協調ワークスペースドライバと
協調フレームワークのプロトタイプ整備

設計書

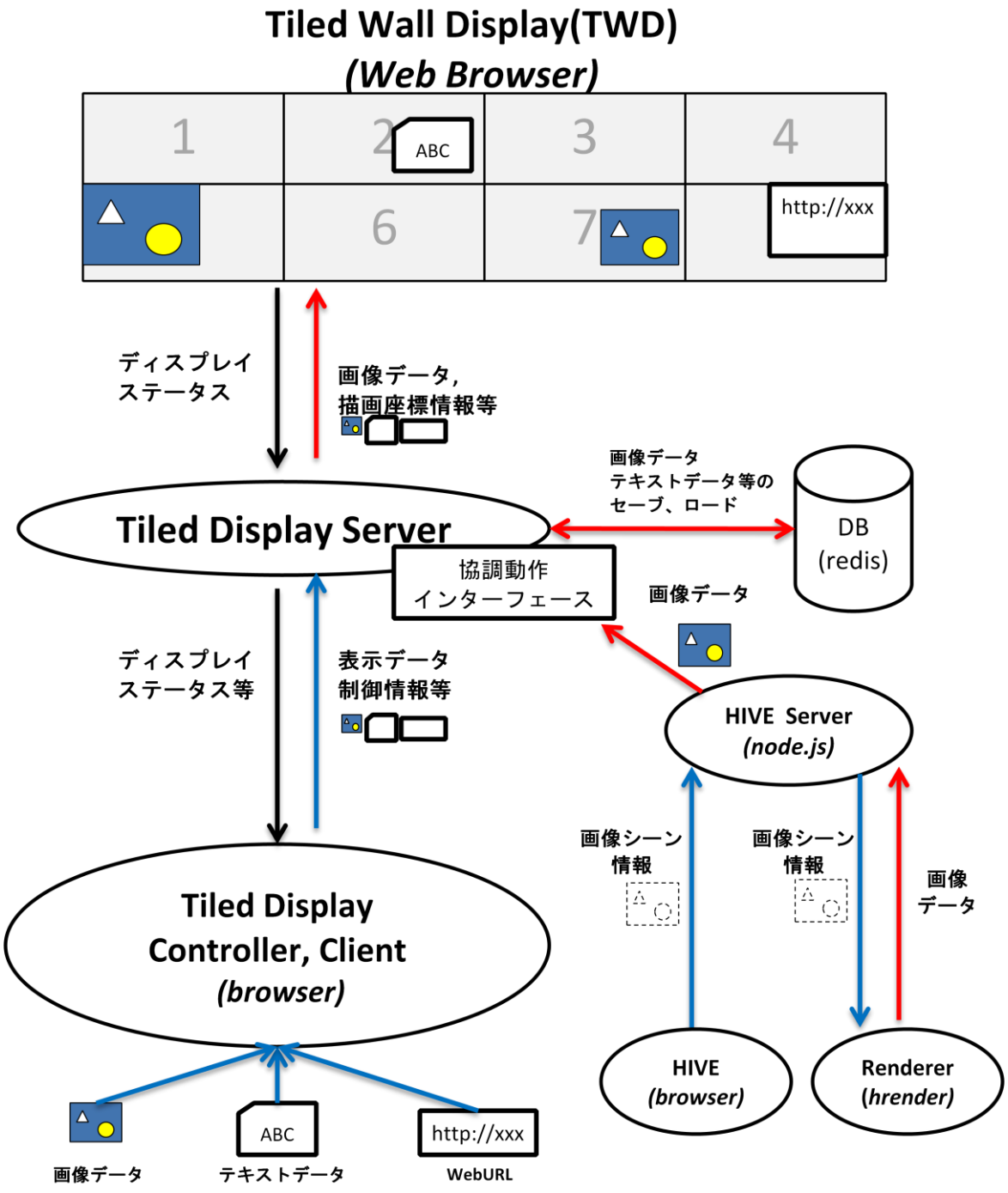
1. 目次

1. システムの構成について 3
1.1 全体の構成 3
1.2 サーバの構成 4
1.2.1 サーバ 4
1.2.2 データベース 4
1.2.3 その他ソフトウェア 4
1.3 クライアントの構成 4
1.3.1 コントローラ 5
1.3.2 ディスプレイ 6
2. 通信用 API の仕様 7
2.1 メタバイナリ 8
2.2 リクエスト/レスポンスコマンド 8
2.3 更新通知コマンド 9
3. データ構造について 10
4. コントローラについて 11
4.1 コンテンツの登録について 11
4.2 仮想ディスプレイ全体の設定について 11
4.3 コンテンツ/ディスプレイの設定について 11
5. ディスプレイについて 13
5.1 ディスプレイの登録について 13
6. 画面操作について 13
7. 動作環境について 14

1. システムの構成について

1.1 全体の構成

本システムは，node. js 及び websockets を用いたクライアントサーバプログラムであり，複数のアプリケーションや, 複数のユーザが共同作業を行える，巨大なスクリーンスペースを，仮想ディスプレイとして提供する．以下に構成図を示す.



1.2 サーバの構成

1.2.1 サーバ

サーバは，node.js を使用して，socket.io 及び websocket による通信に対応している．それぞれの通信方法によって受付ポートを分けている．

受付ポート	通信方式	クライアント	HTTP アクセス時
ポート A	socket.io	socket.io を用いたコントローラによって使用される	スタートページが表示される
ポート B	websocket	websocket を用いたディスプレイによって使用される	
ポート C	websocket	websocket を用いたコントローラによって使用される	

1.2.2 データベース

データベースは，redis を使用して，高速なレスポンスを実現している．

1.2.3 その他ソフトウェア

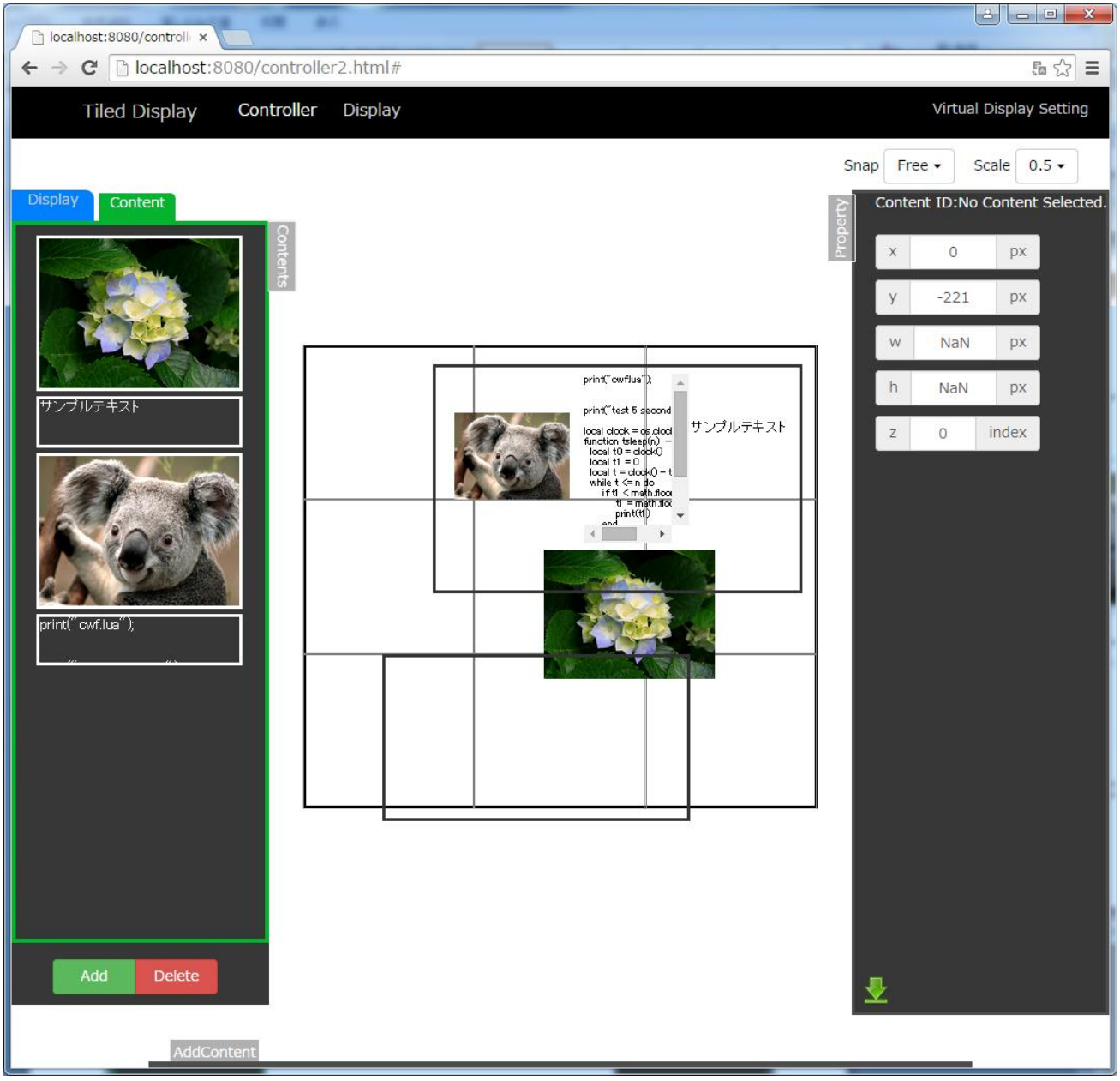
ウェブページレンダリング用に PhantomJS，及び phantom.js の npm ラッパーである phantomjs を使用している．

1.3 クライアントの構成

クライアントサイドは，仮想ディスプレイに対してコンテンツの追加登録等の操作を行う「コントローラ」と，表示のみを行う「ディスプレイ」から構成されている．

1.3.1 コントローラ

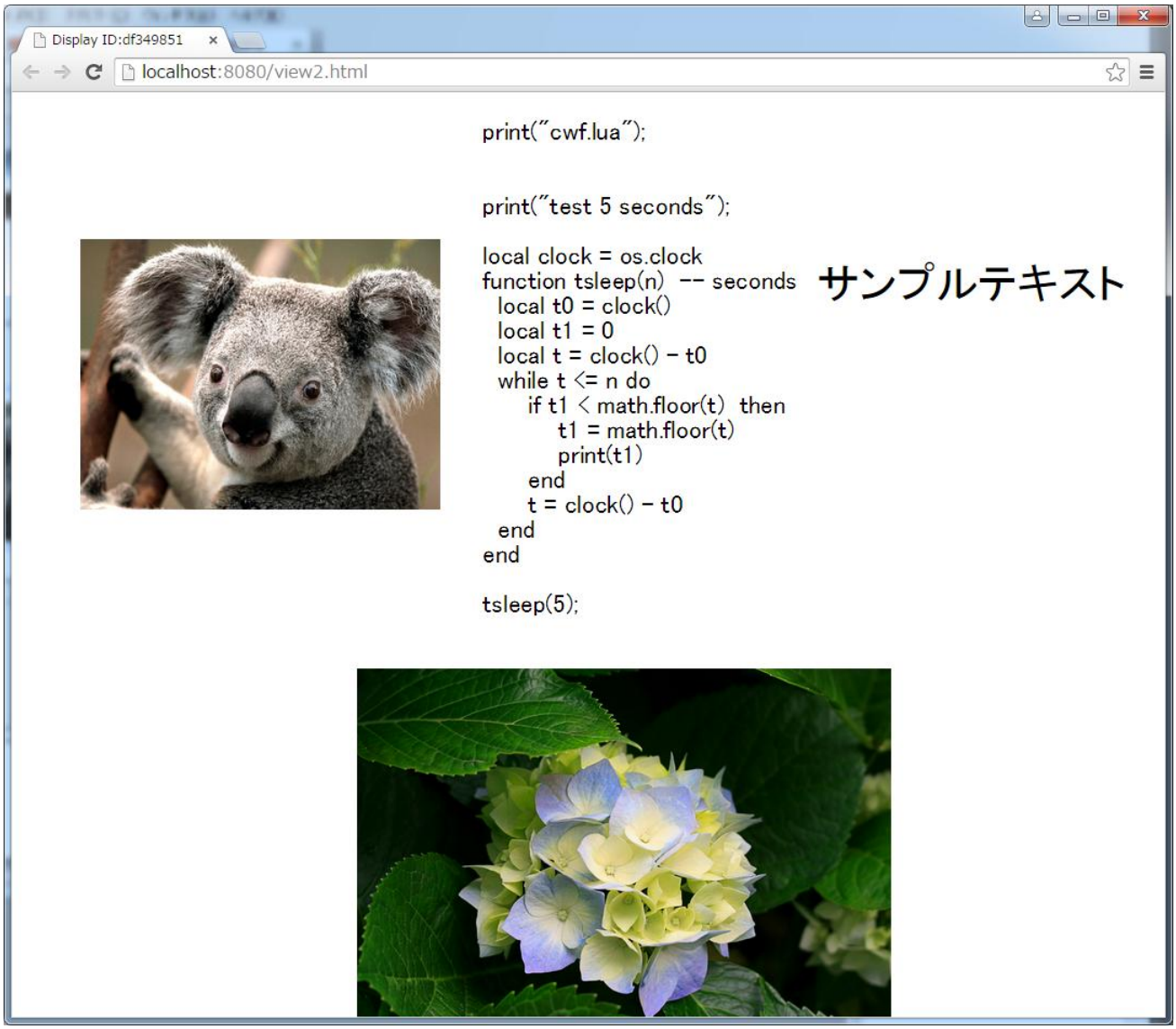
コントローラは，socket.io を用いてサーバと通信を行っている．
コントローラの画面イメージを以下に示す．



コントローラ

1.3.2 ディスプレイ

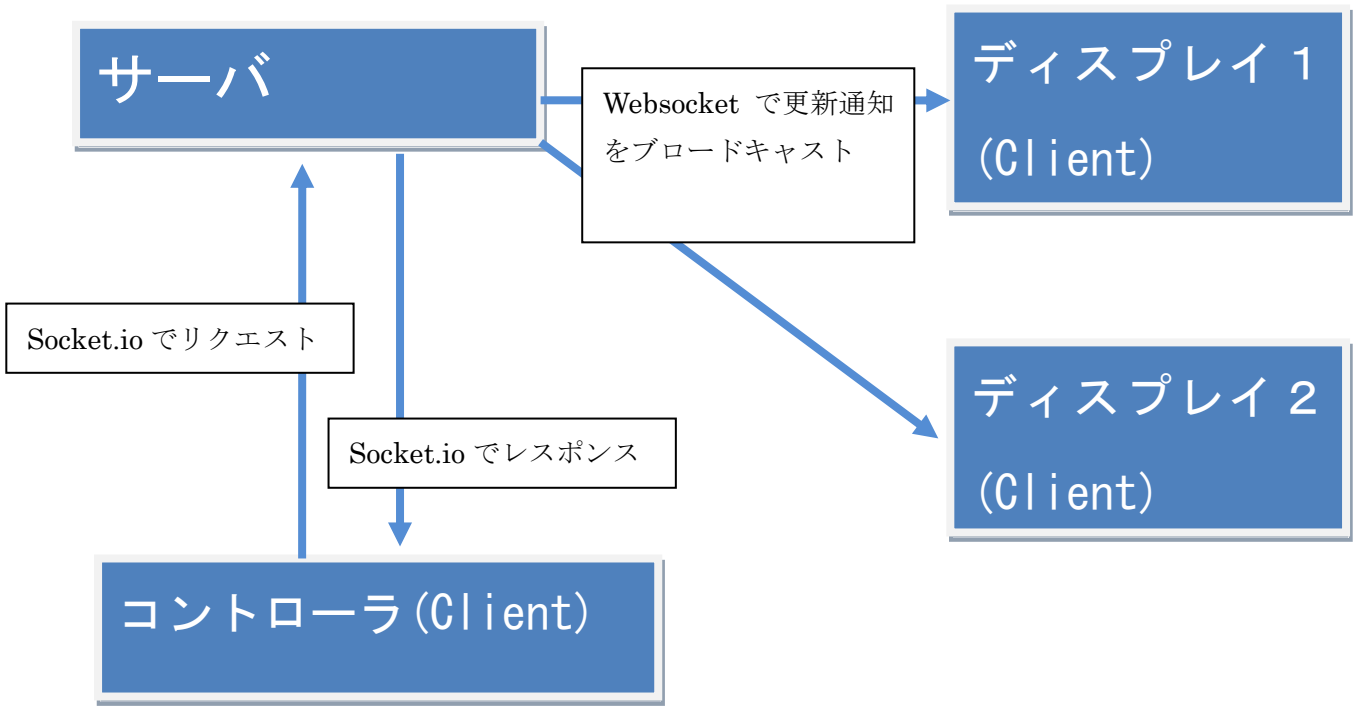
ディスプレイは，コントローラで設定したコンテンツを表示する．HTML5 の機能である websocket を用いてサーバと通信している．ディスプレイのイメージを以下に示す．



2. 通信用 API の仕様

クライアントサーバ間の通信についての仕様を記載する.

サーバではクライアントから websocket または socekt.io でメタバイナリを受け取り，メタデータに記載されているコマンドによって処理を行う．処理を実行した後，クライアントに対して，レスポンスを含んだメタバイナリを websocket または soceket.io にて送信する.



コントローラからリクエストを送ったときの通信の流れ

データの更新が発生した際は，更新通知をクライアントに対してブロードキャストする.
ディスプレイは，更新通知を受け取ると，サーバへコンテンツ/ウィンドウ情報取得リクエストを送り，コンテンツ/ウィンドウ情報を取得する.

また，ディスプレイ新規表示時には，ディスプレイからサーバへ，ディスプレイ登録リクエストを発行し，自身のディスプレイを登録する.

2.1 メタバイナリ

クライアントサーバ間でやり取りするデータである，メタバイナリフォーマットのフォーマットを以下に示す.

ヘッダ	"MetaBin:"	8byte
バージョン	UInt32	4byte
メタデータサイズ	UInt32	4byte
メタデータ	Ascii 文字列 (JSON)	メタデータサイズ byte
コンテンツデータ	Binary など	残りの byte

コンテンツデータは，具体的に以下の値が入る.
メタデータの type=text の場合：UTF8 文字列
メタデータの type=url の場合: URL エンコードされた文字列
メタデータの type=image の場合: 画像ファイルのバイナリ

バージョンは，現在常に 1 が入る.

2.2 リクエスト/レスポンスコマンド

サーバで受け付けているコマンドの一覧を以下に示す.

サーバへのリクエストコマンド	サーバからのレスポンスコマンド	実行内容	更新通知
reqAddContent	doneAddContent	コンテンツを追加します	update
reqGetContent	doneGetContent	コンテンツを追加します	無し
reqGetMetaData	doneGetMetaData	メタデータを取得します	無し
reqDeleteContent	doneDeleteContent	登録されているコンテンツを削除します	update
reqUpdateContent	doneUpdateContent	コンテンツを更新します	updateTransform
reqUpdateTransform	doneUpdateTransform	コンテンツのメタデータを更新します	updateTransform
reqAddWindow	doneAddWindow	ウィンドウを追加します	updateWindow
reqDeleteWindow	doneDeleteWindow	ウィンドウを削除します	update
reqGetWindow	doneGetWindow	ウィンドウを取得します	無し

reqUpdateWindow	doneUpdateWindow	ウィンドウのメタデータを更新します	updateWindow
reqUpdateVirtualDisplay	doneUpdateVirtualDisplay	仮想ディスプレイのメタデータを更新します	updateWindow
reqGetVirtualDisplay	doneGetVirtualDisplay	仮想ディスプレイのメタデータを取得します	無し
reqShowWindowID	doneShowWindowID	ウィンドウ ID を画面に表示するリクエストです。全ての Display に更新通知 showWindowID が送信されます。	showWindowID

2.3 更新通知コマンド

サーバから送信される更新通知コマンドの一覧を以下に示す.

サーバからの更新通知コマンド	通知内容
update	指定の ID または全てのコンテンツ/ウィンドウのデータが更新されたことを通知します。
updateTransform	指定のIDまたは全てのコンテンツ/ウィンドウのメタデータが更新されたことを通知します
updateWindow	指定の ID または全てのウィンドウが、追加または更新されたことを通知します。
showWindowID	ウィンドウ ID を表示する必要があることを通知します。

3. データ構造について

本システムは，データベースとして redis を使用しており，サーバによって受け付けたコンテンツやウィンドウ情報を保存している．以下に，DB のデータ構造を示す．

キー	内容	意味
tiled_server:s:default:virtual_display	splitX	仮想ディスプレイ全体の x 方向分割数
	splitY	仮想ディスプレイ全体の y 方向分割数
	orgWidth	仮想ディスプレイ全体の幅
	orgHeight	仮想ディスプレイ全体の高さ
tiled_server:s:default:content:[コンテンツ ID]	バイナリまたはテキストデータ	コンテンツの実データ
tiled_server:s:default:metadata:[コンテンツ ID]	id	コンテンツ ID
	type	コンテンツの種類
	posx	x 座標
	posy	y 座標
	width	幅
	height	高さ
	orgWidth	初期幅
	orgHeight	初期高さ
	zIndex	z インデックス
	visible	表示状態
	mime	mime タイプ
tiled_server:s:default:window:[ウィンドウ ID]	id	ウィンドウ ID
	type	window
	posx	x 座標
	posy	y 座標
	posx	x 座標
	posy	y 座標
	width	幅
	height	高さ
	orgWidth	初期幅
	orgHeight	初期高さ
	visible	表示状態
tiled_server:s:tiled_server:sessions	default	セッション ID

4. コントローラについて

コントローラページでは，コンテンツとディスプレイに対して各種操作を行うページである．具体的には，コンテンツの登録，削除，移動，拡大縮小，ディスプレイの削除，移動，拡大縮小，仮想ディスプレイ全体の大きさの更新，仮想ディスプレイ分割数の変更を行うことができる．

4.1 コンテンツの登録について

コントローラにて，画像ファイル，テキスト，テキストファイル，URL をコンテンツとして登録することができる．画像ファイルは jpg，gif，png，bmp 形式に対応している．また，URL はサーバにて png 形式の画像としてレンダリングされて登録される．

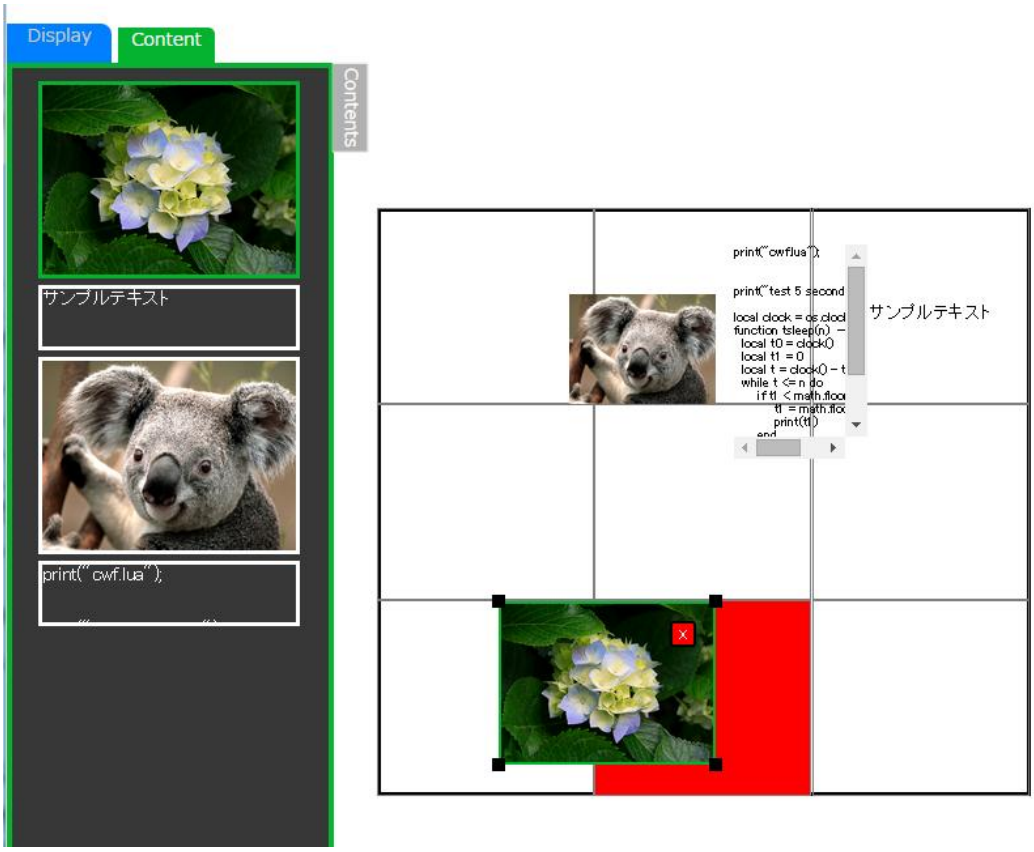
コンテンツの種類	形式	備考
テキスト	文字列	
テキストファイル	txt	
画像	jpg, gif, png, bmp	
URL	png	サーバサイドレンダリング

4.2 仮想ディスプレイ全体の設定について

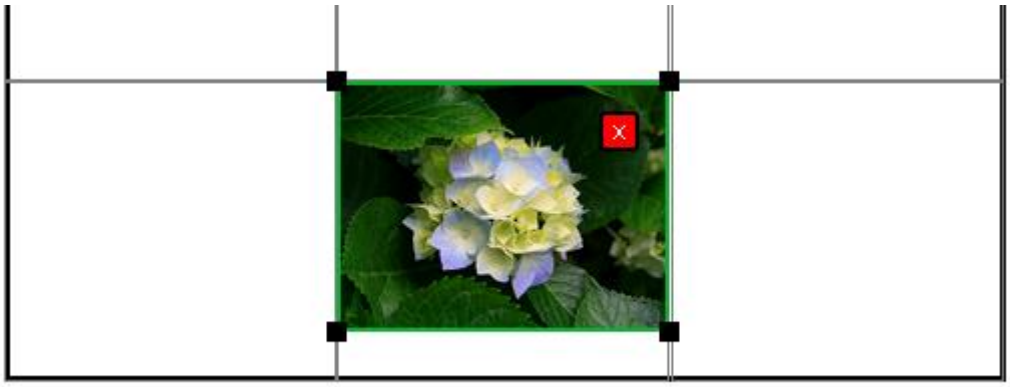
仮想ディスプレイ全体の設定として，仮想ディスプレイ全体の幅，高さ，分割数を設定することができる．

4.3 コンテンツ/ディスプレイの設定について

登録したコンテンツ及びディスプレイは，仮想ディスプレイを表す矩形上に自由に配置し，移動，拡大縮小を行うことができる．また，スナップ設定を有効にすることで，仮想ディスプレイの分割領域に対してフィットするように配置することができる．
分割した領域に対してスナップ配置しているイメージを以下に示す．



分割配置した領域に対してスナップ中の状態



スナップ後の状態

5. ディスプレイについて

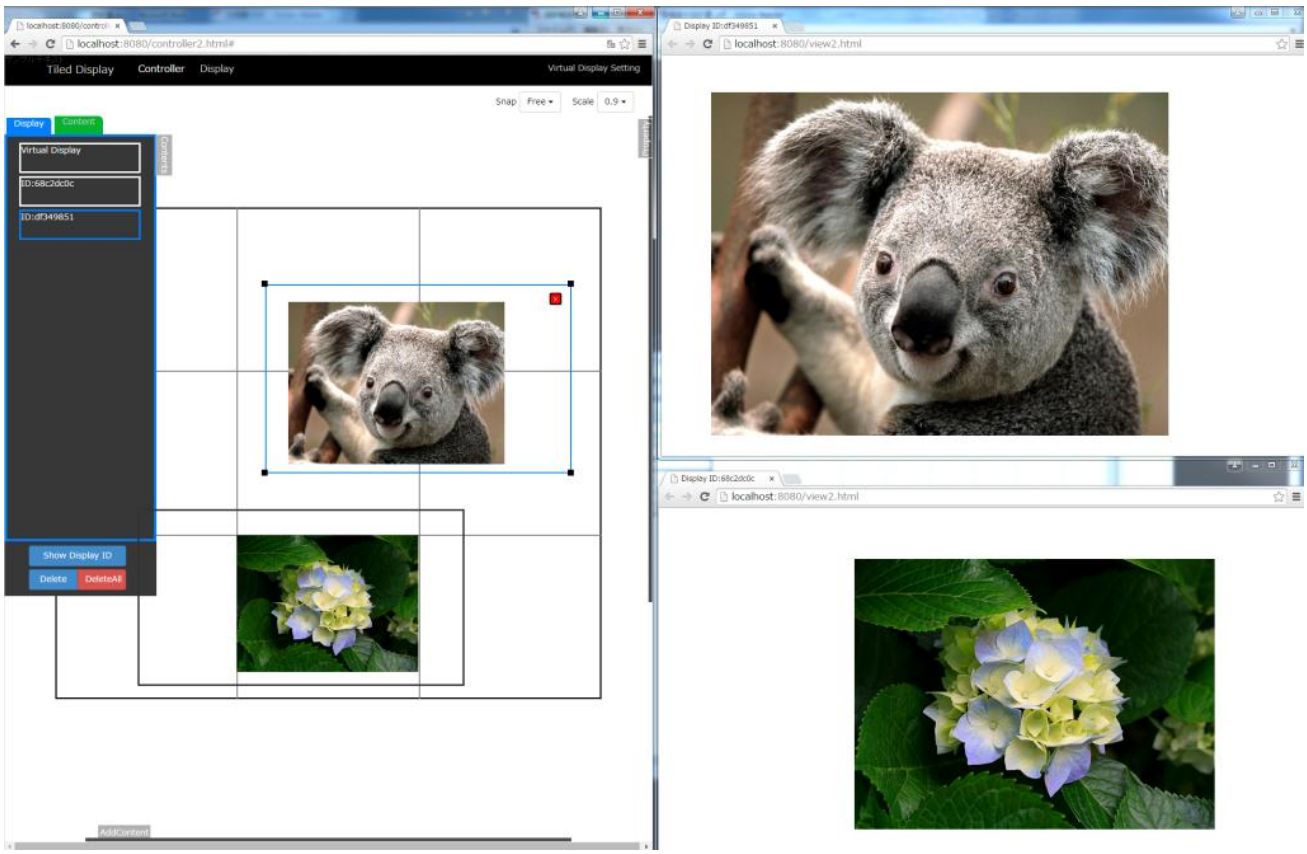
ディスプレイページでは，コントローラで操作した結果のコンテンツを表示することができるページである．操作は全てコントローラより行う仕様となっている．

5.1 ディスプレイの登録について

サーバが起動した状態でディスプレイページを開くと，websocket の API を用いて自動的にサーバに登録される．登録後は，コントローラにてコンテンツと同様に配置，移動することができ，ディスプレイページを閉じると自動的に登録が解除される．

6. 画面操作について

操作中の画面サンプルを以下に示す．



ディスプレイ操作のイメージ
（左がコントローラ，右がディスプレイ）

画面操作については，別紙「利用説明書」を参照の事．

7. 動作環境について

Windows, Linux, MacOSX の GoogleChrome, Firefox 及び Safari で動作する..
各ブラウザの最新のバージョンにて動作確認を行う.