C++ Advanced

Defining a Vector Graphic

Properties of Vector Graphics

Persistence, Serialization, and Files

Lesson Objectives

- Know what vector graphics are
- Know how and why vector graphics are used
- Be able to implement VectorGraphic and Point classes

What are Vector Graphics?

- Define bitmap graphics
    - Unstructured
    - A list of colored dots
    - E.g., bmp, jpg, gif, tiff
- Define Vector Graphics
    - Structured
    - Geometry
    - Points, lines, curves, fonts, pen styles
- Vector graphics get rendered to bitmaps

Working with bitmaps

- What’s the color of pixel at x, y?
- Set the color of pixel at x, y?
- Draw to screen – draw to paper - pixelated
- Screen scraping – search pixels for dots that connect into words
- Bitmap graphic transformations
  - Translation
  - Scaling, Skewing
  - Rotation
- Render and change

Working with Vector Graphics

- Draw Rectangle
- Display List
- Vector graphic transformations
  - Translation
  - Scaling, Skew
  - Rotation
- Render and change

Concept level - abstraction

- Bitmap graphics are lower level concepts
- Vector graphics are higher level – why?
- Draw to screen, draw to paper, scale
  - Pixalation

Properties of Vector Graphics

- Vector graphics are shapes
- Described by how they are drawn
- Fill
  - None
  - Color
  - Shaded
- Stroke
  - Solid
  - Dotted
  - Thickness
  - Color

Curves, Lines, and Points

- Point – x,y,(z) position
  - Graphical coordinate system (y-down)
  - Pixel coordinates
  - World coordinates
- Shapes – line, rectangle, circle, ellipse…
- Line – two or more points
- Curve drawing excluded from class
  - But our design supports addition of curves

Fill and Stroke

- Think of stroke as the pen used to draw lines (color, solid, dotted, width)
- Think of fill as the color and pattern used to fill a shape (e.g., rectangle)
- We'll defer fill and stroke

Closed and open shapes

- Shape is drawn by stroking lines between points
- An open shape does not draw a line between the last point and first point
- A closed shape draws line between last and first shape

VectorGraphic class

- We’ll ignore actual drawing and focus design and implementation of class interface
- Should Vector graphic be abstract?
  - What are pros/cons?

Points: Reference or Value objects?

- Value object
  - Simple concrete classes used like primitive types
    - Often copied
    - May overload operators
- Reference object
  - May be polymorphic
  - Care must be taken with operator overloads since they are not polymorphic

Reference/Value example

- Implement operator=
- What’s it look like for Value type?
- What’s it look like for polymorphic type?
- There is a pattern for this
  - Prototype/Clone

Point

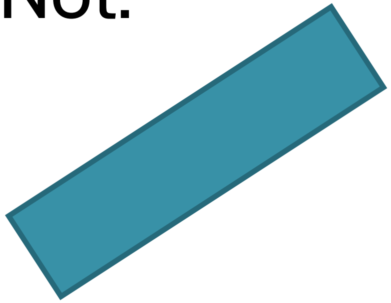
- Reference or Value?

Closed and Open Shapes

- Via Attribute?
  - What’s an attribute?
- setOpen(true) or setClosed(false)
  - Exposes implementation detail (Bool)
  - Which one?
- close() and open()
  - Better, but close and open mean many things
- closeShape() and openShape
  - More specific, but user may expect shape parameter

Bounds

- Shapes are usually constrained to enclosing rectangular boundary
- Often described by two points
  - Upper Left and lower right
  - Assumes horizontal orientation
  - Not:



Choosing a collection

- Use std::list
- We'll defer discussion of choosing the best collection
  - List,
  - Vector
  - Deque
  - Map
  - Set

Persistence, Serialization, and Files

- Persistence refers specifically to the ability to retain data structures between program executions, such as, for example, an image editing program saving complex selections or a word processor saving undo history

Transparent Persistence

- Persistence of object state via abstractions
- Objects not destructed when program closed
  - Instead – moved to permanent storage
- Developer doesn’t worry (or care about)
  - File, database, RAM, hard disk
  - Delimiters, file formats

Serialization

- Current de-facto means of persisting objects
- Object may be written to or read from a stream
- Requires that
  - Class support standard serialize interface
  - Member classes support serialize
- Both Java and .NET provide as part of framework

File Formats

- Consumer applications commonly use file-format based persistence
  - E.g., RTF for word processing documents
  - Many applications use custom file format
    - Data not portable across different applications
- XML is a common file format
  - Many parsers available
  - XML provides mechanism but does not describe content

Who is responsible for persistence?

- The object instances themselves?
- The user (i.e., client of the objects)
- Object responsibilities
  - It’s own business rules (e.g., graphics)
  - Persistence is a different responsibility
    - With different reasons for changing
      - File format changes
      - Multiple clients with different persistence mechanisms
- I like to make persistence outside the class

Our XML persistence format

```
<VectorGraphic closed ="true">
  <Point x="0" y="0"/>
  <Point x="10" y="0">
  </Point>
  <Point x="10" y="10"/>
  <Point x="0" y="10"/>
</VectorGraphic>
```

- Can have multiple same sub elements
- Only one of each attribute type allowed

- Element start
  - <Vector…
- Element end
  - <Vector/>
  - Or
  - <Vector>…</Vector>
- Attribute
  - <Element att=“val”/>

XML Parsing

- In production use an OTS XML parser
- For class, write your own
- Use standard streams and string operators
  - Streams, stringstreams
  - Peek – allows you to look at next input char without removing from stream
  - getline

Stream state and exceptions

- Exception handling added to C++ after streams added
- By default streams throw no exceptions
  - For backward compatibility
- To enable stream exceptions call

```
exceptions(flags);

// E.g.,
exceptions(std::ios::eofbit |
std::ios::failbit | std::ios::badbit);
```