

2023 年度 卒業論文

カラーユニバーサルデザイン
(CUD) の学習支援ツールの開発

指導教員 須田 宇宙 准教授

千葉工業大学 情報ネットワーク学科
須田研究室

2032116 氏名 永溝 柊介

提出日 2024 年 1 月 19 日

目次

1	緒言	5
2	色覚異常とカラーユニバーサルデザイン（CUD）について	6
2.1	色覚異常について	6
2.2	色覚異常の見え方と CUD について	6
2.3	CUD の普及に向けた活動	7
3	CUD 教育について	8
3.1	概要	8
3.2	CUD 教育における課題	8
4	開発した学習ツールについて	9
4.1	開発の目的	9
4.2	対象とする学習者	9
4.3	想定している利用方法	9
4.4	利用環境	9
4.5	学習の流れ	10
4.6	学習ツールの構成	12
4.6.1	第 1 部 文字の強調	12
4.6.2	第 2 部 色の組み合わせ	13
4.6.3	第 3 部 グラフの作成	14
5	実装方法	15
5.1	概要	15
5.2	サーバサイド	15
5.2.1	処理内容	15
5.2.2	使用している言語	15
5.3	DB	16
5.3.1	DB の設計	16
5.3.2	DB の詳細	16
5.3.3	使用した DB	18
6	検証	19
6.1	実施内容	19
6.2	評価項目	19
7	結果	20

8	考察	20
9	結言	21
	謝辞	22
付録 A	作成したプログラム	24
A.1	app.js	24
A.2	layout-1a.ejs	53
A.3	func.js	57

表目次

1	テーブル一覧	16
2	text テーブル	16
3	color テーブル	17
4	tc テーブル	17
5	font テーブル	17
6	eva テーブル	18

図目次

1	P型色覚者の見え方	6
2	改善後のグラフの見え方	7
3	サイトマップ	11
4	1部の学習画面例	12
5	2部の学習画面例	13
6	3部の学習画面例	14
7	開発したシステムの構成	15
8	CUD 評価結果	20

1 緒言

遺伝子の異常により通常と色の見え方が異なる色覚を持つ色覚異常者は、男性で 5 %，女性で 0.2 % 存在する [1]。色覚異常には、主に赤を感じづらい P 型色覚，緑を感じづらい D 型色覚，青を感じづらい T 型色覚がある。色覚の多様性に配慮し，より多くの人に伝わりやすいデザインとしてカラーユニバーサルデザイン（CUD）がある。

学生や社会人になると，ゼミや業務等で資料を作成する機会が増加する。そのため，CUD の講習では資料作成を実践し評価する機会が設けられている。しかし，講習を受けた学生の 20 % が CUD に配慮していない資料を作成した [2]。先行研究では，講習内容を即座に理解することが困難なためと考察されている。そのため，CUD に配慮した工夫を一つ一つ実践する機会が必要だと考える。

学習者は別色覚での見え方を確認することで CUD をより理解できると考えられるが，現状では別媒体のシミュレータを使う必要がある。また，評価者にとって学習者の作成物の CUD 適合度を即座に判断することが難しく，評価や改善案を即座に提示することは困難である。そのため，学習の際は，別色覚での見え方を即座に確認でき，自動で評価・改善案の提示を行う環境が望ましい。

そこで本研究では，これらの環境を持ち，CUD に配慮した工夫を項目ごとに実践できる学習ツールの開発を行う。そして，開発した学習ツールを学習者が利用することで CUD に配慮した資料を作成できるか検証することを目的とする。

2 色覚異常とカラーユニバーサルデザイン（CUD）について

本章では、色覚異常と CUD について説明する。

2.1 色覚異常について

人間の目には、赤、緑、青の 3 種類をそれぞれ感じる機能を持つ錐体があり、その錐体によって色を識別している。この錐体の一部が十分に機能しないために、通常とは色の見え方が異なる色覚となる場合がある。このような色覚を持つ色覚異常者は、日本人の場合、男性で 5 %、女性で 0.2 % 存在する。

色覚異常者は、主にどの色を感じづらい色覚かによって区別される。主に赤を感じづらい色覚である P 型色覚は、全体の約 25 % を占める。主に緑を感じづらい色覚である D 型色覚は、全体の約 75 % を占める。主に青を感じづらい色覚である T 型色覚は、全体の約 0.02 % を占める。

2.2 色覚異常の見え方と CUD について

図 1 は、同じグラフに対しての色覚による見え方の違いを表している。左は通常の色覚での見え方、右は P 型色覚での見え方を表している。線 1 は緑色、線 2 は赤色であり、通常の色覚では区別がつきやすい。しかし、P 型色覚ではどちらも茶色に近い色に見えるため区別がつきづらい。このように、通常の色覚者には伝わりやすい情報であっても、色覚異常者には伝わりづらくなっている場合がある。

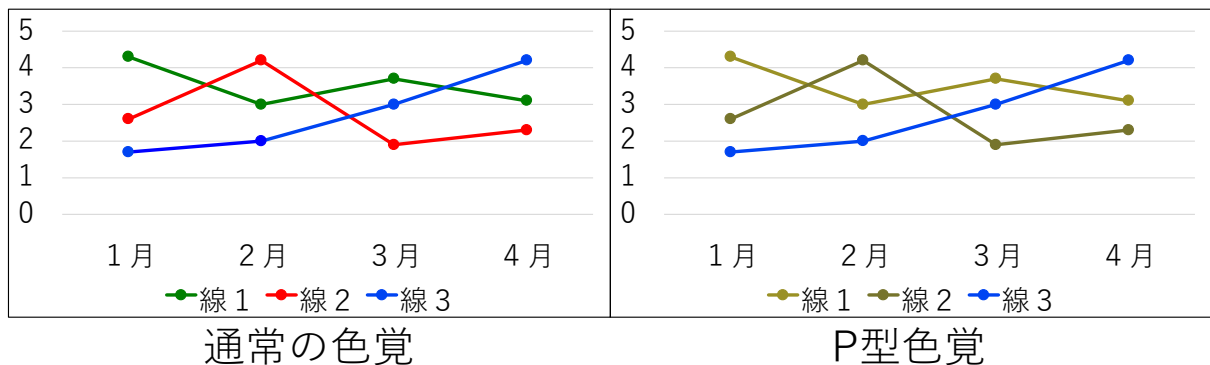


図 1 P 型色覚者の見え方

色覚の多様性に配慮し、より多くの人に伝わりやすい配色やデザインとして、CUDがある。図2は、図2のグラフをCUDに配慮し、改善した例である。図1と同様、左は通常の色覚での見え方、右はP型色覚での見え方を表している。改善後のグラフでは、線2を橙色にし、点線にすることで区別が付きやすくした。このように、色覚異常者にも正確に伝えるためには、CUDに配慮する必要がある。

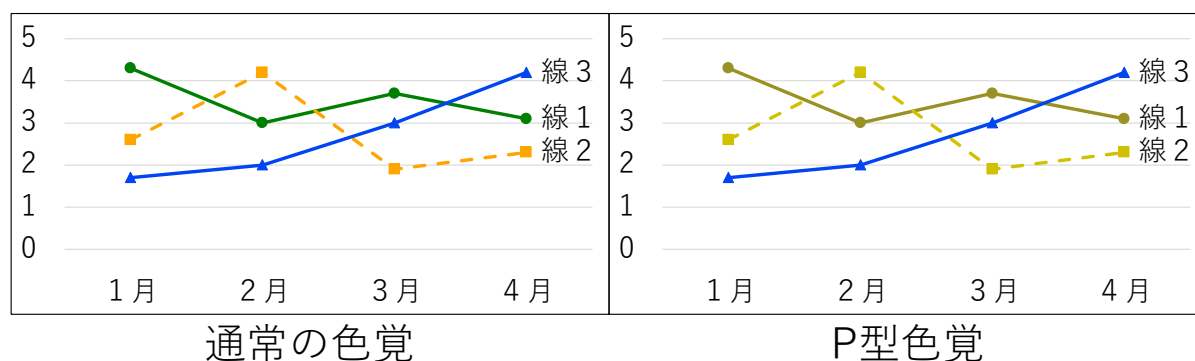


図2 改善後のグラフの見え方

2.3 CUDの普及に向けた活動

CUDの普及に向けた活動として、東京都は「カラーユニバーサルデザインガイドライン」を作成している[3]。また、大阪府は「色覚障がいのある人に配慮した色使いのガイドライン」を作成している[4]。これらのように、一部の自治体がCUDの認知度を高めることを目的とし、CUDのガイドラインを作成している。

また、一部の大学、専門学校や会社では、講義や研修等を通してCUDに関する講習を行っている。

3 CUD 教育について

本章では、CUD 教育の内容と現状の課題について説明する。

3.1 概要

一般的に、色覚異常に関する講習では、主に色覚異常の見え方や色覚異常者にも伝わりやすいデザインである CUD についての説明を行う。CUD 教育では、学習者が色覚異常に関する知識を得るだけでなく、資料等を作成する際に CUD に配慮した工夫を取り入れることができるようになることが求められる。そのため、講義形式での説明に加え、学習者に CUD に配慮したデザインを体験してもらう場面が存在する。

大阪医療福祉専門学校での CUD に関する授業では、色選びの体験ワークを行った [5]。まず、学習者にゴミ分別に関する図の配色を考えさせた。そして、色覚異常者の見え方を体験できるフィルター式眼鏡を掛けて、選択した色が色覚異常者でも区別できる配色となっているか確認させた。このように、実際に CUD に配慮した図の作成を体験させる授業が行われた。

また、東京女子大学での講義では、色覚異常に関する説明の後に CUD に配慮した資料作成を実践する機会を設けていた [2]。この講義では、まず、東京都が作成したガイドライン等を参考に、色覚異常や色覚異常者にも伝わりやすい資料作りとして色のバリアフリーについての説明を行った。その上で、受講者に色覚異常者に配慮した発表資料を作成させる等、CUD を取り入れた資料作成を実践する機会を設けている。

これらのように、学習者に資料等を作成する際に CUD に配慮した工夫を取り入れることができるようになることを目的とした教育が行われている。

3.2 CUD 教育における課題

受講者に資料作成を実践させた授業に関する先行研究によると、講習を受けた学生の 20 %が CUD に配慮していない資料を作成した [2]。先行研究では、初めて色覚異常の知識に触れる学生にとって複数の別色覚にいきなり配慮することは難しく、CUD を直ぐに理解して取り入れることは困難であるためと考察されている。そのため、資料作成を実践する前段階として、CUD に配慮した工夫を一つ一つ実践する機会が必要だと考える。

学習者は別色覚での見え方を確認することで CUD をより理解できると考えられる。別色覚での見え方の確認は、シミュレータを利用することで可能となるが、現状では資料作成ツールとは別媒体のシミュレータを通す必要がある。また、評価者にとって学習者の作成物の CUD 適合度を即座に判断することは難しく、評価や改善案を即座に提示することは困難である。そのため、学習の際は、別色覚での見え方を即座に確認でき、自動で評価・改善案の提示を行う環境が望ましい。

4 開発した学習ツールについて

4.1 開発の目的

これらの問題を受け、学習者が CUD に配慮した工夫を項目ごとに実践でき、自動で別色覚での見え方と評価・改善案を表示する学習ツールの開発を行った。

4.2 対象とする学習者

CUD に関する講習等では、色覚異常や CUD に関する説明を受けた後に、色覚異常の見え方や CUD に配慮したデザインを体験する場面がある。そのため、色覚異常や CUD についてある程度学習している者を対象とした。

4.3 想定している利用方法

本学習ツールは、実際の資料作成の際に CUD に配慮した工夫をどのように取り入れるかを学習させることを目的としている。そのため、CUD の講習等で CUD に配慮した工夫の取り入れ方を教育する場合や、CUD について学習した者が実際に資料作成を行う前段階として、本学習ツールを利用することを想定している。

4.4 利用環境

本学習ツールは、iPad や PC での使用を想定して、Web 上で動作するよう開発した。

4.5 学習の流れ

例として、第1部1項目目の内容である文字の強調における配色について学習する際の学習の流れについて説明する。この例における学習画面の遷移を図3に示す。

この項目では、まず文字の強調に使用する色を選択する。

①選択肢から色を選択すると、下の文の一部の色が変化する。赤が選択されている場合は、文の一部が赤色になっている。

②評価ボタンが押されると、評価画面が表示される。評価画面で別色覚での見え方と評価・改善案が表示される。

③この項目において、赤、緑、青は一部の色覚では黒との区別が付きづらい色に見えるため、文字の強調には適さない。適さない色を選択した場合は、色を再選択する。

④橙、黄緑、空色は赤、緑、青に比べどの色覚でも黒との区別が付きやすい色となっている。適した色を選択した場合、他の適する選択肢を試していない場合は別の色を再選択する。

⑤適する色を全て試した場合は、次の項目に進む。

この一連の流れを項目ごとに行い、CUDに配慮した工夫を一つ一つ学習する。

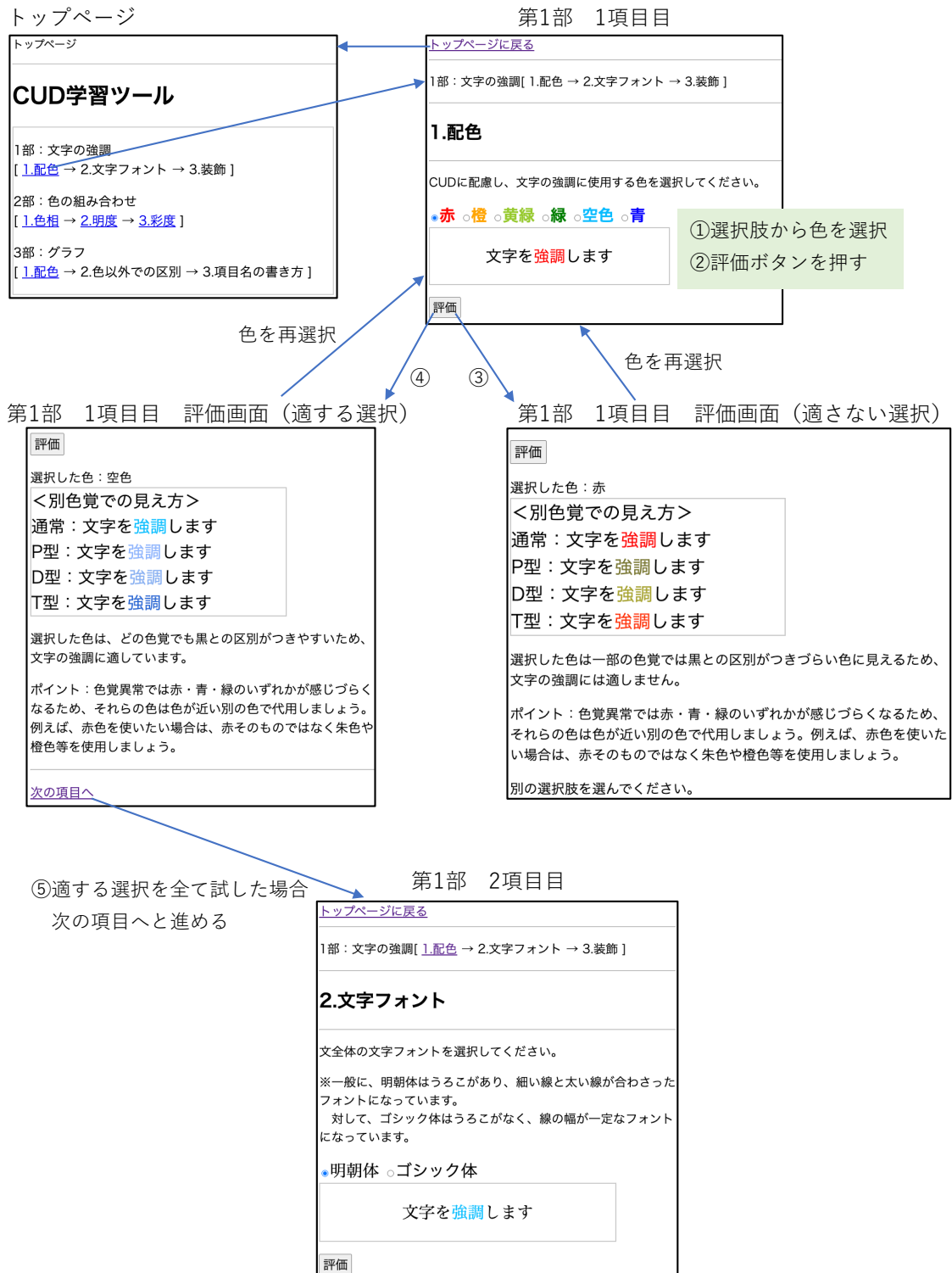


図3 サイトマップ

4.6 学習ツールの構成

本学習ツールは3部構成となっている。

4.6.1 第1部 文字の強調

1部では文字の強調を行う際の工夫について学習する。項目として、配色、フォント、装飾の順に学習する。図4は配色の学習画面である。

配色では、強調に使用する色を選択する。選択肢は、赤、橙、黄緑、緑、空色、青の6種類である。色覚異常は赤、青、緑のいずれかが感じづらくなるため、それらは黒との区別がつきづらい色に見える場合がある。それらの色を選択した場合は文字の強調には適さないという評価とした。

フォントでは、選択肢から文全体のフォントを選択する。選択肢は明朝体とゴシック体の2種類である。明朝体には游明朝体やヒラギノ明朝等、複数の明朝体フォントが存在する。これは、ゴシック体も同様である。そのため、一概にどちらかが適さないフォントであると断定することは難しい。一般に、明朝体はうろこがあり、細い線と太い線が合わさったフォントとなっている。対して、ゴシック体はうろこがなく、線の幅が一定なフォントになっている。そこでこの学習ツールでは、これらの特徴からCUDの観点で適しているか判断した。線の一部が細くなっているフォントは色面積が小さくなり、色による判別がつきづらくなるため、色による文字の強調を行う際には適さないという評価とした。

装飾では、選択肢から選択することで、強調する文字を変化させる。選択肢として、「アンダーラインを引く」と「太字にする」がある。これらを選択することで、強調する文字を変化させる。この項目では、色以外で強調を伝える方法を理解させることを目的としているため、これらの選択肢を選択して試した場合に、次の項目に進めるようにした。

1部：文字の強調[1.配色 → 2.文字フォント → 3.装飾]

1.配色

CUDに配慮し、文字の強調に使用する色を選択してください。

☒赤 ☐橙 ☐黄緑 ☐緑 ☐空色 ☐青

文字を強調します

評価

図4 1部の学習画面例

4.6.2 第2部 色の組み合わせ

2部では、色を組み合わせる際の工夫について学習する。項目として、色相、明度、彩度の順に学習する。図5は色相での学習画面である。2部では、全ての項目で、指定された背景色に対する文字の色を選択するという学習の流れとなっており、選択肢が項目によって異なっている。それぞれ扱っている色のうち1色が背景色となっており、それ以外が選択肢となっている。背景色は「背景色を変更する」から変更することができる。

色相では、赤、橙、黄色、黄緑、緑、空色、青、紫の8種類を扱っている。色相とは、赤、緑、青等の色味の種類のことである。色相で別となっている2つの色の区別がつきづらい場合がある色覚が存在するため、別色覚での見え方を確認した際に、区別がつきづらい配色となる場合は適さないという評価とした。

明度では、明るめの橙、暗めの橙、明るめの緑、暗めの緑の4種類を扱っている。明度とは、色の明るさのことである。明度が高いほど白に近づき、低いほど黒に近くなる、2色を組み合わせる場合は、明度に差をつけることで色覚によらず区別しやすい配色となるため、背景色と同じ明度の色を選択した場合は適さないという評価とした。

彩度では、赤、ピンク、青、水色の4種類を扱っている。彩度とは、色の鮮やかさのことである。彩度が高いほど明瞭な原色に近づき、低いほど白や黒等の無彩色に近づく。彩度が低い色同士で組み合わせると、色による区別がつきづらい場合があるため、それらを組み合わせた場合は適さないという評価とした。

1.色相

＜背景色を変更する＞

[橙](#) [黄色](#) [黄緑](#) [緑](#) [空色](#) [青](#) [紫](#)

CUDに配慮し、背景色に対する中の文字の色を選択してください。

▶ 色相とは

● [橙](#) ○ [黄色](#) ○ [黄緑](#) ○ [緑](#) ○ [空色](#) ○ [青](#) ○ [紫](#)

カラーユニバーサルデザイン

図5 2部の学習画面例

4.6.3 第3部 グラフの作成

3部では、グラフを作成する際の工夫について学習する。この学習ツールでは、折れ線グラフを扱った。項目として、線の色、色以外での区別、項目名の書き方の順に学習する。図6は線の色での学習画面である。線の色では、グラフに存在する3本の線の配色を選択する。色の選択肢や配色に対する評価は、2部の1項目目である色相と同じである。

色以外での区別では、線の種類とマーカーの図形を選択する。線の種類は、実線に加え間隔の異なる破線2種類の合計3種類を扱っている。マーカーの図形は丸、三角、四角の3種類を扱っている。線ごとに別々の組み合わせを用いることで、色に頼らず線の区別がつくようになる。この学習ツールでは線の種類、マーカーの種類がそれぞれ3種類となっているため、それぞれで一つずつ選択していなかった場合は適さないという評価とした。

項目名の書き方では、項目名の表示方法を選択する。選択肢は、「別枠にまとめる」と「線の近くに表示する」の2種類である。項目名は、別枠にまとめて表示するのではなく、それぞれの線の近くに表示することで、色に頼らなくとも項目名と線の対応が伝わるようになる。そのため「別枠にまとめる」を選択した場合は適さないという評価とした。

1.線の色

CUDに配慮し、グラフの線に使用する色を選択してください。
※横軸の値が0（グラフ内の一番左）において、上から順番に線1、線2、線3とします。

線1： ●赤 ○橙 ○黄色 ○黄緑 ○緑 ○空色 ○青 ○紫
線2： ○赤 ○橙 ○黄色 ○黄緑 ●緑 ○空色 ○青 ○紫
線3： ○赤 ○橙 ○黄色 ○黄緑 ○緑 ○空色 ●青 ○紫

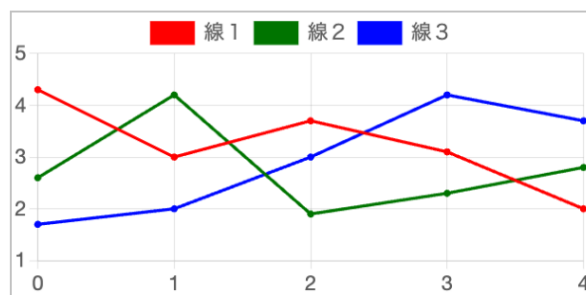


図6 3部の学習画面例

5 実装方法

5.1 概要

本学習ツールは、学習者が Web ブラウザを使用して学習する。

図 7 は開発したシステムの構成を表している。本学習ツールでは、クライアントサイドでの要求に対し、サーバサイドで処理を行い、結果を返している。

例として、学習者の選択した内容に対する評価を行う際は、学習者が選択した内容をサーバサイドに送っている。そして、サーバサイドにあるデータベース (DB) の情報から学習者の選択に対する評価等の情報を取り出し、クライアントサイドに返すことで、学習者側のブラウザに評価等を表示させる。

このように、本学習ツールでは、学習者の選択に対してサーバサイドで処理を行う。

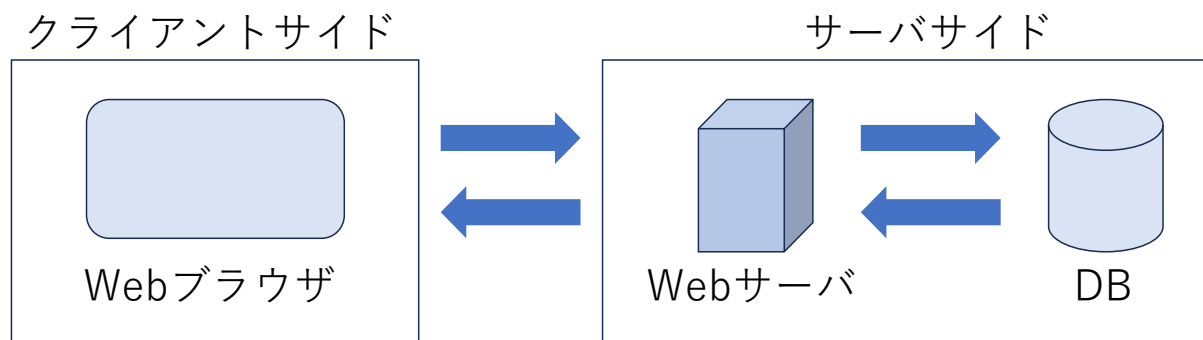


図 7 開発したシステムの構成

5.2 サーバサイド

5.2.1 処理内容

サーバサイドでは、クライアントサイドから送られた情報をもとに、学習者が選択した内容によって表示する評価等を変更する処理を行い、クライアントサイドに結果を返す処理を行う。

5.2.2 使用している言語

本学習ツールの開発には、Node.js のフレームワークの一つである Express.js を利用した。

Node.js は、サーバーサイドの JavaScript 実行環境の一つである。Node.js は大量なアクセスに強くリアルタイムの処理に適していること等から、Web アプリケーションの開発等によく使われる。サーバサイドで学習者の選択によって評価等の表示項目を変更する処理を行うために、サーバーサイドの実行環境である Node.js を使用して開発した。

Express.js は Node.js のフレームワークの一つである。Express.js を利用することで、Node.js を使用して Web アプリケーションを作成するために必要な機能を容易に実装することができる。Node.js を使用した開発を容易にすることを目的として、Express.js を利用した。

5.3 DB

DB には、項目ごとの選択肢や学習者の選択に対する評価に必要な情報等を持たせている。

5.3.1 DB の設計

開発したシステムの DB の設計を表 1 に示す。DB には、text テーブル、color テーブル、tc テーブル、font テーブル、eva テーブルの 5 つのテーブルがある。

表 1 テーブル一覧

テーブル名	内容
text	項目ごとに表示する文章
color	選択肢となる色の情報
tc	項目と選択肢の対応関係
font	選択肢となるフォントの情報
eva	2 色を組み合わせた際の評価

5.3.2 DB の詳細

表 2 は text テーブルの構成である。text テーブルには、項目ごとに表示する文章等の情報を格納している。このテーブルから、評価として表示する文章等を取り出して、学習者に提示している。id は項目の番号、item は項目名、que は学習者への指示、good は学習者の選択が適していた場合の評価、bad は適していなかった場合の評価、adv は CUD に配慮するためのアドバイスを文章として格納している。

表 2 text テーブル

カラム名	内容
id	項目番号
item	項目名
que	指示文
good	適した場合の評価
bad	適していない場合の評価
adv	アドバイス

表 3 は color テーブルの構成である．このテーブルには選択肢となる色の情報を格納している．別色覚での見え方を再現したカラーコード等が入っており，これらのデータを利用して学習者の選択に対して別色覚での見え方を表示している．id は番号，name は色の名前，ccode は色のカラーコード，pcode, dcode, scode は別色覚での見え方を再現したカラーコードを色覚ごとに格納している．

表 3 color テーブル

カラム名	内容
id	番号
name	名前
ccode	カラーコード（通常の色覚）
pcode	カラーコード（P 型色覚）
dcode	カラーコード（D 型色覚）
scode	カラーコード（T 型色覚）

表 4 は tc テーブルの構成である．このテーブルは，text テーブルと color テーブルの中間テーブルである．項目ごとに選択肢となる色が異なるため，項目と選択肢の対応関係を格納している．t-id は text テーブルの id，c-id は color テーブルの id を表す．

表 4 tc テーブル

カラム名	内容
t-id	項目番号
c-id	色番号

表 5 は font テーブルの構成である．このテーブルには，文字の強調について学ぶ際の選択肢となるフォントの情報を格納している．id は番号，face は CSS でフォントを指定する際の値，name はフォントの名前，gb は CUD に配慮したフォントであるかを 0 か 1 で格納している．

表 5 font テーブル

カラム名	内容
id	番号
face	値
name	名前
gb	評価

表 6 は **eva** テーブルの構成である．このテーブルには，2 色のカラーコードと組み合わせた際の評価となる情報を格納している．2 色を組み合わせた際に色覚異常の見え方でも区別が付きやすい組み合わせかどうかこのテーブルの情報をもとに判断している．**cola**，**colb** はそれぞれ別のカラーコード，**gb** はその 2 色の組み合わせが **CUD** に配慮されているかを 0 か 1 で格納している．

表 6 **eva** テーブル

カラム名	内容
cola	カラーコード 1
colb	カラーコード 2
gb	評価

5.3.3 使用した DB

DB には **SQLite** を使用している．**SQLite** はアプリケーションに組み込むことで利用する簡易的な DB である．主要なデータベースに比べて大規模な Web アプリには不向きな反面，セットアップが容易であるため，本学習ツールでは **SQLite** を利用した開発を行った．

6 検証

この章では、開発した学習ツールを用いて行った検証について説明する。

6.1 実施内容

開発した学習ツールを用いて、学生 10 名を対象に検証した。対象者には、東京都が作成した CUD のガイドラインを参考に色覚異常と CUD について事前に説明した [3]。そして、対象者には学習ツールを利用した後に、Microsoft PowerPoint を用いてスライドを作成してもらった。そのスライドを CUD に配慮されているか項目別に評価した。

6.2 評価項目

評価項目は、適切なフォント、色の組み合わせ、強調目的の色、ハッチング、装飾の 5 項目である。これは先行研究での評価項目に、装飾に関する項目を加えた。配色に関する項目については、医学及びメディアデザイン学の博士号を持つ作者が開発した色のシミュレータを用い、通常の色覚に加え、P 型色覚、D 型色覚、T 型色覚での見え方を確認し評価した [6]。

「適切なフォント」では、使用しているフォントが CUD に配慮されているか評価した。線の一部が細くなっているフォントは色面積が小さくなり、色による判別がつきづらくなるため、色による文字の強調を行う際には適さないフォントである。そのため、線の幅が一定なフォントが使用されているかという観点で評価した。

「色の組み合わせ」では、2 色を組み合わせで使用する場合に、どの色覚であっても区別が付きやすい配色となっているか評価した。

「強調目的の色」では、文字の一部の色を別の色にすることで強調する場合に、どの色覚でも強調していることが伝わりやすい配色となっているか評価した。通常の色覚では黒と明確に区別できる色でも、別の色覚では黒と区別がつきづらい色に見える場合がある。複数の色覚での見え方を確認し、どの色覚でも明確に区別が付き、強調が伝わる配色となっているか評価した。

「ハッチング」では、折れ線グラフを作成する際に、それぞれの線を色以外の情報で区別できるようになっているか評価した。折れ線グラフを作成する際は、実線と破線を組み合わせ、マーカの図形を線ごとに別の種類にすることで、色に頼らずに線を区別できるため、色覚によらず伝わりやすいグラフとなる。これらの工夫が取り入れられているか評価した。

「装飾」では、色以外で文字の強調が伝わる資料となっているか評価した。文字を強調する際は、配色に加え、アンダーラインを引くことや、強調する文字を太くする等を行うことで、色以外の情報で文字の強調を伝えることができる。これらの工夫が取り入れられているか評価した。

7 結果

項目ごとに、CUD に配慮した工夫が取り入れられていた資料の割合を図 8 に示す。

配色に関する項目について、強調目的の色で CUD に配慮されていた資料は 90 %であり、先行研究に比べて 10 ポイント増加している。色の組み合わせでは、全ての資料が CUD に配慮した配色となっていた。

色以外の工夫に関する項目について、適切なフォントでは、全ての資料が CUD に配慮したフォントとなっていた。また、ハッチングで CUD に配慮されていた資料は 90 %であった。一方、新たに評価項目として加えた装飾で CUD に配慮されていた資料は 70 %であった。

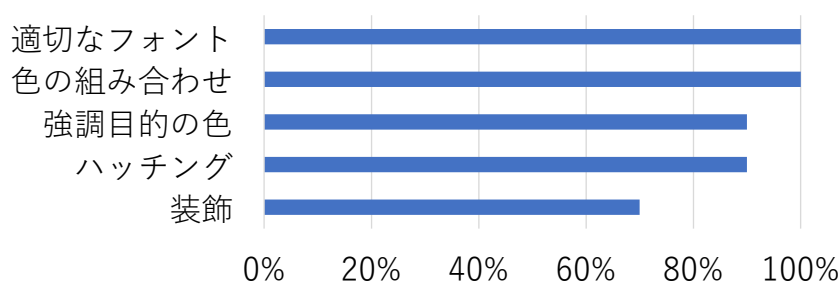


図 8 CUD 評価結果

8 考察

配色に関する項目で、先行研究に比べて CUD に配慮した資料が増加している。開発した学習ツールを利用することで、CUD に配慮した資料を作成できるという結果が得られた。そのため、学習者が CUD に配慮した工夫を項目ごとに実践することで、CUD に配慮した資料を作成できると考えられる。また、配色に関する項目で CUD に配慮した資料が増加した要因として、本学習ツールで別色覚での見え方や評価を即座に確認できるようになったことで、様々な配色を試すことが容易であったことが考えられる。

一方、評価項目の一つである装飾の観点では、CUD に配慮された資料は 70 %であった。本学習ツールでは、配色に関する項目の後に装飾を学習する。複数の色覚で区別が付きやすい配色を行った後に装飾に関する工夫を実践する流れであることや、配色と異なり複数回試すことがない項目であることから、装飾の重要性が伝わりにくくなっていたと考えられる。これらから、学習の流れや表示項目を改善することが望ましい。

9 結言

遺伝子の異常により通常と色の見え方が異なる色覚を持つ色覚異常者は、男性で 5 %，女性で 0.2 % 存在する．色覚の多様性に配慮し，より多くの人に伝わりやすいデザインとしてカラーユニバーサルデザイン（CUD）がある．

学生や社会人になると，ゼミや業務等で資料を作成する機会が増加する．そのため，CUD の講習では資料作成を実践し評価する機会が設けられている．しかし，講習を受けた学生の 20 % が CUD に配慮していない資料を作成した．

本研究では，CUD に配慮した工夫を一つ一つ実践する機会が必要だと考え，これらを行える学習ツールを開発した．開発した学習ツールを学習者が利用した結果，配色に関する項目で，先行研究に比べて CUD に配慮した資料が増加した．このことから，開発した学習ツールを利用することで，CUD に配慮した資料を作成できるという結果が得られた．一方，評価項目の一つである装飾の観点で CUD に配慮された資料は 70 % であり，他項目に比べて工夫が行われていないため，学習の流れや表示項目等の改善が望まれる．

謝辞

本論文の執筆にあたりご指導くださった須田先生に感謝申し上げます。また，研究室のメンバーには多くの手助けを頂きました。本当にありがとうございました。

参考文献

- [1] 岡部正隆・伊藤啓・橋本知子: “ユニバーサルデザインにおける色覚バリアフリーへの提言”, <https://www.nig.ac.jp/color/handout1.pdf>, 2024/1/9 参照
- [2] 菅宮恵子: “色覚異常を考慮した教材資料作成実習の実践報告とその評価”, 教職・学芸員課程研究, 2号 (2020), p.14-23, 2024/1/9 参照
- [3] 東京都福祉保健局生活福祉部地域福祉推進課: “東京都カラーユニバーサルデザインガイドライン”, <https://www.fukushi.metro.tokyo.lg.jp/kiban/machizukuri/kanren/color.files/colorudguideline.pdf>, 2024/1/9 参照
- [4] 府民文化部府政情報室広報広聴課: “色覚障がいのある人に配慮した色使いのガイドライン”, <https://www.pref.osaka.lg.jp/koho/shikikaku/guide1.html>, 2024/1/9 参照
- [5] 一般社団法人日本カラーコーディネーター協会: “大阪医療福祉専門学校様 特別授業「CUD」のご報告”, <https://www.j-color.or.jp/2023/03/15/blog-entry-884/>, 2024/1/9 参照
- [6] 浅田一憲: “色のシミュレータ”, <https://asada.website/cvssimulator/j/index.html>, 2024/1/9 参照

付録 A 作成したプログラム

A.1 app.js

```
1  const express = require("express");
2  const app = express();
3
4  const sqlite3 = require('sqlite3').verbose();
5  const db = new sqlite3.Database('test.db');
6
7  app.set('view engine', 'ejs');
8  app.use("/public", express.static("public"));
9  app.use(express.json())
10 app.use(express.urlencoded({ extended: true }));
11
12 app.get("/", (req, res) => {
13   res.render('top');
14 });
15
16 app.get("/question/1/:count", (req, res) => {
17
18   let textdata;
19   let col = new Array(3);
20   //回数を数える
21   var prev1 = 0;
22   var prev2 = 0;
23   var count = Number(req.params.count);
24   //console.log(count);
25   // テキスト
26   let sql = "select id,item,que"
27     + " from text"
28     + " where text.id = 1"
29     + ";";
30
31   db.serialize( () => {
32     db.all(sql, (error, data) => {
33       if( error ) {
```

```

34         res.render('show', {mes:"エラーです"});
35     }
36     //console.log(data);
37     textdata = data;
38 })
39 })
40
41 // 選択肢
42 let sqlb = "select color.id,color.name,ccode"
43     + " from tc inner join color"
44     + " on (color.id=tc.c_id)"
45     + " where t_id = 1"
46     + ";";
47
48 db.serialize( () => {
49     db.all(sqlb, (error, choices) => {
50         if( error ) {
51             res.render('show', {mes:"エラーです"});
52         }
53         //console.log(choices);
54         col[0] = choices[0].ccode;
55         res.render('layout_1a', {e:0,textdata:textdata,choices:choices,col:col,count:
56     })
57 })
58 });
59
60 app.post("/question/1/answer", (req, res) => {
61
62     let textdata;
63     let gb;
64     let col = new Array(5);
65     var count = Number(req.body.count);
66     var prev1 = Number(req.body.prev1);
67     var prev2 = Number(req.body.prev2);
68
69     // 選択された色のカラーコード
70     let sql = "select name,ccode,pcode,dcode,scode"
71         + " from color"

```

```

72     + " where id = " + req.body.choice
73     + ";";
74
75     db.serialize( () => {
76         db.all(sql, (error, cho) => {
77             if( error ) {
78                 res.render('show', {mes:"エラーです"});
79             }
80             col[0] = cho[0].ccode;
81             col[1] = cho[0].pcode;
82             col[2] = cho[0].dcode;
83             col[3] = cho[0].scode;
84             col[4] = cho[0].name;
85         })
86     })
87
88     //選択された色の評価
89     let sqla = "select gb"
90         + " from eva"
91         + " where cola = 1"
92         + " and colb = " + req.body.choice
93         + ";";
94
95     db.serialize( () => {
96         db.all(sqla, (error, eva) => {
97             if( error ) {
98                 res.render('show', {mes:"エラーです"});
99             }
100             gb = eva[0].gb;
101             if(gb == 1){
102                 if(prev1 != req.body.choice && prev2 != req.body.choice){
103                     count += 1;
104                     prev2 = prev1;
105                     prev1 = req.body.choice;
106                 }
107             }
108             //console.log("prev1:"+prev1);
109             //console.log("count:"+count);

```

```

110     })
111   })
112
113   // テキスト
114   let sqlb = "select *"
115     + " from text"
116     + " where text.id = 1"
117     + ";";
118
119   db.serialize( () => {
120     db.all(sqlb, (error, data) => {
121       if( error ) {
122         res.render('show', {mes:"エラーです"});
123       }
124       //console.log(data);
125       textdata = data;
126     })
127   })
128
129   // 選択肢
130   let sqlc = "select color.id,color.name,ccode"
131     + " from tc inner join color"
132     + " on (color.id=tc.c_id)"
133     + " where t_id = 1"
134     + ";";
135
136   db.serialize( () => {
137     db.all(sqlc, (error, choices) => {
138       if( error ) {
139         res.render('show', {mes:"エラーです"});
140       }
141       res.render('layout_1a', {e:1,textdata:textdata,choices:choices,col:col,gb:gb,
142     })
143   })
144   });
145
146   app.get("/question/2/:col", (req, res) => {
147

```

```

148     let textdata;
149     let col = [req.params.col,""];
150     let fon;
151
152     // テキスト
153     let sqlb = "select id,item,que"
154         + " from text"
155         + " where text.id = 2"
156         + ";";
157
158     db.serialize( () => {
159         db.all(sqlb, (error, data) => {
160             if( error ) {
161                 res.render('show', {mes:"エラーです"});
162             }
163             //console.log(data);
164             textdata = data;
165         })
166     })
167
168     // 選択肢
169     let sqlc = "select *"
170         + " from font"
171         + ";";
172
173     db.serialize( () => {
174         db.all(sqlc, (error, choices) => {
175             if( error ) {
176                 res.render('show', {mes:"エラーです"});
177             }
178             //console.log(choices);
179             fon = choices[0].face;
180             res.render('layout_1b', {e:0,textdata:textdata,choices:choices,col:col,fon:fon});
181         })
182     })
183 });
184
185 app.post("/question/2/answer", (req, res) => {

```

```

186
187     let textdata;
188     let col = new Array(2);
189     let fon = req.body.choice;
190     let gb;
191
192     // 選択されていた色のカラーコード
193     let sql = "select ccode, pcode"
194         + " from color"
195         + " where ccode = '" + req.body.col + "'"
196         + ";";
197
198     db.serialize( () => {
199         db.all(sql, (error, cho) => {
200             if( error ) {
201                 res.render('show', {mes:"エラーです"});
202             }
203             col[0] = cho[0].ccode;
204             col[1] = cho[0].pcode;
205         })
206     })
207
208     // テキスト
209     let sqlb = "select *"
210         + " from text"
211         + " where text.id = 2"
212         + ";";
213
214     db.serialize( () => {
215         db.all(sqlb, (error, data) => {
216             if( error ) {
217                 res.render('show', {mes:"エラーです"});
218             }
219             //console.log(data);    // ③
220             textdata = data;
221         })
222     })
223

```

```

224 // 選択肢
225 let sqlc = "select *"
226   + " from font"
227   + ";";
228
229 db.serialize( () => {
230   db.all(sqlc, (error, choices) => {
231     if( error ) {
232       res.render('show', {mes:"エラーです"});
233     }
234     for(i=0;i<choices.length;i++){
235       if(choices[i].face == fon) gb = choices[i].gb;
236     }
237     res.render('layout_1b', {e:1,textdata:textdata,choices:choices,col:col,fon:fo
238   })
239 })
240 });
241
242 app.get("/question/3/:col", (req, res) => {
243
244   let col = [req.params.col,""];
245
246   // テキスト
247   let sqlb = "select id,item,que"
248     + " from text"
249     + " where text.id = 3"
250     + ";";
251
252   db.serialize( () => {
253     db.all(sqlb, (error, data) => {
254       if( error ) {
255         res.render('show', {mes:"エラーです"});
256       }
257       //console.log(data);
258       res.render('layout_1c', {e:0,textdata:data,col:col,li:"",bo:"",ch:["",""]});
259     })
260   })
261 });

```

```

262
263 app.post("/question/3/answer", (req, res) => {
264
265     let col = new Array(2);
266     let gb;
267     let ch = new Array(2);
268     let li;
269     let bo;
270
271     if(req.body.line == 1 ){
272         ch[0] = "checked";
273         li = "underline";
274         gb = 1;
275     } else{
276         ch[0] = "";
277         li = "none";
278         gb = 0;
279     }
280     if(req.body.bold == 1 ){
281         ch[1] = "checked";
282         bo = "bold";
283     } else{
284         ch[1] = "";
285         bo = "normal";
286         gb = 0;
287     }
288
289     // 選択されていた色のカラーコード
290     let sql = "select ccode,pcode"
291         + " from color"
292         + " where ccode = '" + req.body.col + "'"
293         + ";";
294
295     db.serialize( () => {
296         db.all(sql, (error, eva) => {
297             if( error ) {
298                 res.render('show', {mes:"エラーです"});
299             }

```



```

300         col[0] = eva[0].ccode;
301         col[1] = eva[0].pcode;
302     })
303 })
304
305 // テキスト
306 let sqlb = "select *"
307     + " from text"
308     + " where text.id = 3"
309     + ";";
310
311 db.serialize( () => {
312     db.all(sqlb, (error, data) => {
313         if( error ) {
314             res.render('show', {mes:"エラーです"});
315         }
316         //console.log(data);
317         res.render('layout_1c', {e:1,textdata:data,col:col,gb:gb,li:li,bo:bo,ch:ch});
318     })
319 })
320 });
321
322 app.get("/question/4/:bcol/:count", (req, res) => {
323
324     let textdata;
325     let col = new Array(3);
326     let bcol = [req.params.bcol,""];
327     let backid = new Array(2);
328     //回数を数える
329     var prev1 = 0;
330     var prev2 = 0;
331     var count = Number(req.params.count);
332
333     // テキスト
334     let sql = "select id,item,que"
335         + " from text"
336         + " where text.id = 4"
337         + ";";

```

```

338
339 db.serialize( () => {
340     db.all(sql, (error, data) => {
341         if( error ) {
342             res.render('show', {mes:"エラーです"});
343         }
344         //console.log(data);
345         textdata = data;
346     })
347 })
348
349 //背景色の id
350 let sqla = "select id,name"
351     + " from color"
352     + " where ccode = '" + bcol[0] + "'"
353     + ";";
354
355 db.serialize( () => {
356     db.all(sqla, (error, back) => {
357         if( error ) {
358             res.render('show', {mes:"エラーです"});
359         }
360         backid[0] = back[0].id;
361         backid[1] = back[0].name;
362     })
363 })
364 // 選択肢
365 let sqlb = "select color.id,color.name,ccode"
366     + " from tc inner join color"
367     + " on (color.id=tc.c_id)"
368     + " where t_id = 4"
369     + " and color.ccode != '" + bcol[0] + "'"
370     + ";";
371
372 db.serialize( () => {
373     db.all(sqlb, (error, choices) => {
374         if( error ) {
375             res.render('show', {mes:"エラーです"});

```

```

376     }
377     //console.log(choices);
378     col[0] = choices[0].ccode;
379     res.render('layout_2a', {e:0,textdata:textdata,choices:choices,col:col,bcol:b
380   })
381 })
382 });
383
384 app.post("/question/4/answer", (req, res) => {
385
386   let textdata;
387   let gb;
388   let backid = req.body.backid;
389   let col = new Array(5);
390   let bcol = new Array(4);
391   var count = Number(req.body.count);
392   var prev1 = Number(req.body.prev1);
393   var prev2 = Number(req.body.prev2);
394
395   // 選択された色のカラーコード
396   let sql = "select name,ccode,pcode,dcode,scode"
397     + " from color"
398     + " where id = " + req.body.choice
399     + ";";
400
401   db.serialize( () => {
402     db.all(sql, (error, cho) => {
403       if( error ) {
404         res.render('show', {mes:"エラーです"});
405       }
406       col[0] = cho[0].ccode;
407       col[1] = cho[0].pcode;
408       col[2] = cho[0].dcode;
409       col[3] = cho[0].scode;
410       col[4] = cho[0].name;
411     })
412   })
413

```

```

414 // 背景色のカラーコード
415 let sqld = "select ccode,pcode,dcode,scode"
416   + " from color"
417   + " where ccode = '" + req.body.bcol + "'"
418   + ";";
419
420 db.serialize( () => {
421   db.all(sqld, (error, chob) => {
422     if( error ) {
423       res.render('show', {mes:"エラーです"});
424     }
425     bcol[0] = chob[0].ccode;
426     bcol[1] = chob[0].pcode;
427     bcol[2] = chob[0].dcode;
428     bcol[3] = chob[0].scode;
429   })
430 })
431
432 //選択された色の評価
433 let sqla = "select gb"
434   + " from eva"
435   + " where (cola = " + backid + " and colb = " + req.body.choice + ")"
436   + " or (cola = " + req.body.choice + " and colb = " + backid + ")"
437   + ";";
438
439 db.serialize( () => {
440   db.all(sqla, (error, eva) => {
441     if( error ) {
442       res.render('show', {mes:"エラーです"});
443     }
444     gb = eva[0].gb;
445     if(gb == 1){
446       if(prev1 != req.body.choice && prev2 != req.body.choice){
447         count += 1;
448         prev2 = prev1;
449         prev1 = req.body.choice;
450       }
451     }

```

```

452         //console.log("prev1:"+prev1);
453         //console.log("count:"+count);
454     })
455 })
456
457 // テキスト
458 let sqlb = "select *"
459     + " from text"
460     + " where text.id = 4"
461     + ";";
462
463 db.serialize( () => {
464     db.all(sqlb, (error, data) => {
465         if( error ) {
466             res.render('show', {mes:"エラーです"});
467         }
468         //console.log(data);
469         textdata = data;
470     })
471 })
472
473 // 選択肢
474 let sqlc = "select color.id,color.name,ccode"
475     + " from tc inner join color"
476     + " on (color.id=tc.c_id)"
477     + " where t_id = 4"
478     + " and color.id != " + backid
479     + ";";
480
481 db.serialize( () => {
482     db.all(sqlc, (error, choices) => {
483         if( error ) {
484             res.render('show', {mes:"エラーです"});
485         }
486         res.render('layout_2a', {e:1,textdata:textdata,choices:choices,col:col,bcol:bcol});
487     })
488 })
489 });

```

```

490
491 app.get("/question/5/:bcol", (req, res) => {
492
493     let textdata;
494     let col = new Array(3);
495     let bcol = [req.params.bcol,""];
496     let backid;
497
498     // テキスト
499     let sql = "select id,item,que"
500         + " from text"
501         + " where text.id = 5"
502         + ";";
503
504     db.serialize( () => {
505         db.all(sql, (error, data) => {
506             if( error ) {
507                 res.render('show', {mes:"エラーです"});
508             }
509             //console.log(data);
510             textdata = data;
511         })
512     })
513
514     //背景色の id
515
516     let sqla = "select id"
517         + " from color"
518         + " where ccode = '" + bcol[0] + "'"
519         + ";";
520
521     db.serialize( () => {
522         db.all(sqla, (error, back) => {
523             if( error ) {
524                 res.render('show', {mes:"エラーです"});
525             }
526             backid = back[0].id;
527         })

```

```

528   })
529   // 選択肢
530   let sqlb = "select color.id,color.name,ccode"
531     + " from tc inner join color"
532     + " on (color.id=tc.c_id)"
533     + " where t_id = 5"
534     + " and color.ccode != '" + bcol[0] + "'"
535     + ";";
536
537   db.serialize( () => {
538     db.all(sqlb, (error, choices) => {
539       if( error ) {
540         res.render('show', {mes:"エラーです"});
541       }
542       //console.log(choices);
543       col[0] = choices[0].ccode;
544       res.render('layout_2b', {e:0,textdata:textdata,choices:choices,col:col,bcol:bcol});
545     })
546   })
547 });
548
549 app.post("/question/5/answer", (req, res) => {
550
551   let textdata;
552   let gb;
553   let backid = req.body.backid;
554   let col = new Array(5);
555   let bcol = new Array(4);
556
557   // 選択された色のカラーコード
558   let sql = "select name,ccode,pcode,dcode,scode"
559     + " from color"
560     + " where id = " + req.body.choice
561     + ";";
562
563   db.serialize( () => {
564     db.all(sql, (error, cho) => {
565       if( error ) {

```

```

566         res.render('show', {mes:"エラーです"});
567     }
568     col[0] = cho[0].ccode;
569     col[1] = cho[0].pcode;
570     col[2] = cho[0].dcode;
571     col[3] = cho[0].scode;
572     col[4] = cho[0].name;
573 })
574 })
575
576 // 背景色のカラーコード
577 let sqld = "select ccode,pcode,dcode,scode"
578     + " from color"
579     + " where ccode = '" + req.body.bcol + "'"
580     + ";";
581
582 db.serialize( () => {
583     db.all(sqld, (error, chob) => {
584         if( error ) {
585             res.render('show', {mes:"エラーです"});
586         }
587         bcol[0] = chob[0].ccode;
588         bcol[1] = chob[0].pcode;
589         bcol[2] = chob[0].dcode;
590         bcol[3] = chob[0].scode;
591     })
592 })
593
594 //選択された色の評価
595 let sqla = "select gb"
596     + " from eva"
597     + " where (cola = " + backid + " and colb = " + req.body.choice + ")"
598     + " or (cola = " + req.body.choice + " and colb = " + backid + ")"
599     + ";";
600
601 db.serialize( () => {
602     db.all(sqla, (error, eva) => {
603         if( error ) {

```



```

604         res.render('show', {mes:"エラーです"});
605     }
606     gb = eva[0].gb;
607 })
608 })
609
610 // テキスト
611 let sqlb = "select *"
612     + " from text"
613     + " where text.id = 5"
614     + ";";
615
616 db.serialize( () => {
617     db.all(sqlb, (error, data) => {
618         if( error ) {
619             res.render('show', {mes:"エラーです"});
620         }
621         //console.log(data);
622         textdata = data;
623     })
624 })
625
626 // 選択肢
627 let sqlc = "select color.id,color.name,ccode"
628     + " from tc inner join color"
629     + " on (color.id=tc.c_id)"
630     + " where t_id = 5"
631     + " and color.id != " + backid
632     + ";";
633
634 db.serialize( () => {
635     db.all(sqlc, (error, choices) => {
636         if( error ) {
637             res.render('show', {mes:"エラーです"});
638         }
639         res.render('layout_2b', {e:1,textdata:textdata,choices:choices,col:col,bcol:bcol});
640     })
641 })

```

```

642 });
643
644 app.get("/question/6/:bcol", (req, res) => {
645
646     let textdata;
647     let col = new Array(3);
648     let bcol = [req.params.bcol,""];
649     let backid;
650
651     // テキスト
652     let sql = "select id,item,que"
653         + " from text"
654         + " where text.id = 6"
655         + ";";
656
657     db.serialize( () => {
658         db.all(sql, (error, data) => {
659             if( error ) {
660                 res.render('show', {mes:"エラーです"});
661             }
662             //console.log(data);
663             textdata = data;
664         })
665     })
666
667     //背景色の id
668
669     let sqla = "select id"
670         + " from color"
671         + " where ccode = '" + bcol[0] + "'"
672         + ";";
673
674     db.serialize( () => {
675         db.all(sqla, (error, back) => {
676             if( error ) {
677                 res.render('show', {mes:"エラーです"});
678             }
679             backid = back[0].id;

```

```

680     })
681   })
682   // 選択肢
683   let sqlb = "select color.id,color.name,ccode"
684     + " from tc inner join color"
685     + " on (color.id=tc.c_id)"
686     + " where t_id = 6"
687     + " and color.ccode != '" + bcol[0] + "'"
688     + ";";
689
690   db.serialize( () => {
691     db.all(sqlb, (error, choices) => {
692       if( error ) {
693         res.render('show', {mes:"エラーです"});
694       }
695       //console.log(choices);
696       col[0] = choices[0].ccode;
697       res.render('layout_2c', {e:0,textdata:textdata,choices:choices,col:col,bcol:bcol});
698     })
699   })
700 });
701
702 app.post("/question/6/answer", (req, res) => {
703
704   let textdata;
705   let gb;
706   let backid = req.body.backid;
707   let col = new Array(5);
708   let bcol = new Array(4);
709
710   // 選択された色のカラーコード
711   let sql = "select name,ccode,pcode,dcode,scode"
712     + " from color"
713     + " where id = " + req.body.choice
714     + ";";
715
716   db.serialize( () => {
717     db.all(sql, (error, cho) => {

```

```

718         if( error ) {
719             res.render('show', {mes:"エラーです"});
720         }
721         col[0] = cho[0].ccode;
722         col[1] = cho[0].pcode;
723         col[2] = cho[0].dcode;
724         col[3] = cho[0].scode;
725         col[4] = cho[0].name;
726     })
727 })
728
729 // 背景色のカラーコード
730 let sqld = "select ccode,pcode,dcode,scode"
731     + " from color"
732     + " where ccode = '" + req.body.bcol + "'"
733     + ";";
734
735 db.serialize( () => {
736     db.all(sqld, (error, chob) => {
737         if( error ) {
738             res.render('show', {mes:"エラーです"});
739         }
740         bcol[0] = chob[0].ccode;
741         bcol[1] = chob[0].pcode;
742         bcol[2] = chob[0].dcode;
743         bcol[3] = chob[0].scode;
744     })
745 })
746
747 //選択された色の評価
748 let sqla = "select gb"
749     + " from eva"
750     + " where (cola = " + backid + " and colb = " + req.body.choice + ")"
751     + " or (cola = " + req.body.choice + " and colb = " + backid + ")"
752     + ";";
753
754 db.serialize( () => {
755     db.all(sqla, (error, eva) => {

```

```

756         if( error ) {
757             res.render('show', {mes:"エラーです"});
758         }
759         gb = eva[0].gb;
760     })
761 })
762
763 // テキスト
764 let sqlb = "select *"
765     + " from text"
766     + " where text.id = 6"
767     + ";";
768
769 db.serialize( () => {
770     db.all(sqlb, (error, data) => {
771         if( error ) {
772             res.render('show', {mes:"エラーです"});
773         }
774         //console.log(data);
775         textdata = data;
776     })
777 })
778
779 // 選択肢
780 let sqlc = "select color.id,color.name,ccode"
781     + " from tc inner join color"
782     + " on (color.id=tc.c_id)"
783     + " where t_id = 6"
784     + " and color.id != " + backid
785     + ";";
786
787 db.serialize( () => {
788     db.all(sqlc, (error, choices) => {
789         if( error ) {
790             res.render('show', {mes:"エラーです"});
791         }
792         res.render('layout_2c', {e:1,textdata:textdata,choices:choices,col:col,bcol:bcol});
793     })

```

```

794     })
795   });
796
797   app.get("/question/7", (req, res) => {
798     let textdata;
799     let col = [["FF0000",""],["008000",""],["0000FF",""]];
800     // テキスト
801     let sql = "select id,item,que"
802       + " from text"
803       + " where text.id = 7"
804       + ";";
805
806     db.serialize( () => {
807       db.all(sql, (error, data) => {
808         if( error ) {
809           res.render('show', {mes:"エラーです"});
810         }
811         //console.log(data);
812         textdata = data;
813       })
814     })
815
816     // 選択肢
817     let sqlb = "select color.id,color.name,ccode"
818       + " from tc inner join color"
819       + " on (color.id=tc.c_id)"
820       + " where t_id = 7"
821       + ";";
822
823     db.serialize( () => {
824       db.all(sqlb, (error, choices) => {
825         if( error ) {
826           res.render('show', {mes:"エラーです"});
827         }
828         //console.log(choices);
829         res.render('layout_3a', {e:0,textdata:textdata,choices:choices,col:col});
830       })
831     })

```

```

832 });
833
834 app.post("/question/7/answer", (req, res) => {
835
836     let textdata;
837     let gb = 1;
838     let id = [req.body.s0, req.body.s1, req.body.s2];
839     let col = [[], [], []];
840     // 選択された色のカラーコード
841     let s = "select name, ccode, pcode, dcode, scode"
842         + " from color"
843         + " where id = ";
844
845     let sqls = [ s+id[0]+";" , s+id[1]+";" , s+id[2]+";" ];
846
847     let i = 0;
848     for(let sql of sqls){
849         db.serialize( () => {
850             db.all(sql, (error, data) => {
851                 if( error ) {
852                     return res.render('show', {mes:"エラーです"});
853                 }
854                 col[i][0] = data[0].ccode;
855                 col[i][1] = data[0].pcode;
856                 col[i][2] = data[0].dcode;
857                 col[i][3] = data[0].scode;
858                 col[i][4] = data[0].name;
859                 i++;
860                 //console.log(col);
861             })
862         })
863     }
864
865     //選択された色の評価
866     var sqla = new Array();
867     var a;
868     for(let j=0; j<2; j++){
869         for(let k=j+1; k<3; k++){

```

```

870     a = "select gb"
871     + " from eva"
872     + " where (cola = " + id[j] + " and colb = " + id[k] + ")"
873     + " or (cola = " + id[k] + " and colb = " + id[j] + ")"
874     + ";";
875     sqli.push(a);
876   }
877 }
878 for(let sql of sqli){
879   db.serialize( () => {
880     db.all(sql, (error, eva) => {
881       if( error ) {
882         return res.render('show', {mes:"エラーです"});
883       }
884       if(eva[0].gb == 0){
885         gb = 0;
886       }
887     })
888   })
889 }
890
891 // テキスト
892 let sqlb = "select *"
893   + " from text"
894   + " where text.id = 7"
895   + ";";
896
897 db.serialize( () => {
898   db.all(sqlb, (error, data) => {
899     if( error ) {
900       res.render('show', {mes:"エラーです"});
901     }
902     //console.log(data);
903     textdata = data;
904   })
905 })
906
907 // 選択肢

```



```

908     let sqlc = "select color.id,color.name,ccode"
909         + " from tc inner join color"
910         + " on (color.id=tc.c_id)"
911         + " where t_id = 7"
912         + ";";
913
914     db.serialize( () => {
915         db.all(sqlc, (error, choices) => {
916             if( error ) {
917                 res.render('show', {mes:"エラーです"});
918             }
919             res.render('layout_3a', {e:1,textdata:textdata,choices:choices,col:col,gb:gb});
920         })
921     })
922 });
923
924 app.get("/question/8/:col", (req, res) => {
925
926     let col = [[req.params.col.substr(0,6),""],[req.params.col.substr(6,6),""],[req.p
927     let check = new Array(6).fill(["checked","", ""]);
928     // テキスト
929     let sql = "select id,item,que"
930         + " from text"
931         + " where text.id = 8"
932         + ";";
933
934     db.serialize( () => {
935         db.all(sql, (error, data) => {
936             if( error ) {
937                 res.render('show', {mes:"エラーです"});
938             }
939             //console.log(data);
940             res.render('layout_3b', {e:0,textdata:data,col:col,check:check});
941         })
942     })
943 });
944
945 app.post("/question/8/answer", (req, res) => {

```

```

946     let col = [[req.body.a0s0,""],[req.body.a0s1,""],[req.body.a0s2,""]];
947     let check = [[","","",""],[","","",""],[","","",""],[","","",""],[","","",""],[","","",""]];
948     let c = [req.body.s0c0,req.body.s1c0,req.body.s2c0,req.body.s0c1,req.body.s1c1,req.body.s2c1];
949     let gb = 1;
950     //console.log(c);
951
952     //チェックされた場所
953     for(let i=0;i<6;i++){
954         check[i][c[i]] = "checked";
955     }
956     //console.log(check);
957     //評価（線の種類）
958     for(let i=0;i<3;i++){
959         for(let j=i+1;j<3;j++){
960             if(c[i] == c[j]){
961                 gb = 0;
962                 break;
963             }
964         }
965     }
966
967     //評価（ポイントの図形）
968     for(let i=3;i<6;i++){
969         for(let j=i+1;j<6;j++){
970             if(c[i] == c[j]){
971                 gb = 0;
972                 break;
973             }
974         }
975     }
976
977     //カラーコード
978     let s = "select pcode"
979         + " from color"
980         + " where ccode = '";
981
982     let sqls = [ s+req.body.a0s0+"'";" , s+req.body.a0s1+"'";" , s+req.body.a0s2+"'";" ]
983     let i = 0;

```

```

984     for(let sql of sqls){
985         db.serialize( () => {
986             db.all(sql, (error, data) => {
987                 if( error ) {
988                     return res.render('show', {mes:"エラーです"});
989                 }
990                 col[i][1] = data[0].pcode;
991                 i++;
992                 //console.log(col);
993             })
994         })
995     }
996
997     // テキスト
998     let sql = "select *"
999         + " from text"
1000         + " where text.id = 8"
1001         + ";";
1002
1003     db.serialize( () => {
1004         db.all(sql, (error, data) => {
1005             if( error ) {
1006                 res.render('show', {mes:"エラーです"});
1007             }
1008             //console.log(data);
1009             res.render('layout_3b', {e:1,textdata:data,col:col,check:check,gb:gb,c:c});
1010         })
1011     })
1012 });
1013
1014 app.get("/question/9/:col/:mark", (req, res) => {
1015
1016     let col = [[req.params.col.substr(0,6),""],[req.params.col.substr(6,6),""],[req.p
1017     let check = ["checked",""];
1018     let mark = [req.params.mark.substr(0,1),req.params.mark.substr(1,1),req.params.ma
1019     //console.log(col);
1020     //console.log(mark);
1021

```

```

1022 // テキスト
1023 let sql = "select id,item,que"
1024   + " from text"
1025   + " where text.id = 9"
1026   + ";";
1027
1028 db.serialize( () => {
1029   db.all(sql, (error, data) => {
1030     if( error ) {
1031       res.render('show', {mes:"エラーです"});
1032     }
1033     //console.log(data);
1034     res.render('layout_3c', {e:0,textdata:data,col:col,check:check,mark:mark});
1035   })
1036 })
1037 });
1038
1039 app.post("/question/9/answer", (req, res) => {
1040   let col = [[req.body.a0s0,""],[req.body.a0s1,""],[req.body.a0s2,""]];
1041   let check = ["",""];
1042   let mark = [req.body.m0,req.body.m1,req.body.m2,req.body.m3,req.body.m4,req.body.m5];
1043   let gb;
1044
1045   //評価
1046   if(req.body.c0 == "right"){
1047     gb = 1;
1048     check[1] = "checked";
1049   }else{
1050     gb = 0;
1051     check[0] = "checked";
1052   }
1053
1054   //カラーコード
1055   let s = "select pcode"
1056     + " from color"
1057     + " where ccode = '";
1058
1059   let sqls = [ s+req.body.a0s0+"'";" , s+req.body.a0s1+"'";" , s+req.body.a0s2+"'";" ]

```

```

1060     let i = 0;
1061     for(let sql of sqls){
1062         db.serialize( () => {
1063             db.all(sql, (error, data) => {
1064                 if( error ) {
1065                     return res.render('show', {mes:"エラーです"});
1066                 }
1067                 col[i][1] = data[0].pcode;
1068                 i++;
1069                 //console.log(col);
1070             })
1071         })
1072     }
1073
1074     // テキスト
1075     let sql = "select *"
1076         + " from text"
1077         + " where text.id = 9"
1078         + ";";
1079
1080     db.serialize( () => {
1081         db.all(sql, (error, data) => {
1082             if( error ) {
1083                 res.render('show', {mes:"エラーです"});
1084             }
1085             //console.log(data);
1086             res.render('layout_3c', {e:1,textdata:data,col:col,check:check,mark:mark,gb:g
1087         })
1088     })
1089 });
1090
1091 app.use(function(req, res, next) {
1092     res.status(404).send(' ページが見つかりません');
1093 });
1094
1095 app.listen(8080, () => console.log("Example app listening on port 8080!"));

```

A.2 layout-1a.ejs

```
1  <!Doctype html>
2  <html>
3  <head>
4    <meta charset="utf-8">
5    <script src="/public/js/func.js"></script>
6    <title>CUDA 学習ツール</title>
7  </head>
8  <style type="text/css">
9    span#phrase {
10      color:<%= "#" + col[0] %>;
11    }
12    span#a1 {
13      color:<%= "#" + col[1] %>;
14    }
15    span#a2 {
16      color:<%= "#" + col[2] %>;
17    }
18    span#a3 {
19      color:<%= "#" + col[3] %>;
20    }
21    body{
22      font-size: 1.5rem;
23    }
24    #cho,#sen,#ano{
25      font-size: 2rem;
26    }
27    #cho{
28      font-weight: bold;
29    }
30    #sen,#ano{
31      border: 2px solid;
32      border-color:#c8c8c8;
33      width: <%= 2*15 %>rem;
34    }
35    #sen{
```

```

36     text-align:center;
37 }
38 #sim{
39     text-align:left;
40 }
41 .button{
42     font-size: 1.5rem;
43 }
44 </style>
45 <body>
46     <a href="/">トップページに戻る</a>
47     <hr>
48     <p>1 部：文字の強調 [ 1. 配色 → 2. 文字フォント → 3. 装飾 ]</p>
49     <hr>
50     <h2><%= textdata[0].item %></h2>
51     <hr>
52
53     <form action="/question/1/answer" method="post">
54     <div id="que">
55         <p><%= textdata[0].que %></p>
56     </div>
57     <div id="cho">
58         <% let c;
59         for (let row of choices){
60             if(row.ccode == col[0]) c = "checked";
61             else c = ""; %>
62         <font color="<%= '#' + row.ccode %>">
63             <input type="radio" name="choice" id="<%= row.ccode %>" value="<%= row.id %>"
64             </font>
65         <% } %>
66     </div>
67     <div id="sen">
68         <p>文字を<span id="phrase">強調</span>します</p>
69     </div>
70
71     <div id="button">
72         <input type="hidden" name="id" value="<%= textdata[0].id %>">
73         <p><input type="submit" value="評価" class="button"></p>

```

```

74     </div>
75
76     <input type="hidden" name="count" value="<%= count %>">
77     <input type="hidden" name="prev1" value="<%= prev1 %>">
78     <input type="hidden" name="prev2" value="<%= prev2 %>">
79 </form>
80
81 <% if(e == 1){ %>
82     <div id="acho">
83         選択した色：<%= col[4] %>
84     </div>
85     <div id="ano">
86         <別色覚での見え方><br>
87         <div id="sim">
88             通常：文字を<span id="phrase">強調</span>します<br>
89             P 型：文字を<span id="a1">強調</span>します<br>
90             D 型：文字を<span id="a2">強調</span>します<br>
91             T 型：文字を<span id="a3">強調</span>します
92         </div>
93     </div>
94     <div id="eva">
95         <% if(gb == 0) { %>
96             <p><%= textdata[0].bad %></p>
97             <p>ポイント：<%= textdata[0].adv %></p>
98             <p>別の選択肢を選んでください。</p>
99         <% }else{ %>
100             <p><%= textdata[0].good %></p>
101             <p>ポイント：<%= textdata[0].adv %></p>
102             <hr>
103             <% if(count >= 3) { %>
104                 <p><a href="<%= '/question/2/' + col[0] %>">次の項目へ</a></p>
105             <% } else { %>
106                 <p>別の選択肢も試みましょう。文字の強調に適する色を 3 つ以上試すことで次
107                 の項目に移ることができます。</p>
108             <% } %>
109         </div>
110     <% } %>

```



```
111    </body>
112    </html>
```

A.3 func.js

```
1  function choiceColor(code) {
2      const phrase = document.querySelector("#phrase");
3      phrase.style.color = "#" + code;
4  }
5
6  function choiceFont(font) {
7      const phrase = document.querySelector("#phrase");
8      phrase.style.fontFamily = font;
9  }
10
11 function choiceLine() {
12     const phrase = document.querySelector("#phrase1");
13     phrase.style.textDecoration = "underline";
14 }
15 function choiceBold() {
16     const phrase = document.querySelector("#phrase2");
17     phrase.style.fontWeight = "bold";
18 }
19 function choiceClearLine() {
20     const phrase = document.querySelector("#phrase1");
21     phrase.style.textDecoration = "none";
22 }
23 function choiceClearBold() {
24     const phrase = document.querySelector("#phrase2");
25     phrase.style.fontWeight = "normal";
26 }
```