IntelligentBox Users Guide

IntelligentBox Ver. 5.11.23

written by Yoshihiro Okada,

on 7th, Dec. 1998.

This guide describes the usage of the IntelligentBox(IB) system.

CONTENTS

1. Overview of the IntelligentBox system

1) Available Machines

Silicon Graphics Inc. Graphics Workstations(all types of machines)

2) Development environment

Programming Language: C and C++

Library: OpenGL and ViewKit(bundled in the C++ translator released by SGI)

3) Concerned system

S-Products(used as a base platform for the first version of IntelligentBox)

N-World(SGI Machine version of S-Products)

2. Use and Distribution

1) Distribution software: ib.tar.gz(execution file and sample files)

2) Restriction for use and distribution is described in readme-e.doc

3. Install and Execution

1) uncompress ib.tar.gz file by gunzip to create ib.tar file.

Ex.) gunzip ib.tar.gz

2) extract ib.tar file by tar to create ib directory.

Ex.) tar xvf ib.tar

3) execute ib file existing under ib/bin/ to open the IntelligentBox window.

Please refer ib.cshrc file to define environment variables. However, it is possible to execute ib without defining them.

4) Directories

ib/

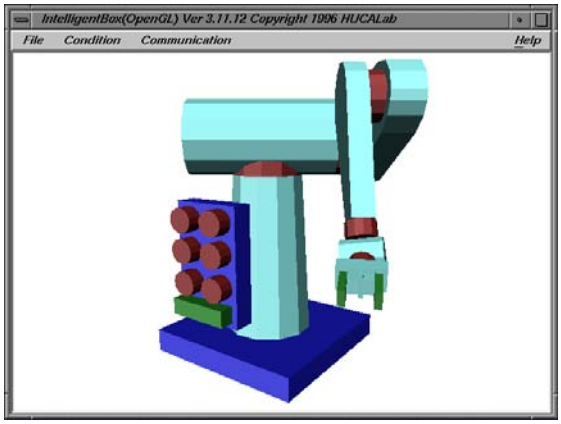
agents/	externally defined agent files(*.dll).
bin/	execution files.
boxdata/	box data(*.box, original format files of IntelligentBox).
doc/	documents(*.doc).
dxfdata/	3D model data(*.dxf files of AutoCAD).
ee/	interface definition files for expression editor(don't care).

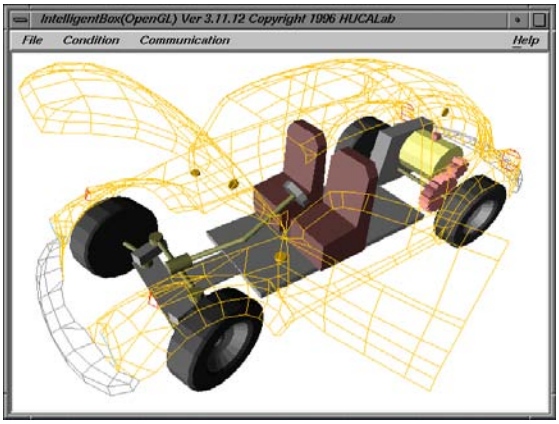
expdata/	expression data(don't care).
geodata/	3D model data(*.geo files of ViewPoint).
guide/	this file(MS Word97: guide-e.doc) and the html file(index-e.html).
ib.cshrc	sample file for defining environment variables.
include/	include files.
labandata/	don't care.
lib/	dynamic link objects(*.so files).
libsrc/	dynamic link object source files.
movie/	movies(*.mov files).
objdata/	3D model data(*.obj files of Wavefront).
primdata/	3D model data(*.dat, original format files of IntelligentBox).
server/	host names file(hosts file).
smdata/	3D model data(*.sm.txt files, don't care).
sound/	sounds(*.wav files).
src/	source program files.
texture/	texture data files for texture mapping.
tools/	tools directory.

4. Summary of IntelligentBox

4.1 Concept

We have been studying 3D software development systems. Our goal is to establish software architecture that makes it easier to construct various 3D graphics applications. We have already developed a prototype system called IntelligentBox. This system provides users with the ability to construct 3D graphics applications by means of combining individually existing 3D functional objects through direct





manipulations on a computer screen. In this system, all functional objects are called boxes. Each box has a unique function and a 3D visible shape. With the use of these boxes, developers can construct interactive 3D graphics applications without making any programs. In the IntelligentBox system, the construction of composite boxes from individually existing primitive boxes is regarded as the construction of 3D graphics applications.

4.2 Fundamental mechanism

IntelligentBox employs the same essential mechanisms as IntelligentPad, which is a synthetic media system developed at Hokkaido University. Therefore, IntelligentBox is regarded as a 3D extension of IntelligentPad. The IntelligentBox system has four basic mechanisms inherited from the IntelligentPad system. They are (1) an MVC sructute, (2) a Parent-Child relationship, (3) a Message sending protocol for a slot-connection and (4) a Model-sharing mechanism. The following four sections explain these mechanisms briefly.

4.2.1 An MVC structure of a box

An MVC structure of IntelligentBox is similar to the Model-View-Controller(MVC) modeling used for the window system of the Smalltalk-80[6]. The pair of a view and a controller is called a display object. Then, each box consists of two individual objects, i.e., a model and a display object. A model holds states of a box. Indeed, states of a box are stored in variables called slots. A display object defines how a box appears on a computer screen and defines how the box reacts to user operations. It decodes input events and reacts adequately, or it sends an adequate message to its corresponding model if the user operation is one that changes its slot value.

Some boxes provided by the IntelligentBox system are geometrical-information handling boxes such as a RotationBox and an ExpandBox. According to user manipulations, a RotationBox rotates and a specified face of an ExpandBox moves up or down. These reactions are regarded as functions of those boxes. These geometrical-information handling boxes have one degree of freedom, e.g., rotation, elasticity, and so on. These boxes have a slot value that means one of these degrees of freedom. Through direct manipulations on a box, its associated slot value changes. Furthermore, its visual image simultaneously changes according to the slot value change. In this way, each box reacts to users manipulations according to its function.

4.2.2 Parent-child relationship between boxes

The slot connection is the connection between a slot of one box and a slot of another box. There is the restriction that the slot connection is available only when there is a parent-child relationship between two boxes. A parent-child relationship between two boxes is defined as a relationship between their two display objects.

4.2.3 Message sending protocol for slot-connections

The slot-connection connects a child box slot to a parent box slot by the following message sending protocol. These messages transfer data between two mutually connected slots. This data transfer works to combine the two functions of the corresponding child box and its parent box because each box has a unique function associated with its slot value. There are three standard messages, i.e., a set message, a gimme message and an update message. These messages have the following formats:

(1) parent box set <slotname> <value>.

(2) parent box gimme <slotname>.

(3) child box update.

A <value> in a format (1) means any value kept by the child box that issues this set message, and a <slotname> in formats (1) and (2) means a user-selected slot of the parent box that receives these two messages. Each box has two procedures associated with a set message and a gimme message, which are executed when it receives either a set message or a gimme message.

Each box possesses five flags that control the above message-flows, i.e., the set flag, the gimme flag, the update flag, the update-acceptance flag and the through flag. These five flags are properties of the DisplayObject. A box works as an input device if its set flag is set true. Contrarily, a box works as an output device if its gimme flag is set true. A box sends update messages to all of its child boxes to tell them that its slot values have changed if its update flag is set true. Child boxes take an action depending upon their states of the set flags and the gimme flags after they receive an update message or changes its slot values. The update-acceptance flag specifies whether a box can accept an update message or not. If two boxes do not need the functions of their intermediate boxes, the three messages should be directly transferred between them. In this case, the through flag is used and should be set true.

4.2.4 Model-sharing and Model substitution mechanism

The MD structure allows more than one box to share the same model. We call this

mechanism Model-Sharing. In Model-Sharing, data stored in the slots of a model are shared among several different boxes because the model is shared among these boxes. This mechanism makes it possible to transfer data between any two boxes physically apart from each other without sharing any common parent box. As mentioned above, the slot connection between two boxes does not work if they have no parent-child relationship. Hence, the IntelligentBox as well as the IntelligentPad provides the model-sharing to supplement this weakness.
The IntelligentBox allow users to build or expire this linkage. After expiring it, the original model part of the box will be reassigned as its model. We call this mechanism model-substitution.

4.3 Development of 3D graphics applications using IntelligentBox

The IntelligentBox system enables users to construct a composite complex(intelligent) box by means of combining already existing boxes interactively on a computer screen. This construction process is regarded as a development process of 3D graphics applications. After you once develop a new box that has a special function, you can use it combined with other already existing boxes whenever you want. Furthermore, you can distribute it freely to other end-users.

5. IB window and its menus

5.1 Pulldown menus

At the top column of the IB window, there are pulldown menus[File | Condition | Communication | Windows].
In the following, (T) means a toggle switch.
[File]

[Load Polyhedron(s)] Load model(s)(polyhedron(s)) from *.dat, *.sm.txt, *.obj, *.dxf, *.geo format files.

[Save All Polyhedrons] Save all of model(polyhedron) existing in the main window in *.dat, *.sm.txt, *.obj formats.

[Load Box(es)] Load box(es) from *.box format files.

[Save All Boxes] Save all of boxes existing in the main window in *.box format.

[Kill All] Kill all of boxes and polyhedrons existing in the main window.

[Test]　Don't care

[DebugMode](T)　When this flag is set true, the system issues detailed reports or messages.

[Exit]　Quit the IB system.

[Condition]

[GlobalAxis](T)　Expose the global axis(the long white axis) or not.

[CenterAxis](T)　Expose the center axis(the short red axis) or not.　The center axis means the aim point of the eye direction.

[FixToOrigin]　Align the global axis with the center axis.

[Change BGColor]　Change the background color(Black or White).

[Default Light]　The parameters of the default light.

[Enable](T)　Enable or disable the default light.

[ShowLightVector](T)　Expose the direction vector of the default light.

[SelectColor]　Set color factors of the default light.

[Ambient]　Ambient color of the default light.

[Diffuse]　Diffuse color of the default light.

[Specular]　Specular color of the default light.

[DrawMode]　Change the drawing mode

[Shade(Flat)]　Flat shading mode.

[Shade(Gouraud)]　Gouraud shading mode.

[Shade(Texture)]　Currently this is the same as [Shade(Flat)].

[Wireframe]　Wireframemode.

[Shade(Texture)]　Gouraud shading and hardware texture mapping mode.

[FrontFace](T)　Render or not front faces.

[BackFace](T)　Render or not back faces.

[SenseMode]　Change sense mode

[FrontFace](T)　All front faces are objectives of user manipulations.

[BackFace](T)　All back faces are objectives of user manipulations.

[ActivateProcess](T)　When this toggle switch is set false, any process Boxes do not work.

[Measurement Window](T)　Expose the Measurement window.

[Subsidiary Window](T)　Expose the Subsidiary window.

[TreeGraph Window](T)　Expose the TreeGraph window.

[Button Panels](T)　Expose or not the button panels.

[Communication] Menus for network(Socket) connections used by a RoomBox

[SERVER](T) Activate the server which receives messages sent from other RoomBoxes at different machines.

[CLIENT] Check states of other servers of different machines

[Send] Send an arbitrary message to the machine selected by the host selection menu.

5.2 Primitive selection buttons

Below the pulldown menus, there is the group of the buttons[collect | none | points | segments | faces | bodies]. These buttons are primitive selection buttons. Here, primitives mean points, segments, faces and polyhedrons. Primitives corresponding to the last pushed button become objectives of user operations. The primitive nearest to the mouse cursor becomes the current selected primitive. A selected primitive is highlighted. For instance, after pushing the [faces] button, only one of the faces of all polyhedrons is highlighted. Indeed, this is the face nearest to the mouse cursor. Furthermore, after clicking the mouse right button, the menu corresponding to the selected primitive will appear. There are a points-menu, a segments-menu, a faces-menu and a bodies-menu. These menus are for modeling operations.

[none] Objectives of user operations are none.

[points] Points are objectives of user operations.

[segments] Segments are objectives of user operations.

[faces] Faces are objectives of user operations.

[bodies] Bodies are objectives of user operations.

[collect] Multiple primitives are selected.

How to select multiple primitives(for example, multiple faces):

Push the [faces] button and the [collect] button sequentially to enter the multiple face-selection mode. Move the mouse cursor and highlight one of faces. At this time, click the mouse right button with pushing the SHIFT-KEY to register the currently highlighted face. Please register other faces you want in this way.

When you want to cancel the register operation, please highlight the corresponding face and click the mouse right button with pushing the SHIFT-KEY.

When you want to quit the multiple primitive-selection mode, push one of buttons except the [collect] button.

5.3 Eye-motion mode buttons

Below the group of the primitive selection buttons, there is another group of the buttons[zoom+|rotate|translate|global|fixed-aim]. These buttons are for eye-motion modes.

While no object is selected, after clicking the mouse left button, it is possible to enter eye-motion mode. According to the mouse movement, eye-direction is changed.

[zoom+] Change the view angle.

[rotate] Rotate the eye position.

[translate] Translate the eye position.

[global] Don't care because not used.

[fixed-aim] Don't care because not used.

5.4 Main menu

While no object is selected, after clicking the mouse right button, the main menu will appear.

[Load Polyhedron(s)] same as the [Load Polyhedron(s)] of the pulldown menu[File].

[Save All Polyhedrons] same as the [Save All Polyhedrons] of the pulldown menu[File].

[Load Box(es)] same as the [Load Box(es)] of the pulldown menu[File].

[Save All Boxes] same as the [Save All Boxes] of the pulldown menu[File].

[Kill All] same as the [Kill All] of the pulldown menu[File].

[Move LightVector] Move the position of the default light.

[TranslateXYZtoZXY] Translate XYZ coordinate values of the vertices of all objects appearing on the main window into ZXY coordinate values.

[Flip SensibilityMode] Change the Sensibility flag of the system.

Each object(polyhedron) also has its own Sensibility flag.

1) If the Sensibility flag of the system is true, all objects are sensible(selectable).

2) If the Sensibility flag of the system is false, the objects whose Sensibility flag is true are sensible.

You can change the Sensibility flag of a certain object by choosing the menu item [Flip Sensibility] of the bodies-menu or the Box menu.

[Flip VisibilityMode] Change the Visibility flag of the system.

As well as the Sensibility flag, each object also has its own Transparent flag.

1) If the Visibility flag of the system is true, all objects are visible.

2) If the Visibility flag of the system is false, the objects whose Transparent flag is true are invisible.

You can change the Transparent flag of a certain object by choosing the menu item [Flip Visibility] of the bodies-menu or the Box menu.

5.5 Box menu

If the highlighted object has already become a box, after clicking the mouse right button, it is possible to open the box menu. You can define a parent-child relationship between two boxes and connect their slots mutually by the box menu.

6. Operations

6.1 Change eye position.

When there is no highlighted object, it is possible to enter the eye motion mode after clicking the mouse left button. The eye position and the view angle change according to the mouse movement. Clicking the mouse left button again is to exit from the eye-motion mode.

6.2 Load model(s)(polyhedron(s)) data.

It is possible to read a model(polyhedron) data file by choosing the menu item [Load Polyhedron(s)] of the pulldown menu [File] or of the main menu. At the present, the IntelligentBox can read *.dat, *.sm.txt, *.obj, *.dxf and *.geo format files. After choosing the menu, the corresponding polyhedron(s) will appear on the main window. However, these have not become boxes yet.

6.3 Primitive selection and Primitive menus

After pushing any primitive selection buttons, the corresponding primitive nearest to the mouse cursor become highlighted. Then, please click the mouse right button with pushing the CTRL-KEY to open the primitive menu corresponding to the current

highlighted primitive. After this, the primitive menu will appear.
The primitive-menus are provided for shape modeling. With these menus, it is possible to modify polyhedrons' shapes. However, these operations are a little unstable so that the IB system sometimes breaks down. When using these operations, it is necessary to take care. Please make backup files through the following operations.
1) When there is no highlighted object, after clicking the mouse right button with pushing the S-KEY, the backup files named 'backuppolyhedrons.data' and 'backupboxes.box' will be generated.
2) When there is any highlighted object, after clicking the mouse right button with pushing the S-KEY, the backup files named 'backuppolyhedron.dat' and 'backupbox.box' will be generated.
When the IB system breaks down unfortunately, please start the IB system again and read these backup files.

6.4 Create, Load and Save box
How to transform a polyhedron into a Box:
1) Change the primitive selection mode into the [bodies].
2) Move the mouse cursor to highlight(select) one of the polyhedrons already read from files. Then, please click the mouse right button with pushing the CTRL-KEY to open the bodies-menu(one of the primitive-menu).
3) Choose the menu item [Transform...] to open the menu list of the DisplayObject names and furthermore choose one of them.
Through these operations(1),(2)and(3), it is possible to transform a polyhedron into a box.
Please try to create a RotateBox.

How to save a box into a file.
4) Choose the menu item [Save...] of the box menu and set a name you want in the file selection-dialogue box.

How to load a box from a file.
5) Choose the menu item [Load Box(es)] of the main menu or of the pulldown menu [File] to read a *.box file.
After this, any objects will appear. These objects have already become boxes.

6.5 Box manipulations

All boxes react to user manipulations. When clicking the mouse left button while any box is highlighted, it is possible to enter the box manipulation mode. After this, for example, a RotateBox rotates according to a user' mouse moving.

Please click the mouse left button while any RotateBox is highlighted. At this time, a warning message window appears. At the beginning, a RotateBox doesn't work correctly because it doesn't know how it rotates itself. At first, users have to select one of the faces. Its normal direction becomes the rotation axis.

When a warning message window appears, please open the box menu and choose [SpecialMenu...]. After this, the submenu will appear. This submenu is RotateBox's unique menu. Furthermore, choose [SelectFaceOnRotate] of this submenu and select one of the faces by the mouse cursor. In this way, the RotateBox comes to rotate on the selected face.

As a trial, please click the mouse left button again while any RotateBox is highlighted. After this, the RotateBox will rotate according to your mouse moving. After clicking the mouse left button again, you quit the box manipulation mode.

6.6 Inspect model slots

As mentioned above, each box consists of the two individual parts, i.e., a model and a display object. A model holds states of a box. Indeed, states are stored in variables called slots. For instance, a RotateBox holds a rotation angle in its model slot called a ratio. This value means a rotation angle of the RotateBox. This value is normalized in zero to one(0 degrees to 360 degrees). It is possible to see slot values by choosing menu item [ShowSlot] after choosing the [SpecialMenu...] of the box menu. After this, a dialog window will appear. In this dialog, it is possible to see and modify slot values.

As a trial, please rotate a RotateBox and see its ratio slot.

As you checked, the rotation function of a RotateBox is linked to its ratio slot value. In this way, a function of a box is linked to one of its slot values. Therefore, it is possible to link two functions of two boxes by means of connecting their two slots each other. This is called a slot connection. The slot connection is available between two boxes only when there is a parent-child relationship between them.

6.7 How to define parent-child relationship

The following operations make a parent-child relationship between two boxes.

1) First, highlight(select) one box that you want to assign as a child, and choose [Store...] of the box menu.

After this, a confirmation message window will appear. Please push the [OK] button.

2) Next highlight(select) one box that you want to assign as a parent, and choose [Compose] of the box menu.

The following operations break a parent-child relationship.

1) please highlight(select) the box already defined as a child of any box, and choose the [Decompose] of the box menu.

After this, the confirmation message window will appear. Please push the [OK] button. Then the parent-child relationship will be broken.

6.8 How to connect slots

When there is a parent-child relationship between two boxes, it is possible to connect their two slots according to the following operations.

1) First, highlight(select) the child box, and choose the [SpecialMenu..] of the box menu.

After this, the submenu will appear.

2) choose [Connection Sheet] of this submenu to open the connection-sheet dialog window.

3) In this dialog window, choose one of the child box slots and one of the parent box slots to combine them.

Furthermore, set the toggle switches, i.e., the set flag, gimme flag etc.

4) Finally push the [OK] button to apply the current setting and to quit.

6.9 How to make a shared copy

As described in the section 4.2.1 and 4.2.4, it is possible to make a shared copy through the following operations.

1) Highlight(select) the box of which you want to make a copy and choose the [SharedCopy] of the box menu.

After this, a copy of the highlighted box will appear on the main window.

The shared copies and its original box share the same common model. Therefore, these boxes hold the same slot values. As a result, these boxes virtually work as the connectors mutually connected by an invisible cable. With using the shared copies, it is possible to transmit and receive any data through them between some boxes even if they are physically apart from each other without having the common parent box.

7. About already defined Boxes

You can check what boxes have already been developed through opening the box-names list.

This list will appear after choosing the [Transform...] of the bodies-menu.

You can also read the descriptions concerning the individual boxes by choosing the menu item [WhatIs...] of the box menu.

8. How to define(make) a new box

As explained in the section 4.3, when you want a new function, you have to make a program to define the function as a function of a certain box.

However, this distribution version of the IB system does not include source files.

9. Developers' version of IntelligentBox

Recently, we have been making another distribution version of the IB system for developers. If there are readers who want to get the developers' distribution version, we can send it to them.

Please contact the author of this document: okada@i.kyushu-u.ac.jp.