グリーンクロマキーを使ったコンテンツ作成について

常三島技術部門
情報システムグループ　　片岡　由樹 (KATAOKA Yoshiki)

１．はじめに

徳島大学理工学部では毎年 8 月上旬に地域貢献として「科学体験フェスティバル in 徳島」を実施している。第 23 回（令和 1 年度，メインテーマは「ふしぎワールド」）の中の一つのブースとしてカメラやプロジェクタを使ったブースを企画した。そのコンテンツとしてカメラを使い，グリーンクロマキーを応用したコンテンツを作成した。その際の技術的な事項や失敗談などを報告する。

２．グリーンクロマキーとは

グリーンクロマキーとは，カメラ等により撮影した映像を合成する時に使用する技術である。日常的に公共放送の天気予報において天気図をバックに気象予報士が解説している光景をよく見かけると思う。緑色をした背景スクリーンの手前に人物を配置し，映像を撮り，その映像の緑の部分を他の映像に差し替える・合成する技術である。技術的には緑である必要はなく，人物を合成する事が多いので肌の色や服の配色を考慮し緑色が使用される事が多い。グリーン以外にはブルーが次に多いくらいだろう（図１）。

３．コンテンツの構想

カメラを使ったコンテンツでグリーンのスクリーンを用意して画面合成するのは簡単だ。スクリーンを用意せずに人物だけを切り出し背景として他の画像を画面合成する手段はコロナ禍の影響で Web 会議アプリを使うとよく見かけるものになっているだろう。スクリーンを用意せずに人物だけを切り出し背景として他の画像を画面合成する手段をブースのコンテンツとして用意していた。それは背景として FIND/47 というプロジェクト[1]で公開されている日本各地の情景写真を採用し，バーチャル旅行を体験する趣向にした。FIND/47 とは Wikipedia によると経済産業省が 2020 年東京オリンピックを視野に，日本各地の風景写真素材の公開により訪日外国人旅行需要を高める目的で開始された写真プロジェクトである。観光予報プラットフォーム推進協議会が運営して，写真は全てクリエイティブコモンズライセンス（表示 4.0 国際）になっている。ライセンス順守の為に使用した写真一覧をパネル（図２）にして会場に設置した。

(a)　　(b)　　(c)

図１　グリーンクロマキーによる合成例
(a)スタジオ撮影，© PictureYouth [CC BY 2.0]
(b)背景画像，© Nicholas A. Tonelli from Pennsylvania, USA[CC BY 2.0]
(c)合成画像，© PictureYouth [CC BY 2.0]

図２　写真一覧のパネル

バーチャル旅行だけではあまり面白くないのではないかと思い，緑の物体（以下，オブジェクト）を持ち背景に隠れた映像から宝探しするものを思いついた。その背景として同様にブースのコンテンツとして用意した徳島大学マスコットキャラクター（図３）を動かしたアプリ[2]の画面を活用する事にした。

図３　徳島大学マスコット
キャラクターとくぽん

４．背景コンテンツ

背景コンテンツとして徳島大学マスコットキャラクターを動かしたアプリを活用する事にしたが，当然編集する必要がある。プログラムは Processing を使ってコーディングしている。幸いアプリではキャラクターをクラスで定義（クラスのファイルも別に記述）していたので流用は容易であった。キャラクターを動かすのがメインではないので尻尾を振り歩く単純な動作にした。また，画面を 15 分割（3x5）にして一箇所に動くとくぽん（可読性の為に以下，カタカナ表記）を配置した。頭と尾を振りながら歩行するトクポンであるが，正面と背面の２種類用意した。

他の箇所には動かないトクポンや岩石のイラストや徳島大学に居そうな動物（猫やカラス），何もない空間を配置させた。20 秒経過すれば「動くトクポン」を再配置するようにした。配置場所はランダムである。何もない空間やダミーとなる岩石・動物の種類や数も同様にランダムに変更される。図４に背景コンテンツのサンプルを表示する。実際には後述するが画面上には表示していない。

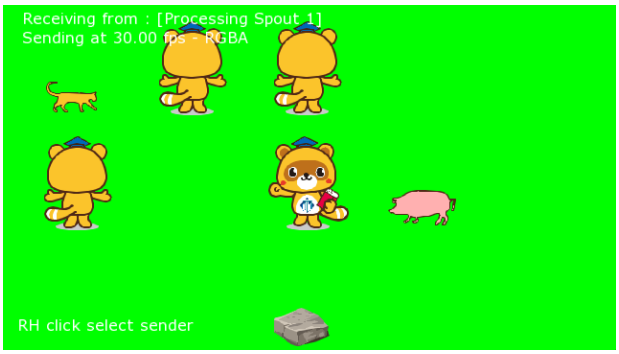

図４　背景コンテンツ
（SpoutToNDI のキャプチャ）

５．グリーンクロマキーによる合成

カメラ画像からグリーンの箇所を背景にするというクロマキー処理であるが，プログラミングすると大変だ。もちろん画面描画の draw 関数内で PImage のラスタ走査によるピクセルごとの処理を loadPixels 関数や pixels 配列，updatePixels 関数を用いて実装する事も考えたが，背景コンテンツは基本的に java ランタイムで実行されるので処理が遅い。そして細かい修正の手間が大きい。動作させるパソコンのメモリにとっては厳しいかもしれないがほかのアプリケーションを使って実現する事にした。そのアプリケーションとは OBS Studio（以下，OBS）である（図５）。

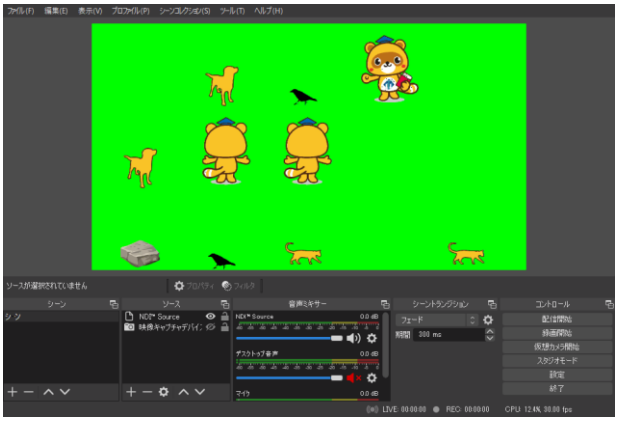

図５　OBS Studio

ソースとして映像キャプチャデバイスを追加して，Webカメラの映像を手前に表示する。そして，配置的にカメラ映像の後方に背景コンテンツのウィンドウキャプチャを追加する。そのうえで画面いっぱいにサイズを調整しておく。映像キャプチャデバイスのプロパティではなくフィルタ（図６）に変更を加える。映像キャプチャデバイス項目を選択して，右クリックしてフィルタを選択する。エフェクトフィルタとしてクロマキーを追加する。フィルタ適用前と適用後の例を図７，図８に示す。

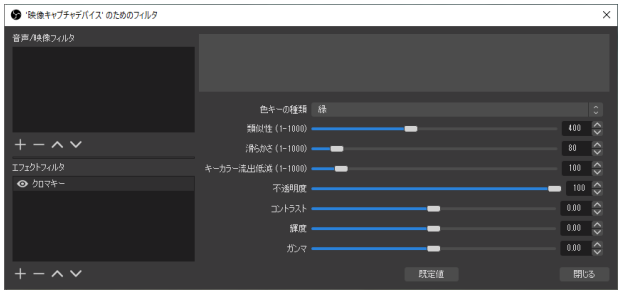

図６　エフェクトフィルタ

図７　フィルタ適用前

図８　フィルタ適用後

６．背景コンテンツの取り込み

執筆段階の今から思えば他の方法はいくらでもある事に気づく。イベントに向けての準備段階では気づかず，変わった方法で対応する事になったのだが，順に説明しながら弁明していこう。

OBS において背景コンテンツのウィンドウキャプチャを追加するのだが，背景コンテンツは全画面表示で作られている。アクティブなアプリケーションを切替（Alt+Tab）すればいいのだが，操作できなくなり取り込めないと思ってしまった。マルチモニタ環境にすれば問題ないのであるが，解決方法は思いつかない。イベントでの動作環境でマルチモニタ環境にする予定もなかったというのもある。後で気づくのだが，たとえシングルモニタでも OBS の出力（プレビュー）を手前に表示すればいいだけだ。アプリケーションの切替えという方法を思いつかなかった。普段からマルチモニタ環境にしていなかった事もあり，OS のショートカットなどを熟知していなかったのもあり，思いつかなかったと言い訳しておこう。

他にもタスクビューで仮想ディスプレイを使うという方法もある。HDMI ダミープラグによりダミーディスプレイを使うという方法もある。

背景コンテンツの全画面表示を止めて，ウィンドウ表示にする事にして，ウィンドウフレームを OBS において出力画面より大きくして表示領域から外せばよかった。イベントに向けての準備段階では OBS について不慣れだった為にサイズ変更や配置がかなり自由に変更できる事を学習する前で，その手段も思いつかなかった。

結局，他の PC で実行した画面が取り込めたらなぁと漠然と思い調査をしたところ，画面共有に関する技術がある事を知り，面白く感じて試してみた。それはネットワークを介して画面共有する技術で Mac では Syphon，Win なら Spout という技術である。私の開発環境および動作環境は Windows であるので Spout に関してさらに調査をした。Processing で Spout を使うためのライブラリが公開されている。これを使う事により Spout により画面が出力できる。その画面を受け取る方である OBS で Spout を扱えたら無事解決しそうであるが，OBS では対応していなかった。同じような画像共有技術である NDI についてはプラグインを入れる事により対応できることが

分かった。OBS で Syphon や Spout を将来的にも取り扱わないという記事を見かけたのでOBS では NDI 一択のようだ。それならば Processing で NDI を扱えたらと思ったのだがそのようなライブラリ等は公開されていない。でも何とかできないかと調査をすると Spout を NDI に変換するアプリケーション（SpoutToNDI）が存在している。それらを組み合わせればいい。どうせなら同じ PC 上でいいと判断し，それらすべてを PC にインストールしプログラミングを修正した。

７．実行環境のまとめ

７．１　ハードウェア

以下のハードウェアにて実行した。

・Windows パソコン　1 台
・モニタ　1 台
・Web カメラ
（モニタの正面が映るように配置）
・キーボード・マウス（起動時のみ）

７．２　ソフトウェア

以下のソフトウェアにて実行した。Ndi のランタイムをインストールする必要があった。

・Processing
・Processing addon Minim
・Processing addon Spout
・OpenBroadcastStudio(OBS)
・OpenBroadcastStudio Plugin nbi
・SpoutToNDI

７．３　プログラム（Processing）

Spout は画面に表示する必要がなく createSender 関数で送信先を作成し，sendTexture 関数で PGraphics インスタンスを引数にすればよい。したがって画面には経過時間のみを表示するプログラムにした。
配置換えのタイミングで鼓の音をならしてタイミングがわかるようにした。送信された画面は SpoutToNDI プログラムで確認できる。

７．４　プログラム（SpoutToNDI）

Spout はソース名を指定して送信されているので，ソース名を指定して受取り ndi に変換されて出力されている。Ndi ソース名も変更はなしで，30fps の RGBA で出力する。

Spout で送信された画面は 1080p である。SpoutToNDI ソフトウェアでの表示はウィンドウサイズに合わせて画面が自動で拡大縮小表示される。変換後のデータは 1080p のままである。

７．５　OBS 設定

グリーンクロマキーによる合成の説明と同様にソースとして NDI ソースを追加しておく。自身の PC 名と SpoutToNDI で確認した Ndi ソース名の組み合わせから得られる NDI ソース名をプロパティにて指定する。

次に Web カメラを映像キャプチャデバイスとして追加し，フィルタにクロマキーを指定する。あとは OBS のプレビュー画面をモニタいっぱいに全体表示する。

７．６　バッチファイル

実行しやすくするためにバッチファイルを用意した（表１，表２）。イベント中にトラブルにより再起動などが必要な時にも手早く復帰できる事が期待できる。起動するのは Processing と OBS と SpoutToNDI である。Processing は実行形式でスケッチを指定して最小化して起動した。OBS は OBSPortable を使用し設定をコレクションとして保存しておいた。タスクトレイに最小化してコレクションを指定して起動させた。OBSPortable は一部だけパラメータ有りのコマンド起動に対応している。複数のアプリ起動などの場合は OBSPortable.exe ではなく実際のアプリ実行ファイル本体の obs64.exe にてコマンド起動をする必要がある。SpoutToNDI は設定を変更しなければ前回の設定で動作するので起動するだけである。

終了するバッチファイル（表１）も作成したが，処理が少し乱暴であるので動作は保証できない。

表１　終了するバッチファイル

```
start taskkill /IM obs64.exe
start taskkill /IM "Spout to NDI.exe"
start taskkill /F /IM processing-java.exe /T
```

表２　開始するバッチファイル

```
if not "%~0"=="%~dp0.¥%~nx0" (
  start /min cmd /c,"%~dp0.¥%~nx0" %*
  exit
)
cd C:¥Fes¥

start /min C:¥Fes¥processing3¥processing-java.exe --
sketch= "C:¥Fes¥Data¥OBS_treasure¥Treasure" --run

start /min C:¥Fes¥OBSPortable¥OBSPortable.exe --mi
nimize-to-tray --collection "2019Tresure"

"C:¥Fes¥SpoutToNDI¥bin¥Spout to NDI.exe"
```

８．体験の仕方

モニタの前で緑の物（オブジェクト）を手に持ってカメラに映る。何も持たなければ自分自身が映っているだけである。左右反転して鏡と同じような表示にしている。オブジェクトを持っていると緑色の背後に図柄がでてくる。刻々と変化していく図柄から動くトクポンを見つけるようにオブジェクトを右左，上下に移動させる。制限時間内に見つけると達成感が得られるという体験だ。

オブジェクトとしては緑色の箱(仙台土産)や Web カメラ（logicool）が入っていた箱や緑色を印刷した紙(ラミネート加工)や緑のスポンジなどを用意した。

９．イベントでの状況について

イベントにて多くの児童がブースを体験していった。どのように体験するのかを掲示しておいたが，やはり説明が足らなくてスタッフが必要であった。なかには緑色の服を着ている児童がいておもしろい映像になっていた。

問題点も色々と見つかった。私の開発環境である居室においてはとくに緑色のオブジェクトは問題なく使えるものだけを厳選したつもりであった。設置場所が変わり，調光や部屋自体の明るさもあり，一部でうまくクロマキー領域と認識されなかった。OBS のクロマキーの色をカスタム色にして微調整できるが，カメラでとらえた認識する色に空間的・時間的な影響で明るさのばらつきが大きく微調整をしても状況は変わらない。また，ラミネート加工をした紙は手軽に色々な形の透過領域をつくれるので数種類用意していたが，光の反射の具合によりカメラでの映像が緑になっていなくて白色に，つまり光の反射の色に認識された。これについてはオブジェクト自身を発光させるように改善予定である。

もう一つの問題点は児童が何気なくオブジェクトをカメラ近くに持っていき画面全体を緑にしてしまう不正をする事がある。今回は特にカメラ前で体験していただく形だが，次に実施する時があるなら立ち位置の場ミリを設置予定である。

１０．さいごに

コンテンツを完成させるための技術的な課題が発生し，その解決方法を試行錯誤するわけであるが，必ずしも最適なものを選べるとは限らない。技術力と経験などでより良いものが選択できるようになるのだろう。今回はかなり遠回りになったがコンテンツは作成できた。遠回りになったからこそ新しい技術や知見に触れる事ができた。執筆時点で他の方法があったと自分で気づくという事で，技術力も向上していると思いたい。

この報告は失敗ではないが反省点が含まれている。技術職員の記述する報告にはもっと失敗を掲載してもいいと思う。失敗をしないで成功するという事は優秀な技術者か，冒険・チャレンジをしていない人ではないだろうか。私自身には馬鹿な失敗も多いのだが，読者の方には笑っていただき，参考にできるところは参考にしていただければ幸いである。

参考文献

[1] https://find47.jp/

[2] 片岡由樹，マスコットキャラクターを動かせ，実験・実習技術研究会2020鹿児島大学プログラム・報告集，p43，2020