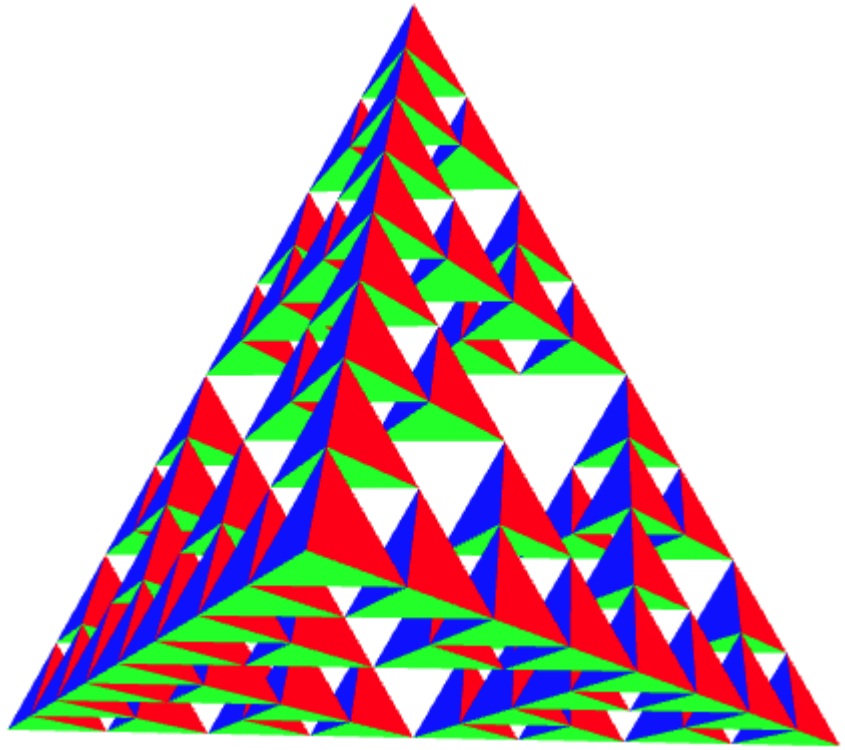
TÖL105M TÖLVUGRAFÍK

Fyrirlestur 6: Litarar og GLSL



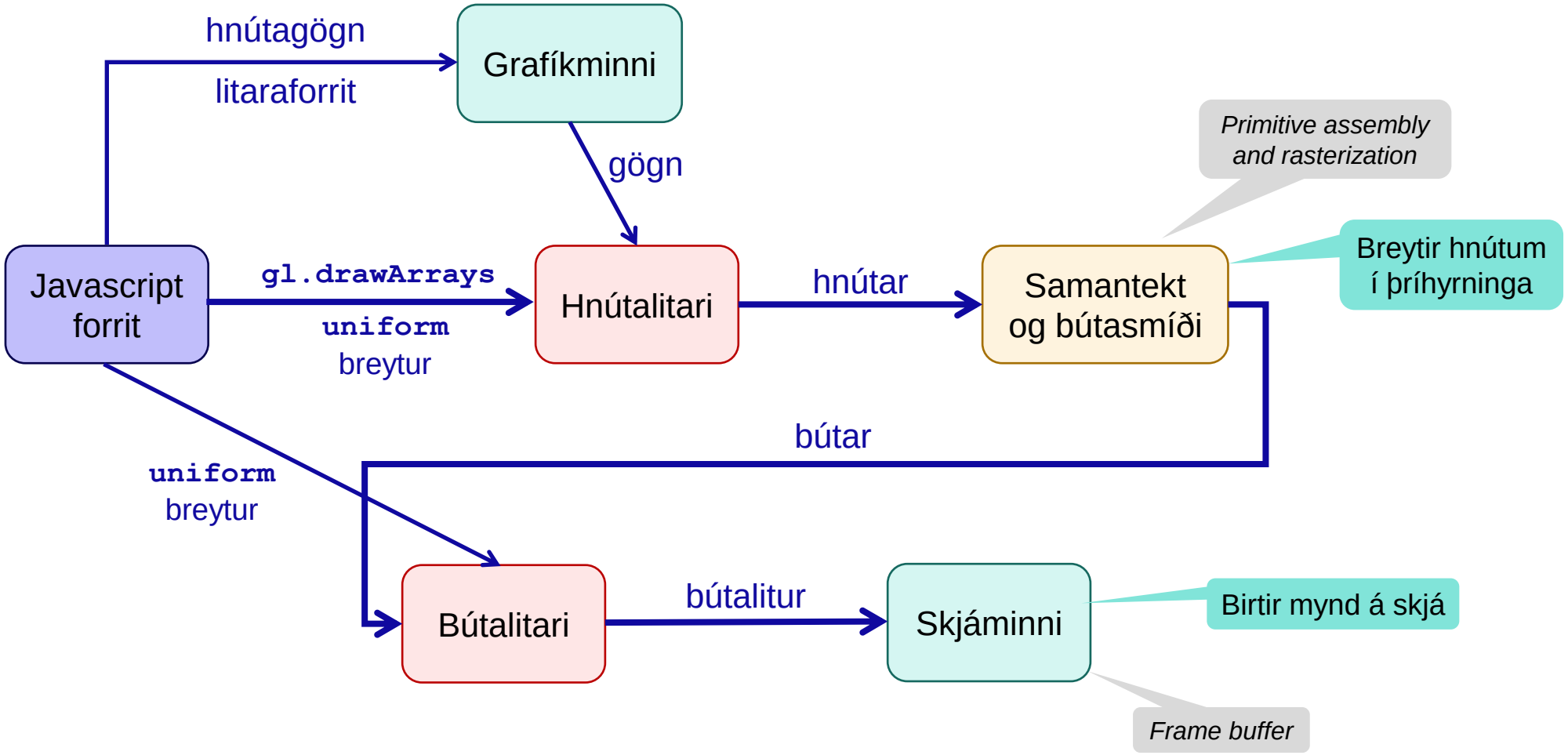
Hjálmtýr Hafsteinsson
Haust 2024

Í þessum fyrirlestri

- Litarar (*shaders*)
  - Hvers vegna litarar?
- GLSL forritunarmálið
  - Saga og samhengi
  - Gagnatög og aðgerðir
  - Minnissvæði á grafíkkorti

1.8, aukaefni

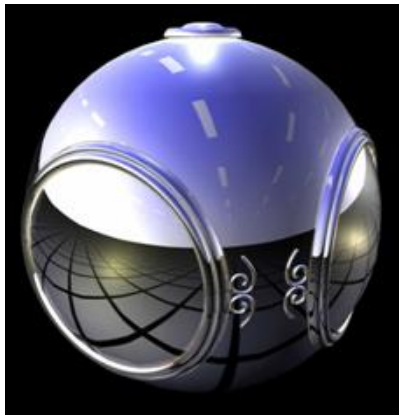
Grafíkpípa (*graphics pipeline*)



Kostir litara

- Auðvelt að gera tiltekna grafíkvinnslu samhliða (*parallel*)
- Látum grafíkörgjörva sjá um þessa vinnslu
  - Forritum þá sjálfstætt
- Notkun:
  - Staðsetja og kvarða hluti líkansins
  - Útfæra lýsingu í líkaninu
  - Mynsturvörpun (*texture mapping*)
  - ...

Væri hægt að gera þetta allt í notendaforritinu, en þá yrði það mun hægvirkara



Forritun litara

- Upphaflega skrifaðir í vélarmáli fyrir GPU
  - Ekki flytjanlegt milli grafíkkorta
- Nvidia hannaði Cg (*C for graphics*) fyrir þeirra kort
- OpenGL hannaði GLSL (*OpenGL Shading Language*) fyrir OpenGL forrit
- Microsoft hannaði HLSL (*High Level Shading Language*) fyrir DirectX forrit

Nvidia er hætt að þróa Cg

GLSL og HLSL eru mjög svipuð. HLSL er nánast eins og Cg

Öll málin eru byggð á C forritunarmálinu

WebGPU hefur sitt eigin litaramál: WGSL

GLSL

- Sérhannað til að skrifa litara
  - Hentar mjög vel til þess, ekki endilega gott almennt forritunarmál
  - Gagnatög og virkjar passa vel fyrir litara
- GLSL fyrst hluti af OpenGL 2.0 (2004)
- Eftir OpenGL 3.1 (2009) þurfa öll OpenGL/WebGL forrit að hafa litara
  - Í það minnsta "*pass-through*" litara

Svipuð þróun hefur verið í DirectX forritum
− reyndar oftast aðeins á undan OpenGL

Gagnatög í GLSL

- Skalartög: **`int`**, **`float`**, **`bool`**
- Vigurtög: **`vec2`**, **`vec3`**, **`vec4`**
  - Þetta eru vigrar af gerðinni **`float`**, líka til **`ivec*`** og **`bvec*`**
- Fylkjatög: **`mat2`**, **`mat3`**, **`mat4`**
  - Sömuleiðis til **`imat*`** og **`bmat*`**
- Hægt að nota smiði til að upphafsstilla breytur:
  - **`vec3 a = vec3(1.0, 0.0, 0.5);`**
  - **`vec4 b = vec4(1.0);`**

Setur öll fjögur gildin sem 1.0

Notkun vigra

- Vigrar notaðir í margvíslegum tilgangi í GLSL
  - Sem hnit (*x*, *y*, *z*, *w*), litir (*r*, *g*, *b*, *a*), mynsturhnit (*s*, *t*, *p*, *q*)
- Hægt að vísa í vigrana á marga vegu:

```
vec3 a;

a[1] = 0.2;
a.y = 0.2;
a.g = 0.2;
a.t = 0.2;
```

Allar fjórar skipanirnar eru jafngildar!

```
vec2 b;

b.x   // löglegt
b.z   // ólöglegt
```

```
vec4 color = ...;
float blue;

blue = color.b;
```

"Hræring" (*swizzling*)

- Mjög sniðug og öflug leið til að vinna með einstaka þætti í vigrum

```
vec4 a = vec4(1.0, 0.0, 0.2, 1.0);
vec2 b;

b = a.xy;
a.xyz = a.zyx;
b = b.xx
a.xx = b              // ólöglegt, tvö x vinstra megin
a.rgb = vec3(0.4, 0.8, 0.3);
b = a.ry;             // ólöglegt
```

Má ekki blanda saman tilvísunaraðferðum

Fyrirlestraæfingar

1. Hvert er lokagildið á **`a`** eftir þessar GLSL skipanir?

```
vec4 a = vec4(1.0, 2.0, 3.0, 4.0);
a.wzyx = a.xxyz;
```

2. Skrifið hnútalitarabút sem notar **`uniform`** breytuna **`time`** (ásamt **`sin()`**) til að stækka og minnka þríhyrning á reglubundinn hátt.

3. Hvernig væri hægt að teikna þríhyrningana tvo í **`twoTriangles`** með því að senda bara þrjá hnúta yfir í grafíkminnið?
   Vísbending: Bæta við **`uniform`** breytu

Notkun fylkja

- Fylki eru annaðhvort 2x2, 3x3 eða 4x4
  - Reyndar núna líka hægt að búa til *n*x*m*, með *n*, *m* = 2, 3, 4
- Ekki hægt að nota hræringu (*swizzling*) með fylkjum
- Notuð venjuleg fylkjatilvísun:

```
mat4 m;
m[0][0] = 1.0;
```

- Hægt að vísa í einn dálk í einu (en ekki eina línu):

```
mat4 m;
m[1] = vec4(0.1);
m[2].xy = vec2(0.1, 2.0);
```

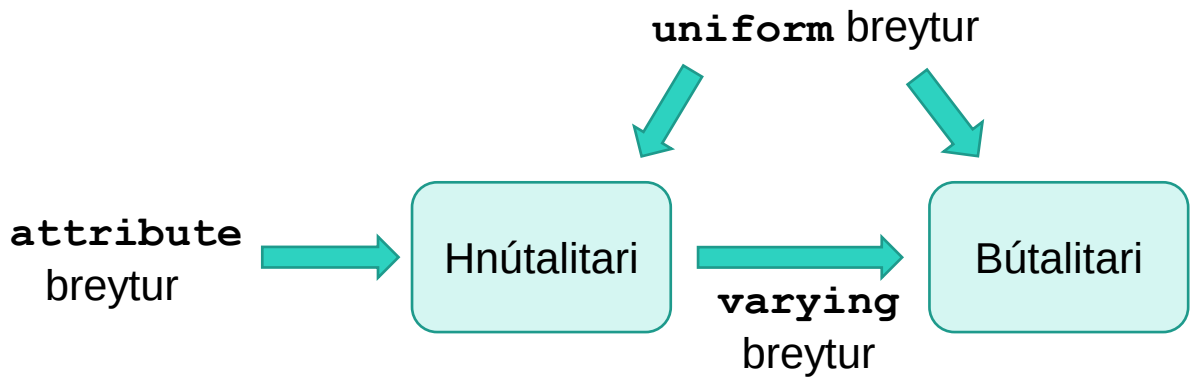
Setur dálk 1 (annan dálk) allan sem 0.1

`m[2]` er vigur, svo hér má nota hrærun

Sérstakar breytur

- Þrjár sérstakar gerðir breyta:
  - **`attribute`** breytur: fá nýtt gildi fyrir hvern hnút
  - **`uniform`** breytur: sama gildi fyrir alla teikningu (einskonar "*global*" breytur)
  - **`varying`** breytur: færa gildi frá hnútalitara til bútalitara

- Athugið:
  - Ekki hægt að flytja gildi frá liturum til notendaforrits



Dæmi um sérstakar breytur

- Hluti af forriti sem færir tening um líkan
  - Vörpunarfylki færir öll hnitin á sama hátt

```
attribute vec4 vPosition;
attribute vec4 vColor;
uniform mat4 rotation;
varying vec4 fColor;

void main()
{
    fColor = vColor;
    gl_Position = rotation * vPosition;
}
```

Hnit og litur fyrir hvern hnút

Vörpunarfylki sem gildir fyrir alla hnúta

Litur fluttur á milli. WebGL sér um að brúa (*interpolate*) gildin

Ný nöfn á breytugerðum

- Í nýrri útgáfum af GLSL eru ekki lengur notuð nöfnin **`attribute`** og **`varying`**
- Nota frekar **`in`** fyrir **`attribute`**-breytur í hnútalitara og **`varying`** breytur í bútalitara og **`out`** fyrir **`varying`** breytur í hnútalitara

```
in vec4 vPosition;
in vec4 vColor;
uniform mat4 rotation;
out vec4 fColor;

void main()
{
    fColor = vColor;
    gl_Position = rotation * vPosition;
}
```

```
in vec4 fColor;

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

Dæmigerð notkun á vörpunarfylki til að breyta hnitum hnúta

WebGL 1.0 hefur eldri gerð af GLSL málinu

Virkjar (*operators*) í GLSL

- Allir helstu virkjar
  - Svipað og í C++/Java:
    - Útreikningur: `+`, `-`, `*`, `/`, `%`, `<<`, `>>` `++`, `--`, ...
    - Samanburður: `==`, `!=`, `<`, `>`, `<=`, `>=`
    - Rökvirkjar: `&&`, `||`, `!`,
  - Virka á öll tög sem passa

```
mat3 m;
vec3 u, v;
float t;

u = v + t;
u = u + v;
v = m * u;
```

Vigur + skalar

Vigur + vigur (stak fyrir stak)

Fylki * vigur (fylkjamargföldun)

Innbyggð föll í GLSL

- Hornaföll:
  - **`sin`**, **`cos`**, **`tan`**, **`asin`**, **`acos`**, **`atan`**
- Önnur föll:
  - **`pow`**, **`exp`**, **`log`**, **`sqrt`**, **`abs`**, **`max`**, **`min`**, **`mix`** ...
- Vigurföll:
  - **`cross`**, **`dot`**, **`distance`**, **`length`**, ...

Sjá fleiri föll á *Quick Reference Card*

Stýriskipanir

- Ítrun: **`for(;;)`**, **`while( )`**, **`do{ } while( )`**
  - Frekar sjaldan notað, því aðeins unnið með einn hnút/bút í einu
- Stýriskipun: **`if( )`**, **`if( ) { } else { }`**
  - Nokkuð mikið notað til að ákveða hreyfingu hnútar eða lit á bút
- Föll
  - Viðföng send með "*value-return*"
  - Endurkvæmni ekki leyfð

Stærri sýnidæmi um litaraforrit:

- Shadertoy
- GLSL Sandbox

Sýniforrit: Hreyfa hnúta

- Tvær **`uniform`**-breytur:
  - **`time`**: fjöldi msek, uppfærist sífellt
  - **`rcolor`**: slembilitur þegar smellt á músarhnapp

Aðeins notuð í hnútalitara

Aðeins notuð í bútalitara

```
uniform float time;
attribute vec4 vPosition;
void main() {
    vec4 t = vPosition;
    t.y = 0.5*sin(0.003*time + 4.0*t.x) + 0.5*sin(0.003*time + 8.0*t.y);
    gl_Position = t;
}
```

Notar **`time`** til að breyta *y*-hniti á reglubundinn hátt

```
precision mediump float;
uniform vec4 rcolor;
void main() {
    gl_FragColor = rcolor;
}
```

Sjá **`waveTriangle`**

Fyrirlestraæfingar

1. Hvert er lokagildið á `a` eftir þessar GLSL skipanir?

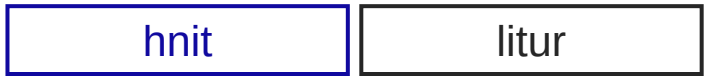
```
vec4 a = vec4(1.0, 2.0, 3.0, 4.0);
a.wzyx = a.xxyz;
```

2. Skrifið hnútalitarabút sem notar **`uniform`** breytuna **`time`** (ásamt **`sin()`**) til að stækka og minnka þríhyrning á reglubundinn hátt.

3. Hvernig væri hægt að teikna þríhyrningana tvo í **`twoTriangles`** með því að senda bara þrjá hnúta yfir í grafíkminnið?
   Vísbending: Bæta við **`uniform`** breytu

Margir eiginleikar (*attributes*)

- Oft fleiri en einn eiginleiki á hverjum hnút
  - Til dæmis bæði hnit og litur
- Tvær lausnir:
  - Tvö sjálfstæð minnissvæði (*buffers*)



  - Flétta eiginleikana saman (*interleaved*)

Tvö minnissvæði

- Tvö sjálfstæð fylki í JS-forriti og tvö sjálfstæð minnissvæði í grafíkminni

Kóði fyrir hnit

```
var vBuffer = gl.createBuffer();
gl.bindBuffer( gl.ARRAY_BUFFER, vBuffer);
gl.bufferData( gl.ARRAY_BUFFER, flatten(vertices), gl.STATIC_DRAW );

var vPosition = gl.getAttribLocation(program, "vPosition");
gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(vPosition);
```

Minnissvæði fyrir hnit

Tengja við **vPosition** breytu

Kóði fyrir liti

```
var cBuffer = gl.createBuffer();
gl.bindBuffer( gl.ARRAY_BUFFER, cBuffer );
gl.bufferData( gl.ARRAY_BUFFER, flatten(colors), gl.STATIC_DRAW );

var vColor = gl.getAttribLocation( program, "vColor" );
gl.vertexAttribPointer( vColor, 4, gl.FLOAT, false, 0, 0 );
gl.enableVertexAttribArray( vColor );
```

Minnissvæði fyrir liti

Tengja við **vColor** breytu

Sjá **colorTriangle2**

Tvö minnissvæði

- Þegar litarabreyta er tengd við minnissvæði með **`gl.vertexAttribPointer`** þá er tengt við virkt (*active*) minnissvæði
  - Virkjum minnissvæði með **`gl.bindBuffer`**
  - Köllum því á **`gl.bindBuffer`** rétt á undan **`gl.vertexAttribPointer`**
- Notum svipaða aðferð þegar við teiknum marga hluti
  - Hver þeirra í sjálfstæðu minnissvæði

Sjá **`twoTriangles`**

Fyrir hvern sjálfstæðan hlut:

- Virkja minnissvæði hans [**`bl.bindBuffer`**]
- Tengja það við **`attribute`** breytu í hnútalitara [**`gl.vertexAttribPointer`**]
- Setja möguleg gildi í **`uniform`** breytu(r) [**`gl.uniform*`**]
- Teikna hlutinn með **`gl.drawArrays`**

Flétta saman

- Eitt fylki í JS-forriti og eitt minnissvæði í grafíkminni
  - Gögn fléttuð saman í báðum

```
var vBuffer = gl.createBuffer();
gl.bindBuffer( gl.ARRAY_BUFFER, vBuffer);
gl.bufferData( gl.ARRAY_BUFFER, flatten(vertices), gl.STATIC_DRAW );

var vPosition = gl.getAttribLocation(program, "vPosition");
gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 4*8, 0);
gl.enableVertexAttribArray(vPosition);

var vColor = gl.getAttribLocation( program, "vColor" );
gl.vertexAttribPointer(vColor, 4, gl.FLOAT, false, 4*8, 4*4 );
gl.enableVertexAttribArray( vColor );
```

Aðeins eitt minnissvæði fyrir bæði hnit og liti

Tengja við **`vPosition`** breytu

Tengja við **`vColor`** breytu

Sjá **`colorTriangle1`**

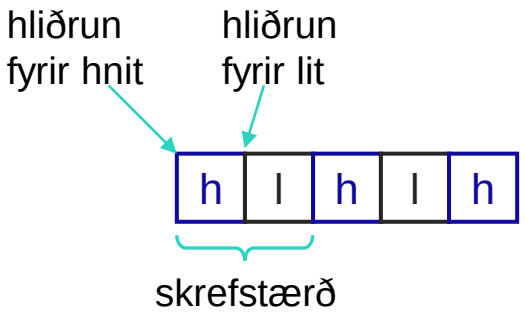
Flétta saman

- Skipunin **`gl.vertexAttribPointer`** tengir **`attribute`**-breytur í litara við minnissvæði

```
gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 4*8, 0);
...
gl.vertexAttribPointer(vColor, 4, gl.FLOAT, false, 4*8, 4*4 );
```

skrefstærð (*stride*)

hliðrun (*offset*)



Hnitin eru nú **`vec4`**, sem er 4*4 (=16) bæti
Skrefstærðin er því 2*4*4 (=32) bæti

Formúla fyrir gildi *i* er:
*i*stride + offset*

Fyrirlestraæfingar

1. Hvert er lokagildið á `a` eftir þessar GLSL skipanir?

```
vec4 a = vec4(1.0, 2.0, 3.0, 4.0);
a.wzyx = a.xxyz;
```

2. Skrifið hnútalitarabút sem notar `uniform` breytuna `time` (ásamt `sin()`) til að stækka og minnka þríhyrning á reglubundinn hátt.

3. Hvernig væri hægt að teikna þríhyrningana tvo í `twoTriangles` með því að senda bara þrjá hnúta yfir í grafíkminnið?
   Vísbending: Bæta við `uniform` breytu