```
/***********************************************************************
$Source: /afs/athena.mit.edu/user/c/crcraig/Tmp/Slider/RCS/Slider.c,v $
$Author: crcraig $  $Date: 90/05/18 18:42:59 $
$Revision: 1.5 $
***********************************************************************/

#include <X11/IntrinsicP.h>
#include <X11/StringDefs.h>
#include "SliderP.h"
#include "double.h"

/***********************************************************************
**                      Default geometry values                      **
***********************************************************************/

#define AtSlider_Default_Major_Axis_Length 100
#define AtSlider_Default_Grabber_Width     20

#define AXISPAD 5
/***********************************************************************
**             Forward function prototype declarations               **
***********************************************************************/

static void ClassInitialize(Widget);
static void Initialize(Widget, Widget);
static void Destroy(Widget);
static void Redisplay(AtSliderWidget, XEvent *, Region);
static void Resize(AtSliderWidget);
static void SetValues(AtSliderWidget, AtSliderWidget, AtSliderWidget);

void MoveGrabber(AtSliderWidget, XEvent *, String *, Cardinal *);
void StartDrag(AtSliderWidget, XEvent *, String *, Cardinal *);
void EndDrag(AtSliderWidget, XEvent *, String *, Cardinal *);
void DragGrabber(AtSliderWidget, XEvent *, String *, Cardinal *);

static void GetGCs(AtSliderWidget);
static void FreeGCs(AtSliderWidget);
static void CalcSize(AtSliderWidget);
static void CalcDefaultDimensions(AtSliderWidget, AtSliderWidget);
static Boolean PointInRect(int, int, int, int, int, int);
static void SetNewValue(AtSliderWidget, double);
static void DrawGrabber(AtSliderWidget, double, Boolean);
static void UpdateGrabber(AtSliderWidget, double);

/***********************************************************************
**                          AtSlider Resources                       **
***********************************************************************/

#define offset(field) XtOffset(AtSliderWidget, slider.field)
static XtResource resources[] = {
  { XtNvalue, XtCValue, XtRDouble, sizeof(double),
      offset(value), XtRString, "0.0" },
  { XtNallowInput, XtCAllowInput, XtRBoolean, sizeof(Boolean),
      offset(allowInput), XtRImmediate, (caddr_t) True },
  { XtNmarkerWidth, XtCMarkerWidth, XtRShort, sizeof(short),
      offset(markerWidth), XtRImmediate, (caddr_t) 0 },
  { XtNmarkerHeight, XtCMarkerHeight, XtRShort, sizeof(short),
      offset(markerHeight), XtRImmediate, (caddr_t) 12 },
  { XtNvalueChangedCallback, XtCValueChangedCallback, XtRCallback,
      sizeof(XtCallbackList), offset(valueChangedCallback),
      XtRCallback, NULL },
};
#undef offset
```

```
/***********************************************************************
**             AtSlider Actions and default Translations              **
***********************************************************************/

static char defaultTranslations[] = "<Btn2Down>:   MoveGrabber() \n\
                                     <Btn1Down>:   StartDrag() \n\
                                     <Btn1Motion>: DragGrabber() \n\
                                     <Btn1Up>:     EndDrag() \n";

static XtActionsRec actions[] =
{
  {"MoveGrabber", MoveGrabber},
  {"StartDrag", StartDrag},
  {"EndDrag", EndDrag},
  {"DragGrabber", DragGrabber},
};

/***********************************************************************
**                     The AtSlider Class Record                      **
***********************************************************************/

AtSliderClassRec atSliderClassRec = {
  { /* core fields */
    /* superclass               */      (WidgetClass) &widgetClassRec,
    /* class_name               */      "AtSlider",
    /* widget_size              */      sizeof(AtSliderRec),
    /* class_initialize         */      ClassInitialize,
    /* class_part_initialize    */      NULL,
    /* class_inited             */      FALSE,
    /* initialize               */      Initialize,
    /* initialize_hook          */      NULL,
    /* realize                  */      XtInheritRealize,
    /* actions                  */      actions,
    /* num_actions              */      XtNumber(actions),
    /* resources                */      resources,
    /* num_resources            */      XtNumber(resources),
    /* xrm_class                */      NULLQUARK,
    /* compress_motion          */      TRUE,
    /* compress_exposure        */      TRUE,
    /* compress_enterleave      */      TRUE,
    /* visible_interest         */      FALSE,
    /* destroy                  */      Destroy,
    /* resize                   */      Resize,
    /* expose                   */      Redisplay,
    /* set_values               */      SetValues,
    /* set_values_hook          */      NULL,
    /* set_values_almost        */      XtInheritSetValuesAlmost,
    /* get_values_hook          */      NULL,
    /* accept_focus             */      NULL,
    /* version                  */      XtVersion,
    /* callback_private         */      NULL,
    /* tm_table                 */      defaultTranslations,
    /* query_geometry           */      NULL,
    /* display_accelerator      */      NULL,
    /* extension                */      NULL,
  },

};

WidgetClass atSliderWidgetClass = (WidgetClass) &atSliderClassRec;



#define Axis(widget, field) (widget->slider.axis->axis.field)
```

```
/*********************************************************************
**                    Class Initialize Method                       **
*********************************************************************/

static void ClassInitialize(Widget w)

{
  AtRegisterDoubleConverter();
  AtRegisterFontSizeConverter();
}

/*********************************************************************
**                   Instance Initialize Method                     **
*********************************************************************/

static void Initialize(Widget req, Widget new)

{
  AtSliderWidget rw = (AtSliderWidget) req;
  AtSliderWidget nw = (AtSliderWidget) new;

  nw->slider.axis = (AtAxisObject) XtCreateWidget("axis", atAxisObjectClass,
                                                  nw, NULL, 0);
  GetGCs(nw);

  if ((rw->core.width == 0) || (rw->core.height == 0))
    CalcDefaultDimensions(rw, nw);

  CalcSize(nw);
}

/*********************************************************************
**                          Destroy Method                          **
*********************************************************************/

static void Destroy(Widget sw)

{
  FreeGCs((AtSliderWidget) sw);
}

/*********************************************************************
**                         Redisplay Method                         **
*********************************************************************/

static void Redisplay(AtSliderWidget sw, XEvent *event, Region region)

{
  int axisside, otherside;
  XPoint points[6];

/* Clear window and draw axis (scale) */
  XClearWindow(XtDisplay(sw), XtWindow(sw));
  AtAxisDraw(XtDisplay(sw), XtWindow(sw), sw->slider.axis);

/* Set up list of points for border */
  points[0].x = sw->slider.axis_x1;
  points[0].y = sw->slider.axis_y1;
  if (Axis(sw, vertical))  {
    axisside = (Axis(sw, mirror) ? sw->slider.slider_x2 :
                   sw->slider.slider_x1);
    otherside = (Axis(sw, mirror) ? sw->slider.slider_x1 :
                   sw->slider.slider_x2);
    points[1].x = axisside;
    points[1].y = sw->slider.slider_y1;
    points[2].x = otherside;
    points[2].y = sw->slider.slider_y1;
    points[3].x = otherside;
    points[3].y = sw->slider.slider_y2;
    points[4].x = axisside;
    points[4].y = sw->slider.slider_y2;
  }
  else  {
    axisside = (Axis(sw, mirror) ? sw->slider.slider_y1 :
                   sw->slider.slider_y2);
    otherside = (Axis(sw, mirror) ? sw->slider.slider_y2 :
                   sw->slider.slider_y1);
    points[1].x = sw->slider.slider_x1;
    points[1].y = axisside;
    points[2].x = sw->slider.slider_x1;
    points[2].y = otherside;
    points[3].x = sw->slider.slider_x2;
    points[3].y = otherside;
    points[4].x = sw->slider.slider_x2;
    points[4].y = axisside;
  }
  points[5].x = sw->slider.axis_x2;
  points[5].y = sw->slider.axis_y2;

  /* Draw the border lines and grabber */
  if (XtIsRealized(sw))  {
    XDrawLines(XtDisplay(sw), XtWindow(sw), sw->slider.normalgc, points, 6,
               CoordModeOrigin);
    DrawGrabber(sw, sw->slider.value, True);
  }
}

/*********************************************************************
**                          Resize Method                           **
*********************************************************************/

static void Resize(AtSliderWidget sw)

{
  CalcSize(sw);
}

/*********************************************************************
**                         SetValues Method                         **
*********************************************************************/

static void SetValues(AtSliderWidget cur, AtSliderWidget req,
                      AtSliderWidget new)

{

}

/**********************************************************************/
/**********************************************************************/
/*                          Action Routines                           */
/**********************************************************************/
/**********************************************************************/

static Boolean currentdrag = False;

/**********************************************************************/
```

```
void MoveGrabber(AtSliderWidget sw, XEvent *event, String *params,
                 Cardinal *num_params)

{
  int pixelval;
  double newval;

  if ((event->type == ButtonPress) && sw->slider.allowInput &&
      PointInRect(event->xbutton.x, event->xbutton.y, sw->slider.slider_x1,
                  sw->slider.slider_y1, sw->slider.slider_x2,
                  sw->slider.slider_y2))  {
    pixelval = (Axis(sw, vertical) ? event->xbutton.y : event->xbutton.x);
    newval = AtScalePixelToUser(Axis(sw, scale), pixelval);
    DrawGrabber(sw, sw->slider.value, False);
    if (newval < Axis(sw, min))
      newval = Axis(sw, min);
    else if (newval > Axis(sw, max))
      newval = Axis(sw, max);

    if (sw->slider.value != newval)  {
      DrawGrabber(sw, sw->slider.value, False);
      SetNewValue(sw, newval);
      DrawGrabber(sw, newval, True);
    }
  }
}

/**********************************************************************/

void StartDrag(AtSliderWidget sw, XEvent *event, String *params,
               Cardinal *num_params)

{
  if ((event->type == ButtonPress) && sw->slider.allowInput)
    if (PointInRect(event->xbutton.x, event->xbutton.y, sw->slider.grabber_x1,
                    sw->slider.grabber_y1, sw->slider.grabber_x2,
                    sw->slider.grabber_y2))
      currentdrag = True;
}

/**********************************************************************/

void EndDrag(AtSliderWidget sw, XEvent *event, String *params,
             Cardinal *num_params)

{
  currentdrag = False;
}

/**********************************************************************/

void DragGrabber(AtSliderWidget sw, XEvent *event, String *params,
                 Cardinal *num_params)

{
  int pixelval;
  double newval;

  if ((event->type == MotionNotify) && sw->slider.allowInput &&
      (PointInRect(event->xmotion.x, event->xmotion.y, sw->slider.grabber_x1,
                   sw->slider.grabber_y1, sw->slider.grabber_x2,
                   sw->slider.grabber_y2) || currentdrag))  {
    currentdrag = True;
    pixelval = (Axis(sw, vertical) ? event->xmotion.y : event->xmotion.x);
    newval = AtScalePixelToUser(Axis(sw, scale), pixelval);
    if (newval < Axis(sw, min))
      newval = Axis(sw, min);
    else if (newval > Axis(sw, max))
      newval = Axis(sw, max);

    if (sw->slider.value != newval)  {
      UpdateGrabber(sw, newval);
      SetNewValue(sw, newval);
    }
  }
}

/**********************************************************************/
/**********************************************************************/
/*                          Support Routines                          */
/**********************************************************************/
/**********************************************************************/

static void GetGCs(AtSliderWidget sw)

{
  XGCValues gcvalues;
  XtGCMask  valuemask;

  valuemask = GCForeground | GCBackground;
  gcvalues.foreground = Axis(sw, axisColor);
  gcvalues.background = sw->core.background_pixel;
  sw->slider.normalgc = XtGetGC(sw, valuemask, &gcvalues);

  gcvalues.foreground = sw->core.background_pixel;
  sw->slider.cleargc = XtGetGC(sw, valuemask, &gcvalues);
}

/**********************************************************************/

static void FreeGCs(AtSliderWidget sw)

{
  XtReleaseGC(sw, sw->slider.normalgc);
  XtReleaseGC(sw, sw->slider.cleargc);
}

/**********************************************************************/

static void CalcSize(AtSliderWidget sw)

{
  int length, width;

/* set grabber dimensions to fill widget */
  length = (Axis(sw, vertical) ? sw->core.height : sw->core.width);
  width = (Axis(sw, vertical) ? sw->core.width : sw->core.height);
  AtAxisComputeTicSpacing(sw->slider.axis, length);
  sw->slider.axis_width = AtAxisWidth(sw->slider.axis);
  sw->slider.grabber_width = width - sw->slider.axis_width - 4;
  sw->slider.grabup = sw->slider.markerHeight / 2;
  sw->slider.grabdown = sw->slider.markerHeight - sw->slider.grabup;

/* set locations for axis endpoints, calculate rectangles */
  if (Axis(sw, vertical))  {
    sw->slider.axis_x2 = (Axis(sw, mirror) ? width - sw->slider.axis_width :
                          sw->slider.axis_width);
```

```
    sw->slider.axis_y2 =  sw->core.height - sw->slider.grabdown -
                          AXISPAD;
    sw->slider.axis_x1 = sw->slider.axis_x2;
    sw->slider.axis_y1 = AXISPAD + sw->slider.grabup;

    sw->slider.slider_x1 = (Axis(sw, mirror) ? 1 : sw->slider.axis_x1);
    sw->slider.slider_y1 = sw->slider.axis_y1 - sw->slider.grabup - 1;
    sw->slider.slider_x2 = (Axis(sw, mirror) ? sw->slider.axis_x2 :
                             sw->core.width - 1);
    sw->slider.slider_y2 = sw->slider.axis_y2 + sw->slider.grabdown;
  }
  else  {
    sw->slider.axis_x1 = AXISPAD + sw->slider.grabdown;
    sw->slider.axis_y1 = (Axis(sw, mirror) ? sw->slider.axis_width :
                           sw->core.height - sw->slider.axis_width);
    sw->slider.axis_x2 = sw->core.width - sw->slider.grabup - AXISPAD;
    sw->slider.axis_y2 = sw->slider.axis_y1;

    sw->slider.slider_x1 = sw->slider.axis_x1 - sw->slider.grabup - 1;
    sw->slider.slider_y1 = (Axis(sw, mirror) ? sw->slider.axis_y1 : 1);
    sw->slider.slider_x2 = sw->slider.axis_x2 + sw->slider.grabdown;
    sw->slider.slider_y2 = (Axis(sw, mirror) ? sw->core.height - 1 :
                             sw->slider.axis_y2);
  }
  AtAxisSetLocation(sw->slider.axis, sw->slider.axis_x1, sw->slider.axis_y1,
                    sw->slider.axis_x2, sw->slider.axis_y2);
}

/**********************************************************************/

static void CalcDefaultDimensions(AtSliderWidget rw, AtSliderWidget nw)

{
  if (Axis(nw, vertical))  {
    if (rw->core.height == 0)
      nw->core.height = AtSlider_Default_Major_Axis_Length;
    if (rw->core.width == 0)  {
      AtAxisComputeTicSpacing(nw->slider.axis, nw->core.height);
      nw->core.width = AtSlider_Default_Grabber_Width +
        AtAxisWidth(nw->slider.axis) + 3;
    }
  }
  else  {
    if (rw->core.width == 0)
      nw->core.width = AtSlider_Default_Major_Axis_Length;
    if (rw->core.height == 0)  {
      AtAxisComputeTicSpacing(nw->slider.axis, nw->core.width);
      nw->core.height = AtSlider_Default_Grabber_Width +
        AtAxisWidth(nw->slider.axis) + 3;
    }
  }
}

/**********************************************************************/

static Boolean PointInRect(int i, int j, int x1, int y1, int x2, int y2)

{
  if ((i < x1) || (j < y1))
    return False;
  if ((i > x2) || (j > y2))
    return False;
  return True;
}
```

```
/**********************************************************************/

static void SetNewValue(AtSliderWidget sw, double newval)

{
  static double val;

  val = newval;
  sw->slider.value = newval;
  XtCallCallbacks(sw, XtNvalueChangedCallback, (caddr_t) &newval);
}

/**********************************************************************/

static void DrawGrabber(AtSliderWidget sw, double value, Boolean onoff)

{
  int userpix, x, y;
  unsigned int width, height;

  userpix = AtScaleUserToPixel(Axis(sw, scale), value);

  if (Axis(sw, vertical))  {
    x = sw->slider.slider_x1 + 2;
    y = userpix - sw->slider.grabup;
    width = sw->slider.grabber_width;
    height = sw->slider.markerHeight;
  }
  else  {
    x = userpix - sw->slider.grabup;
    height = sw->slider.grabber_width;
    y = sw->slider.slider_y2 - 1 - sw->slider.grabber_width;
    width = sw->slider.markerHeight;
  }

  sw->slider.grabber_x1 = x;
  sw->slider.grabber_y1 = y;
  sw->slider.grabber_x2 = x + width;
  sw->slider.grabber_y2 = y + height;

  if (XtIsRealized(sw))
    XFillRectangle(XtDisplay(sw), XtWindow(sw),
                   (onoff ? sw->slider.normalgc : sw->slider.cleargc),
                   x, y, width, height);
}

/**********************************************************************/

#define max(a, b)  ((a) > (b) ? (a) : (b))
#define min(a, b)  ((a) < (b) ? (a) : (b))

static void UpdateGrabber(AtSliderWidget sw, double newval)

{
  int old, new, min, max, x, y;
  unsigned int width, height;

  old = AtScaleUserToPixel(Axis(sw, scale), sw->slider.value);
  new = AtScaleUserToPixel(Axis(sw, scale), newval);
  min = min(old, new);
  max = max(old, new);

  if ((max - min) < sw->slider.markerHeight)
```

```
    if (Axis(sw, vertical))  {
      x = sw->slider.slider_x1 + 2;
      y = min - sw->slider.grabup;
      width = sw->slider.grabber_width;
      height = max - min;

      if (XtIsRealized(sw))
        XFillRectangle(XtDisplay(sw), XtWindow(sw),
                       (old == max ? sw->slider.normalgc : sw->slider.cleargc),
                       x, y, width, height);

      y = min + sw->slider.grabdown;
      if (XtIsRealized(sw))
        XFillRectangle(XtDisplay(sw), XtWindow(sw),
                       (old == max ? sw->slider.cleargc : sw->slider.normalgc),
                       x, y, width, height);
      sw->slider.grabber_y1 = new - sw->slider.grabup;
      sw->slider.grabber_y2 = new + sw->slider.grabdown;
    }

    else  {
      x = min + sw->slider.grabdown;
      y = sw->slider.slider_y2 - 1 - sw->slider.grabber_width;
      width = max - min;
      height = sw->slider.grabber_width;

      if (XtIsRealized(sw))
        XFillRectangle(XtDisplay(sw), XtWindow(sw),
                       (old == min ? sw->slider.normalgc : sw->slider.cleargc),
                       x, y, width, height);

      x = min - sw->slider.grabup;
      if (XtIsRealized(sw))
        XFillRectangle(XtDisplay(sw), XtWindow(sw),
                       (old == min ? sw->slider.cleargc : sw->slider.normalgc),
                       x, y, width, height);

      sw->slider.grabber_x1 = new - sw->slider.grabup;
      sw->slider.grabber_x2 = new + sw->slider.grabdown;
    }

  else  {
    DrawGrabber(sw, sw->slider.value, False);
    DrawGrabber(sw, newval, True);
  }
}
```