Rの基礎と画像データ処理

2024/04/22

HI4 45号 山口惺司

# 実験目的

動画を創作し動画の原理と制作技術の基礎を習得する。また、Rの基本的な使い方ができ、画像データを取り扱うことができる。

# 実験項目

1. Rの基本的な使い方と画像処理用パッケージのインストール
2. 動画制作
3. Rの画像処理パッケージを用いた画像データの処理

# 課題内容

1. Rの基本的な使い方と画像処理用パッケージのインストール

課題1-1 Rの基本的な使い方について、以下を実施する。

(a). ベクトルおよびデータフレームの使用例について、スクリプト、実行結果を示し、説明する。

(b). 自作関数の実行例について、スクリプト、実行結果を示し、説明する。

課題1-2 画像処理用パッケージのインストールと動作確認について、以下を実施する。

(a). 各自のPCにパッケージimagerをインストールする。

(b). 以下のスクリプトを実行して、画像データが表示されることを、各自のPCを用いてデモを見せる。

Rスクリプト

library(imager)

plot(boats)

1. 動画制作

課題2 複数の静止画像をフレームデータ(ビデオの1画像)として、時系列に並べて動画を制作する。

・フレームデータを動画制作ソフトに取り込み、10 秒以内で動画を制作する。このとき、必要に応じて新たなフレーム画像を作成し、加除修正する。

・完成した動画のフレーム周期（フレームレート）を変化して見え方（動きの滑らかさや面白さなど）を各自で評価する。

・制作した作品は、動画ファイル（.mp4）にして保存する。このときファイル名は、「出席番号・氏名－課題2」とすること。

1. Rの画像処理用パッケージを用いた画像データの処理

課題3

・課題2で利用したフレーム(ビデオの1画像)データをRに取り込み、画像処理パッケージを利用した処理を実施する。

・処理する画像は3枚

・なお、実施する処理は、各自の興味あるものをする。

# 課題2の考察

課題2の動画制作では「Krita」というソフトウェアを使用して動画を制作した。

10fpsで10秒つまり、100枚の画像で動画を制作した。

工夫した点はリアリティを再現した点である。

８枚の画像を図1に示す。

一見、同じ画像に見えるが、全部微妙に変わっている。

これは、スナイパーがスコープを覗いた時の手振れを再現したものになる。

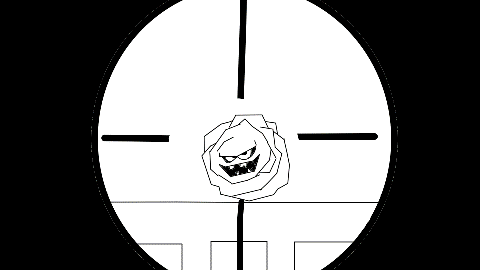
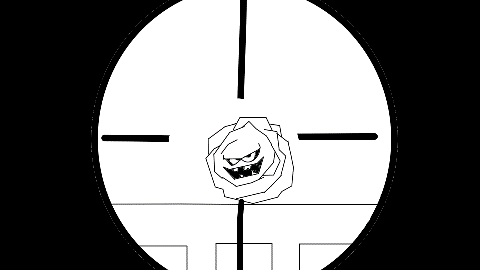
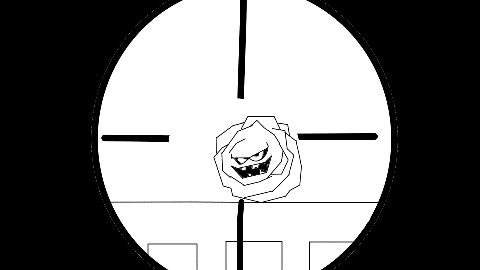
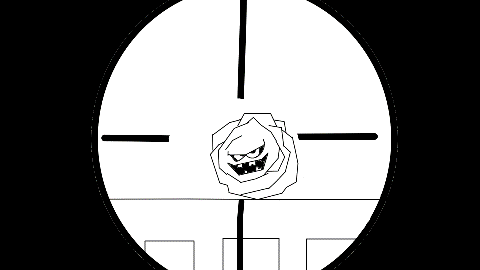
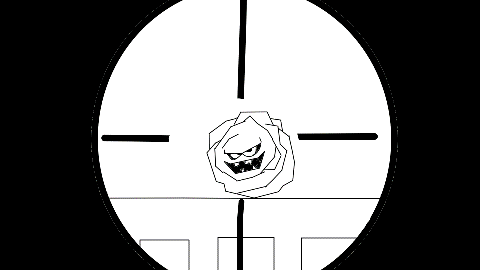
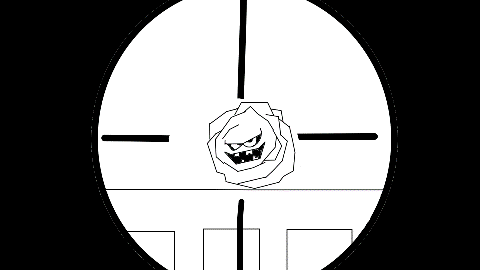
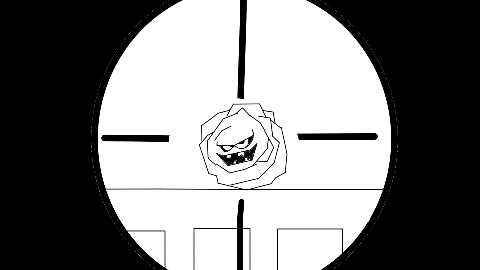
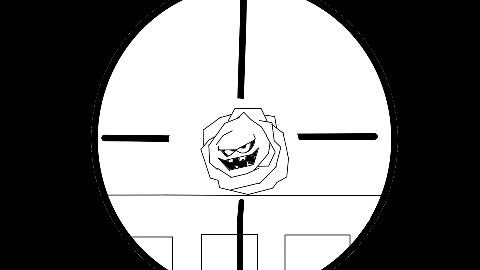
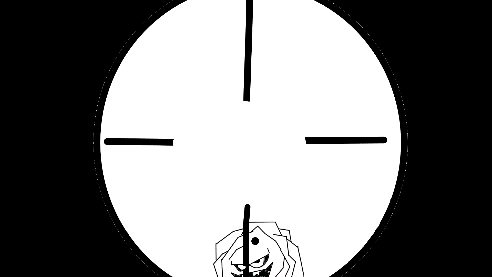
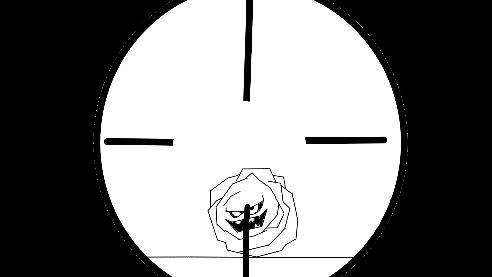
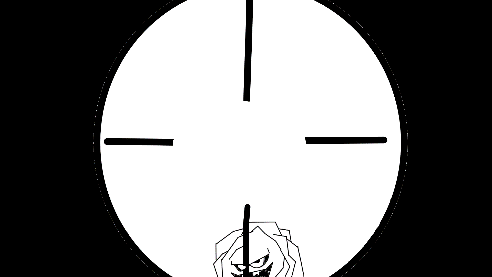
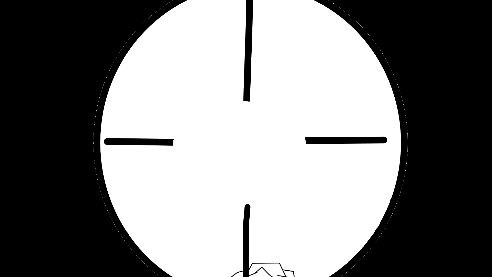
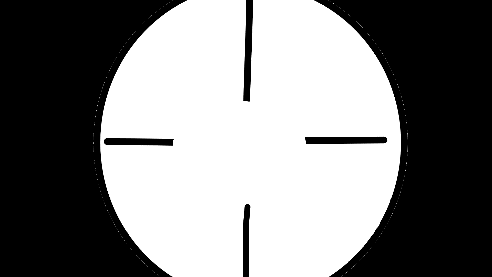


図1 53～61フレーム、手振れを再現した画像

また、銃を撃った時の反動を再現したフレームを図2に示す。

時計 が含まれている画像

自動的に生成された説明時計 が含まれている画像

自動的に生成された説明ダイアグラム, ロゴ

自動的に生成された説明

図2　72～79フレーム目、銃を撃った時の反動の再現

# 課題3の考察

ソースコードを以下に示す。

1 kadai3 <- function(){

2 library(imager)

3

4 img1 <- load.image("imgs/53.png")

5 img1 <- mirror(img1, "y")

6 dev.new()

7 plot(img1)

8

9 img2 <- load.image("imgs/36.png")

10 dev.new()

11 img2 <- isoblur(img2, 10) %>% plot(main="Isotropic blur, sigma=10")

12

13 img3 <- load.image("imgs/37.png")

14 img3 <- deriche(img3, 2, order=2, axis="x")

15 dev.new()

16 plot(img3)

17 }

図3 画像処理Rスクリプト

では、それぞれの画像処理について解説する。

まず、img1について、53フレーム目の画像処理前の画像を図5、画像処理後の画像を図6に示す。

ダイアグラム

自動的に生成された説明 ロゴ が含まれている画像

自動的に生成された説明

図4 フレーム53画像処理前 図5 フレーム53画像処理後

これは図3、5行目のmirror関数を使用して上下を反転させている。

mirror関数の使い方は以下の通りである。

mirror(im, axis)

imは画像の引数でaxisは反軸の引数である。axisの値をyからxに変えると、左右の反転もできる。

次に、img2について、36フレーム目の画像処理前の画像を図7、画像処理後の画像を図8に示す。

ダイアグラム

自動的に生成された説明 ダイアグラム

自動的に生成された説明

図6 フレーム36画像処理前 図7 フレーム36画像処理後

これは図3、11行目のisoblur関数を使用して、画像をボケさせている。

isoblur関数の使い方は以下の通りである。

isoblur(im, sigma)

imは画像の引数、sigmaはボケの標準偏差である。sigmaの値を大きくするとボケが大きくなる。

最後に、img3について、37フレーム目の画像処理前の画像を図9、画像処理後の画像を図10に示す。

ダイアグラム, 設計図

自動的に生成された説明 ダイアグラム, 図形

中程度の精度で自動的に生成された説明

図8 フレーム37画像処理前 図9 フレーム37画像処理後

これは図4、14行目のderiche関数を使用してガウシアンフィルタをかけている。

deriche関数の使い方は以下の通りである。

deriche(im, sigma, order, axis)

imは画像の引数、sigmaはフィルタの標準偏差、orderは0, 1, 2の値が入り、0はスムージングフィルタ、1,2は1次微分,2次微分によるフィルタ、axisは軸の引数である。

# 研究事項

**フレームレートについて**

フレームレートとは1秒間に表示される静止画の数を表している。

今回の実験で制作した動画は10fpsだが、普段我々が使用しているテレビやゲーム、映画などのフレームレートはいくつぐらいなのか気になったので調べた。

まず、映画は一般的に24fpsで作られている。映画のフレームレートは1920年代に決定された標準で、映画的な質感を与える効果がある。また、もし60fpsで映画を作った場合、安っぽく見えるらしい。

次に、テレビの映像、これは一般的に30fpsで作られている。

そしてゲームでは一般的に30~240fpsほどを使用する。なぜこれだけばらつきがあるかというと、ゲームによって要求されるフレームレートが違うからだ、ほとんど動きのないゲームだと低いフレームレートでも問題ないが、より高速なアクションを求められるゲームだとフレームレートが高くないと何をやっているかわからない。

また、フレームレートが高ければ高いほどPCが処理をするのに時間がかかるため、同じゲームでも人によって使用するフレームレートが違うこともある。

# 感想

・今回10fpsで10秒の動画を制作したが、24fpsで2時間の映画を作るアニメ関係の仕事に就いている人はとてもすごいと感じた。

・3年生の時にPythonで二値化や画像の反転のプログラムを書いたが、Rのimagerライブラリを使用することでプログラムを書く手間が省けることに感動した。

# 参考文献

https://educon.jp/knowledge/10242/#:~:text=24fps%3A%20%E6%98%A0%E7%94%BB%E3%81%A7%E4%B8%80%E8%88%AC%E7%9A%84,%E3%81%95%E3%82%8C%E3%82%8B%E3%83%95%E3%83%AC%E3%83%BC%E3%83%A0%E3%83%AC%E3%83%BC%E3%83%88%E3%81%A7%E3%81%99%E3%80%82