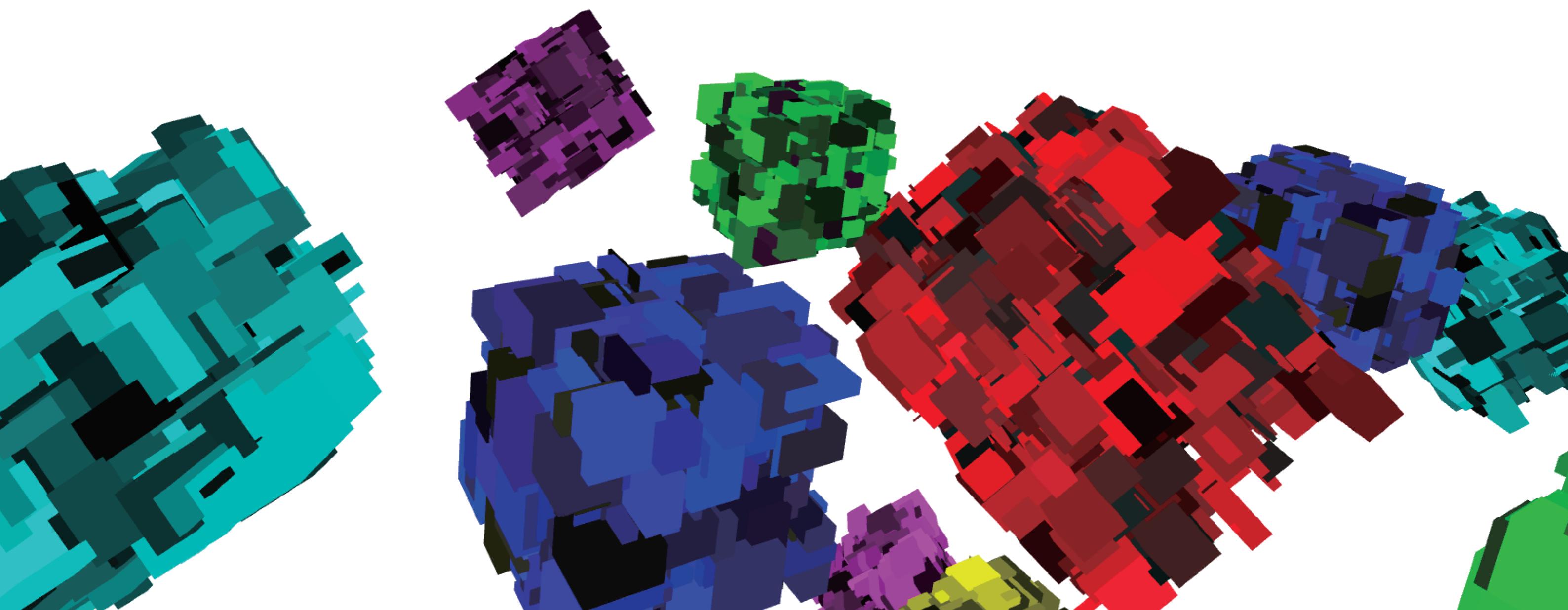


Ryo Nishikado: Portfolio / 2017/4~ 2019/7

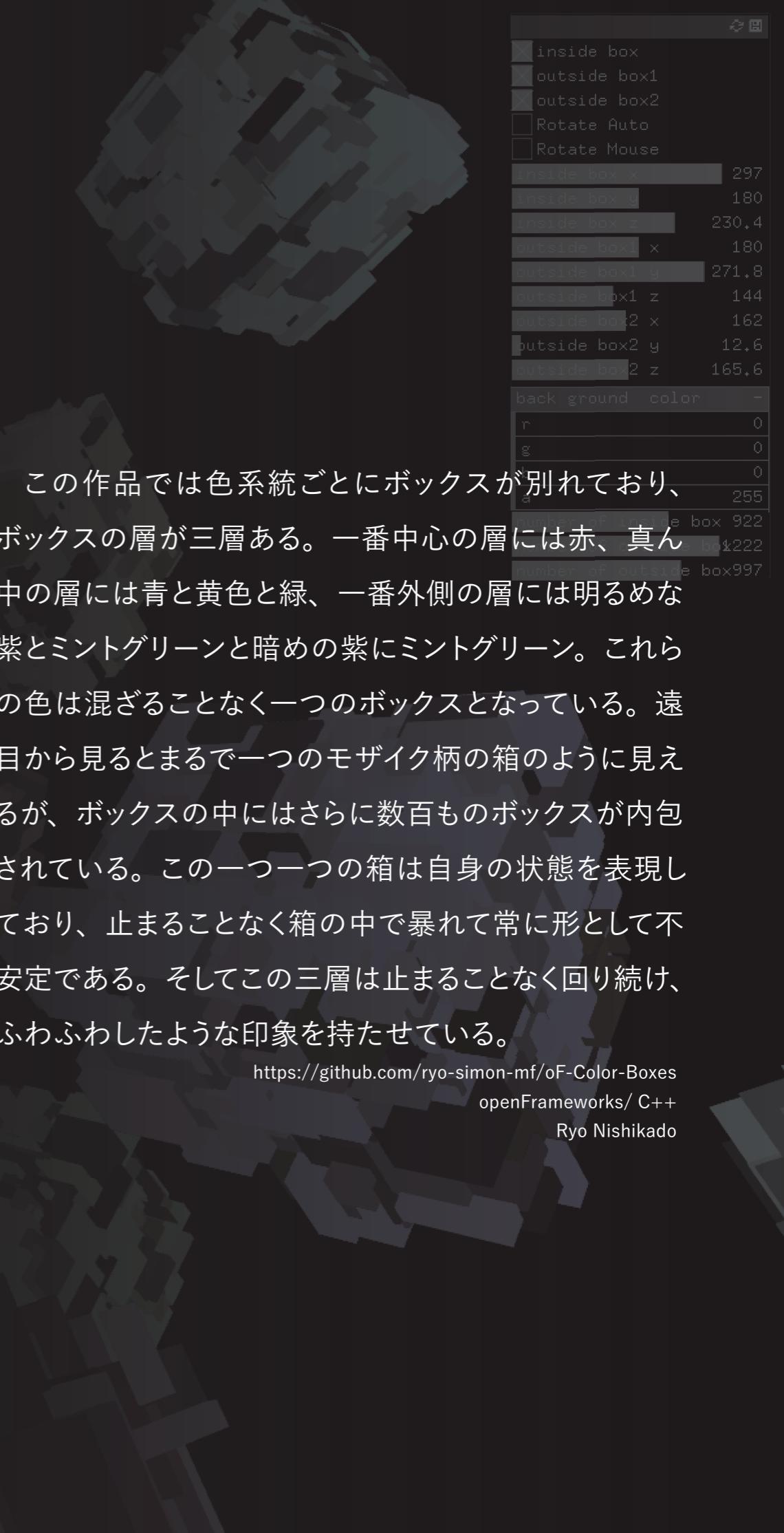
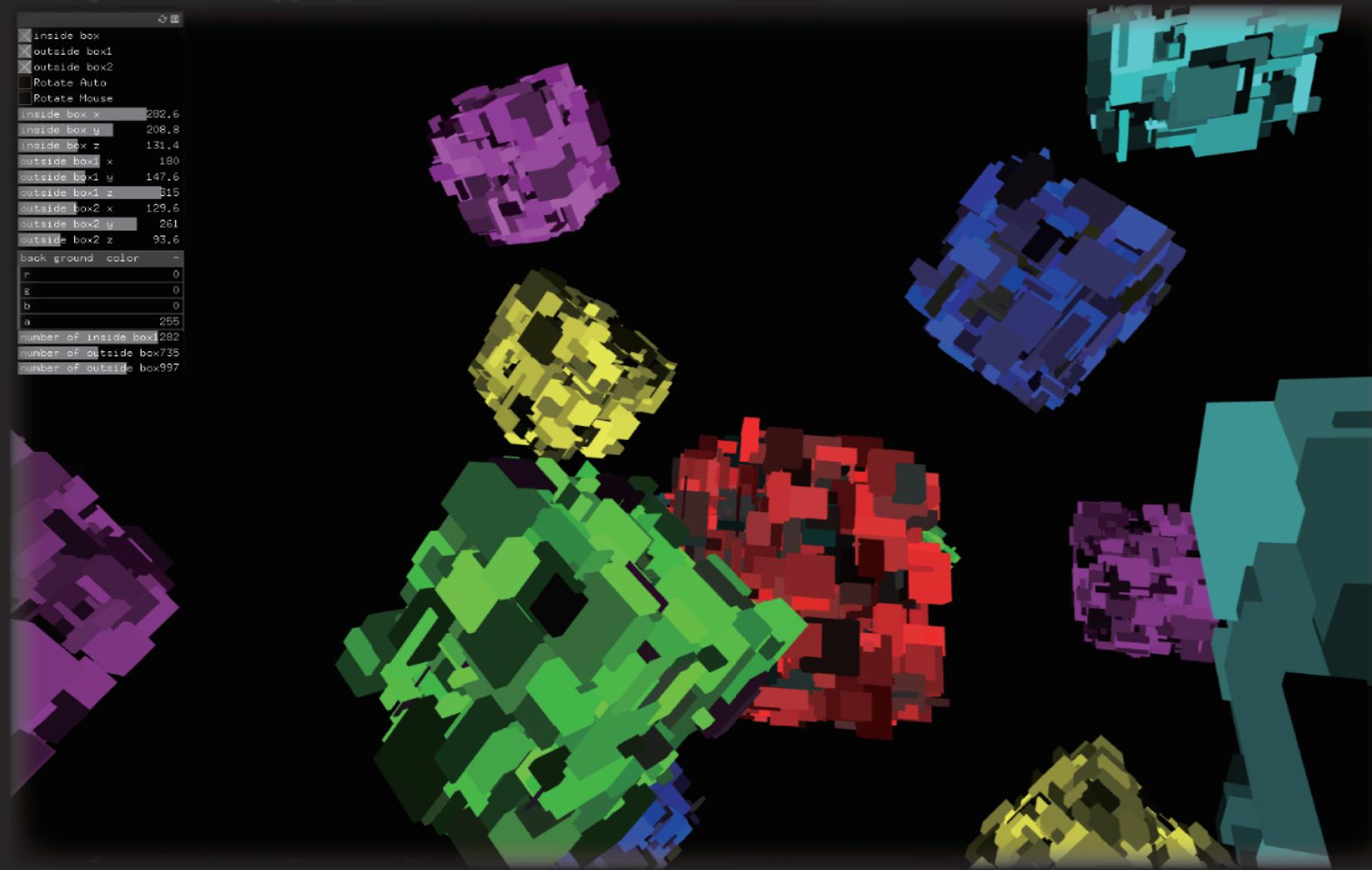
リョウ ニシカド:ポートフォリオ



```
1 #include "ofApp.h"
2
3 //...
4 void ofApp::setup(){
5     canvas.setResolution(1000, 1000);
6     gui.setup();
7     gui.add( toggle_P.setup("inside box", false));
8     gui.add( toggle_1.setup("outside box1", false));
9     gui.add( toggle_4.setup("outside box2", false));
10    gui.add( toggle_2.setup("Rotate Auto", false));
11
12    gui.add( int_slider_1.setup("number of inside box", 282));
13    gui.add( int_slider_2.setup("number of outside box ", 1000, 1, 0, 1500));
14
15    bHide = false;
16    open = true;
17    open1 = true;
18    open2 =true;
19    open3=true;
20    open4=true;
21
22    on = false;
```

Color Boxes

Color Boxes



```
import processing.core.*;
import processing.data.*;
import processing.event.*;
import processing.opengl.*;

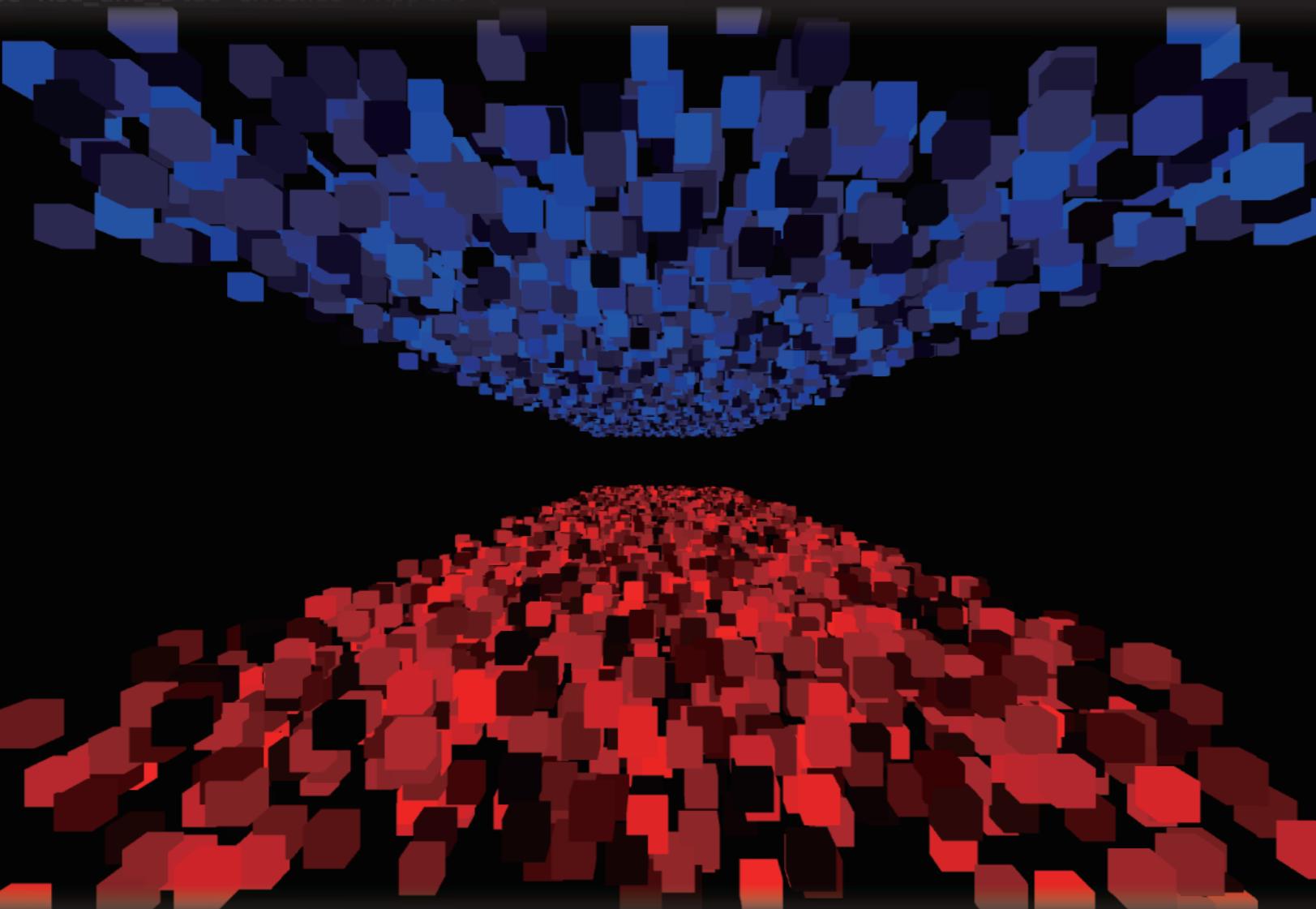
import java.util.*;
import java.util.List;
import java.io.File;
import java.io.BufferedReader;
import java.io.PrintWriter;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.IOException;

public class Red_and_Blue extends PApplet {

    int num;
    r_color;
    b_color;
    //_
    //boxes
    //_
    public
    //full
    //_
    //cbo
    //for
    //cbo
    //cbo
    //_
    for(i
        cbs
        cbs
    }
    public
    noStr
    backg
    smooth
    //_
    for(int i =0;i<cbsr.length;i++){
        cbsr[i].display();
        cbsb[i].display();
    }
    //_
    //for(boxes a:cbox){//boxes配列を1つずつaへ入力しdisplayする
    //a.display();
    //}
    //}
    //}
    class b_colorbox extends colorbox implements boxes{//colorboxの
        b_colorbox(float x,float y,float z){
            super(x,y,z);
        }
    }
}
```

Red and Blue

Red and Blue



赤を地面、青を天井に見立て、毎フレームごとにボックスの配置をランダムに変えて奥行きがあるパースのように座標を組み込むことにより、奥から手前の方に流动的に見えるよう表現している。奥から手前に流れてくるように見せることでトンネルをイメージさせるが、時間経過による変化はほとんど無いに等しい。一向に変わることがない、普遍的な同じような構造を目にすることで先の見えない不安感を感じさせる。

<https://github.com/ryo-simon-mf/Processing-Red-and-Blue>
Processing / Java
Ryo Nishikado

Clear File Vase

Clear File Vase



ただ水を入れてこぼさずその状態を保ったままでいる花瓶はおもしろくない。だけどただ水がこぼれる花瓶もおもしろくない。だとすればどのような花瓶がおもしろいか。それはある程度水を入れた状態を保持し、普通では考えられないこぼれ方をする花瓶だと思う。

この花瓶は約 1.2L の液体を入れることができ、一定時間水を入れた後に角の部分から噴水のように水が放出される。主な材料として、水を出す箇所を限定するために六角形に切ったクリアファイルと、それを接合するためにテープの 2 点のみを使用して製作した。

六角形に切ったクリアファイルの点が 4 以上、辺が 8 以上重なる箇所と折り曲げたときに鋭角になる箇所は構造上水が漏れやすい。よって、これらの箇所などの接合は、テープを用いた独自に考案した特殊な貼り方を用いることで水が漏れるのを一定時間防ぎ、また噴水のように水が放出するのをコントロールするとことを可能にした。

120×120×165mm(W×H×D)

Ryo Nishikado



Eye Have You

Eye Have You



昔は有線のインターネットが主であったが、今では無線でのインターネットが主流となっている。その影響により、今では昔よりもインターネットの象徴であった LAN ポートを目にするすることは少なくなっている。

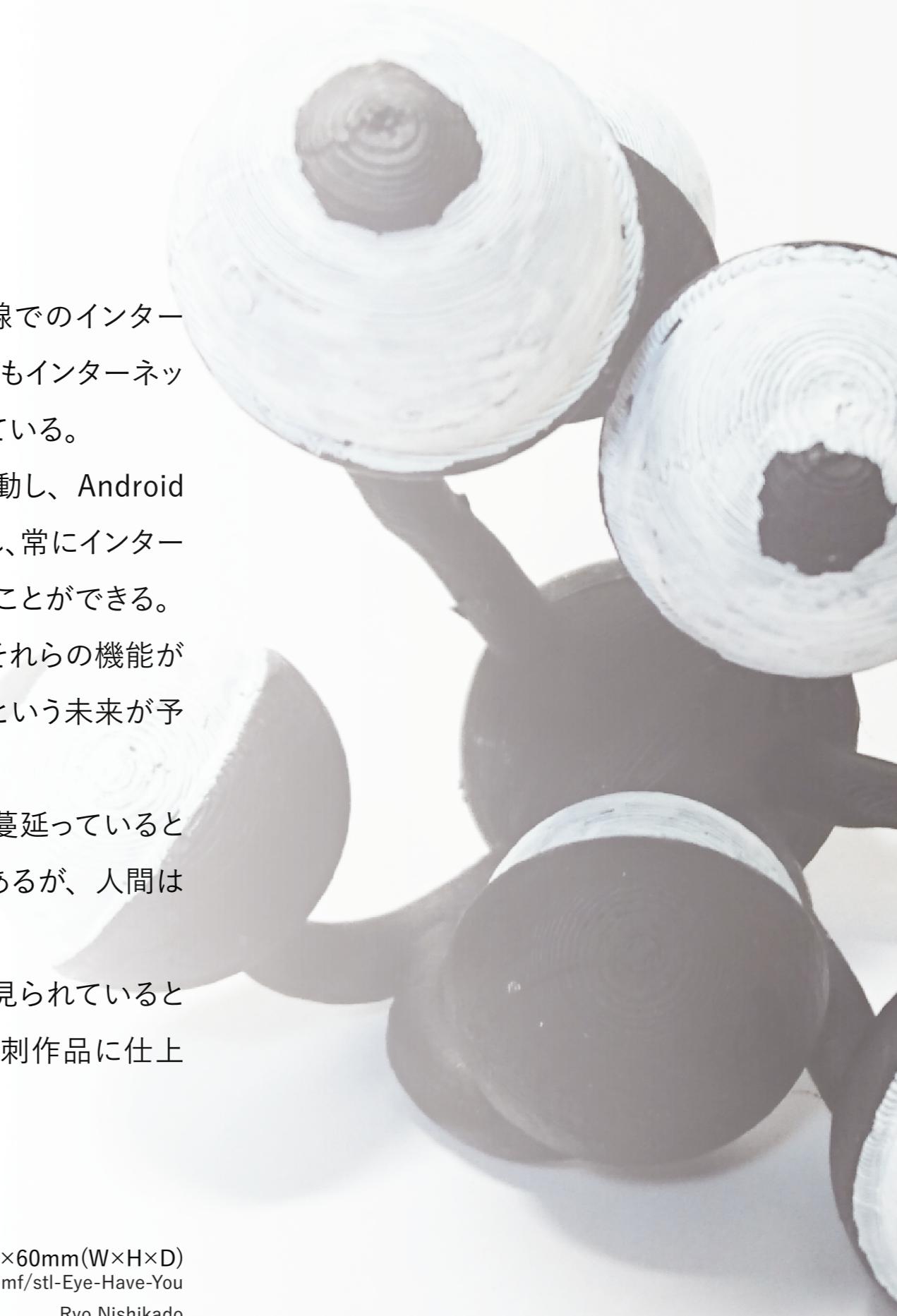
今では Apple 製品は「Hey!Siri!」と言えば Siri が起動し、Android 製品で「Ok!Google!」と言えば Google Assistant が起動し、常にインターネットに繋がり様々なことを調べたり、音楽を流したりすることができる。

またさらに Society5.0 における住宅の IoT 化によってそれらの機能が端末のみならず、家のどこにいても使用することができるという未来が予見することができる。

しかし、それは自分の身の回りに常にインターネットが蔓延っているということであり、インターネットに常に見られていることであるが、人間はそれを目視することができない。

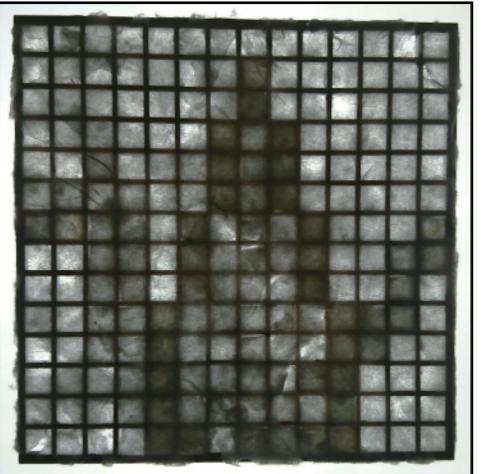
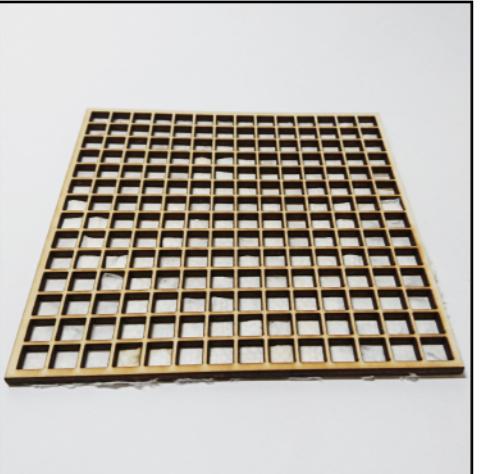
そして、この作品は我々現代人は常にインターネットに見られているという意味を込め、実用的なアタッチメントではなく社会風刺作品に仕上げた物である。

80×60×60mm(W×H×D)
<https://github.com/ryo-simon-mf/stl-Eye-Have-You>
Ryo Nishikado



Japanese Paper Door Display

漉き紙障子ディスプレイ



日本では古来から和紙を作るには紙漉きという技術を使われており、現在でもその紙漉きは伝統工芸として残っている。また、紙漉きでは主原材料であるパルプの量により光の漏れ具合を調節することができる。そして、私たちの身の回りでは、この紙漉きで作られた和紙は障子に使用されることが多い。

障子は日本で伝統的に使われていた部屋の仕切りであり、和紙の特徴を引き継いでいるため、光を拡散させてぼやかしながら透過させる。そしてその光を障子を通して拡散しほやかしながら透過させることによって、障子をはさんで離れた空間は少しだけ向こうの様子を想像することで空間の向こうを知覚させる「やわらかい空間認識」をしている。

この障子に見立てた作品は一見ただの正方形がずらずらと並んでいるが、光を透かすとある生物が浮かび上がる。

子供のころに読んだ日本の昔話を思い出して

そう、「鶴の恩返し」の鶴である。

150×150×15mm(W×H×D)

Soma S, Mizuki H, Ryo N



Pour Water

水を注ぐ



触覚とは皮膚や粘膜の表面に何かが触れた時に感じる人間の五感の中の感覚の一つである。

この触覚というのは人間の中でもどの感覚よりも先に出来上がるため、他の感覚に比べると改めて感じられること自体が希薄である。日常的な行動に対しても常に触覚は存在しているが、それに対して触覚を意識することは非常に少ない。そこで日常的に感じるであろうカップに「水を注ぐ」という行為を視覚的、触覚的に再体験させる。

Processing/ Arduino/ Java/ C++

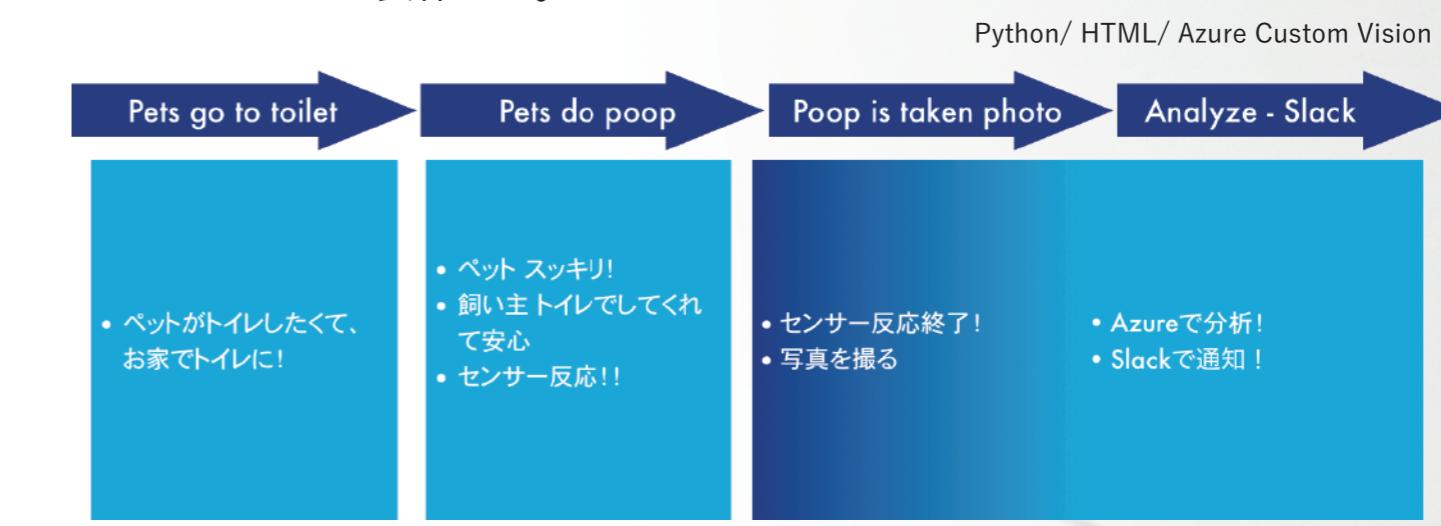
Ryo Nishikado

Toilecher



人間の健康に关心が寄せられているのと同様に、昨今ではペットの健康にも大きな关心が寄せられている。しかし、ペットと人間の共通言語が少ない現在、体調を把握する方法は人間に比べて非常に少ない。そこでペットから発せられる視覚的情報の微々たる変化からを継続的にログを収集することで健康管理ができるのではないかと思い至った。センサーやコンピューターの小型化により、連続的な観察を行うことが昔に比べて容易になったからである。

そこで、小型コンピュータである「Raspberry Pi」と Microsoft 社が提供しているクラウドサービスの Azure で提供される画像認識サービスである「Custom Vision」を活用してペットの健康管理をするシステム及びプロダクトを製作した。



https://github.com/ryo-simon-mf/submit-Practice_of_Open_Design
Ryo Nishikado