学生証番号：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿　名前：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

6. Java とインタラクション

インタラクション(interaction)という英語は inter+action から合成されたものであり、相互作用とかやりとりと訳されることが多い。その基本は、人間が何かアクション(操作、行動)をすると、相手側（人間または機械システム）がそのアクションに対応したリアクションをする、という点にある。即ち、アクションとリアクションの対、それが場合によっては反復されるということがインタラクションという概念の基本的要件であるといえる。

真のインタラクションを java で実現するにはイベント駆動モデルを理解しなければならない。イベント駆動は、OS 上でソフトウェア的に発生させる割り込みを実現したもので、この仕組みによってマウスやキーボードからの入力を処理する。たとえば、マウスのボタンをクリックしたときに処理を行うとしたら、マウスボタンのクリックはイベントとして扱い、イベントに対する処理する内容をイベントハンドラと呼ぶ。

6.1 キー入力

sample 5.1 はキーイベントの例である。キー入力で簡単なキーボード演奏ができるアプリケーションである。

```
/* sample 5.1 */
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.sound.midi.*;

public class JavaKeyboard extends Frame
{

        public static void main(String[] args) throws Exception
        {
                JavaKeyboard keyboard = new JavaKeyboard();
        }

    /**  キーボードマップ  */
        private static int[] KEYBOARD_MAP =
        {KeyEvent.VK_A, KeyEvent.VK_Z, KeyEvent.VK_S, KeyEvent.VK_X,
        KeyEvent.VK_C, KeyEvent.VK_F, KeyEvent.VK_V, KeyEvent.VK_G,
        KeyEvent.VK_B, KeyEvent.VK_N, KeyEvent.VK_J, KeyEvent.VK_M,
        KeyEvent.VK_K, KeyEvent.VK_COMMA, KeyEvent.VK_L, KeyEvent.VK_PERIOD,
        KeyEvent.VK_SLASH, KeyEvent.VK_COLON, KeyEvent.VK_BACK_SLASH,
        KeyEvent.VK_CLOSE_BRACKET};

        private static int LEFT_NOTE_NO = 56;
        private Font labelFont = new Font("Monospaced", Font.PLAIN, 16);
        private Label label =
                 new Label("(C4) [C] [V] [B] [N] [M] [,] [.] [/] (C5)", Label.CENTER);
        private Synthesizer synthesizer;
        private MidiChannel channel;
```

```
public JavaKeyboard()
{
        super();

// GUI 生成/配置
        setTitle(getClass().getName());
        setLayout(new BorderLayout());
        label.setFont(labelFont);
        add(label, BorderLayout.CENTER);
        addWindowListener(new WindowAdapter() {
                public void windowClosing(WindowEvent e)
                {
                        close();
                }
        });

//  キーイベントの設定
        addKeyListener(new KeyAdapter() {
                public void keyPressed(KeyEvent e)
                {
                        for(int i = 0; i < KEYBOARD_MAP.length; i++)
                        {
                                if(e.getKeyCode() == KEYBOARD_MAP[i])
                                {
                                        noteOn(LEFT_NOTE_NO + i);
                                        return;
                                }
                        }
                        return;
                }
                public void keyReleased(KeyEvent e)
                {
                        for(int i = 0; i < KEYBOARD_MAP.length; i++)
                        {
                                if(e.getKeyCode() == KEYBOARD_MAP[i])
                                {
                                        noteOff(LEFT_NOTE_NO + i);
                                        return;
                                }
                        }
                        return;
                }
        });

// Synthesizer とその MidiChannel を取得
        try
        {
                synthesizer = MidiSystem.getSynthesizer();
                MidiChannel[] channelList = synthesizer.getChannels();
                channel = channelList[0];
                if(channel == null)
                {
                        close();
                }
                synthesizer.open();
```

```
                }
                catch(Exception e)
                {
                        close();
                }

                setVisible(true);
                pack();
        }

        /** Note on する */
        public void noteOn(int noteNo)
        {
                channel.noteOn(noteNo, 127);
        }

        /** Note off する */
        public void noteOff(int noteNo)
        {
                channel.noteOff(noteNo, 127);
        }

        /** Synthesizer を閉じ，終了する */
        public void close()
        {
                if(synthesizer != null)
                {
                        synthesizer.close();
                }
                System.exit(0);
        }
}
```

キーイベントを処理する方法は二つあり、そのひとつがFrame を継承するaddKeyListener を使う方法で、もう一つがアプレットを継承して処理する方法である。

- 演習１６（＊）

event を用いて押されたキーの文字を出力するプログラムを作成せよ。

- 挑戦１３

画面に現在の時刻を表示し、キーボードで時間が合わせられる、時計プログラムを作成せよ。

6.2 マウス入力

sample 5.1 はアプレットでマウスイベントを処理する例である。

```
/* sample 5.1 */
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class MouseObject extends Applet {
```

```
        Image img, img2;
        int x = 0, y = 0;

        public void init() {
                img = getImage(getClass().getResource("back.gif"));
                img2 = getImage(getClass().getResource("obj.gif"));

                addMouseListener(new MouseAdapter() {
                        public void mousePressed(MouseEvent e) {
                                x = e.getX();
                                y = e.getY();
                                repaint();
                        }
                });
        }

        public void paint(Graphics g) {
                g.drawImage(img, 0, 0, this);
                g.drawImage(img2, x, y, this);
        }
}
```

この場合は、アプレットを継承し、addMouseListener でマウスイベントを処理している。

- 演習１７

白い画面を用意し、マウスでクリックするとその座標に黒い点を表示するアプレットを作成せよ。(点は、drawRect(四角の左上の x 座標, 四角の左上の y 座標, 幅, 高さ)メソッドを使用し、幅と高さを 1 にして処理せよ。)

- 挑戦１４

任意の位置に円が表示され、マウスポインターをその上に移動させると、異なる場所に円が跳ねる(移動する)アップレットを開発せよ。

7. マルチメディア・アプリケーションの開発

マルチメディアは、シングルメディアより効果的に情報を伝達する能力があるため近年のすべてのコンピュータアプリケーションはマルチメディアを利用しているといっても過言ではない。それを可能にした要因としては、コンピュータの処理能力向上、高速通信ネットワークの発達等が上げられる。

本章では本実験を通して体験した内容を統合したマルチメディア・アプリケーションの例を実行してみながら、マルチメディア・アプリケーションを体験してみる。さらに、自由課題として、マルチメディアを扱う、何らかの目的のある応用アプリケーションの開発に挑戦する。

7.1 シミュレーション

シミュレーション(simulation)とは、現実の問題を模擬した装置を準備し、その装置を用いた実験を行い、結果を分析・予測する手法である。実験装置を、コンピュータを用いて作成するものを、特

にコンピュータシミュレーション、シミュレーションを行なうためのツールをシミュレータ(simulator)と呼ぶ。昔は単純にコンピュータの中で計算を行い、結果だけを数値で見せてくれるシミュレーションが一般的であったが、最近のコンピュータシミュレーションはマルチメディア技術を総動員しているため、シミュレーションで結果だけじゃなく途中の過程までもわかりやすく確認できる。（サンプルは、実験のホームページ参考）

- 演習１８（＊）

sample6.1 のソースコードを読んでから実行し，その結果を確認せよ。

7.2 ゲーム

マルチメディアが一番早く浸透した分野の一つがコンピュータゲームである。昔はカラーグラフィックスデバイスが極めて高価であったため、殆どの PC ユーザはモノクロデバイスを利用した。しかし、コンピュータゲームのおかげで一気に世界中にカラーグラフィックスデバイスが広まり、PC にカラーグラフィックスデバイスが標準となるきっかけになった。また、コンピュータゲームは面白さを追求するため、できるだけの手段を使いユーザを満足させようとするが、その中で一番大きい役割をしているのがマルチメディアであり、他のアプリケーションに先立って PC にマルチメディアが普及されるきっかけともなった。コンピュータゲームは殆どが実時間でユーザとインタラクションを行うため、他のマルチメディア・アプリケーションと比べても難易度が一番高いともいえるだろう。（例は実験のホームページ参考）

- 演習１９（＊）

sample6.2 を実行してみよう。

- 挑戦１５

sample6.2 に効果音（自機の発射音、敵の発射音、爆発音）を追加せよ。

7.3 自由課題

画像、音、ユーザ入力(キーあるいはマウス)を全て扱うアプリケーションを開発する。必須ではないが、できるだけマルチスレッドのプログラムにしてほしい。内容は自由だが、レポートにはアプリケーションの目的(何のためのものなのか)を書かなければならないので事前に良く考えてからマルチメディアアプリケーションの制作に取り組もう。