OpenGL Quick Reference

Framebuffer Objects: QOpenGLFrameBuffer

- wraps a single OpenGL framebuffer object with renderbuffers used as color attachments for multi-sampled surfaces and textures used for color attachments for normal surfaces.
- depth/stencil attachments are optional, but are implemented with renderbuffers if requested.
- encapsulates all aspects of depth/stencil attachments include various depth bit-sizes
- wraps texture construction and configuration
- wraps glBlitFramebuffer call
- unlike QOpenGLBuffer objects you must have a valid OpenGL context before calling a constructor

Helper Objects

enum Attachment { NoAttachment, CombinedDepthStencil, Depth };

QOpenGLFrameBufferFormat
- helper class used to describe parameters for fbo and fbo surfaces

void setAttachment(QOpenGL FramebufferObject::Attachment attachment);

void setSamples(int samples);

void setMipmap(bool torf);

void setAttachment(Attachment attachment);

void setTextureTarget (GLenum target);

void setInternalTesxtureFormat (GLenuminternalFormat);

Helper Functions:

void **initBindFbo**()
- creates an openGL framebuffer object

```
glGenFramebuffers(1, &internalFbo);
glBindFramebuffer(GL_FRAMEBUFFER, internalFbo);
```

void **initTexture**(int target, int width, int height, bool mipmap)
- constructs a texture object;

```
glGenTexture(1, &internalTexture)
glBindTexture(target, internalTexture);
glTexParameter(target, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameter(target, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameter(target, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameter(target, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glTexImage2D(target, 0, GL_RGBA[8], width, height, 0, GL_RGBA, GL_UNSIGNEDBYTE);
if (mipmap)
  for(int level=0; width>1 || height>1; )
    width=max(1,width>>1);
    height=max(1,height>>1);
    glTexImage2D(target, ++level, GL_RGBA[8], width, height, 0, GL_RGBA, GL_
      UNSIGNED_BYTE, NULL);
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, target,
  internalTexture);
glBindTexture(target, 0);
```

void **initColorBuffer**(int width, int height, int samples)
- constructs a multi-sample color render buffer;

```
glGenRenderbuffers(1, & internalColorBuffer);
glBindRenderbuffer(GL_RENDERBUFFER, internalColorBuffer);
glRenderbufferStorageMultisample(GL_RENDERBUFFER, samples, GL_RGBA8,
width, height, GL_RENDERBUFFER, internalColorBuffer);
glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0,
GL_RENDERBUFFER,internalColorBuffer);
```

void **initPackedDepthStencil**(int samples, int width, int height)
- constructs a packed depth stencil render buffer;

```
glGenRenderbuffers(1, &internalDepthBuffer);
glBindRenderbuffer(GL_RENDERBUFFER, internalDepthBuffer);
if (samples == 0) then
  glRenderbufferStorage(GL_RENDERBUFFER, GL_DEPTH24_STENCIL8);
else
  glRenderbufferStorageMultismaple(GL_RENDERBUFFER, samples, GL_
  DEPTH24_STENCIL8, width, height)
glFrameBufferRenderBuffer(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT,
  GL_RENDERBUFFER, internalDepthBuffer);
glFrameBufferRenderBuffer(GL_FRAMEBUFFER, GL_STENCIL_ATTACHMENT,
  GL_RENDERBUFFER, internalDepthBuffer);
```

void **initComponentDepth**(int samples, int width, int height)
- constructs a depth render buffer;

```
glGenRenderbuffers(1, &internalDepthBuffer);
glBindRenderbuffer(GL_RENDERBUFFER, internalDepthBuffer);
if (samples == 0)
glRenderbufferStorage(GL_RENDERBUFFER, GL_DEPTH_COMPONENT[24/16], width,
 height);
 else
  glRenderBufferStorageMultisample(GL_RENDERBUFFER, samples, GL_DEPTH_
    COMPONENT[24/16], width, height);
 glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT,
  GL_RENDERBUFFER,internalDepthBuffer);
```

void **initComponentStencil**(int samples, int width, int height)
- constructs a stencil render buffer;

```
glGenRenderbuffers(1, &internalStencilBuffer);
glBindRenderbuffer(GL_RENDERBUFFER, internalStencilBuffer);
if (samples == 0)
  glRenderbufferStorage(GL_RENDERBUFFER, GL_STENCIL_INDEX[8], width, height);
else
  glRenderBufferStorageMultisample(GL_RENDERBUFFER, samples, GL_STENCIL_
    INDEX[8], width, height);
glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_STENCIL_ATTACHMENT,
  GL_RENDERBUFFER,internalStencilBuffer);
```

void **initDepthStencil**(Attachment attachment, int samples, int width, int height)
- construct depth/stencil buffer(s)

```
if (attachment==CombinedDepthStencil)
  if (the OpenGL driver supports Packed_Depth_Stencil)
    initPackedDepthStencil(samples, width, height);
  else
    initComponentDepth(samples, width, height);
    initComponentStencil(samples, width, height);
else if (attachment==Depth)
  initComponentDepth(samples, width, height);
```

void **init**(int width, int height, Attachment attachment, GLenum target, GLenum internalFormat, bool mipmap)
- create a non-multisampled FBO with a Texture color surface

```
initBindFbo()
initTexture(samples, width, height);
initDepthStencil(attachment, 0, width, height);
glCheckFramebufferStatus(GL_FRAMEBUFFER) == GL_FRAMEBUFFER_COMPLETE;
```

void **init**(int width, int height, Attachment attachment, int samples)
- create a multisampled FBO with a Renderbuffer color surface

```
initBindFbo()
initColorBuffer(target, width, height);
initDepthStencil(attachment, 0, width, height);
glCheckFramebufferStatus(GL_FRAMEBUFFER) == GL_FRAMEBUFFER_COMPLETE;
```

void **init**(int width, int height, const QOpenGLFramebufferObjectFormat& format)
- create either a non-multisampled or multisampled FBO depending on the format's samples() value.

```
if (format.samples() == 0)
  Init(width, height, format.attachment(), format.target(), format.internalFormat(),
    format.mipmap());
else
  Init(width, height, format.attachment(), format.samples());
```

Constructors

QOpenGLFramebufferObject(int width, int height, Attachment attachment, GLenum target=GL_TEXTURE_2D, GLenum internalFormat=GL_RGBA[8]);
- calls *Init(width, height, attachment, target, internalFormat, false);*

QOpenGLFramebufferObject(const QSize &size, Attachment attachment, GLenum target=GL_TEXTURE_2D, GLenum internalFormat=GL_RGBA[8]));
- calls *Init(size.width(), size.height(), attachment, target, internalFormat, false);*

QOpenGLFramebufferObject(int width, int height, GLenum target = GL_TEXTURE_2D)
- calls *Init(width, height, NoAttachment, target, GL_RGBA[8], false);*

QOpenGLFramebufferObject(const QSize& size, GLenum target = GL_TEXTURE_2D)
- calls *Init(size.width(), size.height(), NoAttachment, target, GL_RGBA[8], false);*

QOpenGLFramebufferObject(int width, int height, const QOpenGLFramebuffer ObjectFormat &format)
- calls *Init(width, height, format);*

QOpenGLFramebufferObject(const QSize &size, const QOpenGLFramebuffer ObjectFormat &format)
- calls *Init(size.width(), size.height(), format);*

Methods

bool **bind**();
- calls *glBindFramebuffer(GL_FRAMEBUFFER, internalFbo);*

bool **release**();
- calls *glBindFramebuffer(GL_FRAMEBUFFER, 0);*

void **setAttachment**(Attachment attachment)
- rebuilds fbo attachments to the new enum value (see the constructors for more information)

GLuint **takeTexture**();
- takes ownership of the texture created in the constructor
- rebuilds a new texture next time the *bind*() is called

Queries

GLuint	**handle**	() const;
int	**width**	() const;
int	**height**	() const;
QSize	**size**	() const;
QOpenGLFramebufferObjectFormat	**format**	() const
GLuint	**texture**	() const;
QImage	**toImage**	() const;
QImage	**toImage**	(bool flipped)const;

Static Public Members

static bool **bindDefault**();
- calls *glBindFramebuffer(GL_FRAMEBUFFER, 0);*

static bool **hasOpenGLFramebufferObjects**();

static bool **hasOpenGLFramebufferBlit**();

static void **blitFramebuffer**(QOpenGLFramebufferObject *target,
QOpenGLFramebufferObject *source,
GLbitfield buffers = GL_COLOR_BUFFER_BIT,
GLenum filter = GL_NEAREST);
- calls *blitFramebuffer(target, QRect(QPoint(0,0), target?target->size():source->size()),*
source, QRect(QPoint(0,0), source?source->size():target->size()),
buffers, filter);

static void **blitFramebuffer**(QOpenGLFramebufferObject *target, const QRect &targetRect,
QOpenGLFramebufferObject *source, const QRect &sourceRect,
GLbitfield buffers = GL_COLOR_BUFFER_BIT,
GLenum filter = GL_NEAREST);
- calls *blitFramebuffer(target, targetRect, source, sourceRect, buffers, filter, 0, 0);*

static void **blitFramebuffer**(QOpenGLFramebufferObject *target, const QRect &targetR,
QOpenGLFramebufferObject *source, const QRect &sourceR,
GLbitfield buffers,
GLenum filter,
int readColorAttachmentIndex,
int drawColorAttachmentIndex);
- blits between framebuffers; either target or source may be NULL (but not both)

```
GLuint prevFbo = 0; glGetIntegerv(GL_FRAMEBUFFER_BINDING, &prevFbo);
glBindFramebuffer(GL_READ_FRAMEBUFFER, source ? source->handle() : 0);
glBindFramebuffer(GL_DRAW_FRAMEBUFFER, target ? target->handle() : 0);
glReadBuffer(GL_COLOR_ATTACHMENT0 + readColorAttachmentIndex);
GLenum drawBuf = GL_COLOR_ATTACHMENT0 + drawColorAttachment; glDrawBuffers(1, &drawBuf);
glBlitFramebuffer(sourceR.left(), sourceR.top(), sourceR.left() + sourceR.width(), sourceR.top() + sourceR.height(),
        targetR.left(), targetR.top(), targetR.left() + targetR.width(), targetR.top() + targetR.height(),
        buffers, filter);
glBindFramebuffer(GL_FRAMEBUFFER, prevFbo);
```