Product Information 2

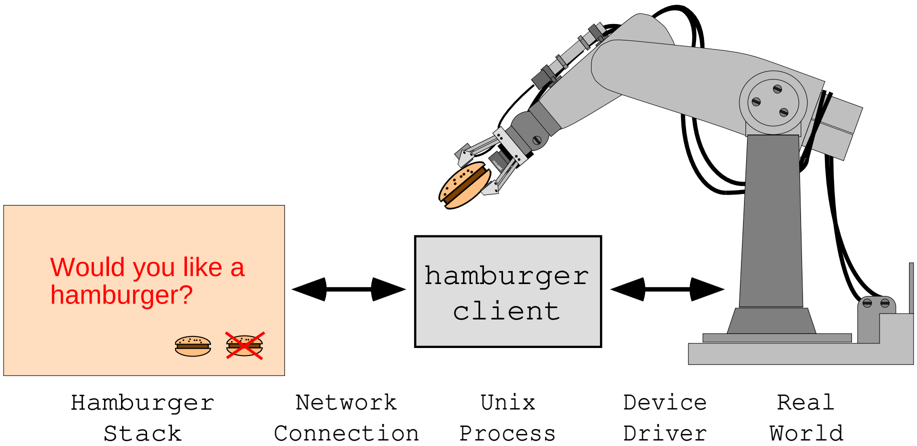
Open Windows With HyperLook*

HyperLook is an interactive application design system, that lets you quickly develop advanced multimedia systems, via simple direct manipulation, property editors, and object oriented programming.

You design interfaces by taking fully functional components from an object warehouse. You lay them out in your own window, configure them with menus and property editors, define their appearance in colorful PostScript† fonts and graphics, and write scripts to customize their behavior.

Figure 1

Interfacing HyperLook to the real world.



You can write applications in C or other languages, that communicate with HyperLook by sending high level messages across the network. You need not worry about details like layout, look and feel, or fonts and colors. You can edit HyperLook applications while they're running, or package them into a runtime form.

---

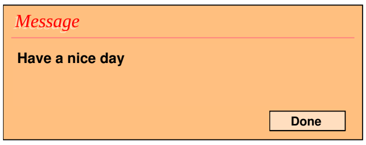
*. HyperLook is a trademark of the Turing Institute Ltd.
†. PostScript is a registered trademark of Adobe Systems Inc.

HyperLook is totally extensible and open ended. It comes with a toolkit of user interface classes, and warehouses of pre-configured components with useful behavior. The next few pages show some ways HyperLook can be used.

Applications To Go!

HyperLook uses the stack/card/object model. Windows are called stacks. Each stack can have several cards, and objects live on the cards. Below is a HyperLook stack with two buttons and a text field. Its name is “Hamburger” because of the buttons. This stack was created using the HyperLook graphics editor, no programming was involved.

On the right is a hamburger stack. Below is its associated message stack (at a different scale). You have chosen to eat today!



The message stack is a standard system stack. In this example we want to press either of the hamburger buttons and have a message come up in the message stack. If the crossed button is pressed the message says “Please reconsider my offer!”, if the other hamburger is pressed the message is “Have a nice day” and the hamburger stack disappears.

This behavior is obtained by providing *scripts* for the two buttons. HyperLook scripts are written in PostScript. A script contains *methods* for an object. The uncrossed button’s script is:

```
/Action {
   [(Have a nice day)] /Init /Message Send
   MyStack HideStack
} def
```

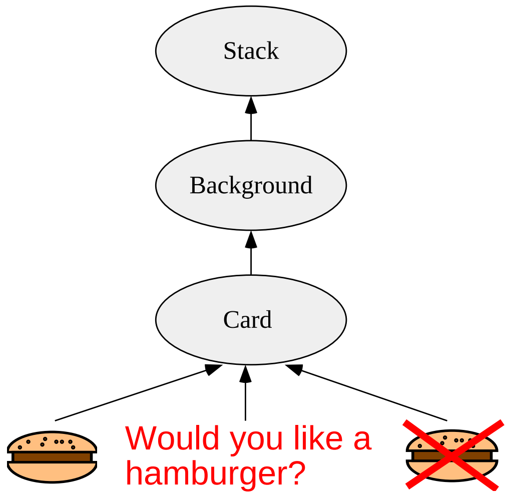
When the button is pressed it receives an **`Action`** message. The Action method is then executed. This sends the Message stack an **`Init`** message. The Message stack's **`Init`** method shows the stack if it's hidden or not loaded and displays the argument string, which in this case is"Have a nice day". Once the message has been delivered the stack is hidden. This is done by the code "**`MyStack HideStack`**".

The second button has a simpler script, since the Hamburger stack does not need to be hidden:

```
/Action {
    [(Please reconsider my offer!)] /Init /Message Send
} def
```

So, what happens when the buttons are pressed? Almost all HyperLook programming can be done by passing messages between objects. The Hamburger stack is an object. So are the two buttons. An exploded view of the objects in the stack is shown below.

An exploded view of the Hamburger stack, Messages follow the direction of the arrows.



You can see that the two hamburgers are children of the *card*, which is the child of the *background*, which has the *stack* as its parent. When a button is pressed an **`Action`** message is sent to it. If the button doesn't have an **`Action`** method, the message passesto the button's parent. If the parent can't handle the message, it is passed on. If the stack can't handle the message it is passed to a client if there is one. If there isn't a client then the message disappears. When we add the **Action** method to a button's script, the message is handled by the button as happens here.

The Hamburger in C

Programming can be done in HyperLook's scripting language. C or other languages can also be used. If we compile and execute the C program below we have the same effect as the scripts above. In this program the C process is responsible for handling the **Action** messages.

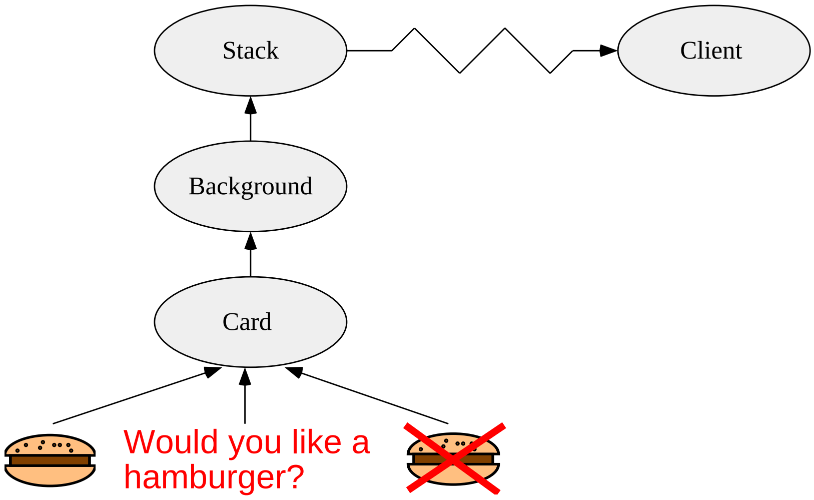
```
#include <hyperlook.h>
int yes()
{
    hl_send0("Message", "Init",
        hl_new_string("Have a nice day"),
        0);
    return -1;
}

int no()
{
    hl_send0("Message", "Init",
         hl_new_string("Please reconsider my offer!"),
         0);
    return 1;
}

main(argc, argv)
{
    hl_start(argv[0]);
    hl_path(0);
    hl_connect("Hamburger");
    hl_show("Hamburger");
    hl_register(yes, "Hamburger:yes", "Action");
    hl_register(no, "Hamburger:no", "Action");
    hl_listen(hl_forever);
    hl_hide("Hamburger");
    hl_stop();
}
```

The yes and no C functions correspond to the scripts of the two buttons, whose names are **yes** and **no**. The **main** function is resonsible for setting up the connection between the client and HyperLook. When this happess the conmmunication path is as shown below.

An exploded view of the Hamburger stack, Messages follow the direction of the arrows.



The **hl_connect** function sets up the network connection between the client and the Hamburger stack. Whenever a message passes to the client it is matched against all the registered patterns of message, and dispatched accordingly.

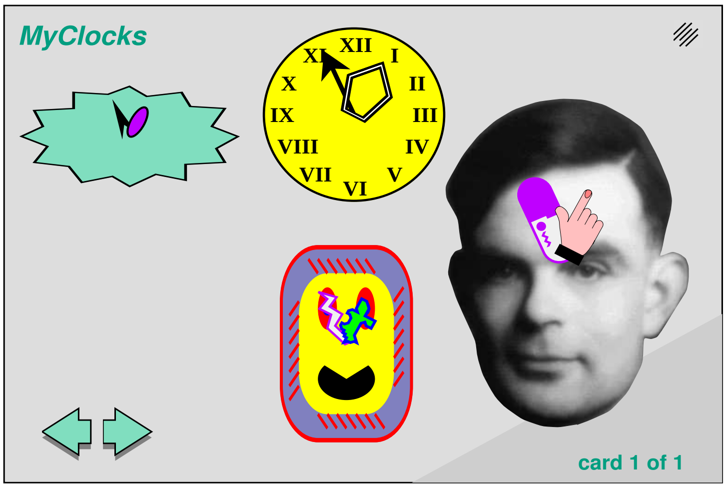
If the uncrossed button (called **yes**) is pressed it gets an **Action** message. Since it has no method for this message, it is passed up the hierarchy. None of its ancestors can handle the message, which is passed to the client. The client dispatches the message to the yes function, which in turn sends a message to the Message stack. The return value of **-1** means that the dispatch loop started with **hl_listen** ends, the Hamburger stack is hidden and the process exits.

Creating New Classes

With HyperLook it is possible to create new classes of object with interesting functionality. These objects can be used by other people without having to understand the programming details. This is the use of abstract data types and encapsulation in a visual environment.

To illustrate what can be done the picture below shows several clocks. Each clock is constructed from three drawings.

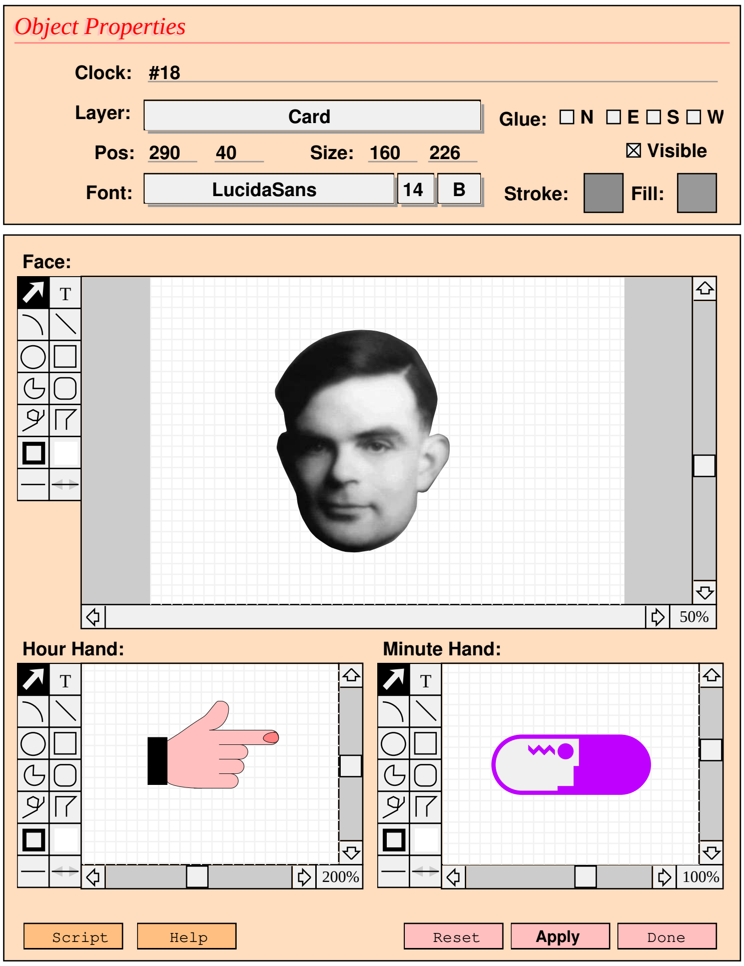
Some designer clocks. Each clock is an instance of a user constructed class.



There is one drawing for the face, and one each for the hands. The clock is a new class. It has its own property editor, shown below. HyperLook is organized to make it easy for designers to use the clock object.

The components can be given any look by using the graphics editors in the property sheet. If a clock is installed in a warehouse stack then that version of the clock class can be loaded into a stack from a menu.

The property editor for the designer clocks. Each of the three components has its own graphics editor. Many designs of clock can be constructed without programming.



This class, the demonstration stack and the property sheet were created inside 2 hours by an experienced HyperLook programmer.

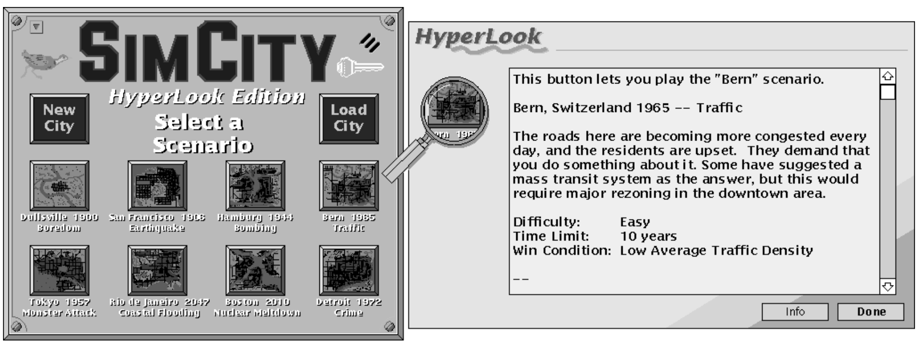
Sophisticated Applications

HyperLook is a powerful platform for developing production quality applications. It's ease of use not only enables non-programmers to design graphical user interfaces, but also gives skilled programmers the leverage to develop sophisticated multimedia applications. Instead of trying to make an interface that works, you can concentrate on making the interface you want. HyperLook supports rapid prototyping and iterative refinement, so you can perfect your application in response to user feedback.

*SimCity**, the award winning city simulation game from Maxis, ported to HyperLook by DUX, demonstrates the capabilities of HyperLook. It integrates real time audio, animation, colorful PostScript fonts and imaging, on-line help, OPEN LOOK†, and customized user interface components such as graphical buttons, pie menus, zooming and scrolling views, and fancy window frames.

On-line help

Here we have an example of the integrated help facilities of HyperLook, which can be implemented without programming. Help for any object can be obtained with the **Help** key. Designers can tailor the help text, or even produce graphical help.

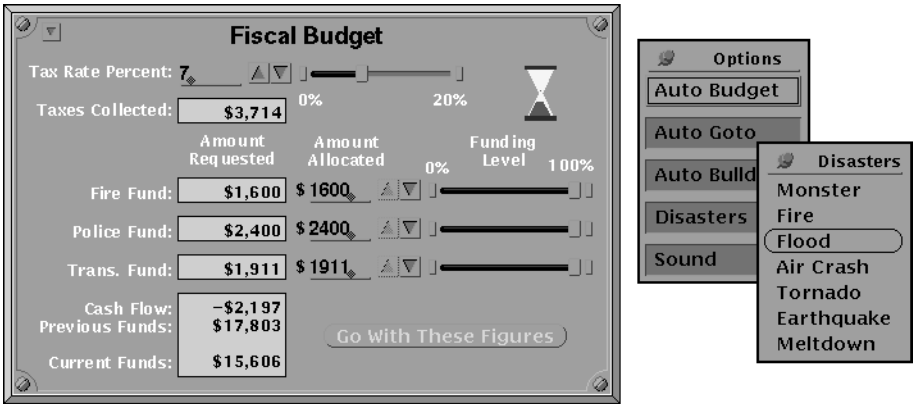


The SimCity scenario selector with help information being displayed. The magnifying glass contains an image of what the help message refers to.

*. SimCity is a registered trademarh of Maxis Software.
†. OPEN LOOK is a registered trademark of UNIX System Laboratories Inc.

OPEN LOOK objects

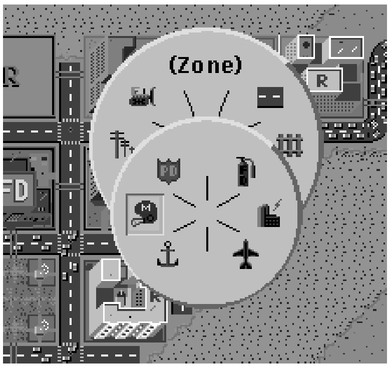
An example of OPEN LOOK objects, including pinned menus.



A full set of OPEN LOOK objects is available with HyperLook. The OPEN LOOK objects are as easy to work with as the other HyperLook objects.

Pie Menus

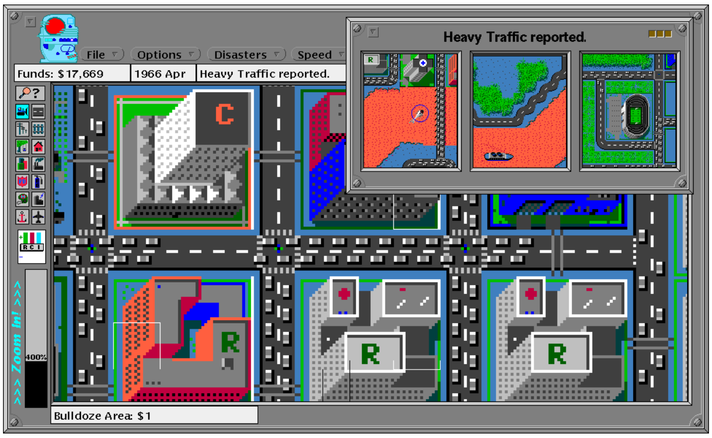
Nested Pie menus let you select options with rapid and natural gestures.



The picture above shows pie menus running under HyperLook. Pie menus let you change editing tools with quick gestures and a minimum of mouse movement. Only HyperLook provides this kind of power and ease of use.

Power and speed

Multiple panning and zooming views of a city scene.



The above picture shows panning and zooming views of an animated city scene. HyperLook provides a stimulating combination of high speed raster graphics animation and PostScript imaging.

Sound

HyperLook has an audiotool application which is used by SimCity. The HyperLook Audiotool lets you play multiple sounds at the same time. It lets you define channel, volume, pitch and synchronisation. The tool can handle multiple (virtual) channels. Each application can allocate its own channel over which to play sounds. Channels can be switched on or off by the user. This means that you can switch off SimCity sounds without switching off your mail alarm bell!

Contact Information

HyperLook provides more functionality, with greater ease of use than any other OpenWindows[*] tool. Earlier versions of the program have been in use at more than 300 sites worldwide. Its value is proven.

The examples show that HyperLook is useful to a wide range of people: from non-programmers to experts, all of whom want an environment which allows them to experiment with and develop useful tools and applications.

For more information about HyperLook contact:

The Turing Institute Limited
36 North Hanover Street
Glasgow G1 2AD
Scotland
Tel: +44 41 552 6400
Fax: +44 41 552 2985
Email: hyperlook@turing.com

*. OpenWindows is a trademark of Sun Microsystems Inc.