前項と同じく、実行してみましょう。

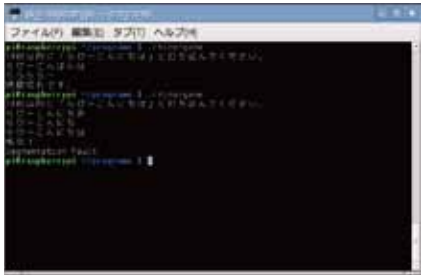

図 5.4.1　イメージ

イメージのような動作をしたでしょうか？
ここまでできれば、初心者卒業です！

5.5　プログラム例：”グラフィックプログラミング”に挑戦！

パソコンやスマートフォンで目にするような、ボタンやテキストボックスがあって、絵が描いてあって･･･というようなプログラムを、GTK というライブラリを利用して作ってみましょう。
目標とするプログラムは、以下のようなイメージです。

図 5.5.1

まず、下記**コマンドリスト**を打ち込み、GTK をインストールします。

コマンドリスト 5.7

```
pi@raspberrypi ~ $ sudo apt-get install libgtk-3-dev
（問いかけが表示されたら、[Y] キー→[Enter] キーを押下して決定）
```

```
pi@raspberrypi ~ $ sudo apt-get install libgtk-3-dev
パッケージリストを読み込んでいます ... 完了
依存関係ツリーを作成しています
状態情報を読み取っています ... 完了
以下の特別パッケージがインストールされます :
  autopoint debhelper gettext gir1.2-atk-1.0 gir1.2-freedesktop
〜〜〜〜〜〜〜〜中略〜〜〜〜〜〜〜〜
この操作後に追加で 61.9 MB のディスク容量が消費されます。
続行しますか [Y/n]?
```

続いて、下記コマンドでディレクトリを作成の上、nano を起動します。

コマンドリスト 5.8

```
pi@raspberrypi ~ $ sudo mkdir gui
pi@raspberrypi ~ $ sudo cd gui
pi@raspberrypi ~/gui $ sudo nano main.c
（問いかけが表示されたら、[Y] キー→[Enter] キーを押下して決定）
```

用意ができたら、以下のプログラムを打ち込みましょう。

プログラムソース 5.5.1

```
#include <stdio.h>
#include <gtk/gtk.h>
#include <stdlib.h>

GtkWidget *window;

// キャンバスに図を描画する関数
gboolean draw_canvas (GtkWidget *widget, cairo_t *ct, gpointer data)
{
    // キャンバスの幅と高さを取得
    guint width = gtk_widget_get_allocated_width(widget),
          height = gtk_widget_get_allocated_height(widget);
     // 色を準備
```

プログラムソース 5.5.1

```
    // 色を準備
    GdkRGBA color_Red = {1.0, 0, 0, 1.0};
    // 円を描画
    gdk_cairo_set_source_rgba(ct, &color_Red);
    cairo_arc(ct, width * 0.5, height * 0.5, MIN(width, height)*0.3 , 0.0, 2.0 * G_PI );
    cairo_fill(ct);
    //
    return FALSE;
}

void button_clicked (GtkWidget *widget, gpointer data)
{
int child,status;
//fork() 関数を使うと、自分 ( アプリケーション ) の分身を全く同じ状態で作り出す。
その後、起動に成功した ( 親の ) 場合は正の数、失敗した場合は -1、自身が分身した側
( 子 ) である場合は 0 を返す。
    if ((child = fork()) < 0) {
        perror("fork");
    }
    if(child == 0){
// 自身がコピーされたプロセスであれば、自身を chromium に変身させる。これにより、
元のランチャーを残したまま chromium が起動することになる。execlp("chromium",
"chromium","http://www.youtube.com","-e","Fullscreen",NULL);
    }
}

void button2_clicked (GtkWidget *widget, gpointer data)
{
    // アプリケーション終了
    gtk_main_quit();
}

int main (int argc, char *argv[])
{
```

```
GtkWidget *box, *button, *button2 , *label,
        *hbox1, *hbox2, *hbox3 , *canvas1;
 //GTK を初期化
gtk_init(&argc, &argv);

// ウィンドウ作成
window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
// 破棄の方法を指定
g_signal_connect(window, "delete-event",
                G_CALLBACK(gtk_main_quit), NULL );

// ウィンドウサイズを指定
gtk_window_set_default_size(GTK_WINDOW(window), 500, 300);

// ボックスを作成、ウィンドウに配置
box = gtk_box_new(GTK_ORIENTATION_VERTICAL, 0);
gtk_box_set_homogeneous(GTK_BOX(box), TRUE);
gtk_container_add(GTK_CONTAINER(window), box);

// 水平方向のボックス 1 を作成、ボックスへ入れる
hbox1 = gtk_box_new(GTK_ORIENTATION_HORIZONTAL,10);
gtk_box_set_homogeneous(GTK_BOX(hbox1), TRUE);
gtk_box_pack_start(GTK_BOX(box), hbox1, TRUE, TRUE, 0);

// 水平方向のボックス 2 を作成、ボックスへ入れる
hbox2 = gtk_box_new(GTK_ORIENTATION_HORIZONTAL,10);
gtk_box_set_homogeneous(GTK_BOX(hbox2), TRUE);
gtk_box_pack_start(GTK_BOX(box), hbox2, TRUE, TRUE, 0);

水平方向のボックス 3 を作成、ボックスへ入れる
hbox3 = gtk_box_new(GTK_ORIENTATION_HORIZONTAL,10);
gtk_box_set_homogeneous(GTK_BOX(hbox3), TRUE);
gtk_box_pack_start(GTK_BOX(box), hbox3, TRUE, TRUE, 0);

// ラベルを作成
label = gtk_label_new("Hello World!");
gtk_box_pack_start(GTK_BOX(hbox1), label, TRUE, TRUE, 0);
```

```
    // キャンバスを作成、draw_canvas で描画するよう定める
    canvas1 = gtk_drawing_area_new();
    gtk_widget_set_size_request(canvas1, 100, 100);
    gtk_box_pack_start(GTK_BOX(hbox1), canvas1, TRUE, TRUE, 0);

    g_signal_connect(canvas1, "draw",
                     G_CALLBACK(draw_canvas), NULL );

    // ボタン 1 を配置
    button = gtk_button_new_with_label("Youtube");
    g_signal_connect(button, "clicked",
                       G_CALLBACK(button_clicked), NULL);
    gtk_box_pack_start(GTK_BOX(hbox3), button, TRUE, TRUE, 10);

     // ボタン 2 を配置
    button2 = gtk_button_new_with_label("Quit");
g_signal_connect(button2, "clicked"
(,
                       G_CALLBACK(button2_clicked), NULL);
    gtk_box_pack_start(GTK_BOX(hbox3), button2, TRUE, TRUE, 10);

     // 表示する
    gtk_widget_show_all(window);

    // ループ
    gtk_main();
}
```

解説

ラベルとキャンバスを配置し、ためしに円を描画しています。
Youtube ボタンを押すとブラウザにて Youtube が起動し、Quit ボタンで終了するというシンプルなプログラムです。

プログラムを無事書き終えたら、/home/pi/gui/main.c(~/gui/main.c) に保存を行います。

続いてコンパイルを行います。
ターミナルを開き、以下のようにコマンドを入力し、実行して下さい。

コマンドリスト 5.8

```
pi@raspberrypi ~/gui $ sudo gcc main.c -o main `pkg-config
--libs --cflags gtk+-3.0`
```

解説

gcc は、あらかじめ Raspbian にインストールされているコンパイラです。
半角スペースで続けて文字列を打ち込むことで、様々なオプションをつけることができます。
ここでは以下のような内容でコンパイルを行っています。

pi@raspberrypi ~/gui $ **sudo gcc main.c -o main `pkg-config --libs --cflags gtk+-3.0` -o main**

main という名前で実行ファイルを出力

`pkg-config --libs --cflags gtk+-3.0`

gtk を使うと明示します。おまじないだと思っていて下さい。

エラーが出なければ成功です。
次のコマンドを入力し、実行ファイルを実行しましょう。

コマンドリスト 5.4.1

```
pi@raspberrypi /c $ sudo ./main
```

他にもラジオボタン、チェックボックス、タイマー、画像など様々なパーツを組み合わせて GUI アプリケーションを作成することができます。