目次

1 本体 5
1.1 C++ . . . 5
Animation.cpp . . . 5
Bezier.cpp . . . 7
Button.cpp . . . 8
CommonText.cpp . . . 11
Draw.cpp . . . 12
DrawGraph.cpp . . . 15
DrawObject.cpp . . . 18
DrawText.cpp . . . 22
Font.cpp . . . 26
Grading.cpp . . . 27
Kinect.cpp . . . 30
KinectBody.cpp . . . 31
MaiKagami.cpp . . . 34
Main.cpp . . . 36
ModeSelect.cpp . . . 37
Nfc.cpp . . . 38
PartMain.cpp . . . 41
PartOption.cpp . . . 43
PartOptionPop.cpp . . . 44
PartPause.cpp . . . 48
PartPlay.cpp . . . 51
PartResult.cpp . . . 52
PartResultMain.cpp . . . 54
PauseScreen.cpp . . . 56
PlayScreen.cpp . . . 58
PlayScreenObject.cpp . . . 60
Result.cpp . . . 63
Scene.cpp . . . 64
SettingPop.cpp . . . 65
Song.cpp . . . 68
Songs.cpp . . . 71
SongSelect.cpp . . . 74
SongSelectCommon.cpp . . . 76
SongSelectCover.cpp . . . 80

SongSelectMain.cpp . . . 83
StartScreen.cpp . . . 85
ThroughDetail.cpp . . . 86
ThroughMain.cpp . . . 89
ThroughOption.cpp . . . 91
ThroughPause.cpp . . . 92
ThroughPlay.cpp . . . 95
ThroughResult.cpp . . . 96
ThroughResultMain.cpp . . . 98
ThroughResultObject.cpp . . . 100
ThroughStart.cpp . . . 104
Top.cpp . . . 105
TopMain.cpp . . . 106
Touch.cpp . . . 107
User.cpp . . . 108
1.2 ヘッダファイル . . . 109
Animation.h . . . 109
Bezier.h . . . 110
Button.h . . . 111
CommonText.h . . . 113
Draw.h . . . 114
DrawGraph.h . . . 116
DrawObject.h . . . 117
DrawText.h . . . 119
Font.h . . . 121
Grading.h . . . 122
Kinect.h . . . 123
KinectBody.h . . . 124
MaiKagami.h . . . 125
Main.h . . . 126
ModeSelect.h . . . 127
Nfc.h . . . 128
PartDefine.h . . . 129
PartMain.h . . . 130
PartOption.h . . . 131
PartOptionPop.h . . . 132
PartPause.h . . . 134
PartPlay.h . . . 136
PartResult.h . . . 137
PartResultDefine.h . . . 138

PartResultMain.h . . . 139
PauseScreen.h . . . 140
PlayScreen.h . . . 141
PlayScreenObject.h . . . 142
Result.h . . . 143
Scene.h . . . 144
SeetingPop.h . . . 145
Song.h . . . 147
Songs.h . . . 149
SongSelect.h . . . 150
SongSelectCommon.h . . . 151
SongSelectCover.h . . . 152
SongSelectDefine.h . . . 153
SongSelectMain.h . . . 154
StartScreen.h . . . 155
stdafx.h . . . 156
ThroughDefine.h . . . 157
ThroughDetail.h . . . 158
ThroughMain.h . . . 160
ThroughOption.h . . . 161
ThroughPause.h . . . 162
ThroughPlay.h . . . 163
ThroughResult.h . . . 164
ThroughResultDefine.h . . . 165
ThroughResultMain.h . . . 166
ThroughResultObject.h . . . 167
ThroughStart.h . . . 169
Top.h . . . 170
TopMain.h . . . 171
Touch.h . . . 172
User.h . . . 173

2 Web サーバ 174
2.1 HTML . . . 174
list.html . . . 174
2.2 PHP . . . 176
api_video.php . . . 176
api_history.php . . . 177
api_add.php . . . 178
list.php . . . 179

login.php . . . 181
logout.php . . . 184
main.php . . . 186
make_history_table.php . . . 188
music.php . . . 189
video.php . . . 191
2.3 CSS . . . 193
css/style.css . . . 193
2.4 Javascript . . . 203
js/drawerNav.js . . . 203
js/scroll.js . . . 204

3 NFC リーダ用 Android アプリケーション 205
MainActivity.java . . . 205
MyNfc.java . . . 206
UsbMessage.java . . . 208
AndroidManifest.xml . . . 209

1 本体

1.1 C++

Animation.cpp

```cpp
#include "Animation.h"

// アニメーションの進行割合を更新して戻り値へ
// 終了時刻, タイプ, 遅延
double Animation::UpdateRate(Easing ease) {
    double r, rate;
    if (duration)
        r = (double)t / duration;
    else
        r = 1;
    switch (ease) {
    case EaseInOut_SINE:
        rate = (1 - cos(r * M_PI)) / 2;
        break;
    case EaseOut_SINE:
        rate = sin(r * M_PI / 2);
        break;
    case EaseIn_SINE:
        rate = 1 - cos(r * M_PI / 2);
        break;
    case EaseInOut_QUAD:
        rate = r < 0.5 ? r * r * 2 : - (r - 1) * (r - 1) * 2 + 1;
        break;
    case LinerInEaseOut_QUAD:
        rate = r < 0.5 ? r * 4 / 3 : - (r - 1) * (r - 1) * 4 / 3 + 1;
        break;
    case EaseInLinerOut_QUAD:
        rate = r < 0.5 ? r * r * 4 / 3 : (r * 4 - 1) / 3;
        break;
    case EaseOutBack_QUAD:
        // rate = - (r - 2.0 / 3) * (r - 2.0 / 3) * 3 + 4.0 / 3;
        rate = - (r - 3.0 / 4) * (r - 3.0 / 4) * 2 + 9.0 / 8;
        break;
    case LINER: default:
        rate = r;
        break;
    }
    if (t < duration)
        t++;
    return rate;
}

// パラメータ代入
// void Animation::SetRate(MyTime _duration, int _ease = LINER) {
void Animation::SetDuration(MyTime _duration) {
    duration = _duration <= 0 ? 0 : _duration;
//  ease = _ease;
}

// アニメーション時刻を強制変更 (引数に0を入れれば時刻初期化)
void Animation::SetTime(MyTime _t) {
    t = _t;
}

```

```
55  MyTime Animation::GetTime() {
56      return t;
57  }
58
59  void Animation::Reset() {
60      SetTime(0);
61  }
```

Bezier.cpp

```cpp
#include "Bezier.h"

Bezier::Bezier(const double x1, const double y1, const double x2, const double y2) {
    this->x1 = x1;
    this->x2 = x2;
    this->y1 = y1;
    this->y2 = y2;
}

// 計算
double Bezier::Calc(const double x) {
    double now_x, now_abs, pre_abs = 1, y;
    double t;
    int c = 0;
    const int N = 1000; // 大きいほど精度良・計算量多

    for (c = 0; c < N + 1; c++) {
        t = (double)c / N;
        now_x = t*t*t + 3 * t*t*(1 - t)*x2 + 3 * t*(1 - t)*(1 - t)*x1;
        now_abs = x - now_x > 0 ? x - now_x : now_x - x;
        if (pre_abs < now_abs) {
            t = (double)(c - 1) / N;
            break;
        }
        pre_abs = now_abs;
    }
    y = t*t*t + 3 * t*t*(1 - t)*y2 + 3 * t*(1 - t)*(1 - t)*y1;

    return y;
}
```

Button.cpp

```
1  #include "Button.h"
2
3  // ボタン
4  Button::Button(const int num, Touch *touch)
5      : Draw(0, BUTTON_POS + num * BUTTON_INTERVAL) {
6      this->num = num;
7      this->touch = touch;
8  }
9
10 int Button::GetTouch() {
11     return touch->Get(num);
12 }
13
14 // 三角形のボタン
15 TriangleButton::TriangleButton(Font *font, Touch *touch, const char *str,
       const int direction, const int num, char *colorName)
16     : Button(num, touch) {
17     text = new MyDrawText(font, str, WIDTH * 0.94, GetY(), 2, 30);
18     myDrawTriangle2 = new MyDrawTriangle2(WIDTH * 0.97, GetY(), WIDTH *
           0.03, direction, colorName);
19 }
20
21 void TriangleButton::ContentView() {
22     myDrawTriangle2->View();
23     text->View();
24 }
25
26 TriangleButton::~TriangleButton() {
27     delete myDrawTriangle2;
28     delete text;
29 }
30
31 // 説明文付き三角形のボタン
32 TriangleButton2::TriangleButton2(Font *font, Touch *touch, const char *
       title, const char *str, const int direction, const int num, const
       float x, const char *colorName)
33     : Button(num, touch) {
34     float pos = GetY();
35     text = new MyDrawText(font, title, x, pos - HEIGHT * 0.03, 0, 30,
           colorName);
36     descriptionText = new MyDrawTexts(font, str, x, pos + HEIGHT * 0.01,
           0, 20, 15);
37
38     float width = WIDTH * 0.35;
39     myDrawBox = new MyDrawBox(x + width / 2, pos, width + WIDTH * 0.05,
           HEIGHT * 0.09, 2, colorName);
40     myDrawTriangle2 = new MyDrawTriangle2(WIDTH * 0.97, pos, WIDTH * 0.03,
            direction, colorName);
41 }
42
43 void TriangleButton2::ContentView() {
44     descriptionText->View();
45     myDrawBox->View();
46     myDrawTriangle2->View();
47     text->View();
48 }
49
50 TriangleButton2::~TriangleButton2() {
51     delete myDrawTriangle2;
52     delete myDrawBox;
53     delete descriptionText;
```

```
54      delete text;
55  }
56
57
58  // 文字右寄せボタン
59  CircleButton::CircleButton(Font *font, Touch *touch, const char *str,
        const int num, char *colorName)
60      : Button(num, touch) {
61      text = new MyDrawText(font, str, WIDTH * 0.94, GetY(), 2, 30);
62      myDrawCircle = new MyDrawCircle(WIDTH * 0.97, GetY(), WIDTH * 0.015,
            7, colorName);
63  }
64
65  // 文字中央寄せボタン
66  CircleButton::CircleButton(Font *font, Touch *touch, const char *str,
        const int num, const float x, char *colorName)
67      : Button(num, touch) {
68      text = new MyDrawText(font, str, x, GetY(), 1, 30);
69      myDrawCircle = new MyDrawCircle(WIDTH * 0.97, GetY(), WIDTH * 0.015,
            7, colorName);
70  }
71
72  void CircleButton::ContentView() {
73      myDrawCircle->View();
74      text->View();
75  }
76
77  CircleButton::~CircleButton() {
78      delete text;
79      delete myDrawCircle;
80  }
81
82  // 文字が丸の中にあるボタン
83  CircleButton2::CircleButton2(Font *font, Touch *touch, const char *str,
        const int num, char *colorName)
84      : Button(num, touch) {
85      float r = WIDTH * 0.045;
86      float x = WIDTH - r - 4;
87      text = new MyDrawText(font, str, x, GetY(), 1, 30, "Black");
88      myDrawCircle = new MyDrawCircle(x, GetY(), r, colorName);
89  }
90
91  void CircleButton2::ContentView() {
92      myDrawCircle->View();
93      text->View();
94  }
95
96  CircleButton2::~CircleButton2() {
97      delete text;
98      delete myDrawCircle;
99  }
100
101 // 画像付きのボタン
102 CircleGraphButton::CircleGraphButton(Touch *touch, const int num, const
        char *fileName)
103     : Button(num, touch) {
104     float r = WIDTH * 0.075;
105     myDrawCircle = new MyDrawCircle(WIDTH, GetY(), r);
106     myDrawGraph = new MyDrawGraph(WIDTH  - 35, GetY(), fileName);
107 }
108
109 void CircleGraphButton::Load() {
110     myDrawGraph->Load();
```

```
111 }
112
113 void CircleGraphButton::ContentView() {
114     myDrawCircle->View();
115     myDrawGraph->View();
116 }
117
118 void CircleGraphButton::Release() {
119     myDrawGraph->Release();
120 }
121
122 CircleGraphButton::~CircleGraphButton() {
123     delete myDrawCircle;
124     delete myDrawGraph;
125 }
126
127 // 画像、テキスト付きのボタン
128 CircleGraphTextButton::CircleGraphTextButton(Font *font, Touch *touch,
        const char *str, const int num, const char *fileName)
129     : Button(num, touch) {
130     float x = WIDTH * 0.965;
131     float y = GetY();
132     float r = WIDTH * 0.026;
133     text = new MyDrawText(font, str, x - r - 12, y, 2, 30);
134     myDrawCircle = new MyDrawCircle(x, y, r, "Blue");
135     myDrawGraph = new MyDrawGraph(x, y, fileName, 0.6);
136 }
137
138 void CircleGraphTextButton::Load() {
139     myDrawGraph->Load();
140 }
141
142 void CircleGraphTextButton::ContentView() {
143     text->View();
144     myDrawCircle->View();
145     myDrawGraph->View();
146 }
147
148 void CircleGraphTextButton::Release() {
149     myDrawGraph->Release();
150 }
151
152 CircleGraphTextButton::~CircleGraphTextButton() {
153     delete text;
154     delete myDrawCircle;
155     delete myDrawGraph;
156 }
```

CommonText.cpp

```
1  #include "CommonText.h"
2
3  // タイトル表示用
4  DrawTitle::DrawTitle(Font *font, const char *str)
5      : MyDrawTextLine(font, str, WIDTH * 0.65, HEIGHT * 0.21, 1, 50, WIDTH
           * 0.4, 3) {}
6
7  // サブタイトル表示用
8  DrawSubtitle::DrawSubtitle(Font *font, const char *str)
9      : MyDrawText(font, str, WIDTH * 0.65, HEIGHT * 0.25, 1, 30) {}
```

Draw.cpp

```cpp
#include "Draw.h"

// 色指定
Color::Color(const char *color) {
    ChangeColor(color);
}

// 色取得
int Color::Get() {
    return c;
}

void Color::ChangeColor(const char *color) {
    if (!strcmp(color, "White"))
        c = GetColor(255, 255, 255); // 白色
    else if (!strcmp(color, "Blue"))
        c = GetColor(127, 210, 234); // 青色
    else if (!strcmp(color, "Black"))
        c = GetColor(0, 0, 0); // 黒色
    else if (!strcmp(color, "Yellow"))
        c = GetColor(255, 255, 0); // 黄色
}

// 表示位置用クラスコンストラクタ
Pos::Pos() {
    x = 0; y = 0;
}

// 表示位置用クラスコンストラクタ
Pos::Pos(const float x, const float y) {
    this->x = x / SIZE_RATE; this->y = y / SIZE_RATE;
}

// 表示位置変更
void Pos::ChangePos(const float x, const float y) {
    this->x = x / SIZE_RATE; this->y = y / SIZE_RATE;
}

// アニメーション用パラメータセット Jaity
void Pos::SetPosAnimation(float target_x, float target_y, Easing ease) {
    if (GetTime() != 0)
        return;
    default_x = GetX();
    default_y = GetY();
    this->target_x = target_x;
    this->target_y = target_y;
    ease_pos = ease;
//  SetRate(duration, ease);
    // SetDuration(duration);
}


// アニメーション更新 Jaity
void Pos::Update() {
    double r = UpdateRate(ease_pos);
    float nx = default_x + (target_x - default_x) * r;
    float ny = default_y + (target_y - default_y) * r;
    ChangePos(nx, ny);
}

```

```
61  // x座標取得
62  float Pos::GetX() {
63      return x * SIZE_RATE;
64  }
65
66  // y座標取得
67  float Pos::GetY() {
68      return y * SIZE_RATE;
69  }
70
71  // 描画用クラスコンストラクタ
72  Draw::Draw(){}
73
74  // 描画用クラスコンストラクタ
75  Draw::Draw(const float x, const float y) : Pos(x, y) {}
76
77  // 描画
78  void Draw::View() {
79      if (viewFlag) {
80          SetDrawBlendMode(DX_BLENDMODE_ALPHA, alpha); // 透明度設定
81          ContentView(); // 内容表示
82          SetDrawBlendMode(DX_BLENDMODE_NOBLEND, 0); // 透明度解除
83      }
84  }
85
86  void Draw::SetViewFlag(const boolean flag) {
87      viewFlag = flag;
88  }
89
90  // 透明度指定
91  void Draw::SetAlpha(const int alpha) {
92      this->alpha = alpha;
93  }
94
95  int Draw::GetAlpha() {
96      return alpha;
97  }
98
99  void Draw::SetAlphaAnimation(int target_alpha, Easing ease) {
100     if (GetTime() != 0)
101         return;
102     default_alpha = GetAlpha();
103     this->target_alpha= target_alpha;
104     ease_alpha = ease;
105 }
106
107 void Draw::Update() {
108     double r = UpdateRate(ease_alpha);
109     int na = default_alpha + (target_alpha - default_alpha) * r;
110     SetAlpha(na);
111     Pos::Update();
112 }
113
114 Draw2::Draw2(const int pos) {
115     p = pos;
116 }
117
118 Draw2::Draw2(const float x, const float y, const int pos) {
119     p = pos;
120     ChangePos(x, y);
121 }
122
123 void Draw2::ChangePos() {
```

```
124         float a = 0;
125         switch (p) {
126         case 1:
127             a -= GetWidth() / 2;
128             break;
129         case 2:
130             a -= GetWidth();
131             break;
132         }
133 
134         Draw::ChangePos(xx + a, yy - GetHeight() / 2);
135     }
136 
137     void Draw2::ChangePos(const float x, const float y) {
138         xx = x;
139         yy = y;
140         Draw2::ChangePos();
141     }
142 
143     float Draw2::GetX() {
144         return xx;
145     }
146 
147     float Draw2::GetY() {
148         return yy;
149     }
```

DrawGraph.cpp

```cpp
#include "DrawGraph.h"


// 画像初期化(座標指定なし、あとから指定する場合)
// MyDrawGraph(ファイル名)
MyDrawGraph::MyDrawGraph(const char *fileName) {
    ex = 1.0;
    this->fileName = fileName;
}

// 画像初期化
// MyDrawGraph(x座標、y座標、ファイル名、拡大率) // 拡大率は省略可能、省略した場合等倍
MyDrawGraph::MyDrawGraph(const float x, const float y, const char *fileName, const double ExRate) : Draw(x, y) {
    ex = ExRate;
    this->fileName = fileName;
}

// ファイル名変更
void MyDrawGraph::ChangeFile(const char *fileName) {
    this->fileName = fileName;
}

// 画像ロード
void MyDrawGraph::Load() {
    handle = LoadGraph(fileName.c_str()); // 画像のロード
}

// 画像表示
void MyDrawGraph::ContentView() {
    SetDrawMode(DX_DRAWMODE_BILINEAR);
    DrawRotaGraphF(x, y, ex / SIZE_RATE, 0, handle, TRUE, FALSE); // 描画
    SetDrawMode(DX_DRAWMODE_NEAREST);
}

// 画像大きさ変更
void MyDrawGraph::ChangeEx(const double ExRate) {
    ex = ExRate;
}
// 画像大きさ取得
double MyDrawGraph::GetEx() {
    return ex;
}

void MyDrawGraph::SetExAnimation(double target_ex, Easing ease) {
    if (GetTime() != 0)
        return;
    default_ex = GetEx();
    this->target_ex = target_ex;
    ease_ex = ease;
}

void MyDrawGraph::Update() {
    double r = UpdateRate(ease_ex);
    float nex = default_ex + (target_ex - default_ex) * r;
    ChangeEx(nex);
    Draw::Update();
}

```

```
59  // 画像を解放
60  void MyDrawGraph::Release() {
61      DeleteGraph(handle);
62  }
63
64  // 動画初期化
65  MyDrawMovie::MyDrawMovie(const char *filename) : MyDrawGraph(filename) {
66      speed = sp = 1.0;
67  }
68
69  // 動画初期化
70  MyDrawMovie::MyDrawMovie(const float x, const float y, const char *
        filename, const double ExRate)
71      : MyDrawGraph(x, y, filename, ExRate) {
72      speed = sp = 1.0;
73  }
74
75  // 動画表示
76  void MyDrawMovie::ContentView() {
77      if (!CheckHandleASyncLoad(handle)) {
78          SetDrawMode(DX_DRAWMODE_BILINEAR);
79          DrawRotaGraphF(x, y, ex / SIZE_RATE, 0, handle, TRUE, TRUE); // 描
                画
80          SetDrawMode(DX_DRAWMODE_NEAREST);
81      }
82  }
83
84  // 指定したフレームに移動
85  void MyDrawMovie::Seek(const int flame) {
86      Stop();
87      if(flame == -1)
88          SeekMovieToGraphToFrame(handle, startFlame);
89      else
90          SeekMovieToGraphToFrame(handle, flame);
91  }
92
93  // 再生
94  void MyDrawMovie::Start() {
95      if (!CheckHandleASyncLoad(handle)) {
96          SetSpeed();
97          if (GetNowFlame() == GetEndFlame())
98              Seek();
99          if(GetMovieStateToGraph(handle) == 0)
100             PlayMovieToGraph(handle);
101     }
102 }
103
104 // 再生停止
105 void MyDrawMovie::Stop() {
106     PauseMovieToGraph(handle);
107 }
108
109 // スピード変更
110 void MyDrawMovie::ChangeSpeed(double speed) {
111     this->sp = speed;
112 }
113
114 // スピードセット
115 void MyDrawMovie::SetSpeed() {
116     if (speed != sp) {
117         Stop();
118         Seek();
119         speed = sp;
```

```
120             SetPlaySpeedRateMovieToGraph(handle, speed);
121         }
122     }
123
124     void MyDrawMovie::SetPart() {
125         if (sf != startFlame || ef != endFlame) {
126             startFlame = sf;
127             endFlame = ef;
128             Stop();
129             Seek();
130         }
131     }
132
133     // スピード取得
134     double MyDrawMovie::GetSpeed() {
135         return sp;
136     }
137
138     // 最初のフレーム数取得
139     int MyDrawMovie::GetStartFlame() {
140         return startFlame;
141     }
142
143     // 最後のフレーム数取得
144     int MyDrawMovie::GetEndFlame() {
145         if (endFlame == -1)
146             return GetAllFlame();
147         return endFlame;
148     }
149
150     // 現在のフレーム数取得
151     int MyDrawMovie::GetNowFlame() {
152         return TellMovieToGraphToFrame(handle);
153     }
154
155     // 動画のフレーム数取得
156     int MyDrawMovie::GetAllFlame() {
157         return GetMovieTotalFrameToGraph(handle) - 1;
158     }
159
160     // スタートフレーム指定
161     void MyDrawMovie::SetStartFlame(const int flame) {
162         sf = flame;
163     }
164
165     // エンドフレーム指定
166     void MyDrawMovie::SetEndFlame(const int flame) {
167         ef = flame;
168     }
169
170     MyDrawMovie::~MyDrawMovie() {
171         DeleteGraph(handle);
172     }
```

DrawObject.cpp

```
1  #include "DrawObject.h"
2
3  // 円初期化（塗りつぶしあり）
4  // MyDrawCircle(x座標、y座標、半径、色）　※色は省略可能、省略した場合青色
5  MyDrawCircle::MyDrawCircle(const float x, const float y, const float
       radius, const char *colorName)
6      :Draw(x, y), Color(colorName) {
7      r = radius / SIZE_RATE;
8      w = 0;
9  }
10
11 // 円初期化（塗りつぶしなし）
12 // MyDrawCircle(x座標、y座標、半径、線の太さ、色）　※色は省略可能、省略した場合青色
13 MyDrawCircle::MyDrawCircle(const float x, const float y, const float
       radius, const float width, const char *colorName)
14     :Draw(x, y), Color(colorName) {
15     r = radius / SIZE_RATE;
16     w = width / SIZE_RATE;
17 }
18
19 // 円表示
20 void  MyDrawCircle::ContentView() {
21     boolean flag = TRUE;
22     if (w != 0)
23         flag = FALSE;
24     DrawCircleAA(x, y, r, 100, Color::Get(), flag, w);
25 }
26
27 // 角度付きの円初期化(塗りつぶしなし)
28 // MyDrawCircleGauge(x座標、y座標、半径、角度（%指定）、線の太さ、色）　※色は省略可能、省略した場合青色
29 MyDrawCircleGauge::MyDrawCircleGauge(const float x, const float y, const
       float radius, const double degree, const float width, const char *
       colorName)
30     :MyDrawCircle(0, 0, width, colorName), Pos(x, y){
31     r = radius / SIZE_RATE;
32     ChangeDegree(degree); // 角度を%からラジアンに変更
33 }
34
35 // 角度付きの円描画
36 void MyDrawCircleGauge::ContentView() {
37     for (double i = 0; i < rad; i += 0.02) {
38         float xx = (Pos::x + r * sin(i)) * SIZE_RATE;
39         float yy = (Pos::y - r * cos(i)) * SIZE_RATE;
40         MyDrawCircle::ChangePos(xx, yy);
41         MyDrawCircle::ContentView();
42     }
43     MyDrawCircle::ChangePos(GetEndX() * SIZE_RATE, GetEndY() * SIZE_RATE);
44     MyDrawCircle::ContentView();
45 }
46
47 // 角度付きの円　角度を%からラジアンに変更して保存
48 void MyDrawCircleGauge::ChangeDegree(const double degree) {
49     rad = 2 * M_PI * degree / 100;
50 }
51
52 // 角度付きの円　最終X座標を取得
53 float  MyDrawCircleGauge::GetEndX() {
54     return Pos::x + r * sin(rad);
```

```
55  }
56
57  // 角度付きの円　最終Y座標を取得
58  float  MyDrawCircleGauge::GetEndY() {
59      return Pos::y - r * cos(rad);
60  }
61
62  // 線初期化（座標指定なし、あとで指定する場合）
63  // MyDrawLine(長さ、色)　※色は省略可能、省略した場合青色
64  MyDrawLine::MyDrawLine(const float width, const char *colorName)
65      : Color(colorName) {}
66
67  // 線初期化（座標指定あり）
68  // MyDrawLine(始点x座標、始点y座標、終点x座標、終点y座標、長さ、色)　※色は省略可能、省略した場合青色
69  MyDrawLine::MyDrawLine(const float x1, const float y1, const float x2, const float y2, const float width, const char *colorName)
70      : Color(colorName) {
71      ChangePos(x1, y1, x2, y2);
72      w = width / SIZE_RATE;
73  }
74
75  // 線表示
76  void MyDrawLine::ContentView() {
77      DrawLineAA(x1, y1, x2, y2, Color::Get(), w);
78  }
79
80  // 線の座標指定
81  void MyDrawLine::ChangePos(const float x1, const float y1, const float x2, const float y2) {
82      this->x1 = x1 / SIZE_RATE;
83      this->y1 = y1 / SIZE_RATE;
84      this->x2 = x2 / SIZE_RATE;
85      this->y2 = y2 / SIZE_RATE;
86  }
87
88  // 三角形初期化
89  MyDrawTriangle::MyDrawTriangle(const char *colorName)
90      : Color(colorName) {}
91
92  // 三角形初期化
93  // MyDrawTriangle(点1x座標、点1y座標、点2x座標、点2y座標、点3x座標、点3y座標、長さ、色)　※色は省略可能、省略した場合青色
94  MyDrawTriangle::MyDrawTriangle(const float x1, const float y1, const float x2, const float y2, const float x3, const float y3, const char *colorName)
95      : Color(colorName) {
96      ChangePos(x1, y1, x2, y2, x3, y3);
97  }
98
99  // 三角形表示
100 void MyDrawTriangle::ContentView() {
101     DrawTriangleAA(x1, y1, x2, y2, x3, y3, Color::Get(), TRUE);
102 }
103
104 // 三角形　座標指定
105 void MyDrawTriangle::ChangePos(const float x1, const float y1, const float x2, const float y2, const float x3, const float y3) {
106     this->x1 = x1 / SIZE_RATE;
107     this->y1 = y1 / SIZE_RATE;
108     this->x2 = x2 / SIZE_RATE;
109     this->y2 = y2 / SIZE_RATE;
```

```
110         this->x3 = x3 / SIZE_RATE;
111         this->y3 = y3 / SIZE_RATE;
112     }
113
114     // 正三角形初期化
115     // MyDrawTriangle2(x座標、y座標、一辺の長さ、方向、色)  ※色は省略可能、省略した場合青色
116     // 方向(0:上向き、1:下向き、2:左向き)
117     MyDrawTriangle2::MyDrawTriangle2(const float x, const float y, const float width, const int direction, const char *colorName)
118         : MyDrawTriangle(colorName) {
119         w = width;
120         d = direction;
121
122         float x1, x2, x3, y1, y2, y3;
123         float a = w * sqrt(3) / 4;
124
125         x1 = x2 = x3 = x;
126         y1 = y2 = y3 = y;
127
128         switch (d)
129         {
130         case 0:
131             x2 -= w / 2; x3 += w / 2;
132             y1 -= a; y2 += a; y3 += a;
133             break;
134         case 1:
135             x2 -= w / 2; x3 += w / 2;
136             y1 += a; y2 -= a; y3 -= a;
137             break;
138         case 2:
139             y2 -= w / 2; y3 += w / 2;
140             x1 += a; x2 -= a; x3 -= a;
141             break;
142         }
143
144         ChangePos(x1, y1, x2, y2, x3, y3);
145     }
146
147     // 四角形初期化(塗りつぶしあり)
148     // MyDrawBox(x座標、y座標、横の長さ、縦の長さ、色)  ※色は省略可能、省略した場合白色
149     MyDrawBox::MyDrawBox(const float x, const float y, const float width, const float height, const char *colorName)
150         :Draw(x, y), Color(colorName) {
151         w = width / SIZE_RATE;
152         h = height / SIZE_RATE;
153         l = 0;
154     }
155
156     // 四角形初期化(塗りつぶしなし)
157     // MyDrawBox(x座標、y座標、横の長さ、縦の長さ、線の太さ、色)  ※色は省略可能、省略した場合青色
158     MyDrawBox::MyDrawBox(const float x, const float y, const float width, const float height, const float line, const char *colorName)
159         :Draw(x, y), Color(colorName) {
160         w = width / SIZE_RATE;
161         h = height / SIZE_RATE;
162         l = line / SIZE_RATE;
163     }
164
165     // 四角形表示
166     void  MyDrawBox::ContentView() {
```

```
167     boolean flag = TRUE;
168     if (l != 0)
169         flag = FALSE;
170     float x1 = x - w / 2;
171     float y1 = y - h / 2;
172     float x2 = x + w / 2;
173     float y2 = y + h / 2;
174 
175     DrawBoxAA(x1, y1, x2, y2, Color::Get(), flag, l);
176 }
177 
178 // 四角形サイズ変更
179 void MyDrawBox::ChangeSize(const float width, const float height) {
180     w = width / SIZE_RATE;
181     h = height / SIZE_RATE;
182 }
183 
184 // 進捗バー初期化
185 MyDrawBar::MyDrawBar(const float x, const float y, const float width,
        const float height, const char *colorName)
186     :MyDrawBox(x + width / 2, y, width, height, colorName) {
187     MyDrawBar::x = x;
188     MyDrawBar::y = y;
189 }
190 
191 // 進捗バーサイズ変更
192 void MyDrawBar::ChangeSize(const float width, const float height) {
193     MyDrawBox::ChangeSize(width, height);
194     ChangePos(x + width / 2, y);
195 }
```

DrawText.cpp

```cpp
#include "DrawText.h"

// テキスト初期化
//     MyDrawText（フォントポインタ、表示文字、x座標、y座標、ポジション情報、フォントサイ
// ポジション情報（0：左寄せ、1：中央寄せ、2：右寄せ）
MyDrawText::MyDrawText(Font *font, const char *str, const float x, const float y, const int pos, const int point, const char *colorName)
    : Color(colorName) , Draw2(pos) {
    s = str; // 文字列
    ChangeFont(font, point);
    MyDrawText::point = point;
    ChangePos(x, y);
}

// テキスト表示
void MyDrawText::ContentView() {
    DrawStringFToHandle(x, y, s.c_str(), Color::Get(), f); // 文字表示
}

// テキスト変更
void MyDrawText::ChangeText(char *str) {
    s = str;
    ChangePos();
}

// フォントサイズ変更
void MyDrawText::ChangeFont(Font *font, const int point) {
    f = font->Get(point); // フォント情報
}

// テキストの縦取得
float MyDrawText::GetHeight() {
    int line = 1; // 行数
    for (int i = 0; i < strlen(s.c_str()); i++) {
        if (s.c_str()[i] == '\n')
            line++;
    }

    return (float)point * (1 + 1 / 3) * line;
}

// テキストの幅取得
float MyDrawText::GetWidth() {
    return  (float)GetDrawStringWidthToHandle(s.c_str(), (int)strlen(s.c_str()), f) * SIZE_RATE;
}

// 縦書きテキスト初期化
//     MyDrawText（フォントポインタ、表示文字、x座標、y座標、ポジション情報、フォントサイ
// ポジション情報（0：下寄せ、1：中央寄せ、2：上寄せ）
MyDrawTextV::MyDrawTextV(Font *font, const char *str, const float x, const float y, const int pos, const int point, const char *colorName)
    : MyDrawText(font, str, x, y, 0, point, colorName) {
    switch (pos)
    {
    case 0:
```

```
54          RotCenterX = 0;
55          break;
56      case 1:
57          RotCenterX = GetWidth() / SIZE_RATE / 2;
58          break;
59      case 2:
60          RotCenterX = GetWidth() / SIZE_RATE;
61          break;
62      }
63  }
64
65  // 縦書きテキスト表示
66  void MyDrawTextV::ContentView() {
67      SetDrawMode(DX_DRAWMODE_BILINEAR);
68      DrawRotaStringToHandle(x, y, 1, 1, RotCenterX, GetHeight() / SIZE_RATE
            / 2, - 1.0 / 2.0 * 3.141592, Color::Get(), f, -1, FALSE, s.c_str
            ());
69      SetDrawMode(DX_DRAWMODE_NEAREST);
70  }
71
72  // 複数行のテキスト
73  //
        MyDrawTexts（フォントポインタ、表示文字、x座標、y座標、ポジション情報、フォントサ
74  // ポジション情報（0：左寄せ、1：中央寄せ、2：右寄せ）
75  MyDrawTexts::MyDrawTexts(Font *font, const char *str, const float x, const
         float y, const int pos, const int point, const float lineInterval,
        const char *colorName)
76      : Color(colorName) , Draw(x, y) {
77
78      p = pos; // 位置情報
79      inter = lineInterval; // 間隔
80      strcpy_s(color, sizeof(color), colorName);
81      this->point = point;
82      f = font;
83
84      ChangeText(str);
85  }
86
87  // 複数行のテキスト表示
88  void MyDrawTexts::ContentView() {
89      for (int i = 0; i < l; i++)
90          myDrawText[i]->ContentView();
91  }
92
93  // 複数行のテキスト表示位置変更
94  void MyDrawTexts::ChangePos(const float x, const float y) {
95      Draw::ChangePos(x, y);
96      float height = myDrawText[0]->GetHeight();
97      float yy = y - (height + inter) / 2 * (l - 1);
98      for (int i = 0; i < l; i++) {
99          myDrawText[i]->ChangePos(myDrawText[i]->GetX(), yy);
100         yy += height + inter;
101     }
102 }
103
104 // 複数行のテキスト表示文字変更
105 void MyDrawTexts::ChangeText(const char *str) {
106     for (int i = 0; i < l; i++)
107         delete myDrawText[i];
108
109     l = 0;
110     char a[256];
```

```
111     int i, j;
112     for (i = 0, j = 0; i < strlen(str); i++) {
113         a[j++] = str[i];
114         if (str[i + 1] == '\n' || i == strlen(str) - 1) {
115             a[j] = '\0';
116             myDrawText[l] = new MyDrawText(f, a, GetX(), 0, p, point,
                    color);
117             l++; i++; j = 0;
118         }
119     }
120     if (i == 0) {
121         myDrawText[0] = new MyDrawText(f, str, GetX(), 0, p, point, color
                );
122         l = 1;
123     }
124     ChangePos(GetX(), GetY());
125 }
126
127 // 複数行のテキスト表示幅取得
128 float MyDrawTexts::GetWidth() {
129     float max = 0;
130     for (int i = 0; i < l; i++) {
131         if (max < myDrawText[i]->GetWidth())
132             max = myDrawText[i]->GetWidth();
133     }
134     return max;
135 }
136
137 // 複数行のテキスト表示高さ取得
138 float MyDrawTexts::GetHeight() {
139     return  myDrawText[0]->GetHeight() * l;
140 }
141
142 // 複数行のテキストデストラクタ
143 MyDrawTexts::~MyDrawTexts() {
144     for (int i = 0; i < l; i++)
145         delete myDrawText[i];
146 }
147
148 // アンダーライン付きテキスト
149 //
      MyDrawTextLine（フォントポインタ、表示文字、x座標、y座標、ポジション情報、フォント
150 // ポジション情報（0：左寄せ、1：中央寄せ、2：右寄せ）
151 MyDrawTextLine::MyDrawTextLine(Font *font, const char *str, const float x,
        const float y, const int pos, const int point, const float lineLength
      , const float lineWidth, const char *colorName)
152     : Color(colorName), Draw(x, y) {
153     myDrawText = new MyDrawText(font, str, x, y, pos, point, colorName);
154     l = lineLength / SIZE_RATE;
155     w = lineWidth / SIZE_RATE;
156     this->pos = pos;
157     ChangePos(x, y);
158 }
159
160 // アンダーライン付きテキスト描画
161 void MyDrawTextLine::ContentView() {
162     myDrawText->View();
163     DrawLineAA(x1, y1, x2, y2, Color::Get(), w);
164 }
165
166 // アンダーライン付きテキスト表示位置変更
167 void MyDrawTextLine::ChangePos(const float x, const float y) {
```

```
168         x1 = x / SIZE_RATE - l / 2;
169         x2 = x / SIZE_RATE + l / 2;
170         y1 = y2 = (y + myDrawText->GetHeight() * 0.9) / SIZE_RATE;
171 
172         float xx;
173         switch (pos)
174         {
175         case 0:
176             xx = x1 * SIZE_RATE + 10;
177             break;
178         case 1:
179             xx = x;
180             break;
181         case 2:
182             xx = x2 * SIZE_RATE - 10;
183             break;
184         }
185         myDrawText->ChangePos(xx, y);
186     }
187 
188     // アンダーライン付きテキスト表示文字変更
189     void MyDrawTextLine::ChangeText(char *str) {
190         myDrawText->ChangeText(str);
191         ChangePos(GetX(), GetY());
192     }
193 
194     // アンダーライン付きテキストデストラクタ
195     MyDrawTextLine::~MyDrawTextLine() {
196         delete myDrawText;
197     }
```

Font.cpp

```cpp
#include "Font.h"

// フォント指定
Font::Font() {
    for(int i = 0; i < FONT_NUM; i++)
        id[p[i]] = CreateFontToHandle("M+ 1c", p[i] / SIZE_RATE, 1,
            DX_FONTTYPE_ANTIALIASING);
}

// フォント取得
int Font::Get(int point) {
    return id[point];
}

// フォントデストラクタ
Font::~Font() {
    for (int i = 0; i < FONT_NUM; i++)
        DeleteFontToHandle(id[p[i]]); // フォントデータを削除
}
```

Grading.cpp

```
1  #include "Grading.h"
2
3  FlameGrading::FlameGrading(FILE *modelfp) {
4      this->modelfp = modelfp;
5      modelflame = 0, j = 0;
6  }
7
8  int FlameGrading::Mark(float joints[JointType_Count][3], const int
       userflame) {
9      const int MAX = 1024;
10     while (j != JointType_Count || userflame > modelflame) {
11         char modelline[MAX];
12         if (fgets(modelline, MAX, modelfp) == NULL)
13             break;
14         sscanf(modelline, "%d:", &modelflame);
15         int num;
16         if (modelflame == 0)
17             num = 1;
18         else
19             num = (int)log10((double)modelflame) + 1;
20
21         char *line = modelline + num + 1;
22         for (j = 0; j < JointType_Count; j++) {
23             if (sscanf(line, "%f,%f,%f|", &model[j][0], &model[j][1], &
                   model[j][2]) != 3)
24                 break;
25             char str[256];
26             line = line + sprintf(str, "%f,%f,%f|", model[j][0], model[j
                   ][1], model[j][2]);
27         }
28     }
29     return (int)FlameMark(joints, model);
30 }
31
32 // 2関節間の点数計算
33 float FlameGrading::JointMark(float joints[JointType_Count][3], float
       model[JointType_Count][3], int x, int y) {
34     float userv, modelv;
35     float inner = 0, userlen = 0, modellen = 0;
36     for (int i = 0; i < 3; i++) {
37         userv = joints[x][i] - joints[y][i];
38         modelv = model[x][i] - model[y][i];
39         inner += userv * modelv;
40         userlen += pow(userv, 2);
41         modellen += pow(modelv, 2);
42     }
43     userlen = (float)sqrt(userlen);
44     modellen = (float)sqrt(modellen);
45     float point = (inner / (userlen * modellen));
46     if (point > 1)
47         point = 100;
48     else if (point > 0)
49         point = (1 - acos(point) * 2 / DX_PI) * 100;
50     else
51         point = 0;
52     return point * 1;
53 }
54
55 // 1フレームあたりの点数計算
56 float FlameGrading::FlameMark(float joints[JointType_Count][3], float
       model[JointType_Count][3]) {
```

```
57      const int MAX = 24;
58      float sum = 0;
59      int jointNum[MAX][2] = {
60          { JointType_SpineBase       , JointType_SpineMid },
61          { JointType_SpineBase       , JointType_HipLeft },
62          { JointType_SpineBase       , JointType_HipRight },
63          { JointType_SpineMid        , JointType_SpineShoulder },
64          { JointType_Neck            , JointType_SpineShoulder },
65          { JointType_Neck            , JointType_Head },
66          { JointType_ShoulderLeft    , JointType_ElbowLeft },
67          { JointType_ShoulderLeft    , JointType_SpineShoulder },
68          { JointType_ElbowLeft       , JointType_WristLeft },
69          { JointType_WristLeft       , JointType_HandLeft },
70          { JointType_HandLeft        , JointType_HandTipLeft },
71          { JointType_HandLeft        , JointType_ThumbLeft },
72          { JointType_ShoulderRight   , JointType_ElbowRight },
73          { JointType_ShoulderRight   , JointType_SpineShoulder },
74          { JointType_ElbowRight      , JointType_WristRight },
75          { JointType_WristRight      , JointType_HandRight },
76          { JointType_HandRight       , JointType_HandTipRight },
77          { JointType_HandRight       , JointType_ThumbRight },
78          { JointType_HipLeft         , JointType_KneeLeft },
79          { JointType_KneeLeft        , JointType_AnkleLeft },
80          { JointType_AnkleLeft       , JointType_FootLeft },
81          { JointType_HipRight        , JointType_KneeRight },
82          { JointType_KneeRight       , JointType_AnkleRight },
83          { JointType_AnkleRight      , JointType_FootRight } };
84      for (int i = 0; i < MAX; i++)
85          sum += JointMark(joints, model, jointNum[i][0], jointNum[i][1]);
86      return  sum / MAX;
87  }
88
89  Grading::Grading() {
90      bezier = new Bezier(1.2, 0, 0.5, 1);
91  }
92
93  void Grading::Mark(const char *model, const char *user) {
94      const int SCORE_FLAME = 500; // 一区切りあたりのフレーム
95
96      const char BAR_NUM = 9; // タイミング、表情のバーの数
97      FILE *userfp, *modelfp[BAR_NUM];
98      FlameGrading *flameGrading[BAR_NUM];
99      const int MAX = 1024;
100     char userline[MAX];
101     int i = 0, userflame = 0, sum = 0, count = 0, scoreCount = 0;
102     int timingSum[BAR_NUM] = {};
103     max = 0;
104
105     if ((userfp = fopen(user, "r")) == NULL) {
106         printf("file open error!!\n");
107         exit(EXIT_FAILURE);
108     }
109
110     for (int i = 0; i < BAR_NUM; i++) {
111         if ((modelfp[i] = fopen(model, "r")) == NULL) {
112             printf("file open error!!\n");
113             exit(EXIT_FAILURE);
114         }
115         flameGrading[i] = new FlameGrading(modelfp[i]);
116     }
117
118     while (fgets(userline, MAX, userfp) != NULL) {
119         float user[JointType_Count][3];
120         sscanf(userline, "%d:", &userflame);
```

```
121
122             int num;
123             if (userflame == 0)
124                 num = 1;
125             else
126                 num = (int)log10((double)userflame) + 1;
127
128             char *line = userline + num + 1;
129             for (i = 0; i < JointType_Count; i++) {
130                 if (sscanf(line, "%f,%f,%f|", &user[i][0], &user[i][1], &user[
                        i][2]) != 3)
131                     break;
132                 char str[256];
133                 line = line + sprintf(str, "%f,%f,%f|", user[i][0], user[i
                        ][1], user[i][2]);
134             }
135             if (i != JointType_Count)
136                 continue;
137
138             for (int k = 0; k < BAR_NUM; k++) {
139                 int x = k - BAR_NUM / 2;
140                 int point = (int)flameGrading[k]->Mark(user, userflame + x *
                        5);
141                 timingSum[k] += point;
142                 if (x == 0) {
143                     sum += point;
144                     score[max] += point;
145                     if (userflame >= SCORE_FLAME * (max + 1)) {
146                         score[max] = (int)(bezier->Calc((double)score[max] / (
                                count - scoreCount) / 100) * 100);
147                         score[max] = Adjust(score[max]);
148                         scoreCount = count;
149                         max++;
150                     }
151                 }
152             }
153             count++;
154         }
155         score[max] = (int)(bezier->Calc((double)score[max] / (count -
                scoreCount) / 100) * 100);
156         score[max] = Adjust(score[max]);
157         max++;
158         total = (int)(bezier->Calc((double)sum / count / 100) * 100);
159         total = Adjust(total);
160         timing = 0;
161         for (int k = 1; k < BAR_NUM; k++) {
162             if (timingSum[timing] < timingSum[k])
163                 timing = k;
164         }
165         printfDx("%d\n", timing);
166     }
167
168     int Grading::Adjust(int point) {
169         if (point > 100)
170             return 100;
171         else if (point < 0)
172             return 0;
173         return point;
174     }
175
176     Grading::~Grading() {
177         delete bezier;
178     }
```

Kinect.cpp

```
#include "Kinect.h"

// コンストラクタ
Kinect::Kinect() {
    if (KINECT_FLAG) {
        HRESULT hr;

        hr = GetDefaultKinectSensor(&m_pKinectSensor);
        if (FAILED(hr))
            exit(-1);

        if (m_pKinectSensor)
        {
            hr = m_pKinectSensor->Open();

            if (SUCCEEDED(hr))
            {
                kinectBody = new KinectBody(m_pKinectSensor);
//              kinectColor = new KinectColor(m_pKinectSensor);
            }
        }

        if (!m_pKinectSensor || FAILED(hr))
            exit(-1);
    }
}

void Kinect::Update() {
    if (KINECT_FLAG) {
        static boolean updateFlag = TRUE; // 更新用のフラグ、2回に一回しか処理をしない
        if (updateFlag) {
            kinectBody->Update(); // 骨格情報update
//          kinectColor->Update(); // 色情報update

            updateFlag = FALSE;
        }
        else {
            updateFlag = TRUE;
        }
    }
}

// デストラクタ
Kinect::~Kinect()
{
    delete kinectBody;
//  delete kinectColor;

    // close the Kinect Sensor
    if (m_pKinectSensor)
        m_pKinectSensor->Close();

    SafeRelease(m_pKinectSensor);
}
```

KinectBody.cpp

```
1  #include "KinectBody.h"
2
3  // コンストラクタ
4  KinectBody::KinectBody(IKinectSensor *m_pKinectSensor) {
5      userFlag = new boolean();
6      HRESULT hr;
7
8      IBodyFrameSource* pBodyFrameSource = NULL;
9
10     hr = m_pKinectSensor->get_BodyFrameSource(&pBodyFrameSource);
11
12     if (SUCCEEDED(hr))
13     {
14         hr = pBodyFrameSource->OpenReader(&m_pBodyFrameReader);
15     }
16
17     SafeRelease(pBodyFrameSource);
18 }
19
20 void KinectBody::Update() {
21     *userFlag = FALSE;
22     userJoints[0].Position.Z = 100;
23
24     if (!m_pBodyFrameReader)
25     {
26         return;
27     }
28
29     IBodyFrame* pBodyFrame = NULL;
30
31     HRESULT hr = m_pBodyFrameReader->AcquireLatestFrame(&pBodyFrame);
32
33     if (SUCCEEDED(hr))
34     {
35
36         INT64 nTime = 0;
37
38         hr = pBodyFrame->get_RelativeTime(&nTime);
39
40         IBody* ppBodies[BODY_COUNT] = { 0 };
41
42         if (SUCCEEDED(hr))
43         {
44             hr = pBodyFrame->GetAndRefreshBodyData(_countof(ppBodies),
                   ppBodies);
45         }
46         if (SUCCEEDED(hr))
47         {
48             hr = pBodyFrame->GetAndRefreshBodyData(_countof(ppBodies),
                   ppBodies);
49         }
50
51         if (SUCCEEDED(hr))
52         {
53             for (int i = 0; i < BODY_COUNT; ++i) {
54                 IBody* pBody = ppBodies[i];
55                 if (pBody)
56                 {
57                     BOOLEAN bTracked = false;
58                     hr = pBody->get_IsTracked(&bTracked);
59
```

```
60                      if (SUCCEEDED(hr) && bTracked) {
61                          Joint joints[JointType_Count];
62                          hr = pBody->GetJoints(_countof(joints), joints);
63                          if (SUCCEEDED(hr))
64                          {
65                              if (joints[0].Position.Z < userJoints[0].
                                    Position.Z) {
66                                  *userFlag = TRUE;
67                                  for (int i = 0; i < JointType_Count; i++)
68                                      userJoints[i] = joints[i];
69                              }
70                          }
71                      }
72                  }
73              }
74          }
75
76          for (int i = 0; i < _countof(ppBodies); ++i)
77          {
78              SafeRelease(ppBodies[i]);
79          }
80      }
81      SafeRelease(pBodyFrame);
82  }
83
84  // 距離を測定
85  boolean KinectBody::CheckDistance() {
86      const int maxMis = 60;
87      static int miss = maxMis;
88      if (KINECT_FLAG) {
89          if (*userFlag) {
90              if (miss == 0 && userJoints[0].Position.Z >= 1.5)
91                  miss = 0;
92              else if (miss != 0 && userJoints[0].Position.Z >= 2.0)
93                  miss = 0;
94              else
95                  miss = maxMis;
96          }
97          else {
98              miss++;
99          }
100     }
101     else {
102         if (CheckHitKey(KEY_INPUT_N) == 1)
103             miss = 0;
104         else
105             miss = maxMis;
106     }
107     if (miss < maxMis)
108         return TRUE;
109     else
110         return FALSE;
111 }
112
113 void KinectBody::StartSave(const char *fileName) {
114     if ((fp = fopen(fileName, "w")) == NULL) {
115         printf("file open error!!\n");
116         exit(EXIT_FAILURE);
117     }
118 }
119
120 // 保存
121 void KinectBody::JointSave(const int flame) {
122     static boolean flag = TRUE;
```

```
123         if (flag) {
124             fprintf(fp, "%d:", flame);
125             if (*userFlag) {
126                 for (int i = 0; i < JointType_Count; i++)
127                     fprintf(fp, "%f,%f,%f|", userJoints[i].Position.X,
                            userJoints[i].Position.Y, userJoints[i].Position.Z);
128                 putc('\n', fp);
129             }
130             else {
131                 fprintf(fp, "-1\n");
132             }
133             flag = FALSE;
134         }
135         else {
136             flag = TRUE;
137         }
138     }
139
140     void KinectBody::FinishSave() {
141         fclose(fp);
142     }
143
144     KinectBody::~KinectBody() {
145         SafeRelease(m_pBodyFrameReader);
146     }
```

MaiKagami.cpp

```
#include "MaiKagami.h"

MaiKagami::MaiKagami() {
    touch = new Touch();
    font = new Font();
    songs = new Songs(font); // 曲一覧作成
    user = new User();
    kinect = new Kinect(); // キネクト
    top = new Top(font, user);
    songSelect = new SongSelect(font, touch, songs, user);
    throughMain = new ThroughMain(font, touch, songs, kinect);
    throughResultMain = new ThroughResultMain(font, touch, songs, user);
    partMain = new PartMain(font, touch, songs, kinect);
    partResultMain = new PartResultMain(font, touch, songs);
    scene = TOP;
}

// 全体の算計
void MaiKagami::Update() {
    static int lastScene = TOP;
    touch->Check();
    switch (scene) {
    case TOP:
        scene = top->Switch(scene);
        break;
    case SONG_SELECT:
        scene = songSelect->Switch(scene);
        break;
    case THROUGH:
        scene = throughMain->Switch(scene);
        break;
    case THROUGH_RESULT:
        scene = throughResultMain->Switch(scene);
        break;
    case PART:
        scene = partMain->Switch(scene);
        break;
    case PART_RESULT:
        scene = partResultMain->Switch(scene);
        break;
    }

    if(lastScene != SONG_SELECT && scene == SONG_SELECT)
        songSelect->SetScene(MAIN);
    if (scene == THROUGH_OPTION) {
        songSelect->SetScene(OPTION1);
        scene = SONG_SELECT;
    }
    if (scene == PART_OPTION) {
        songSelect->SetScene(OPTION2);
        scene = SONG_SELECT;
    }

    lastScene = scene;

    kinect->Update();
    top->Update(scene); // トップ画面計算
    songSelect->Update(scene); // 曲選択画面計算
    throughMain->Update(scene); // 通し練習プレイ画面計算
    throughResultMain->Update(scene); // 通し練習結果画面計算
    partMain->Update(scene); // 部分練習プレイ画面計算
```

```
62      partResultMain->Update(scene); // 部分練習結果画面表示
63  }
64
65  // 全体の描画
66  void MaiKagami::View() {
67      top->View(); // トップ画面表示
68      songSelect->View(); // 曲選択画面表示
69      throughMain->View(); // 通し練習プレイ画面表示
70      throughResultMain->View(); // 通し練習結果画面表示
71      partMain->View(); // 部分練習プレイ画面表示
72      partResultMain->View(); // 部分練習結果画面表示
73  }
74
75  // デストラクタ
76  MaiKagami::~MaiKagami() {
77      delete top;
78      delete songSelect;
79      delete throughMain;
80      delete throughResultMain;
81      delete partMain;
82      delete partResultMain;
83      delete kinect;
84  }
```

Main.cpp

```cpp
#include "MaiKagami.h"
#include "Draw.h"
#include "DxLib.h"

int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int) {
    SetGraphMode(WIDTH / SIZE_RATE, HEIGHT / SIZE_RATE, 32); // ウィンドウサイズ設定
    SetBackgroundColor(0, 0, 0); // 背景色設定
    ChangeWindowMode(TRUE), DxLib_Init(), SetDrawScreen(DX_SCREEN_BACK); // ウィンドウモード変更と初期化と裏画面設定
    SetAlwaysRunFlag(TRUE); // バックグラウンドでも処理を実行
    MaiKagami *maiKagami = new MaiKagami(); // 舞鏡クラス作成

    SetUseASyncLoadFlag(TRUE); // 非同期読み込みon
                               // while(裏画面を表画面に反映, メッセージ処理, 画面クリア)
    while (ScreenFlip() == 0 && ProcessMessage() == 0 && ClearDrawScreen() == 0) {
        maiKagami->Update(); // 舞鏡画面計算
        maiKagami->View(); // 舞鏡画面表示
    }

    delete maiKagami; // 舞鏡クラス開放
    DxLib_End(); // DXライブラリ終了処理

    return 0;
}
```

ModeSelect.cpp

```
1  #include "ModeSelect.h"
2
3  // モード選択初期化
4  ModeSelectButton::ModeSelectButton(Font *font, Touch *touch) {
5      char *through = "一曲を通して練習できます。
           \nあなたにあったスピードで練習でき、\n分析・採点が行われます。";
6      char *part = "練習区間を設定して集中して練習で
           き\nます。練習途中であっても、スピード\
           nの変更や巻き戻しが自由にできます。";
7      button[0] = new TriangleButton2(font, touch, "通し練習モー
           ド", through, 2, 0, WIDTH * 0.57, "Blue");
8      button[1] = new TriangleButton2(font, touch, "部分練習モー
           ド", part, 2, 2, WIDTH * 0.57, "Yellow");
9      button[2] = new CircleButton2(font, touch, "戻る", 4);
10 }
11
12 int ModeSelectButton::Switch(const int scene) {
13     if (button[0]->GetTouch() == 1)
14         return OPTION1;
15     if (button[1]->GetTouch() == 1)
16         return OPTION2;
17     if (button[2]->GetTouch() == 1)
18         return MAIN;
19     return scene;
20 }
21
22 // モード選択ボタン計算
23 void ModeSelectButton::ContentUpdate() {
24     if (nowScene == MODE)
25         viewFlag = TRUE;
26     else
27         viewFlag = FALSE;
28 }
29
30 // モード選択ボタン表示
31 void ModeSelectButton::ContentView() {
32     for (int i = 0; i < 3; i++)
33         button[i]->View();
34 }
35
36 // モード選択削除
37 ModeSelectButton::~ModeSelectButton() {
38     for (int i = 0; i < 3; i++)
39         delete button[i];
40 }
```

Nfc.cpp

```cpp
#include "Nfc.h"
#include "Main.h"
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <string.h>

#define PORT 9999        // ポート番号
#define IP "127.0.0.1"   // IP番号(ローカルホスト)

void strReplace(std::string& str, const std::string& from, const std::
    string& to);

void Nfc::Init()
{
    WSADATA data;
    WSAStartup(MAKEWORD(2, 0), &data);
}

char* Nfc::GetId()
{
    if (CheckHitKey(KEY_INPUT_S)) // Sキー(スキップ)が押されたら
        return "daichi";
    if (!NFC_FLAG) // NFC_FLAGがfalseだったら
        return "\0";

    // 接続に失敗したときのエラー処理
    // またnfc監視を初めてから1秒間の間は0を返す
    calledCont++;
    if (!Connect(IP, PORT) || calledCont < 10) {
        return "\0";
    }

    int recvsize;                    // 受信データ長
    char recvMessage[5] = {"\0"};    // 受信バッファ
    char data[256] = { "\0" };       // 受信したIDを格納する変数
    char* p1 = data;                 // 実際にreturnするデータ

    // 受信
    // tcp/ip通信では4バイトごと送信される
    // つまり一回受信しただけでは完全に受信されていない可能性がある
    // なので何度か受信されたか確認することによって完全に受信させる
    while (true) {
        RECVSTATUS status = Recv(
            recvMessage,         // 受信データ格納用の配列
            sizeof(recvsize),    // 受信データ長
            &recvsize);          // 受信データ長のポインタ

        switch (status) {
        // データが来ていないとき
        case RECV_STILL:
            continue;
        // 成功
        case RECV_SUCCESSED:
            strcat_s(data, sizeof(data), recvMessage);
            for (int i = 0; i < 5; i++) {
                recvMessage[i] = '\0';
            }
            continue;
        // 切断orエラー
```

```
60          case RECV_FAILED:
61              break;
62          }
63
64          break;
65      }
66
67      // 制御文字の削除
68      // 処理を減らすためにデータがあるときのみ調べる
69      if (data[0] != '\0') {
70          // 文字列がナルになるか配列の範囲内でループ
71          // 確認されている制御文字はstx(0x02),\r(0x0a),\n(0x0d)
72          for (int i = 0; *(p1 + i) == '\0' || i < 256; i++) {
73              // まずstxの削除をする
74              // これは文頭につくのでアドレスを1つインクリメントする
75              if (*(p1 + i) == 0x02) {
76                  p1++;
77              }
78              // 残り二つを削除
79              // これは文末につくのでナルで上書きする
80              if (*(p1 + i) == 0x0a || *(p1 + i) == 0x0d)  {
81                  *(p1 + i) = '\0';
82              }
83          }
84      }
85
86      return p1;
87  }
88
89  // 接続
90  bool Nfc::Connect(const char* Ip, u_short Port)
91  {
92      // sockaddr_in 構造体のセット
93      struct sockaddr_in dstAddr;
94      memset(&dstAddr, 0, sizeof(dstAddr));
95      dstAddr.sin_port = htons(Port);
96      dstAddr.sin_family = AF_INET;
97      dstAddr.sin_addr.s_addr = inet_addr(Ip);
98
99      // ソケットの生成
100     m_DstSocket = socket(AF_INET, SOCK_STREAM, 0);
101
102     // 接続
103     if (connect(m_DstSocket, (struct sockaddr *) & dstAddr, sizeof(dstAddr
        ))
104         == SOCKET_ERROR) {
105         return false;
106     }
107     // ソケットを非同期モードにする
108     u_long val = 1;
109     ioctlsocket(m_DstSocket, FIONBIO, &val);
110     return true;
111 }
112
113 // 受信
114 RECVSTATUS Nfc::Recv(char* pData, int DataSize, int *pRecvSize)
115 {
116     int n = recv(m_DstSocket, pData, DataSize, 0);
117     if (n < 1) {
118         // データが来ていない
119         if (WSAGetLastError() == WSAEWOULDBLOCK) {
120             return RECV_STILL;
121         // 切断orエラー
```

```
122             } else {
123                 return RECV_FAILED;
124             }
125         }
126         *pRecvSize = n; // 受信データ長取得
127         return RECV_SUCCESSED;
128     }
129 
130     void Nfc::reset_calledCont() { calledCont = 0; }
131 
132     /**
133     * 文字列中から文字列を検索して別の文字列に置換する
134     * @param str  : 置換対象の文字列。上書かれます。
135     * @param from : 検索文字列
136     * @param to   : 置換後の文字列
137     */
138     void strReplace(std::string& str, const std::string& from, const std::
        string& to) {
139         std::string::size_type pos = 0;
140         while (pos = str.find(from, pos), pos != std::string::npos) {
141             str.replace(pos, from.length(), to);
142             pos += to.length();
143         }
144     }
```

PartMain.cpp

```
#include "PartMain.h"

PartMain::PartMain(Font *font, Touch *touch, Songs *songs, Kinect *kinect)
    {
    partStart = new PartStart(font);
    partPlay = new PartPlay(font, songs, touch, kinect);
    partPause = new PartPause(font, songs, touch);
}

void PartMain::ContentLoad() {
    scene = PART_START;
    partStart->Load();
    partPlay->Load();
    partPause->Load();
}

MainScene PartMain::Switch(const MainScene scene) {
    switch (this->scene)
    {
    case PART_COUNTDOWN:
    case PART_PLAY:
    case PART_START:
        this->scene = partPlay->Switch(this->scene);
    case PART_PAUSE:
    case PART_SETTING:
    case PART_SETTING_PART:
    case PART_SETTING_SPEED:
    case PART_REWIND:
        this->scene = partPause->Switch(this->scene);
        break;
    }

    switch (this->scene)
    {
    case PART_NEXT:
        Delete();
        return PART_RESULT;
    case PART_BACK_SONG_SELECT:
        Delete();
        return SONG_SELECT;
    }
    return PART;
}

void PartMain::ContentUpdate() {
    if (nowScene == PART) {
        Load();
        partStart->Update(scene);
        partPlay->Update(scene);
        partPause->Update(scene);
    }
}

void PartMain::ContentView() {
    partPlay->View();
    partStart->View();
    partPause->View();
}

void PartMain::ContentDelete() {
    partStart->Delete();
```

```
61      partPlay->Delete();
62      partPause->Delete();
63  }
64
65  PartMain::~PartMain() {
66      delete partStart;
67      delete partPlay;
68      delete partPause;
69  }
```

PartOption.cpp

```
1  #include "PartOption.h"
2
3  PartOptionPreview2::PartOptionPreview2(Font *font, Songs *songs, Touch *
       touch)
4      : PartOptionPreview(font, songs, touch, OPTION2, OPTION2_PART,
           OPTION2_SPEED) {
5      button[0] = new CircleButton(font, touch, "スタート!", 3, WIDTH * 0.8);
6      button[1] = new CircleButton2(font, touch, "戻る", 4);
7  }
8
9  int PartOptionPreview2::Switch(const int scene) {
10     if (button[0]->GetTouch() == 1) {
11         Song *song = songs->GetSong(songs->GetNowSong());
12         song->danceMovie->SetPart();
13         return NEXT2;
14     }
15     if (button[1]->GetTouch() == 1)
16         return MODE;
17     return PartOptionPreview::Switch(scene);
18 }
19
20 void PartOptionPreview2::ContentView() {
21     for (int i = 0; i < 2; i++)
22         button[i]->View();
23     PartOptionPreview::ContentView();
24 }
25
26 PartOptionPreview2::~PartOptionPreview2() {
27     for (int i = 0; i < 2; i++)
28         delete button[i];
29 }
30
31 PartOptionButton::PartOptionButton(Font *font, Songs *songs, Touch *touch)
32     : PartOptionPop(font, songs, touch, OPTION2, OPTION2_PART,
           OPTION2_SPEED, new PartOptionPreview2(font, songs, touch)) {}
```

PartOptionPop.cpp

```
1  #include "PartOptionPop.h"
2
3  PartOptionSpeedPop::PartOptionSpeedPop(Font *font, Songs *songs, Touch *
       touch, const int mainScene, const int speedScene)
4      : SpeedPop(font, songs, touch) {
5      this->mainScene = mainScene;
6      this->speedScene = speedScene;
7  }
8
9  int PartOptionSpeedPop::Switch(const int scene) {
10     if (button->GetTouch() == 1) {
11         song->danceMovie->SetSpeed();
12         return mainScene;
13     }
14     return scene;
15 }
16
17 void PartOptionSpeedPop::ContentUpdate() {
18     static int lastScene = mainScene;
19     if (nowScene == speedScene) {
20         if (lastScene == nowScene)
21             SpeedPop::ContentUpdate();
22         viewFlag = TRUE;
23     }
24     else {
25         viewFlag = FALSE;
26     }
27     lastScene = nowScene;
28 }
29
30 PartOptionPartPop::PartOptionPartPop(Font *font, Songs *songs, Touch *
       touch, const int mainScene, const int partScene)
31     : PartPop(font, songs, touch) {
32     this->mainScene = mainScene;
33     this->partScene = partScene;
34 }
35
36 int PartOptionPartPop::Switch(const int scene) {
37     if (button->GetTouch() == 1) {
38         song->danceMovie->SetPart();
39         return mainScene;
40     }
41     return scene;
42 }
43
44 void PartOptionPartPop::ContentUpdate() {
45     static int lastScene = mainScene;
46     if (nowScene == partScene) {
47         if (lastScene == mainScene)
48             PartPop::Init();
49         else if (lastScene == nowScene)
50             PartPop::ContentUpdate();
51         viewFlag = TRUE;
52     }
53     else {
54         viewFlag = FALSE;
55     }
56     lastScene = nowScene;
57 }
58
```

```
59  PartOptionPreview::PartOptionPreview(Font *font, Songs *songs, Touch *
        touch, const int mainScene, const int partScene, const int speedScene)
         {
60      this->songs = songs;
61      this->mainScene = mainScene;
62      this->partScene = partScene;
63      this->speedScene = speedScene;
64      button[0] = new CircleButton(font, touch, "区間", 0, WIDTH * 0.8);
65      button[1] = new CircleButton(font, touch, "", 2, WIDTH * 0.8);
66      message = new MyDrawText(font, "変更したいものを選んでくださ
            い", WIDTH * 0.75, HEIGHT * 0.45, 1, 30);
67      caption[0] = new MyDrawText(font, "開始：
            ", WIDTH * 0.8, HEIGHT * 0.53, 2, 30);
68      caption[1] = new MyDrawText(font, "終了：
            ", WIDTH * 0.8, HEIGHT * 0.555, 2, 30);
69      caption[2] = new MyDrawText(font, "速度：
            ", WIDTH * 0.8, HEIGHT * 0.6, 2, 30);
70      para[0] = new MyDrawText(font, "", WIDTH * 0.81, HEIGHT * 0.53, 0, 30,
             "Yellow");
71      para[1] = new MyDrawText(font, "", WIDTH * 0.81, HEIGHT * 0.555, 0,
            30, "Yellow");
72      para[2] = new MyDrawText(font, "", WIDTH * 0.81, HEIGHT * 0.6, 0, 30,
            "Yellow");
73  }
74
75  int PartOptionPreview::Switch(const int scene) {
76      if (button[0]->GetTouch() == 1)
77          return partScene;
78      if (button[1]->GetTouch() == 1)
79          return speedScene;
80      return scene;
81  }
82  void PartOptionPreview::ContentUpdate() {
83      Song *song = songs->GetSong(songs->GetNowSong());
84      if (nowScene == mainScene) {
85          char str[256];
86          sprintf_s(str, sizeof(str), "× %1.1lf", song->danceMovie->GetSpeed
                ());
87          para[0]->ChangeText(song->GetPart(song->StartPart())->GetName());
88          para[1]->ChangeText(song->GetPart(song->EndPart())->GetName());
89          para[2]->ChangeText(str);
90          viewFlag = TRUE;
91      }
92      else {
93          viewFlag = FALSE;
94      }
95  }
96
97  void PartOptionPreview::ContentView() {
98      for (int i = 0; i < 2; i++)
99          button[i]->View();
100     message->View();
101     for (int i = 0; i < 3; i++) {
102         caption[i]->View();
103         para[i]->View();
104     }
105 }
106
107 PartOptionPreview::~PartOptionPreview() {
108     for (int i = 0; i < 2; i++)
109         delete button[i];
110     delete message;
111     for (int i = 0; i < 3; i++) {
112         delete caption[i];
```

```
113             delete para[i];
114         }
115     }
116
117     PartOptionPop::PartOptionPop(Font *font, Songs *songs, Touch *touch, const
            int mainScene, const int partScene, const int speedScene,
        PartOptionPreview *partOptionPreview) {
118         this->songs = songs;
119         this->mainScene = mainScene;
120         this->partScene = partScene;
121         this->speedScene = speedScene;
122         this->partOptionPreview = partOptionPreview;
123         speedPop = new PartOptionSpeedPop(font, songs, touch, mainScene,
                speedScene);
124         partPop = new PartOptionPartPop(font, songs, touch, mainScene,
                partScene);
125     }
126
127     int PartOptionPop::Switch(const int scene) {
128         if (scene == mainScene)
129             return partOptionPreview->Switch(scene);
130         if (scene == partScene)
131             return partPop->Switch(scene);
132         if(scene == speedScene)
133             return speedPop->Switch(scene);
134         return scene;
135     }
136
137     void PartOptionPop::Load() {
138         speedPop->Load();
139         partPop->Load();
140         partOptionPreview->Load();
141     }
142
143     void PartOptionPop::Delete() {
144         speedPop->Delete();
145         partPop->Delete();
146         partOptionPreview->Delete();
147     }
148
149     void PartOptionPop::ContentUpdate() {
150         speedPop->Update(nowScene);
151         partPop->Update(nowScene);
152         partOptionPreview->Update(nowScene);
153         Song *song = songs->GetSong(songs->GetNowSong());
154
155         if (nowScene == mainScene || nowScene == partScene || nowScene ==
                speedScene) {
156             viewFlag = TRUE;
157             if (nowScene == mainScene) {
158                 song->ChangeStart(0);
159                 song->ChangeEnd(0);
160             }
161         }
162         else {
163             viewFlag = FALSE;
164         }
165     }
166
167     void PartOptionPop::ContentView() {
168         speedPop->View();
169         partPop->View();
170         partOptionPreview->View();
171     }
```

```
172
173 PartOptionPop::~PartOptionPop
174
175 () {
176     delete speedPop;
177     delete partPop;
178     delete partOptionPreview;
179 }
```

PartPause.cpp

```
1  #include "PartPause.h"
2
3  PartPauseButton::PartPauseButton(Touch *touch, Songs *songs) {
4      this->songs = songs;
5      button[0] = new CircleGraphButton(touch, 0, "img/pause.png");
6      button[1] = new CircleGraphButton(touch, 2, "img/pause.png");
7  }
8
9  void PartPauseButton::Load() {
10     for(int i = 0; i < 2; i++)
11         button[i]->Load();
12 }
13
14 int PartPauseButton::Switch(const int scene) {
15     static int lastScene = PART_START;
16     if (button[0]->GetTouch() == 1)
17         return PART_PAUSE;
18     if (button[1]->GetTouch() > 0)
19         return PART_REWIND;
20     if (scene == PART_REWIND)
21         return lastScene;
22     lastScene = scene;
23     return scene;
24 }
25
26 void PartPauseButton::ContentUpdate() {
27     Song *song = songs->GetSong(songs->GetNowSong());
28     if (button[1]->GetTouch() > 0 && nowScene == PART_REWIND) {
29         if(song->danceMovie->GetStartFlame() <= song->danceMovie->
               GetNowFlame())
30             song->danceMovie->Seek(song->danceMovie->GetNowFlame()-5);
31     }
32
33     switch (nowScene)
34     {
35     case PART_PLAY:
36     case PART_COUNTDOWN:
37     case PART_START:
38     case PART_REWIND:
39         viewFlag = TRUE;
40         break;
41     default:
42         viewFlag = FALSE;
43         break;
44     }
45 }
46
47 void PartPauseButton::ContentView() {
48     for (int i = 0; i < 2; i++)
49         button[i]->View();
50 }
51
52 void PartPauseButton::Delete() {
53     for (int i = 0; i < 2; i++)
54         button[i]->Release();
55 }
56
57 PartPauseButton::~PartPauseButton() {
58     for (int i = 0; i < 2; i++)
59         delete button[i];
60 }
```

```
61
62  PartPauseScreen::PartPauseScreen(Font *font, Songs *songs, Touch *touch)
63      : PauseScreen(font, songs, touch, PART_PAUSE, PART_START,
            PART_BACK_SONG_SELECT, PART_SETTING) {}
64
65  PartOptionPreview3::PartOptionPreview3(Font *font, Songs *songs, Touch *
        touch)
66      : PartOptionPreview(font, songs, touch, PART_SETTING,
            PART_SETTING_PART, PART_SETTING_SPEED) {
67      button = new CircleButton2(font, touch, "戻る", 4);
68      blackBox = new BlackBox();
69  }
70
71  int PartOptionPreview3::Switch(const int scene) {
72      if (button->GetTouch() == 1)
73          return PART_PAUSE;
74      return PartOptionPreview::Switch(scene);
75  }
76
77  void PartOptionPreview3::ContentView() {
78      blackBox->View();
79      button->View();
80      PartOptionPreview::ContentView();
81  }
82
83  PartOptionPreview3::~PartOptionPreview3() {
84      delete button;
85      delete blackBox;
86  }
87
88  PartPauseSetting::PartPauseSetting(Font *font, Songs *songs, Touch *touch)
89      : PartOptionPop(font, songs, touch, PART_SETTING, PART_SETTING_PART,
            PART_SETTING_SPEED, new PartOptionPreview3(font, songs, touch)) {}
90
91  PartPause::PartPause(Font *font, Songs *songs, Touch *touch) {
92      partPauseButton = new PartPauseButton(touch, songs); // ポーズボタン画
            面
93      partPauseScreen = new PartPauseScreen(font, songs, touch);
94      partPauseSetting = new PartPauseSetting(font, songs, touch);
95      flag = FALSE;
96  }
97
98  void PartPause::Load() {
99      partPauseButton->Load();
100     partPauseScreen->Load();
101     partPauseSetting->Load();
102 }
103
104 int PartPause::Switch(const int scene) {
105     switch (scene)
106     {
107     case PART_COUNTDOWN:
108     case PART_PLAY:
109     case PART_START:
110     case PART_REWIND:
111         return partPauseButton->Switch(scene);
112     case PART_PAUSE:
113         return partPauseScreen->Switch(scene);
114     case PART_SETTING:
115     case PART_SETTING_PART:
116     case PART_SETTING_SPEED:
117         return partPauseSetting->Switch(scene);
118     }
119     return scene;
```

```
120 }
121
122 void PartPause::ContentUpdate() {
123     partPauseButton->Update(nowScene);
124     partPauseScreen->Update(nowScene);
125     partPauseSetting->Update(nowScene);
126
127     switch (nowScene)
128     {
129     case PART_SETTING:
130     case PART_PLAY:
131     case PART_PAUSE:
132     case PART_SETTING_PART:
133     case PART_SETTING_SPEED:
134     case PART_COUNTDOWN:
135     case PART_REWIND:
136     case PART_START:
137         viewFlag = TRUE;
138         break;
139     default:
140         viewFlag = FALSE;
141         break;
142     }
143 }
144
145 void PartPause::ContentView() {
146     partPauseButton->View();
147     partPauseScreen->View();
148     partPauseSetting->View();
149 }
150
151 void PartPause::Delete() {
152     partPauseButton->Delete();
153     partPauseScreen->Delete();
154     partPauseSetting->Delete();
155 }
156
157 PartPause::~PartPause() {
158     delete partPauseButton;
159     delete partPauseScreen;
160     delete partPauseSetting;
161 }
```

PartPlay.cpp

```
1  #include "PartPlay.h"
2
3  PartStart::PartStart(Font *f)
4      : StartSceen(f, PART_START, PART_PLAY) {}
5
6  // 部分練習画面
7  PartPlay::PartPlay(Font *font, Songs *songs, Touch *touch, Kinect *kinect)
8      : PlayScreen(font, songs, touch, kinect, PART_START, PART_COUNTDOWN,
           PART_PLAY, PART_NEXT) {}
```

PartResult.cpp

```cpp
#include "PartResult.h"

PartResult::PartResult(Font *font, Songs *songs, Touch *touch) {
    this->songs = songs;
    this->font = font;
    title = new MyDrawTextLine(font, "採点結果", WIDTH * 0.5, HEIGHT * 0.15, 1, 60, WIDTH * 0.5, 4);
    button = new CircleButton2(font, touch, "次へ", 4);
}

void PartResult::Load() {
    song = songs->GetSong(songs->GetNowSong());
    song->coverGraph->Load();
    song->coverGraph->ChangePos(WIDTH * 0.3, HEIGHT * 0.26);
    song->drawSongTitle->ChangePos(WIDTH * 0.6, HEIGHT * 0.24);
    partMax = song->GetPartNum();
    for (int i = 0; i < partMax; i++) {
        SongPart *songPart = song->GetPart(i);
        float y = HEIGHT * 0.35 + HEIGHT * 0.35 * i / (partMax - 1);
        part[i] = new MyDrawText(font, songPart->GetName(), WIDTH * 0.27, y, 1, 30);
        circle[i] = new MyDrawCircle(WIDTH * 0.42, y, 16, "Blue");
        char str[256];
        sprintf_s(str, sizeof(str), "×%1.1lf", song->danceMovie->GetSpeed());
        speed[i] = new MyDrawText(font, str, WIDTH * 0.62, y, 1, 30);
        score[i] = new MyDrawText(font, "A", WIDTH * 0.77, y, 1, 30);
        if (i < song->StartPart() || i > song->EndPart()) {
            part[i]->SetAlpha(100);
            circle[i]->ChangeColor("White");
            circle[i]->SetAlpha(100);
            speed[i]->ChangeText("-");
            speed[i]->SetAlpha(100);
            score[i]->ChangeText("-");
            score[i]->SetAlpha(100);
        }
    }
}

int PartResult::Switch(const int scene) {
    if (button->GetTouch() == 1)
        return PART_RESULT_FINISH;
    return scene;
}

void PartResult::ContentUpdate() {
    if (nowScene == PART_RESULT_TOP || nowScene == PART_RESULT_FINISH)
        viewFlag = TRUE;
    else
        viewFlag = FALSE;
}

void PartResult::ContentView() {
    title->View();
    button->View();
    song->coverGraph->View();
    song->drawSongTitle->View();
    for (int i = 0; i < partMax; i++) {
        part[i]->View();
        circle[i]->View();
        speed[i]->View();
```

```
59          score[i]->View();
60      }
61  }
62
63  PartResult::~PartResult() {
64      delete title;
65      delete button;
66      for (int i = 0; i < partMax; i++) {
67          delete part[i];
68          delete circle[i];
69          delete speed[i];
70          delete score[i];
71      }
72  }
73
74  PartFinish::PartFinish(Font *font, Touch *touch) {
75      blackBox = new BlackBox();
76      button[0] = new CircleButton(font, touch, "もう一
            度", 0, WIDTH * 0.75, "White");
77      button[1] = new CircleButton(font, touch, "部分練
            習", 1, WIDTH * 0.75, "White");
78      button[2] = new CircleButton(font, touch, "通し練
            習", 2, WIDTH * 0.75, "White");
79      button[3] = new CircleButton(font, touch, "曲選択画
            面", 3, WIDTH * 0.75, "White");
80  }
81
82  int PartFinish::Switch(const int scene) {
83      if (button[0]->GetTouch() == 1)
84          return PART_RESULT_BACK_PLAY;
85      if (button[1]->GetTouch() == 1)
86          return PART_RESULT_BACK_PART_OPTION;
87      if (button[2]->GetTouch() == 1)
88          return PART_RESULT_BACK_THROUGH_OPTION;
89      if (button[3]->GetTouch() == 1)
90          return PART_RESULT_BACK_SONG_SELECT;
91      return scene;
92  }
93
94  void PartFinish::ContentUpdate() {
95      if (nowScene == PART_RESULT_FINISH)
96          viewFlag = TRUE;
97      else
98          viewFlag = FALSE;
99  }
100
101 void PartFinish::ContentView() {
102     blackBox->View();
103     for (int i = 0; i < 4; i++)
104         button[i]->View();
105 }
106
107 PartFinish::~PartFinish() {
108     delete blackBox;
109     for (int i = 0; i < 4; i++)
110         delete button[i];
111 }
```

PartResultMain.cpp

```cpp
#include "PartResultMain.h"

PartResultMain::PartResultMain(Font *font, Touch *touch, Songs *songs) {
    partResult = new PartResult(font, songs, touch);
    partFinish = new PartFinish(font, touch);
}

void PartResultMain::ContentLoad() {
    scene = PART_RESULT_TOP;
    partResult->Load();
    partFinish->Load();
}

MainScene PartResultMain::Switch(const MainScene scene) {
    switch (this->scene)
    {
    case PART_RESULT_TOP:
        this->scene = partResult->Switch(this->scene);
        break;
    case PART_RESULT_FINISH:
        this->scene = partFinish->Switch(this->scene);
        break;
    }
    if (this->scene == PART_RESULT_BACK_PLAY) {
        Delete();
        return PART;
    }
    if (this->scene == PART_RESULT_BACK_SONG_SELECT) {
        Delete();
        return SONG_SELECT;
    }
    if (this->scene == PART_RESULT_BACK_THROUGH_OPTION) {
        Delete();
        return THROUGH_OPTION;
    }
    if (this->scene == PART_RESULT_BACK_PART_OPTION) {
        Delete();
        return PART_OPTION;
    }
    return PART_RESULT;
}

void PartResultMain::ContentUpdate() {
    if (nowScene == PART_RESULT) {
        Load();
        partResult->Update(scene);
        partFinish->Update(scene);
    }
}

void PartResultMain::ContentView() {
    partResult->View();
    partFinish->View();
}

void PartResultMain::ContentDelete() {
    partResult->Delete();
    partFinish->Delete();
}

PartResultMain::~PartResultMain() {
```

```
62        delete partResult;
63        delete partFinish;
64    }
```

PauseScreen.cpp

```
1  #include "PauseScreen.h"
2
3  PauseScreen::PauseScreen(Font *font, Songs *songs, Touch *touch, const int
       pauseScene, const int startScene, const int songSelectScene, const
       int settingScene) {
4      this->songs = songs;
5      this->pauseScene = pauseScene;
6      this->startScene = startScene;
7      this->songSelectScene = songSelectScene;
8      this->settingScene = settingScene;
9      blackBox = new BlackBox();
10     title = new MyDrawText(font, "- ポーズ -", WIDTH * 0.95, HEIGHT *
           0.45, 2, 40, "Yellow");
11     button[0] = new CircleGraphTextButton(font, touch, "戻
           る", 0, "img/play.png");
12     button[1] = new CircleGraphTextButton(font, touch, "はじめか
           ら", 1, "img/rewind.png");
13     button[2] = new CircleGraphTextButton(font, touch, "曲選択
           へ", 2, "img/back.png");
14     button[3] = new CircleGraphTextButton(font, touch, "設定変
           更", 3, "img/setting.png");
15 }
16
17 void PauseScreen::Load() {
18     for (int i = 0; i < 4; i++)
19         button[i]->Load();
20 }
21
22 int PauseScreen::Switch(const int scene) {
23     Song *song = songs->GetSong(songs->GetNowSong());
24     // 戻るボタン
25     if (button[0]->GetTouch() == 1)
26         return startScene;
27     // 頭出しボタン
28     if (button[1]->GetTouch() == 1) {
29         song->danceMovie->Seek();
30         return startScene;
31     }
32     // 曲選択へ戻るボタン
33     if (button[2]->GetTouch() == 1)
34         return songSelectScene;
35     // 設定ボタン
36     if (button[3]->GetTouch() == 1)
37         return settingScene;
38     return scene;
39 }
40
41 void PauseScreen::ContentUpdate() {
42     if (nowScene == pauseScene)
43         viewFlag = TRUE;
44     else
45         viewFlag = FALSE;
46 }
47
48 void PauseScreen::ContentView() {
49     blackBox->View();
50     title->View();
51     for (int i = 0; i < 4; i++)
52         button[i]->View();
53 }
54
```

```
55 void PauseScreen::Delete() {
56     for (int i = 0; i < 4; i++)
57         button[i]->Release();
58 }
59
60 PauseScreen::~PauseScreen() {
61     delete blackBox;
62     delete title;
63     for (int i = 0; i < 4; i++)
64         delete button[i];
65 }
```

PlayScreen.cpp

```
1  #include "PlayScreen.h"
2
3  PlayScreen::PlayScreen(Font *font, Songs *songs, Touch *touch, Kinect *
       kinect, const int startScene, const int countDownScene, const int
       playScene, const int finishScene) {
4      this->startScene = startScene;
5      this->countDownScene = countDownScene;
6      this->playScene = playScene;
7      this->finishScene = finishScene;
8      this->songs = songs;
9      this->kinect = kinect;
10     playBar = new PlayBar(font);
11     countDown = new CountDown(font, countDownScene, playScene);
12 }
13
14 void PlayScreen::Load() {
15     song = songs->GetSong(songs->GetNowSong());
16     song->danceMovie->ChangeEx(1.2);
17     song->danceMovie->ChangePos(WIDTH * 0.5, HEIGHT * 0.5);
18     song->danceMovie->Seek();
19     song->drawSongTitle->ChangePos(WIDTH * 0.2, HEIGHT * 0.03);
20     playBar->Load(song);
21     viewFlag = TRUE;
22     kinect->kinectBody->StartSave("FILE/test.txt");
23 }
24
25 int PlayScreen::Switch(const int scene) {
26     if (kinect->kinectBody->CheckDistance() == FALSE) // ユーザが2
           mより近かったら
27         return startScene;
28     else if (scene == startScene)
29         return countDownScene;
30     else if (scene == countDownScene)
31         return countDown->Switch(scene);
32     else if (scene == playScene) {
33         if (song->danceMovie->GetNowFlame() == song->danceMovie->
               GetEndFlame()) {
34             song->danceMovie->Stop();
35             return finishScene;
36         }
37     }
38     return scene;
39 }
40
41 void PlayScreen::ContentUpdate() {
42     playBar->Update();
43     countDown->Update(nowScene);
44
45     if (nowScene == playScene) {
46         song->danceMovie->Start();
47         kinect->kinectBody->JointSave(song->danceMovie->GetNowFlame());
48     }
49     else {
50         song->danceMovie->Stop();
51     }
52 }
53
54 void PlayScreen::ContentView() {
55     song->danceMovie->View();
56     song->drawSongTitle->View();
57     playBar->View();
```

```
58      countDown->View();
59  }
60
61  void PlayScreen::Delete() {
62      kinect->kinectBody->FinishSave();
63  }
64
65
66  PlayScreen::~PlayScreen() {
67      delete playBar;
68      delete countDown;
69  }
```

PlayScreenObject.cpp

```
1  #include "PlayScreenObject.h"
2
3  // 進捗バー
4  PlayBar::PlayBar(Font *font) {
5      barAll = new MyDrawBar(WIDTH * 0.41, HEIGHT * 0.055, WIDTH * 0.56,
           10);
6      barNow = new MyDrawBar(WIDTH * 0.41, HEIGHT * 0.055, 0, 10, "Blue");
7      circle[0] = new MyDrawCircle(WIDTH * 0.41, HEIGHT * 0.055, 12);
8      circle[1] = new MyDrawCircle(WIDTH * 0.41, HEIGHT * 0.055, 5, "White"
           );
9      this->font = font;
10 }
11
12 void PlayBar::Load(Song *song) {
13     this->song = song;
14     song->LoadPart();
15 }
16
17 void PlayBar::Update() {
18     int nowFlame = song->danceMovie->GetNowFlame();
19     int startFlame = song->danceMovie->GetStartFlame();
20     int lastFlame = song->danceMovie->GetEndFlame();
21
22     for (int i = 0; i < song->GetPartNum(); i++) {
23         SongPart *songPart = song->GetPart(i);
24         float x = WIDTH * 0.41 + WIDTH * 0.56 * (float)(songPart->GetFlame
               () - startFlame) / (lastFlame - startFlame);
25         part[i] = new MyDrawTextV(font, songPart->GetName(), x, HEIGHT *
               0.056, 0, 16);
26         if (songPart->GetFlame() >= startFlame && songPart->GetFlame() <=
               lastFlame)
27             part[i]->SetViewFlag(TRUE);
28         else
29             part[i]->SetViewFlag(FALSE);
30     }
31
32     float now = WIDTH * 0.56 * (float)(nowFlame - startFlame) / (lastFlame
           - startFlame);
33     barNow->ChangeSize(now, 10);
34     for (int i = 0; i < 2; i++)
35         circle[i]->ChangePos(WIDTH * 0.41 + now, HEIGHT * 0.055);
36     for (int i = song->GetPartNum() - 1; i >= 0; i--) {
37         SongPart *songPart = song->GetPart(i);
38         if (nowFlame < lastFlame && nowFlame >= songPart->GetFlame()) {
39             part[i]->ChangeColor("Blue");
40             part[i]->ChangeFont(font, 20);
41         }
42         else {
43             part[i]->ChangeColor("White");
44             part[i]->ChangeFont(font, 16);
45         }
46         lastFlame = songPart->GetFlame();
47     }
48 }
49
50 void PlayBar::View() {
51     barAll->View();
52     barNow->View();
53     for (int i = 0; i < 2; i++)
54         circle[i]->View();
55     for (int i = 0; i < song->GetPartNum(); i++)
```

```
56          part[i]->View();
57  }
58
59  PlayBar::~PlayBar() {
60      delete barAll;
61      delete barNow;
62      for (int i = 0; i < 2; i++)
63          delete circle[i];
64  }
65
66  // カウントダウン画面再生三角形
67  PlayTriangle::PlayTriangle(const float x, const float y)
68      : MyDrawTriangle("Yellow") {
69      const float w = 100, ex = 12;
70      float x1 = x - w + ex, x2 = x - w + ex, x3 = x + w + ex;
71      float y1 = y + w, y2 = y - w, y3 = y;
72      ChangePos(x1, y1, x2, y2, x3, y3);
73  }
74
75  // カウントダウン画面
76  CountDown::CountDown(Font *font, const int thisScene, const int playScene)
        {
77      this->thisScene = thisScene;
78      this->playScene = playScene;
79      const float x = WIDTH * 0.5; // 円の中心（x座標）
80      const float y = HEIGHT * 0.5; // 円の中心（y座標）
81      const float r = WIDTH * 0.2; // 円の半径
82      text = new MyDrawText(font, "準備をしてください", x, y + r + 80, 1, 40);
83      circle = new MyDrawCircle(x, y, r, 3, "White"); // 縁が白色の円
84      countCircle1 = new MyDrawCircleGauge(x, y, r, 0, 5, "Blue");    // ゲ
            ージ
85      countCircle2 = new MyDrawCircle(0, 0, 12, "Blue");  // ゲージの先の円
86  }
87
88  int CountDown::Switch(const int scene) {
89      if (++count == max)
90          return playScene;
91      return scene;
92  }
93
94  void CountDown::ContentUpdate() {
95      if (nowScene == thisScene) {
96          viewFlag = TRUE;
97          countCircle1->ChangeDegree((double)count / max * 100);
98          countCircle2->ChangePos(countCircle1->GetEndX() * SIZE_RATE,
                countCircle1->GetEndY() * SIZE_RATE);
99      }
100     else {
101         count = 0;
102         viewFlag = FALSE;
103     }
104 }
105
106 void CountDown::ContentView() {
107     text->View();
108     circle->View();
109     countCircle1->View();
110     countCircle2->View();
111 }
112
113 CountDown::~CountDown() {
114     delete text;
115     delete circle;
```

```
116         delete countCircle1;
117         delete countCircle2;
118     }
```

Result.cpp

```cpp
#include "Result.h"

Result::Result(Songs *songs, User *user) {
    this->songs = songs;
    this->user = user;
}

void Result::Calc() {
    Song *song = songs->GetSong(songs->GetNowSong());
    char buf[256];
    sprintf(buf, "song/%s/model.txt", song->GetFolder());
    Mark(buf, "FILE/test.txt");
    strcpy(comment, "Bメロからサビに入ってからサビの終わりにかけてが苦手\nのように思います。そこを重点的に練習しましょう。");
    point[0] = 2;
    point[1] = 2;
    point[2] = 1;
    point[3] = 1;
    timing = 2;
    expression = 4;
}

// 送信
void Result::Send() {
    Song *song = songs->GetSong(songs->GetNowSong());
    // printfDx("%d\n", song->GetSongId()); // 曲ID
    // printfDx("%s\n", user->GetUserId()); // ユーザーID
}

float Result::GetTotal() {
    return total;
}

// 部位別得点取得
void Result::GetPoint(int x[4]) {
    for (int i = 0; i < 4; i++)
        x[i] = point[i];
}

// コメント取得
char *Result::GetComment() {
    return comment;
}

// タイミング取得
int Result::GetTiming() {
    return timing;
}

// 表情取得
int Result::GetExpression() {
    return expression;
}

// 区間別得点取得
int Result::GetScore(int x[100]) {
    for (int i = 0; i < max; i++)
        x[i] = score[i];
    return max;
}
```

Scene.cpp

```cpp
#include "Scene.h"

// 計算
void SubScene::Update(const int scene) {
    nowScene = scene;
    ContentUpdate();
}

void SubScene::Load() {
    viewFlag = TRUE;
}

void SubScene::Delete() {
    viewFlag = FALSE;
}

// 表示
void SubScene::View() {
    if (viewFlag)
        ContentView();
}

// 表示中かどうか確認する(TRUE:表示中、FALSE:非表示中)
boolean SubScene::CheckView() {
    return viewFlag;
}

// ロード
void Scene::Load() {
    if (loadFlag == 2)
        return;

    if (loadFlag == 0) {
        ContentLoad();
        loadFlag = 1;
    }

    if (loadFlag == 1 && GetASyncLoadNum() == 0) {
        viewFlag = TRUE;
        loadFlag = 2;
    }
}

// 削除
void Scene::Delete() {
    ContentDelete();
    viewFlag = FALSE;
    loadFlag = 0;
}
```

SettingPop.cpp

```
#include "SeetingPop.h"

// ポップアップ用四角形（黒色半透明全画面）
BlackBox::BlackBox()
    : MyDrawBox(WIDTH / 2, HEIGHT / 2, WIDTH, HEIGHT, "Black") {
    MyDrawBox::SetAlpha(220);
}

// スピードオプション表示
SpeedOption::SpeedOption(Font *font, Songs *songs, Touch *touch) {
    this->songs = songs;
    button[0] = new TriangleButton(font, touch, "UP", 0, 0);
    button[1] = new TriangleButton(font, touch, "DOWN", 1, 1);
    float height = BUTTON_POS + BUTTON_INTERVAL / 2;
    speed[0] = new MyDrawText(font, "スピー
        ド", WIDTH * 0.72, height, 0, 30);
    speed[1] = new MyDrawText(font, "× 1.0", WIDTH * 0.86, height, 0, 30,
        "Yellow");
}

void SpeedOption::Check() {
    Song *song = songs->GetSong(songs->GetNowSong());
    if (button[0]->GetTouch() == 1)
        song->ChangeSpeed(1);
    if (button[1]->GetTouch() == 1)
        song->ChangeSpeed(-1);
    char str[256];
    sprintf_s(str, sizeof(str), "× %1.1lf", song->danceMovie->GetSpeed());
    speed[1]->ChangeText(str);
}

void SpeedOption::View() {
    for (int i = 0; i < 2; i++) {
        button[i]->View();
        speed[i]->View();
    }
}

SpeedOption::~SpeedOption() {
    for (int i = 0; i < 2; i++) {
        delete button[i];
        delete speed[i];
    }
}

// 区間設定オプション表示
PartOption::PartOption(Font *font, Songs *songs, Touch *touch) {
    this->songs = songs;
    button[0] = new TriangleButton(font, touch, "", 0, 0);
    button[1] = new TriangleButton(font, touch, "", 1, 1);
    button[2] = new TriangleButton(font, touch, "", 0, 2);
    button[3] = new TriangleButton(font, touch, "", 1, 3);
    float height = BUTTON_POS + BUTTON_INTERVAL / 2;
    part[0] = new MyDrawText(font, "始め：", WIDTH * 0.67, height, 0, 30);
    part[1] = new MyDrawText(font, "", WIDTH * 0.79, height, 0, 30, "
        Yellow");
    part[2] = new MyDrawText(font, "終わり：
        ", WIDTH * 0.67, height + BUTTON_INTERVAL * 2, 0, 30);
    part[3] = new MyDrawText(font, "", WIDTH * 0.79, height +
        BUTTON_INTERVAL * 2, 0, 30, "Yellow");
}
```

```
57
58  void PartOption::Init() {
59      song = songs->GetSong(songs->GetNowSong());
60      part[1]->ChangeText(song->GetPart(song->StartPart())->GetName());
61      part[3]->ChangeText(song->GetPart(song->EndPart())->GetName());
62  }
63
64  void PartOption::Check() {
65      song = songs->GetSong(songs->GetNowSong());
66      if (button[0]->GetTouch() == 1)
67          song->ChangeStart(1);
68      if (button[1]->GetTouch() == 1)
69          song->ChangeStart(-1);
70      if (button[2]->GetTouch() == 1)
71          song->ChangeEnd(1);
72      if (button[3]->GetTouch() == 1)
73          song->ChangeEnd(-1);
74      part[1]->ChangeText(song->GetPart(song->StartPart())->GetName());
75      part[3]->ChangeText(song->GetPart(song->EndPart())->GetName());
76  }
77
78  void PartOption::View() {
79      for (int i = 0; i < 4; i++) {
80          button[i]->View();
81          part[i]->View();
82      }
83  }
84
85  PartOption::~PartOption() {
86      for (int i = 0; i < 4; i++) {
87          delete button[i];
88          delete part[i];
89      }
90  }
91
92  // スピードオプションポップアップ
93  SpeedPop::SpeedPop(Font *font, Songs *songs, Touch *touch) {
94      this->songs = songs;
95      speedOption = new SpeedOption(font, songs, touch);
96      blackBox = new BlackBox();
97      button = new CircleButton2(font, touch, "決定", 4);
98      text = new MyDrawText(font, "- 速度設定 -", WIDTH * 0.95, HEIGHT *
            0.45, 2, 40);
99  }
100
101 void SpeedPop::Load() {
102     song = songs->GetSong(songs->GetNowSong());
103 }
104
105 void SpeedPop::ContentUpdate() {
106     speedOption->Check();
107 }
108
109 void SpeedPop::ContentView() {
110     blackBox->View();
111     speedOption->View();
112     button->View();
113     text->View();
114 }
115
116 SpeedPop::~SpeedPop() {
117     delete speedOption;
118     delete blackBox;
```

```
119     delete button;
120     delete text;
121 }
122 
123 // 区間設定オプションポップアップ
124 PartPop::PartPop(Font *font, Songs *songs, Touch *touch) {
125     this->songs = songs;
126     blackBox = new BlackBox();
127     partOption = new PartOption(font, songs, touch);
128     button = new CircleButton2(font, touch, "決定", 4);
129     text = new MyDrawText(font, "- 区間設定 -", WIDTH * 0.95, HEIGHT *
            0.45, 2, 40);
130 }
131 
132 void PartPop::Load() {
133     song = songs->GetSong(songs->GetNowSong());
134 }
135 
136 void PartPop::ContentUpdate() {
137     partOption->Check();
138 }
139 
140 void PartPop::ContentView() {
141     blackBox->View();
142     partOption->View();
143     button->View();
144     text->View();
145 }
146 
147 void PartPop::Init() {
148     partOption->Init();
149 }
150 
151 PartPop::~PartPop() {
152     delete partOption;
153     delete blackBox;
154     delete button;
155     delete text;
156 }
```

Song.cpp

```
1  #include "Song.h"
2
3  // 履歴セット
4  void SongHistory::Set(const int history[2]) {
5      for (int i = 0; i < 2; i++)
6          this->history[i] = history[i];
7  }
8
9  // 履歴取得
10 void SongHistory::Get(int *history[2]) {
11     for (int i = 0; i < 2; i++)
12         *history[i] = this->history[i];
13 }
14
15 // 曲名、アーティスト情報
16 DrawSongTitle::DrawSongTitle(Font *font, const char *title, const char *
       artist) {
17     songTitle = new MyDrawTextLine(font, title,0, 0, 1, 30, WIDTH * 0.35,
           2); // テキスト初期化
18     songArtist = new MyDrawText(font, artist, 0, 0, 2, 20); // テキスト初
           期化
19 }
20
21 void DrawSongTitle::ChangePos(const float x, const float y) {
22     Pos::ChangePos(x, y);
23     songTitle->ChangePos(x, y);
24     songArtist->ChangePos(x + WIDTH * 0.17, y + HEIGHT * 0.025);
25 }
26
27 void DrawSongTitle::View() {
28     songTitle->View();
29     songArtist->View();
30 }
31
32 DrawSongTitle::~DrawSongTitle() {
33     delete songTitle;
34     delete songArtist;
35 }
36
37 // パート情報
38 void SongPart::Set(const int flame, const char *name) {
39     this->flame = flame;
40     strcpy_s(this->name, sizeof(this->name), name);
41 }
42
43
44 // フレーム数取得
45 int SongPart::GetFlame() {
46     return flame;
47 }
48
49 // パート名取得
50 char *SongPart::GetName() {
51     return name;
52 }
53
54 Song::Song(Font *font, const int id, const char *title, const char *artist
       , const char *folder) {
55     char cover[256], movie[256];
56     strcpy_s(Song::folder, sizeof(Song::folder), folder); // フォルダ
```

```
57      sprintf_s(cover, sizeof(cover), "song/%s/cover.jpg", folder); // カバー画像
58      sprintf_s(music, sizeof(music), "song/%s/music.mp3", folder); // 音楽
59      sprintf_s(movie, sizeof(movie), "song/%s/movie.ogv", folder); // 動画
60      Song::id = id;
61      n = new int();
62      songPartNum = new int();
63      start = new int();
64      end = new int();
65      *start = 0;
66      *end = 0;
67      for(int i = 0; i < 256; i++)
68          songPart[i] = new SongPart();
69
70      drawSongTitle = new DrawSongTitle(font, title, artist);
71      coverGraph = new MyDrawGraph(cover);
72      coverWhite = new MyDrawGraph("img/box.png");
73      danceMovie = new MyDrawMovie(movie);
74      songHistory = new SongHistory();
75  }
76
77  // 曲IDを取得
78  int Song::GetSongId() {
79      return id;
80  }
81
82  // 現在の位置IDを取得
83  int Song::GetNow() {
84      return *n;
85  }
86
87  // 位置IDをセット
88  void Song::SetNow(const int n) {
89      *Song::n = n;
90  }
91
92  // 動画の再生スピードを変更
93  void Song::ChangeSpeed(int num) {
94      const double s[6] = { 1.0, 0.9, 0.8, 0.7, 0.6, 0.5 };
95      static int x = 0;
96      if (num == 1 && x > 0) {
97          x -= 1;
98          danceMovie->ChangeSpeed(s[x]);
99      }
100     if (num == -1 && x < 5) {
101         x += 1;
102         danceMovie->ChangeSpeed(s[x]);
103     }
104 }
105
106 // 動画の開始位置を変更
107 void Song::ChangeStart(int num) {
108     if (num == 1 && *start > 0)
109         (*start) -= 1;
110     if (num == -1 && *start < *end)
111         (*start) += 1;
112     danceMovie->SetStartFlame(GetPart(*start)->GetFlame());
113 }
114
115 // 動画の終了位置を変更
116 void Song::ChangeEnd(int num) {
117     if (num == 1 && *end > *start)
118         (*end) -= 1;
```

```
119         if (num == -1 && *end < GetPartNum() - 1)
120             (*end) += 1;
121 
122         if (*end + 1 == GetPartNum())
123             danceMovie->SetEndFlame(danceMovie->GetAllFlame());
124         else
125             danceMovie->SetEndFlame(GetPart(*end + 1)->GetFlame());
126 
127     }
128 
129     int Song::StartPart() {
130         return *start;
131     }
132 
133     int Song::EndPart() {
134         return *end;
135     }
136 
137     // パート情報をロード
138     void Song::LoadPart() {
139         char part[256];
140         sprintf_s(part, sizeof(part), "song/%s/part.csv", folder);
141         SetUseASyncLoadFlag(FALSE);
142         int file = FileRead_open(part, FALSE);
143         SetUseASyncLoadFlag(TRUE);
144         char buf[256];
145         int flame;
146         *songPartNum = 0;
147         while (FileRead_scanf(file, "%d,%[^\n\r]", &flame, buf) != EOF) {
148             songPart[*songPartNum]->Set(flame, buf);
149             (*songPartNum)++;
150         }
151     }
152 
153     // パート情報取得
154     SongPart *Song::GetPart(int num) {
155         return songPart[num];
156     }
157 
158     // パート数取得
159     int Song::GetPartNum() {
160         return *songPartNum;
161     }
162 
163     // フォルダ取得
164     char *Song::GetFolder() {
165         return folder;
166     }
```

Songs.cpp

```
1  #include "Songs.h"
2
3  // 曲数
4  #define NUMSONGS 256
5
6  LPBYTE ReadData(HINTERNET hRequest, LPDWORD lpdwSize);
7
8  Songs::Songs(Font *font) {
9      n = 0;
10     SetUseASyncLoadFlag(FALSE);
11     int file = FileRead_open("song/song.csv", FALSE);
12     SetUseASyncLoadFlag(TRUE);
13     char buf[3][256];
14     int id = 0;
15     while (FileRead_scanf(file, "%d,%[^,\n\r],%[^,\n\r],%[^\n\r]", &id,
           buf[0], buf[1], buf[2]) != EOF) {
16         song[n] = new Song(font, id, buf[1], buf[2], buf[0]);
17         n++;
18     }
19     FileRead_close(file);
20 }
21
22 // 曲数取得
23 int Songs::GetSongNum() {
24     return n;
25 }
26
27 // 曲取得
28 Song *Songs::GetSong(int x) {
29     return song[x];
30 }
31
32 // 現在選択されている曲取得
33 int Songs::GetNowSong() {
34     for (int i = 0; i < n; i++) {
35         if (song[i]->GetNow() == 0)
36             return i;
37     }
38     return 0;
39 }
40
41 LPBYTE ReadData(HINTERNET hRequest, LPDWORD lpdwSize)
42 {
43     LPBYTE lpData = NULL;
44     LPBYTE lpPrev = NULL;
45     DWORD  dwSize;
46     DWORD  dwTotalSize = 0;
47     DWORD  dwTotalSizePrev = 0;
48
49     for (;;) {
50         WinHttpQueryDataAvailable(hRequest, &dwSize);
51         if (dwSize > 0) {
52             dwTotalSizePrev = dwTotalSize;
53             dwTotalSize += dwSize;
54             lpData = (LPBYTE)HeapAlloc(GetProcessHeap(), 0, dwTotalSize);
55 //          if (lpPrev != NULL) {
56 //              CopyMemory(lpData, lpPrev, dwTotalSizePrev);
57 //              HeapFree(GetProcessHeap(), 0, lpPrev);
58 //          }
59             WinHttpReadData(hRequest, lpData + dwTotalSizePrev, dwSize,
                   NULL);
```

```
60              lpPrev = lpData;
61          } else
62              break;
63      }
64
65      *lpdwSize = dwTotalSize;
66
67      return lpData;
68  }
69
70  int Songs::LoadHistory(const char *userId) {
71      // ここでサーバに接続して前回と前々回の点数を受信
72      HINTERNET       hSession, hConnect, hRequest;
73      URL_COMPONENTS urlComponents;
74      WCHAR           szHostName[256], szUrlPath[2048];
75      // URL
76      WCHAR           szUrl[256] = L"http:// globalstudios.jp/mai-archive/
            api_history.php?user=";
77      WCHAR           szUserId[18];
78  //  printfDx("%d, %d, %d, %d, %d, %d, %d, %d\n", userId[0], userId[1],
        userId[2],userId[3],userId[4],userId[5], userId[6], userId[7]);
79  //  printfDx("%d, %d, %d, %d, %d, %d\n", 'd', 'a', 'i', 'c', 'h', 'i');
80      mbstowcs(szUserId, userId, 256);
81      wcscat(szUrl, szUserId);
82      LPBYTE          lpData;
83      DWORD           dwSize;
84
85      hSession = WinHttpOpen(L"Sample Application/1.0",
86          WINHTTP_ACCESS_TYPE_DEFAULT_PROXY,
87          WINHTTP_NO_PROXY_NAME,
88          WINHTTP_NO_PROXY_BYPASS,
89          0);
90      if (hSession == NULL)
91          return -1;
92
93      ZeroMemory(&urlComponents, sizeof(URL_COMPONENTS));
94      urlComponents.dwStructSize = sizeof(URL_COMPONENTS);
95      urlComponents.lpszHostName = szHostName;
96      urlComponents.dwHostNameLength = sizeof(szHostName) / sizeof(WCHAR);
97      urlComponents.lpszUrlPath = szUrlPath;
98      urlComponents.dwUrlPathLength = sizeof(szUrlPath) / sizeof(WCHAR);
99
100
101     if (!WinHttpCrackUrl(szUrl, lstrlenW(szUrl), 0, &urlComponents)) {
102         WinHttpCloseHandle(hSession);
103         return -1;
104     }
105
106     // 接続
107     hConnect = WinHttpConnect(hSession, szHostName, INTERNET_DEFAULT_PORT,
            0);
108     if (hConnect == NULL) {
109         WinHttpCloseHandle(hSession);
110         return -1;
111     }
112
113     hRequest = WinHttpOpenRequest(hConnect,
114         L"GET",
115         szUrlPath,
116         NULL,
117         WINHTTP_NO_REFERER,
118         WINHTTP_DEFAULT_ACCEPT_TYPES,
119         0);
120     if (hRequest == NULL) {
```

```
121             WinHttpCloseHandle(hConnect);
122             WinHttpCloseHandle(hSession);
123             return -1;
124         }
125
126         if (!WinHttpSendRequest(hRequest, WINHTTP_NO_ADDITIONAL_HEADERS, 0,
                WINHTTP_NO_REQUEST_DATA, 0, WINHTTP_IGNORE_REQUEST_TOTAL_LENGTH,
                0)) {
127             WinHttpCloseHandle(hRequest);
128             WinHttpCloseHandle(hConnect);
129             WinHttpCloseHandle(hSession);
130             return 0;
131         }
132
133         WinHttpReceiveResponse(hRequest, NULL);
134
135         // ボディ取得
136         lpData = ReadData(hRequest, &dwSize);
137 //      printfDx((char*)lpData);
138         for (int i = 0; i < NUMSONGS; i++) {
139             char* temp = NULL;
140             char* ctx;// 内部的に使用するので深く考えない
141
142             if (i == 0) {
143                 temp = strtok_s((char*)lpData, "\n", &ctx);
144
145             } else {
146                 temp = strtok_s(0, "\n", &ctx);
147             }
148             if (temp == NULL)break;
149             int history[2];
150             int hoge;
151             sscanf_s(temp, "%d||%d||%d", &hoge, &history[0], &history[1]);
152             // 以下の式を実行することによってデータを保存
153             // song[Search(<曲ID>)]->songHistory->Set(＜前回と前々回の点数（配
                列ポインタ）＞);
154             song[Search(hoge)]->songHistory->Set(history);
155         }
156         HeapFree(GetProcessHeap(), 0, lpData);
157
158         WinHttpCloseHandle(hRequest);
159         WinHttpCloseHandle(hConnect);
160         WinHttpCloseHandle(hSession);
161
162         return 0;
163     }
164
165     int Songs::Search(const int songId) {
166         for (int i = 0; i < n; i++) {
167             if (song[i]->GetSongId() == songId)
168                 return i;
169         }
170         return -1;
171     }
```

SongSelect.cpp

```
1  #include "SongSelect.h"
2
3  // 曲選択画面ボタン初期化
4  SongSelectButton::SongSelectButton(Font *font, Touch *touch) {
5      button[0] = new TriangleButton(font, touch, "前の曲へ", 0, 0);
6      button[1] = new CircleButton(font, touch, "決定", 1);
7      button[2] = new TriangleButton(font, touch, "次の曲へ", 1, 2);
8      button[3] = new CircleButton2(font, touch, "終了", 4);
9  }
10
11 // 曲選択画面ボタン画面切り替え
12 int SongSelectButton::Switch(const int scene) {
13     if (button[1]->GetTouch() == 1)
14         return MODE;
15     if (button[3]->GetTouch() == 1)
16         return BACK;
17     return scene;
18 }
19
20 // 曲選択画面ボタン計算
21 void SongSelectButton::ContentUpdate() {
22     if (nowScene == MAIN)
23         viewFlag = TRUE;
24     else
25         viewFlag = FALSE;
26 }
27
28 // 曲選択画面ボタン表示
29 void SongSelectButton::ContentView() {
30     for (int i = 0; i < 4; i++)
31         button[i]->View();
32 }
33
34 SongSelectButton::~SongSelectButton() {
35     for (int i = 0; i < 4; i++)
36         delete button[i];
37 }
38
39 // 終了用ポップアップ
40 SongSelectPop::SongSelectPop(Font *font, Touch *touch) {
41     this->touch = touch;
42     blackBox = new BlackBox();
43     title = new MyDrawText(font, "- 終了 -", WIDTH * 0.75, HEIGHT * 0.4,
           1, 40, "Blue");
44     message = new MyDrawText(font, "本当に終
           了\nしますか？", WIDTH * 0.75, HEIGHT * 0.45, 1, 30);
45     button[0] = new CircleButton(font, touch, "は
           い", 1, WIDTH * 0.75, "White");
46     button[1] = new CircleButton(font, touch, "いい
           え", 2, WIDTH * 0.75, "White");
47 }
48
49 int SongSelectPop::Switch(const int scene) {
50     if (touch->Get(1) == 1)
51         return BACK_TOP;
52     if (touch->Get(2) == 1)
53         return MAIN;
54     return scene;
55 }
56
57 void SongSelectPop::ContentUpdate() {
```

```
58      if (nowScene == BACK)
59          viewFlag = TRUE;
60      else
61          viewFlag = FALSE;
62  }
63
64  void SongSelectPop::ContentView() {
65      blackBox->View();
66      title->View();
67      message->View();
68      for (int i = 0; i < 2; i++)
69          button[i]->View();
70  }
71
72  SongSelectPop::~SongSelectPop() {
73      delete blackBox;
74      delete title;
75      delete message;
76      for (int i = 0; i < 2; i++)
77          delete button[i];
78  }
```

SongSelectCommon.cpp

```
1  #include "SongSelectCommon.h"
2
3  // 曲選択画面タイトルロード
4  SongSelectTitle::SongSelectTitle(Font *font) {
5      title = new DrawTitle(font, "");
6      subTitle = new DrawSubtitle(font, "");
7  }
8
9  // 曲選択画面タイトル計算
10 void SongSelectTitle::ContentUpdate() {
11     static int lastScene = -100;
12     if (lastScene != nowScene) {
13         lastScene = nowScene;
14         switch (nowScene)
15         {
16         case MODE:
17             title->ChangeText("Mode Select");
18             subTitle->SetViewFlag(FALSE);
19             break;
20         case OPTION1:
21             title->ChangeText("Option");
22             subTitle->ChangeText("通し練習モード");
23             subTitle->SetViewFlag(TRUE);
24             subTitle->ChangeColor("Blue");
25             break;
26         case OPTION2:
27         case OPTION2_PART:
28         case OPTION2_SPEED:
29             title->ChangeText("Option");
30             subTitle->ChangeText("部分練習モード");
31             subTitle->SetViewFlag(TRUE);
32             subTitle->ChangeColor("Yellow");
33             break;
34         default:
35             title->ChangeText("Song Select");
36             subTitle->SetViewFlag(FALSE);
37             break;
38         }
39     }
40 }
41
42 // 曲選択画面タイトル表示
43 void SongSelectTitle::ContentView() {
44     title->View(); // テキスト表示
45     subTitle->View(); // テキスト表示
46 }
47
48 SongSelectTitle::~SongSelectTitle() {
49     delete title;
50     delete subTitle;
51 }
52
53 // 曲選択画面カバー画像初期化
54 SongInformation::SongInformation(Font *font, Songs *songs, Touch *touch,
       User *user) {
55     this->songs = songs;
56     this->touch = touch;
57     this->user = user;
58     n = songs->GetSongNum();
59     for (int i = 0; i < n; i++) {
60         songCover[i] = new SongSelectCover(font, songs->GetSong(i), i);
```

```
61          // songCover[i]->Change(0, n);  //
                Updateに統合したから要らなくなった Jaity
62      }
63
64      float x = HEIGHT * 0.35;
65      myDrawBox = new MyDrawBox(WIDTH * 0.5, HEIGHT * 0.5, 170, 1000);
66      myDrawBox->SetAlpha(90); // 透明度指定
67      grad[0] = new MyDrawGraph(WIDTH * 0.5, HEIGHT * 0.22-30, "img/grad1.
            png");
68      grad[1] = new MyDrawGraph(WIDTH * 0.5, HEIGHT * 0.8, "img/grad2.png");
69  //  box = new MyDrawGraph(WIDTH * 0.5, x, "img/box.png");
70      songLast[0] = new MyDrawText(font, "前回  :
            --点", WIDTH * 0.75, HEIGHT * 0.36, 0, 24); // テキスト初期化
71      songLast[1] = new MyDrawText(font, "前々回:
            --点", WIDTH * 0.75, HEIGHT * 0.385, 0, 24); // テキスト初期化
72  }
73
74
75  void SongInformation::Load() {
76      songs->LoadHistory(user->GetUserId());
77      for (int i = 0; i < 2; i++)
78          grad[i]->Load();
79      for (int i = 0; i < n; i++)
80          songCover[i]->Load(n);
81      nowSong = songCover[songs->GetNowSong()];
82      nowSong->drawSongTitle->ChangePos(WIDTH * 0.79, HEIGHT * 0.3);
83      viewFlag = TRUE;
84  }
85
86  void SongInformation::ContentUpdate() {
87      int direct = 0;  // increase or decrease of IDs  Jaity
88      static int lastScene = nowScene;
89      int *last[2] = { new int(), new int() }; // 履歴保存用
90
91      switch (nowScene)
92      {
93      case MAIN:
94          if (nowScene == lastScene) {
95              // ボタン0が押されたら
96              if(touch->Input2(0)) {
97                  direct = 1;  // Jaity
98                  for (int i = 0; i < n; i++) {
99                      songCover[i]->coverGraph->Reset();
100                     songCover[i]->coverWhite->Reset();
101                 }
102             }
103
104             // ボタン2が押されたら
105             if(touch->Input2(2)) {
106                 direct = -1;  // Jaity
107                 for (int i = 0; i < n; i++) {
108                     songCover[i]->coverGraph->Reset();
109                     songCover[i]->coverWhite->Reset();
110                 }
111             }
112         }
113
114         for (int i = 0; i < n; i++)
115             songCover[i]->Update(direct, n);  // Updateに引数追加 Jaity
116
117         nowSong = songCover[songs->GetNowSong()];
118         nowSong->drawSongTitle->ChangePos(WIDTH * 0.79, HEIGHT * 0.3);
119         nowSong->songHistory->Get(last);
```

```
120             for (int i = 0; i < 2; i++) {
121                 char str[256];
122                 char text[2][10] = { "前回  ", "前々回" };
123                 if (*last[i] == -1)
124                     sprintf_s(str, sizeof(str), "%s：  --点", text[i]);
125                 else
126                     sprintf_s(str, sizeof(str), "%s：%3d点", text[i], *last[i
                        ]);
127                 songLast[i]->ChangeText(str);
128             }
129 
130             if(lastScene == MODE)
131                 nowSong->danceMovie->Release();
132             break;
133         case MODE:
134             if (lastScene != MODE) {
135                 if (lastScene == OPTION1) {
136                     nowSong->danceMovie->Seek(0);
137                 }
138                 else {
139                     nowSong->danceMovie->Load();
140                     nowSong->LoadPart();
141                     nowSong->danceMovie->ChangeSpeed(nowSong->danceMovie->
                        GetSpeed());
142                 }
143             }
144             break;
145         case OPTION1:
146             if(lastScene != OPTION1)
147                 nowSong->danceMovie->Seek(0);
148             break;
149         }
150         lastScene = nowScene;
151     }
152 
153     // 曲選択画面カバー画像表示
154     void SongInformation::ContentView() {
155         nowSong->drawSongTitle->View();
156         for (int i = 0; i < 2; i++) {
157             songLast[i]->View();
158         }
159         switch (nowScene)
160         {
161         case BACK:
162         case MAIN:
163             myDrawBox->View();
164     //      box->View();
165             for (int i = 0; i < n; i++)
166                 songCover[i]->Draw(nowScene);
167             for (int i = 0; i < 2; i++)
168                 grad[i]->View();
169             break;
170         case MODE:
171         case OPTION1:
172         case OPTION2:
173         case OPTION2_PART:
174         case OPTION2_SPEED:
175             nowSong->Draw(nowScene);
176             break;
177         }
178     }
179 
180     void SongInformation::Delete() {
```

```
181         for (int i = 0; i < 2; i++)
182             grad[i]->Release();
183         for (int i = 0; i < n; i++)
184             songCover[i]->Release();
185         SubScene::Delete();
186     }
187 
188     SongInformation::~SongInformation() {
189     //  delete box;
190         delete myDrawBox;
191         for (int i = 0; i < n; i++)
192             delete songCover[i];
193         for (int i = 0; i < 2; i++) {
194             delete grad[i];
195         }
196     }
```

SongSelectCover.cpp

```cpp
#include "SongSelectCover.h"
#include "Animation.h"

SongSelectCover::SongSelectCover(Font *font, Song *song, const int now)
    : Song(*song) {
    char *folder = "";
    SetNow(now);
}

void SongSelectCover::Load(int max) {
    danceMovie->ChangePos(WIDTH * 0.5, HEIGHT * 0.57);
    danceMovie->ChangeEx(0.5);
    coverGraph->Load();
    coverWhite->Load();
    Change(0, max);

    coverGraph->ChangePos(WIDTH * 0.5, CalcY());
    coverGraph->SetAlpha(CalcAlpha());
    coverGraph->ChangeEx(CalcEx());
    coverWhite->ChangePos(WIDTH * 0.5, CalcY());
    coverWhite->SetAlpha(CalcAlphaWhite());
    coverWhite->ChangeEx(CalcEx());
    playFlag = FALSE;
}

void SongSelectCover::Release() {
    coverGraph->Release();
    coverWhite->Release();
}

// 表示位置の計算
void SongSelectCover::Update(int num, int max) {
    // static int t = 0;    // 邪魔 Jaity
    Change(num, max);
    int n = GetNow();
    int duration = 20;
    float y = CalcY();

    if (n == -2 && num > 0 || n == max - 3 && num < 0) {
        coverGraph->SetDuration(0);
        coverGraph->SetPosAnimation(WIDTH * 0.5, y);
        coverWhite->SetDuration(0);
        coverWhite->SetPosAnimation(WIDTH * 0.5, y);
    }
    else if (coverGraph->GetTime() == 0) { // 最初だけ
        coverGraph->SetDuration(duration);
        coverGraph->SetPosAnimation(WIDTH * 0.5, y, Animation::
            EaseOut_SINE);
        coverWhite->SetDuration(duration);
        coverWhite->SetPosAnimation(WIDTH * 0.5, y, Animation::
            EaseOut_SINE);
    }

    coverGraph->SetExAnimation(CalcEx(), Animation::EaseOut_SINE);
    coverGraph->SetAlphaAnimation(CalcAlpha(), Animation::EaseOut_SINE);
    coverWhite->SetExAnimation(CalcEx(), Animation::EaseOut_SINE);
    coverWhite->SetAlphaAnimation(CalcAlphaWhite(), Animation::
        EaseOut_SINE);

    coverGraph->Update(); // アニメーション更新
    coverWhite->Update();
```

```
59  }
60
61  void SongSelectCover::Draw(int scene) {
62      int n = GetNow();
63      if (n <= 6) {   // 移動中を考えて 5 も描画
64          coverWhite->View();
65          coverGraph->View();
66      }
67      switch (scene) {
68      case OPTION1:
69          if(playFlag)
70              StopMusic();
71          playFlag = FALSE;
72          danceMovie->Start();
73          danceMovie->View();
74          break;
75      case OPTION2:
76          if (playFlag)
77              StopMusic();
78          playFlag = FALSE;
79          break;
80      default:
81          if (n == 0 && !playFlag) {
82  //          PlayMusic(music, DX_PLAYTYPE_LOOP); // 重いので一時的に消去
        Jaity
83              playFlag = TRUE;
84          }
85          else if (n != 0) {
86              playFlag = FALSE;
87          }
88          break;
89      }
90  }
91
92  // 曲の位置IDを変更
93  void SongSelectCover::Change(int num, int max) {
94      int n = GetNow();
95      n = (n + num + max + 2) % max - 2;
96      SetNow(n);
97  }
98
99  // y座標を算出して取得
100 float SongSelectCover::CalcY() {
101     int n = GetNow();
102     float y;
103
104     n = n < 6 ? n : 6;
105
106     if (n <= -1)
107         y =  HEIGHT * 0.35 - 30 + 150 * n;
108     else if (n == 0)
109         y = HEIGHT * 0.35;
110     else
111         y = HEIGHT * 0.35 + 30 + 150 * n;
112
113     return y;
114 }
115
116 int SongSelectCover::CalcAlpha() {
117     return GetNow() ? 180 : 255;
118 }
119
120 int SongSelectCover::CalcAlphaWhite() {
121     return GetNow() ? 0 : 255;
```

```
122 }
123
124 double SongSelectCover::CalcEx() {
125     return GetNow() ? 0.7 : 1.0;
126 }
```

SongSelectMain.cpp

```cpp
#include "SongSelectMain.h"

SongSelect::SongSelect(Font *font, Touch *touch, Songs *songs, User *user) {
    songSelectTitle = new SongSelectTitle(font); // 曲選択画面タイトル初期化
    songSelectButton = new SongSelectButton(font, touch);
    songInformation = new SongInformation(font, songs, touch, user); // 選択中の曲初期化
    songSelectPop = new SongSelectPop(font, touch);
    modeSelectButton = new ModeSelectButton(font, touch); // モード選択ボタン初期化
    throughOptionButton = new ThroughOptionButton(font, songs, touch); // 通し練習オプションボタン初期化
    partOptionButton = new PartOptionButton(font, songs, touch); // 部分練習オプションボタン初期化
}

// 曲選択画面ロード
void SongSelect::ContentLoad() {
    songInformation->Load(); // カバー画像ロード
    songSelectTitle->Load(); // タイトルロード
    songSelectPop->Load();   // 終了用ポップアップロード
    songSelectButton->Load(); // 曲選択ボタンロード
    modeSelectButton->Load(); // モード選択ボタンロード
    throughOptionButton->Load(); // 通し練習オプション画面ボタンロード
    partOptionButton->Load(); // 部分練習オプション画面ボタンロード
}

void SongSelect::SetScene(const int scene) {
    this->scene = scene;
}

// 曲選択画面場面切り替え
MainScene SongSelect::Switch(const MainScene scene) {
    switch (this->scene)
    {
    case BACK:
        this->scene = songSelectPop->Switch(this->scene);
        break;
    case MAIN:
        this->scene = songSelectButton->Switch(this->scene);
        break;
    case MODE:
        this->scene = modeSelectButton->Switch(this->scene);
        break;
    case OPTION1:
        this->scene = throughOptionButton->Switch(this->scene);
        break;
    case OPTION2:
    case OPTION2_PART:
    case OPTION2_SPEED:
        this->scene = partOptionButton->Switch(this->scene);
        break;
    }

    switch (this->scene)
    {
    case BACK_TOP:
        Delete();
```

```
55          return TOP;
56      case NEXT1:
57          Delete();
58          return THROUGH;
59      case NEXT2:
60          Delete();
61          return PART;
62      default:
63          return SONG_SELECT;
64      }
65  }
66
67  // 曲選択画面計算
68  void SongSelect::ContentUpdate() {
69      if (nowScene == SONG_SELECT) {
70          Load();
71          songInformation->Update(scene);
72          songSelectTitle->Update(scene);
73          songSelectPop->Update(scene);
74          songSelectButton->Update(scene);
75          modeSelectButton->Update(scene);
76          throughOptionButton->Update(scene);
77          partOptionButton->Update(scene);
78      }
79  }
80
81  // 曲選択画面表示
82  void SongSelect::ContentView() {
83      songInformation->View(); // カバー表示
84      songSelectTitle->View(); // タイトル表示
85      songSelectPop->View(); // 終了用ポップアップ表示
86      songSelectButton->View(); // 曲選択ボタン表示
87      modeSelectButton->View(); // モード選択ボタン表示
88      throughOptionButton->View(); // 通し練習オプション画面ボタン表示
89      partOptionButton->View(); // 部分練習オプション画面ボタン表示
90  }
91
92  void SongSelect::ContentDelete() {
93      songInformation->Delete(); // カバー削除
94      songSelectTitle->Delete(); // タイトル削除
95      songSelectPop->Delete();     // 終了用ポップアップ削除
96      songSelectButton->Delete(); // 曲選択ボタン削除
97      modeSelectButton->Delete(); // モード選択ボタン削除
98      throughOptionButton->Delete(); // 通し練習オプション画面ボタン削除
99      partOptionButton->Delete(); // 部分練習オプション画面ボタン削除
100 }
101
102 SongSelect::~SongSelect() {
103     delete songInformation;
104     delete songSelectButton;
105     delete songSelectTitle;
106     delete songSelectPop;
107     delete throughOptionButton;
108     delete partOptionButton;
109 }
```

StartScreen.cpp

```cpp
#include "StartScreen.h"

StartSceen::StartSceen(Font *f, const int startScene, const int playScene) {
    this->startScene = startScene;
    this->playScene = playScene;
    blackBox = new BlackBox();
    myDrawGraph = new MyDrawGraph(WIDTH * 0.5, HEIGHT * 0.45, "img/start.png");
    myDrawGraph->Load();
    wait = new MyDrawText(f, "", WIDTH * 0.5, HEIGHT * 0.3, 1, 40);
    caution = new MyDrawText(f, "本体から2メートル以上離れてください", WIDTH * 0.5, HEIGHT * 0.67, 1, 46, "Blue");
    annotation = new MyDrawTexts(f, "※2メートル以内に入ると\n自動的に曲が一時停止します", WIDTH * 0.5, HEIGHT * 0.75, 1, 36, 20);
}

void StartSceen::Load() {
    wait->ChangeText("準備中…");
    annotation->SetViewFlag(TRUE);
}

void StartSceen::ContentUpdate() {
    if (nowScene == startScene) {
        viewFlag = TRUE;
    }
    else {
        viewFlag = FALSE;
        if (nowScene = playScene) {
            wait->ChangeText("一時停止中");
            annotation->SetViewFlag(FALSE);
        }
    }
}

void StartSceen::ContentView() {
    blackBox->View();
    myDrawGraph->View();
    wait->View();
    caution->View();
    annotation->View();
}

StartSceen::~StartSceen() {
    delete myDrawGraph;
    delete wait;
    delete caution;
    delete annotation;
    delete blackBox;
}
```

ThroughDetail.cpp

```
1  #include "ThroughDetail.h"
2
3  ThroughFinish::ThroughFinish(Font *font, Touch *touch) {
4      blackBox = new BlackBox();
5      button[0] = new CircleButton(font, touch, "おすすめ練
           習", 0, WIDTH * 0.75, "White");
6      button[1] = new CircleButton(font, touch, "もう一
           度", 1, WIDTH * 0.75, "White");
7      button[2] = new CircleButton(font, touch, "部分練
           習", 2, WIDTH * 0.75, "White");
8      button[3] = new CircleButton(font, touch, "曲選択画
           面", 3, WIDTH * 0.75, "White");
9  }
10
11 ThroughResultScene ThroughFinish::Switch(const ThroughResultScene scene) {
12     if (button[0]->GetTouch() == 1)
13         return THROUGH_RESULT_BACK_PART_OPTION;
14     if (button[1]->GetTouch() == 1)
15         return THROUGH_RESULT_BACK_PLAY;
16     if (button[2]->GetTouch() == 1)
17         return THROUGH_RESULT_BACK_PART_OPTION;
18     if (button[3]->GetTouch() == 1)
19         return THROUGH_RESULT_BACK_SONG_SELECT;
20     return scene;
21 }
22
23 void ThroughFinish::ContentUpdate() {
24     if (nowScene == THROUGH_RESULT_FINISH)
25         viewFlag = TRUE;
26     else
27         viewFlag = FALSE;
28 }
29
30 void ThroughFinish::ContentView() {
31     blackBox->View();
32     for (int i = 0; i < 4; i++)
33         button[i]->View();
34 }
35
36 ThroughFinish::~ThroughFinish() {
37     delete blackBox;
38     for (int i = 0; i < 4; i++)
39         delete button[i];
40 }
41
42 ThroughDetailScreen::ThroughDetailScreen(Font *font, Songs *songs, Touch *
       touch, Result *result) {
43     this->songs = songs;
44     this->result = result;
45     title = new DrawTitle(font, "採点結果");
46     timingBar = new TimingBar(font);
47     expressionBar = new ExpressionBar(font);
48     resultComment = new ResultComment(font);
49     resultBody = new ResultBody(font);
50     resultGraph = new ResultGraph(font);
51     button = new CircleButton2(font, touch, "次へ", 4);
52 }
53
54 ThroughResultScene ThroughDetailScreen::Switch(const ThroughResultScene
       scene) {
55     if (button->GetTouch() == 1)
```

```
56          return THROUGH_RESULT_FINISH;
57      return scene;
58  }
59
60  void ThroughDetailScreen::Load() {
61      Song *song = songs->GetSong(songs->GetNowSong());
62      int point[4];
63      int score[100];
64      int max = result->GetScore(score);
65      timingBar->Load(result->GetTiming());
66      expressionBar->Load(result->GetExpression());
67      result->GetPoint(point);
68      resultBody->Load(point);
69      resultComment->Load(result->GetComment());
70      resultGraph->Load(score, max, song);
71  }
72
73  void ThroughDetailScreen::ContentUpdate() {
74      if (nowScene == THROUGH_RESULT_DETAIL || nowScene ==
            THROUGH_RESULT_FINISH)
75          viewFlag = TRUE;
76      else
77          viewFlag = FALSE;
78  }
79
80  void ThroughDetailScreen::ContentView() {
81      title->View();
82      timingBar->View();
83      expressionBar->View();
84      resultComment->View();
85      button->View();
86      resultBody->View();
87      resultGraph->View();
88  }
89
90  void ThroughDetailScreen::Delete() {
91      resultGraph->Delete();
92      resultBody->Delete();
93  }
94
95  ThroughDetailScreen::~ThroughDetailScreen() {
96      delete title;
97      delete timingBar;
98      delete expressionBar;
99      delete resultComment;
100     delete button;
101     delete resultGraph;
102 }
103
104 ThroughDetail::ThroughDetail(Font *font, Songs *songs, Touch *touch,
        Result *result) {
105     throughDetailScreen = new ThroughDetailScreen(font, songs, touch,
            result);
106     throughFinish = new ThroughFinish(font, touch);
107 }
108
109 ThroughResultScene ThroughDetail::Switch(const ThroughResultScene scene) {
110     switch (scene)
111     {
112     case THROUGH_RESULT_DETAIL:
113         return throughDetailScreen->Switch(scene);
114     case THROUGH_RESULT_FINISH:
115         return throughFinish->Switch(scene);
116     }
```

```
117     return scene;
118 }
119
120 void ThroughDetail::Load() {
121     throughDetailScreen->Load();
122     throughFinish->Load();
123 }
124
125 void ThroughDetail::ContentUpdate() {
126     throughFinish->Update(nowScene);
127     throughDetailScreen->Update(nowScene);
128     if (nowScene == THROUGH_RESULT_DETAIL || nowScene ==
        THROUGH_RESULT_FINISH)
129         viewFlag = TRUE;
130     else
131         viewFlag = FALSE;
132 }
133
134 void ThroughDetail::ContentView() {
135     throughDetailScreen->View();
136     throughFinish->View();
137 }
138
139 void ThroughDetail::Delete() {
140     throughDetailScreen->Delete();
141 }
142
143 ThroughDetail::~ThroughDetail() {
144     delete throughDetailScreen;
145     delete throughFinish;
146 }
```

ThroughMain.cpp

```
1  #include "ThroughMain.h"
2
3  ThroughMain::ThroughMain(Font *font, Touch *touch, Songs *songs, Kinect *
       kinect) {
4      throughStart = new ThroughStart(font);
5      throughPlay = new ThroughPlay(font, songs, touch, kinect);
6      throughPause = new ThroughPause(font, songs, touch);
7  }
8
9  void ThroughMain::ContentLoad() {
10     scene = THROUGH_START;
11     throughStart->Load();
12     throughPlay->Load();
13     throughPause->Load();
14 }
15
16 MainScene ThroughMain::Switch(const MainScene scene) {
17     switch (this->scene)
18     {
19     case THROUGH_COUNTDOWN:
20     case THROUGH_PLAY:
21     case THROUGH_START:
22         this->scene = throughPlay->Switch(this->scene);
23     case THROUGH_PAUSE:
24     case THROUGH_SETTING:
25         this->scene = throughPause->Switch(this->scene);
26         break;
27     }
28
29     switch (this->scene)
30     {
31     case THROUGH_NEXT:
32         Delete();
33         return THROUGH_RESULT;
34     case THROUGH_BACK_SONG_SELECT:
35         Delete();
36         return SONG_SELECT;
37     }
38     return THROUGH;
39 }
40
41 void ThroughMain::ContentUpdate() {
42     if (nowScene == THROUGH) {
43         Load();
44         throughStart->Update(scene);
45         throughPlay->Update(scene);
46         throughPause->Update(scene);
47     }
48 }
49
50 void ThroughMain::ContentView() {
51     throughPlay->View();
52     throughStart->View();
53     throughPause->View();
54 }
55
56 void ThroughMain::ContentDelete() {
57     throughStart->Delete();
58     throughPlay->Delete();
59     throughPause->Delete();
60 }
```

```
61
62 ThroughMain::~ThroughMain() {
63     delete throughStart;
64     delete throughPlay;
65     delete throughPause;
66 }
```

ThroughOption.cpp

```
1  #include "ThroughOption.h"
2
3  ThroughOptionButton::ThroughOptionButton(Font *font, Songs *songs, Touch *
       touch) {
4      this->songs = songs;
5      speedOption = new SpeedOption(font, songs, touch);
6      button[0] = new CircleButton(font, touch, "スタート!", 2);
7      button[1] = new CircleButton2(font, touch, "戻る", 4);
8  }
9
10 int ThroughOptionButton::Switch(const int scene) {
11     if (button[0]->GetTouch() == 1)
12         return NEXT1;
13     if (button[1]->GetTouch() == 1)
14         return MODE;
15     return scene;
16 }
17
18 // モード選択ボタン計算
19 void ThroughOptionButton::ContentUpdate() {
20     static int lastScene = TOP;
21     if (nowScene == OPTION1) {
22         viewFlag = TRUE;
23         Song *song = songs->GetSong(songs->GetNowSong());
24         song->danceMovie->SetStartFlame();
25         song->danceMovie->SetEndFlame();
26         if (lastScene == nowScene) {
27             speedOption->Check();
28         }
29     }
30     else {
31         viewFlag = FALSE;
32     }
33     lastScene = nowScene;
34 }
35
36 // オプション画面ボタン表示
37 void ThroughOptionButton::ContentView() {
38     speedOption->View();
39     for(int i = 0; i < 2; i++)
40         button[i]->View();
41 }
42
43 // オプション画面ボタン削除
44 ThroughOptionButton::~ThroughOptionButton() {
45     for(int i = 0; i < 2; i++)
46         delete button[i];
47     delete speedOption;
48 }
```

ThroughPause.cpp

```cpp
#include "ThroughPause.h"

ThroughPauseButton::ThroughPauseButton(Touch *touch) {
    pauseButton = new CircleGraphButton(touch, 0, "img/pause.png");
}

void ThroughPauseButton::Load() {
    pauseButton->Load();
}

int ThroughPauseButton::Switch(const int scene) {
    if (pauseButton->GetTouch() == 1)
        return THROUGH_PAUSE;
    return scene;
}

void ThroughPauseButton::ContentUpdate() {
    switch (nowScene)
    {
    case THROUGH_PLAY:
    case THROUGH_COUNTDOWN:
    case THROUGH_START:
        viewFlag = TRUE;
        break;
    default:
        viewFlag = FALSE;
        break;
    }
}

void ThroughPauseButton::ContentView() {
    pauseButton->View();
}

void ThroughPauseButton::Delete() {
    pauseButton->Release();
}

ThroughPauseButton::~ThroughPauseButton() {
    delete pauseButton;
}

ThroughPauseScreen::ThroughPauseScreen(Font *font, Songs *songs, Touch *
    touch)
    : PauseScreen(font, songs, touch, THROUGH_PAUSE, THROUGH_START,
        THROUGH_BACK_SONG_SELECT, THROUGH_SETTING) {}

int ThroughPauseSetting::Switch(const int scene) {
    if (button->GetTouch() == 1) {
        song->danceMovie->SetSpeed();
        return THROUGH_PAUSE;
    }
    return scene;
}

void ThroughPauseSetting::ContentUpdate() {
    if (nowScene == THROUGH_SETTING) {
        SpeedPop::ContentUpdate();
        viewFlag = TRUE;
    }
    else {
```

```
60          viewFlag = FALSE;
61      }
62  }
63
64  ThroughPause::ThroughPause(Font *font, Songs *songs, Touch *touch) {
65      throughPauseButton = new ThroughPauseButton(touch); // ポーズボタン画
            面
66      throughPauseScreen = new ThroughPauseScreen(font, songs, touch);
67      throughPauseSetting = new ThroughPauseSetting(font, songs, touch);
68      flag = FALSE;
69  }
70
71  void ThroughPause::Load() {
72      throughPauseButton->Load();
73      throughPauseScreen->Load();
74      throughPauseSetting->Load();
75  }
76
77  int ThroughPause::Switch(const int scene) {
78      switch (scene)
79      {
80      case THROUGH_COUNTDOWN:
81      case THROUGH_PLAY:
82      case THROUGH_START:
83          return throughPauseButton->Switch(scene);
84      case THROUGH_PAUSE:
85          return throughPauseScreen->Switch(scene);
86      case THROUGH_SETTING:
87          return throughPauseSetting->Switch(scene);
88      }
89      return scene;
90  }
91
92  void ThroughPause::ContentUpdate() {
93      throughPauseButton->Update(nowScene);
94      throughPauseScreen->Update(nowScene);
95      throughPauseSetting->Update(nowScene);
96
97      switch (nowScene)
98      {
99      case THROUGH_SETTING:
100     case THROUGH_PLAY:
101     case THROUGH_PAUSE:
102     case THROUGH_COUNTDOWN:
103     case THROUGH_START:
104         viewFlag = TRUE;
105         break;
106     default:
107         viewFlag = FALSE;
108         break;
109     }
110 }
111
112 void ThroughPause::ContentView() {
113     throughPauseButton->View();
114     throughPauseScreen->View();
115     throughPauseSetting->View();
116 }
117
118 void ThroughPause::Delete() {
119     throughPauseButton->Delete();
120     throughPauseScreen->Delete();
121     throughPauseSetting->Delete();
122 }
```

```
123
124 ThroughPause::~ThroughPause() {
125     delete throughPauseButton;
126     delete throughPauseScreen;
127     delete throughPauseSetting;
128 }
```

ThroughPlay.cpp

```
1  #include "ThroughPlay.h"
2
3  ThroughPlay::ThroughPlay(Font *font, Songs *songs, Touch *touch, Kinect *
       kinect)
4      : PlayScreen(font , songs, touch, kinect, THROUGH_START,
           THROUGH_COUNTDOWN, THROUGH_PLAY, THROUGH_NEXT) {}
```

ThroughResult.cpp

```cpp
#include "ThroughResult.h"

ThroughResult::ThroughResult(Font *font, Songs *songs, Touch *touch,
    Result *result) {
    this->songs = songs;
    this->result = result;
    title        = new MyDrawTextLine(font, "採点結果", WIDTH * 0.5, HEIGHT
        * 0.15, 1, 60, WIDTH * 0.5, 4);                    // 採点結果
    circle       = new MyDrawCircle(WIDTH * 0.5, HEIGHT * 0.5, WIDTH *
        0.3, 10, "WHITE");           // 縁が白色の円
    pointCircle  = new MyDrawCircleGauge(WIDTH * 0.5, HEIGHT * 0.5, WIDTH
        * 0.3, 0, 6);          // 青色の弧
    pointCircle2 = new MyDrawCircle(0, 0, 16);                         // 弧
        の先の円
    button       = new CircleButton2(font, touch, "次
        へ", 4);                                          // 次へボタン
    text         = new MyDrawText(font, "総合得点", WIDTH * 0.5, HEIGHT *
        0.42, 1, 46, "White");        // 「総合得点」
    point1       = new MyDrawGraph(WIDTH * 0.33, HEIGHT * 0.51, "", 0.23);
                           // 得点(100の位)
    point2       = new MyDrawGraph(WIDTH * 0.42, HEIGHT * 0.51, "", 0.23);
                             // 得点(10の位)
    point3       = new MyDrawGraph(WIDTH * 0.51, HEIGHT * 0.51, "", 0.23);
                             // 得点(1の位)
    unit         = new MyDrawText(font, "点", WIDTH * 0.58, HEIGHT * 0.54,
         0, 46, "White");           // 「点」
    last         = new MyDrawText(font, "前回 --点", WIDTH * 0.5, HEIGHT *
        0.58, 1, 36, "White");   // 前回の点数
}

void ThroughResult::Load() {
    float total = result->GetTotal();
    char buf[256];

    // 円のバー
    pointCircle->ChangeDegree(total);
    pointCircle2->ChangePos(pointCircle->GetEndX() * SIZE_RATE,
        pointCircle->GetEndY() * SIZE_RATE);

    // 得点の100の位の画像を読み込み
    if (total == 100) {
        point1->ChangeFile("img/1.png");
        point1->Load();
        point1->SetViewFlag(TRUE);
        total = 0;
    }
    else {
        point1->SetViewFlag(FALSE);
    }

    // 得点の10の位の画像を読み込み
    sprintf(buf, "img/%d.png", (int)(total / 10));
    point2->ChangeFile(buf);
    point2->Load();

    // 得点の一の位の画像を読み込み
    sprintf(buf, "img/%d.png", (int)total % 10);
    point3->ChangeFile(buf);
    point3->Load();

```

```
48      song = songs->GetSong(songs->GetNowSong()); // 現在選択中の曲を取得
49      song->coverGraph->Load(); // 曲カバー画像をロード
50      song->coverGraph->ChangePos(WIDTH * 0.3, HEIGHT * 0.26); // カバー画像
            の表示位置変更
51      song->drawSongTitle->ChangePos(WIDTH * 0.6, HEIGHT * 0.24); // 曲タイ
            トルの表示位置変更
52
53      // 前回の点数
54      int *history[2] = { new int(), new int() };
55      song->songHistory->Get(history);
56      if(*history[0] == -1)
57          sprintf(buf, "前回 --点");
58      else
59          sprintf(buf, "前回 %d点", *history[0]);
60      last->ChangeText(buf);
61  }
62
63  ThroughResultScene ThroughResult::Switch(const ThroughResultScene scene) {
64      if (button->GetTouch() == 1) // ボタンが押されたら
65          return THROUGH_RESULT_DETAIL; // 詳細ページに飛ぶ
66      return scene;
67  }
68
69  void ThroughResult::ContentUpdate() {
70      if (nowScene == THROUGH_RESULT_TOP) // シーンが
            THROUGH_RESULT_TOPだったら
71          viewFlag = TRUE; // 画面を表示
72      else // それ以外は
73          viewFlag = FALSE; // 画面を非表示
74  }
75
76  void ThroughResult::ContentView() {
77      title->View();                  // タイトル表示
78      song->coverGraph->View();       // 曲カバー画像表示
79      song->drawSongTitle->View();    // 曲タイトル表示
80      circle->View();                 // 白色の円表示
81      pointCircle->View();            // 青色の弧を表示
82      pointCircle2->View();           // 弧の先の円を表示
83      text->View();                   // 「総合得点」表示
84      point1->View();                 // 得点(10の位)表示
85      point2->View();                 // 得点(1の位)表示
86      point3->View();                 // 得点(1の位)表示
87      unit->View();                   // 「点」を表示
88      last->View();                   // 前回の得点を表示
89      button->View();                 // ボタンを表示
90  }
91
92  ThroughResult::~ThroughResult() {
93      delete title;                   // タイトル削除
94      delete circle;                  // 曲カバー画像削除
95      delete pointCircle;             // 青色の弧削除
96      delete pointCircle2;            // 弧の先の円削除
97      delete button;                  // ボタンを削除
98      delete text;                    // 「総合得点」削除
99      delete point1;                  // 得点(10の位)削除
100     delete point2;                  // 得点(1の位)削除
101     delete point3;                  // 得点(1の位)削除
102     delete unit;                    // 「点」を削除
103     delete last;                    // 前回の得点削除
104 }
```

ThroughResultMain.cpp

```cpp
#include "ThroughResultMain.h"

ThroughResultMain::ThroughResultMain(Font *font, Touch *touch, Songs *songs, User *user) {
    result = new Result(songs, user);
    throughResult = new ThroughResult(font, songs, touch, result);
    throughDetail = new ThroughDetail(font, songs, touch, result);
}

void ThroughResultMain::ContentLoad() {
    scene = THROUGH_RESULT_TOP;
    result->Calc();
    result->Send(); // 送信
    throughResult->Load();
    throughDetail->Load();
}

MainScene ThroughResultMain::Switch(const MainScene scene) {
    switch (this->scene)
    {
    case THROUGH_RESULT_TOP:
        this->scene = throughResult->Switch(this->scene);
        break;
    case THROUGH_RESULT_DETAIL:
    case THROUGH_RESULT_FINISH:
        this->scene = throughDetail->Switch(this->scene);
        break;
    }
    if (this->scene == THROUGH_RESULT_BACK_PLAY) {
        Delete();
        return THROUGH;
    }
    if (this->scene == THROUGH_RESULT_BACK_SONG_SELECT) {
        Delete();
        return SONG_SELECT;
    }
    if (this->scene == THROUGH_RESULT_BACK_PART_OPTION) {
        Delete();
        return PART_OPTION;
    }
    return THROUGH_RESULT;
}

void ThroughResultMain::ContentUpdate() {
    if (nowScene == THROUGH_RESULT) {
        Load();
        throughResult->Update(scene);
        throughDetail->Update(scene);
    }
}

void ThroughResultMain::ContentView() {
    throughResult->View();
    throughDetail->View();
}

void ThroughResultMain::ContentDelete() {
    throughResult->Delete();
    throughDetail->Delete();
}

```

```
61  ThroughResultMain::~ThroughResultMain() {
62      delete throughResult;
63      delete throughDetail;
64      delete result;
65  }
```

ThroughResultObject.cpp

```cpp
#include "ThroughResultObject.h"

ScoreBar::ScoreBar(Font *font, const float y, const char *title, const char *para1, const char *para2)
 : Draw(WIDTH * 0.475, y) {
    const float height = 110;
    this->title = new MyDrawTextLine(font, title, GetX(), GetY(), 0, 24, WIDTH * 0.3, 2);
    mark = new MyDrawGraph(0, GetY() + height - 50, "img/mark.png");
    score = new MyDrawText(font, "", 0, GetY() + height - 55, 1, 30);
    para[0] = new MyDrawText(font, para1, GetX() - 100, GetY() + height, 2, 20);
    para[1] = new MyDrawText(font, para2, GetX() + 100, GetY() + height, 0, 20);
    for (int i = 0; i < 8; i++)
        box[i] = new MyDrawBox(GetX() - 77 + i * 22, GetY() + height, 20, 40);
}

void ScoreBar::Load(const int p) {
    char point[10];
    sprintf_s(point, sizeof(point), "%d", p);
    mark->Load();
    const float x = GetX() - 88 + p * 22;
    mark->ChangePos(x, mark->GetY());
    score->ChangePos(x, score->GetY());
    score->ChangeText(point);
    for (int i = 0; i < 8; i++) {
        if (i < p)
            box[i]->SetAlpha();
        else
            box[i]->SetAlpha(100);
    }
}

void ScoreBar::ContentView() {
    title->View();
    mark->View();
    score->View();
    for (int i = 0; i < 2; i++)
        para[i]->View();
    for (int i = 0; i < 8; i++)
        box[i]->View();
}

ScoreBar::~ScoreBar() {
    delete title;
    delete mark;
    delete score;
    for (int i = 0; i < 2; i++)
        delete para[i];
    for (int i = 0; i < 8; i++)
        delete box[i];
}

TimingBar::TimingBar(Font *font) : ScoreBar(font, HEIGHT * 0.44, "タイミング", "slow", "early") {}

ExpressionBar::ExpressionBar(Font *font) : ScoreBar(font, HEIGHT * 0.54, "表
情", "bad", "good") {}
```

```
54
55  ResultComment::ResultComment(Font *font)
56      : Draw(WIDTH * 0.6, HEIGHT * 0.64) {
57      title = new MyDrawTextLine(font, "コメン
            ト", GetX(), GetY(), 0, 24, WIDTH * 0.55, 2);
58      comment = new MyDrawTexts(font, "", GetX(), GetY() + 66, 1, 20, 16);
59  }
60
61  void ResultComment::Load(const char *str) {
62      comment->ChangeText(str);
63  }
64
65  void ResultComment::ContentView() {
66      title->View();
67      comment->View();
68  }
69
70  ResultComment::~ResultComment() {
71      delete title;
72      delete comment;
73  }
74
75  ResultBody::ResultBody(Font *font)
76      : Draw(WIDTH * 0.8, HEIGHT * 0.53) {
77      body = new MyDrawGraph(GetX(), GetY(), "img/man.png");
78      part[0] = new MyDrawText(font, "左
            手", GetX() - 106, GetY() - 68, 1, 20);
79      part[1] = new MyDrawText(font, "右
            手", GetX() + 140, GetY() - 55, 1, 20);
80      part[2] = new MyDrawText(font, "左
            足", GetX() - 100, GetY() + 68, 1, 20);
81      part[3] = new MyDrawText(font, "右
            足", GetX() + 122, GetY() + 55, 1, 20);
82      point[0] = new MyDrawText(font, "", GetX() - 147, GetY() - 70, 1, 30,
            "Yellow");
83      point[1] = new MyDrawText(font, "", GetX() + 99, GetY() - 57, 1, 30, "
            Yellow");
84      point[2] = new MyDrawText(font, "", GetX() - 141, GetY() + 66, 1, 30,
            "Yellow");
85      point[3] = new MyDrawText(font, "", GetX() + 81, GetY() + 53, 1, 30, "
            Yellow");
86  }
87
88  void ResultBody::Load(const int point[4]) {
89      body->Load();
90      for (int i = 0; i < 4; i++) {
91          switch (point[i]) {
92          case 1:
93              this->point[i]->ChangeText("A");
94              break;
95          case 2:
96              this->point[i]->ChangeText("B");
97              break;
98          case 3:
99              this->point[i]->ChangeText("C");
100             break;
101         }
102     }
103 }
104
105 void ResultBody::ContentView() {
106     body->View();
107     for (int i = 0; i < 4; i++) {
108         part[i]->View();
```

```
109             point[i]->View();
110         }
111     }
112
113     void ResultBody::Delete() {
114         body->Release();
115     }
116
117     ResultBody::~ResultBody() {
118         delete body;
119         for (int i = 0; i < 4; i++) {
120             delete part[i];
121             delete point[i];
122
123         }
124     }
125
126     ResultGraph::ResultGraph(Font *font)
127         : Draw(WIDTH * 0.65, HEIGHT * 0.31) {
128         this->font = font;
129         frame[0] = new MyDrawLine(GetX() - w / 2, GetY() - h / 2, GetX() - w /
                2, GetY() + h / 2, 6, "White");
130         frame[1] = new MyDrawLine(GetX() - w / 2, GetY() + h / 2, GetX() + w /
                2, GetY() + h / 2, 6, "White");
131         scale = new MyDrawTexts(font, "100\n・\n・\n・\n・\n50\n・\n・\n・\n・
                \n0", GetX() - w / 2 - WIDTH * 0.025, GetY(), 1, 20, 4);
132     }
133
134     void ResultGraph::Load(const int *point, const int num, Song *song) {
135         pointMax = num;
136         partMax = song->GetPartNum();
137         for (int i = 0; i < num; i++) {
138             float x1 = GetX() - w / 2 + (float)i / (num - 1) * w;
139             float y1 = GetY() + h / 2 - (float)point[i] / 100 * h;
140             if (i > 0) {
141                 float x2 = GetX() - w / 2 + (float)(i - 1) / (num - 1) * w;
142                 float y2 = GetY() + h / 2 - (float)point[i - 1] / 100 * h;
143                 line[i - 1] = new MyDrawLine(x1, y1, x2, y2, 3);
144             }
145             dot[i] = new MyDrawCircle(x1, y1, 10, "Yellow");
146             if (point[i] > 80)
147                 dot[i]->SetViewFlag(TRUE);
148             else
149                 dot[i]->SetViewFlag(FALSE);
150         }
151
152         for (int i = 0; i < partMax; i++) {
153             SongPart *songPart = song->GetPart(i);
154             float x = GetX() - w / 2 + w * (float)songPart->GetFlame() / song
                    ->danceMovie->GetEndFlame();
155             part[i] = new MyDrawTextV(font, songPart->GetName(),  x, GetY() +
                    HEIGHT * 0.075, 2, 16);
156         }
157     }
158
159     void ResultGraph::ContentView() {
160         for (int i = 0; i < partMax; i++)
161             part[i]->View();
162         for (int i = 0; i < pointMax - 1; i++)
163             line[i]->View();
164         for(int i = 0; i < pointMax; i++)
165             dot[i]->View();
166
167         for (int i = 0; i < 2; i++)
```

```
168             frame[i]->View();
169         scale->View();
170     }
171
172     void ResultGraph::Delete() {
173         for (int i = 0; i < pointMax; i++) {
174             delete dot[i];
175             if (i > 0)
176                 delete line[i - 1];
177         }
178         for (int i = 0; i < partMax; i++)
179             delete part[i];
180     }
181
182     ResultGraph::~ResultGraph() {
183         for (int i = 0; i < 2; i++)
184             delete frame[i];
185         scale->View();
186     }
```

ThroughStart.cpp

```
1  #include "ThroughStart.h"
2
3  ThroughStart::ThroughStart(Font *f)
4      : StartSceen(f, THROUGH_START, THROUGH_PLAY){}
```

Top.cpp

```cpp
#include "Top.h"

// トップロゴ
TopLogo::TopLogo(const float y)
    : MyDrawGraph(WIDTH * 0.5, y, "img/logo.png") {
}

// NFCタッチメッセージコンストラクタ
TopTouchMessage::TopTouchMessage(Font *font, const float y)
    : MyDrawText(font, "-カードをタッチしてください-", WIDTH * 0.5, y, 1, 46) {
    Init(); // 初期化
}

// NFCタッチメッセージ初期化
void TopTouchMessage::Init() {
    t = 0;
}

// NFCタッチメッセージ計算
void TopTouchMessage::Update() {
    if (t > 180)
        SetAlpha(0);
    else if (t > 120)
        SetAlpha((180 - t) * 255 / 60);
    else if (t > 60)
        SetAlpha(255);
    else
        SetAlpha(t * 255 / 60);
    t++;
    t %= 240;
}

// NFCタッチメッセージ表示
void TopTouchMessage::View() {
    MyDrawText::View(); // 文字表示
}

// NFCタッチボタンコンストラクタ
TopTouchButton::TopTouchButton(Font *font)
    : Pos(WIDTH, NFC_POS) {
    float r = WIDTH / 12;
    text = new MyDrawTexts(font, "ここに\nタッチ！", GetX() - r, GetY(), 2, 40, 20);
    circle = new MyDrawCircle(GetX(), GetY(), r);
}

// NFCタッチボタン表示
void TopTouchButton::View() {
    text->View(); // 円表示
    circle->View(); // テキスト表示
}

TopTouchButton::~TopTouchButton() {
    delete text;
    delete circle;
}
```

TopMain.cpp

```cpp
#include "TopMain.h"

// トップ画面コンストラクタ
Top::Top(Font *font, User *user) {
    this->user = user;
    f = font;
    topLogo = new TopLogo(HEIGHT / 3); // ロゴ初期化
    topTouchMessage = new TopTouchMessage(f, HEIGHT * 0.42); // NFCタッチメッセージ初期化
    topTouchButton = new TopTouchButton(f); // NFCタッチボタン初期化
}

// トップ画面初期化
void Top::ContentLoad() {
    StopMusic();
    topLogo->Load();
    topTouchMessage->Init();
    nfc.Init();
}

// 場面の切り替え
MainScene Top::Switch(const MainScene scene) {
    char* id = nfc.GetId();
    if (id[0] != '\0') {
        user->SetUserId(id);
        // printfDx("id:%s", id);
        nfc.reset_calledCont();
        Delete();
        return SONG_SELECT;
    }
    return TOP;
}

// トップ画面計算
void Top::ContentUpdate() {
    if (nowScene == TOP) {
        Load();
        topTouchMessage->Update(); // NFCタッチメッセージ計算
    }
}

// トップ画面表示
void Top::ContentView() {
    topLogo->View(); // ロゴ表示
    topTouchMessage->View(); // NFCタッチメッセージ表示
    topTouchButton->View(); // NFCタッチボタン表示
}

void Top::ContentDelete() {
    topLogo->Release();
}

Top::~Top() {
    delete topLogo;
    delete topTouchButton;
    delete topTouchMessage;
}
```

Touch.cpp

```cpp
#include "Touch.h"

// 確認
void Touch::Check() {
    int k = KEY_INPUT_1;
    for (int i = 0; i < 5; i++) {
        if (CheckHitKey(k++))
            key[i]++;
        else
            key[i] = 0;
    }
}

// 取得
int Touch::Get(int num) {
    return key[num];
}

boolean Touch::Input(int num, int wait, int duration) {
    int key = Get(num);
    return key == 1 || wait <= key && !((key - wait) % duration);
}

boolean Touch::Input2(int num, int wait1, int duration1, int wait2, int duration2) {
    int key = Get(num);
    return key == 1
        || wait1 <= key && key < wait2 && !((key - wait1) % duration1)
        || wait2 + (wait2 - wait1) % duration1 <= key && !((key - wait2 - (wait2 - wait1) % duration1) % duration2);
}
```

User.cpp

```cpp
#include "User.h"

// ユーザーIDを格納
void User::SetUserId(const char *userId) {
    strcpy_s(this->userId, sizeof(this->userId), userId);
}

// ユーザーIDを収得
char *User::GetUserId() {
    return userId;
}
```

1.2 ヘッダファイル

Animation.h

```
1  #ifndef __ANIMATION_H_INCLUDED__
2  #define __ANIMATION_H_INCLUDED__
3
4  #include "Animation.h"
5  #define _USE_MATH_DEFINES  // 円周率M_PIを使うため
6  #include <math.h>
7  #include "DxLib.h"
8
9  typedef unsigned long MyTime;
10
11 class Animation {
12 public:
13     MyTime GetTime();
14     void Reset();
15     void SetDuration(MyTime);
16     enum Easing {
17         LINER,          // 線形
18         EaseInOut_SINE, // 正弦波(遅早遅)
19         EaseOut_SINE,   // 正弦波(早遅)
20         EaseIn_SINE,    // 正弦波(遅早)
21         EaseInOut_QUAD, // 2次式
22         LinerInEaseOut_QUAD,    // 1次=>2次
23         EaseInLinerOut_QUAD,    // 2次=>1次
24         EaseOutBack_QUAD, // 二次式(ちょっとはみ出て戻る)
25     };
26 protected:
27     double UpdateRate(Easing);
28     void SetTime(MyTime);
29 private:
30     MyTime t = 0;       // アニメーションの現在時刻
31     MyTime duration;    // アニメーション動作時間
32 //  int ease = LINER;   // アニメーション種類
33 };
34
35
36 #endif
```

Bezier.h

```
1  #ifndef __BEZIER_H_INCLUDED__
2  #define __BEZIER_H_INCLUDED__
3
4  #include <math.h>
5
6  class Bezier {
7  public:
8      Bezier(const double x1, const double y1, const double x2, const double
           y2);
9      double Calc(const double x); // 計算
10 private:
11     double x1, x2, y1, y2;
12 };
13
14 #endif
```

Button.h

```
1  #ifndef __BUTTON_H_INCLUDED__
2  #define __BUTTON_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "DrawText.h"
6  #include "DrawObject.h"
7  #include "DrawGraph.h"
8  #include "Touch.h"
9
10 #define BUTTON_POS HEIGHT * 0.5
11 #define BUTTON_INTERVAL HEIGHT * 0.05
12
13 // ボタン関係
14 class Button : public Draw {
15 public:
16     Button(const int num, Touch *touch);
17     int GetTouch(); // そのボタンが押されているフレーム数を取得
18 private:
19     virtual void ContentView() = 0; // 表示用関数
20     Touch *touch; // タッチ
21     int num; // ボタン番号
22 };
23
24 // 三角形のボタン
25 class TriangleButton : public Button {
26 public:
27     TriangleButton(Font *font, Touch *touch, const char *str, const int
        direction, const int num, char *colorName = "Blue");
28     void ContentView();
29     ~TriangleButton();
30 private:
31     MyDrawTriangle2 *myDrawTriangle2;
32     MyDrawText *text; // ボタンの文字
33 };
34
35 // 説明文付き三角形のボタン
36 class TriangleButton2 : public Button {
37 public:
38     TriangleButton2(Font *font, Touch *touch, const char *title, const
        char *str, const int direction, const int num, const float x,
        const char *colorName);
39     void ContentView();
40     ~TriangleButton2();
41 private:
42     MyDrawText *text; // ボタンの文字
43     MyDrawTexts *descriptionText;
44     MyDrawTriangle2 *myDrawTriangle2;
45     MyDrawBox *myDrawBox;
46 };
47
48 // 円のボタン
49 class CircleButton : public Button {
50 public:
51     CircleButton(Font *font, Touch *touch, const char *str, const int num,
         char *colorName = "Blue"); // 文字右寄せボタン
52     CircleButton(Font *font, Touch *touch, const char *str, const int num,
         const float x, char *colorName = "Blue"); // 文字中央寄せボタン
53     void ContentView();
54     ~CircleButton();
55 private:
56     MyDrawText *text; // ボタンの文字
```

```
57      MyDrawCircle *myDrawCircle;
58  };
59
60  // 文字が丸の中にあるボタン
61  class CircleButton2 : public Button {
62  public:
63      CircleButton2(Font *font, Touch *touch, const char *str, const int num
            , char *colorName = "Blue");
64      void ContentView();
65      ~CircleButton2();
66  private:
67      MyDrawText *text; // ボタンの文字
68      MyDrawCircle *myDrawCircle;
69  };
70
71  // 画像付きのボタン
72  class CircleGraphButton : public Button {
73  public:
74      CircleGraphButton(Touch *touch, const int num, const char *fileName);
75      void ContentView();
76      void Load();
77      void Release();
78      ~CircleGraphButton();
79  private:
80      MyDrawCircle *myDrawCircle;
81      MyDrawGraph *myDrawGraph;
82  };
83
84  // 画像、テキスト付きのボタン
85  class CircleGraphTextButton : public Button {
86  public:
87      CircleGraphTextButton(Font *font, Touch *touch, const char *str, const
            int num, const char *fileName);
88      void Load();
89      void ContentView();
90      void Release();
91      ~CircleGraphTextButton();
92  private:
93      MyDrawCircle *myDrawCircle;
94      MyDrawGraph *myDrawGraph;
95      MyDrawText *text;
96  };
97
98  #endif
```

CommonText.h

```
1  #ifndef __COMMONTEXT_H_INCLUDED__
2  #define __COMMONTEXT_H_INCLUDED__
3
4  #include "Main.h"
5  #include "DrawText.h"
6
7  class DrawTitle : public MyDrawTextLine {
8  public:
9      DrawTitle(Font *font, const char *str);
10 };
11
12 class DrawSubtitle : public MyDrawText {
13 public:
14     DrawSubtitle(Font *font, const char *str);
15 };
16
17 #endif
```

Draw.h

```
1  #ifndef __DRAW_H_INCLUDED__
2  #define __DRAW_H_INCLUDED__
3
4  #include <math.h>
5  #include "DxLib.h"
6  #include "Main.h"
7  #include "Animation.h"
8
9  // 色関係
10 class Color {
11 public:
12     Color(const char *color);
13     void ChangeColor(const char *color);
14 protected:
15     int Get();
16 private:
17     int c;
18 };
19
20 // 表示位置用クラス
21 class Pos : public Animation {
22 public:
23     void ChangePos(const float x, const float y); // 座標変更
24     void SetPosAnimation(float target_x, float target_y, Easing ease =
           LINER);  // Jaity
25     void Update();  // アニメーション更新
26     float GetX(); // x座標取得
27     float GetY(); // y座標取得
28 protected:
29     Pos();
30     Pos(const float x, const float y); // 初期化
31     float x, y;
32 private:
33     float target_x, target_y;   // アニメーション時の目標座標
34     float default_x, default_y; // アニメーション開始時の座標
35     Easing ease_pos;
36 };
37
38 // 描画用クラス
39 class Draw : public Pos {
40 public:
41     Draw();
42     Draw(const float x, const float y);
43     void View();
44     void SetAlpha(const int alpha = 255); // 透明度指定
45     int GetAlpha();
46     void SetAlphaAnimation(int alpha = 255, Easing ease = LINER);
47     void Update();  // アニメーション更新
48     void SetViewFlag(const boolean viewFlag);
49 private:
50     virtual void ContentView() = 0; // 表示メソッド
51     int alpha = 255; // 透明度
52     int target_alpha, default_alpha;
53     Easing ease_alpha;
54     boolean viewFlag = TRUE;
55 };
56
57 // 描画用クラス（位置指定あり）
58 class Draw2 : public Draw {
59 public:
60     Draw2(const int position);
```

```
61      Draw2(const float x, const float y, const int pos);
62      void ChangePos();
63      void ChangePos(const float x, const float y);
64      float GetX(); // x座標取得
65      float GetY(); // y座標取得
66      virtual float GetWidth() = 0;
67      virtual float GetHeight() = 0;
68  protected:
69      int p; // ポジション情報
70  private :
71      float xx, yy; // 座標
72  };
73
74  #endif
```

DrawGraph.h

```
1  #ifndef __DRAWGRAPH_H_INCLUDED__
2  #define __DRAWGRAPH_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "Draw.h"
6  #include <string>
7
8  // 画像関係
9  class MyDrawGraph : public Draw{
10 public:
11     MyDrawGraph(const char *fileName); // 初期化
12     MyDrawGraph(const float x, const float y, const char *fileName, const
           double ExRate = 1.0); // 初期化
13     void ContentView(); // 表示
14     void Load();
15     void ChangeEx(const double ExRate); // 倍率変更
16     double GetEx();
17     void ChangeFile(const char *fileName);
18     void SetExAnimation(double target_ex, Easing ease = LINER);
19     void Update();
20     void Release();
21 protected:
22     std::string fileName; // ファイル名
23     int handle; // 画像用ハンドル
24     double ex; // 表示倍率
25     double target_ex, default_ex;
26     Easing ease_ex;
27 };
28
29 // 動画関係
30 class MyDrawMovie : public MyDrawGraph {
31 public:
32     MyDrawMovie(const char *filename); // 初期化
33     MyDrawMovie(const float x, const float y, const char *filename, const
           double ExRate); // 初期化
34     void ContentView(); // 表示
35     void Seek(const int flame = -1); // 指定したフレームに移動
36     void Start(); // 再生
37     void Stop(); // 停止
38     void ChangeSpeed(double speed); // スピード変更
39     void SetSpeed(); // スピードセット
40     void SetPart(); // 区間セット
41     double GetSpeed(); // スピード取得
42     int GetStartFlame(); // 最初のフレーム数取得
43     int GetEndFlame(); // 最後のフレーム数取得
44     int GetNowFlame(); // 現在のフレーム数取得
45     int GetAllFlame(); // 動画のフレームズ数取得
46     void SetStartFlame(const int flame = 0); // スタートフレーム指定
47     void SetEndFlame(const int flame = -1); // エンドフレーム指定
48     ~MyDrawMovie();
49 private:
50     double speed, sp;
51     int startFlame = 0, endFlame = -1, sf, ef;
52 };
53
54 #endif
```

DrawObject.h

```
1  #ifndef __DRAWOBJECT_H_INCLUDED__
2  #define __DRAWOBJECT_H_INCLUDED__
3
4  #include "DxLib.h"
5  #define _USE_MATH_DEFINES
6  #include "Main.h"
7  #include "Draw.h"
8
9  // 円関係
10 class MyDrawCircle : public Draw, public Color{
11 public:
12     MyDrawCircle(const float x, const float y, const float radius, const
           char *colorName = "Blue"); // 円初期化（塗りつぶしあり）
13     MyDrawCircle(const float x, const float y, const float radius, const
           float width, const char *colorName = "Blue"); // 円初期化（塗りつ
           ぶしなし）
14     void ContentView();
15 private:
16     float r, w; // 半径、線の太さ
17 };
18
19 // 角度付きの円(線のみ)
20 class MyDrawCircleGauge : public MyDrawCircle , public Pos{
21 public:
22     MyDrawCircleGauge(const float x, const float y, const float radius,
           const double degree, const float width, const char *colorName = "
           Blue");
23     void ContentView();
24     void ChangeDegree(const double degree);
25     float GetEndX();
26     float GetEndY();
27 private:
28     float r; // 半径
29     double rad; // 角度（ラジアン）
30 };
31
32 // 線
33 class MyDrawLine : public Draw, public Color {
34 public:
35     MyDrawLine(const float width, const char *colorName = "Blue");
36     MyDrawLine(const float x1, const float y1, const float x2, const float
            y2, const float width, const char *colorName = "Blue");
37     void ChangePos(const float x1, const float y1, const float x2, const
           float y2);
38 private:
39     void ContentView();
40     float x1, y1, x2, y2, w;
41 };
42
43 class MyDrawTriangle : public Draw, public Color {
44 public:
45     MyDrawTriangle(const char *colorName = "Blue");
46     MyDrawTriangle(const float x1, const float y1, const float x2, const
           float y2, const float x3, const float y3, const char *colorName =
           "Blue");
47     void ChangePos(const float x1, const float y1, const float x2, const
           float y2, const float x3, const float y3);
48 private:
49     void ContentView();
50     float x1, y1, x2, y2, x3, y3;
51 };
```

```
52
53  // 正三角形関係
54  class MyDrawTriangle2 : public MyDrawTriangle {
55  public:
56      MyDrawTriangle2(const float x, const float y, const float width, const
            int direction, const char *colorName = "Blue");
57  private:
58      int d;
59      float w;
60  };
61
62  // 四角形関係
63  class MyDrawBox : public Draw, public Color {
64  public:
65      MyDrawBox(const float x, const float y, const float width, const float
            height, const char *colorName = "White"); // 四角形初期化（塗りつ
           ぶしあり）
66      MyDrawBox(const float x, const float y, const float width, const float
            height, const float line, const char *colorName = "Blue"); // 四
           角形初期化（塗りつぶしなし）
67      void ContentView();
68      void ChangeSize(const float width, const float height);
69  private:
70      float w, h, l; // 幅、高さ、線の太さ
71  };
72
73  class MyDrawBar : public MyDrawBox {
74  public:
75      MyDrawBar(const float x, const float y, const float width, const float
            height, const char *colorName = "White"); // 四角形初期化（塗りつ
           ぶしあり）
76      void ChangeSize(const float width, const float height);
77  private:
78      float x, y; // 座標
79  };
80
81  #endif
```

DrawText.h

```
#ifndef __DRAWTEXT_H_INCLUDED__
#define __DRAWTEXT_H_INCLUDED__

#include <string.h>
#include <string>
#include "DxLib.h"
#include "Font.h"
#include "Draw.h"

// テキスト関係
class MyDrawText : public Color, public Draw2{
public:
    MyDrawText(Font *font, const char *str, const float x, const float y,
        const int pos, const int point, const char *colorName = "White");
        // pos=左寄せ:0 / 中央寄せ:1 / 右寄せ:2
    void ContentView(); // 描画
    void ChangeText(char *str); // テキスト変更
    void ChangeFont(Font *font, const int point); // フォントサイズ変更
    float GetHeight(); // 縦取得
    float GetWidth(); // 幅取得
protected:
    int f, point; // フォント情報、ポジション情報、フォントサイズ
    std::string s; // 文字
};

// 縦書きテキスト
class MyDrawTextV : public MyDrawText {
public:
    MyDrawTextV(Font *font, const char *str, const float x, const float y,
         const int pos, const int point, const char *colorName = "White");
         // pos=左寄せ:0 / 中央寄せ:1 / 右寄せ:2
    void ContentView();
private:
    float RotCenterX;
};

// 複数行のテキスト
class MyDrawTexts : public Color, public Draw {
public:
    MyDrawTexts(Font *font, const char *str, const float x, const float y,
         const int pos, const int point, const float lineInterval, const
        char *colorName = "White");
    void ContentView();
    void ChangePos(const float x, const float y);
    void ChangeText(const char *str); // テキスト変更
    float GetWidth(); // 幅取得
    float GetHeight(); // 高さ取得
    ~MyDrawTexts();
private:
    MyDrawText *myDrawText[256];
    Font *f;
    int l = 0, p, inter, point; // 行数, ポジション情報, 間隔、ポイント数
    char color[100];
};

// アンダーライン付きテキスト
class MyDrawTextLine : public Color, public Draw {
public:
    MyDrawTextLine(Font *font, const char *str, const float x, const float
         y, const int pos, const int point, const float lineLength, const
        float lineWidth, const char *colorName = "White");
```

```
54      void ContentView();
55      void ChangePos(const float x, const float y);
56      void ChangeText(char *str); // テキスト変更
57      ~MyDrawTextLine();
58  private:
59      MyDrawText *myDrawText;
60      int pos;
61      float x1, x2, y1, y2, w, l; // 座標、線の太さ、線の長さ
62  };
63
64  #endif
```

Font.h

```
1  #ifndef __FONT_H_INCLUDED__
2  #define __FONT_H_INCLUDED__
3
4  #include <map>
5  #include "DxLib.h"
6  #include "Main.h"
7
8  #define FONT_NUM 10
9
10 // フォント関係
11 class Font {
12 public:
13     Font(); // ポイント数セット
14     int Get(int point); // フォントID取り出し
15     ~Font();
16 private:
17     std::map <int, int> id;
18     int p[FONT_NUM] = {16, 20, 24, 30, 36, 40, 46, 50 , 60, 100};
19 };
20
21 #endif
```

Grading.h

```
1  #ifndef __GRADING_H_INCLUDED__
2  #define __GRADING_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include <Kinect.h>
6  #include "Bezier.h"
7
8  class FlameGrading {
9  public:
10     FlameGrading(FILE *modelfp);
11     int Mark(float joints[JointType_Count][3], const int userflmae);
12 private:
13     float JointMark(float joints[JointType_Count][3], float model[
           JointType_Count][3], int x, int y); // 2関節間の点数計算
14     float FlameMark(float joints[JointType_Count][3], float model[
           JointType_Count][3]); // 1フレームあたりの点数計算
15     FILE *modelfp;
16     int modelflame, j;
17     float model[JointType_Count][3];
18 };
19
20 class Grading {
21 public:
22     Grading();
23     void Mark(const char *model, const char *user);
24     ~Grading();
25 protected:
26     int total; // 総合得点
27     char comment[1024]; // コメント
28     int point[4]; // 部位別得点
29     int score[100] = {}; // 区間別得点
30     int max; // 区間別得点の点の数
31     int timing; // タイミング
32     int expression; // 表情
33 private:
34     int Adjust(int point); // 点数が0~100の範囲になるように調整
35     Bezier *bezier;
36 };
37
38 #endif
```

Kinect.h

```
1  #ifndef __KINECT_H_INCLUDED__
2  #define __KINECT_H_INCLUDED__
3
4
5  #include "stdafx.h"
6  #include <strsafe.h>
7  #include "DxLib.h"
8  #include "Main.h"
9  #include "KinectBody.h"
10 // #include "KinectColor.h"
11
12 class Kinect
13 {
14 public:
15     Kinect(); // コンストラクタ
16     void Update(); // 更新
17     ~Kinect(); // デストラクタ
18
19     KinectBody *kinectBody; // 骨格情報
20 //  KinectColor *kinectColor; // 色情報
21 private:
22     // Current Kinect
23     IKinectSensor *m_pKinectSensor;
24 };
25
26 #endif
```

KinectBody.h

```
1  #ifndef __KINECTBODY_H_INCLUDED__
2  #define __KINECTBODY_H_INCLUDED__
3
4  #include <Kinect.h>
5  #include "DxLib.h"
6  #include "stdafx.h"
7  #include "Main.h"
8
9  class KinectBody
10 {
11 public:
12     KinectBody(IKinectSensor *m_pKinectSensor); // コンストラクタ
13     void Update(); // 更新
14     boolean CheckDistance(); // 距離を測定
15     void StartSave(const char *fileName);
16     void JointSave(const int flame); // 保存
17     void FinishSave();
18     ~KinectBody(); // デストラクタ
19 private:
20     Joint userJoints[JointType_Count]; // 関節座標情報
21     boolean *userFlag; // ユーザーの状態(TRUE:認識されている / FALSE:認識
           されていない)
22
23     // Body reader
24     IBodyFrameReader*       m_pBodyFrameReader;
25     FILE *fp;
26 };
27
28 #endif
```

MaiKagami.h

```
1  #ifndef __MAIKAGAMI_H_INCLUDED__
2  #define __MAIKAGAMI_H_INCLUDED__
3
4  #include "TopMain.h"
5  #include "SongSelectMain.h"
6  #include "Songs.h"
7  #include "ThroughMain.h"
8  #include "ThroughResultMain.h"
9  #include "PartMain.h"
10 #include "PartResultMain.h"
11 #include "Touch.h"
12 #include "User.h"
13 #include "Kinect.h"
14 #include "DxLib.h"
15
16 class MaiKagami
17 {
18 public:
19     MaiKagami(); // コンストラクタ
20     void Update(); // 計算
21     void View(); // 表示
22     ~MaiKagami(); // デストラクタ
23 private:
24     MainScene scene; // シーン
25     Font *font; // フォント
26     Top *top; // トップ画面
27     SongSelect *songSelect; // 曲選択画面
28     ThroughMain *throughMain; // 通し練習プレイ画面
29     ThroughResultMain *throughResultMain; // 通し練習結果画面
30     PartMain *partMain; // 部分練習プレイ画面
31     PartResultMain *partResultMain; // 部分練習結果画面
32     Songs *songs; // 曲一覧
33     Touch *touch;
34     User *user; // ユーザー情報
35     Kinect *kinect; // キネクト関係
36 };
37
38 #endif
```

Main.h

```
1  #ifndef __MAIN_H_INCLUDED__
2  #define __MAIN_H_INCLUDED__
3
4  #define SIZE_RATE   2
5  #define WIDTH       1080
6  #define HEIGHT      1920
7
8  #define NFC_POS     HEIGHT * 0.85
9  #define NFC_FLAG    FALSE   // NFCカードを読み込むかどうか(TRUE:読み込む/
       FALSE:読み込まない)
10 #define KINECT_FLAG TRUE    // KINECTを使用するかどうか(TRUE:使用する/
       FALSE:使用しない)
11
12 typedef enum {
13     TOP,
14     SONG_SELECT,
15     THROUGH,
16     THROUGH_RESULT,
17     PART,
18     PART_RESULT,
19     THROUGH_OPTION,
20     PART_OPTION
21 } MainScene;
22
23 #endif
```

ModeSelect.h

```
1  #ifndef __MODESELECT_H_INCLUDED__
2  #define __MODESELECT_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "Button.h"
6  #include "SongSelectDefine.h"
7  #include "Font.h"
8  #include "Scene.h"
9  #include "Touch.h"
10
11 class ModeSelectButton : public SubScene{
12 public:
13     ModeSelectButton(Font *font, Touch *touch);
14     int Switch(const int scene);
15     void ContentUpdate();
16     void ContentView();
17     ~ModeSelectButton();
18 private:
19     Button *button[3];
20 };
21
22 #endif
```

Nfc.h

```
1  #ifndef __NFC_H_INCLUDED__
2  #define __NFC_H_INCLUDED__
3
4  #include <WinSock2.h>
5  #include <Ws2tcpip.h>
6  #include "DxLib.h"
7  #pragma comment(lib, "ws2_32.lib")
8
9  #ifndef _RECVSTATUS_
10 #define _RECVSTATUS_
11
12 // 受信状態
13 enum RECVSTATUS
14 {
15     RECV_STILL,      // データが来ていない
16     RECV_SUCCESSED, // 成功
17     RECV_FAILED      // 切断orエラー
18 };
19
20 #endif
21
22 class Nfc
23 {
24 public:
25     // 初期化
26     void Init();
27     // ユーザーIDの取得
28     char* GetId();
29     // ソケットとの接続
30     bool Connect(const char* Ip, u_short Port);
31     // 受信
32     RECVSTATUS Recv(char* pData, int DataSize, int *pRecvSize);
33     // calledContのリセット
34     // 読み込みが完了、または読み込みの開始前にこれを呼び出してください
35     void reset_calledCont();
36 private:
37     // ソケット
38     SOCKET m_DstSocket;
39     // nfcの監視が始まってからGetId()が呼び出された回数
40     int calledCont = 0;
41 };
42
43 #endif
```

PartDefine.h

```
1  #ifndef __PARTDEFINE_H_INCLUDED__
2  #define __PARTDEFINE_H_INCLUDED__
3
4  typedef enum {
5      PART_BACK_SONG_SELECT,
6      PART_START,
7      PART_COUNTDOWN,
8      PART_PLAY,
9      PART_PAUSE,
10     PART_REWIND,
11     PART_SETTING,
12     PART_SETTING_PART,
13     PART_SETTING_SPEED,
14     PART_NEXT
15 } PartScene;
16
17 #endif
```

PartMain.h

```
1  #ifndef __PARTMAIN_H_INCLUDED__
2  #define __PARTMAIN_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "Scene.h"
6  #include "Main.h"
7  #include "Font.h"
8  #include "Songs.h"
9  #include "Touch.h"
10 #include "PartDefine.h"
11 #include "PartPlay.h"
12 #include "PartPause.h"
13
14 class PartMain : public Scene {
15 public:
16     PartMain(Font *font, Touch *touch, Songs *songs, Kinect *kinect);
17     MainScene Switch(const MainScene scene);
18     ~PartMain();
19 private:
20     void ContentLoad();
21     void ContentUpdate();
22     void ContentView();
23     void ContentDelete();
24     PartStart *partStart;
25     PartPlay *partPlay;
26     PartPause *partPause;
27     int scene;
28 };
29
30 #endif
```

PartOption.h

```
1  #ifndef __PARTOPTION_H_INCLUDED__
2  #define __PARTOPTION_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "PartOptionPop.h"
6  #include "SongSelectDefine.h"
7
8  class PartOptionPreview2 : public PartOptionPreview {
9  public:
10     PartOptionPreview2(Font *font, Songs *songs, Touch *touch);
11     int Switch(const int scene);
12     ~PartOptionPreview2();
13 private:
14     void ContentView();
15     Button *button[2];
16 };
17
18 class PartOptionButton : public PartOptionPop {
19 public:
20     PartOptionButton(Font *font, Songs *songs, Touch *touch);
21 };
22
23
24 #endif
```

PartOptionPop.h

```
1  #ifndef __PARTOPTIONPOP_H_INCLUDED__
2  #define __PARTOPTIONPOP_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "Scene.h"
6  #include "Font.h"
7  #include "Songs.h"
8  #include "Song.h"
9  #include "Touch.h"
10 #include "SeetingPop.h"
11 #include "Button.h"
12
13 // スピード変更用ポップアップ
14 class PartOptionSpeedPop : public SpeedPop {
15 public:
16     PartOptionSpeedPop(Font *font, Songs *songs, Touch *touch, const int
       mainScene, const int speedScene);
17     int Switch(const int scene);
18 private:
19     void ContentUpdate();
20     int mainScene, speedScene;
21 };
22
23 // 区間変更用ポップアップ
24 class PartOptionPartPop : public PartPop {
25 public:
26     PartOptionPartPop(Font *font, Songs *songs, Touch *touch, const int
       mainScene, const int partScene);
27     int Switch(const int scene);
28 private:
29     void ContentUpdate();
30     int mainScene, partScene;
31 };
32
33 // オプション画面の動画とボタン
34 class PartOptionPreview : public SubScene {
35 public:
36     PartOptionPreview(Font *font, Songs *songs, Touch *touch, const int
       mainScene, const int partScene, const int speedScene);
37     virtual int Switch(const int scene);
38     ~PartOptionPreview();
39 protected:
40     Songs *songs;
41     void ContentUpdate();
42     void ContentView();
43     Button *button[2];
44     MyDrawText *message, *caption[3], *para[3];
45     int mainScene, partScene, speedScene;
46 };
47
48 class PartOptionPop : public SubScene {
49 public:
50     PartOptionPop(Font *font, Songs *songs, Touch *touch, const int
       mainScene, const int partScene, const int speedScene,
       PartOptionPreview *partOptionPreview);
51     int Switch(const int scene);
52     void Load();
53     void Delete();
54     ~PartOptionPop();
55 private:
56     Songs *songs;
```

```
57      void ContentUpdate();
58      void ContentView();
59      int mainScene, partScene, speedScene;
60      PartOptionSpeedPop *speedPop;
61      PartOptionPartPop *partPop;
62      PartOptionPreview *partOptionPreview;
63  };
64
65
66  #endif
```

PartPause.h

```
1  #ifndef __PARTPAUSE_H_INCLUDED__
2  #define __PARTPAUSE_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "Button.h"
6  #include "Font.h"
7  #include "Songs.h"
8  #include "PartDefine.h"
9  #include "SeetingPop.h"
10 #include "PauseScreen.h"
11 #include "Touch.h"
12 #include "Scene.h"
13 #include "PartOptionPop.h"
14
15 // ポーズボタン画面
16 class PartPauseButton : public SubScene {
17 public:
18     PartPauseButton(Touch *touch, Songs *songs);
19     void Load();
20     int Switch(const int scene);
21     void Delete();
22     ~PartPauseButton();
23 private:
24     void ContentUpdate();
25     void ContentView();
26     Songs *songs;
27     CircleGraphButton *button[2]; // ボタン
28 };
29
30 // ポーズ画面
31 class PartPauseScreen : public PauseScreen {
32 public:
33     PartPauseScreen(Font *font, Songs *songs, Touch *touch);
34 };
35
36 class PartOptionPreview3 : public PartOptionPreview {
37 public:
38     PartOptionPreview3(Font *font, Songs *songs, Touch *touch);
39     int Switch(const int scene);
40     ~PartOptionPreview3();
41 private:
42     BlackBox *blackBox;
43     void ContentView();
44     Button *button;
45 };
46
47 // 設定変更画面
48 class PartPauseSetting : public PartOptionPop {
49 public:
50     PartPauseSetting(Font *font, Songs *songs, Touch *touch);
51 };
52
53 // ポーズ関係
54 class PartPause : public SubScene {
55 public:
56     PartPause(Font *font, Songs *songs, Touch *touch);
57     void Load();
58     int Switch(const int scene);
59     void Delete();
60     ~PartPause();
61 private:
```

```
62      void ContentUpdate();
63      void ContentView();
64      Songs *songs;
65      boolean flag; // ポーズ中かどうかのフラグ
66      PartPauseButton *partPauseButton; // ポーズボタン画面
67      PartPauseScreen *partPauseScreen; // ポーズ画面
68      PartPauseSetting *partPauseSetting; // 設定変更画面
69  };
70
71  #endif
```

PartPlay.h

```
1  #ifndef __PARTPLAY_H_INCLUDED__
2  #define __PARTPLAY_H_INCLUDED__
3
4  #include "StartScreen.h"
5  #include "PlayScreen.h"
6  #include "PartDefine.h"
7
8  class PartStart : public StartSceen {
9  public:
10     PartStart(Font *f);
11 };
12
13 // 部分練習画面
14 class PartPlay : public PlayScreen {
15 public:
16     PartPlay(Font *font, Songs *songs, Touch *touch, Kinect *kinect);
17 };
18
19 #endif
```

PartResult.h

```
1  #ifndef __PARTRESULT_H_INCLUDED__
2  #define __PARTRESULT_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "DrawText.h"
6  #include "DrawObject.h"
7  #include "Button.h"
8  #include "Songs.h"
9  #include "Scene.h"
10 #include "PartResultDefine.h"
11 #include "SeetingPop.h"
12 #include "Touch.h"
13
14 class PartResult : public SubScene {
15 public:
16     PartResult(Font *font, Songs *songs, Touch *touch);
17     void Load();
18     int Switch(const int scene);
19     ~PartResult();
20 private:
21     void ContentUpdate();
22     void ContentView();
23     MyDrawTextLine *title; // 採点結果画面タイトル
24     MyDrawText *part[100], *speed[100], *score[100];
25     MyDrawCircle *circle[100];
26     Button *button; // 次へボタン
27     Song *song;
28     Songs *songs;
29     Font *font;
30     int partMax;
31 };
32
33 class PartFinish : public SubScene {
34 public:
35     PartFinish(Font *font, Touch *touch);
36     int Switch(const int scene);
37     ~PartFinish();
38 private:
39     void ContentUpdate();
40     void ContentView();
41     BlackBox *blackBox;
42     Button *button[4];
43 };
44
45 #endif
```

PartResultDefine.h

```
1  #ifndef __PARTRESULTDEFINE_H_INCLUDED__
2  #define __PARTRESULTDEFINE_H_INCLUDED__
3
4  typedef enum {
5      PART_RESULT_BACK_SONG_SELECT,
6      PART_RESULT_BACK_PLAY,
7      PART_RESULT_TOP,
8      PART_RESULT_FINISH,
9      PART_RESULT_BACK_THROUGH_OPTION,
10     PART_RESULT_BACK_PART_OPTION
11 } PartResultScene;
12
13 #endif
```

PartResultMain.h

```
#ifndef __PARTRESULTMAIN_H_INCLUDED__
#define __PARTRESULTMAIN_H_INCLUDED__

#include "DxLib.h"
#include "Scene.h"
#include "Font.h"
#include "Touch.h"
#include "Songs.h"
#include "PartResultDefine.h"
#include "PartResult.h"

class PartResultMain : public Scene {
public:
    PartResultMain(Font *font, Touch *touch, Songs *songs);
    MainScene Switch(const MainScene scene);
    ~PartResultMain();
private:
    void ContentLoad();
    void ContentUpdate();
    void ContentView();
    void ContentDelete();
    int scene;
    PartResult *partResult;
    PartFinish *partFinish;
};

#endif
```

PauseScreen.h

```
1  #ifndef __PAUSESCREEN_H_INCLUDED__
2  #define __PAUSESCREEN_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "Button.h"
6  #include "Font.h"
7  #include "Songs.h"
8  #include "SeetingPop.h"
9  #include "Touch.h"
10 #include "Scene.h"
11
12 // ポーズ画面
13 class PauseScreen : public SubScene {
14 public:
15     PauseScreen(Font *font, Songs *songs, Touch *touch, const int
           pauseScene, const int startScene, const int songSelectScene, const
            int settingScene);
16     void Load();
17     int Switch(const int scene);
18     void Delete();
19     ~PauseScreen();
20 private:
21     void ContentUpdate();
22     void ContentView();
23     int pauseScene,  startScene, songSelectScene, settingScene;
24     Songs *songs;
25     BlackBox *blackBox; // 背景半透明黒の四角形
26     MyDrawText *title;
27     CircleGraphTextButton *button[4];
28 };
29
30 #endif
```

PlayScreen.h

```
1  #ifndef __PLAYSCREEN_H_INCLUDED__
2  #define __PLAYSCREEN_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "Songs.h"
6  #include "DrawText.h"
7  #include "DrawObject.h"
8  #include "Button.h"
9  #include "SeetingPop.h"
10 #include "Scene.h"
11 #include "Touch.h"
12 #include "Kinect.h"
13 #include "PlayScreenObject.h"
14
15 // 通し練習画面
16 class PlayScreen : public SubScene {
17 public:
18     PlayScreen(Font *font, Songs *songs, Touch *touch, Kinect *kinect,
        const int startScene, const int countDownScene, const int
        playScene, const int finishScene);
19     void Load();
20     int Switch(const int scene);
21     void Delete();
22     ~PlayScreen();
23 private:
24     void ContentUpdate();
25     void ContentView();
26     int startScene, countDownScene, playScene, finishScene;
27     Kinect *kinect;
28     Songs *songs;
29     Song *song;
30     PlayBar *playBar; // 進捗バー
31     CountDown *countDown; // カウントダウン画面
32 };
33
34 #endif
```

PlayScreenObject.h

```
1  #ifndef __PLAYSCREENOBJECT_H_INCLUDED__
2  #define __PLAYSCREENOBJECT_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "DrawObject.h"
6  #include "SeetingPop.h"
7  #include "DrawText.h"
8
9  // 進捗バー
10 class PlayBar {
11 public:
12     PlayBar(Font *font);
13     void Load(Song *song);
14     void Update();
15     void View();
16     ~PlayBar();
17 private:
18     Font *font;
19     Song *song;
20     MyDrawBar *barAll, *barNow;
21     MyDrawCircle *circle[2];
22     MyDrawTextV *part[10];
23 };
24
25 // カウントダウン画面用再生三角形
26 class PlayTriangle : public MyDrawTriangle {
27 public:
28     PlayTriangle(const float x, const float y);
29 };
30
31
32 // カウントダウン画面
33 class CountDown : public    SubScene {
34 public:
35     CountDown(Font *font, const int thisScene, const int playScene);
36     int Switch(const int scene);
37     ~CountDown();
38 private:
39     void ContentUpdate();
40     void ContentView();
41     int count; // カウンタ
42     int thisScene, playScene; //
           CountDown画面のシーン、プレイモードのシーン
43     BlackBox *blackBox;
44     MyDrawText *text;
45     MyDrawCircle *circle;
46     MyDrawCircleGauge *countCircle1;
47     MyDrawCircle *countCircle2;
48     PlayTriangle *playTriangle;
49     const int max = 120;
50 };
51
52 #endif
```

Result.h

```
#ifndef __RESULT_H_INCLUDED__
#define __RESULT_H_INCLUDED__

#include "DxLib.h"
#include "Songs.h"
#include "User.h"
#include "Grading.h"

class Result : public Grading {
public:
    Result(Songs *songs, User *user);
    void Calc(); // 得点計算
    void Send(); // 送信
    float GetTotal(); // 総合得点取得
    void GetPoint(int x[4]); // 部位別得点取得
    char *GetComment(); // コメント取得
    int GetTiming(); // タイミング取得
    int GetExpression(); // 表情取得
    int GetScore(int x[100]); // 区間別得点取得
private:
    Songs *songs;
    User *user;
};

#endif
```

Scene.h

```
1  #ifndef __SCENE_H_INCLUDED__
2  #define __SCENE_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "Main.h"
6
7  // サブ場面定義
8  class SubScene {
9  public:
10     void Update(const int scene); // 更新
11     void View(); // 表示
12     void Load(); // ロード
13     void Delete(); // 削除
14     boolean CheckView(); // 表示中かどうか確認する(TRUE:表示中、
           FALSE:非表示中)
15 protected:
16     int nowScene;
17     boolean viewFlag = FALSE; // 表示用フラグ(TRUE:表示、FALSE:非表示)
18     virtual void ContentView() = 0; // 表示詳細
19     virtual void ContentUpdate() = 0; // 更新詳細
20 };
21
22 // 場面定義
23 class Scene : public SubScene {
24 protected:
25     void Load(); // ロード
26     void Delete(); // 削除
27 private:
28     virtual void ContentLoad() = 0; // ロード詳細
29     virtual void ContentDelete() = 0; // 削除詳細
30     int loadFlag = 0; // ロード確認用(0:未ロード、1:ロード中、2:ロード
           完了)
31 };
32
33 #endif
```

SeetingPop.h

```
#ifndef __SETTINGPOP_H_INCLUDED__
#define __SETTINGPOP_H_INCLUDED__

#include "DxLib.h"
#include "DrawObject.h"
#include "DrawText.h"
#include "Button.h"
#include "Font.h"
#include "Songs.h"
#include "Touch.h"
#include "Scene.h"

// ポップアップ用四角形（黒色半透明全画面）
class BlackBox : public MyDrawBox {
public:
    BlackBox();
private:
};

// スピードオプション表示
class SpeedOption {
public:
    SpeedOption(Font *font, Songs *songs, Touch *touch);
    void Check();
    void View();
    ~SpeedOption();
private:
    Button *button[2];
    MyDrawText *speed[2];
    Songs *songs;
};

// 区間設定オプション表示
class PartOption {
public:
    PartOption(Font *font, Songs *songs, Touch *touch);
    void Init();
    void Check();
    void View();
    ~PartOption();
private:
    Button *button[4];
    MyDrawText *part[4];
    Songs *songs;
    Song *song;
};

// スピードオプションポップアップ
class SpeedPop : public SubScene {
public:
    SpeedPop(Font *font, Songs *songs, Touch *touch);
    void Load();
    ~SpeedPop();
protected:
    void ContentUpdate();
    void ContentView();
    Songs *songs;
    Song *song;
    SpeedOption *speedOption;
    BlackBox *blackBox;
    Button *button;
```

```
62      MyDrawText *text;
63  };
64
65  // 区間設定オプションポップアップ
66  class PartPop : public SubScene {
67  public:
68      PartPop(Font *font, Songs *songs, Touch *touch);
69      void Load();
70      ~PartPop();
71  protected:
72      void ContentUpdate();
73      void ContentView();
74      void Init();
75      Songs *songs;
76      Song *song;
77      PartOption *partOption;
78      BlackBox *blackBox;
79      Button *button;
80      MyDrawText *text;
81  };
82
83
84  #endif
```

Song.h

```
1  #ifndef __SONG_H_INCLUDED__
2  #define __SONG_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "Draw.h"
6  #include "DrawGraph.h"
7  #include "DrawText.h"
8  #include "Font.h"
9
10 // 履歴
11 class SongHistory {
12 public:
13     void Set(const int history[2]); // 履歴セット
14     void Get(int *history[2]); // 履歴取得
15 private:
16     int history[2] = { -1, -1 }; // 曲ID,
17 };
18
19 // 曲名、アーティスト情報
20 class DrawSongTitle : public Pos {
21 public:
22     DrawSongTitle(Font *font, const char *title, const char *artist);
23     void ChangePos(const float x, const float y);
24     void View();
25     ~DrawSongTitle();
26 private:
27     MyDrawTextLine *songTitle; // 曲名表示
28     MyDrawText *songArtist; // アーティスト表示
29 };
30
31 // パート情報
32 class SongPart {
33 public:
34     void Set(const int flame, const char *name);
35     int GetFlame(); // フレーム数取得
36     char *GetName(); // パート名取得
37 private:
38     int flame;
39     char name[256]; // 文字
40 };
41
42 class Song {
43 public:
44     Song(Font *font, const int id, const char *title, const char *artist,
        const char *folder);
45     int GetSongId(); // 曲IDを取得
46     int GetNow(); // 現在の位置IDを取得
47     void SetNow(const int n); // 位置IDをセット
48     void ChangeSpeed(int num); // 動画の再生速度変更
49     void ChangeStart(int num); // 動画の開始位置変更
50     void ChangeEnd(int num); // 動画の終了位置変更
51     int StartPart();
52     int EndPart();
53     void LoadPart(); // パート情報ロード
54     SongPart *GetPart(int num); // パート情報取得
55     int GetPartNum(); // パート数取得
56     char *GetFolder(); // フォルダ取得
57     DrawSongTitle *drawSongTitle; // 曲名、アーティスト表示
58     MyDrawGraph *coverGraph; // カバー画像
59     MyDrawGraph *coverWhite; // カバー画像の背景の白
```

```
60      MyDrawMovie *danceMovie; // 動画
61      SongHistory *songHistory;
62  protected:
63      char music[256], folder[256]; // 音楽ファイル、フォルダ
64  private:
65      int id, *n, *songPartNum, *start, *end; // ID、現在の番号, 曲数、開始
            、終了
66      SongPart *songPart[256];
67  };
68
69  #endif
```

Songs.h

```
#ifndef __SONGS_H_INCLUDED__
#define __SONGS_H_INCLUDED__

#include <Windows.h>
#include <winhttp.h>
#include <wchar.h>

#include "Song.h"
#include "DxLib.h"
#include "DrawGraph.h"
#include "Font.h"

#pragma comment (lib, "winhttp.lib")

class Songs {
public:
    Songs(Font *font);
    int GetSongNum(); // 曲数取得
    Song *GetSong(int x);
    int GetNowSong();
    // 履歴ロード
    // 成功したら0,エラーならば-1を返す
    int LoadHistory(const char *userId);
private:
    Song *song[256];
    int Search(const int songId);
    int n; // 曲数
};

#endif
```

SongSelect.h

```
1  #ifndef __SONGSELECT_H_INCLUDED__
2  #define __SONGSELECT_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "Main.h"
6  #include "DrawObject.h"
7  #include "Drawtext.h"
8  #include "Button.h"
9  #include "Touch.h"
10 #include "Font.h"
11 #include "SeetingPop.h"
12 #include "Scene.h"
13 #include "SongSelectDefine.h"
14
15 // 曲選択画面ボタン関係
16 class SongSelectButton : public SubScene{
17 public:
18     SongSelectButton(Font *font, Touch *touch); // 初期化
19     int Switch(const int scene);
20     void ContentUpdate();
21     void ContentView(); // 表示
22     ~SongSelectButton();
23 private:
24     Button *button[4];
25 };
26
27 // 終了用ポップアップ
28 class SongSelectPop : public SubScene {
29 public:
30     SongSelectPop(Font *font, Touch *touch);
31     int Switch(const int scene);
32     void ContentUpdate();
33     void ContentView();
34     ~SongSelectPop();
35 private:
36     Touch *touch;
37     BlackBox *blackBox;
38     MyDrawText *title;
39     MyDrawText *message;
40     Button *button[2];
41 };
42
43 #endif
```

SongSelectCommon.h

```
1  #ifndef __SONGSELECTCOMMON_H_INCLUDED__
2  #define __SONGSELECTCOMMON_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "DrawGraph.h"
6  #include "CommonText.h"
7  #include "Touch.h"
8  #include "SongSelectCover.h"
9  #include "Songs.h"
10 #include "SongSelectDefine.h"
11 #include "Scene.h"
12 #include "User.h"
13
14 // 曲選択画面タイトル
15 class SongSelectTitle : public SubScene {
16 public:
17     SongSelectTitle(Font *font); // 初期化
18     void ContentUpdate(); // 計算
19     void ContentView(); // 表示
20     ~SongSelectTitle();
21 private:
22     DrawTitle *title;
23     DrawSubtitle *subTitle;
24 };
25
26 // 曲選択画面カバー関係
27 class SongInformation : public SubScene {
28 public:
29     SongInformation(Font *font, Songs *songs, Touch *touch, User *user);
        // 初期化
30     void Load();
31     void ContentView(); // 表示
32     void ContentUpdate();
33     void Delete();
34     ~SongInformation();
35 private:
36     int n, now;
37     User *user;
38     Touch *touch;
39     SongSelectCover *songCover[256];
40     Songs *songs;
41     SongSelectCover *nowSong;
42     MyDrawGraph *grad[2]; // カバー画像
43 //  MyDrawGraph *box; // カバー画像
44     MyDrawBox *myDrawBox;
45     MyDrawText *songLast[2];
46 };
47
48 #endif
```

SongSelectCover.h

```
#ifndef __SONGSELECTCOVER_H_INCLUDED__
#define __SONGSELECTCOVER_H_INCLUDED__

#include <string>
#include "DxLib.h"
#include "Draw.h"
#include "DrawText.h"
#include "DrawGraph.h"
#include "DrawObject.h"
#include "Font.h"
#include "SongSelectDefine.h"
#include "Songs.h"

class SongSelectCover : public Song{
public:
    SongSelectCover(Font *font, Song *song, const int now);
    void Load(int);
    void Release();
    void Update(int, int);
    void Draw(int scene);
private:
    void Change(int num, int max);
    float CalcY();
    int CalcAlpha();
    int CalcAlphaWhite();
    double CalcEx();
    boolean playFlag = 0;
    Font *font;
};

#endif
```

SongSelectDefine.h

```
1  #ifndef __SONGSELECTDEFINE_H_INCLUDED__
2  #define __SONGSELECTDEFINE_H_INCLUDED__
3
4  typedef enum {
5      BACK_TOP,
6      BACK,
7      MAIN,
8      MODE,
9      OPTION1,
10     OPTION2,
11     OPTION2_PART,
12     OPTION2_SPEED,
13     NEXT1,
14     NEXT2
15 } SongSelectScene;
16
17 #endif
```

SongSelectMain.h

```
#ifndef __SONGSELECTMAIN_H_INCLUDED__
#define __SONGSELECTMAIN_H_INCLUDED__

#include "DxLib.h"
#include "ModeSelect.h"
#include "Touch.h"
#include "SongSelectCover.h"
#include "Songs.h"
#include "Font.h"
#include "SongSelectCommon.h"
#include "SongSelect.h"
#include "SongSelectDefine.h"
#include "ThroughOption.h"
#include "PartOption.h"
#include "Scene.h"
#include "User.h"

// 曲選択画面関係
class SongSelect : public Scene{
public:
    SongSelect(Font *font, Touch *touch, Songs *songs, User *user);
    MainScene Switch(const MainScene scene);
    void SetScene(const int scene);
    ~SongSelect();
private:
    SongSelectTitle *songSelectTitle; // 曲選択画面タイトル
    SongInformation *songInformation; // 選択中の曲
    SongSelectButton *songSelectButton; // ボタン関係
    SongSelectPop *songSelectPop; // 終了用ポップアップ
    ModeSelectButton *modeSelectButton; // モード選択ボタン
    ThroughOptionButton *throughOptionButton; // 通し練習オプションボタン
    PartOptionButton *partOptionButton; // 部分練習オプションボタン
    int scene = MAIN;;
    void ContentLoad();
    void ContentUpdate();
    void ContentView();
    void ContentDelete();
};

#endif
```

StartScreen.h

```
1  #ifndef __STARTSCREEN_H_INCLUDED__
2  #define __STARTSCREEN_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "DrawGraph.h"
6  #include "DrawText.h"
7  #include "DrawObject.h"
8  #include "SeetingPop.h"
9  #include "Scene.h"
10
11 class StartSceen : public SubScene {
12 public:
13     StartSceen(Font *f, const int startScene, const int playScene);
14     void Load();
15     ~StartSceen();
16 private:
17     void ContentUpdate();
18     void ContentView();
19     int startScene;
20     int playScene;
21     MyDrawGraph *myDrawGraph;
22     MyDrawText *wait;
23     MyDrawText *caution;
24     MyDrawTexts *annotation;
25     BlackBox *blackBox;
26 };
27
28 #endif
```

stdafx.h

```
1  //
       ------------------------------------------------------------------------------
2  // <copyright file="stdafx.h" company="Microsoft">
3  //     Copyright (c) Microsoft Corporation.  All rights reserved.
4  // </copyright>
5  //
       ------------------------------------------------------------------------------
6
7  // include file for standard system and project includes
8
9  #pragma once
10
11 #ifndef WIN32_LEAN_AND_MEAN
12 #define WIN32_LEAN_AND_MEAN             // Exclude rarely-used stuff from
       Windows headers
13 #endif
14
15 // Windows Header Files
16 #include <windows.h>
17
18 #include <Shlobj.h>
19
20 // Direct2D Header Files
21 #include <d2d1.h>
22
23 // Kinect Header files
24 #include <Kinect.h>
25
26 #pragma comment (lib, "d2d1.lib")
27
28 #ifdef _UNICODE
29 #if defined _M_IX86
30 #pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft
       .Windows.Common-Controls' version='6.0.0.0' processorArchitecture='x86
       ' publicKeyToken='6595b64144ccf1df' language='*'\"")
31 #elif defined _M_X64
32 #pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft
       .Windows.Common-Controls' version='6.0.0.0' processorArchitecture='
       amd64' publicKeyToken='6595b64144ccf1df' language='*'\"")
33 #else
34 #pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft
       .Windows.Common-Controls' version='6.0.0.0' processorArchitecture='*'
       publicKeyToken='6595b64144ccf1df' language='*'\"")
35 #endif
36 #endif
37
38 // Safe release for interfaces
39 template<class Interface>
40 inline void SafeRelease(Interface *& pInterfaceToRelease)
41 {
42     if (pInterfaceToRelease != NULL)
43     {
44         pInterfaceToRelease->Release();
45         pInterfaceToRelease = NULL;
46     }
47 }
```

ThroughDefine.h

```
#ifndef __THROUGHDEFINE_H_INCLUDED__
#define __THROUGHDEFINE_H_INCLUDED__

typedef enum {
    THROUGH_BACK_SONG_SELECT,
    THROUGH_START,
    THROUGH_COUNTDOWN,
    THROUGH_PLAY,
    THROUGH_PAUSE,
    THROUGH_SETTING,
    THROUGH_NEXT
} ThroughScene;

#endif
```

ThroughDetail.h

```
1  #ifndef __THROUGHDETAIL_H_INCLUDED__
2  #define __THROUGHDETAIL_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "CommonText.h"
6  #include "ThroughDefine.h"
7  #include "Button.h"
8  #include "SeetingPop.h"
9  #include "Scene.h"
10 #include "ThroughResultObject.h"
11 #include "ThroughResultDefine.h"
12 #include "Result.h"
13
14 class ThroughFinish : public SubScene {
15 public:
16     ThroughFinish(Font *font, Touch *touch);
17     ThroughResultScene Switch(const ThroughResultScene scene);
18     void ContentUpdate();
19     void ContentView();
20     ~ThroughFinish();
21 private:
22     BlackBox *blackBox;
23     Button *button[4];
24 };
25
26 class ThroughDetailScreen : public SubScene {
27 public:
28     ThroughDetailScreen(Font *font, Songs *songs, Touch *touch, Result *
        result);
29     ThroughResultScene Switch(const ThroughResultScene scene);
30     void Load();
31     void Delete();
32     ~ThroughDetailScreen();
33 private:
34     void ContentUpdate();
35     void ContentView();
36     DrawTitle *title;
37     TimingBar *timingBar;
38     ExpressionBar *expressionBar;
39     ResultComment *resultComment;
40     ResultBody *resultBody;
41     ResultGraph *resultGraph;
42     Button *button;
43     Songs *songs;
44     Result *result;
45 };
46
47 class ThroughDetail : public SubScene {
48 public:
49     ThroughDetail(Font *font, Songs *songs, Touch *touch, Result *result);
50     ThroughResultScene Switch(const ThroughResultScene scene);
51     void Load();
52     void Delete();
53     ~ThroughDetail();
54 private:
55     void ContentUpdate();
56     void ContentView();
57     ThroughDetailScreen *throughDetailScreen;
58     ThroughFinish *throughFinish;
59 };
60
```

```
61  #endif
```

ThroughMain.h

```
1  #ifndef __THROUGHMAIN_H_INCLUDED__
2  #define __THROUGHMAIN_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "Font.h"
6  #include "Songs.h"
7  #include "Main.h"
8  #include "DrawText.h"
9  #include "ThroughStart.h"
10 #include "ThroughDefine.h"
11 #include "ThroughPlay.h"
12 #include "ThroughPause.h"
13 #include "Touch.h"
14 #include "Scene.h"
15
16 class ThroughMain : public Scene{
17 public:
18     ThroughMain(Font *font, Touch *touch, Songs *songs, Kinect *kinect);
19     MainScene Switch(const MainScene scene);
20     ~ThroughMain();
21 private:
22     void ContentLoad();
23     void ContentUpdate();
24     void ContentView();
25     void ContentDelete();
26     ThroughStart *throughStart;
27     ThroughPlay *throughPlay;
28     ThroughPause *throughPause;
29     int scene;
30 };
31
32 #endif
```

ThroughOption.h

```
1  #ifndef __THROUGHOPTION_H_INCLUDED__
2  #define __THROUGHOPTION_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "Button.h"
6  #include "Font.h"
7  #include "Draw.h"
8  #include "Touch.h"
9  #include "SongSelectDefine.h"
10 #include "Songs.h"
11 #include "SeetingPop.h"
12 #include "Scene.h"
13 #include "SongSelectDefine.h"
14
15 class ThroughOptionButton : public SubScene{
16 public:
17     ThroughOptionButton(Font *font, Songs *songs, Touch *touch);
18     int Switch(const int scene);
19     ~ThroughOptionButton();
20 private:
21     Songs *songs;
22     void ContentUpdate();
23     void ContentView();
24     SpeedOption *speedOption;
25     Button *button[2];
26     MyDrawText *speed[2];
27 };
28
29 #endif
```

ThroughPause.h

```
#ifndef __THROUGHPAUSE_H_INCLUDED__
#define __THROUGHPAUSE_H_INCLUDED__

#include "DxLib.h"
#include "Button.h"
#include "Font.h"
#include "Songs.h"
#include "ThroughDefine.h"
#include "SeetingPop.h"
#include "PauseScreen.h"
#include "Touch.h"
#include "Scene.h"

// ポーズボタン画面
class ThroughPauseButton : public SubScene {
public:
    ThroughPauseButton(Touch *touch);
    void Load();
    int Switch(const int scene);
    void Delete();
    ~ThroughPauseButton();
private:
    void ContentUpdate();
    void ContentView();
    CircleGraphButton *pauseButton; // 一時停止用ボタン
};

// ポーズ画面
class ThroughPauseScreen : public PauseScreen {
public:
    ThroughPauseScreen(Font *font, Songs *songs, Touch *touch);
};

// 速度変更画面
class ThroughPauseSetting : public SpeedPop {
public:
    ThroughPauseSetting::ThroughPauseSetting(Font *font, Songs *songs,
        Touch *touch) : SpeedPop(font, songs, touch) {};
    int Switch(const int scene);
    void ContentUpdate();
};

// ポーズ関係
class ThroughPause : public SubScene {
public:
    ThroughPause(Font *font, Songs *songs, Touch *touch);
    void Load();
    int Switch(const int scene);
    void Delete();
    ~ThroughPause();
private:
    void ContentUpdate();
    void ContentView();
    Songs *songs;
    boolean flag; // ポーズ中かどうかのフラグ
    ThroughPauseButton *throughPauseButton; // ポーズボタン画面
    ThroughPauseScreen *throughPauseScreen; // ポーズ画面
    ThroughPauseSetting *throughPauseSetting; // 速度変更画面
};

#endif
```

ThroughPlay.h

```
1  #ifndef __THROUGHPLAY_H_INCLUDED__
2  #define __THROUGHPLAY_H_INCLUDED__
3
4  #include "PlayScreen.h"
5  #include "ThroughDefine.h"
6
7  // 通し練習画面
8  class ThroughPlay : public PlayScreen {
9  public:
10     ThroughPlay(Font *font, Songs *songs, Touch *touch, Kinect *kinect);
11 };
12
13 #endif
```

ThroughResult.h

```
#ifndef __THROUGHRESULT_H_INCLUDED__
#define __THROUGHRESULT_H_INCLUDED__

#include "DxLib.h"
#include "DrawText.h"
#include "DrawObject.h"
#include "Button.h"
#include "Songs.h"
#include "Scene.h"
#include "ThroughDefine.h"
#include "ThroughResultDefine.h"
#include "Touch.h"
#include "Result.h"

class ThroughResult : public SubScene {
public:
    ThroughResult(Font *font, Songs *songs, Touch *touch, Result *result);
    void Load();
    ThroughResultScene Switch(const ThroughResultScene scene);
    ~ThroughResult();
private:
    void ContentUpdate();
    void ContentView();
    MyDrawTextLine      *title;          // 採点結果画面タイトル
    MyDrawCircle        *circle;         // 得点を表示する円(縁白色)
    MyDrawCircleGauge   *pointCircle;    // 得点を示す角度指定の円
    MyDrawCircle        *pointCircle2;   // 得点を示す円
    MyDrawText          *text;           // テキスト(総合得点)
    MyDrawGraph         *point1, *point2, *point3;      // 得点
    MyDrawText          *unit;           // 単位(点)
    MyDrawText          *last;           // 前回の得点
    Button              *button;         // 次へボタン
    Song                *song;
    Songs               *songs;
    Result              *result;         // 結果関係
};

#endif
```

ThroughResultDefine.h

```
1  #ifndef __THROUGHRESULTDEFINE_H_INCLUDED__
2  #define __THROUGHRESULTDEFINE_H_INCLUDED__
3
4  typedef enum {
5      THROUGH_RESULT_BACK_SONG_SELECT,
6      THROUGH_RESULT_BACK_PLAY,
7      THROUGH_RESULT_TOP,
8      THROUGH_RESULT_DETAIL,
9      THROUGH_RESULT_FINISH,
10     THROUGH_RESULT_BACK_PART_OPTION
11 } ThroughResultScene;
12
13 #endif
```

ThroughResultMain.h

```
1  #ifndef __THROUGHRESULTMAIN_H_INCLUDED__
2  #define __THROUGHRESULTMAIN_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "Scene.h"
6  #include "ThroughResult.h"
7  #include "ThroughDetail.h"
8  #include "ThroughResultDefine.h"
9  #include "Result.h"
10
11 class ThroughResultMain : public Scene {
12 public:
13     ThroughResultMain(Font *font, Touch *touch, Songs *songs, User *user);
14     MainScene Switch(const MainScene scene);
15     ~ThroughResultMain();
16 private:
17     void ContentLoad();
18     void ContentUpdate();
19     void ContentView();
20     void ContentDelete();
21     ThroughResult *throughResult;
22     ThroughDetail *throughDetail;
23     Result *result;
24     ThroughResultScene scene;
25 };
26
27 #endif
```

ThroughResultObject.h

```
1  #ifndef __THROUGHRESULTOBJECT_H_INCLUDED__
2  #define __THROUGHRESULTOBJECT_H_INCLUDED__
3
4  #include "Dxlib.h"
5  #include "DrawText.h"
6  #include "DrawObject.h"
7  #include "DrawGraph.h"
8  #include "Song.h"
9
10 // 得点バー
11 class ScoreBar : public Draw {
12 public:
13     ScoreBar(Font *font, const float y, const char *title, const char *
       para1, const char *para2);
14     void Load(const int p);
15     ~ScoreBar();
16 private:
17     void ContentView();
18     MyDrawTextLine *title;
19     MyDrawText *para[2];
20     MyDrawBox *box[8];
21     MyDrawGraph *mark;
22     MyDrawText *score;
23 };
24
25 // タイミング得点バー
26 class TimingBar : public ScoreBar {
27 public:
28     TimingBar(Font *font);
29 };
30
31 // 表情得点バー
32 class ExpressionBar : public ScoreBar {
33 public:
34     ExpressionBar(Font *font);
35 };
36
37 // コメント表示
38 class ResultComment : public Draw {
39 public:
40     ResultComment(Font *font);
41     void Load(const char *str);
42     ~ResultComment();
43 private:
44     void ContentView();
45     MyDrawTextLine *title;
46     MyDrawTexts *comment;
47 };
48
49 // 体のパーツ別採点結果表示
50 class ResultBody : public Draw {
51 public:
52     ResultBody(Font *font);
53     void Load(const int point[4]);
54     void Delete();
55     ~ResultBody();
56 private:
57     void ContentView();
58     MyDrawGraph *body;
59     MyDrawText *part[4];
60     MyDrawText *point[4];
```

```
61  };
62
63  // 区間別採点グラフ表示
64  class ResultGraph : public Draw {
65  public:
66      ResultGraph(Font *font);
67      void Load(const int *ponit, const int num, Song *song);
68      void Delete();
69      ~ResultGraph();
70  private:
71      void ContentView();
72      MyDrawBox *myDrawBox;
73      MyDrawTexts *scale; // 目盛り
74      MyDrawCircle *dot[100]; // 点
75      MyDrawLine *line[100]; // 点
76      MyDrawLine *frame[2]; // 枠線
77      MyDrawTextV *part[64];
78      Font *font;
79      const float w = WIDTH * 0.6, h = HEIGHT * 0.13;
80      int pointMax = 0, partMax = 0;
81  };
82
83  #endif
```

ThroughStart.h

```
1  #ifndef __THROUGHSTART_H_INCLUDED__
2  #define __THROUGHSTART_H_INCLUDED__
3
4  #include "StartScreen.h"
5  #include "ThroughDefine.h"
6
7  class ThroughStart : public StartSceen{
8  public:
9      ThroughStart(Font *f);
10 };
11
12 #endif
```

Top.h

```
1  #ifndef __TOP_H_INCLUDED__
2  #define __TOP_H_INCLUDED__
3
4  #include "DxLib.h"
5  #include "DrawText.h"
6  #include "DrawGraph.h"
7  #include "DrawObject.h"
8  #include "Font.h"
9
10 // トップロゴ
11 class TopLogo : public MyDrawGraph {
12 public:
13     TopLogo( const float y);
14 };
15
16 // NFCタッチメッセージ関係
17 class TopTouchMessage : public MyDrawText {
18 public:
19     TopTouchMessage(Font *font, const float y);
20     void Init(); // 初期化
21     void Update(); // 計算
22     void View(); // 表示
23 private:
24     int t; // 時間
25 };
26
27 // NFCタッチボタン関係
28 class TopTouchButton : public Pos{
29 public:
30     TopTouchButton(Font *font);
31     void View(); // 表示
32     ~TopTouchButton();
33 private:
34     MyDrawTexts *text;
35     MyDrawCircle *circle;
36 };
37
38 #endif
```

TopMain.h

```
#ifndef __TOPMAIN_H_INCLUDED__
#define __TOPMAIN_H_INCLUDED__

#include "Nfc.h"
#include "DxLib.h"
#include "Main.h"
#include "Font.h"
#include "Top.h"
#include "Scene.h"
#include "User.h"

// トップ画面関係
class Top : public Scene {
public:
    Top(Font *font, User *user);
    MainScene Switch(const MainScene scene);
    ~Top();
private:
    Font *f;
    TopLogo *topLogo; // トップロゴ
    TopTouchMessage *topTouchMessage; // NFCタッチメッセージ
    TopTouchButton *topTouchButton; // NFCタッチボタン
    Nfc nfc; // NFC監視
    User *user;
    void ContentUpdate(); // 計算
    void ContentView(); // 表示
    void ContentLoad();
    void ContentDelete();
};

#endif
```

Touch.h

```
1  #ifndef __TOUCH_H_INCLUDED__
2  #define __TOUCH_H_INCLUDED__
3
4  #include "DxLib.h"
5
6  class Touch {
7  public:
8      void Check(); // 確認
9      int Get(int num); // 取得
10     boolean Input(int num, int wait = 30, int duration = 10);  // キーが入
           力されているか
11     boolean Input2(int num, int wait1 = 30, int duration1 = 10, int wait2
           = 100, int duration2 = 6);  // キーが入力されているか
12 private:
13     int key[5];
14 };
15
16
17 #endif
```

User.h

```
#ifndef __USER_H_INCLUDED__
#define __USER_H_INCLUDED__

#include "DxLib.h"

class User {
public:
    void SetUserId(const char *userId); // ユーザーIDを格納
    char *GetUserId(); // ユーザーIDを収得
private:
    char userId[64];
};
#endif
```

2 Web サーバ

2.1 HTML

list.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Mai-Archive</title>
    <meta charset="UTF-8" />
    <meta name="keywords" content="Global Studios, グローバルスタジオ" />
    <meta name="description" content="" />
    <meta name="author" content="Global Studios" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="shortcut icon" href="img/favicon.ico" />
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
        awesome/4.5.0/css/font-awesome.min.css">
    <link rel = "stylesheet" type = "text/css" href = "css/style.css">
</head>
<body>

<header>
        <div class="siteheader shadow1">
            <a href=""><img src="img/logo.png"></a>
            <nav>
                <ul id="menu">
                    <li><a href="">USER ID: daichi</a></li>
                    <li><a href="">アーカイブリスト</a></li>
                    <li><a href="">Mai-Archiveについて</a></li>
                    <li><a href="">ログアウト</a></li>
                </ul>
                <button id="menubutton"><span></span></button>
            </nav>
        </div>
</header>
    <section id="header-back"></section>

    <!-- Contents -->
    <section class="each_menu">
        <div class="wrap">
            <div class="left_menu">
                <a href=""></a><img src="img/Aset.png">
            </div>
            <div class="right_menu">
                <h1>Perfume - FLASH</h1>
                <p>Score: 82</p>
                <p>2016-07-21   13:45</p>
            </div>
        </div>
    </section>

    <section class="each_menu">
        <div class="wrap">
            <div class="left_menu">
                <a href=""></a><img src="img/Bset.png">
            </div>
            <div class="right_menu">
                <h1>最&amp;高 -   きゃりーぱみゅぱみゅ</h1>
                <p>Score: 77</p>
                <p>2016-07-21   13:28</p>
```

```
55                  </div>
56              </div>
57          </div>
58      </section>
59
60      <!-- Script -->
61      <script src="js/lib/jquery.min.js"></script>
62      <script src="js/drawerNav.js"></script>
63  </body>
64  </html>
```

2.2 PHP

api_video.php

```php
<?php
/*
  $db['host'] = "localhost";  // DBサーバのurl
  $db['user'] = "daichi";
  $db['pass'] = "yoshitake";
  $db['dbname'] = "mai-archive";
*/


  $db['host'] = "mysql318.db.sakura.ne.jp";  // DBサーバのurl
  $db['user'] = "globalstudios";
  $db['pass'] = "ni-towakame";
  $db['dbname'] = "globalstudios_mai-archive";

  $mysqli = new mysqli($db['host'], $db['user'], $db['pass']);
  if ($mysqli->connect_errno) {
      print('<p>データベースへの接続に失敗しました。</p>' . $mysqli->
          connect_error);
      exit();
    }

  $mysqli->query("SET NAMES utf8");

  $mysqli->select_db($db['dbname']);

  $query = "update history set video_url = '" . $_GET['video_url'] . "'
      where user_id = '" . $_GET['user'] . "' AND date = '" .$_GET['date']
       . "';";

  $result = $mysqli->query($query);
  if (!$result) {
    print('クエリーが失敗しました。' . $mysqli->error);
    $mysqli->close();
    exit();
  }

  echo "1";
?>
```

api_history.php

```php
<?php

  $db['host'] = "localhost";  // DB server's url
  $db['user'] = "daichi";
  $db['pass'] = "yoshitake";
  $db['dbname'] = "mai-archive";

/*
  $db['host'] = "mysql318.db.sakura.ne.jp";  // DB sever's url
  $db['user'] = "globalstudios";
  $db['pass'] = "ni-towakame";
  $db['dbname'] = "globalstudios_mai-archive";
*/


  $mysqli = new mysqli($db['host'], $db['user'], $db['pass']);
  if ($mysqli->connect_errno) {
      print('<p>データベースへの接続に失敗しました。</p>' . $mysqli->
          connect_error);
      exit();
    }

  $mysqli->select_db($db['dbname']);

  $userid = $_GET["user"];
  $num_fetch = 2; //number of scores fetched from the database

  //Get the max number of music id
  $query = "SELECT MAX(music_id) FROM music";
  $result = $mysqli->query($query);
  $num_music = $result->fetch_assoc()["MAX(music_id)"];

  for ($i = 1; $i <= $num_music; $i++) {
    $j = 0;
    $text = $i;
    $query = "SELECT * FROM history WHERE user_id = '". $userid . "'  AND
        music_id = '". $i ."' order by history_id DESC LIMIT ". $num_fetch
        ;
    $result = $mysqli->query($query);
    while ($row = $result->fetch_assoc()) {
      if ($j < $num_fetch) {
        $text .= "||" . $row["score"];
        $j++;
      }
    }
    for (; $j < $num_fetch; $j++) {
      $text .= "||-1";
    }
    echo $text;
    if ($i != $num_music) {
      echo "\n";
    }
  }
?>
```

api_add.php

```php
<?php
/*
  $db['host'] = "localhost";  // DBサーバのurl
  $db['user'] = "daichi";
  $db['pass'] = "yoshitake";
  $db['dbname'] = "mai-archive";
*/


  $db['host'] = "mysql318.db.sakura.ne.jp";  // DBサーバのurl
  $db['user'] = "globalstudios";
  $db['pass'] = "ni-towakame";
  $db['dbname'] = "globalstudios_mai-archive";

  $mysqli = new mysqli($db['host'], $db['user'], $db['pass']);
  if ($mysqli->connect_errno) {
      print('<p>データベースへの接続に失敗しました。</p>' . $mysqli->
          connect_error);
      exit();
    }

  $mysqli->query("SET NAMES utf8");

  $mysqli->select_db($db['dbname']);

  $query = "insert into history(user_id, music_id, date, score, part, body
      , timing, expression, comment) values('" . $_GET["user"] . "', '" .
      $_GET["song"]. "', '" . $_GET["date"] . "', '" . $_GET["total"] . "
      ', '" . $_GET["part"] . "' ,'" .  $_GET["body"] . "', '" . $_GET["
      timing"] . "', '" . $_GET["expression"] . "', '" . $_GET["comment"]
      . "');";

  $result = $mysqli->query($query);
  if (!$result) {
    print('クエリーが失敗しました。' . $mysqli->error);
    $mysqli->close();
    exit();
  }

  echo "1";
?>
```

list.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Mai-Archive</title>
    <meta charset="UTF-8" />
    <meta name="keywords" content="Global Studios, グローバルスタジオ" />
    <meta name="description" content="" />
    <meta name="author" content="Global Studios" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="shortcut icon" href="img/favicon.ico" />
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css">
    <link rel = "stylesheet" type = "text/css" href = "css/style.css">
</head>

<body>
<?php
  session_start();

  $db['host'] = "localhost";  // DB server's url
  $db['user'] = "daichi";
  $db['pass'] = "yoshitake";
  $db['dbname'] = "mai-archive";

/*
  $db['host'] = "mysql318.db.sakura.ne.jp";  // DB sever's url
  $db['user'] = "globalstudios";
  $db['pass'] = "ni-towakame";
  $db['dbname'] = "globalstudios_mai-archive";
*/


  $mysqli = new mysqli($db['host'], $db['user'], $db['pass']);
  if ($mysqli->connect_errno) {
      print('<p>データベースへの接続に失敗しました。</p>' . $mysqli->connect_error);
      exit();
    }

  $mysqli->select_db($db['dbname']);

  $userid = $_SESSION["USERID"];
  //$userid = $_GET["user"];
  $num_fetch = 2; //number of scores fetched from the database

  //Get the max number of music id
  $query = "SELECT MAX(music_id) FROM music";
  $result = $mysqli->query($query);
  $num_music = $result->fetch_assoc()["MAX(music_id)"];

  $text = array();
  for ($i = 1; $i <= $num_music; $i++) {
    $j = 0;
    $text[$i] = $i;
    $query = "SELECT * FROM history WHERE user_id = '". $userid . "'  AND music_id = '". $i ."' order by history_id DESC LIMIT ". $num_fetch ;
    $result = $mysqli->query($query);
    while ($row = $result->fetch_assoc()) {
      if ($j < $num_fetch) {
        $text[$i] .= "||" . $row["score"];
```

```
58          $j++;
59        }
60      }
61      for (; $j < $num_fetch; $j++) {
62        $text[$i] .= "||-1";
63      }
64    }
65  echo '
66  <header>
67          <div class="siteheader shadow1">
68              <a href=""><img src="img/logo.png"></a>
69              <nav>
70                  <ul id="menu">
71                      <li><a href="">USER ID: '. $userid .'</a></li>
72                      <li><a href="">アーカイブリスト</a></li>
73                      <li><a href="">Mai-Archiveについて</a></li>
74                      <li><a href="logout.php">ログアウト</a></li>
75                  </ul>
76                  <button id="menubutton"><span></span></button>
77              </nav>
78          </div>
79  </header>
80  <section id="header-back"></section>';
81
82  for ($i=1; $i <= $num_music; $i++) {
83      $pieces = explode("||", $text[$i]);
84      if ($pieces[1] != "-1") {
85          $query = "SELECT * FROM music WHERE music_id = '" . $pieces[0] ."
                ';";
86          $result = $mysqli->query($query);
87          $row = $result->fetch_assoc();
88          echo '<section class="each_menu">
89          <div class="wrap">
90              <div class="left_menu">
91                  <a href="music.php?id=' . $pieces[0] . '"></a><img src="
                        img/music/' . $pieces[0] . '.jpg">
92              </div>
93              <div class="right_menu">
94                  <h1>' . $row["artist"] . ' - ' . $row["song"] .'</h1>
95                  <p>1st Latest Score: ' . $pieces[1] . '</p>';
96          if ($pieces[2] != "-1") {
97              echo '<p>2nd Latest Score: ' . $pieces[2] . '</p>';
98          }
99          echo '</div></div></section>';
100     }
101 }
102 ?>
103     <!-- Script -->
104     <script src="js/lib/jquery.min.js"></script>
105     <script src="js/drawerNav.js"></script>
106 </body>
107 </html>
```

login.php

```php
<?php
ini_set( 'display_errors', 1 ); //display error messages
//require 'password.php';
// セッション開始
session_start();

$db['host'] = "localhost";  // DBサーバのurl
$db['user'] = "daichi";
$db['pass'] = "yoshitake";
$db['dbname'] = "mai-archive";

/*
$db['host'] = "mysql318.db.sakura.ne.jp";  // DBサーバのurl
$db['user'] = "globalstudios";
$db['pass'] = "ni-towakame";
$db['dbname'] = "globalstudios_mai-archive";
*/

// エラーメッセージの初期化
$errorMessage = "";
$db_hashed_pwd = "";

// ログインボタンが押された場合
if (isset($_POST["login"])) {
  // 1. ユーザIDの入力チェック
  if (empty($_POST["userid"])) {
    $errorMessage = "ユーザIDが未入力です。";
  } else if (empty($_POST["password"])) {
    $errorMessage = "パスワードが未入力です。";
  }

  // 2. ユーザIDとパスワードが入力されていたら認証する
  if (!empty($_POST["userid"]) && !empty($_POST["password"])) {
    // mysqlへの接続
    $mysqli = new mysqli($db['host'], $db['user'], $db['pass']);
    if ($mysqli->connect_errno) {
      print('<p>データベースへの接続に失敗しました。</p>' . $mysqli->
          connect_error);
      exit();
    }

    // データベースの選択
    $mysqli->select_db($db['dbname']);

    // 入力値のサニタイズ
    $userid = $mysqli->real_escape_string($_POST["userid"]);

    // クエリの実行
    $query = "SELECT * FROM user WHERE user_id = '" . $userid . "'";
    $result = $mysqli->query($query);
    if (!$result) {
      print('クエリーが失敗しました。' . $mysqli->error);
      $mysqli->close();
      exit();
    }

    while ($row = $result->fetch_assoc()) {
      // パスワード(暗号化済み)の取り出し
      $db_hashed_pwd = password_hash($row['password'], PASSWORD_DEFAULT);
    }
```

```
60
61      // データベースの切断
62      $mysqli->close();
63
64      // ３．画面から入力されたパスワードとデータベースから取得したパスワードのハッシュを比較します。
65      //if ($_POST["password"] == $pw) {
66      if (password_verify($_POST["password"], $db_hashed_pwd)) {
67        // ４．認証成功なら、セッションIDを新規に発行する
68        session_regenerate_id(true);
69        $_SESSION["USERID"] = $_POST["userid"];
70        header("Location: list.php");
71        exit;
72      }
73      else {
74        // 認証失敗
75        $errorMessage = "ユーザIDあるいはセキュリティコードに誤りがあります。";
76      }
77    } else {
78      // 未入力なら何もしない
79    }
80  }
81
82  ?>
83    <!DOCTYPE html>
84  <html>
85  <head>
86    <title>Mai-Archive</title>
87    <meta charset="UTF-8" />
88    <meta name="keywords" content="Global Studios, グローバルスタジオ" />
89    <meta name="description" content="" />
90    <meta name="author" content="Global Studios" />
91    <meta name="viewport" content="width=device-width, initial-scale=1.0">
92      <link rel="shortcut icon" href="img/favicon.ico" />
93      <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css">
94    <link rel = "stylesheet" type = "text/css" href = "css/style.css">
95  </head>
96    <body>
97    <?php
98    echo '
99  <header>
100       <div class="siteheader shadow1">
101         <a href=""><img src="img/logo.png"></a>
102         <nav>
103           <ul id="menu">
104             <li><a href="">Mai-Archiveについて</a></li>
105             <li><a href="login.php">ログインしていません。</a></li>
106           </ul>
107           <button id="menubutton"><span></span></button>
108         </nav>
109       </div>
110  </header>
111  <section id="header-back"></section>';
112    ?>
113    <!-- $_SERVER['PHP_SELF']はXSSの危険性があるので、actionは空にしておく -->
114    <!--<form id="loginForm" name="loginForm" action="<?php print($_SERVER['PHP_SELF']) ?>" method="POST">-->
115    <section class="each_menu"><div class="wrap">
116    <h2>Mai-Card表面のユーザIDと裏面のセキュリティコードを入力してください。</h2><br />
```

```
117     <form id="loginForm" name="loginForm" action="" method="POST">
118     <div><?php echo $errorMessage ?></div>
119     <label for="userid">ユーザID</label><br /><input type="text" id="userid"
            name="userid" value="">
120     <br>
121     <label for="password">セキュリティコード</label><br /><input type="
            password" id="password" name="password" value="">
122     <br /><br />
123     <input type="submit" id="login" name="login" value="ログイ
            ン" class="login_button">
124     </form>
125   </div>
126     </section>
127     </body>
128     <!-- Script -->
129     <script src="js/lib/jquery.min.js"></script>
130     <script src="js/drawerNav.js"></script>
131   </html>
```

logout.php

```php
<?php
session_start();

if (isset($_SESSION["USERID"])) {
  $errorMessage = "ログアウトしました。";
}
else {
  $errorMessage = "セッションがタイムアウトしました。";
}
// セッション変数のクリア
$_SESSION = array();
// クッキーの破棄は不要
//if (ini_get("session.use_cookies")) {
//    $params = session_get_cookie_params();
//    setcookie(session_name(), '', time() - 42000,
//        $params["path"], $params["domain"],
//        $params["secure"], $params["httponly"]
//    );
//}
// セッションクリア
@session_destroy();
?>

<!doctype html>
<html>
  <head>
    <title>Mai-Archive</title>
  <meta charset="UTF-8" />
  <meta name="keywords" content="Global Studios, グローバルスタジオ" />
  <meta name="description" content="" />
  <meta name="author" content="Global Studios" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="shortcut icon" href="img/favicon.ico" />
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css">
  <link rel = "stylesheet" type = "text/css" href = "css/style.css">
</head>
  <body>

    <?php
  echo '
<header>
    <div class="siteheader shadow1">
      <a href=""><img src="img/logo.png"></a>
      <nav>
        <ul id="menu">
          <li><a href="">Mai-Archiveについて</a></li>
          <li><a href="login.php">ログインしていません。</a></li>
        </ul>
        <button id="menubutton"><span></span></button>
      </nav>
    </div>
</header>
<section id="header-back"></section>';

  ?>
  <section class="each_menu">
    <div class="wrap">
      <a href="login.php"></a><h1>&lt; ログイン画面に戻る。</h1>
    </div>
  </section>
```

```
61
62    <section class="each_menu">
63    <div class="wrap">
64      <div><?php echo $errorMessage; ?></div>
65    </div>
66    </section>
67    </body>
68    <!-- Script -->
69    <script src="js/lib/jquery.min.js"></script>
70    <script src="js/drawerNav.js"></script>
71  </html>
```

main.php

```php
<?php
session_start();

// ログイン状態のチェック
if (!isset($_SESSION["USERID"])) {
  header("Location: logout.php");
  exit;
}
?>

<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Mai-Archive</title>
  </head>
  <body>
  <h1>Mai-Archive</h1>
  <!-- ユーザIDにHTMLタグが含まれても良いようにエスケープする -->
  <h2>ようこそ<?=htmlspecialchars($_SESSION["USERID"], ENT_QUOTES); ?>さん</h2><hr />

<?php
  $db['host'] = "localhost";  // DBサーバのurl
  $db['user'] = "daichi";
  $db['pass'] = "yoshitake";
  $db['dbname'] = "mai-archive";

/*
  $db['host'] = "mysql318.db.sakura.ne.jp";  // DBサーバのurl
  $db['user'] = "globalstudios";
  $db['pass'] = "ni-towakame";
  $db['dbname'] = "globalstudios_mai-archive";
*/


  $mysqli = new mysqli($db['host'], $db['user'], $db['pass']);
  if ($mysqli->connect_errno) {
      print('<p>データベースへの接続に失敗しました。</p>' . $mysqli->connect_error);
      exit();
    }

  $mysqli->select_db($db['dbname']);

  $userid = $mysqli->real_escape_string($_SESSION["USERID"]);

   // クエリの実行
    $query = "SELECT * FROM video WHERE user_name = '" . $_SESSION["USERID"] . "' order by id";
    $result = $mysqli->query($query);
    if (!$result) {
      print('クエリーが失敗しました。' . $mysqli->error);
      $mysqli->close();
      exit();
    }
    while ($row = $result->fetch_assoc()) {
      echo '<p>' . $row['artist'];
      echo ' - ' . $row['song'] . '</p>';
      echo '<iframe width="560" height="315" src="https://www.youtube.com/embed/' . $row['url'] .'" frameborder="0" allowfullscreen></
```

```
            iframe></p><hr />';
58      }
59
60  ?>
61    <a href="logout.php">ログアウト</a>
62    </body>
63  </html>
```

make_history_table.php

```php
<?php
  $db['host'] = "localhost";  // DBサーバのurl
  $db['user'] = "daichi";
  $db['pass'] = "yoshitake";
  $db['dbname'] = "mai-archive";

/*
  $db['host'] = "mysql318.db.sakura.ne.jp";  // DBサーバのurl
  $db['user'] = "globalstudios";
  $db['pass'] = "ni-towakame";
  $db['dbname'] = "globalstudios_mai-archive";
*/


  $mysqli = new mysqli($db['host'], $db['user'], $db['pass']);
  if ($mysqli->connect_errno) {
      print('<p>データベースへの接続に失敗しました。</p>' . $mysqli->
          connect_error);
      exit();
    }

  $mysqli->select_db($db['dbname']);

  $userid = $_GET["user"];

  $query = "CREATE TABLE `history_" . $userid ."` (
  `id` int(11) NOT NULL,
  `date` text NOT NULL,
  `artist` text NOT NULL,
  `song` text NOT NULL,
  `prev_score` int(11) NOT NULL,
  `prev_score2` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;";
  $result = $mysqli->query($query);
  if (!$result) {
    print('クエリーが失敗しました。' . $mysqli->error);
    $mysqli->close();
    exit();
  }

  $query = "ALTER TABLE `history_" . $userid . "`
  ADD PRIMARY KEY (`id`);";
  $result = $mysqli->query($query);
  if (!$result) {
    print('クエリーが失敗しました。' . $mysqli->error);
    $mysqli->close();
    exit();
  }

  $query = "ALTER TABLE `history_" . $userid . "`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;";
  $result = $mysqli->query($query);
  if (!$result) {
    print('クエリーが失敗しました。' . $mysqli->error);
    $mysqli->close();
    exit();
  }

  echo "<h1>Made the User History Table: history_" . $userid . "</h1>";
?>
```

music.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Mai-Archive</title>
    <meta charset="UTF-8" />
    <meta name="keywords" content="Global Studios, グローバルスタジオ" />
    <meta name="description" content="" />
    <meta name="author" content="Global Studios" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="shortcut icon" href="img/favicon.ico" />
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css">
    <link rel = "stylesheet" type = "text/css" href = "css/style.css">
</head>

<body>
<?php
  session_start();

  $db['host'] = "localhost";  // DB server's url
  $db['user'] = "daichi";
  $db['pass'] = "yoshitake";
  $db['dbname'] = "mai-archive";

/*
  $db['host'] = "mysql318.db.sakura.ne.jp";  // DB sever's url
  $db['user'] = "globalstudios";
  $db['pass'] = "ni-towakame";
  $db['dbname'] = "globalstudios_mai-archive";
*/


  $mysqli = new mysqli($db['host'], $db['user'], $db['pass']);
  if ($mysqli->connect_errno) {
      print('<p>データベースへの接続に失敗しました。</p>' . $mysqli->connect_error);
      exit();
    }

  $mysqli->select_db($db['dbname']);

  $userid = $_SESSION["USERID"];
  $musicid = $_GET["id"];

echo '
<header>
        <div class="siteheader shadow1">
            <a href=""><img src="img/logo.png"></a>
            <nav>
                <ul id="menu">
                    <li><a href="">USER ID: '. $userid .'</a></li>
                    <li><a href="">アーカイブリスト</a></li>
                    <li><a href="">Mai-Archiveについて</a></li>
                    <li><a href="logout.php">ログアウト</a></li>
                </ul>
                <button id="menubutton"><span></span></button>
            </nav>
        </div>
</header>
<section id="header-back"></section>';

```

```
60  $query = "SELECT * FROM music WHERE music_id = '" . $musicid ."';";
61  $result = $mysqli->query($query);
62  $row = $result->fetch_assoc();
63
64  echo '
65  <section class="each_menu">
66    <div class="wrap">
67      <div class="music_main">
68        <h1>' . $row["artist"] . ' - ' . $row["song"] .'</h1>
69        <img src="img/music/' . $musicid . '.jpg">
70      </div>
71        <div class="youtube_video">
72        <iframe src="https://www.youtube.com/embed/' . $row["video_url"] .'"
              frameborder="0" allowfullscreen></iframe>
73        </div>
74    </div>
75  </section>';
76
77
78      $query = "SELECT * FROM history WHERE music_id = '" . $musicid ."' AND
            user_id = '". $userid ."' order by history_id desc;";
79    $result = $mysqli->query($query);
80    while ($row = $result->fetch_assoc()) {
81      echo '<section class="each_menu">
82      <div class="wrap">
83        <div class="left_menu_music">
84          <a href="./video.php?id='. $row["history_id"] .'"></a><img src="
              http://i.ytimg.com/vi/'. $row["video_url"] .'/maxresdefault.
              jpg">
85        </div>
86        <div class="right_menu_music">
87          <p>'.$row["date"].'</p>
88          <h1>Score: ' . $row["score"] .'</h1>';
89    echo '</div></div></section>';
90    }
91  ?>
92      <!-- Script -->
93      <script src="js/lib/jquery.min.js"></script>
94      <script src="js/drawerNav.js"></script>
95  </body>
96  </html>
```

video.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Mai-Archive</title>
    <meta charset="UTF-8" />
    <meta name="keywords" content="Global Studios, グローバルスタジオ" />
    <meta name="description" content="" />
    <meta name="author" content="Global Studios" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="shortcut icon" href="img/favicon.ico" />
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css">
    <link rel = "stylesheet" type = "text/css" href = "css/style.css">
</head>

<body>
<?php
  session_start();

  $db['host'] = "localhost";  // DB server's url
  $db['user'] = "daichi";
  $db['pass'] = "yoshitake";
  $db['dbname'] = "mai-archive";

/*
  $db['host'] = "mysql318.db.sakura.ne.jp";  // DB sever's url
  $db['user'] = "globalstudios";
  $db['pass'] = "ni-towakame";
  $db['dbname'] = "globalstudios_mai-archive";
*/


  $mysqli = new mysqli($db['host'], $db['user'], $db['pass']);
  if ($mysqli->connect_errno) {
      print('<p>データベースへの接続に失敗しました。</p>' . $mysqli->connect_error);
      exit();
    }

  $mysqli->select_db($db['dbname']);

  $userid = $_SESSION["USERID"];
  $historyid = $_GET["id"];

echo '
<header>
        <div class="siteheader shadow1">
            <a href=""><img src="img/logo.png"></a>
            <nav>
                <ul id="menu">
                    <li><a href="">USER ID: '. $userid .'</a></li>
                    <li><a href="">アーカイブリスト</a></li>
                    <li><a href="">Mai-Archiveについて</a></li>
                    <li><a href="logout.php">ログアウト</a></li>
                </ul>
                <button id="menubutton"><span></span></button>
            </nav>
        </div>
</header>
<section id="header-back"></section>';

```

```
60
61
62      $query = "SELECT * FROM history WHERE history_id = '" . $historyid ."'
            AND user_id = '". $userid ."';";
63    $result = $mysqli->query($query);
64    $row = $result->fetch_assoc();
65
66    $query = "SELECT * FROM music WHERE music_id = '" . $row["music_id"] ."
          ';";
67    $result = $mysqli->query($query);
68    $row = $result->fetch_assoc();
69
70    echo '<section class="each_menu"><div class="wrap">';
71    echo '<a href="./music.php?id='. $row["music_id"] .'"></a>';
72    echo '<h1>&lt; '.$row["artist"].' - ' . $row["song"] .'</h1>';
73    echo '</div></section>';
74
75
76    $query = "SELECT * FROM history WHERE history_id = '" . $historyid ."'
          AND user_id = '". $userid ."';";
77    $result = $mysqli->query($query);
78    $row = $result->fetch_assoc();
79
80    echo '<section class="each_menu"><div class="wrap">';
81    echo '<div class="youtube_video">
82          <iframe src="https://www.youtube.com/embed/' . $row["video_url"] .
              '" frameborder="0" allowfullscreen></iframe>
83          </div>';
84    echo '</div></section>';
85
86    echo '<section class="each_menu"><div class="wrap">';
87    echo '<h1>Total Score: '.$row["score"].'</h1>';
88    echo '<h1>'.$row["comment"].'</h1>';
89    echo '</div></section>';
90
91    echo '<section class="each_menu"><div class="wrap">';
92    $pieces = explode("/", $row["part"]);
93    for ($i=0; $i < sizeof($pieces) - 1 ; $i+=2) {
94      echo "<h1>".$pieces[$i].": ".$pieces[$i+1]."</h1>";
95    }
96    echo '</div></section>';
97
98
99    echo '<section class="each_menu"><div class="wrap">';
100   $pieces = explode("/", $row["body"]);
101   $body = array("右腕", "左腕", "右足", "左足");
102   for ($i=0; $i < sizeof($pieces); $i++) {
103     echo "<h1>".$body[$i].": ".$pieces[$i]."</h1>";
104   }
105   echo '</div></section>';
106
107   echo '<section class="each_menu"><div class="wrap">';
108   echo '<h1>タイミング: '.$row["timing"].'</h1>';
109   echo '<h1>表情: '.$row["expression"].' / 8 </h1>';
110   echo '</div></section>';
111 ?>
112     <!-- Script -->
113     <script src="js/lib/jquery.min.js"></script>
114     <script src="js/drawerNav.js"></script>
115 </body>
116 </html>
```

2.3 CSS

css/style.css

```
1  * {
2      padding: 0;
3      margin: 0;
4  }
5
6  body {
7      width: 100%;
8      font-family: "Lato", "Lucida Grande", "Hiragino Kaku Gothic ProN", "ヒ
           ラギノ角ゴ ProN W3", "Meiryo", "メイリオ", sans-serif;
9      font-weight: normal;
10     font-size: 16px;
11     color: #fff;
12     background-color: #519bbe;
13
14 }
15
16 header {
17     width: 100%;
18     height: 56px;
19     background-color: rgba(98,183,226,0.93);
20     color: #fff;
21     font-size: 16px;
22     line-height: 16px;
23     position: fixed;
24     top: 0;
25     margin-bottom: 116px;
26     z-index:1;
27 }
28
29 header img {
30     margin-top: 4px;
31     height: 32px;
32 }
33 header h1 {
34     margin-bottom: 32px;
35 }
36
37 header a {
38     /*display:block;*/
39     margin-bottom: -20px;
40 }
41
42 #header-back {
43     width: 100%;
44     height: 56px;
45     background-color: rgba(98,183,226,1);;
46 }
47
48 .siteheader {
49     padding: 8px 16px;
50 }
51
52 #head-left {
53     float: left;
54 }
55
56 #head-right {
```

```
57      float: right;
58  }
59
60  h1, h2, h3 {
61      font-weight: normal;
62  }
63
64  p {
65      font-size: 16px;
66      line-height: 28px;
67  }
68
69  a {
70      text-decoration: none;
71      color: #fff;
72  }
73
74  li {
75      list-style: none;
76  }
77
78  img {
79      max-width: 100%;
80  }
81
82  .icon {
83      font-size: 32px;
84      margin-right: 8px;
85  }
86
87  #tweets {
88      margin: 0 auto;
89      width: 100%;
90      max-width: 736px;
91  }
92
93  .twitter-tweet {
94      margin: 16px auto !important;
95      width: 100% !important;
96      max-width: 480px !important;
97  }
98
99  @media screen and (min-width: 784px) {
100     .twitter-tweet {
101         margin: 16px 16px 16px 0! important;
102         width: 360px !important;
103         float: left !important;
104     }
105
106     .twitter-tweet:nth-of-type(2n) {
107         margin: 16px 0 16px 0 !important;
108     }
109 }
110
111 .wrap {
112     max-width: 996px;
113     margin: 0 auto;
114     padding: 24px 16px;
115     overflow: hidden;
116     zoom: 1;
117     position: relative;
118 }
119
120 .wrap a {
```

```
121     display: :block;
122     width: 100%;
123     position: absolute;
124     top: 0;
125     left: 0;
126     width: 100%;
127     height: 100%;
128 }
129 
130 .left_content {
131     float: left;
132     width: 48%;
133     margin-right: 4%;
134     overflow:hidden;
135     zoom:1;
136 }
137 
138 .right_content {
139     float: left;
140     width: 48%;
141     overflow:hidden;
142     zoom:1;
143 }
144 
145 .left_menu {
146     float: left;
147     width: 28%;
148     margin-right: 8%;
149     overflow:hidden;
150     zoom:1;
151 }
152 
153 .right_menu {
154     float: left;
155     width: 64%;
156     overflow:hidden;
157     zoom:1;
158 }
159 
160 @media screen and (min-width: 800px) {
161     .left_menu {
162         width: 20%;
163     }
164 
165     .right_menu {
166         width: 72%;
167     }
168 }
169 
170 .left_menu_music {
171     float: left;
172     width: 42%;
173     margin-right: 8%;
174     overflow:hidden;
175     zoom:1;
176 }
177 
178 .right_menu_music {
179     float: left;
180     width: 50%;
181     overflow:hidden;
182     zoom:1;
183 }
184 
```

```
185 @media screen and (min-width: 800px) {
186     .left_menu_music {
187         width: 40%;
188     }
189 
190     .right_menu_music {
191         width: 52%;
192     }
193 }
194 
195 .soldout_menu {
196     color: #ddd;
197 }
198 
199 .soldout {
200     color: #FFC40D;
201 }
202 
203 .each_menu {
204     background-color: #519bbe;
205     border-bottom: 1px solid #fff;
206 }
207 
208 .each_menu:last-of-type {
209     border-bottom: 0px solid #fff;
210 }
211 
212 #menu_icon {
213     background-color: #5caba6;
214     text-align: center;
215 }
216 
217 #menu_icon .wrap img {
218     width: 40%;
219     max-width: 320px;
220     margin-bottom: 32px;
221 }
222 
223 #menu_info {
224     background-color: #16a086;
225 }
226 
227 #menu_buttons {
228     background-color: #5caba6;
229 }
230 
231 #buttons {
232     margin: 0 auto;
233     width: 100%;
234     max-width: 320px;
235 }
236 
237 #buttons .twin {
238     float: left;
239     width: 42%;
240     margin: 0 2%;
241     padding: 4% 2%;
242     text-align: center;
243     font-size: 16px;
244     line-height: 24px;
245     margin-bottom: 16px;
246 }
247 
248 #likes_button {
```

```
249     overflow: hidden;
250     background-color: #169086;
251     border: none;
252     width: 86%;
253     margin-left: 2%;
254     color: #fff;
255     font-size: 24px;
256     padding: 16px;
257     text-align: center;
258 }
259 
260 #report_button {
261     background-color: #cf3c22;
262 }
263 
264 #tweet_button {
265     background-color: #da532c;
266 }
267 
268 #menu_tweets {
269     background-color: #16a086;
270 }
271 
272 #camera_button {
273     background-color: #169086;
274 }
275 
276 .report {
277     font-size: 40px;
278     margin-bottom: 48px;
279 }
280 
281 #soldout_button {
282     background-color: #da532c;
283 }
284 
285 #onsell_button {
286     background-color: #169086;
287 }
288 
289 #soldout_button {
290     border: none;
291     width: 100%;
292     color: #fff;
293     font-size: 24px;
294     padding: 16px;
295     text-align: center;
296     max-width: 560px;
297     display: block;
298     margin: 0 auto;
299     margin-bottom: 32px;
300 }
301 
302 #onsell_button {
303     border: none;
304     width: 100%;
305     color: #fff;
306     font-size: 24px;
307     padding: 16px;
308     text-align: center;
309     max-width: 560px;
310     display: block;
311     margin: 0 auto;
312 }
```

```
313
314 #tweet_button {
315     font-size: 24px;
316     padding: 16px;
317     text-align: center;
318     max-width: 560px;
319     margin: 0 auto;
320     margin-bottom: 32px;
321     display: block;
322 }
323
324 #camera_button {
325 /*  width: 100%;*/
326     font-size: 24px;
327     padding: 16px;
328     text-align: center;
329     max-width: 560px;
330     margin: 0 auto;
331     margin-bottom: 32px;
332     display: block;
333 }
334
335 #camera_button input[type="file"] {
336   opacity: 0;
337   filter: progid:DXImageTransform.Microsoft.Alpha(opacity=0);
338     margin: 0 auto;
339   font-size: 100px;
340   cursor: pointer;
341 }
342
343 textarea {
344     width: 100%;
345     max-width: 592px;
346     height: 160px;
347     font-size: 16px;
348     resize: none;
349     display: block;
350     margin: 0 auto;
351     margin-bottom: 40px;
352 }
353
354 .icon_camera_button {
355     font-size: 24px;
356     margin-right: 8px;
357 }
358
359 .icon_tweet_button {
360     font-size: 30px;
361     margin-right: 4px;
362 }
363
364 #day_next {
365     float: left;
366     margin-left: 8px;
367 }
368
369 #day_prev {
370     float: left;
371     margin-right: 8px;
372 }
373
374 #prev {
375     margin-right: 8px;
376 }
```

```
377 /*
378 @media screen and (max-width: 352px) {
379     #day_prev, #head-left, #day_next, #head-right {
380         font-size: 14px;
381         line-height: 24px;
382     }
383 
384     header {
385         height: 72px;
386         margin-bottom: 88px;
387     }
388 
389     #header-back {
390         height: 72px;
391     }
392 
393     header img {
394         height: 24px;
395     }
396 }
397 */
398 
399 .err {
400     margin-bottom: 8px;
401     text-align: center;
402     color: #FFC40D;
403 }
404 
405 .logout {
406     background-color: #169086;
407     text-align:center;
408     width: 100%;
409     max-width: 592px;
410     margin: 0 auto;
411 }
412 
413 .twi-none {
414     margin-top: 16px;
415     font-size: 20px;
416 }
417 
418 #logout_button {
419     overflow: hidden;
420     border: none;
421     color: #fff;
422     font-size: 24px;
423     padding: 16px;
424     text-align: center;
425     display: inline-block;
426 }
427 
428 .login {
429     background-color: #169086;
430     text-align:center;
431     width: 100%;
432     max-width: 592px;
433     margin: 0 auto;
434     margin-top: 24px;
435 }
436 
437 
438 
439 
440 
```

```
441 button {
442     display: block;
443     cursor: pointer;
444     border: none;
445     outline: none;
446 }
447 
448 .siteheader button {
449     width: 48px;
450     height: 48px;
451     background: none;
452     position: absolute;
453     right: 0;
454     bottom: 100%;
455     color: #f8f8f8;
456     font-size: 32px;
457     line-height: 40px;
458     display: block;
459 }
460 
461 .siteheader button:before,
462 .siteheader button:after,
463 .siteheader button span {
464     margin: -1px 8px 0;
465     width: 32px;
466     position: absolute;
467     display: block;
468     border-top: solid 2px #f0f0f0;
469     -webkit-transition: all 0.3s ease;
470     transition: all 0.3s ease;
471     -webkit-transform: translate3d(0,0,0);
472     transform: translate3d(0,0,0);
473 }
474 
475 .siteheader button:before{
476     content: "";
477     top: 30%;
478 }
479 
480 .siteheader button:after{
481     content: "";
482     top: 70%;
483 }
484 
485 .siteheader button span {
486     top: 50%;
487 }
488 
489 .siteheader button:active {
490     background-color: #b7d5cd;
491 }
492 
493 .siteheader .opened button:before {
494     top: 50%;
495     -webkit-transform: rotate(45deg);
496     transform: rotate(45deg);
497 }
498 
499 .siteheader .opened button:after {
500     top: 50%;
501     -webkit-transform: rotate(-45deg);
502     transform: rotate(-45deg);
503 }
504 
```

```
505 .siteheader .opened button span {
506     opacity: 0;
507 }
508 
509 .siteheader nav {
510     width: 100%;
511     position: absolute;
512     top: 100%;
513     left: 0;
514     color: #f8f8f8;
515     background-color: rgba(98,183,226,0.93);
516 }
517 
518 .siteheader nav ul {
519     width: 100%;
520     max-height: 0;
521     border-top: solid 1px rgba(0,0,0,0);
522     overflow: hidden;
523     -webkit-transition: all 0.3s ease;
524     transition: all 0.3s ease;
525     -webkit-transform: translate3d(0,0,0);
526     transform: translate3d(0,0,0);
527 }
528 
529 .siteheader nav.opened ul {
530     max-height: 200px;
531     border-color: #f8f8f8;
532 }
533 
534 .siteheader nav li {
535     height: 40px;
536 }
537 
538 .siteheader nav li a {
539     padding: 0 16px;
540     width: 100%;
541     height: 100%;
542     line-height: 40px;
543     display: block;
544     color: #f8f8f8;
545 }
546 
547 .siteheader nav li:active {
548     background-color: #FFC40D;
549 }
550 
551 .music_main {
552     margin-bottom: 24px;
553 }
554 
555 .music_main img {
556     width: 50%;
557     display: block;
558     margin: 0 auto;
559 }
560 
561 .youtube_video {
562     position: relative;
563     width: 100%;
564     padding-top: 56.25%;
565 }
566 .youtube_video iframe {
567     position: absolute;
568     top: 0;
```

```
569         left: 0;
570         width: 100% !important;
571         height: 100% !important;
572     }
573
574     .music_main h1 {
575         text-align: center;
576         margin-bottom: 24px;
577     }
578
579     input {
580         width: 100%;
581     }
582
583     input#login {
584         border-style: none;
585         padding: 24px 0;
586         background-color: #62b7e2;
587         color: #fff;
588         font-size: 18px;
589     }
```

2.4 Javascript

js/drawerNav.js

```
1  $(function() {
2    $('#menubutton').click(function() {
3      $('.siteheader nav').toggleClass('opened');
4    });
5  }());
```

js/scroll.js

```
$(function(){

    $('a[href^=#]').click(function() {
        var href= $(this).attr("href");
        var target = $(href == "#" || href == "" ? 'html' : href);
        var position = target.offset().top + 32;
        $('body,html').animate({scrollTop:position}, 500, 'swing');
        return false;
    });

    $('#scrollup').click(function() {
        $('body,html').animate({scrollTop:0}, 400, 'swing');
    });
});
```

3 NFC リーダ用 Android アプリケーション

MainActivity.java

```
1  package com.example.kousukenezu.mai_kagamilogin;
2
3  import android.support.v7.app.AppCompatActivity;
4  import android.os.Bundle;
5  import android.text.SpannableStringBuilder;
6  import android.view.View;
7  import android.widget.*;
8
9  public class MainActivity extends AppCompatActivity {
10
11     MyNfc myNfc;
12     UsbMessage usbMessage = null;
13     TextView text;
14
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.activity_main);
19
20         usbMessage = new UsbMessage(9999);
21         myNfc = new MyNfc(this, usbMessage);
22         usbMessage.setMainActivity(this);
23         text = (TextView)findViewById(R.id.textView);
24
25         /* 送信ボタンイベント */
26         findViewById(R.id.button).setOnClickListener(new View.
               OnClickListener() {
27             @Override
28             public void onClick(View v) {
29                 EditText edit = (EditText)findViewById(R.id.editText);
30                 SpannableStringBuilder sp = (SpannableStringBuilder)edit.
                       getText(); // データ取得
31                 usbMessage.sendData(sp.toString()); // 送信
32             }
33         });
34
35     }
36
37     // 再開時
38     @Override
39     protected void onResume() {
40         super.onResume();
41         myNfc.resume();
42     }
43 }
```

MyNfc.java

```
1  package com.example.kousukenezu.mai_kagamilogin;
2
3
4  import android.os.Parcelable;
5
6  import android.app.PendingIntent;
7  import android.content.Intent;
8  import android.content.IntentFilter;
9  import android.content.IntentFilter.MalformedMimeTypeException;
10 import android.nfc.NdefMessage;
11 import android.nfc.NdefRecord;
12 import android.nfc.NfcAdapter;
13
14 // ========================================
15 // 1,NFCが読み込まれる
16 // 2,インテントを取得
17 // 3,アクティビティが切り替わる
18 // 4,MainActivityのonResume()が呼び出される
19 // 5.MyNfcのresume()が呼び出される
20 // 6.文字列の取得
21 // ========================================
22
23
24 public class MyNfc {
25
26     MainActivity mainActivity = null;
27     UsbMessage usbMessage = null;
28     NfcAdapter mNfcAdapter;
29     PendingIntent mNfcPendingIntent;
30     IntentFilter[] mNdefExchangeFilters;
31     IntentFilter[] mWriteTagFilters;
32
33     String body = null;  // 読み込んだ文字列を格納する
34
35     MyNfc(MainActivity mainActivity, UsbMessage usbMessage) {
36         this.mainActivity = mainActivity;  // メインアクティビティの登録
37         this.usbMessage = usbMessage;      // usbメッセージの登録
38
39         mNfcAdapter = NfcAdapter.getDefaultAdapter(mainActivity);   //
               NFCアダプタ
40
41         mNfcPendingIntent = PendingIntent.getActivity(
42                 mainActivity, 0, new Intent(mainActivity, getClass()).
                       addFlags(
43                         Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);
44         IntentFilter ndefDetected = new IntentFilter(NfcAdapter.
               ACTION_NDEF_DISCOVERED);
45
46         try {
47             ndefDetected.addDataType("text/plain");
48         } catch (MalformedMimeTypeException e) {
49             e.printStackTrace();
50         }
51         mNdefExchangeFilters = new IntentFilter[]{ndefDetected};
52
53         IntentFilter tagDetected = new IntentFilter(NfcAdapter.
               ACTION_TAG_DISCOVERED);
54         mWriteTagFilters = new IntentFilter[]{tagDetected};
55     }
56
57     //   getNdefMessages()を呼び出し、nfcを読み込む
```

```
58      public void resume(){
59          // NfcAdapterがタグを読み込んだときif文通過
60          if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(mainActivity.
                getIntent().getAction())) {
61              NdefMessage[] msg = getNdefMessages(mainActivity.getIntent());
62              byte[] payload = msg[0].getRecords()[0].getPayload();
63              remove_en(new String(payload));
64              mainActivity.text.setText(body);
65              usbMessage.sendData(this.body);
66              mainActivity.setIntent(new Intent());
67          }
68      }
69
70      // ncf作成時に余分に書き込まれる"en"の削除
71      private void remove_en(String body) {
72          this.body = body.replaceFirst("en", "");
73      }
74
75      // 読み込みはここで行われる
76      NdefMessage[] getNdefMessages(Intent intent) {
77          //  Parse the intent
78          NdefMessage[] msgs = null;
79          String action = intent.getAction();
80          if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(action)
81                  || NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action)) {
82              Parcelable[] rawMsgs = intent.getParcelableArrayExtra(
                    NfcAdapter.EXTRA_NDEF_MESSAGES);
83              if (rawMsgs != null) {
84                  msgs = new NdefMessage[rawMsgs.length];
85                  for (int i = 0; i < rawMsgs.length; i++) {
86                      msgs[i] = (NdefMessage) rawMsgs[i];
87                  }
88              } else {
89                  // Unknown tag type
90                  byte[] empty = new byte[] {};
91                  NdefRecord record = new NdefRecord(NdefRecord.TNF_UNKNOWN,
                        empty, empty, empty);
92                  NdefMessage msg = new NdefMessage(new NdefRecord[] {
93                          record
94                  });
95                  msgs = new NdefMessage[] {
96                          msg
97                  };
98              }
99          } else {
100             mainActivity.finish();
101         }
102         return msgs;
103     }
104 }
```

UsbMessage.java

```
1  package com.example.kousukenezu.mai_kagamilogin;
2
3  import java.util.concurrent.Executors;
4  import java.io.IOException;
5  import java.io.PrintStream;
6  import java.net.ServerSocket;
7  import java.net.Socket;
8
9
10 public class UsbMessage {
11     private int mPort;  // ポート番号
12     String data; // 送信用データ
13     static boolean mode_read = true;  // 読み取りモードならtrue
14     MainActivity mainActivity;
15     ServerSocket serverSocket;
16     Socket clientSocket;
17
18     public void setMainActivity(MainActivity mainActivity){
19         this.mainActivity = mainActivity;
20     }
21
22     /* 送信用スレッド */
23     private final Runnable mSendTask = new Runnable() {
24         public void run() {
25             if (mode_read) {
26                 try {
27                     ServerSocket serverSocket = new ServerSocket(mPort);
                           // サーバ側のソケット
28                     Socket clientSocket = serverSocket.accept();
29                     if(clientSocket.isConnected()) {
30                         final PrintStream os = new PrintStream(
                               clientSocket.getOutputStream());
31                         os.println(data);  // データ送信
32                     }
33                     serverSocket.close();
34                     clientSocket.close();  // ソケットの後処理
35                 } catch (IOException e) {
36                 }
37                 mode_read = true;
38             }
39         }
40     };
41
42     UsbMessage(int port) {
43         mPort = port;
44     }
45
46     // 送信用スレッドを起動する
47     public void sendData(String d) {
48         if(mode_read) {
49             data = d;
50             Executors.newSingleThreadExecutor().execute(mSendTask);
51         }
52     }
53 }
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.kousukenezu.mai_kagamilogin"
    android:versionCode="1"
    android:versionName="1.0">

    <uses-feature
        android:name="android.hardware.nfc"
        android:required="true" />

    <uses-permission android:name="android.permission.NFC" />

    <uses-feature android:name="android.hardware.usb.accessory" />

    <uses-permission android:name="android.permission.INTERNET" />

    <uses-sdkandroid
        android:minSdkVersion="10"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
                    />
            </intent-filter>
            <intent-filter>
                <action android:name="android.nfc.action.NDEF_DISCOVERED"
                    />
                <action android:name="android.hardware.usb.action.
                    USB_ACCESSORY_ATTACHED" />

                <category android:name="android.intent.category.DEFAULT" /
                    >

                <data android:mimeType="text/plain" />
            </intent-filter>

            <meta-data
                android:name="android.nfc.action.TAG_DISCOVERED"
                android:resource="@xml/nfc_filter" />
        </activity>

        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
        <meta-data
            android:name="android.hardware.usb.action.
                USB_ACCESSORY_ATTACHED"
            android:resource="@xml/accessory_filter" />
    </application>

</manifest>
```