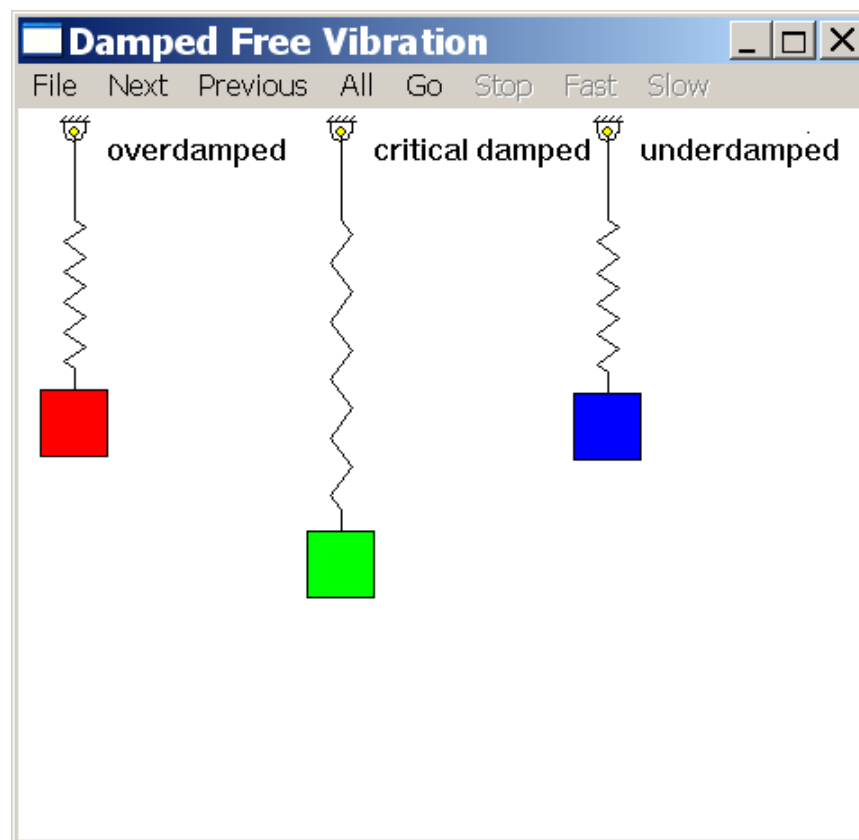


QuickAnimation™

バージョン 2.3

ユーザーズガイド



SoftIntegration の連絡先

住所 SoftIntegration, Inc.
216 F Street, #68
Davis, CA 95616
電話 + 1 530 297 7398
Fax + 1 530 297 7392
Web <http://www.softintegration.com>
メール info@softintegration.com

Copyright ©2004-2009 by SoftIntegration, Inc. All rights reserved.
December, 2009

著作権者の文書による事前の許可なしには、この著作物のいかなる部分もどのような形式またはどのような方法によってであれ、複写、検索システムへの保存、または送信してはなりません。

SoftIntegration, Inc. はこの文書に記載された QuickAnimationTM の著作権者です。

SoftIntegration, Inc. makes no representations, expressed or implied, with respect to this documentation, or the software it describes, including without limitations, any implied warranty merchantability or fitness for a particular purpose, all of which are expressly disclaimed. Users should be aware that included in the terms and conditions under which SoftIntegration is willing to license QuickAnimationTM as a provision that SoftIntegration, and their distribution licensees, distributors and dealers shall in no event be liable for any indirect, incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of QuickAnimationTM, and that liability for direct damages shall be limited to the amount of purchase price paid for QuickAnimationTM.

Ch, SoftIntegration, One Language for All, and QuickAnimation are either registered trademarks or trademarks of SoftIntegration, Inc. in the United States and/or other countries. Microsoft, MS-DOS, Windows, Windows 95, Windows 98, Windows Me, Windows NT, Windows 2000, and Windows XP are trademarks of Microsoft Corporation. Solaris and Sun are trademarks of Sun Microsystems, Inc. Unix is a trademark of the Open Group. HP-UX is either a registered trademark or a trademark of Hewlett-Packard Co. Linux is a trademark of Linus Torvalds. All other trademarks belong to their respective holders.

目次

1	オブジェクトの表示とアニメーションのための QuickAnimation	1
1.1	はじめに	1
1.2	QuickAnimation TM のユーザーインターフェース	2
1.3	入力データ形式	2
1.3.1	一般描画プリミティブ	3
1.3.2	力学的描画プリミティブ	6
1.4	QuickAnimation TM の使用例	8
1.4.1	例 1: データ形式	8
1.4.2	例 2: 減衰自由振動の位置の表示	10
1.4.3	例 3: 減衰自由振動のアニメーション	15
2	オブジェクトの Web ベースの表示とアニメーション	21
2.0.4	CGI スクリプトファイルの記述	21
2.0.5	Web サーバーの構築とセットアップ	21
	索引	22

1 章

オブジェクトの表示とアニメーションのための QuickAnimation

1.1 はじめに

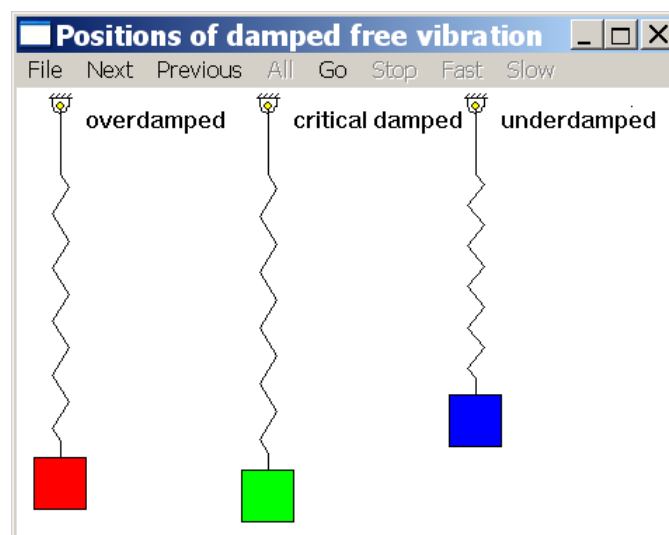


図 1.1: 3つの振動系の位置を示す QuickAnimation ウィンドウ

QuickAnimationTM は指定した x-y 座標データに基づき、様々なオブジェクトのクイックアニメーションと表示を行うためのプログラムです。他の SoftIntegration 製品と同様、簡潔さと使いやすさがこのユーティリティプログラムの特徴となっています。QuickAnimationTM は、二次元力学系のアニメーションに特に適しています。たとえば、図 1.1 に示されている QuickAnimationTM ウィンドウでは、メニューバーと過減衰 (overdamped)、臨界減衰 (critical damped)、および減衰振動 (underdamped) 系が表示されています。表示オブジェクトは、上にタイトルが記載されたウィンドウ内に大きく描画されています。アニメーションのための `qanimate` の QuickAnimationTM プログラムの使用方法の詳細については、この章で記述されています。

1.2 QuickAnimationTM のユーザーインターフェース

オブジェクトに対する QuickAnimationTM データファイルの拡張子は .qnm です。このデータファイルは、QuickAnimationTM コマンド **qanimate** により、

```
qanimate datafile.qnm
```

のように起動し、図 1.1 に示されるような QuickAnimationTM ウィンドウを開くことができます。

QuickAnimation ウィンドウ内のメニューバーは、アニメーションシステムを操作する一連のメニューを含んでいます。File メニューを使用すると、プログラムを終了することができます。Next ボタンおよび Prev ボタンはアニメーションのフレームを制御し、All ボタンは全フレームを一度に表示します。Fast および Slow ボタンはアニメーションの速度を変更します。Go および Stop ボタンはアニメーションを開始および停止します。システムは、Prev ボタンで一方向、Next ボタンで逆方向のように、どちらの方向にも動かすことができます。Go ボタンが押されると、システムは Prev または Next ボタンで直前に指定された方向に動きます。

1.3 入力データ形式

QuickAnimation データファイルの典型的な形式は図 1.2 で示されています。この形式は、以下の表記で指定されます。

- Typewriter 書式のテキストはキーワードです。
- *Emphasized* 書式のテキストはユーザーにより指定されるものです。
- 角かっこ '[']' 内のテキストはオプションです。
- 縦棒文字 '|' は "OR" (論理和) 条件を指定します。

行の最初の '#' 文字は、コメントの境界を定めています。QuickAnimation はこの行上の '#' 文字に続く文字をすべて無視します。力学系のタイトルは、title キーワードに続く二重引用符 '"' で囲まれた文字列で指定されます。fixture キーワードを使用すると、以降の行で固定オブジェクトを定義可能です。primitives は、表示され、またはアニメーション化されるシステムの一般および力学コンポーネントの定義に使用されます。animate は、アニメーション用のデータの入力を開始します。animate キーワードに続く各行は、primitive 上のスーパースクリプトで示されるような、アニメーションの 1 つのフレームを表します。restart オプションは、アニメーション終了時にそれを最初から再開することを指定します。reverse オプションは、アニメーション終了時にそれを最後のフレームから最初のフレームまで逆方向に動作させることを指定します。stopped キーワードに続く primitives は、アニメーション停止時にのみ表示されます。フレームは、複数のプリミティブを含むことが可能なデータセットから構成されます。継続文字 '\' は、複数行にわたるデータセットにおけるプリミティブ間の橋渡しに使用されます。 n 個のフレームをもつアニメーションには、 n 個のデータセットが必要とされます。

```

# comment
title "title string"
fixture
primitives
animate [ restart | reverse ]
primitives1 [ stopped primitives1 ]
primitives2 [ stopped primitives2 ]

.
.
.

primitivesn [ stopped primitivesn ]

```

図 1.2: QuickAnimation データ形式

1.3.1 一般描画プリミティブ

図 1.3 は、QuickAnimation 用に使用可能な一般描画プリミティブを示しています。これらのプリミティブを使用すると、QuickAnimation プログラムへのテキスト (Text) の挿入はもちろん、弧 (Arc)、直線 (Line)、セグメント (Segment)、円 (Circle)、多角形 (Polygon)、および矩形 (Rectangle) も描画することができます。これらのプリミティブを描画する構文は、図 1.4 で示されています。ひとつの例として、直線の描画を考えます。line の次に始点と終点の x 、 y 座標を指定した直線を定義します (すなわち、line 0 0 2 3 は、直交座標系で原点から点 (2,3) への直線を描画します)。複数の直線は、line ステートメントの後にさらに座標点を追加することにより、互いに連結されます。同様に、円は図 1.4 に示されている構文にしたがいその中心点と半径を指定することにより、描画されます。各一般描画プリミティブで使用可能な様々なオプションは、図 1.5 内で表示され、色とフォントのオプションの例は図 1.6 で示されています。

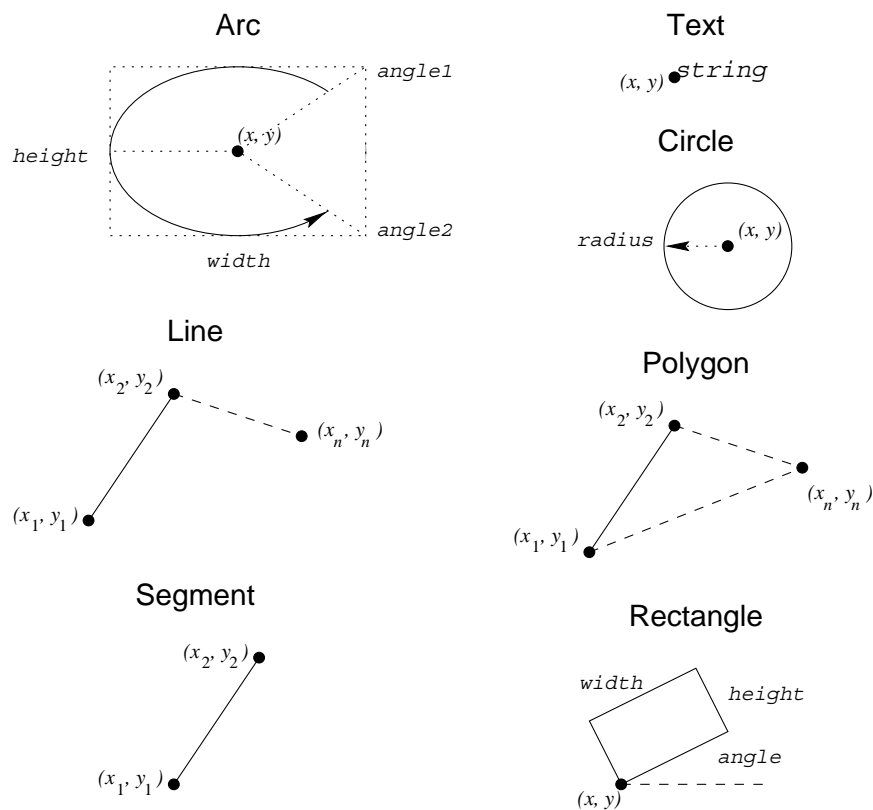


図 1.3: 一般描画プリミティブのグラフィック表現

```

line  $x_1 y_1 x_2 y_2$  [ ...  $x_n y_n$  ]
arc  $x y$  width height angle1 angle2
segment  $x_1 y_1 x_2 y_2$ 
rectangle  $x y$  width height [ angle angle ]
polygon  $x_1 y_1 x_2 y_2 x_3 y_3 \dots x_n y_n$ 
text  $x y$  string
circle  $x y$  radius
dot  $x y$ 

```

図 1.4: 一般描画プリミティブの構文

```

line
segment    ...
            [ pen color ]
            [ linewidth pixelwidth ]
            [ linestyle solid |
              dashed [ length pixellength ] |
              dotted [ gap pixelgap ] ]
            [ capstyle butt | round | projecting ]
            [ joinstyle miter | round | bevel ]
            [ depth depth ]

arc
circle
polygon
rectangle ...
            [ pen color ]
            [ fill color [ intensity percent ]
              [ pattern number ] ]
            [ linewidth pixelwidth ]
            [ linestyle solid |
              dashed [ length pixellength ] |
              dotted [ gap pixelgap ] ]
            [ capstyle butt | round | projecting ]
            [ joinstyle miter | round | bevel ]
            [ depth depth ]

text
...
            [ pen color ]
            [ depth depth ]
            [ font fontname ]

dot
...
            [ pen color ]
            [ depth depth ]

```

図 1.5: 一般描画プリミティブのオプション

```

... color { red | blue | yellow | white | black | grey90 ... }
... font { fixed | 6x13 | 6x13bold | lucidasanstypewriter-12 ... }
}

```

図 1.6: 色とフォントのオプションの例

色とフォントは、Unix の X Window System および Windows で指定されます。Linux では、色の名前の一覧はディレクトリ `/usr/X11R6/lib/X11` にあるファイル `rgb.txt` の中に見出されます。Solaris では、このファイルは `/usr/openwin/lib` にあります。

1.3.2 力学的描画プリミティブ

力学的描画プリミティブは、力学系のばねや連結点のような典型的な力学的コンポーネントを簡単に作成できるように QuickAnimation に同梱されています。QuickAnimation で使用可能な力学的描画プリミティブは、一般描画プリミティブから派生しています。たとえば、リンクは直線で接続された2つの円の組み合わせです。使用可能なすべての力学的描画プリミティブは、図 1.7 に示されています。これらのプリミティブは、力学系のアニメーション作成に使用される基本的ツールです。

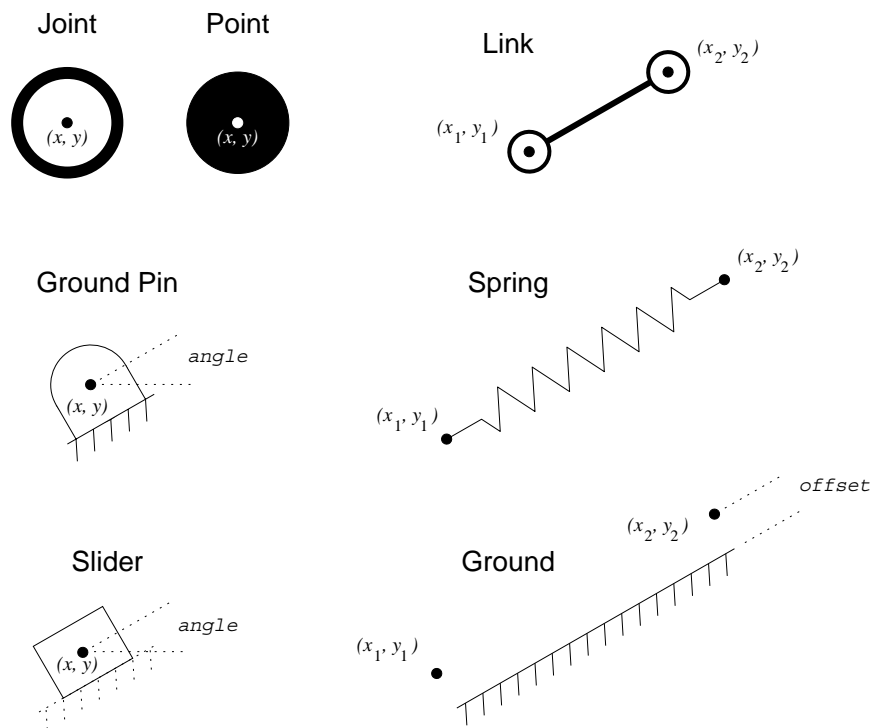


図 1.7: 力学的描画プリミティブのグラフィック表現

点 (Point) と連結点 (Joint)

point プリミティブは、基本的には中塗りされた円です。これは通常、力学系上の点を強調するために使用されます。point の一般的構文は、

```
point x1 y1 [x2 y2 ... xn yn] [trace]
```

であり、ここで $x_1 y_1 \dots x_n y_n$ は連結点の座標を指定し、trace はオプションのパラメータで、この点がアニメーションの最中にトレースされるかどうかの指定に使われます。たとえば、トレースされる、座標 (1,3) の点を作成するには以下のコマンドを指定します。

```
point 1 3 trace
```

プリミティブ joint は point と非常によく似通っています。これは point プリミティブと構文上は同じですが、中塗りされていない円で構成されています。連結点は、2つのリンクまたは他の力学的コンポーネント間の接続を表します。

リンク (Link)

すでに述べたように、link プリミティブは2つの円プリミティブと1つの直線プリミティブで形成される力学的コンポーネントです。このプリミティブは通常、平面4節機構のような力学的リンクのアニメーション作成に使用されます。link の一般的構文は以下のとおりです。

```
link  $x_1$   $x_2$   $x_2$   $y_2$  [...  $x_n$   $y_n$ ].
```

最初のリンクの端点座標は、 (x_1, y_1) と (x_2, y_2) で指定されます。追加リンクは、そのリンクの他方の端点座標を指定することにより、最後のリンクに加えることができます。共通の端点で隣接する2つのリンクを作成する典型的な例として、以下のものがあります。

```
link 1 1 1 4 4 4
```

この例では、最初のリンクの端点は座標 (1,1) と (1,4) です。そして次のリンクは (1,4) で最初のリンクに追加され、その他方の端点は (4,4) です。QuickAnimation の実行中には、端点座標間の余分なスペースは無視されることに注意してください。この例でのこれらのスペースは、各端点を区別しやすくするために入れられています。

平面 (Ground)

ground プリミティブはアニメーションの参照領域を表します。このプリミティブは静止していて、その位置は固定されています。ground の構文は以下のとおりです。

```
ground  $x_1$   $y_1$   $x_2$   $y_2$  [offset pixeloffset]
      [ticks forward | backward]
```

オプション offset、pixeloffset は、この平面の x と y 座標に対して、この平面が離れて配置されている距離を指定します。さらに、ticks オプションが使用され、その値が forward である場合、この平面上の順方向は (x_1, y_1) から (x_2, y_2) であることを示しています。同様に、ticks の値が backward である場合は、その逆になります。オプション ticks の既定値は forward です。たとえば、

```
ground 0 0 10 0 offset 2
```

は、 $x=0$ から $x=10$ の方向の、 $y=0$ の線から 2 単位下の ground セクションを作成します。

平面ピン (Ground Pin)

力学系を直接 ground に接続するために、groundpin プリミティブが使用され、要求する接続を指定します。このプリミティブの構文は以下のとおりです。

```
groundpin  $x$   $y$  [angle angle]
```

座標 (x, y) はこの平面ピンの中心であり、オプション引数の angle angle は水平位置からの角度を度単位で指定します。原点上に、 45° の回転角度をもつ平面ピンを作成するには、以下のステートメントを宣言します。

```
groundpin 0 0 angle 45
```

スライダー (Slider)

slider プリミティブは、矩形描画プリミティブから生成されます。このプリミティブは、並進運動のみが可能な力学系のブロックメンバを表します。平面ピンと同様に、スライダーは、傾斜した表面を平行移動できるように、角度を指定することができます。この構文は以下のとおりです。

```
slider xy [angle angle]
```

例として、(3,4) の位置にある、平面と 30° の傾斜面をすべるスライダーを考えます。このメカニズムのスライダー部分は、以下のステートメントで作成します。

```
slider 3 4 angle 30
```

ばね (Spring)

ばねは多くの力学系の代表的なコンポーネントです。QuickAnimation における spring プリミティブにより、モデル化し、アニメーション化可能な力学系の数を飛躍的に増やすことができます。この構文は以下のとおりです。

```
spring  $x_1$   $y_1$   $x_2$   $y_2$ 
```

ここで、座標 (x_1, y_1) と (x_2, y_2) は、このばねの端点を指定します。(1,1) から (3,5) のばねを作成するには、QuickAnimation データファイル内に以下のステートメントを入れます。

```
spring 1 1 3 5
```

1.4 QuickAnimation™ の使用例

これらのサンプルのソースコードは Ch で提供され、CHHOME/demos/qanimate ディレクトリ (Windows では C:/Ch/demos/qanimate、Unix では /usr/local/ch/demos/qanimate) に置かれています。

1.4.1 例 1: データ形式

図 1.8 におけるデータファイルは、QuickAnimation ファイル内の一般および力学的プリミティブをどのように指定するかを示しています。図 1.9 はこのデータファイルが qanimate プログラムで実行されたときの表示を示しています。

```

# this is a comment
title "This is a Title"

fixture
#no fixture

animate

# low level primitives:
line 0 0 1 1.5 2 2 pen red \
    line 3 3 4 4
line 5 5 12 5 linestyle dashed length 2 pen green linewidth 1
line 5 6 12 6 linestyle dashed length 5 pen green linewidth 1
line 5 7 12 7 linestyle dotted gap 1 pen red linewidth 2
line 5 8 12 8 linestyle dotted gap 5 pen red linewidth 2
arc 11 11 4 4 0 270 fill grey90 linewidth 5
arc 12 12 10 11 0 90 13 13 5 5 0 360 linewidth 2 pen blue
segment 14 14 15 15 16 16 17 17 pen red
#color of text cannot be changed in Windows for now
text 18 5 string1 pen rgb:ffff/ffff/0
text 18 7 "This is a string2" pen red
text 18 9 "This is a string3" \
    font *-lucidatypewriter-medium-***-12-***-***-***
circle 22 16 2 \
    stopped line 14 17 17 20 text 17.2 20 "center of circle"
rectangle 15 18 1 1 pen red fill grey
rectangle 17 20 2 1 angle 30

# higher linkage primitives
joint 18 18
point 19 19
link 20 20 21 21
groundpin 22 22 25 25 angle 30
link 22 22 25 25
polygon 4 10 5 10 6 13 3.5 14 fill green
spring 10 1 15 1
ground 17 1.0 19 2.0
# The traced trajectory shown on the upper left
point 0 20 trace
point 3 23 trace
point 6 25 trace
point 10 20 trace

```

図 1.8: サンプルデータファイル sample.qnm

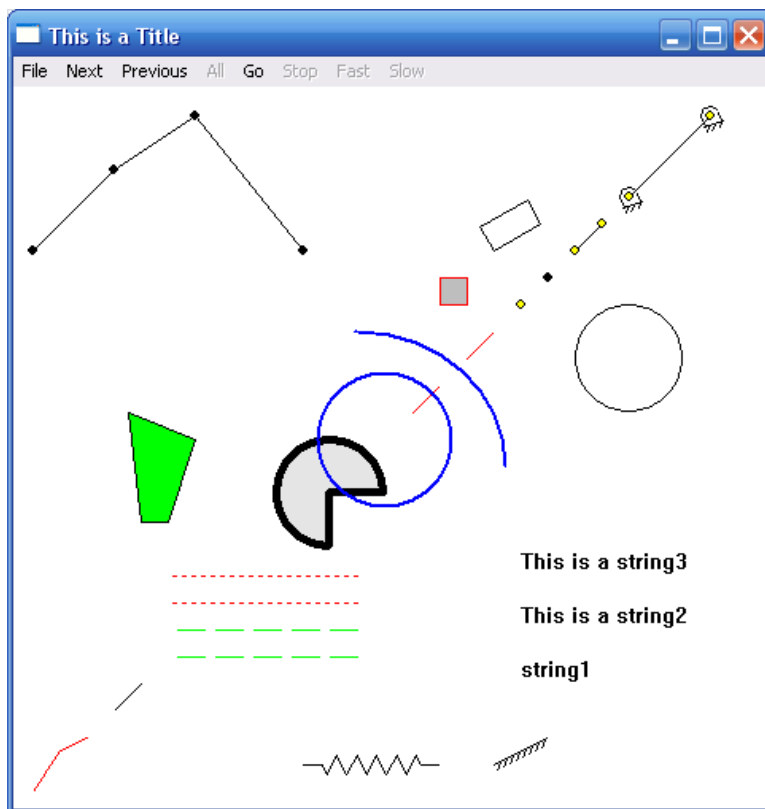


図 1.9: サンプルデータファイル sample.qnm に基づく QuickAnimation の表示

1.4.2 例 2: 減衰自由振動の位置の表示

図 1.10 の QuickAnimation™ データファイルは図 1.1 に示される振動系の表示に使用可能です。# で始まる最初の行は、コメント行です。次に、このアニメーションのタイトルがセットされています。そして固定オブジェクトが指定されています。まず、平面ピンと連結点が以下のように、(0,6) に配置されます。

```
groundpin 0 6 angle 180 joint 0 6
```

そして、2 番目のものが (4, 6)、3 番目のものが (8, 6) に配置されます。3 つのテキスト文字列 *overdamped*、*critical damped*、および *underdamped* は平面ピンのとなりに配置されます。表示領域は、テキスト文字列の幅を考慮することなくプリミティブのデータに基づいて自動的に計算されますので、

```
dot 11 6 pen white
```

の指定を用いると、テキスト文字列 *underdamped* が完全に表示されます。以下の指定

```
rectangle -0.500000 -0.818212 1.000000 1.000000 fill red \
spring 0 6 0 0.181788 \
```

は、過減衰 (*overdamped*) 振動系に対する矩形とばねを描画します。次の 2 つの継続行は、臨界減衰 (*critical damped*) 振動系に対する矩形とばねを描画します。最後の 2 つの行は、減衰 (*underdamped*) 振動系用のものです。固定オブジェクトとして表示するには、これらのプリミティブを継続シンボルなしに複数行で指定することもできます。

```
# qanimate data for animation of vibration systems
title "Positions of damped free vibration"
fixture
groundpin 0 6 angle 180 joint 0 6
text 0.5 6 "overdamped"
groundpin 4 6 angle 180 joint 4 6
text 4.5 6 "critical damped"
groundpin 8 6 angle 180 joint 8 6
text 8.5 6 "underdamped"
dot 11 6 pen white
rectangle -0.5 -0.818212 1 1 fill red \
spring 0 6 0 0.181788 \
rectangle 3.5 -1.049575 1 1 fill green \
spring 4 6 4 -0.049575 \
rectangle 7.5 0.412908 1 1 fill blue \
spring 8 6 8 1.412908
```

図 1.10: 図 1.1 で示される振動系に対する QuickAnimation™ ファイル

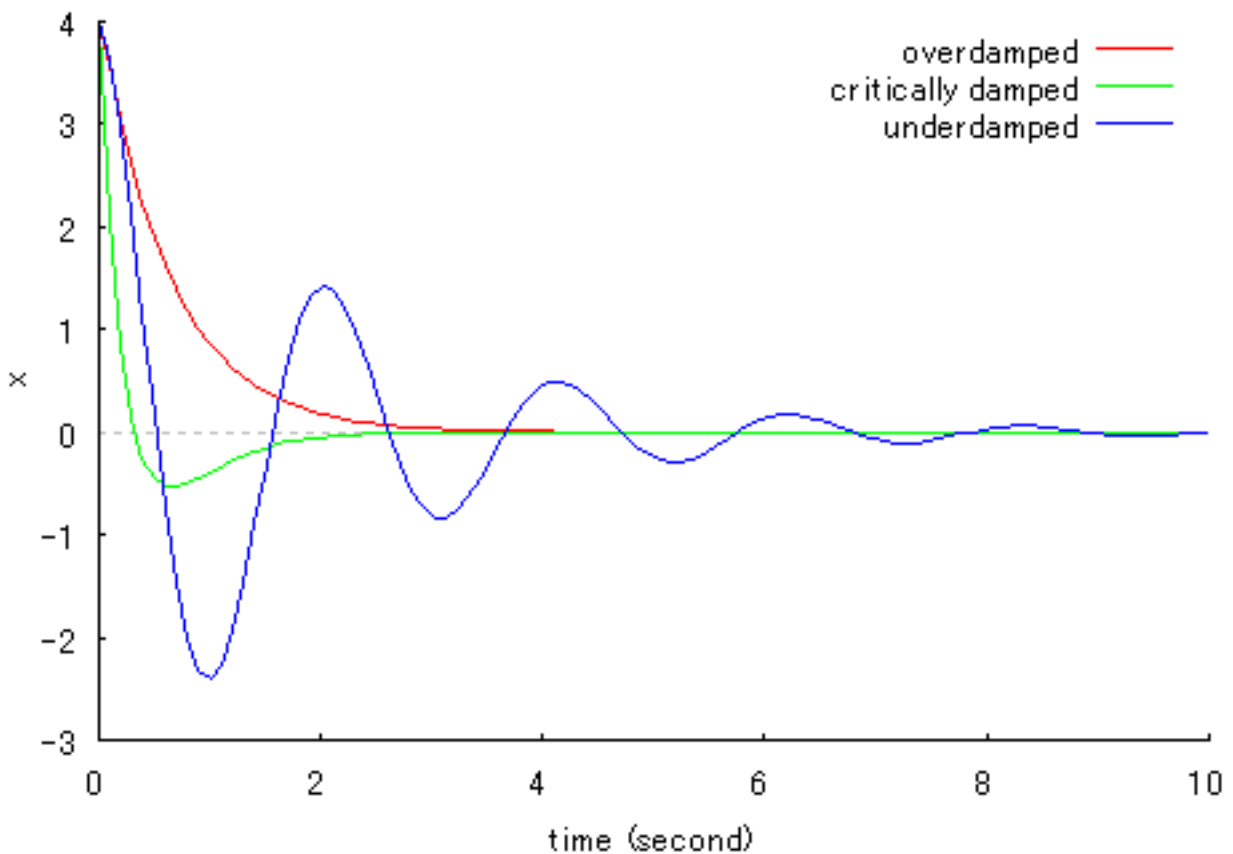


図 1.11: 3 つの減衰自由振動

```

/*****
* File: vibration.cpp
* Display the positions of damped free vibrations of
* overdamped, critical damped, underdamped systems.
* Note: The details about this damped free vibration can be found in
* an exercise in Chapter 6 Functions in the book
* "C for Engineers and Scientists: An Interpretive Approach"
* by Harry H. Cheng, published by McGraw-Hill, 2009,
* ISBN: 0073376051, ISBN-13: 978-0073376059.
*****/
#include <stdio.h>
#include <math.h>
#include <chplot.h>

/* The amplitude of the vibration is 4 */
double overdamped(double t) {
    return 4.2*exp(-1.57*t) - 0.2*exp(-54.2*t);
}

double criticaldamped(double t) {
    return 4*(1-3*t)*exp(-3*t);
}

double underdamped(double t) {
    return 4*exp(-0.5*t)*sin(3*t+M_PI/2);
}

int main() {
    double t0, tf;
    int num = 100;          // number of points for plotting
    CPlot plot;

    t0 = 0;
    tf = 10;
    plot.title("Damped Free Vibration");
    plot.label(PLOT_AXIS_X, "time (second)");
    plot.label(PLOT_AXIS_Y, "x");
    plot.func2D(t0, tf, num, overdamped);
    plot.legend("overdamped", 0);
    plot.func2D(t0, tf, num, criticaldamped);
    plot.legend("critically damped", 1);
    plot.func2D(t0, tf, num, underdamped);
    plot.legend("underdamped", 2);
    plot.plotting();
    return 0;
}

```

Program 1: 図 1.11 で示される減衰振動のための変動を生成するプログラム

減衰自由振動の 3 つのカテゴリ、すなわち、過減衰、臨界減衰、および減衰振動は、書籍 *C for Engineers and Scientists: An Interpretive Approach* (by Harry H. Cheng, published by McGraw-Hill, 2009) 内の「Chapter 6 Functions」の「Exercises」に記述されています。以下の例が提供されています。

1. **Overdamped** (過減衰)

$$y_1(t) = 4.2e^{-1.57t} - 0.2e^{-54.2t} \quad (1.1)$$

このケースでは、振動は起こりません。運動は衰退し、図 1.11 に示されるように、 x は長い時間をかけてゼロに近づきます。

2. **Critically damped** (臨界減衰)

$$y_2(t) = 4(1 - 3t)e^{-3t} \quad (1.2)$$

臨界減衰系に対しても、運動は非周期的です。質点は、やはり急激に平衡位置に達します。

3. **Underdamped** (減衰振動)

$$y_3(t) = 4e^{-0.5t} \sin(3t + \pi/2) \quad (1.3)$$

このケースでは、プログラム 1 で生成され、図 1.11 で示されるような振動が発生します。この解は、指数関数的に減少する調和関数になります。しかしながら、これは減衰運動ですので、結局、本体は、時間 t の大きな値に対して、平衡位置に近づきます。

図 1.1 は、時間 t が 2 秒のときの、方程式 (1.1)、(1.2)、および (1.3) を用いた上記 3 つの減衰自由振動の位置を示しています。プログラム 2 は、図 1.10 で示される QuickAnimation™ データの作成に使用されます。

図 1.1 で表示される振動系は、Ch コマンドシェルで以下のコマンドを入力することにより作成されます。

```
position.c | qanimate
```

または

```
position.c > tmp1.qnm  
qanimate tmp1.qnm
```


1章. オブジェクトの表示とアニメーションのための QUICKANIMATION

1.4. QUICKANIMATION™ の使用例

```
/* *****
* File: position.c
* Display the position when t is 2 seconds for damped free vibration of
* overdamped, critical damped, underdamped systems.
* Run this program in Ch as follows:
*     position.c | qanimate
*     or
*     positions.c > tmp1.qnm
*     qanimate tmp1.qnm
* See CHHOME/docs/qanimate.pdf for detailed description of this program.
* Note: The details about this damped free vibration can be found in
* an exercise in Chapter 6 Functions in the book
* "C for Engineers and Scientists: An Interpretive Approach"
* by Harry H. Cheng, published by McGraw-Hill, 2009,
* ISBN: 0073376051, ISBN-13: 978-0073376059.
* *****/
#include <stdio.h>
#include <math.h>

/* The amplitude of the vibration is 4 */
double overdamped(double t) {
    return 4.2*exp(-1.57*t) - 0.2*exp(-54.2*t);
}
double criticaldamped(double t) {
    return 4*(1-3*t)*exp(-3*t);
}
double underdamped(double t) {
    return 4*exp(-0.5*t)*sin(3*t+M_PI/2);
}

int main() {
    double t, t0, tf;           // time
    double y1, y2, y3;          // displacement
    double pin1x = 0, pin1y=7, // pin 1
           pin2x = 4, pin2y=7, // pin 2
           pin3x = 8, pin3y=7; // pin 3

    /* A comment line starting with # */
    printf("# qanimate data for positions of vibration systems\n");
    /* The title displayed on the animation */
    printf("title \"Positions of damped free vibration\"\n");
    printf("fixture\n");
    /* The primitives following fixture */
    printf("groundpin %f %f angle 180 joint %f %f\n", pin1x, pin1y, pin1x, pin1y);
    printf("line %f %f %f %f\n", pin1x, pin1y, pin1x, pin1y-1 );
    printf("text %f %f \"overdamped\"\n", pin1x+0.5, pin1y);
    printf("groundpin %f %f angle 180 joint %f %f\n", pin2x, pin2y, pin2x, pin2y);
    printf("line %f %f %f %f\n", pin2x, pin2y, pin2x, pin2y-1 );
    printf("text %f %f \"critical damped\"\n", pin2x+0.5, pin2y);
    printf("groundpin %f %f angle 180 joint %f %f\n", pin3x, pin3y, pin3x, pin3y);
    printf("line %f %f %f %f\n", pin3x, pin3y, pin3x, pin3y-1 );
    printf("text %f %f \"underdamped\"\n", pin3x+0.5, pin3y);
    printf("dot 11 7 pen white\n"); // to display all text corretly

    t = 2; // 2 seconds
    y1 = overdamped(t);
    y2 = criticaldamped(t);
    y3 = underdamped(t);
    printf("rectangle %f %f %f %f fill red \\n", -0.5, y1-1.0, 1.0, 1.0);
    printf("spring %f %f %f %f %f \\n", pin1x, pin1y-1, pin1x, y1);
    printf("rectangle %f %f %f %f fill green \\n", pin2x-0.5, y2-1.0, 1.0, 1.0);
    printf("spring %f %f %f %f %f \\n", pin2x, pin2y-1, pin2x, y2);
    printf("rectangle %f %f %f %f fill blue \\n", pin3x-0.5, y3-1.0, 1.0, 1.0);
    printf("spring %f %f %f %f %f \\n", pin3x, pin3y-1, pin3x, y3);
    return 0;
}
```

Program 2: 図 1.10 における QuickAnimation™ データファイル作成用プログラム

1.4.3 例 3: 減衰自由振動のアニメーション

上記の減衰自由振動の動きは、停止しようとするエレベータのようでもあります。仮にこの動きが減衰振動であれば、乗っていて非常に不快に感じ、そしてこの動きが過減衰であるならば、乗っていて遅く感じることでしょう。臨界振動ならば、迅速かつスムーズな乗り心地が提供されます。QuickAnimation™ における動作のアニメーションは、プログラム 3 で作成可能です。

プログラム 3 の結果は、図 1.12 に示されています。このアニメーションを作成するデータセットは for ループで生成されます。図 1.13 は、このアニメーションデータファイルで生成された Quick-Animation アニメーションを表示しています。

```

/*****
* File: vibration.c
* Animate the damped free vibration of
* overdamped, critical damped, underdamped systems.
* The output is for animation coordinate data.
* (1) Run this program in Ch as follows:
*     vibration.c | qanimate
*     or
*     vibration.c > tmp1.gnm
*     qanimate tmp1.gnm
* (2) Click "Go" to view the animation
* See CHHOME/docs/qanimate.pdf for detailed description of this program.
* Note: The details about this damped free vibration can be found in
* an exercise in Chapter 6 Functions in the book
* "C for Engineers and Scientists: An Interpretive Approach"
* by Harry H. Cheng, published by McGraw-Hill, 2009,
* ISBN: 0073376051, ISBN-13: 978-0073376059.
*****/
#include <stdio.h>
#include <math.h>

/* The amplitude of the vibration is 4 */
double overdamped(double t) {
    return 4.2*exp(-1.57*t) - 0.2*exp(-54.2*t);
}

double criticaldamped(double t) {
    return 4*(1-3*t)*exp(-3*t);
}

double underdamped(double t) {
    return 4*exp(-0.5*t)*sin(3*t+M_PI/2);
}

int main() {
    double t, t0, tf;           // time
    double y1, y2, y3;         //displacement
    double pin1x = 0, pin1y=7, // pin 1
           pin2x = 4, pin2y=7, // pin 2
           pin3x = 8, pin3y=7; // pin 3

    /* A comment line starting with # */
    printf("# qanimate data for animation of vibration systems\n");
    /* The title displayed on the animation */
    printf("title \"Damped Free Vibration\"\n");
    printf("fixture\n");
    /* The primitives following fixture */
    printf("groundpin %f %f angle 180 joint %f %f\n", pin1x, pin1y, pin1x, pin1y);
    printf("line %f %f %f %f\n", pin1x, pin1y, pin1x, pin1y-1);
    printf("text %f %f \"overdamped\"\n", pin1x+0.5, pin1y);
    printf("groundpin %f %f angle 180 joint %f %f\n", pin2x, pin2y, pin2x, pin2y);
    printf("line %f %f %f %f\n", pin2x, pin2y, pin2x, pin2y-1);
    printf("text %f %f \"critical damped\"\n", pin2x+0.5, pin2y);
    printf("groundpin %f %f angle 180 joint %f %f\n", pin3x, pin3y, pin3x, pin3y);
    printf("line %f %f %f %f\n", pin3x, pin3y, pin3x, pin3y-1);
    printf("text %f %f \"underdamped\"\n", pin3x+0.5, pin3y);
    printf("dot 11 7 pen white\n"); // to display all text corretly
    printf("animate restart\n");

    t0 = 0;
    tf = 10;
    for(t = t0; t<tf; t += 0.01) {
        y1 = overdamped(t);
        y2 = criticaldamped(t);
        y3 = underdamped(t);
        printf("rectangle %f %f %f %f fill red \\n", -0.5, y1-1.0, 1.0, 1.0);
        printf("spring %f %f %f %f \\n", pin1x, pin1y-1, pin1x, y1);
        printf("rectangle %f %f %f %f fill green \\n", pin2x-0.5, y2-1.0, 1.0, 1.0);
        printf("spring %f %f %f %f \\n", pin2x, pin2y-1, pin2x, y2);
        printf("rectangle %f %f %f %f fill blue \\n", pin3x-0.5, y3-1.0, 1.0, 1.0);
        printf("spring %f %f %f %f \\n", pin3x, pin3y-1, pin3x, y3);
    }
    return 0;
}

```

Program 3: 減衰自由振動のアニメーションを作成するプログラム

```

# qanimate data for animation of vibration systems
title "Positions of damped free vibration"
fixture
groundpin 0 6 angle 180 joint 0 6
text 0.5 6 "overdamped"
groundpin 4 6 angle 180 joint 4 6
text 4.5 6 "critical damped"
groundpin 8 6 angle 180 joint 8 6
text 8.5 6 "underdamped"
dot 11 6 pen white
rectangle -0.500000 -0.818212 1.000000 1.000000 fill red \
spring 0 6 0 0.181788 \
rectangle 3.500000 -1.049575 1.000000 1.000000 fill green \
spring 4 6 4 -0.049575 \
rectangle 7.500000 0.412908 1.000000 1.000000 fill blue \
spring 8 6 8 1.412908

.
.
.

rectangle -0.500000 -0.999999 1.000000 1.000000 fill red \
spring 0.000000 6.000000 0.000000 0.000001 \
rectangle 3.500000 -1.000000 1.000000 1.000000 fill green \
spring 4.000000 6.000000 4.000000 -0.000000 \
rectangle 7.500000 -0.995843 1.000000 1.000000 fill blue \
spring 8.000000 6.000000 8.000000 0.004157

```

図 1.12: プログラム 3 で生成された QuickAnimation™ データ

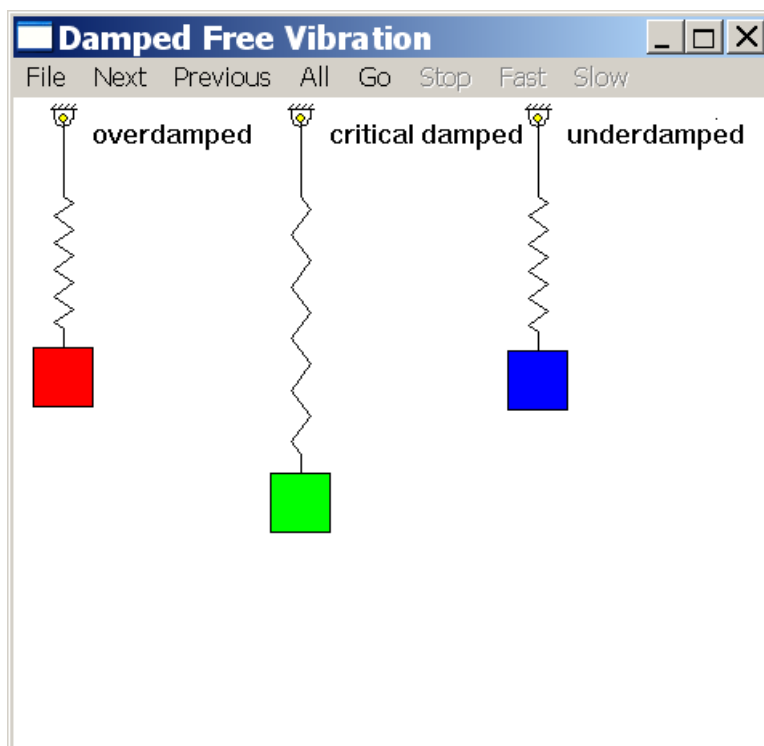


図 1.13: 減衰自由振動のスナップショットを示す QuickAnimation ウィンドウ

QuickAnimation アニメーションを用いた減衰自由振動のアニメーションに対しては、2つの追加的なプログラムがあります。これらのプログラムは、プログラム 3 で示されるプログラム `vibration.c` と同じアニメーションを生成しますが、アニメーションデータの扱い方が特殊です。プログラム `vibration2.ch` は、アニメーション座標データを最初に一時ファイルに出力します。このアニメーションデータファイルは次に、コマンド `qanimate` により処理されます。QuickAnimation™ 終了後、この一時アニメーションデータファイルは削除されます。このケースでは、プログラム `vibration2.ch` が ChIDE で直ちに実行可能になり、アニメーションが生成されます。プログラム `vibration3.c` もアニメーションを生成します。このアニメーションデータは、関数 `popen()` を用いて QuickAnimation プログラム `qanimate` に直接パイプ出力され、自動的にアニメーションを生成します。コマンド `qanimate` は Ch プログラム内部のコマンドとして起動されます。プログラムを実行すると、単純に希望するアニメーションが生成されます。

プログラム `vibration2.ch` のリスト

```

/*****
* File: vibration2.ch
* Animate the damped free vibration of
* overdamped, critically damped, underdamped systems.
* The output is for animation coordinate data.
* (1) Run this program in Ch.
* (2) Click "Go" to view the animation
* See CHHOME/docs/qanimate.pdf for detailed description of this program.
* Note: The details about this damped free vibration can be found in
* an exercise in Chapter 6 Functions in the book
* "C for Engineers and Scientists: An Interpretive Approach"
* by Harry H. Cheng, published by McGraw-Hill, 2009,
* ISBN: 0073376051, ISBN-13: 978-0073376059.
*****/
#include <stdio.h>
#include <math.h>

/* The amplitude of the vibration is 4 */
double overdamped(double t) {
    return 4.2*exp(-1.57*t) - 0.2*exp(-54.2*t);
}

double criticaldamped(double t) {
    return 4*(1-3*t)*exp(-3*t);
}

double underdamped(double t) {
    return 4*exp(-0.5*t)*sin(3*t+M_PI/2);
}

int main() {
    double t, t0, tf;           // time
    double y1, y2, y3;         // displacement
    double pin1x = 0, pin1y=7, // pin 1
           pin2x = 4, pin2y=7, // pin 2
           pin3x = 8, pin3y=7; // pin 3
    FILE *stream;
    char qnmFileName[1024];     // data file name for qanimate

    tmpnam(qnmFileName);
    stream = fopen(qnmFileName, "w");
    if (stream==NULL) {
        fprintf(stderr, "Error: cannot open '%s'\n", qnmFileName);
        exit(1);
    }

    /* The first line of the animation file must start with #qanimate */
    fprintf(stream, "# qanimate data for animation of vibration systems\n");
    /* The title displayed on the animation */
    fprintf(stream, "title \"Damped Free Vibration\"\n");

```

```

fprintf(stream, "fixture\n");
/* The primitives following fixture */
fprintf(stream, "groundpin %f %f angle 180 joint %f %f\n", pin1x, pin1y, pin1x, pin1y);
fprintf(stream, "line %f %f %f %f\n", pin1x, pin1y, pin1x, pin1y-1 );
fprintf(stream, "text %f %f \"overdamped\"\n", pin1x+0.5, pin1y);
fprintf(stream, "groundpin %f %f angle 180 joint %f %f\n", pin2x, pin2y, pin2x, pin2y);
fprintf(stream, "line %f %f %f %f\n", pin2x, pin2y, pin2x, pin2y-1 );
fprintf(stream, "text %f %f \"critical damped\"\n", pin2x+0.5, pin2y);
fprintf(stream, "groundpin %f %f angle 180 joint %f %f\n", pin3x, pin3y, pin3x, pin3y);
fprintf(stream, "line %f %f %f %f\n", pin3x, pin3y, pin3x, pin3y-1 );
fprintf(stream, "text %f %f \"underdamped\"\n", pin3x+0.5, pin3y);
fprintf(stream, "dot 11 7 pen white\n"); // to display all text corretly
fprintf(stream, "animate restart\n");

t0 = 0;
tf = 10;
for(t = t0; t<tf; t += 0.01) {
    y1 = overdamped(t);
    y2 = criticaldamped(t);
    y3 = underdamped(t);
    fprintf(stream, "rectangle %f %f %f %f fill red \\n", -0.5, y1-1.0, 1.0, 1.0);
    fprintf(stream, "spring %f %f %f %f \\n", pin1x, pin1y-1, pin1x, y1);
    fprintf(stream, "rectangle %f %f %f %f fill green \\n", pin2x-0.5, y2-1.0, 1.0, 1.0);
    fprintf(stream, "spring %f %f %f %f \\n", pin2x, pin2y-1, pin2x, y2);
    fprintf(stream, "rectangle %f %f %f %f fill blue \\n", pin3x-0.5, y3-1.0, 1.0, 1.0);
    fprintf(stream, "spring %f %f %f %f \\n", pin3x, pin3y-1, pin3x, y3);
}
fclose(stream);
qanimate $qnmFileName
remove(qnmFileName);
return 0;
}

```

プログラム vibration3.c のリスト

```

/*****
* File: vibration3.c
* Animate the damped free vibration of
* overdamped, critical damped, underdamped systems.
* The output is for animation coordinate data.
* (1) Run this program in Ch.
* (2) Click "Go" to view the animation
* See CHHOME/docs/qanimate.pdf for detailed description of this program.
* Note: The details about this damped free vibration can be found in
* an exercise in Chapter 6 Functions in the book
* "C for Engineers and Scientists: An Interpretive Approach"
* by Harry H. Cheng, published by McGraw-Hill, 2009,
* ISBN: 0073376051, ISBN-13: 978-0073376059.
*****/
#include <stdio.h>
#include <math.h>

/* The amplitude of the vibration is 4 */
double overdamped(double t) {
    return 4.2*exp(-1.57*t) - 0.2*exp(-54.2*t);
}

double criticaldamped(double t) {
    return 4*(1-3*t)*exp(-3*t);
}

double underdamped(double t) {
    return 4*exp(-0.5*t)*sin(3*t+M_PI/2);
}

int main() {
    double t, t0, tf;           // time
    double y1, y2, y3;         // displacement

```

1 章 . オブジェクトの表示とアニメーションのための QUICKANIMATION

1.4. QUICKANIMATION™ の使用例

```
double pin1x = 0, pin1y=7, // pin 1
       pin2x = 4, pin2y=7, // pin 2
       pin3x = 8, pin3y=7; // pin 3
FILE *stream;

stream = popen("qanimate","w");    // open qanimate pipe
if (stream==NULL) {
    fprintf(stderr, "Error: popen() failed\n");
    exit(1);
}

/* A comment line starting with # */
fprintf(stream, "# qanimate data for animation of vibration systems\n");
/* The title displayed on the animation */
fprintf(stream, "title \"Damped Free Vibration\"\n");
fprintf(stream, "fixture\n");
/* The primitives following fixture */
fprintf(stream, "groundpin %f %f angle 180 joint %f %f\n", pin1x, pin1y, pin1x, pin1y);
fprintf(stream, "line %f %f %f %f\n", pin1x, pin1y, pin1x, pin1y-1 );
fprintf(stream, "text %f %f \"overdamped\"\n", pin1x+0.5, pin1y);
fprintf(stream, "groundpin %f %f angle 180 joint %f %f\n", pin2x, pin2y, pin2x, pin2y);
fprintf(stream, "line %f %f %f %f\n", pin2x, pin2y, pin2x, pin2y-1 );
fprintf(stream, "text %f %f \"critical damped\"\n", pin2x+0.5, pin2y);
fprintf(stream, "groundpin %f %f angle 180 joint %f %f\n", pin3x, pin3y, pin3x, pin3y);
fprintf(stream, "line %f %f %f %f\n", pin3x, pin3y, pin3x, pin3y-1 );
fprintf(stream, "text %f %f \"underdamped\"\n", pin3x+0.5, pin3y);
fprintf(stream, "dot 11 7 pen white\n"); // to display all text corretly
fprintf(stream, "animate restart\n");

t0 = 0;
tf = 10;
for(t = t0; t<tf; t += 0.01) {
    y1 = overdamped(t);
    y2 = criticaldamped(t);
    y3 = underdamped(t);
    fprintf(stream, "rectangle %f %f %f %f fill red \\n", -0.5, y1-1.0, 1.0, 1.0);
    fprintf(stream, "spring %f %f %f %f \\n", pin1x, pin1y-1, pin1x, y1);
    fprintf(stream, "rectangle %f %f %f %f fill green \\n", pin2x-0.5, y2-1.0, 1.0, 1.0);
    fprintf(stream, "spring %f %f %f %f \\n", pin2x, pin2y-1, pin2x, y2);
    fprintf(stream, "rectangle %f %f %f %f fill blue \\n", pin3x-0.5, y3-1.0, 1.0, 1.0);
    fprintf(stream, "spring %f %f %f %f \\n", pin3x, pin3y-1, pin3x, y3);
}
pclose(stream);
return 0;
}
```

2 章

オブジェクトの Web ベースの表示とアニメーション

QuickAnimationTM プログラムは、オブジェクトとアニメーションの Web ベースの表示プログラムを開発する際に便利です。

2.0.4 CGI スクリプトファイルの記述

<FORM> タグをもつ HTML ドキュメントから送信されるデータ処理には、サーバー側にあるスクリプトファイルが必要とされます。いったんデータが、Common Gateway Interface (CGI) またはその他の仕組みを通してサーバーに渡されると、必要な手順が実行され、その結果がクライアントに返されます。Ch における CGI の詳細については、Ch CGI Toolkit User's Guide を参照してください。QuickAnimationTM を出力フォームのコンテンツタイプとして使用するには、CGI スクリプトがステートメント

```
Response.setContentType("application/x-qnm");
```

で指定されている必要があります。これは出力が QuickAnimationTM アプリケーションであることを示しています。

2.0.5 Web サーバーの構築とセットアップ

Unix オペレーティングシステムで Netscape Web サーバーから QuickAnimationTM アプリケーションを実行するには、以下の行を `server_home_dir/https-80_or_http/config` ディレクトリにある Netscape WWW 構築ファイル **mime.types** に追加します。

```
type=application/x-qnm      exts=qnm
```

Apache Web サーバーに対しては、以下の行

```
application/x-qnm      qnm
```

をファイル `server_home_dir/conf/mime.types` に追加します。変更を有効にするには、Web サーバーを再起動する必要があります。

索引

QuickAnimation™, 1

CGI, 21

CGI プログラミング, 20

qanimate, 1

一般描画プリミティブ, 3

一般プリミティブ, 3

コメント, 2

著作権情報, ii

力学的描画プリミティブ, 6

力学的プリミティブ, 6