

# UnityMobileStreaming を用いたタイルドディスプレイのタッチ操作の実現

情報・通信工学科 学籍番号:1111017 成見研究室 伊藤 一輝

## 1.はじめに

近年では iPhone や Android 端末の急速な普及に伴い、高解像度のマルチタッチ操作が可能なディスプレイが多く見られるようになった。マーケティングにおいてはデジタルサイルネージといった大型ディスプレイに適した高解像度映像コンテンツがタイルドディスプレイシステムを利用して提供されるようになってきている。タイルドディスプレイシステムとは、複数の小さなディスプレイを格子状に並べることで1つの大きなディスプレイとして見せる技術である[1](図1)。

ただし複数のディスプレイによるタッチ操作を実現するには何らかの工夫が必要である。

一方モバイル端末向けゲーム開発においてゲームエンジンおよび統合開発環境である Unity3D[2] (以下 Unity)が普及してきた。成見研究室の高橋は非力なモバイル端末でも快適な物理演算を実現するクラウド技術として UnityMobileStreaming を開発しモバイル端末の可能性を広げた[3]。

本研究では対象を Unity を使ったゲームアプリケーションに限り、モバイル端末を使ってタイルドディスプレイを実現することを目指す。

## 2.既存研究

### 2.1 タイルドディスプレイ

タイルドディスプレイのメリットは1枚の大型ディスプレイより安価であり持ち運びが可能なことである。安枝の研究[4]では持ち運び可能なモバイルタイルドディスプレイが開発されたがタッチ操作が実装されていないという問題があった。

### 2.2 SAGE

SAGE[5]はアメリカのイリノイ大学 EVL で開発されたソフトウェアである。壁に取り付けた複数のディスプレイをタ



図1:タイルドディスプレイ(参考文献[1]より)

イルドディスプレイとして利用しタッチ操作を可能にした。しかし、当研究ではモバイル端末をタイルドディスプレイとして使用する点が異なっている。

## 3.システム構成

### 3.1 Android 向け Unity アプリの仮想化

UnityMobileStreaming は Unity で作成されたゲームをサーバとクライアントに処理を分け、Android クライアントからの入力を受けてサーバ上で演算処理を実行し、その結果を映像として返すシステムである(図2)。

### 3.2 画面分割の実装

映像出力は 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、クライアントに送信する。複数のクライアントに対し指定した画像を送信するためには、サーバ側での画面分割処理が必要となる。スクリーンショットで取得した画像データに対し新しく空のテクスチャを用意し[6]ディスプレイに該当する指定領域の画像データをピクセル単位で読み込み、テクスチャを PNG ファイルにエンコードして[7]クライアントへ送信する手法を考案した(図3)。

### 3.3 座標の管理

クライアントから操作入力でタッチされた座標はサーバ側にとってはどのタブレットから送信されてきた座標なのか判別が困難である。そこでクライアント側では予め ID を割り振っておくことで、どのタブレットから送信された情報であるかを確立させ座標データの計算を可能にした。

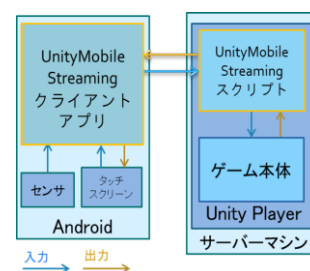


図2:システム構成

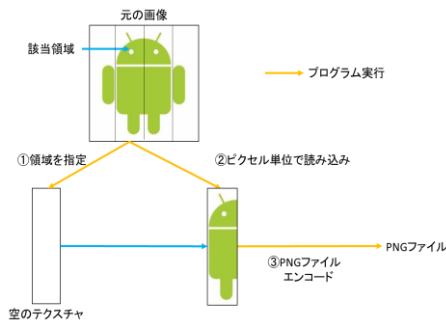


図 3:画像の分割

表 1:平均通信速度結果

接続台数(台)	1	2	4
平均通信速度(Mbps)	7.22	7.39	6.5

表 2:解像度と台数に応じた平均描画フレームレート (fps)

解像度	台数(台)	1	2	4
	360×260	20.42	11.28	9.58
720×520	18.03	11.38	10.64	
1440×1040	18.30	10.88	8.10	

### 3.4 ベゼルデータをまたぐドラッグ操作の開発

ベゼルデータを間に挟むドラッグ操作に対応するために複数の指でタッチされた座標を1つの点として扱いドラッグする方法を考案した。この場合、複数の指を操作するためベゼル内に指があっても他の指がタッチし続けるため判定が可能となる。

## 4.評価

各タブレットの接続台数に対して平均通信速度と描画フレームレートの評価を行った。測定時における評価の環境は以下の通りである。

- サーバ CPU: Intel Core i7 CPU 920
- サーバ GPU: NVIDIA GeForce GTX 670
- サーバ OS: Windows8.1
- サーバ解像度: 1920×1080 ピクセル
- クライアント端末: Nexus 7 (2013 年版)
- クライアント OS: Android4.4.2
- クライアント解像度: 1920×1200 ピクセル
- クライアントの Wi-Fi リンク速度: 54Mbps

### 4.1 平均通信速度

解像度を 720×520 に固定して Android タブレット 1 台、2 台、4 台における通信速度を計測した。タブレットに送信した画像の合計バイト数から通信速度を求め、デモプログラムが実行されてから計測終了までの約 150 秒間に実行中における平均通信速度を表 1 に示す。

### 4.2 描画フレームレート

UnityMobileStreaming はサーバからクライアントに単位時間辺りに送信されているコマ数である描画フレームレートが重要になる。UnityMobileStreaming は毎フレーム画面をキャプチャするが、通信はゲーム動作とは非同期であるため、実際にクライアントに送信している画像は通信時の最新の画面のみである。約 0.5 秒毎にその間実際にクライアントに送

信した画像の回数から描画フレームレートを計算し約 150 秒間の平均描画フレームレートを表 2 に示す。

### 4.3 結果と比較

表 1 よりタブレット 1、2 台による平均通信速度に変化はあまりないが 4 台の場合、平均通信速度が他に比べて遅くなっていることが分かる。表 2 では描画フレームレートはタブレットの台数に主に依存し、解像度にはそれ程影響は受けていないことが分かる。

## 5.まとめと今後の課題

Android 端末を 4 台用いることで、1つのタイルドディスプレイとしてタッチ操作とドラッグ操作を実現した。また隣接したタブレット同士のベゼルを考慮したタッチ操作を可能にした。

今後の課題として拡大や縮小といったピンチイン・アウトの操作、画面のスクロールに用いられるフリック操作といったマルチタッチ操作があげられる。今回は Unity アプリケーションだけでタッチ操作を実現しているが、どんなアプリケーションでも対応する方法が望ましい。

## 参考文献

- [1]—タイルドディスプレイシステム  
<http://japan.cnet.com/digital/av/20414727/>
- [2]—Unity3D: <http://japan.unity3d.com/>
- [3]—高橋 悠、Android 向け Unity アプリの仮想化、電気通信大学、エンタテイメントコンピューティング、2014
- [4]—安枝 光 モバイルタイルドディスプレイの開発、電気通信大学、情報処理学会第 76 回全国大会、2014
- [5]—SAGE2: <http://sage2.sagecommons.org/project/>
- [6]—Unity のオフスクリーンレンダリング  
[http://qiita.com/tsubaki\\_t1/items/ee8419c054d17ade32ee](http://qiita.com/tsubaki_t1/items/ee8419c054d17ade32ee)
- [7]—Texture2D を png ファイルとしてファイル保存  
[http://ft-lab.ne.jp/cgi-bin-unity/wiki.cgi?page=unity\\_script\\_texture2d\\_save\\_png\\_file](http://ft-lab.ne.jp/cgi-bin-unity/wiki.cgi?page=unity_script_texture2d_save_png_file)