Introduction to X Windows

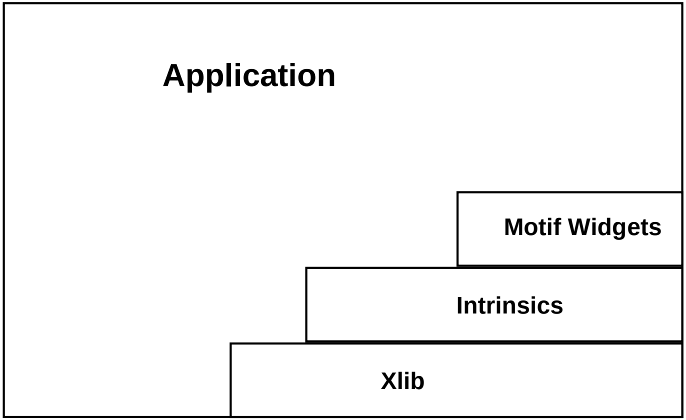
Professor Raymond Zavodnik

Fachhochschule Regensburg, Germany, April 7, 2012

What is X?

- X is a windowing system that promotes a portable graphics development/sharing environment over a network
- X uses the *client/server* architecture paradigm
  - The client can run anywhere on the network
  - The server posseses the graphics resources requested by the client
  - Everything runs in a multiprocessor environment
  - The X Protocol in an IEEE standard
- X is a *layered* system
- We begin with a discussion of Xlib

Layers of X



Server and Client

- A One-way Request
  - Goes from Client to Server
  - E.g. open a window
- A Round-trip Request
  - Client requests a resource from the Server
  - Server returns it over the network
  - For example, request the pixel coordinates of the mouse position
  - Is slower than a One-way
- Events are sent from the Server to the Client
- Events must be explcitly requested by the Client
- It is the job of the Server to manage all Client windows
- X Graphics operations are also carried out by the Server

Obtaining a Screen

- In X a **Screen** is a Graphics Terminal
- The Server manages many screens over the network
- A **Display** is a set of Screens with Keyboard and Mouse
- Obtaining a Screen results in a connection to the X Server
- See the Function `XOpenDisplay()`

Obtaining a Window

- **Windows** are hierarchically arranged
  - `root` window covers all screens
  - `root` window can be partially or completely covered by child windows
  - One child is usually “on top”

  item Windows have borders of 0 or more pixels width
- Each Window has its own pixel (integer) coordinate system, with origin in the upper left-hand corner
- A Window’s contents can become invalid. If this happens the Server sends the Client an `Expose` event
- It is the job of the client to react to this event

- A Window’s contents can be stored offscreen in the form of a **pixmap**
- Pixmaps and Windows are more or less interchangeable and are called **drawables**
- Create a Window on a Screen with `XCreateWindow()`
- Usually several **Attributes** must be set before the Window can be created
- Some of these are of type `XSetWindowAttributes`
- Others, e.g. **Visual** require a separate discussion
- The result is an *unmapped* Window
- Make it visible (only possible with `InputOutput` Window) with `XMapWindow()`

Window Attributes

- Before to create a Window you must specify its *attributes*
- The most relevant are:
  - The Window's **Background**
  - The Window's **Border**
  - A Colormap (later), default is `CopyFromParent`. This has to match with the Window's original Visual
  - An **Event Mask**:
    1. `KeyPress`,`KeyRelease`
    2. `ButtonPress`,`ButtonRelease`
    3. `PointerMotion`,`ButtonMotion` (several types)
  - Other attributes are of course `width, height, x,y` and `depth`

Visuals and Colormaps

- A **Visual** is a structure (of type `XVisualInfo`) that contains information about the system's color capabilities:
  1. `red_mask, green_mask, blue_mask`
  2. `bits_per_rgb`
  3. `colormapsize`
- Possible capabilities are:
  1. `StaticGray`
  2. `StaticColor`
  3. `PseudoColor`
  4. `DirectColor`
  5. `TrueColor`
- For OpenGL we will need much more!
- The function `XGetVisualInfo()` delivers a list of Visuals that match a specified depth and Screen class

Colormaps in X

- Colormaps translate color specifications to hardware colors in the **Framebuffer**
- In this course we will only consider `TrueColor`
- `PseudoColor` is very difficult in OpenGL because many shades of a color must be loaded into the Colortable for rendering
- Because of the network nature of X Colormaps should also be *shareable* to prevent color flashing
- Because your selected Visual is not usually default, a Colormap will have to be carefully selected
- OpenGL does support **Color Index Mode**, but this is only necessary for porting older applications that do not have `TrueColor`

Doing Graphics in X Windows

- You need a **Graphics Context**
- Later, we will need to extend this context for OpenGL
- A Graphics Context is like a crayon that is used for drawing on a Window
- This is a resource on the Server
- Includes parameters such as points, lines, rectangles, arcs, text, background and foreground colors, line-width, etc.
- A Graphics Context is created with the Xlib function `XCreateGC()`
- As with Window these attributes are specified either in a bit mask or in the `XGCValues` structure

Dealing with Events

- This far, programs are only non-interactive
- But we have **Events** on the Server that can be requested by the Client
- The Client specifies the Events of interest with an `event-mask` window attribute
- Events happen in an asynchronous fashion
- The program thus receives events as they occur (mostly), deciphers the event type (see above) and then reacts correspondingly in a `switch` statement
- Drawing on the Graphics Context usually happens when an Expose event has been received. This is important!

Programming Project

- Write a small program that opens a display and prints out information about that display, e.g. resolution, pixel-depth.
- Extend your program to create a Window that uses `TrueColor` and returns the pixel coordinates of a mouse click.
- Extend this program further to create a Graphics Context and draw a red rectangle with the black text "Hello, World" in the middle