

## Stencil Buffer & Decals

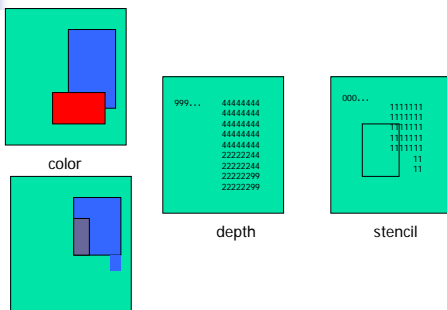
- Decals
- Stencil buffer & OpenGL commands
- Using the stencil buffer to apply polygonal decals
- Using the stencil buffer to apply text decals

## Decals

2 step process:

1. As surface to be stenciled is written into frame buffer, mark what pixels it modifies
2. Scan convert decal into frame buffer restricted to the pixels marked in step 1

## Decals



## Stencil Buffer

- Same spatial resolution as color and depth buffers
- Usually (and at least) 8-bits, but can vary
- Used to hold values related to elements being written into frame buffer



## OpenGL Commands

- `glStencilFunc()` - sets function to test stencil bits with
- `glStencilMask()`, `glStencilMaskSeparate()` - specifies which bits in Stencil Buffer are involved
- `glStencilOp()`, `glStencilOpSeparate()` - specifies operation to perform as result of stencil test and depth test



## `glStencilFunc()`

- `glStencilFunc(GLenum func, GLint ref, GLuint mask)`
- Specifies test to perform on reference value and masked bits in stencil buffer
- *func* - test function e.g., `GL_LEQUAL`, `GL_ALWAYS`
- *ref* - reference value for test
- *mask* - ANDed with *ref* & stencil value - selects what bits to use



## `glStencilMask()`

- `glStencilMask(GLuint mask)`
- Enables and disables writing of individual bits in the stencil planes



## `glStencilMaskSeparate()`

- `glStencilMaskSeparate(GLenum face, GLuint mask)`
- Face - `GL_FRONT`, `GL_BACK`, `GL_FRONT_AND_BACK`
- Enables and disables writing of individual bits in the stencil planes

## glStencilOp()

- `glStencilOp(GLenum sfail, GLenum dpfail, GLenum dppass)`
- Specifies what action to take as a result of stencil test and depth test: `GL_KEEP`, `GL_ZERO`, `GL_REPLACE`, etc.
- `sfail` - fails stencil test
- `dpfail` - passes stencil test, fails depth test
- `dppass` - passes both stencil and depth test

## glStencilOpSeparate()

- `glStencilOpSeparate(GLenum face, GLenum sfail, GLenum dpfail, GLenum dppass)`
- Specifies what action to take as a result of stencil test and depth test: `GL_KEEP`, `GL_ZERO`, `GL_REPLACE`, etc.
- `sfail` - fails stencil test
- `dpfail` - passes stencil test, fails depth test
- `dppass` - passes both stencil and depth test

## Applying decals

- Draw decal right on top of surface
- Draw it into buffer wherever surface was drawn (don't draw it where surface is not visible)
- Don't do depth testing

## Step 1

- Put '1' in stencil buffer wherever surface is drawn in frame buffer

```
glEnable(GL_STENCIL_TEST); // enable the stencil test
glStencilFunc(GL_ALWAYS,1,1); // always place a 1 in the stencil buffer
glStencilOp(GL_KEEP,GL_ZERO,GL_REPLACE);
// if stencil fails (it won't - why?), keep stencil value
// else if depth fails, put a zero in stencil buffer
// else, replace value in stencil buffer with ref
draw base polygon
```



## Step 2

---

- Draw decal wherever stencil has a '1'

```
glStencilFunc(GL_EQUAL,1,1); // test if 1 bit is set in stencil buffer,
glStencilMask(GL_FALSE);   // turn off stencil writing ('0' ?)
glDisable(GL_DEPTH_TEST);  // don't do depth test (so it 'passes')
draw decal polygon
```

```
glEnable(GL_DEPTH_TEST);
glDisable(GL_STENCIL_TEST);
```



## Step 2 for text

---

- Draw text decal wherever stencil has a '1'

```
// with depth test off and stencil writing off as in previous slide
glLineWidth(6);
set_material("whiteMatteMaterial");
<transforms to get it where you want it to go>
glStencilFunc(GL_EQUAL,1,1); // if 1 bit is set,
glutStrokeCharacter(GLUT_STROKE_MONO_ROMAN,'5');
glutStrokeCharacter(GLUT_STROKE_MONO_ROMAN,'8');
glutStrokeCharacter(GLUT_STROKE_MONO_ROMAN,'1');
```