15　複数人で遊べるペイントツール製作

12 番　　髙橋　流星

指導教員　佐々木　建

1.　はじめに

私は今までペイントソフトを使う機会があり、どのような機能があればより楽しく描くことができるかなどを考えることがあった。そこで自分でも面白いと感じるものを製作できないかと思い、卒業研究でペイントツールの製作をすることにした。

今回は単純に 1 台のパソコンでのペイントツールではなく、複数台のパソコンでキャンバスを共有するペイントツールを製作し、そのツールを通して、JavaScript での描画・通信処理や HTML5 のデザインの有用性、チャット機能についての研究を進めながら知識を深める。

2.　研究概要

2.1　目的

複数台のパソコンで 1 つのキャンバスを共有し、それに絵を描くことができるペイントツールの製作を目的とする。

ペイント機能として必要なペン・消しゴムなどの実装をはじめ、他にもコミュニケーションを取るためのツールとしてチャット機能を付加した。

2.2　開発環境

開発環境は以下に示す通りである。

表 1　開発環境

OS	Windows10
ブラウザ	Google Chrome
言語	JavaScript,Node.js,HTML5

2.3　Node.js について

本来、JavaScript はユーザーのブラウザで動作するプログラミング言語であるが、Node.js はサーバー側で動作する JavaScript のひとつである。

ノンブロッキング I/O を採用しており、さらに Socket.io というリアルタイムな通信を実現するライブラリを扱うことができ、これを利用することで、今回製作することになったキャンバスの共有機能やチャット機能を実現することができた。

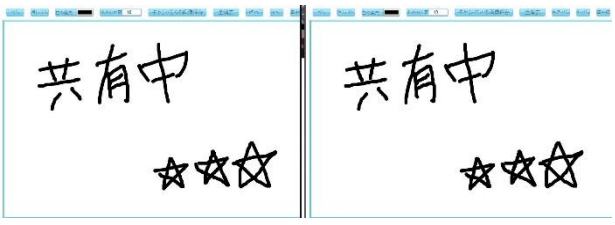
3.　実装機能説明

3.1　キャンバスの共有化

基本となるキャンバスの共有化の機能は node.js と socket.io の技術を用いた JavaScript を使うことにより、実現することができた。

このペイントツールにはサーバー側とクライアント側（複数）のそれぞれにキャンバスがあり、クライアント側で描画された内容は、一旦サーバー側に送信され、さらにその内容をアクセスしている全クライアントに再送信するという技術を使うことにより、１つのキャンバスの共有化を実現している。

図 1　キャンバスの共有



3.2　ペイント機能の実装及び描画の共有化

ペイント機能として実装されている機能は以下の通りである。

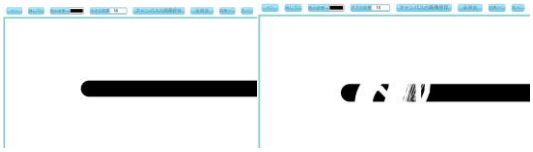
① ペン・消しゴム機能

ペイント機能の『ペン』を選択することで、キャンバス上でマウスをドラッグ操作したときにキャンバスに線が描画される。また、ペイント機能の『色』の変更や『サイズ』の指定などにより色を変えたり、ペンの太さを変えたりしての自由な線を描くことができる。

ペイント機能の『消しゴム』は基本的な操作はペンと同様であり、消しゴムを選択している間は色の変更は反映されず、キャンバス上に描かれている内容を消す処理のみ行う。

これらの描画等の動作は複数のクライアントに反映される。

図 2　ペン機能と消しゴム機能

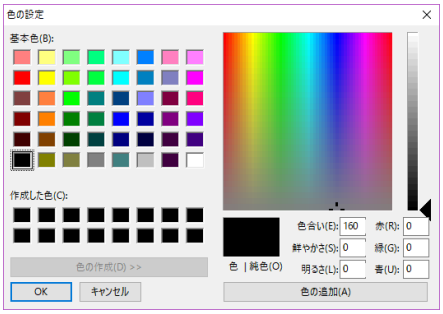


② ペンの色変更のための『カラーピッカー』

カラーピッカー（図 3）はペンの色を変更するための機能である。色は RGB や色合い・鮮やかさ・明るさを自由に選択・作成し、幅広い色を使用することができる。

一度選択した色は、『作成した色』欄に 16 色まで保存でき、パレットとしての機能（再利用が可能）も担ってくれる。

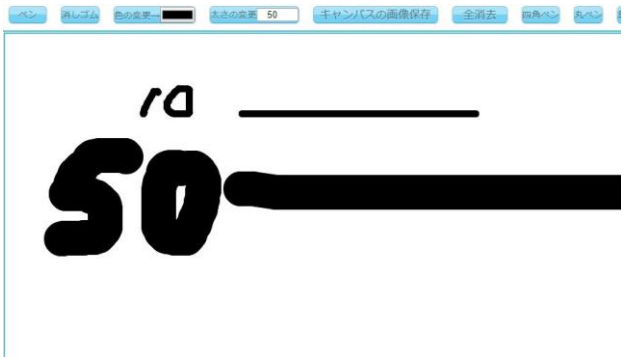
図 3　カラーピッカー



③ ペンサイズ変更のためのサイズ指定欄

ペイント機能のペン・消しゴムのサイズをクライアントごとに設定するための機能である。初期値は『10』であるが、『1～100』の範囲内でサイズを自由に設定することができる。

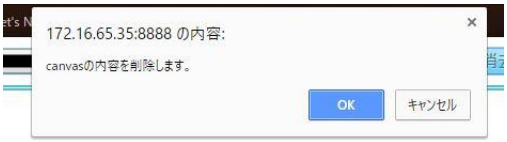
図 4　ペンサイズの変更



④ キャンバスの全消去機能

共有キャンバスを初期状態の何も描画されていない状態に戻すための機能である。接続しているユーザーが自由に使うことができる。使用する際に一度確認用のホップアップメッセージが出るようになっており、消去すると全クライアントのキャンバスが初期状態になる。

図 5　全消去確認用のホップアップ

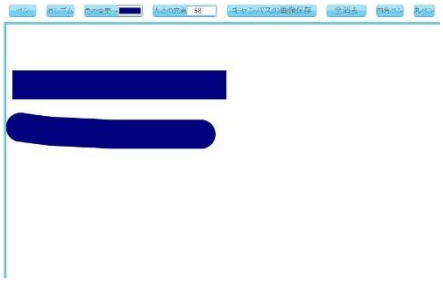


⑤ 四角ペン・丸ペン切り替え機能

ペイント機能のペンの形を変えるための機能である。丸型と四角型の 2 種類があり、それぞれを選択することでペン･消しゴムの形が変更される。

初期の状態では『丸型』の状態となっている。

図６　丸ペンと四角ペンによる描画



3.3　キャンバスの画像保存機能

共有キャンバスの上にある『キャンバスの画像保存』ボタンを押すことでキャンバスの描画内容を Jpeg 形式で保存することが可能である。

もともとキャンバス上で『名前を付けて画像を保存』は可能であるが、この機能は素早く画像を保存するための『スクリーンショット』的な意味合いが強い。

3.4　チャット機能の実装

共有キャンバスのすぐ下のスペースにクライアント間でのコミュニケーションを円滑にするためにチャット欄を設置した。

技術としてはキャンバスの共有の基礎と同様 Node.js を使用した通信を行っている。

機能としてはチャット機能としては基本的な、メッセージの送受信、ユーザーネームの設定のみだ。一度に送れるメッセージは 50 文字までと制限しており、それ以上を打ち込んだ状態で送信した場合は 50 文字目より先の内容は切れた状態で内容が表示されない。

他には、メッセージの内容を何も入力してない状態での送信は行うことができず、ユーザーネームが未設定の場合は『名無しさん』と自動表示されるようの設定をした。チャット欄で入力された文字がタグなどの認識をされないように HTML5 のタグ認識される文字記号などは認識されないように設定した。

4.　ＣＳＳによるデザインの変更

デザインを簡素なものから青を基調とした目に優しく視覚的にもわかりやすいデザインに変更した。初期の状態ではツール部分がリストタグで構成していたが、ボタンに変更することでどのツールを選択したのかをよりわかりやすくした。デザインも他のパーツと一貫性を持たせるものに変更した。キャンバス部分の縁取りやチャットのメッセージログの欄も他と統一したデザインにすることでまとまりがでるように変更した。

図７　ボタンのデザイン

図８　チャットメッセージ欄

5.　検証

今回、Node.js を用いた JavaScript で製作したキャンバス共有型のペイントツールを通し、よかった感じた点と課題点について検証する。

良かったと感じた点は、まず、ブラウザさえインストールされていればクライアント側で用意するものがほとんどない点である。厳密にいえば Google Chrome での動作を前提としているのでブラウザの準備はいるが、それ以外に必要なものは何もなく、サーバーが起動している限り誰でもすぐに利用できるというのは非常に扱いやすく良いと感じた。動作状況も酷いラグが生じるようなことも無く、絵を描く際に妨げになるようなことはないように感じた。

『チャット機能』の実装の際にも、キャンバスの共有と同じ仕組みを利用している点が多く、比較的楽に実装することができた。これによりクライアント間でのコミュニケーションが取りやすくなった。

HTML5 を使っているため、CSS などでデザインにも幅広い拡張性を持っている。機能をさらに

増やしてもデザインによっては、より使いやすいペイントツールを作り上げていけるように感じた。

問題点については、製作段階の初期の頃にあったもので『ツールの操作まで全クライアントと共有してしまう』という動作があった。原因として考えられるのは、使用している『ペン』が全クライアントで 1 つのものを使用していることが原因と思われる。それの改善策として各クライアントに別々のペンを与えることで他人のツール操作の影響を受けることを防ぐことができた。しかし、クライアント間で特定の順番に操作した際に影響を受けることが多々あり発見しだい修正はしたが、まだ隠れた問題点があるのではないかと懸念される。

他にはペイント機能の充実化を目指したものの優れた機能を持つブラシなどの実装ができなかったことである。ペンサイズの変更、色の指定、ペンの形を変更などの基本的な操作はできたが、塗りつぶしや、キャンバス上の色を抽出するスポイト機能、エアブラシなどの絵を描くのに特化した機能を実装することはできなかった。その理由の多くがクライアント側で描画できたとしてもサーバー側にそれらの描画情報を処理させることが難しく実現できなかったことである。これらは今回残ってしまった大きな課題である。

他に難点を挙げると途中から参加したクライアントにはそれ以前の描画内容は反映されていない点があるが、描画情報をサーバーサイドに保存してあるので他の人が入室したタイミングでその情報を送ることができれば改善するのではないかと考えている。

6. おわりに

キャンバス共有型のペイントツールの製作を通して、使用した言語に対する理解を深めつつ、様々な課題点が見えた。

あまり触れることのなかった JavaScript やそれに加えて初めて触れる技術である Node.js を使ってみて、各ツールの共有を一つ一つ行うことの大変さがとても感じられた。一つのツールを応用することで容易に実装できる機能もあれば、新たに実装するうえで描画処理や通信処理をどう解決していくかを考える必要があるものもあった。
そのために、一つツールを増やすのに何日も掛かることもあれば、実装ができなかった機能も多くあり、それらの処理をどうやって解決することができるかを更に考えつつ、実装するにはどのような技術が必要になるのかを展開させることが重要であると感じた。

しかし、実装できた機能の中には、『カラーピッカー』を使用した色の変更などは自分が考えたものをそのまま実装できた形になっている。もともとは何色か使いやすい色を用意しておくだけの予定だったが、この機能を知り実装するために試行錯誤を繰り返し、実装にこぎつけた。実装できなかったツールも多くある反面、実現できた機能も多々あり、徐々に Node.js や JavaScript に対しての理解を深めることができたのが実感した。

7. 参考文献・参考サイト等

YoheiM.NET
(http://www.yoheim.net/)
Node.js から Socket.IO を使うための事前知識
(https://qiita.com/ij_spitz/items/2c66d501f29bff3830f7)
LIG
(https://liginc.co.jp/)
ONE-RUN
(https://st40.xyz/one-run/3/all/)
Socket.io とは何か?リアルタイム Web アプリケーションを実現する技術をまとめてみた
(https://qiita.com/masarufuruya/items/2bd5dfe03096057af63f)
kigiroku
(https://kigiroku.com/frontend/canvas_draw.html)
q-Az
(https://q-az.net/canvas-drawing-pad/)
while(is プログラマ)
(http://am-yu.net/2014/02/13/canvas-pbbs-fill/)