スマートフォンを利用した複数画面の連携表示と動的なレイアウト変更によるアプリケーション

田中　潤†　　太田　高志‡

我々は，つまむ動作（pinch: ピンチ）によって，複数のスマートフォンで実行しているアプリケーションの連携を動的に設定するインターフェース，Pin-chを開発した．スクリーンをまたいで表示が連続するようにしたり実行中のレイアウトの変更に反応するようにアプリケーションを設計することで，単独画面ではできなかった新しいインタラクションを持つ様々なコンテンツを考えることができる．また，物理的な接触を必要とすることから，スマートフォンを持つ複数の人での利用を考えると，ネットワークや電話を介さない直接的なコミュニケーションを促進するメディアとしても期待できるだろう．本稿では，接続の仕組みと，それを利用したアプリケーションの開発について説明し，実際に作成したアプリケーション例を紹介する．作成例はプロトタイプ的なものであるが，インターフェースの実効性を検証することができた．

Multi-Smartphone-Display Applications that React Dynamically to Display's Layout

JUN TANAKA†　TAKASHI OHTA‡

We devised an interface to relate applications running on multiple smartphones merely by "pinching" the touch screens of juxtaposed devices. By making a virtual display field by multiple screens, or an application to react the change of screen's layout dynamically, we could design various contents of new interaction that a single screen unable to offer. As physical contact is essential to the interface, we can expect direct human communication would occur by playing the contents. In this paper, we introduce the interface and its mechanism, and also report about the prototype applications. Although the applications are simple ones, they demonstrate the validity of our approach.

1. はじめに

複数の画面を単一の仮想表示領域として構成するマルチディスプレイ環境の利用は，同じ作品を表示するのでも単独画面よりも大きなインパクトを与える効果が期待できる．また，多くの人が一度に鑑賞し，参加することが可能である．しかしながら，単一の画面として利用する限りではコンテンツの表現的な手法に変化がもたらされるわけではない．我々は，マルチディスプレイの可能性を追求していくなか[1][2]で，画面の構成をアプリケーションの実行中にも動的に変更し，表示もそれに併せて変化することができればより多様なコンテンツ表現を生み出すことができるのではないかと考えた．画面の構成を頻繁に入れ替えることを想定して，モバイル性と比較的大きな画面を備えているスマートフォンをプラットフォームとして仕組みを実装することとした．

† 東京工科大学大学院
Graduate School of Tokyo University of Technology
‡ 東京工科大学メディア学部
School of Media Science, Tokyo University of Technology

インタラクティブ性を実現することを考えると画面の連携に対してその都度設定画面を開いて登録するようなものは許容できない．我々は，スマートフォンで採用されているタッチディスプレイを利用し，隣接して並べた二つの画面の表面に指をすべらせ，ちょうど二つのものをつまみ合わせるような動作だけでそれらの連携を実現するインターフェースを設計した．スマートフォンのアプリケーションは，デバイス自体を傾けたり回転させたりして表示を直接操作するようなアナロジーを持たせているものが多く作成されている．画面を並べてつまみ合わせると表示が連携するというものもその延長にあるデザインと言えるだろう．我々の目的は，アプリケーションの表現に多様性をもたらすための新たなプラットフォームとして複数画面の動的な連携環境を利用できるような仕組みを整えることであり，また，そのためのインターフェースの提案と実証のためのアプリケーションの制作と検証を行うことである．

本発表では，基盤技術となるインターフェースの仕組みと，アプリケーション設計について説明し，それを利用して作成したアプリケーション例について紹介

する．作成したアプリケーションはプロトタイプとして機能検証が主な目的であり，本インターフェースを利用した表現の多様性を十分示すだけのバリエーションにはまだ不足しているが，課題の抽出と今後の発展の可能性を示唆することができたと考えている．

2. 関連研究

マルチディスプレイについての研究は，高解像度で広大な表示画面を確保する目的で科学的な可視化目的のために多く行われている[3][4]．また，仮想現実感に関連した研究においても，CAVEのような没入感を持たせる目的で複数画面を利用することが行われている[5]．我々の目的はこれらの研究のように表示環境自体の実現にあるのではなく，コンテンツ表現の多様性を模索するために，複数画面の構成を変更するのに併せて表示内容が反応するようなアプリケーションを創ることを支援する環境を用意することである．複数のスマートフォン画面を利用して，一つの仮想的な表示領域を作成する試みとしてはJunkyard Jumbotron[6]というアプリケーションがある．また，本研究と同じように，画面の組み合わせの変化で画面が反応する仕組みをもたせた，ゲーム用のデバイスとしてShifteo Cubesが発表された[7]．本研究と前者との違いは，静的な仮想画面を構築することではなく，動的な画面の構成変更と，それに反応するアプリケーションのデザインを目的としていることである．前者は始めに一斉に全画面の構成を設定する必要があるが，本研究のアプローチでは，画面はアプリケーションの実行中においても画面の追加や削除，並び順や画面方向などの変更が繰り返し可能である．また，後者との違いは，専用機器ではなく一般に普及するスマートフォンをプラットフォームとすることである．複数の人がそれぞれの所有するデバイスを持ち寄って使用することが想定されるため，表示デバイスとは別に管理用サーバーなどの中央制御のシステムを用意するのではなく，各デバイスが個々に独立して互いに連携する非集中的な機構としていることである．これにより，専用に用意されたデバイスではなく，その場に居合わせた人が持つデバイスを組み合わせて利用することができるため，人の直接のコミュニケーションを図るような用途への展開も期待できると考えている．

3. インターフェースの概要と設計

我々が開発したPin-ch（ピンチ）は，指でつまむ動作（pinch: ピンチ）によって，複数の異なるデバイス

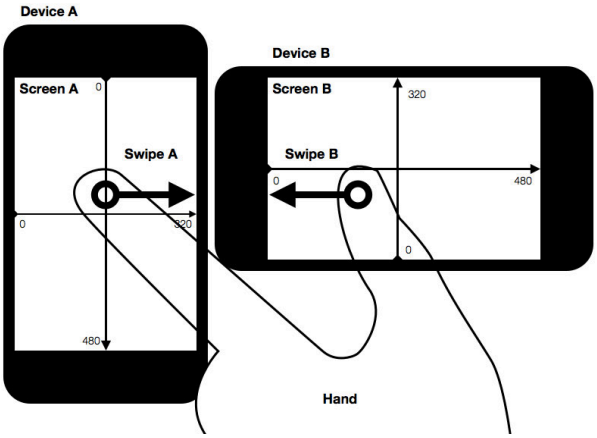

図1 つまむ動作でデバイスのスクリーンを接続するイメージ

のスクリーンを柔軟にレイアウトするための手法である．**図**1に示すように，操作者が2つのデバイスを隣接させ，各スクリーン上に1本ずつ置いた指を同時に引き合わせることで，2つのデバイスが互いを識別し，同時にスクリーンの相対的位置情報に基づいた表示を行うことができるようになる．

3.1 アルゴリズムの概要

隣接する2つのデバイスのスクリーンをまたぐように指をつまみ合わせると，スクリーンをなぞる動作（swipe: スワイプ）が2つの異なるデバイス上で「同時」かつ「相反方向」に，さらに「同一直線上」に発生する．Pin-chではこの特性を利用することで，ネットワーク上の複数のデバイスの識別，さらに相対的位置関係を適用したスクリーンレイアウトを実現する．

各デバイスはタッチスクリーン上で発生するスワイプの情報を随時記録し，これを無線ネットワークを介して他のデバイスと共有する．隣接する2つのデバイスのスクリーンを指でつまみ合わせると，2つのデバイス上で「同時」かつ「相反方向」にスワイプが発生する．この条件を用いることで，ネットワーク上に存在する複数のデバイスの中から，互いのデバイスを識別し，相対的な位置関係を推測ことができる．また，2つのスワイプは「同一直線上」に発生するため，その座標情報の差分を利用することで，互いのスクリーンの座標を統合することが可能となる．これに基づいてデバイス間通信を行うことで，デバイスの位置関係を表示コンテンツに反映させることが可能となる．

3.2 連携の確立の方法

各デバイスは，自機のタッチスクリーン上でスワイプ動作が行われる度にその情報を記録し，最新のスワイプ情報を随時更新する．更新記録に含まれる情報は**表**1のようになる．

スワイプ情報が更新される度に，デバイスは無線

表1 スワイプの更新記録に含まれる情報

属性	解説	値の例
Timestamp	スワイプ動作の発生時刻	…007.47…
Screen Size	スクリーンの大きさ（pixel）	{320, 480}
Location	スワイプの終点（pixel）	{320, 360}
Direction	スワイプの方向	right
Screen Orientation	スクリーンの姿勢	portrait
Device Roll	デバイスのY軸回転（radians）	0.123456…
Device Pitch	デバイスのX軸回転（radians）	0.123456…
Device Yaw	デバイスのZ軸回転（radians）	0.123456…

ネットワーク上の他のデバイスに対し更新の通知を行う．各デバイスはスワイプの更新通知を受け取ると，自機の最新のスワイプ情報と比較し，それらが同一のピンチ動作によって行われたものかを判定する．判定は次の3つ条件に基づく．第1に，2つのスワイプ動作が同時に行われていること．ピンチ動作による2本の指の動きは，必然的に同時に発生する．第2に，2つのデバイスのスクリーンが同じ面を向いていること．背中合わせの2つのデバイス上でピンチ動作は発生しない．そして第3に，2つのスワイプの方向が相反していること．これらの判定基準を満たす2つのスワイプを，同一のピンチ動作によるものと推定する．スワイプの組み合わせが判定基準を満たし，同じピンチ動作によるものと推定できる場合には，2つのデバイス間でPeer-to-Peerのコネクションを生成し，スワイプの組み合わせを保持する．

3.3　スクリーン座標の相対位置の取得

2つのデバイスの連携が確立した場合，両デバイスのスクリーンの相対的位置関係を，保持したスワイプの組み合わせから算出する．**図**2に示すスワイプAとスワイプBによって，デバイスA，デバイスBがマッチングされた場合，スクリーンの相対位置を算出する主な手順は次の通りである（**図**3を合わせて参照）．

1) デバイスAのスクリーン座標の中心に，デバイスBのスクリーン座標を重ねる．
2) デバイスAのスワイプの終点とスクリーンの中心点の差分だけ，デバイスBのスクリーン座標を移動させる．
3) スワイプAとスワイプBの方向をラジアン値に変換し，その差分だけのデバイスBのスクリーン座標を回転させる．
4) デバイスBのスワイプの終点とスクリーンの中心点の差分だけ，デバイスBのスクリーン座標を逆方向に移動させる．

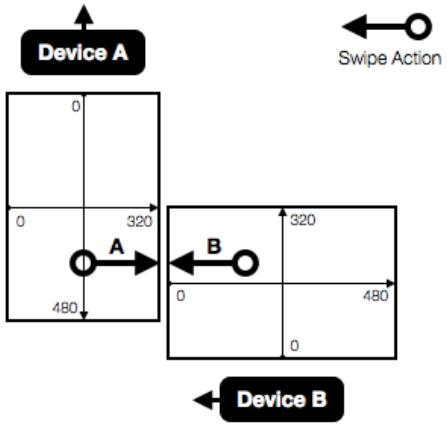

図2 デバイスA・B上で行われるスワイプ動作

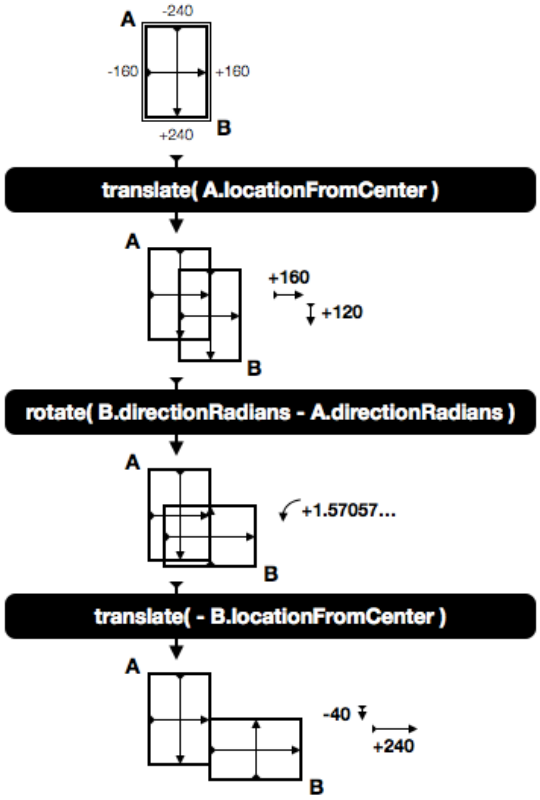

図3 デバイスA・Bのスクリーンの相対位置の算出手順

この手順を行うことで，各デバイスは接続相手のデバイスのスクリーンが，自機のスクリーンの座標上のどの位置にあたるのかを把握することができる．加えて，自機のスクリーン座標上にある特定の点が，接続相手のスクリーン座標上に置き換えた場合にどの点になるかを，相互変換することが可能となる．この算出方法は2つのデバイスのスクリーンサイズの異なる機能するため，大型のタブレットデバイスとスマートフォンなど，利用機会の異なるデバイスを同士を連携させることも可能である．

3.4　3台以上のデバイスの連携

1台のデバイスが複数のデバイスに対してコネクションを生成することで，3台以上のデバイスを連鎖的に連携させることも可能である．ピンチ動作によって結び付けられた2つのデバイスの間には，Peer-to-Peerのコネクションが生成される．基本的にPin-chでは，（スワイプ情報の更新通知を除き）結び付けられた隣接するデバイス同士でのみ通信が行われるが，リレー方式でデータを転送することで，連鎖的に接続された全てのデバイスに対して命令を行うことができる．データの転送を行う際，データにユニークな識別子を付加することで，再び同じ命令を処理することを回避する．

3.5　連携解除の方法

デバイス間の接続を解除する方法は，現段階で2つある．第1に，デバイスを振る（シェイク）動作をもって解除する方法がある．デバイスに内蔵された加速度センサーを用いて，一定以上の加速度が認められた場合に，該当デバイスに対するすべての接続を解除する．デバイスを振る動作は，スマートフォンにおいて「取り消し」のための一般的なインターフェースでもある．第2に，デバイスの回転（傾き）によって解除するという方法もある．同じくデバイスに内蔵されているジャイロセンサーを用いて，接続時の姿勢から一定以上の回転があった場合に該当デバイスに対するすべての接続を解除する．こちらはジグソーパズルのピースを取り外すイメージに近く，前述のシェイクよりも，解除に対してより敏感なインターフェースと言える．デバイスを手に持った状態での使用では前者，卓上に置いて使用する場合には後者といった具合に，コンテンツの利用形態に合わせて解除方法を選択することを想定している．

4.　アプリケーションの開発

我々はPin-chの機能をプログラミング・フレームワークとして整備した．アプリケーション開発者はPin-chフレームワークを利用することで，Pin-chによるダイナミックなスクリーンレイアウトを利用したアプリケーションを開発することができる．Pin-chフレームワークは，ネットワークアクセスからスワイプ情報の更新，デバイスのマッチング，接続，解除，データ通信，そしてスクリーンの相対的位置関係に基づく座標変換処理までを一貫して担う．アプリケーション内の任意のビューオブジェクトを紐付けるだけで，Pin-chフレームワークは自動的に「ピンチ」動作によるデバイスの接続を通知し，座標変換とデータ通信のためのインターフェースを提供する．これらのインターフェースを利用して，アプリケーション開発者は，スクリーンの位置関係をどのようにコンテンツに反映させるかを実装する．

Pin-chフレームワークは現在，iPhoneおよびiPod touch，iPad用アプリケーション開発に利用できるiOS用のObjective-Cライブラリとして実装している．ネットワーク通信はWi-Fiを介して行い，Bonjourプロトコルを用いてWi-Fiネットワーク上のPin-ch対応アプリケーションを実行しているデバイスのIPアドレスを自動的に検出する．デバイス間で行われるPeer-to-Peerの通信プロトコルについては，現段階ではTCPとUDPをアプリケーションによって使い分けている．

4.1　アプリケーション「鈴虫」

Pin-chの動作検証を目的として，我々が最初に開発したアプリケーションである．スクリーン上の草むらにいる鈴虫を，指でつついて移動させるというシンプルなものである（**図4**）．鈴虫の近くを指先でつつく（タップする）と，鈴虫は指の反対方向へと逃げていく．ここでPin-chを利用して複数のデバイスをつなぎ合わせると，双方のスクリーン上の草むらが連結され，鈴虫はスクリーンの枠を超えて隣のデバイスへと逃げていくようになる．また，鈴虫の移動に伴い，スピーカーから再生される鈴虫の鳴き声も，移動先のデバイスから発せられるようになる．デバイスを上下左右のどの方向に接続しても，鈴虫が正しい方向へとアニメーションしていくことを示した．

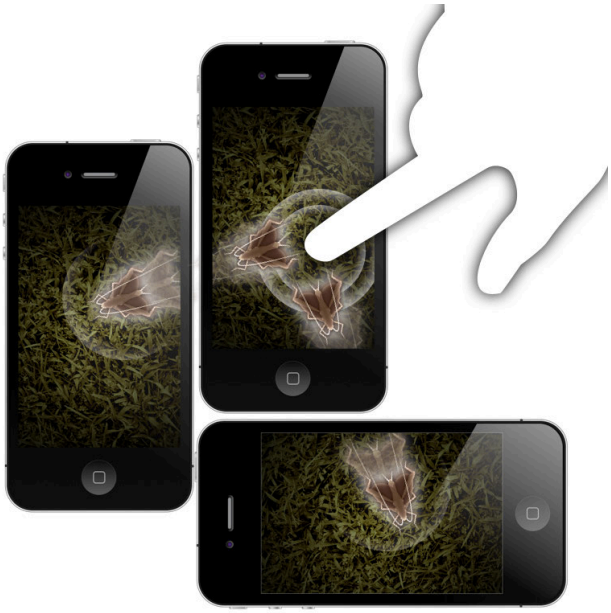

図4 アプリケーション「鈴虫」

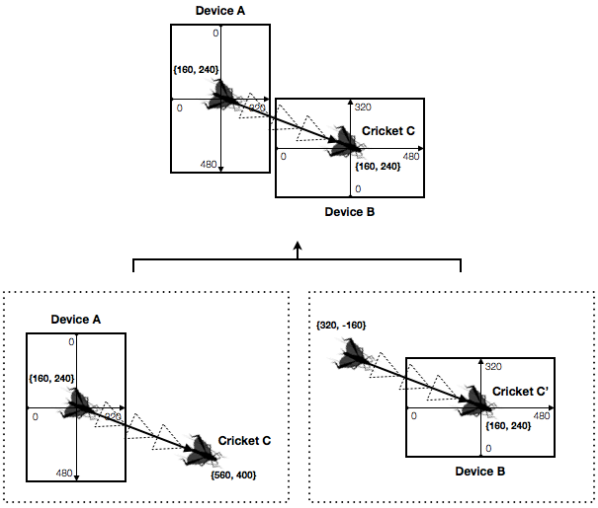

図5 スクリーンをまたぐアニメーションの実装

このアプリケーションでは，スクリーンをまたいで鈴虫が移動していくアニメーションを実装している．**図**5のように，デバイスAのスクリーン上にあるクリケットCが，スクリーンの枠を超えてデバイスBのスクリーン上へと移動していくようなアニメーションを実現する場合，手順は以下のようになる．

1) デバイスAのスクリーン座標における移動の始点と終点を設定する．
2) Pin-chフレームワークの提供するインタフェースを使い，デバイスBのスクリーン座標における移動の始点と終点を算出する．
3) デバイスBのスクリーン座標における始点の位置に，デバイスA上にあるクリケットCをコピーしたクリケットC’を設置する．
4) クリケットCとクリケットC’に対し，同時に移動アニメーションを適用する．
5) デバイスAのスクリーン上からクリケットCを取り除く．

4.2 アプリケーション「Tuneblock」

Tuneblockは，連結可能なミュージック・シーケンサーである．ユーザは各デバイス上で作成した音の配列を，Pin-chを利用してリアルタイムで繋ぎあわせていくことができる（**図**6）．各デバイスのスクリーン上にはオルゴールを模した16列8行のグリッドが表示されている．ユーザが任意の位置をタップすることでグリッド上に突起が追加され，スクリーンを左から右へ向かって移動する走査線が突起に接触すると，突起の縦方向の位置に対応した音が再生される．走査線がスクリーンの右端に達すると，再び左端から右端へと移動を繰り返す．この走査線の移動は，Pin-chを利用して複数のデバイスをつなぎ合わせることで，デバイス間で同期されるようになる．デバイスを横方向に接続すると，スクリーンの右端へ達した走査線は右隣のデバイスへと移動するようになり，縦方向に接続したデバイスの走査線は同一直線上に揃えられる．つまり，横方向へつなぐことでメロディの延長となり，縦方向へつなぐことで重奏となる．

Tuneblockの特徴のひとつとして，演奏中でもスクリーンのレイアウトを自由に変更できることが挙げられる．レイアウトの変更に素早く対応するため，Tuneblockでは演奏の走査線が1列進行する度に，Pin-chによって接続されたスクリーンのヒットテストを行なっている．走査線の延長線上の座標に接続されたスクリーンの1列目のグリッドが存在する場合に，相手のデバイスへメッセージを送り演奏を開始させる．走査線が最後列に達した場合にも，同様に右隣にスクリーンがあるか確認し，必要に応じてメッセージを送る．また，Tuneblockでは演奏のループを実現するために，3.4項で述べたPin-chのメッセージ中継機能を利用している．スクリーンのレイアウトが変更される度に，接続されたデバイス全体における演奏の始点と終点が自動的に設定される．走査線が終点に達すると，終点のデバイスは演奏が終了したことを隣接するデバイスへ通知し，通知が始点のデバイスへと到達するまで各デバイスは自身の隣接するデバイスへと通知を中継していく．通知が始点まで達すると，再び演奏が開始される．

図6 アプリケーション「Tuneblock」

5. 評価

5.1 動作検証

Pin-chを使ったスクリーンレイアウトの即時性，ならびに座標の正確性について検証を行った．結果の概要は**表**2の通りである．まず，通信プロトコルとしてTCP，UDPをそれぞれ用いた場合の，ピンチ動作からスクリーンの接続が成立するまでの時間を計測した．各10回計測を行い，結果はTCPが平均1.68秒，UDPではほぼ即時に反映された．同様に接続解除までの時間は，TCPが0.84秒，UDPがほぼ即時の反映であった．また，接続されたスクリーン座標のずれについても計測した．こちらも10回計測を行い，結果は平均16.3ピクセルとなった．親指と人差し指でピンチする際，親指が寝ている状態だと誤差が拡大する傾向があった．

結果として，「動的なレイアウト変更」に耐えるインタフェースとして十分な速度を実現できたと言えるだろう．座標の誤差についても，スクリーンが「繋がっている」ことを実感する上では十分に許容できる誤差である．しかしながら，デバイスがアクセスするWi-Fiネットワークの混雑状況によっては，レイアウトの反映が大きく遅延する場合もある．より安定した環境を実現するため，今後はBluetoothを介した接続オプションも検討していきたい．

5.2 展示会での反応

去る2011年9月に開催された東京ゲームショウ2011にて，Tuneblockの展示を行った．TuneblockをインストールしたiPod touchを6台用意し，展示中はバッテリーの状態によって2台から4台をデモ機として稼働させ，来場者が自由に試遊できるようにした．来場者からは「演奏がつながる様子が手品みたい」「いろんな使い方ができそう」「ゲームの開発に使いたい」「こういうのが欲しかった」等の肯定的な感想が多く寄せられた．ただし操作している様子を観察していると，一度接続したデバイスを再び異なるレイアウトで接続する際，解除と再接続が円滑に行えていないケースが見受けられた．接続状態の遷移をフィードバックする方法については，改めて検討する余地がある．

表2 検証項目と結果

検証項目	結果
接続成立までの平均時間（TCP）	1.68 秒（10回の平均値）
接続解除までの平均時間（TCP）	0.84 秒（10回の平均値）
接続成立までの平均時間（UDP）	ほぼ即時
接続解除までの平均時間（UDP）	ほぼ即時
スクリーン座標位置の平均誤差	16.3 px（10回の平均値）

6. おわりに

「指でつまんで画面を繋げる」というインターフェースがもたらす面白さとして，直接的で直感的な操作であることと併せて，実際にデバイスを手に持って配置操作をすることがあるだろう．それに加えて，コンテンツの表示が配置の変化に動的に反応することで従来には無い面白さを持つアプリケーションの表現を創ることができると期待している．

本アプローチは，単に画面の連携が可能となる機構を用意しただけではなく，そのプラットフォームとして今や多くの人が所有するスマートフォンを利用している．従って，友人が何名か集うような時をアプリケーション利用の機会の一つとして想定することができるだろう．そのような用途を想定したとき，ゲームなどのエンターテイメント用途のものだけでなく，広告用途のアプリケーションや，直接物理的な接触が必要なインターフェースの特性を生かしてネットワーク越しではなく人が対面で直接コミュニケーションをすることを促するソーシャルメディアへと展開する可能性もあるのではないかと考えている．

参 考 文 献

1) Takashi Ohta: Dynamically Reconfigurable Multi-Display Environment for CG Contents, Proceedings of Advances in Computer Entertainment Technology 2008, pp. 416 (2008).
2) 太田高志，田中潤，KAON（顔音）: 顔認識を利用したインタラクティブ・コンテンツ，芸術科学会論文誌，Vol.10, No.3, pp. 148-156 (2011).
3) Tao Ni, et al.: A survey of Large High-Resolution Display Technologies, Techniques, and Applications, Proceedings of Virtual Reality Conference, pp. 223-236 (2006).
4) Sungwon Nam, et al.: Multi-Application Inter-Tile Synchronization on Ultra-High-Resolution Display Walls, Proceedings of the ACM SIGMM Conference on Multimedia Systems, pp. 145-156 (2010).
5) Carolina Cruz-Neira et al.: Surround-screen projection-based virtual reality: the design and implementation of the cave, SIGGRAPH'93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pp. 135–142, New York, NY, USA (1993). ACM.
6) Rick Borovoy, Brian Knep: Junkyard Jumbotron, http://jumbotron.media.mit.edu/
7) D. Merrill, J. Kalanithi, P. Maes: Siftables: Towards Sensor Network User Interfaces, Proceedings of the First International Conference on Tangible and Embedded Interaction (TEI'07), February 15-17 in Baton Rouge, Louisiana, USA.