



# How to Use the Limo Example

GBB Version 4.1

---

## Overview

### What this document is about

This tutorial describes how to use the limo-scheduling demonstration example provided with GBB. Topics are as follows:

- Loading and running the limo application
- What the graphics windows show you
- Entering limo orders and exploring unit instances
- Using the Control Shell Window
- Quitting the application

### The limo example

The limo application schedules limousine service for customers traveling between various locations. The application acts as an intelligent dispatcher for a limousine service called the Lazy Limo Company. The basic idea of a limousine service is to accommodate people who typically make reservations in advance for a ride from a designated pickup place, at a designated time, to a designated drop-off place, at an estimated time. A limousine service typically uses limousines or vans that can transport several passengers at one time.

### Developed using GBB and ChalkBox

The limo application was developed using GBB, and it uses the GBB Graphics System to display graphical information. The GBB Graphics System is a graphical user interface developed using the ChalkBox™ graphics toolkit.

### For more information

- **About the limo application**—For a general description of the limo example, see the document titled *The Limo Example*.
- **About the GBB Graphics System**—To learn how to create graphical blackboard windows in a GBB application and customize them to display the information you need, see the tutorial titled

*How to Use the GBB Graphics System with the Limo Example.* For information on how to use your mouse with the GBB Graphics System and how to use the pop-up menus and dialogs, see the *Interacting with ChalkBox* manual.

- **About GBB**—For more information about the GBB concepts discussed in this tutorial, see the GBB documentation set.

---

## Loading and running the limo application

### Loading start-up information

In the Lisp listener window, make the directory containing GBB current and load the `startup.lisp` file into Lisp by entering the following form:

```
(load "startup.lisp")
```

Lisp loads the `startup.lisp` file and several other files.

**Tip:** You must specify a complete path name if the `startup.lisp` file is stored in a directory other than your default directory. For example, the form might be:  

```
(load "/local/gbb/v-410/startup.lisp").
```

### Switching from another GBB application

If you are switching to the limo example from any other GBB application (for example, the ecosystem example or elevator example), simply reset GBB by calling the **reset-gbb** generic function, instead of loading the `startup.lisp` file.

### Loading and running the limo example

To load the GBB `limo-example` module and run the application, enter forms in the Lisp listener window as follows:

- 1 Load the `limo-example` module:

```
(load-kti-module :limo-example)
```

- 2 Change to the `limo-example` package:

```
(in-package :limo-example)
```

- 3 Activate the GBB Graphics System and run the limo example:

```
(limo-example t)
```

**Note:** If your application window is not sufficiently large or your monitor does not have sufficient resolution, a Chain Manager window will be displayed, to allow you to iteratively select among individual windows. For information about using the Chain Manager window, see "Small screen displays."

## Building a suspended Lisp image

Lisp loads *all* of the required GBB files each time you load the `:limo-example` module. Therefore, to avoid reloading all those files each time you want to run the limo example (or another GBB example), you can build a suspended Lisp image containing the commonly used GBB modules. Then, when you load the `:limo-example` module to run the limo example, only the files specific to the limo example are loaded.

For example, the following forms build an image containing the `gbb`, `gbb-graphics`, and `agenda-shell` modules:

```
(load-kti-module '(:gbb :agenda-shell :gbb-graphics))

(setq excl::*read-init-files* nil)

(kti-tools:save-image "my-gbb")
```

## Recovering from errors

If you cause an error while performing the activities in this tutorial, see "Error recovery" at the end of this document.

---

## What the graphics windows show you

### Predefined configuration

The limo application displays the following windows in a predefined configuration on your screen:

- **Five blackboard windows**, which are windows that display information about unit instances on the blackboard.
- **The Control Shell Window**, which enables you to monitor the operations of the control shell and interactively stop and restart it. The title of the window changes to reflect the KS that is currently being executed. (For example, when the Lisp listener window indicates that it is time to place an order, the window will be titled `Get-Order-Interactive-KS`, since the `get-order-interactive-ks` KS is the KS that brings up the limo order dialog.)
- **The Lisp listener window**, which displays a trace of all control-shell activities (consisting of descriptive lines for each event that is signalled).
- **The GBB logo window**, which is an important mouse location, because the main ChalkBox menu and GBB Graphics System menu are available when the mouse is located over the logo window.
- **The Mouse Documentation window**, which provides a handy source of information on the functions of mouse buttons and keystroke/mouse-button combinations.

Window arrangement

If your monitor has sufficient resolution and a sufficiently large application window, all the windows described above will appear directly on your monitor. On a low-resolution monitor or a monitor on which the application window size is insufficient, a special Chain Manager window allows you to iteratively select among individual windows. For information about using the Chain Manager window, see "Small screen displays."

**What the blackboard windows show**

Blackboard windows are a special type of window provided by the GBB Graphics System to display a one- or two-dimensional view of a selected area of the blackboard database. The five blackboard windows displayed by the limo application have been predefined to show the unit instances stored on particular space dimensions in the blackboard database.

The windows show the following information:

- **Route Map**—The blackboard window titled Route Map shows the places serviced by the limos and the roads used to service the places. (The places at which a sign is displayed are designated as waiting places for the limos.)
- **Orders Received**—The blackboard window titled Orders Received shows orders received. It displays a line indicating the most direct route, as the crow flies, between the origin and destination places. Notice that, since you haven't submitted an order yet, no lines are drawn.
- **Planned Tasks**—The blackboard window titled Planned Tasks shows tasks planned for the limos. It displays lines representing the routes between the origin and destination places. Notice that initially the only planned task is the waiting task. The dot representing the Lazy Limo Co. is green (representing the waiting task), because all the limos are waiting there.
- **Time/Place View**—The blackboard window titled Time/Place View contains a graph that shows the places that will be visited by the limos (or at which they will be waiting) against the period of time during which the places will be visited. Notice that initially all limos are waiting at Lazy Limo Co. (The green line represents the waiting task of the limos.)
- **Task Status**—The blackboard window titled Task Status contains a graph that shows the status (*to-pickup-fare*, *to-waiting-place*, *waiting*, and *carrying-fare*) of the waiting and to-pickup-fare tasks against the time interval of the statuses. The graph shows that initially all limos are waiting at Lazy Limo Co.

**Note:** The type of mouse click you use to select an operation from a pop-up menu depends on your window toolkit.

---

## Entering a limo order

Next, enter a limo order by following the instructions below. Watch the order as it is displayed in the Orders Received window (as the blue line between J-

Mart and Golden Age Village). Watch the planned route of the assigned limo as it is displayed in the Planned Tasks window. To understand why the application chose the planned route, check the possible routes in the Route Map window.

### Using the limo order dialog

You enter an order for a limo by using the limo order dialog. To bring up the limo order dialog, click left on the button labeled “Place an order.” In the dialog, specify the origin and destination places, the pickup time, and the number of passengers to be picked up, as follows:

- 1 Click left on **J-Mart** in the list of origin place names, to indicate the place at which the customer wants to be picked up.
- 2 Click left on **Golden Age Village** in the list of destination place names, to indicate the place to which the customer wants to be delivered.
- 3 Enter **10:00** (indicating 10 A.M.) for the pickup time.
- 4 Click left on the box labeled **2** for the number of passengers.
- 5 Click left on the **Place Order** button, or simply press Return.

### KS activity

Based on the limo order you entered, GBB runs KSs that determine:

- Which limo is available to pick up two passengers at J-Mart at 10 A.M. and deliver them to Golden Age Village
- The fastest and most efficient route for servicing these customers, given the possible routes (as defined to the application)

As GBB runs the KSs, it displays descriptive lines in the Lisp listener window about the events signalled by the KSs. Note the message displayed in the Lisp listener window by the assign-limo-ks KS:

```
Assigning limo n to pickup 2 passengers at 10:00 from
J-Mart. Drop off is at 11:24 at Golden Age Village.
```

### Planned tasks that resulted

In the Planned Tasks window, GBB now displays:

- Several **empty** tasks (in green), as follows:
  - **The dot at Lazy Limo Co.**—Indicates several empty tasks with the `waiting` status. The empty tasks represent the waiting activity of the limos that have not left Lazy Limo Co. yet.
  - **The line segment between Lazy Limo Co. and Green Acres**—Indicates an empty task with the `to-pickup-fare` status. This empty task represents the route of the limo assigned to carry the fare from Lazy Limo Co. (the limo’s current location) to J-Mart (the origin).
  - **The line segment between Green Acres and J-Mart**—Indicates the same empty task as the line segment between Lazy Limo Co. and Green Acres.

**Note:** Moving in the opposite direction, this line segment also represents the `carrying-fare` task. However, until the window is refreshed during a subsequent step

(or if you resize the window), the line segment is green, since the empty task is drawn after the carrying-fare task and therefore temporarily overwrites it.

- **The dot at Golden Age Village**—Indicates an empty task with the waiting status. This empty task represents the waiting activity of the limo that completed the carrying-fare task (from J-Mart to Golden Age Village) and is now waiting for its next fare.
- **The carrying-fare task** (in red) showing the application's planned route for servicing the customers. The planned route is the most direct route, by road, from J-Mart (the origin) to Golden Age Village (the destination). The route is from J-Mart to Green Acres, to City Hall, to Monroe Square Garden, to Dave's Deli, to Larry's Laundry, and, finally, to Golden Age Village. Notice that other possible routes exist (see the Route Map window), but the selected route is the shortest.

### Information displayed on a time line

The graph in the Time/Place View window shows the places that were visited during each task against the period of time during which the places were visited, as follows:

- **The carrying-fare task** (in red), showing the route of the limo from J-Mart to Golden Age Village on a time line. These line segments represent the activity of the limo transporting the passengers.
- **Several empty tasks** (in green), as follows:
  - **A solid line at Lazy Limo Co.**—Indicates several empty tasks with the waiting status. This line represents the waiting activity of the limos that haven't left Lazy Limo Co. yet.
  - **Line segments from Lazy Limo Co. to Green Acres and Green Acres to J-Mart**—Indicate an empty task with the to-pickup-fare status. This line represents the en-route activity of the limo assigned to carry the fare from J-Mart to Golden Age Village.
  - **A line at Golden Age Village**—Indicates the empty task with the waiting status. This line represents the waiting activity of the limo that carried the fare from J-Mart to Golden Age Village *after* it dropped off the passengers.

## Exploring unit instances

Now that GBB has processed a limo order, you can explore the unit instances on the tasks and map space instances that were used by the KSs to arrive at the planned route. These unit instances are shown in the Planned Tasks window.

### Find Units Near Point operation

Find unit instances by performing the following steps:

- 1 **Find unit instances that are near the Lazy Limo Co. place instance** by clicking left on the dot at Lazy Limo Co. in the Planned Tasks window.

GBB displays the Units menu, which lists the unit instances located near the point on which you clicked. In this case, the menu lists six `empty` instances and the `Lazy Limo Co` instance. Five of the `empty` instances have a `waiting` status, representing limos that are waiting at the Lazy Limo Co., and the other `empty` instance has a `to-pickup-fare` status, representing a limo that is on its way from the Lazy Limo Co. to pick up a fare.

- 2 Inspect the `empty 7` unit instance** by clicking left on it and then on the Inspect button.

GBB displays an inspector window, which shows the name of the unit instance and the value of each slot and dimension of the unit instance. (On some window toolkits, the window toolkit's native inspector window is used; on others, ChalkBox's inspector window is used.) The `path` slot indicates the assigned route of the limo from Lazy Limo Co. to J-Mart. To exit the Units menu, click left on the Exit button.

- 3 Find unit instances near J-Mart** by clicking left on the dot at J-Mart in the Planned Tasks window. GBB displays the Units menu, which shows the `empty 7` instance, as well as the `j-mart` instance and `carrying-fare 1` instance.

- 4 Inspect the `carrying-fare 1` instance** by clicking left on it and then on the Inspect button. Inspect the `path` slot, in order to see all the places that make up the planned route for the limo. Then, to quit the inspector window, click left on the Exit button (or use the technique appropriate to your window toolkit's inspector window). Quit the inspector window and then the Units menu.

By inspecting various unit instances in the Planned Tasks window, you can see that planned tasks for limos that are empty (with the `to-pickup-fare`, `to-waiting-place`, or `waiting` status) are shown in green and planned tasks for limos that are carrying passengers (with the `carrying-fare` status) are shown in red.

### Inspecting more unit instances

In the Task Status window, which shows instances stored on the `time-interval` and `task-status` dimensions of the `tasks` space instance, inspect some unit instances as follows:

- 1** Click left on the solid line displayed for the `waiting` task status, close to the right side of the line. GBB displays the Units menu, showing the unit instances near the point on which you clicked.
- 2** Click left on any one of the `empty` instances with the time interval `(9:00 5:00)` and then click left on the Inspect button. GBB displays the inspector window. Notice that the instance (like the other instances with time interval `(9:00 5:00)`) represents waiting activity at the Lazy Limo Co. Then, quit the inspector window.
- 3** Select the `empty` instance with the time interval `(11:24 5:00)`, and repeat the inspection process. Notice that this instance represents waiting activity at Golden Age Village. Quit the inspector window and then the Units menu.

---

## Entering a second limo order

### Using the limo order dialog

Bring up the limo order dialog by clicking on the “Place an order” button. Then perform the following steps to submit a request for a limo:

- 1 Click left on **9/16 Convenience Store** in the list of origin place names, to indicate the place at which the customer wants to be picked up.
- 2 Click left on **Acme Manufacturing** in the list of destination place names, to indicate the place to which the customer wants to be delivered.
- 3 Enter **10:30** for the pickup time.
- 4 Click left on the box labeled **1** for the number of passengers.
- 5 Click left on the **Place Order** button, or press Return.

### KS activity

GBB runs the KSs that determine:

- Which limo is available to pick up one passenger at 9/16 Convenience Store, at 10:30 A.M., and deliver her to Acme Manufacturing
- The fastest and most efficient route for servicing this customer, given the possible routes (as defined to the application)

Then, GBB displays a message in the Lisp listener window indicating which limo will pick up the passenger, the time of pickup, the origin place, the drop-off time, and the destination place. Notice that the new order is shown in the Orders Received window and the newly planned tasks are shown in the Planned Tasks window.

## Zooming in and out

### A more focused view

Next, try zooming in on, and out of, an area of the graph in the Time/Place View window.

To determine how to zoom in, position the mouse cursor anywhere on the graph in the Time/Place View window and then see the mouse documentation to determine which mouse click to use to perform the zoom operation. Zoom in on the instances located near J-Mart, Green Acres, City Hall, and Monroe Square Garden at approximately 9:40, in order to see more detail (such as the exact time) about the instances.

For example, if you’re running the limo example on Allegro Common Lisp for Windows, the procedure is as follows:

- 1 Position the mouse cursor at the top left corner of the rectangular area containing the instances you want to zoom in on.
- 2 Simultaneously press the Alt button and click the left mouse button.
- 3 Holding the mouse button down, drag the mouse to the lower right corner of the rectangular area.
- 4 Release the button.

Then, you can zoom out again. See the mouse documentation for instructions. GBB redisplay the graph as it was before you zoomed in.

### Zoom to fit

Try using the Zoom To Fit operation, which you initiate from the Graph Operations Menu. Zoom To Fit finds the bounds of the data that is to be displayed or is already being displayed and displays or redisplay the data with its bounds. For example, if you've already zoomed in on part of a dimension (by using Zoom), Zoom To Fit finds the whole piece of that dimension, based on its bounds, and redisplay that whole piece.

---

## Using the Control Shell Window

### What the window shows

The Control Shell Window gives you easy access to information about control-shell operations, and it provides buttons that allow you to invoke actions such as turning on stepping, suspending trace printing in the Lisp listener window, and even stopping the currently executing control-shell scheduling loop. The Control Shell Window continuously indicates the current status of control shell operations (triggering KSs, executing preconditions, executing KSAs, quiescence).

The Control Shell Window also provides graphs that show historical information about the current scheduling loop and several previous loops, for the following items:

- The number of events processed
- The number of KSs triggered
- The number of KSs activated
- The number of KSAs pending

The historical information is plotted as minimum and maximum numbers, reading right to left on a time scale. As the oldest maximum drops of the graph, the newer maximum is the largest number shown.

### Watching the indicators change

To observe changing information in the Control Shell Window, follow the steps below:

1 Enter the following limo order:

- The origin place is **Piggly Wiggly**.
- The destination place is **Mile Long Mall**.
- The pickup time is **10:40**.
- The number of passengers is **1**.

Watch the control-shell phase indicator in the Control Shell window. Notice that it moves quickly among the phases of the control shell's scheduling loop, thereby always showing the current status of control-

shell operations. Also notice that the cycle number is updated each time a new cycle of the scheduling loop begins.

Notice the historical information about events, triggered KSs, activated KSs, and pending KSAs displayed in the Control Shell Window. GBB updates these graphs during each cycle of the control-shell scheduling loop, to provide information about control-shell activities over time.

- 2 Enter another limo order, as follows: specify **Seventh National Bank** for the origin, **Board of Trade** for the destination, **10:40** for the pickup time, and **2** for the number of passengers. Again, watch the information provided in the Control Shell window.
- 3 Turn on control-shell stepping for event processing, as follows:
  - a Click left on the Step Options (|>>) button. GBB displays a menu of control-shell stepping options.
  - b Click left on **Process event** and OK.
- 4 Again, enter a limo order: specify **Mile Long Mall** for the origin, **Board of Trade** for the destination, **2:00** for the pickup time, and **1** for the number of passengers.

The control shell pauses before processing the next event that is signalled, and a dialog titled Control Shell Stepper is displayed (positioned where your mouse cursor is). To continue stepping through control-shell operations, click left on the Continue button.

Then, turn off control-shell stepping by clicking left on the Quit button in the Control Shell Stepper dialog.

---

## Quitting the application

To quit the limo application:

- 1 Bring up the GBB Graphics System menu by clicking left on the GBB logo window in the upper right corner of your screen.
- 2 Click on **Exit Limo Example**.

GBB exits the control shell and deletes the blackboard windows, the Control Shell Window, and the window containing the “Place an order” button.

---

## Other considerations

The sections below explain the following conditions:

- Graphics windows that are overlaid
- Error recovery in Allegro Common Lisp

## Small screen displays

If you run the limo example on a screen with resolution of 800 x 600 pixels or less or if the size of your application window is insufficient to display all the graphics windows, the configuration of the windows that are displayed is different than on a larger screen.

### Chain Manager window

On a small display, the GBB graphics windows remain overlaid, and you must use the Chain Manager window to access the underlying windows. (The Lisp listener window is always displayed on your screen, however.) The Chain Manager window contains several buttons, as described in the table that follows:

Click left on this button	To perform this action
Previous	Bring to the front the preceding graphics window. (If the Order Received window is displayed, the previous window is the Route Map window; if the Route Map window is displayed, the previous window is the Control Shell Window, and so on.)
Next	Bring to the front the immediately following graphics window. (If the Control Shell Window is displayed, the next window is the Route Map window, and so on.)
Select	Bring up the Chained Windows menu and select the one that is to be brought to the front.
Remove	Bring up the Chained Windows menu and select the window or windows you want to remove from the chain. Removed windows are displayed on the screen along with the chained window that is at the front of the chain.
Add	Bring up the Chained Candidates menu and select the window or windows you previously removed from the chain and want to add back into the chain.

## Error recovery

### When using Allegro Common Lisp

If you're running the limo application on Allegro Common Lisp and either of the following conditions apply, the problem described below can occur:

- You're running Lisp on UNIX and not using the emacs interface
- You're running Lisp on Windows and not using either the emacs interface or Allegro CL's IDE

When an error is signalled, the error process and the normal Lisp listener process are both trying to read from the same stream. (More specifically, the

debugger and the limo example are trying to read input at the same time, and they succeed in reading only alternate lines.)

To exit from an error, enter `:pop` twice in the Lisp listener window to ensure you're really out of the error.

---

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Knowledge Technologies International, Inc.

**U.S. Government Restricted Rights Legend:** Use, duplication, and disclosure by the Government are subject to restrictions of Restricted Rights for Commercial Software developed at private expense as specified in DOD FAR 52.227-7013 (c)(1)(ii).

Copyright © 2000 by Knowledge Technologies International, Inc. All rights reserved.

GBB, ChalkBox, KTI Tools, and the KTI logo are trademarks of Knowledge Technologies International, Inc. Other brand or product names are trademarks or registered trademarks of their respective holders.

May 2000

Knowledge Technologies International, Inc.  
401 Main Street  
Amherst, MA 01002  
Phone: (800) 577-8990, (413) 256-8990  
Fax: (413) 256-3179  
E-mail: [gbb-info@KTIworld.com](mailto:gbb-info@KTIworld.com)  
WWW: [www.KTIworld.com](http://www.KTIworld.com)



---

KNOWLEDGE  
TECHNOLOGIES  
INTERNATIONAL

---

[www.KTIworld.com](http://www.KTIworld.com)