



The Limo Example

GBB Version 4.1

Overview

About the application

The limo example is a very basic blackboard application that was developed using GBB™. The application, which schedules limousine service for customers traveling between various locations, is a simple one you can use to become familiar with GBB and its graphics capabilities.

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.

Background information

Developed by GBB students

The limo example was written collectively by a group of students who were learning to use GBB during a three-day laboratory session in a GBB training course. The students chose limo scheduling as their application's purpose because it presented an interesting planning and scheduling problem, and it required only intuitive expertise (that is, it required no significant acquisition of knowledge on their part). The students had constraints, such as limited time in which to consider the problem and varied levels of Common Lisp expertise.

The students were very successful in using GBB to develop the limo application. During their three days of hands-on training in using GBB, they were able to design, develop, and deliver the application described in this document.

Limited scope

The purpose of the limo application is to act as an intelligent dispatcher for a limousine service called the Lazy Limo Company.

To simplify the application design and development process, the students limited the scope of their domain. They defined a limited number of pickup and drop-off locations, called **places**. They also limited the **roads** that connect the pickup and drop-off places and thereby form a route among the various places. (As a result, a limo cannot move directly between every place; in some cases, it must travel through other places to get to the desired destination.) For each road, the students defined the normal amount of time a limo requires for travel.

The goal

The students' goal in designing the limo application was that it would schedule limos according to the following criteria:

- **Use of resources**—Maximize the use of resources (for example, require ride-sharing when necessary).
- **Route determination**—Determine the shortest route between two places when assigning a route to a limo (where a route can pass through one or more places and is made up of roads). If a limo is delayed en route because of such incidents as an accident or construction roadblock, the application must find an alternate route for the limo.
- **Distribution of limos**—Keep the limos distributed throughout the entire service area.

All of these goals were not achieved in this implementation, however.

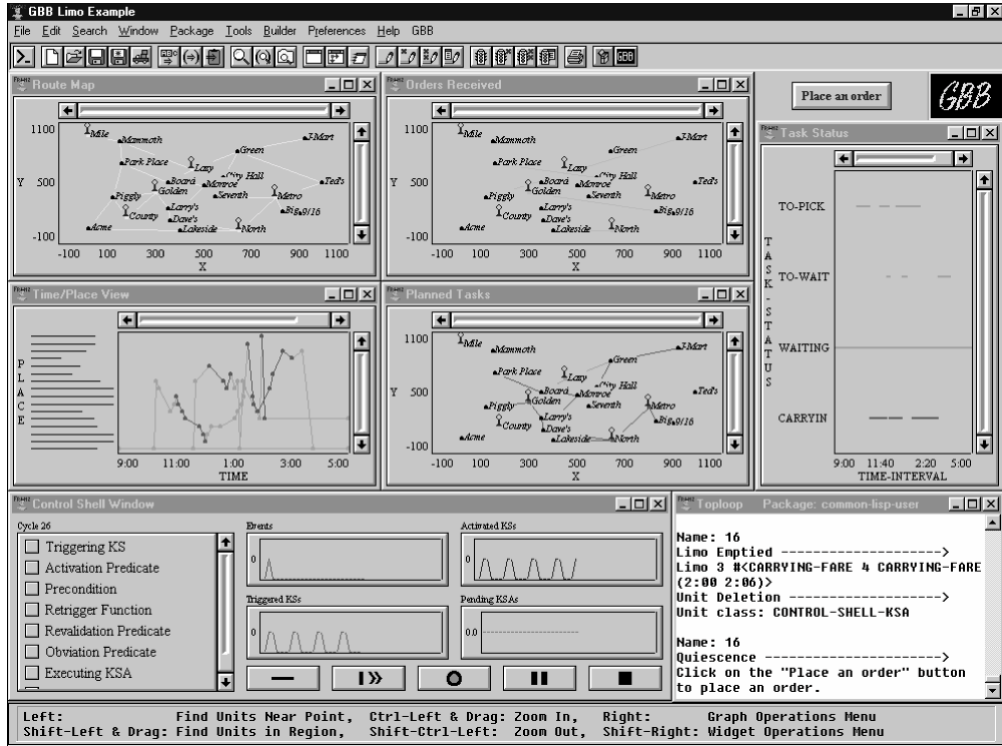
Sample session

Screen illustration

The screen on the next page illustrates a portion of the limo application session. During the illustrated portion of the session, the Lazy Limo Co. serviced four orders, as indicated by the four line segments in Blackboard Window 3.

Blackboard Window 2 shows the routes of the various limos that serviced the four orders. (The routes include the roads travelled to the pickup places and then to the drop-off places.) Blackboard Window 4 indicates the time intervals during which all the Lazy Limo Co. limos had various statuses (en route to a pickup place, en route to a waiting place, waiting at a waiting place, and carrying a fare).

The Control Shell Window (labeled `GET-ORDER-KS` at the time at which the screen was captured) indicates that the control shell is currently executing an activation of the `get-order-ks` knowledge source (KS). It also provides in graph form historical information about events, triggered KSs, activated KSs, and pending KS activations (KSAs). GBB updates these graphs during each cycle of the control-shell scheduling loop, to provide information about control-shell activities over time.



Application data

Places serviced by limos

The following table lists the places that are serviced by the Lazy Limo Co. and indicates the roads that connect certain places and the normal travel time for each road.

Place	Connecting places	Normal travel time (in minutes)
9/16 Convenience Store	Big View Apartments	4
Acme Manufacturing	Lakeside Park	25
	Piggly Wiggly	16
Big View Apartments	9/16 Convenience Store	4
	Metro Airport	8
Board of Trade	Lazy Limo Co	5
	Monroe Square Garden	6
	Park Place	30
City Hall	Green Acres	20
	Seventh National Bank	9

Place	Connecting places	Normal travel time (in minutes)
County Hospital*	Golden Age Village	8
	Piggly Wiggly	4
Dave's Deli	Larry's Laundry and Video	8
	North Station	6
Golden Age Village*	County Hospital	8
	Larry's Laundry and Video	4
	Piggly Wiggly	9
Green Acres	City Hall	20
	J-Mart	20
	Lazy Limo Co.	24
	Ted's Wharf	31
J-Mart	Green Acres	20
Lakeside Park	Acme Manufacturing	25
	North Station	6
Larry's Laundry and Video	Golden Age Village	4
	Dave's Deli	8
Lazy Limo Co.*	Board of Trade	5
	Green Acres	24
	Mammoth Multicinema	30
Mammoth Multicinema	Lazy Limo Co.	30
	Mile Long Mall	6
	Park Place	28
Metro Airport*	Big View Apartments	8
	North Station	31
	Seventh National Bank	12
	Ted's Wharf	18
Mile Long Mall*	Mammoth Multicinema	6
Monroe Square Garden	Board of Trade	6
	City Hall	8
	Dave's Deli	24
North Station*	Dave's Deli	6
	Lakeside Park	6
	Metro Airport	31
Park Place	Board of Trade	30
	Mammoth Multicinema	28
	Piggly Wiggly	29

Place	Connecting places	Normal travel time (in minutes)
Piggly Wiggly	Acme Manufacturing	16
	County Hospital	4
	Golden Age Village	9
	Park Place	29
Seventh National Bank	City Hall	9
	Metro Airport	12
Ted's Wharf	Green Acres	31
	Metro Airport	18

* Designated as a waiting place for the limos

The structure of the application

Unit classes

Some of the unit classes used in the application are listed below:

- **Place**—Represents places that are serviced by the Lazy Limo Co.
- **Limo**—Represents the limos owned by the Lazy Limo Co.
- **Order**—Represents customer reservations, called **orders**.
- **Road**—Represents the roads that connect the places.
- **Empty**—Represents the tasks being performed by a limo when it is not carrying a passenger.
- **Carrying-fare**—Represents the fastest possible route for servicing a customer order.
- **Task**—An abstract unit class (which is not intended to be instantiated) that represents the tasks being performed by a limo during a given time interval. The **empty** and **carrying-fare** classes inherit from this class.

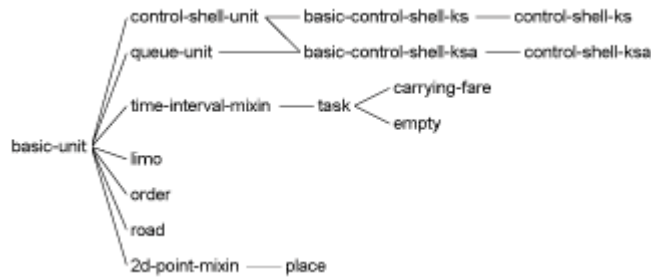
The following table provides more information about these unit classes.

Unit class	Description
Place	<p>Represents the places. Slots are as follows:</p> <ul style="list-style-type: none"> • <code>Location</code>—Nonlink slot (inherited from the <code>2d-point-mixin</code> unit class) containing the coordinate point that indicates the location of the place on the map. • <code>Waiting-place</code>—Nonlink slot containing a logical value that indicates that the place stored in the <code>location</code> slot is a waiting place for limos. • <code>Roads</code>—Link slot for which instances are linked to the <code>connects</code> slot of the <code>road</code> class. • <code>Waiting-tasks</code>—Link slot for which instances are linked to the <code>waiting-at</code> slot of the <code>empty</code> class. • <code>Orders</code>—Link slot for which instances are linked to the <code>route</code> slot of the <code>order</code> class.
Limo	<p>Represents the limos. Slots are as follows:</p> <ul style="list-style-type: none"> • <code>Capacity</code>—Nonlink slot containing the number of passengers that the limo can hold. • <code>Tasks</code>—Link slot for which instances are linked to the <code>limo</code> slot of the <code>empty</code> class.
Order	<p>Represents the customer reservations, which indicate the origin and destination places, number of passengers, and pickup time. Slots are as follows:</p> <ul style="list-style-type: none"> • <code>Route</code>—Link slot for which instances are linked to the <code>orders</code> slot of the <code>place</code> class. • <code>N-passengers</code>—Nonlink slot containing the number of passengers to be serviced according to the order. • <code>Pickup-time</code>—Nonlink slot containing the time (as an integer indicating the number of minutes past 9 A.M.) at which the passengers are to be picked up at the place of origin. • <code>Arrival-time</code>—Nonlink slot containing the time (as an integer indicating the number of minutes past 9 A.M.) at which the passengers are to be dropped off at the destination place.

Unit class	Description
Road	<p>Represents standard routes the limos will take between two particular places. Slots are as follows:</p> <ul style="list-style-type: none"> • <code>Connects</code>—Link slot for which instances are linked to the <code>roads</code> slot of the <code>place</code> class. • <code>Normal-time</code>—Nonlink slot containing the amount of time the limo normally needs to travel the road. • <code>Tasks</code>—Link slot for which instances are linked to the <code>roads</code> slot of the <code>task</code> class.
Carrying-fare	<p>Represents the fastest possible route from the place of origin to the destination place. Inherits slots from the <code>task</code> unit class.</p>
Empty	<p>Represents the tasks being performed by a limo when it is not carrying a passenger. Inherits slots from the <code>task</code> unit class.</p>
Task	<p>An abstract unit class that represents the tasks being performed by a limo during a given time interval. Slots are as follows:</p> <ul style="list-style-type: none"> • <code>Time-interval</code>—Nonlink slot (inherited from the <code>time-interval-mixin</code> unit class) containing the time interval during which the limo has the status in the <code>status</code> slot. • <code>Status</code>—Nonlink slot containing the status of the limo during the time interval stored in the <code>interval</code> slot. • <code>Path</code>—Nonlink slot containing the time/place list that indicates the route of the limo during the time interval stored in the <code>interval</code> slot. • <code>Waiting-at</code>—Link slot for which instances are linked to the <code>waiting-tasks</code> slot of the <code>place</code> class. • <code>Limo</code>—Link slot for which instances are linked to the <code>tasks</code> slot of the <code>limo</code> class. • <code>N-passengers</code>—Nonlink slot containing the number of passengers riding in the limo during the time interval stored in the <code>interval</code> slot. • <code>Roads</code>—Link slot for which instances are linked to the <code>tasks</code> slot of the <code>road</code> class.

Unit-class hierarchy

The hierarchy of unit classes in the limo application is shown below. Note that the `basic-unit` class is the highest-level class, since in GBB it is the default superclass unit class.



Spaces

To store the unit classes, the application uses the following spaces:

- **Map**—Represents the map of the service area and, therefore, stores instances of the `place` and `road` unit classes
- **Limos**—Stores instances of the `limo` unit class
- **Orders**—Stores instances of the `order` unit class
- **Tasks**—Stores instances of the `empty` and `carrying-fare` unit classes

The table on the next page provides more information about the spaces.

Space	Description
Map	<p>The space's dimensions represent the coordinate points of locations on the service-area map:</p> <ul style="list-style-type: none"> • <code>X</code>—The x-axis coordinate; an ordered dimension with bounds of 0 and 1000. • <code>Y</code>—The y-axis coordinate; an ordered dimension with bounds of 0 and 1000. <p>Stores instances of the <code>road</code> and <code>place</code> unit classes.</p>
Limos	<p>The space's single dimension, <code>capacity</code>, represents the number of people the limo can hold. The dimension is ordered and has bounds of 0 and 1000. Stores instances of the <code>limo</code> unit class.</p>

Orders

The space's dimensions represent the information that makes up an order:

- Time—Points in time; an ordered dimension with bounds of -1 and 480.
- X—The x-axis coordinate; an ordered dimension with bounds of 0 and 1000.
- Y—The y-axis coordinate; an ordered dimension with bounds of 0 and 1000.
- Pickup-time—The time at which the passengers

Dimension-value computations

For extracting dimension values

The limo application uses several dimension-value computations. Dimension-value computations specify how the application is to obtain dimension values from source data (such as a list, structure, array, CLOS object, and so on). The source data can be located in a slot value of a unit instance or computed using another dimension-value computation.

The limo application uses the following dimension-value computations:

- **Place-list-dvc**—Specifies how to obtain a list of places. The application uses this computation to obtain the values of the `x` and `y` dimensions for the `order` and `road` unit classes.
- **Time-place-list-dvc**—Specifies how to obtain a list of time and place pairs. The application uses this computation to obtain the value of the `x`, `y`, `time`, and `place` dimensions for the `task` unit class.

This dimension-value computation was defined in order to create a unit class consisting of a temporally connected sequence of `x`, `y` points, thereby enabling the application to display line segments on the `x` and `y` axes. Thus, the application can show the route of a limo during a particular time interval, rather than simply showing its location at a point in time, on the map.

- **Task-status-dvc**—Specifies how to obtain a status for limos (for example, `:to-pickup-fare`). The application uses this computation to obtain the value of the `task-status` dimension for the `task` unit class.
- **2d-point-mixin-dvc**—Specifies how to obtain a simple `x`, `y` location of a place. The application uses this computation to obtain the value of the `x` and `y` dimensions for the `2d-point-mixin` unit class. (This computation is used implicitly in the `place` unit class: the `place` unit class inherits from the `2d-point-mixin` unit class.)
- **Time-interval-mixin-dvc**—Specifies how to obtain a time interval. The application uses this computation in two ways: (1) to obtain the value of the `time-interval` dimension for the `time-interval-mixin` unit class and (2) in retrieval patterns in the `find-candidate-tasks` and `find-place-to-wait-ks` functions. (This computation is used implicitly in the `task` unit class: the `task` unit class inherits from the `time-interval-mixin` unit class.)

The knowledge sources

Simplistic use

In keeping with the simplistic design of the application, only five knowledge sources (KSs) are used, and they aren't highly knowledgeable. The KSs are as follows:

- **Initial-ks**—Initializes the system
- **Get-order-ks**—Obtains the next order

- **Plan-fare-ks**—Determines the fastest route for a limo, when an order is obtained
- **Assign-limo-ks**—Assigns a limo to the fastest possible route from the place of origin to the destination
- **Find-place-to-wait-ks**—Finds the most appropriate waiting place for a limo when it becomes empty

The control mechanism used by the application is GBB’s Agenda control shell, which is based on a precondition/action model. Only one of the KSs uses a precondition function. The purpose of a precondition function is to determine whether to activate the triggered KS and to return an execution rating that controls the execution order of the resulting KSAs. For each other KS, a constant rating is defined. Given a constant rating, the triggered KS will always be activated and the specified rating will be assigned to all KSAs.

The following table describes the KSs used by the application:

KS	Triggered by	Purpose
Initial-ks	System initialization	Reads map data and creates unit instances for places, roads between places, and limos; creates the initial waiting task for each limo unit class; and resets path-finding variables.
Get-order-ks	Queue quiescence	Obtains the next order from the keyboard, when queue quiescence occurs (that is, when no more executable KSAs remain in the queue of pending KSAs; creates an instance of the order unit class and sets the value of the route, pickup-time, and n-passengers slots.
Plan-fare-ks	Creation of an instance of the order unit class	Determines the fastest route from the origin to the destination, and creates an instance of the carrying-fare unit class and sets the value of the path, interval, and n-passengers slots.
Assign-limo-ks	Creation of an instance of the carrying-fare unit class	Determines whether a limo is available for the fare; if so, returns the list of tasks (which is limited to the waiting or to-waiting-place tasks) that the selected limo is doing during the interval of the proposed task, and assigns the limo to the fare by splicing the new task into the task list for that limo.

KS	Triggered by	Purpose
Find-place-to-wait-ks	Emptying of a limo	<p>Finds the most appropriate waiting place for the empty limo.</p> <p>For this KS, a precondition function called <code>end-place-is-waiting-place-p</code> determines whether the destination place at which the limo is currently located is a waiting place; if so, the precondition function need not schedule a KS to plan a route to a waiting place.</p>

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-400/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:

```
(initialize-chalkbox)
```

- 4 Run the `limo-example` module:

```
(limo-example)
```

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" in the document titled *How to Use the Limo Example*.

Pausing/resuming the application

You can cause the application to pause at any time by clicking on the pause/continue button in the Control Shell Window. (If the Chain Manager window is displayed, click on the Previous button to display the Control Shell Window.) The pause/continue button is the second button from the right-hand side of the window (with two vertical bars, suggestive of the pause control on an audio tape or CD player).

To continue from a pause, click again on the pause/continue button in the Control Shell Window.

Quitting the application

Perform the following steps to quit the 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.

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")
```

For more information

- **On using the limo example**—For information on using the limo application (for example, submitting limo requests and exploring the unit

instances that were used by the KSs to arrive at the proposed route), see the document titled *How to Use the Limo Example*.

- **On using the GBB Graphics System**—For information on using the GBB Graphics System with the limo application, see the document titled *How to Use the GBB Graphics System with the Limo Example*.
- **On GBB**—For a wide range of information about GBB, see the GBB document set.

Where to find the source code

The source code for the limo example is stored in the `limo-example.lisp` and `limo-example-graphics.lisp` files of the `examples` subdirectory.

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
WWW: www.KTIworld.com



KNOWLEDGE
TECHNOLOGIES
INTERNATIONAL

www.KTIworld.com