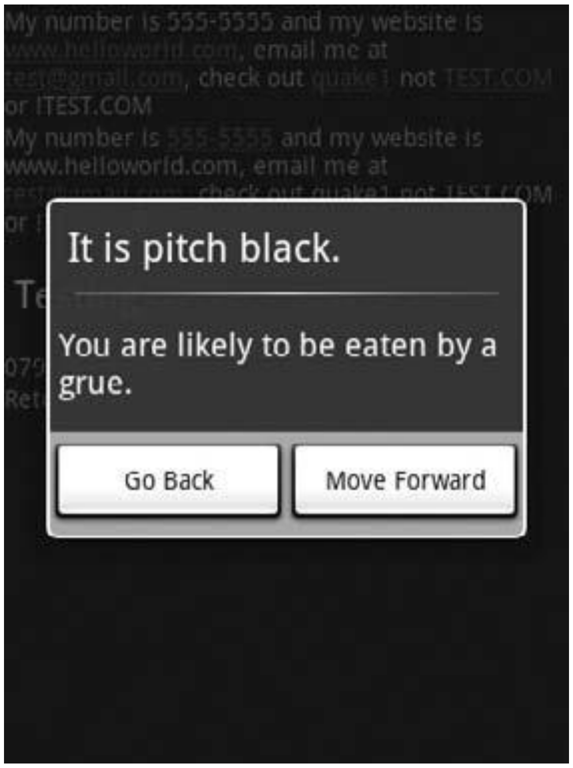
Introducing Dialogs

Dialog boxes are a common UI metaphor in desktop and web applications. They're used to help users answer questions, make selections, confi rm actions, and read warning or error messages. An *Android Dialog* is a fl oating window that partially obscures the Activity that launched it.

As you can see in Figure 5-5, Dialog boxes are not full screen and can be partially transparent. They generally obscure the Activities behind them using a blur or dim filter.



My number is 555-5555 and my website is www.helloworld.com, email me at test@gmail.com, check out quake1 not TEST.COM or ITEST.COM
My number is 555-5555 and my website is www.helloworld.com, email me at test@gmail.com, check out quake1 not TEST.COM

It is pitch black.

You are likely to be eaten by a grue.

Figure 5-5

There are three ways to implement a Dialog box in Android:

❑ **Using a Dialog-Class Descendent** As well as the general-purpose AlertDialog class, Android includes several specialist classes that extend Dialog. Each is designed to provide specific Dialog-box functionality. Dialog-class-based screens are constructed entirely within their calling Activity, so they don't need to be registered in the manifest, and their life cycle is controlled entirely by the calling Activity.

❑ **Dialog-Themed Activities** You can apply the Dialog theme to a regular Activity to give it theappearance of a Dialog box.

❑ **Toasts** Toasts are special non-modal transient message boxes, often used by Broadcast Receivers and background services to notify users of events. You learn more about Toasts in Chapter 8.

Introducing the* Dialog *Class

The Dialog class implements a simple fl oating window that is constructed entirely within an Activity.
To use the base Dialog class, you create a new instance and set the title and layout as shown below:

```
Dialog d = new Dialog(MyActivity.this);
// Have the new window tint and blur the window it
// obscures.
Window window = d.getWindow();
window.setFlags(WindowManager.LayoutParams.FLAG_BLUR_BEHIND,
WindowManager.LayoutParams.FLAG_BLUR_BEHIND);
d.setTitle("Dialog Title");
d.setContentView(R.layout.dialog_view);
TextView text = (TextView)d.findViewById(R.id.dialogTextView);
text.setText("This is the text in my dialog");
```

Once it's confi gured to your liking, use the show method to display it. d.show();

The Alert Dialog *Class*

The AlertDialog class is one of the most versatile Dialog implementations. It offers various options that let you construct screens for some of the most common Dialog-box use cases, including:

❑ Presenting a message to the user offering one to three options in the form of alternative buttons. This functionality is probably familiar to you if you've done any desktop programming, where the buttons presented are usually a selection of OK, Cancel, Yes, or No.

❑ Offering a list of options in the form of check buttons or radio buttons.

❑ Providing a text entry box for user input.

To construct the Alert Dialog user interface, create a new AlertDialog.Builder object, as shown below:

```
AlertDialog.Builder ad = new AlertDialog.Builder(context);
```

You can then assign values for the title and message to display, and optionally assign values to be used for any buttons, selection items, and text input boxes you wish to display. That includes setting event listeners to handle user interaction.

The following code gives an example of a new Alert Dialog used to display a message and offer two button options to continue. Clicking on either button will automatically close the Dialog after executing the attached Click Listener.

```
Context context = MyActivity.this;
String title = "It is Pitch Black";
String message = "You are likely to be eaten by a grue.";
String button1String = "Go Back";
String button2String = "Move Forward";
AlertDialog.Builder ad = new AlertDialog.Builder(context);
ad.setTitle(title);
ad.setMessage(message);
ad.setPositiveButton(button1String,
new OnClickListener() {
public void onClick(DialogInterface dialog,
int arg1) {
eatenByGrue();
}
});
ad.setNegativeButton(button2String,
new OnClickListener(){
public void onClick(DialogInterface dialog,
int arg1) {
// do nothing
}
});
ad.setCancelable(true);
ad.setOnCancelListener(new OnCancelListener() {

public void onCancel(DialogInterface dialog) {
eatenByGrue();
}
});
```

To display an Alert Dialog that you've created call show. ad.show();

Alternatively, you can override the onCreateDialog and onPrepareDialog methods within your Activity to create single-instance Dialogs that persist their state. This technique is examined later in this chapter.

Specialist Input Dialogs

One of the major uses of Dialog boxes is to provide an interface for user input. Android includes several specialist Dialog boxes that encapsulate controls designed to facilitate common user input requests.

They include the following:

❑ **DatePickerDialog** Lets users select a date from a DatePicker View. The constructor includes a callback listener to alert your calling Activity when the date has been set.

❑ **TimePickerDialog** Similar to the DatePickerDialog, this Dialog lets users select a time from a TimePicker View.

❑ **ProgressDialog** A Dialog that displays a progress bar beneath a message textbox. Perfect for keeping the user informed of the ongoing progress of a time-consuming operation.