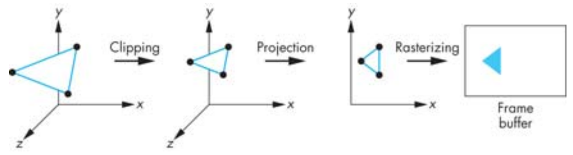
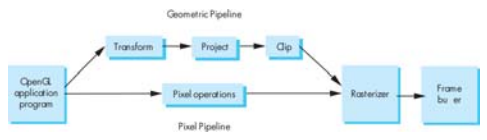
Buffers

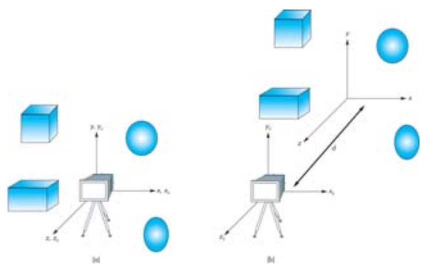
Niels Jørgen Christensen
IMM . DTU

Output pipeline

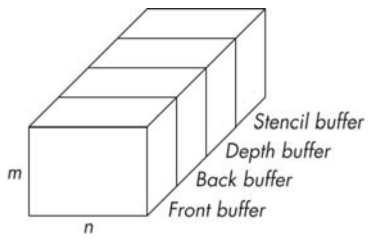


Framebuffer

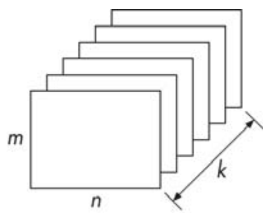




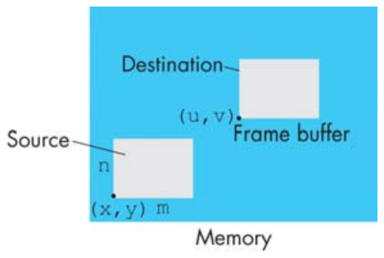
OpenGL Buffers >3.1



Buffer – bit planes



Source - Destination



19-11-2013

Color buffer, depth buffer

- glColorMask (...) Write to color buffer on/off
- glEnable(GL_DEPTH_TEST)
- glDisable(GL_DEPTH_TEST)
- glDepthMask (…) Write to depth buffer on/off
- glDepthFunc (…)   Depth test

19-11-2013

Stencil buffer

- Compare framebuffer, depth buffer (and framebuffer objects)
- Use of stencil buffer

19-11-2013

Blending

- Source color
- Destination color
- glBlendFunc (…) Calculation of new values

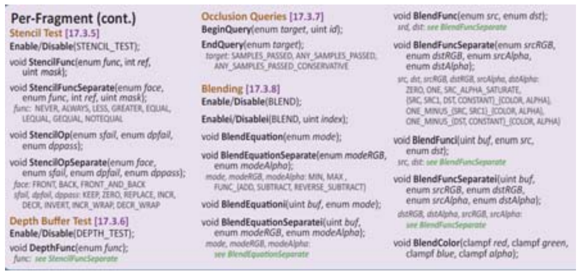
19-11-2013

Clip plane

- Clip Plane: N,dist
- Which side

19-11-2013

19-11-2013



Per-Fragment (cont.)

Stencil Test [17.3.5]

Enable/Disable(STENCIL_TEST);

void **StencilFunc**(enum *func*, int *ref*, uint *mask*);

void **StencilFuncSeparate**(enum *face*, enum *func*, int *ref*, uint *mask*);
func: NEVER, ALWAYS, LESS, GREATER, EQUAL, LEQUAL, GEQUAL, NOTEQUAL

void **StencilOp**(enum *sfail*, enum *dpfail*, enum *dppass*);

void **StencilOpSeparate**(enum *face*, enum *sfail*, enum *dpfail*, enum *dppass*);
face: FRONT, BACK, FRONT_AND_BACK
sfail, dpfail, dppass: KEEP, ZERO, REPLACE, INCR, DECR, INVERT, INCR_WRAP, DECR_WRAP

Depth Buffer Test [17.3.6]

Enable/Disable(DEPTH_TEST);

void **DepthFunc**(enum *func*);
func: *see StencilFuncSeparate*

Occlusion Queries [17.3.7]

BeginQuery(enum *target*, uint *id*);

EndQuery(enum *target*);
target: SAMPLES_PASSED, ANY_SAMPLES_PASSED, ANY_SAMPLES_PASSED_CONSERVATIVE

Blending [17.3.8]

Enable/Disable(BLEND);

Enablei/Disablei(BLEND, uint *index*);

void **BlendEquation**(enum *mode*);

void **BlendEquationSeparate**(enum *modeRGB*, enum *modeAlpha*);
mode, modeRGB, modeAlpha: MIN, MAX, FUNC_{ADD, SUBTRACT, REVERSE_SUBTRACT}

void **BlendEquationi**(uint *buf*, enum *mode*);

void **BlendEquationSeparatei**(uint *buf*, enum *modeRGB*, enum *modeAlpha*);
mode, modeRGB, modeAlpha: *see BlendEquationSeparate*

void **BlendFunc**(enum *src*, enum *dst*);
src, dst: *see BlendFuncSeparate*

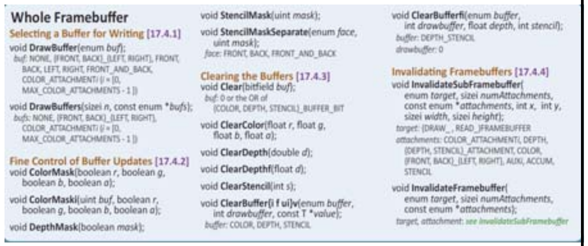
void **BlendFuncSeparate**(enum *srcRGB*, enum *dstRGB*, enum *srcAlpha*, enum *dstAlpha*);
src, dst, srcRGB, dstRGB, srcAlpha, dstAlpha: ZERO, ONE, SRC_ALPHA_SATURATE, {SRC, SRC1, DST, CONSTANT}_{COLOR, ALPHA}, ONE_MINUS_{SRC, SRC1}_{COLOR, ALPHA}, ONE_MINUS_{DST, CONSTANT}_{COLOR, ALPHA}

void **BlendFunci**(uint *buf*, enum *src*, enum *dst*);
src, dst: *see BlendFuncSeparate*

void **BlendFuncSeparatei**(uint *buf*, enum *srcRGB*, enum *dstRGB*, enum *srcAlpha*, enum *dstAlpha*);
dstRGB, dstAlpha, srcRGB, srcAlpha: *see BlendFuncSeparate*

void **BlendColor**(clampf *red*, clampf *green*, clampf *blue*, clampf *alpha*);

19-11-2013



Whole Framebuffer

Selecting a Buffer for Writing [17.4.1]

void **DrawBuffer**(enum *buf*);
buf: NONE, {FRONT, BACK}_{LEFT, RIGHT}, FRONT, BACK, LEFT, RIGHT, FRONT_AND_BACK, COLOR_ATTACHMENTi (i = [0, MAX_COLOR_ATTACHMENTS - 1])

void **DrawBuffers**(sizei *n*, const enum **bufs*);
bufs: NONE, {FRONT, BACK}_{LEFT, RIGHT}, COLOR_ATTACHMENTi (i = [0, MAX_COLOR_ATTACHMENTS - 1])

Fine Control of Buffer Updates [17.4.2]

void **ColorMask**(boolean *r*, boolean *g*, boolean *b*, boolean *a*);

void **ColorMaski**(uint *buf*, boolean *r*, boolean *g*, boolean *b*, boolean *a*);

void **DepthMask**(boolean *mask*);

void **StencilMask**(uint *mask*);

void **StencilMaskSeparate**(enum *face*, uint *mask*);
face: FRONT, BACK, FRONT_AND_BACK

Clearing the Buffers [17.4.3]

void **Clear**(bitfield *buf*);
buf: 0 or the OR of {COLOR, DEPTH, STENCIL}_BUFFER_BIT

void **ClearColor**(float *r*, float *g*, float *b*, float *a*);

void **ClearDepth**(double *d*);

void **ClearDepthf**(float *d*);

void **ClearStencil**(int *s*);

void **ClearBuffer{i f ui}v**(enum *buffer*, int *drawbuffer*, const T **value*);
buffer: COLOR, DEPTH, STENCIL

void **ClearBufferfi**(enum *buffer*, int *drawbuffer*, float *depth*, int *stencil*);
buffer: DEPTH_STENCIL
drawbuffer: 0

Invalidating Framebuffers [17.4.4]

void **InvalidateSubFramebuffer**(enum *target*, sizei *numAttachments*, const enum **attachments*, int *x*, int *y*, sizei *width*, sizei *height*);
target: {DRAW_, READ_}FRAMEBUFFER
attachments: COLOR_ATTACHMENTi, DEPTH, {DEPTH, STENCIL}_ATTACHMENT, COLOR, {FRONT, BACK}_{LEFT, RIGHT}, AUXi, ACCUM, STENCIL

void **InvalidateFramebuffer**(enum *target*, sizei *numAttachments*, const enum **attachments*);
target, attachment: *see InvalidateSubFramebuffer*

19-11-2013

Color buffer

- glDrawBuffer (buf)
- Depth test - On/off:
  glEnable/glDisable(GL_DEPTH_TEST)
- Write depth buffer - on/off: glDepthMask
- Depth test details: glDepthFunc

- Write to colorbuffer - On/off: glColorMask)

19-11-2013

Depth buffer

- Depth test - On/off:
  glEnable/glDisable(GL_DEPTH_TEST)
- Write depth buffer - on/off: glDepthMask
- Depth test details: glDepthFunc

- Write to colorbuffer - On/off: glColorMask)

19-11-2013

Stencil Buffer

- Write to stencilbuffer - On/off: glColorMask
- Stencil test - On/off:
  glEnable/glDisable(GL_STENCIL_TEST)
- Write color/depth buffer - on/off: glStencilMask
- Stencil test details: glStencilFunc(func, ref,

19-11-2013

Blending

19-11-2013

Per-Fragment operations

- glEnable/glDisable (…..)
  - GL_DEPTH_TEST
  - GL_STENCIL_TEST
  - GL_BLEND
- glColorMask

Whole Framebuffer - Select buffer for writing

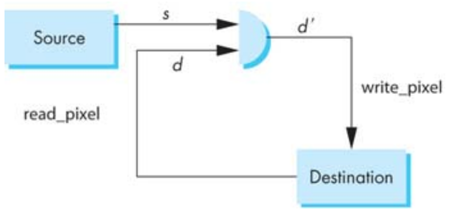
- DrawBuffer (buffer)
  - NONE
  - FRONT
  - BACK
  - LEFT
  - RIGHT

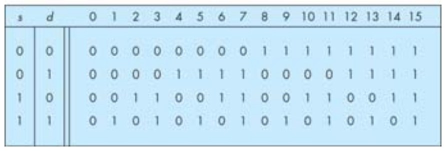
Clearing the Buffer

- glClear (Color_BUFFER_BIT)
- glClearColor (r, g, b, a)
- glClearDepth (d)
- glClearStencil (s)

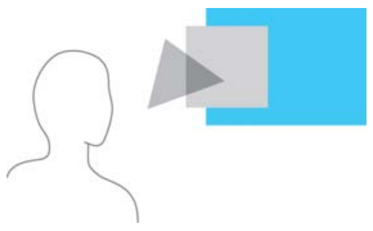
Fine Control - Buffer Update

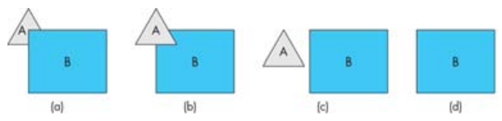
- glColorMask(r, g, b, a)
- glDepthMask (m)
- glStencilMask (m)

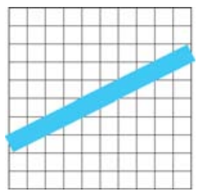


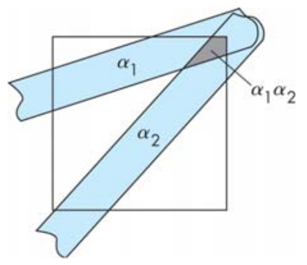


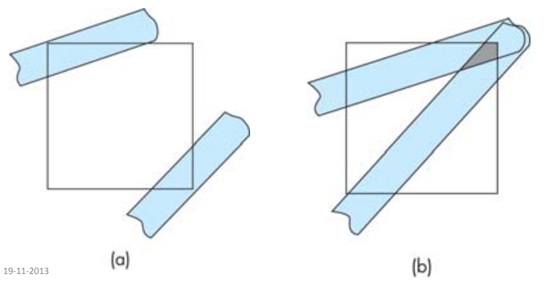
s	d	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

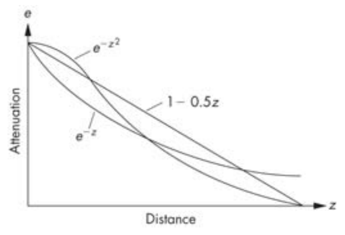


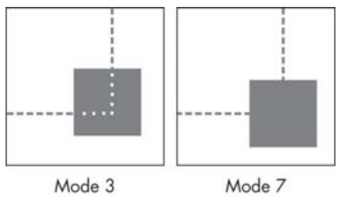




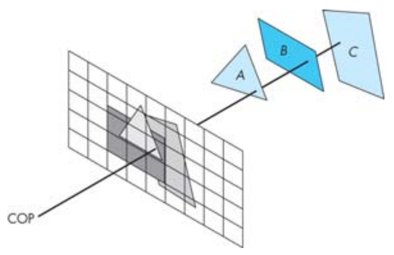

α_1
$\alpha_1\alpha_2$
α_2


(a)
(b)

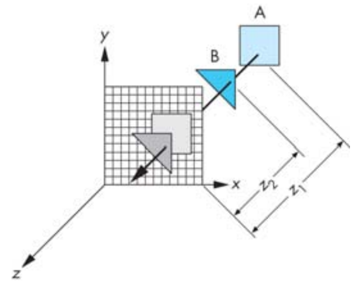

e
e^{-z^2}
$1-0.5z$
e^{-z}
Attenuation
Distance
z


Mode 3
Mode 7

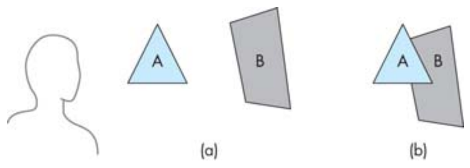
Depth buffer



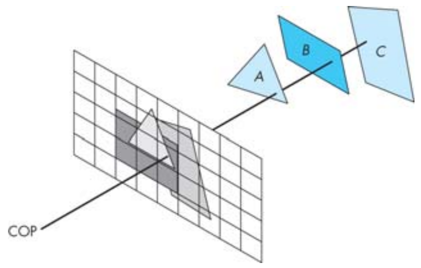
19-11-2013



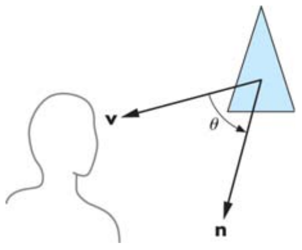
19-11-2013



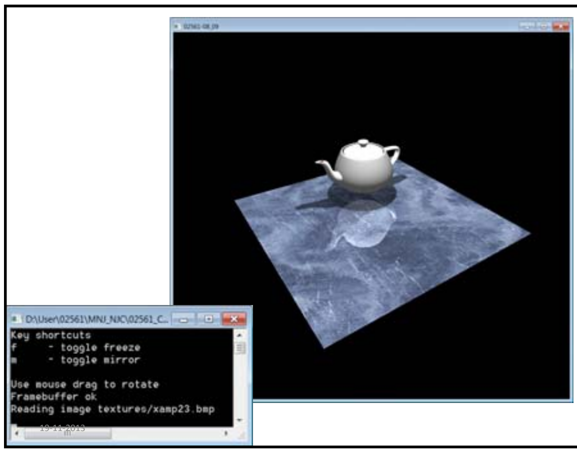
19-11-2013



19-11-2013

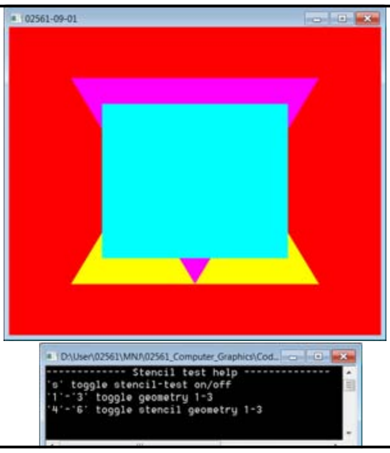


19-11-2013

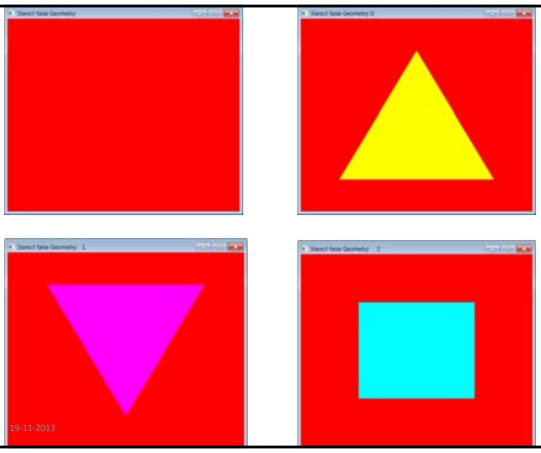


```
Key shortcuts
f     - toggle freeze
m     - toggle mirror

Use mouse drag to rotate
Framebuffer ok
Reading image textures/xamp23.bmp
```

19-11-2013



```
------------- Stencil test help --------------
's' toggle stencil-test on/off
'1'-'3' toggle geometry 1-3
'4'-'6' toggle stencil geometry 1-3
```



```
void setupStencil(mat4& projection, mat4& modelView){
glClear(GL_STENCIL_BUFFER_BIT); // Clear stencil buffer (set values to 0)
glStencilFunc(GL_ALWAYS, 1, 1); //Test always success, value written 1
glColorMask(false, false, false, false); //Disable writting in color buffer
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE); //Stencil & Depth test passes => replace
existing value in stencil buffer

for (int i=0;i<meshes.size();i++){
if (drawStencil[i]){
meshes[i].drawMesh(projection, modelView); // render object to stencil
}
}

glColorMask(true, true, true, true);    //Enable writting in color buffer
glStencilFunc(GL_EQUAL, 1, 1);         //Draw only to color buffer where stencil buffer is 1
glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP); //Stencil buffer read only
}
```

```
if (enableStencil){
glEnable(GL_STENCIL_TEST);
setupStencil(projection, modelView);
}
```

```
void keyboard(unsigned char c, int x, int y){
switch (c){
case '1':
drawMesh[0] = !drawMesh[0];
cout << "draw mesh 0: "<<drawMesh[0]<<endl;
break; case '2':
drawMesh[1] = !drawMesh[1];
cout << "draw mesh 1: "<<drawMesh[1]<<endl;
break; case '3':
drawMesh[2] = !drawMesh[2];
cout << "draw mesh 2: "<<drawMesh[2]<<endl;
break; case '4':
drawStencil[0] = !drawStencil[0];
cout << "draw stencil 0: "<<drawStencil[0]<<endl;
break; case '5':
drawStencil[1] = !drawStencil[1];
cout << "draw stencil 1: "<<drawStencil[1]<<endl;
break; case '6':
drawStencil[2] = !drawStencil[2];
cout << "draw stencil 2: "<<drawStencil[2]<<endl;
break; case 's':
case 'S':
enableStencil = !enableStencil;
cout << "Enable stencil "<<enableStencil<<endl;
break;
```

```
void display() {
glClearColor(1.0, 0.0, 0.0, 1.0); // red background
   glClear(GL_COLOR_BUFFER_BIT);

mat4 projection = Ortho2D(-15.0f, 15.0f, -15.0f, 15.0f);
mat4 modelView;

if (enableStencil){
glEnable(GL_STENCIL_TEST);
setupStencil(projection, modelView);
}

glUseProgram(shaderProgram);

for (int i=0;i<meshes.size();i++){
if (drawMesh[i]){
meshes[i].drawMesh(projection, modelView);
}
}

glDisable(GL_STENCIL_TEST);
glutSwapBuffers();
}
```

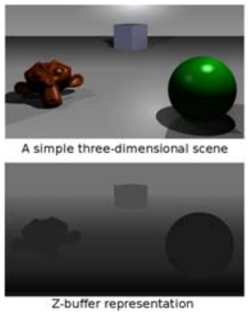
```
GLuint buildFrameBufferObject(int width, int height, GLuint textureId) {
GLuint framebufferObjectId,
renderBufferId;
glGenFramebuffers(1, &framebufferObjectId);
glGenRenderbuffers(1, &renderBufferId);
glBindRenderbuffer(GL_RENDERBUFFER, renderBufferId);
glRenderbufferStorage(GL_RENDERBUFFER, GL_DEPTH_COMPONENT24, width, height);
glBindFramebuffer(GL_FRAMEBUFFER, framebufferObjectId);
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D,
textureId, 0);
glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT,
GL_RENDERBUFFER, renderBufferId);

GLenum frameBufferRes = glCheckFramebufferStatus(GL_DRAW_FRAMEBUFFER);

cout << getFrameBufferStatusString(frameBufferRes)<<endl;

glBindFramebuffer(GL_DRAW_FRAMEBUFFER, 0);
return framebufferObjectId;
}
```

```
void drawMirror(mat4 &projection, mat4 &view) {
vec4 oldClipPlane = clipPlane;

mat4 model;
glDisable(GL_DEPTH_TEST);
glColorMask(false, false, false, false);
glEnable (GL_STENCIL_TEST) ;
glStencilFunc (GL_NEVER, 1 , 1) ;
glStencilOp (GL_REPLACE, GL_KEEP, GL_KEEP) ;
drawMeshObject(projection,  model, view, planeObject);
glEnable (GL_DEPTH_TEST);
glStencilFunc (GL_EQUAL , 1 , 1 );
glStencilOp (GL_KEEP, GL_KEEP, GL_KEEP);
glColorMask(true, true, true, true);

clipPlane = vec4(0,-1,0,0);
model = Scale(1,-1,1) * Translate(teapotPosition);
glFrontFace(GL_CW);
drawMeshObject(projection, model, view, teapotObject);
glFrontFace(GL_CCW);
glDisable (GL_STENCIL_TEST);
clipPlane = oldClipPlane;
}
```

-

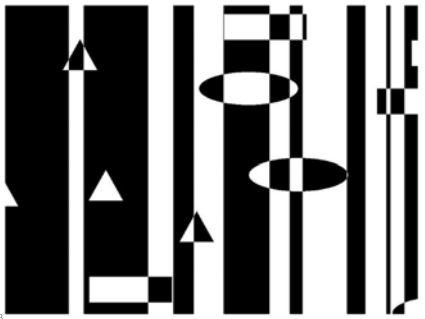
```
void drawPlane(mat4 projection, mat4 view) {
if (draw_mirror == 1) {
teapotObject.color = vec4(0.5f, 0.5f, 0.5f, 0.5f);
drawMirror(projection, view);
teapotObject.color = vec4(1.0f, 1.0f, 1.0f, 1.0f);
}
glClear(GL_DEPTH_BUFFER_BIT);
glEnable(GL_BLEND);
glBlendFunc(GL_ONE, GL_ONE);
planeObject.color = vec4(0.7f, 0.3f, 0.5f, 1.0f);
mat4 model;
drawMeshObject(projection,  model, view, planeObject);
glDisable(GL_BLEND);
}
```

Z-buffer – Depth-buffer



19-11-2013

Stencil Buffer – user example



19-11-2013

Stencil Buffer - Example

- Color of shapes change as they pass over other shapes
- Stencil buffer is filled with 1s where-ever a white stripe is drawn and 0s elsewhere
- A black colored shape is drawn where the stencil buffer is 0, and a white shape is drawn where the buffer is 1

19-11-2013

19-11-2013

19-11-2013

```
glEnable(GL_STENCIL_TEST); // by default not enabled
glStencilMask(stencilMask); // allow writing to stencil buffer, by default (0xFF) no mask.
glClearStencil(clearStencilValue); // clear stencil value, by default = 0
glStencilFunc(func, ref, mask); // by default GL_ALWAYS, 0, 0xFF, always pass stencil test
glStencilOp(fail,zfail,zpass); // by default GL_KEEP, GL_KEEP, GL_KEEP, dont change stencil buffer
glClear(GL_STENCIL_BUFFER_BIT); // clear stencil buffer, fill with (clearStencilValue & stencilMask
```

19-11-2013