VGUI
Tutorial for application programmers

Karen McGaul
Robotics Research Group
Department of Engineering Science
University of Oxford

September 1, 2000

Contents

Chapter 1

Introduction

The purpose of this document is to provide an overview of the GUI (graphical user interface) used with Target. It should provide all the information necessary to write applications but does not go into details about the design and internal workings of the new GUI system. It is assumed that the reader has a reasonable knowledge of C++.

1.1 Getting started

This document consists of example programs which introduce different aspects of GUI. This is what you will need to do to compile and run the examples:

Make a copy of the examples directory. Source code and shared objects are provided. (kym**** Note - shared objects not made yet, but there is a makefile)

```
unix% mkdir ~/examples
unix% cd $IUEROOT/Documentation/Tutorials/
unix% cp vgui/examples/*.C ~/examples/
unix% cp vgui/examples/*.so ~/examples/
```

In order to run the examples use

```
unix% ./example.so <--factory=gui-toolkit> <image>
```

If you want to recompile the examples use

```
unix% make
```

The user interface consists of two parts:

- an OpenGL area/window used to display images and 3D models (this is called a tableau)
- a GUI toolkit used to create menus, buttons, etc.

If one of these examples is run with no preference given on the command line for a GUI toolkit then application will look to find a suitable GUI toolkit that is registered. If more than one GUI toolkit is registered then one will be selected at random. If you wish to specify a GUI toolkit (for example Foo) then you will need to add `--factory=Foo` to the end of the command line or set the variable `vgui` to Foo.

Chapter 2

Displaying images and 2D objects

2.1 Displaying an image

This first example simply displays an image in a window. The image is loaded into a tableau which displays images (`vgui_image_tableau`) and this is put inside a 2D viewer tableau (`vgui_viewer2d`). The 2D viewer adds the functionality that would commonly be required for viewing 2D images and objects such as zooming and moving which part of the image is displayed.

The viewer2D provides the following functionality:

control and right mouse button	zoom out
control and left mouse button	zoom in
control and middle mouse button	image can be dragged around the display area while the button is held down

```
// display-image.cxx
//
// This example displays an image in a window.

#include <vgui/vgui.h>
#include <vgui/vgui_image_tableau.h>
#include <vgui/vgui_viewer2D.h>

int main(int argc, char **argv)
{
  vgui::init(argc, argv);

  // Load an image into an image tableau:
  vgui_image_tableau image( argv[1] );

  // Put the image tableau inside a 2D viewer tableau:
  vgui_viewer2D viewer(&image);

  // Create a window, add the tableau and show it on screen:
  return vgui::run(&viewer, image.width(), image.height());
```

```
}
```

2.2 Displaying 2D objects

In order to display 2D objects we need to use a different tableau called `vgui_easy2D`. This tableau has functions which allow us to draw points, lines and other geometric objects. As in the previous example we include this into a 2D viewer tableau.

```
// draw-line.cxx
//
// This example draws a line in an OpenGL area in the window:

//#include ¡vbl/ArgParse.h¿
#include <vgui/vgui.h>
#include <vgui/vgui_image_tableau.h>
#include <vgui/vgui_easy2D.h>
#include <vgui/vgui_viewer2D.h>

int main(int argc, char **argv)
{
  vgui::init(argc, argv);

  // Use an easy2D tableau which allows us to draw 2D objects:
  vgui_easy2D easy2D("unnamed");

  // Add a line:
  easy2D.set_foreground(0,0,1);
  easy2D.set_line_width(2);
  easy2D.add_line(100,100,200,400);

  // Put into a viewer2D tableau to get zooming, etc:
  vgui_viewer2D viewer(&easy2D);

  // Create a window, add the tableau and show it on screen:
  return vgui::run(&viewer, 512, 512);
}
```

2.3 Displaying images and 2D objects together

To display images and 2D objects together we simply "stack" tableaux with the required functionality. So all we need to do is add an image tableau to an easy2D tableau.

```
// 2d-example.cxx
//

#include <vgui/vgui.h>
#include <vgui/vgui_image_tableau.h>
#include <vgui/vgui_easy2D.h>
#include <vgui/vgui_viewer2D.h>

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

```
  vgui::init(argc, argv);

  // Load an image into an image.tableau
  vgui_image_tableau image(argv[1] ? argv[1] : "/users/fsm/images/pig.jpg");

  // Put the image.tableau into a easy2D tableau
  vgui_easy2D easy2D(&image);

  // Add a point, line, and infinite line
  easy2D.set_foreground(0,1,0);
  easy2D.set_point_radius(5);
  easy2D.add_point(10, 20);

  easy2D.set_foreground(0,0,1);
  easy2D.set_line_width(2);
  easy2D.add_line(100,100,200,400);

  easy2D.set_foreground(0,1,0);
  easy2D.set_line_width(2);
  easy2D.add_infinite_line(1,1,-100);

  // Put the easy2D tableau into a viewer2D tableau:
  vgui_viewer2D viewer(&easy2D);


  // Create a window, add the tableau and show it on screen:
  return vgui::run(&viewer, image.width(), image.height());
}
```

Chapter 3

Menus and dialog boxes

3.1 Adding a menubar

To make a menubar we simply create a `vgui_menu` (a menu description) and pass that in when we call `vgui::produce_window`.

```
// menubar-example.cxc x
//
// This example displays an image with a menu bar.

#include <vgui/vgui.h>
#include <vgui/vgui_menu.h>
#include <vgui/vgui_image_tableau.h>
#include <vgui/vgui_viewer2D.h>

// Set up a dummy callback function for the menu to call (for
// simplicity all menu items will call this function):
static void dummy(const void*)
{
  cerr << "Dummy function called" << endl;
}

// Create a vgui.menu:
vgui_menu create_menus()
{
  vgui_menu file;
  file.add("Open",dummy,(vgui_key)'O',vgui_CTRL);
  file.add("Quit",dummy,(vgui_key)'R',vgui_SHIFT);

  vgui_menu image;
  image.add("Center image",dummy);
  image.add("Show histogram",dummy);

  vgui_menu bar;
  bar.add("File",file);
  bar.add("Image",image);

  return bar;
}

int main(int argc, char ** argv)
```

```
{
  vgui::init(argc,argv);

  // Load an image into an image tableau:
  vgui_image_tableau image("/users17/CVlectures/images/basement00.mit");
  vgui_viewer2D viewer(&image);

  // Create a window, but this time we also pass in a vgui˙menu.
  return vgui::run(&viewer, 512, 512, create_menus());
}
```

3.2 Adding a pop-up menu

A pop-up menu is also specified by a `vgui_menu`. In this next example we take the same `vgui_menu` we used to specify the menubar, but this time we add it to a pop-up menu.

```
// popup-example.cc
//
// This example displays an image with a popup-menu.

#include <vgui/vgui.h>
#include <vgui/vgui_window.h>
#include <vgui/vgui_adaptor.h>
#include <vgui/vgui_menu.h>
#include <vgui/vgui_image_tableau.h>
#include <vgui/vgui_viewer2D.h>

// Set up a dummy callback function for the menu to call (for
// simplicity all menu items will call this function):
static void dummy(const void*)
{
  cerr << "Dummy function called" << endl;
}

// Create a vgui.menu:
vgui_menu create_menus()
{
  vgui_menu file;
  file.add("Open", dummy);
  file.add("Quit", dummy);

  vgui_menu image;
  image.add("Center image", dummy);
  image.add("Show histogram", dummy);

  vgui_menu bar;
  bar.add("File",file);
  bar.add("Image",image);

  return bar;
}

int main(int argc, char ** argv)
{
  vgui::init(argc,argv);

  // Load an image into an image tableau:
  vgui_image_tableau image("/users/kym/target/target/vgui/kym_examples/images/bt.000.mit");
  vgui_viewer2D viewer(&image);
```

```
  // Create a window and add the tableau:
  vgui_window *win = vgui::produce_window(512, 512);
  win->get_adaptor()->set_tableau(&viewer);

  // Add our menu items to the base pop-up (this menu appears when the user clicks the right
  // menu button on the tableau:
  win->get_adaptor()->include_in_popup(create_menus());
  win->get_adaptor()->bind_popups();
  win->show();
  return vgui::run();
}
```

Click the right mouse button anywhere on the image to get the pop-up menu to appear. The implementation of pop-up menus in VGUI is very simple: you can only attach a menu to a tableau and the default method to get it to "pop up" is a right mouse click.

Other items in the pop-up menu that we didn't specifiy are the default items for an image tableau. You would see these items in a pop-up menu if you right clicked on any of the previous examples. If you wish to replace the default pop-up menu (instead of adding to it) then instead of `include_in_popup`, use the function `add_popup`.

3.3 Adding a dialog box

To make a dialog box, create a `vgui_dialog`. In this example the dialog box is shown when we select the menu item called "Dialog",

```
// dialog-example.cxx
//
// This example makes a dialog box and attaches it to a menu item.

#include <vbl/vbl_arg.h>
#include <vbl/vbl_bool_ostream.h>
#include <vgui/vgui.h>
#include <vgui/vgui_menu.h>
#include <vgui/vgui_dialog.h>
#include <vgui/vgui_image_tableau.h>
#include <vgui/vgui_viewer2D.h>

// Make a vgui.dialog:
static void test_dialog(const void*)
{
  static int int_value = 2;
  static long long_value = 3;
  static float float_value = 3.1;
  static double double_value = 4.2;
  static vcl_string string_value = "dialog test";
  static bool bool_value = true;
  static vcl_string file_value = "/tmp/myfile.txt";
  static vcl_string regexp = "*.txt";
   static vcl_string color_value = "blue";

  static int choice_value = 1;
  vcl_vector<vcl_string> labels;
  labels.push_back(vcl_string("fltk"));
```

```
  labels.push_back(vcl_string("motif"));
  labels.push_back(vcl_string("gtk"));
  labels.push_back(vcl_string("glut"));
  labels.push_back(vcl_string("glX"));

  vgui_dialog mydialog("My dialog");
  mydialog.field("int value", int_value);
  mydialog.field("long value", long_value);
  mydialog.field("float value", float_value);
  mydialog.field("double value", double_value);
  mydialog.field("string value", string_value);
  mydialog.checkbox("bool value", bool_value);
  mydialog.choice("choice value", labels, choice_value);
  mydialog.message("This is a message");
  mydialog.inline_file("file browser", regexp, file_value);
  mydialog.inline_color("color value", color_value);

  if (mydialog.ask())
  {
    cerr << "int_value : " << int_value << endl;
    cerr << "long_value : " << long_value << endl;
    cerr << "float_value : " << float_value << endl;
    cerr << "double_value : " << double_value << endl;
    cerr << "string_value : " << string_value << endl;
    cerr << "bool_value : " << vbl_bool_ostream::true_false(bool_value) << endl;
    cerr << "choice_value : " << choice_value << " " << labels[choice_value] << endl;
    cerr << "file_value: " << file_value << endl;
    cerr << "color_value: " << color_value << endl;
  }
}

// Create a vgui.menu with an item which shows the dialog box:
vgui_menu create_menus()
{
  vgui_menu test; test.add("Dialog", test_dialog);

  vgui_menu bar;
  bar.add("Test",test);

  return bar;
}

int main(int argc, char ** argv)
{
  vgui::init(argc,argv);

  vgui_image_tableau image("/users/pcp/bt/mit/bt.000.mit");
  vgui_viewer2D viewer(&image);

  // Create a window with a menu, add the tableau and show it on screen:
  return vgui::run(&viewer, image.width(), image.height(), create_menus());
}
```

Chapter 4

Handling events

4.1 Getting mouse position

When a mouse button is pressed it produces an event which is passed to the `handle_impl` function of the tableau. In order to use a mouse event we need to write our own tableau which is derived from a `vgui_tableau`.

In this next example we get the mouse position when the left mouse button is pressed on the image. The mouse position is given in viewport coordinates with the lower-left corner as the origin. (In general the viewport will be the area displaying OpenGL).

```
// mouse-position.cc
//
// This example displays the mouse position when the left mouse button is pressed.

#include <vgui/vgui.h>
#include <vgui/vgui_image_tableau.h>
#include <vgui/vgui_viewer2D.h>

class my_tableau : public vgui_image_tableau
{
public:
  my_tableau(char const *f) : vgui_image_tableau(f){ }
  ~my_tableau() { }

  bool handle(const vgui_event &e)
  {
    if (e.type == vgui_BUTTON_DOWN && e.button == vgui_LEFT && e.modifier == 0)
    {
      cout << "selecting at " << e.wx << " " << e.wy << endl;
      return true; // event has been used
    }

    // We are not interested in other events, so pass event to base class:
    return vgui_image_tableau::handle(e);
  }
};
```

```
int main(int argc,char **argv)
{
  vgui::init(argc, argv);

  // Load an image into our tableau:
  my_tableau my_tab("/users/kym/target/target/vgui/kym_examples/images/basement00.tif");

  // Put the tableau inside a 2D viewer tableau:
  vgui_viewer2D viewer(&my_tab);

  // Start event loop, using easy method.
  return vgui::run(&viewer, 512, 512);
}
```

We can use this to draw a line onto the image. In this next example a line is drawn between two points selected by pressing the left mouse button.

```
// mouse-draw-line.cxx
//
// This example a line is drawn by clicking on the end points.

#include <vgui/vgui.h>
#include <vgui/vgui_tableau.h>
#include <vgui/vgui_drag_mixin.h>
#include <vgui/vgui_projection_inspector.h>
#include <vgui/vgui_image_tableau.h>
#include <vgui/vgui_viewer2D.h>
#include <vgui/vgui_easy2D.h>

class my_tableau : public vgui_easy2D
{
public:
  float start_x;
  float start_y;

  my_tableau(vgui_image_tableau *imtab) :
    vgui_easy2D(imtab)
    {
      start_x = -1;
      start_y = -1;
    }
  ~my_tableau() { }

  bool handle(const vgui_event &e)
  {
    if (e.type == vgui_BUTTON_DOWN && e.button == vgui_LEFT && e.modifier == 0)
    {
      // Convert the coordinates from viewport coords to image coords:
      vgui_projection_inspector pi;
      float ix, iy;
      pi.window_to_image_coordinates(int(e.wx),int(e.wy), ix,iy);

      if (start_x == -1)
      {
        start_x = ix;
        start_y = iy;
      }
      else
      {
        this->set_line_width(3);
        this->add_line(start_x, start_y, ix, iy);
        start_x = start_y = -1;
      }
      cout << "selecting at " << e.wx << " " << e.wy << endl;
```

```
      return true; // event has been used
    }

    // We are not interested in other events, so pass event to base class:
    return vgui_easy2D::handle(e);
  }
};

int main(int argc,char **argv)
{
  vgui::init(argc, argv);

  // Load an image into an image tableau:
  vgui_image_tableau image("/users/kym/target/target/vgui/kym_examples/images/basement00.tif");

  // Put the image tableau inside our tableau:
  my_tableau my_tab(&image);

  vgui_viewer2D viewer(&my_tab);

  // Create a window, add the tableau and show it on screen:
  return vgui::run(&viewer, 512, 512);
}
```

4.2 Rubberbanding

A nicer way to add a line to an image is to use rubberbanding. VGUI contains a rubberbanding tableau (vgui_rubberbander) which takes a vgui_easy2D as its child. This next example allows the user to select either a line, circle or polygon to rubberband.

```
// rubberband.cxx
//
#include <vcl/vcl_cassert.h>
#include <vgui/vgui_image_tableau.h>
#include <vgui/vgui_zoomer.h>
#include <vgui/vgui_easy2D.h>
#include <vgui/vgui_load.h>
#include <vgui/vgui_rubberbander.h>
#include <vgui/vgui_menu.h>
#include <vgui/vgui.h>

static vgui_rubberbander* rubber;

static void point(const void *)
{
  cerr << __FILE__ " hello, fsm?" << endl;
  rubber->rubberband_point();
}

static void line(const void*)
{
  rubber->rubberband_line();
}

static void infinite_line(const void*)
{
  rubber->rubberband_infinite_line();
}
```

```
static void circle(const void*)
{
  rubber->rubberband_circle();
}

static void polygon(const void*)
{
  rubber->rubberband_polygon();
}
static void linestrip(const void *)
{
  rubber->rubberband_linestrip();
}
```
// **Create a vgui.menu:**
```
vgui_menu create_menus()
{
  vgui_menu objs;
  objs.add("point",point);
  objs.add("line",line);
  objs.add("infinite line", infinite_line);
  objs.add("circle",circle);
  objs.add("polygon", polygon);
  objs.add("linestrip",linestrip);
  vgui_menu bar;
  bar.add("Objects",objs);

  return bar;
}

int main(int argc, char **argv)
{
  vgui::init(argc, argv);

  assert(argv[1]);
  vgui_image_tableau image(argv[1]);
  vgui_easy2D easy(&image);
  rubber = new vgui_rubberbander(&easy);
  vgui_zoomer zoom(rubber);
  vgui_load load(&zoom);

  load.set_image(image.width(), image.height());

  return vgui::run(&load, image.width(), image.height(), create_menus());
}
```

Chapter 5

Displaying 3D objects

3D objects can be added using the easy3D tableau and putting it inside a viewer3D tableau gives functionality to zoom and move the object.

Pressing the shift key and left mouse button then dragging the mouse moves the object in 3 dimensions.

```
// 3d-example.cxx
//
// This example displays a point with 3 lines radiating from it in different
// non-coplanar directions (along the X, Y and Z axes).

//#include ¡vbl/ArgParse.h¿
#include <vgui/vgui.h>
#include <vgui/vgui_easy3D.h>
#include <vgui/vgui_viewer3D.h>

int main(int argc, char **argv)
{
  vgui::init(argc, argv);

  // Create a easy3D tableau:
  vgui_easy3D list3D;

  list3D.set_point_radius(15);
  // Add a point at the origin
  list3D.set_foreground(1,1,1); list3D.add_point(0,0,0);

  // Add a line in the xaxis:
  list3D.set_foreground(1,1,0);
  list3D.add_line(1,0,0, 4,0,0);
  list3D.add_point(5,0,0);

  // Add a line in the yaxis:
  list3D.set_foreground(0,1,0);
  list3D.add_line(0,1,0, 0,4,0);
  list3D.add_point(0,5,0);

  // Add a line in the zaxis:
  list3D.set_foreground(0,0,1);
  list3D.add_line(0,0,1, 0,0,4);
  list3D.add_point(0,0,5);
```

```
  // Add the easy3D tableau to a viewer3D tableau (this
  // gives us functionality to zoom and move the 3D object):
  vgui_viewer3D viewer(&list3D);

  return vgui::run(&viewer, 512, 512, "3d-example");
}
```

Chapter 6

More complicated GUIs

There are several reasons why you may wish to write your application without using the VGUI pattern. One reason is that only menus and dialog boxes are available using the pattern provided by VGUI. If you want to use more complicated widgets in your GUI (e.g. sliders, buttons) then you will have to write the application for one particular GUI toolkit only.

Alternatively you may be very familiar with a particular toolkit and not want to learn the VGUI pattern for menus etc (but remember this will make your code less portable).

As a first example here is the display image example again, but this time written in GTK. We still use the tableau code from VGUI to create the OpenGL area where the image is displayed and the `vgui_gtk_adaptor` class to plug the OpenGL area into our GTK code.

```
// gtk-display-image.cxx
//

#include <vgui/vgui.h>
#include <vgui/vgui_image_tableau.h>
#include <vgui/vgui_viewer2D.h>
#include <vgui/impl/gtk/vgui_gtk_adaptor.h>
#include <gtk/gtk.h>
#include <gtkgl/gtkglarea.h>

int main(int argc, char ** argv)
{
  // IMPORTANT NOTE, pass in "gtk" to init:
  vgui::init("gtk");
  // DO NOT DOUBLE INIT!  gtk_init(&argc, &argv);

  // Set up the tableaux as before:
  vgui_image_tableau image("/users/kym/target/target/vgui/kym_examples/images/basement00.tif");
  vgui_viewer2D viewer(&image);

  // Plug into a GL context :
  //vgui_gtk_adaptor *ct = new vgui_gtk_adaptor("Gtk Window", 512,512,0,0);
  vgui_gtk_adaptor *ct = new vgui_gtk_adaptor();
  GtkWidget *glarea= (( vgui_gtk_adaptor *)ct)->get_gtkwidget();
  gtk_widget_set_usize(glarea, 512, 512);
```

```
  gtk_widget_show(glarea);

  ct->set_tableau(&viewer);
  // Use GTK to create the window etc:
  GtkWidget *window= gtk_window_new(GTK_WINDOW_TOPLEVEL);
  GtkWidget *holderbox= gtk_vbox_new (FALSE, 0);

  gtk_box_pack_start (GTK_BOX (holderbox), glarea, TRUE, TRUE, 2);
  gtk_container_add(GTK_CONTAINER(window),GTK_WIDGET(holderbox));

  gtk_widget_show(holderbox); gtk_widget_show(window);
  return vgui::run();
}
```

In this next example we are writing a movie player and we want to use buttons to stop and start the movie. In order to do this the application has been written in GTK.

```
// gtk-movie-player.cxx
//

#include <vbl/vbl_sprintf.h>
#include <vgui/vgui_deck.h>
#include <vgui/vgui.h>

#include <vgui/vgui_load.h>
#include <vgui/vgui_image_tableau.h>
#include <vgui/vgui_zoomer.h>
#include <vcl/vcl_cstdlib.h>
#include <vgui/impl/gtk/vgui_gtk_adaptor.h>
#include <gtk/gtk.h>
#include <gtkgl/gtkglarea.h>

int idlecallbackid= -1;
int timer= 30;

gint idlecallback( gpointer context)
{
  ((vgui_deck*)context)->next();
  ((vgui_deck*)context)->post_redraw();
  return TRUE;
}

gint playimage(GtkWidget*, gpointer deckptr)
{
  idlecallbackid = gtk_timeout_add(timer, idlecallback, deckptr);
  return TRUE;
}

gint stopimage(GtkWidget*, gpointer deckptr)
{
  gtk_timeout_remove(idlecallbackid);
  idlecallbackid= -1;
  return TRUE;
}

int main(int argc, char ** argv)
{
  vgui::init("gtk", argc, argv);

  vgui_deck deck;

  // Load the images into image tableaux and add them to the deck:
  for (int i=1; i<10; ++i)
```

```
  {
    vbl_sprintf name("/users/pcp/bt/mit/bt.%03d.mit", i);
    vgui_image_tableau *image = new vgui_image_tableau(name);
    deck.add(image);
  }

  // then a zoomer :
  vgui_zoomer zr(&deck);

  // set initial image region :
  vgui_load vs(&zr);
  vs.set_ortho(0,400,400,0);


  // plug into a GL context :
  vgui_gtk_adaptor *ct = new vgui_gtk_adaptor;
  //vgui˙gtk˙adaptor *ct = vgui˙gtk˙adaptor();
  GtkWidget *glarea= ct->get_gtkwidget();
  gtk_widget_set_usize(GTK_WIDGET(glarea), 400, 400);

  // and set up the context
  ct->set_tableau(&vs);

  GtkWidget *window= gtk_window_new (GTK_WINDOW_TOPLEVEL);
  GtkWidget *buttonbox= gtk_hbox_new (FALSE, 0);
  GtkWidget *holderbox= gtk_vbox_new (FALSE, 0);
  GtkWidget *playbutton= gtk_button_new_with_label ("Play");
  GtkWidget *stopbutton= gtk_button_new_with_label ("Stop");

  gtk_box_pack_start (GTK_BOX (buttonbox), stopbutton, TRUE, TRUE, 5);
  gtk_box_pack_start (GTK_BOX (buttonbox), playbutton, TRUE, TRUE, 3);
  gtk_box_pack_start (GTK_BOX (holderbox), buttonbox, TRUE, TRUE, 4);
  gtk_box_pack_start (GTK_BOX (holderbox), glarea, TRUE, TRUE, 0);

  gtk_container_add(GTK_CONTAINER(window),GTK_WIDGET(holderbox));
  gtk_signal_connect (GTK_OBJECT (playbutton), "clicked", GTK_SIGNAL_FUNC (playimage), &deck);
  gtk_signal_connect (GTK_OBJECT (stopbutton), "clicked", GTK_SIGNAL_FUNC (stopimage), &deck);

  gtk_widget_show(playbutton);
  gtk_widget_show(stopbutton);
  gtk_widget_show(buttonbox);
  gtk_widget_show(holderbox);
  gtk_widget_show(glarea);
  gtk_widget_show(window);

  vgui::run();
}
```