

# **An Interactive Software Paradigm for Data Visualisation**

Derek Law

Department of Statistics

University of Auckland

# Outline

- Motivation for the research

# Outline

- Motivation for the research
- A basic idea underlying a number of interactive software paradigms

# Outline

- Motivation for the research
- A basic idea underlying a number of interactive software paradigms
- Taligent's Model-View-Presenter (MVP)

# Motivation

- A statistical visualisation toolkit was written to ease the process of creating 3D interactive graphics applications with GUI, using OpenGL and a GUI toolkit (e.g. Gtk+)

# Motivation

- A statistical visualisation toolkit was written to ease the process of creating 3D interactive graphics applications with GUI, using OpenGL and a GUI toolkit (e.g. Gtk+)
- In search of an interactive software paradigm to put onto the toolkit that i) allows linking of graphical objects within a plot or between plots ii) ensures the maintainability and reusability of the user created components

# Motivation

- A statistical visualisation toolkit was written to ease the process of creating 3D interactive graphics applications with GUI, using OpenGL and a GUI toolkit (e.g. Gtk+)
- In search of an interactive software paradigm to put onto the toolkit that i) allows linking of graphical objects within a plot or between plots ii) ensures the maintainability and reusability of the user created components
- MVC, and more recently MVP, became the “buzz acronym” these days but they are confusing both at theoretical and practical standpoint

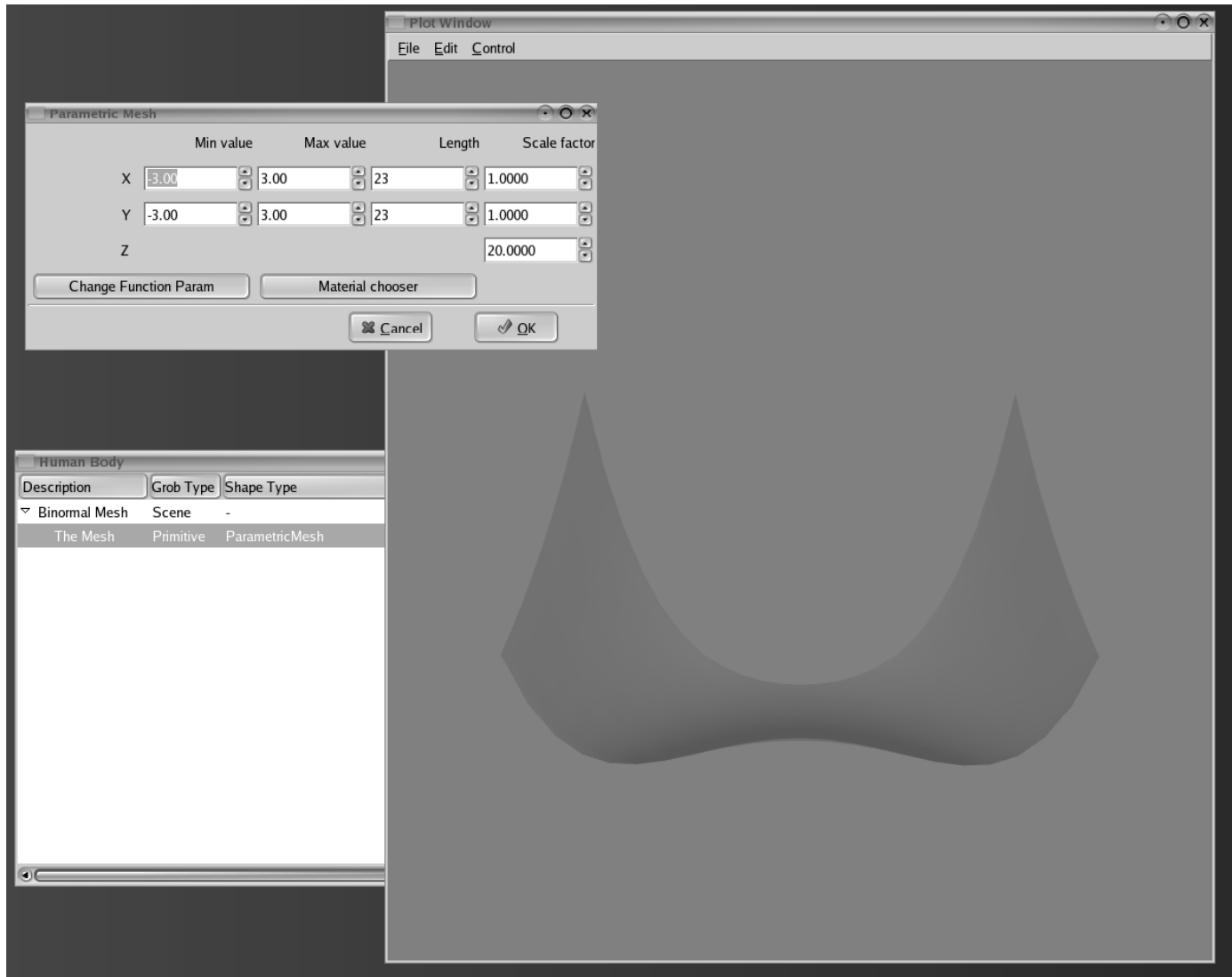
# Motivation

- A statistical visualisation toolkit was written to ease the process of creating 3D interactive graphics applications with GUI, using OpenGL and a GUI toolkit (e.g. Gtk+)
- In search of an interactive software paradigm to put onto the toolkit that i) allows linking of graphical objects within a plot or between plots ii) ensures the maintainability and reusability of the user created components
- MVC, and more recently MVP, became the “buzz acronym” these days but they are confusing both at theoretical and practical standpoint
- A toy example was written to try and study the control flow of the Taligent’s MVP framework and clarify some of the mysteries behind it

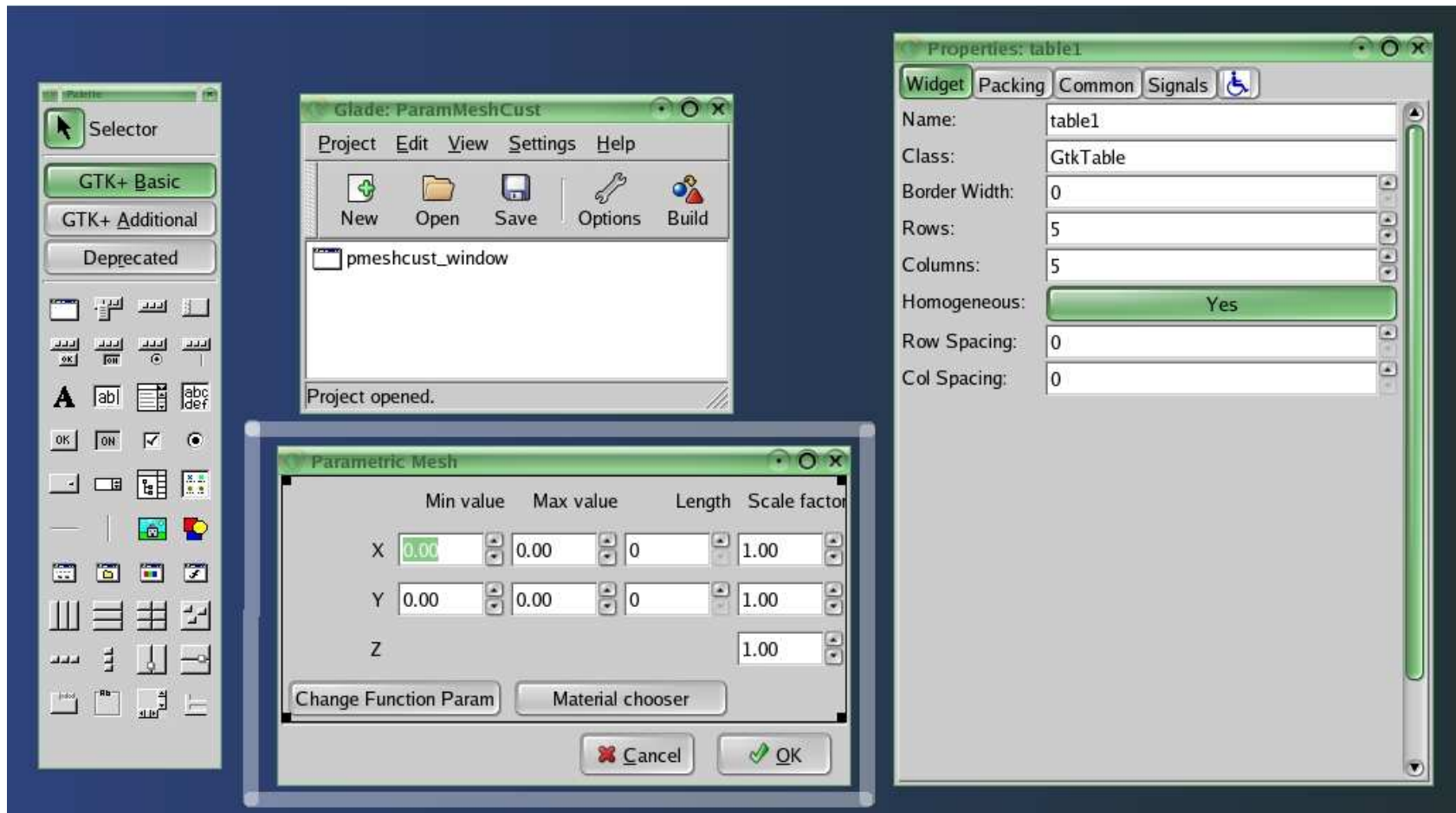


# Widget Based Paradigm

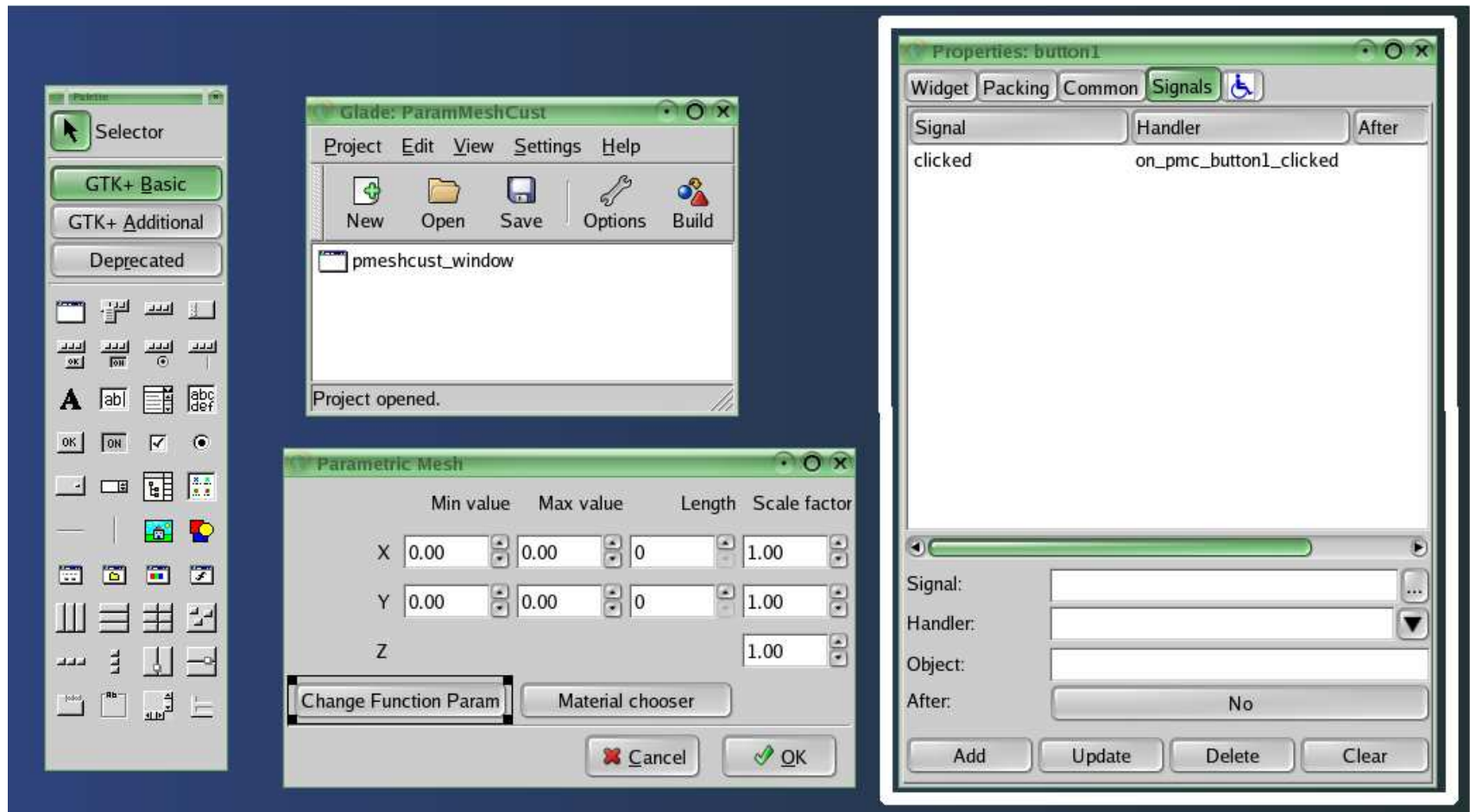
# Widget Based Paradigm



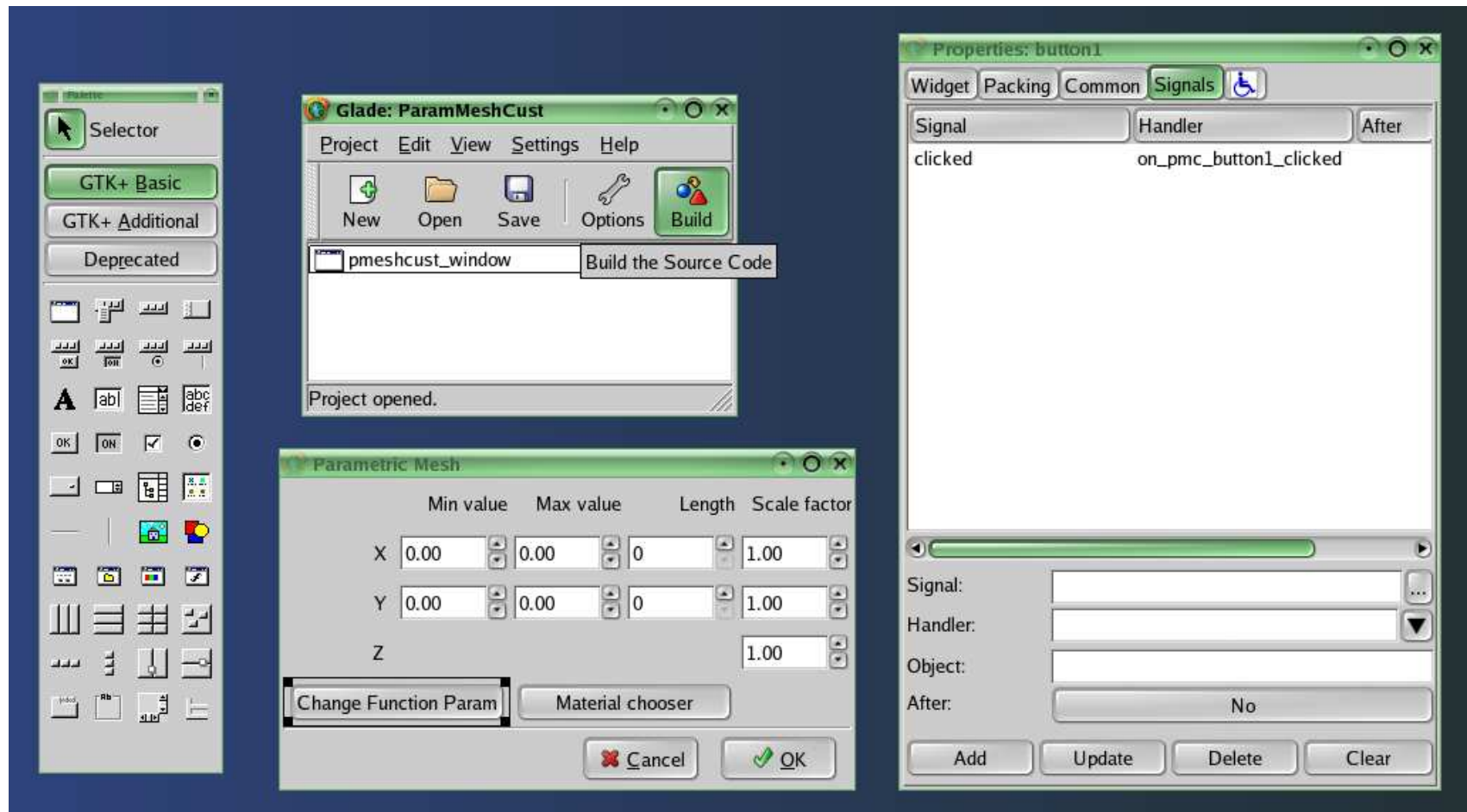
# Widget Based Paradigm



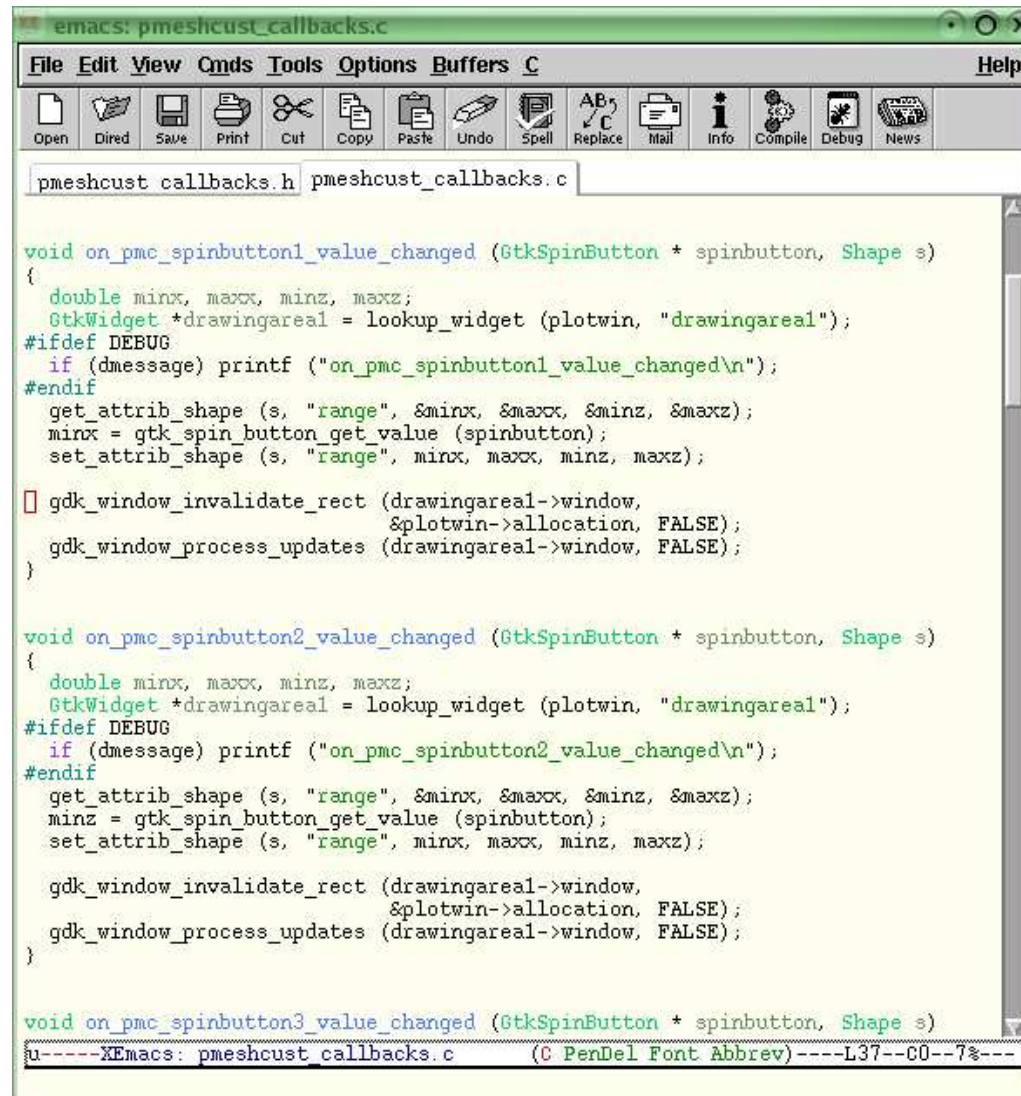
# Widget Based Paradigm



# Widget Based Paradigm



# Widget Based Paradigm



```
emacs: pmeshcust_callbacks.c
File Edit View Cmds Tools Options Buffers C Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail Info Compile Debug News

pmeshcust_callbacks.h pmeshcust_callbacks.c

void on_pmc_spinbutton1_value_changed (GtkSpinButton * spinbutton, Shape s)
{
    double minx, maxx, minz, maxz;
    GtkWidget *drawingareal = lookup_widget (plotwin, "drawingareal");
#ifdef DEBUG
    if (dmessage) printf ("on_pmc_spinbutton1_value_changed\n");
#endif
    get_attrib_shape (s, "range", &minx, &maxx, &minz, &maxz);
    minx = gtk_spin_button_get_value (spinbutton);
    set_attrib_shape (s, "range", minx, maxx, minz, maxz);

    gdk_window_invalidate_rect (drawingareal->window,
                               &plotwin->allocation, FALSE);
    gdk_window_process_updates (drawingareal->window, FALSE);
}

void on_pmc_spinbutton2_value_changed (GtkSpinButton * spinbutton, Shape s)
{
    double minx, maxx, minz, maxz;
    GtkWidget *drawingareal = lookup_widget (plotwin, "drawingareal");
#ifdef DEBUG
    if (dmessage) printf ("on_pmc_spinbutton2_value_changed\n");
#endif
    get_attrib_shape (s, "range", &minx, &maxx, &minz, &maxz);
    minz = gtk_spin_button_get_value (spinbutton);
    set_attrib_shape (s, "range", minx, maxx, minz, maxz);

    gdk_window_invalidate_rect (drawingareal->window,
                               &plotwin->allocation, FALSE);
    gdk_window_process_updates (drawingareal->window, FALSE);
}

void on_pmc_spinbutton3_value_changed (GtkSpinButton * spinbutton, Shape s)
{
    double minx, maxx, minz, maxz;
    GtkWidget *drawingareal = lookup_widget (plotwin, "drawingareal");
#ifdef DEBUG
    if (dmessage) printf ("on_pmc_spinbutton3_value_changed\n");
#endif
    get_attrib_shape (s, "range", &minx, &maxx, &minz, &maxz);
    maxx = gtk_spin_button_get_value (spinbutton);
    set_attrib_shape (s, "range", minx, maxx, minz, maxz);

    gdk_window_invalidate_rect (drawingareal->window,
                               &plotwin->allocation, FALSE);
    gdk_window_process_updates (drawingareal->window, FALSE);
}

u-----XEmacs: pmeshcust_callbacks.c (C PenDel Font Abbrev)----L37--C0--78--
```

# Widget Based Paradigm Problems

# Widget Based Paradigm Problems

Program logic is straightforward to understand, but



# Widget Based Paradigm Problems

Program logic is straightforward to understand, but

- UI components are not fine-grained enough

# Widget Based Paradigm Problems

Program logic is straightforward to understand, but

- UI components are not fine-grained enough
- UI influences our program logic

# Widget Based Paradigm Problems

Program logic is straightforward to understand, but

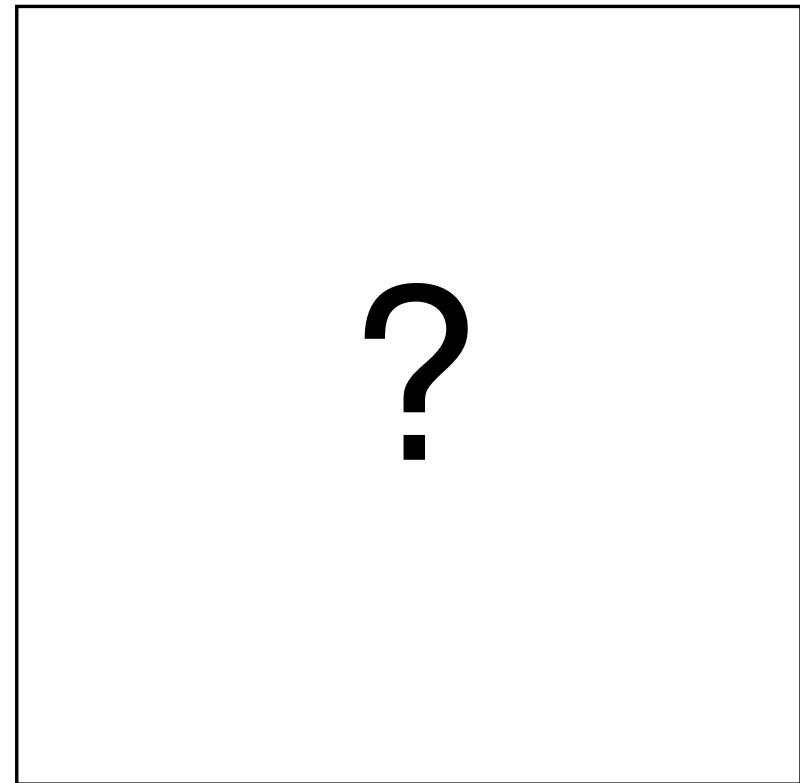
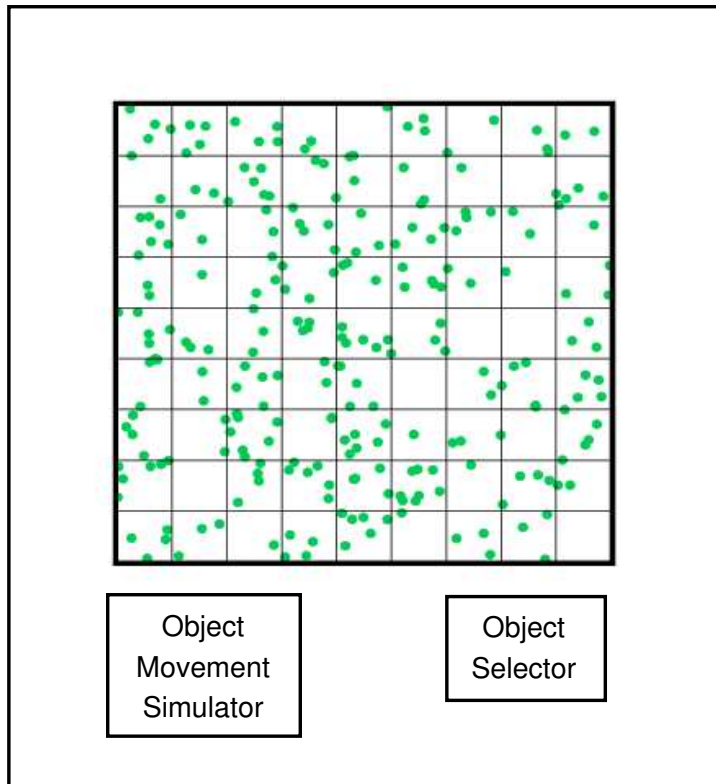
- UI components are not fine-grained enough
- UI influences our program logic
- UI code can easily get tangled up with data representations

# Widget Based Paradigm Problems

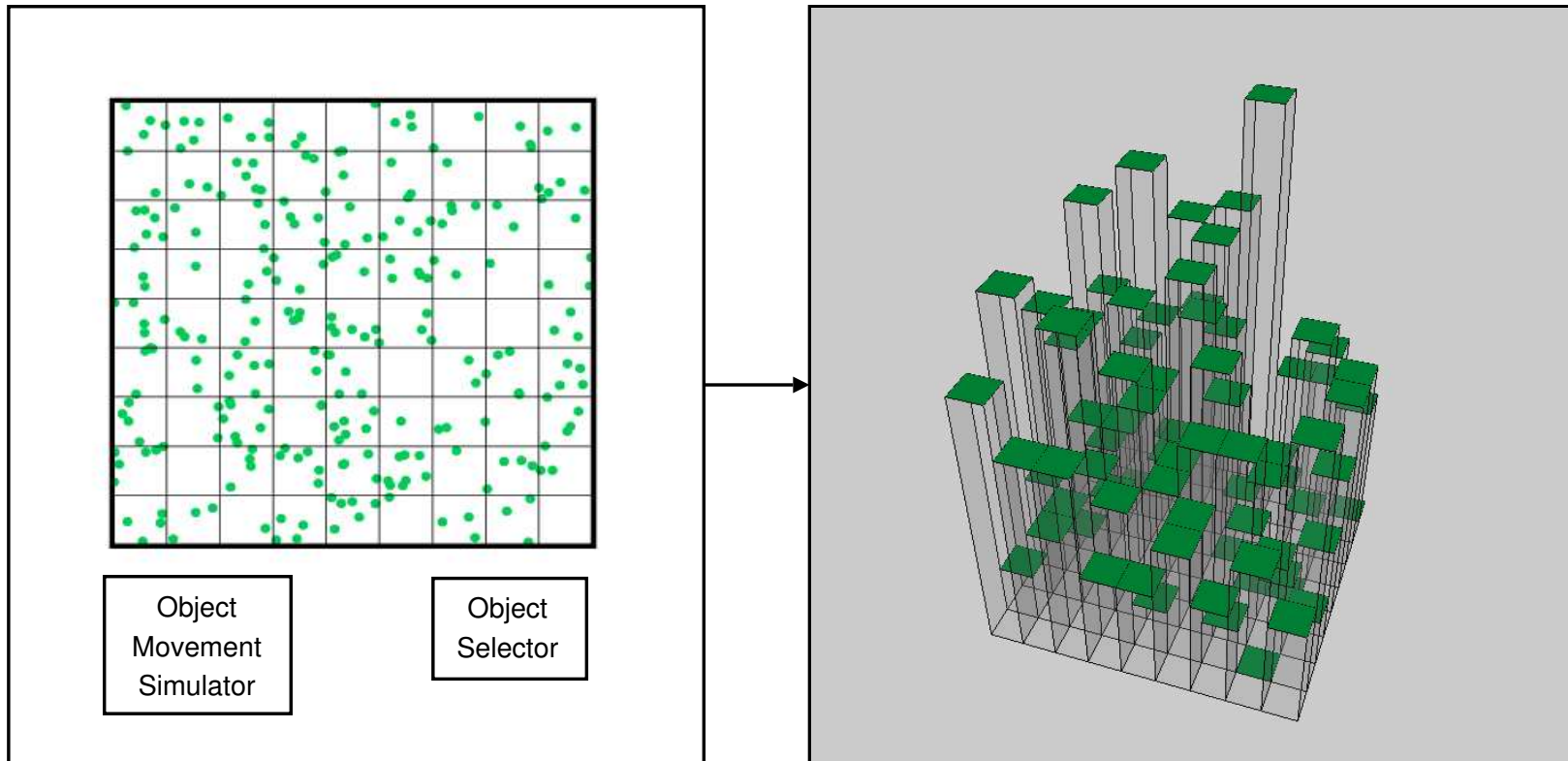
Program logic is straightforward to understand, but

- UI components are not fine-grained enough
- UI influences our program logic
- UI code can easily get tangled up with data representations
- Limited extensibility and reusability

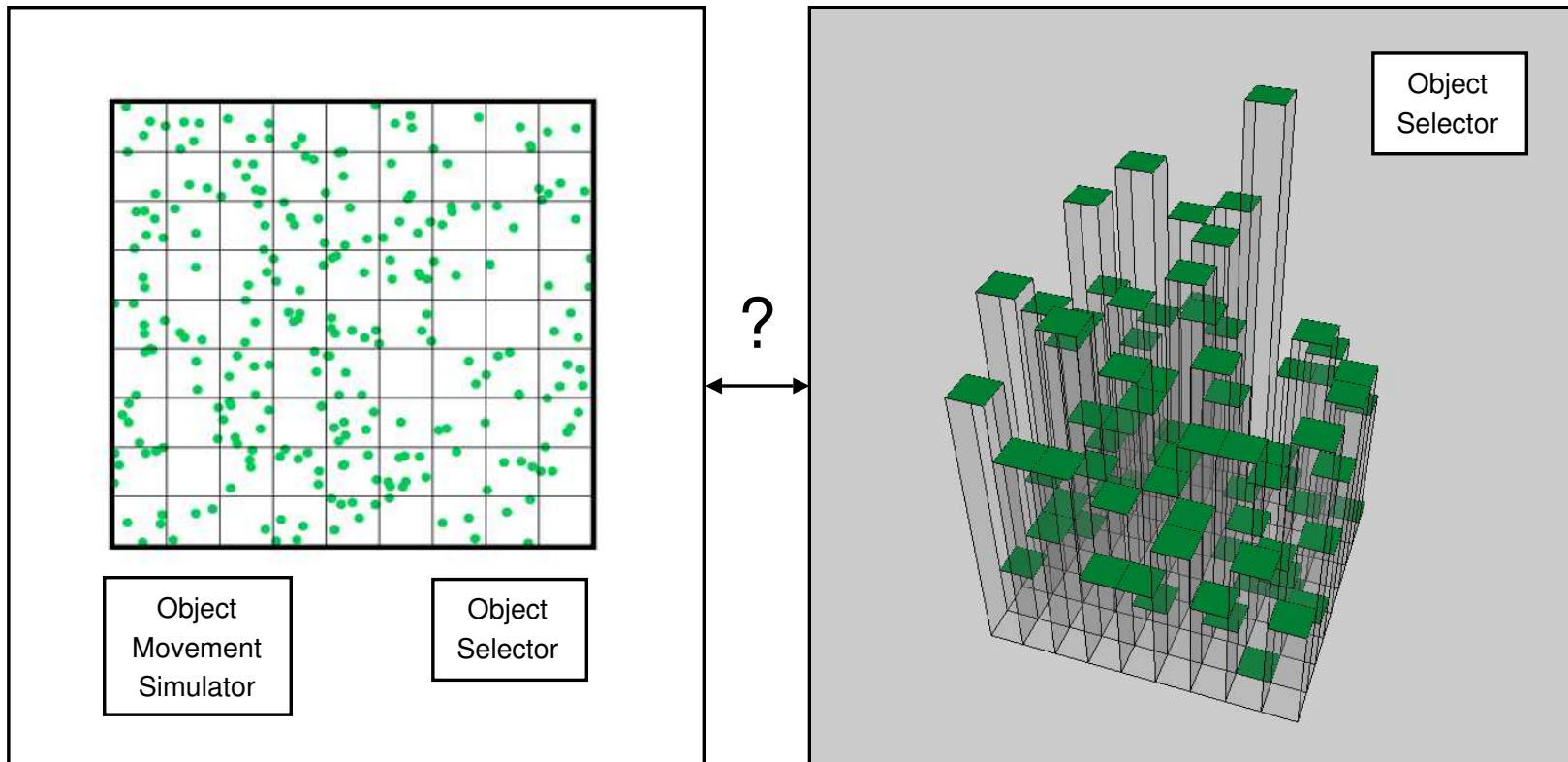
# Widget Based Paradigm Problems



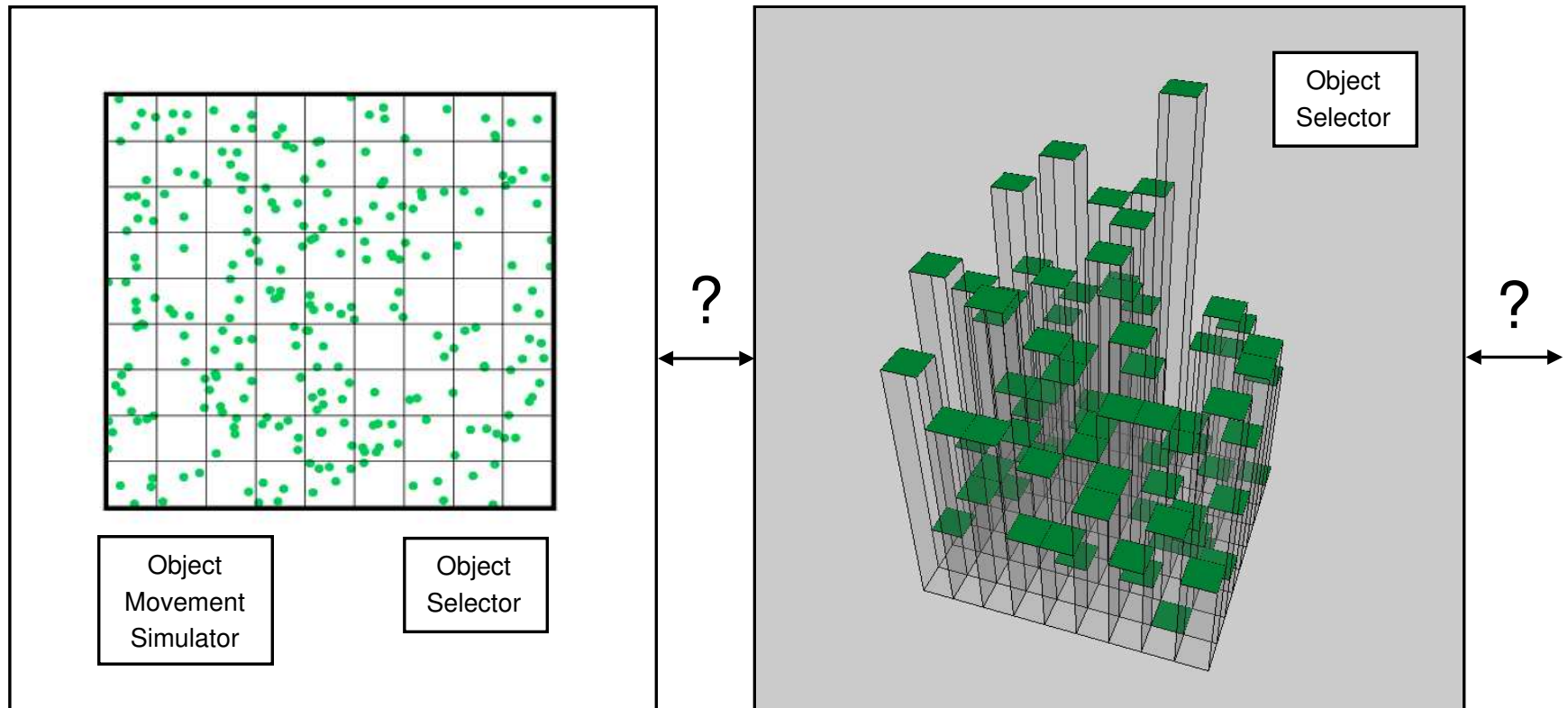
# Widget Based Paradigm Problems



# Widget Based Paradigm Problems



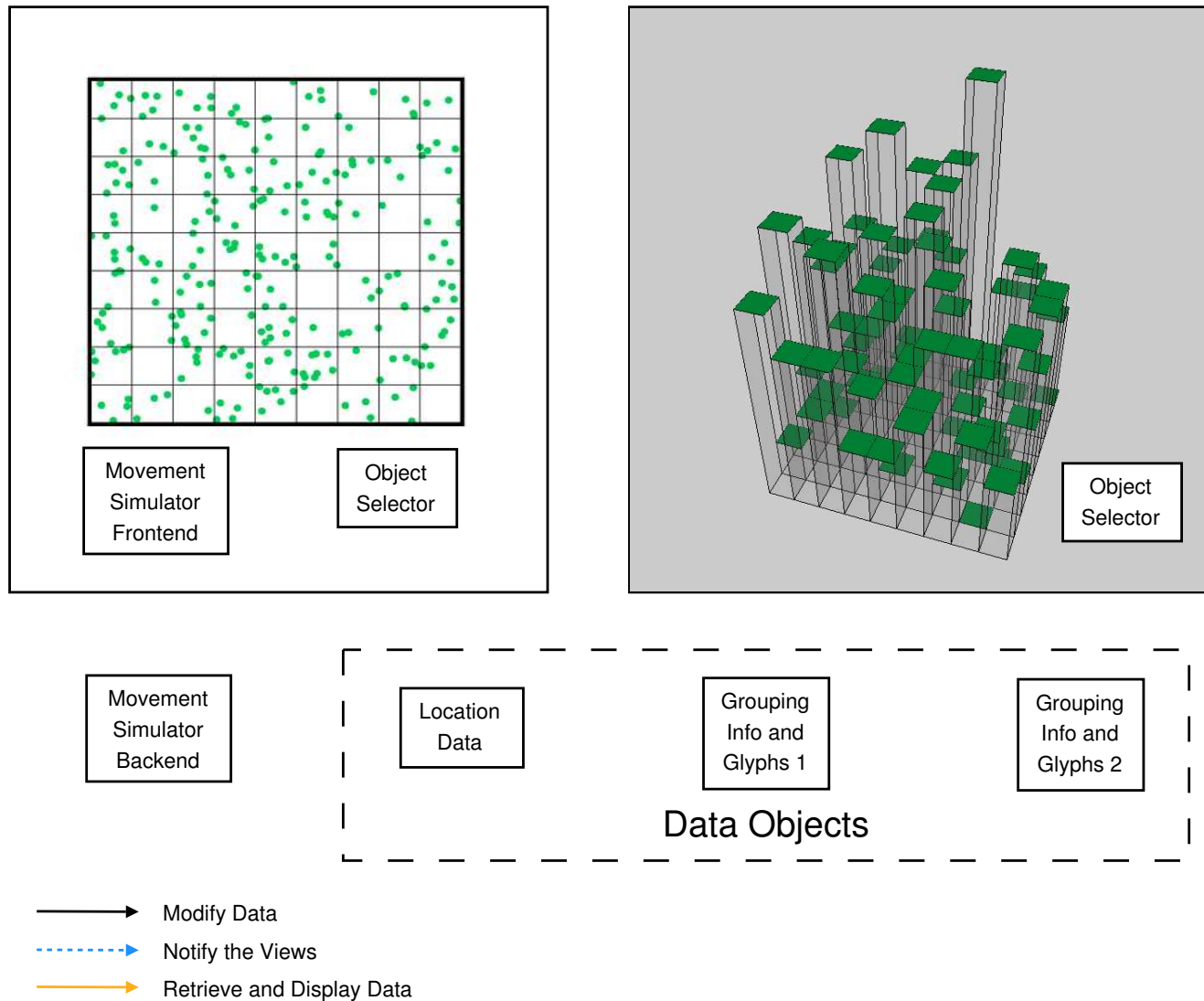
# Widget Based Paradigm Problems



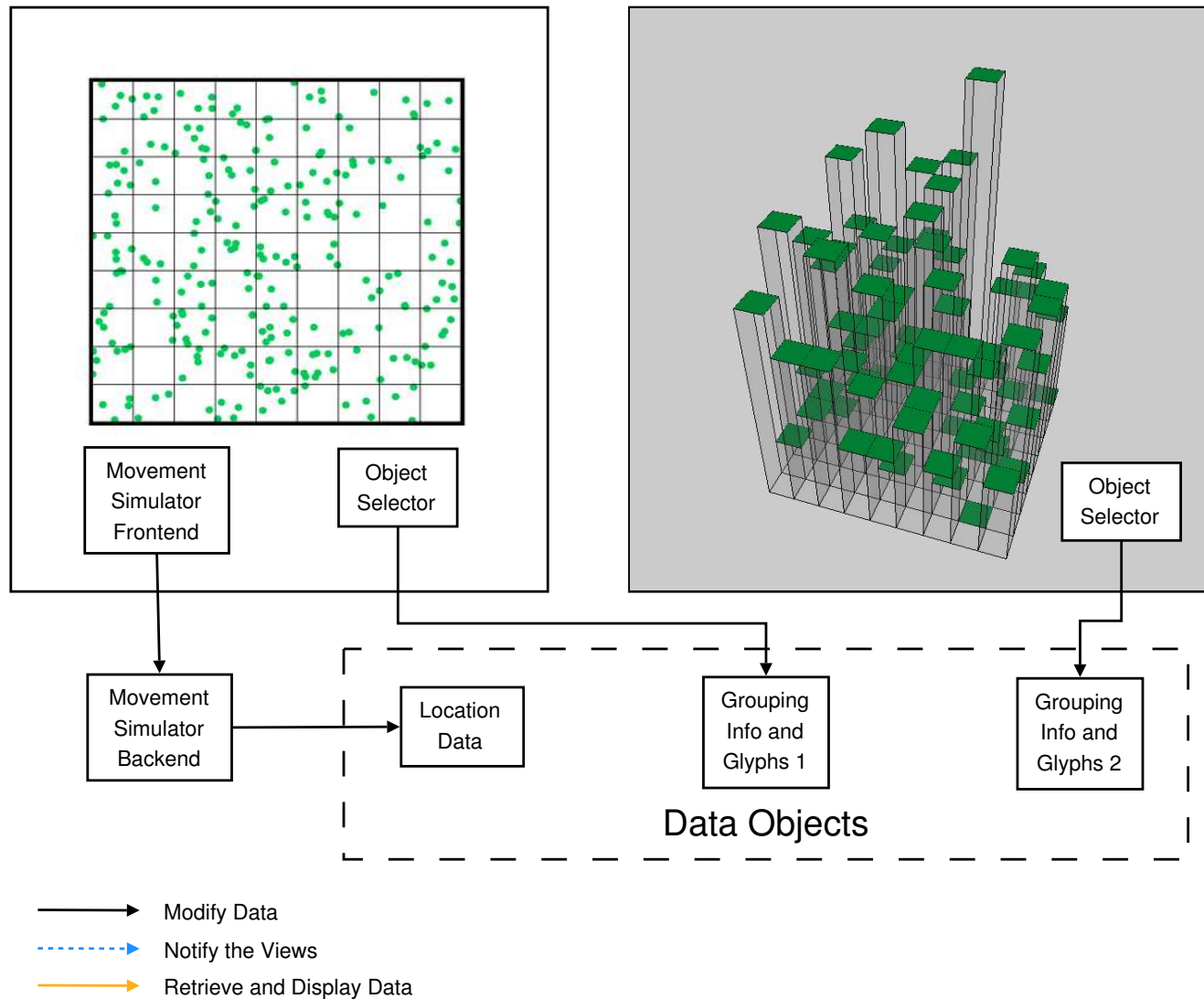


# The Two Stage Solution

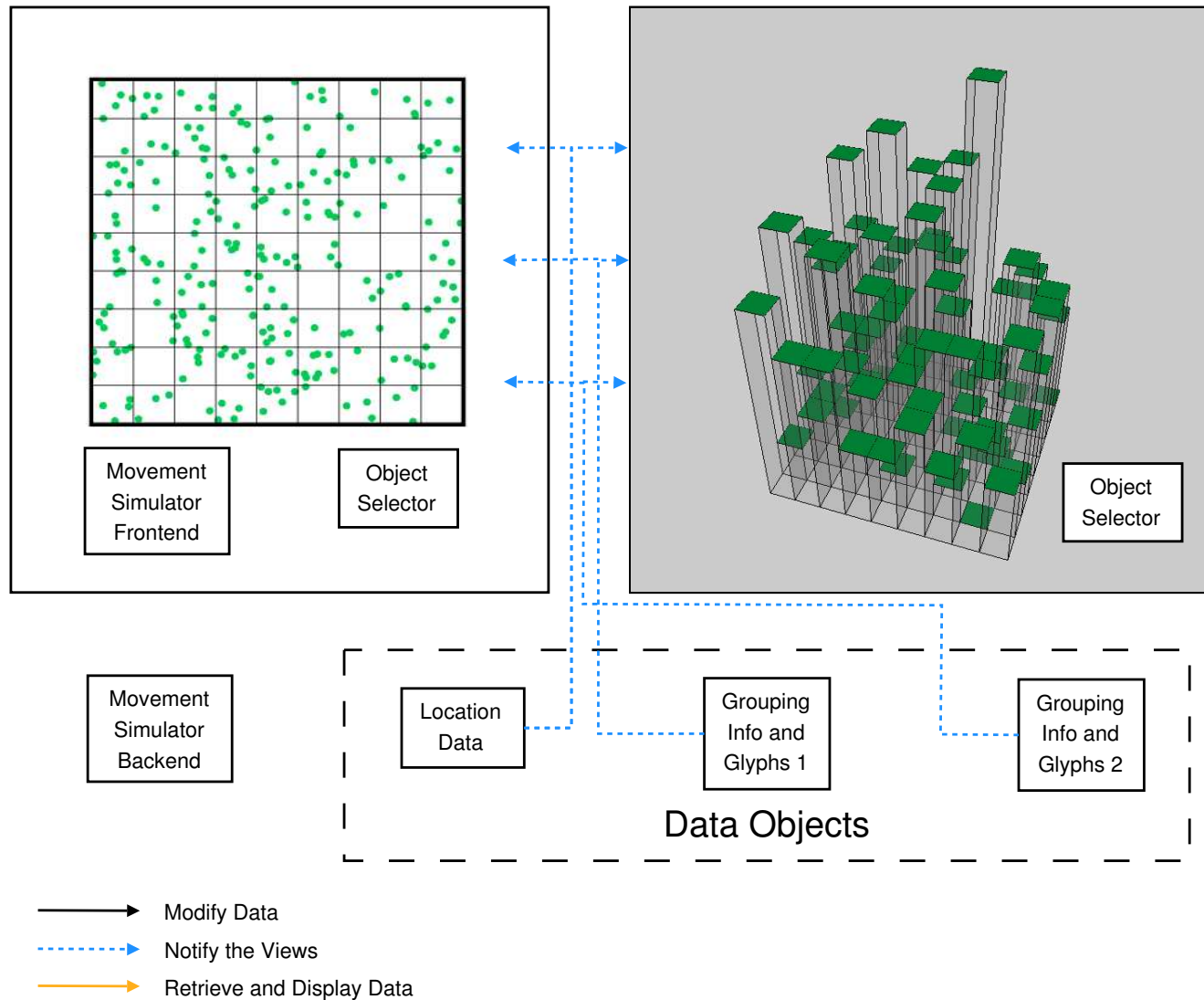
# The Two Stage Solution



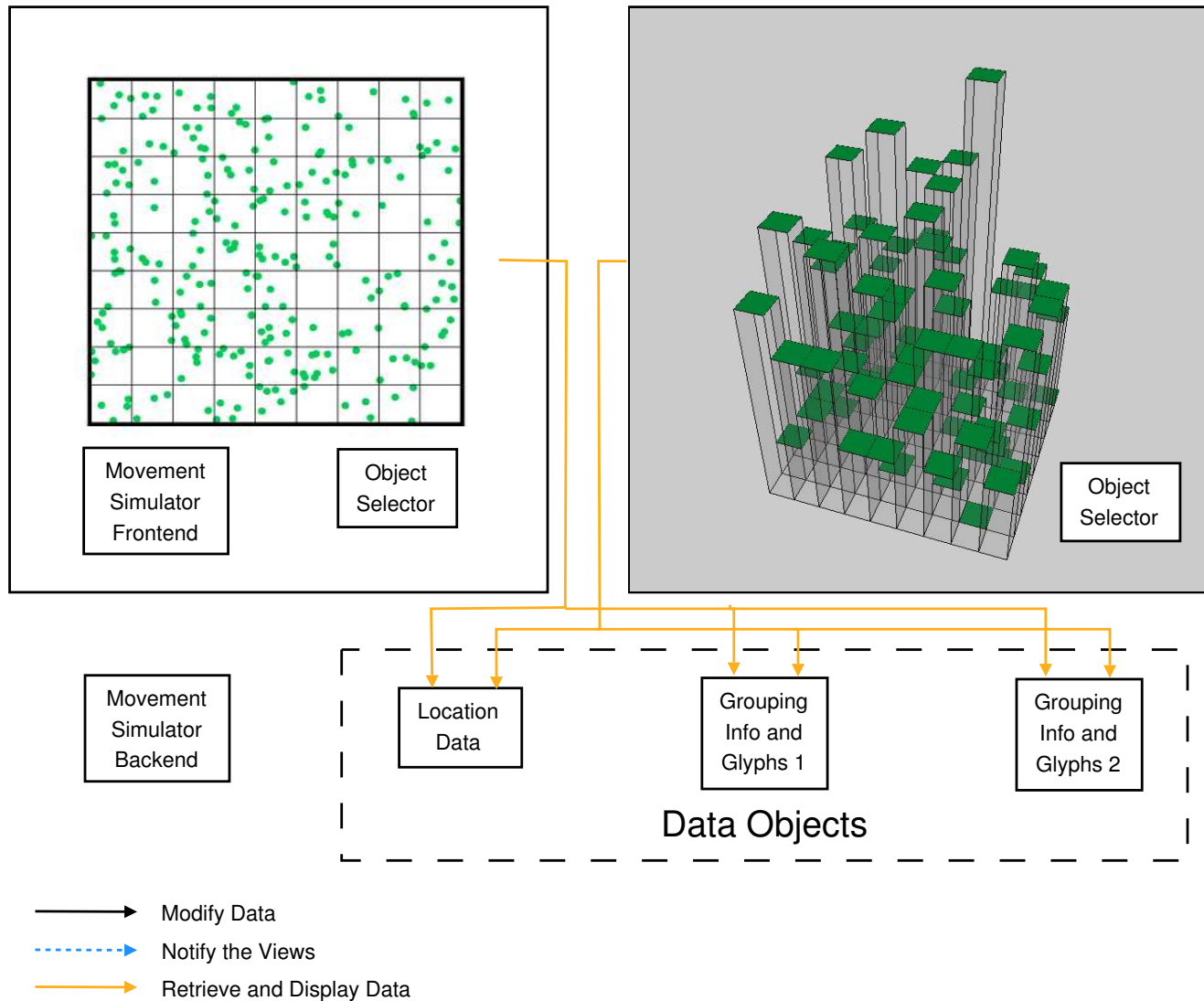
# The Two Stage Solution



# The Two Stage Solution



# The Two Stage Solution



# Consequence: View as Observer to Data

# Consequence: View as Observer to Data

- Increased complexity, but

# Consequence: View as Observer to Data

- Increased complexity, but
- Data objects do not need to know the objects referring to them any more so only need to have a single interface to manipulate and gain access to the data



# Consequence: View as Observer to Data

- Increased complexity, but
- Data objects do not need to know the objects referring to them any more so only need to have a single interface to manipulate and gain access to the data
- Data objects become reusable

# Consequence: View as Observer to Data

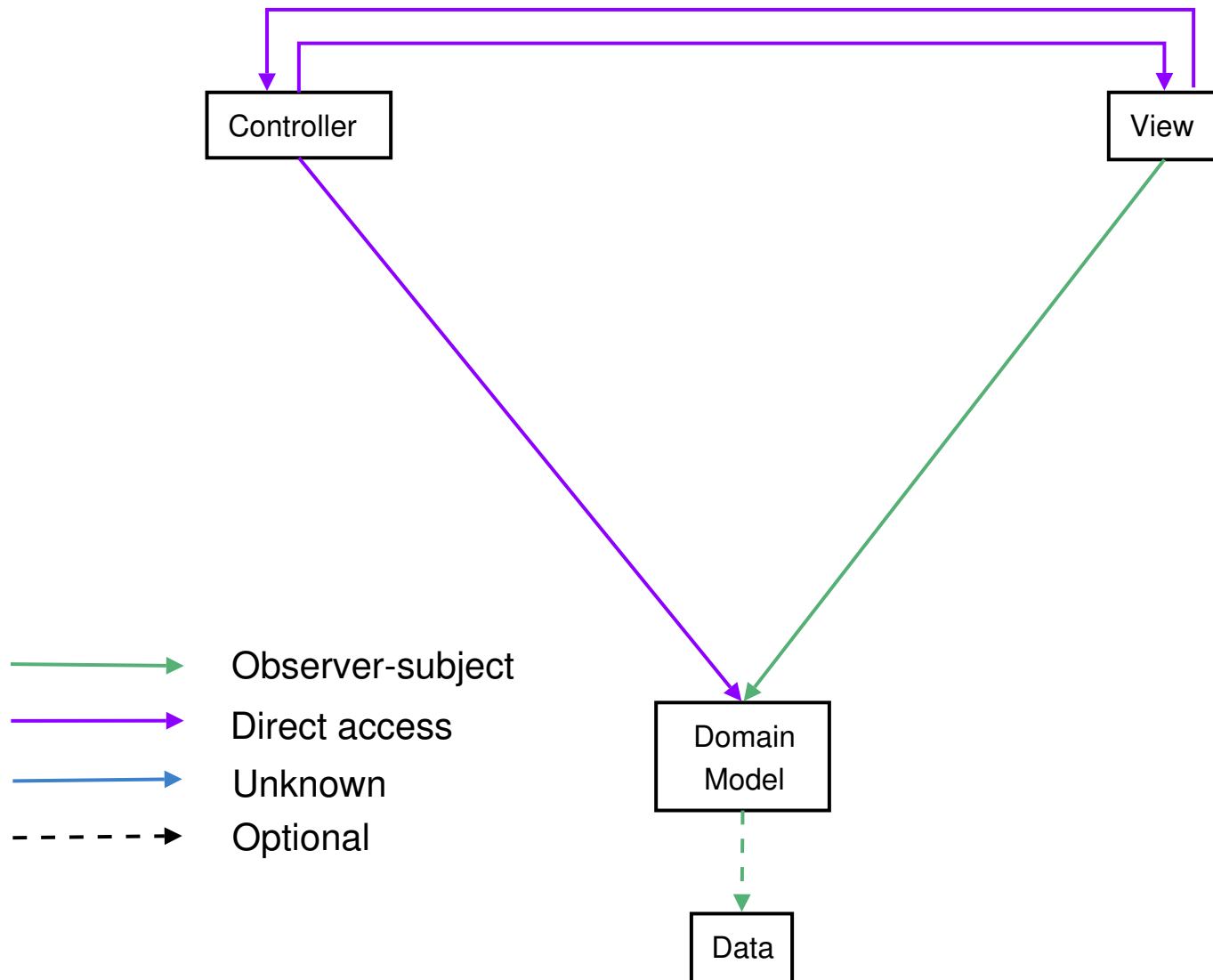
- Increased complexity, but
- Data objects do not need to know the objects referring to them any more so only need to have a single interface to manipulate and gain access to the data
- Data objects become reusable
- Plot windows can be replaced without affecting other components

# Consequence: View as Observer to Data

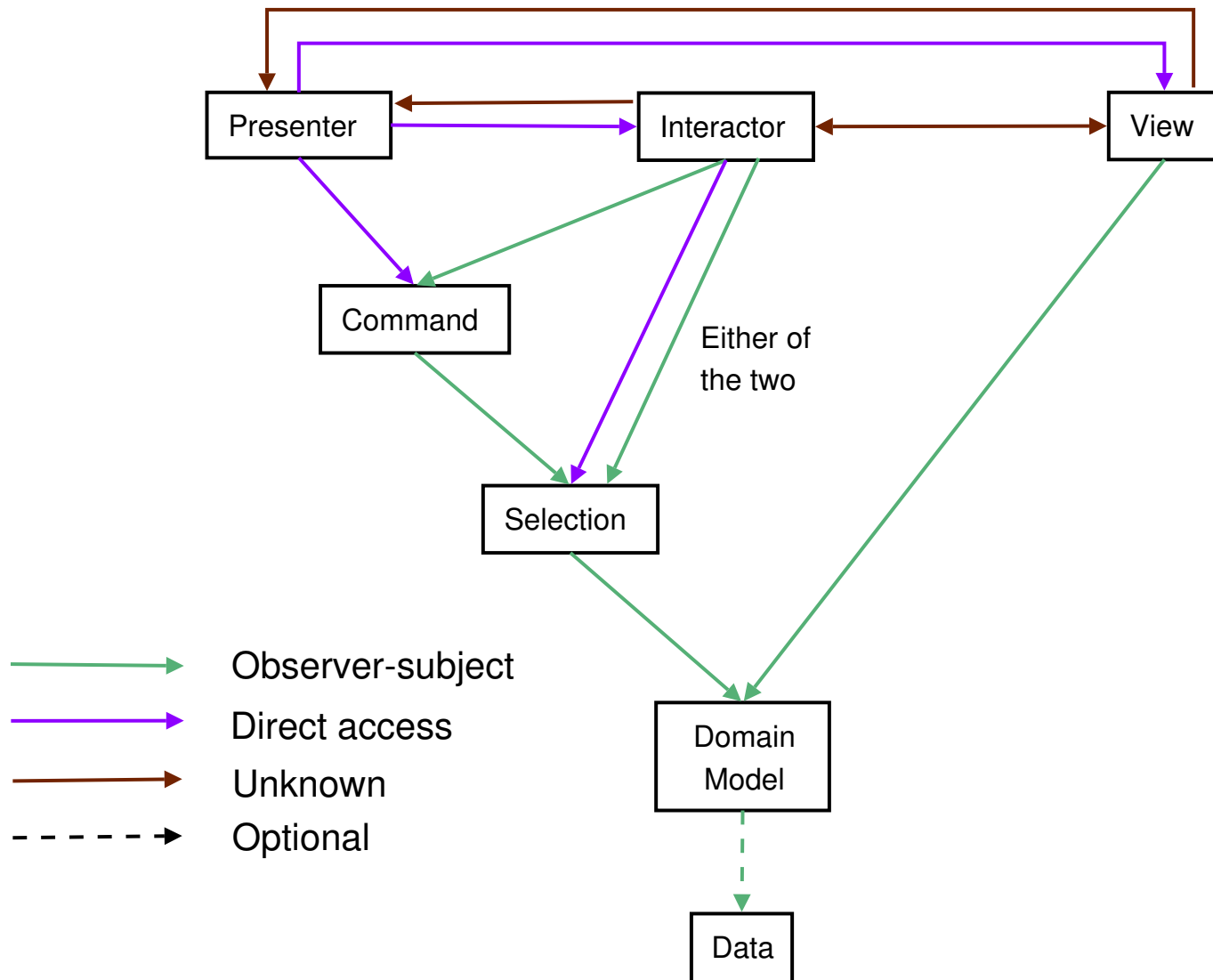
- Increased complexity, but
- Data objects do not need to know the objects referring to them any more so only need to have a single interface to manipulate and gain access to the data
- Data objects become reusable
- Plot windows can be replaced without affecting other components

The separation of the view and data and the use of the Observer pattern form the basis of Model-View-Controller (MVC) and Model-View-Presenter (MVP)

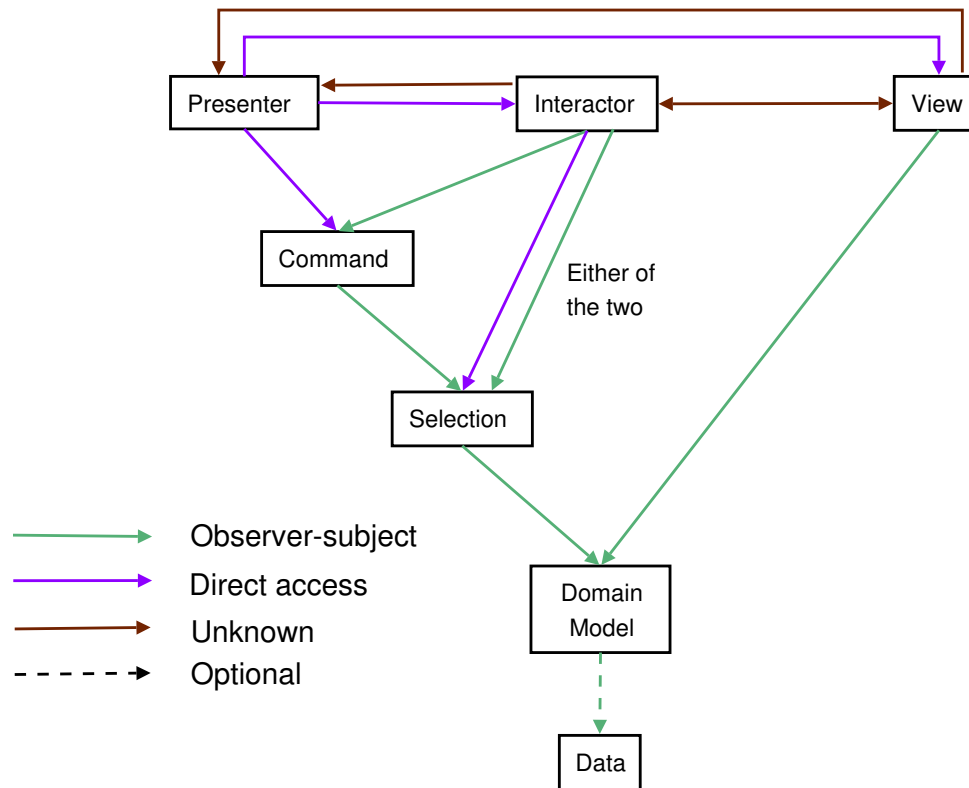
# Taligent's MVP



# Taligent's MVP

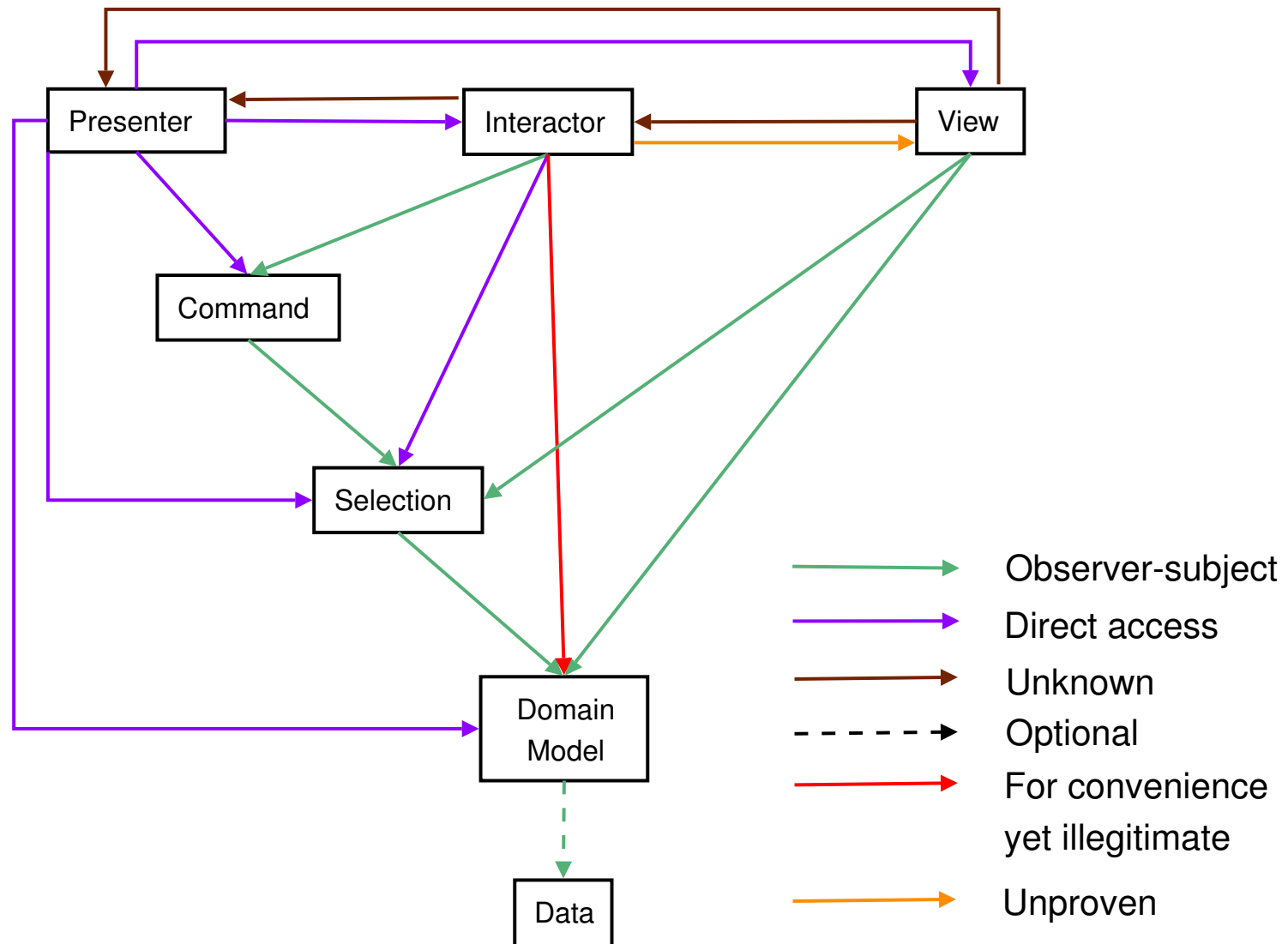


# Taligent's MVP



“Triads” can be applied on a per window basis but other implementations apply the construct to simple structures (e.g. strings) and build up using lists or the composite pattern

# Taligent's MVP (Modified for OpenGL)

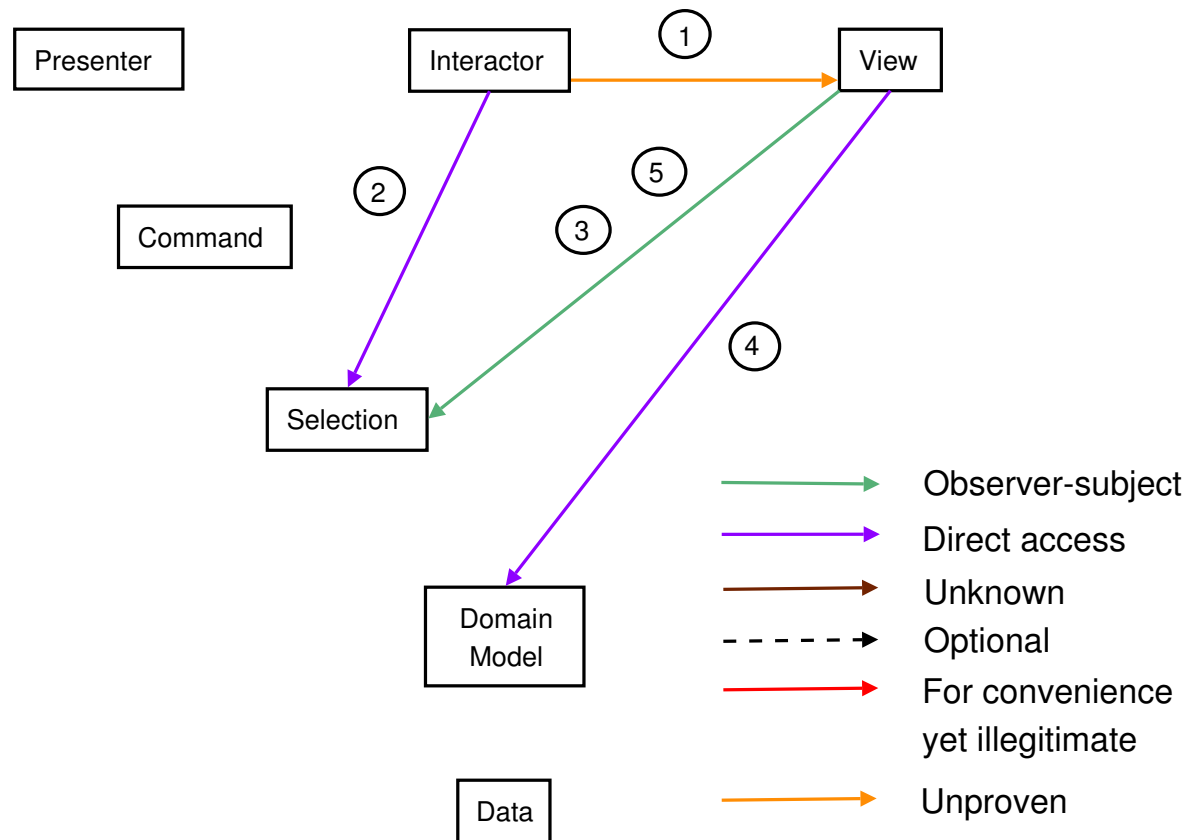


# Control Flow (Modified Version)



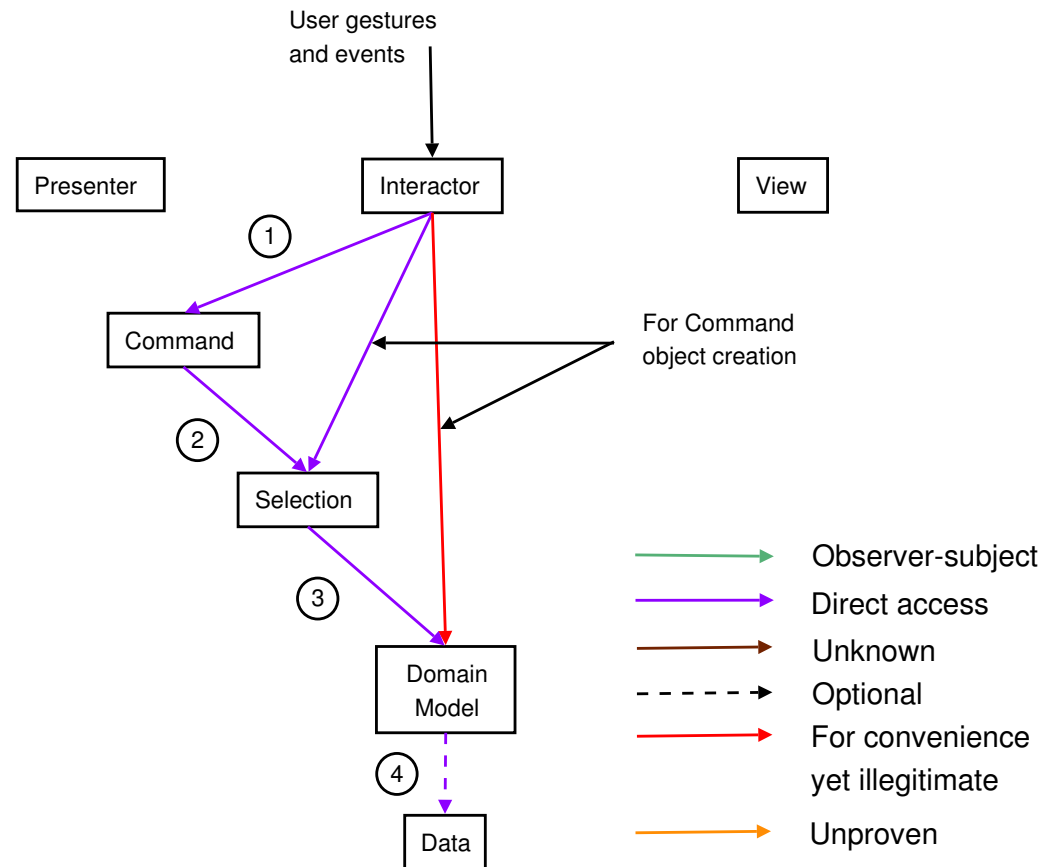
# Control Flow (Modified Version)

## Selecting Observations and Rendering Selected Obs



# Control Flow (Modified Version)

## User Gestures/Events Interpretation and Command Execution



# Conclusion

- Benefits of Model-View separation and Observing View are obvious

# Conclusion

- Benefits of Model-View separation and Observing View are obvious
- Have not yet concluded on *the best suited* framework among the various versions of MVC/MVP for this particular toolkit

# Conclusion

- Benefits of Model-View separation and Observing View are obvious
- Have not yet concluded on *the best suited* framework among the various versions of MVC/MVP for this particular toolkit
- Taligent's MVP looks good on paper but requires some further studies to assess its effectiveness in practice, as is the effect of the additional links in the modified version.

# Conclusion

- Benefits of Model-View separation and Observing View are obvious
- Have not yet concluded on *the best suited* framework among the various versions of MVC/MVP for this particular toolkit
- Taligent's MVP looks good on paper but requires some further studies to assess its effectiveness in practice, as is the effect of the additional links in the modified version.
- The steep learning curve is a major drawback and proper documentations and assisting tools will be necessary

# Some Well-known Software Paradigms

- Smalltalk-80 MVC
- Presentation-Abstraction-Control (PAC)
- Widget based framework
- VisualWorks MVC
- Taligent's MVP
- Dolphin Smalltalk's MVP
- Humble View

# Selected Papers and Articles

- S. Burbeck (1992). "Applications Programming in Smalltalk-80: How to use Model-View-Controller.
- M. Potel (1996), MVP: Model-View-Presenter - The Taligent Programming Model for C++ and Java
- A. Bower, B. McGlashan (2000), Twisting the Triad: The Evolution of the Dolphin Smalltalk MVP Application Framework
- J. Carter (2001-2002), Most Valuable Player? - Tutorial Series on Taligent's Model View Presenter
- M. Fowler (2006), GUI Architectures (online article)
- Gamma et. al. (1995), Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley.



# Quote of the day

“Many people in those days considered it impractical to use a virtual machine. I wonder what our prior selves would have thought to see me running Smalltalk 80 in a virtual machine written in VisualWorks running in the VisualWorks virtual machine on Windows XP running in a VMware virtual machine running on Ubuntu.”

– GUI architectures by Martin Fowler, in his effort towards an in-depth study of the classic MVC used in Smalltalk-80

# Mysteries of Taligent's MVP

- Unknown links and their respective roles if they exist, especially the Interactor-View and Interactor-Presenter link

# Mysteries of Taligent's MVP

- Unknown links and their respective roles if they exist, especially the Interactor-View and Interactor-Presenter link
- Selection component: defining and rendering selection regions without any knowledge about the View (see later)

# Mysteries of Taligent's MVP

- Unknown links and their respective roles if they exist, especially the Interactor-View and Interactor-Presenter link
- Selection component: defining and rendering selection regions without any knowledge about the View (see later)
- Implementation of the Command component (partially solved)

# Mysteries of Taligent's MVP

- Apparent contradiction between Presenter's role of interpreting user gestures and events from Interactor and the addMenuAndCommand method addressed in the "class diagram" in the original paper

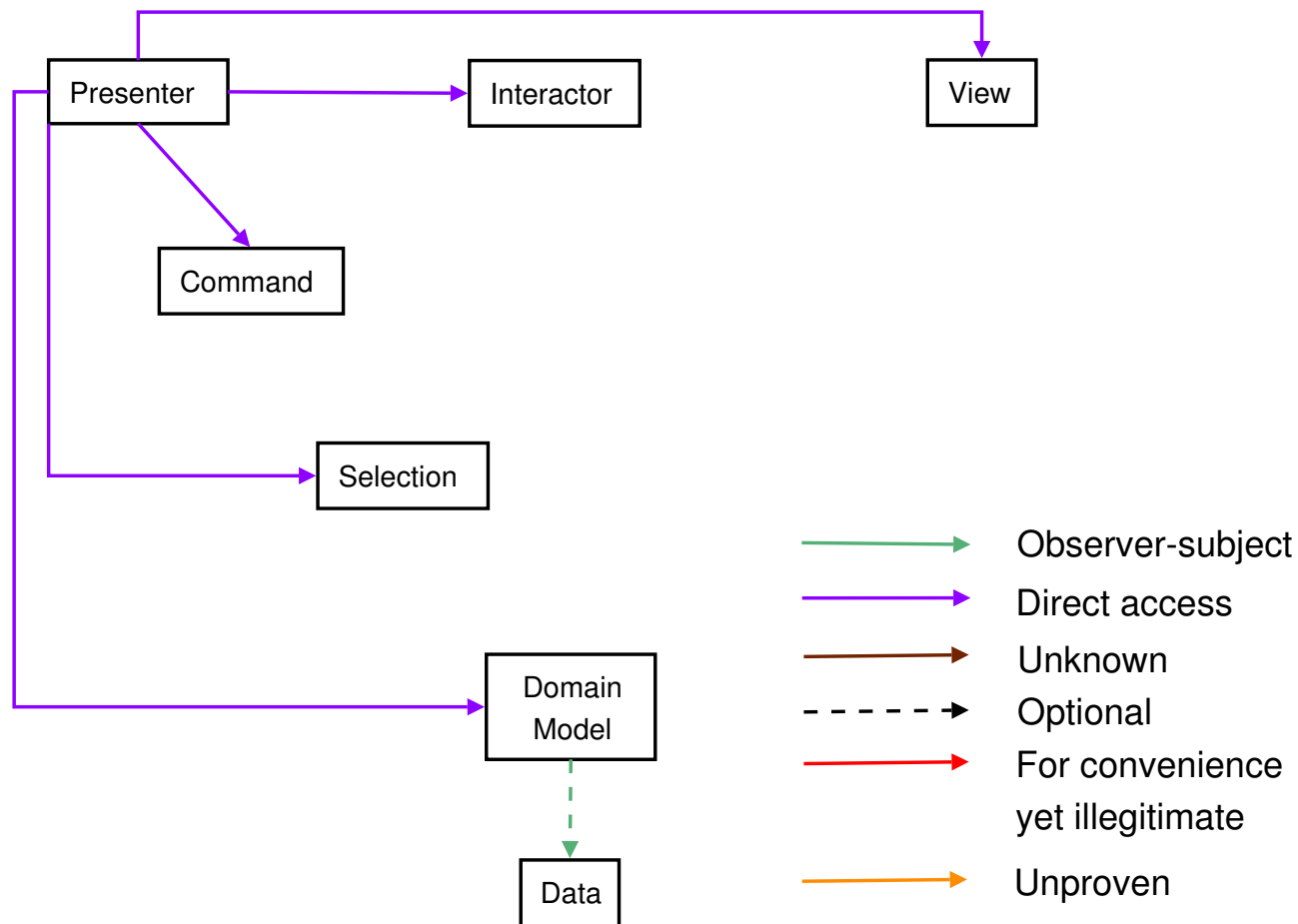
# Mysteries of Taligent's MVP

- Apparent contradiction between Presenter's role of interpreting user gestures and events from Interactor and the addMenuAndCommand method addressed in the "class diagram" in the original paper
- The appropriate timing when the addMenuAndCommand method, addressed in the "class diagram" in the original paper, should be called

# Control Flow (Modified Version)

# Control Flow (Modified Version)

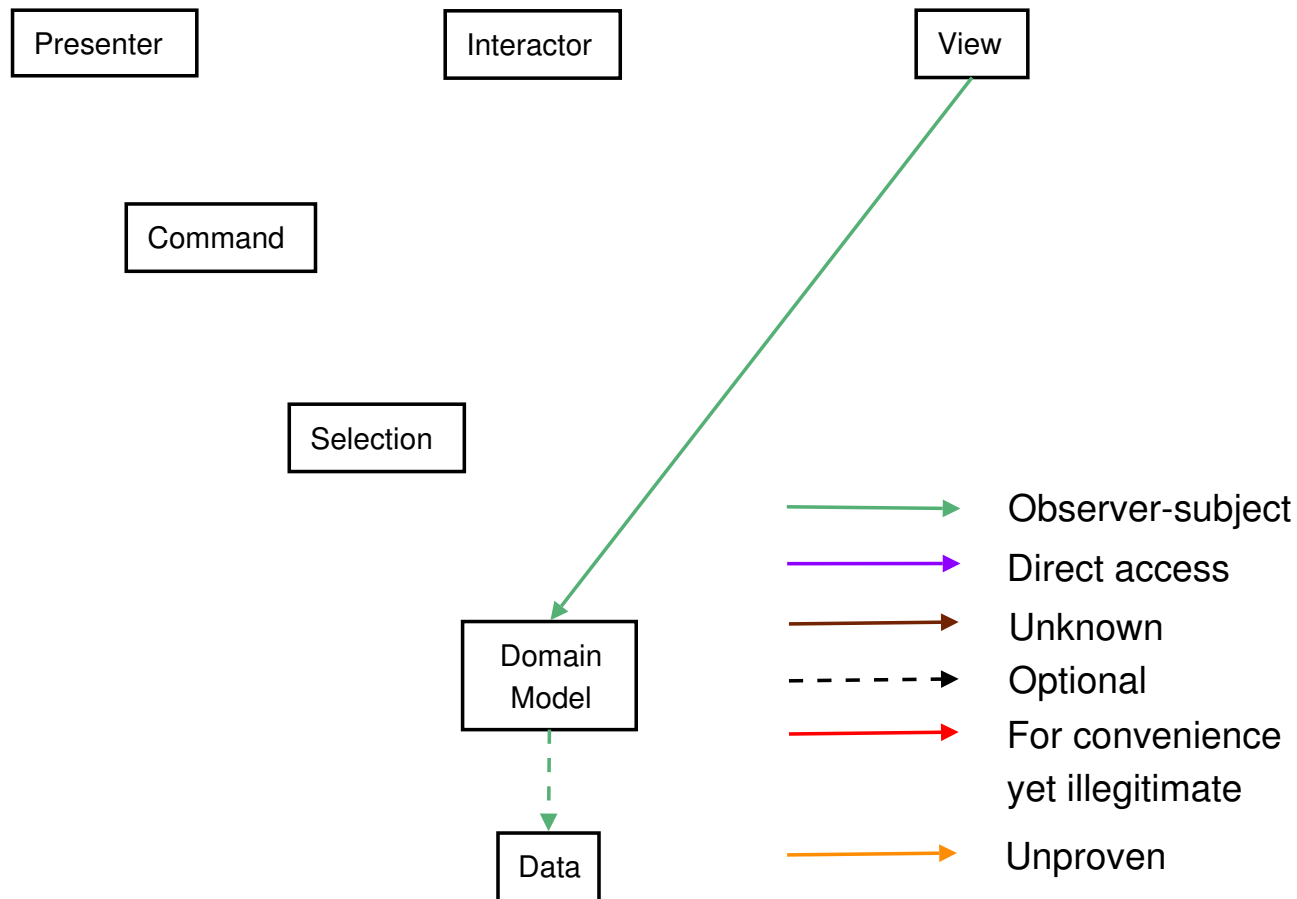
## Triad Creation





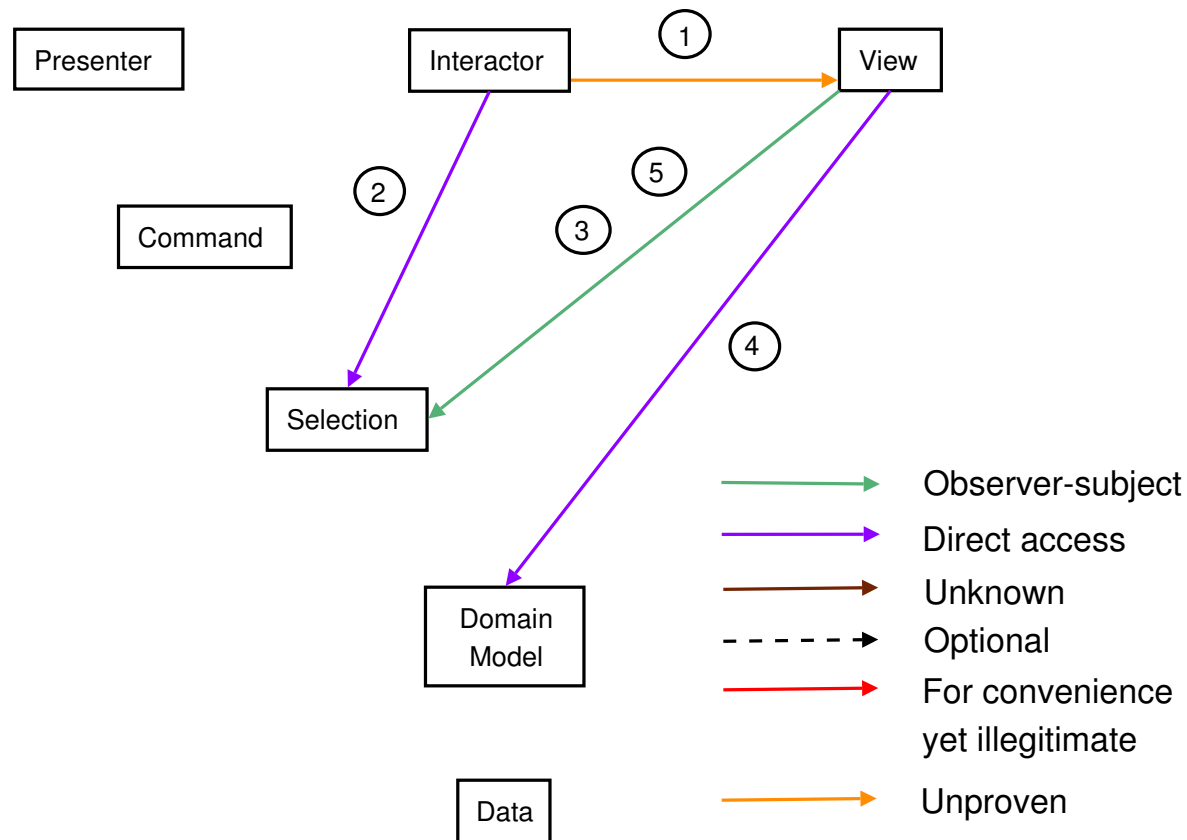
# Control Flow (Modified Version)

## Rendering the Data



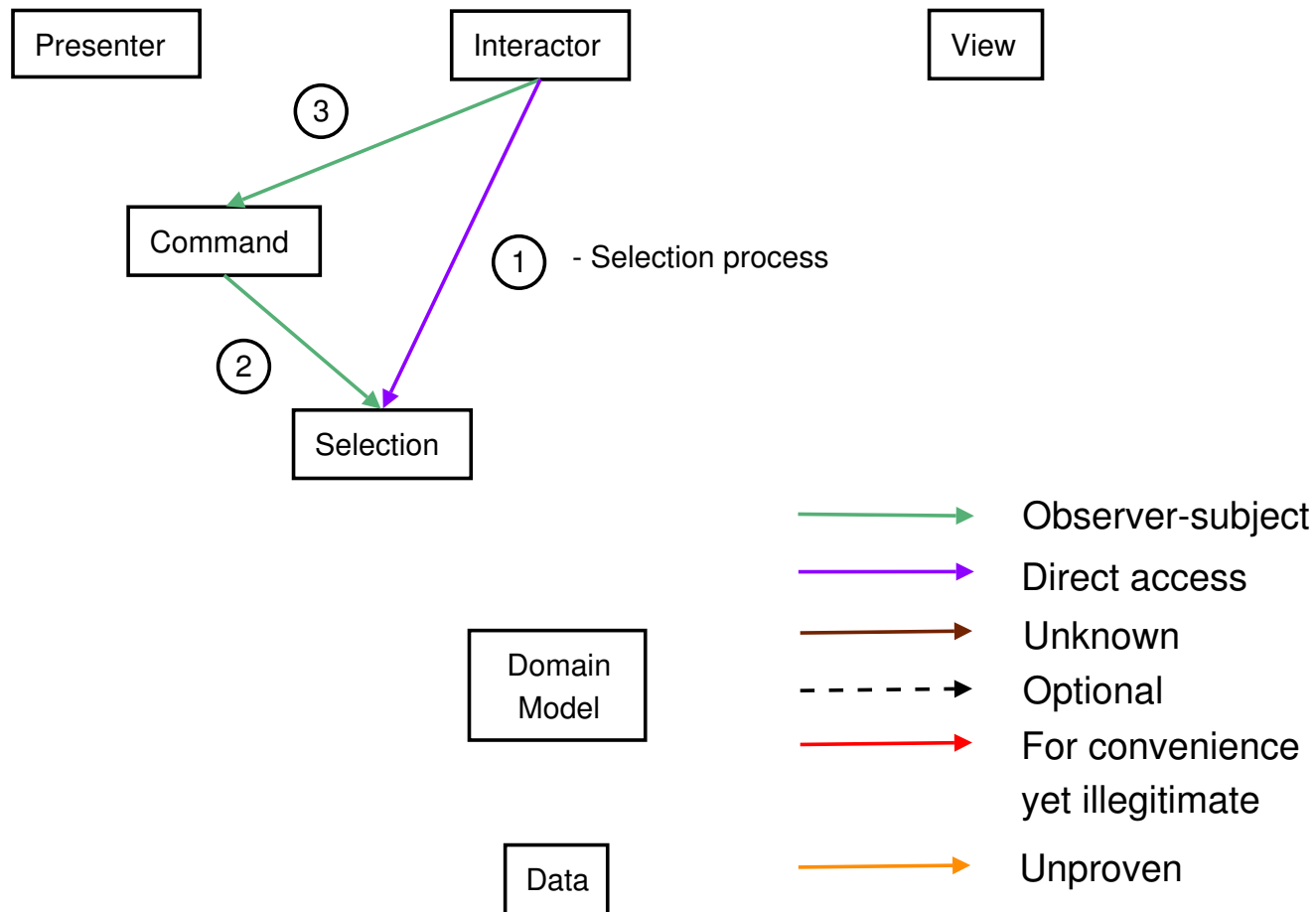
# Control Flow (Modified Version)

## Selecting Observations and Rendering Selected Obs



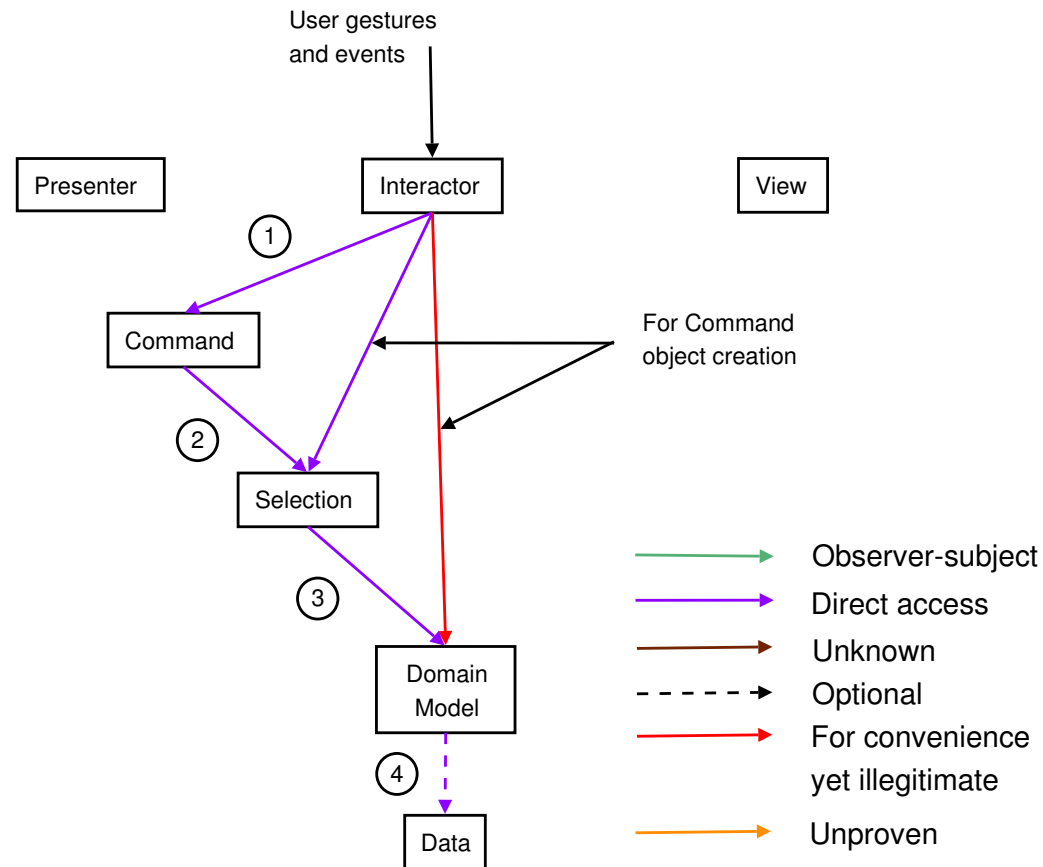
# Control Flow (Modified Version)

## Update Available Commands



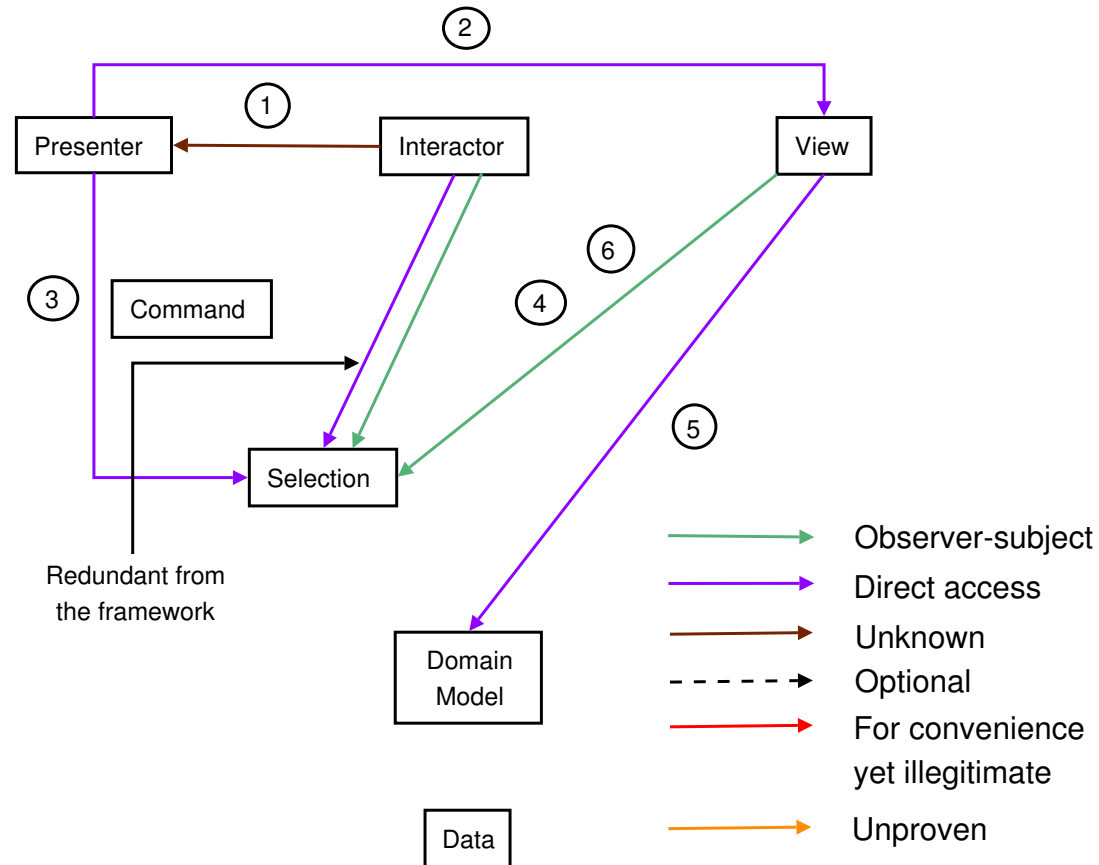
# Control Flow (Modified Version)

## User Gestures/Events Interpretation and Command Execution



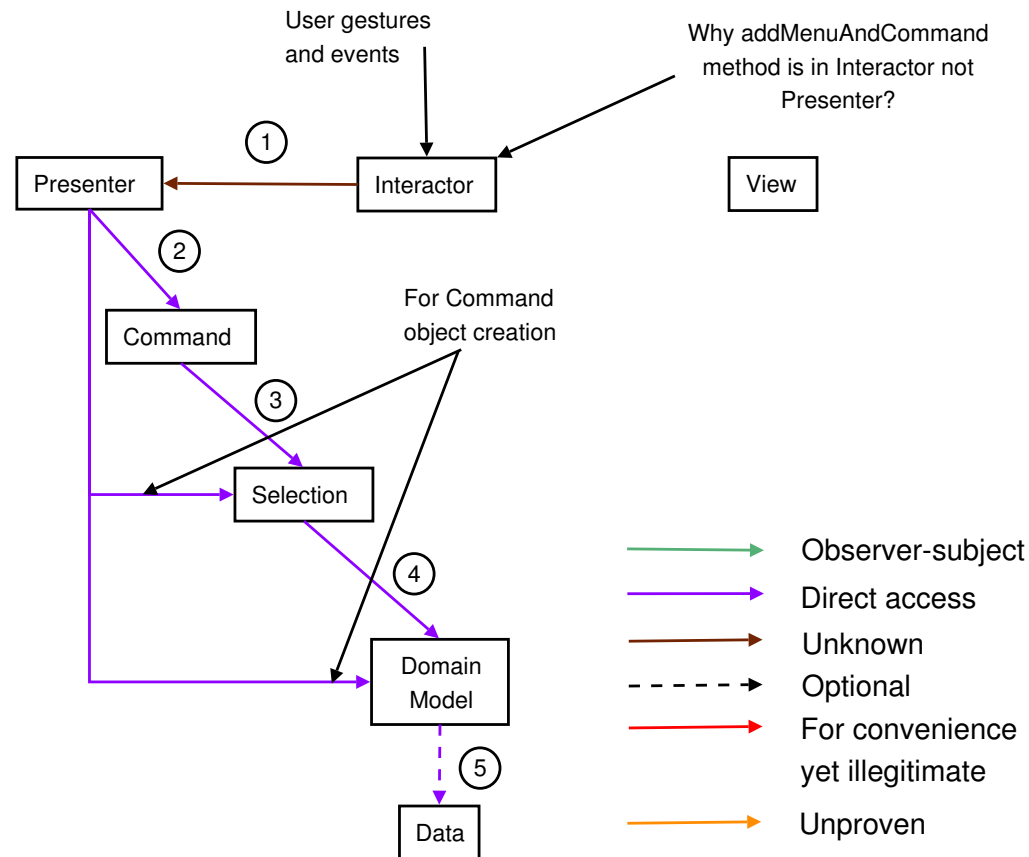
# Control Flow (Alternatives)

## Selecting Observations and Rendering Selected Obs (b)



# Control Flow (Alternatives)

## User Gestures/Events Interpretation and Command Execution (b)



# Additional Links for the Modified Taligent's M

- Interactor-View link: In OpenGL, need to render the selectable objects in the selection buffer and feedback buffer

# Additional Links for the Modified Taligent's M

- Interactor-View link: In OpenGL, need to render the selectable objects in the selection buffer and feedback buffer
- Interactor-Model link: For pluggability, Command object creation requires the type of Selection and Model. It also eases command executions.

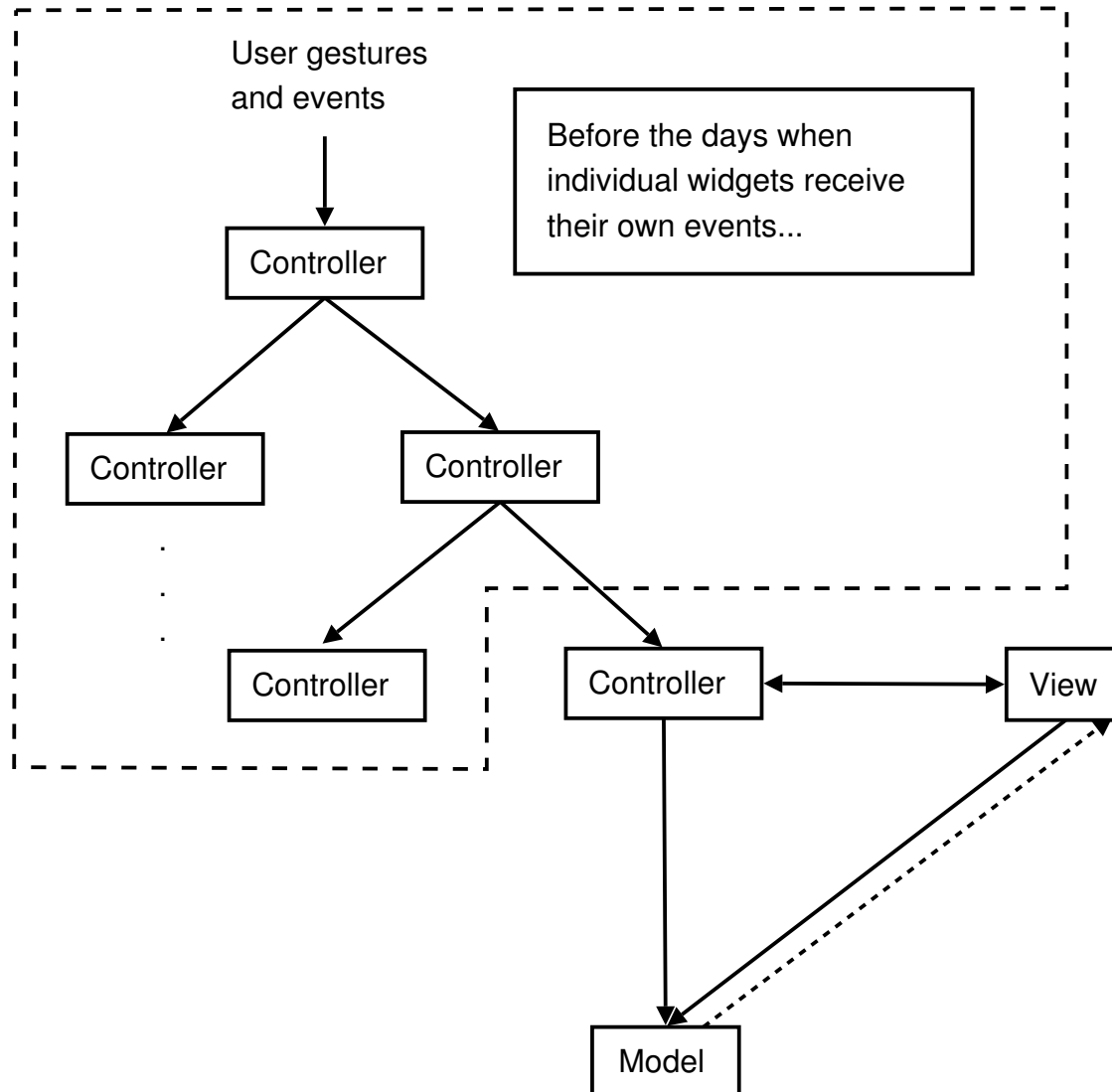


# Additional Links for the Modified Taligent's M

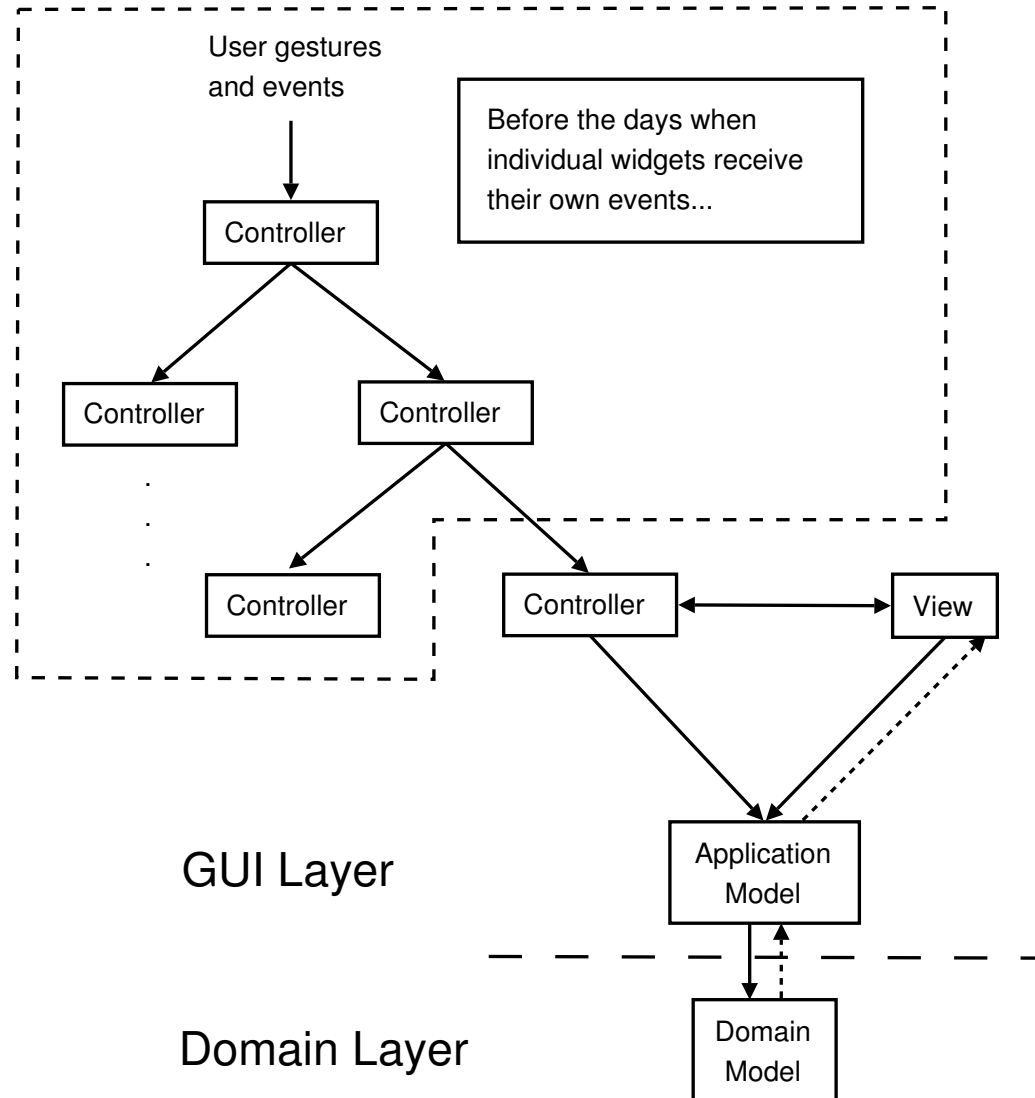
- Interactor-View link: In OpenGL, need to render the selectable objects in the selection buffer and feedback buffer
- Interactor-Model link: For pluggability, Command object creation requires the type of Selection and Model. It also eases command executions.
- View-Selection observer link: For linked selection, a single Selection component cannot render selection regions for different Views. Have to be done in View instead.

# MVC-MVP Evolution

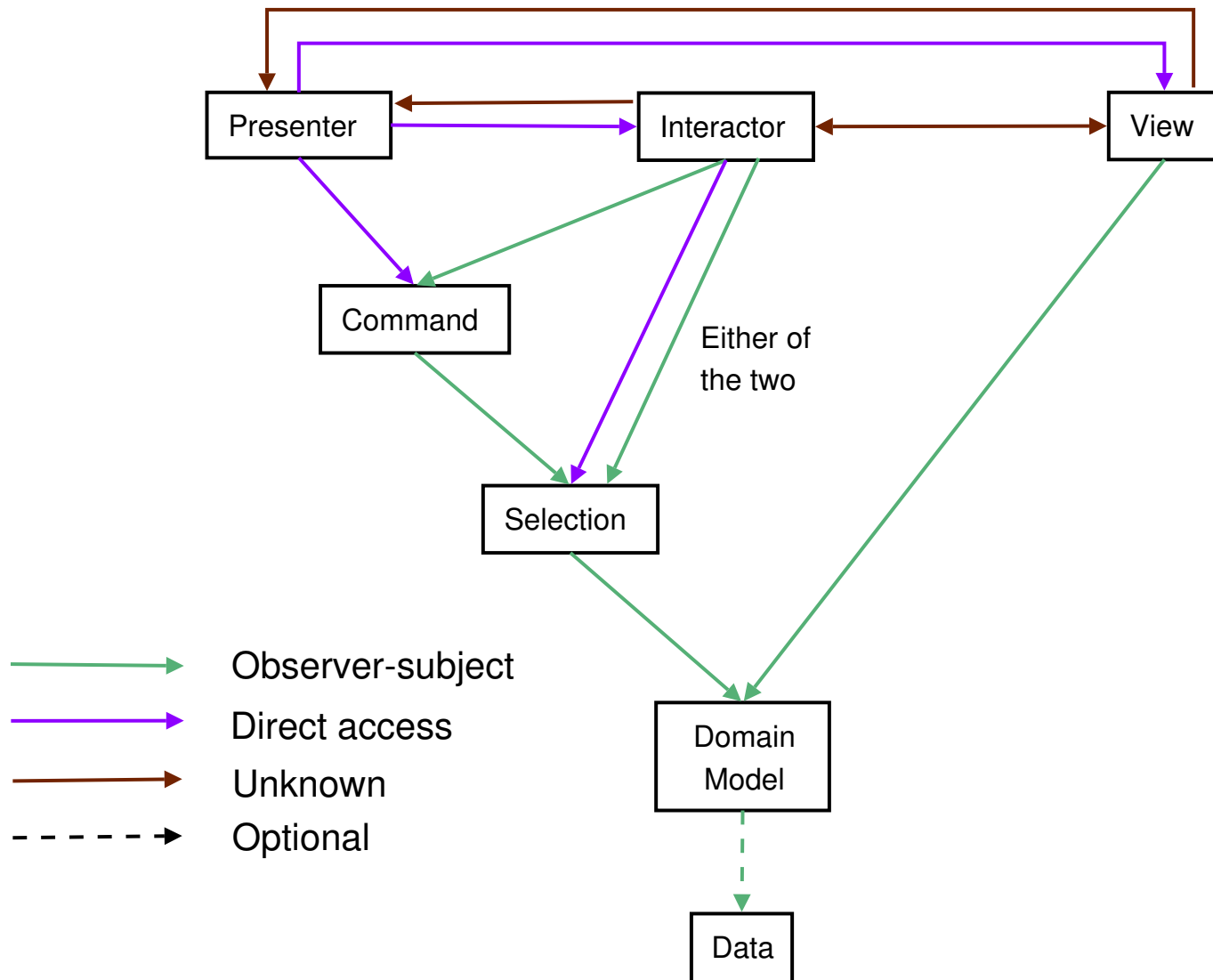
# MVC-MVP Evolution



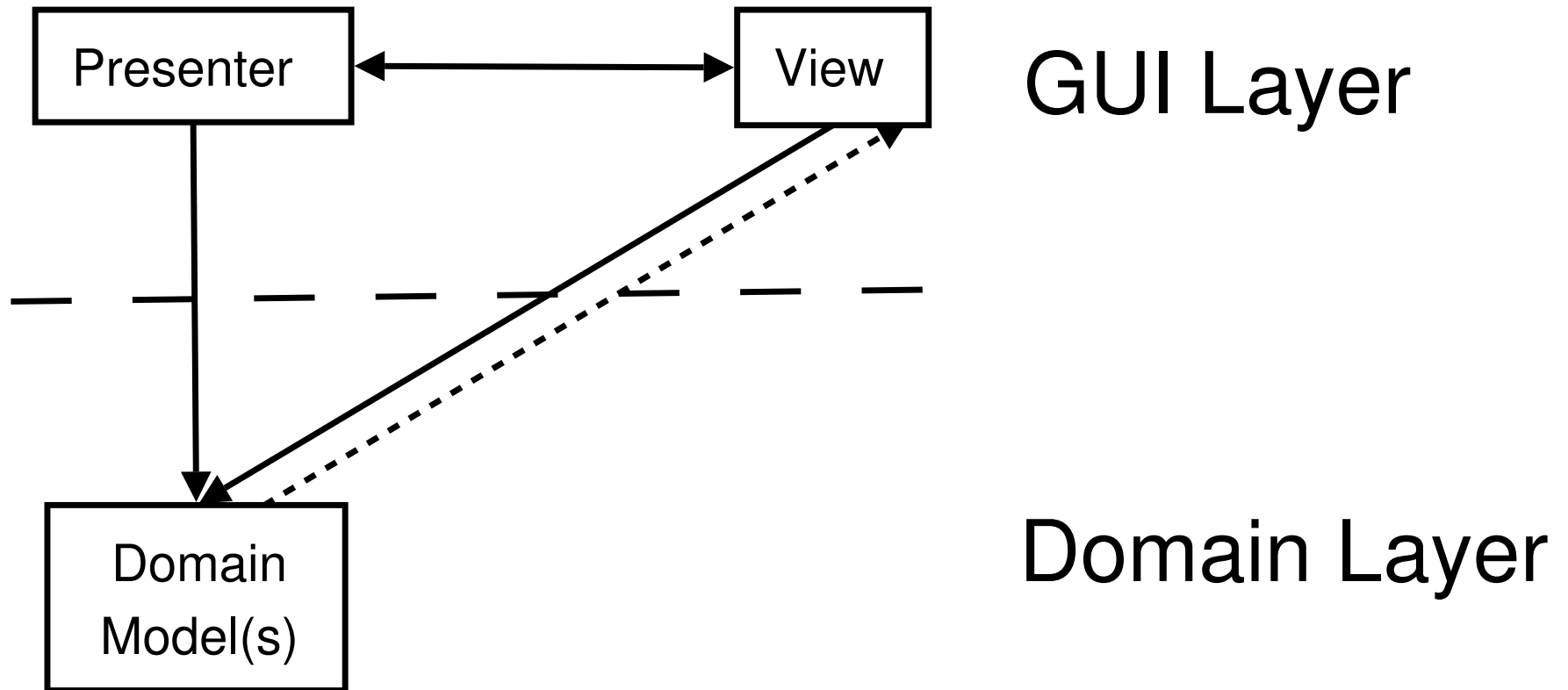
# MVC-MVP Evolution



# MVC-MVP Evolution



# MVC-MVP Evolution



# Taligent's MVP Class Diagram

