Subject: Re: BufferPainter Clip Crash - Fatal error: Invalid memory access!
Posted by devilsclaw on Thu, 13 Apr 2023 14:36:59 GMT
View Forum Message <> Reply to Message

I re-wrote how the mouse handler works by overloading it and it fixed the problem for me

below is sample code of what I did. I made it work more like how the java mouse handling works

```
//Header
enum {
  MOUSE_NONE   = (0 << 0),
  MOUSE_LEFT   = (1 << 1),
  MOUSE_RIGHT  = (1 << 2),
  MOUSE_MIDDLE = (1 << 3),
  MOUSE_DOWN   = (1 << 4),
  MOUSE_UP     = (1 << 5),
  MOUSE_DOUBLE = (1 << 6),
  MOUSE_TRIPLE = (1 << 7),
  MOUSE_DRAG   = (1 << 8),
  MOUSE_MOVE   = (1 << 9),
};

virtual Upp::Image MouseEvent(int event, Upp::Point p, int zdelta, Upp::dword keyflags);
virtual void MouseWheel(Upp::Point p, int zdelta, Upp::dword keyflags);
virtual void MouseClicked(Upp::Point p, Upp::dword keyflags, unsigned int type, int clicks);
virtual void MouseReleased(Upp::Point p, Upp::dword keyflags, unsigned int type);
virtual void MouseDrag(Upp::Point p, Upp::dword keyflags, unsigned int type);
virtual void MouseMove(Upp::Point p, Upp::dword keyflags);
virtual void MouseEnter(Upp::Point p, Upp::dword keyflags);
virtual void MouseLeave();

//CPP
void frm_network::MouseWheel(Upp::Point p, int zdelta, Upp::dword keyflags) {
}

void frm_network::MouseEnter(Upp::Point p, Upp::dword keyflags) {
}

void frm_network::MouseLeave() {
}

void frm_network::MouseDrag(Upp::Point p, Upp::dword keyflags,  unsigned int type) {
}

void frm_network::MouseClicked(Upp::Point pos, Upp::dword keyflags, unsigned int type, int
clicks) {
}
```

```
void frm_network::MouseReleased(Upp::Point p, Upp::dword keyflags, unsigned int type) {
}

void frm_network::MouseMove(Upp::Point p, Upp::dword keyflags) {
}

Upp::Image frm_network::MouseEvent(int event, Upp::Point p, int zdelta, Upp::dword keyflags) {
  int mouse_event = event & ~(LEFT | RIGHT | MIDDLE);
  Upp::dword mouse_flags =  keyflags & (Upp::K_MOUSELEFT | Upp::K_MOUSERIGHT |
Upp::K_MOUSEMIDDLE | Upp::K_MOUSEDOUBLE | Upp::K_MOUSETRIPLE);
  if(mouse_event == REPEAT || mouse_event == CURSORIMAGE) {
    goto exit;
  }

  switch(mouse_event) {
    case MOUSEWHEEL: {
      MouseWheel(p, zdelta, keyflags);
      goto exit;
    }
    case MOUSEENTER: {
      MouseEnter(p, keyflags);
      goto exit;
    }
    case MOUSELEAVE: {
      MouseLeave();
      goto exit;
    }
  }

  if(mouse_event != UP && keyflags == 0) {
    mouse_state = 0;
  }

  if((mouse_state & MOUSE_UP)) {
    mouse_state = 0;
  }

  //Add rejection of other buttons once one is in use
  if((mouse_state & MOUSE_LEFT) != 0 && (keyflags & (Upp::K_MOUSERIGHT |
Upp::K_MOUSEMIDDLE)) != 0) {
    goto exit;
  }
  if((mouse_state & MOUSE_RIGHT) != 0 && (keyflags & (Upp::K_MOUSELEFT |
Upp::K_MOUSEMIDDLE)) != 0) {
    goto exit;
  }
  if((mouse_state & MOUSE_MIDDLE) != 0 && (keyflags & (Upp::K_MOUSERIGHT |
```

```
Upp::K_MOUSELEFT)) != 0) {
    goto exit;
  }

  if((keyflags & Upp::K_MOUSELEFT) == Upp::K_MOUSELEFT) {
    mouse_state |= MOUSE_LEFT;
  } else if((keyflags & Upp::K_MOUSERIGHT) == Upp::K_MOUSERIGHT) {
    mouse_state |= MOUSE_RIGHT;
  } else if((keyflags & Upp::K_MOUSEMIDDLE) == Upp::K_MOUSEMIDDLE) {
    mouse_state |= MOUSE_MIDDLE;
  }

  if((keyflags & Upp::K_MOUSETRIPLE) == Upp::K_MOUSETRIPLE) {
    mouse_state &= ~(MOUSE_DOWN | MOUSE_UP | MOUSE_DOUBLE | MOUSE_MOVE |
MOUSE_DRAG);
    mouse_state |= MOUSE_TRIPLE;
  } else if((keyflags & Upp::K_MOUSEDOUBLE) == Upp::K_MOUSEDOUBLE) {
    mouse_state &= ~(MOUSE_DOWN | MOUSE_UP  | MOUSE_MOVE | MOUSE_DRAG);
    mouse_state |= MOUSE_DOUBLE;
  } else if(mouse_event == DOWN) {
    mouse_state &= ~(MOUSE_UP | MOUSE_MOVE | MOUSE_MOVE | MOUSE_DRAG);
    mouse_state |= MOUSE_DOWN;
  } else if(mouse_event == UP) {
    mouse_state &= ~(MOUSE_DOWN | MOUSE_DOUBLE | MOUSE_TRIPLE | MOUSE_MOVE |
MOUSE_DRAG);
    mouse_state |= MOUSE_UP;
  } else if(mouse_event == DRAG) {
    mouse_state &= ~(MOUSE_DOWN | MOUSE_MOVE | MOUSE_UP);
    mouse_state |= MOUSE_DRAG;
  } else if((mouse_state & MOUSE_DRAG) != MOUSE_DRAG) {
    mouse_state |= MOUSE_MOVE;
  }

  if(mouse_state & MOUSE_DRAG) {
    MouseDrag(p, keyflags, mouse_state & (MOUSE_LEFT | MOUSE_RIGHT |
MOUSE_MIDDLE));
  } else if((mouse_state & MOUSE_DOWN) && (mouse_state & MOUSE_MOVE) == 0) {
    MouseClicked(p, keyflags, mouse_state & (MOUSE_LEFT | MOUSE_RIGHT |
MOUSE_MIDDLE), 1);
  } else if((mouse_state & MOUSE_DOUBLE) && (mouse_state & MOUSE_MOVE) == 0) {
    MouseClicked(p, keyflags, mouse_state & (MOUSE_LEFT | MOUSE_RIGHT |
MOUSE_MIDDLE), 2);
  } else if((mouse_state & MOUSE_TRIPLE) && (mouse_state & MOUSE_MOVE) == 0) {
    MouseClicked(p, keyflags, mouse_state & (MOUSE_LEFT | MOUSE_RIGHT |
MOUSE_MIDDLE), 3);
  } else if(mouse_state & MOUSE_UP) {
    MouseReleased(p, keyflags, mouse_state & (MOUSE_LEFT | MOUSE_RIGHT |
MOUSE_MIDDLE));
```

```
  } else if(mouse_state & MOUSE_MOVE) {
    MouseMove(p, keyflags);
  }

exit:
  return Upp::Image::Arrow();
}
```