

## 1 課題概要

ドローエディタを作成した。フローチャートが書けること、アニメーションができること、使いやすいことの3点を目指して制作した。作成したドローエディタは次のようなことができる。

- 図形
  - － 様々な図形を描く。
  - － 図形を塗りつぶす
  - － 直線，矢印を描く。
  - － フリーハンドで曲線を描く。
  - － 線の太さを指定できる。
  - － 文字を書く。
  - － 文字のフォントを指定する。
  - － 画像を読み込む。
- 色
  - － 色を選択する。
  - － 色を取得する。
  - － グラデーションをつける。
  - － 背景色を指定する。
- カーソル
  - － カーソルが十字で表示される。
  - － スペースキーでマウスのボタンと同じ操作をする。
  - － 方向キーでカーソルの位置を微調整する。
  - － カーソルの座標を表示する。
- グリッド
  - － グリッドの間隔を指定する。
  - － グリッドを表示する。
- レイヤー
  - － レイヤーの追加，削除をする。
  - － レイヤーの表示，非表示，単独のレイヤーのみの表示をする。
- 編集
  - － 図形を選択する。
  - － 図形を複数選択する。
  - － 図形のグループ化，解除をする。
  - － 図形を削除する。
  - － 図形のカット，コピー，ペーストをする。
  - － 図形を拡大，縮小，回転，移動する。
  - － 操作を戻す，やり直す。
  - － 画像ファイルとして書き出す。
  - － 保存する。あらかじめ保存しておいたデータを開く。
  - － キーボードショートカットで操作する。
- アニメーション

- 変換行列を指定して自由に図形を動かす。

## 2 設計方針

ViewPanel に描画する部分では、「Model-View-Controller」モデルを使用した MVC モデルと呼ばれる。

Model では、主にデータを保持している。Observable クラスを継承することによって、変更された時に View で描画が行われるようになっている。

View では、Model が変更された時に再描画されるようになっている。

Controller では、主に ViewPanel でのイベントを受け取り、Model を操作する。ただし、グリッド関連の変数はここで保持している。

描画する部分以外では View と Controller を分離しないで、M<sub>i</sub> と (V+C) に分離したモデルを使用している。Animation と AnimationDialog, DrawModel と LayerDialog, Pen と ShapeDialog, Pen と ColorDialog, ViewController と CursorDialog がこのモデルになっている。

分離することによってクラスの役割が明確になるため、複数人で開発する際に分業ができる。コードも綺麗にまとまりやすくなる。また、データを保持しているクラスは一つだけであるので、複数の値を更新しなければならないことが少なくなり、更新時エラーを減少させることができる。

次に、Figure クラスの設計について記述する。Figure を変形させるための方法が 3 段階ある。

まず、1 段階目は初期位置を指定する段階である。マウスをドラッグする動作に対応する。この時に左上の点の座標、幅、高さを指定することで、位置と大きさをかえることができる。この時、shape に形が格納される。

2 段階目は shape を描画時にアフィン変換することで変形する段階である。選択した図形を変形するためにドラッグしている時や、アニメーションで図形を動かす時に対応している。

3 段階目は、shape の中身をアフィン変換する段階である。選択した図形を変形し終わった時に対応する。shape を直接変えることで実行時間が短縮できることを期待して実装した。

しかし、Shape クラスは Serializable を実装していないので、変数 affine を記録しておくことでコピーや復元を行った。

次に、デザインについて記述する。

描画するフレームはメニューバーと ViewPanel のみにした。シンプルな方が描きやすいと考えたからである。必要な情報はその都度設定すればよいので、ダイアログで表示、入力させる形式をとった。

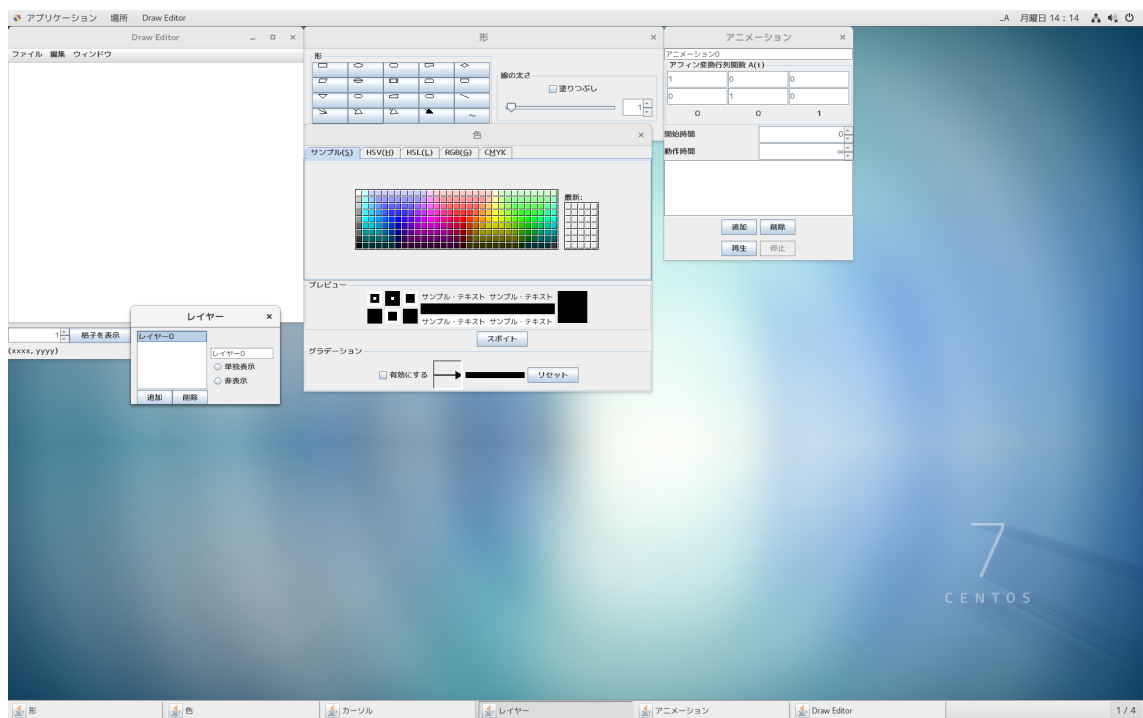


図 1 ドローエディタ全体のスクリーンショット

スペースキーでマウスと同じ操作ができるようにした．マウスをクリックする時に座標がずれてしまうのを防げるため，楽に描くことができる．

アニメーションではより柔軟な移動をできるようにしたかったので，アフィン変換行列を直接与えることができるようにした．

主なクラスのクラス図を図 2 に示した．ここに示したクラスの他に，Figure を継承したクラス，ShapeButton を継承したクラス，Operation を継承したクラスが存在する．

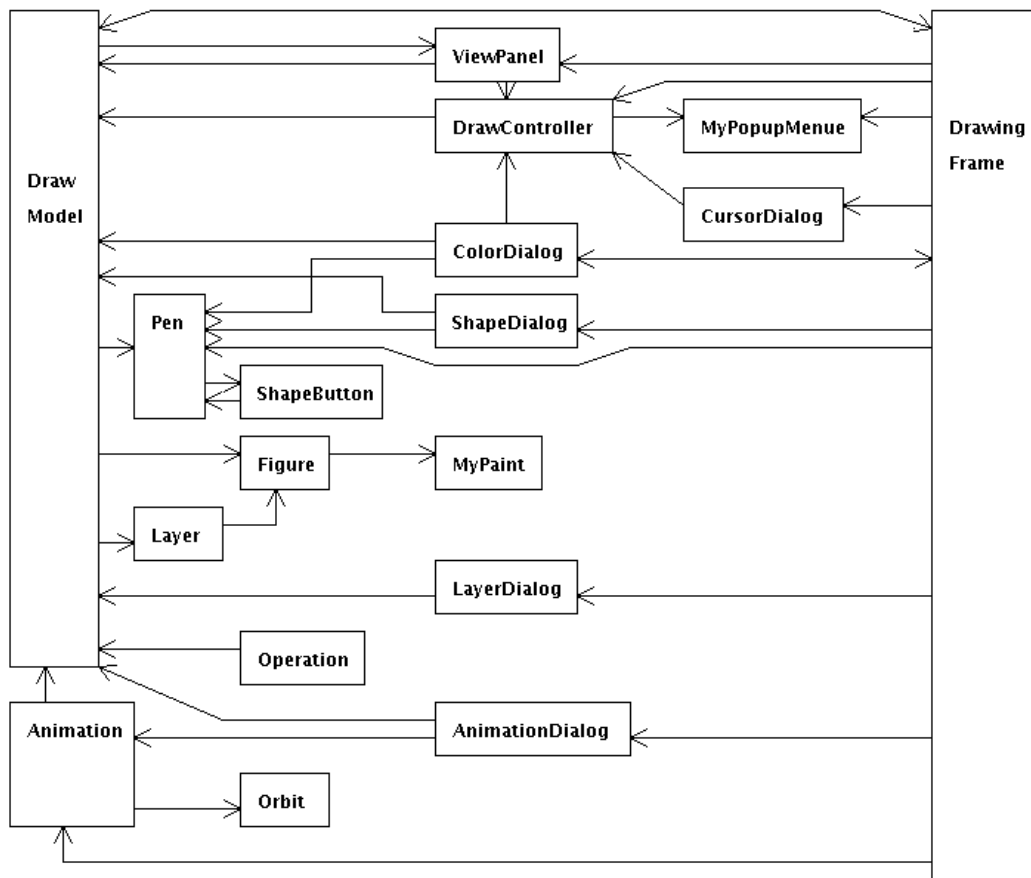


図 2 クラス図

### 3 プログラムの説明

ここでは、図形の初期位置を指定する段階のみ詳しく説明する。

マウスをドラッグすることで初期位置を決めることができる。

まず、マウスのボタンが押されたことを DrawController が検知し、mousePressed が呼び出される。そこから pressed が呼ばれる。スペースキーが押された時も keyPressed が呼び出され、pressed が呼ばれる。pressed では編集モードでないことを確認して、DrawModel の createFigure を呼ぶ。そこから Pen の createFigure が呼ばれる。色の情報が付け加えられ、選択されている ShapeButton から新しい Figure が作成される。作成された Figure は、現在選択されているレイヤーの中の Figure のリストに付け加えられます。また、DrawingModel 内の drawingFigure にも保持されます。

次にマウスがドラッグされている時は、DrawController、DrawModel、Layer、Figure の順に処理が渡されていき、Figure の reshape が呼ばれる。通常はここで makeShape が呼ばれ、Figure 内の Shape が変更される。また、x, y, width, height などの初期位置を決める変数とその都度設定される。

マウスが離された時も、処理が渡されていき、Figure の completed が呼ばれる。通常は何もしないが、文字を表

示させるときには文字列を入力するダイアログを表示させる。

すべてのクラスと、主要な各関数の説明は Javadoc を参照すると良い。

## 4 実行例

まず、様々な形を使ったフローチャートを描いたものを図 3 に示した。

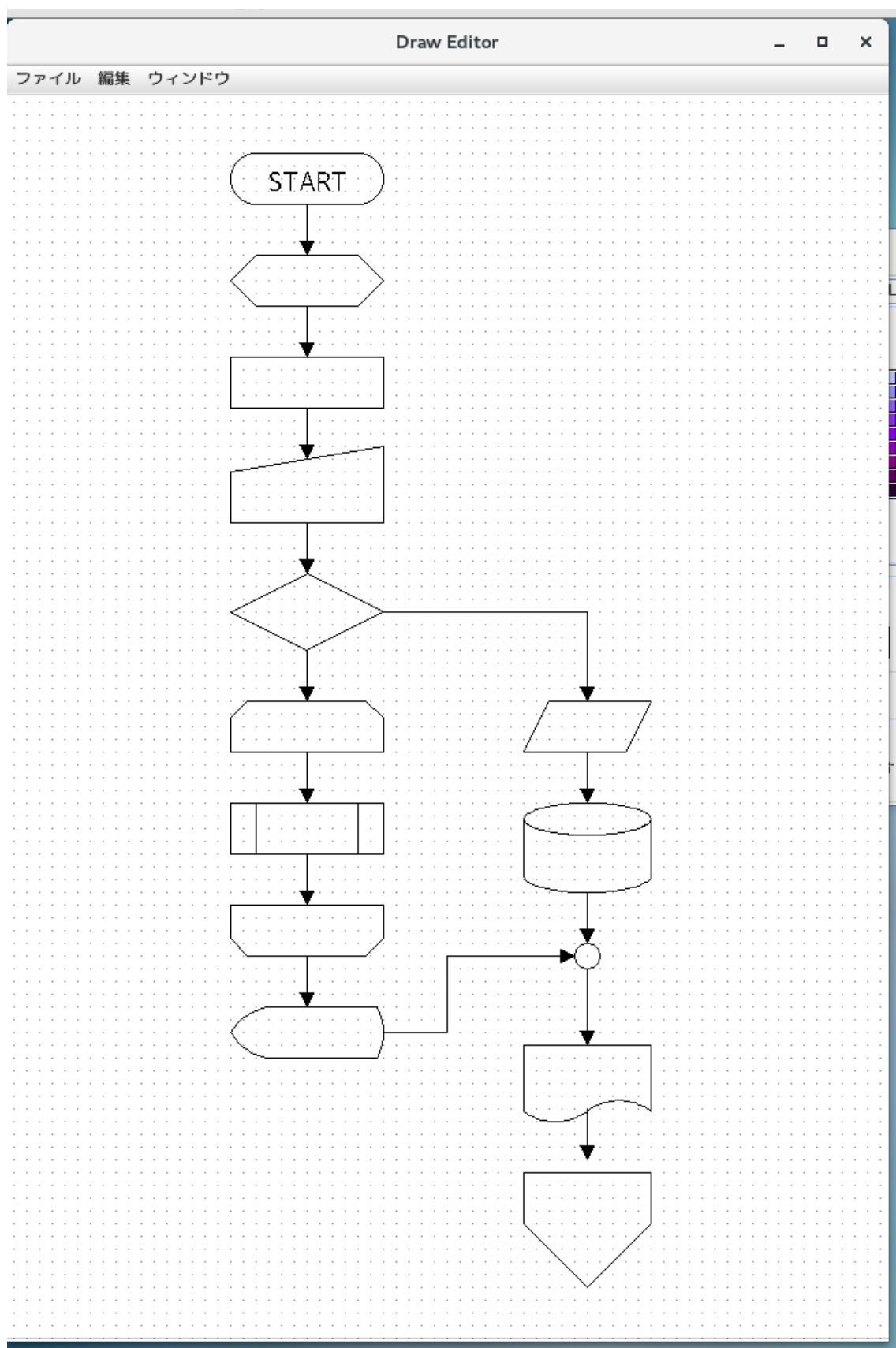


図3 フローチャートを描いた画面

次に図形を複数選択してある様子，コピーアンドペーストをした図形，変形した図形を描いたものを図 4 に示した．

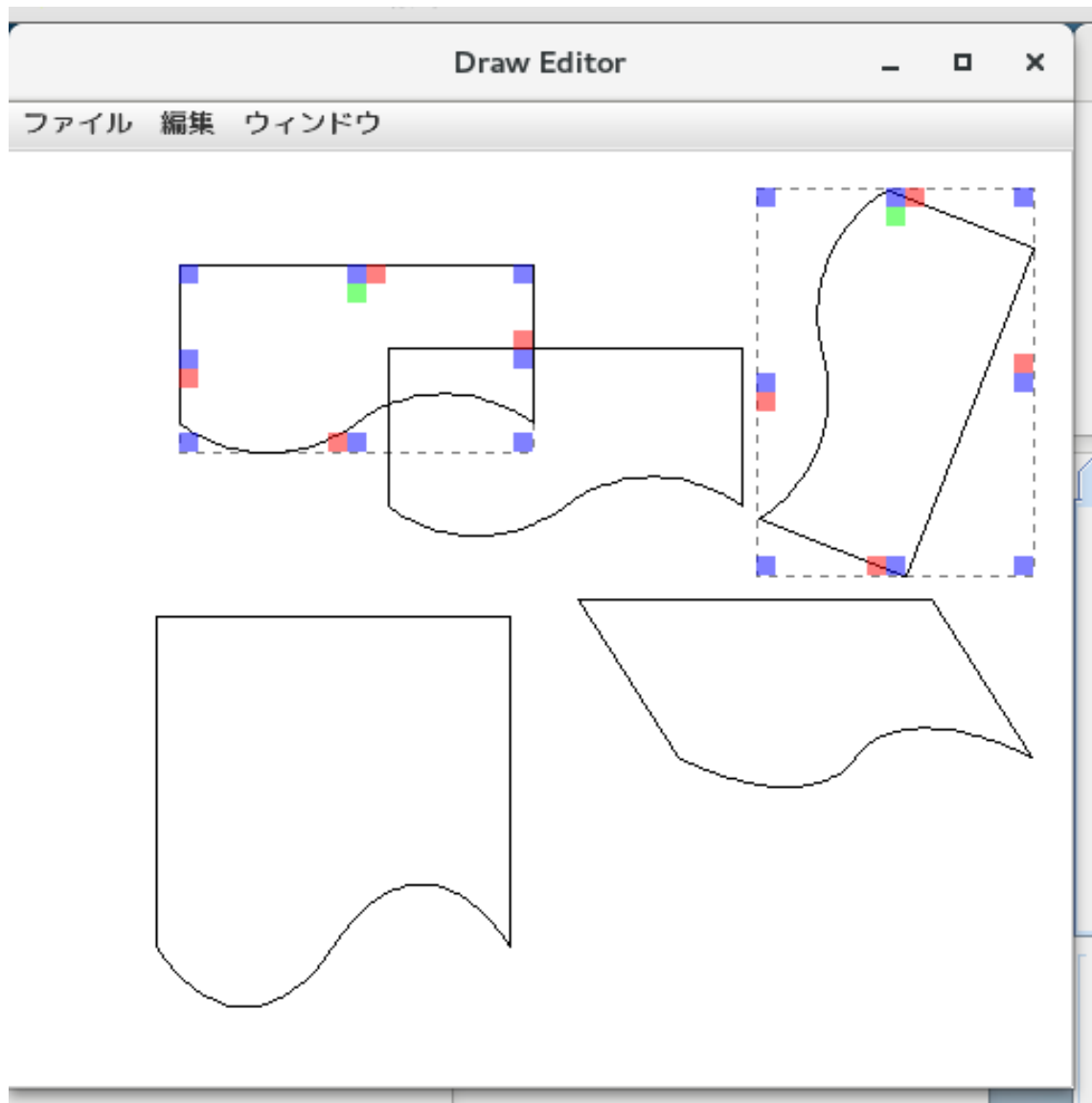


図 4 図形の複数選択と変形

さらに，グラデーション機能を使った実行結果を図 5 に示した．図 6 はその時の設定画面である．

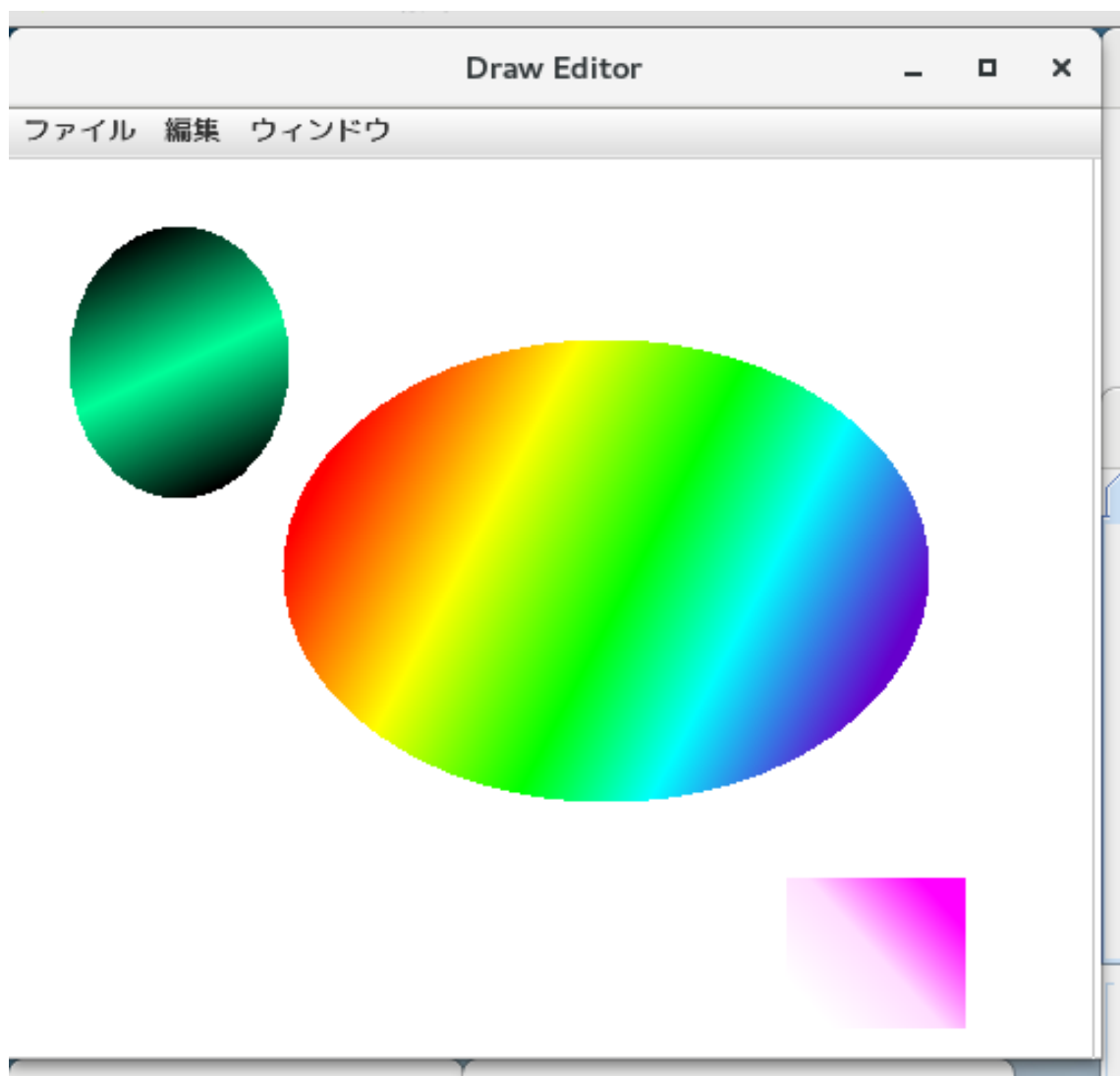


図 5 グラデーション





図 6 グラデーションの設定画面

背景色を変更した様子，アニメーションを設定している画面の様子を図 7 に示した．実際に動かしてみると正常に動いていることがわかる．

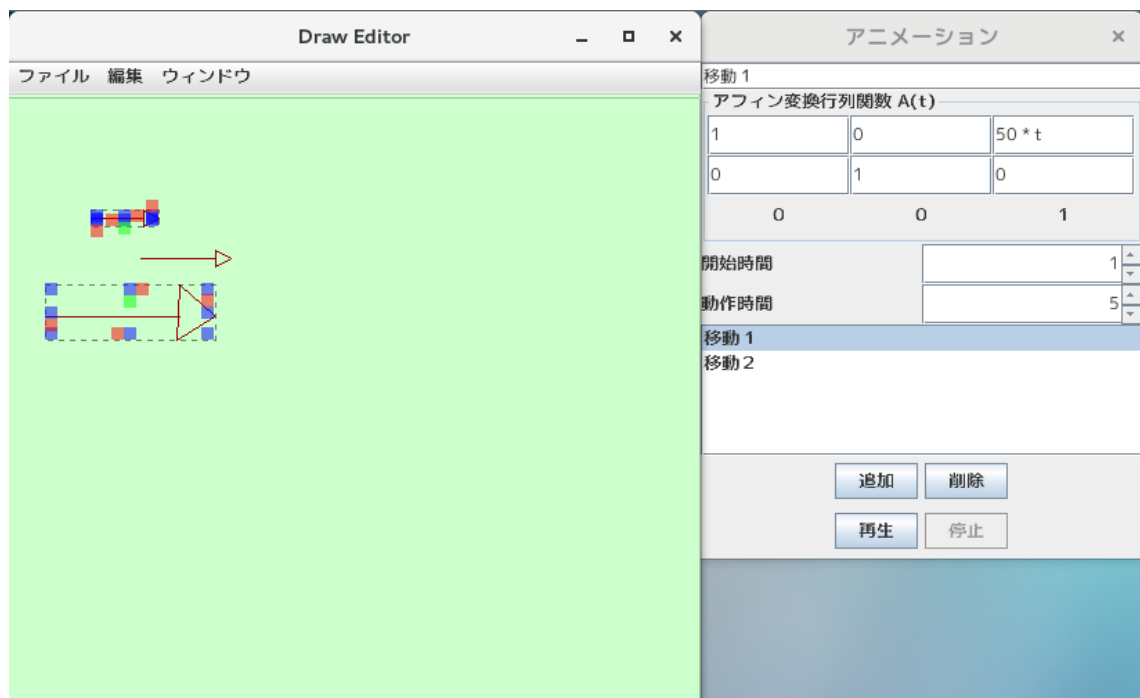


図 7 アニメーションの設定と背景色の変更

## 5 考察

様々な機能を追加していったが、クラスを分けることによって、あまり混乱せずにプログラムを書くことができた。また、Figure や Layer, コンポーネントのようにものとしてクラスを扱うことができることで、コードが描きやすかった。オブジェクト指向の大きな利点であると考える。

例えば Figure 型に RectFigure と OvalFigure を代入してもそれらが区別され、オーバーライドした関数を呼び出すことができた。

DrawFrame に getColor, getAnimation, setAnimation 関数を入れるような構造になってしまった。クラス間でやり取りできるような構造を作ることによって、関係のない場所に関数を書いてしまうことを減らせると考える。

当初のとおり、フローチャートが書けるようになり、アニメーションもできるようになった。また、戻ったり削除したりするような編集作業を行えるようにしたことで使いやすいエディタになった。さらに、選択できるようにすることは予定していなかったが実装できた。逆に、アニメーションをもっと使いやすくしようと思っていたができなかった。

アニメーションの指定が自由にできる反面、直感的ではないので、そこを改良したい。

## 6 感想

Java はこの課題をやる前から触れてはいたが、この課題をやることによって新しく学んだことがあったのでよかった。具体的には、インターフェースの使い方は知っていたが理解はいまひとつであったものが、授業を通してインターフェースやオブジェクト指向の理解が深まった。

J3 課題は楽しく取り組めた。提出期限が短かったことが大変だった。

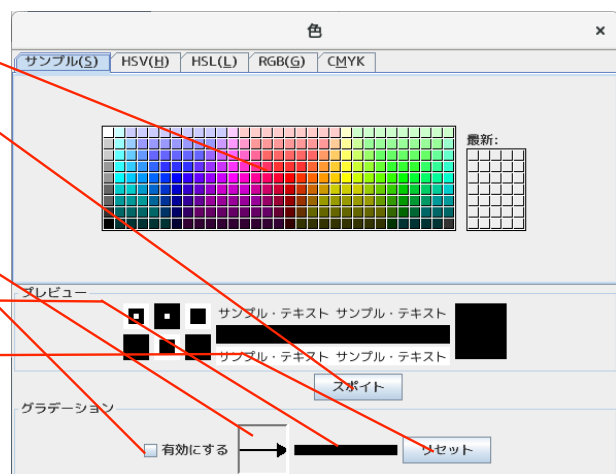
## 7 付録．操作マニュアル

### 基本的な図形の描き方

1. 描きたい形を選択します。
2. 図形を塗りつぶすかどうか指定します。  
塗りつぶさない場合は、線の太さを指定します。



3. 色を指定します。  
パネルから色を取得することもできます。
4. グラデーションを設定することができます。
  - A グラデーションを使用する際には  
チェックを入れてください。
  - B グラデーションの向きを指定します。
  - C 上で色を選択したのちにクリックし、  
グラデーションの色を指定します。
  - D グラデーションを  
再設定する際に、リセットを押します。



5. ビューでドラッグして図形を描きます。

## 編集

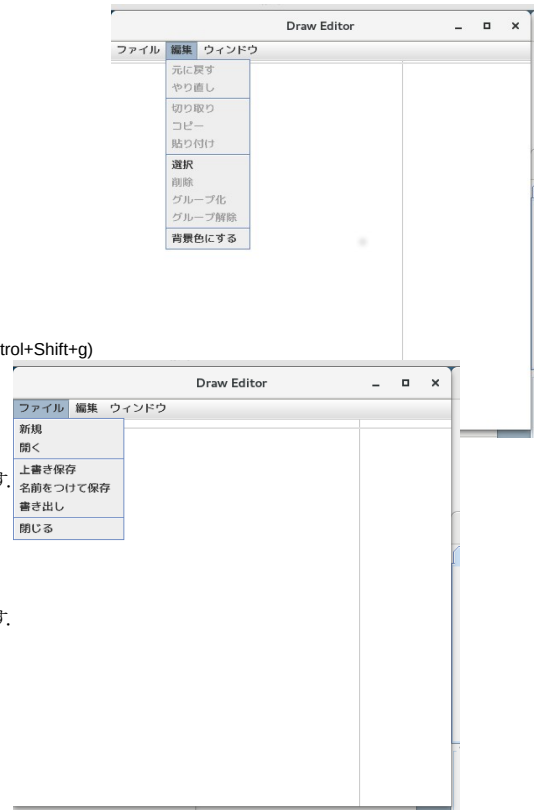
- 元の戻す : 操作を1つずつ戻します, (Control+z)
- やり直し : 戻した操作をやり直します, (Control+Shift+z)
- 切り取り : 選択した図形を切り取ります, (Control+x)
- コピー : 選択した図形をコピーします, (Control+c)
- ペースト : 図形を貼り付けます, (Control+v)
- 選択 : 図形選択モードに切り替えます, (Control+e)
- 削除 : 選択した図形を削除します, (Control+d)
- グループ化 : 選択した図形をグループ化します, (Control+g)
- グループ解除 : 選択したグループ化された図形を解除します, (Control+Shift+g)
- 背景色にする : 選択されている色を背景色にします,

## ファイル

- 新規 : 現在のビューを破棄して初めから描き直せます,
- 開く : 保存されたファイルを開きます,
- 上書き保存 : 上書き保存をします,
- 名前をつけて保存 : 名前をつけて保存します,
- 書き出し : ピクセルデータを画像ファイルに書き出します,
- 閉じる : このドローエディタを閉じます,

## ウィンドウ

- 一度閉じたウィンドウを再度開きます,

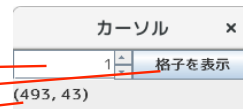


## グリッド

グリッドの間隔を指定できます。

グリッドの表示・非表示を切り替えます。

カーソルの位置座標が表示されます。



カーソル

1 格子を表示

(493, 43)

## レイヤー

レイヤーの一覧が表示されます。

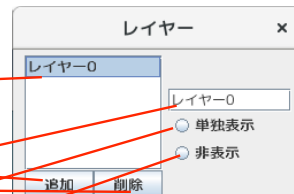
選択されているレイヤーの次に新しいレイヤーを追加します。

選択されているレイヤーを削除します。

レイヤーの名前を設定できます。

選択されているレイヤーのみを表示します。

レイヤーごとに表示・非表示を切り替えられます。



レイヤー

レイヤー0

レイヤー0

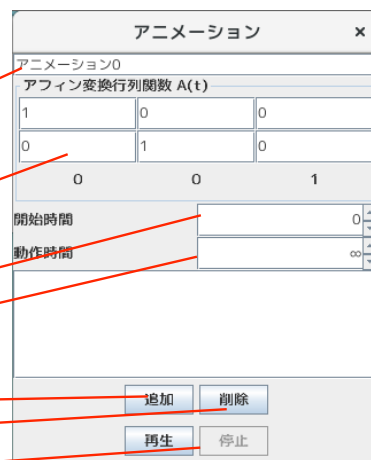
☐ 単独表示

☐ 非表示

追加 削除

## アニメーション

- 動かしたい図形を選択します。
- アニメーションの名前を指定できます。
- アニメーションに適用するアフィン変換行列を時間  $t$  [秒] の関数で入力してください。四則演算のみ入力することができます。
- 再生開始後、動作を開始するまでの時間を指定します。
- 動作している時間を指定します。
- 指定したアニメーションを追加します。
- アニメーションを削除します。
- アニメーションを再生・停止します。



アニメーション

アニメーション0

アフィン変換行列関数  $A(t)$

1	0	0
0	1	0
0	0	1

開始時間 0

動作時間  $\infty$

追加 削除

再生 停止

スペースキーを押すことでクリックするのと同じ動作をします。  
矢印キーでカーソルを移動させることができます。