



The Elevator Example

GBB Version 4.1

Overview

About this document

This document describes the GBB demonstration application that acts as an intelligent dispatcher for a set of elevators in a building. Information about the application is in three parts:

- Data used by the application
- The structure of the application
- How to load and run the application

This document assumes you have already run the GBB limo example and are therefore familiar with how the GBB Graphics System works. If you are not already familiar with the GBB Graphics System, see the document titled *How to Use the GBB Graphics System with the Limo Example*

About the application

The elevator example is a basic blackboard application that performs the following activities:

- Plans the up and down movement of the elevators and the actions of the elevators (such as opening of doors)
- Simulates both the elevators and the behavior of the elevators

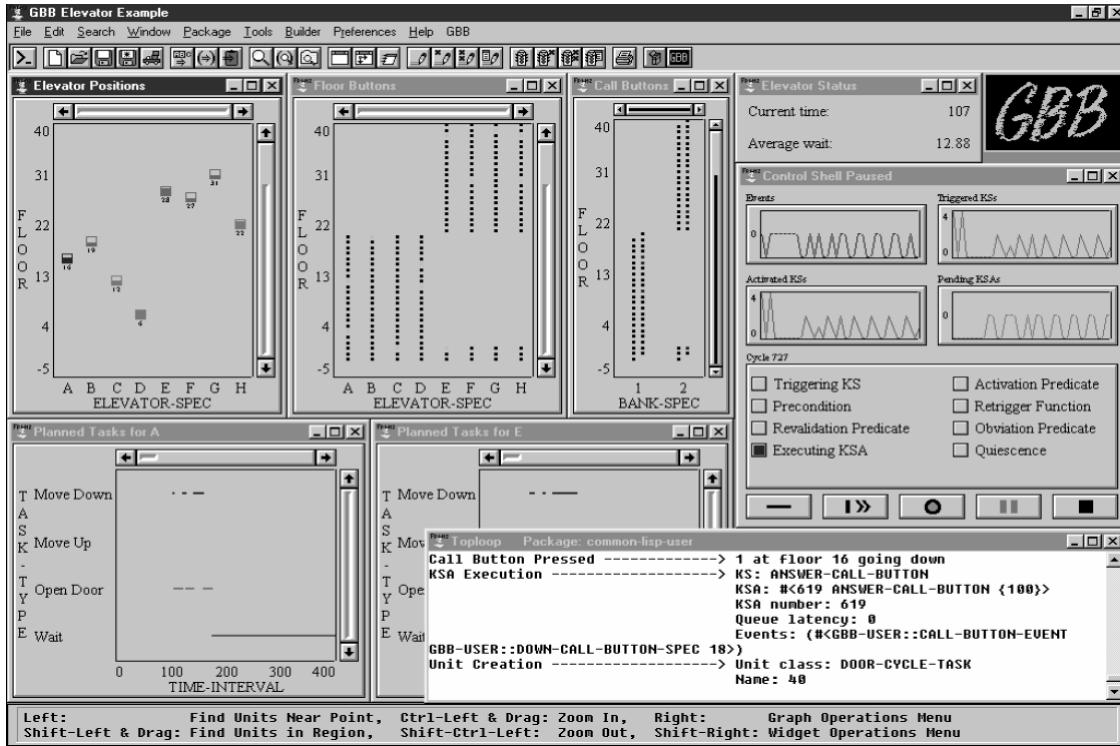
The application's planner component plans the tasks that control the elevators' movements on the basis of ongoing input from the application's simulator component.

The simulator generates elevator orders. An order contains the information needed by the simulator to represent a person approaching a bank of elevator cabs, pressing an up or down button, waiting for the elevator doors to open, entering the elevator cab, pressing one or more destination-floor buttons, and then exiting the elevator cab at the desired floor. An initial set of elevator orders is generated by the simulator component when the application starts

up. Successive sets of orders are generated by the simulator in a random manner.

Sample session

The screen below shows the elevator application's graphical interface:



Background information

Developed by GBB students

Like the limo-scheduling example, the elevator example was written collectively by students in a GBB training course. During three days of hands-on training in using GBB, the students were able to design, develop, and deliver the application described in this document.

Limited scope

The elevator application is limited in scope. For example, time is represented by an integer counter (the clock) that indicates the number of seconds that have elapsed since the start of the application. The clock is incremented only after all simulation and planning activities (determining what to do and creating a task to handle it) have been completed for the current time tick.

The application makes no attempt to introduce any time constraints on the planner. The planner always runs until it has planned every task it can for the current time tick. Also, the application makes no provisions for handling plan failures, such as elevator breakdowns.

Application data

Elevator cabs

The following table lists the static data about the elevator cabs, as found in the input file named `building-info.lisp`.

Elevator cab name	(x y) location
"A"	(20 40)
"B"	(40 40)
"C"	(60 40)
"D"	(80 40)
"E"	(20 60)
"F"	(40 60)
"G"	(60 60)
"H"	(80 60)

Note that this data isn't used by the application; it's simply used by the graphics system to give a bird's eye view of the elevator cabs in the building. (Looking down at the building, you'd see the row of elevator cabs A–D aligned with the row of elevator cabs E–H, with an aisle between the rows.)

Elevator banks

The following table lists the static data about the banks of elevators. This data is found in the input file named `elevator-info.lisp`.

Name	Elevator cab	Floors
"Bank 1"	("A" "B" "C" "D")	((-2 20))
"Bank 2"	("E" "F" "G" "H")	((-2 0) (21 40))

Elevator orders

The table below lists the elevator orders in the `elevator-orders.lisp` file, which is read by the application when the control shell starts up. This data represents the initial set of elevator orders generated by the simulator component. (Successive sets of orders are generated in a random manner.)

Time	Bank	Call-button floor	Up/down	Destination floor	Weight	Departure floor	Entry time	Exit time
2	"Bank 1"	0	:up	(57)	325	7	10	6
2	"Bank 1"	8	:down	(0)	180	0	14	6
2	"Bank 2"	0	:up	(22)	345	22	14	6
17	"Bank 2"	0	:up	(38)	345	38	14	6
20	"Bank 2"	28	:down	(0)	345	0	14	6
25	"Bank 1"	11	:up	(20)	345	0	14	6
100	"Bank 2"	22	:down	(0)	165	0	10	6

The structure of the application

Separate blackboards

The application uses three blackboards, as follows:

- **Building-info blackboard**—Contains information about the building and the elevators (the number of elevators, how they are grouped into banks, which floors each elevator bank serves, and so on). This information is used by the `planner` and `simulator` blackboards.
- **Planner blackboard**—Contains information used by the application's planner, which controls the elevators' actions.
- **Simulator blackboard**—Contains information used by the application's simulator, which provides input to the planner and simulates the execution of tasks that are planned by the planner.

Each blackboard is described in more detail below. Tables at the back of the document provide more specific information about (1) the spaces comprising each blackboard and (2) the unit classes of the instances stored on the space instances.

The building-info blackboard

Space hierarchy

The building-info blackboard represents the space-definition hierarchy rooted at the building-info space. The inheriting spaces are as follows:

- `Elevator-specs`—Stores instances of the `elevator-spec` unit class
- `Bank-specs`—Stores instances of the `bank-spec` unit class
- `Call-button-specs`—Stores instances of the `call-button-spec`, `up-call-button-spec`, and `down-call-button-spec` unit classes

- `Floor-button-specs`—Stores instances of the `floor-button-spec` unit class

Unit classes

The `building-info` blackboard stores instances of the following unit classes:

- `Bank-spec`—Represents static information about each elevator bank
- `Elevator-spec`—Represents static information about each elevator
- `Floor-button-spec`—Represents dynamic information about each floor button inside an elevator
- `Call-button-spec`—Represents dynamic information about each elevator call button located outside each bank of elevators
- `Up-call-button-spec`—Used only to print a message when the up call-button is pressed
- `Down-call-button-spec`—Used only to print a message when the down call-button is pressed

The planner blackboard

Space hierarchy

The planner blackboard represents the space-definition hierarchy rooted at the planner space. There is only one inheriting space, the `tasks` space.

Unit classes

The planner blackboard stores instances of the following unit classes:

- `Basic-task`—Serves as a superclass unit-class to provide slots to the task class (as well as to the `sim-action` class on the simulator blackboard)
- `Task`—Represents a task that has been planned for an elevator; serves as a superclass unit-class for the following unit classes:
 - `Move-task`—Represents a planned task for moving an elevator; serves as a superclass unit-class for the following unit classes:
 - ◊ `Move-up-task`—Represents a planned task for moving an elevator up
 - ◊ `Move-down-task`—Represents a planned task for moving an elevator down
 - `Door-cycle-task`—Represents a task that has been planned for an elevator to open its doors, wait, and close its doors
 - `Wait-task`—Represents a task that has been planned for an elevator to wait while awaiting the next planned task

The simulator blackboard

Space hierarchy

The simulator blackboard represents the space-definition hierarchy rooted at the simulator space.

The inheriting spaces are as follows:

- Cabs—Stores instances of the cab unit class
- Sim-actions—Stores instances of the sim-action unit class
- Orders—Stores instances of the order unit class
- Clock—Stores the instance of the world-clock unit class

Unit classes

The simulator blackboard stores instances of the following unit classes:

- Basic-task—Serves as a superclass unit-class to provide slots to the sim-action class (as well as to the task class on the planner blackboard)
- Sim-action—Represents actions being performed by the simulator; serves as a superclass unit-class for the following unit classes:
 - Move-sim-action—Represents an action being performed that moves an elevator; serves as a superclass unit-class for the following unit classes:
 - ◇ Move-up-sim-action—Represents an action being performed that moves an elevator up
 - ◇ Move-down-sim-action—Represents an action being performed that moves an elevator down **dv 195.0.0026 0.5(r)9.7(t)-(-a)-12(0qor)Tj /Fcy.64 44**

Dimension-value computations

For extracting dimension values

The elevator 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 elevator application uses the following dimension-value computations:

- Elevator-type-dvc
- Elevator-name-spec-dvc
- Elevator-location-dvc
- Elevator-floors-dvc
- Bank-name-spec-dvc
- Bank-spec-from-elevator-spec-dvc
- Floor-range-dvc
- Order-call-button-dvc
- Task-type-dvc

Knowledge sources

Performing problem-solving activities

The application uses the following knowledge sources (KSs) to perform problem-solving activities:

- Initial-ks—Initializes the system; triggered by system initialization
- Read-orders—Reads orders from a file; also triggered by system initialization
- Generate-random-orders—Generates additional orders with random characteristics; triggered by queue quiescence
- Simulator-interface—Issues a command to the simulator to perform an action; triggered when the planner component creates or modifies a task
- Simulation-update—Performs all simulator actions (signaling button presses, opening doors, moving the elevator cab, and so on); triggered when the clock is updated
- Complete-action—Handles late simulator actions, and creates a new simulator action for the next scheduled task; triggered when the simulator component completes an action
- Answer-call-button—Plans a task (which might modify the existing plan) to send an elevator to answer a call button; triggered when a call button is pressed

- Floor-button-request-ks—Adds a task (which might modify the existing plan) to take the elevator to the designated floors; triggered when a floor button is pressed
- Update-modified-task—Updates a task after significant changes have been made; triggered when the planner component modifies a plan
- Clock-ks—Updates the clock when all the planning and simulation is complete for the current clock tick; triggered by queue quiescence

Agenda control shell

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, update-modified-task, uses a precondition function; the purpose of the precondition function is to determine how important it is to update the task. 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 activations of the KS.

Unit-class hierarchy

The hierarchy of unit classes in the elevator application is shown below. In the hierarchy, the basic-unit class is the highest-level class, since in GBB it is the default superclass unit class.



Loading and running the elevator example

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

Loading and running the elevator example

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

- 1 Load the `elevator-example` module:

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

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

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

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

```
(elevator-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.”

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. 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). If the Chain Manager window is displayed, click on the Previous button to display the Control Shell Window.

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

Error recovery under Allegro Common Lisp

If you're running the elevator 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 IDE in Allegro CL

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 elevator 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.

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 Elevator Example**.

GBB exits the control shell and deletes the blackboard windows, the Elevator Status window, and the Control Shell Window.

Building a suspended Lisp image

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

Small screen displays

If either of the following conditions exists, the configuration of the windows that are displayed is different than on a larger screen and the GBB Graphics System provides a Chain Manager Window:

- You are running the elevator example on a screen with resolution of 800 x 600 pixels or less.

- The size of your application window is insufficient to display all the graphics windows.

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 Floor Buttons window is displayed, the previous window is the Elevator Positions window.)
Next	Bring to the front the immediately following graphics window. (If the Planned Tasks for E window is displayed, the next window is the Elevator Positions window.)
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.

Where to find the source code

Examples subdirectory

The source code for the elevator example is stored in the following files of the examples subdirectory:

- `elevator-building-info.lisp`
- `elevator-example-graphics.lisp`
- `elevator-example.lisp`
- `elevator-orders.lisp`

Tables

The spaces and unit classes used by the elevator application are described by blackboard in the tables that follow.

The building-info blackboard

Spaces The building-info blackboard's spaces are described below:

Space	Description
building-info	The space has no dimensions (since it isn't intended to store unit instances).
bank-specs	Stores instances of the bank-spec unit class. It has only one dimension: <ul style="list-style-type: none">• floors—The floors serviced by the elevator bank; an ordered dimension with bounds of *top-floor* and *bottom-floor*
floor-button-specs	Stores instances of the floor-button-spec unit class. Its dimensions represent the current status of the floor button: <ul style="list-style-type: none">• elevator-spec—The elevator containing the floor button; an enumerated dimension that supports dynamically defined labels• floor—The floor number for which the button was pressed; an ordered dimension with bounds of *top-floor* and *bottom-floor*• lit?—An indicator of whether the floor-button light is lighted; an enumerated dimension with a value of <code>t</code> or <code>nil</code>
call-button-specs	Stores instances of the call-button-spec unit class. Its dimensions represent the current status of the elevator call-button: <ul style="list-style-type: none">• bank-spec—The elevator bank with which the call button is associated; an enumerated dimension that supports dynamically defined labels• floor—The floor at which the call button was pressed; an ordered dimension with bounds of *top-floor* and *bottom-floor*• lit?—An indicator of whether the call-button light is lighted; an enumerated dimension with a value of <code>t</code> or <code>nil</code>

Space	Description
elevator-specs	<p data-bbox="565 247 1427 325">Stores instances of the <code>elevator-spec</code> unit class. Its dimensions represent the static characteristics of the elevator:</p> <ul data-bbox="565 325 1427 758" style="list-style-type: none"> <li data-bbox="565 325 1427 409">• <code>elevator-type</code>—The type of elevator; an enumerated dimension with a value of <code>:passenger</code> or <code>:freight</code> <li data-bbox="565 409 1427 493">• <code>bank-spec</code>—The elevator bank in which the elevator is located; an enumerated dimension that supports dynamically defined labels <li data-bbox="565 493 1427 577">• <code>x</code>—The x-axis coordinate specifying the location of the elevator in the building; an ordered dimension with bounds of 0 and 100 <li data-bbox="565 577 1427 661">• <code>y</code>—The y-axis coordinate specifying the location of the elevator in the building; an ordered dimension with bounds of 0 and 100 <li data-bbox="565 661 1427 758">• <code>floors</code>—The floors serviced by the elevator; an ordered dimension with bounds of *top-floor* and *bottom-floor*

Unit classes

The building-info blackboard's unit classes are described below:

Unit class	Description
bank-spec	Represents static information about the elevator bank. Defines the following slots: <ul style="list-style-type: none">• floors—Nonlink slot containing a list of floors serviced by the elevator bank• elevator-specs—Link slot for which instances are linked to the bank-spec slot of the elevator-spec class• call-button-specs—Link slot for which instances are linked to the bank-spec slot of the call-button-spec class
floor-button-spec	Represents the status of a floor-indicator button. Defines the following slots: <ul style="list-style-type: none">• elevator-spec—Link slot for which instances are linked to the floor-button-specs slot of the elevator-spec class• floor—Nonlink slot containing a floor number• lit?—Nonlink slot containing an indicator of whether the floor-button light is lighted
up-call-button-spec	Represents a call button requesting an elevator going up; inherits from the call-button-spec unit classes.
down-call-button-spec	Represents a call button requesting an elevator going down; inherits from the call-button-spec unit classes.
call-button-spec	Represents the status of an elevator call-button. Defines the following slots: <ul style="list-style-type: none">• bank-spec—Link slot for which instances are linked to the call-button-specs slot of the bank-spec class• orders—Link slot for which instances are linked to the call-button-spec slot of the order class• keyed—Nonlink slot containing a value indicating whether the call button requires a key for access (not used)• floor—Nonlink slot containing the number of the floor at which the call button was pressed• lit?—Nonlink slot containing an indicator of whether the call-button light is lighted

Unit class	Description
elevator-spec	<p data-bbox="578 260 1414 323">Represents static information about the elevator; inherits from GBB's 2d-point-mixin unit class. Defines the following slots:</p> <ul data-bbox="578 344 1414 1339" style="list-style-type: none"> <li data-bbox="578 344 1414 407">• 2d-point—Nonlink slot containing a bird's eye location of the elevator in the building <li data-bbox="578 428 1414 491">• tasks—Link slot for which instances are linked to the elevator-spec slot of the task class <li data-bbox="578 512 1414 575">• sim-actions—Link slot for which instances are linked to the elevator-spec slot of the sim-action class <li data-bbox="578 596 1414 659">• bank-spec—Link slot for which instances are linked to the elevator-specs slot of the bank-spec class <li data-bbox="578 680 1414 743">• floor-button-specs—Link slot for which instances are linked to the elevator-spec slot of the floor-button-spec class <li data-bbox="578 764 1414 827">• cab—Link slot for which instances are linked to the elevator-spec slot of the cab class <li data-bbox="578 848 1414 911">• capacity—Nonlink slot containing the capacity of the elevator in pounds <li data-bbox="578 932 1414 995">• type—Nonlink slot containing a value, :passenger or :freight, indicating the elevator type <li data-bbox="578 1016 1414 1079">• floors—Nonlink slot containing a list of the floors that the elevator services <li data-bbox="578 1100 1414 1163">• start/stop penalty—Nonlink slot containing the number of seconds the elevator takes to start moving after it stops <li data-bbox="578 1184 1414 1247">• speed—Nonlink slot containing the speed of the elevator, measured in the number of floors per second <li data-bbox="578 1268 1414 1339">• door-cycle-time—Nonlink slot containing the minimum amount of time the elevator takes to open its doors, wait a moment, and close its doors

The planner blackboard

Spaces

The planner blackboard's spaces are described below:

planner	The space has no dimensions (since it isn't intended to store unit instances).
tasks	<p>Stores instances of the <code>task</code> unit class. Its dimensions represent a planned task for an elevator:</p> <ul style="list-style-type: none">• <code>time-interval</code>—The time interval of the task; an ordered dimension with bounds of -1 and the value of *max-time*• <code>bank-spec</code>—The elevator bank in which the elevator is located; an enumerated dimension that supports dynamically defined labels• <code>elevator-spec</code>—The elevator; an enumerated dimension that supports dynamically defined labels• <code>floors</code>—The floors that the elevator is to service; an ordered dimension with bounds of *top-floor* and *bottom-floor*• <code>task-type</code>—The type of task; an enumerated dimension that supports dynamically defined labels

Unit classes

The planner blackboard's unit classes are described below:

Unit class	Description
basic-task	Abstract class that provides slots to the inheriting planner task class (as well as the simulator <code>sim-action</code> class); inherits from GBB's <code>time-interval-mixin</code> unit class. Defines the following slots: <ul style="list-style-type: none">• <code>time-interval</code>—Nonlink slot containing the time interval of the task• <code>floors</code>—Nonlink slot containing a starting and ending floor number, indicating a range of floors
task	Represents a task planned for an elevator; inherits from the <code>basic-task</code> class and provides slots to the inheriting <code>move-task</code> , <code>door-cycle-task</code> , and <code>wait-task</code> classes. Defines the following slots: <ul style="list-style-type: none">• <code>sim-action</code>—Link slot for which instances are linked to the <code>tasks</code> slot of the <code>sim-action</code> class• <code>elevator-spec</code>—Link slot for which instances are linked to the <code>tasks</code> slot of the <code>elevator-spec</code> class• <code>duration</code>—Nonlink slot containing a value indicating the duration of the task• <code>next-task</code>—Link slot for which instances are linked to the <code>previous-task</code> slot of the <code>task</code> class• <code>previous-task</code>—Link slot for which instances are linked to the <code>next-task</code> slot of the <code>task</code> class
move-task	Represents a planned task that moves the elevator up or down; inherits from the <code>task</code> class.
move-up-task	Represents a planned task that moves the elevator up; inherits from the <code>move-task</code> class.
move-down-task	Represents a planned task that moves the elevator down; inherits from the <code>move-task</code> class.
door-cycle-task	Represents the sequence of opening the doors, waiting with the doors open, and closing the doors; inherits from the <code>task</code> class.
wait-task	Represents the sequence of closing the doors and waiting with the doors closed; inherits from the <code>task</code> class.

The simulator blackboard

Spaces The simulator blackboard's spaces are described below:

Space	Description
simulator	The space has no dimensions (since it isn't intended to store unit instances).
cabs	Stores instances of the <code>cab</code> unit class. Its dimensions represent the dynamic characteristics of the elevator cab: <ul style="list-style-type: none">• <code>elevator-spec</code>—Static information about the elevator (used for planning the actions of the elevator cab); an enumerated dimension that supports dynamically defined labels• <code>floor</code>—The floor at which the cab is currently located; an ordered dimension with bounds of *top-floor* and *bottom-floor*• <code>weight</code>—The total weight of the people currently riding in the elevator cab; an ordered dimension with bounds of 0 and the value of *max-weight*• <code>status</code>—The status of the cab; an enumerated dimension with the following label set: <code>:stopped</code>, <code>:doors-open</code>, <code>:out-of-service</code>, <code>:moving-up</code>, <code>:moving-down</code>• <code>x</code>—The x-axis coordinate specifying the location of the elevator in the building; an ordered dimension with bounds of 0 and 100• <code>y</code>—The y-axis coordinate specifying the location of the elevator in the building; an ordered dimension with bounds of 0 and 100
sim-actions	Stores instances of the <code>sim-action</code> unit class. Its dimensions represent actions being performed by the simulator: <ul style="list-style-type: none">• <code>time-interval</code>—The time interval of the action; an ordered dimension with bounds of -1 and the value of *max-time*• <code>elevator-spec</code>—Static information about the elevator; an enumerated dimension that supports dynamically defined labels• <code>floors</code>—The floors that the elevator will stop at or pass by; an ordered dimension with bounds of *top-floor* and *bottom-floor*• <code>executed?</code>—An indicator of whether the simulated actions have been executed; an enumerated dimension with a value of <code>t</code> or <code>nil</code>
clock	Stores instances of the <code>world-class</code> unit class. It has only one dimension: <ul style="list-style-type: none">• <code>time</code>—The current time; an ordered dimension with bounds of -1 and the value of *max-time*

Space	Description
orders	<p>Stores instances of the <code>order</code> unit class. Its dimensions represent an elevator order:</p> <ul style="list-style-type: none">• <code>time</code>—The time at which the order was placed; an ordered dimension with bounds of -1 and the value of *max-time*• <code>start-floor</code>—The floor at which the call button was pressed; an ordered dimension with bounds of *top-floor* and *bottom-floor*• <code>up/down</code>—An indicator of whether the call button was an up or down button; an enumerated dimension with a value of <code>:up</code> or <code>:down</code>• <code>answered</code>—An indicator of whether an elevator has stopped at the requested floor; an enumerated dimension with a value of <code>t</code> or <code>nil</code>• <code>bank-spec</code>—The elevator bank selected by the call button; an enumerated dimension that supports dynamically defined labels

Unit classes

The simulator blackboard's unit classes are described below:

Unit class	Description
<code>basic-task</code>	<p>Abstract class that provides slots to the inheriting simulator <code>sim-action</code> class (as well as the planner task class); inherits from GBB's <code>time-interval-mixin</code> unit class. Defines the following slots:</p> <ul style="list-style-type: none">• <code>time-interval</code>—Nonlink slot containing the time interval of the task• <code>floors</code>—Nonlink slot containing a starting and ending floor number, indicating a range of floors
<code>sim-action</code>	<p>Represents actions being performed by the simulator; inherits from the <code>basic-task</code> class. Defines the following slots:</p> <ul style="list-style-type: none">• <code>task</code>—Link slot for which instances are linked to the <code>sim-action</code> slot of the task class• <code>new</code>—Nonlink slot containing a <code>t</code> value if the simulated action is a new one• <code>elevator-spec</code>—Link slot for which instances are linked to the <code>sim-actions</code> slot of the <code>elevator-spec</code> class• <code>cab</code>—Link slot for which instances are linked to the <code>sim-action</code> slot of the <code>cab</code> class
<code>move-sim-action</code>	<p>Represents an action that moves the elevator up or down; inherits from the <code>sim-action</code> class. Defines the following slots:</p> <ul style="list-style-type: none">• <code>speed</code>—Nonlink slot containing a value indicating the speed of the elevator in floors per second• <code>penalty</code>—Nonlink slot containing a value indicating any acceleration or deceleration penalty time remaining for the action
<code>move-up-sim-action</code>	<p>Represents an action that moves the elevator up; inherits from the <code>move-sim-action</code> class.</p>
<code>move-down-sim-action</code>	<p>Represents an action that moves the elevator down; inherits from the <code>move-sim-action</code> class.</p>
<code>door-cycle-up-sim-action</code>	<p>Represents the sequence of opening the doors, waiting with the doors open, and closing the doors, to be followed by an action that moves the elevator up; inherits from the <code>sim-action</code> class.</p>
<code>door-cycle-down-sim-action</code>	<p>Represents the sequence of opening the doors, waiting with the doors open, and closing the doors, to be followed by an action that moves the elevator down; inherits from the <code>sim-action</code> class.</p>
<code>door-cycle-wait-sim-action</code>	<p>Represents the sequence of opening the doors and waiting with the doors open; inherits from the <code>sim-action</code> class.</p>

Unit class	Description
wait-sim-action	Represents the sequence of opening the doors and waiting with the doors closed; inherits from the <code>sim-action</code> class.
cab	<p>Represents the current condition of the elevator cab. Defines the following slots:</p> <ul style="list-style-type: none"> • <code>elevator-spec</code>—Link slot for which instances are linked to the <code>cab</code> slot of the <code>elevator-spec</code> class • <code>orders</code>—Link slot for which instances are linked to the <code>cab</code> slot of the <code>order</code> class • <code>sim-action</code>—Link slot for which instances are linked to the <code>cab</code> slot of the <code>sim-action</code> class • <code>status</code>—Nonlink slot containing a value indicating whether the elevator cab is currently stopped or moving • <code>current-floor</code>—Nonlink slot containing a number indicating the floor at which the cab is currently • <code>current-weight</code>—Nonlink slot containing a number indicating the current weight of the cab
world-clock	Represents the current time. Defines one slot, called <code>time</code> , which is a nonlink slot containing the time.

Unit class	Description
order	<p data-bbox="519 254 1383 430">Represents a person approaching a bank of elevators on a particular floor, pressing an up or down button, waiting for the elevator doors to open, entering the elevator, pressing one or more destination floor buttons, and then exiting the elevator at the desired floor. Defines the following slots:</p> <ul data-bbox="519 430 1383 1266" style="list-style-type: none"> <li data-bbox="519 430 1383 514">• <code>time</code>—Nonlink slot containing the time at which the order originates <li data-bbox="519 514 1383 598">• <code>cab</code>—Link slot for which instances are linked to the <code>orders</code> slot of the <code>cab</code> class <li data-bbox="519 598 1383 682">• <code>call-button-spec</code>—Link slot for which instances are linked to the <code>orders</code> slot of the <code>call-button-spec</code> class <li data-bbox="519 682 1383 766">• <code>call-button-actions</code>—Nonlink slot containing either <code>:up</code> or <code>:down</code> <li data-bbox="519 766 1383 850">• <code>dest-floors</code>—Nonlink slot containing the number of each floor for which a floor button will be pressed when the person gets on <li data-bbox="519 850 1383 934">• <code>departure-floor</code>—Nonlink slot containing the number of the floor at which the person exits the elevator <li data-bbox="519 934 1383 1018">• <code>weight</code>—Nonlink slot containing the total weight of the people on the elevator <li data-bbox="519 1018 1383 1102">• <code>entry-time</code>—Nonlink slot containing a number indicating the amount of time in seconds it takes the person to get on the elevator <li data-bbox="519 1102 1383 1186">• <code>exit-time</code>—Nonlink slot containing a number indicating the amount of time in seconds it takes the person to get off the elevator <li data-bbox="519 1186 1383 1266">• <code>answered</code>—Nonlink slot containing a value indicating whether an elevator has stopped at the floor at which the call button was pressed

For more information

- **On using the elevator example**—If you are running the elevator example on an x86 computer running Windows 95/98/NT, you should also refer to the document titled *A Walk Through the Elevator Example*. The document introduces basic GBB concepts, gives an overview of the GBB Graphics System windows used by the GBB elevator example, and then describes how to use elevator example with Windows.
- **On GBB**—For a wide range of information about GBB, see the GBB document set.

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, 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
E-mail: gbb-info@KTImworld.com
Fax: (413) 256-3179
WWW: www.KTImworld.com



KNOWLEDGE
TECHNOLOGIES
INTERNATIONAL

www.KTIworld.com