Brushwork (Iteration #1) Design Discussion

CSCI-3081: Program Design and Development

How did you handle key design decisions in your group?

1. **Discuss:** What is the relationship between Tools and Masks in you design?

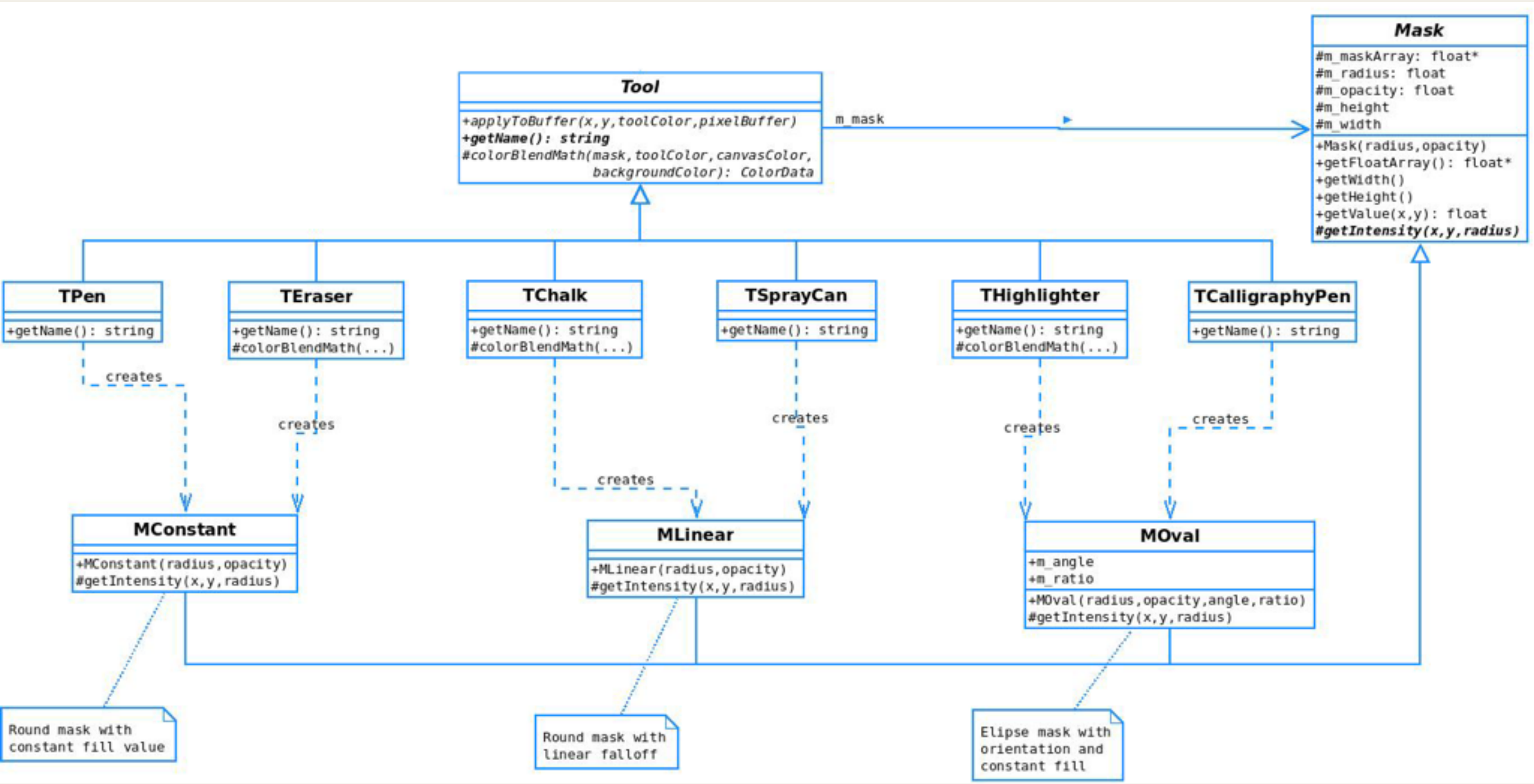
   e.g., Do you have a Mask class?

   e.g., Do you have subclasses of Tool and/or Mask?

2. **Write down or diagram:** What are the top 3-4 designs that you or others at your table used or considered?

3. **Write down:** What are the pros and cons of each?

Our TA Solution



How did you handle key design decisions in your group?

1. **Discuss:** How do you create new Tools?

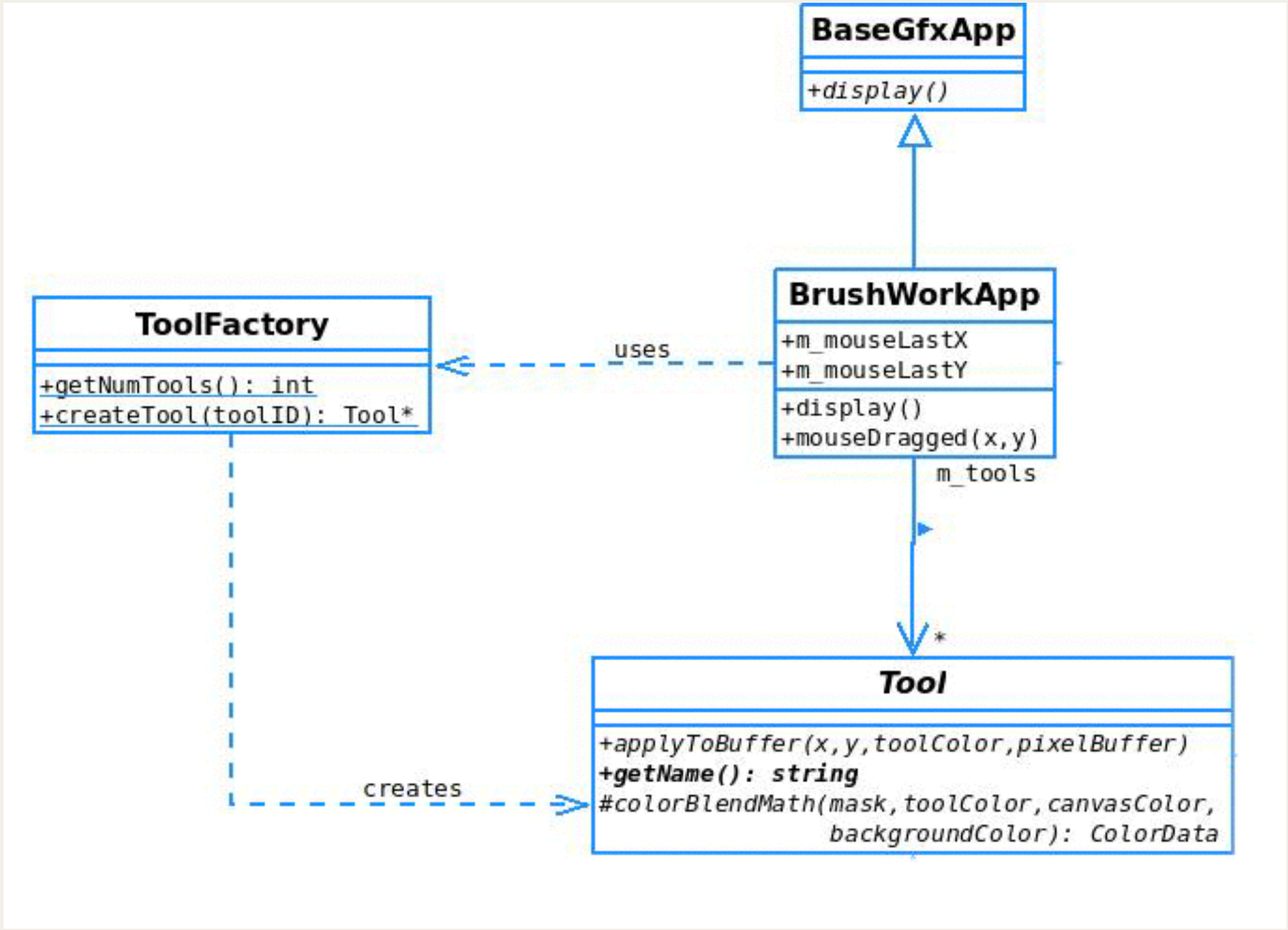
   e.g., Do you use a factory?

   e.g., How many files and how many places in the code need to be edited when a new tool is added?

2. **Write down or diagram:** What are the top 3-4 designs that you or others at your table used or considered?

3. **Write down:** What are the pros and cons of each?

Our TA Solution



How did you handle key design decisions in your group?

1. **Discuss:** How do you handle the “special case” of the eraser?

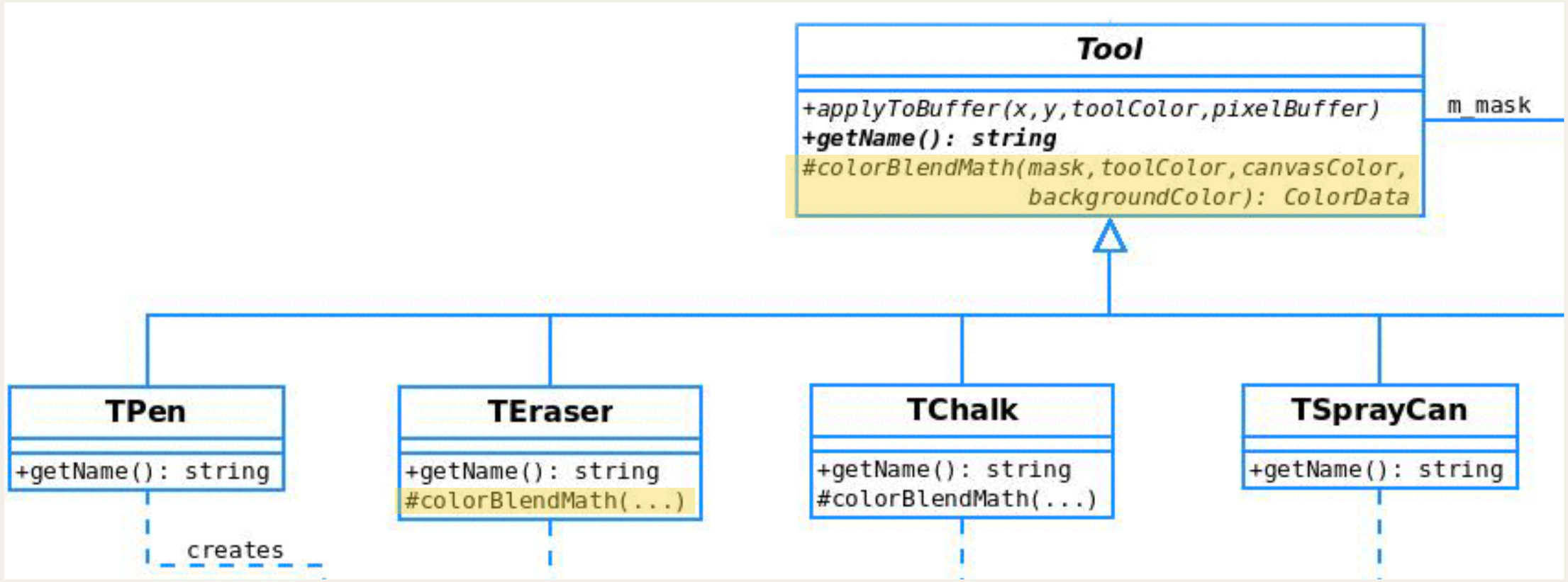
   e.g., How do you pass the background color to the eraser?

   e.g., How does the eraser apply itself to the canvas, does it’s apply method duplicate code used for the other tools?

2. **Write down or diagram:** What are the top 3-4 designs that you or others at your table used or considered?

3. **Write down:** What are the pros and cons of each?

Our TA Solution



Just the code that will need to change for the eraser is isolated in it's own function "colorBlendMath", and a default implementation is provided in Tool.cpp.
The *template method* "applyToBuffer" in Tool.cpp calls colorBlendMath.

```
25  ColorData Tool::colorBlendMath(float mask, ColorData toolColor, ColorData canvasColor, ColorData backgroundColor) {
26      return toolColor*mask + canvasColor*(1.0-mask);
27  }
28
29  void Tool::applyToBuffer(int toolX, int toolY, ColorData toolColor, PixelBuffer* buffer) {
30      if (m_mask == NULL) {
31          return;
32      }
33
34      int left_bound = std::max(toolX-m_mask->getWidth()/2, 0);
35      int right_bound = std::min(toolX+m_mask->getWidth()/2, buffer->getWidth()-1);
36      int lower_bound = std::max(toolY-m_mask->getHeight()/2, 0);
37      int upper_bound = std::min(toolY+m_mask->getHeight()/2, buffer->getHeight()-1);
38
39      for (int y = lower_bound; y <= upper_bound; y++) {
40          for (int x = left_bound; x <= right_bound; x++) {
41              int mask_x = x - (toolX-m_mask->getWidth()/2);
42              int mask_y = y - (toolY-m_mask->getHeight()/2);
43              float mask_value = m_mask->getValue(mask_x, mask_y);
44              ColorData current = buffer->getPixel(x, y);
45
46              // Because we interpolate the pixels, colors overlap
47              // and increase intensity quickly. We found that cubing
48              // the mask intensity compensated for this.
49              float slimmed_mask_value = powf(mask_value,3);
50
51              ColorData c = colorBlendMath(slimmed_mask_value, toolColor, current, buffer->getBackgroundColor());
52
53              buffer->setPixel(x, y, c);
54          }
55      }
56  }
```

Eraser.cpp overrides colorBlendMath to do what is needed for the eraser.

```
23  //  Eraser does not blend colors with the toolColor.  Here we are overriding the
24  //      superclass's colorBlendMath to set the canvasColor to the backgroundColor.
25  ColorData TEraser::colorBlendMath(float mask, ColorData toolColor, ColorData canvasColor, ColorData backgroundColor) {
26      return backgroundColor*mask + canvasColor*(1-mask);
27  }
```

One more interesting design decision…

Rather than passing the background color into Eraser's constructor, we get the background color from the current buffer. Planning for future change, we did this so that the background color will be valid even if we add some new feature to change the background color interactively while the program is running.

Inside Eraser.cpp's applyToBuffer() method:

```
51          ColorData c = colorBlendMath(slimmed_mask_value, toolColor, current, buffer->getBackgroundColor());
```

For all Tools, BrushWorkApp passes the current pixel buffer into the the applyToBuffer function whenever it is called.

```
23  /// This is the superclass for Tools.
24  class Tool
25  {
26  public:
27      Tool();
28      virtual ~Tool();
29
30      virtual void applyToBuffer(int toolX, int toolY, ColorData toolColor, PixelBuffer* buffer);
31      virtual std::string getName() = 0;
32
33  protected:
34      virtual ColorData colorBlendMath(float mask, ColorData toolColor, ColorData canvasColor, ColorData backgroundColor);
35      Mask *m_mask;
36  };
```

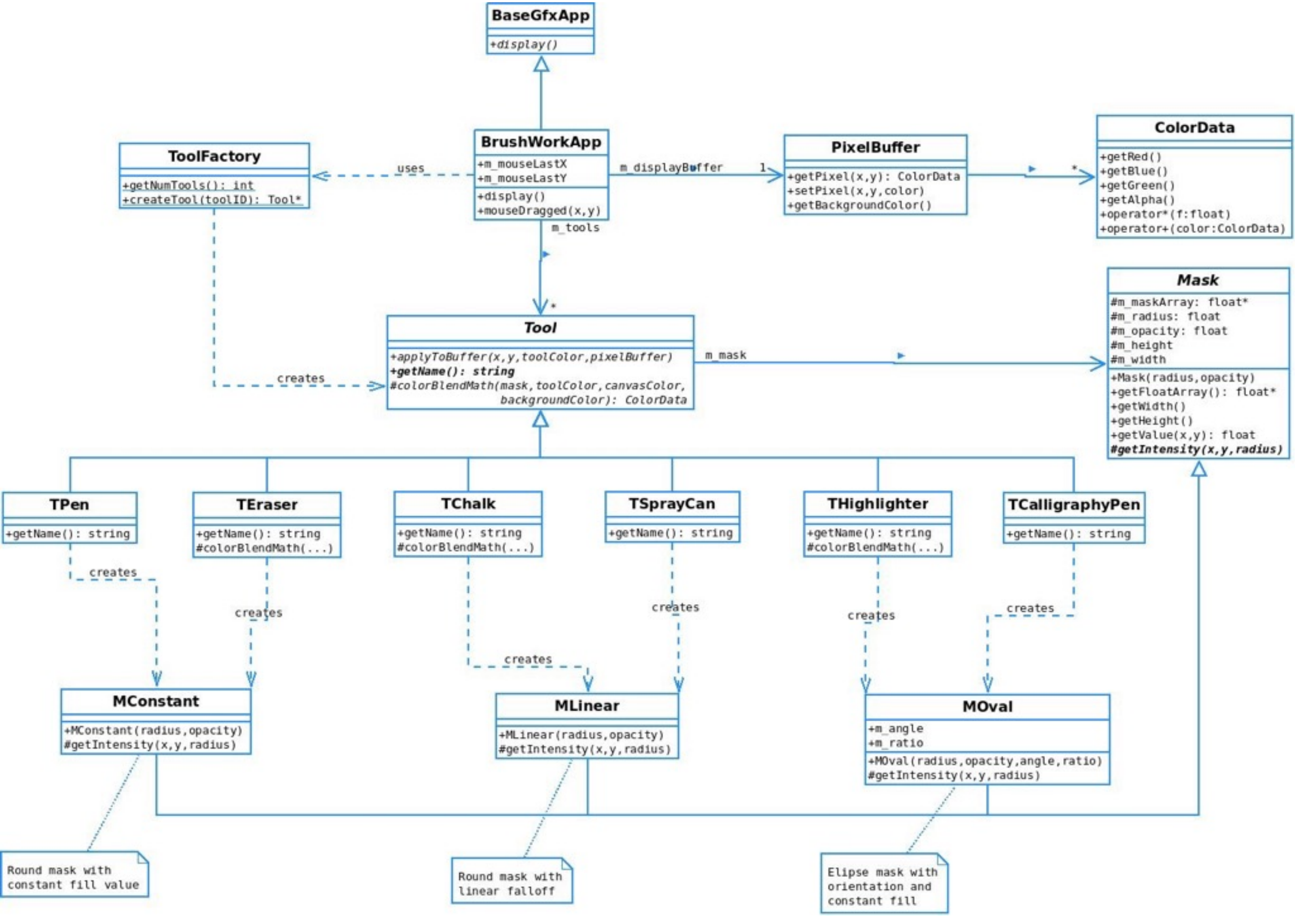
How did you handle key design decisions in your group?

1. **Discuss:** Were there any other important or controversial design decisions in your team's implementation?

   e.g., What did you write about in your design document?

2. **Write down or diagram:** What are the top 3-4 designs that you or others at your table used or considered?

3. **Write down:** What are the pros and cons of each?


BaseGfxApp
+display()
BrushWorkApp
+m_mouseLastX
+m_mouseLastY
+display()
+mouseDragged(x,y)
uses
ToolFactory
+getNumTools(): int
+createTool(toolID): Tool*
m_displayBuffer
1
PixelBuffer
+getPixel(x,y): ColorData
+setPixel(x,y,color)
+getBackgroundColor()
*
ColorData
+getRed()
+getBlue()
+getGreen()
+getAlpha()
+operator*(f:float)
+operator+(color:ColorData)
m_tools
*
Tool
+applyToBuffer(x,y,toolColor,pixelBuffer)
+getName(): string
#colorBlendMath(mask,toolColor,canvasColor,
backgroundColor): ColorData
creates
m_mask
Mask
#m_maskArray: float*
#m_radius: float
#m_opacity: float
#m_height
#m_width
+Mask(radius,opacity)
+getFloatArray(): float*
+getWidth()
+getHeight()
+getValue(x,y): float
#getIntensity(x,y,radius)
TPen
+getName(): string
TEraser
+getName(): string
#colorBlendMath(...)
TChalk
+getName(): string
#colorBlendMath(...)
TSprayCan
+getName(): string
THighlighter
+getName(): string
#colorBlendMath(...)
TCalligraphyPen
+getName(): string
creates
creates
creates
creates
creates
creates
MConstant
+MConstant(radius,opacity)
#getIntensity(x,y,radius)
MLinear
+MLinear(radius,opacity)
#getIntensity(x,y,radius)
MOval
+m_angle
+m_ratio
+MOval(radius,opacity,angle,ratio)
#getIntensity(x,y,radius)
Round mask with
constant fill value
Round mask with
linear falloff
Elipse mask with
orientation and
constant fill

Moodle: Check Current Assignments Heading into Spring Break.

UNIVERSITY OF MINNESOTA
PROFESSOR DANIEL F. KEEFE