Blending

- Blending and Alpha blending
- Depth buffer and Depth test
- Alpha test
- Frame buffer
- Stencil buffer

Ana Gil Luezas
Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

Colores RGBA

El valor por defecto de la componente Alpha es 1 (≈255):

Un color RGB se completa con A=1 (opaco)

Se utiliza para modelar objetos traslúcidos:

A = 1 -> opaco

A = 0 -> transparente

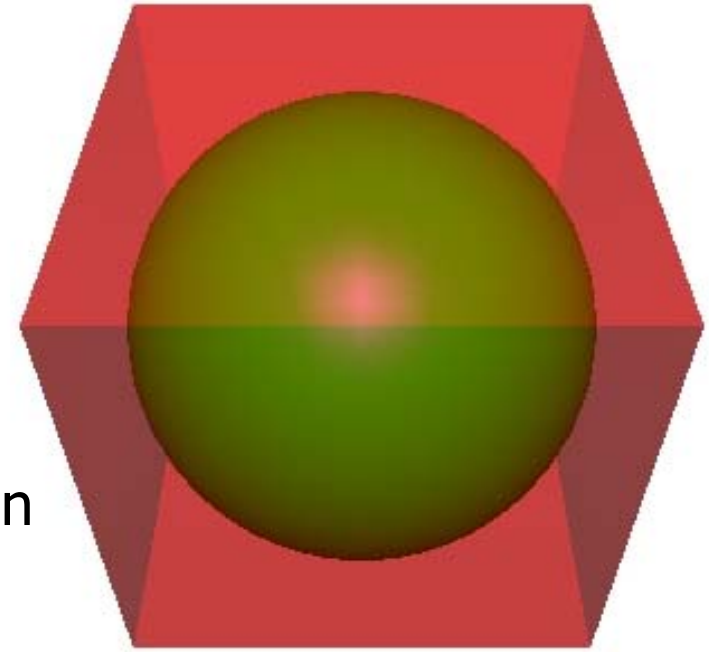
Los colores de objetos que se superponen en la vista se combinan para determinar el color final:

Cuando un objeto traslúcido

aparece delante de otro

Blending: suma ponderada de los colores correspondientes a distintos objetos.

Alpha blending:
los factores de ponderación se determinan en función de la componente alpha.



Depth Buffer and depth test

El *depth buffer* o *Z-buffer* contiene la distancia con respecto a la cámara (al plano cercano) de cada píxel (componente Z del fragmento). Los valores están en el rango [0, 1], siendo 0 el más cercano y 1 el más lejano.

Inicialización:

```
glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH );
// en IG1App::iniWinOpenGL()
glEnable(GL_DEPTH_TEST);     // en scene.init()
```

Cada vez que se renderiza: void display()

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

El *back color buffer* queda con el color de fondo en todos los píxeles, y el *Z-buffer* con el valor 1 en todos los píxeles.

```
scene.render(camera);          // ->
glutSwapBuffers();
```

Depth Buffer and depth test

- Al realizarse la proyección de los vértices, se calcula la distancia relativa a la cámara y se guarda en la componente z del vértice.
- Se realiza el relleno de triángulos: genera los fragmentos del interior mediante interpolación de los valores de los vértices.

  Datos de cada fragmento: coordenadas de ventana (x, y, z) (proyectadas y ajustadas al puerto de vista), color (r, g, b, a), coordenadas de textura (s, t)
- Y se procesa cada fragmento

  Depth test por defecto GL_LESS (se puede configurar):

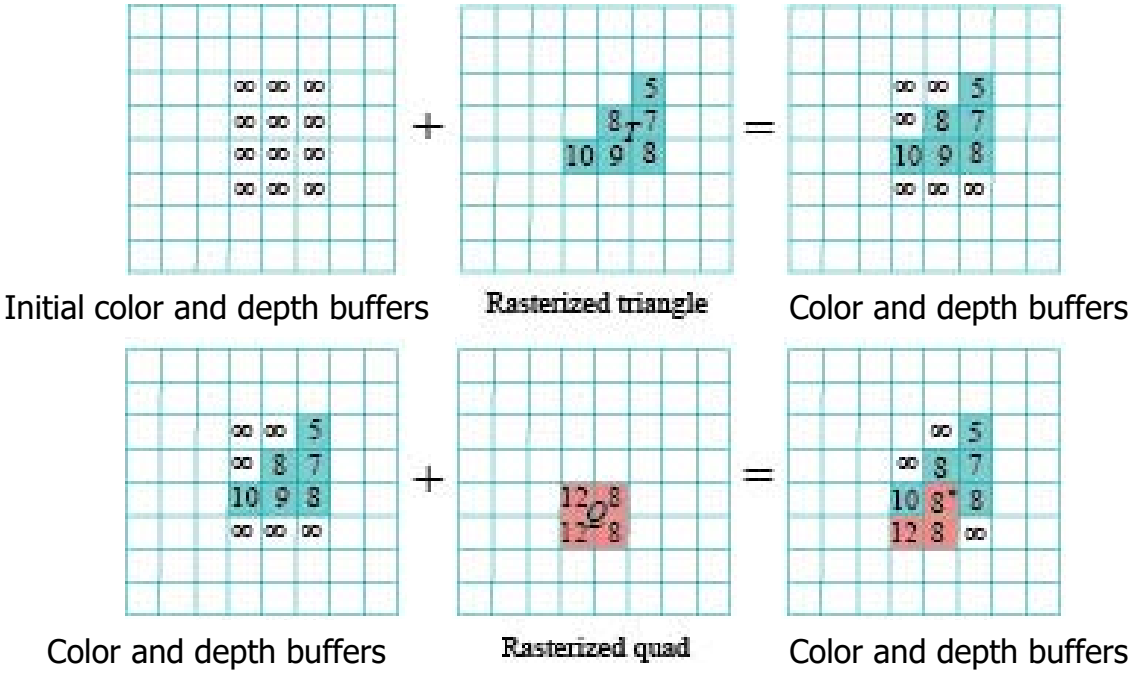
  Cuando se procesa un fragmento, se compara la distancia del fragmento con el valor del Z-buffer.

  Si es menor, el fragmento en proceso reemplaza el valor de ambos buffers (el de colores y el de profundidad).

  En otro caso, el fragmento queda descartado y no modifica ningún buffer.

Color and Depth Buffers

- glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  T->render(…); // triángulo azul opaco
  Q->render(…); // rectángulo rojo traslúcido



Con el test de profundidad por defecto activo

y sin blending

↓

Un fragmento pasa el test si su profundidad es menor que la del buffer

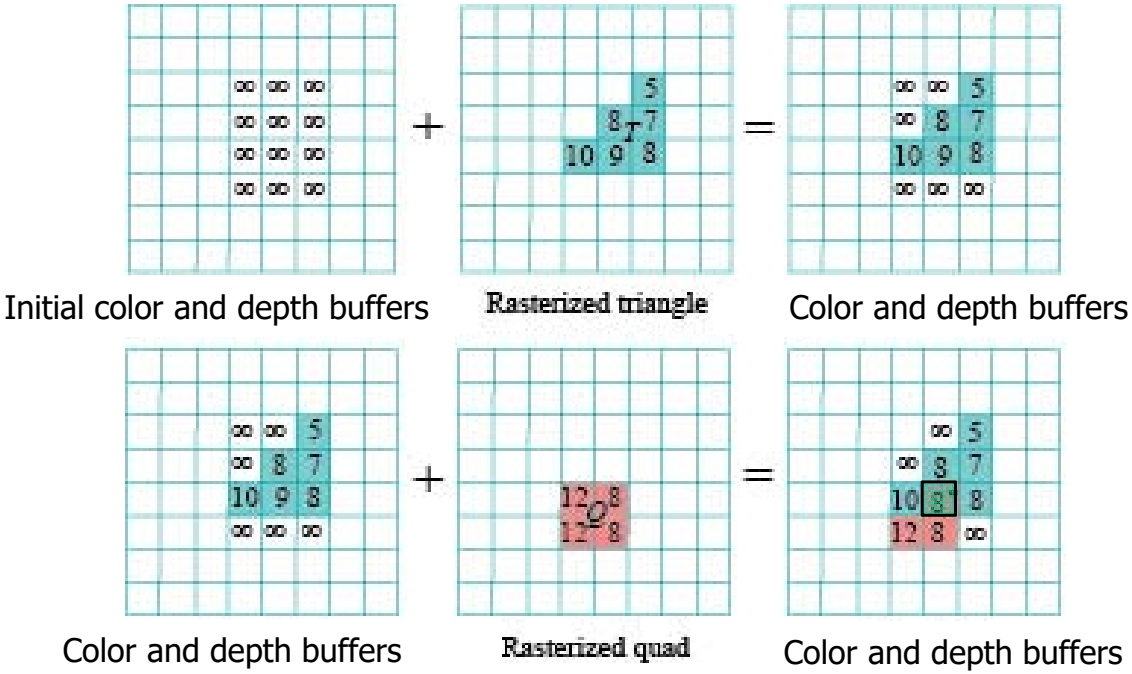
y sobrescribe ambos buffers con los valores del nuevo fragmento

Color and Depth Buffers

- glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  T->render(...); // triángulo azul opaco

  Q->render(...); // rectángulo rojo traslúcido



Con el test de profundidad por defecto activo

y blending activo

Un fragmento pasa el test si su profundidad es menor que la del buffer

y mezcla el color del buffer con el color del fragmento. El valor del Z-buffer se reemplaza

Blending equation

Activar y desactivar el *Blending*

glEnable(GL_BLEND) / glDisable(GL_BLEND)

Ecuación: la mezcla de los dos colores se obtiene con los factores de blending que estén establecidos

dstColor = srcBFactor * srcColor + dstBFactor * dstColor

siendo

srcColor = (srcR, srcG, srcB, srcA) el color RGBA del fragmento en proceso (source Color),

dstColor = (dstR, dstG, dstB, dstA) el color del Color Buffer correspondiente al mismo píxel (destination Color)

y srcBFactor y dstBFactor los correspondientes factores de blending

Blending equation

Configuración de los factores de la ecuación:

dstColor = srcBFactor * srcColor + dstBFactor * dstColor

Los factores de la ecuación se establecen con:

glBlendFunc(srcBFactor, dstBFactor) ambos factores en [0, 1]:

glBlendFunc(1, 0) -> valores por defecto

glBlendFunc(0.5, 0.5) -> mismo peso para los dos colores

Alpha blending: se utiliza la componente Alpha de los colores

glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

Alpha Blending

Para transparencias se utiliza la componente Alpha de los colores

glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

Si el fragmento en curso pasa el test de profundidad y

- es opaco (srcA = 1) -> srcBF =1 y dstBF = 0

  reemplaza al color del buffer

- es transparente (srcA=0) -> srcBF =0 y dstBF = 1

  el color del buffer no se modifica

- es translucido (por ejemplo srcA=0.5) -> srcBF = dstBF = 0.5

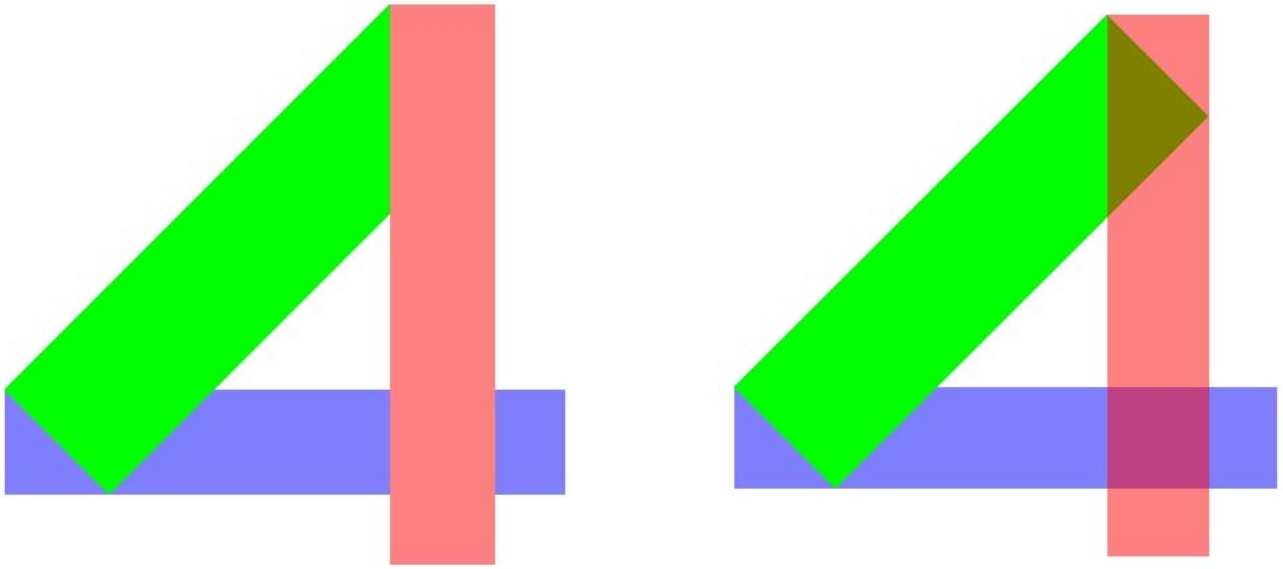
  el color del buffer se multiplica por 0.5 y se suma el del fragmento también multiplicado por 0.5

Alpha Blending

- glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  Rectángulos: rojo traslúcido (cercano), azul translúcido (lejano) y verde opaco (en medio)

  Con el test de profundidad por defecto y blending activos



Orden: rojo, verde, azul -> azul, verde, rojo

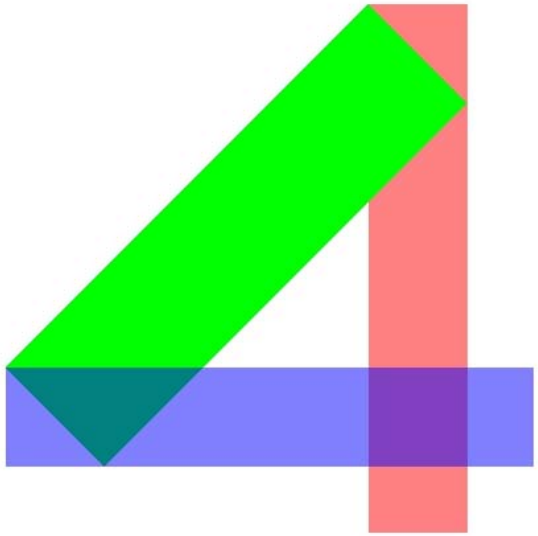
La profundidad es relativa al punto de vista (posición de la cámara)

Alpha Blending

- glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  Rectángulos: rojo traslúcido (cercano), azul translúcido (lejano) y verde opaco (en medio)

  Test de profundidad desactivado y blending activo



Orden: rojo, verde, azul

Alpha Blending

- Orden de renderizado con objetos opacos y traslúcidos

  1- Test de profundidad activado por defecto

  Dibujar los objetos opacos

  2- Usar el buffer de profundidad solo para lectura:

  glDepthMask(GL_FALSE);

  realiza el test pero no modifica el Z-Buffer

  glEnable(GL_BLEND); glBlendFunc(...);

  Dibujar los objetos traslúcidos

  Los que están delante de los opacos mezclarán el color

  3- glDepthMask(GL_TRUE)

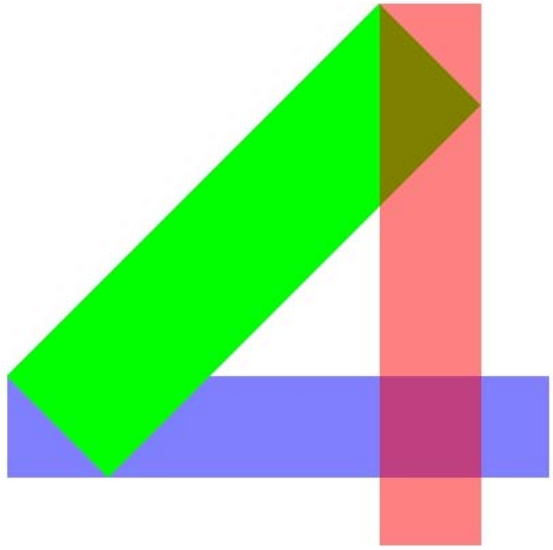
  glDisable(GL_BLEND)

Alpha Blending

- glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  Rectángulos: rojo traslúcido (cercano), azul translúcido (lejano) y verde opaco (en medio)

  Test de profundidad activado para opacos y solo lectura para traslucidos (y blending activo)



Orden (primero opacos): verde, rojo, azul

Alpha test

Las texturas con fondo transparente pueden corresponder a objetos:

- Opacos: se renderizan sin blending y con el depth test habitual, pero para que el fondo no afecte al renderizado, hay que activar el alpha test y configurarlo con glAlphaFunc(GL_GREATER, 0.0):
  Solo pasan el test los fragmentos No trasparentes.

- Traslúcidos: se renderizan con blending y desactivando la escritura en el depth buffer. También se puede utilizar el alpha test.

Alpha test

Alpha test: no utiliza buffer, permite descarta los fragmentos cuya componente alpha NO cumple la función:

```
glAlphaFunc(cond, val);
```

Que podemos configurar:

cond: GL_GREATER, GL_EQUAL, GL_LESS, …

val: un alpha valor en [0, 1]

Por ejemplo, para descartar los fragmentos transparentes (alpha = 0)

```
glAlphaFunc(GL_GREATER, 0.0);
```

El test se activa / desactiva con

```
glEnable(GL_ALPHA_TEST)   / glDisable(GL_ALPHA_TEST)
```

Texturas translucidas

- El método load de la clase Texture permite cargar imágenes bmp con distintos valores de transparencia para el canal alpha.

  void load(const string & BMP_Name, GLubyte alpha = 255);

  Por defecto el canal alpha de todos los texels se establece a opaco (255), pero podemos especificar cualquier valor entre 0 y 255 para hacer que la textura sea translúcida.

  Otras opciones:

  void load(const string & BMP_Name, u8vec3 color, GLubyte alpha);

  Para hacer que los texels de un color concreto tengan un valor alpha concreto. Los demás texels quedan opacos. Habitualmente se usa para hacer transparente el color de fondo.

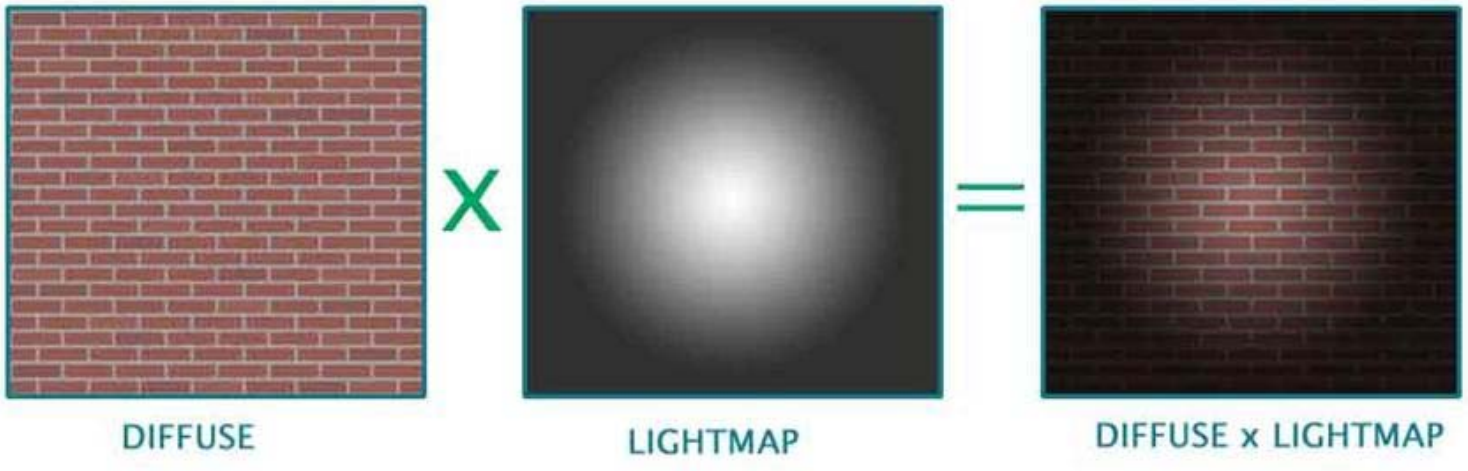
  void load(const string & BMP_Name, u8vec3 colorKey, GLubyte alphaKey , GLubyte alpha);

Multitexturas

Se puede asociar más de una textura a un objeto, y combinar los colores de las imágenes para obtener el color del objeto.

Hay que utilizar tantas unidades de textura como imágenes queramos utilizar simultáneamente.

Por ejemplo: una imagen se mezcla con otra para añadir luz a la original, renderizando el rectángulo con dos unidades de textura activas



Multitexturas

Frame Buffer

El Frame Buffer consta de varios buffers del mismo tamaño

1. Colors buffers: front and back
2. Depth buffer: depth test
3. Stencil buffer: stencil test

Se configura al crear la ventana

```
glutInitDisplayMode( GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH |
                     GLUT_STENCIL);
```

Se reinician a los valores establecidos con

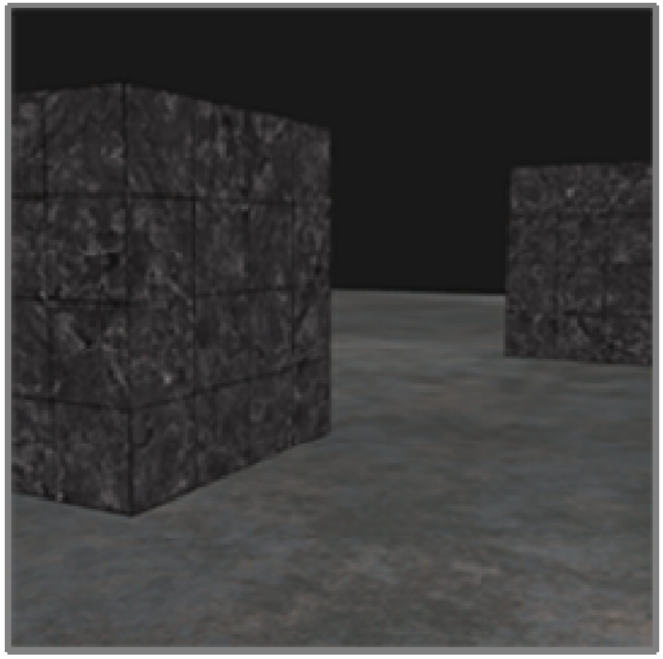
```
glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT |
         GL_STENCIL_BUFFER_BIT);
```

Los test permiten descartar fragmentos para que no aporten color al Color Buffer.

Se pueden utilizar frame buffers auxiliares (Frame Buffer Objects)

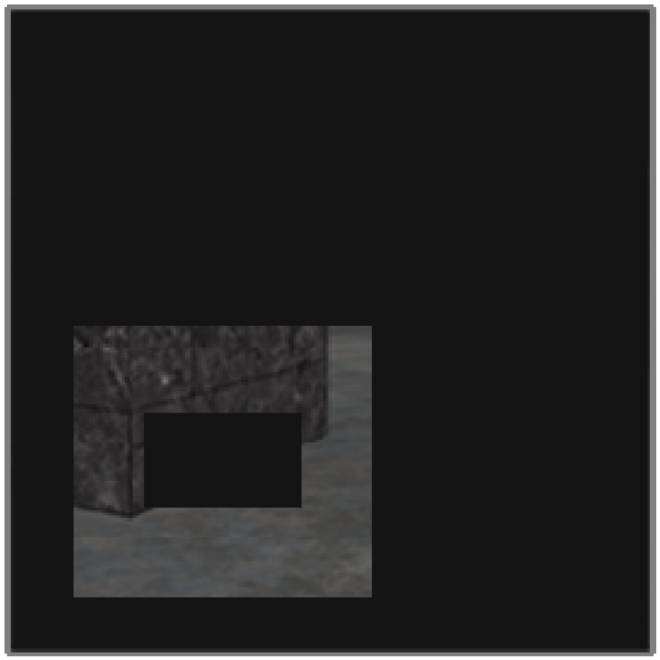
Stencil Buffer

1. Activar la escritura en el Stencil Buffer.
2. Renderizar objetos específicos escribiendo solo en el stencil buffer.



Color buffer

Stencil buffer



After stencil test

3. Desactivar la escritura en el Stencil buffer.
4. Renderizar objetos utilizando el contenido del Stencil buffer con el test