实际上这些都是在Render中进行改变的：

drawBatchedTriangles

drawBatchedQuads（在这里也是调用的opengl绘制三角形）

实际上两者的行为是一样的：

void Renderer::drawBatchedTriangles()

{

//TODO: we can improve the draw performance by insert material switching command before hand.

int indexToDraw = 0;

int startIndex = 0;

//Upload buffer to VBO

if(\_filledVertex <= 0 || \_filledIndex <= 0 || \_batchedCommands.empty())

{

return;

}

if (Configuration::getInstance()->supportsShareableVAO())

{

//Bind VAO

GL::bindVAO(\_buffersVAO);

//Set VBO data

glBindBuffer(GL\_ARRAY\_BUFFER, \_buffersVBO[0]);

// option 1: subdata

// glBufferSubData(GL\_ARRAY\_BUFFER, sizeof(\_quads[0])\*start, sizeof(\_quads[0]) \* n , &\_quads[start] );

// option 2: data

// glBufferData(GL\_ARRAY\_BUFFER, sizeof(quads\_[0]) \* (n-start), &quads\_[start], GL\_DYNAMIC\_DRAW);

// option 3: orphaning + glMapBuffer

glBufferData(GL\_ARRAY\_BUFFER, sizeof(\_verts[0]) \* \_filledVertex, nullptr, GL\_DYNAMIC\_DRAW);

void \*buf = glMapBuffer(GL\_ARRAY\_BUFFER, GL\_WRITE\_ONLY);

memcpy(buf, \_verts, sizeof(\_verts[0])\* \_filledVertex);

glUnmapBuffer(GL\_ARRAY\_BUFFER);

glBindBuffer(GL\_ARRAY\_BUFFER, 0);

glBindBuffer(GL\_ELEMENT\_ARRAY\_BUFFER, \_buffersVBO[1]);

glBufferData(GL\_ELEMENT\_ARRAY\_BUFFER, sizeof(\_indices[0]) \* \_filledIndex, \_indices, GL\_STATIC\_DRAW);

}

else

{

#define kQuadSize sizeof(\_verts[0])

glBindBuffer(GL\_ARRAY\_BUFFER, \_buffersVBO[0]);

glBufferData(GL\_ARRAY\_BUFFER, sizeof(\_verts[0]) \* \_filledVertex , \_verts, GL\_DYNAMIC\_DRAW);

GL::enableVertexAttribs(GL::VERTEX\_ATTRIB\_FLAG\_POS\_COLOR\_TEX);

// vertices

glVertexAttribPointer(GLProgram::VERTEX\_ATTRIB\_POSITION, 3, GL\_FLOAT, GL\_FALSE, kQuadSize, (GLvoid\*) offsetof(V3F\_C4B\_T2F, vertices));

// colors

glVertexAttribPointer(GLProgram::VERTEX\_ATTRIB\_COLOR, 4, GL\_UNSIGNED\_BYTE, GL\_TRUE, kQuadSize, (GLvoid\*) offsetof(V3F\_C4B\_T2F, colors));

// tex coords

glVertexAttribPointer(GLProgram::VERTEX\_ATTRIB\_TEX\_COORD, 2, GL\_FLOAT, GL\_FALSE, kQuadSize, (GLvoid\*) offsetof(V3F\_C4B\_T2F, texCoords));

glBindBuffer(GL\_ELEMENT\_ARRAY\_BUFFER, \_buffersVBO[1]);

glBufferData(GL\_ELEMENT\_ARRAY\_BUFFER, sizeof(\_indices[0]) \* \_filledIndex, \_indices, GL\_STATIC\_DRAW);

}

//Start drawing vertices in batch

for(const auto& cmd : \_batchedCommands)

{

auto newMaterialID = cmd->getMaterialID();

if(\_lastMaterialID != newMaterialID || newMaterialID == MATERIAL\_ID\_DO\_NOT\_BATCH)

{

//Draw quads

if(indexToDraw > 0)

{

**// 通过索引进行绘制**

**glDrawElements(GL\_TRIANGLES, (GLsizei) indexToDraw, GL\_UNSIGNED\_SHORT, (GLvoid\*) (startIndex\*sizeof(\_indices[0])) );**

**\_drawnBatches++;**

**\_drawnVertices += indexToDraw;**

**startIndex += indexToDraw;**

**indexToDraw = 0;**

}

//Use new material

cmd->useMaterial();

\_lastMaterialID = newMaterialID;

}

indexToDraw += cmd->getIndexCount();

}

//Draw any remaining triangles

if(indexToDraw > 0)

{

**glDrawElements(GL\_TRIANGLES, (GLsizei) indexToDraw, GL\_UNSIGNED\_SHORT, (GLvoid\*) (startIndex\*sizeof(\_indices[0])) );**

**\_drawnBatches++;**

**\_drawnVertices += indexToDraw;**

}

if (Configuration::getInstance()->supportsShareableVAO())

{

//Unbind VAO

GL::bindVAO(0);

}

else

{

glBindBuffer(GL\_ARRAY\_BUFFER, 0);

glBindBuffer(GL\_ELEMENT\_ARRAY\_BUFFER, 0);

}

\_batchedCommands.clear();

\_filledVertex = 0;

\_filledIndex = 0;

}