161A09
2016/3/2

MagicalSheet: PC画面における特定色を消すための透明シートの実現とその応用

丹羽 佑輔[1,a)]　大囿 忠親[1,b)]　新谷 虎松[1,c)]

概要：本研究では，PC のデスクトップ画面上の指定した領域の特定の色を消すことが可能である MagicalSheet を試作した．ユーザーは，MagicalSheet を用いることで赤シートを用いた暗記学習を PC のデスクトップ画面上で行うことが可能となる．デスクトップ画面上のある領域内の特定の色を消すためには，その領域に表示されたアプリケーションの描画処理を拡張する必要があり困難である．MagicalSheet では，デスクトップ画面を撮影し，撮影した画像から特定色を消す処理を行い，デスクトップ画面上に重ねて表示することで実現した．MagicalSheet のソフトウェアの利点として，コンテンツを作成するアプリケーションの制約は色のみであり，また，既存のアプリケーションの変更は必要無い点である．例えば，プレゼンテーションソフトでスライドを作成しプレゼンテーションモードで表示することで，MagicalSheet のコンテンツとして利用することができる．

MagicalSheet: Realizing a Transparent Sheet to Erase Specified Colors on a PC Screen

Yusuke Niwa[1,a)]　Tadachika Ozono[1,b)]　Toramatsu Shintani[1,c)]

Abstract: We implemented a system to erase specified colors on a PC screen, called MagicalSheet. We enhanced the drawing process on a PC screen to erase specified colors. Because the system erases specified colors of an image captured by transparent layers, any modification of existing desktop applications is not required. We can program the erase operation by using HTML5 technologies on the transparent layer, and it allows for easy and fast prototyping.

1. はじめに

紙媒体上に赤色で書かれた文字を，プラスチック製の赤色の半透明シートを利用して隠すことで記憶学習を行うための教材などがある．また，OHP を用いたプレゼンテーションで，黒色のシートを用いて，部分的にスライドの内容を隠す表現がある．これらは，重ね合わせにより，コンテンツの視覚的情報を合成あるいは遮断する例である．Web 上には，文字による情報資源が豊富である．これらの情報を教材として利用し学習する際，赤色の半透明シートと同様の操作により PC のデスクトップ画面上で Web ページの内容を隠すことができると便利である．実際に紙媒体上に印刷する必要もなく，端末やディスプレイ上で記憶学習を行うことができる．

本研究では，PC のデスクトップ画面上の指定した領域の特定の色を消すことが可能である MagicalSheet を試作した．ユーザーは，MagicalSheet を用いることで赤シートを用いた暗記学習を PC のデスクトップ画面上で行うことが可能となる．赤色の半透明シートを使って赤文字を隠す方法では，その性質上，シートで覆う領域全てが赤くなってしまい他の字も見づらくなり，長時間利用していると視界に余韻が残ってしまう点がある．MagicalSheet では，消去対象の色を透明にするため，対象とする色を持つ部分以

[1] 名古屋工業大学大学院 情報工学専攻
Department of Computer Science and Engineering, Graduate School of Engineering, Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya, Aichi, 466-8555 Japan
[a)] yusuken@toralab.org
[b)] ozono@toralab.org
[c)] tora@toralab.org

外の領域はシートで覆う前と後で変化することがないため，現実の赤シートのように見づらくなることはない．

デスクトップ画面上のある領域内の特定の色を消すためには，その領域に表示されたアプリケーションの描画処理を拡張する必要があり困難である．MagicalSheet では，デスクトップ画面を撮影し，撮影した画像から特定色を消す処理を行い，デスクトップ画面上に重ねて表示することで実現した．

MagicalSheet のソフトウェアの利点として，デスクトップ画面を撮影し画像データを処理するため，既存のアプリケーションの描画処理を拡張する必要は無い．例えば，プレゼンテーションソフトでスライドを作成してプレゼンテーションモードで表示したり，イラストソフトでのイラスト作成し画像ファイルとして出力して画像閲覧ソフトでフルスクリーン化して表示することで，MagicalSheet のコンテンツとして利用することができる．

2. 関連研究

半透明の GUI を他のアプリケーションの GUI に重ね，重なった部分の視覚的な情報の取得や変化させる研究が存在する [1]．MagicalSheet では，デスクトップ画面のスクリーンショットを取得しラスタデータを取得する処理を行う為，既存のアプリケーションとイベント情報のやりとりを行わない．

Apple 社の AppleScript など OS に特化したアプリケーション間通信機能は古くから実現されており，プログラミングなしでアプリケーション間通信を可能としている例も挙げられる．ここでは，事前に定められたプロトコルに従い，プログラムされたアプリケーション間での柔軟な連携が可能である．しかしながら，近年では Web アプリケーションが広範囲に利用されており，デスクトップ画面アプリケーションおよび Web アプリケーション間での連携に関しても考慮する必要がある．例えば，InterTwine[2] では，既存アプリケーションを改変することで，アプリケーション間連携を実現しているが，非専門家には容易ではない．複数のアプリケーションを協調的に利用することで，それぞれのアプリケーションを単純に組み合わせるのみでは得ることができない機能を実現することが可能である．

異なるアプリケーションの情報を取得する方法として，アプリケーションの GUI の情報を利用する方法がある．既存アプリケーションの GUI で用いられているアイコンなどのスクリーンショットによって，目的の GUI を検索し操作の自動化を行う研究もある [3] [4]．アプリケーションの操作に関するチュートリアルビデオは，操作方法を覚える上で便利であるが，チュートリアルビデオを見ながら対象とするアプリケーションを操作するのは困難である．アプリケーションが提供する GUI の情報およびユーザの操作情報を利用しチュートリアルのビデオの一時停止および再生，巻き戻しの制御を行う研究がある [5]．MagicalSheet は，PC のデスクトップ画面上のスクリーンショット情報を用いて，画像処理によって，既存のアプリケーションの GUI に応じて表示する内容を変更することが可能である．

タブレットデバイスの画面上または空中に重ねた物理デバイスの位置に応じた情報を表示する研究がある [6][7]．SWB は，PC のデスクトップ画面上で Web コンテンツの重畳表示を可能とし，Web コンテンツの表示位置に応じてコンテンツの内容を変えることが可能である．

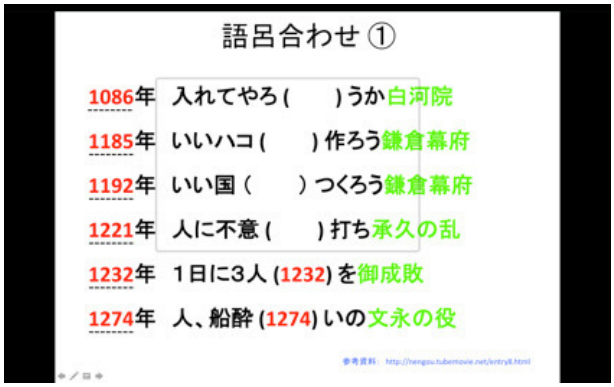

図 1 透明シートの内部の赤色を隠している様子．
Fig. 1 The transparent sheet hides red colored characters under the sheet.

図 2 MagicalSheet の操作メニュー．
Fig. 2 The action menu of MagicalSheet.

3. MagicalSheet の機能と実装

本節では，MagicalSheet の機能と実装を示す．MagicalSheet は，本研究室で開発した Silhouette Web Browser (SWB) を用いて作成した．SWB は，Mac OS 上で動作する WebKit ベースの Web コンテンツビューア環境であり，Web コンテンツの背景を透明にしデスクトップ画面上に表示することが可能である．HTML5 によって記述されたコンテンツを表示することが可能であるため，JavaScript による動的なコンテンツを表示することも可能である．MagicalSheet が利用する SWB の機能は，スクリーンショット

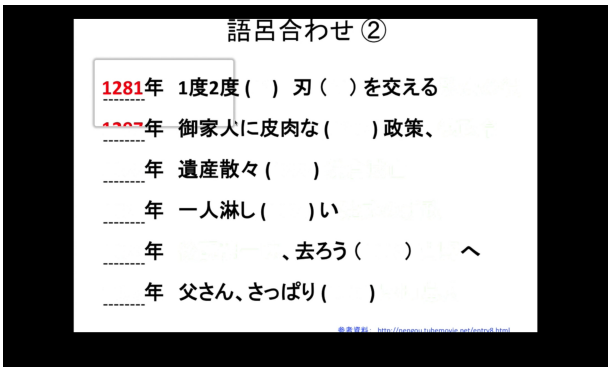

図 3 透明シートの縮小.
Fig. 3 Shrinking the transparent sheet.

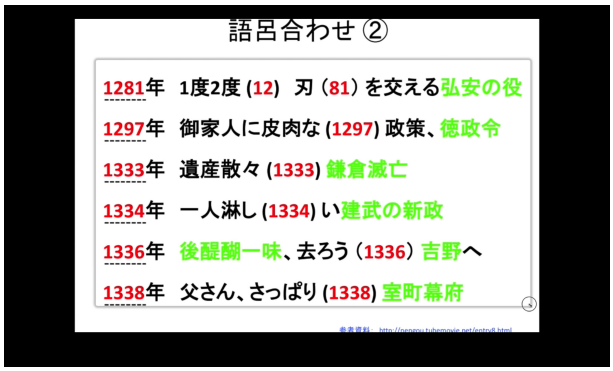

図 4 透明シートの拡大.
Fig. 4 Expanding the transparent sheet.

機能と，レイヤー順序制御である．

図 1 は，PowerPoint スライドショー中の画面上の一部に MagicalSheet の透明シートを重ね，シートの内部の赤色を隠している様子である．図中の背面は PowerPoint によって描画されているスライドの内容である．図の中央部分に影付きの枠が表示されている部分が，MagicalSheet の透明シートの領域である．

図 2 は，MagicalSheet の操作メニューと透明シートが表示されている状態を示す．MagicalSheet を起動するとデスクトップ画面の左下に MagicalSheet の操作メニューが表示される．操作メニューには 7 つのボタンがあり，上から，選択色透明ボタン，赤透明ボタン，緑透明ボタン，青透明ボタン，シート反転ボタン，シート追加ボタン，シート削除ボタンである．選択色透明ボタン，赤透明ボタン，緑透明ボタン，青透明ボタンをクリックすることで，透明シートによって透明にする色を設定するボタンであり，透過の有効または無効を切り替えることができる．赤透明ボタンを押した場合は，透明シート内の赤色が透明になる．これらのボタンの領域からマウスカーソルが離れてから 5 秒経過すると，操作メニューが自動的に非表示になる．画面の左下の灰色のバー上の部分をクリックすることで，操作メニューが表示される．

MagicalSheet は，デスクトップ画面を撮影し，撮影した画像から特定色を消す処理を行い，デスクトップ画面上に重ねて表示することで実現した．MagicalSheet の透明シートを常に最前面に表示するためにレイヤー順序制御を利用する．

3.1 透明シートの移動・サイズ変更

MagicalSheet で表示している透明シートは SWB で管理している 1 つのレイヤー上で表示している．このレイヤーは SWB の機能により，常に最前面表示されているため，このレイヤーで表示している透明シートや設定ボタンの Web コンテンツは常に最前面に表示されるようになる．

透明シートの位置は，ドラッグ操作によって変更可能である．透明シートのサイズは，透明シートの境界部分をドラッグすることで変更可能である．図 3 は，透明シート縮小した様子，図 4 は，透明シート拡大した様子である．

3.2 色の透明処理

JavaScript で SWB の機能を呼び出すことでデスクトップ画面の画像データを得ることができる．その画像データを，HTML5 で定義された Canvas 関連の API を用いてラスタデータの処理を行う．

透明処理を行うに際して，RGB 空間を HSV 空間に変換後，各画素と特定色の差分を取り，その差分がある範囲に収まっている場合は透明にする色であるとみなし背景色に置き換える．HSV 空間上で透明処理を行う理由として，H 成分が人の色覚と類似しているためである．

RGB 空間の点 (R, G, B) から HSV 空間の点 (H, S, V) への色変換は次の式を用いた．

$$R' = R/255, G' = G/255, B' = B/255$$

$$C_{max} = max(R', G', B'), C_{min} = min(R', G', B')$$

$$\Delta = C_{max} - C_{min}$$

$$H = \begin{cases} 0^\circ & , \Delta = 0 \quad (1a) \\ 60^\circ \times \dfrac{G' - B'}{\Delta} \ mod\ 6 & , \mathrm{C}_{max} = R' (1b) \\ 60^\circ \times \dfrac{B' - R'}{\Delta} + 2 & , \mathrm{C}_{max} = G' (1c) \\ 60^\circ \times \dfrac{R' - G'}{\Delta} + 4 & , \mathrm{C}_{max} = B' (1d) \end{cases}$$

$$S = \begin{cases} 0 & , \mathrm{C}_{max} = 0 \quad (2a) \\ \dfrac{\Delta}{C_{max}} & , \mathrm{C}_{max} \neq 0 \quad (2b) \end{cases}$$

$$V = C_{max}$$

各色の差分の有効範囲は以下の通りである．

$$(dH, dS, dV) = |(H, S, V) - (H_p, S_p, V_p)|$$

$$dH \leq 0.15, dS \leq 0.80, dV \leq 0.80$$

赤の場合，$(H_p, S_p, V_p) = (0, 1, 1)$ である．緑の場

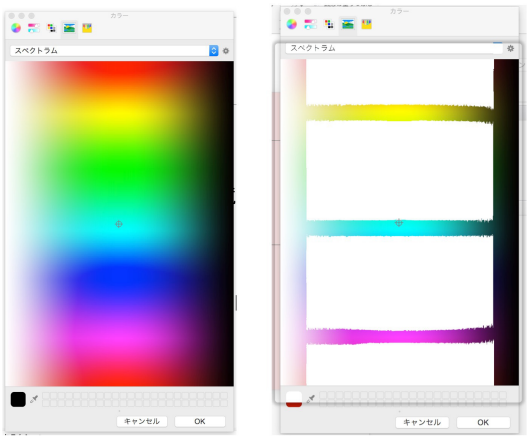

図 5 画像処理前 (左) と後 (右) の例
Fig. 5 Left: before the processing; Right: after the processing.

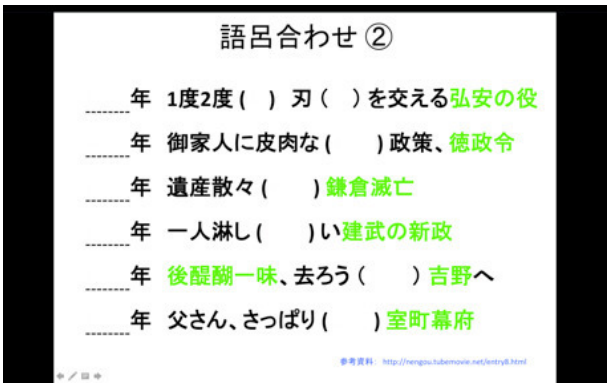

図 6 透明シートをフルスクリーン化して，スライド中の全ての赤色を隠している様子.
Fig. 6 The transparent sheet covers the whole slide and hides red colored characters in the slide.

合，$(H_p, S_p, V_p) = (3/10, 1, 1)$ である．青の場合，$(H_p, S_p, V_p) = (6/10, 1, 1)$ である．図 5 は，画像処理前 (左) と後 (右) のスペクトラムの例である．図中の左は，透明シートで覆っていない状態である．図中の右は，OS が提供する色選択インタフェース上に表示された色のスペクトラム上に透明シートを重ねて画像処理を行った結果である．赤・緑・青を特定色として設定しているため，それらの色に類似した部分が白色に置き換えられていることが確認できる．白く変化している 4 つの部分のうち，上から 1 つめと 4 つ目の領域が赤色，2 つ目が緑色，3 つ目が青色の画像処理によるものである．

3.3 フルスクリーン化

フルスクリーン化したい透明シートを選択して Ctrl + Shift + F キー を押す．元に戻す時は再度，同じキーを押す．図 6 は，透明シートをフルスクリーン化して，スライド中の全ての赤色を隠している様子である．

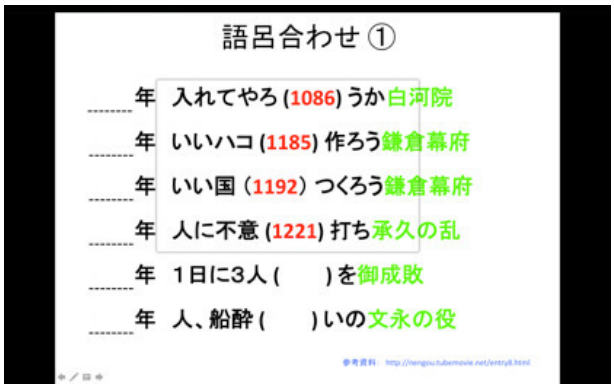

図 7 透明シートの内部の赤色を表示している例.
Fig. 7 The transparent sheet shows red colored characters under the sheet.

3.4 画面上から透明にする文字色を選択

操作メニュー上の「赤」「緑」「青」のボタンを押すことで，透明にする色を選択する．選択後に透明シートをクリックすると，選択した色が透明になる．これら以外の色をの文字を透明にしたい場合は，デスクトップ画面上から色を選択して，それを透明にする色として扱うことができる．「無」のボタンを押すと画面が赤枠で囲まれた状態になり，この状態で画面上をクリックすると，そのクリックした位置の色が取得され，色選択ボタンに色の名前が設定される．再度ボタンを押すと色を確定した状態になる．確定した状態で再度ボタンを押すと，選択が解除される．

3.5 透明シートの反転

反転モードで透明シートを用いた場合，透明シートの外部の特定色が透明になる．操作メニューのシート反転ボタンを押すとことで，シートの内側/外側の対象色を透明にするかどうかを設定できる．反転ボタン中に「■」が表示されている場合はシートの内側の対象色を透明にし，「■」が表示されていない場合はシートの外側の対象色を透明にする．図 7 は，シートの内部の赤色を表示している例である，図 1 の状態でシート反転ボタンを押した結果である．

反転モードを実現するために 2 つのシートを用いる．1 つ目の透明シートは，フルスクリーン化してデスクトップ画面上に固定する透明シートであり，デスクトップ画面全体を画像処理した画像を表示する．2 つ目の透明シートは，ユーザーが自由に動かせる透明シートであり，画像処理していないデスクトップ画面の画像を表示する．

3.6 シートの追加・削除

操作メニューの「+」が書かれた透明シート追加ボタンを押すと，画面上に透明シートが追加される．操作メニューの「×」が書かれた透明シート削除ボタンを押すと，全透明シートに「×」ボタンが表示されるので，削除したいシート上の「×」ボタンをクリックすると，そのシートが画面

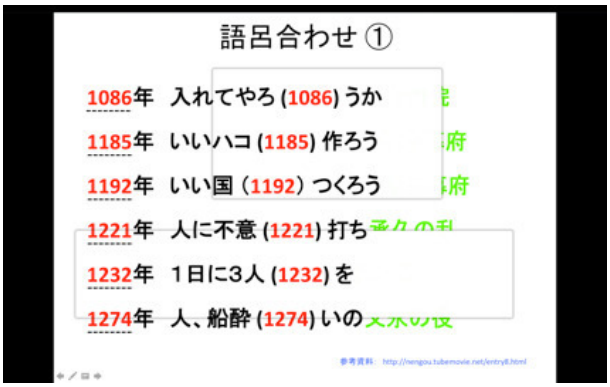

図 8 複数の透明シートを表示して，それぞれの緑色の部分を隠している例.

Fig. 8 The transparent sheets hide green colored characters under the sheets.

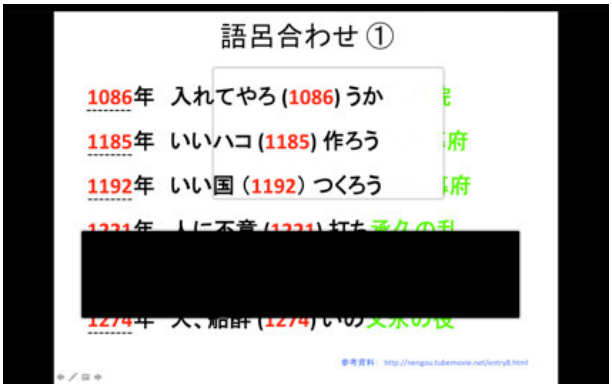

図 9 選択した透明シートを黒色に塗りつぶして，背後のコンテンツを隠している例.

Fig. 9 The user can change the transparent sheet to a black colored sheet.

から削除される．再度，透明シート削除ボタンを押すと全透明シート上の「×」ボタンが消える．図 8 は，複数の透明シートを表示して，それぞれの緑色の部分を隠している例である．

3.7 黒シート化

黒シート化したいシートを選択して Ctrl + Shift + W キー を押す．元に戻す時は再度，同じキーを押す．図 9 は，選択した透明シートを黒色に塗りつぶして，背後のコンテンツを隠している例である．

3.8 複数シートを組み合わせた応用

2 つの透明シートを利用して，一部分だけ表示するという利用も可能である．方法は，1 つの透明シートを黒シート化してフルスクリーン化し，もう 1 つの透明シートを覗き穴として利用する．図 10 は，複数シートを組み合わせ「覗き穴」を実現した例である．

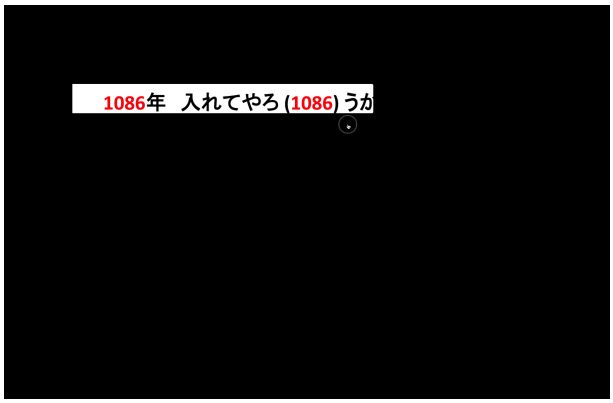

図 10 複数シートを組み合わせ「覗き穴」を実現した例

Fig. 10 Two transparent sheets realizing a peephole.

4. 考察

MagicalSheet と現実の赤シートを視覚的，領域的の側面から比較した．視覚的な側面では，現実の赤シートを使って赤文字を隠す方法では，その性質上，シートで覆う領域全てが赤くなってしまい他の字も見づらくなる，隠す文字はシートの色に類似した色でなければならない，長時間利用していると，視界に余韻が残ってしまう点がある．MagicalSheet では，対象とする色を背景色で塗りつぶすため，対象とする色を持つ部分以外に影響しない．対象とする色を持つ部分以外の領域は，シートで覆う前と後で変化することがないため，現実の赤シートのように見づらくなることはない．対象とする色は，任意に設定可能であるため，現実の赤シートのように，隠す文字はシートに類似した色である必要はない．

領域的な側面では，現実の赤シートでは，シートのサイズが決まっており，変更することができない．MagicalSheet では，シートのサイズを縮小・拡大，シートの追加・削除が可能である．例えば，複数行あるコンテンツの中から一行だけ隠したい場合は，現実の赤シートでは，その行に合う様にシートを切るなどして別に用意する必要があるが，MagicalSheet のシートはその行に合う様にサイズを変更することで対応できる．複数のシートを用いて，穴埋め問題のような利用方法も可能である．具体的には，単一色の文字のみで作成されたコンテンツであっても，隠したい場所ごとに，シートを追加・配置して，単一色を透過対象の色に設定することで，シートを重ねた部分のみが消え，穴埋め問題のように利用できる．MagicalSheet のシートは，シートの領域内の対象色を透明にするモードだけでなく，シートの領域外の対象色を透明にするモードもあるため，シートで覆った部分だけ対象色を表示するといった利用も可能である．

MagicalSheet のソフトウェアの利点として，既存のアプリケーションの変更は必要無い点である．デスクトップ画

面を撮影し画像データを処理するため，既存のアプリケーションの描画処理を拡張する必要は無い．例えば，プレゼンテーションソフトでスライド作成しプレゼンテーションモードで表示，イラストソフトでのイラスト作成し画像ファイルとして出力して画像閲覧ソフトでフルスクリーン化して表示することで，MagicalSheet のコンテンツとして利用することができる．MagicalSheet は，背景色は単色，文字色は赤・緑・青であるコンテンツを対象とするが，文字色については赤・緑・青以外の色に対応させるため，デスクトップ画面上の任意箇所の色を抽出して，それを対象の色とすることができる．一般的な資料作成ソフトウェアでは印刷機能および印刷プレビューもあるため，その機能を用いて表示したプレビュー内容を MagicalSheet のコンテンツとして利用することもできる．また，Web ブラウザ上で表示しているインターネット上に存在する既存のリソースを MagicalSheet のコンテンツとして再利用することもできる．Web カメラとプレビュー用のソフトウェアを用いて，メモ帳，ホワイトボードに手書きした内容や既存の印刷物を Web カメラで撮影しデスクトップ画面上に表示することで，MagicalSheet のコンテンツとして利用することもできる．

MagicalSheet の透明シートを複数利用した応用例として，透明シートで覆った部分だけコンテンツを表示し，それ以外の部分を黒で塗りつぶす例を挙げた．この応用は，透明シートで覆った部分だけしかコンテンツが見えないため，聴衆などのフォーカスコントロールに利用できる．現在は黒シート化したシートは不透明だが，透明度を設定して，フォーカスしてほしいとこを明るく表示し，それ以外の部分を暗く表示するといったことも可能である．

MagicalSheet の画像処理では，特定色を消すときに単色に置き換えているため，コンテンツの背景色は単色のみに対応している．コンテンツの背景色が単色であることを前提とし，文字が画面を占める面積が背景色の占める面積よりも小さい場合，最も面積を占める色を背景色として自動的に設定することが可能である．または，背景色もユーザーが画面上から選択できるようにすることも可能である．背景が模様の場合は，コンテンツの表示部分で覆われた部分の背景を推定することができないため利用できない．

MagicalSheet は, シートのクリック時にスクリーンショットを撮影し，特定色を消す画像処理を行う．画像処理後のシートの移動・縮小・拡大では，クリック時に処理した画像を利用する．そのため，MagicalSheet は動画など描画内容が動的なコンテンツには不向きであるが，暗記学習用のコンテンツではコンテンツの性質上，内容が静的であるため，この用途では問題ないと考える．

MagicalSheet は，最前面表示とデスクトップ画面撮影機能を提供できる SWB 上で動作する．Google Chrome はデスクトップ画面のスクリーンショットを提供する API も存在するため，画像データ取得し透明処理を行い表示することは可能である．ただし，デスクトップ画面撮影機能を提供する Web ブラウザはフルスクリーン表示でない限り，OS のウィンドウ枠が表示されてしまう問題がある．

5. おわりに

本研究では，PC のデスクトップ画面上の指定した領域の特定の色を消すことが可能である MagicalSheet を試作した．ユーザーは，MagicalSheet を用いることで赤シートを用いた暗記学習を PC のデスクトップ画面上で行うことが可能となる．デスクトップ画面上のある領域内の特定の色を消すためには，その領域に表示されたアプリケーションの描画処理を拡張する必要があり困難である．MagicalSheet では，デスクトップ画面を撮影し，撮影した画像から特定色を消す処理を行い，デスクトップ画面上に重ねて表示することで実現した．今回は PowerPoint を暗記学習用コンテンツの作成環境として利用したが，文字の配置，文字色・背景色の編集操作できるアプリケーションであれば同じく作成環境として利用可能である．

参考文献

[1] E.A.Bier, M.Stone, K.Pier, W.Buxton, T.DeRose Toolglass and Magic Lenses: The See-through Interface, SIGGRAPH ’93, pp.73-80, 1993.
[2] Adam Fourney, Ben Lafreniere, Parmit Chilana, Michael Terry: InterTwine: creating interapplication information scent to support coordinated use of software, UIST ’14: Proceedings of the 27th annual ACM symposium on User interface software and technology, pp.429-438, 2014.
[3] Tom Yeh, Tsung-Hsiang Chang, Robert C. Miller Sikuli: using GUI screenshots for search and automation, UIST ’09 Proceedings of the 22nd annual ACM symposium on User interface software and technology, pp.183-192, 2009.
[4] Morgan Dixon, James Fogarty: Prefab: implementing advanced behaviors using pixel-based reverse engineering of interface structure, CHI ’10 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp.1525-1534, 2010.
[5] Suporn Pongnumkul, Mira Dontcheva, Wilmot Li, Jue Wang, Lubomir Bourdev, Shai Avidan, Michael F. Cohen: Pause-and-play: automatically linking screencast video tutorials with applications, UIST ’11 Proceedings of the 24th annual ACM symposium on User interface software and technology, pp.135-144, 2011.
[6] Andrea Bianchi, So-Ryang Ban, Ian Oakley, Designing a Physical Aid to Support Active Reading on Tablets, CHI ’15 Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, pp.699-708, 2015.
[7] Julian Seifert, Sebastian Boring, Christian Winkler, Florian Schaub, Fabian Schwab, Steffen Herrdum, Fabian Maier, Daniel Mayer, Enrico Rukzio: Hover Pad: interacting with autonomous and self-actuated displays in space, UIST ’14 Proceedings of the 27th annual ACM symposium on User interface software and technology, pp.139-147, 2014.