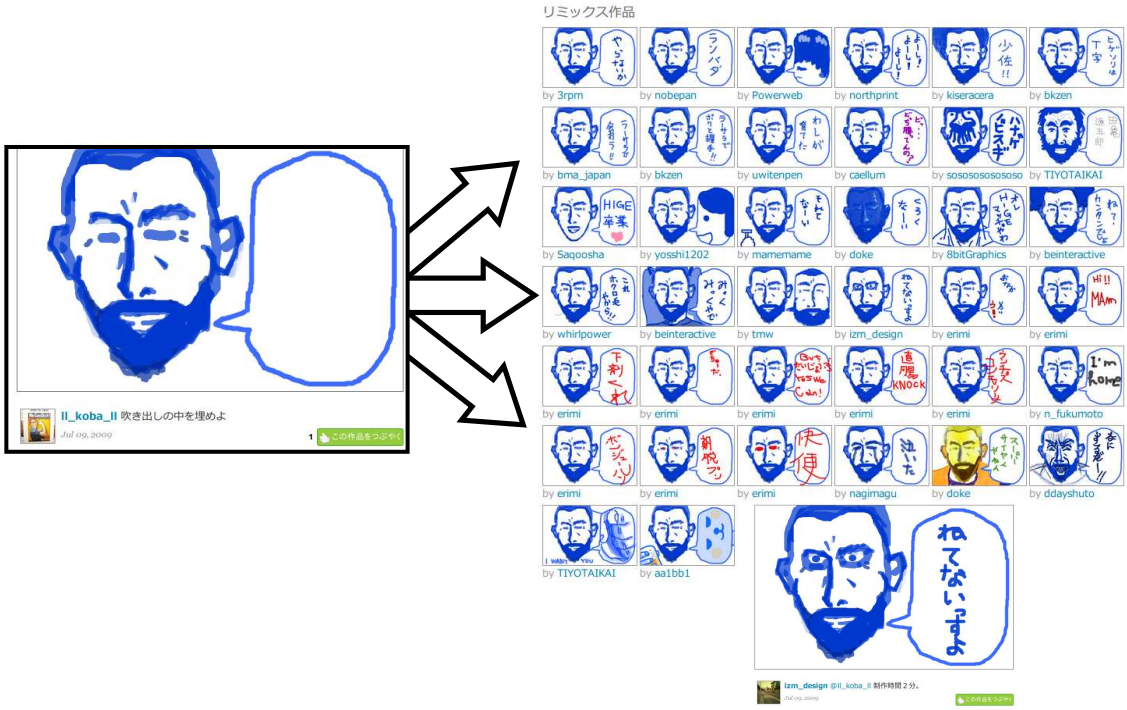

図 5.19: 吹き出し画像からの派生（TwitPaint）

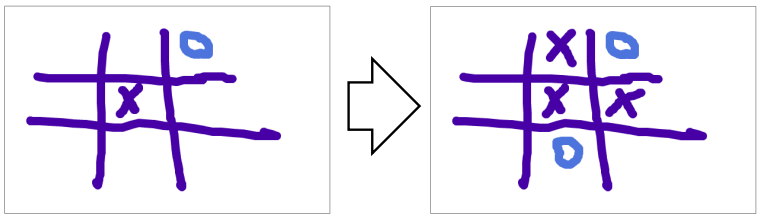

図 5.20: 三目並べ（TwitPaint）

5.5.2 絵の公開先とコミュニケーション

TwitPaint の他サイトからのアクセスのうち 69%が Twitter.com 経由[6] で，9%が検索エンジン経由，残りがその他のサイト経由であったことから，Twitter を用いたコミュニケーションが TwitPaint の大きな特徴であり，多くの参加者を集めた要因になったと考えられる．一方，Willustrator の開発や運営を中心的に行っていた時期（2006-2007 年頃）は，まだ Twitter はユーザ数も少なく現在ほど流行っていなかった．当時は Web 上でのコミュニケーションはブログが中心であり，Willustrator も描いた絵がブログに載せられることを想定していた．この Twitter とブログという絵の公開先の違いや，Twitter に自動投稿されるというコミュニケーションへの直結度，TwitPaint のランキング（みんなの作品），お題といった機能が，絵を通じたコミュニケーションの活性化や絵を描くモチベーション，そして参加者数の違いにつながったと考えられる．

[6]Twitter の Web クライアントからのアクセスを含めると 72%以上

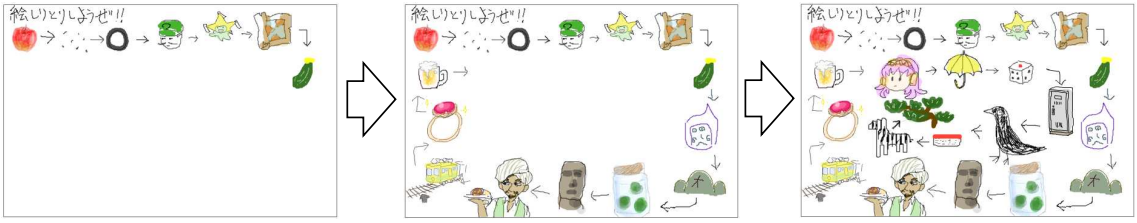

図 5.21: しりとり（TwitPaint）

図 5.22: 名画を 1 人で模写（TwitPaint）

バイラル効果

Twitter とブログでは，人づてに情報が伝わって広まるバイラル（口コミ）効果に差がある．Twitter は短い発言のしやすさから，これまで Web 上に情報を発信しなかったユーザからの投稿も多い．また，更新頻度がブログに比べて多いため情報の広まるスピードが速い．

TwitPaint に絵が投稿されると，自動的に Twitter にも URL が投稿され，それを見た Twitter でつながりのある知人などに伝わる．さらにその絵が面白ければ，コメントや ReTweet を通じて Twitter 上で話題になり情報が広がる．また，話題になって訪問者が増えることで TwitPaint の利用者がさらに増えるという循環になっている．

一方，Twitter では情報が広まるのは早いが，数日でタイムラインから流れてしまうため，話題が「熱しやすく冷めやすい」傾向がある．ブログでは情報の広まるスピードは遅いが，不特定多数に公開され，検索エンジンを通じて公開後しばらくしてから見られることも多いため，長期間見られるような絵に適していると言える．

モチベーション

Twitter のバイラル効果は絵を描く人にとってのモチベーションにつながると考えられる．特に絵がうまい人にとっては，描いた絵を誰かに見てもらえることや，それが話題になることは楽しみであるため，より人に見せやすく，話題になりやすい TwitPaint はそのような人に向いている．絵がうまい人が集まることで，よりバイラル効果も高まり，派生によって作られる絵も増えるという循環を生む．

また Twitter はチャットのようにリアルタイムに近い早さでコミュニケーションできるため，投稿したあとコメントなどの反応がすぐに返ってくるという楽しさがある．この素早いレスポンスによる楽しさが，TwitPaint で絵を描く人にとってのモチベーションにつながると考えられる．

図 5.23: 名画を複数人で模写（TwitPaint）

ランキングとお題を通じたコミュニケーション

TwitPaint では，ランキング（みんなの作品）やお題も，Twitter や TwitPaint 上でのコミュニケーションの活性化に貢献していた．

みんなの作品の「話題賞」「元ネタ賞」や「継続賞」では，優れた絵や面白い派生ツリーが上位に浮かび上がるようになっており，上位の絵やツリーは Twitter 上でも話題になりやすい．そしてランキングで上位に載ることや，話題になることは絵を描く人にとってのモチベーションにもつながる．このように派生データはランキング手法に用いることで，話題性のある絵や力作の抽出につながり，さらにはコミュニケーションの活性化・モチベーション向上につながっていると言える．

TwitPaint のお題も，直接絵を描かない人にもリクエストを通じて参加するきっかけを与えられる点や，何を描いていいか分からない人にヒントを与える点で，コミュニケーションの活性化につながっていた．

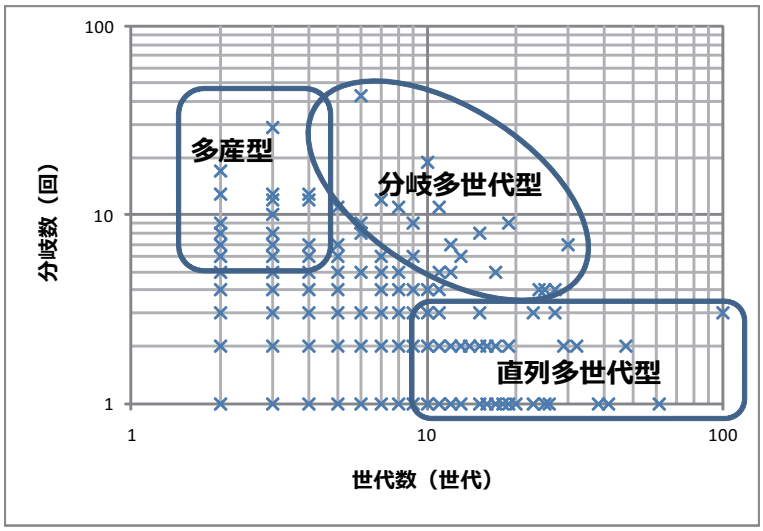

図 5.24: TwitPaint の派生ツリーの分類

5.5.3 派生ツリーの分類

Willustrator と TwitPaint は共に派生によって絵を描けることを特徴とし，この派生機能が活用される中で，様々な派生ツリーが形成された．派生ツリー内の分岐数[7] や世代数，ツリーの利用のされ方やユーザ層に基づいて，「直列多世代型」「分岐多世代型」「多産型」の大きく 3 つのタイプに分類した．図 5.24 は TwitPaint の計 5796 個のツリーの分布およびそれらの分類を示すグラフである．以下にそれぞれの分類の特徴を記すとともに，派生機能を活かした表現やコミュニケーション，熟達者と非熟達者の参加について議論する．

直列多世代型

「直列多世代型」の派生ツリーの定義は，1 つの親画像に 1 つの子画像という関係が何世代も連続的に続くような，世代数が多く分岐が少ないツリーとする．主に TwitPaint の利用事例で述べた「力作絵画」や，漫画のようなストーリーを続けていくような利用目的にこのタイプが多かった．直列多世代型のツリーでは，絵のうまい人が少人数（1~2 人）で一つの絵や話を作っていく点が特徴的である．

5796 個のツリーのうち 10 世代以上のツリーは 55 個であり，直列多世代型ツリーの数はそれほど多くない．これは，作成に関われる人が少数の絵がうまい人に限定される点と，一つの絵や話を仕上げるのに時間や労力がかかるためである．

直列多世代型ツリーは基本的に最初から最後まで少数の絵のうまい人のみが参加しており，他の人，特にあまり絵のうまくない人が途中でツリーに割り込んだり，ツリーを分岐するといったことはあまりされない．その理由として，絵のうまい人が力作を完成させようとしている途中で，その絵の上に落書きなどをして邪魔をしてしまわな

[7] ツリー内の各画像の子画像数の合計

いようにしているためと考えられる．また，絵にあまり自信の無い人が，絵のうまい人達と並んで描くことに心理的抵抗を感じることも原因として考えられる．

分岐多世代型

「分岐多世代型」は，世代数が多く，途中で時々枝分かれもするようなツリーとする．これは主に TwitPaint の三目並べやしりとりのような遊びをする例に多く見られた．直列多世代型と異なり，分岐多世代型のツリーでは絵のあまりうまくない人も参加している点や，絵そのものよりも遊びやコミュニケーションを楽しんでいるという点が特徴的である．

このような参加のしやすさに違いが生まれる理由としては，遊びで描くような絵の場合，誰でも描けるような単純な絵しか描かないといったことや，お互い絵がうまくない人同士であれば，絵のうまい下手をあまり気にせず描けるため心理的抵抗が低いといったことが考えられる．

ツリーが絵のうまい人から始まると途中からの参加や分岐が難しくなるため直列多世代型になりやすい傾向にあるが，ツリー生成初期の時点で絵のあまりうまくない人が参加したり，ラフな絵から始まると分岐多世代型になりやすい．つまりツリーが直列になるか分岐するかはツリー初期の時点で決まる傾向があると考えられる．

多産型

一つの親画像から多数の子画像が作られたツリーを「多産型」とする．Willustrator では図 5.16 のようにキャラクターから派生させて編集するものに多産型のツリーが見られた．TwitPaint でも同様に「吹き出しテンプレート」から派生させて絵を描くものが多かった．

Willustrator は，表 5.2 の最大世代数（5 世代）が少ないことからも分かるように，多世代型のツリーが少なく，多産型のツリーが多かった．これは，Willustrator は TwitPaint のような 140 ストローク制限が無く，また三目並べやしりとりのような遊びをするユーザが少なかったためであると考えられる．

多産型のツリーではツリーの初期にテンプレートや素材，塗り絵用の線画といった再利用されやすい絵が作られる．そのため何か少し絵に足すだけで簡単に新しい絵を作れる場合が多いため，分岐多世代型と同様に非熟達者が参加しやすい形と言える．また，同じ親/祖先の絵を各ユーザが様々な形に派生させて絵を描くことで，1 つの絵がどのように変化したのかという派生の流れを見て楽しむことができるというのも特徴である．

5.5.4 Web 上で使えるイラストツールの課題と展望

Willustrator と TwitPaint の運用から得られた知見を元に，今後の Web 上で使えるイラストツールの課題や将来の可能性について検討する．

非熟達者の継続的な利用

「5.1 はじめに」で述べたように，Willustrator と TwitPaint では非熟達者による絵の共有や編集を目標の一つとしていた．そして，この目標のために Web 上で簡単に共有できる仕組みや，一から絵を描かなくても派生・再編集によって絵を作れる仕組み，SNS を通じたコミュニケーションの中で気軽に絵を使えるといった仕組みを構築した．一方，図 5.12 のグラフから判断すると，1~2 枚書いておしまいというユーザも多く，継続的に多数の絵を描いているユーザは少数である．さらに，こうしたユーザを観察したところ，その大半が絵の熟達者であった．

サービスを運営する中で，非熟達者が継続的に利用しにくい原因・課題がまだいくつもあることが分かってきた．克服すべき主な課題として,「絵を一般に公開することへの心理的抵抗の解消」「絵を描く人のモチベーションを高める工夫の必要性」「絵の素材と更なる活用場面の提供」などが挙げられる．以下では，このような課題をふまえ，非熟達者を含む広く一般の人のためのツールにしてゆくための施策について検討する．

Twitter や SNS との連携によるコミュニケーション

TwitPaint では Twitter と連携することで，優れた絵の存在が素早く人づてに広まり，また絵を描く人のモチベーション向上につながった．Twitter を含むソーシャルネットワークシステム (SNS) は Web 上での主要なコミュニケーションツールになっているため，これらの外部 Web アプリケーションとの連携が今後のイラストツールにとって重要な課題となる．Twitter や SNS などとうまく連携することで，絵の作成がより直接的にコミュニケーションにつなげられるようになり，また Web 上でのコミュニケーションを絵の編集に反映できるようになると考えられる．

Willustrator と TwitPaint は共に描いた絵は基本的に全て Web 上に公開となっていた．しかし絵に自信がない人にとっては自分の描いた絵を公開することは心理的抵抗が大きく，公開範囲が広いほどその抵抗は大きい．また Twitter では公開範囲を細かく調節することは難しいが，他の SNS と連携することで手軽かつ柔軟に公開・共有範囲を限定するための工夫も必要である．

自分の描いた絵から派生させて誰かが絵を描いたり，描いた絵に対して誰かから反応がくるのは嬉しいものである．気軽にコミュニケーションをとりやすい知人だけに限定して公開・共有することにより，より反応が返ってきやすくするというモチベーション向上の仕組みも，絵の公開における心理的抵抗を解消する上で重要となる．

Willustrator や TwitPaint では，絵を描くために Willustrator や TwitPaint のサイトに移動する必要がある．もし，様々な SNS 上で直接絵を描くことができれば，より手軽に絵をコミュニケーションに活用できる．Web ブラウザの拡張機能を活用し，絵を描く機能を各サイトに埋め込むことで実現できると考えられる．

派生情報の分析とランキング手法

TwitPaint の「元ネタ賞」や「継続賞」では，派生情報を元に特徴的な絵をランキング表示しており，面白い絵の発見や，絵を描く人のモチベーション向上につながった．ここでは派生機能が重要な役割を果たしていたと言える．

「絵の派生」は Willustrator や TwitPaint の持つ個性的な特徴であり，今回のように多くの絵の派生情報を記録した事例はまだ珍しく，分析の余地は大きい．今後のイラストツールにおいても，派生機能を採用し，派生情報をより詳細に記録・分析することで，新しいランキング手法などへの応用が期待できる．

図解用素材の提供と共有

Willustrator では利用事例で挙げたように図解やモックアップ作成ツールとして利用されていた．図解のような用途では，より多くの人に利用機会があり，非熟達者でもパーツを組み合わせるだけで絵を描けるため，有力な発展の方向性と言える．

しかし一方で，Willustrator では図解に適した素材が少なく，必ずしも非熟達者にとって図解を作成しやすい環境ではなかった．図解用の基本的で高品質な素材をあらかじめ多数用意・提供し，さらにユーザによって描かれた素材を共有し充実させることで，非熟達者を含むより広いユーザにとって活用しやすいイラストツールになると考えられる．

また，これまでの議論で述べたような Twitter や SNS，ランキング，お題などを通じたコミュニケーションと，それらによるバイラル効果やモチベーションの向上によって，より効果的に素材が集まり，利用されることが期待できる．

5.6 関連研究

5.6.1 Web 上で使えるドローツール

Web 上で使えるドローツールは Willustrator 以降，様々な Web アプリケーションが開発されており，特に高機能なものとして Aviary 社の Raven[Aviary, 2011b] がある．ドローツールの中で作図に特化した Web アプリケーションも多く，Cacoo[Cacoo, 2011] や Gliffy[Gliffy, 2011]，Creately[Creately, 2011] などがある．このように Web 上で利用できるドローツール，作図ツールはすでに数多くあるが，派生を活かしたドローツールは現在のところ Willustrator のみである．

5.6.2 Web 上で使えるペイントツール

Web 上の掲示板やチャット上で使えるペイントツールは比較的古くからあり，しぃペインター [しぃペインター, 2011] のような Java Applet で実装されたものが使われていた．最近では Web ブラウザで動作する高機能なペイント･画像編集ツールも数多く登場しており，Picnic[Picnic, 2011] や Adobe 社の Photoshop Express[Adobe, 2011]，Aviary 社

の Phoenix[Aviary, 2011a]，SumoPaint[SumoPaint, 2011] などがある．また，Twitter に投稿できるペイントツールには，TwitDraw[TwitDraw, 2011] や drawTwit[drawTwit, 2011] などがあるが，他の人の描いた絵から派生させられるものは TwitPaint のみである．

5.6.3 CSCW による絵の作成

Computer Supported Cooperative Work(CSCW) の分野で，複数人で絵を描くための研究が古くから行われており，ネットワーク越しに複数人で同時に絵を描くソフトウェアとしては，Greenberg らによるペイントツールの GroupSketch[Greenberg, 1992a] やドローツールの GroupDraw[Greenberg, 1992b] がある．

また Web 上で利用できる非同期のシステムとしては Web コンテンツへのアノテーションシステムの Annotea[Kahan, 2001] がある．Annotea そのものは RDF を用いて Web 上に様々なアノテーションを付けるためのシステムであるが，Amaya[8] というブラウザを使うことで Web 上に SVG を書き込んで協同で編集することができる．

Web 上で同期，非同期を両立したシステムに永田らの Nota[Nagata, 2007a][9] がある．Nota ではドラッグ＆ドロップによる図形の配置や，フリーハンドで線を引けるといった直接編集操作により簡単に Web ページを作成でき，さらに複数のユーザで同時編集することで Web 上での情報共有やコラボレーションを支援することができる．

Willustrator や TwitPaint のような派生型のシステムも，複数人がコンテンツ作成に関わるという点では一種の CSCW であるが，上記のような複数人で共有された 1 つのコンテンツを作成するシステムとは利用の仕方が異なる．派生型のシステムでは 1 つの絵は基本的に 1 人で編集するが，派生させることでより手軽に絵を描けるようになり，また派生させることによってユーザ間や絵の間で緩いつながりが生まれる．

5.6.4 コンテンツの派生

アーティストの安斎らは連画 [Anzai, 2000][10] プロジェクトにて，他の人の描いた絵から連想することで，参加者が次々と紙の上に絵を描き，線でつないでいくというワークショップを行っている．さらに，このワークショップで作られた絵をコンピュータに取り込むことで，派生の様子を辿ってみられるビューアを開発した．Willustrator や TwitPaint の絵がツリーのように次々と派生していくイメージはこの連画に近い．

Processing[11] を拡張したプログラミング環境の Share[Assogba, 2010] では，プログラミングをするアーティストがソースコードを他のアーティストと共有でき，簡単に引用することができる．さらに全てのコードの引用を記録し，グラフ状に可視化することで，多くの人に参考にされた有用なコードを発見することができる．

Web ブラウザ上で Flash を作成・共有することのできる Wonderfl[Wonderfl, 2011] では，プログラムのソースコードを共有し，さらにそれを他の人が派生 (Fork) させて

[8] http://www.w3.org/Amaya/
[9] http://nota.jp/
[10] http://www.renga.com/
[11] http://processing.org/

新しいプログラムを書くことができる．Flash を使った作品を Web 上で編集・共有し，派生させながら作品を発展させるという利用スタイルは Willustrator や TwitPaint によく似ている．

5.7 まとめ

本章では，Web 環境に溶け込むインタフェースとして，Web 環境のブログや SNS に PC 環境のイラストツールを溶け込ませる「Willustrator」「TwitPaint」という 2 つのシステムを提案，構築した．さらに両ツールを長期運用し，得られたデータや事例，知見などを元に相違や特徴，問題点の分析を行った．

ドローツールとペイントツールという描画方法の違いや，ブログと Twitter という絵の公開先の違いから，ユーザ数や画像の枚数，描かれる絵に大きな差が生じた．TwitPaint は Twitter のバイラル効果によって素早く情報を広め，さらに絵を描く人のモチベーションを高めたことが利用者の増加につながったと考えられる．両ツールに共通する特徴である派生に関するデータや事例を分析することで，派生ツリーは大きく「直列多世代型」「分岐多世代型」「多産型」の 3 つに分類することができ，派生ツリーの種類ごとに利用目的やユーザ数，ユーザ層が異なることが分かった．そしてこれらの知見を元に，今後の Web 上で使えるイラストツールの課題や将来の可能性について検討した．Twitter や SNS との連携によって絵の作成をより直接的にコミュニケーションにつなげることや，派生情報のより詳細な分析によって新しいランキング手法や可視化手法を実現すること，図解用素材の提供と共有によって非熟達者にとって活用しやすいイラストツールにすることなどが今後の課題であり，我々の目指す方向性である．

第 6 章

ユビキタス環境に溶け込む

概要

本章では，ユビキタス環境に溶け込むインタフェースとして，生活空間に PC 環境のビデオチャットを溶け込ませる「なめらカーテン」を提案する．現在，ビデオチャットには主に Skype や iChat，Polycom などのアプリケーションが用いられている．これらのアプリケーションは，ビデオ会議やテレビ電話のようにディスプレイの側に座って集中的に会話するときは便利であるが，生活空間の中で気軽に使うには問題も多い．例えば，一瞬だけ相手と話したい場合でも，PC のすぐ側に近づいてマウス操作をする必要がある．また，通信をしていない状態の部屋の様子はお互いに伝わりにくい．しかし，常時ビデオチャットをしようとすると，生活空間におけるプライバシーが問題になる．そこで，生活空間にビデオチャットを溶け込ませることで，プライバシーを保護しながら，いつでも手軽に遠隔地の相手と会話できるシステム「なめらカーテンを」提案，構築する．なめらカーテンでは，カーテンのメタファを採り入れたカーテン型デバイスによって，直感的にプライバシーレベルを制御し，コミュニケーション形態の柔軟な移行を可能にする．さらに，なめらカーテンを用いた 14ヶ月以上の期間実証実験により有効性を検証する．

6.1 はじめに

インターネットが普及したことで，PC を使ったビデオチャットも広く利用されるようになった．ビデオチャット用のアプリケーションとしては，Skype や iChat，Polycom などが用いられている．

これらのアプリケーションは，ビデオ会議やテレビ電話のようにディスプレイの側に座って集中的に会話するときは便利であるが，生活空間の中で気軽に使うには問題も多い．例えば，一瞬だけ相手と話したい場合でも，PC のすぐ側に近づいてマウス操作をする必要がある．また，従来のビデオチャット用アプリケーションでは基本的にコンピュータ（ディスプレイ）の前にいる時しか通信をしないため，通信をしていない状態の部屋の様子はお互いに伝わりにくい．しかし，常時ビデオチャットをしようとすると，生活空間におけるプライバシーが問題になる．

本研究では，生活空間にビデオチャットを溶け込ませることで，これらの問題を解決する「なめらカーテン」を提案，構築し，有効性を検証する．

6.2 なめらカーテン

なめらカーテンは，カーテンのメタファを用いることで，直感的な操作でプライバシーを柔軟に制御し，生活空間での常時ビデオチャットを可能にするシステムである．

なめらカーテンでは，図 6.1,6.2 のような「なめらカーテン端末」を 2 箇所に設置し，両者間を常時ビデオチャットでつないだ状態にする．そして端末のディスプレイの前についたカーテンの開け閉めに応じてプライバシーレベルを調節し，コミュニケーション形態を変える．

以下，なめらカーテンの特徴，カーテンメタファ，デバイス構成，そして利用方法について述べる．

6.2.1 特徴

なめらカーテンでは，ビデオチャットを生活空間に溶け込ませるため，カーテンのメタファを採り入れたカーテン型デバイスによって，直感的にプライバシーレベルを制御し，コミュニケーション形態の柔軟な移行を可能にする．

なめらカーテンの特徴は，以下の 4 つにまとめられる．

- 日常空間での常時連続利用
- 柔軟なプライバシー制御
- シンプルで分かりやすい操作
- 生活空間に馴染むデザイン

次にカーテンメタファと，コミュニケーションシステムへの適用方法について詳しく述べる．

図 6.1: なめらカーテンの利用風景

カーテンメタファ

カーテンの主な機能や特徴としては「アンビエント性」「プライバシー保護」「使い方の分かりやすさ」「家への馴染みの良さ」などがある．

日常生活において，カーテン（特にレースのカーテン）は家庭のプライバシーを守る役割を持っている．例えば，人々は外部の視線が気になるとき，カーテンを閉めて家の中の様子を隠す．カーテンを閉めると，家の中の詳細な様子は外部から見られることはなくなるが，時間によって変化する明るさや，窓の前を通り過ぎていく人の影などのアンビエントな情報は感じることができる．すなわち，カーテンを閉じることでプライバシーを保持しつつ，外部からの情報を完全には遮断せず，穏やかに情報を共有することができる．

そして，カーテンは多くの家で利用されており，布地の柔らかな質感もあって，日常空間に馴染みやすい．使い方も「開け閉めするだけ」とシンプルで分かりやすく，大人から子供まで多くの人が日々利用して，その機能や使い方をよく知っているのも特徴である．

コミュニケーションシステムへの適用

本研究ではまず，カーテンのプライバシー面での特性を利用し，カーテンの開いた状態と閉じた状態をそれぞれ「ビデオチャット型の直接的なコミュニケーション」「アンビエントなコミュニケーション」に対応づけた．ユーザはカーテンの開閉量を変えることで，映像や音といったコミュニケーションチャンネルの情報量を調節する．

また，カーテンをある程度閉じて相手から覗かれにくくした場合，自分側のなめら

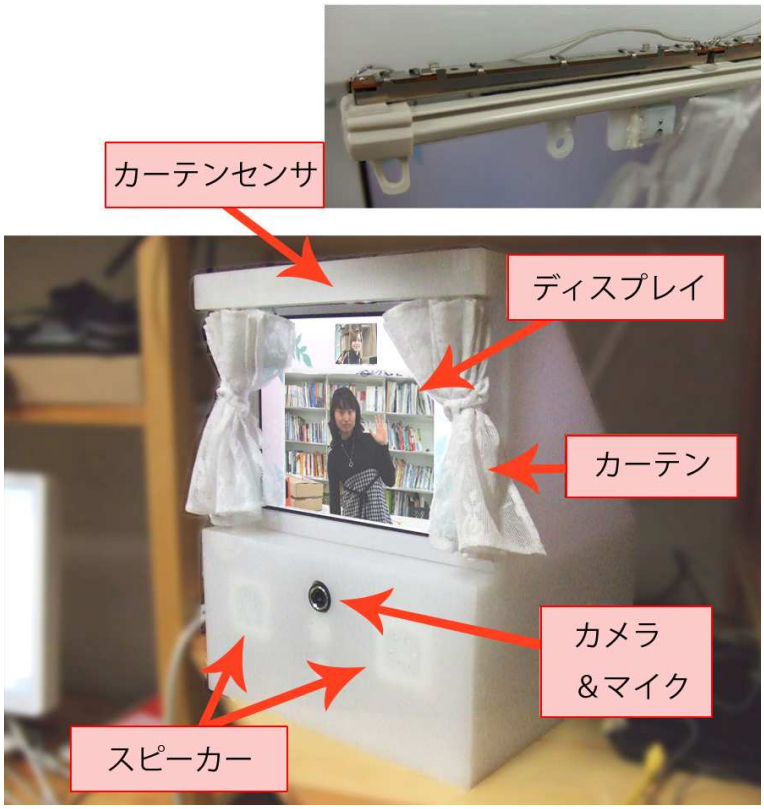

図 6.2: なめらカーテンの外観

カーテン端末のディスプレイもカーテンに隠れ，相手の様子を覗きにくくなる．こうすることで相手から一方的に覗き見られるという状態を和らげ，一定の公平感を与える．ただし，あくまで一方的な覗き見を「緩く」防ぐためのものであり，一瞬だけ相手の様子を見たい時などはカーテンをめくって見るといった使い方も可能である．

さらに，カーテンの開閉状態がコミュニケーションの意思を示すことにもつながる．相手がカーテンを開いている時はビデオチャットが可能な状態であり，反対に閉じている場合はアンビエントなコミュニケーションを望んでいるというように，お互いの意思や状況を推測することもできる．

6.2.2 機能

次に，なめらカーテンの具体的な機能として「ぼかしフィルタと音量調整」「チャイム機能」について紹介する．

ぼかしフィルタと音量調整

寝起きや着替え中，化粧中など，他人に見せたくない映像を隠す方法として，前章で述べたGreenbergらの研究[Neustaedter, 2003][Boyle, 2000]と同様に，映像にぼかしフィルタをかけるようにした．カーテンを完全に開いた状態（図6.3上）では映像と音声をそのまま相手側の端末で表示・再生する．反対に完全に閉じた状態（図6.3下）ではほとんど様子が分からない程度にぼかした映像を表示し，音声は遮断する．そしてカーテンを開き具合によって，映像のぼかしや音声の大きさが連続的に変化する．

なめらカーテン端末の画面は図6.4のようになっており，上部に自分の姿が表示され，下部に相手の姿が大きく表示される．カーテンを閉めると自分の姿が映った映像にぼかしがかかり（図6.3左），相手からどのように見えているか，すなわちどの程度ぼかしがかかっているかを確認することができる．

音声に関して，カーテンは両方向の音を遮断する．すなわち，カーテンを閉めると自分の声が相手に届かなくなり，同時に相手の声も聞こえなくなる．逆に相手がカーテンを閉めると相手の声が聞こえなくなり，同時に自分の声も相手に届かなくなる．

チャイム機能

前述のように，カーテンが閉じられている場合は相互に音声が伝わらなくなるため，なめらカーテン越しに呼びかけて話をしたい場合に明示的なコミュニケーションの開始が困難になる．このような場合に，家の玄関チャイムのように音を鳴らすことで「コミュニケーションを取りたい」という合図を送り，カーテンを開けてもらう「チャイム機能」を用意した．相手を呼びたい時は「カーテンを素早く[1] 2回以上開閉する」という操作をすると，相手側の端末でチャイム音を鳴らすことができる．

なお，チャイム音を鳴らす操作方法としては「物理的なスイッチを付ける」「カーテンの前に一定時間立つ」といった方法も検討した.「カーテンを素早く2回開閉する」という方法を採用した理由としては，急いでコミュニケーションを取りたいという意思をカーテンを用いたジェスチャで表現できることや，なめらカーテンの通常の操作とほぼ重複せず，誤認識も少ないこと，新たに入力デバイスを増やすことなくカーテンだけで全ての操作ができることなどが挙げられる．

6.2.3 利用の流れ

互いに遠隔地にいるユーザAとユーザBの2人がシステムを使用すると仮定し，基本的な利用方法を説明する．

まず双方のカーテンが開かれていると，Aの端末にはB側の映像，Bの端末にはA側の映像が映る．Aが端末の前に立って話しかけると，B側の端末にはAの姿が表示され，Aの声が再生される．両者が話しているときは，それぞれ図6.4のような画面になり，画面上部に小さく自分の映像が表示され，下部に大きく相手の映像が表示される．

[1] 3秒以内に

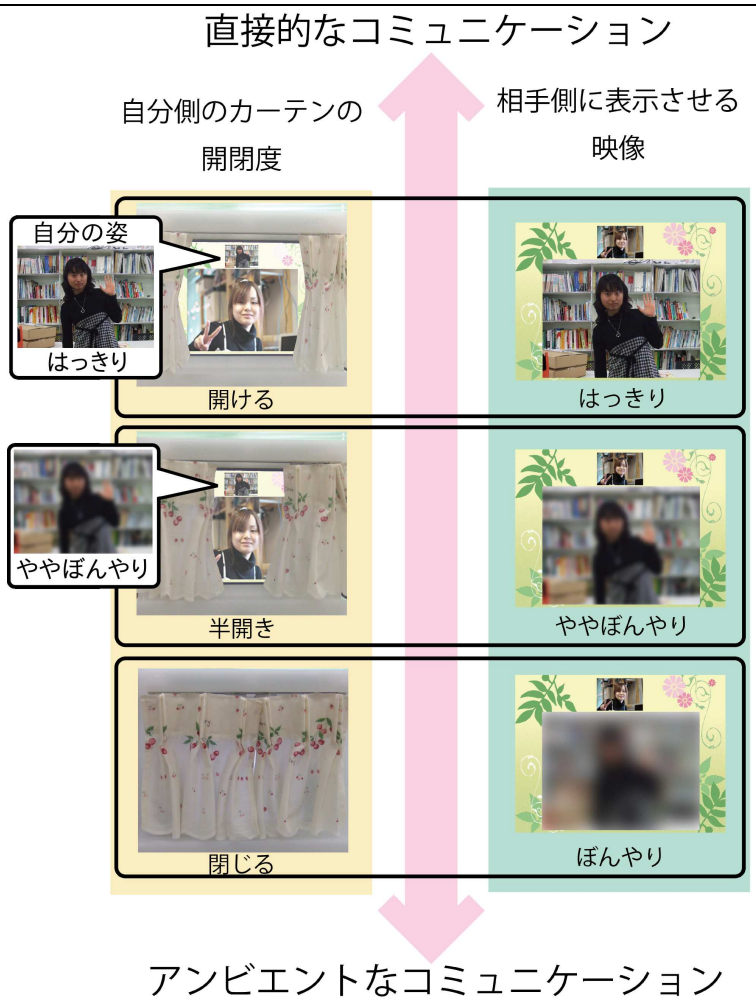

図 6.3: カーテンの開閉に応じたコミュニケーション形態の変化

ここで A が着替えのためにカーテンを閉めると，B 側端末の映像にはぼかしフィルタがかかって A の姿が見えにくくなり，A の話す声の音量が小さくなる．また，同時に A 側のディスプレイでも B の姿はカーテンに隠れて見えにくくなる．

そして A が完全にカーテンを閉めると，B 側の画面には強くぼかしがかかって A の姿が見えなくなり，A の声も聞こえなくなる．また A 側のディスプレイも完全にカーテンに隠れるため B の姿はほとんど見えなくなり，B の音声も聞こえなくなる．そのため A が一方的に B の様子を覗き見ることはできない．

その後しばらくして，B が A に話しかけたいと思ったとき，A の部屋の明かりは点いているようなのでなめらカーテンで聞こうとする．しかし A はカーテンを閉じているため声で呼び出すことはできない．そこで A を呼ぶためカーテンを 2 回素早く開閉すると，A 側の端末でチャイム音が鳴る．それに気づいた A がカーテンを開け，B は A と会話をした．

図 6.4: なめらカーテン端末の画面例.

6.2.4 実装

本章では，なめらカーテンの実装について，ハードウェア／ソフトウェアの観点から述べる．

ハードウェア

なめらカーテンのハードウェアは，カーテンとその開閉状況を取得するカーテンセンサ，液晶ディスプレイ，Web カメラ，マイク，スピーカー，およびこれらのデバイスを制御する小型パソコンから構成される．

カーテンセンサは図 6.5 のように，市販のカーテンレールにスライダ型可変抵抗（以下，スライダ）を組み込んだものである．スライダは，Phidget InterfaceKit[2] を介して小型パソコンに接続してされ，カーテンの開閉度を検出できるようになっている．

カメラとマイクは，多様な利用場面（1 対 1，1 対多，多対多）に対応するため，広角の Web カメラ（Logicool Qcam Pro for Notebooks) と無指向性のマイク（Panasonic RP-VC151）を使用した．

これらのデバイスを，アクリル板製のケースに格納した．設置や運用がしやすいようにケースの内部は上下 2 段に仕切られており，上段にはディスプレイと制御基板，Phidget InterfaceKit を収納し，下段にはカメラ／マイク／スピーカーと小型パソコンを収納する．

[2]http://www.phidgets.com/

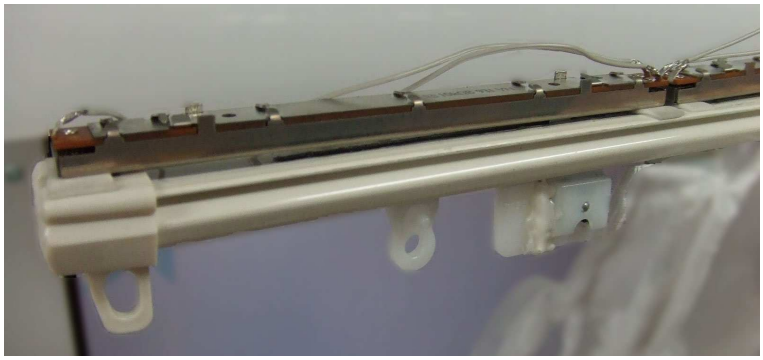

図 6.5: カーテンセンサ: カーテンレールにスライダセンサを組み込んでいる

ソフトウェア

ソフトウェアは，画面表示／デバイス制御などを行う端末側のプログラムと，両端末の間で映像や音声，カーテン開閉度を中継するサーバから構成される．端末側のプログラムは Adobe Flex[3] ，サーバは Flash Media Server[4] を用いて開発した．

端末側ではカメラ・マイクおよびカーテンセンサからそれぞれ映像・音声・カーテンの開閉値（以下，カーテンデータ）を取得し，サーバを介して他方の端末に送信する．相手の端末に直接データを送らず，サーバを中継したのは，端末の設置のしやすさと実装のしやすさを考慮したためである．端末は基本的に遠隔地に置くため，それぞれ異なる NAT（Network Address Translation）環境下に端末を設置することが多く，そのままでは端末間で直接通信することができない．そこで，それぞれの端末から NAT 外のサーバに接続することで，ルータなどを設定する手間を省き，家などでの設置をしやすくした．また，Flash Media Server を用いることで，映像や音声のストリーミングを使ったプログラムを比較的容易に実装できることも理由の一つである．

そして，サーバを経由して受信したカーテンデータに応じて，映像のぼかし具合や音声の音量を調整して，ディスプレイ／スピーカーに出力する（図 6.6）．

6.3 実証実験

なめらカーテンの効果を調査するため，実証実験を行った．遠隔地の親しいグループ間でなめらカーテンを日常的に利用してもらい，その様子を記録，観察することで提案方法の有効性と課題を検証した．

[3] http://www.adobe.com/jp/products/flex/
[4] http://www.adobe.com/jp/products/flashmediaserver/

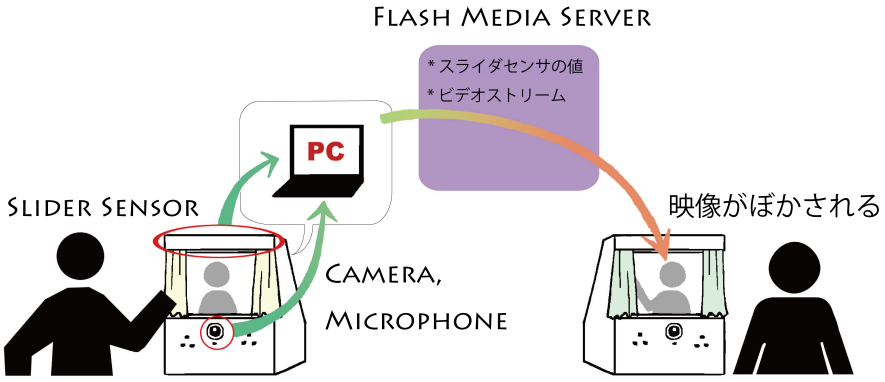

図 6.6: システム構成: サーバ経由で映像と音およびスライダセンサの値が送られる

図 6.7: 実証実験の設置風景．左: S 研究室，右: T 研究室

6.3.1 実験内容

大学内の離れた 2 つの部屋（以下，S 研究室と T 研究室）になめらカーテン端末をひとつずつ設置した（図 6.7）．両部屋のレイアウトと端末の置き場所は図 6.8 のようになっている．想定利用環境である生活空間に近づけるため，S 研究室ではなめらカーテンをソファのある部屋中央の休憩スペースに向け，T 研究室ではソファやテーブルのすぐ側に設置した．

S 研究室と T 研究室では，同じ研究グループに所属する学生と教員が常駐しており，S 研究室には基本的に学生のみが，T 研究室には教員と学生の双方が常駐している．両研究室は約 60m 離れた別々のビルにあり，歩いて約 5 分程度かかる．こうしたことから，従来は電話や Skype などがコミュニケーション手段として頻繁に使用されていた．

実験期間は約 14ヶ月間[5] で，合計 15 人以上の学生と教員が使用した．

実験期間の途中，一時的に T 研究室に設置した端末のカーテン丈を図 6.7 右上のよ

[5]本論文執筆時までの期間であり，現在も継続的に利用している．

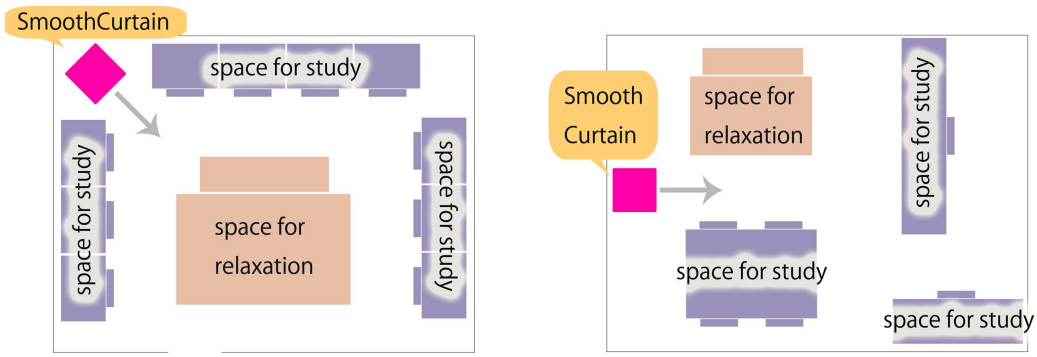

図 6.8: 実証実験における各部屋のレイアウト．左: S 研究室，右: T 研究室

うに Web カメラが隠れる程度の長さに変更し，2 週間程度運用した．これは，なめらカーテン端末ではカーテンを閉じた際にもカメラが物理的に見えているため，プライバシー面での不安を与えるのではないかと考えたためである．

6.3.2 結果

なめらカーテンの利用状況のデータと様々な利用事例を記す．

利用状況のデータ

実験期間のうち 2010 年 12 月に約 1ヶ月ほど[6]，カーテンの操作を記録した．運用当初はシステムの改良を重視していたため利用状況を記録しておらず，大学の休暇期間中のデータなどを省略したため，利用状況を記録・分析した期間は運用期間に比べて短くなっている．

それぞれの部屋のカーテンの開閉状況は図 6.9，6.10 のようになっている．T 研究室ではカーテンを閉じていることが多く，中間状態が少なかったのに対し，S 研究室では中間状態となっていることが比較的多かった．この利用状況の違いについては，T 研究室側にいる教員が集中して作業したい時にカーテンを閉じるケースが多かったことや，S 研究室側にいる学生の話し声や笑い声などが漏れ過ぎないように，学生自らカーテンをやや閉めた状態にしていたことが関係している．

1ヶ月のカーテンの開け閉めの遷移を示した図 6.11 からも分かるように，S 研究室に比べ，T 研究室ではカーテンを開閉する頻度が高かった．これは，T 研究室の教員側から S 研究室の学生に対して話しかける（カーテンを開く・チャイム機能を使用する）ケースが多かったことや，上述したように T 研究室側の教員が集中して作業する際に閉める機会が多かったことが関係している．

[6]運用当初はシステムの改良を重視していたため利用状況を記録しておらず，大学の休暇期間中のデータなどを省略したため，利用状況を記録・分析した期間は運用期間に比べて短くなっている．

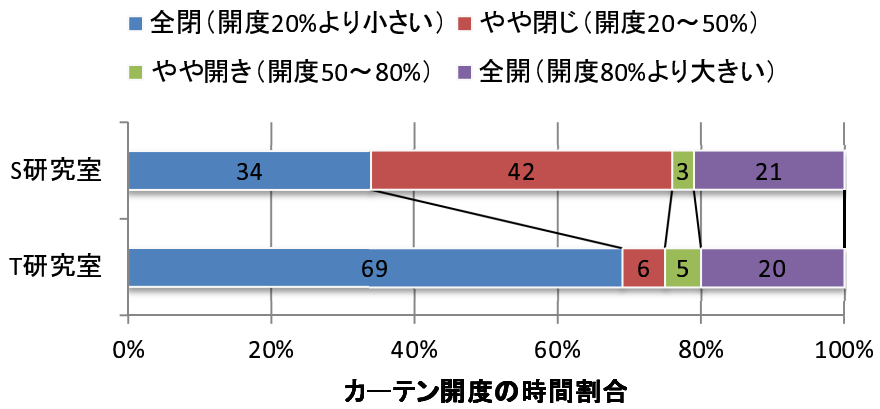

図 6.9: 各部屋のカーテン開度の時間割合

直接的なコミュニケーションの事例

両端末のカーテンが開いた状態では,「特定の人の在室状況を聞く」「探し物について何か知らないか聞く」といったちょっとした質問をするという使い方が多く見られた．利用者へのインタビューでも，両方のカーテンが完全に開いた状態の時は「手軽に相手に話しかけることができる」「違和感なく使用することができた」という声が多く得られた．

やや特殊な使い方としては，別の部屋にいる学生の誕生日祝いをするという事例も見られた．

アンビエントなコミュニケーションの事例

なめらカーテンを半開きの状態で使う場合，カーテン越しに見える画面の明るさや，少しだけ聞こえてくる音を頼りにしたアンビエントなコミュニケーションが見られた．

例えば，カーテン越しに部屋の明かりが点いているかどうかを見て（図 6.12），教員や学生が部屋にいるかどうかを知り，相手の部屋に移動するといった場面が度々見られた．また，両研究室同士でのイベント（ミーティングや飲み会）の時間が近づいた時，S 研究室の明かりが消えたことで人が移動したことを知り，T 研究室の人も急いで移動を始めるといったことがあった．

一方，音を頼りにした事例としては，S 研究室に一人でいた学生が，T 研究室での賑やかな声を聞いてカーテンを開けたところ，他の学生が T 研究室に集まっていることを知り，その学生も T 研究室に足を運んでみるといったことがあった．

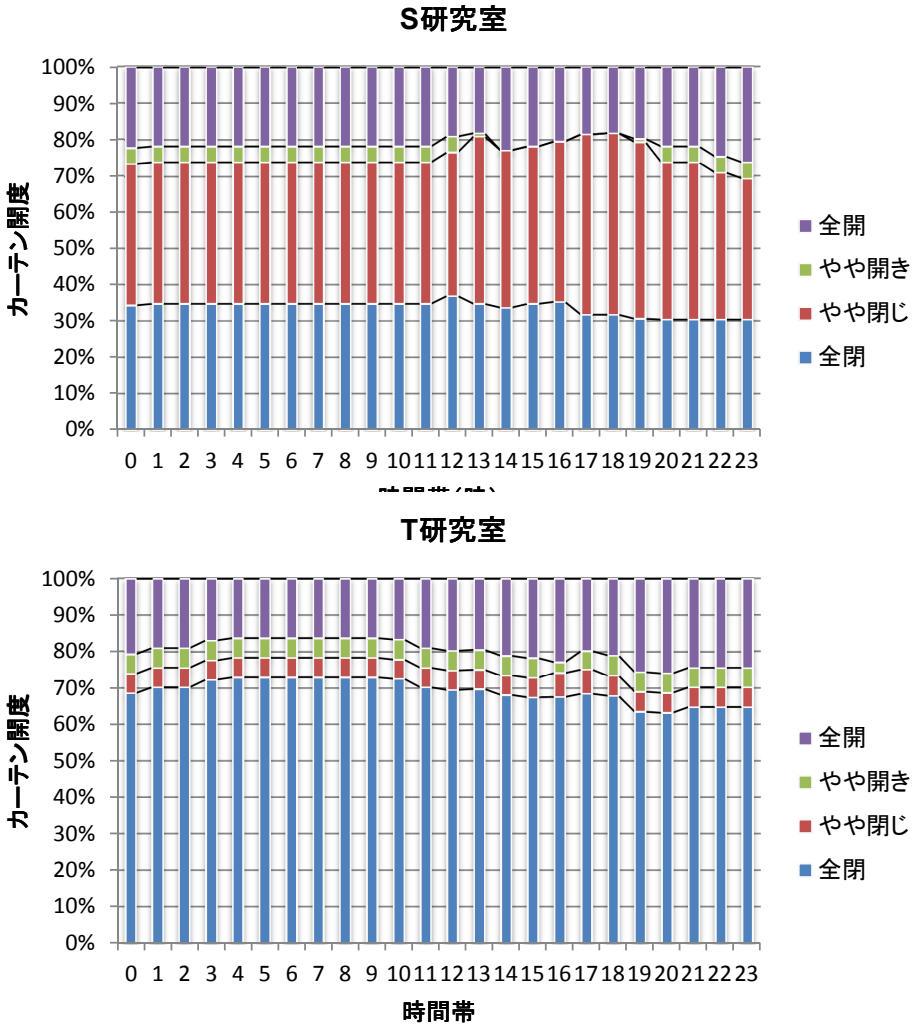

図 6.10: 時間帯別のカーテン開度

プライバシー制御の事例

プライバシー保護のためカーテンを操作した事例としては，部屋の中のソファで寝ようとした時にその姿を見られないようにカーテンを閉める（図 6.13）といったことがあった．また，部屋の中でプライベートな会話や電話をする時，会話の内容を聞かれないようにカーテンを閉めた，という利用者もいた．

実験内容で述べたように，カーテンを閉じた場合でもカメラが見えていることが，プライバシー面での不安を与えるのではないかと考え，一時的にカーテン丈を伸ばしてカメラが隠れるようにした．その後，数人の利用者に意見を聞いたところ，以下のようなコメントが得られた．

- カーテンの丈が変わっていたことに気がつかなかった．

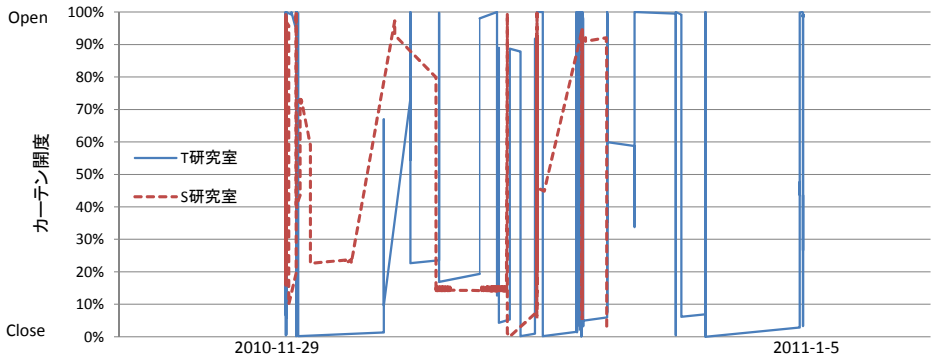

図 6.11: 各部屋のカーテンの開け閉めの遷移

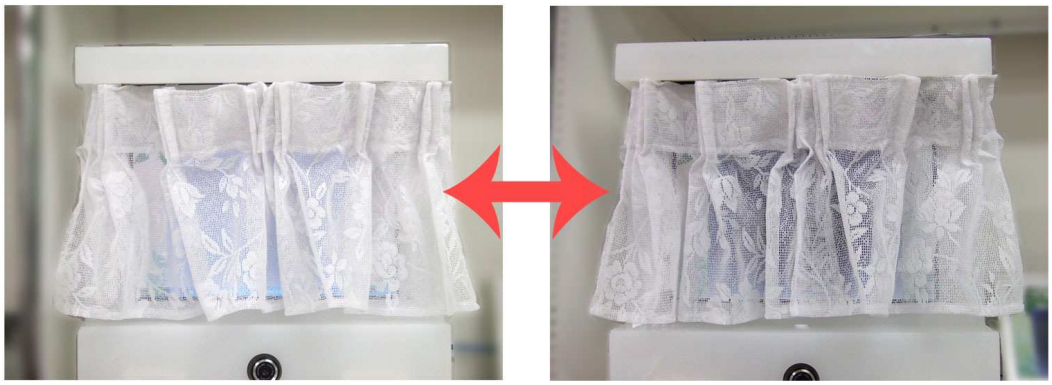

図 6.12: カーテン越しに部屋の照明の状態を知る

- カメラが物理的に露出していることは元から気にならなかったので，カーテン丈が長くなっても特に使用感に変化はない．
- いつもディスプレイに注目して利用していたので，実際に相手の姿が見えるディスプレイの高さよりもカーテンが長いことには，逆に違和感を感じる．

音量を制御する事例

カーテン操作によって，スピーカー音量やマイク音量を制御する場面が度々見られた．

スピーカー音量を制御する事例としては，T 研究室にいた教員が集中して作業したい時，端末のスピーカーから聞こえる S 研究室の賑やかな話し声が気になり，カーテンの開き具合を小さくして音量を下げる場面が週 2~3 回程度観察された．

また，マイク音量を制御する事例として，T 研究室の教員が大きな音の出る工作機

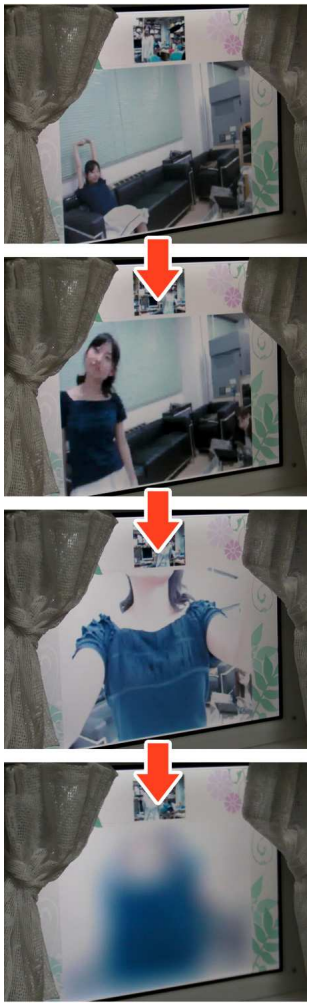

図 6.13: 部屋の中のソファで寝ようとした時に，相手にその姿を見られないようカーテンを閉めた．

械を使うとき，機械の動作音が相手に迷惑であると考え，S 研究室に騒音が聞こえないようにカーテンを閉めるといったことがあった．

カーテンを半開きにすることの多かった利用者へのインタビューでは，その理由として以下のようなコメントを得た．

- 自分の声が相手側に聞こえすぎるのを防ぎたかった．
- 相手側から突然笑い声が聞こえて驚くことがあり，作業の集中力が切れてしまうため若干閉めた状態にしていた．
- 用事のないときは，相手側からの音声が小さく聞こえる状態が良かった．

また，カーテンの開閉で映像と同時に音量も変化することに違和感はあったかとインタビューしたところ，特に違和感なく使用できたという意見が多く得られた．

チャイム機能の利用事例

チャイム機能を利用した事例としては以下のようなものがあった．

S 研究室にいた学生が T 研究室の教員に相談をしようとしていたが，T 研究室のカーテンは完全に閉じられていた．しかし部屋の明かりが点いていたことから教員が在室していると考え，チャイム音を鳴らしたところ，教員が気づいて T 研究室のカーテンを開け，ビデオチャットで相談をした．

また別の事例としては，夜中，T 研究室にいた教員が出前を頼む際，S 研究室側のカーテンは半開き状態で，明かりがついており人影が動いているのが見えた．そこで S 研究室でも誰か一緒に出前を頼みたい人がいないかどうか聞いてみるため，大きな声を出して呼ぶ代わりにチャイムを鳴らしたところ，S 研究室の一人が気づいてカーテンを開け，ビデオチャットで即座に聞いた．

6.3.3 考察

実験結果を踏まえて，コンセプトで挙げた「日常空間での常時利用」「プライバシーの制御」「操作の分かりやすさ」「日常空間に馴染むデザイン」という 4 つの課題について考察する．

日常空間での常時利用

本研究の目的の 1 つは，日常空間で常時利用できるコミュニケーション手法の実現である．それも，単にアンビエントなコミュニケーション手法として常時利用するだけではなく，直接的なコミュニケーションをすることもでき，状況に応じて両者の間を移行できることがポイントとなる．

まず直接的なコミュニケーションが行われたかどうかに関しては，6.3.2 の事例で述べたように，カーテンが完全に開いた状態で多くのユーザが手軽に相手に話しかけることができていた．特に，わざわざ電話をかけて聞くほどでもない些細な話題，質問に多く使われていた点が特徴的である．これは常時接続によって会話を始めるための敷居が下がったことが理由として考えられる．常時接続ではない一般的なビデオ会議システムや電話では会話のきっかけを掴みづらいのに対し，なめらカーテンでは常時接続されていることでお互いの相手の様子を見て相手に話しかけるタイミングが取りやすい．

次に，コミュニケーション形態の移行に関しては，6.3.2 のプライバシー保護や 6.3.2 の音量調節の事例のように，直接的なコミュニケーションからアンビエントなコミュニケーションへの移行が行われていた．逆方向への移行として，6.3.2 で述べたようにチャイム機能を利用することで，直接的なコミュニケーションへ移行することができた．また，6.3.2 のアンビエントなコミュニケーションの事例で挙げたような，アンビ

エントな情報から相手の様子を知り，直接会いに行くという行動も一種のコミュニケーション形態の移行と言える．

研究室の中で従来から使われていた内線電話やSkypeなど他のコミュニケーション手段との関係について述べる．まず，なめらカーテンを使い始めてから内線電話を使う頻度は減った．話しかける前に相手の様子を見ることのできるなめらカーテンの方が，電話に比べてより話しかけやすいためと考えられる．一方，Skypeのようなインスタントメッセンジャー（以下，Skype）はなめらカーテン導入後も利用されていた．両者の利用頻度は,「会話内容」「作業状態」「即時性」「Skypeの利用経験」によって変化する傾向が見られた．以下，具体的な事例について述べる．まず，会話内容が周りの人にあまり聞かれたくないような個人的な内容である場合，なめらカーテンよりSkypeが利用される傾向があった．次に，ユーザの事前の作業状態に近いメディアが優先的に利用される傾向にあった．すなわち，一方の研究室で複数人で会話が行われている状況では，多くの場合なめらカーテンが利用され，逆に個人でPCで作業していて，かつ話しかけたい人がSkypeにいる場合は，Skypeが優先的に利用された．さらに，他研究室にいる特定の人にすぐにコミュニケーションを取りたい場合は，Skypeと比べて割り込み能力が高く即時性に優れる点から，なめらカーテンが優先的に利用された．最後に，Skypeの利用経験も利用頻度に影響を与えた．学生の半数程度はSkypeを日常的には利用しておらず，特に雑談／議論している時はSkypeに気づかないことも多かった．よって，なめらカーテンの方が（お互い研究室にいる場合は）信頼できるコミュニケーションチャンネルとして機能した．このようにSkypeとなめらカーテンはお互い有用なメディアとして使い分けられる傾向にあった．

プライバシーの制御

生活空間で利用する上での課題の一つが「柔軟なプライバシーの制御」であった．本実験では，6.3.2のプライバシー制御の事例で挙げたように，仮眠をとるときや私的な会話をするときにカーテンの操作が行われていたことから，有効に機能が利用されていたと言える．

常時連続利用するコミュニケーションシステムを室内に置くということに対して，特にプライバシー面での不安や抵抗を感じることも懸念された．しかし，今回の実験ではそのような不安や抵抗感などを訴えるといった人は無く，肯定的に受け入れられていたと言える．肯定的に受け入れられた理由としては，まず第一になめらカーテンが日常的なコミュニケーション手段として実際に活用されており，十分な実用性や利便性があったという点が大きい．そして，上記の柔軟なプライバシー制御ができるということも不安の解消につながっていたと言える．ただし，本実験環境が大学の研究室という，家に比べてプライバシー的な問題が起こりにくい環境であることも関係している．家ではよりプライベートな時間・空間が増えるため，カーテンを閉じる機会も増えると考えられるが，そのような状況でも有効に機能するかについては，さらなる検証が必要である．

操作方法の分かりやすさ

プライバシー制御の操作方法に関しては 6.3.2 で述べたように，ほとんどの利用者が「カーテンを閉めることでこちらの様子を隠す」という使い方をすぐに理解していた．これは「プライバシーを守るためにカーテンを閉める」というカーテンのメタファが有効に働いたためと考えられる．

「カーテンの開閉によってマイクやスピーカーの音量が変化する」という点に関しても多くのユーザは違和感なく使用していた．カーテンによって音が伝わりにくくなるのは，カーテンメタファとの対応付けはやや弱い[7] ものの，単純な機能のため，一度カーテンを開閉して使ってみることですぐに理解できる機能であると言える．

「カーテンを素早く開閉する」というチャイム機能のジェスチャ操作は，カーテンのメタファには無いものであり，操作方法を知らないと使えない機能である．しかしながら，利用者にチャイム機能と操作方法の説明をするとすぐに使い方を覚えた．何らかの操作方法の提示が必要であるが，ジェスチャ操作自体はシンプルで分かりやすいものだと考えられる．ただし，チャイム機能の操作方法としては，3.2.2 で挙げたように「物理的なスイッチ」や「カーテンの前に一定時間立つ」といった他の方法も考えられる．これらの操作方法と比較することで，より分かりやすい操作方法を検討したい．

カーテンを閉めた際に，相手側では「画面にぼかしがかかる」ことで見えにくくなるのに対し，自分側は「カーテンによって画面が隠される」ことによって見えにくくなるという違いがあり，お互いに見えにくくなる仕組みが非対称となっている．この非対称性は実世界では起こらない事象ではあるものの，利用者からは非対称性についての疑問やコメントなどは特に無く，違和感なく受け入れられていた．その理由として，カーテンを閉める際に，自分側の画面上部に表示されている「自分の映った映像」にぼかしがかかるため，相手側に表示されている「自分の映った映像」にもぼかしがかかることをユーザが推測・理解できた，ということが挙げられる．すなわち，お互いに見えにくくなる仕組みは非対称であるものの，その仕組みをユーザが容易に理解できたことで，非対称性が実用上の問題にはならなかったと言える．

6.2.1 で述べたように,「カーテンによって画面が隠される」ことで一方的な覗き見を「緩く」防ぐようになっているが，閉めたカーテンをめくって覗くという使い方も可能である．しかし，実際にはそのような使い方はあまりされていなかった．その理由として，まず，なめらカーテンに取り付けられたカーテンが小さく，さらに波状になってため，布をめくりにくかったことが挙げられる．さらに，カーテンが少し開いていれば相手の様子（カーテンを開けているかなど）は大体分かるため，わざわざカーテンをめくって相手を見る必要性があまり無かった．また，カーテンをめくって見るのは相手を「覗き見」をするのは周囲の目を気にしてやりにくい，という心理的な抵抗感も理由として考えられる．

[7] あまり一般的ではないが，遮音カーテンのメタファに近いと言える

実験内容でも述べたようになめらカーテンは生活空間での利用を主に想定している．そのため本実験環境である大学の研究室の中でも，特にリビングのような生活空間に近い環境を選んで設置・運用をした．結果，カーテン端末のデザインに関しては特に不満などは聞かれず，違和感なく受け入れられていた．今後は家庭環境へ導入し，様々な生活空間で運用することで，より生活空間に馴染むデザインについて検証していきたい．

実際に生活空間に設置する場合，現在のなめらカーテン端末はやや大きく，設置場所が限定されてしまうため，端末の小型化が課題である．端末を小型化し，複数設置することにより，1 対多のコミュニケーションをするという発展も考えられる．

端末のカーテンを閉じた際にカメラが見えている点に関しては，一時的にカーテン丈を変更して調査したものの，6.3.3 で述べたように利用者は特に気にしていなかった．利用者のコメントにもあるように，端末を見るときの意識は主にディスプレイに向いており，カメラが見えているかどうかは端末をデザインする上でそれほど大きな問題にはならないようである．もちろん，寝室などのプライバシー保護が強く求められる空間では，さらに配慮したデザインが必要になる可能性がある．例えば「物理的にカメラにふたをできるようにする」「カメラの向きを変えられるようにする」といった方法で安心感を与えるデザインが考えられる．このようなデザインについても様々な生活空間での検証をしていきたい．

6.4 関連研究

本章では，本研究に関連の深い研究を「直接的なコミュニケーション」「アンビエントなコミュニケーション」「プライバシーの制御」という 3 分野から取り上る．

6.4.1 直接的なコミュニケーション

Xerox の Video Wall[Root, 1988] は，遠隔の 2 つの場所を大きなディスプレイで繋ぎ，常時接続することで二つの空間が相互に交わる感覚を得ることができる．VideoWindow System[Boyle, 2000] は大規模なビデオ会議システムであり，壁に高画質のテレビを設置し，原寸大に相手の映像を表示する．さらにスピーカーも複数設置し，相手の立ち位置に応じて音声の聞こえる方向も変換する．t-room[Yamashita, 2007] は，巨大な 6 枚のディスプレイを 8 角形型（うち 2 辺にはディスプレイはない）に並べ，天井から 8 台の Web カメラで撮影した映像を相手側のディスプレイに表示するシステムである．遠隔で同じ部屋にいる感覚を共有し，実世界でのやりとりが遠隔地においてもシームレスに通用する環境を目指している．HyperMirror[Morikawa, 1998] は，お互いの等身大の映像を大型スクリーンで共有することで，あたかも一緒にいるような感覚を与える．遠距離の 2 箇所間で撮影／合成した映像をスクリーンに投影し，対話者の片方が相手の所に出向いているような映像を実現する．

これらの研究は，遠隔地間で高い臨場感を共有することを主眼としており，ミーティングルームなどの環境には適しているが，日常生活空間で常時利用することはあまり想定されておらずプライバシー制御が問題になる．

6.4.2 アンビエントコミュニケーション

SyncDecor[Tsujita, 2009] は，相手の状態をさりげなく知らせるために，遠隔地に置かれた日用品（e.g. ゴミ箱，ランプなど）が同期して動作するシステムである．影電話 - Teleshadow plus[Hashimoto, 2007] は，影を使う事でプライベートな状況においても同じ空間にいるようなやりとりができる行灯型の遠隔コミュニケーションシステムである．つながり感通信 [Miyajima, 2003] は，人の存在やそれに伴う様々な手がかり情報（動き方，気配など）を用いた，親しい関係にある人と人との間での遠隔コミュニケーションを支援するシステムである．

これらのコミュニケーションシステムは日常空間において常時利用できるが，直接的なコミュニケーション機能を備えないため，単体で具体的な意思伝達をすることは困難である．

6.4.3 プライバシーの制御

Greenberg ら [Kuzuoka, 2000] はメディア空間におけるプライバシーレベルの柔軟な調節を行なうため，人形などの物理的な「代替品」を用い，直接的なコミュニケーションとアンビエントなコミュニケーション形態を柔軟に移行する手法を提案している．また，その応用として物理的なつまみを回すことで Web カメラの映像をぼかしてプライバシー保護を行なう手法を提案している [Neustaedter, 2003][Boyle, 2000]．Buxton の Door Mouse[Buxton, 1995] では，プライバシーを保護する行為として，ドアを閉める動作を検出することで，メディア空間においてのプライバシー保護を実現している．同じく児玉ら [Kodama, 2005] は，ドアの開き具合を計測し，Web カメラの映像の透明度や音声の音量を変化させる，プライバシー制御手法を提案している．また，ノート PC のディスプレイの開閉状況を取得して，デスクトップアイコンや Web ブラウザの表示状況などのプライバシー制御を行う事例 [Masui, 2004a] も存在する．

これらの研究から，プライバシーを制御するために物理的な代理品を利用することや，実世界におけるプライバシー保護行為のメタファを活用することが有効と考えられる．遠隔地間を常時つなぐ場合は，このようなプライバシー制御に加えて，どちらかが一方的に覗き込むことを防ぐ「公平さ」が求められる．また，Door Mouse や児玉らのシステムではプライバシーを制御するためのメタファおよび物理デバイスとして室内のドアを用いているが，プライバシー制御の対象となるビデオチャットの画面と物理デバイスが離れているためやや操作の直接性に欠ける．物理デバイスとプライバシー制御対象との関連性を分かりやすくするために，両者を一体化することが望ましい．

6.5 結論

本章では，ユビキタス環境に溶け込むインタフェースとして，生活空間にPC環境のビデオチャットを溶け込ませる「なめらカーテン」を提案，構築した．なめらカーテンでは，カーテンのメタファを採り入れたカーテン型デバイスによって，直感的にプライバシーレベルを制御し，コミュニケーション形態の柔軟な移行を可能にする．さらに，なめらカーテンを用いた14ヶ月以上の期間実証実験を行ない,「日常空間での常時利用」「プライバシーの制御」「操作の分かりやすさ」「日常空間に馴染むデザイン」といった観点から本研究の有効性を確認した．

第 7 章

ユビキタス環境に溶け込むインタフェースの開発手法

概要

本章では，我々がこれまで開発してきたユビキタス環境に溶け込むインタフェースの中でも，開発事例の多い「GUI とデバイスを連携したシステム」の開発手法を紹介する．PC 環境や Web 環境をユビキタス環境を溶け込ませようとすると，従来の PC 環境と同様にディスプレイや GUI が併用されることが多い．しかし，様々なセンサ／デバイスなどのハードウェアと連携した操作を行う点で，従来のマウス／キーボード操作を前提とした GUI とは大きく性質が異なっている．そのため，従来の PC 向けの GUI 開発ツールが適用しにくく，開発プロセスが複雑化しやすかった．我々は多数のシステム開発を通じ，こうした問題を解決するためのノウハウを貯め，一定の開発手法を確立してきた．我々の開発手法のポイントは，GUI とデバイス制御プログラムの間に「デバイスサーバ」と言われるミドルウェアを設けることで，両者のプログラムを分離し「疎結合」な状態にするという点である．こうすることで，GUI とデバイスというレイヤーの異なる開発を並行してスムーズに進めることができるようになった．本章では，こうした GUI-デバイス複合型システムの開発手法について，具体的な開発方法や様々な事例，課題を交えながら紹介する．

7.1 はじめに

近年，ユビキタスコンピューティングを想定したアプリケーションの研究開発が盛んになるにつれて，その開発に適した様々なツールキットが登場してきた．たとえば，Phidgets，Gainer，Arduino などのセンサ／アクチュエータを用いたシステム開発を支援するためのツールキットが注目されている．Phidgets[Greenberg, 2002] や Gainer[Kobayashi, 2006] は，USB でパソコンに接続するデバイス群 と，それらを制御するライブラリ群から構成される．Arduino[Mellis, 2007] は，USB 経由でプログラムを書き換え可能な汎用 I/O デバイスと，Processing[1] に統合された開発環境からなる．こうしたツールキットを活用することで，多様なセンサ／アクチュエータを PC から比較的手軽に扱えるようになってきた．

しかし，6 章で紹介したなめらカーテンのような「ユビキタス環境に溶け込むインタフェース」の開発では，単にセンサを扱うだけでなく，複数のセンサを組み合わせたロジックを記述したり，GUI (Graphical User Interface) や Web サービスと連携させたりといった，複雑なソフトウェア開発工程が必要となる．こうした状況では，多様なハードウェアとソフトウェアを組み合わせる点から，従来の PC や携帯端末向けに確立されたマウス・キーボードを前提とした UI 設計や，インタフェースビルダーのような開発ツールを単純に適用できないケースも多い．

一方，ユビキタスコンピューティングにおける開発手法としては，ユビキタスコンピューティングにおけるプログラミングモデル [Tei, 2008] の提案などが行われているが，これらはプログラム内部のやや抽象的なモデル化に焦点を当てており，実際のアプリケーションを開発する上での具体的／効率的な実装方法については議論されていない．

こうした中，我々は UI 研究の立場から「ユビキタス環境に溶け込むインタフェース」を含む，多数のシステム開発を行ってきた．特に，我々がこれまでに開発したシステムの中では「GUI とデバイスを連携したシステム」の事例が多く，例えば以下のようなシステムを開発してきた．

- なめらカーテン [Handa, 2009]
- ハングル ガングル [Kadomura, 2011]
- DrawerFinder [Komatsuzaki, 2011]
- LunchCommunicator [Kotani, 2011]
- MouseField [Masui, 2004b]
- インタラクティブな掃除機 [Ogasawara, 2007]
- Complete Fashion Coordinator [Tsujita, 2010]

[1] http://processing.org/

- IODisk [Tsukada, 2010b]
- タグタンス [Tsukada, 2008]
- EyeWish [Watanabe, 2009]

これらの GUI とデバイスを組み合わせたシステム（以下，GUI-デバイス複合型システム）は，いずれも構成が複雑になりがちで，複数人で開発することも多い．そのため，上述したツールキットや開発手法を単に用いるだけでは，なかなか効率的に開発が進まないという状況も増えてくる．

こうした問題を改善するために，我々は何度も GUI-デバイス複合型システムの開発を繰り返す中でノウハウを貯め，実装方法を洗練させてきた．我々の開発手法のポイントは，GUI とデバイス制御プログラムの間に「デバイスサーバ」と言われるミドルウェアを設けることで，両者のプログラムを分離し「疎結合」な状態にするという点である．こうすることで，GUI とデバイスというレイヤーの異なる開発を並行してスムーズに進めることができるようになった．また，我々がこれまでに開発してきた，デバイス制御のためのソフトウェア群「MobiServer[Tsukada, 2010a]」を活用することで，デバイスサーバの開発を支援する点や，GUI 側の開発に「Flash」を用いることで GUI デザインの自由度を高めるといった点も重要な特徴である．本章では，こうした GUI-デバイス複合型システムの開発手法について，具体的・実践的な実装方法や様々な事例，課題などを交えながら紹介する．

7.2 ユビキタスコンピューティングにおける UI 開発の問題

ユビキタスコンピューティングにおいて UI を開発する際，大きく分けて「非 WIMP 方式の GUI」「GUI とデバイスの同時開発」「デバッグと分担作業」といった点が問題になる．

7.2.1 非 WIMP 方式の GUI

ユビキタスコンピューティングにおけるシステム開発においても，多くの場合において，従来のデスクトップ環境と同様に情報の表示や操作にディスプレイおよび GUI が使われる．たとえば，家電機器などにディスプレイが埋め込まれた情報アプライアンスや，リビングのテレビやモバイル端末と連携したシステム，デジタルサイネージなど，色々な場所に大小様々なディスプレイやプロジェクタが利用されている．

一方，ユビキタスコンピューティング向けの GUI は従来の PC 向けの GUI とは性質が異なっている．従来の PC 向けの GUI，いわゆる WIMP (Window, Icon, Menu, Pointing device) 方式の GUI は，マウスやキーボードを使った近接操作を前提に設計されている．しかしユビキタス環境では，マウスやキーボードに限らず多様なセンサやデバイスを用いてインタラクションを行うため，WIMP 方式とは違ったデザインの GUI が求められる．例えば，ターンテーブル型のデバイスを使って写真や動画をコントロー

図 7.1: IODisk 写真ビューア

ルする IODisk は，ディスクの回転操作にあわせて，画面上のコンテンツも円盤状に回転・アニメーションする GUI となっている（図 7.1）．また，IC タグのつけられた物を置いたり，動かしたりすることで音楽などのコンテンツを操作をする MouseField（図 7.2）では，例えば置いた CD ジャケットを回転する動作により音量を操作する．このような操作方法を視覚的に表現するため，MouseField では図 7.2 の画面内のように円弧状に変形したデザインのボリュームインジケータとなっている．さらに MouseField では CD ジャケットを上下に動かすことで再生リストを操作する．このとき画面内の CD ジャケットがアニメーションすることで視覚的なフィードバックを返す．このように独自のデバイスを使って操作をする場合，従来の GUI 部品とは異なった動きや形の GUI が必要になる．

また，ユビキタス環境では「なにか別の作業をしながら利用する」「立ったり歩いているとき使う」「テレビのように機器と離れた状態で利用する」「あまり操作せず閲覧が中心」といった生活の中での多様な利用状況に応じた GUI をデザインする必要がある．そのため「机の前で椅子に座り，キーボードやマウスで比較的集中して操作する」という限定された状況を前提にデザインされた既存の GUI に比べて，求められるデザインのバリエーションが広い．PC 用の GUI と携帯端末の GUI，テレビ向けの GUI がそれぞれ大きく異なるように，画面のサイズや画面からの距離，利用時間，場所に応じて，GUI 部品の大きさや画面内の情報量，レイアウト，画面遷移の仕方が大きく変わる．

このように，ユビキタスコンピューティング向けの GUI では，PC 向けの GUI とは異なる多様なデザインが求められるため，VisualStudio のユーザインタフェースエディタや Xcode のインタフェースビルダー，HTML オーサリングツールの Dreamweaver といった，WIMP に最適化された従来の GUI デザインツールを単純に適用することが難しい．上で述べた IODisk や MouseField のように，従来の GUI 部品とは動きや形が大きく異なる GUI は，GUI デザインツール上でボタンなどの既存の GUI 部品を配置す

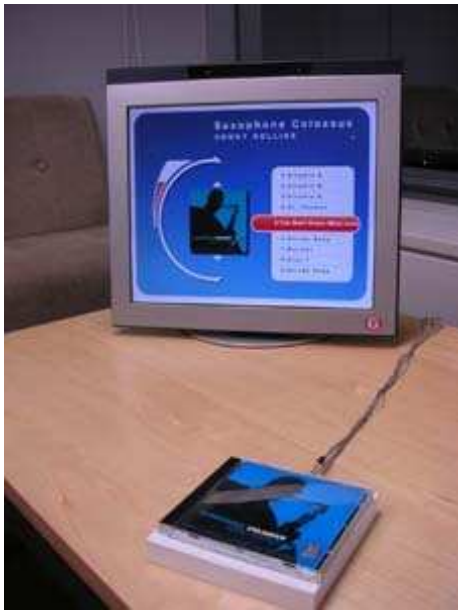

図 7.2: MouseField

るだけでは作ることができない.

7.2.2 GUI とデバイスの同時開発

従来のマウスやキーボードを前提とした GUI との大きな違いとして，様々なセンサなどのデバイスを用いた「多様な操作方法」が挙げられる．そのためシステムを作る際，これまでのように画面の中だけでなく，ハードウェアの設計や実装，制御まで考える必要がある．

研究開発初期のプロトタイピングの段階では，まだシステムの仕様がはっきりしていないことが多く，GUI とデバイスをお互いにすり合わせながら実装することになる．この場合，種類の違う 2 つの開発を調整しながら同時並行に進めるという難しさがある．また，GUI とデバイス制御は開発のレイヤーが大きく離れている点も問題となる．デバイス制御はハードウェア寄りの低いレイヤーなのに対し，GUI はユーザ寄りの比較的高いレイヤーになる．レイヤー同士が近い開発であれば，使用する言語や開発環境が似ていることが多く，同時にまとめて開発しやすいが，レイヤーの異なる GUI とデバイス制御ではそれらが大きく異なるため同時に開発しにくい.

7.2.3 デバッグと分担作業

前述のように，GUI とデバイスを組み合わせたシステムでは両者の連携が重要となる．しかし，両者のプログラムをまとめて 1 つにしたり，両者の依存関係が強く，単体

では動かせない「密結合」した状態になると，デバッグや分担といった開発作業が難しくなる．

システムが密結合になると，例えば「片方を直すともう片方も動かなくなってしまう」といった事態が起こりやすくなる．その結果，GUI とデバイスの双方をまとめて直していくことになり,「どちらかに集中して開発しにくくなる」「問題の切り分けがしにくくなるためデバッグが複雑になる」といったことにつながる．

さらに密結合しすぎると複数人での分担作業が難しくなる．GUI 担当とハードウェア担当に分かれて作業する場合，密結合していると「片方を直している間，もう片方の担当者が作業できない」ということになり効率が悪い．

7.3 GUI-デバイス複合型開発

ここでは，ユビキタスコンピューティングのためのデバイスと GUI を組み合わせた開発における様々な問題を考慮した，我々の開発手法を紹介する．

本開発手法のポイントは，デバイス側と GUI 側のプログラム・開発作業を分離することにある．そのために，まず図 7.3 のように，デバイスと GUI の間には「デバイスサーバ」と言われるミドルウェアを設け，GUI 側のプログラムが無い状態でもデバイスの制御・開発・デバッグをできるようにする．また，GUI 側の開発には主に Adobe 社の「Flash」を用いることで，GUI デザインの自由度を高める．さらに，マウス・キーボードによるエミュレーション機能などを用意することで， GUI 部分だけを切りだして開発・デバッグできるようにする．

こうして GUI とデバイス制御を別々のプログラムに分離し「疎結合」な状態にすることで，レイヤーの異なる開発作業をうまく切り分けながら両者を連携できるようにした．その結果，GUI とデバイスのどちらかに集中して開発したり，問題の発生箇所を切り分けてデバッグしたり，複数の担当者が関わる作業をスムーズなものにすることができる．また，両プログラム間は TCP Socket や HTTP といったプロトコルで通信することで，LAN やインターネット経由で結合テストできるようになり，プログラムやハードウェアを別の PC に度々移し替える手間も減る．

以下では，我々の手法の特徴である「Flash」「デバイスサーバ」「疎結合」についてそれぞれ詳しく述べる．

7.3.1 Flash

Adobe Flash は主に RIA (Rich Internet Application) と言われる高度な Web アプリケーションの UI 作成や Web 上でのアニメーション表現，動画の再生などに広く用いられている技術である．最近では Adobe AIR という技術によって，Flash を用いたデスクトップアプリケーションの作成も可能になっている．

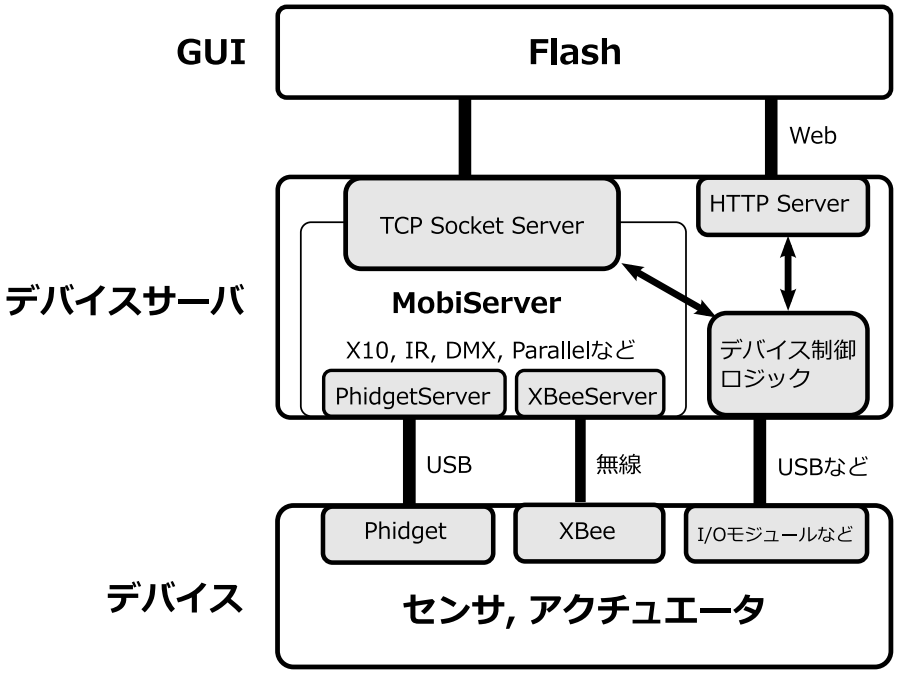

図 7.3: GUI-デバイス複合システムの構成

Flash を用いる理由

Flash は一般的に Web サイトやアニメーション作成向けのツールとされているが，GUI-デバイス複合型開発に適した特徴を備えている．

Flash が GUI-デバイス複合開発に向いている最大の理由は，他の GUI 開発ツールに比べて「非 WIMP 方式の GUI を開発しやすい」ためである．7.2.1 節で述べたように，非 WIMP 方式の GUI では「マウスやキーボード以外の多様なセンサやデバイスを用いたインタラクション」や「従来の GUI 部品とは異なった動きや形の GUI」といった多様なデザインが求められる．WIMP に最適化された従来の GUI デザインツールはデザインの自由度が低く，多様なデザインの実現が難しいのに対し，Flash はデザインの自由度が高い．Flash はグラフィックツールやアニメーション用のタイムラインを内蔵しており，独自の形状・動きを持つ GUI 部品を作りやすく，それらを柔軟に配置し，動かすことができるためである．また，単なるグラフィックツールとも異なり，プログラミング言語（ActionScript）や開発環境を備えているため，複雑な機能の UI も作ることができる．

これまでに Flash を使ったことのあるデザイナーや UI エンジニアが多く，そのようなデザイナーとの共同作業がしやすいことも理由として挙げられる．7.2.2 節で述べたように GUI とデバイス制御はレイヤーが大きく異なるため，それぞれのレイヤーを得意とする人同士で作業すると効率が良い．デザイナーや UI エンジニアにとっては，新たにツールを覚えることなく，既存のノウハウを活かすことができる．

その他の理由として，Flash はデスクトップ，モバイル端末，Web ブラウザなど多くのプラットフォーム上で動作するため，様々なシステム構成に柔軟に対応できることや，C++や C#，Java といった他の高級言語に比べて動画（ストリーミング）や音声，画像などマルチメディアコンテンツを扱いやすいことが挙げられる．

7.3.2 デバイスサーバ

Flash は上述のように GUI-デバイス複合型開発に適した特性を備えるが，OS の機能に直接アクセスすることができないため[2]，直接外部デバイスを制御することは難しい．そこで，我々はデバイスと Flash を仲介するためのミドルウェア（以下，デバイスサーバ）を用意した．デバイスサーバの主な役割は，PC につながった各種センサやアクチュエータを制御し，Flash から直接利用できるような API を提供することである．

Flash に対して API を提供する方法としては，TCP Socket サーバや CGI のような HTTP サーバとして動作させる．これは Flash が TCP Socket または HTTP 経由で外部ネットワークにアクセスできるためである．TCP Socket は HTTP に比べて高速で双方向通信可能なため，リアルタイムなデバイスの制御に適している．一方，HTTP はシンプルなプロトコルであり，Flash 本来の使い方として Web 上での HTTP 通信に長けていることもあって，比較的実装しやすい．

TCP Socket/HTTP サーバとデバイスを制御するプログラムがデバイス制御ロジックである．デバイス制御ロジックはアプリケーションごとに固有であり，デバイスサーバの中心的なプログラムとなる．

センサやアクチュエータは Phidget や Gainer などの I/O モジュールを利用して USB などで PC に接続して制御する．

デバイスサーバは，Flash のクライアントと接続されていない状態，すなわちデバイスサーバ単体でもデバイスの動作を確認できる機能も開発を進める上で重要となる．デバイスサーバ自体に簡単な GUI を持たせることで，手動でデバイスを制御したり，センサのパラメータや各デバイスの動作状況のログを表示する．

MobiServer

デバイスサーバには多数の物理的なデバイスを制御する機能が求められる．こうした実装を支援し，汎用的なデバイスサーバとしても使えるソフトウェア群「MobiServer」を開発した [Tsukada, 2010a]．

MobiServer は，以下のような多様なデバイス群を制御するためのミドルウェアの総称である．

- PhidgetServer: USB 接続のセンサ・アクチュエータ群である Phidget を制御する
- X10Server: 電灯線通信を行う X10 デバイスを用いて照明や家電を制御する

[2]たとえば，Windows 環境においては，Win32API を介してシリアルポートを開くことはできず，外部 DLL(Dynamic Link Library) を直接呼び出すこともできない．

- IRServer: USB 接続の学習リモコンを用いて，エアコンやテレビなどの家電を制御する
- ParallelServer: 安価な USB パラレル変換モジュールを用いてデジタル I/O を制御する
- DMXServer: フルカラー LED 照明などを制御する
- XBeeServer: 無線通信モジュール XBee を組み合わせた無線センサシステムを制御する

これらの MobiServer はそれぞれ TCP Socket サーバやデバイス動作を確認するための GUI を備えており，単体でも 1 つのデバイスサーバとして機能する．そのため単純に 1 つのデバイス制御をするだけであれば，新たにデバイスサーバを書かなくてもよい．一方，MobiServer が対応していないデバイスを用いる場合や，複数のデバイスを組み合わせた複雑なシステムを作る場合，MobiServer と別のプログラム（デバイス制御ロジック）を組み合わせて利用することもできる．こうすることで必要最小限のプログラムを書くだけでデバイスサーバを構築することができる．MobiServer と他のプログラムの間の通信は，Flash と同様に TCP Socket を用いることが多い．

MobiServer はいずれも Web サイト[3] からダウンロードできる．MobiServer のうち PhidgetServer については後述するが，その他の MobiServer の詳細な機能や使い方については Web サイトまたは解説論文 [Tsukada, 2010a] を参照されたい．

PhidgetServer

MobiServer のうち，我々の開発でも頻繁に利用している PhidgetServer を例に紹介する．Phidgets[Greenberg, 2002] は図 7.4 のような USB で PC に接続できるセンサやアクチュエータ（スライダ，スイッチ，LED，モーターなど），汎用 I/O ボード（アナログ入力・デジタル入出力）などのキットで，PhidgetServer ではこれら様々なデバイスを制御する．

PhidgetServer は TCP Socket サーバとして動作し，TCP クライアントからテキストのコマンドを送信することでデバイスを制御したり，センサの状態をサーバから通知することができる．送受信されるコマンドは表 7.1 のようなテキストとなっている．

PhidgetServer の画面は図 7.5 のようになっており，出力系デバイスの操作や，入力系デバイスの状態およびログを表示するための GUI を備えている．この GUI により，TCP Socket クライアントが接続されていない状態でもデバイスの動作を確認できるため，デバイス単体での開発やデバッグ作業をすすめやすくなっている．

PhidgetServer 以外の他の MobiServer も，コマンドや GUI に違いはあるものの，基本的な使い方は PhidetServer とほぼ同様である．

[3] http://mobiquitous.com/mobiserver/

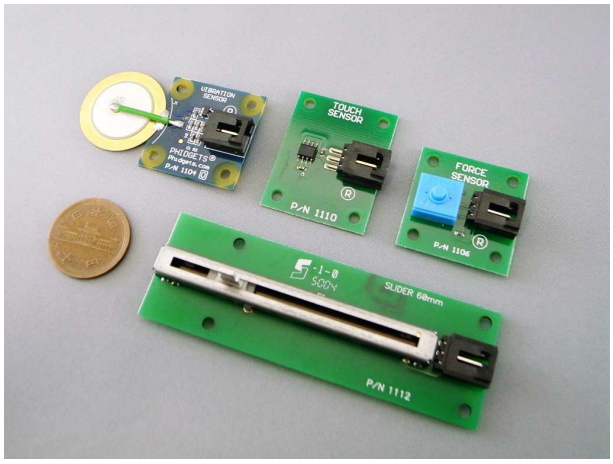

図 7.4: Phidgets のセンサの 1 例．上段 左:振動センサ, 中央:タッチセンサ, 右:圧力センサ, 下段:スライダ

表 7.1: PhidgetServer のコマンド例

Phidgets,In,InterfaceKit,27249,Analog,0,512

Phidget InterfaceKit (ID:27249) のアナログ入力ポート 0 が 512 へと変化

Phidgets,Out,Servo,3633,0,230

Phidget ServoMotor (ID:3633) のポート 0 の角度を 230 に変更

デバイスサーバの開発環境

デバイスサーバの開発環境や OS に関して原則的には制約は無いが，我々がデバイスサーバを開発する際は，主として OS に Windows，開発環境に VisualStudio，プログラミング言語に C#を用いている．その理由としては，まず Windows から既存のデバイスを利用するためのドライバやツールキット，情報が充実していることが挙げられる．また，最近ではネットブックと呼ばれる Windows 搭載の小型のノート PC が 3 万円前後で手に入るため，システム内に直接 PC を組み込むような場合でも，コストを押さえてコンパクトに作れるということも大きな理由となっている．そして，Windows のアプリケーションを開発するにあたっては，C++などに比べて C#が比較的容易であり，Java などと比べて低レイヤー（ハードウェア寄り）のプログラミングをしやすい．

上記のような理由から，MobiServer はいずれも C#で書かれており，Windows 上で動作する．そして我々が MobiServer を用いたデバイスサーバを作る際も，開発環境や動作環境を合わせた方が開発しやすいため，いずれのデバイスサーバも同じような構成となっている．

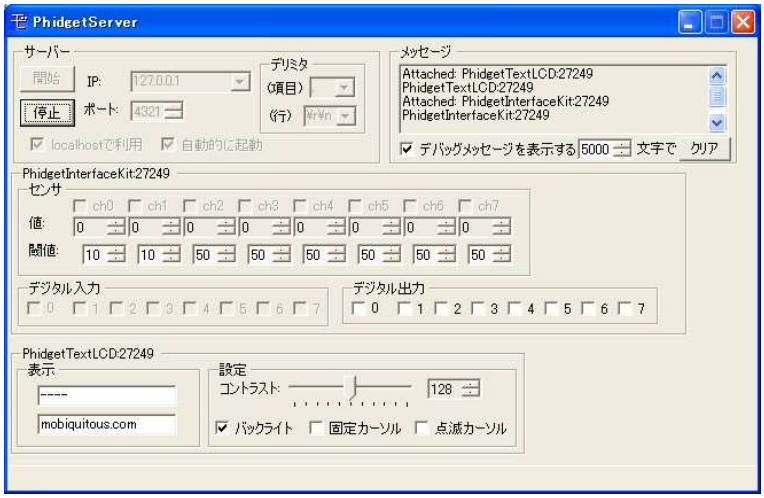

図 7.5: PhidgetServer

7.4 システム構成の分類

図 7.3 では GUI-デバイス複合型開発での基本的なシステム構成を示した．一方，具体的なアプリケーションに目を向けてみると，「Flash」「デバイスサーバ」「デバイス」という構成要素は共通するものの，要素の数や通信方法などの詳細は異なる．ユビキタスコンピューティングでは多様な利用形態があり得るため，システム構成のバリエーションも広い．

実際のシステム構成はアプリケーションによって異なるものの，ジャンルに応じて以下のようにいくつかの典型的なパターンに分類できる．

- 入出力デバイス系：デバイスで GUI を操作する
- コミュニケーションデバイス系：GUI を備えた複数の機器同士で通信する
- 実世界 Web 系：Web とデバイスが連携する

以下では，それぞれの分類の特徴および適したシステムの構成方法について述べる．

7.4.1 入出力デバイス系

センサを用いて GUI を操作，または GUI に応じてアクチュエータで出力するシステムを指す．GUI とデバイスが 1 対 1 の関係になっており，Flash とデバイスサーバを使った最も基本的な構成で実装できる（図 7.6）．

我々が開発したアプリケーションの中では，IODisk [Tsukada, 2010b] や Mouse-Field [Masui, 2004b]，ハングル ガングル [Kadomura, 2011]，インタラクティブな掃除機 [Ogasawara, 2007]，EyeWish [Watanabe, 2009] が入出力デバイス系のアプリケーションである．

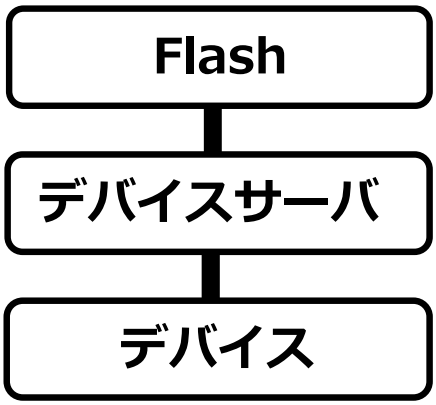

図 7.6: 入出力デバイス系

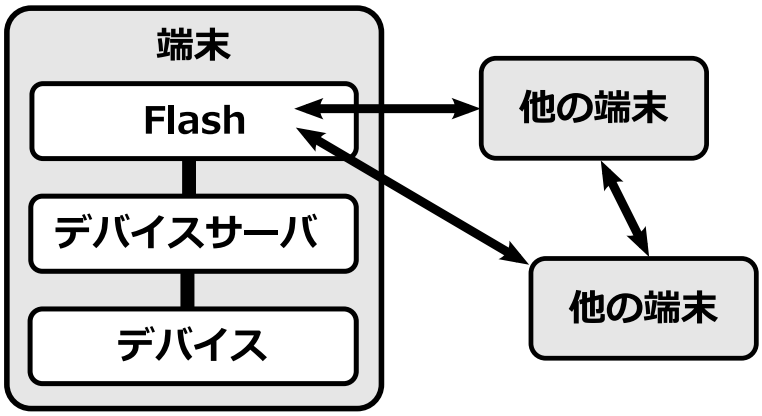

図 7.7: コミュニケーションデバイス系

7.4.2 コミュニケーションデバイス系

図 7.7 のようにディスプレイが組み込まれた機器（端末）を複数用いて，コミュニケーションするシステムを指す．各端末は Flash とデバイスサーバの基本構成で実装し，端末間の通信は Flash を利用してインターネット経由で通信すると良い．Flash を使った通信方法としては，TCP Socket や HTTP，ストリーミング用の RTMP[4] ，Flash 間で P2P 通信ができる RTMFP[5] などがある．

我々が開発したアプリケーションのうち，なめらカーテン [Handa, 2009] と Lunch-Communicator [Kotani, 2011] がコミュニケーションデバイス系のアプリケーションと言える．

[4]http://www.adobe.com/devnet/rtmp.html
[5]http://www.adobe.com/products/flashmediaserver/rtmfp_faq/

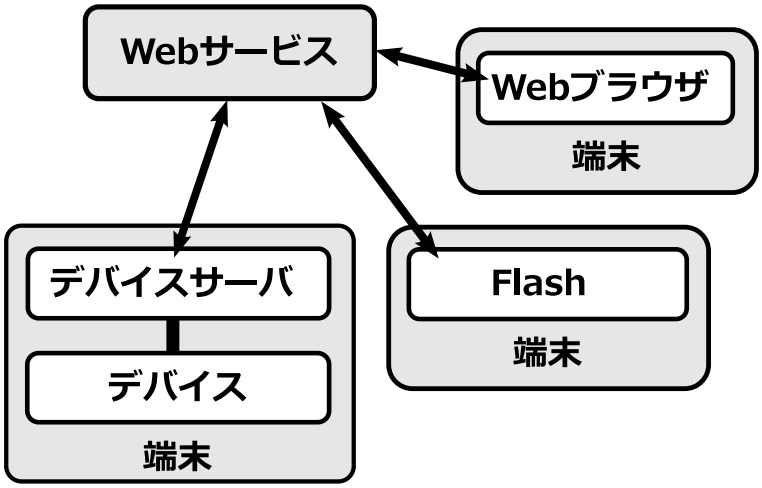

図 7.8: 実世界 Web 系

7.4.3 実世界 Web 系

図 7.8 のように実世界デバイスと Web サービスが連携したシステムを指す．Web 上の文書，写真，動画といったデータを利用するシステムや，Twitter・Facebook などのソーシャルネットワークおよび集合知を活用するシステム，またはセンサデータなどを Web 上に保存・共有するライフログ系のシステムがある．Web サービスを介して複数のアプリケーションやデバイスが連携できるのも特徴である．

デバイスサーバと Web サービス間の通信は REST (Representational State Transfer) や SOAP[6] を用いて HTTP 上で行う．既存の Web サービスを利用する場合は Web API が提供されていることも多い．また，チャットのようなリアルタイム性の高い通信をする場合は Flash の TCP Socket 通信なども利用できる．

我々が開発したアプリケーションのうち，Complete Fashion Coordinator [Tsujita, 2010] やタグタンス [Tsukada, 2008]，DrawerFinder [Komatsuzaki, 2011] が実世界 Web 系である．

7.5 開発事例

我々がこれまでに研究開発してきたシステムの中から，上述した 3 種類のシステム構成の代表的な開発事例をそれぞれ紹介する．入出力デバイス系として「IODisk」，コミュニケーションデバイス系として「なめらカーテン」，実世界 Web 系として「タグタンス」を取りあげる．

いずれのシステムの開発にもこれまでに述べた「Flash を用いた GUI や，MobiServer を利用したデバイスサーバによる構成」「複数人での分担開発，デバッグ作業」といっ

[6] http://www.w3.org/TR/soap12-part0/

表 7.2: 各事例の疎結合度とプロトコルの複雑さ

	疎結合度	プロトコルの複雑さ
IODisk	高	シンプル
なめらカーテン	低	複雑
タグタンス	高	やや複雑

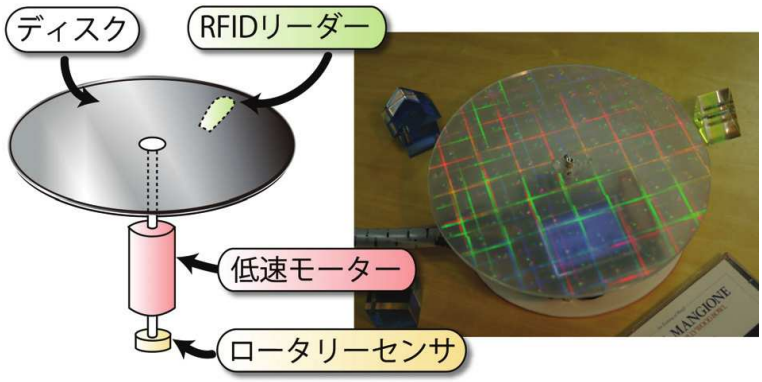

図 7.9: IODisk

た開発手法が取り入れられている．しかし，表 7.2 のように，それぞれシステムごとに疎結合の度合いやプロトコルの複雑さなどが異なっており，それによって開発の効率にも違いがあった．疎結合度が高くプロトコルがシンプルな IODisk の開発は非常にスムーズであったが，疎結合度がやや低めでプロトコルが複雑ななめらカーテンは効率さに欠ける点もあった．

ここでは，それぞれの開発事例の紹介を通して，本提案の有効性や課題について議論する．

7.5.1 IODisk

IODisk[Tsukada, 2010b] とは DJ が用いるターンテーブル風のデバイス/操作によって，写真や動画といったデジタルコンテンツを閲覧するシステムである（図 7.9）．

図 7.9 のディスクを少し回すと，手を離しても内蔵されたモーターによりそのまま一定速度で回り続ける．回転しているディスクに手で摩擦をかけることで，回転速度を細かく調整できる．そして，この回転速度に応じてコンテンツの再生速度を変えるというものである．

例えば IODisk を用いて複数の動画を切り替えながら見る場合，ゆっくりディスクを回すと再生開始，回転を速めることで早送り，逆回転により巻き戻し，回転を止めることで一時停止といった操作をする（図 7.10）．また，ディスクを瞬間的に急速回転

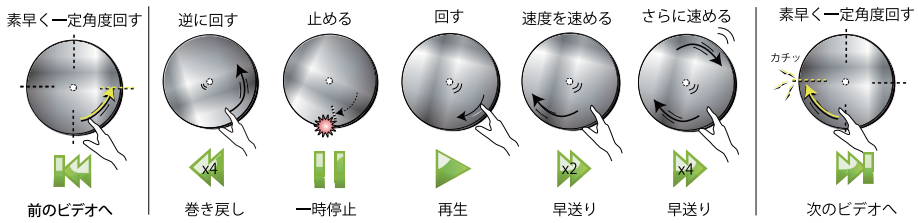

図 7.10: IODisk の操作

することで次のビデオに移動するといった操作もできる．そして，同様の操作方法で写真アルバムのスライドショーをコントロールする．

システム構成

IODisk は図 7.11 のような構成になっている．ディスク型デバイスは主にロータリーセンサとモーターから成り，いずれも Phidget および PhidgetServer により PC から制御する．ロータリーセンサから送られる回転角の情報は，ディスク制御ロジックによって回転方向や速度に応じて再生や停止，アルバム移動といったコマンドに解釈される．そして TCP Socket サーバを通じて，表 7.3 のコマンド文字列が改行区切りで写真ビューア[7] ・動画プレーヤー[8] などのアプリケーションに送られる．

表 7.3: IODisk で GUI 側アプリケーションに送られるコマンド

コマンド文字列	操作内容
next	次のビデオへ移動
previous	前のビデオへ移動
pause	一時停止
stop	停止
forward	再生
ff1 ～ ff4	早送り (4 段階)
reverse	巻き戻し
fr1 ～ fr4	巻き戻し (4 段階)

開発作業の分割

IODisk は GUI 担当とデバイス担当の 2 人で開発した．図 7.11 のうち，GUI 部分である写真・動画ビューアを一人が開発し，もう一人がデバイスサーバ（主にディスク

[7]https://github.com/tsuka-lab/IODiskPhotoViewer にてソースコードを公開
[8]https://github.com/tsuka-lab/IODiskVideoPlayer にてソースコードを公開

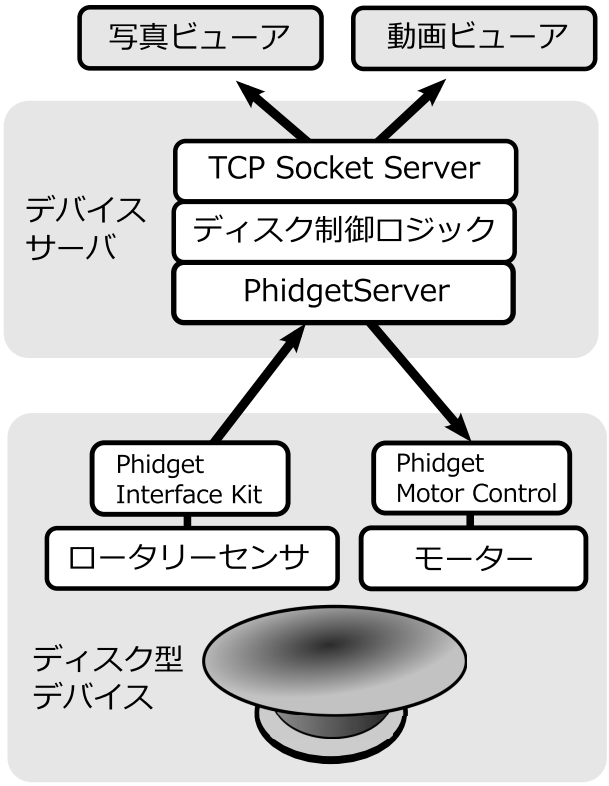

図 7.11: IODisk のシステム構成

制御ロジック）とディスク型デバイスを開発した．

IODisk では GUI 部分をデバイス関連の処理から明確に分離することができ，開発全体を通してそれぞれほぼ単独で開発作業を進めることが可能であった．システム構成で述べたように，GUI-デバイスサーバ間の通信は「表 7.3 のコマンドを GUI 側に送る」というシンプルなプロトコルであり，両者のプログラムを組み合わせて動かす際も大きな問題は起こらなかった．

ディスク制御ロジックとして開発した「IODiskServer」というプログラムは，単独でディスク型デバイスを制御する機能に加えて，GUI 側へ手動でコマンドを送る機能を持たせた．この機能により，ディスク型デバイスが手元に無い状態でも GUI の動きをテストすることができた．さらに，GUI 側である写真・動画ビューアでは，キーボード操作でコマンドをエミュレートできるようにしており，IODiskServer が無い状態，すなわち単独のプログラムとして動作するようになっていた．

7.5.2 なめらカーテン

なめらカーテン [Handa, 2009] とは，カーテンメタファを用いることで日常空間でもプライバシーを守りながら常時利用できる遠隔ビデオチャットシステムである．

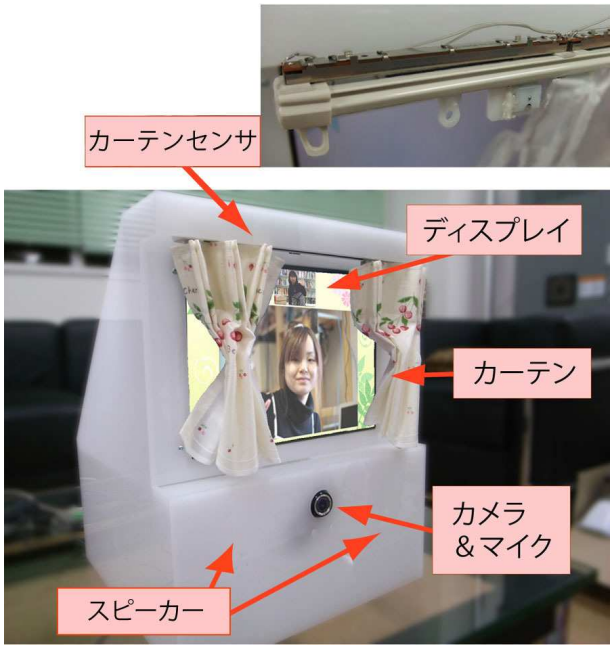

図 7.12: なめらカーテン端末

もし，ビデオチャットの接続を閉じず，遠隔地同士で常につながった状態になれば，離れた部屋にいる人の存在や様子をいつでも知ることができ，より気軽に話しかけることができるなど，遠隔地にいる人同士でのコミュニケーションがしやすくなると考えられる．しかし，家の中のような日常空間ではお互いのプライバシーが問題になる．

そこで，部屋の窓の外から中を覗かれないようにするというカーテンの機能に着目し，ビデオチャットにカーテンのメタファを取り入れることでプライバシーを直感的に制御できるようにした．

図 7.12 のようにビデオチャット専用端末にカーテン状の装置を取り付け，本物のカーテンと同じ感覚で開け閉めできるようにした．相手にこちらの様子を見られたくないときは，カーテンを閉じることで相手側に映る映像をぼかして見えにくくすることができる．

このカーテンの開き具合に応じてぼかし具合が変化し，どれぐらい詳細な様子を相手に伝えるかを簡単に調整できる．カーテンを半開きにしておけば「人が動いている」「話の内容は聞こえないけどにぎやかになった」といった大まかな様子だけを知る/伝えることができ，たとえば，カーテンを完全に閉じた状態であっても「相手側の部屋の明かりがついているか（部屋に人がいるかどうか)」といったことは分かるようになっている．このように詳細を省いた大まかな情報だけを常時やりとりすることで，相手の部屋で何か変化があったときにカーテンを開けて話しかけてみる，といったコミュニケーションのきっかけが生まれやすくなる．

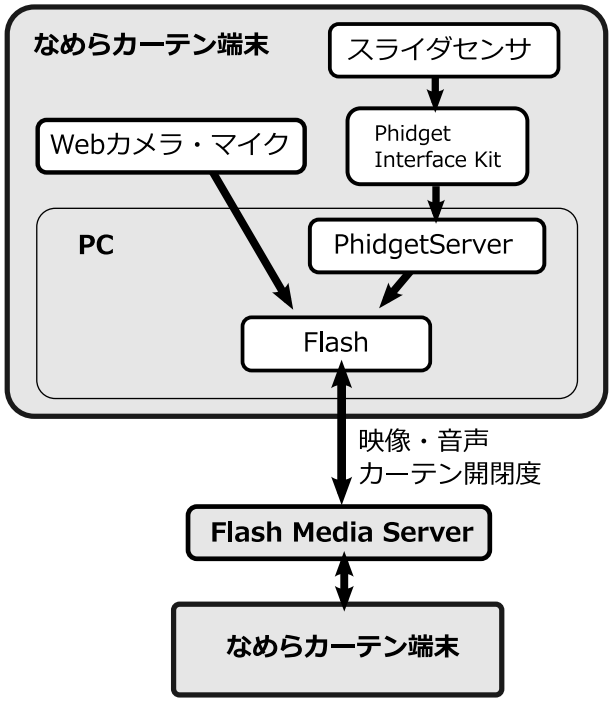

図 7.13: なめらカーテンのシステム構成

システム構成

なめらカーテンは図 7.13 のような構成になっている．

カーテンレールの部分にスライダセンサが取り付けられており，Phidget および PhidgetServer によりカーテンの開き具合を取得する．そして，このカーテンの開き具合およびストリーミング映像・音声を，Flash で書かれたプログラム[9] から，Flash 用のストリーミングサーバの FlashMediaServer 経由で相手に送る．Flash および Flash-MediaServer を利用することで，端末間での情報のやりとりや，映像・音声のストリーミングを比較的容易に実現できた．

開発作業の分割

なめらカーテンは 3 人で開発した．メインの一人がシステム全体の開発に関わり，残りの二人はそれぞれ GUI 側，デバイス側を補助するという体制であった．図 7.13 のうち，GUI にあたる部分が「Flash」，デバイスサーバにあたるのが「PhidgetServer」，そしてデバイスにあたるのが「Web カメラ・マイク」「スライダセンサ」「Phidget Interface Kit」になる．デバイスサーバは，すでに開発済みであった PhidgetServer を単独で用いたため，新たな開発はしていない．

開発の前半では，GUI とデバイスを別々に開発し，それぞれの開発においてデバイ

[9] https://github.com/tsuka-lab/SmoothCurtainAir にてソースコードを公開

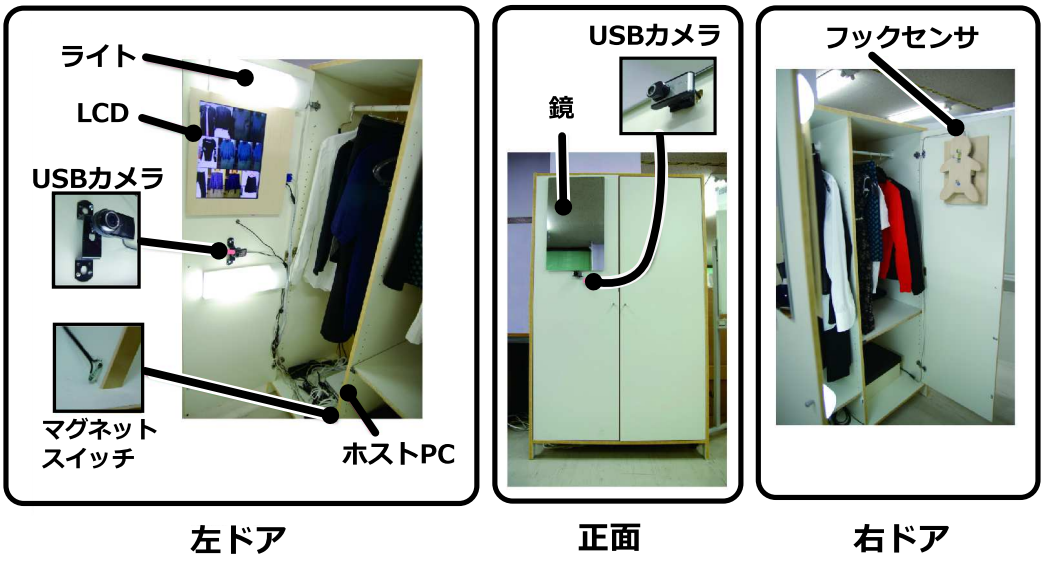

図 7.14: タグタンス

スサーバである PhidgetServer を活用する機会が多かった．例えば GUI 側では，スライダセンサが無い状態でも，PhidgetServer の画面上で値を変えることで仮想的にスライダを動かすことができた．デバイス側では，スライダセンサが正常に動いていることを PhidgetServer の画面上に表示される値で確認することができた．

開発作業の問題点

なめらカーテンの開発では GUI とデバイスサーバでの処理の切り分けが不十分で，デバイスに関連する処理が GUI 側に含まれてしまい，GUI 側の処理が複雑になるという問題があった．例えば，スライダセンサから送られる値の範囲は端末によって少しずつばらつきがあり，これを原因とする分かりにくいバグの発生につながるということがあった．このスライダセンサの問題に対し，GUI 側（Flash）で対処したため，GUI とデバイスがやや密結合な状態になった．さらに，GUI 側では「ストリーミング」「PhidgetServer との通信」「Web カメラ・マイクの制御」など担当する処理が多いことも重なり，コードが複雑になってしまった．

本開発手法を厳格に適用するならば，このようなセンサ値の正規化といった処理はデバイスサーバのデバイス制御ロジックで行うべきであった．このデバイス制御ロジックは，図 7.13 の「Flash」と「PhidgetServer」の間に入ることになる．こうすることで疎結合な状態を保ち，テストをしやすくできると考えられる．

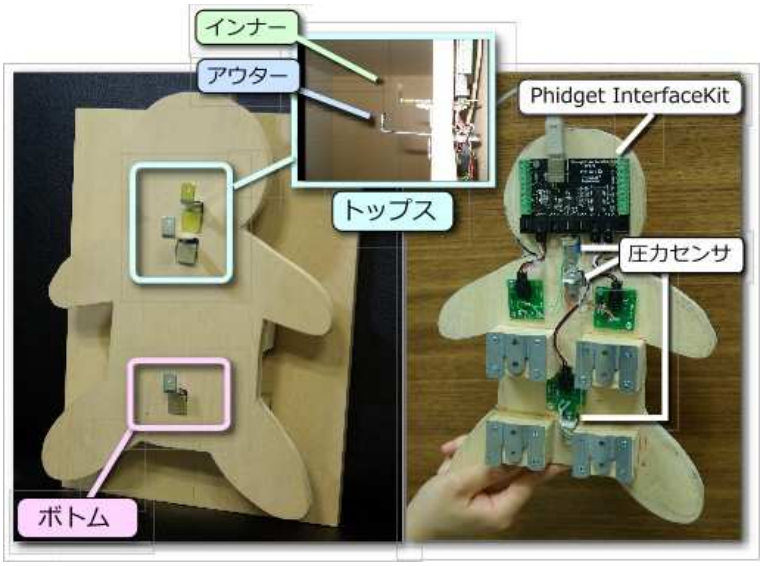

図 7.15: タグタンスのフックセンサ

7.5.3 タグタンス

タグタンスとは家庭で利用される観音開きのタンスを利用し，手軽に服を撮影，分類して，さらに Web 上で管理・共有できるシステムである [Tsukada, 2008]．図 7.14 のように扉の内側にフックが取り付けられており，ここにハンガーを掛けることで反対側の扉に取り付けられたカメラによって服の写真が撮られる．フック部分は図 7.15 のようにアウター・インナー・ボトム用の 3 つのフックがついており，いずれかのフックにかけることで撮影時にタグづけ（分類）される．また，フックには圧力センサが取り付けられており服の重さも同時に記録される．そして，写真や撮影日時，タグ，重さといったデータは写真共有サービスの Flickr にアップロードされる．

Last-Minute Coordinator

タグタンスを利用したアプリケーションの 1 つとして，日々の服選びを支援するシステム「Last-Minute Coordinator」を開発した．Last-Minute Coordinator では「自分の服とその組み合わせを手軽に一覧できる」「これまでに着た服の組み合わせの履歴や，その日に会う人をもとに服を推薦してくれる」さらに「SNS を通じて友達に服の相談ができる」といったことを実現する．

Last-Minute Coordinator は一般的なタッチパネル付き PC 上で利用する．図 7.16 のようにアウター・インナー・ボトムごとに一覧でき，それぞれのリストをスクロールすることで洋服の組み合わせを確かめることができる．

この服の一覧の中から条件によって服を絞り込む．たとえば，カジュアル・フォーマル・セミフォーマルといった条件を選ぶと，過去に選択した組み合わせや服の種類

図 7.16: Last-Minute Coordinator

をもとに，適していない服は薄く，おすすめの服が強調表示される．

さらに,「どちらがいいと思う？」「この組み合わせは変じゃないか？」といったことを他の人に相談してみたいときは，Twitter や Facebook といった SNS を使って聞くことができる．Last-Minute Coordinator から迷っている服の組み合わせをいくつか選んで投稿すると，Web 上に動的に投票用ページが作られ，その URL が SNS に投稿される．そして，SNS から誰かが投票およびコメントすると集計結果が表示され，服選びの参考にすることができる．

システム構成

タグタンスおよび Last-Minute Coordinator は図 7.17，図 7.18 のような構成になっている．

タグタンスのフック（圧力センサ）は Phidget と PhidgetServer で，ライトとドア開閉センサは USB パラレル変換モジュールと ParallellServer でそれぞれ PC から制御される．フックにハンガーが掛かると USB カメラで服が撮影され，その写真がタグ情報とともに Flickr にアップロードされる．撮影した服を一覧する服ビューア[10] （Flash）は，Flickr の Web API を利用して写真を取得する．

Last-Minute Coordinator も服ビューアと同様に Flickr から写真を取得する．また，SNS 用に投票ページおよび集計ページを生成する CGI のサーバがあり，このサーバ経由で SNS に投稿する．

[10]https://github.com/tsuka-lab/TansuViewer にてソースコードを公開

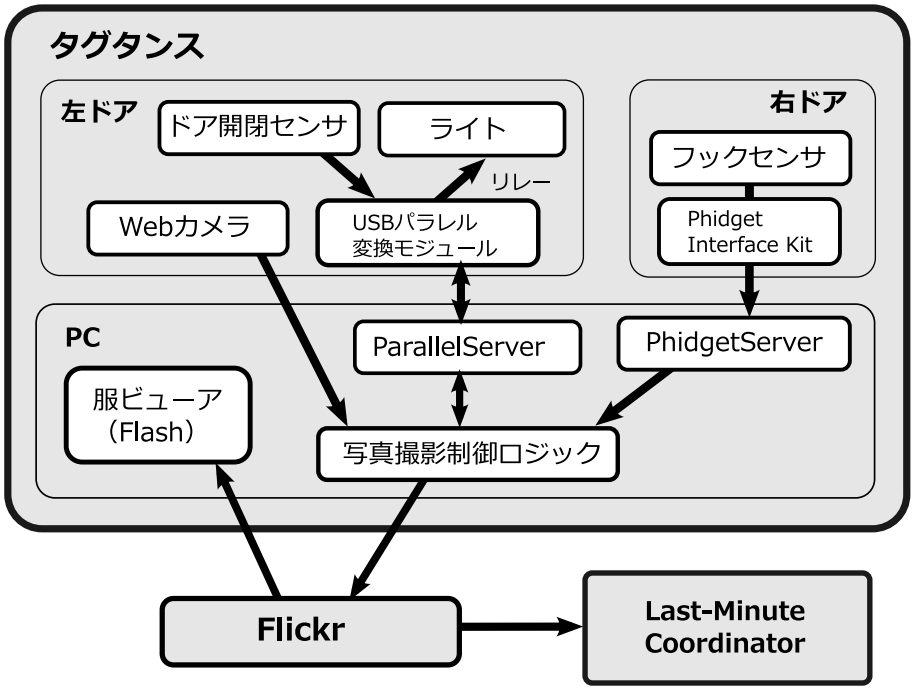

図 7.17: タグタンスのシステム構成

システムの特徴

タグタンスのシステムの特徴は，GUI とデバイスの連携に Web サービス（Flickr）を利用している点である．タグタンスは写真やメタデータを Flickr にアップロード・保存し，服ビューアは Flickr 経由でそのデータを取得している．今回は Flickr を利用したが，実際には他の写真共有サービスでも問題ない．

Flickr のような既存の Web サービスを利用した理由として「API を共通化できる」「Web API 用ライブラリにより開発しやすい」「Web ブラウザからでもアクセスできる」「ネットワーク環境からでも利用しやすい」といったことが挙げられる．タグタンスで取得したデータは複数のアプリケーションから利用されることを想定しており，どのアプリケーションからでも共通した API で扱えることが望ましい．そして Web API (REST や SOAP) のように簡単で一般的な API ほど扱いやすい．特に Flickr のようなメジャーな Web サービスでは各種言語用のライブラリが提供されており，アプリケーションを開発しやすい．また Web サービスであれば，専用アプリケーションに限らず，PC やモバイル端末の Web ブラウザからデータを閲覧・編集できるのも便利な点である．さらに，HTTP 通信はファイヤーウォールの影響を受けにくいため，セキュリティの厳しいネットワーク内でも設置・利用しやすいというメリットもある．

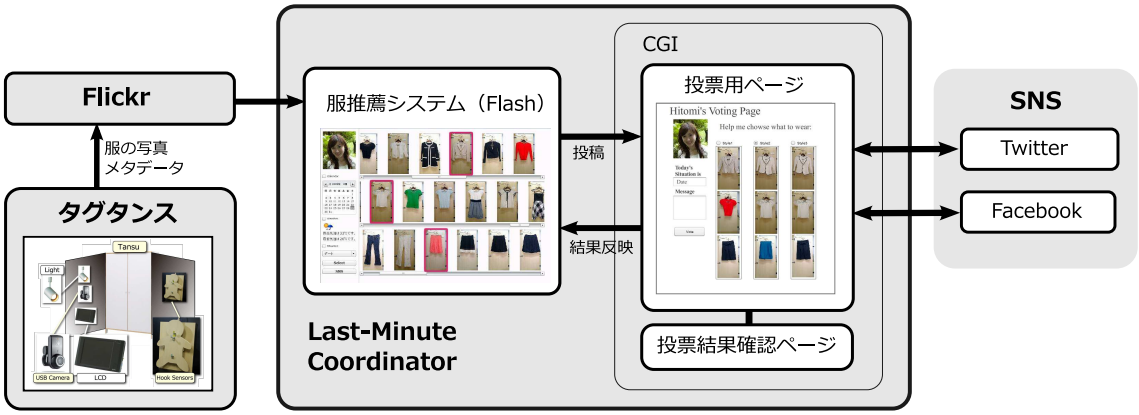

図 7.18: Last-Minute Coordinator のシステム構成

開発作業の分割

図 7.17 のうち，GUI にあたる部分が「服ビューア」および「Last-Minute Coordinator」，デバイスサーバにあたるのが「写真撮影制御ロジック」「ParallelServer」「Phidget-Server」，そしてデバイスにあたるのが左右のドアに含まれるセンサやライト，Phidget Interface Kit などである．

タグタンスおよび Last-Minute Coordinator はシステムの構成要素が比較的多く複雑であったため，開発に時間はかかったものの，それぞれの構成要素がいずれも疎結合な状態であったため，効率的に開発を進めることができた．デバイスサーバの中心となる写真撮影制御ロジックは，GUI である服ビューアや Last-Minute Coordinator と外部の Web サービス（Flickr）を介してデータ（服の写真とメタデータ）をやりとりしている．そのため，デバイスと GUI のプログラムは互いに依存することなく動作し，それぞれほぼ単独で開発作業を進めることができた．実際これらのシステムの開発には 4 人ほど関わっていたが，それぞれ，タグタンスデバイス・服ビューア（Flash）・服推薦システム（Flash）・投票用ページ（CGI）というように分担，独立して作業することが可能であった．

7.6 課題と展望

我々の GUI-デバイス複合開発には，問題が起こりやすいケースやいくつかの課題がある．ここでは「プロトコルが複雑な場合」「複数の開発言語を利用する場合」「限定された動作環境」「実用・商用アプリケーション開発」という 4 つの課題について，それぞれ解決方法や今後の展望について議論する．

7.6.1 プロトコルが複雑な場合

単一の言語で開発する場合と異なり，我々の手法ではGUI-デバイス間で通信をするためのプロトコルを設計する必要がある．そのプロトコルが複雑になると，開発が難しくなるという問題がある．

IODiskやタグタンスは基本的に一方向通信の単純なプロトコルであり，通信フォーマットも単純なカンマ区切りテキストやREST，XMLという一般的なテキストベースであったため，開発も比較的スムーズであった．一方，なめらかーテンはやや複雑な双方向の通信プロトコルであり，情報量も多く，バイナリフォーマットで通信内容を確認しにくい．そのため開発作業が難しいものとなった．中でも通信関連のデバッグ作業が難しくなる．

このような問題への対策としては，まずできる限りシンプルなプロトコル，通信フォーマットを選ぶということである．通信フォーマットはMobiServerで用いている単純なカンマ区切りテキストやREST，JSON，YAMLなどテキストベースの物が望ましい．しかし，要求仕様によってプロトコルが複雑になるのは時に避けられない問題である．なめらかーテンで行っていたように，ログ出力やエラー出力を詳細・丁寧にするといった，できるだけデバッグの手間を軽減する工夫が必要である．

7.6.2 複数の開発言語を利用する場合

我々の開発手法では，GUI側は前述のようにFlashを，デバイスサーバの実装にはC#を利用している．このように2種類の言語を利用するのはメリットとデメリットがある．

メリットは，それぞれのレイヤーに適した言語を使えることである．そして，GUIデザインやハードウェア開発に慣れた人であれば，これまでの知識やスキルをほぼそのまま活かせる．

一方，デメリットとして，1人でGUIとデバイスの両方を開発する場合，2種類の言語や開発環境を使い分ける必要がある．どちらかに詳しくなければ新たに言語やツールを習得する必要がある．また，単一言語であれば必要なかった，両者の連携という手間がかかる．

もちろん，なめらかーテンのようにGUIとデバイス間をMobiServerだけで連携できるような単純なシステムであれば，実際に使用する言語はGUI側の1種類で済む．しかしシステムが複雑なものになるに従って，レイヤーの分離や複数の言語の使い分けが必要になる場合も多い．将来的にミドルウェアが充実し，より多くのデバイスや通信方式に対応できれば，ある程度システムが複雑になっても単一の言語だけで開発できるケースが増えると考えられる．

7.6.3 限定された動作環境

我々の実装するデバイスサーバはWindows上での動作を前提とするため，現在のところデバイスの制御はPCの利用が前提となっている．最近では手のひらサイズの

超小型 PC や，ネットブックのような比較的小型で安価な PC も存在するが，さらなる小型化や省電力化，低コスト化には限界がある．

ユビキタスコンピューティングでは様々な場所や機器にいくつものコンピュータやデバイスを組み込むことが多いため，小型化や省電力化，低コスト化は重要な課題である．現状ではまだ難しいものの，将来的に MobiServer のようなミドルウェアを Android や Embedded Linux，Windows Mobile といったモバイル・組み込み系の OS で動かすことができれば，大幅な小型化や省電力化が期待できる．

7.6.4 実用・商用アプリケーション開発

我々の開発手法は主にプロトタイプの開発および実験環境での利用を想定しており，研究段階の試行錯誤・調整を繰り返すような段階の開発には向いているが，実用段階・商用のアプリケーションを開発にあたっては課題が残る．実用・商用アプリケーションを開発する際は，上述した低コスト化や小型化に加えて，パッケージ化による設置・設定のしやすさを新たに考慮する必要がある．

例えば MobiServer はデバイスの細かなパラメータ調整や動作確認をするための開発者向けの GUI を備えているが，開発者以外のユーザ自身がそれらの設定項目を操作するのは難しく現実的でない．システム内部について詳しくないユーザにとっても設定・操作しやすい GUI が別途必要になる．

また，本手法で開発したシステムは PC やデバイスといったハードウェアや，デバイスサーバ，Flash を用いた GUI などのソフトウェアで構成されるが，これらをユーザ自身で構成し設置するのは手間がかかり難しい．できるだけこれらのシステムを単一のパッケージに収めて簡単に導入できるような工夫が求められる．

7.7 まとめ

本章では，我々がこれまで開発してきたユビキタス環境に溶け込むインタフェースの中でも，開発事例の多い「GUI とデバイスを連携したシステム」の開発手法を紹介した．

GUI-デバイス複合システムの開発では，既存のデスクトップアプリケーション用の GUI デザインツールが使いにくいことや，GUI とデバイスというレイヤーの違いから両者の適切な分離・連携が問題となる．このような問題に対して，我々はこれまでに Flash とデバイスサーバを連携した手法により「IODisk」「タグタンス」「なめらカーテン」といったシステムを開発してきた．

個別のシステムに求められる要件に応じてシステムの構成方法は変わるものの，利用目的や利用形態によっていくつかのパターンに分類でき，それぞれのパターンに適した構成・通信手段がある．また，この開発手法において問題の起こりやすいケースとその対処法，将来の課題などについて検討した．

ユビキタスコンピューティングにおける開発手法はまだ確立しておらず，多くの人がいまだ試行錯誤を繰り返している手探りな段階である．我々を含めたユビキタスコ

ンピューティングの開発に関わる人が，それぞれのノウハウやツールといった開発手法を公開・共有し，改善してゆくことで，汎用的で効果的な開発手法を実現・確立してゆきたい．本稿がそのための一助となり，ユビキタスコンピューティングの発展につながることを期待している．

第 8 章

関連研究

概要

本章では，本研究に関係する研究および技術として，実世界指向インタフェースおよび非アプリケーション指向の環境・アーキテクチャを紹介し，本研究の位置づけについて述べる．

8.1 実世界指向インタフェース

実世界指向インタフェースとは，現実世界の事物を利用して，実世界における人間の活動を支援するインタフェースである [Rekimoto, 1996]．1991 年に発表された Mark Weiser による論文「The Computer for the 21st Century」や，1993 年に発表された「Computer Augmented Environments: Back to the Real World」によって注目を集めるようになった．それ以前の HCI 研究は，CUI や GUI といったスクリーンの中での活動を対象としていたのに対し，実世界指向インタフェースの研究は，スクリーンの外側，すなわち現実世界での人間の活動を対象とした点で，HCI 研究の大きな流れの転換となった．

実世界指向インタフェースの研究領域は広く，これまでに様々な関連コンセプトが提案されている．以下では実世界指向インタフェースと関連の深いコンセプトとして「拡張現実」「情報アプライアンス」「実世界 GUI」「タンジブル・ビット」「Calm Computing」を紹介し，さらに実世界指向インタフェースと溶け込むインタフェースとの関係を明らかにする．

8.1.1 拡張現実

実世界の物や場所にコンピュータ情報を貼り付けて提示する手法を「拡張現実（Augmented Reality）」と呼ぶ．

狭義の拡張現実は，仮想世界の情報を実映像に重ねて表示し，空間的に一致させるというものである．このような拡張現実の例として Feiner らによる KARMA がある [Feiner, 1993]．これはレーザープリンタのメンテンナンスを支援するシステムで，ユーザの位置情報に応じて，スイッチの説明やトレイを動かす方向を透過型 HMD（Head Mounted Display）に表示する．

一方，広義の拡張現実は，コンピュータ情報を光や音などのさまざまな形態で物や場所に関係付けて提示する手法である [Siio, 2002]．このような拡張現実の例として，DigitalDesk[Wellner, 1993] や Augmented Surface[Rekimoto, 1999b] がある．DigitalDesk では机の上の情報をカメラで読み取り，プロジェクタで情報を投影する．こうすることで紙に書いた単語を指さすとその語義を表示したり，ペンで紙に描いた図形を他の場所に表示するといった操作が可能になる．Augmented Surface も同様にプロジェクタとカメラを組み合わせたシステムである．机の上にノート PC やモバイル機器を置くと，机の表面がノート PC の画面を拡張する作業空間として機能するようになり，机の上の他のノート PC やモバイル機器に対してドラッグ＆ドロップで情報を渡すことができる．

拡張現実により，ユビキタス環境に機能や情報を溶け込ませることができる．特に KARMA や DigitalDesk，Augmented Surface のように，PC 環境で行うような複雑な作業をこなす機能を，ユビキタス環境に溶け込ませるのに敵していると考えられる．

しかし，ちょっとしたシンプルな機能や情報を溶け込ませるにはやや大げさであり，日常空間に溶け込ませるには問題が多い．HMD の装着は面倒であるため「平易性」を

損ね，そもそも日常空間でHMDを常に身につけるのは不自然・邪魔であり「親和性」が低い．また，プロジェクタやカメラを設置するシステムは大掛かりになりがちで目立ちやすい．しかも一度システムを設置すると動かしにくいため，机の位置も動かしにくくなるなど，日常空間への「親和性」が低い．

8.1.2 情報アプライアンス

情報アプライアンスとは，Normanの提案するシンプルで単機能な情報機器である[Norman, 1999]．PCのような現在の情報機器は，一つの機器に機能を詰め込みすぎたために複雑で使いにいものになっているとNormanは指摘している．そこで，目的に応じて機能を限定した情報機器とすることで，操作の複雑さを減らし，ユーザに「コンピュータを操作している」と感じさせないようにすることを提案している．また，ネットワークを介して情報アプライアンス同士が連携することで，一つの情報アプライアンスではできない作業を可能にする．

情報アプライアンスは，日常空間にシンプルな機能や情報を溶け込ませるのに適している．例えば，なめらカーテンは一種の情報アプライアンスであり，ビデオチャットによるコミュニケーションという機能だけに特化した情報機器と言える．

一方，情報アプライアンスは，多くの機能を用いる複雑なタスクには向いていない．情報アプライアンス同士で連携できるとはいえ，PCでは1つにまとまっていた機能が別々の機器に分かれることで，操作が複雑になってしまうためである．また，情報アプライアンスという物理的な物が増えすぎると，スペースの限られた空間では邪魔になって馴染みにくいという問題もある．

8.1.3 実世界GUI

増井らは，あらゆる場所においてGUIと同じような操作を可能にする実世界GUIを提案しており[Masui, 2000]，さらに実世界GUIに向いた入力装置としてFieldMouseを提案している[Siio, 1999]．FieldMouseはマウスや加速度センサのような相対移動検出装置とバーコードリーダのようなID検出装置を一体化した装置で，壁や紙の上に貼ったバーコードなどのIDを認識した後，そのFieldMouseを動かす事で各種操作をする．紙にはスライダやボリュームのようなGUI部品が印刷されており，これを利用してスピーカのボリュームなどを制御する．このGUIの印刷された紙をコピーして，それぞれの家電機器の操作に適した場所に貼っておくことで，一種のリモコンとして使うことができる．

FieldMouseとは逆に，実世界の物を動かすことでコンピュータの操作を行う入力デバイスがMouseFieldである[Masui, 2004b]．MouseFieldは，RFIDリーダと動き検出装置を組み合わせた装置で，RFIDを持つ物体を当てて動かすことで操作する．MouseFieldを応用した音楽再生装置のPlayStand++では，RFIDが組み込まれた音楽CDのジャケットをMouseFieldの上に置くことで再生を開始し，上下に動かすことで選曲，回転することで音量を調節する．

FieldMouse と MouseField は，ともにシンプルで安価な装置により汎用的な操作を実現している点が特徴である．また，マウスのような PC 環境に近い操作ができるため，PC 環境をユビキタス環境に溶け込ますのに向いていると考えられる．

8.1.4 タンジブル・ビット

「タンジブル・ビット」は GUI と異なる新しいユーザインタフェースデザインのためのパラダイムを目指す研究アプローチである [Ishii, 2002].「タンジブル」とは「触れて感知できる実体がある」という意味であり，石井らは情報に物理的実体を与え，直接触れて感知・操作できるようにする「タンジブルユーザインタフェース」を提案している．GUI は入力のみがタンジブルで，出力（ディスプレイ装置）はタンジブルではないのに対し，タンジブルユーザインタフェースは，タンジブルな情報表現を用いることで，表現（出力）メディアそのものを直接操作（入力）のメカニズムとして利用できることが特徴となっている．石井らはこのようなタンジブルユーザインタフェースの特徴として「直接操作性」「タンジブルな表現とインタンジブルな表現とのシームレスな融合」「専用インタフェース」「マルチユーザ・マルチハンド・インタラクション」を挙げている．

8.1.5 Calm Technology

ユビキタスコンピューティングを提唱した Mark Weiser は，後に「Calm Technology（穏やかな技術）」という言葉を使うようになった [Weiser, 1996]．現在の情報技術の多くは穏やかではなく，人々を情報攻めにするが，書き慣れたペンや履き心地のよい靴，日曜日の朝の新聞配達は穏やかで快適な技術であるとしている．

Weiser によれば，Calm Technology とは，ものに対する「注意」が「中心」と「外（周辺）」の間を自由に行き来できるようにするものである．車の運転中，人の注意の中心は道路やラジオや乗客にあり，エンジンのノイズには無い．しかし，突然エンジンが変なノイズを発すると，それまで注意の外にあったノイズへと注意の中心が移る．このように，ものを注意の外に置くことで，他のより多くのものに対して注意を払うことができるようになり，外にあったものを中心に持ってくることで，そのものに集中してコントロールできるようになる．

Weiser がユビキタスコンピューティングを提唱した論文「The Computer for the 21st Century」の中で，現在の情報機器は注意の集中を必要とするものであり，コンピュータは周辺や背景の中に消え去って見えなくなるべきだと繰り返し主張している [Weiser, 1991].「見えなくなる」とは，物理的に無くなるということではなく，Calm Technology で言うところの「注意の外に置かれる」ことを意味している．

「情報アプライアンス」や「タンジブル・ビット」といった実世界指向インタフェースに関わる様々なコンセプトにおいても「コンピュータが見えなくなるべき」という同様の主張がなされている．情報アプライアンスでは，コンピュータやテクノロジーが隠されて視界からも意識からも消える「Invisible Computer（見えないコンピュータ）」

を最終目標としており [Norman, 1999]，タンジブル・ビットにおいても，インタフェースを認知的に「透明」にすることを目標としている [Ishii, 2002]．

そして溶け込むインタフェースの目標も Calm Computing に近い．溶け込むインタフェースの究極的な姿は，アプリケーションという形態が分解消滅して，その内包する機能が周囲に溶け込んだ状態であり，そのような状態ではコンピュータを意識せず，快適に情報と接することができる．しかし，単純に意識しなくなれば良いというわけではない．Calm Computing において外から中心へと注意が移るように，普段は意識しない状態でも，必要となったときはすぐに集中して使えることが望ましい．

8.1.6　実世界指向と溶け込むインタフェースの関係

溶け込むインタフェースは，実世界環境だけでなく従来からある PC や Web といった環境を含み，それらを融合するという点が，実世界指向との最も大きな違いになる．

実世界指向インタフェースは「ポスト GUI」とも言われるように，従来の CUI や GUI の抱えていた問題を実世界化によって解決しようとする立場と言える．それに対し，溶け込むインタフェースでは実世界化にもまた問題があると考え，PC 環境や Web 環境，そしてユビキタス環境のそれぞれの良し悪しを認めた上で，それらの違いを越えて自在に組み合わせようとする立場と言える．もちろん，実世界 GUI のように実世界と GUI を組み合わせたインタフェースもあり，両者を厳密に区別することはできない．実世界 GUI は比較的溶け込むインタフェースの考え方に近いとも言える．

8.2　非アプリケーション指向

「2 章 背景」で述べたように，現在利用されている主要なコンピュータ環境では，様々な機能がアプリケーションという固定化された形で利用されている．これらは，言わば「アプリケーション指向」の環境である．

一方で，アプリケーション指向ではない環境やアーキテクチャも存在する．これらの環境では，アプリケーションという概念が希薄であったり，アプリケーションの存在をあまり意識させないようになっている．以下ではそのような「非アプリケーション指向」の環境やアーキテクチャを紹介する．

8.2.1　Squeak

Squeak[Squeak, 2011] は，オブジェクト指向プログラミング言語の Smalltalk 環境であり，一種のデスクトップ OS でもある．Squeak では，OS を含むあらゆる機能が「オブジェクト」として存在しており，それらを自由に組み合わせることで新しいオブジェクトを作り，利用することができる．そのためアプリケーションという概念が希薄である．

Squeak にはテキストエディタやプレゼンツールといった，PC 環境のアプリケーションによく似たツールも存在する．しかし，これらのツールもやはりオブジェクトであ

り,「ツールの機能の一部を抜き出す」「機能を追加する」といったことが自由にできる.

このように機能が固定化された通常のアプリケーションと異なり,「オブジェクト中心」の Squeak は機能の柔軟性が高い.

8.2.2 超漢字

超漢字 [超漢字, 2011] は，ハイパーテキストであらゆる情報を管理する OS である.

超漢字にはファイルやディレクトリは存在せず,「実身」と呼ばれるドキュメント同士を,「仮身」と呼ばれるリンクでつなぐことで全ての情報を管理する．仮身をクリックして実身を開くことでドキュメントを閲覧/編集できる．ドキュメントには文字だけでなく，絵や表を埋め込むこともでき，それらの絵や表もまた実身である.

このように，アプリケーションを用いた操作が無い[1] ため，ドキュメントを「直接的に閲覧/編集」できるのが特徴である.

8.2.3 Live Tiles

スマートフォン OS の Android や iOS ではホーム画面にアプリケーションのアイコンが並ぶ「アプリケーション指向」の UI であるのに対し，Windows Phone 7 では「天気」や「友達の発言」といった情報がタイル状に並ぶ Live Tiles という UI を採用している [Microsoft, 2011].

Live Tiles の各タイルをタップするとアプリケーションが起動するため，完全な「非アプリケーション指向」とは言えないが，アプリケーションを起動することなく,「より直接的に情報を閲覧」できる点が特徴である.

8.2.4 OpenDoc

OpenDoc は Apple 社の開発した複合ドキュメント作成技術である[2]．複合ドキュメントとはテキストに加えてスプレッドシート（表），画像，動画などが埋めこまれたドキュメントであるり，一般的には Word のようなワープロ，Excel のような表計算ソフト，画像処理ソフトを利用して各要素を編集する．一方，OpenDoc ではそれらの編集機能が 1 つに統合されており，アプリケーションを使い分ける必要がない.

8.2.5 SOA

SOA（Service Oriented Architecture）とは，大規模なコンピュータシステムを構築する際の手法の一つである.「決済する」「在庫状況を照会する」といった業務上の一処理に相当するソフトウェアをそれぞれ「サービス」とし，サービスをネットワーク上で連携することでシステムを構築する.

[1] 正確にはアプリケーションも存在するが，それらはあくまで他の OS などと連携するための補助的な機能である.

[2] 現在は利用されていない

プログラムを部品化し，それらを組み合わせて全体のシステムを構成するプログラミング手法としては，オブジェクト指向やコンポーネント指向などが存在する．しかし，オブジェクト指向やコンポーネント指向は処理の単位が小さく，業務の変化にあわせて素早く組み合わせを変更するのは難しい．SOA では，実際の業務処理単位に合わせた，より大きな単位の処理をするソフトウェアで構成することにより，素早く組み合わせを変更できるようにした．

8.3 まとめ

本章では，本研究に関係する研究および技術として，実世界指向インタフェースおよび非アプリケーション指向の環境・アーキテクチャを紹介し，本研究の位置づけについて述べた．

第 9 章

溶け込むインタフェースの考察と展望

概要

本章では，溶け込むインタフェースに関する考察および展望について述べる．

9.1 溶け込むインタフェースの考察

本研究では環境間の不和を解消する「溶け込むインタフェース」を提案し，そのコンセプトを検証するための 3 種類のシステムを提案，構築した．提案コンセプトについて検証するとともに，溶け込ませる上での重要事項や課題について述べる．

9.1.1 提案コンセプトの検証

本研究にて提案，構築した「PC 環境に溶け込む」「Web 環境に溶け込む」「ユビキタス環境に溶け込む」という 3 種類のシステムに基づき，溶け込むインタフェースの特徴である平易性，親和性，流動性について検証する．

平易性

溶け込むインタフェースにおける平易性とは，複数のアプリケーションの組み合わせによる操作の複雑さを解消し，本来やりたいことをシンプルで素早く直接的にできることである．

ソーシャル顔アイコンでは，Web 環境の SNS を PC 環境に溶け込ませることで，Web ブラウザを開くことなく，複数のサイトを横断して素早く人の発言を一覧できるようになった．また，PC 環境らしいドラッグ＆ドロップによって直接的にアイコン化した人の管理ができる．

Willustrator や TwitPaint では，PC 環境のイラストツールを Web 環境であるブログや SNS 溶け込ませることで，Web 上で素早く直接的に絵を編集できるようになった．画像ファイルの管理やアップロードの手間が無くなり，シンプルに絵を管理することができる．

なめらカーテンでは，PC 環境のビデオチャットをユビキタス環境である生活空間に溶け込ませることで，離れた部屋にいる人とすぐに話せるようになった．また，カーテンの開閉というシンプル操作だけで，プライバシー制御が可能になった．

親和性

溶け込むインタフェースにおける親和性とは，異なるアプリケーションや利用環境を組み合わせる時，外観や操作方法を環境に馴染ませて違和感を無くし，直感的に使えることである．

ソーシャル顔アイコンでは，Web 環境の SNS を PC 環境に溶け込ませるために，アイコン型 UI を用いた．コンパクトなアイコンによって面積の限られたデスクトップ上にも並べやすくなり，従来のアイコンと同様の操作方法でデスクトップ上で直感的に使えることができるようになった．

Willustrator や TwitPaint では，PC 環境のイラストツールを Web 環境であるブログや SNS 溶け込ませるために，テキストエリアなどと同様にブラウザ上で絵を編集で

き，様々なWebページと同様にURLの受け渡しによって描いた絵を他の人と共有できるようになった．

なめらカーテンでは，PC環境のビデオチャットをユビキタス環境である生活空間に溶け込ませるために，カーテンのメタファを採り入れたカーテン型デバイスを用いた．カーテンは生活空間に置かれても違和感が無く，外からの視線を遮るというカーテンと同様の概念モデルによって直感的にプライバシーを制御することができた．

流動性

溶け込むインタフェースにおける「流動性」とは，環境間や環境内でよりスムーズに情報が流れるようにすることである．また，一部の環境に限られていた機能を他の環境でも使えるようにすることで，データの流動性を高める．

ソーシャル顔アイコンでは，Web環境のSNSをPC環境に溶け込ませることで，SNSの情報がデスクトップ上に常駐的に表示されるようになった．また，発言が少ない人であってもタイムラインに情報が埋もれにくくなった．

WillustratorやTwitPaintでは，PC環境のイラストツールをWeb環境であるブログやSNS溶け込ませることで，Web上でスムーズに絵を共有できるようになり，絵を用いたコミュニケーションが活性化した．

なめらカーテンでは，PC環境のビデオチャットをユビキタス環境である生活空間に溶け込ませることで，部屋同士のコミュニケーションが活性化した．

9.1.2 溶け込むインタフェースの様々な要素

溶け込むインタフェースの特徴として挙げた，平易性，親和性，流動性は特に重要な要素である．ここではさらに，全てのアプリケーションに適用できるとは限らないが，溶け込ませる上で考慮すべき要素として「アプリケーションの常在」「環境らしさ」「非阻害性」について述べる．

アプリケーションの常在

アプリケーションが環境の中に常時存在することで，そのアプリケーションを必要なとき素早く使えるようになり，平易性を高めることにつながる．また，アプリケーションが常存することで，アプリケーションの起動や終了を意識する必要が無くなる．

アプリケーションを常在させるとは，実際には，ユビキタス環境でアプリケーションを実空間内に常設したり，PC環境でアプリケーションをバックグラウンドで常に動かすということである．なめらカーテンでは，なめらカーテン端末を生活空間内に常設することで，いつでもすぐにビデオチャットできるようにした．ソーシャル顔アイコンでは，顔アイコンを常にデスクトップに置くことで，ブラウザを立ち上げることなく，一目で素早く閲覧できるようにした．

環境らしさ

アプリケーションの利用環境への親和性を高める場合，その環境に特徴的な外観や操作方法，概念といった「その環境らしさ」を考え，アプリケーションの外観や操作に採り入れることが重要になる．

例えば「Web 環境らしさ」として，Web ブラウザやハイパーテキストのページ，ハイパーリンク，ページ遷移，戻る/進む，URL 文字列，各種フォームといった要素が挙げられる．反対に「Web 環境らしくない」ものであると同時に「PC 環境らしい」ものとして，ファイルやディレクトリといった概念や，ドラッグ＆ドロップ，ダブルクリックといった操作が挙げられる．

ユビキタス環境では空間ごとに「その環境らしさ」は異なるが，操作にメタファを採り入れる場合に，そのものらしさを考える必要がある．カーテンのメタファを取り入れたなめらカーテンでは,「カーテンらしさ」として，レース生地，波打った布地といった外観や，中央から左右に開く操作を再現している．

非阻害性

アプリケーションの親和性を高める上で，アプリケーションが目立ちすぎず控えめに振る舞い，人の意識や行動を阻害しないことも重要である．すなわち，情報の提示方法や量を控えめにして過剰に人の注意を引かないようにし，ユーザを長時間拘束しないようにする．控えめな情報提示とは，例えば，小さく表示する/一部だけ表示する/中央ではなく周辺に表示する/必要ないときは隠れる/ユーザから少し離れた場所に置く/音を出さない，といった提示方法である．また，ユーザを長時間拘束しないとは，視覚を占有しない/短時間で使える/姿勢が固定されない，といったことを指す．

9.1.3 溶け込むインタフェースの課題

溶け込むインタフェースの課題や，溶け込むインタフェースを適用しにくい場面について議論する．

開発作業が複雑になるケース

2 種類の利用環境で同時に開発を行う場合，システムが複雑になり，開発作業が難しくなることがある．これは，利用環境ごとに使用する言語やツール，デバッグ手法が異なるためである．

PC 環境でアプリケーションを開発する場合，C++/C#/Objective-C/Java といった言語で，VisualStudio のような統合開発環境を使うのが一般的である．Web 環境でサーバサイドの開発をする場合，PHP/Perl/Ruby といったスクリプト言語に加えて，Ruby on Rails のような Web アプリケーションフレームワークをすることが多い．また，Web 環境のクライアントサイドの開発では，HTML/CSS/JavaScript といった言語を使用する．ユビキタス環境のアプリケーション開発では，使用する言語や開発環境は様々である．スマートフォン向けアプリケーションの開発では，Objective-C/Java/JavaScript

といった言語で，iOS，Android 向けの統合開発環境を利用する．ユビキタス環境のアプリケーションで，デバイスやセンサをマイコンで制御する場合，手軽な統合開発環境として Arduino の利用が増えている．

7 章の GUI-デバイス複合型のアプリケーション開発手法では，異なる 2 種類の UI 開発作業の難しさと，GUI とデバイスの開発作業を分離して，スムーズに開発を進める手法について述べた．

ユビキタス環境の溶け込みにくさ

ユビキタス環境は，親和性を高めにくく，溶け込みにくいことがある．親和性を高めにくいとは，すなわちアプリケーションの外観や操作性を環境に馴染ませるのが難しいということである．

まず，ユビキタス環境の外観，すなわち実空間の見た目は多種多様であるため，一定の手法で馴染ませることが難しい．特に生活空間はユーザの趣味嗜好や文化，気候，季節などに大きく左右されるため，同じアプリケーションでもユーザや利用時間によって外観を変える必要がある．なめらカーテンではディスプレイの背景画像を季節ごとに変化させたように，例えば壁紙を変えることで外観を調整できるようにするするといった対応が考えられる．

ユビキタス環境の操作性も，外観と同様に多種多様であることが問題となる．また，操作の親和性を高める上で，適切なメタファを用いることが重要であるが，常にアプリケーションに適した実世界のメタファがあるとは限らない．無理にメタファを取り入れると，むしろ使いにくくなってしまうこともある．

プライバシー問題

溶け込み度合が高くなると，プライバシーが問題になりやすい．すなわち「サイズが小さい」「外観が周囲によく馴染んでいる」といった溶け込み度合の高いシステムは，ユーザがその存在に気づきにくくなり，知らないうちにプライベートな情報を他人に知られてしまう，ということが起こりやすくなる．

また，実際にはプライバシー上の問題がない場合でも，ユーザに不安を感じさせてしまうことがある．例えば「カメラがあると誰かに監視されているのではないかと思う」「操作ミスや設定ミスでうっかりプライベートな情報が漏れていないか不安」「Web 上に書いた情報が誰に見られるのか分からず不安」といった場合である．

特になめらカーテンのように「人を撮影する」「声を録音する」「一部の人だけに情報を見せる」といった，プライベートな情報を扱う場合は，プライバシー上の問題が起こらないようにした上で，さらに「不安を感じさせないデザイン」が求められる．そして，不安を感じさせないためには「機能の明確さ」「シンプルなプライバシー制御操作」「情報の公開範囲の明示」「公平感」といったことが重要になると考えられる．

機能が明確で理解しやすければ，それだけ操作ミスや設定ミスの不安が減る．「特定の相手としか会話できない」というように，あえて機能を減らすことがプライバシーの保護や安心感につながると考えられる．

操作性とプライバシー制御の両立は難しい．ユーザが手動でプライバシーを制御しようとすると，それだけ操作の複雑さが増してしまうが，コンピュータが自動でプライバシーを制御するとユーザは不安に感じるためである．操作性とプライバシーの適度なバランスを考え，できるだけシンプルな操作でプライバシーを制御できることが望ましい．

また，情報の公開範囲を明示することが,「誰に見られているのか分からない」という不安の軽減につながると考えられる．なめらカーテンのコミュニケーションは1対1であり，相手の様子が常に画面に映っているため公開範囲は明確である．もし，1対多や多対多のコミュニケーションをする場合は，誰と対話しているのか分かるように表示を工夫する必要がある．

そして，情報の送り手と受け手が対等な関係であると感じられるように，ユーザに「公平感」を与えることも安心感につながると考えられる．例えば，なめらカーテンでは，カーテンを閉めると「自分側の様子を隠す」と同時に「相手の様子も見えにくくなる」ように，一方的にのぞき見られないようにすることで公平感を与えている．また，お互いに同じような機器を使い，同じような画面を見るといったことも公平感につながる．

9.2 展望

溶け込むインタフェースをよりいっそう推し進めるための指針や将来展望について述べる．

9.2.1 捨てる発想

溶け込むインタフェースではアプリケーションという形態を分解消滅させることを目指している．そのためこれまでのように「新たに何らかのアプリケーションを作る」とはむしろ真逆の発想が必要となる．

これまでのアプリケーション指向の発想では，何かをもっと良くしようとするとき,「こんな機能を付けたらどうか」「こんな画面にしよう」といったアプリケーションの中身を考えてしまいがちであった．

それに対し，溶け込ませるインタフェースを実現するには，まず「その何かを捨てるにはどうしたら良いか」を考えると良い．すなわち「いつも使っているこのアプリケーションを捨てる（使わずに済むようにする）にはどうしたら良いか」「この Web サイトを捨てる（見ずに済むようにする）にはどうしたら良いか」「この道具を捨てるにはどうしたら良いか」ということをまず最初に考える．

このように，まず「捨てようとする」ことで，必然的にそれが無くなっても困らないようにする代替手段を考えることになる．代替手段を考えようとすると，そもそもそれで何をしたかったのかという「真の目的」や「必要不可欠な機能や情報」を考えざるを得なくなる．また，代替手段として別の物を増やすのではなく,「本当に必要なものだけをどこか（環境）に溶け込ませられないか」を考えると良い．

このようにして,「SNS を捨てる（見なくて済む)」「Web ブラウザを捨てる（起動しなくて済む)」にはどうするかを考えた結果,「よく見る人の情報だけいつでもすぐに見られればよい」となり，ソーシャル顔アイコンが生まれた[1]．また,「デスクトップアプリケーションのドローツールを捨てる」にはどうするかを考えた結果,「ブログに載せる画像が描ければ十分」「他のイラスト同士を組み合わせられれば十分」となり，Willustrator が生まれた．

この「捨てる発想」は発想の飛躍につながる．特に普段からよく使っているものこそ，それを捨てようなどとは思わないものである．あえてそれを捨てようとしてみることで，全く新しい視点で考えられるようになる．究極的には「何もかも捨てるにはどうしたら良いか」を考えると，さらなる発想の飛躍が期待できる．

9.2.2 なんでも API 化

溶け込むインタフェースの課題として「開発作業が複雑になるケース」を挙げたように，今後，自在に機能を組み合わせて溶け込ませていくには，その開発作業をより簡単にしていく必要がある．

そこで重要となるのが「API（Application Programming Interface)」である．API とはアプリケーションのプログラミングをしやすくするためのインタフェースで，具体的にはハードウェアやソフトウェアの機能を第三者でも利用できるようにするライブラリの情報などを意味する．また，Web 上では多くの Web サイトが Web API を公開しており，Web サイト内のデータを第三者が利用できるようにしている．これら複数の Web API を組み合わせて，新しいアプリケーションを作るマッシュアップと呼ばれる開発が盛んに行われている．

API を整備することは，それまで 1 つのアプリケーションにまとまっていた機能や情報を分解することにつながる．溶け込むインタフェースでは，アプリケーションを分解し，その中の機能を自在に組み合わせられる状態を目指すため，API を整備することは理想的な状態に近づける一つの手法である．Web API によるマッシュアップが盛んに行われているのは，様々な Web サイトや Web アプリケーションが分解し，それぞれが溶け込みやすくなった結果と言える．

PC 環境や Web 環境にはすでに様々な API が提供されているが，ユビキタス環境における API についてはほとんど整備されていない．モバイル機器で記録した情報や，様々なセンサなどの情報，情報機器のデータなど，あらゆる情報を API を通じて簡単に利用できるようにすることができれば，開発作業が簡単になると考えられる．このような「なんでも API 化」が実現できれば，Web 上でマッシュアップが盛んに行われているように，ユビキタス環境における開発も盛んになると期待できる．

[1]ただしまだ完全に捨て切れてはいない．

9.2.3 究極の溶け込んだ世界

溶け込むインタフェースの提案では，その究極的な姿を,「アプリケーションという形態が分解消滅し，その内包する多様な機能が，現実/仮想に関係なく自由に組み合わりながら周囲に溶け込み，利用できるような状態」と述べた．本研究はそのような状態への一歩に過ぎず，究極的な状態には程遠い．

現在の技術だけでは不可能なこともあり，今後どのように技術が進化するかを予測することは極めて難しいが，溶け込むインタフェースの究極的な状態が完全に実現したとき，一体どのような世界になるかを予測してみたい．

まず，溶け込むインタフェースの指針として「何もかも捨てるにはどうしたら良いかを考える」という発想を挙げたが，本当に何もかも捨てると極めてシンプルな世界になるであろう．また，この指針はコンピュータ以外の様々なものに対しても適用できると考えており，単にものが無くなるというだけではなく，世の中の様々な複雑さを解消できると思われる．

次に，溶け込むインタフェースの特徴である平易性，親和性，流動性が究極的に高まった状態を考えてみる．平易性が極めて高いため，いつでもやりたいと思ったことを即，ストレートに実行できる．そして，親和性が極めて高いため，お互いの環境が完全に調和し，その区別はつかなくなる．すなわち PC や Web やユビキタスといった環境の違いは徐々に薄まり，最終的に一体化する．さらに，流動性が極めて高いため，情報がどこにあるかを意識せず扱える．非常に漠然としているが，環境や情報が渾然一体となった自由な世界である．

本当に何もかも溶け込み一体になるとどうなるのか．アニメ 新世紀エヴァンゲリオンでは，まさにそのような状態を実現しようとする「人類補完計画」とその計画を巡る物語が描かれていた．人類補完計画は，全ての生物を液体状に溶かすことで「人々の個体生命の形を消し去り，人類を完全な単体生物へ人工進化させる」計画であり，計画が実現した世界は人々の不完全な心の隙間を埋める「理想の世界」とされた[2]．人すらも溶け込んでしまうことで，人の思考や心を含むありとあらゆる情報が共有された世界と言える．すでに現在でも，Web によって世界中の多く人による情報共有されるようになった結果,「集合知」と呼ばれる，群衆が一つの知性を生み出すような状況が見られるようになった．人の思考や心まで完全に共有することは極めて難しいが，Twitter 上ではふと思ったことや感じたことなど，ちょっとした心理状況も書かれ，瞬間的に共有されるようになっている．すでに人類補完計画の「理想の世界」へと少しずつ近付きつつあるのかもしれない．

現在のコンピュータの基礎を作った人物もこれに似通った未来予測をしている．J. C. R. Licklider は論文「Man-Computer Symbiosis」の中で,「人間の脳と計算機械を密に結合して相互依存しつつ共生させることで互いの力を補足しあうことができる」と述べており [Licklider, 1960]，人とコンピュータすら溶け込むことでより高いレベルの知的活動に集中できるようになるとしている．また，Douglas Engelbart は論文「Augmenting Human Intellect: A Conceptual Framework」の中でネットワーク強化知性というコン

[2]Wikipedia「新世紀エヴァンゲリオン」より

セプトを提案しており,「人間の能力を増強する」「(外交，経営，社会科学，生命科学，自然科学，法律，設計などの）複雑すぎた状況に対するより高速な理解，より良い理解，より深い理解，またかつては解決不能と思われた問題への解決策をより早く，より良く見つけ出す」と述べている [Engelbart, 1962]．人とコンピュータすら溶け込ませ一体化することが，人の能力を発揮し，世界の複雑さを解決する手がかりと言えるだろう．

9.3 まとめ

本章では，溶け込むインタフェースに関する考察と展望について述べた．まず，本研究で提案した「溶け込むインタフェース」のコンセプトを，構築したシステムに基づき検証した．また，溶け込むインタフェースにとって重要な様々な要素や，課題について考察した．さらに，溶け込むインタフェースを推し進めるための指針や，将来の展望について議論した．

第 10 章

結論

概要

本章では本研究の成果についてまとめ，本論文を総括する．

10.1 本研究の成果

本研究では，環境間の不和という問題を解決するための「溶け込むインタフェース」を提案し，そのコンセプトに基づいたユーザインタフェースを提案，構築，評価した．

本研究の成果は以下のようにまとめられる．

1. 溶け込むインタフェースの提案
   PC，Web，ユビキタスというそれぞれ特性の異なる環境下のアプリケーションを，他の環境と組み合わせた際に生じる問題点を明らかにした上で，その問題を解決し，コンピュータの活用範囲をより一層広げるアプローチとして「溶け込むインタフェース」を提案した．溶け込むインタフェースとは，PC，Web，ユビキタスといった環境の違いを越えて，アプリケーションを他の環境に馴染ませるインタフェースデザインである．環境に馴染ませるための様々なインタフェースデザインの工夫により，素早く直接的に操作できる「平易性」や，外観や操作方法が環境に馴染んで違和感が無い「親和性」，環境間でスムーズに情報が流れる「流動性」を高める．

2. PC 環境，Web 環境，ユビキタス環境にそれぞれ溶け込むシステムの提案，構築，運用
   溶け込むインタフェースの有効性を検証するため，PC 環境，Web 環境，ユビキタス環境において，それぞれ溶け込むインタフェースを提案，構築した．

   PC 環境に溶け込むインタフェースとして，Web アプリケーションであるソーシャルウェブサイトを PC 環境のデスクトップ上に溶け込ませる「ソーシャル顔アイコン」を提案した．ソーシャルウェブサイトの中にいる人をアイコンとしてデスクトップ上に溶け込ませることで人に対してより直接的にアクセスできるインタフェースを備えており，使い慣れたアイコンと同じように操作できる点や，発言の新鮮度や発言頻度といった時間軸情報を視覚的に分かりやすく表示する．

   Web 環境に溶け込むインタフェースとして，Web 環境のブログや SNS に，PC 環境のイラストツールを溶け込ませる「Willustrator」と「TwitPaint」を提案した．Web 上にイラストツールを溶け込ませることで，Web ブラウザ上で絵を編集でき，他の人の描いた絵から「派生」させて新しい絵を作成することのできる．ドロー系である Willustrator は絵を再編集/再利用しやすいという特徴を持つ．一方，ペイント系の TwitPaint は Twitter と連携した絵によるコミュニケーションを特徴とする．両ツールを長期運用し，得られた派生データや事例，知見などを元に，相違や特徴，問題点を分析した．

   ユビキタス環境に溶け込むインタフェースとして，生活空間に PC 環境のビデオチャットを溶け込ませる「なめらカーテン」を提案した．なめらカーテンを用いて，生活空間にビデオチャットを溶け込ませることで，プライバシーを保護しながら，いつでも手軽に遠隔地の相手と会話できる．なめらカーテンでは，カーテンのメタファを採り入れたカーテン型デバイスによって，直感的にプライバシー

レベルを制御し，コミュニケーション形態の柔軟な移行を可能にする．さらに，なめらカーテンを用いた 14ヶ月以上の期間実証実験により有効性を検証した．

3. 提案コンセプトの考察
   2 で提案，構築したシステムを運用して得られたデータや知見を元に，本研究で提案した溶け込むインタフェースについて考察し，有効性や課題を検証した．さらに，溶け込むインタフェースに関する議論を行うとともに，その設計指針や将来展望を示した．

10.2 本論文の総括と結論

PC，Web，ユビキタスコンピューティングといったコンピュータ技術が普及する過程で，その多くの機能は「アプリケーション」という形でユーザに提供され，利用されるようになった．

現在の主要なアプリケーションには,「デスクトップアプリケーション」「Web アプリケーション」「ユビキタスアプリケーション」があり，それぞれ利用環境や動作環境が異なっている．それぞれの環境が持つ利点をうまく組み合わせることで，単一の環境ではできなかった新しいコンピュータの活用方法やコミュニケーションが生まれ，コンピュータの活用範囲がより一層広まると考えられる．

しかし，現在のアプリケーションの多くは，他の環境と組み合わせて使おうとすると様々な不和が生じてしまう．この環境間の不和には，やりたいことを直接的にできない「操作の複雑化」や，操作方法や外観が環境に馴染まない「環境との非調和」，環境間で情報をやり取りできない「情報と非流動」といったものがある．

この環境間の不和を解決するため，本研究では「溶け込むインタフェース」を提案した．溶け込むインタフェースとは，PC，Web，ユビキタスといった環境の違いを越えて，アプリケーションを他の環境に馴染ませるインタフェースデザイン手法である．環境に馴染ませるための様々なインタフェースデザインの工夫により，素早く直接的に操作できる「平易性」や，外観や操作方法が環境に馴染んで違和感が無い「親和性」，環境間でスムーズに情報が流れる「流動性」を高める．そして，様々なアプリケーションが自然な形で互いに混ざり合うことで，PC，Web，ユビキタスがあたかも融合した状態を目指す．

本研究では，主要なアプリケーション環境である「PC 環境」「Web 環境」「ユビキタス環境」において，それぞれ溶け込むインタフェースを提案した．

Web 環境に溶け込むインタフェースとして，PC 環境のアプリケーションであるイラストツールを，Web 環境のブログや SNS といったアプリケーションに溶け込ませる「Willustrator」「TwitPaint」という 2 つのシステムを提案した．Willustrator と TwitPaint は，それぞれ Web ブラウザ上で絵を編集でき，他の人の描いた絵から「派生」させて新しい絵を作成することのできる Web アプリケーションである．ドローツールである Willustrator は絵を絵を再編集/再利用しやすいという特徴を持つ．一方，ペイント系の TwitPaint は Twitter と連携した絵によるコミュニケーションを特徴とす

る．両ツールを長期運用し，得られた派生データや事例，知見などを元に，相違や特徴，問題点を分析した．

PC 環境に溶け込むインタフェースとして，Web 環境のアプリケーションである SNS を，PC 環境に溶け込ませる「ソーシャル顔アイコン」を提案した．ソーシャル顔アイコンは，複数のソーシャルウェブサイトを横断しながら，優先的に見たい人がタイムラインに埋もれてしまうことなく，特定の人物を一目で見つけることのできるインタフェースである．ソーシャルウェブサイトの中にいる人をアイコンとしてデスクトップに置くことで人に対してより直接的にアクセスできるインタフェースを備えており，使い慣れたアイコンと同じように操作できる点や，発言の新鮮度や発言頻度といった時間軸情報を視覚的に分かりやすく表示する点が特徴である．

ユビキタス環境に溶け込むインタフェースとして，PC 環境のアプリケーションであるビデオチャットを生活空間に溶け込ませる「なめらカーテン」を提案した．なめらカーテンは，カーテンメタファを用いたコミュニケーションシステムである．カーテンメタファを用いることで「直接的」「アンビエント」という 2 種類コミュニケーション形態を直感的な操作でなめらかに移行できる．なめらカーテンを用いた 14ヶ月以上の期間実証実験を行ない,「日常空間での常時利用」「プライバシーの制御」「操作の分かりやすさ」「日常空間に馴染むデザイン」といった観点から有効性を確認した．

そして，溶け込むインタフェースに関連する研究を紹介し，本研究の位置づけについて整理した．最後に，溶け込むインタフェースの特徴や課題を明らかにし，今後の展望を示した．

謝辞

学部生の頃から 10 年以上の長きに渡りご指導を頂いた，慶應義塾大学 安村通晃教授に深く感謝致します．また，本研究の副査として有益なご意見，ご助言を頂いた慶應義塾大学 増井俊之教授および小川克彦教授，お茶の水女子大学 椎尾一郎教授に感謝致します．

研究を進めるにあたり，慶応義塾大学およびお茶の水女子大学の先生方，学生の皆様，また Nota Inc. など様々な組織の方々にお世話になりました．学生時代は研究室の先輩として，そして現在は上司として私の研究全般に多大なご協力を頂き，面倒を見ていただきました，お茶の水女子大学 塚田浩二特任助教に深く感謝致します．「Web 上で編集/派生可能なイラストツールの研究」など，Web 関係の様々なプロジェクトを共に進め，なにより同世代の研究者・エンジニア・友人として強い刺激を与えてくれた Nota Inc. の永田周一氏に感謝致します．

お茶の水女子大学では椎尾研究室の皆様と研究活動を共にさせて頂き，楽しく充実した研究生活を送ることができました．なめらカーテンの共同研究者であり，真面目で熱心に研究に取り組んでくれた同研究室の半田智子氏に感謝致します．私が同大学に赴任して以来，毎日のように側で研究を行い，日々の研究生活を助けて頂きました同研究室博士課程の中川真紀氏，沖真帆氏，川上あゆみ氏に感謝致します．

塚田研究室の同僚で，同じ分野の研究者として議論を重ね，数々の助言を頂きました総合研究大学院大学 後藤孝行氏，安村研究室後輩の松田聖大氏に感謝致します．また，研究活動をバックアップして頂きました，お茶の水女子大学研究協力チームの皆様，お茶大アカデミックプロダクションの皆様に感謝致します．

最後に，これまで私を支えてくださった両親に感謝致します．

2012 年 2 月 20 日
神原 啓介

本研究に関する発表

論文誌

1. 神原啓介, 半田智子, 塚田浩二, 椎尾一郎: 日常空間で常時利用するためのカーテンメタファを用いたビデオコミュニケーションシステム, ヒューマンインタフェース学会論文誌, Vol. 13, No. 4, pp. 291–302 Nov. 2011.

2. 神原啓介, 塚田浩二: ユビキタスコンピューティングにおけるGUI-デバイス複合型のアプリケーション開発手法, 日本ソフトウェア科学会論文誌「コンピュータソフトウェア」, Vol. 28, No. 4, pp. 206–221, Nov. 2011.

3. 神原啓介, 塚田浩二: ソーシャル顔アイコン, 日本ソフトウェア科学会論文誌「コンピュータソフトウェア」, Vol.28, No.2, pp. 172–182, May. 2011.

4. 神原啓介, 永田周一, 塚田浩二: Web 上で編集/派生可能なイラストツールの研究, 情報処理学会論文誌, Vol.52, No.4, pp. 1621–1634, Apr. 2011.

国際会議

1. Keisuke Kambara, and Koji Tsukada: DrawerBrowser: Practical Picture Browser for Finding Items in Drawers, Adjunct Proceedings of UbiComp 2011, pp. 601–602, Sep. 2011.

2. Keisuke Kambara, and Koji Tsukada: Onomatopen: Painting Using Onomatopoeia, *Proceedings of 9th International Conference on Entertainment Computing (ICEC 2010)*, pp. 43–54, Sep. 2010.

その他

1. 神原啓介, 塚田浩二: オノマトペを用いたマルチモーダルインタラクション, 2011年度（第25回）人工知能学会全国大会（JSAI2011）論文集, 演題番号1C2-OS4b-12, Jun. 2011.

2. 神原啓介, 塚田浩二: オノマトペン日本ソフトウェア科学会論文誌「コンピュータソフトウェア」, Vol. 27, No. 1, pp. 48–55, Feb. 2010.

3. 神原啓介, 塚田浩二: ソーシャル顔アイコン, 日本ソフトウェア科学会 WISS2009 論文集, pp. 53–56, Dec. 2009.

4. 半田智子, 神原啓介, 塚田浩二, 椎尾 一郎: なめらカーテン, ソフトウェア科学会 WISS2009 論文集, pp. 179–180, Dec. 2009.

5. Tomoko Handa, Keisuke Kambara, Koji Tsukada, and Itiro Siio: SmoothCurtain: privacy controlling video communication device, *Adjunct Proceedings of UbiComp 2009*, pp. 186–187, Oct. 2009.

6. 半田智子, 神原啓介, 塚田浩二, 椎尾一郎: なめらカーテン, 情報処理学会 ヒューマンインタフェースシンポジウム 2009 論文集, pp. 117–120, Sep. 2009.

7. 神原啓介, 塚田浩二: オノマトペン, 日本ソフトウェア科学会 WISS2008 論文集, pp. 79–84, Nov. 2008.

8. 神原啓介, 安村通晃: Willustrator: Web 上での協同イラストレーション, ソフトウェア科学会 WISS2005 論文集, pp. 139–140, Dec. 2005.

参考文献

[Adobe, 2011] Adobe (2011). “Photoshop Express”. http://www.photoshop.com/.

[Anzai, 2000] 安斎利洋・中村理恵子 (2000). 「連画コラボレーションを支援するパノラマ空間ペイントシステム — The Wall」. *IPSJ SIG Notes 2000(13)*, pp. 57–64.

[Apple, 2011] Apple (2011). “Safari Web Clip”. http://www.apple.com/safari/features.html.

[Assogba, 2010] Assogba, Y. and Donath, J. (2010). “Share: A Programming Environment for Loosely Bound Cooperation”. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems*, pp. 961–970, Atlanta, Georgia. ACM.

[Aviary, 2011a] Aviary (2011a). “Phoenix”. http://www.photoshop.com/.

[Aviary, 2011b] Aviary (2011b). “Raven”. http://aviary.com/tools/vector-editor.

[Boyle, 2000] Boyle, M., Edwards, C., and Greenberg, S. (2000). “The Effects of Filtered Video on Awareness and Privacy”. In *Proceedings of CSCW 2000*, pp. 1–10.

[Buxton, 1995] Buxton, W. A. (1995). “Living in Augmented Reality: Ubiquitous Media and Reactive Environments”. In *Proceedings of Imagina*, pp. 215–229.

[Cacoo, 2011] Cacoo (2011). http://cacoo.com/.

[Creately, 2011] Creately (2011). http://creately.com/.

[drawTwit, 2011] drawTwit (2011). http://drawtwit.com/.

[Engelbart, 1962] Engelbart, D. C. (1962). “Augmenting Human Intellect: A Conceptual Framework”. *SRI Summary Report AFOSR-3223*.

[Feiner, 1993] Feiner, S., Macintyre, B., and Seligmann, D. (1993). “Knowledge-based augmented reality”. *Communications of the ACM*, **36**(7) pp.53–62.

[Gliffy, 2011] Gliffy (2011). http://www.gliffy.com/.

[Greenberg, 2002] Greenberg, S. and Boyle, M. (2002). “Customizable Physical Interfaces for Interacting with Conventional Applications”. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST2002)*, pp. 31–40.

[Greenberg, 1992a] Greenberg, S., Roseman, M., and Webster, D. (1992a). “GroupSketch”. In *ACM SIGGRAPH Video Review: Special Edition of the CSCW ’92 Technical Video Program, 87*. ACM.

[Greenberg, 1992b] Greenberg, S., Roseman, M., Webster, D., and Bohnet, R. (1992b). “Issues and experiences designing and implementing two group drawing tools”. In *Proceedings of Hawaii International Conference on System Sciences*, volume 4, pp. 138–150, Kuwaii, Hawaii. IEEE.

[Handa, 2009] Handa, T., Kambara, K., Tsukada, K., and Siio, I. (2009). “SmoothCurtain: privacy controlling video communication device”. In *Adjunct Proceedings of Ubicomp2009*, pp. 186–187.

[Hashimoto, 2007] 橋本翔・安田俊平・小泉麻理子 (2007).「影電話- Teleshadow plus」. http://www.ipa.go.jp/about/jigyoseika/07fy-pro/youth/2007-0362d.pdf.

[Ishii, 2002] 石井裕 (2002).「タンジブル・ビット:情報と物理世界を融合する，新しいユーザ・インタフェース・デザイン」. 情報処理, **43**(3) pp.222–229.

[Kadomura, 2011] 門村亜珠・塚田浩二・馬場哲晃・串山久美子 (2011).「ハングル ガングル: ハングル文字学習のためのインタラクティブ玩具」.『情報処理学会 インタラクション 2011 論文集』, pp. 789–793.

[Kahan, 2001] Kahan, J. and Koivunen, M.-R. (2001). “Annotea: An Open RDF Infrastructure for Shared Web Annotations”. In *Proceedings of the 10th international conference on World Wide Web*, pp. 623–632, Hong Kong. ACM.

[Kobayashi, 2006] Kobayashi, S., Endo, T., Harada, K., and Oishi, S. (2006). “GAINER: a reconfigurable I/O module and software libraries for education”. In *Proceedings of the 2006 Conference on New Interfaces for Musical Expression (NIME 2006)*, pp. 346–351.

[Kodama, 2005] 児玉哲彦・渡邊恵太・安村通晃 (2005).「遮蔽行為による情報機器のプライバシー制御」.『ヒューマンインタフェースシンポジウム 2005 論文集』, pp. 541–546.

[Komatsuzaki, 2011] 小松崎瑞穂・塚田浩二・椎尾一郎 (2011).「DrawerFinder: 収納箱に適した物探し支援システムの提案と運用」.『情報処理学会第 73 回全国大会講演論文集』, pp. 603–604.

[Kotani, 2011] 小谷尚子・塚田浩二・渡邊恵太・椎尾一郎 (2011). 「お弁当箱を介したコミュニケーション支援システム」. 『情報処理学会 インタラクション 2011 論文集』, pp. 239–242.

[Kuzuoka, 2000] Kuzuoka, H. and Greenberg, S. (2000). "Using Digital but Physical Surrogates to Mediate Awareness, Communication and Privacy in Media Spaces". *Personal Technologies*, **3**(4) pp.182–198.

[Licklider, 1960] Licklider, J. C. R. (1960). "Man-Computer Symbiosis". *IRE Transactions on Human Factors in Electronics*, **HFE-1** pp.4–11.

[Mander, 1992] Mander, R., Salomon, G., and Wong, Y. Y. (1992). "A 'Pile' Metaphor for Supporting Casual Organization of Information". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 627–634.

[Masui, 2004a] 増井俊之 (2004a). 『インターフェイスの街角 (74) - PC でセンサを活用する』.

[Masui, 2000] Masui, T. and Siio, I. (2000). "Real-World Graphical User Interfaces". In *Proceedings of the Second International Symposium on Handheld and Ubiquitous Computing (HUC 2000)*, pp. 72–84. Springer Publications.

[Masui, 2004b] Masui, T., Tsukada, K., and Siio, I. (2004b). "MouseField: A Simple and Versatile Input Device for Ubiquitous Computing". In *Proceedings of UbiComp2004*, pp. 319–328. Springer LNCS3205.

[Mellis, 2007] Mellis, D., Banzi, M., Cuartielles, D., and Igoe, T. (2007). "Arduino: An Open Electronics Prototyping Platform". In *alt.chi section of the CHI 2007*.

[Microsoft, 2011] Microsoft (2011). "Windows Phone 7". http://www.microsoft.com/windowsphone/en-gb/features/default.aspx.

[Miyajima, 2003] 宮島麻美・伊藤良浩・伊東昌子・渡邊琢美 (2003). 「つながり感通信：人間関係の維持・構築を目的としたコミュニケーション環境の設計と家族成員間における検証」. ヒューマンインタフェース学会論文誌, **5**(2) pp.171–180.

[Morikawa, 1998] Morikawa, O. and Maesako, T. (1998). "HyperMirror: Toward Pleasant-to-use Video Mediated Communication System". In *Proceedings of CSCW9'8*, pp. 149–158.

[Mozilla, 2011] Mozilla (2011). "Prism". http://prism.mozilla.com/.

[Nagata, 2007a] 永田周一 (2007a). 『NOTA：ユーザー主導型コミュニティ活動支援システムの研究, SFC-MT2006-004』. 慶應義塾大学湘南藤沢学会.

[Nagata, 2007b] 永田周一・安村通晃 (2007b).「Enzin：情報の公開範囲を手軽に変更できるコミュニケーションツール」. 情報処理学会論文誌, **48**(3) pp.1134–1143.

[Neustaedter, 2003] Neustaedter, C. and Greenberg, S. (2003). "The design of a context-aware home media space for balancing privacy and awareness". In *Proceedings of Ubicomp 2003*, pp. 297–314.

[Norman, 1999] Norman, D. A. (1999). *The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution*. The MIT Press.

[Ogasawara, 2007] 小笠原遼子・山木妙子・塚田浩二・渡邊恵太・椎尾一郎 (2007).「インタラクティブな掃除機」.『エンタテインメントコンピューティング 2007 講演論文集』, pp. 71–74.

[Picnic, 2011] Picnic (2011). http://www.picnik.com/.

[Rekimoto, 1999a] Rekimoto, J. (1999a). "Time-Machine Computing: A Time-centric Approach for the Information Environment". In *ACM UIST'99*, pp. 45–54.

[Rekimoto, 1999b] Rekimoto, J. and Saitoh, M. (1999b). "Augmented surfaces: a spatially continuous work space for hybrid computing environments". In *Proceedings of the SIGCHI conference on Human factors in computing systems(CHI 99)*, pp. 378–385. ACM Press.

[Rekimoto, 1996] 暦本純一 (1996).「実世界指向インタフェースの研究動向」. コンピュータソフトウェア, **13**(3) pp.4–18.

[Reportage, 2011] Reportage (2011). http://itunes.apple.com/jp/app/reportage/.

[Root, 1988] Root, R. W. (1988). "Design of a multi-media vehicle for social browsing". In *Proceedings of the 1988 ACM Conference on Computer - Supported Cooperative Work*, pp. 25–38.

[Siio, 1999] Siio, I., Masui, T., and Fukuchi, K. (1999). "Real-world Interaction using the FieldMouse". In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST 99)*, pp. 113–119. ACM Press.

[Siio, 2002] Siio, I., Rowan, J., and Mynatt, E. (2002). "Peek-a-drawer: communication by furniture". In *Extended abstracts on Human factors in computing systems(CHI 2002)*, pp. 582–583. ACM Press.

[Squeak, 2011] Squeak (2011). http://squeak.org/.

[SumoPaint, 2011] SumoPaint (2011). http://www.sumopaint.com/app/.

[Takabayashi, 2003] 高林哲・塚田浩二・増井俊之 (2003). 「顔アイコン: 手軽なファイル転送システム」. 『インタラクション 2003 論文集』, pp. 33–34. 情報処理学会.

[Tei, 2008] 鄭顕志・中川博之・川俣洋次郎・吉岡信和・深澤良彰・本位田真一 (2008). 「ユビキタスコンピューティングにおけるアプリケーション開発手法に関する研究動向」. 日本ソフトウェア科学会論文誌（コンピュータソフトウェア）, **25**(4) pp.121–132.

[Tsujita, 2010] Tsujita, H., Tsukada, K., Kambara, K., and Siio, I. (2010). "Complete Fashion Coordinator: A support system for capturing and selecting daily clothes with social network". In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI2010)*, pp. 127–132.

[Tsujita, 2009] 辻田眸・塚田浩二・椎尾一郎 (2009). 「遠距離恋愛者間のコミュニケーションを支援する日用品”SyncDecor”の提案」. 日本ソフトウェア科学会論文誌 (コンピュータソフトウェア), **26**(1) pp.25–37.

[Tsukada, 2010a] 塚田浩二 (2010a). 「日曜ユビキタスのための手軽なミドルウェア」. 日本ソフトウェア科学会論文誌 (コンピュータソフトウェア), **27**(1) pp.3–17.

[Tsukada, 2010b] Tsukada, K. and Kambara, K. (2010b). "IODisk: Disk-type I/O interface for browsing digital contents". In *Extended Abstracts of UIST2010*, pp. 403–404.

[Tsukada, 2002] 塚田浩二・高林哲・増井俊之 (2002). 「廃れるリンク」. 情報処理学会論文誌, **43**(12) pp.3718–3721.

[Tsukada, 2008] Tsukada, K., Tsujita, H., and Siio, I. (2008). "TagTansu:A Wardrobe to Support Creating a Picture Database of Clothes". In *Adjunct Proceedings of Pervasive2008*, pp. 49–52.

[TwitDraw, 2011] TwitDraw (2011). http://twitdraw.com/.

[Watanabe, 2009] 渡邊恵太・塚田浩二 (2009). 「EyeWish: 目を閉じることを利用したインタラクション手法」. 『ソフトウェア科学会 WISS2009 論文集』, pp. 81–84.

[Weiser, 1991] Weiser, M. (1991). "The Computer for the 21st Century". *Scientific American*, **265** pp.94–100.

[Weiser, 1996] Weiser, M. and Brown, J. S. (1996). "Designing Calm Technology". *POWERGRID JOURNAL*, **1**.

[Wellner, 1993] Wellner, P. (1993). "Interacting with paper on the DigitalDesk". *Communications of the ACM*, **36**(7) pp.86–97.

[Wonderfl, 2011] Wonderfl (2011). http://wonderfl.net/.

[Yamashita, 2007] Yamashita, N., Hirata, K., Takada, T., Harada, Y., Shirai, Y., and Aoyagi, S. (2007). “Effects of Room-sized Sharing on Remote Collaboration on Physical Tasks”. *IPSJ Digital Courier*, **48**(12) pp.788–799.

[しぃペインター, 2011] しぃペ　イ　ン　タ　ー (2011). http://hp.vector.co.jp/authors/VA016309/.

[超漢字, 2011] 超漢字 (2011). http://www.chokanji.com/.

※ URL は 2012 年 2 月現在