W

window gravity When windows are resized, subwindows can be repositioned automatically relative to some position in the window. This attraction of a subwindow to some part of its parent is known as *window gravity*.

window manager Manipulation of windows on the screen, and much of the user interface (policy) is typically provided by a *window manager* client.

window manager shell A subclass of **shell** called **wm-shell** that interacts with the window manager.

X

XY Format The data for a pixmap is said to be in *XY Format* if it is organized as a set of bitmaps representing individual bit planes, with the planes appearing from most to least significant in bit order.

Z

Z Format The data for a pixmap is said to be in *Z Format* if it is organized as a set of pixel values in scanline order.

t

tile A pixmap can be replicated in two dimensions to *tile* a region. The pixmap itself is also known as a tile.

timer A CLUE object that provides support for animation and other types of time-sensitive user interfaces. A timer causes **:timer** events to be dispatched to a specific contact for processing.

timestamp A time value, expressed in milliseconds, typically since the last server reset. Timestamp values wrap around (after about 49.7 days). The server, given its current time is represented by timestamp T, always interprets timestamps from clients by treating half of the timestamp space as being earlier in time than T and half of the timestamp space as being later in time than T. One timestamp value (named CurrentTime) is never generated by the server; this value is reserved for use in requests to represent the current server time.

top-level contact A contact whose parent is a root. A top-level contact is usually a composite at the top of a hierarchy of other contacts created by an application program.

top-level-session A subclass of **shell** that is used to communicate with a session manager.

top-level-shell A subclass of **shell** that provides full window manager interaction.

transient-shell A subclass of **shell** that a window manager typically will unmap when its owner becomes unmapped or iconified and will not allow to be individually iconified.

true color A degenerate case of direct color in which the subfields in the pixel value directly encode the corresponding RGB values. That is, the colormap has predefined read-only RGB values. The values are typically (near-)linear increasing ramps.

type An arbitrary atom used to identify the interpretation of property data. Types are completely uninterpreted by the server; they are solely for the benefit of clients.

u

unmanaged A contact that is not under geometry management control.

user interface A set of abstract interface objects used to control the dialog between an application and its human user.

v

viewable A window is *viewable* if it and all of its ancestors are mapped. This does not imply that any portion of the window is actually visible. Graphics requests can be performed on a window when it is not viewable, but output will not be retained unless the server is maintaining backing store.

visible A region of a window is *visible* if someone looking at the screen can actually see it; that is, the window is viewable and the region is not occluded by any other window.

screen A server can provide several independent *screens*, which typically have physically independent monitors. This would be the expected configuration when there is only a single keyboard and pointer shared among the screens.

selection A *selection* can be thought of as an indirect property with dynamic type. That is, rather than having the property stored in the server, it is maintained by some client (the *owner*). A selection is global in nature, being thought of as belonging to the user (but maintained by clients), rather than being private to a particular window subhierarchy or a particular set of clients. When a client asks for the contents of a selection, it specifies a selection *target type*. This target type can be used to control the transmitted representation of the contents.

For example, if the selection is "the last thing the user clicked on" and that is currently an image, then the target type might specify whether the contents of the image should be sent in XY Format or Z Format. The target type can also be used to control the class of contents transmitted; that is, asking for the looks (fonts, line spacing, indentation, and so forth) of a paragraph selection, rather than the text of the paragraph. The target type can also be used for other purposes; the semantics is not constrained by the protocol.

sensitivity A condition in which a user interface component of an application will accept input. Conversely, when a contact is insensitive, events of particular types are not dispatched to the contact and are ignored.

server The *server* provides the basic windowing mechanism. It handles IPC connections from clients, demultiplexes graphics requests onto the screens, and multiplexes input back to the appropriate clients.

server grabbing The server can be *grabbed* by a single client for exclusive use. This prevents processing of any requests from other client connections until the grab is complete. This is typically only a transient state for such things as rubber-banding and pop-up menus, or to execute requests indivisibly.

server resource Windows, pixmaps, cursors, fonts, gcontexts, and colormaps are known as resources. They all have unique identifiers associated with them for naming purposes. The lifetime of a resource is bounded by the lifetime of the connection over which the resource was created. See resource.

shell A composite that handles the duties required by standard conventions for top-level X windows.

sibling Children of the same parent window are known as *sibling* windows.

static color A degenerate case of pseudo color in which the RGB values are predefined and read-only.

static gray A degenerate case of gray scale in which the gray values are predefined and read-only. The values are typically (near-)linear increasing ramps.

stacking order Sibling windows can *stack* on top of each other. Windows above both obscure and occlude lower windows. This is similar to paper on a desk. The relationship between sibling windows is known as the *stacking order*.

state A slot of **contact** that controls the visual effect of the contact.

stipple A bitmap that is used to tile a region to serve as an additional clip mask for a fill operation with the foreground color.

r

realization — See contact realization.

redirecting control — Window managers (or client programs) may choose to enforce window layout policy in various ways. When a client attempts to change the size or position of a window, the operation can be *redirected* to a specified client, rather than the operation actually being performed.

reply — Information requested by a client program is sent back to the client with a *reply*. Both events and replies are multiplexed on the same connection. Most requests do not generate replies. However, some requests generate multiple replies.

representation type — The type of representation of a resource value. For example, a color value might be represented either as a namestring ("red"), a pixel value, an RGB triplet, an HSV triplet, and so on.

request — A command to the server is called a *request*. It is a single block of data sent over a connection.

resource — A value of the user interface that can be changed by the user in a resource database via CLX functions **add-resource**, **get-resource**, and so forth. See server resource.

resource class, complete — See complete resource class.

resource database — Conceptually, a set of resource name/value pairs (or resource bindings). CLX defines functions for storing and retrieving interface resources from a resource database.

resource name, complete — See complete resource name.

RGB values — *Red*, *green*, and *blue* intensity values used to define color. These values are always represented as 16-bit unsigned numbers, with zero being the minimum intensity and 65535 being the maximum intensity. The values are scaled by the server to match the display hardware.

root — A special composite contact used to represent an entire display screen.

root window — Each screen has a *root window* covering it. It cannot be reconfigured or unmapped, but otherwise acts as a full-fledged window. A root window has no parent.

s

save set — The *save set* of a client is a list of other client's windows that, if they are inferiors of one of the client's windows at connection close, should not be destroyed and that should be remapped if it is unmapped. Save sets are typically used by window managers to avoid lost windows if the manager should terminate abnormally.

scanline — A list of pixel or bit values viewed as a horizontal row (all values having the same y coordinate) of an image, with the values ordered by increasing x coordinate.

scanline order — An image represented in *scanline order* contains scanlines ordered by increasing y coordinate.

override-shell A subclass of **shell** used to override the window manager. This subclass contains pop-up menus and other temporary objects that the user can never resize and so on.

p

padding Some padding bytes are inserted in the data stream to maintain alignment of the protocol requests on natural boundaries. This increases ease of portability to some machine architectures.

parent window If C is a child of P, then P is the *parent* of C.

passive grab Grabbing a key or button is a *passive grab*. The grab activates when the key or button is actually pressed.

pixel value An *n*-bit value, where *n* is the number of bit planes used in (that is, the depth of) a particular window or pixmap. For a window, a pixel value indexes a colormap to derive an actual color to be displayed.

pixmap A three dimensional array of bits. A pixmap is normally thought of as a two dimensional array of pixels, where each pixel can be a value from 0 to $(2^n) - 1$, where *n* is the depth (z axis) of the pixmap. A pixmap can also be thought of as a stack of *n* bitmaps.

plane When a pixmap or window is thought of as a stack of bitmaps, each bitmap is called a *plane* or *bit plane*.

plane mask Graphics operations can be restricted to only affect a subset of bit planes of a destination. A *plane mask* is a bit mask describing which planes are to be modified, and it is stored in a graphics context.

pointer The pointing device attached to the cursor and tracked on the screens.

pointer grabbing A client can actively *grab* control of the pointer, and button and motion events will be sent to that client rather than the client to which the events would normally have been sent.

pointing device Typically a mouse or tablet, or some other device with effective dimensional motion. There is only one visible cursor defined by the core protocol, and it tracks whatever pointing device is attached as the pointer.

pop-up One of the uses of a top-level shell (for example, a menu that pops up when a command button contact is activated). Setting the **state** of a shell to **:mapped** is sometimes referred to as *mapping* or *popping up* the shell. Setting the **state** of a shell to **:withdrawn** or **:iconic** is sometimes referred to as *unmapping* or *popping down* the shell.

property Windows can have associated *properties*, consisting of a name, a type, a data format, and some data. The protocol places no interpretation on properties; they are intended as a general-purpose naming mechanism for clients. For example, clients might share information such as resize hints, program names, and icon formats with a window manager via properties.

property list The list of properties that have been defined for a window.

pseudo color A class of colormap in which a pixel value indexes the colormap to produce independent red, green, and blue values. That is, the colormap is viewed as an array of triples (RGB values). The RGB values can be changed dynamically.

k

key grabbing Keys on the keyboard can be passively *grabbed* by a client. When the key is pressed, the keyboard is then actively grabbed by the client.

keyboard grabbing A client can actively *grab* control of the keyboard, and key events will be sent to that client rather than the client to which the events would normally have been sent.

keysym An encoding of a symbol on a keycap on a keyboard.

m

managed A contact under geometry management control.

mapped A window is said to be *mapped* if a map call has been performed on it. Unmapped windows and their inferiors are never viewable or visible.

modifier keys SHIFT, CONTROL, META, SUPER, HYPER, ALT, Compose, Apple, CAPS LOCK, Shift Lock, and similar keys are called *modifier keys*.

monochrome A special case of static gray, in which there are only two colormap entries.

o

obscure A window is *obscured* if some other window obscures it. For example, window A obscures window B if:

- Both windows are viewable **:input-output** windows
- Window A is higher in the global stacking order than window B
- The rectangle defined by the outside edges of window A intersects the rectangle defined by the outside edges of window B

Notice that window borders are included in the calculation, and that a window can be obscured and yet still have visible regions. See occlude (there is a fine distinction between obscure and occlude).

occlude A window is *occluded* if some other window occludes it. For example, window A occludes window B if:

- Both windows are mapped
- Window A is higher in the global stacking order than window B
- The rectangle defined by the outside edges of window A intersects the rectangle defined by the outside edges of window B

Notice that window borders are included in the calculation. See obscure (there is a fine distinction between occlude and obscure).

grab Keyboard keys, the keyboard, pointer buttons, the pointer, and the server can be *grabbed* for exclusive use by a client. In general, these facilities are not intended to be used by normal applications but are intended for various input and window managers to implement various styles of user interfaces.

gcontext Shorthand for graphics context.

graphics context Various information for graphics output is stored in a *graphics context* (or gcontext), such as foreground pixel, background pixel, line width, clipping region, and so forth. A graphics context can only be used with drawables that have the same root and the same depth as the graphics context.

gray scale A degenerate case of pseudo color, in which the red, green, and blue values in any given colormap entry are equal, thus producing shades of gray. The gray values can be changed dynamically.

h

hot spot A cursor has an associated *hot spot* that defines a point in the cursor that corresponds to the coordinates reported for the pointer.

i

identifier Each resource has an *identifier*, a unique value associated with it that clients use to name the resource. An identifier can be used over any connection to name the resource.

inferiors All of the subwindows nested below a window: the children, the children's children, and so on.

initialization See contact initialization.

input event See event.

input focus Normally a window defining the scope for processing of keyboard input. If a generated keyboard event would normally be reported to this window or one of its inferiors, the event is reported normally; otherwise, the event is reported with respect to the focus window. The input focus also can be set such that all keyboard events are discarded and that the focus window is dynamically taken to be the root window of whatever screen the pointer is on at each keyboard event.

input-only window A window that cannot be used for graphics requests. *input-only* windows are invisible, and can be used to control such things as cursors, input event generation, and grabbing. *input-only* windows cannot have *input/output* windows as inferiors.

input/output window The normal kind of opaque window, used for both input and output. Input/output windows can have both *input/output* and input-only windows as inferiors.

insensitivity See sensitivity.

interactive-stream A contact subclass designed to integrate CLUE with the conventional stream-based I/O of Common Lisp.

event compression Ignoring (or compressing) certain redundant input events. Compression of redundant events is controlled by the class slots **compress-exposures** and **compress-motion**, which are shared by all instances of a contact class.

event loop The fundamental application control structure: wait for an event, figure out how to handle it, process the event, then go back and wait for the next one. In CLUE, the event loop is implemented using the **process-next-event** function.

event mask Events are requested relative to a window. The set of event types a client requests relative to a window are described using an *event mask*.

event propagation Device-related events *propagate* from the source window to ancestor windows until some client has expressed interest in handling that type of event, or until the event is discarded explicitly.

event specification A notation for describing a certain sort of event. CLUE input processing consists of matching an event with an event specification found in a contact's **event-translations** slot and then calling actions associated with the matching event specification.

event synchronization Certain race conditions are possible when demultiplexing device events to clients (in particular deciding where pointer and keyboard events should be sent when in the middle of window management operations). The event synchronization mechanism allows synchronous processing of device events.

event source The smallest window containing the pointer is the *source* of a device related event.

event translation The process of determining which contact action functions will be executed. An event translation is a list found in a contact's **event-translations** slot associating an event specification with one or more action names. Also see class event translations.

exposure event Servers do not guarantee to preserve the contents of windows when windows are obscured or reconfigured. *Exposure* events are sent to clients to inform them when contents of regions of windows have been lost.

extension Named *extensions* to the core protocol can be defined to extend the system. Extension to output requests, resources, and event types are all possible, and expected.

f

focus window Another term for the input focus.

font A matrix of glyphs (typically characters). The protocol does no translation or interpretation of character sets. The client simply indicates values used to index the glyph array. A font contains additional metric information to determine inter-glyph and inter-line spacing.

g

geometry management The process whereby a composite controls the geometrical properties of its child contacts; the composite is referred to as the geometry manager.

glyph An image, typically of a character, in a font.

containment A window contains the pointer if the window is viewable and the hot spot of the cursor is within a visible region of the window or a visible region of one of its inferiors. The border of the window is included as part of the window for containment. The pointer is in a window if the window contains the pointer but no inferior contains the pointer.

content The single child of a shell. The basic geometry management policy implemented by the **shell** class constrains a shell and its content to have the same width and height; size changes to one are automatically applied to the other.

coordinate system The coordinate system has x horizontal and y vertical, with the origin [0, 0] at the upper left. Coordinates are discrete and are in terms of pixels. Each window and pixmap has its own coordinate system. For a window, the origin is at the inside upper left, inside the border.

cursor The visible shape of the pointer on a screen. It consists of a hot-spot, a source bitmap, a shape bitmap, and a pair of colors. The cursor defined for a window controls the visible appearance when the pointer is in that window.

d

depth The depth of a window or pixmap is number of bits per pixel it has. The depth of a graphics context is the depth of the drawables it can be used in conjunction with for graphics output.

descendant If W is an inferior of A, then W is a *descendant* of A.

device Keyboards, mice, tablets, track-balls, button boxes, and so forth, are all collectively known as input *devices*. The core protocol only deals with two devices: the keyboard and the pointer.

direct color A class of colormap in which a pixel value is decomposed into three separate subfields for indexing. One subfield indexes an array to produce red intensity values, the second subfield indexes a second array to produce blue intensity values, and the third subfield indexes a third array to produce green intensity values. The RGB values can be changed dynamically.

dispatching an event The process of finding the appropriate contact and its actions.

double-click A sequence of two clicks of the same button in rapid succession.

drawable Both windows and pixmaps can be used as sources and destinations in graphics operations. These are collectively known as *drawables*. However, an **:input-only** window cannot be used as a source or destination in a graphics operation.

e

event Clients receive information asynchronously via *events*. These events can be either asynchronously generated from devices, or generated as side effects of client requests. Events are grouped into types; events are never sent to a client by the server unless the client has specifically asked to be informed of that type of event, but clients can force events to be sent to other clients. Events are typically reported relative to a window.

class event translations — Event translations that belong to all instances of a contact class. A class event translation is created by the **defevent** macro.

class resources — Resources defined for each instance of a contact class. Also see constraint resources.

click — A **:button-press** event followed immediately by a **:button-release** event for the same button, with no intervening change in pointer position or modifier key state.

client — An application program connects to the window system server by some interprocess communication (IPC) path, such as a TCP connection or a shared memory buffer. This program is referred to as a *client* of the window system server. More precisely, the client is the IPC path itself. A program with multiple paths open to the server is viewed as multiple clients by the protocol. Resource lifetimes are controlled by connection lifetimes, not by program lifetimes.

clipping regions — In a graphics context, a bitmap or list of rectangles can be specified to restrict output to a particular region of the window. The image defined by the bitmap or rectangles is called a *clipping region*.

colormap — A set of entries defining color values. The colormap associated with a window is used to display the contents of the window. Each pixel value indexes the colormap to produce RGB values that drive the guns of a monitor. Depending on hardware limitations, one or more colormaps can be installed at one time, such that windows associated with those maps display with correct colors.

composite — A subclass of **contact** representing contacts that are the parents of other contacts. A composite provides geometry management and input focus management services for the contacts that are its children.

complete resource class — A list of symbols containing the class of the contact, the class of the contact's **parent** (and so on), and the class of the **contact-display** to which the contact belongs. The complete resource class is one of the two items used as a key by a CLUE application in order to access a contact resource value in a resource database.

complete resource name — A list of symbols containing the **name** of the contact, the **name** of the contact's **parent** (and so on), and the name of the **contact-display** to which the contact belongs. The complete resource name is one of the two items used as a key by a CLUE application in order to access a contact resource value in a resource database.

connection — The IPC path between the server and client program. A client program typically has one connection to the server over which requests and events are sent.

constraint resources — Resources defined for each child belonging to a member of a composite class. Constraint resources are typically used to control the parent's geometry management policy. Also see class resources.

contact — The basic CLUE object for programming a user interface.

contact-display — The CLUE object type that represents a connection to an X server and that supports an event loop for application input.

contact initialization — The process of collecting initial values for all contact attributes. No server resources (windows and so on) are actually allocated until contact realization.

contact realization — The process of allocating contact resources. This process completes contact creation.

GLOSSARY

a

access control list — X maintains a list of hosts from which client programs can be run. By default, only programs on the local host can use the display, plus any hosts specified in an initial list read by the server. This *access control list* can be changed by clients on the local host. Some server implementations can also implement other authorization mechanisms in addition to or in place of this mechanism. The action of this mechanism can be conditional based on the authorization protocol name and data received by the server at connection setup.

action — A function that is designed to handle an input event. CLUE input processing consists of matching an event with an event specification found in a contact's **event-translations** slot and then calling actions associated with the matching event specification.

active grab — A grab is *active* when the pointer or keyboard is actually owned by the single grabbing client.

ancestors — If W is an inferior of A, then A is an *ancestor* of W.

atom — A unique ID corresponding to a string name. Atoms are used to identify properties, types, and selections.

b

backing store — When a server maintains the contents of a window, the off-screen saved pixels are known as a *backing store*.

before action — An action of a **contact-display** that is called when an event is dispatched to a contact, but before any other contact input processing is performed.

bit gravity — When a window is resized, the contents of the window are not necessarily discarded. It is possible to request the server to relocate the previous contents to some region of the window. This attraction of window contents for some location of a window is known as *bit gravity*.

bitmap — A pixmap of depth one.

button grabbing — Buttons on the pointer can be passively *grabbed* by a client. When the button is pressed, the pointer is then actively grabbed by the client.

byte order — For image (pixmap/bitmap) data, byte order is defined by the server, and clients with different native byte ordering must swap bytes as necessary. For all other parts of the protocol, the byte order is defined by the client, and the server swaps bytes as necessary.

c

callback — A function that represents a connection between a contact and the rest of an application program. A contact calls a callback function in order to report the results of the user interface component that it represents.

children — First-level subwindows of a window.