在创建了一个Image之后，通过Image创建Texture2D，在默认的情况下会调用下面的函数：

**bool Texture2D::initWithMipmaps(MipmapInfo\* mipmaps, int mipmapsNum, PixelFormat pixelFormat, int pixelsWide, int pixelsHigh)**

{

//the pixelFormat must be a certain value

CCASSERT(pixelFormat != PixelFormat::NONE && pixelFormat != PixelFormat::AUTO, "the \"pixelFormat\" param must be a certain value!");

CCASSERT(pixelsWide>0 && pixelsHigh>0, "Invalid size");

if (mipmapsNum <= 0)

{

CCLOG("cocos2d: WARNING: mipmap number is less than 1");

return false;

}

if(\_pixelFormatInfoTables.find(pixelFormat) == \_pixelFormatInfoTables.end())

{

CCLOG("cocos2d: WARNING: unsupported pixelformat: %lx", (unsigned long)pixelFormat );

return false;

}

const PixelFormatInfo& info = \_pixelFormatInfoTables.at(pixelFormat);

if (info.compressed && !Configuration::getInstance()->supportsPVRTC()

&& !Configuration::getInstance()->supportsETC()

&& !Configuration::getInstance()->supportsS3TC()

&& !Configuration::getInstance()->supportsATITC())

{

CCLOG("cocos2d: WARNING: PVRTC/ETC images are not supported");

return false;

}

//Set the row align only when mipmapsNum == 1 and the data is uncompressed

if (mipmapsNum == 1 && !info.compressed)

{

unsigned int bytesPerRow = pixelsWide \* info.bpp / 8;

if(bytesPerRow % 8 == 0)

{

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 8);

}

else if(bytesPerRow % 4 == 0)

{

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 4);

}

else if(bytesPerRow % 2 == 0)

{

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 2);

}

else

{

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

}

}else

{

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

}

if(\_name != 0)

{

GL::deleteTexture(\_name);

\_name = 0;

}

glGenTextures(1, &\_name);

GL::bindTexture2D(\_name);

**// 这里设置了纹理参数antialiasEnabled，默认为true，也就是说默认纹理过滤采用的GL\_LINEAR过滤方式**

if (mipmapsNum == 1)

{

**glTexParameteri( GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, \_antialiasEnabled ? GL\_LINEAR : GL\_NEAREST);**

}else

{

**glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, \_antialiasEnabled ? GL\_LINEAR\_MIPMAP\_NEAREST : GL\_NEAREST\_MIPMAP\_NEAREST);**

}

**glTexParameteri( GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, \_antialiasEnabled ? GL\_LINEAR : GL\_NEAREST );**

**glTexParameteri( GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP\_TO\_EDGE );**

**glTexParameteri( GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP\_TO\_EDGE );**

#if CC\_ENABLE\_CACHE\_TEXTURE\_DATA

if (\_antialiasEnabled)

{

TexParams texParams = {(GLuint)(\_hasMipmaps?GL\_LINEAR\_MIPMAP\_NEAREST:GL\_LINEAR),GL\_LINEAR,GL\_NONE,GL\_NONE};

VolatileTextureMgr::setTexParameters(this, texParams);

}

else

{

TexParams texParams = {(GLuint)(\_hasMipmaps?GL\_NEAREST\_MIPMAP\_NEAREST:GL\_NEAREST),GL\_NEAREST,GL\_NONE,GL\_NONE};

VolatileTextureMgr::setTexParameters(this, texParams);

}

#endif

// clean possible GL error

GLenum err = glGetError();

if (err != GL\_NO\_ERROR)

{

cocos2d::log("OpenGL error 0x%04X in %s %s %d\n", err, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_);

}

// Specify OpenGL texture image

int width = pixelsWide;

int height = pixelsHigh;

for (int i = 0; i < mipmapsNum; ++i)

{

unsigned char \*data = mipmaps[i].address;

GLsizei datalen = mipmaps[i].len;

if (info.compressed)

{

glCompressedTexImage2D(GL\_TEXTURE\_2D, i, info.internalFormat, (GLsizei)width, (GLsizei)height, 0, datalen, data);

}

else

{

glTexImage2D(GL\_TEXTURE\_2D, i, info.internalFormat, (GLsizei)width, (GLsizei)height, 0, info.format, info.type, data);

}

if (i > 0 && (width != height || ccNextPOT(width) != width ))

{

CCLOG("cocos2d: Texture2D. WARNING. Mipmap level %u is not squared. Texture won't render correctly. width=%d != height=%d", i, width, height);

}

err = glGetError();

if (err != GL\_NO\_ERROR)

{

CCLOG("cocos2d: Texture2D: Error uploading compressed texture level: %u . glError: 0x%04X", i, err);

return false;

}

width = MAX(width >> 1, 1);

height = MAX(height >> 1, 1);

}

\_contentSize = Size((float)pixelsWide, (float)pixelsHigh);

\_pixelsWide = pixelsWide;

\_pixelsHigh = pixelsHigh;

\_pixelFormat = pixelFormat;

\_maxS = 1;

\_maxT = 1;

\_hasPremultipliedAlpha = false;

\_hasMipmaps = mipmapsNum > 1;

// shader

setGLProgram(GLProgramCache::getInstance()->getGLProgram(GLProgram::SHADER\_NAME\_POSITION\_TEXTURE));

return true;

}