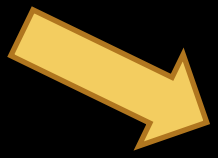


TEKSTURE

Teksture

- stvarne površine, pogotovo ako su od prirodnih materijala (drvo, stijena) imaju neki karakterističan uzorak i to je ono što zovemo tekstura
- iscrtavanje takvih nepravilnih uzoraka i šara pomoću osnovnih grafičkih elemenata kao što su linije i trokutići je neuvjerljivo i vrlo neefikasno
- OpenGL omogućava pridruživanje („lijepljenje”) sličica na plohe i na taj način može se simulirati izgled teksture i stvoriti privid nekog materijala ili postići začudne vizualne efekte

Primjer

- tekstura drveta dobivena je fotografiranjem radnog stola i pridružena površini kvadra
- etiketa je skinuta s boce, skenirana i potom „nalijepljena” na valjak

Teksture u WebGL-u

- „čisti” OpenGL-u (recimo u C-u) nema rutine koje bi omogućavale manipulaciju sa slikama tako da za učitavanje slika treba koristiti neke druge knjižnice ili to treba samostalno isprogramirati (izvedivo za neke jednostavnije grafičke formate)
- prednost WebGL-a je da za teksture može koristiti već postojeći i dobro podržani DOM objekt Image() pa je rad sa slikama znatno pojednostavljen
- nažalost, sve ostalo je preuzeto iz OpenGL-a i oslanja se na uobičajenu paradigmu stroja stanja te je stoga izrazito arhaično (i gotovo iritantno)

Učitavanje tekstura

```
async function loadImage(url) {
   const slika = new Image();
   slika.src = url;
   try {
     await slika.decode();
   } catch(err) { console.log('Greska: ’, err); }
   return slika;
 }

 async function WebGLaplikacija() {
   var slika = await loadImage('starigrad.jpg');
   if (!slika.width) { alert('Nema teksture!’); return; }

   ...
```

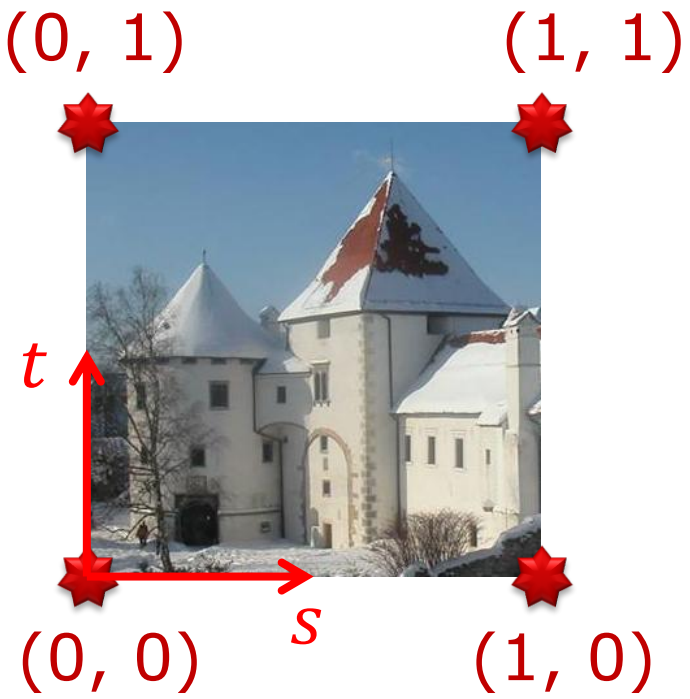
Inicijalizacija

```
tekstura0 = gl.createTexture();
gl.bindTexture(gl.TEXTURE_2D, tekstura0);

// postavljanje raznih parametara
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER,
     gl.LINEAR); // mora se definirati
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER,
     gl.LINEAR); // ovo je default

// preokreće sliku tako da je donji kut (0, 0)
gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, true);

// dodavanje slike (DOM objekt Image())
gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA,
  gl.UNSIGNED_BYTE, slika);
```

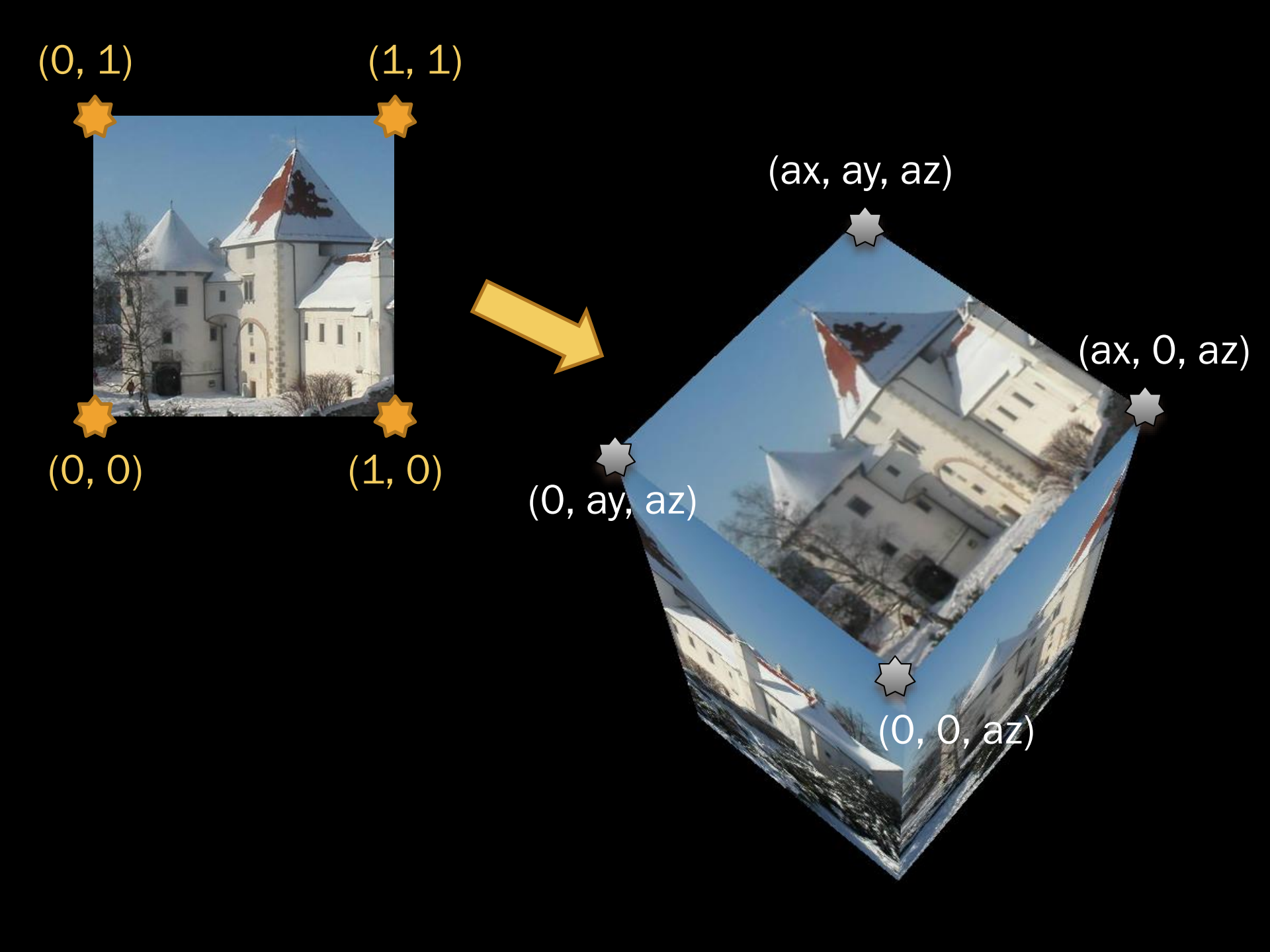
Vezanje (lijepljenje) tekstura

- ukoliko se tekstura želi nalijepiti na neku plohu koordinatama vrhova trokuta koji definiraju tu plohu treba pridružiti koordinate tekstura
- koordinate tekstura prenose se u programe za sjenčanje kao `attribute` varijable pri čemu se mogu koristiti i isprepleteni spremnici
- svakom vrhu pridružuje se koordinata tekstura, tj. koja točka slike treba biti preslikana u taj vrh
- interpolacijom u procesoru za sjenčanje fragmenata svakom fragmentu pridružuje se odgovarajući dio slike

Koordinate tekstura s i t



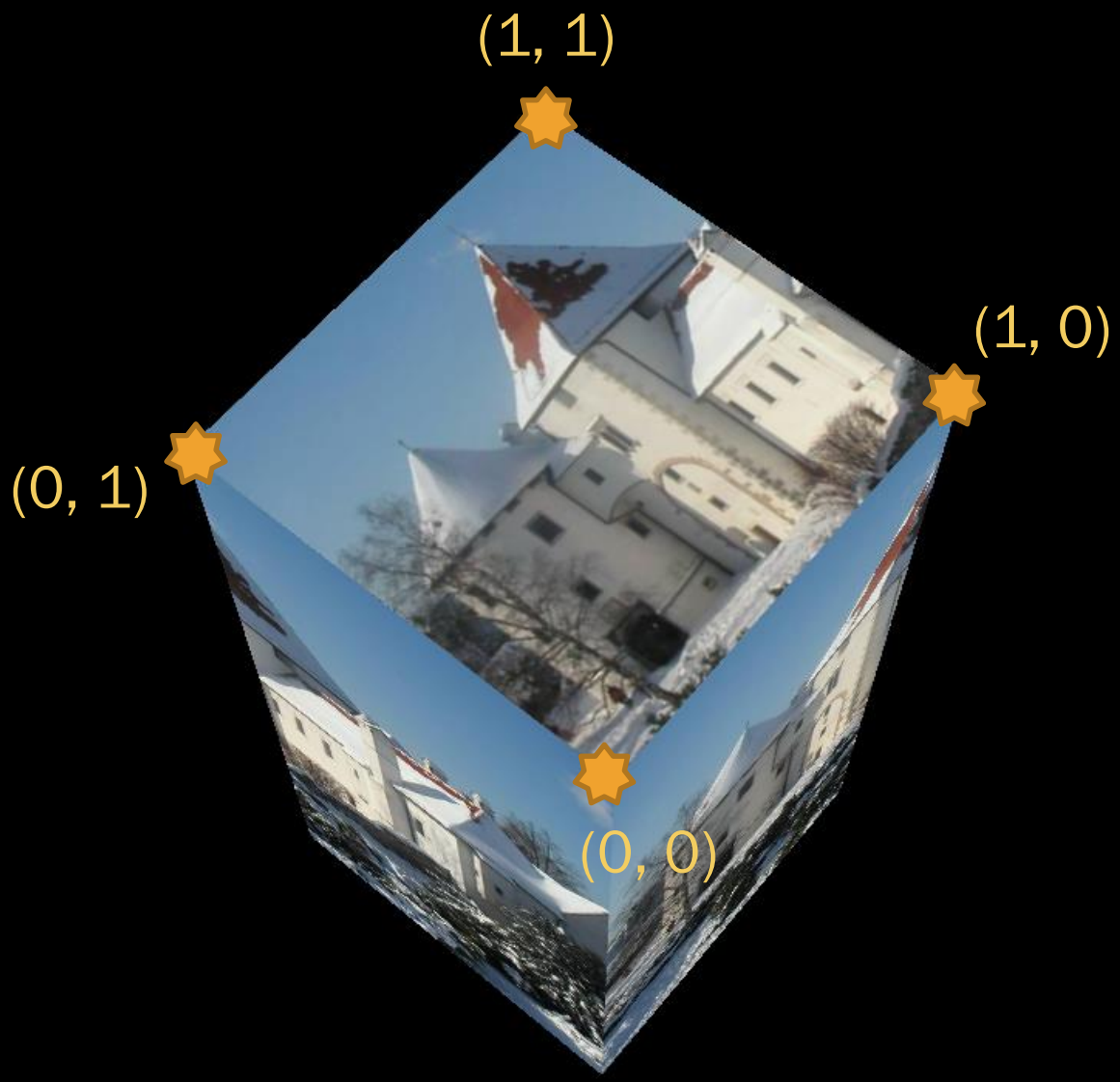
- teksturi se pridružuju koordinate koje se umjesto sa x i y uobičajeno označavaju sa s i t (ponekad u i v)
- uvijek su normirane od 0 do 1 (to vrijedi i kad tekstura nije kvadratična!)

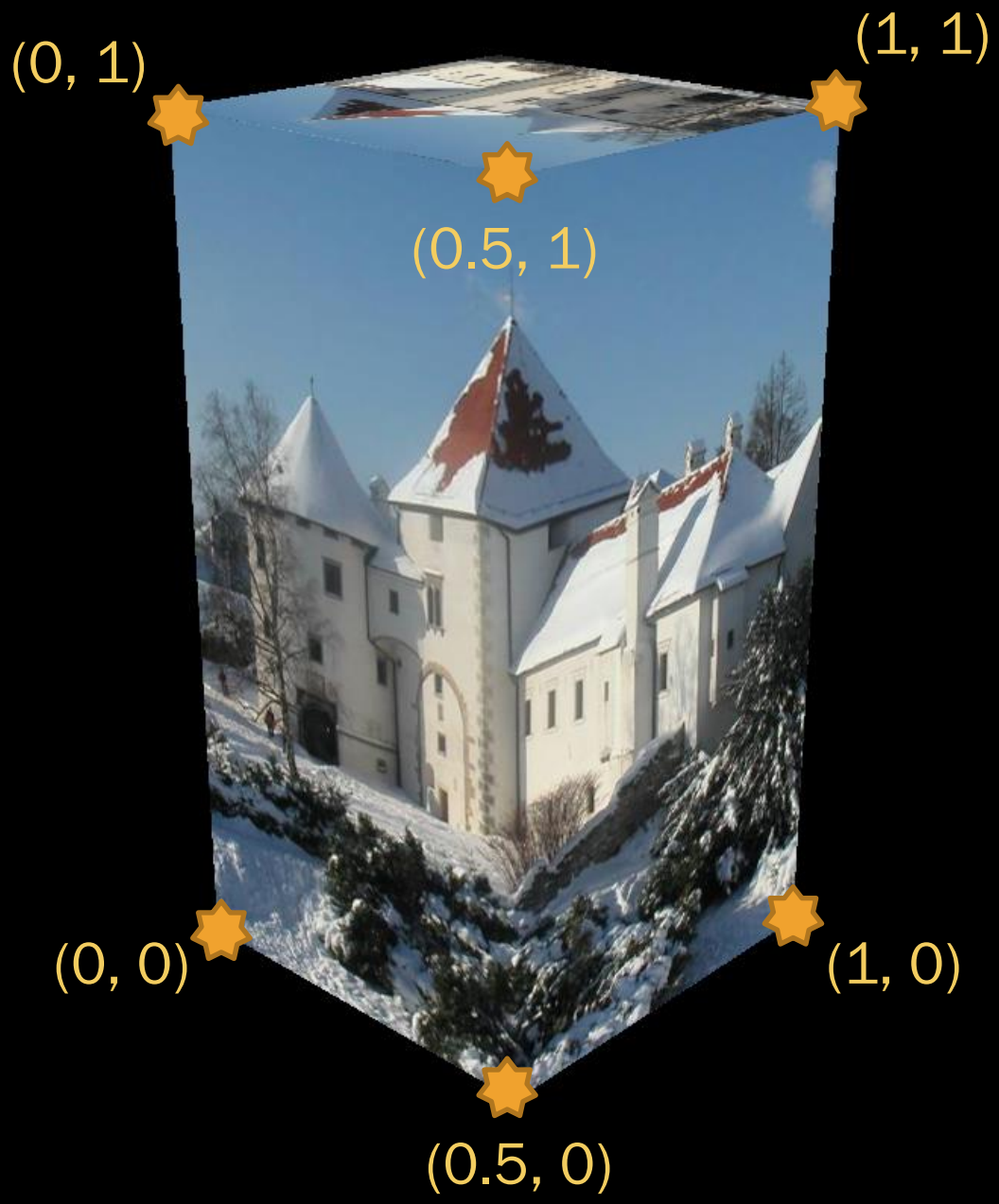

(0, 1)
(1, 1)
(0, 0)
(1, 0)
(ax, ay, az)
(ax, 0, az)
(0, ay, az)
(0, 0, az)

Koordinate vrhova i tekstura

- koordinate vrhova i tekstura mogu se pospremiti u posebne ili isprepletene spremnike
- primjer gornje plohe kvadra u isprepletenom spremniku: na svaki vrh kvadra lijepi se jedan ugao teksture

```
[ 0,  0, az,    0, 0], // gornja ploha
[ax,  0, az,    1, 0],
[ax, ay, az,    1, 1],
[ 0, ay, az,    0, 1]
```


(1, 1)
(1, 0)
(0, 1)
(0, 0)


(0, 1)
(1, 1)
(0.5, 1)
(0, 0)
(1, 0)
(0.5, 0)

Spremanje u spremnike

```
gl.bindBuffer(gl.ARRAY_BUFFER, gl.createBuffer());

gl.enableVertexAttribArray(GPUprog1.a_vrhXYZ);
gl.enableVertexAttribArray(GPUprog1.a_teksturaST);

gl.vertexAttribPointer(GPUprog1.a_vrhXYZ, 3,
gl.FLOAT, false, 20, 0);
gl.vertexAttribPointer(GPUprog1.a_teksturaST, 2,
gl.FLOAT, false, 20, 12);

gl.bufferData(gl.ARRAY_BUFFER, new
Float32Array(vrhovi), gl.STATIC_DRAW);
```

Postavljanje aktivne teksture

- ako se koristi više tekstura treba prije iscrtavanja postaviti koja je aktivna

```
gl.activeTexture(gl.TEXTURE0);
gl.uniform1i(program.u_slika, 0);
gl.bindTexture(gl.TEXTURE_2D, tekstura[0]);
gl.drawArrays(gl.TRIANGLE_STRIP, 0, 4);

gl.activeTexture(gl.TEXTURE1);
gl.uniform1i(program.u_slika, 1);
gl.bindTexture(gl.TEXTURE_2D, tekstura[1]);
gl.drawArrays(gl.TRIANGLE_STRIP, 4, 4);
```

Program za sjenčanje vrhova

- koordinate tekstura samo se prosljeđuju dalje u program za sjenčanje fragmenata

```
#version 300 es
in vec4 a_vrhXYZ;
in vec2 a_texturaST;
uniform mat4 u_mTrans;
out vec2 v_texturaST;

void main() {
  v_texturaST = a_texturaST;
  gl_Position = u_mTrans * a_vrhXYZ;
}
```

Program za sjenčanje fragmenata

- u programu za sjenčanje fragmenata interpolacija se odvija automatski, skoro kao da je fiksni protočni sustav

```
#version 300 es
precision highp float;
uniform sampler2D u_slika;
in vec2 v_texturaST;
out vec4 bojaPiksela;

void main() {
  bojaPiksela = texture(u_slika, v_texturaST);
}
```

Program za sjenčanje vrhova

- ipak, imamo mogućnost dodati boje ili osvjetljenje

```
#version 300 es
precision highp float;
uniform sampler2D u_slika;
uniform vec4 u_boja;
uniform float u_mix;
in vec2 v_texturaST;
out vec4 bojaPiksela;

void main() {
  bojaPiksela = vec4((1.0 - u_mix) * u_boja.rgb +
    u_mix * texture(u_slika, v_texturaST).rgb, 1);
}
```