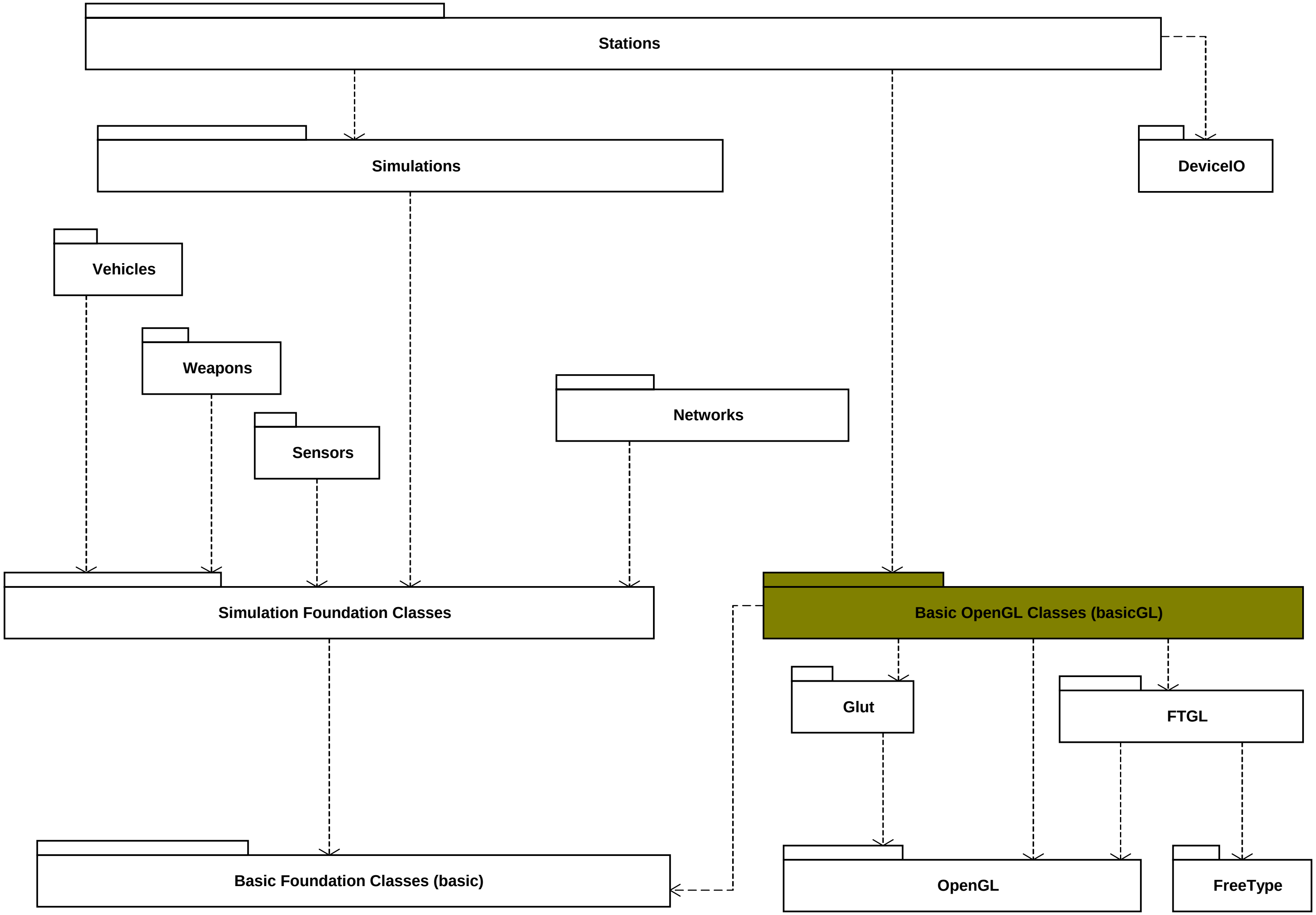
OpenEaagles Simulation Framework

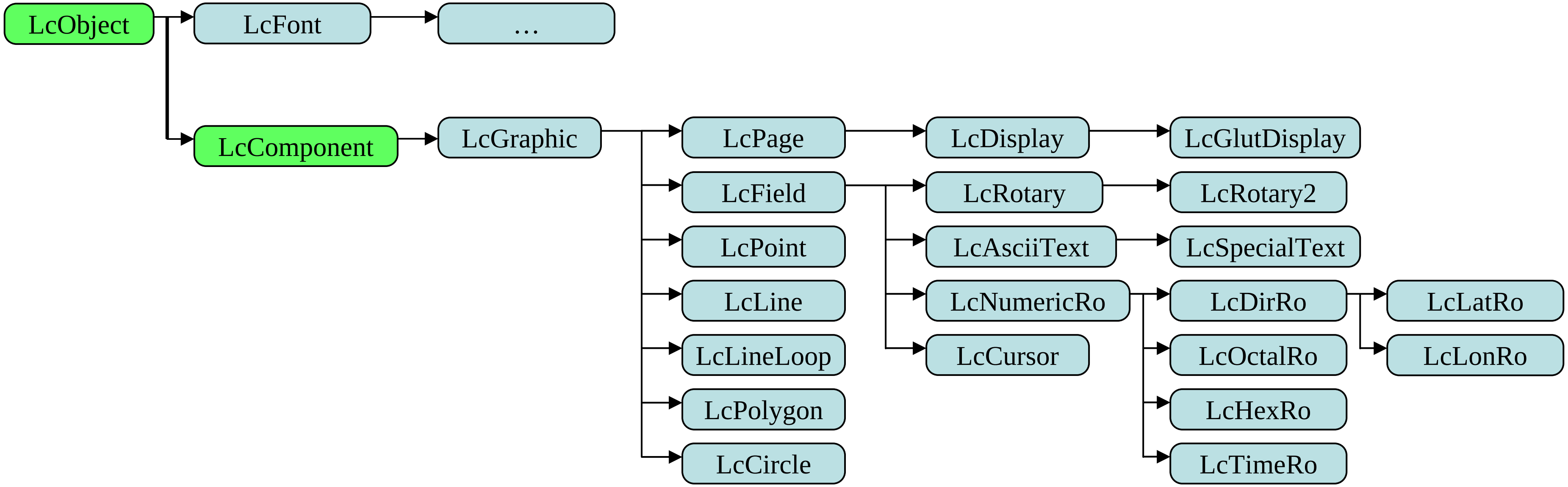
BasicGL Foundation Classes & Instrument Library

Revised: Nov 14, 2006

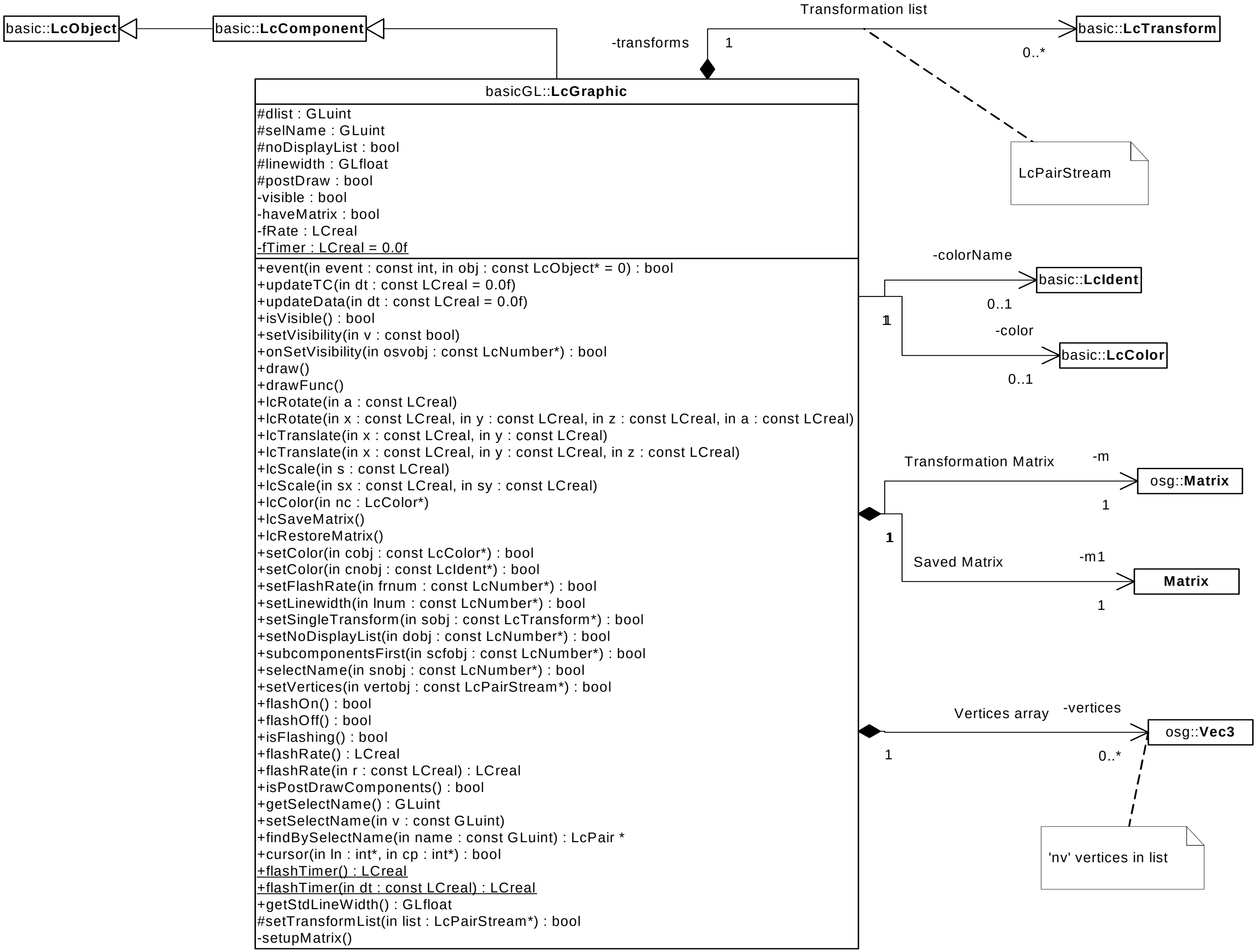
Software Layers &Packages


Stations
Simulations
DeviceIO
Vehicles
Weapons
Sensors
Networks
Simulation Foundation Classes
Basic OpenGL Classes (basicGL)
Glut
FTGL
Basic Foundation Classes (basic)
OpenGL
FreeType

BasicGL
Class Hierarchy



LcGraphic Station Structure



basic::**LcObject**

basic::**LcComponent**

Transformation list

-transforms

1

basic::**LcTransform**

0..*

LcPairStream

basicGL::**LcGraphic**

#dlist : GLuint
#selName : GLuint
#noDisplayList : bool
#linewidth : GLfloat
#postDraw : bool
-visible : bool
-haveMatrix : bool
-fRate : LCreal
-fTimer : LCreal = 0.0f

+event(in event : const int, in obj : const LcObject* = 0) : bool
+updateTC(in dt : const LCreal = 0.0f)
+updateData(in dt : const LCreal = 0.0f)
+isVisible() : bool
+setVisibility(in v : const bool)
+onSetVisibility(in osvobj : const LcNumber*) : bool
+draw()
+drawFunc()
+lcRotate(in a : const LCreal)
+lcRotate(in x : const LCreal, in y : const LCreal, in z : const LCreal, in a : const LCreal)
+lcTranslate(in x : const LCreal, in y : const LCreal)
+lcTranslate(in x : const LCreal, in y : const LCreal, in z : const LCreal)
+lcScale(in s : const LCreal)
+lcScale(in sx : const LCreal, in sy : const LCreal)
+lcColor(in nc : LcColor*)
+lcSaveMatrix()
+lcRestoreMatrix()
+setColor(in cobj : const LcColor*) : bool
+setColor(in cnobj : const LcIdent*) : bool
+setFlashRate(in frnum : const LcNumber*) : bool
+setLinewidth(in lnum : const LcNumber*) : bool
+setSingleTransform(in sobj : const LcTransform*) : bool
+setNoDisplayList(in dobj : const LcNumber*) : bool
+subcomponentsFirst(in scfobj : const LcNumber*) : bool
+selectName(in snobj : const LcNumber*) : bool
+setVertices(in vertobj : const LcPairStream*) : bool
+flashOn() : bool
+flashOff() : bool
+isFlashing() : bool
+flashRate() : LCreal
+flashRate(in r : const LCreal) : LCreal
+isPostDrawComponents() : bool
+getSelectName() : GLuint
+setSelectName(in v : const GLuint)
+findBySelectName(in name : const GLuint) : LcPair *
+cursor(in ln : int*, in cp : int*) : bool
+flashTimer() : LCreal
+flashTimer(in dt : const LCreal) : LCreal
+getStdLineWidth() : GLfloat
#setTransformList(in list : LcPairStream*) : bool
-setupMatrix()

-colorName

basic::**LcIdent**

0..1

1

-color

basic::**LcColor**

0..1

Transformation Matrix

-m

osg::**Matrix**

1

1

Saved Matrix

-m1

Matrix

1

Vertices array

-vertices

osg::**Vec3**

1

0..*

'nv' vertices in list

BasicGL – LcGraphic

- Purpose: Base class for the graphic objects. Provides lists of vertices and texture coordinates, transformation matrix, color, texture id, line width, flash rate, visibility, …

- Derived from LcComponent
  - Will manage a list of graphic subcomponents
  - Standard graphic system parent/child relationship
    - Parent attributes are passed down to child nodes
    - Child nodes can override parent attributes
  - updateTC() used to update object's R/T dynamics
  - updateData() used to send data to child components

- Member functions
  - virtual void draw()
    - Draw the graphic object, and all its child components, with the defined attributes (e.g., color, line width, transformation matrix).
    - Most cases, this function does not need to be over ridden by derived classes
  - virtual void drawfunc()
    - Called by draw()
    - Performs the actual drawing (OpenGL)
    - Derived classes provide their own drawfunc()

BasicGL – LcGraphic (cont)

- Member functions (cont)
  - setColor() /* Sets the color */
  - setLinewidth() /* Sets the line width */
  - setFlashRate() /* Sets the flash rate (Hz) */
  - setVisibility() /* Sets visibility flag */

  - lcRotate(), lcTranslate(), lcScale()
    - Transformation member functions

  - setSelectName()
    - Sets OpenGL select name
    - Used to select/pick objects

BasicGL – Graphic classes

- Shapes
  - Attributes and vertices are derived from the LcGraphic class
  - LcPoint
  - LcLine
    - New attribute: segment
      - Draw line segments if true, otherwise draw one continuous line
  - LcLineLoop
  - LcPolygon
  - LcCircle
    - New attributes: radius, filled, slices

- LcField
  - Abstract text field class (created only as a derived class)
  - New Attributes
    - Line number and character position
    - Field width
    - Justification (left, right, center, none)
    - Highlight (brighter text)
    - Underline (use underline font)
    - Reverse video (use reverse video font)

BasicGL – Graphic classes (cont)

- Readouts
  - Derived from LcField
  - AsciiText
    - Standard text readout
  - Rotary and Rotary2
    - Selects and draws one component from its list of components
  - NumericReadout
    - Numerical readout
    - New attributes
      - Format (e.g., "+0#.#“ is a 5 character field, floating point number w/sign and leading zeros)
      - Default value
      - Characters to substitute for ‘+’ and ‘-’ (e.g., ‘E’ and ‘W’)
      - Maximum value
      - Character to fill field with of maximum value is exceeded

BasicGL – Graphic classes (cont)

- Readouts (cont)
  - Classes derived from NumericReadout
    - TimeReadout
      - Displays time in various formats
        - » Input value of time is in seconds
      - Example formats
        - » “HH:MM:SS.S” is a 10 character field to display hours, minutes and seconds to one decimal place
        - » “HH:MM” is a 5 character field to display hours and minutes only
    - LatitudeReadout and LongitudeReadout
      - Input values are in degrees (positive north and east)
      - Displayed in various common formats
    - HexReadout and OctalReadout classes

BasicGL – LcFont

- LcFont
  - Base class for various character fonts
- LcBitmapFont
  - Simple bitmap font
- LcFtglFont
  - OpenGL True Type fonts
  - Uses the FTGL graphics library from GNU
- LcStrokeFont
  - Uses SGI’s stroke font

BasicGL – LcPage

- LcPage
  - Manages a collection of graphic objects as a group (i.e., page)
    - Multifunction Display (MFD) formats
      - ADI and HSI formats
      - Moving map and situation display formats
      - System status formats
    - Control Display Unit (CDU) and Up Front Controller (UFC) formats
      - Waypoint lists
      - Radio pages
  - Contains a list of subpages
    - Partitioned pages
    - Pop up page menus
  - Derived from LcGraphic
    - Has its own ability to draw
  - LcPage's derived classes manage the flow of data from the simulation subsystem to the graphic components

BasicGL – LcDisplay

- LcDisplay
  - Manage collections of pages (LcPage) as a display system
    - MFD, CDU, HUD, HMD, …
    - The selection of pages, master menus, pop-up page events, …
  - Each LcDisplay contains it’s own fonts, color table, textures, standard line width, etc.
  - Derived from LcPage, which is derived from LcGraphic
    - Can provide it’s own background page and menus
    - Can provide it’s own graphic components
  - Derived classes manage the flow of data from major simulation systems to the pages
  - Can contain a list of sub-displays

BasicGL – GlutDisplay

- GlutDisplay
  - Manages a LcDisplay as a GLUT main window or sub-window.
  - Manages GLUT event call backs
    - Reshape
    - Keyboard Event
    - Special Event
    - Mouse Event
    - Draw Event

Instrument Library

- The instrument library is a library that contains classes which when used can replicate various analog instruments.

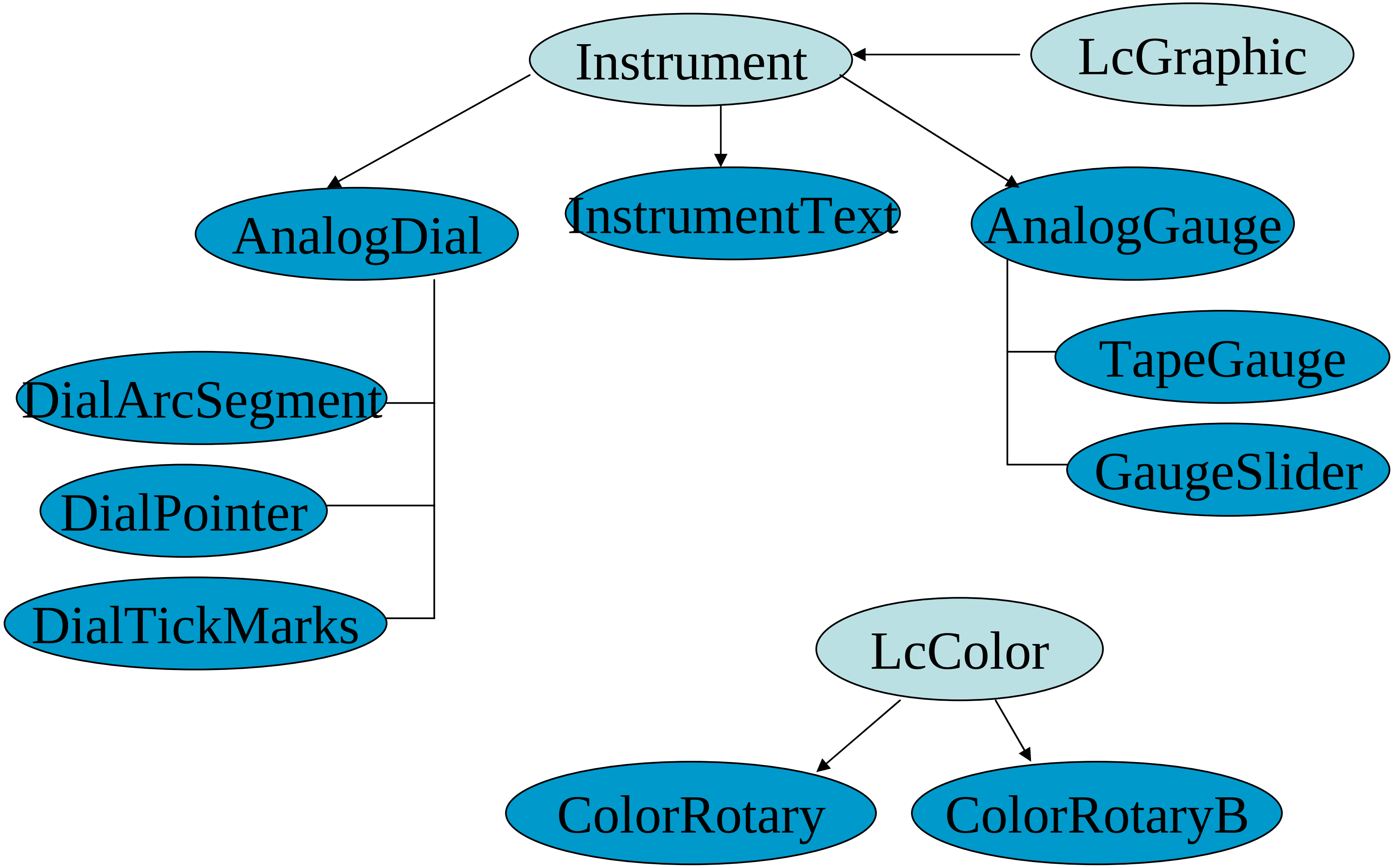
- All instruments are derived from the Instrument class, which itself is derived from LcGraphic.

- These instrument-derived classes can be used alone, or as pieces to create a complex instrument.

- The instrument library depends on BasicGL, which defines the core graphic classes.

- All instruments are data driven, and are easily created and initialized via a *.d input file

Instrument Library (cont)



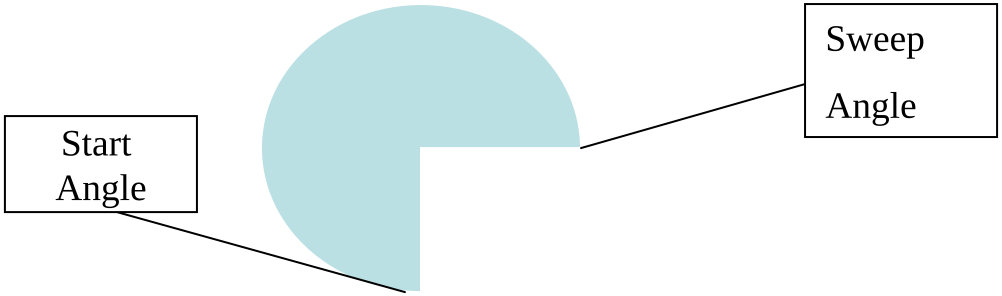
Instrument Library (cont)

- Instrument class
  - Core class for all instruments
  - Member functions
    - setInstVal() – updates our instruments position, based upon either a linear or non-linear scale
      - Also called by passing the token UPDATE_INSTRUMENTS to our instrument classes event function

- ColorRotary class
  - Given an input value, selects a color from a list of colors and associated breakpoint values

Instrument Library (cont)

- AnalogDial class
  - Creates either a mobile or stationary circular background for an analog instrument using:
    - a) startAngle – angle to start drawing from
    - b) SweepAngle – how far around the circle to draw
    - c) DialRadius – how big of a circle to draw
    - d) Mobile – tells us if this object will rotate with instrument position



Instrument Library (cont)

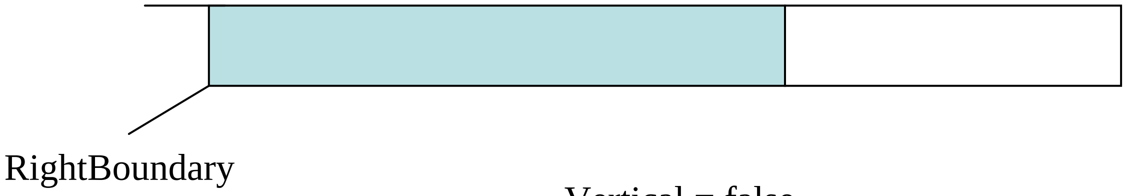
- DialTickMarks class
  - create tick marks around a circular dial
    - length – determines the tick mark length (inches)
    - quantity – how many tick marks to use
    - tickGraphic – use this if you want to specify something other than a line representation for a tick mark (such as a polygon, or texture)

- DialArcSegment class
  - creates a circular portion of a gauge, with a start and sweepAngle
    - This is basically an AnalogDial, only it is not drawn as a filled circle, and is more flexible in it’s drawing (linewidth, etc..)

- DialPointer class
  - the pointer that will rotate around the dial based on the data it is given, and the scaling table it is using
    - It default to a line, but it can draw and rotate complex graphics, including textures

Instrument Library (cont)

- AnalogGauge class
  - Creates either a vertical or horizontal gauge that will increase or decrease given a specific value
  - Slots
    - leftBoundary – point, in inches, to start the left side of the gauge drawing from
    - rightBoundary – right side boundary of the gauge
    - outlined – determines if we fill the gauge or just use an outline
    - vertical – if true, we draw a vertical gauge, else we are horizontal
    - Note: leftBoundary and rightBoundary assume the top boundary and bottom boundary if you specify the verical flag a false.



Vertical = false

Instrument Library (cont)

- GaugeSlider class
  - creates a “slider” that will translate up and down the gauge based on the instrument value
    - every component of a GaugeSlider will translate with the instrument position, from a complex textured graphic to an input field to a numeric readout.
    - you can send the UPDATE_VALUE event to readouts while they are sliding, and they will show the digital output of the gauge.

- InstrumentText class
  - allow readouts and other graphics to recognize what a colorRotary is. This way you can send the value to the instrumentText, and it will ask it’s colorRotary what color it should be, and it will tell all of it’s components to be that color

Instrument Library (cont)

- TapeGauge class
  - takes a graphic, or texture, and translates it up and down depending on the instrument value. It also allows you to scissor a portion of the tape out, so it resembles a true analog gauge.
  - Slots
    - tapeGraphic – the graphic you want to translate
    - scissorX – the x position of where you want to start the scissor
    - scissorWidth – how much of the graphic width you want to scissor
    - scissorY – the y position of where you want to start the scissor
    - scissorHeight – how much of the graphic height you want to scissor