# Rendera till textur

Man kan rendera scener som inte visas direkt, utan som läggs i en textur för att användas i senare renderingssteg.

Exempel på tillämpningar:

- Environment mapping på “riktigt”. Rendera omgivningen till textur.
- Spatiella filtreringar
- Temporalfiltrering
- Viktigt för skuggor

Möjliggör multipassrenderingar.

# Rendera till textur

Kan göras på flera sätt:

glReadPixels + glTexImage
Dålig, går via CPU.

glCopyTexImage
glCopyTexSubImage
Kopiering inom VRAM

pBuffers (Pixel buffers)
Frame buffer objects (FBO) - preferred!
Skriv direkt till egen buffer.

# Framebuffer objects, användning

Skapa referens, samma princip som för texturer:

```
glGenFramebuffers(1, &fb);
```

Ange textur:

```
glFramebufferTexture2D(GL_FRAMEBUFFER,
GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, tex, 0);
```

Dito för renderbuffer:

```
glGenRenderbuffers(1, &rb);
glBindRenderbuffer(GL_RENDERBUFFER, rb);
glRenderbufferStorage(GL_RENDERBUFFER,
GL_DEPTH_COMPONENT24, width, height);
```

# Framebuffer objects, användning

Välj aktiv FBO:

```
glBindFramebuffer(GL_FRAMEBUFFER, fb)
```

Stäng av FBO, rendera till vanliga frame buffern:

```
glBindFramebuffer(GL_FRAMEBUFFER, 0)
```

OBS! Gamla OpenGL-installationer lägger till en massa EXT-suffix!

# Debugging av FBO

Under utveckling bör man fånga upp felmeddelanden från FBO. Det görs med glCheckFramebufferStatus.

Typiskt problem: Om man inte har rätt paramerar till sin textur så kan den underkännas som textur till en FBO! Destinationstexturen bör vara inställd på närmaste granne-interpolation.

Det är svårare att få ihop en fungerande FBO än man tror! (Vi har färdig kod till labbarna!)

# Exempel med glCopyTexSubImage

```
void display()
{
    glBindTexture(GL_TEXTURE_2D, minitexid);

    glViewport(0, 0, width, height);

// Draw what should go in the texture
    glClearColor(1,1,0.5,0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    DrawTextureScene();
    glFlush();

// Copy result to the texture
    glBindTexture(GL_TEXTURE_2D, tex);
    glCopyTexSubImage2D(GL_TEXTURE_2D, 0, 0, 0, 0, 0, width, height);

    glViewport(0, 0, lastw, lasth);

// Render final image using the generated texture
    glClearColor(0.3, 0.3, 0.7,0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    DrawMainScene();
    glutSwapBuffers();
}
```

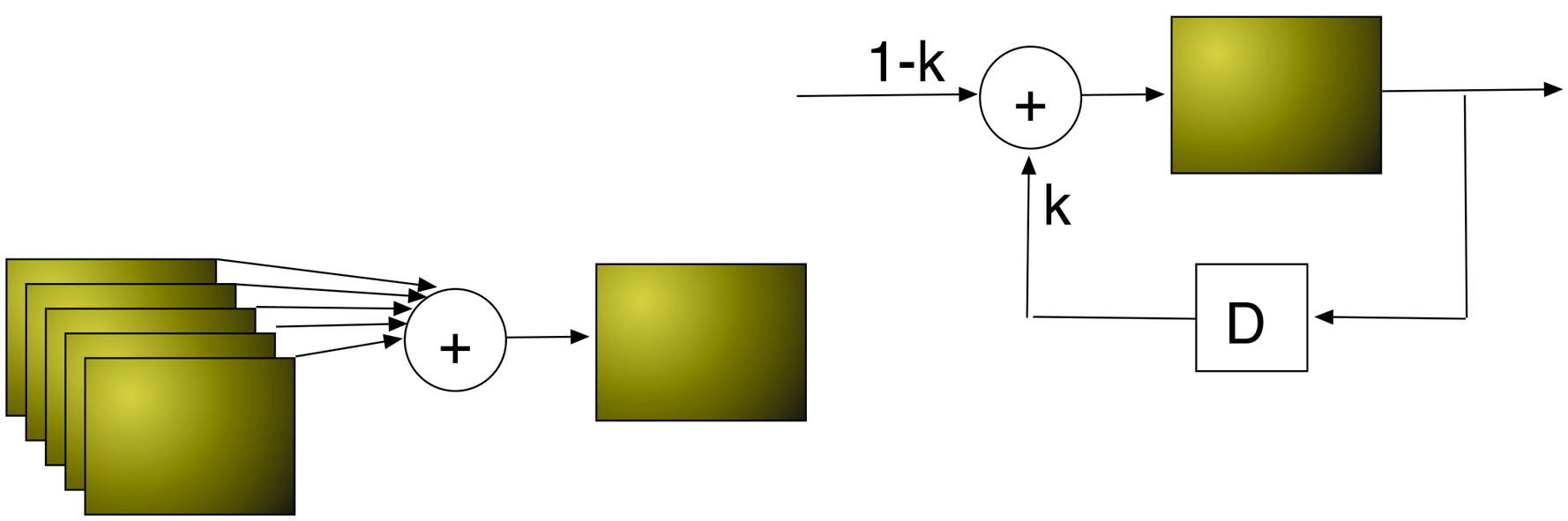
# Exempel med FBO

```
void display()
{
// render to the FBO
    glBindFramebuffer(GL_FRAMEBUFFER, fb);
    glBindTexture(GL_TEXTURE_2D, minitexid);

    glViewport(0, 0, width, height);

// (draw something here, rendering to texture)
    glClearColor(1,1,0.5,0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    DrawTextureScene();

    glViewport(0, 0, lastw, lasth);

// render to the window, using the texture
    glBindFramebuffer(GL_FRAMEBUFFER, 0);
    glBindTexture(GL_TEXTURE_2D, tex);

    glClearColor(0,0,0,0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    DrawMainScene();
    glutSwapBuffers();
}
```

# Temporalfiltrering

Filtrering i tiden

Enkel tillämpning av att rendera till textur

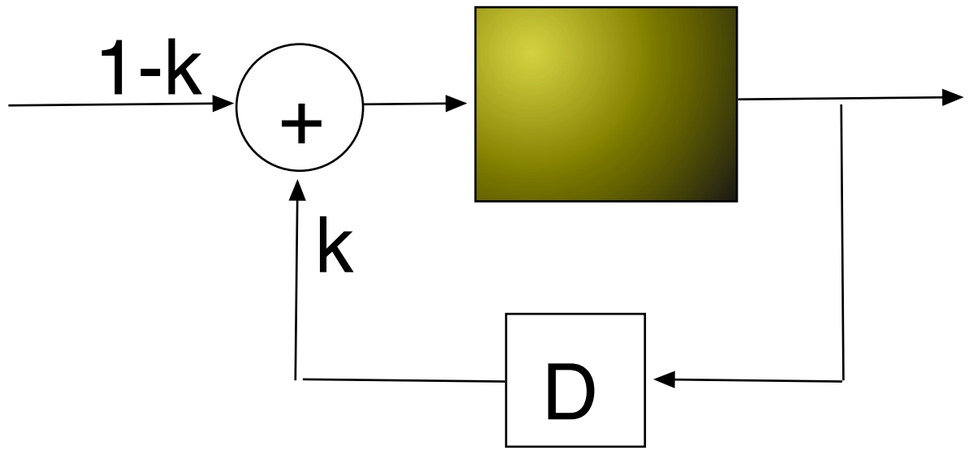


# Rekursiv temporalfiltrering

Ger en ”eftersläpning”

Bilden sammanvägs med en tidigare

Kräver bara att en enda äldre bild sparas!



# Rekursiv temporalfiltrering på GPU

- Rendera scenen. Godtyckling innehåll plus en bakgrundtextur, RTF-buffer. Varje pixels värde sammanvägs med bakgrundstexturen över hela bilden

Om du använder glCopyTexSubImage: Kopiera den färdiga bilden till texturen.

(Något annorlunda om man använder FBO.)

# CPU-kod

Select the texture (RTF buffer)

```
glActiveTexture( GL_TEXTURE1 );
glBindTexture(GL_TEXTURE_2D, rtftex);
```

Render scene.

Copy texture:

```
glBindTexture(GL_TEXTURE_2D, rtftex);
glCopyTexSubImage2D(GL_TEXTURE_2D, 0, 0, 0, 0,
                    0, 512, 512);
```

# GPU-kod

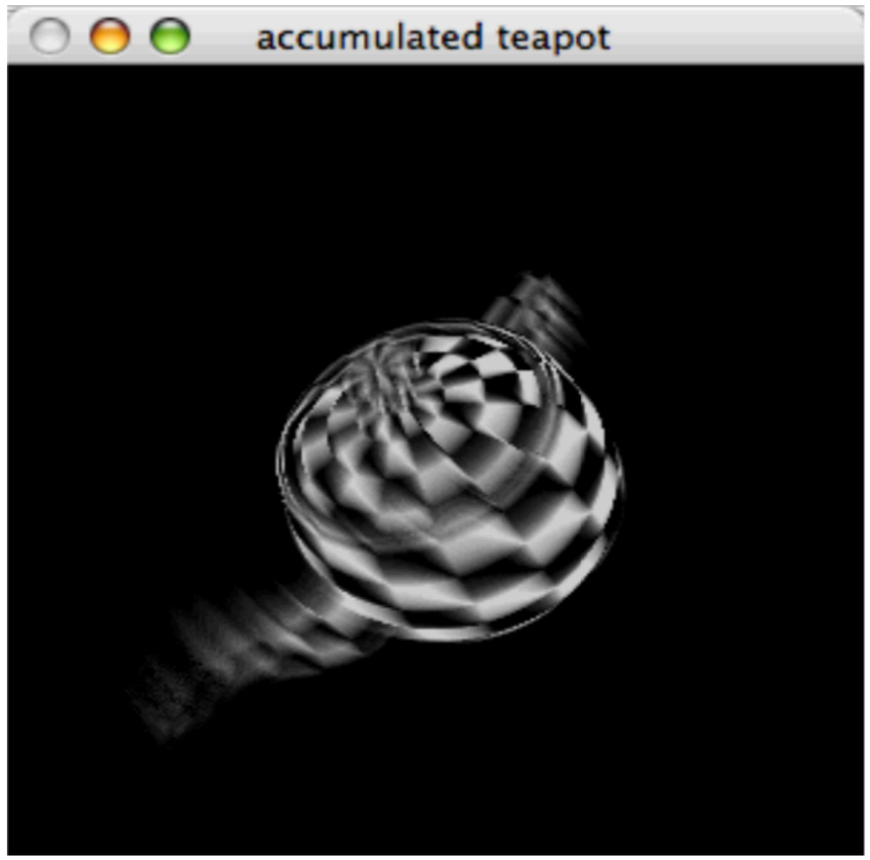
Vertexshader: Skärmkoordinater till texturkoordinater

```
out vec2 screenCoord;
screenCoord = vec2(gl_Position) / gl_Position.w
    / 2.0 + vec2(0.5);
```

Fragmentshader: utför sammanvägning mellan renderingsresultat och textur:

```
outColor = texture(tex, texCoord) * (1.0-k) +
    texture(rtftex, screen- Coord) * k;
```

# Rörelseoskärpa



# Vi har sett...

- Stencilbuffern
- Projicerade texturer
- Rendering till textur

=> Nu har vi viktiga verktyg för flera sorters skuggor!
(Nästa föreläsning!)