

# CAD Algorithms for Physical Design - GTK+-based GUI

Christos P Sotiriou

1

CE439 - CAD Algorithms II 8/3/2016

## GTK+ Documentation

- ▶ **GTK/GDK (Gimp Tool Kit +) and Cairo**
  - ▶ <http://www.gtk.org/>
  - ▶ <https://developer.gnome.org/gtk2/stable/>
  - ▶ <https://developer.gnome.org/gdk2/stable/>
  - ▶ <https://developer.gnome.org/gtk2/stable/gtk2-General.html>
  - ▶ <http://zetcode.com/gui/gtk2/gtkevents/>
  - ▶ <https://developer.gnome.org/gdk2/stable/gdk2-Cairo-Interaction.html>
  - ▶ <http://cairographics.org/documentation/>
  - ▶ <https://developer.gnome.org/glib/stable/>



cairo



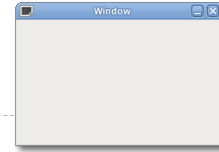
▶ 2

CE439 - CAD Algorithms II 8/3/2016

## GTK+ Widgets - 1

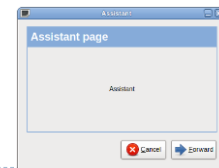
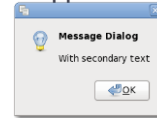
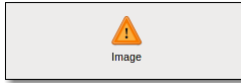
### ▶ Windows

- ▶ GtkDialog — Create popup windows
- ▶ GtkMessageDialog
- ▶ GtkWindow — Toplevel which can contain other widgets
- ▶ GtkAboutDialog — Display information about an application
- ▶ GtkAssistant



### ▶ Display Widgets

- ▶ GtkImage
- ▶ GtkLabel — A widget that displays a small to medium amount of text
- ▶ GtkProgressBar — A widget which indicates progress visually
- ▶ GtkStatusbar — Report messages of minor importance to the user
- ▶ GtkInfoBar
- ▶ GtkStatusIcon — Display an icon in the system tray
- ▶ GtkSpinner



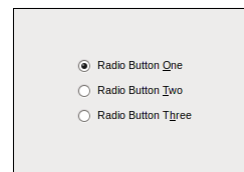
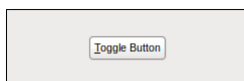
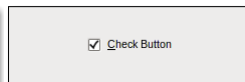
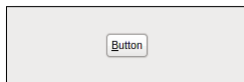
▶ 3

CE439 - CAD Algorithms II 8/3/2016

## GTK+ Widgets - 2

### ▶ Buttons and Toggles

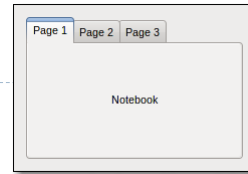
- ▶ GtkButton — A widget that creates a signal when clicked on
- ▶ GtkCheckButton — Create widgets with a discrete toggle button
- ▶ GtkRadioButton — A choice from multiple check buttons
- ▶ GtkToggleButton — Create buttons which retain their state
- ▶ GtkLinkButton — Create buttons bound to a URL
- ▶ GtkScaleButton — A button which pops up a scale
- ▶ GtkVolumeButton



▶ 4

CE439 - CAD Algorithms II 8/3/2016

## GTK+ Widgets - 3



### ▶ Layout Containers

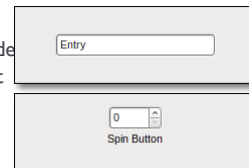
- ▶ GtkHBox
- ▶ GtkVBox
- ▶ GtkHButtonBox — A container for arranging buttons horizontally
- ▶ GtkVButtonBox
- ▶ GtkFixed — A container which allows you to position widgets at fixed coordinates
- ▶ **GtkHPaned** — A container with two panes arranged horizontally
- ▶ **GtkVPaned**
- ▶ GtkLayout — Infinite scrollable area containing child widgets and/or custom drawing
- ▶ GtkNotebook — A tabbed notebook container
- ▶ GtkTable — Pack widgets in regular patterns
- ▶ GtkExpander — A container which can hide its child
- ▶ GtkOrientable — An interface for flippable widgets

▶ 5

CE439 - CAD Algorithms II 8/3/2016

## GTK+ Widgets - 4

- ▶ Action-based menus and toolbars
  - ▶ GtkUIManager — Constructing menus and toolbars from an XML description
- ▶ Interface builder
  - ▶ GtkBuilder — Build an interface from an XML UI definition
- ▶ Scrolling
  - ▶ **GtkHScrollbar** — A horizontal scrollbar
  - ▶ **GtkVScrollbar**
  - ▶ GtkAdjustment — A GObject representing an adjustable bound
  - ▶ GtkRange — Base class for widgets which visualize an adjustment
  - ▶ GtkScrolledWindow — Adds scrollbars to its child widget
- ▶ Numeric/Text Data Entry
  - ▶ GtkEntry — A single line text entry field
  - ▶ GtkEntryBuffer — Text buffer for GtkEntry
  - ▶ GtkEntryCompletion — Completion functionality for GtkEntry
  - ▶ GtkHScale — A horizontal slider widget for selecting a value from a range
  - ▶ GtkVScale
  - ▶ GtkSpinButton — Retrieve an integer or floating-point number from the user
  - ▶ GtkEditable — Interface for text-editing widgets

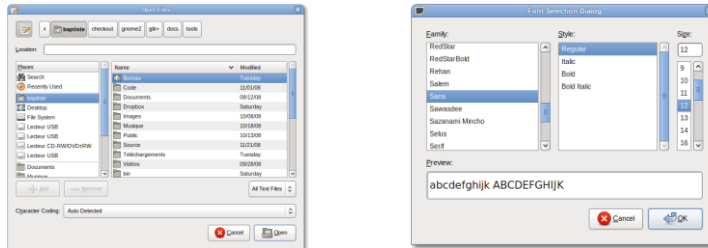


▶ 6

CE439 - CAD Algorithms II 8/3/2016

## GTK+ Widgets - 5

- ▶ **Selectors (File/Font/Color/Input Devices)**
  - ▶ **GtkFileChooserDialog** — A file chooser dialog, suitable for "File/Open" or "File/Save" commands
  - ▶ **GtkFileChooserWidget** — File chooser widget that can be embedded in other widgets
  - ▶ **GtkFileFilter** — A filter for selecting a file subset
  - ▶ **GtkFontSelectionDialog** — A dialog box for selecting fonts



▶ 7

CE439 - CAD Algorithms II 8/3/2016

## GDK Cursors

- ▶ <https://developer.gnome.org/gdk3/stable/gdk3-Cursors.html>

Creates a new cursor from the set of builtin cursors. Some useful ones are:

- GDK\_RIGHT\_PTR (right-facing arrow)
- GDK\_CROSSHAIR (crosshair)
- GDK\_XTERM (I-beam)
- GDK\_WATCH (busy)
- GDK\_FLEUR (for moving objects)
- GDK\_HAND1 (a right-pointing hand)
- GDK\_HAND2 (a left-pointing hand)
- GDK\_LEFT\_SIDE (resize left side)
- GDK\_RIGHT\_SIDE (resize right side)
- GDK\_TOP\_LEFT\_CORNER (resize northwest corner)
- GDK\_TOP\_RIGHT\_CORNER (resize northeast corner)
- GDK\_BOTTOM\_LEFT\_CORNER (resize southwest corner)
- GDK\_BOTTOM\_RIGHT\_CORNER (resize southeast corner)
- GDK\_TOP\_SIDE (resize top side)
- GDK\_BOTTOM\_SIDE (resize bottom side)
- GDK\_SB\_H\_DOUBLE\_ARROW (move vertical splitter)
- GDK\_SB\_V\_DOUBLE\_ARROW (move horizontal splitter)

▶ 8

CE439 - CAD Algorithms II 8/3/2016

# Widget Properties - 1

► All widgets possess properties:

**GtkCellRendererText**  
 GtkCellRendererText — Renders text in a cell

**Functions**

```
GtkCellRenderer * gtk_cell_renderer_text_new ()
void gtk_cell_renderer_text_set_fixed_height_from_font ()
```

**Properties**

gboolean	align-set	Read / Write
PangoAlignment	alignment	Read / Write
PangoAttrList *	attrib-bes	Read / Write
gchar *	background	Write
GtkColor *	background-gdk	Read / Write
gboolean	background-set	Read / Write
gboolean	editable	Read / Write

gdouble	scale	Read / Write
gboolean	scale-set	Read / Write
gboolean	single-paragraph-mode	Read / Write
gint	size	Read / Write
gdouble	size-points	Read / Write
gboolean	size-set	Read / Write
PangoStretch	stretch	Read / Write
gboolean	stretch-set	Read / Write
gboolean	strikethrough	Read / Write
gboolean	strikethrough-set	Read / Write
PangoStyle	style	Read / Write
gboolean	style-set	Read / Write
gchar *	text	Read / Write
PangoUnderline	underline	Read / Write
gboolean	underline-set	Read / Write
PangoVariant	variant	Read / Write
gboolean	variant-set	Read / Write
gint	weight	Read / Write
gboolean	weight-set	Read / Write
gint	width-chars	Read / Write
PangoWrapMode	wrap-mode	Read / Write
gint	wrap-width	Read / Write

**The "scale" property**

"scale" gdouble

Font scaling factor.  
 Flags: Read / Write  
 Allowed values: >= 0  
 Default value: 1

► 9

# Widget Properties - 2

► Modify

**g\_object\_get ()**

```
void
g_object_get (Gpointer object,
             const gchar *first_property_name,
             ...);
```

Gets properties of an object.

In general, a copy is made of the property contents and the caller is responsible for freeing the memory in the appropriate manner for the type, for instance by calling `g_free()` or `g_object_unref()`.

Here is an example of using `g_object_get()` to get the contents of three properties: an integer, a string and an object:

```
gint intval;
gchar *strval;
GObject *objval;

g_object_get (my_object,
             "int-property", &intval,
             "str-property", &strval,
             "obj-property", &objval,
             NULL);

// Do something with intval, strval, objval

g_free (strval);
g_object_unref (objval);
```

Parameters

object	a GObject
first_property_name	name of the first property to get
...	return location for the first property, followed optionally by more name/return location pairs, followed by <code>NULL</code>

► // set font size hierarchy g\_obj

ale the NULL);

► 10

## Widget Signals - 1

### ► All widgets possess signals:

Signals	
void	accel-closures-changed
gboolean	button-press-event
gboolean	button-release-event
gboolean	can-activate-accel
void	child-notify
gboolean	client-event
void	composited-changed
gboolean	configure-event
gboolean	damage-event
gboolean	delete-event
gboolean	destroy-event
void	direction-changed
void	drag-begin
void	drag-data-delete
void	drag-data-get
void	drag-data-received
gboolean	drag-drop
void	drag-end
gboolean	drag-failed
void	drag-leave
gboolean	drag-motion
gboolean	enter-notify-event
gboolean	event
gboolean	expose-event
gboolean	focus
gboolean	focus-in-event
gboolean	focus-out-event
gboolean	grab-broken-event
void	grab-focus
void	grab-notify
void	hide
void	hierarchy-changed
gboolean	key-press-event
gboolean	key-release-event
gboolean	keynav-failed
gboolean	leave-notify-event
void	map
gboolean	map-event
gboolean	mnemonic-activate
gboolean	motion-notify-event
void	move-focus
gboolean	no-expose-event
void	parent-set
gboolean	popup-menu
gboolean	property-notify-event
gboolean	proximity-in-event
gboolean	proximity-out-event
gboolean	query-tooltip
void	realize
void	screen-changed
gboolean	scroll-event
gboolean	selection-clear-event
void	selection-get
gboolean	selection-notify-event
void	selection-received
gboolean	selection-request-event
void	show
gboolean	show-help
void	size-allocate
void	size-request
void	state-changed
void	style-set
void	unmap
gboolean	unmap-event
void	unrealize
gboolean	visibility-notify-event
gboolean	window-state-event

► 11

CE439 - CAD Algorithms II 8/3/2016

## Widget Signals - 2

### ► <https://developer.gnome.org/gobject/unstable/gobject-Signals.html#g-signal-connect>

#### **g\_signal\_connect()**

```
#define g_signal_connect(instance, detailed_signal, c_handler, data)
```

Connects a GCallback function to a signal for a particular object.

The handler will be called before the default handler of the signal.

See memory management of signal handlers for details on how to handle the return value and memory management of `data`.

#### Parameters

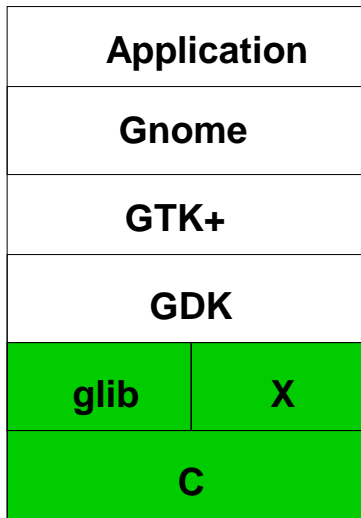
instance	the instance to connect to.
detailed_signal	a string of the form "signal-name::detail".
c_handler	the GCallback to connect.
data	data to pass to <code>c_handler</code> calls.

#### Returns

the handler id (always greater than 0 for successful connections)

► 12

CE439 - CAD Algorithms II 8/3/2016



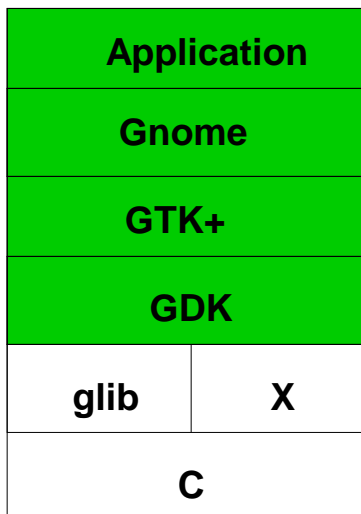
- X11 Graphics Library : *low-level functions to control the display*
- glib : *library of C functions, macros and structs used by GDK, GTK+ and GNOME*
- C Standard Libraries
- Linux System Calls



Figure 1.1: The levels of software for a GNOME application in Linux



CE439 - CAD Algorithms II 8/3/2016



- Gnome application program : *you will write this*
- GNOME : *extension of GTK+, specialized widgets*
- GIMP Toolkit: *organizes the GDK functions into objects providing the functionality of widgets*
- GIMP Drawing Kit : *simplifies access to X functions*



Figure 1.1: The levels of software for a GNOME application in Linux



CE439 - CAD Algorithms II 8/3/2016



## OOP in C



Object oriented programming (OOP) is a method of organizing the data and code.

The GTK and GNOME libraries are written in C.

Some languages (C++, Java) automate object oriented programming and provide language features to enforce the OO data access protocols.

GTK+ and GNOME libraries are written in C, thus the object orientation must be simulated.

▶ 15

CE439 - CAD Algorithms II 8/3/2016



## Terminology



Encapsulation - *object encapsulates its data. Data can only be accessed through function calls designed for that purpose.*

Inheritance - *properties that are available through a base class upon which a new class is based.*

Methods - *the above mentioned functions also used to create a new object from a class.*

Class - *definition code used to construct an object*

Object - *a particular instance of a class*

▶ 16

CE439 - CAD Algorithms II 8/3/2016





## Terminology

---

Each of the “classes” in GTK+ are actually created using **structs**. These **structs** contain data fields which serve the purpose that private data members would in C++. They are not protected from improper access.

“Methods” for each class of object are simple functions, whose first parameter is the type of object on which the function is defined.

Inheritance is achieved by including the entire struct of a base class.

▶ 17

CE439 - CAD Algorithms II 8/3/2016



## Terminology

---

**Event** - *sent to your program when a hardware action such as a keystroke, movement of the mouse, pressing of a mouse button, takes place or when X window changes occur in the display (window appears, window is uncovered, etc)*

**Signal** - *Events are translated into signals. They are sent to your program when one or several hardware actions take place or when certain things occur within a widget (scroll bar moves, button is pressed)*

**Callback function** - *a function which is called by your program upon receipt of a particular signal*

▶ 18

CE439 - CAD Algorithms II 8/3/2016



## Example p.24

```

/** gtkwin.c */
#include <gtk/gtk.h>

int main(int argc, char *argv[]){
    GtkWidget *topLevelWindow;

    gtk_init(&argc, &argv);
    topLevelWindow =
        gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_widget_show(topLevelWindow);
    gtk_main();
    exit(0);
}

```

▶ 19

CE439 - CAD Algorithms II 8/3/2016



## makefile

```

CC=gcc
LDLIBS=`gtk-config --libs`
CFLAGS=-Wall -g `gtk-config --cflags`

gtkwin: gtkwin.o
    $(CC) $(LDLIBS) gtkwin.o -o gtkwin

gtkwin.o: gtkwin.c
    $(CC) $(CFLAGS) -c gtkwin.c

clean:
    rm -f gtkwin
    rm -f *.o

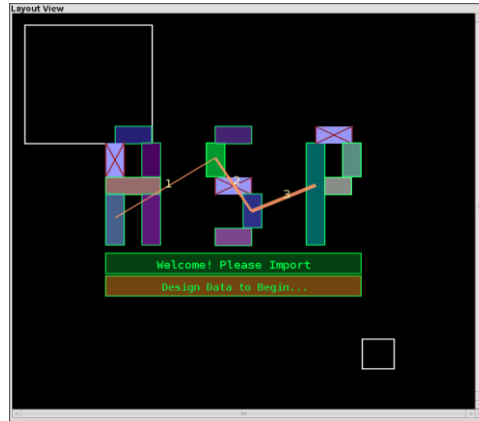
```

▶ 20

CE439 - CAD Algorithms II 8/3/2016

## Layout Canvas with Scrollbars

### ► LayoutView Demo



► 21

CE439 - CAD Algorithms II 8/3/2016

## GtkDrawingArea Widget

**GtkDrawingArea**

GtkDrawingArea — A widget for custom user interface elements

**Functions**

`GtkWidget * gtk_drawing_area_new()`

`void`

**Types and Values**

`struct`

**Object Hierarchy**

```

GObject
├── GInitiallyUnfinalized
├── GtkWidget
├── GtkDrawingArea
│   ├── GtkCurve
│   └── GtkSpinner

```

**Description**

The GtkDrawingArea widget is used for creating custom user interface elements. It's essentially a blank widget; you can draw on `widget->window`. After creating a drawing area, the application may want to connect to:

- Mouse and button press signals to respond to input from the user. (Use `gtk_widget_add_events()` to enable events you wish to receive.)
- The "realize" signal to take any necessary actions when the widget is instantiated on a particular display. (Create GDK resources in response to this signal.)
- The "configure\_event" signal to take any necessary actions when the widget changes size.
- The "expose\_event" signal to handle redrawing the contents of the widget.

The following code portion demonstrates using a drawing area to display a circle in the normal widget foreground color. Note that GDK automatically clears the exposed area to the background color before sending the expose event, and that drawing is implicitly clipped to the exposed area.

► 22

CE439 - CAD Algorithms II 8/3/2016

# Layout Canvas with Scrollbars Example

## Layout Canvas with Scrollbars Example – page 1

```
int main()
{
    GtkWidget *maincansvashbox; // contains maincanvas and vertical scrollbar //
    GtkWidget *maincansvshbox; // contains maincansvashbox and horizontal scrollbar //
    GtkWidget *mainframe; // main canvas frame //

    gtk_init(NULL, NULL);

    aspwindow = gtk_window_new(GTK_WINDOW_TOPLEVEL);

    gtk_window_set_title(GTK_WINDOW(aspwindow), "ASP");
    gtk_window_set_default_size(GTK_WINDOW(aspwindow), 300, 300); // default size //
    g_signal_connect(G_OBJECT(aspwindow), "destroy", G_CALLBACK(quitaction), aspwindow);

    maincansvashbox = gtk_hbox_new(FALSE, 0);
    maincansvshbox = gtk_vbox_new(FALSE, 0);

    mainframe = gtk_frame_new("Layout View");
    maincanvas = gtk_drawing_area_new();
    gtk_widget_set_size_request(maincanvas, maincansvWidth, maincansvHeight);

    // maincanvas Event Handlers //
    g_signal_connect(G_OBJECT(maincanvas), "expose-event", G_CALLBACK(expose), maincanvas);

    gtk_widget_add_events(maincanvas, GDK_SCROLL);
    gtk_widget_add_events(maincanvas, GDK_BUTTON_PRESS_MASK);

    g_signal_connect(G_OBJECT(maincanvas), "scroll-event", G_CALLBACK(scroll), maincanvas);
    g_signal_connect(G_OBJECT(maincanvas), "button-press-event", G_CALLBACK(mousebutton), maincanvas);

    maincansvscrollbaradjustment = gtk_adjustment_new(0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
    maincansvshscrollbaradjustment = gtk_adjustment_new(0.0, 0.0, 0.0, 0.0, 0.0, 0.0); . . .
}
```

▶ 23

CE439 - CAD Algorithms II 8/3/2016

# Layout Canvas with Scrollbars Example

## Layout Canvas with Scrollbars Example – page 2

```
. . .

    maincansvscrollbar = gtk_vscrollbar_new(GTK_ADJUSTMENT(maincansvscrollbaradjustment));
    maincansvshscrollbar = gtk_hscrollbar_new(GTK_ADJUSTMENT(maincansvshscrollbaradjustment));

    g_signal_connect(G_OBJECT(maincansvscrollbar), "change-value", G_CALLBACK(maincansvscroll),
    maincansvscrollbar);
    g_signal_connect(G_OBJECT(maincansvshscrollbar), "change-value", G_CALLBACK(maincansvshscroll),
    maincansvshscrollbar);

    gtk_box_pack_start(GTK_BOX(maincansvashbox), maincanvas, FALSE, FALSE, 0);
    gtk_box_pack_start(GTK_BOX(maincansvashbox), maincansvscrollbar, FALSE, FALSE, 0);

    gtk_box_set_child_packing(GTK_BOX(maincansvashbox), maincanvas, TRUE, TRUE, 0, GTK_PACK_START);
    gtk_box_set_child_packing(GTK_BOX(maincansvashbox), maincansvscrollbar, FALSE, FALSE, 0, GTK_PACK_END);

    gtk_box_pack_start(GTK_BOX(maincansvshbox), maincansvashbox, FALSE, FALSE, 0);
    gtk_box_pack_start(GTK_BOX(maincansvshbox), maincansvshscrollbar, FALSE, FALSE, 0);

    gtk_box_set_child_packing(GTK_BOX(maincansvshbox), maincansvashbox, TRUE, TRUE, 0, GTK_PACK_START);
    gtk_box_set_child_packing(GTK_BOX(maincansvshbox), maincansvshscrollbar, FALSE, FALSE, 0, GTK_PACK_END);

    gtk_container_add(GTK_CONTAINER(mainframe), maincansvshbox);

    g_signal_connect(G_OBJECT(maincanvas), "size-allocate", G_CALLBACK(resizemaincanvas), maincanvas);

    gtk_container_add(GTK_CONTAINER(aspwindow), mainframe);

    gtk_widget_show_all(aspwindow);

    gtk_main();
}
```

▶ 24

CE439 - CAD Algorithms II 8/3/2016

## GTK to Cairo Interaction

- ▶ <http://cairographics.org/documentation/>
- ▶ <http://cairographics.org/manual/>
- ▶ <http://cairographics.org/tutorial/>

```
// Expose-Event Paint Function for maincanvas //
static void maincanvaspaint(GtkWidget *widget, GdkEventExpose *event, gpointer data)
{
    maincanvas_drawable = maincanvas->window; // drawable drawing area window //
    maincanvas_cs = gdk_cairo_create(maincanvas_drawable); // corresponding cairo state //
    // draw ... in Cairo //
    cairo_destroy(maincanvas_cs);
}
```



▶ 25

CE439 - CAD Algorithms II 8/3/2016

## Cairo Drawing Functions in a Nutshell

- ▶ `cairo_move_to(cairo_t *cr, double x, double y);`
- ▶ `cairo_line_to(cairo_t *cr, double x, double y);`
- ▶ `cairo_rectangle(cairo_t *cr, double x, double y, double width, double height);`
- ▶ `cairo_rel_line_to(cairo_t *cr, double x, double y);`
- ▶ `cairo_rel_move_to(cairo_t *cr, double x, double y);`
- ▶ `cairo_select_font_face(cairo_t *cr, const char *family, cairo_font_slant_t slant, cairo_font_weight_t weight);`
- ▶ `cairo_set_font_size(cairo_t *cr, double size);`
- ▶ `cairo_show_text(cairo_t *cr, const char *utf8);`
- ▶ `cairo_set_source_rgb(cairo_t *cr, double red, double green, double blue);`
- ▶ `cairo_set_source_rgba(cairo_t *cr, double red, double green, double blue, double alpha);`

▶ 26

CE439 - CAD Algorithms II 8/3/2016