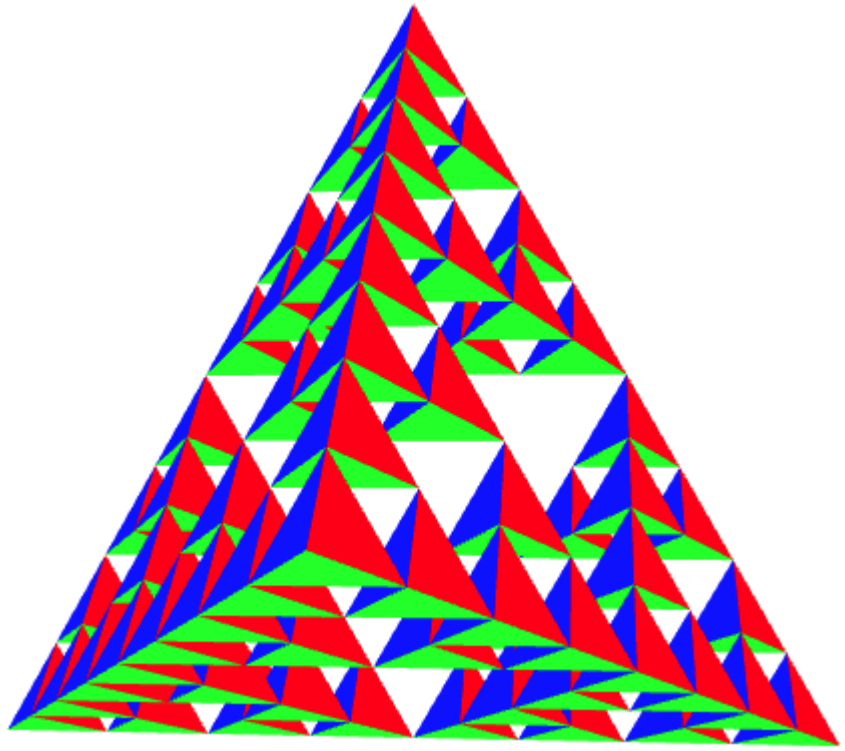
TÖL105M TÖLVUGRAFÍK

Fyrirlestur 4: Litarar

Hjálmtýr Hafsteinsson
Haust 2024



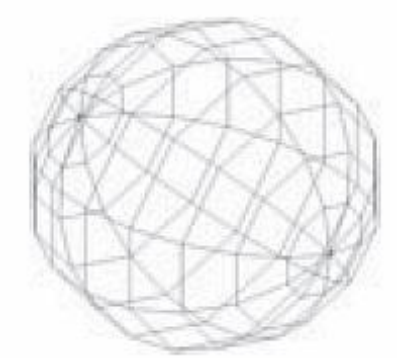
Í þessum fyrirlestri

- Marghyrningar í WebGL
- Endurkvæmt Sierpinski forrit
- Litir í tölvugrafík
  - Skilgreining á litum
- Litarar (*shaders*)
  - Skilgreining
  - Uppsetning
  - Notkun

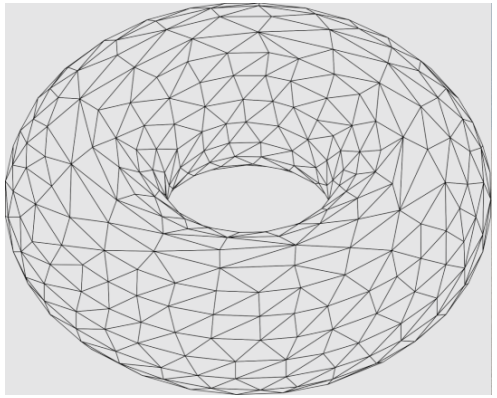
2.8 – 2.9

Marghyrningar (*polygons*)

- Tölvugrafík vinnur mikið með marghyrninga
  - Flóknir hlutir eru brotnir upp í marghyrningagrind
  - Þrívíðir hlutir verða margflötungar (*polyhedra*)

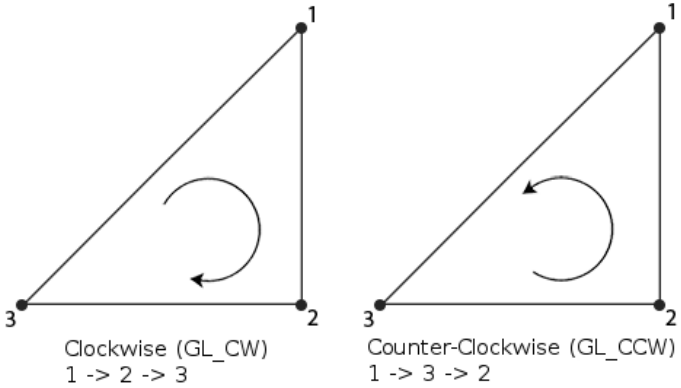


- Í dag eru nær eingöngu notaðir þríhyrningar
  - Flóknari marghyrningar brotnir upp í þríhyrninga (*triangulation*)
  - Þríhyrningavinnsla innbyggð í grafíkkort
  - Þríhyrningar hafa góða eiginleika í þrívídd
    - Þeir "bogna" ekki!

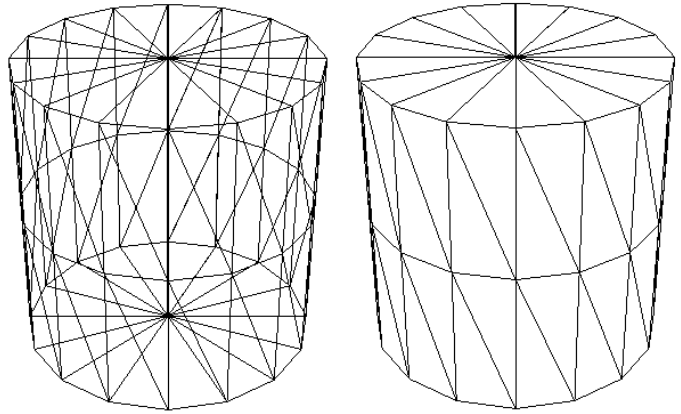


Hnútaröð marghyrninga

- Röð hnútanna ræður því hvernig marghyrningarnir snúa
  - Skiptir ekki máli í tvívídd
  - Sjálfgefið að rangsælisröð (*counter-clockwise*) snúi fram
  - Hægt að breyta því

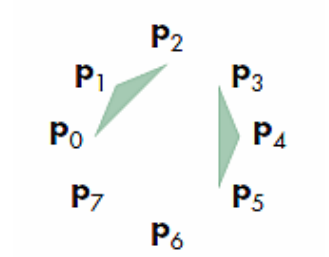


- Í þrívídd ákvarðar hnútaröðin hvaða hnútar eru bakhliðar á hlut og eiga þá ekki að teiknast

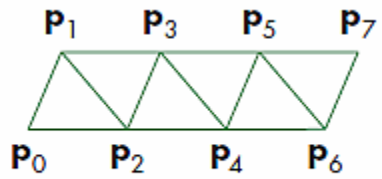


Teikning marghyrninga

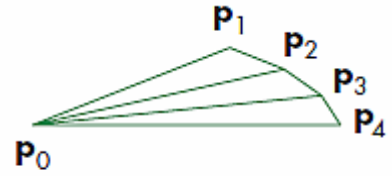
- Notum **`gl.drawArrays`**-fallið með tilteknum stikum:
  - **`gl.TRIANGLES`** Hverjir þrír punktar mynda þríhyrning
  - **`gl.TRIANGLE_STRIP`** Næsti punktur myndar þríhyrning með síðustu tveimur
  - **`gl.TRIANGLE_FAN`** Einn punktur er hluti af öllum þríhyrningunum



gl.TRIANGLES



gl.TRIANGLE_STRIP

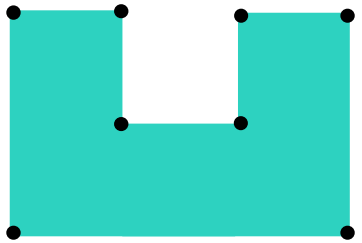


gl.TRIANGLE_FAN

Skoðum þetta betur síðar

Fyrirlestraæfingar

1. Hér til hliðar er marghyrningur með 8 hnútum. Hve marga þríhyrninga þarf til að teikna hann?



2. Litir eru rafsegulbylgjur af mismunandi tíðni. Hvor er með hærri tíðni, rauður eða blár?

3. Hvernig væri hægt að nota `uniform` breytu til að breyta (kvarða, *scale*) stærð Sierpinski þríhyrningsins?

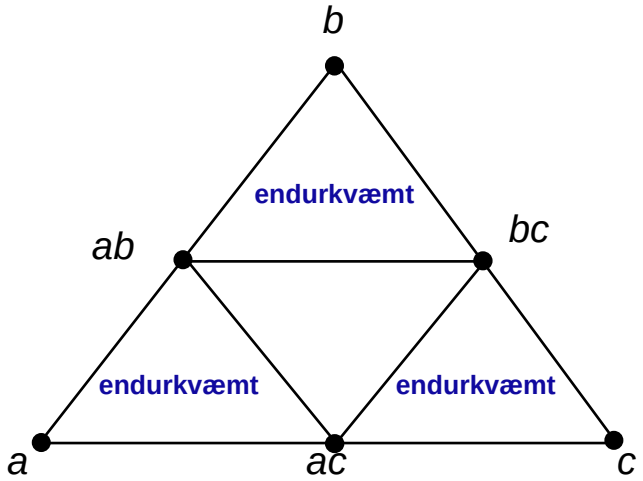
Endurkvæmur Sierpinski (gasket2)

- Endurkvæm uppskipting þríhyrninga:
  - Skipta hverri hlið í tvennt, fáum 4 þríhyrninga
  - Skiptum þremur þeirra endurkvæmt upp

```
function divideTriangle( a, b, c, count )
{
    if ( count === 0 ) {
        triangle( a, b, c );
    }
    else {
        var ab = mix( a, b, 0.5 );
        var ac = mix( a, c, 0.5 );
        var bc = mix( b, c, 0.5 );

        --count;
        divideTriangle( a, ab, ac, count );
        divideTriangle( c, ac, bc, count );
        divideTriangle( b, bc, ab, count );
    }
}
```

Fallið **`triangle`** setur **`a`**, **`b`**, **`c`** aftast í **`points`** fylkið

`mix(u,v,s)` er $(1-s)*u + s*v$



Endurkvæmur Sierpinski

- Skilgreinum hornpunkta eins og áður
- Köllum svo á endurkvæma fallið með fjölda uppskiptinga

```
var vert = [vec2( -1, -1 ), vec2( 0,  1 ), vec2( 1, -1 )];

divideTriangle(vert[0], vert[1], vert[2], NumSubdivide);
```

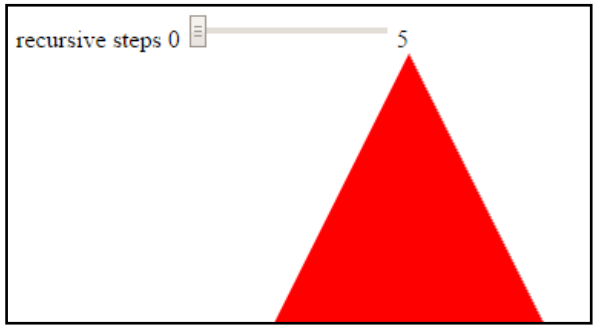
Gott að byrja með 5 eða 6 skiptingar

Fjöldi hnúta verður $3^{\texttt{NumSubdivide}+1}$
Fyrir 5 uppskiptingar verður þetta $3^6 = 729$

```
function render()
{
    gl.clear( gl.COLOR_BUFFER_BIT );
    gl.drawArrays( gl.TRIANGLES, 0, points.length );
}
```

Teiknum nú þríhyrninga, ekki punkta

Breytileg uppskipting



- Sýnisforritið **`gasket5`** leyfir notandanum að stýra uppskiptingunni með sleða (*slider*)
  - Sleðinn er skilgreindur í HTML-skránni

```
recursive steps  0 <input id="slider" type="range"
 min="0" max="5" step="1" value="0" /> 5
```

  - Fallið sem bregst við:

```
document.getElementById("slider").onchange = function(event) {
    numTimesToSubdivide = event.target.value;
    render();
};
```

  - Kóðinn sem býr til hnútana og sendir þá yfir er núna í **`render`**-fallinu

Frátekið minni á grafíkkorti

- Í forritinu **`gasket5`** er tekið frá minnissvæði á grafíkkorti í upphafi og gögnin eru afrituð þangað síðar
  - Leyfir þá breytilegt gagnamagn (innan efri marka)

```
gl.bufferData( gl.ARRAY_BUFFER, 8*Math.pow(3, 6), gl.STATIC_DRAW );
```

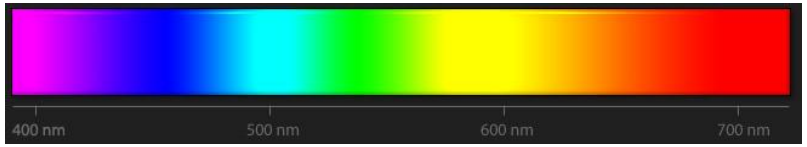
$8*3^6$ bæti frátekin
(tvö **`float`** gildi taka 8 bæti)

```
function render() {
    ...
    gl.bufferSubData(gl.ARRAY_BUFFER, 0, flatten(points));
    ...
}
```

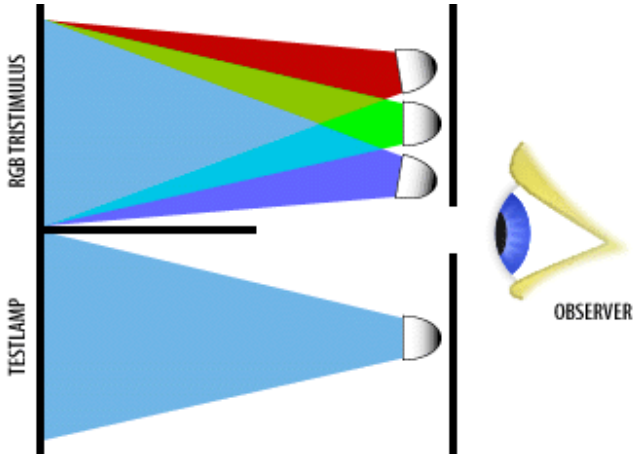
Setur gögnin inn í svæðið sem var frátekið

Litir

- Mjög flókið samspil líffræði (mannsaugað), sálfræði (mannsheilinn) og eðlisfræði (rafsegulbylgjur)
  - Skoðum þetta betur síðar
- Mannsaugað nemur rafsegulbylgjur með bylgjulengdir á bilinu 390nm (fjólublátt) til 700nm (rautt)

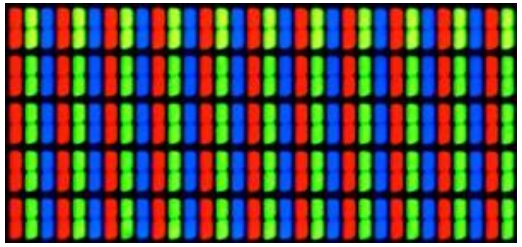


- Ljósgjafar gefa frá sé litróf, sem við skynjum sem tiltekinn lit
- Það er hægt að blekkja augað til að skynja tiltekinn lit með því að blanda saman þremur grunnlitum (*tristimulus theory*)



Skilgreining lita

- Hægt að skilgreina liti á marga vegu
- Við munum nota þrjá grunnliti: Rauður, Grænn, Blár
  - Þetta samsvarar því hvernig litir eru sýndir á skjám



- Tiltökum styrk hvers grunnlitar fyrir sig
  - Oftast á bilinu 0.0 til 1.0 (stundum 0 til 255)
  - Ef hver grunnlitur fær 1 bæti þá getum við búið til 2^{24} (16.777.216) liti
  - Mannsaugað getur í allra mesta lagi greint á milli ~10 milljón lita

Litir í tölvugrafík

- RGB líkanið er þægilegt til að tilgreina liti
  - En oft ekki auðvelt að vita hvað lit gildin tákna

```
var raudur = vec3(1.0, 0.0, 0.0);
var ljosblar = vec3(0.7, 0.8, 0.95);
```

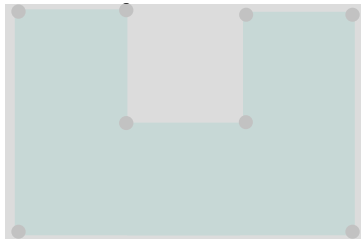
- Munum síðar bæta við fjórða gildinu, *A* eða alfa (*alpha*) sem táknar gegnsæi
  - Notum þá RGBA liti
  - A = 0.0 er alveg gegnsætt, A = 1.0 er alveg ógegnsætt

```
var halfgegnsaer_blar = vec4(0.0, 0.0, 1.0, 0.5);
var venjulegur_gulur = vec4(1.0, 1.0, 0.0, 1.0);
```

Auk þess er hagkvæmara að vinna með 4ra bæta blokkir heldur en 3ja bæta!

Fyrirlestraæfingar

1. Hér til hliðar er marghyrningur með 8 hnútum. Hve marga þríhyrninga þarf til að teikna hann?



2. Litir eru rafsegulbylgjur af mismunandi tíðni. Hvor er með hærri tíðni, rauður eða blár?

3. Hvernig væri hægt að nota `uniform` breytu til að breyta (kvarða, *scale*) stærð Sierpinski þríhyrningsins?

Skilgreining litara

- Litarar eru forrit skrifuð í GLSL (*OpenGL Shading Language*)
  - Hnútalitari (*vertex shader*) er keyrður fyrir hvern hnút líkansins sem er teiknað
    - Gerir ýmsar aðgerðir á hvern hnút, t.d. færa til, setja lit og þvervigur
  - Bútalitari (*fragment shader*) er keyrður fyrir hvern bút (þ.e. mögulegan skjápunkt)
    - Ákvarðar lit bútarins
- Aðeins eitt par af liturum er virkt í einu
  - Sama litaraforrit keyrt á öllum hnútum (eða bútum) og hver keyrsla er algerlega sjálfstæð

Þ.e. einn hnútalitari og einn bútalitari

Hvar eru litararnir?

- Í sjálfstæðum skrám
  - Oft gert í OpenGL forritum, en ekki gott í WebGL
- Sem strengjafastar í JS forritinu
  - Verður dálítið "sóðalegt"
- Skilgreindir í HTML skrá

Bókin notar þessa aðferð, hún er líklega skást

```
...
<script id="vertex-shader" type="x-shader/x-vertex">
attribute vec4 vPosition;

void main() {
    gl_PointSize = 1.0;
    gl_Position = vPosition;
}
</script>
...
```

Kóði fyrir hnútalitara

Uppsetning litara

- Javascript forritið þarf að lesa kóðann fyrir litarana, senda hann yfir á GPU og láta það þýða hann

- Dálítið flókið ferli, svo bókinni fylgir fall (**`initShaders`**) til að gera þetta fyrir okkur:

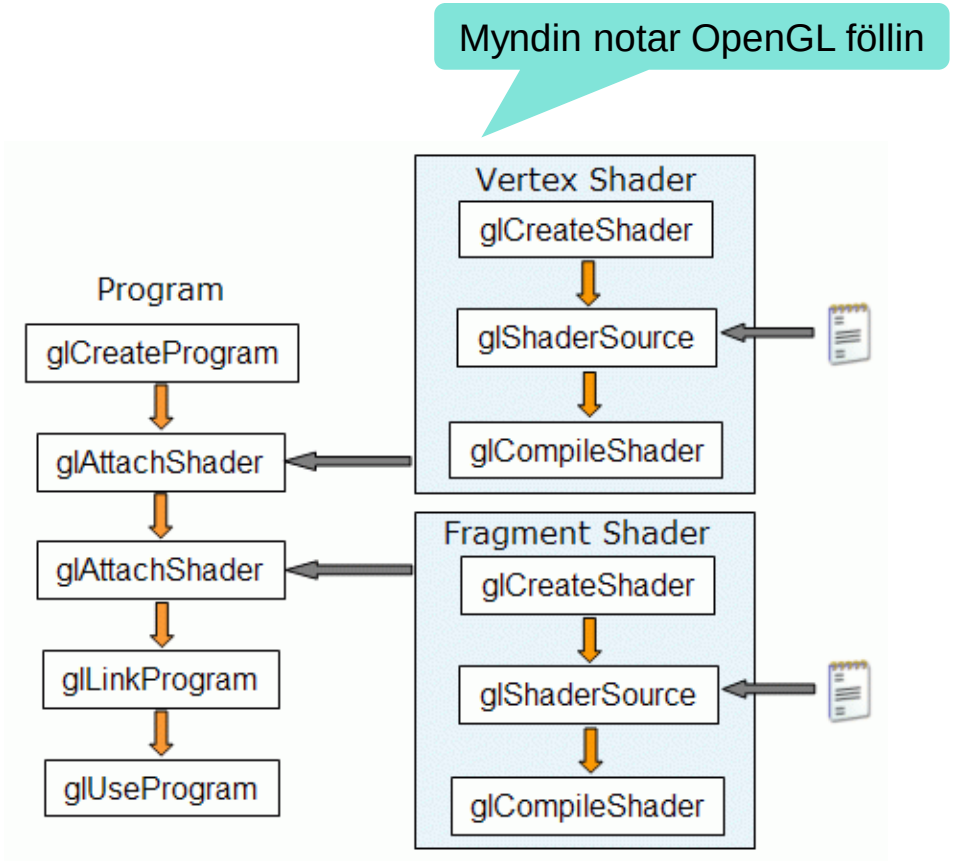
```
...
   var program = initShaders( gl, "vertex-shader", "fragment-shader" );
   gl.useProgram( program );
...
```

Nafn (*id*) á hnútalitara í HTML skrá

Nafn (*id*) á bútalitara í HTML skrá

Virkni `initShaders`

- Búa til forritshlut (**`program`**)
  - Með **`gl.createProgram`**
- Þarf að þýða báða litarana
  - Með **`gl.compileShader`**
  - Skila villum ef þær eru
- Tengja litarana við forritshlutinn
  - Með **`gl.attachShader`**
- Tengja allt saman
  - Með **`gl.linkProgram`**
- Virkja litarara
  - Með **`gl.useProgram`**



Þurfið ekki að kunna þetta, en gott að vita hvað er að gerast, t.d. ef það er villa í litarakóðanum

Hnútalitari

- Þessi litari gerir nánast ekkert (*pass-through shader*)
  - **`vPosition`** er inntaksbreyta sem inniheldur hnit hnútarins
  - Breytan **`gl_Position`** er innbyggð breyta sem sendir staðsetningu hnútarins áfram
  - **`gl_PointSize`** er innbyggð breyta fyrir stærð punktarins (ef teiknaður er punktur)

Algengasta notkun á hnútalitara er að breyta hnitum hnútarins áður en þau eru sett í **`gl_Position`**

```
attribute vec4 vPosition;

void main()
{
    gl_PointSize = 1.0;
    gl_Position = vPosition;
}
```

Bútalitari

- Þetta er lágmarks bútalitari
  - Allir bútalitarar verða að setja lit á núverandi bút
  - Gert með því að setja litagildi í innbyggðu breytuna **`gl_FragColor`**
  - Sum GPU ráða ekki við fulla nákvæmni á kommutölum í GLSL. Skipunin "**`precision mediump float`**" minnkar nákvæmnina (úr 16-bita í 10-bita)

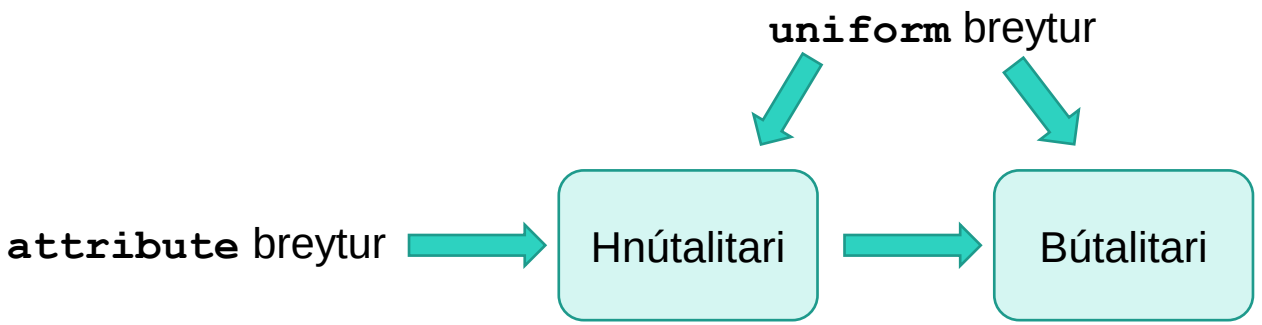
```
precision mediump float;

void main()
{
    gl_FragColor = vec4( 1.0, 0.0, 0.0, 1.0 );
}
```

Samskipti við litara

- Tvær leiðir fyrir JS forrit að senda gögn til litara:
  - Eiginleikabreytur (**`attribute`**)
    - Fyrir ýmsa eiginleika hnúta (hnit, lit, þvervigur, ...)
    - Aðeins sendar inn í hnútalitara (sem getur sent gildi áfram í bútalitara)
  - Almennar breytur (**`uniform`**)
    - Fyrir ýmsar stýriupplýsingar fyrir báða litara
    - Hafa sama gildi fyrir alla hnútana/bútana

Virka eins og víðværar (*global*) breytur

Litarar geta **ekki** sent gögn til baka til JS forrits



Dæmi um `uniform` breytu

- Setja lit Sierpinski þríhyrnings með **`uniform`** breytu:

í Javascript fallinu **`init()`**:

```
...
  colorLoc = gl.getUniformLocation( program, "fColor" );
  gl.uniform4fv( colorLoc, vec4(0.5, 0.0, 1.0, 1.0) );

  render();
};
```

Ná í "staðsetningu" breytunnar

Setja gildi í breytuna

Nafn fallsins fer eftir gerð viðfangsins.
Hér: 4ra staka **`float`** vigur (**`4fv`**)

Bútalitarinn:

```
precision mediump float;
uniform vec4 fColor;

void main()
{
    gl_FragColor = fColor;
}
```

Sýniforritið gasket1-lit

- Býr til 10.000 punkta fyrir þríhyrning Sierpinskis eins og venjulega
- Teiknar helming punktanna með rauðum lit og hinn helminginn með bláum lit

```
function render() {
    gl.clear( gl.COLOR_BUFFER_BIT );

    // Setjum litinn sem rauðann og teiknum helming punktanna
    gl.uniform4fv( colorLoc, vec4(1.0, 0.0, 0.0, 1.0) );
    gl.drawArrays( gl.POINTS, 0, points.length/2 );

    // Setjum litinn sem bláann og teiknum helming punktanna
    gl.uniform4fv( colorLoc, vec4(0.0, 0.0, 1.0, 1.0) );
    gl.drawArrays( gl.POINTS, points.length/2, points.length/2 );
}
```

Bútalitarinn hefur **`uniform`**-breytuna **`fColor`** sem fær þetta gildi

Hvar er byrjað í fylkinu?

Hversu marga punkta á að teikna?

Litarar og færiband

- Litarar hegða sér eins og starfsmaður á færibandi
  - Hnútagildin koma inn hvert af öðru á færibandinu (þ.e. í **`attribute`** breytum)
  - Starfsmaðurinn hefur líka aðgang að almennum stýriupplýsingum (þ.e. **`uniform`** breytum)
  - Hnútaupplýsingar sendar áfram og verða að bútaupplýsingum, sem bútalitari vinnur með (annað færiband!)

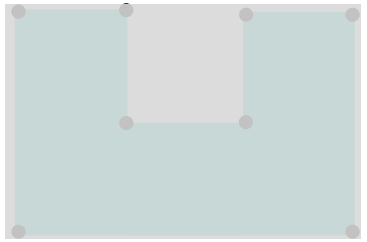
Chaplin - Nútíminn (*Modern Times*)

Forritun litara

- Einfaldir litarar:
  - Skrifa þá inn í HTML skránna með textaritli
- Vandamál:
  - Erfitt að aflúsa, því ekki hægt að skrifa neitt út
  - Væri gott að hafa þróunarumhverfi
- Til einföld þróunartól í vöfrum fyrir HTML og JS

Fyrirlestraæfingar

1. Hér til hliðar er marghyrningur með 8 hnútum. Hve marga þríhyrninga þarf til að teikna hann?



2. Litir eru rafsegulbylgjur af mismunandi tíðni. Hvor er með hærri tíðni, rauður eða blár?

3. Hvernig væri hægt að nota **`uniform`** breytu til að breyta (kvarða, *scale*) stærð Sierpinski þríhyrningsins?