Chapter 8

PyOpenGLによるグラフィクス

本章以降の章では，PyOpenGL を利用して，主に 3 次元のコンピュータグラフィクス (CG: Computer Graphics) について学ぶ．**PyOpenGL** は，3 次元 CG 用のグラフィクスライブラリである **OpenGL** の Python インタフェースである．OpenGL は，各種の 3 次元 CG の専用ハードウェア，PC の場合には 3 次元グラフィクスプロセサ (GPU: Graphic Processing Unit) の利用を前提としたライブラリで，広く利用されている．従来，OpenGL は C ないし C++言語を用いて使われていたが，Python からも利用できるようにしたものが PyOpenGL である．なお，PyOpenGL は標準の Python 環境には含まれていない．PyOpenGL の利用法については授業のウェブページなどを参考にしてもらいたい．

8.1 OpenGL の関連ライブラリ

PyOpenGL で OpenGL を利用する場合には，グラフィクスハードウェアとの API を提供するグラフィクスハードウェア依存の GL ライブラリ，比較的低レベルではあるがグラフィクスハードウェアとは独立でソフトウェアのみで実現されている GLU ライブラリ，OS などとの API を提供する GLUT ライブラリという 3 種類のライブラリを用いる[1]．ここではウィンドウの作成やイベント処理など，OS との橋渡しとなる GLUT ライブラリについて説明する．

GLUT ライブラリ： OS とのインタフェース

GLUT ライブラリは，オペレーティング・システム（OS）との API を提供し，ウィンドウの作成やイベント処理のためのコールバック関数の登録，メインループの起動など関数からなる．すべての関数名は glut で始まる．GLUT におけるコールバック関数の登録とイベント発生時のプログラム実行の遷移は，図 8.1 に示すようになっている．

`glutInit`　GLUT ライブラリの初期化関数
: GLUT ライブラリを初期化する．

`glutInitDisplayMode`　表示モードの指定関数
: 表示モード（主にグラフィクスメモリの利用法）を指定する．カラー表示（RGB），重畳やブレンディング表示（A），奥行き判定（DEPTH），1 回（静止）表示（SINGLE），交互（アニメーション）表示（DOUBLE）などがある．

`glutInitWindowSize`　ウィンドウサイズの指定関数
: 描画ウィンドウの大きさ（幅と高さ）を指定する．

`glutInitWindowPosition`　ウィンドウ位置の指定関数
: 描画ウィンドウの位置（左上隅の座標）を指定する．

`glutCreateWindow`　ウィンドウの作成関数
: 描画ウィンドウを作成する．引数にタイトルバーの文字列を与えるが，bytes 型文字列（b'...'）にしておく方が安全．

[1] C や C++でも同様だが，Java の場合には少し異なる．

`glutReshapeFunc`　ウィンドウサイズ変更時のコールバック関数の登録
ウィンドウの配置変更や大きさ変更などのイベントにより起動されるコールバック関数を登録する．描画ウィンドウのサイズに合わせた処理を行なう．最初に表示される際にも当該イベントが発生する．

`glutDisplayFunc`　描画要求時のコールバック関数の登録
描画要求イベントにより起動されるコールバック関数を登録する．実際に描画する内容を指定する．

`glutKeyboardFunc`　キーボード入力時のコールバック関数の登録
キーボード入力イベントにより起動されるコールバック関数を登録する．

`glutMouseFunc`　マウスイベント発生時のコールバック関数の登録
マウスボタンプレスなどのイベントにより起動されるコールバック関数を登録する．

`glutMotionFunc`　マウスカーソル移動時のコールバック関数の登録
マウスドラッグなどのマウスカーソル移動に伴うイベントにより起動されるコールバック関数を登録する．

`glutIdleFunc`　一定時間経過時のコールバック関数の登録
一定時間の経過で自動的に発生するイベントにより起動されるコールバック関数を登録する．アニメーション表示などに利用される．

`glutMainLoop`　実行ループの起動関数
実行ループを起動する関数で，イベントが発生すると登録されたコールバック関数を呼び出される．

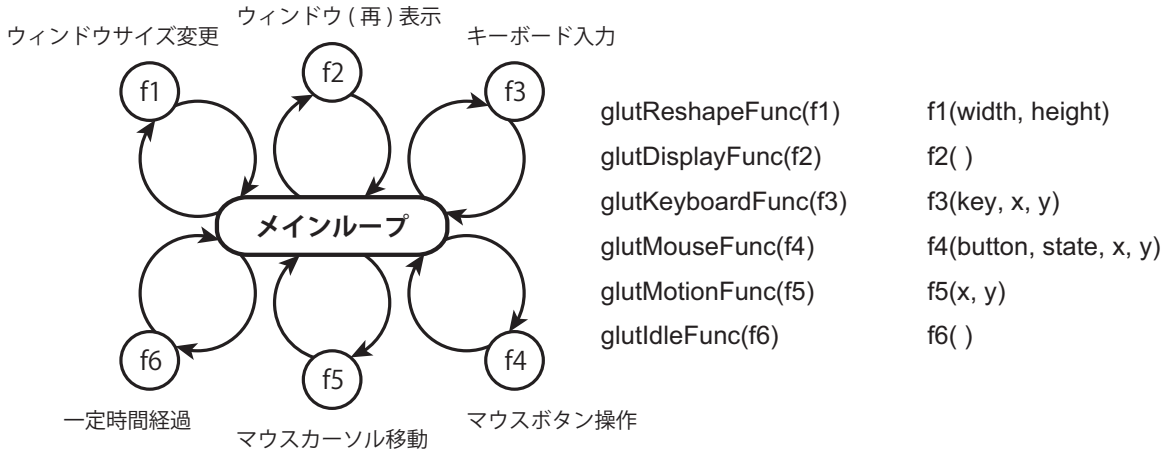

Figure 8.1: GLUT におけるコールバック関数によるイベント処理

8.2 OpenGL による描画

OpenGL では，頂点の集まりとして図形の描画を行なう．プログラムでは，`glBegin(`図形要素の種類`)` と `glEnd()` の間に，頂点，また場合によって色や法線などを与えて図形を構築する．OpenGL で描画できる基本的な図形要素の種類は，表 8.1 のように定められている．それぞれの図形要素の説明図を図 8.2 に示す．

`GL` ライブラリ： OpenGL 命令のインタフェース
OpenGL の命令 (C 言語インタフェースでの関数) は，`gl` で始まる関数によって提供される．

`glBegin`　描画命令の開始関数
以降の `glEnd` までに指定される頂点を描画する．引数として描画要素の種類を与える．

`glEnd`　描画命令の終了関数
`glBegin` からの一連の描画命令の終了を示す．

Table 8.1: OpenGL の図形要素

点	GL_POINTS	与えられた点の集合
線	GL_LINES	頂点 2 つで 1 組の線分の集合
	GL_LINE_STRIP	開いた折れ線
	GL_LINE_LOOP	閉じた折れ線 (ループ)
面	GL_TRIANGLES	頂点 3 つで 1 組の三角形の集合
	GL_QUADS	頂点 4 つで 1 組の四角形の集合
	GL_POLYGON	多角形
	GL_QUAD_STRIP	四角形 (梯子風) のリボン形状
	GL_TRIANGLE_STRIP	三角形 (互い違い) のリボン形状
	GL_TRIANGLE_FAN	三角形の扇型形状

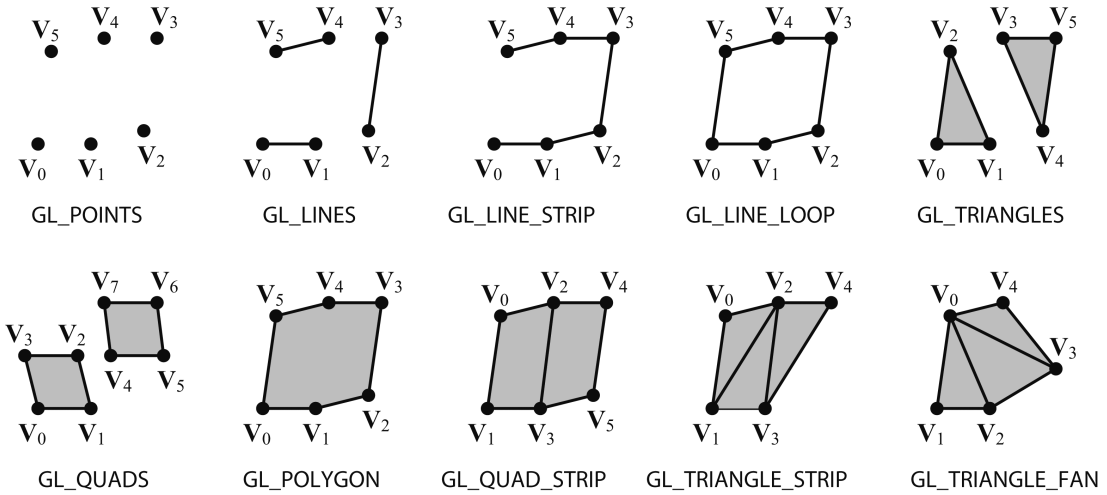

Figure 8.2: OpenGL の図形要素

`glVertex{2,3,4}{i,f,d}[v]`　頂点情報の指定関数
　描画要素の頂点情報を指定する．各文字は，以下の意味を表す．

```
2,3,4          : 2 次元ベクトル，3 次元ベクトル，4 次元ベクトル
i,f,d          : int, float, double
v              : 配列利用の有無 (配列の場合は offset 値を与える)
```

`glColor{3,4}{i,f,d}[v]`　色情報の指定関数
　描画要素の色情報を指定する．各文字の表す意味は，`glVertex` と同様である．

`glNormal{3,4}{i,f,d}[v]`　法線ベクトル情報の指定関数
　描画要素の法線ベクトル情報を指定する．各文字の表す意味は，`glVertex` と同様である．

`glFlush`　描画命令の強制実行関数
　バッファリングされた描画命令を強制的に実行する．

8.3 PyOpenGL による 2 次元グラフィクスのプログラム

PyOpenGL を用いて，第 2 章の例 1「直線群の描画」プログラム `lines.py` と同様の直線を描画する．

例１： PyOpenGL による直線群の描画 (2 次元) — `lines.py`
　PyOpenGL を利用した直線描画プログラムである．GLUT ライブラリによって，描画領域を確保する（`window` 関数）とともに，描画関連のイベントのコールバック関数を登録している（`loop` 関数）．`init` 関数では背景色と描画色を設定している．`reshape` 関数では描画座標系の設定を行なっているが，これ

は 2 次元グラフィクス用の特殊な設定となっている．描画座標系の設定については，第 10 章の「3 次元グラフィクスの基礎」で詳しく説明する．描画内容は display 関数において記述されており，図形要素として線分の GL_LINES が指定されている．図 8.3 のように，Tkinter を用いた 2 次元グラフィクスとは座標系が異なっていることに注意してほしい．

```
import sys                                  # sys モジュールの import
from OpenGL.GL import *                     # GL モジュールの import
from OpenGL.GLU import *                    # GLU モジュールの import
from OpenGL.GLUT import *                   # GLUT モジュールの import

W, H   = (600, 600)                         # OpenGL ウィンドウの幅と高さ
points = ((W/60, H-1), (W/20,  H-1), (W/8,  H-1), (W/4, H-1),
          ( W/2, H-1), ( W-1,  H-1), (W-1,  H/2), (W-1, H/4),
          ( W-1, H/8), ( W-1, H/20), (W-1, H/60)) # 線分の終点, 11 個

def window(width, height):                  # ウィンドウ作成
  '''
  width  - OpenGL ウィンドウの幅
  height - OpenGL ウィンドウの高さ
  GLUT を初期化して，OpenGL ウィンドウを作成する
  '''
  glutInit(sys.argv)                        # GLUT の初期化
  glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE) # 表示モードの指定
  glutInitWindowSize(width, height)         # ウィンドウサイズの指定
  glutInitWindowPosition(0, 0)              # ウィンドウ位置の指定
  glutCreateWindow(b'OpenGL')               # ウィンドウの作成

def init():                                 # OpenGL の初期化
  '''
  OpenGL を初期化する
  '''
  glClearColor(1, 1, 1, 1)                  # 背景色の設定（白）

def reshape(width, height):                 # ウィンドウのサイズ変更に伴うコールバック関数
  '''
  width  - 変更後の OpenGL ウィンドウの幅
  height - 変更後の OpenGL ウィンドウの高さ
  ウィンドウサイズ変更に伴う処理を行う
  '''
  glViewport(0, 0, width, height)           # ビューポートの設定
  glMatrixMode(GL_PROJECTION)               # 投影変換行列の設定開始
  glLoadIdentity()                          # 恒等行列での初期化
  gluOrtho2D(0, width-1, 0, height-1)       # 2 次元ワールド座標系との対応
  glMatrixMode(GL_MODELVIEW)                # モデル変換行列の設定開始
  glLoadIdentity()                          # 恒等行列での初期化

def display():                              # 描画要求に伴うコールバック関数
  '''
  描画要求に伴う処理を行う（放射状の線を描画する）
  '''
  origin = (0, 0)                           # 線分の始点（左下隅）
  glClear(GL_COLOR_BUFFER_BIT)              # 背景の消去
  glBegin(GL_LINES)                         # 線分描画の開始
  glColor3d(0, 0, 0)                        # 描画色の設定（黒）
  for i in range(len(points)):              # 線分描画の反復（終点の個数分）
    glVertex2dv(origin)                     # 線分の始点
    glVertex2dv(points[i])                  # 線分の終点
  glEnd()                                   # 線分描画の終了
  glFlush()                                 # 描画命令の送信

def loop():                                 # コールバック関数の設定とループ起動
  '''
  reshape と display コールバック関数を設定し，ループを起動する
  '''
  glutReshapeFunc(reshape)                  # reshape コールバック関数の登録
```

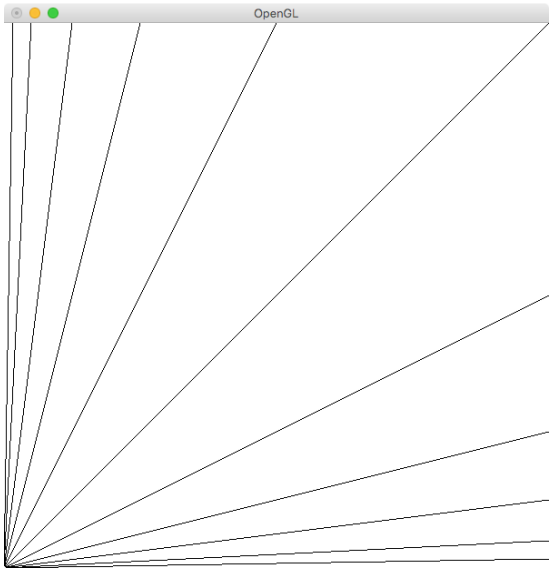

Figure 8.3: PyOpenGL による直線群の描画

```
  glutDisplayFunc(display)                # display コールバック関数の登録
  glutMainLoop()                          # GLUT のメインループ起動

def main():                               # main 関数
  window(W, H)                            # ウィンドウの作成
  init()                                  # OpenGL の初期化
  loop()                                  # コールバック関数の設定とループ起動

if __name__ == '__main__':                # 起動の確認（コマンドラインからの起動）
  main()                                  # main 関数の呼出
```

例２： ピクセルによる直線群の描画 — dotLines.py

例 1 と同じ直線群を描くプログラムであるが，display 関数において点の GL_POINTS を用い，ピクセル単位で描画する部分が異なっている．基本的に第 2 章の例 2「ピクセルによる直線群の描画」dotLines.py と同じことを PyOpenGL で実行している．

```
from OpenGL.GL import *                   # GL モジュールの import
from OpenGL.GLUT import *                 # GLUT モジュールの import
from lines import window, init, reshape, W, H, points # lines モジュールの import

def display():                            # 描画要求に伴うコールバック関数
  '''
  描画要求に伴う処理を行う（放射状の線を点で描画する）
  '''
  glClear(GL_COLOR_BUFFER_BIT)            # 背景の消去
  glBegin(GL_POINTS)                      # 点描画の開始
  glColor3d(0, 0, 0)                      # 描画色の設定（黒）
  for i in range(len(points)):            # 線分描画の反復（終点の個数分）
    if points[i][0] >= points[i][1]:      # x >= y（横長の直線）
      n = int(points[i][0]) + 1           # 表示ピクセルの個数: n = x+1
      d = points[i][1] / points[i][0]     # 1 ピクセルごとの差分
      for x in range(n):                  # ピクセル描画の反復 n 回
        glVertex2i(x, int(d*x+0.5))       # 1 ピクセルの描画
    else:                                 # x < y（縦長の直線）
      n = int(points[i][1]) + 1           # 表示ピクセルの個数: n = y+1
      d = points[i][0] / points[i][1]     # 1 ピクセルごとの差分
      for y in range(n):                  # ピクセル描画の反復 n 回
        glVertex2i(int(d*y+0.5), y)       # 1 ピクセルの描画
  glEnd()                                 # 点描画の終了
  glFlush()                               # 描画命令の送信
```

```
def loop():                                 # コールバック関数の設定とループ起動
  '''
  reshape と display コールバック関数を設定し，ループを起動する
  '''
  glutReshapeFunc(reshape)                  # reshape コールバック関数の登録
  glutDisplayFunc(display)                  # display コールバック関数の登録
  glutMainLoop()                            # GLUT のメインループ起動

def main():                                 # main 関数
  window(W, H)                              # ウィンドウの作成
  init()                                    # OpenGL の初期化
  loop()                                    # コールバック関数の設定とループ起動

if __name__ == '__main__':                  # 起動の確認（コマンドラインからの起動）
  main()                                    # main 関数の呼出
```