RenderCommand有如下几种类型：

enum class Type

{

/\*\* Reserved type.\*/

UNKNOWN\_COMMAND,

/\*\* Quad command, used for draw quad.\*/

QUAD\_COMMAND,

/\*\*Custom command, used for calling callback for rendering.\*/

CUSTOM\_COMMAND,

/\*\*Batch command, used for draw batches in texture atlas.\*/

BATCH\_COMMAND,

/\*\*Group command, which can group command in a tree hierarchy.\*/

GROUP\_COMMAND,

/\*\*Mesh command, used to draw 3D meshes.\*/

MESH\_COMMAND,

/\*\*Primitive command, used to draw primitives such as lines, points and triangles.\*/

PRIMITIVE\_COMMAND,

/\*\*Triangles command, used to draw triangles.\*/

TRIANGLES\_COMMAND

};

对于Sprite而言，它里面保存的是一个TrianglesCommand，同时保存了一个四边形的数据结构V3F\_C4B\_T2F\_Quad，还有一个PolygonInfo信息

class CC\_DLL TrianglesCommand : public RenderCommand

{

public:

/\*\*The structure of Triangles. \*/

struct Triangles

{

/\*\*Vertex data pointer.\*/

V3F\_C4B\_T2F\* verts;

/\*\*Index data pointer.\*/

unsigned short\* indices;

/\*\*The number of vertices.\*/

ssize\_t vertCount;

/\*\*The number of indices.\*/

ssize\_t indexCount;

};

/\*\*Constructor.\*/

TrianglesCommand();

/\*\*Destructor.\*/

~TrianglesCommand();

/\*\* Initializes the command.

@param globalOrder GlobalZOrder of the command.

@param textureID The openGL handle of the used texture.

@param glProgramState The specified glProgram and its uniform.

@param blendType Blend function for the command.

@param triangles Rendered triangles for the command.

@param mv ModelView matrix for the command.

@param flags to indicate that the command is using 3D rendering or not.

\*/

void init(float globalOrder, GLuint textureID, GLProgramState\* glProgramState, BlendFunc blendType, const Triangles& triangles,const Mat4& mv, uint32\_t flags);

/\*\*Deprecated function, the params is similar as the upper init function, with flags equals 0.\*/

CC\_DEPRECATED\_ATTRIBUTE void init(float globalOrder, GLuint textureID, GLProgramState\* glProgramState, BlendFunc blendType, const Triangles& triangles,const Mat4& mv);

/\*\*Apply the texture, shaders, programs, blend functions to GPU pipeline.\*/

void useMaterial() const;

/\*\*Get the material id of command.\*/

inline uint32\_t getMaterialID() const { return \_materialID; }

/\*\*Get the openGL texture handle.\*/

inline GLuint getTextureID() const { return \_textureID; }

/\*\*Get a const reference of triangles.\*/

inline const Triangles& getTriangles() const { return \_triangles; }

/\*\*Get the vertex count in the triangles.\*/

inline ssize\_t getVertexCount() const { return \_triangles.vertCount; }

/\*\*Get the index count of the triangles.\*/

inline ssize\_t getIndexCount() const { return \_triangles.indexCount; }

/\*\*Get the vertex data pointer.\*/

inline const V3F\_C4B\_T2F\* getVertices() const { return \_triangles.verts; }

/\*\*Get the index data pointer.\*/

inline const unsigned short\* getIndices() const { return \_triangles.indices; }

/\*\*Get the glprogramstate.\*/

inline GLProgramState\* getGLProgramState() const { return \_glProgramState; }

/\*\*Get the blend function.\*/

inline BlendFunc getBlendType() const { return \_blendType; }

/\*\*Get the model view matrix.\*/

inline const Mat4& getModelView() const { return \_mv; }

protected:

/\*\*Generate the material ID by textureID, glProgramState, and blend function.\*/

void generateMaterialID();

/\*\*Generated material id.\*/

uint32\_t \_materialID;

/\*\*OpenGL handle for texture.\*/

GLuint \_textureID;

/\*\*GLprogramstate for the command. encapsulate shaders and uniforms.\*/

GLProgramState\* \_glProgramState;

/\*\*Blend function when rendering the triangles.\*/

BlendFunc \_blendType;

/\*\*Rendered triangles.\*/

**Triangles \_triangles;**

/\*\*Model view matrix when rendering the triangles.\*/

Mat4 \_mv;

};

**struct Triangles**

{

/\*\*Vertex data pointer.\*/

V3F\_C4B\_T2F\* verts;

/\*\*Index data pointer.\*/

unsigned short\* indices;

/\*\*The number of vertices.\*/

ssize\_t vertCount;

/\*\*The number of indices.\*/

ssize\_t indexCount;

};

**class CC\_DLL PolygonInfo**

{

public:

/// @name Creators

/// @{

/\*\*

\* Creates an empty Polygon info

\* @memberof PolygonInfo

\* @return PolygonInfo object

\*/

PolygonInfo():

isVertsOwner(true),

rect(cocos2d::Rect::ZERO),

filename("")

{

triangles.verts = nullptr;

triangles.indices = nullptr;

triangles.vertCount = 0;

triangles.indexCount = 0;

};

/\*\*

\* Create an polygoninfo from the data of another Polygoninfo

\* @param other another PolygonInfo to be copied

\* @return duplicate of the other PolygonInfo

\*/

PolygonInfo(const PolygonInfo& other);

// end of creators group

/// @}

/\*\*

\* Copy the member of the other PolygonInfo

\* @param other another PolygonInfo to be copied

\*/

PolygonInfo& operator= (const PolygonInfo &other);

~PolygonInfo();

/\*\*

\* set the data to be a pointer to a quad

\* the member verts will not be released when this PolygonInfo destructs

\* as the verts memory are managed by other objects

\* @param quad a pointer to the V3F\_C4B\_T2F\_Quad object

\*/

**void setQuad(V3F\_C4B\_T2F\_Quad \*quad);**

/\*\*

\* set the data to be a pointer to a triangles

\* the member verts will not be released when this PolygonInfo destructs

\* as the verts memory are managed by other objects

\* @param triangles a pointer to the TrianglesCommand::Triangles object

\*/

**void setTriangles(TrianglesCommand::Triangles triangles);**

/\*\*

\* get vertex count

\* @return number of vertices

\*/

const unsigned int getVertCount() const;

/\*\*

\* get triangles count

\* @return number of triangles

\*/

const unsigned int getTriaglesCount() const;

/\*\*

\* get sum of all triangle area size

\* @return sum of all triangle area size

\*/

const float getArea() const;

Rect rect;

std::string filename;

TrianglesCommand::Triangles triangles;

protected:

bool isVertsOwner;

private:

void releaseVertsAndIndices();

};

在Sprite中会调用PolygonInfo的setQuad函数：

**void PolygonInfo::setQuad(V3F\_C4B\_T2F\_Quad \*quad)**

{

releaseVertsAndIndices();

isVertsOwner = false;

triangles.indices = quadIndices;

triangles.vertCount = 4;

triangles.indexCount = 6;

triangles.verts = (V3F\_C4B\_T2F\*)quad;

}

总体而言：

**Sprite：**V3F\_C4B\_T2F\_Quad PolygonInfo TrangleCommand

**PolygonInfo：**rect filename isVertsOwner triangles

**TrangleCommand：**muterialID textureID glProgramState blendType triangles mv

Sprite通过各个函数填充V3F\_C4B\_T2F\_Quad，然后通过PolygonInfo的setQuad函数设置PolygonInfo的triangles，最后通过Sprite的draw函数init给TrangleCommand，在各个改变的函数中都是直修改的TrangleCommand中的triangles属性