02

応用

音や画像で遊ぼう
—インタラクティブアプリケーションための C++ フレームワーク「Siv3D」—

鈴木 遼（早稲田大学 基幹理工学研究科 表現工学専攻）

Siv3D とは

Siv3D（http://siv3d.jp）は，音や画像を使ったプログラムや，マイク，Web カメラ，Kinect などさまざまなデバイスを使ったアプリケーションを，シンプルな C++ コードで開発できるフレームワークである．最新の C++ 規格を活用する，豊富なアプリケーション・プログラミング・インタフェース（API）により，お絵かきアプリは 9 行，Kinect を用いた人の姿勢のキャプチャは 15 行，ブロックくずしゲームは 28 行といったように，複雑なインタラクションを短いコードで記述できるのが特徴である．

現在 Windows 向けのソフトウェア開発キット（SDK）が無償で公開されているほか，対応プラットフォームを macOS と Linux にも広げた次世代版 Siv3D，「OpenSiv3D」の開発も進められている．

本稿では，Windows 版の Siv3D SDK を用いて，音や画像で遊ぶ 5 つのインタラクティブなアプリケーションを実装し，その仕組みを解説する．C++ プログラミングの文法や，Siv3D の細かい機能についての説明は割愛しているが，コードは 20 ～ 50 行と短いので，プログラミングの経験がない読者にも，音声処理や画像処理プログラミングの雰囲気や面白さが伝わるはずだ．

サンプルコードについて

本稿に掲載しているコードは，疑似コードや抜粋ではなく，Siv3D SDK をインストールした環境でそのままビルドできる完成形の C++ コードである．自分なりにカスタマイズしてみるのもよいだろう． http://siv3d.jp/ipsj17 に同じサンプルコードを掲載している．

コンピュータ画伯

画像処理技術の進歩は目覚ましく，昨今は白黒画像をカラー画像に変換する技術や，線画に自動的に色を塗ってくれるツールが注目を集めている．私たちも何かすごい画像処理のプログラムを作ってみたいものだが，一朝一夕には難しい．そこでまずは，誰でも簡単にできるアルゴリズムから始めるとしよう．それは「適当に描く」ことだ．

目標

コンピュータが，キャンバスのランダムな位置にランダムな色で丸を描き，絵を目標の画像に近づけていくプログラムを作ってみよう．丸を描く直前に絵を保存しておき，もし丸を描いて目標との類似度が離れてしまったら，保存していた絵に差し戻すことで，描けば描くほど目標に近づいていくようにしよう．

実装

2 つの画像がどれだけ類似しているかを，各ピクセルの RGB 成分の差の絶対値を合計することで求める Diff 関数を定義する．この値が小さいほど，2 つの画像は類似していることを意味する—①．Main 関数ではまず，コンピュータに描かせる目標の画像をユーザに選択させて開く．画像は画面内に収まるよう縮小する—②．続いて，同じ大きさの白い画像を用意し—③，その時点での目標との差を計測する—④．メインループ内では，最初に現在の画像を保存しておき—⑤，ランダムな位置に—⑥ランダムな色—⑦と大きさ—⑧で円を描き足す—⑨．Diff 関数で目標との類似度を調べ，差が縮まればそれを採用し—⑩，それ以外の場合は⑤で保存した画像に差し戻す—⑪．これをメインループごとに 100 回繰

図 -1　コンピュータに描いてもらう風景写真

図 -2　実行結果：描き始めてから数分後の様子

図 -3　実行結果：数十分後の様子．印象派画家を彷彿とさせる力強い色彩表現だ

り返し，最後の時点での画像を画面に描画する―⑫．これで，真っ白な画像から始まり，だんだん目標の絵に近づいていくプログラムが完成する（図 -1，2，3）．

```
# include <Siv3D.hpp>

// ① 2つの画像の差分を計算
double Diff(const Image& a, const Image& b){
  double d = 0.0;
  for (auto p : step(a.size))
  {
    d += Abs(int(a[p].r) - int(b[p].r));
    d += Abs(int(a[p].g) - int(b[p].g));
    d += Abs(int(a[p].b) - int(b[p].b));
  }
  return d;
}

void Main() {
  // ② 目標とする画像を開く
  const Image target = Dialog::OpenImage()
    .fit(Window::Size());
  // ③ 同じ大きさの白い画像を用意
  Image image(target.size, Palette::White);
  Image old = image;
  DynamicTexture texture(old);
  double d1 = Diff(target, image); // ④

  while (System::Update()) {
    for (int i = 0; i < 100; ++i) {
      old = image; // ⑤

      // ⑥ 画像内のランダムな位置
      Point pos = RandomPoint(
        image.width, image.height);
      // ⑦ ランダムな色
      ColorF color;
      color.r = Random();
      color.g = Random();
      color.b = Random();
      color.a = Random();
      // ⑧ ランダムな大きさ
      int size = Random(1, 10);
      // ⑨ 円を描いてみる
      Circle(pos, size).write(image, color);

      // ⑩ 目標に近づけば採用
      double d2 = Diff(target, image);
      if (d2 < d1) {
         d1 = d2;
      }
      else {
         image = old; // ⑪
      }
     }

      // ⑫ テクスチャを更新して描画
      texture.fill(image);
      texture.draw();
  }
}
```

「コンピュータ画伯」の完全なプログラム

さらに発展

丸の代わりに直線や曲線を描き込んで絵のタッチを変えてみたり，描き込みの位置を前回描き込んだ位置と近づけることで，人間らしく描かせたりするような工夫をするのも面白いだろう．

あなたの声はどんな形？

共感覚と呼ばれる特殊な知覚を持つ人の中には，音に色がついて見える人がいるといわれている．ちょっと素敵な能力のような気がしてうらやましいが，プログラミングができれば，私たちも似たような感覚を体験できる．

私たちが耳にする声とは，声帯や口の開き方によって特徴づけられる，空気中の圧力変動の波である．この波形信号をマイクで記録し，高速フーリエ変換（FFT）により，音声にどのような周波数成分が含まれているかを計算で求め，その結果を色や形としてグラフィカルに表現すれば，あなたにも声が見えるようになる．

目標

マイクで自分の声を録音し，そこに含まれる周波数成分をリアルタイムで可視化するアプリケーションを作ってみよう．周波数をドレミの音階に対応付けて円周上に配置し，周波数成分の大きさに応じて円を描くことで，画面上に声の形を作り出そう（図-4）．

実装

まず，マイクをセットアップして声の録音を開始する．録音用のバッファは毎秒44,100サンプルで5秒分，バッファがいっぱいになるとバッファの先頭から上書きする—①．カラフルな円を重ねて描くときに，光が飽和するような美しい表現を得るために，色のブレンドモードを加算ブレンド（Additive）にしておこう—②．メインループでは，マイクで録音された直近の音声に，どのような周波数成分が分布しているのかをFFTで解析する—③．結果の配列から，それぞれの音階—④におけるパワー—⑤を求める．ここでの音階sは，ピアノの一番低いラの音を0として，1オクターブ上がるごとに1大きくなる値である．円の位置は，その音階のパワーが大きいほど画面の中心から離れ，その方向は音階によって異なる．ラの音であれば12時の方向に，半音高いラのシャープであれば1時の方向に伸び，1オクターブで1周する—⑥．円の色も音階を色相環に対応させる．ラの音であれば赤色，ラのシャープであればオレンジ色と，こちらも1オクターブで1周させる—⑦．計算した位置と色に基づいてそれぞれの円を描くと声の形が現れる—⑧．

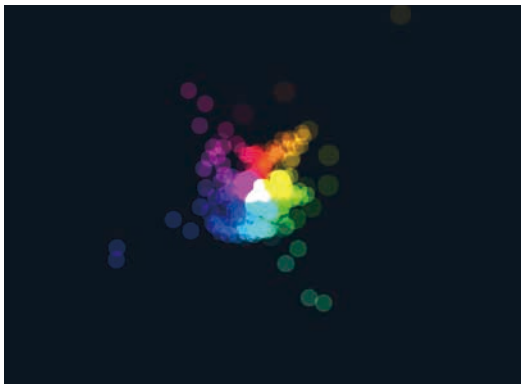

図-4　実行結果：筆者の「声の形」

```
# include <Siv3D.hpp>

void Main() {
  // ①マイク録音開始
  Recorder mic;
  mic.open(0, 5s, RecordingFormat::S44100, true);
  mic.start();

  // ②加算ブレンドを有効に
  Graphics2D::SetBlendState(BlendState::Additive);

  while (System::Update()) {
    // ③FFTで周波数成分を解析
    const auto fft = FFT::Analyze(mic);

    for (int i = 0; i < fft.length(); ++i) {
      // ④音階
      double s = Log2(
      (fft.resolution() * (i + 1)) / 27.5);
      // ⑤パワー
      double p = fft.buffer[i];

      //⑥円の中心位置
      const Vec2 pos =
        Window::Center() +
        Circular(Pow(p, 0.5) * 600, s * TwoPi);
      // ⑦円の色
      const Color c = HSV(s * 360)
        .toColorF(0.05 * s);
      //⑧円を描く
      Circle(pos, 15 - s).draw(c);
    }
  }
}
```

「声の形」の完全なプログラム

さらに発展

声以外にも，口笛やいろいろな楽器を鳴らして，どのような形が現れるか調べてみよう．音の成分の表現方法を，棒グラフやもっと複雑なエフェクトにしてみるなど，ビジュアルを工夫するのもよいだろう．

幸せになれる画像ビューア

旅行先の素敵な景色，おいしそうなランチ，可愛い我が子の成長記録．私たちが SNS に写真や動画をアップロードするのは，他人からの「いいね！」によって，自らの幸せを再確認したいからなのかもしれない．一般的に，ある個人のアカウントを注目している「フォロワー」の人数が多いほど，投稿にはたくさんの反響が寄せられる．数万人単位のフォロワーを抱える有名人の投稿には，ステージに浴びせられる歓声のように，賞賛や羨望のコメントが降り注ぐ．

さて，私たちのような平凡な人間でも，プログラミングができると，そんな人気者の気分を少しだけ味わえるというのはご存じだろうか．用意するのは，あなたが今日撮った写真と，降り注がせたいコメントを列挙したテキストファイルだけである．

目標

自分の撮った写真を開くと，たくさんのコメントが画面を埋め尽くす画像ビューアを作ってみよう．コメントはテキストファイルに改行で区切って記述する．20 ～ 30 パターンのコメントを用意すれば，にぎやかな画面になるだろう（図 -5）．

実装

まず，画面に流れるコメントの文章（String 型）と位置（Vec2 型）を保持する Comment 型を定義しておこう—①．Main 関数の最初で，ユーザにダイアログから写真を選択させる．画面内に収まるようにサイズを調整しておく—②．次に，TextReader クラスを使って，用意しておいたテキストファイルから 1 行ずつコメントを読み込む—③．コメントを表示する初期位置はランダムにする—④．コメント用のフォントもあらかじめ用意しておく．文字にアウトラインを付けるとよく映える—⑤．メインループ内では，読み込んだ写真を背景に描画し—⑥，その上にコメントを表示する—⑦．コメントが画面の右から左へ流れるように座標を制御すると—⑧，どこかで見たことのあるインタフェースになる．画面外に流れたコメントは，座標をリセットして再利用すれば，いつまでもコメントを流せる—⑨．

図 -5　実行結果：題して「ポジティブなコメント群による，優れたランチ・エクスペリエンスの創出」

```
# include <Siv3D.hpp>

// ①コメントの文章と位置
struct Comment {
  String text;
  Vec2 pos;
};

void Main() {
  // ②画像を選択する
  Texture photo(Dialog::OpenImage()
    .fit(Window::Size()));

  //③事前に用意したコメントを読み込む
  TextReader reader(L"comments.txt");
  Array<Comment> comments;
  Comment c;
  while (reader.readLine(c.text)) {
    c.pos.x = Random(1000, 2000); // ④
    c.pos.y = Random(0, 420);
    comments.push_back(c);
  }

  //⑤コメント用のフォント
  Font font(30, Typeface::Bold, FontStyle::Outline);
  font.changeOutlineStyle(TextOutlineStyle(
    Palette::Gray, Palette::White, 1));

  while (System::Update()) {
   photo.draw(); // ⑥

   for (auto& c : comments) {
     font(c.text).draw(c.pos); //⑦
     //⑧コメントを左に流す
     c.pos.x -= 4 + c.text.length * 0.2;
     //⑨画面外に出たらまた右へ
     if (c.pos.x < -500) {
```

```
        c.pos.x = Random(1000, 2000);
        c.pos.y = Random(0, 420);
    }
  }
 }
}
```

「幸せになれる画像ビューア」の完全なプログラム

さらに発展

プログラミングに腕のある読者であれば，画像中の物体を認識して，コメントの内容をカスタマイズしたり，自動生成したりするような追加機能も実装できるだろう．Siv3D には Web カメラで撮影した画像をリアルタイムで表示する機能がある．白雪姫の魔法の鏡さながらに，自分の姿に「イケメン」「お美しい」といったコメントを流してくれるアプリというのはどうだろうか．

めちゃくちゃな音楽プレイヤ

一青窈の「もらい泣き」の音の高さを下げて再生すると，平井堅が歌っているように聞こえるそうだ．私たちは撮った写真の見栄えを良くしたり，加工をしたりすることはよくあるが，音楽に関してはなかなか遊ぶ機会が少ないのではないだろうか．パソコンに保存されているお気に入りの音楽を再生する，ちょっと不思議な音楽プレイヤを作ってみよう．

目標

再生している音楽のテンポ（1 分間に何拍刻むかの速さ）とピッチ（音の高さ）をリアルタイムに変更できる音楽プレイヤを作ろう☆1．画面に基準の線を表示し，マウスカーソルを右に移動させると速いテンポで，左に移動させると遅いテンポで，上に移動させると高いピッチで，下に移動させると低いピッチで再生されるようにしよう（図 -6）．

実装

テンポとピッチを表示するためのフォントを用意

☆1 いわゆる「早送り再生」では，テンポとピッチが同時に速く，高くなるが，今回のプログラムでは，それぞれの要素を独立して操作する．たとえば，ゆっくり再生しても音の高さは変わらないといったことが可能である．

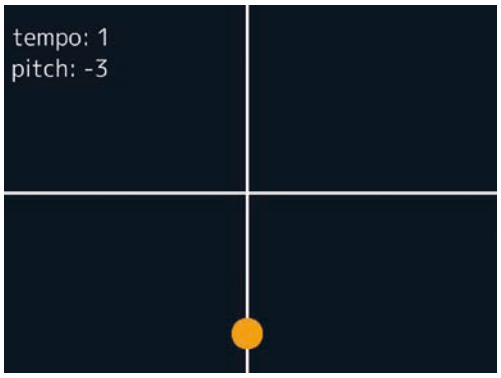

図 -6　実行結果：テンポやピッチを変更すると，聞きなれた楽曲も新鮮に聞こえる

する—①．続いてユーザにダイアログから音楽ファイルを選択させ—②，それを再生する—③．メインループ内では，ユーザがテンポやピッチを制御する基準となる線—④と円—⑤を描く．マウスカーソルが画面の中心よりも右にあればあるほどテンポを速くし—⑥，マウスカーソルが画面の中心よりも上にあればあるほどピッチが高くなるように値を計算する—⑦．計算した結果は，再生中の音楽に設定することで反映される—⑧．最後にこれらの値を画面に表示して完成である—⑨．

```
# include <Siv3D.hpp>

void Main() {
  //①表示用のフォントを用意
  Font font(20);
  //②音楽ファイルを開く
  Sound sound = Dialog::OpenSound();
  //③音楽を再生
  sound.play();

  while (System::Update()) {
   //④線を描く
   Line(0, 240, 640, 240).draw(4);
   Line(320, 480, 320, 0).draw(4);
   //⑤カーソルの位置に円を描く
   Point pos = Mouse::Pos();
   Circle(pos, 20).draw(Palette::Orange);

   // ⑥テンポを計算
   double tempo = Exp2((pos.x - 320) / 240.0);

   //⑦ピッチを計算
   double pitch = -(pos.y - 240) / 60.0;
   // ⑧音楽にテンポとピッチを適用
   sound.changeTempo(tempo);
   sound.changePitchSemitones(pitch);
   //⑨現在のテンポとピッチを表示
   font(L"tempo: ", tempo).draw(20, 20);
   font(L"pitch: ", pitch).draw(20, 60);
  }
}
```

「めちゃくちゃな音楽プレイヤ」の完全なプログラム

さらに発展

いつも聞いているお気に入りの音楽のプレイリストを用意して，テンポやピッチをランダムにアレンジして再生してくれる音楽プレイヤを作ってみよう．

数式の大地を探検する

私たちは地図アプリやゲームで 3D 空間を探検することはよくあるが，実際にこうした 3D のディジタルデータを作る機会は少ない．近頃は 3D プリンタが手の届く価格で利用できるようになったのだから，何か 3D のものを作ってみようではないか．複雑なモデリングソフトは使わず，数学の知識だけで 3D マップを生成できるアプリケーションを作ってみよう．

目標

テキストエリアに x，y の関数の数式を入力すると，そのグラフを 3D 空間上に描いてくれるプログラムを作ろう．キーボード入力で，3D グラフの中を探検できるようにもしてみよう．たった数十文字の数式から生み出されるバリエーション豊かな地形やダンジョンが，あなたの冒険を待っている（図-7，8）．

実装

グラフが見やすいように画面の背景を明るい色に設定する―①．次に，グリッド状の頂点配列で構成される 3D メッシュデータを用意する―②．ユーザが数式を入力するための，2 行× 30 文字分のテキストエリアを，グラフィカルユーザインタフェース（GUI）クラスを使って用意する―③．メインループ内では，ユーザが数式を入力中でなければ，アローキーや W/A/S/D などのキーボード入力で 3D 空間上の視点を操作できるようにする関数を呼ぶ―④．もし，テキストエリア内の数式が変更されたら―⑤，数式をパースし，有効な式であれば数式の文字を黒くする―⑥．そして，数式を用いてメッシュ状の頂点の配列の座標を再計算する．Siv3D の 3D 空間では y 軸が上方向なので，各頂点の y 座標を，x 座標と z 座標から求める―⑦．配列の座標を更新したら，陰影の計算のために必要な法線情報を再構築し―⑧，メッシュデータを新しい配列で更新する―⑨．もし数式のパース時にエラーがあれば，数式の文字を赤くして，ユーザに伝える―⑩．メッシュとその影を描画すれば―⑪，画面に 3D グラフの地形が現れ，その中を探検できる．

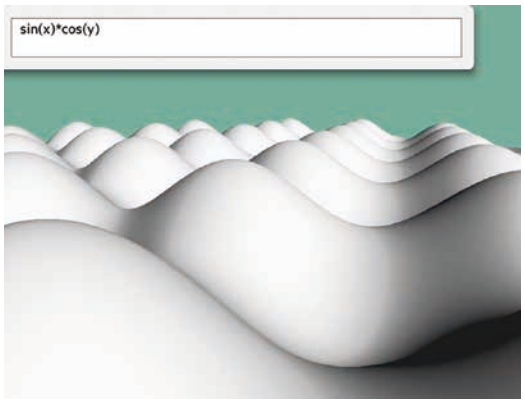

図-7　実行結果：永遠に抜け出せない sin(x)*cos(y) 砂漠

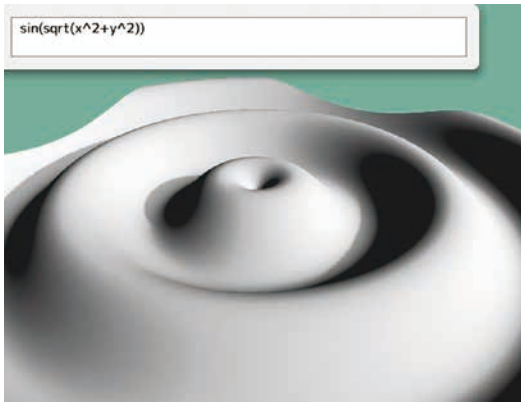

図-8　実行結果：sin(sqrt(x^2+y^2)) 火山とカルデラ地形

```
# include <Siv3D.hpp>

void Main() {
  //①背景を明るい色に
  Graphics::SetBackground(Color(120, 180, 160));

  //②メッシュを用意
  MeshData meshData = MeshData::Grid(25, 160);
  DynamicMesh mesh(meshData);

  //③数式を入力するテキストエリアを用意
  GUI gui(GUIStyle::Default);
  gui.addln(L"exp", GUITextArea::Create(2, 30));

  while (System::Update()) {
   if(!gui.textArea(L"exp").active) {
      Graphics3D::FreeCamera(); //④
   }

   //⑤数式が変更されたら
   if(gui.textArea(L"exp").hasChanged) {
     if(const ParsedExpression
        exp{ gui.textArea(L"exp").text }) {
       gui.textArea(L"exp").style.color
        = Palette::Black; //⑥
```

```
        //⑦数式に基づきメッシュの座標を計算
        for (auto& v : meshData.vertices) {
          v.position.y = exp.evaluateOpt({
            { L"x", v.position.x },
            { L"y", v.position.z } })
            .value_or(0);
        }

        //⑧法線を更新
        meshData.computeNormals();
        //⑨メッシュを更新
        mesh.fillVertices(meshData.vertices);
      }
       else {
         gui.textArea(L"exp").style.color
           = Palette::Red; //⑩
      }
    }

    //⑪メッシュを描画
    mesh.draw().drawShadow();
  }
}
```

「3D グラフ」の完全なプログラム

さらに発展

draw() の引数に色や画像を渡せば，地形に色や模様をつけられる．気に入った形のメッシュを 3D プリンタで扱えるファイル形式で出力し，実体化したものを棚に並べて鑑賞するのも楽しいだろう．

遊べるプログラミングの価値

情報処理技術の活用が進む将来に向け，プログラミング教育に関する議論が盛んに行われている．プログラミング言語の文法やアルゴリズムを丁寧に学ぶことも大切だが，それだけでは，自分で使いたくなるようなソフトウェアを開発できるようになるまでに時間がかかってしまう．本稿で解説した，音や画像を使う方法のように，ちょっとしたコードで遊べる開発を通して，「コンピュータって面白い！」と思える体験を積み重ね，一人ひとりが「作り手」になれるという意識を育んでいくようなことが効果的ではないだろうか．

Siv3D を使う高校生や大学生と話すと，はじめは C++ プログラミングの経験はほとんどなかったが，面白いアイディアやインタラクションを実現するために，Siv3D を使いながら C++ を学んだというユーザに多く出会う．プログラミングを楽しむ人口を増やし，豊かな IT 社会を実現するにあたって，さまざまなツールやフレームワークが提供する「プログラミング体験」が，ますます重要な役割を担っていくだろう．

(2017 年 2 月 27 日受付)

鈴木　遼（学生会員）■ reputeless@gmail.com

Siv3D 開発者．2016 年早稲田大学基幹理工学研究科表現工学専攻修士課程修了．同年同専攻博士後期課程入学．プログラミングツールの研究開発と，若年層へのプログラミング教育活動に力を入れている．