その場でタイルドディスプレイ

井出 拓弥[1]　成見 哲[1]

概要：本研究では, モバイル端末を複数用いてその場で大画面を構成するシステムを開発した. タイルドディスプレイとは, 複数台のディスプレイを組み合わせて一つの大きな画面として使う技術であるが, 通常はデジタルサイネージ用に使用されることが多く持ち運べない. 本システムでは Unity を介したサーバークライアント方式によるタブレット 4 画面のシステムを開発した. 4 つのタブレットの相対位置はインカメラで AR マーカーを認識することで自動的に取得している.

On-the-fly Tiled Display System with Mobile Devices

Ide Takuya[1]　Narumi Tetsu[1]

Abstract: In this research, we developed a tiled display system with mobile devices on the fly. A tiled display is composed of multiple displays to form a big screen, and often used for digital signage, which means not portable. In our system a server-client system is developed using Unity with four displays of tablets. Their relative positions for rendering is automatically determined with an AR marker using front cameras of tablets.

1. はじめに

近年, スマートフォンやタブレットなどのモバイル端末が普及しており, 東京地区のスマートフォン所有率は約 78%, タブレット所有率は 41%である [1]. また,Youtube や Netflix 等の動画配信サービスが普及し, 様々な映像コンテンツが供給されている.

しかし, 高精細な動画を視聴する際にはモバイル端末の画面は小さいと感じる人も多い. それを解決する技術として, タイルドディスプレイが挙げられる. タイルドディスプレイは複数のディスプレイを並べて大画面を実現する技術で, 同じサイズの大画面ディスプレイを購入するよりも安価であることや運搬が容易であるという利点がある.

本研究ではモバイル端末を複数用いて大画面を構成することを考える. モバイル端末を用いてタイルドディスプレイシステムを実現した例として, 2 本の指をつまむように近づけるピンチ操作を行うことで画面を結合させることができるシステム [2], インターネットを介してサーバーに接続し, ウェブブラウザ上で表示するシステム [3] などがある. また, 複数のモバイル端末を組み合わせて用いた例として, 複数のスマートフォンを用いて動画編集を行うシステム [4] などがある.

「その場でタイルドディスプレイ」とは, みんなで持ち寄ったモバイル端末を集めてその場所で大きな画面を作ることを表す. 本研究では第一段階として 4 台のタブレット端末での表示が行えるシステムを開発した. 特にタブレット間の位置をインカメラを用いて自動的に判断させている.

2. UnityMobileStreaming[5]

本研究室で開発された Android 向けプラグインであり,Unity[6] 向けアプリケーションを手軽にクラウド化することができる. 図 1 に示すように, このシステムはサーバクライアント方式で実装されている. サーバとクライアントはソケット通信によって情報のやり取りを行う.

2.1 サーバ

サーバは Unity スクリプトとして作成されている. このスクリプトを Unity 内で動作させることでその Unity プロジェクトをサーバ化することができる.

[1] 電気通信大学
The Univercity of Electro-Communications

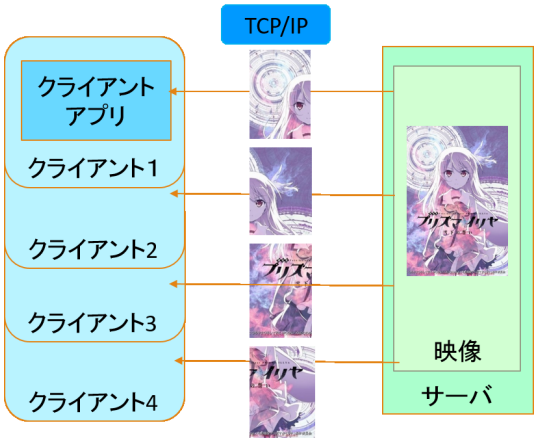

図 1　UnityMobileStreaming システム構成
Fig. 1　System configuration of UnityMobileStreaming

Unity スクリプトによって Unity 実行画面のスクリーンショットを連続で取得し, クライアントへ送信する (図 1). サーバは Unity 内で完結しているため, Unity が動作する環境であれば, Windows に限らず Mac, Linux 等 PC の OS に限らず動作させることができる. Unity スクリプトで作成された UnityMobileStreaming クラスはクライアントから受け取った操作情報をプロパティとして保持する. static クラスとして定義されているため, これらのプロパティは外部のスクリプトから直接参照することができる.

2.2　クライアント

クライアントは Android アプリとして作成されている. サーバの IP アドレスとポート番号を指定して接続する. サーバから受信した映像を画面に表示するとともに, Android 端末のセンサ情報を取得し, 操作情報としてサーバへ送信する.

2.3　4 画面出力

このシステムは 4 画面タイルドディスプレイ出力に対応している. 4 台の Android 端末を対象とし, クライアント毎に異なるポートを使用して通信を行う. その際, サーバ側であらかじめ画像を分割しておき, それぞれのクライアントに画像を送信することで 4 画面タイルドディスプレイを実現する.

3.　「その場でタイルドディスプレイ」システム

3.1　概要

システム構成図を図 2 に示す.UnityMobileStreaming と本システムの差異はモバイル端末のインカメラを用いて自動的に映像の描画範囲を決定することである. インカメラを用いて AR マーカーを認識することでモバイル端末の座標を取得し, AR マーカーとの相対位置によって映像の描画範囲を決定する. 図 3 は 4 画面に対応している例である. 現状では田の字型の 4 画面にのみ対応している.

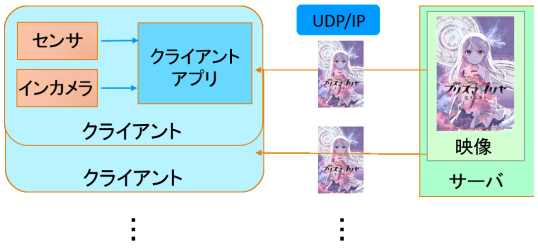

図 2　システム構成
Fig. 2　System configuration

3.2　Unity 内での動画再生

Unity 内で動画を再生するために Texture コンポーネントである MovieTexture を利用した. 事前に Unity プロジェクト内に動画を配置しておき,MovieTexture に配置した動画を読み込むことで Unity の 3D 空間内に動画をインポートすることができる.

3.3　映像の分割

サーバは毎フレームスクリーンショットを取ることで映像を画像として保存し, クライアントに送信する. クライアントはサーバから送られてきた画像を左上, 左下, 右上, 右下の 4 枚の画像に分割する.

3.4　モバイル端末の位置の取得

クライアントで動作する Unity アプリケーション内にモバイル端末のディスプレイの中心を原点とする Unity3D 空間を作成し, 原点に AR マーカーを認識する AR カメラを鉛直上向きに設置する. これにより, AR マーカーをモバイル端末のインカメラに映るようにかざすことで Unity3D 空間内で AR マーカーの座標を認識することができる. AR マーカーを 4 台のモバイル端末の中心かつ上方に設置し, それぞれの端末が認識した AR マーカーの座標によって描画範囲を決定する. AR ライブラリには, Vuforia[7] を用いた.

3.5　AR マーカーのトラッキングの停止

インカメラが AR マーカーのトラッキングを常に行っていると, クライアント側の処理が重くなってしまう. また, 端末の配置によっては切り出し範囲が安定しないこともあり得る. そこで, 加速度を常に監視し, 変化を検出したらトラッキングを開始し, 1500ms 後にトラッキングを停止する. これは加速度を検出した際はモバイル端末を動かして配置を変更しようとしている状態であると想定されるためである.

4.　性能評価

クライアントが 1 台の場合, 4 台で AR 認識を行っている場合, 4 台で AR 認識を止めた場合の 3 つの環境におけ

サーバー画面　　クライアント画面

図 3　4 画面出力結果
Fig. 3　Snapshot with four displays

表 1　動画再生のフレームレート
Table 1　Frame Rate of Playback Movies

クライアント	フレームレート (fps)
1	7.48
4(AR トラッキング OFF)	1.87
4(AR トラッキング ON)	1.70

る動画再生時のフレームレートの測定を行った.

4.1　測定法

0.1 秒毎に表示される画像が変化する動画を作成し, 本システムのサーバで再生する. その動画がクライアントで描画されている様子を動画として撮影し, 描画された画像の枚数をカウントすることでフレームレートを測定する. 測定時の条件は以下の通りである.

- サーバ CPU : Intel(R) Core(TM) i7-4790
- サーバ GPU : GeForce GTX 970
- サーバ OS : Windows10
- クライアント OS : Android6.0.1
- クライアント端末 : Media PadT2 Pro
- 画面解像度 : 839 × 472 ピクセル
- 計測フレーム数 : 200 フレーム

なお, サーバとクライアントは同一ローカルネットワーク内に存在している.

4.2　結果

表 1 にフレームレートの測定結果を示す.1 画面の場合, 平均して 7.48fps, 4 画面の場合では 1.87fps という結果となった. 出力台数の増加とクライアントアプリ上での AR ライブラリの動作が速度低下の原因であると考えられる. 4 画面の AR トラッキングを行っている場合と行っていない場合を比較すると, 行っていない場合の方が約 0.17fps 上昇していることから AR トラッキングはクライアントの動作に影響を及ぼしていたことが確認できた.

5.　まとめと今後の課題

モバイル端末 4 台を使って, 動的にレイアウト変更可能なタイルドディスプレイシステムを開発した. しかし, 本研究の目的である動画の視聴を行うためには現状の速度では不足していることから, 通信速度およびアプリケーション全体の動作速度の向上を行う必要がある. 速度向上および画面の同期を取るためにブロードキャストで映像を送信することを想定しているため, 現状 TCP で通信している UnityMobileStreaming を UDP に変更する必要がある. また, モバイル端末の位置をを自由に調整できるようなシステムにするために, 端末の配置から画面サイズを計算して自動的に調整し, それぞれのモバイル端末に応じて映像を切り出す必要がある. さらに, 現状は画面の位置の取得のために AR マーカーを必要としているが, AR マーカーの代わりに人の顔を認識するなどして,AR ライブラリを用いずに端末の位置を判定する方法が必要になる.

参考文献

[1] メディア定点調査 2017, 博報堂 DY メディアパートナーズのメディア環境研究所, 2017, 6 月 20 日, http://www.garbagenews.net/archives/2170355.html(最終アクセス: 2017/7/5)
[2] Takashi Ohta, Jun Tanaka, ”M’obile’ Tile: Interactively Adjustable Free Shape Multi-Display of Mobile Devices”, School of Media Science Tokyo University of Technology, SA’ 15 Symposium on MGIA, November 02-06, 2015, Kobe, Japan
[3] Julia Schwarz, David Klionsky, Chris Harrison, Paul Dietz, Andy Wilson, ”Phone as a Pixel: Enabling Ad-Hoc, Large-Scale Displays Using Mobile Devices”, Carnegie Mellon Univercity, CHI’ 12, May 5-10, 2012, Austin, Texas, USA
[4] Ryota Shimamoto, Homei Miyashita, ”Movie edit System for Multi-user with SmartPhones ”, エンタテインメントコンピューティングシンポジウム (EC2011), 2011 年 10 月, 日本科学未来館 (東京), 日本
[5] 高橋悠, 成見哲, ”Android 向け Unity アプリケーションのクラウド化”, エンタテインメントコンピューティングシンポジウム (EC2014), 2014 年 9 月, 明治大学 中野キャンパス, 日本
[6] Unity, https://unity3d.com/jp(最終アクセス: 2017/7/26)
[7] Vuforia, https://developer.vuforia.com/ (最終アクセス: 2017/7/2)