Thermal Designer

BRANZ Contract No. 85-024 Technical Report No. 24
Robert Amor
Department of Computer Science
University of Auckland.
December 1991

ABSTRACT

This document describes the creation of *ThermalDesigner* an application system based upon the ALF design manual. *ThermalDesigner* is the first application to integrate the *PlanEntry* geometrical input system and the *Forms* input system into a *Kea* program. This document concentrates mainly on the models of interaction made possible for the user by the use of these systems. This document examines the improved interaction and association between objects through the use of such systems.

CONTENTS

FIGURES

Thermal Designer

BRANZ Contract No. 85-024 Technical Report No. 24

Robert Amor
Department of Computer Science
University of Auckland.
December 1991

1. INTRODUCTION

This report examines the basic structure of the ThermalDesigner system. ThermalDesigner is based upon the ALF design manual produced and published by the Building Research Association of New Zealand (BRANZ). The current ThermalDesigner program is an adaptation of the text based system previously constructed at Auckland University by John Hamer.

ThermalDesigner makes use of several of the new systems added to Kea in the latest revision. Including:

- the Forms interface which allows forms associated with an object to be displayed and filled in by the user;
- the PlanEntry system which allows the user to describe the geometry of the building in a graphical environment, in a similar fashion to the top end CAD packages available to the industry;
- the Material Selection system which allows users to define the composition of a particular object, as in the ALF design manual, but which is extensible to allow the definition of the composition of an object from basic raw materials.

Disclaimer

As the code in ThermalDesigner is based upon the ALF design manual it is only an estimation of the loads on a building and its performance in a given location. As a further caveat, many of the formulae used in this system are taken from the charts in the ALF manual. As these formulae are approximate the results can only be held to be approximations until BRANZ validate the formulae or provide formulae for each of their charts and graphs. However, the encoding of the graphs and charts from the ALF manual into ThermalDesigner has reduced the number of questions that the user must be asked, in particular removing user error in reading off figures from charts and graphs.

1.1 The ALF Design Manual

The ALF design manual, subtitled 'An aid to thermal design of buildings', was written with the following aims (from the introduction to the ALF design manual):

> "This booklet allows you to predict the amount of space heating energy needed in New Zealand houses for 'standard' conditions, of continuous whole-house heating.
>
> With this booklet you can easily compare alternative designs, see the relative sizes of heat gains and losses from different parts of the building, and calculate an index of building thermal performance 'BPI'.
>
> The method considers the effects of thermal mass and solar overheating. It uses a factor called 'ALF', which stands for 'Annual Loss Factor', and accounts for the effects of temperature, solar radiation, and other climatic parameters."

While the first step of this project was to develop a system which mimicked the calculation procedures in the ALF design manual, the major considerations were how to:

- allow easy input of design information;
- provide feedback to the designer on the analysis of the design;
- allow the designer to test modifications and rectify a suspect design with minimal additional work.

Through discussions with Bill Irvine from BRANZ several areas where problems could be determined were detailed. Also explored were ways in which help could be provided at these points. These problems were classified into two classes:

- **Conformance violations**
    - Early determination of likely problems
        - Air leakage ratio
        - R-value ratios within and between areas
    - Isolating weak points in the design
        - Irregular building shapes giving greater joint lengths
        - Examining floor joists on upper storeys
    - Solar gain
    - Overheating (beyond scope of code)
    - R-value increases in walls, roofs, ceilings, etc.
        - Based on cladding materials
        - Including curtains and shades
    - Condensation

- **Other advice**
    - Ratios of insulation in each wall of a space
    - Toxicity of materials, cost advantages, etc.

After further talks with BRANZ it was decided that we could not include aspects of building design beyond the scope of the code into the ThermalDesigner system. This limited the amount of help that could be provided.

1.2 Materials

New Zealand Standards NZS 4218P and NZS 41214 define the methods to be used for calculating the R-value of an object comprising several layers of materials. The R-value is the main piece of information we wish to know about the component materials comprising any particular object in the building. The same information used to determine the R-value can be used in doing take-offs of quantities of materials needed, and to supply bracing information to structural analysis or code checking systems, etc. The reader is refered to the above standards for a complete description of the methods of calculating R-values, however, a brief outline of the method is detailed below.

The thermal resistance of an object is defined in terms of its R-value, in units of m^2.C/W. This is calculated from the thickness of the material (in metres) divided by the conductivity of the material (W/m.C), ie.

R-value = Thickness / Conductivity

In most cases the total resistance of an object is defined as the sum of all the individual resistances of the materials. However, where there is a ventilated air gap between materials, the R-value of all the materials between the ventilated air-gap and the outside (exterior atmosphere) is reduced to a half of their usual values.

A section through an object may be composed of several materials; for example where there is timber framing in a wall with the gaps filled with insulation. In this case the R-value must be calculated by working out the fraction of area represented by each material. This case is known as a thermal bridge. For example, in the case of two materials the R-value is calculated as follows:

1 / R-value = Fb / Rb + (1-Fb) / Rr

where Fb = fraction of area bridged

Rb = R-value of bridging material

Rr = R-value of gap material

In addition to the R-values of the wall materials, there are also small R-values associated with the air-surface on each side of the object. These values are:

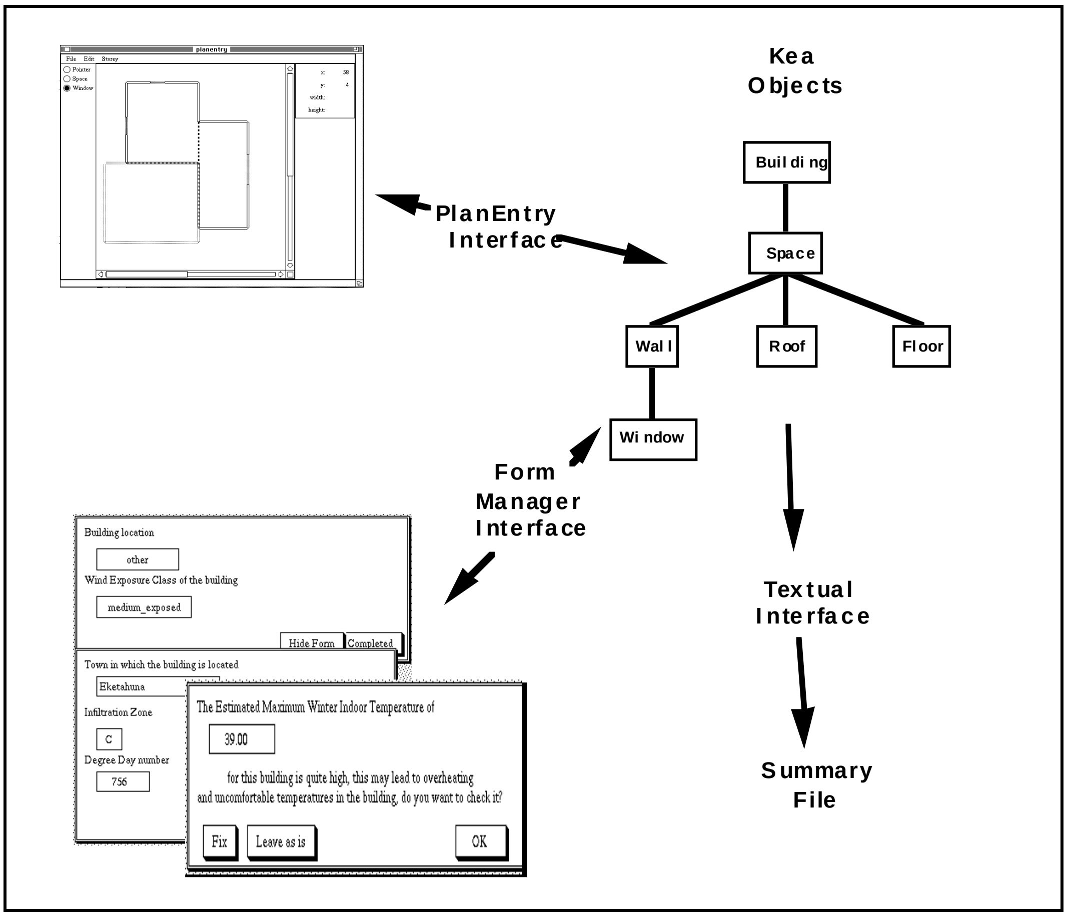
R-value (outside)	0.03 m2.C/W
R-value (inside)	0.09 m2.C/W

A few other unusual situations can affect the calculation of an objects R-value. These are detailed in the New Zealand Standards NZS 4218P and NZS 41214.

2. THERMALDESIGNER

Figure 1 shows an overview of the ThermalDesigner system. From this we see that ThermalDesigner includes the following:

- A graphical plan drawing system (PlanEntry) which provides much of the data required for the ALF calculations
- Forms for additional data entry, display of results, context sensitive help, and provision of feedback to the user when a design does not satisfy thermal design requirements.
- A summary file, summarises the data entered and the results produced
- A Kea program which performs the relevant calculations



Building location
other
Wind Exposure Class of the building
medium_exposed

Town in which the building is located
Eketahuna
Infiltration Zone
C
Degree Day number
756

The Estimated Maximum Winter Indoor Temperature of
39.00
for this building is quite high, this may lead to overheating
and uncomfortable temperatures in the building, do you want to check it?

Figure 1: Thermal Designer Overview

2.1 PlanEntry

In many building applications it is necessary to specify the geometry of the building and the relationship of spaces to each other. The PlanEntry system gives a method for users to enter geometrical information about a building in a manner similar to commercial CAD packages. Figure 2 shows an example of the PlanEntry system in use.

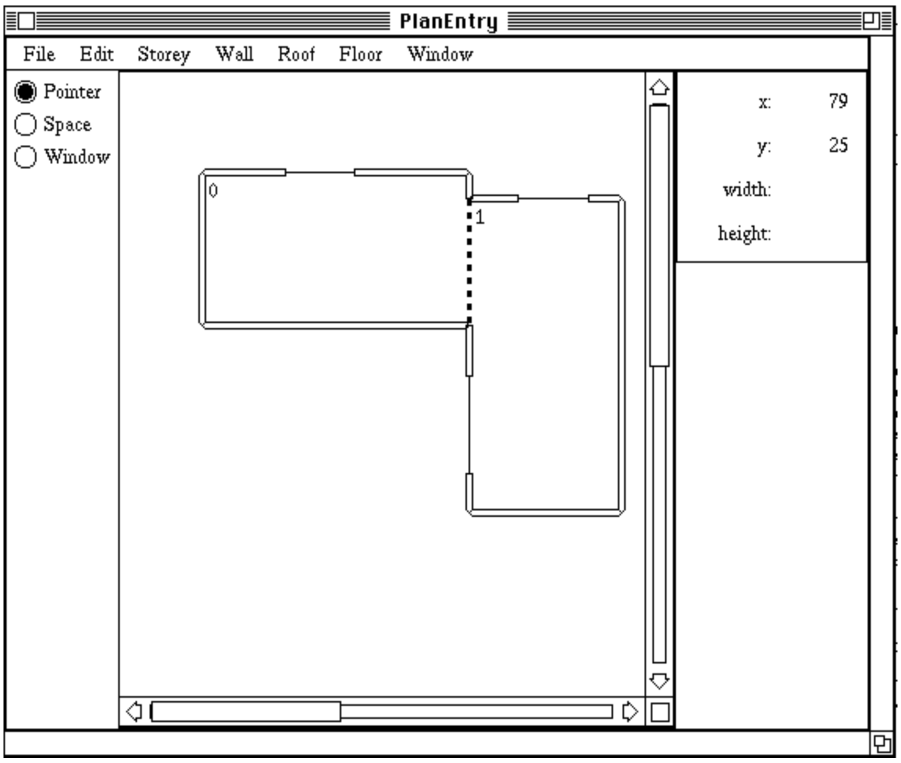


Figure 2: PlanEntry system in use

Users define a building by drawing collections of abutting rectangular spaces, each collection comprising one storey of the building. Windows can be added to any of the external walls of the spaces.

The creation of a space automatically creates the surrounding walls and the roof and floor of the space. Creation and tailoring of the space is achieved using a combination of mouse and menu operations:

- Spaces are created by clicking, dragging and releasing the left mouse button with the Space tool selected. While doing this, the user can see the dimensions of the space in the dimension box to the right of the screen.

- When laying a second space next to an existing space the user can only draw the space up to the edge of the existing space. Internal walls will automatically be inserted into the abutting wall lines.

- The cursor automatically places itself at an edge or corner if it is clicked near an existing edge or corner. This makes it simple to start or end spaces adjacent to an existing space.

- By clicking and dragging with the middle mouse button the user can move an existing space around the drawing. When the button is released, all affected spaces will re-calculate their boundaries.

- By clicking and dragging with the right mouse button, the user can re-size a space in any direction.

- Clicking on an already drawn object with the left mouse button selects that object.

- Dragging with the left mouse button, starting from outside any existing object, will create a Marqui box. When the button is released, all objects within the outlines of the box will be selected.

- Objects in the current selection may be cut, copied and pasted using the appropriate edit menu entries.

Windows are drawn by clicking dragging and releasing the left mouse button. They can only be drawn inside a wall and stretch along the direction of the wall they were started in. When a space is placed abutting an existing wall any overlaid windows in the existing wall are removed.

Four pull down menus, one each for wall, roof, floor, and window materials, allow the user to define a material combination to be used for their building. Each menu has a "currently selected entry" from the list of entries offered. This is set by selecting the appropriate entry from the pull down menu. When a space is layed down, the currently selected materials for wall, window, and roof will be used in constructing the space. Similarly, the currently selected window material is used when inserting windows. The material of an already drawn object can be changed by selecting the object (see above) and then selecting the new material from the appropriate material menu.

Each menu also has a "New..." entry, which permits users to add additional materials to the menu. Selecting this entry invokes the form-based materials selection system described in the next section.

New storeys may be added (or deleted) from the appropriate pull down menu. When a new storey is created the spaces on the current storey are faded so they are still visible but don't interfere with the placement of new spaces. The user can still click to walls and corners of the lower storey giving a simple method for laying abutting spaces on several storeys.

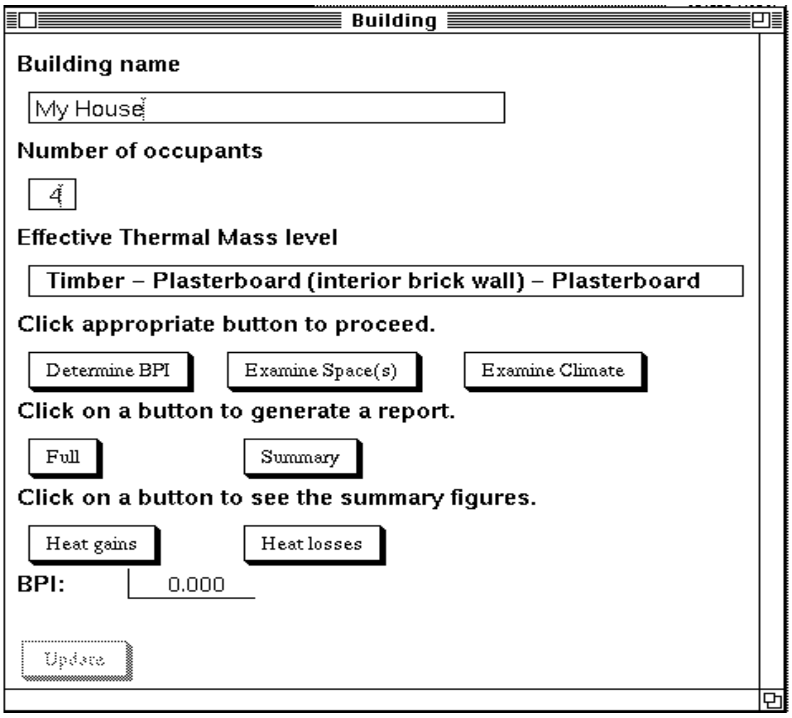
Plans drawn by the PlanEntry system are only made available to the Kea portion of ThermalDesigner when the "Commit" option from the File menu is selected.

2.2 Forms

ThermalDesigner uses forms to:

- request data from the user, including:
  - information for material selection
  - location and climate information
  - information supplementing the geometric data supplied by PlanEntry
- present results;
- provide help;
- provide feedback on design errors;
- provide control of the running application.

These categories are often combined on one form. For example, Figure 3 shows a form requesting input data on the building, but which also has result information (the BPI value) and buttons permitting the user to fill in information on other forms before supplying the information requested on the current form. Thus the form combines data entry, result presentation, and navigational control information on the one form.



Building name

My House

Number of occupants

4

Effective Thermal Mass level

Timber – Plasterboard (interior brick wall) – Plasterboard

Click appropriate button to proceed.

Click on a button to generate a report.

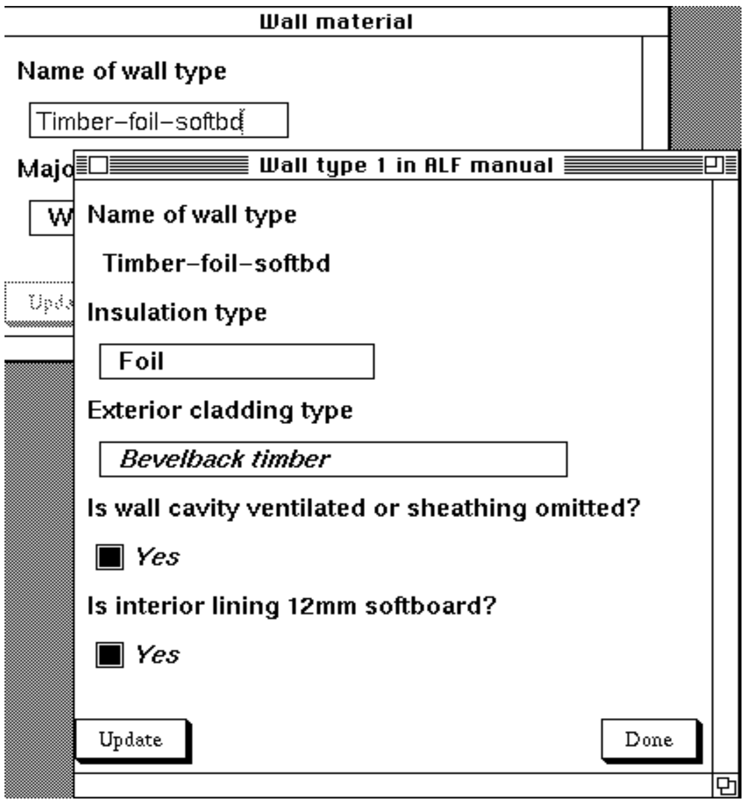
Click on a button to see the summary figures.

BPI: 0.000

Figure 3: A Building form

The user is able to move back and forth between any form on the screen, modifying values as required, by making us of the change facility, and examining the impact this has on the final results.

To aid the user in interacting with the system users can request summary information forms to be displayed. In the example of the building form, these summary forms detail the heat gains and heat losses summed for walls and windows facing in various directions, as in the ALF worksheets. This information is generated by analysing the described building, and when a change is made to any portion of a building, its impact will be mirrored in the summary information displayed in these forms.



Wall type 1 in ALF manual

Name of wall type

Timber–foil–softbd

Insulation type

Foil

Exterior cladding type

Bevelback timber

Is wall cavity ventilated or sheathing omitted?

■ *Yes*

Is interior lining 12mm softboard?

■ *Yes*

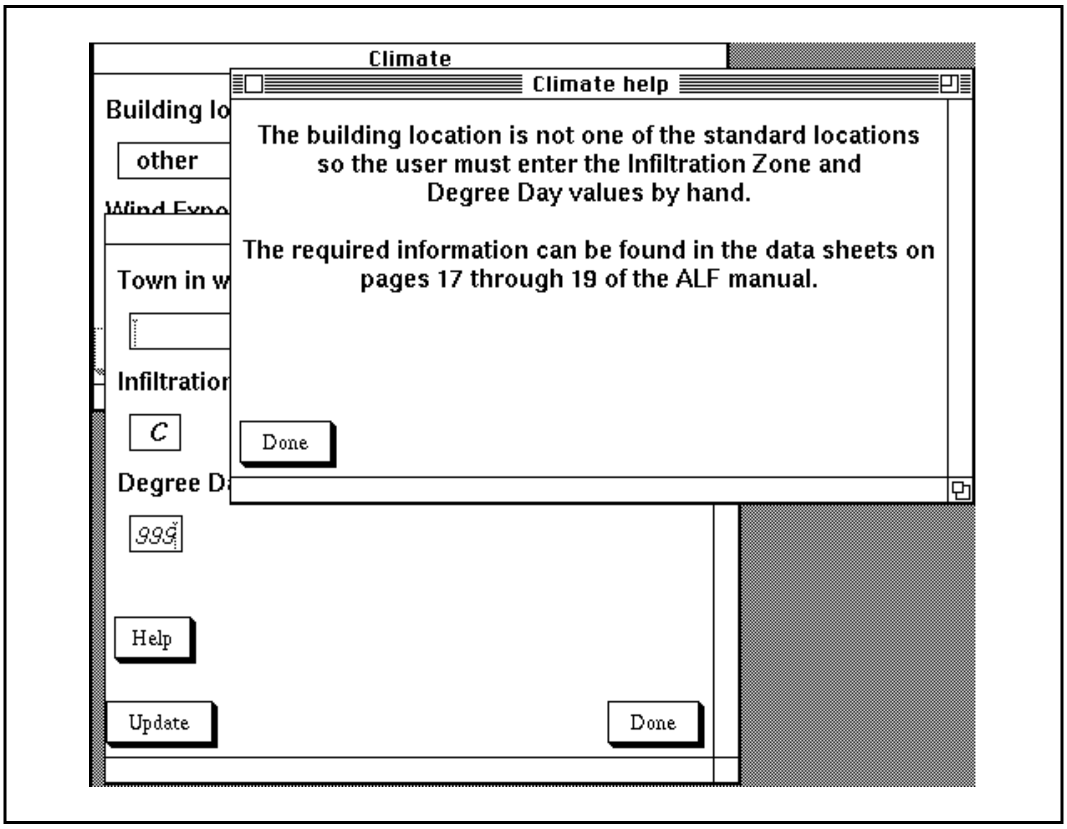
Figure 4: Wall material selection example

2.2.1 Materials selection

It is often problematic to quickly and simply define the material layers composing an object, such as a wall. This is a non-trivial problem for several reasons. In ALF there are a huge number of materials requiring the development of a fast selection method. Materials exist in different forms, and have varying amounts of information defined for them. For example they may be of a pre-defined thickness or the material can be any given thickness. When constructing the layers of an object the user may wish to use pre-fabricated or standard sections for part of the object, or even start with a pre-defined standard object. Hence there are many possible methods for putting the materials together to form the object. This has lead to the development of a form-based materials selection system.

When users select the "New..." option from any of the materials menus of PlanEntry they are presented with forms to describe the composition of the material combination they require. These combinations match the lists of combinations from the ALF manual. Figure 4 shows an example of a wall material specification.

The approach taken in the ThermalDesigner system is as yet by no means ideal, as the example illustrates. However, being written completely in Kea and forms system code, it is readily extensible. This is unlike the prototype WallThermalResistance system developed earlier as part of the ThermalDesigner system. The WallThermalResistance system had a more graphical user interface, but was hard coded in DEC-Windows UIL code, making extensibility a major problem. It is a testament to the flexibility of the new forms system that an adequate replacement to the hard-coded version is able to be constructed with such little loss of usability.



The building location is not one of the standard locations so the user must enter the Infiltration Zone and Degree Day values by hand.

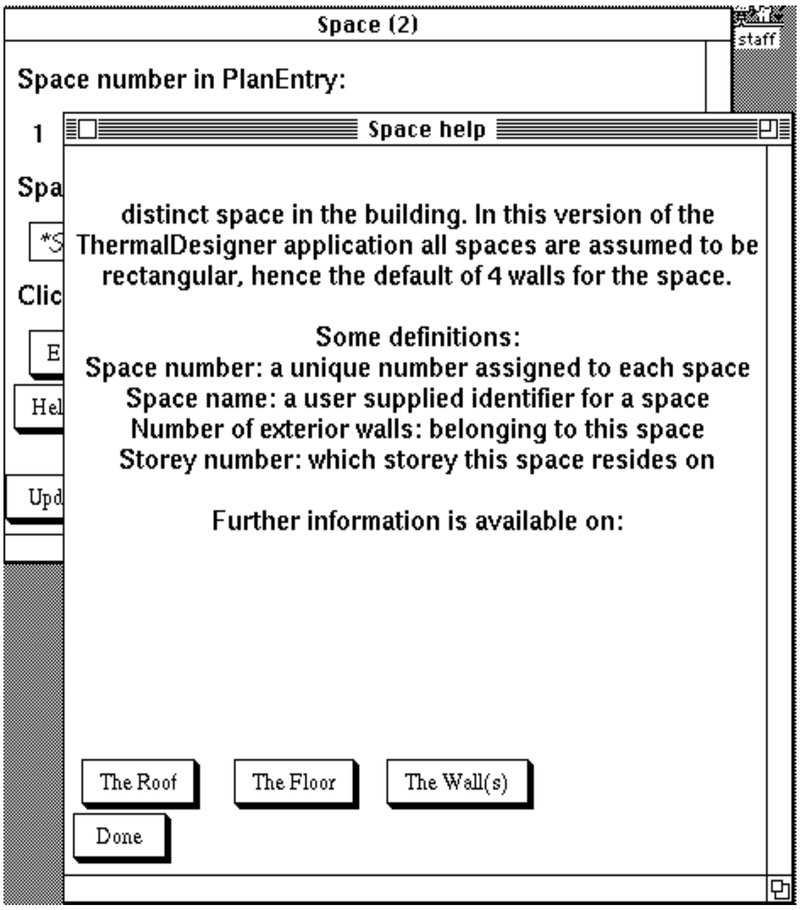
The required information can be found in the data sheets on pages 17 through 19 of the ALF manual.

Fugure 5: Example help form

2.2.2 Form based help

To help guide the user through the ThermalDesigner system, a hypercard like help facility is available from many of the forms. Clicking on a 'Help' button showns the user a message pertaining to the form they are attempting to fill in. For example, in Figure 5, when the user enters a location of *Other* in the main climate form, a secondary form is generated asking for additional information. If the user presses the help button on this secondary form the help explanation is provided.

The help forms explain what information is required from the user. If applicable, they include the relevant section of the code being examined at that point. In most cases, the help form provides a reference to where the information required can be found, either in the ALF manual or in standards documents which describe the constraints on the information required.



distinct space in the building. In this version of the ThermalDesigner application all spaces are assumed to be rectangular, hence the default of 4 walls for the space.

Some definitions:
Space number: a unique number assigned to each space
Space name: a user supplied identifier for a space
Number of exterior walls: belonging to this space
Storey number: which storey this space resides on

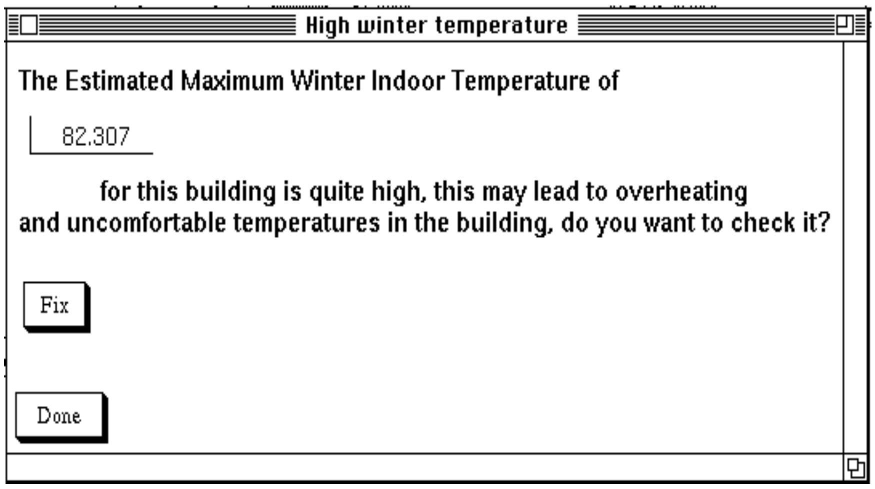
Further information is available on:

Figure 6: Space help form

A help form for a space is shown in Figure 6. This demonstrates the hypercard like aspects of the help system. As well as the information on the space the user can jump to help screens for the roof, floor and wall objects. While this example shows a forward chain of questioning it is entirely possible to have links between any help form in the application in a similar manner to a hypercard stack.

2.2.3 Feedback to the user

ThermalDesigner incorporates several procedures to provide feedback to the user on the performance of the building. Most of this feedback is in the form of warning and help screens that will appear on the screen when a particular condition exists, or has been created, in the building.

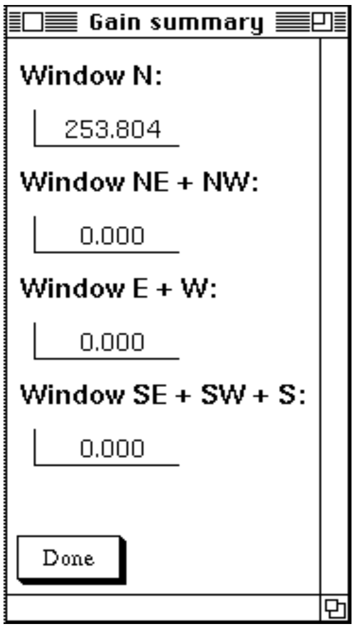

High winter temperature

The Estimated Maximum Winter Indoor Temperature of

82.307

for this building is quite high, this may lead to overheating and uncomfortable temperatures in the building, do you want to check it?

Figure 7: High winter temperature warning form


Gain summary

Window N:

253.804

Window NE + NW:

0.000

Window E + W:

0.000

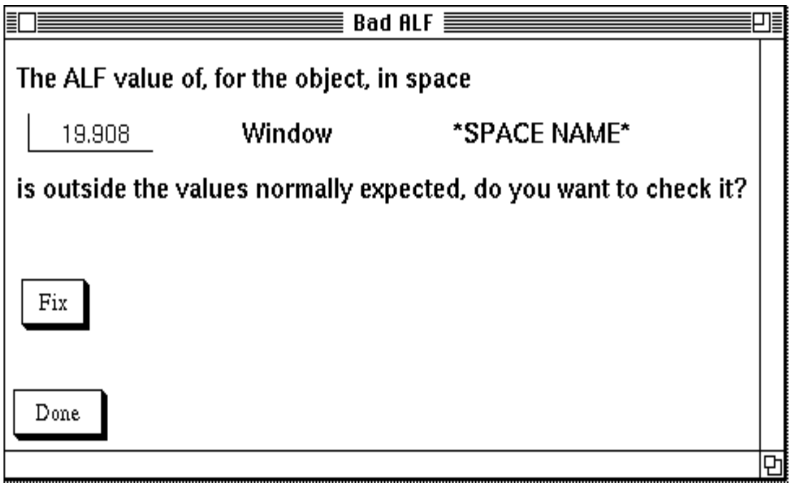
Window SE + SW + S:

0.000

Figure 8: Heat gains summary form

For example one of the values calculated in the ALF worksheet is an estimated maximum winter indoor temperature. This value provides an indication of overheating inside the building. If it is higher than some threshold value, a warning form is generated, as shown in Fig. 7.

Feedback is also provided via forms providing summary information as requested. For example, in Figure 8, the heat gains for the building are displayed on a form when requested from the gain summary button on the building form (Fig. 3).



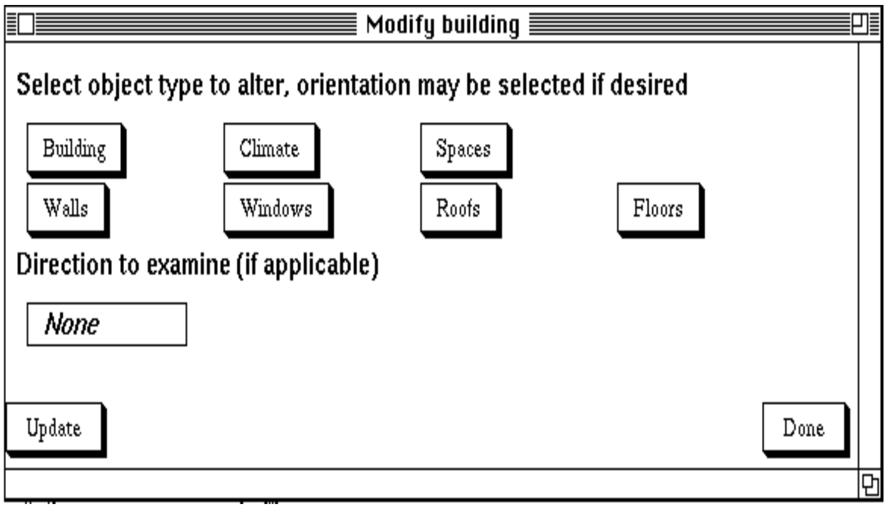
Bad ALF

The ALF value of, for the object, in space

19.908 Window *SPACE NAME*

is outside the values normally expected, do you want to check it?

Figure 9: ALF value out of range form



Modify building

Select object type to alter, orientation may be selected if desired

Direction to examine (if applicable)

None

Figure 10: Building modification form

2.2.4 Modifying and improving a design

As the design may not meet the code requirements, or the designer wishes to examine alternatives to the initial design, there are procedures for allowing changes to the design. For example Figure 9 shows the form which is activated when the current design does not meet the thermal requirements of ALF (ie. BPI not in appropriate range). If at this point the user wishes to modify the design to try and change the BPI for their design then they click on the 'Fix' button. Clicking on the 'Fix' button invokes the form in Figure 10.

The latter form works in a very simple manner. The user may select any object in the design, and its forms will be displayed on the screen, allowing the user to make modifications to the object, and to follow the links on the object forms to the objects associated parts. The user may wish to modify only objects with a certain orientation, for example walls facing north, in which case a direction is specified before selecting the 'Walls' button on the form. Then only walls orientated north will be re-displayed for modification by the designer.

2.3 Limitations

Some assumptions and limitations have been made in the implementation of ThermalDesigner. These are listed below for completeness.

- Spaces are restricted to being of a rectangular shape laid down in an orthogonal matrix.
- All spaces are assumed to be box-like, that is, all walls are of the same height and there can be no sloping roofs.
- All spaces on a single storey are assumed to be of the same height.
- If the user wants to enter a material combination that is not in listed in the ALF manual then they may only give its name and its R-value. It is not possible to list the constituent materials of the object.

3. TECHNICAL ASPECTS

3.1 Class structures

Figure 11 shows a simplified class structure diagram of the Kea portion of ThermalDesigner. The major simplification of Fig. 11 is that most of the classes shown also inherit from one or more external classes. These external classes are of two types: those corresponding to PlanEntry objects, and those corresponding to supplementary forms. A more detailed, but still incomplete, class structure diagram is shown in Appendix 1.

The role of the various classes is fairly obvious from their names. The exception might be the Hole class, which is used to represent portions of an object which are absent, e.g. holes in walls caused by insertion of windows.

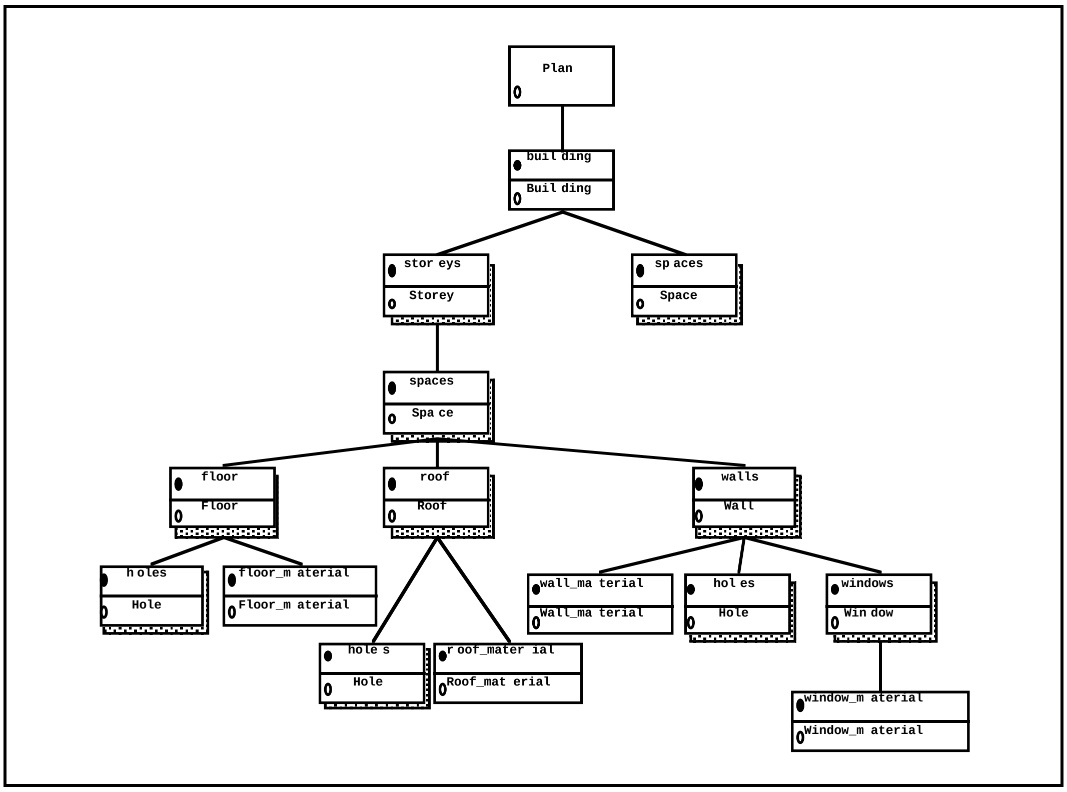


Figure 11: ThermalDesign major class structure

3.2 PlanEntry

The PlanEntry system passes information to a Kea program through the external interface. The fixed set of object types available in PlanEntry translates to a fixed set of external classes available to a Kea program. As the user creates a building by laying spaces and fixing in windows, materials can be defined separately for many of the component objects of the space. The object types available to the user in PlanEntry are:

Space: Added or modified by the user. The creation of a space invokes the creation of the constituent walls, roof and floor bounding the space. These constituent objects inherit the default material types for the objects if specified.

Storey: Added or deleted by the user through the pop down menus, all spaces are associated with a particular storey.

Wall: Created during the addition of the space, a user may change the material type of the wall at any stage.

Floor: Created during the addition of the space, a user may change the material type of the floor at any stage.

Roof: Created during the addition of the space, a user may change the material type of the roof at any stage.

Window: Added or modified by the user. The window must be specified in an exterior wall of the building.

Information about these objects is available via the following external classes:

ExtPlan: Defines the materials available for use in the wall, roof, floor and window objects. This also gives the reference to the building object.

ExtBuilding: Provides a list of the storeys comprising the building and a bounding box for the building.

ExtStorey: Provides a list of the spaces comprising the storey and a bounding box for the storey.

ExtSpace: Provides lists of wall, roof and floor segments comprising the space and a bounding box for the space.

ExtWall: Provides lists of window and separator segments inside the wall, the orientation of the wall in the space, the material type of the wall, and the dimensions of the wall.

ExtFloor: Provides a list of separator segments in the floor, the material type of the floor, and the dimensions of the floor.

ExtRoof: Provides a list of separator segments in the roof, the material type of the roof, and the dimensions of the roof.

ExtWindow: Provides the material type of the window, and the location and dimensions of the window.

ExtSeparator: Provides the location of the hole segment defined by the separator. This is used to denote any segment of a wall, roof or floor that is not exterior to the building. This includes interior walls, ceilings and floors as well as windows in walls.

ExtMenu: Provides access to one of the pull-down material menus in PlanEntry for the addition of items.

ExtMenuEntry: Provides a way of passing a new material name back to PlanEntry to incorporate into the specified pull-down menu.

Two of the external classes are shown as examples below in Fig. 12. These define the top two levels in the hierarchy accessible from PlanEntry: the plan and building classes.

```
external  class  ExtPlan.
    input buildingRef    : integer.
    input wallMenuRef    : integer.
    input roofMenuRef    : integer.
    input floorMenuRef   : integer.
    input windowMenuRef  : integer.
end ExtPlan.

external  class  ExtBuilding.
    determinant  parameter buildingId    : integer.
    input storeyRefs     : list integer.
    input min_x          : float.
    input max_x          : float.
    input min_y          : float.
    input max_y          : float.
    input min_z          : float.
    input max_z          : float.
end ExtBuilding.
```

Figure 12: Example external classes for PlanEntry

```
class Plan inherits ExtPlan.
 public building : Building := new
            Building(buildingId := buildingRef).
 public WallMenuRef : integer := wallMenuRef.
 public wall_menu : Wall_menu := new
            (menuId := WallMenuRef).
 do determine(wall_menu) done.
end Plan.

class Storey inherits ExtStorey.
 public spaces : list Space
 := collect(space in spaceRefs, new
            Space(spaceId := space)).
end Storey.

class Building inherits ExtBuilding, Building_form.
 public storeys : list Storey
 := collect(r in storeyRefs, new Storey(storeyId := r)).
 public spaces : list Space
 := flatten(collect(st in storeys, st^spaces)).
 private floor_area_total : float
 := sum(flatten(collect(f in spaces,
            collect(r in f^floor, r^area)))).
 ...
end Building.
```

Figure 13: ThermalDesigner code accessing external classes

The code in Fig. 13 shows how the connections from the PlanEntry objects to the corresponding Kea objects are dynamically constructed. The code demonstrates several interesting aspects of using the external interface with a Kea program.

Associating an external object with a Kea class is done through the use of references and identifiers passed through the external classes. To associate the external building with a Kea object the code in the Plan class creates a new Building object with the *buildingRef* from the external Plan as the *buildingId* parameter for the new *building* object. This *buildingId* is used as a determinant to associate the new building with the correct external building which has this *buildingId* as a key.

The same process is followed when there are a list of external objects to be linked to. This occurs in the external building class where a list of *storeyRefs* is input. The association between an external storey and the corresponding Kea storey is made by creating a new Kea storey and passing the *storeyRef* through as the *storeyId* determinant for the external storey object. The only difference in this case is that the building class contains a list of storeys, rather than a reference to a single object.

The association of an external material menu with the Kea application is handled in the Plan class. In this example the *wallMenuRef* from the external plan is associated with the creation of a new *wall_menu* and used as the determinant for the external menu for the *wall_menu*.

The code in the building class also illustrates how it is possible to amalgamate information from the external class hierarchy without having to duplicate the whole structure used by the external interface. In a building object, a list of space objects is created rather than having to access the spaces by storey, as one would by following the hierarchy of external classes. To create this list of all spaces in the building, the spaces for each storey are collected and then flattened into one list accessible in the building class.

3.3 Materials selection

The material selection in PlanEntry is a demonstration of the interaction possible between the PlanEntry system and the Forms system through the external interface. In the previous section the method by which a menu of materials from PlanEntry is associated with a list of materials in a Kea application is discussed. In this section we examine the interaction between PlanEntry and the Forms system as materials are created and accessed.

The external interface to PlanEntry menus and their entries are shown in Fig. 14. The *menuId* determinant in *ExtMenu* is provided from the *Plan* object (discussed in the previous section). The *numInitialEntries* determinant provides a mechanism (currently unimplemented) for specifying some number of initial entries to be created for this menu. The *entryListRef* input provides a reference to this menu for all entries in the menu. The *firstEntryRef* input is a reference to the first menu entry in the menu.

The *entryListId* determinant in ExtMenuuEntry specifies which menu the entry belongs to, and corresponds to the *entryListRef* input in ExtMenu. The *entryId* determinant ia a key to the external object corresponding to the menu entry. For the first entry in a menu this is provided from the *firstEntryRef* input. For subsequent entries the *nextEntryRef* input of the previous entry provides the appropriate reference. The *entryName* determinant provides a name for the entry. This is the name that appears in the pull down menu.

```
external class ExtMenu.
    determinant parameter menuId  : integer.
    determinant numInitialEntries  : integer:= 0.
    input entryListRef             : integer.
    input firstEntryRef            : integer.
end ExtMenu.

external class ExtMenuEntry.
    determinant parameter entryListId : integer.
    determinant parameter entryId : integer.
    determinant entryName          : text
                     := '*new menu entry*'.
    input nextEntryRef             : integer.
end ExtMenuEntry.
```

Figure 14: External classes for PlanEntry menus

```
class Wall_menu inherits ExtMenu.
 firstEntry : Wall_material
 == new Wall_material(   menuName := MenuName,
                         entryListId := entryListRef,
                         entryId := firstEntryRef).
 public materials : list Wall_material
 ==     [] if firstEntryRef = -1
        | [firstEntry] + firstEntry^materials.
 do determine(materials) done.
end Wall_menu.

class Wall_material
 inherits Wall_mat_form, ExtMenuEntry.
 parameter menuName : text.
 public materials : list Wall_material
 ==     [] if nextEntryRef = -1
        | [nextEntry] + nextEntry^materials.
 nextEntry : Wall_material
 == new Wall_material(   menuName := menuName,
                         entryListId := entryListId,
                         entryId := nextEntryRef).
 % determined from the wall_material_form
 public classifier type_of_wall : WallClassType
 == wall_type.
 entryName : text := wall_name.
end Wall_material.
```

Figure 15: Kea code interacting with PlanEntry menus

Figure 15 shows the interaction between the Kea wall materials selection classes and the external menu classes. The dynamics of the interaction are as follows:

- A *Wall_menu* object is created with a *menuId* supplied by the *Plan* object. Association with the corresponding external wall menu object is made.

- The unconditional do statement in *Wall_menu* forces evaluation of the *materials* list. Evaluation blocks on the PlanEntry input *firstEntryRef*.
- The first "New..." invoked by the user on the PlanEntry wall material menu causes PlanEntry to construct a new external menu entry object, and to supply a reference to this to the *firstEntryRef* input of the *Wall_menu* object.
- Evaluation of the *materials* list now unblocks. The *firstEntry Wall_material* object is created, and supplied *entryListRef* and *firstEntryRef* as parameters.
- Before attaching to the external menu entry object, the *Wall_material* object must have a value for the *entryName* determinant. This is obtained from the *Wall_mat_form* form associated with the *Wall_material* object. This latter form also allows the user to specify the wall material, in conjunction with subsidiary forms obtained via the *type_of_wall* classifier. Figure 4 provides an example of the resulting forms seen by the user.
- The *Wall_material* object now makes the association with the external menu entry object. The *entryName* supplied as a determinant appear as the new entry's name in the wall material menu.
- Subsequent "New..." invocations cause values to be supplied to the current last menu entry's *nextEntryRef* input. This cause evaluation of the materials list to continue by creation of a new *Wall_material* object.
- Commital of the plan causes a -1 value to be supplied for the current last menu entry's *nextEntryRef* input, terminating the construction of the *materials* list.
- Whenever a PlanEntry wall is constructed with a material specified by one of the material menu entries, the *entryId* of the entry is associated with that wall.
- On plan commital, external wall objects provide the associated material id as an input to the coresponding Kea wall object.
- The Kea wall object can then locate and make use of the appropriate Kea *Wall_material* object by selecting the object with the appropriate *entryId* from the wall materials list in the *Plan* object.

The other materials menus have a similar method of interaction. Figure 16 shows a more detailed class structure diagram of the menu classes.

3.4 Kea and Forms

The previous section has illustrated one way in which form interaction is used by the ThermalDesigner system. In this section we describe how some of the other types of form interaction discussed in section 2.2 are implemented.

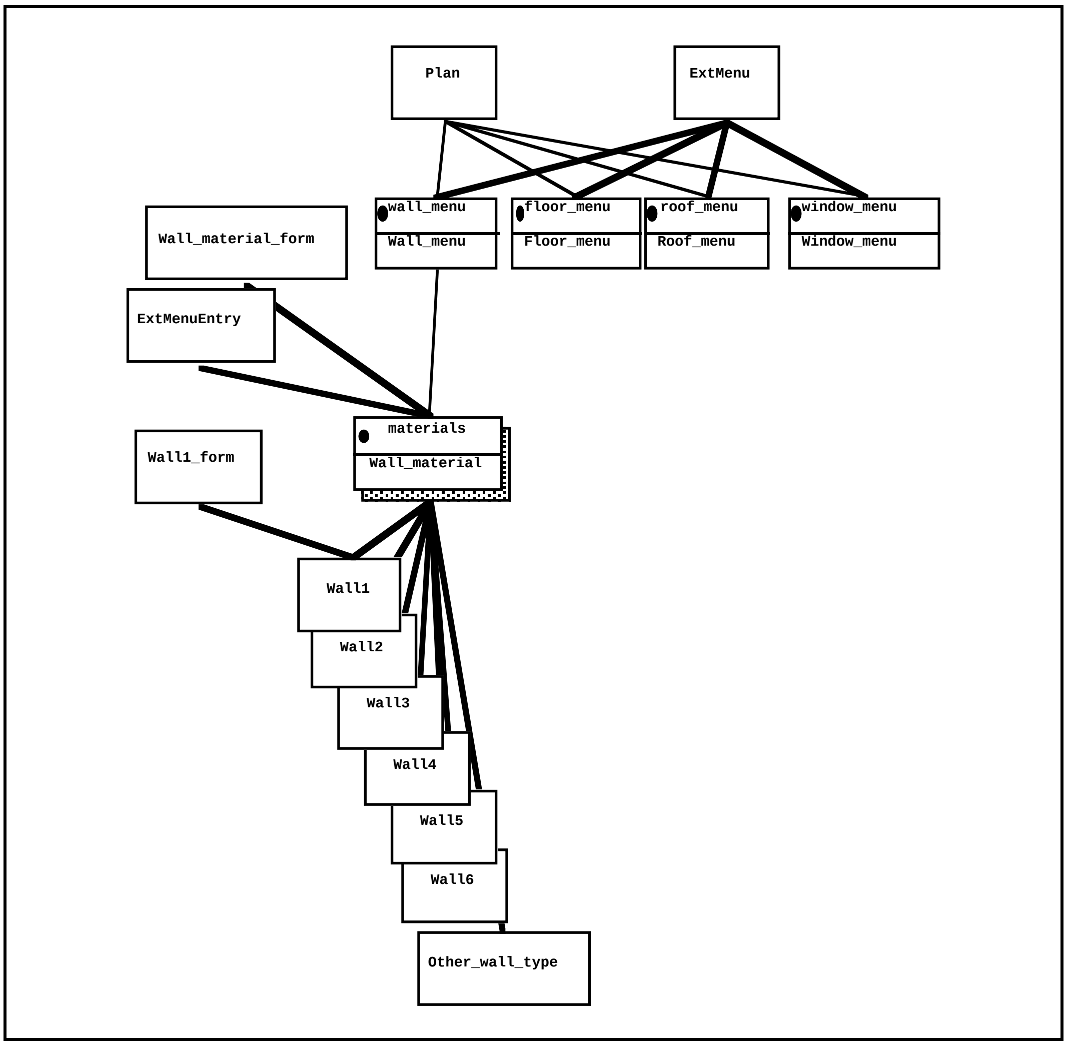


Figure 16: Class structure diagram for menu classes

3.4.1 Help forms

Help forms are brought up by having a procedure associated with the help button which constructs the help form. The space help form of Fig. 6 is displayed by the code of Figure 17, which is included in the *Space* class.

```
...
public space_help: procedure :=
      determine(new Space_form_help).
...
```

Figure 17: Help form invocation

3.4.2 User feedback

Warning forms, such as the one in Fig 7, are displayed autaomatically by using a *when* procedure. For example, Fig. 18 shows how the high winter temperature warning form of Fig 7 is generated.

```
class Building inherits Building_form.
 .....
 when est_max_winter_indoor_temp > 40.0 do
   determine(new Hot_winter
        (temp := est_max_winter_indoor_temp)).
 done.
 .....
end Building.

class Hot_winter inherits Hot_winter_warning.
 parameter temp : float.
 public temp_f : float := temp.
.....
end Hot_winter.

form Hot_winter_warning "High winter temps." @ (440, 40)
 "The Estimated Maximum Winter Indoor Temperature of" @ (5, 5)
 determinant temp : float(10, 3) @ (14, 20)
 "for this building is quite high, this may lead to overheating"
 "and uncomfortable temperatures in the building, do you want to
 check it?" @ (5, 40)
 fix_problem : button("Fix") @ (10, 70)
 done @ (5, 105)
end
```

Figure 18: Warning form implementation

More complicated expressions can be used in when procedures to check for events which are the culmination of results from several different objects. For example, in Figure 19, a when procedure from the space class is triggered when the difference between the largest R-value of a wall in a space and the smallest R-value of a wall in the same space is greater than 1.2, this is an indication that condensation could accumulate on one of the walls in the space.

```
when (max(collect(w in walls, w^r_value)) -
          min(collect(w in walls, w^r_value))) > 1.2 do
 determine(new Bad_wall_ratio(space_name := name)).
done.
```

Figure 19: Complicated when procedure

3.4.3 Modifying designs

Figure 10 shows the building modification form, which allows users to select appropriate objects for modification. Figure 20 shows the implementation behind this form illustrating the mechanism for selecting objects. The operation simply amounts to displaying each of the required forms using the show procedure. In the case of windows, however, a new type of form is constructed as there is no corresponding existing form for the user to examine.

```
class Fix_form inherits Fix_problem_form.
 public show_building : procedure :=
   plan^building^show.
 public show_space : procedure :=
   foreach(s in plan^building^spaces, s^show).
 public show_wall : procedure :=
   foreach(s in plan^building^spaces,
       foreach(w in s^walls,
               w^show if direction = None or w^orientation =
                                  good_orientation)).
 public show_window : procedure :=
   foreach(s in plan^building^spaces,
      foreach(w in s^walls,
         foreach(window in w^windows,
                 determine(new window_info_form(
                           space_name := window^space_name,
                           facing := window^facing,
                           win_type := window^win_type))
                 if direction = None or w^orientation =
                                        good_orientation))).
 public show_roof : procedure :=
   foreach(s in plan^building^spaces,
       foreach(r in s^roof, r^show)).
 public show_floor : procedure :=
   foreach(s in plan^building^spaces,
       foreach(f in s^floor, f^show)).
 public show_climate : procedure := climate^show.
end Fix_form.
```

Figure 20: Building modification form implementation

4 FUTURE WORK

Construction of ThermalDesigner has proven the worth of the graphical extensions to the Kea system. It has also shown that the two layer architecture of Kea is extensible to cope with the degree of fine scale modifications that result from a highly interactive graphical environment.

As befits a resaerch prototype using new facilities, there are many opportunities for further work. In this section we examine some improvements and extensions that could usefully be made to ThermalDesigner.

4.1 Merging forms

The ability to link together associated forms to form one complete form would be useful. A particular example is where the results of filling out a form require further information gathered through the use of an extra form. Conceptually it would be useful to have these forms merge into a single form defining all the information required in the analysis of the object.

4.2 Identifying PlanEntry objects

There are some difficulties experienced in the current process of interaction between PlanEntry and a Kea application. The major one is the lack of feedback from the Kea side to PlanEntry. In the current version it is not possible to feed information back to PlanEntry. This means that when filling in forms about objects in PlanEntry there is no contextual feedback about the object being examined. To partially overcome this problem all spaces in a PlanEntry building are numbered as created and this number allows the user some way of linking the space being discussed in Kea with a space in PlanEntry. However, it would be better to have some method of highlighting the PlanEntry object from Kea, giving better feedback to the user about what information is required.

This lack of feedback also means that no changes made on the Kea side can be applied on the PlanEntry side, forcing the programmer to carefully construct their application to ensure that no inconsistencies can be introduced between the PlanEntry model and the model in the application.

4.3 Difficulties in defining materials with Kea

Trying to define a table of materials in Kea and the FormManager highlighted several problems that are difficult to solve or to avoid in Kea. They are:

- Tables do not exist in the current FormManager. The only way of mimicking this at present is to have all elements in a table represented as individual forms, positioned to emulate the look of a table.

- In FormManager, the only method available to allow the user to select from a fixed range of options is to define an enumerated type of all the possibilities. However, when making a selection of categories to examine, the user has narrowed down the possible candidates to select between, requiring some form of dynamic enumerated type to be defined to allow the user to select the desired material. This is not currently possible in Kea.

- It is hard to represent the information required about each material for easy selection in a standard relational database. The attributes needed to represent materials varies considerably between different materials, making it hard to define sensible relations covering more than a few materials at a time.

4.4 Use of database

The new Kea database interface could be used to allow external storage of materials and climate information, able to be updated independently of the application. In the case of materials, however, the caveats of the previous section must be taken into account.

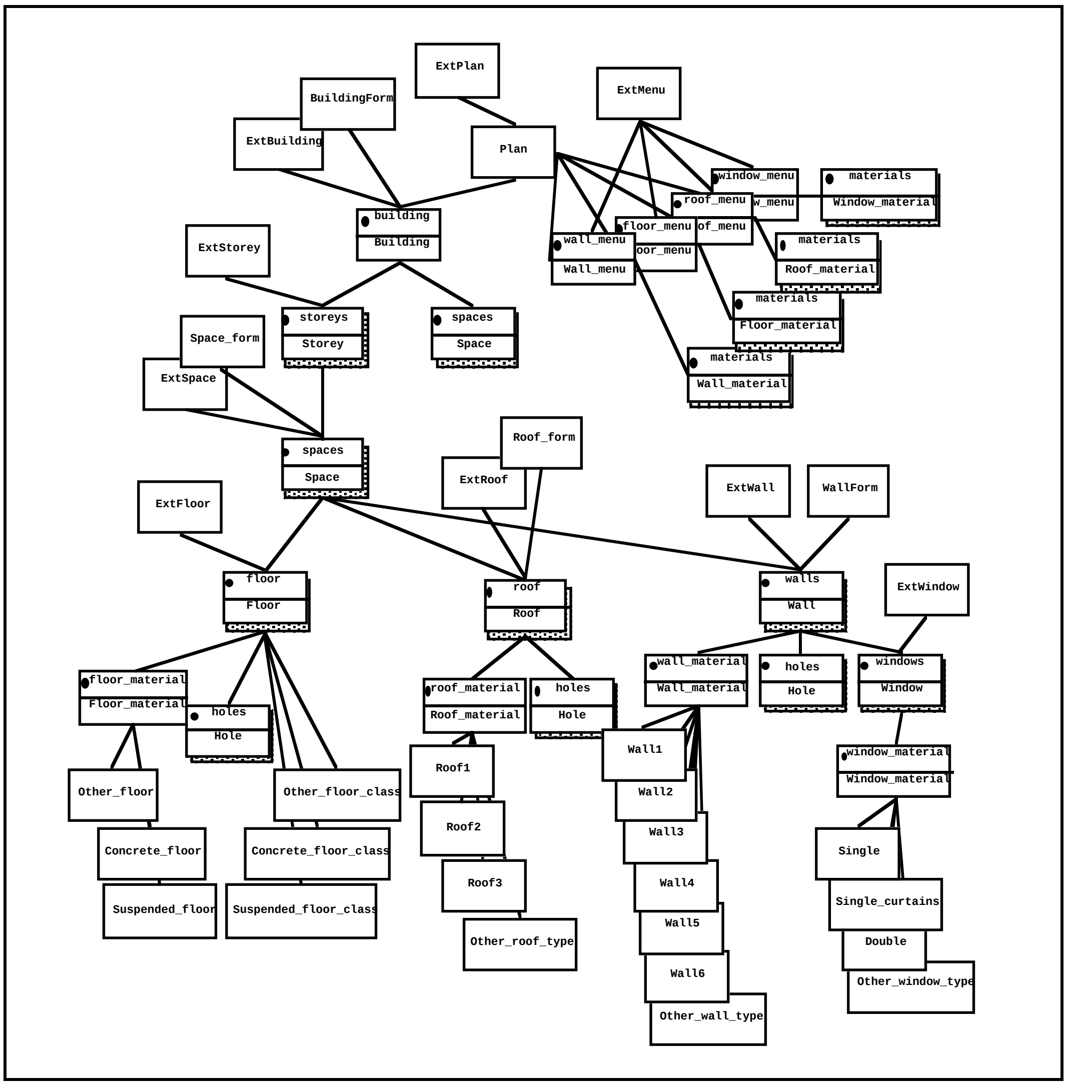
4.5 Non-rectangular spaces

The PlanEntry system is currently limited to rectangular spaces. This is undesirable, and it would be useful to extend the system to cope with non-rectangular geometry as well. The Kea component can be readily adapted to cope with this, but modifying PlanEntry to cope would be a major undertaking.

4.6 Analysis of results

The current display of results leaves much to be desired. Histograms or charts summarising information could be useful, and would provide a challenging extension to the forms system.

APPENDIX 1 EXTENDED CLASS STRUCTURE DIAGRAM



APPENDIX 2 FORMULAE USED IN THERMALDESIGNER

The formulae derived for the graphs in the ALF design manual were calculated by taking points on the graphs and working out equations through the use of Cricket Graph on the Macintosh computers.

ALF graphs (y = c + ax + bx2)

Graph type	c	a	b	R2
Floors	3.5343	4.5608x10-2	-1.1001x10-5	1.000
Air leakage	-3.3678	2.2838x10-2	-6.7557x10-6	0.993
Wall light S	-2.6186	6.0696x10-2	-1.8240x10-5	1.000
Wall light E,W	-3.9841	6.0934x10-2	-1.8473x10-5	0.999
Wall light N	-5.9841	6.0934x10-2	-1.8473x10-5	0.999
Wall dark S	-7.0047	6.3322x10-2	-1.8765x10-5	1.000
Wall dark E,W	-11.529	6.7010x10-2	-2.0688x10-5	1.000
Wall dark NE,NW	-13.005	6.3322x10-2	-1.8765x10-5	1.000
Wall dark N	-16.875	6.5122x10-2	-1.9639x10-5	1.000
Roof light	-2.838	6.3549x10-2	-1.9522x10-5	0.999
Roof dark	-9.7212	6.3364x10-2	-1.9687x10-5	1.000
Window	12.0	0.02143	0.0	

AGF graphs (y= c + ax + bx2)

Window N	429.41	-0.24122	4.1833x10-5	1.000
Window NW,NW	354.59	-0.23459	5.8192x10-5	0.999
Window E,W	221.18	-0.158	4.1833x10-5	0.996
Window S	99.468	-7.1612x10-2	2.4488x10-5	0.996

Suspended floor R-values (y=b + ax)

Wind protection	b	a
A	0.01	0.0
B	0.0	0.05714
C	0.0	0.1

Concrete floor R-values

0.4 + 0.2 * insulation_depth + ratio * (0.464 + 0.052 * insulation_depth)

Air leakage ratio

Initial stage

15 * ratio if zone A 13.333 * ratio if zone B

11.8181 * ratio if zone C10.43478 * ratio if zone D

Final stage

0.09091 * initial_stage if exposed

0.07857 * initial_stage if medium exposed

0.06666 * initial_stage if medium sheltered

0.05714 * initial_stage if sheltered

Internal heat gain

900.0 + 150.0 * #_of_occupants

Overheating

20.0 + 142.857 * (1.0 - 0.38333 * effective_thermal_mass) * ratio

Utilisability graphs (y=c + ax + bx2 + dx3)

M	c	a	b	d	R2
0.0	1.0787	-0.42465	5.1865x10-2	4.3706x10-3	0.999
0.3	1.1022	-0.38544	7.5756x10-3	1.4731x10-2	1.000
0.6	1.118	-0.36126	-5.9524x10-3	1.13678x10-2	1.000
1.0	1.0613	-0.12136	-0.21618	6.734x10-2	1.000
1.5	0.97333	0.23106	-0.51705	0.14205	1.000
2.0	0.95071	0.33541	-0.59551	0.15783	1.000

REFERENCES

Bassett, M.R., Bishop, R.C. and van der Werff, I.S. (1990) *ALF MANUAL, Annual Loss Factor Design Manual*, An aid to thermal design of buildings, Building Research Association of New Zealand.

NZS 4218P (1977) *Minimal Thermal Insulation Requirements for Residential Buildings*, Standards Association of New Zealand.

NZS 41214 (1977) *Methods of Determining the Total Thermal Resistances of Parts of Buildings*, Standards Association of New Zealand.