Drawing Path

Pie(9.0)

openGL->GPU(flush)

```
xref: /frameworks/base/libs/hwui/pipeline/skia/SkiaPipeline.cpp
 Home | History | Annotate | Line# | Navigate | Download
                                                                                         Search
      void SkiaPipeline::renderFrame(const LayerUpdateQueue& layers, const SkRect& clip,
  324
                                     const std::vector<sp<RenderNode>>& nodes, bool opaque,
  325
                                    bool wideColorGamut, const Rect& contentDrawBounds,
  326
                                     sk_sp<SkSurface> surface) {
  327
          renderVectorDrawableCache();
  328
  329
          // draw all layers up front
  330
          renderLayersImpl(layers, opaque, wideColorGamut);
  331
  332
          // initialize the canvas for the current frame, that might be a recording canvas if SKP
  333
          // capture is enabled.
  334
          std::unique_ptr<SkPictureRecorder> recorder;
  335
          SkCanvas* canvas = tryCapture(surface.get());
  336
  337
          renderFrameImpl(layers, clip, nodes, opaque, wideColorGamut, coptentDrawBounds, canvas);
  338
  339
          endCapture(surface.get());
  340
  341
          if (CC UNLIKELY(Properties::debugOverdraw)) {
  342
              renderOverdraw(layers, clip, nodes, contentDrawBounds, surface);
  343
  344
  345
          ATRACE_NAME("flush commands");
  346
          surface->getCanvas()->flush();
  347 }
```

```
xref: /external/skia/src/core/SkCanvas.cpp
 Home | History | Annotate | Line# | Navigate | Down
       void SkCanvas: →flush() {
           this->onFlush();
   786 }
   787
   <del>788</del> void SkCanvas∷<mark>onFlush</mark>() {
   789
           SkBaseDevice* device = this->getDevice();
   790
           if (device) {
   791
                device->flush();
   792
   793 }
   794
```

```
xref: /external/skia/src/core/SkCanvas.cpp

Home | History | Annotate | Line# | Navigate | Down

783
784
784
void SkCanvas::flush() {
    this->onFlush();
786 }
787
788 void SkCanvas::onFlush() {
    SkBaseDevice* device = this->getDevice();
    if (device) {
        device->flush();
    }
    792
    }
793 }
794
```

```
xref: /external/skia/src/gpu/SkGpuDevice.cpp
Home | History | Annotate | Line# | Navigate | Download
                                                                            Search only in
1663
1664 void SkGpuDevice::flush() {
1665 this->flushAndSignalSemaphores(0, nullptr);
1666 }
1667
1668 GrSemaphoresSubmitted SkGpuDevice::flushAndSignalSemaphores(int numSemaphores,
1669
                                                       GrBackendSemaphore signalSemaphores[]) {
1670
        ASSERT_SINGLE_OWNER
1671
        return fRenderTargetContext->prepareForExternalIO(numSemaphores, signalSemaphores);
1672
1673 }
```

```
xref: /external/skia/src/gpu/GrRenderTargetContext.cpp
Home | History | Annotate | Line# | Navigate | Download
                                                                                         Search
      GrSemaphoresSubmitted GrRenderTargetContext::prepareForExternallO(
 1383
              int numSemaphores, GrBackendSemaphore backendSemaphores[]) {
          ASSERT SINGLE OWNER
 1384
          if (this->drawingManager()->wasAbandoned()) { return GrSemaphoresSubmitted::kNo; }
 1385
          SkDEBUGCODE(this->validate()
 1386
          GR_CREATE_TRACE_MARKER_CONTEXT("GrRenderTargetContext", "prepareForExternalIO", fContext);
 1387
 1388
          return this->drawingManager()->prepareSurfaceForExternal10(fRenderTargetProxy.get(),
 1389
 1390
                                                                   numSemaphores,
 1391
                                                                   backendSemaphores);
 1392
             Search only ir
```

```
xref: /external/skia/src/gpu/SkGpuDevice.cpp
 Home | History | Annotate | Line# | Navigate | Download
 1663
      void SkGpuDevice flush() {
          this->flushAndSignalSemaphores(0, nullptr);
 1665
 1666
 1667
 1668
     GrSemaphoresSubmitted SkGpuDevice::flushAndSignalSemaphores(int numSemaphores)
 1669
                                                               GrBackendSemaphore signalSemaphores[]) {
 1670
          ASSERT_SINGLE_OWNER
 1671
          return fRenderTargetContext->prepareForExternalIO(numSemaphores, signalSemaphores);
 1672
 1673
```

```
xref: /external/skia/src/gpu/GrDrawingManager.cpp
Home | History | Annotate | Line# | Navigate | Download
                                                                                          Search
 326 GrSemaphoresSubmitted GrDrawingManager: prepareSurfaceForExternal10(
              GrSurfaceProxy* proxy, int numSemaphores, GrBackendSemaphore backendSemaphores[]) {
 328
          if (this->wasAbandoned()) {
              return GrSemaphoresSubmitted::kNo;
 329
 330
 331
          SkASSERT(proxy);
 332
 333
          GrSemaphoresSubmitted result = GrSemaphoresSubmitted::kNo;
          if (proxy->priy().hasPending[0() || numSemaphores) {
 334
 335
              result = this->flush(proxy, numSemaphores, backendSemaphores);
 336
 337
          if (/proxy->instantiate(fContext->contextPriv().resourceProvider())) {
 338
 339
              return result:
 340
  341
          GrGpu* gpu = fContext->contextPriv().getGpu();
          GrSurface* surface = proxy->priv().peekSurface();
 344
  345
          if (gpu && surface->asRenderTarget()) {
              |gpu->reso|veRenderTarget(surface->asRenderTarget());
              urn result:
   Search
```

```
xref: /external/skia/src/gpu/GrRenderTargetContext.cpp
Home | History | Annotate | Line# | Navigate | Download
 1382 GrSemaphoresSubmitted GrRenderTargetContext::prepareForExternal10(
              int numSemaphores, GrBackendSemaphore backendSemaphores[]) {
 1383
 1384
          ASSERT_SINGLE_OWNER
          if (this->drawingManager()->wasAbandoned()) {     return GrSemaphoresSubmitted::kNo; }
 1385
 1386
          SkDEBUGCODE(this->validate();)
          GR_CREATE_TRACE_MARKER_CONTEXT("GrRenderTargetContext", "prepareForExternal10", fContext);
 1387
 1388
          return this->drawingManager()->prepareSurfaceForExternalIO(fRenderTargetProxy.get(),
 1389
 1390
                                                                   numSemaphores,
 1391
                                                                   backendSemaphores);
 1392 }
```

```
xref: /external/skia/src/gpu/GrDrawingManager.cpp
Home | History | Annotate | Line# | Navigate | Download
                                                                                        Search
 326 GrSemaphoresSubmitted GrDrawingManager::prepareSurfaceForExternal10(
             GrSurfaceProxy* proxy, int numSemaphores, GrBackendSemaphore backendSemaphores[]) {
 327
 328
          if (this->wasAbandoned()) {
 329
              return GrSemaphoresSubmitted: kNo;
 330
 331
         Skassert(proxy);
 332
 333
         GrSemaphoresSubmitted result = GrSemaphoresSubmitted::kNo;
 334
          if (proxy->priv().hasPending+6() || numSemaphores) {
              result = this->flush(proxy, numSemaphores, backendSemaphores);
 335
 336
 337
 338
          if (!proxy=>instantiate(fContext=>contextPriv().resourceProvider())) {
 339
              return result:
 340
 341
 342
         GrGpu* gpu = fContext->contextPriv().getGpu();
 343
         GrSurface* surface = proxy->priv().peekSurface();
 344
 345
          if (gpu && surface->asRenderTarget()) {
 346
              gpu->resolveRenderTarget(surface->asRenderTarget());
 347
 348
          return result:
 349 }
```

```
me | History | Annotate | Line# | Navigate | Download |

GrSemaphoresSubmitted | Flush(GrSurfaceProxy* proxy, int numSemaphores = 0, GrBackendSemaphore backendSemaphores[] = nullptr) {

return this->internalFlush(proxy, GrResourceCache::FlushType::kExternal, numSemaphores, backendSemaphores);
}
```

```
Home | History | Annotate | Line# | Navigate | Download | Section |

96 | GrSemaphoresSubmitted | GrSurfaceProxy* | Proxy |
97 | Ipt | numSemaphores = 0 |
98 | GrBackendSemaphore | backendSemaphores | = nullptr |
99 | return | this->internalFlush(proxy | GrResourceCache::FlushType::kExternal |
100 | numSemaphores | backendSemaphores |
101 | }
```

```
xref: /external/skia/src/gpu/GrDrawingManager.cpp
 Home | History | Annotate | Line# | Navigate | Download
                                                                                     Search only in
 116 // MDB TODO: make use of the 'proxy' parameter.
 117 GrSemaphoresSubmitted GrDrawingManager::internalFlush(GrSurfaceProxy*,
                                                            GrResourceCache: FlushType type.
 118
  119
                                                            int numSemaphores,
  120
                                                            GrBackendSemaphore backendSemaphores[]) {
          GR_CREATE_TRACE_MARKER_CONTEXT GrDrawingManager", "internalFlush", fContext);
  121
  122
          if (fFlushing || this was Abandoned())
  123
              return GrSemaphoresSubmitted::kNo;
  124
  125
  126
          fFlushing = true;
  127
  128
          for (int i = 0; i < f0pLists.count(); ++i) {</pre>
              // Semi-usually the GrOpLists are already closed at this point, but sometimes Ganesh
  129
  130
              // needs to flush mid-draw. In that case, the SkGpuDevice's GrOpLists won't be closed
              // but need to be flushed anyway. Closing such GrOpLists here will mean new
              // GrOpLists will be created to replace them if the SkGpuDevice(s) write to them again.
  133
              fOpLists[i]->makeClosed(*fContext->caps());
  134
  135
  136 #ifdef SK_DEBUG
          // This block checks for any unnecessary splits in the opLists. If two sequential opLists
          // share the same backing GrSurfaceProxy it means the opList was artificially split.
  139
          if (fOpLists.count()) {
  140
              GrRenderTargetOpList + prevOpList = fOpLists[0]->asRenderTargetOpList();
  141
              for (int i = 1; i < fOpLists.count(); ++i) {</pre>
                  GrRenderTargetOpList* curOpList = fOpLists[i]->asRenderTargetOpList();
  142
  143
  144
                  if (prevOpList && curOpList) {
  145
                      SkASSERT(prevOpList->fTarget.get() != curOpList->fTarget.get());
  146
  147
  148
                  prevOpList = curOpList;
  149
  150
  151 #endif
  152
  153
          if (fSortRenderTargets) {
  154
              SkDEBUGCODE(bool result =) SkTTopoSort<GrOpList. GrOpList::TopoSortTraits>(&fOpLists);
  155
              SkASSERT(result);
  156
 157
  158
          GrGpu* gpu = fContext->contextPriv().getGpu();
  159
          GrOpFlushState flushState(gpu, fContext->contextPriv().resourceProvider(),
  160
  161
                                    &fTokenTracker);
```

```
117 GrSemaphoresSubmitted GrDrawingManager::internalFlush(GrSurfaceProxy*,
                                                          GrResourceCache::FlushType type,
119
                                                          int numSemaphores.
120
                                                          GrBackendSemaphore backendSemaphores[]) {
121
        GR_CREATE_TRACE_MARKER_CONTEXT("GrDrawingManager", "internalFlush", fContext);
122
123
        if (fFlushing || this->wasAbandoned()) {
124
            return GrSemaphoresSubmitted::kNo;
125
126
        fFlushing = true;
127
128
        for (int i = 0; i < f0pLists.count(); ++i) {</pre>
129
            // Semi-usually the GrOpLists are already closed at this point, but sometimes Ganesh
130
            // needs to flush mid-draw. In that case, the SkGpuDevice's GrOpLists won't be closed
131
            // but need to be flushed anyway. Closing such GrOpLists here will mean new
132
            // GrOpLists will be created to replace them if the SkGpuDevice(s) write to them again.
133
            fOpLists[i] -> makeClosed(*fContext->caps());
134
135
136 #ifdef SK_DEBUG
137
        // This block checks for any unnecessary splits in the opLists. If two sequential opLists
138
        // share the same backing GrSurfaceProxy it means the opList was artificially split.
139
        if (fOpLists.count()) {
140
            GrRenderTargetOpList* prevOpList = fOpLists[0]->asRenderTargetOpList();
141
            for (int i = 1; i < fOpLists.count(); ++i) {</pre>
142
                GrRenderTargetOpList + curOpList = fOpLists[i]->asRenderTargetOpList();
143
144
                if (prevOpList && curOpList) {
145
                    SkASSERT(prevOpList->fTarget.get() != curOpList->fTarget.get());
146
147
148
                prevOpList = curOpList;
149
150
```

GrResourceAllocator alloc(fContext->contextPriv().resourceProvider());

GrResourceAllocator::AssignError error = GrResourceAllocator::AssignError://NoError:

for (int i = 0; i < f0pLists.count(); ++i) {</pre>

alloc.markEndOfOpList(i);

fOpLists.reset();

fOpLists[i]->gatherProxyIntervals(&alloc);

while (alloc.assign(&startIndex, &stopIndex, &error)) {

```
xref: /external/skia/src/gpu/GrGpu.cpp
Home | History | Annotate | Line# | Navigate | Download
                                                                                         Search Or
                      bounds = &flippedBounds;
  459
  460
                  target->flagAsNeedingResolve(bounds);
  461
  462
              GrTexture* texture = surface->asTexture();
              if (texture && 1 == mipLevels) {
  464
                  texture->texturePriv().markMipMapsDirtv();
  465
  466
  467 }
  468
      GrSemaphoresSubmitted GrGpu: finishFlush(int numSemaphores,
  470
                                              GrBackendSemaphore backendSemaphores[]) -
          GrResourceProvider + resourceProvider = fContext->contextPriv(), resourceProvider();
  471
  472
  473
          if (this->caps()->fenceSyncSupport()) {
  474
              for (int i = 0; i < numSemaphores; ++i) {</pre>
  475
                  sk_sp<GrSemaphore> semaphore:
  476
                   if (backendSemaphores[i].isInitialized()) {
  477
                      semaphore = resourceProvider->wrapBackendSemaphore(
  478
                              backendSemaphores[i]. GrResourceProvider::SemaphoreWrapType::kWillSignal.
  479
                             kBorrow_GrWrapOwnership);
  480
                  } else {
                      semaphore = resourceProvider->makeSemaphore(false);
                  this->insertSemaphore(semaphore, false);
  484
  485
                  if (!backendSemaphores[i].isInitialized()) {
  486
                      semaphore->setBackendSemaphore(&backendSemaphores[i]);
  487
  488
  489
  490
          this->onFinishFlush((numSemaphores > 0 && this->caps()->fenceSyncSupport()));
          return this->caps()->fenceSyncSupport() ? GrSemaphoresSubmitted::kYes
  492
                                                 : GrSemaphoresSubmitted::kNo:
  493 }
```

```
if (GrResourceAllocator::AssignError::kFailedProxyInstantiation == error) {
    for (int i = startIndex; i < stopIndex; ++i) {
        fOpLists[i]->purgeOpsWithUninstantiatedProxies();
    }
}
if (this->executeOpLists(startIndex, stopIndex, %flushState)) {
    flushed = true;
```

정

GrSemaphoresSubmitted result = gpu->finishFlush(numSemaphores, backendSemaphores): 단순히 flush는 이 함수를 따라 갑니다.

```
xref: /external/skia/src/gpu/GrGpu.cpp
Home | History | Annotate | Line# | Navigate | Download
                                                                                        Search On
                      bounds = &flippedBounds:
  459
  460
                  target->flagAsNeedingResolve(bounds);
              GrTexture* texture = surface->asTexture();
              if (texture && 1 == mipLevels) {
                  texture->texturePriv().markMipMapsDirty();
  467
      GrSemaphoresSubmitted GrGpu::finishFlush(int numSemaphores,
  470
                                              GrBackendSemaphore backendSemaphores[]) {
  471
          GrResourceProvider + resourceProvider = fContext->contextPriv(), resourceProvider();
  472
  473
          if (this->caps()->fenceSyncSupport()) {
  474
              for (int i = 0; i < numSemaphores; ++i) {</pre>
  475
                  sk sp<GrSemaphore> semaphore;
  476
                  if (backendSemaphores[i].isInitialized()) {
  477
                      semaphore = resourceProvider->wrapBackendSemaphore(
                             backendSemaphores[i], GrResourceProvider::SemaphoreWrapType::kWillSignal,
  478
  479
                             kBorrow GrWrapOwnership);
  480
                  } else {
                      semaphore = resourceProvider->makeSemaphore(false);
                  this->insertSemaphore(semaphore, false)
  484
  485
                  if (!backendSemaphores[i]. hitialized()) {
                      semaphore->setBackendSemaphore(&backendSemaphores[i]);
  486
  487
  488
  489
          this->onFinishFlush((numSemaphores > 0 && this->caps()->fenceSyncSupport()));
  490
  491
          return this->caps()->fenceSyncSupport() ? GrSemaphoresSubmitted::kYes
  492
                                                 : GrSemaphoresSubmitted::kNo:
  493 }
```