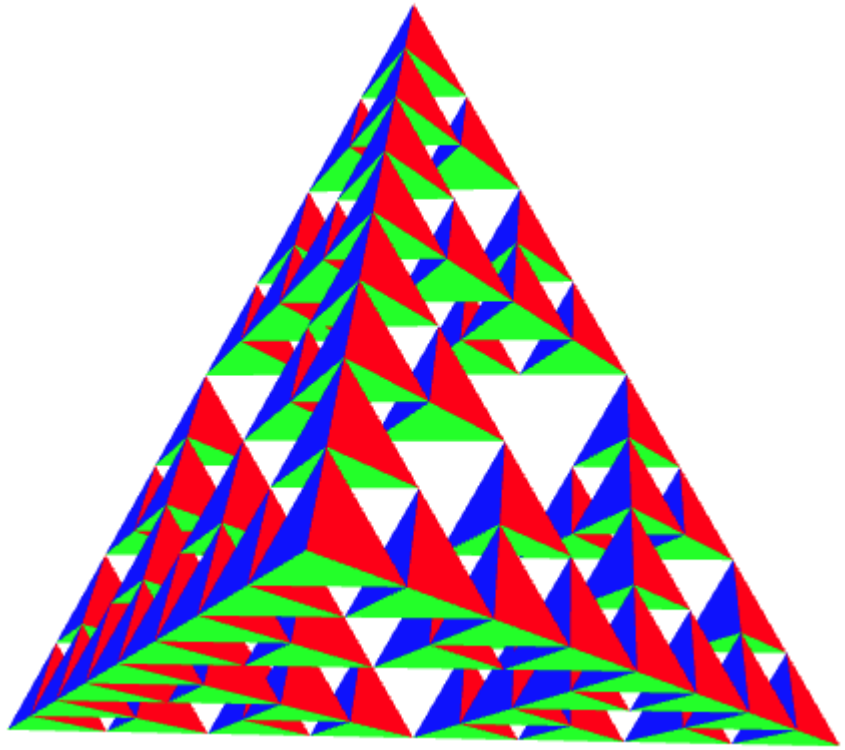
TÖL105M TÖLVUGRAFÍK

Fyrirlestur 2: WebGL forritun



Hjálmtýr Hafsteinsson
Haust 2024

Í þessum fyrirlestri

- Dæmigerð WebGL forrit
  - Tvær skrár: HTML og Javascript
  - Einfalt WebGL forrit
- Forritunarumhverfi
  - Textaritill
  - Vafrari
  - Aflúsun (*debugging*)

2.2 – 2.7

Saga OpenGL

- Kemur frá Silicon Graphics (SGI)
  - Hét þá GL (*Graphics Library*)
- Gert að opnum staðli árið 1992 og kallað OpenGL
  - Engin gluggaföll - til að losna við vandamál vegna ólíkra stýrikerfa
- Stöðugt og samhæft aftur á bak fram að útgáfu 3.0 (2008)
  - Vandamál: Nýtti sér ekki framfarir í grafíkkortum
- Í útgáfu 3.0 og seinni útgáfum er áherslan á GPU
  - Forritanlegir litarar (*shaders*) í GLSL forritunarmálinu

Nýjasta útgáfa er OpenGL 4.6

Næst kemur Vulkan(?)

(Ekki ráðlagt að byrja að læra Vulkan, heldur frekar WebGL!)

Önnur tengd forritasöfn

- OpenGL ES (*Embedded Systems*)
  - Einfölduð útgáfa af OpenGL fyrir smærri tölvur
    - Stutt af Android, iOS, ...
  - Samsvörun:
    - OpenGL ES 1.0 er svipað OpenGL 2.1
    - OpenGL ES 2.0 er svipað OpenGL 3.1
    - OpenGL ES 3.0 er svipað OpenGL 4.3
  - Engir sjálfgefnir litarar í útgáfu 2.0 og síðar
    - Þarf að skrifa litara fyrir hvert forrit

Með forritan-legum liturum

WebGL

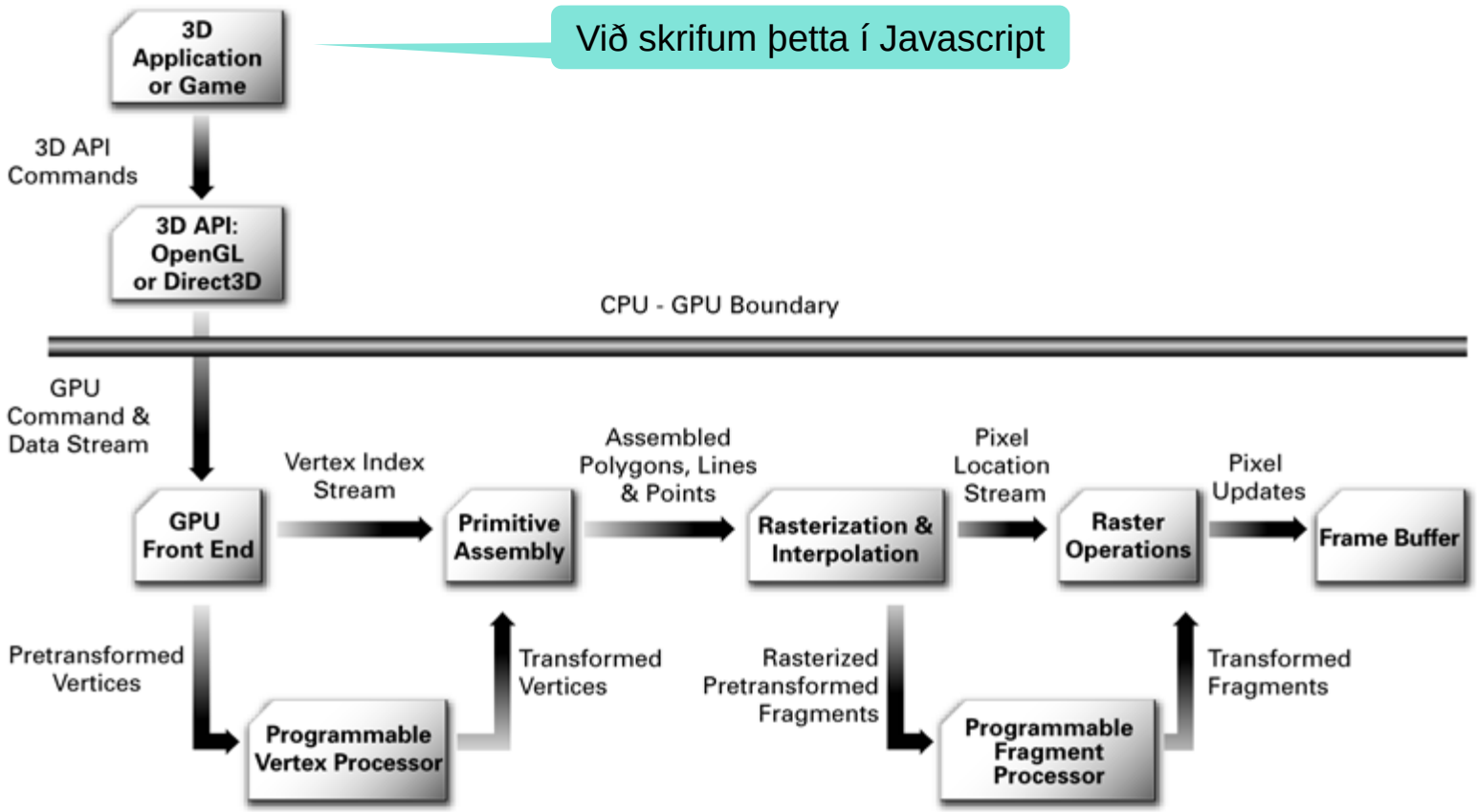
- WebGL er Javascript viðmót á OpenGL ES 2.0
  - Stutt af nær öllum vöfrum í dag:
    - Chrome, Safari, Firefox, Opera, Internet Explorer, ...
  - WebGL forrit samanstanda af:
    - Javascript forriti sem vafrinn keyrir
    - Litaraforritum í GLSL, sem keyra á GPU tölvunnar
  - WebGL 2.0 er frekar nýlegt
    - Byggt á OpenGL ES 3.0
    - Mörg minni tæki ráða ekki ennþá við það

Reynum að hafa þyngsta grafíkkóðann hér. Javascript er hægvirkt!

Ekki mikill gróði fyrir okkur að nota WebGL 2.0

Ræður vafrinn við WebGL 2.0?
webglreport.com

Grafíkpípan (*Graphics pipeline*)



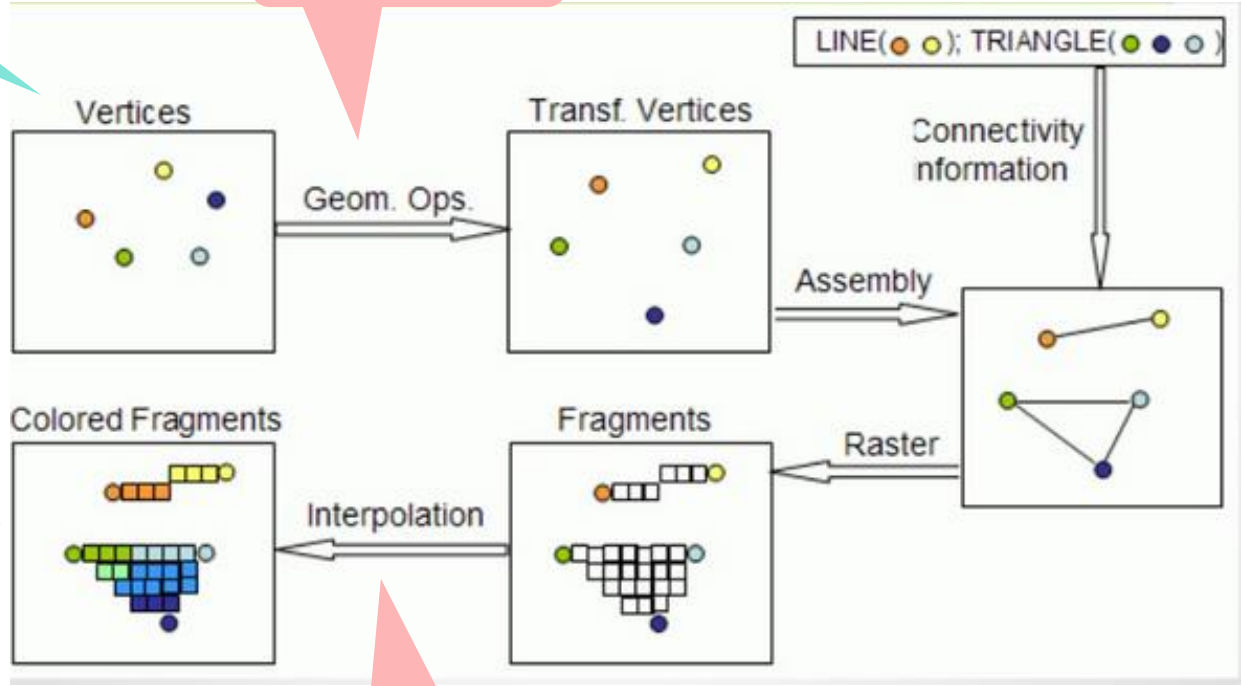
frá: Cg tutorial

Hvað gerist í grafíkpípunni?

Byrjum með hnúta (*vertices*)

Hnútalitari keyrir fyrir hvern hnút

Á að tengja hnútana sem línur eða þríhyrninga?



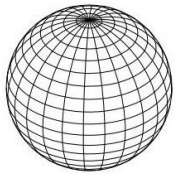
Það sem birtist á skjánum

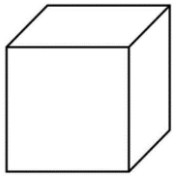
Bútalitari keyrir fyrir hvern bút

Einstök skref

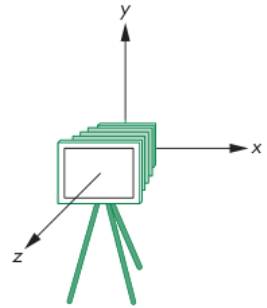
- Líkön (*models*)
  - Hvernig lýsum við hlutum?
    - Stærðfræðiformúla
    - Með hornpunktum og hliðum
    - Samsettir úr einfaldari hlutum (kössum, kúlum, ...)

$$x^2 + y^2 + z^2 = r^2$$





- Áhorfandi (*viewer*)
  - Þarf staðsetningu, stefnu, halla, eiginleika (gerð linsu), ...



- Ljós (*light*)
  - Staðsetja ljósgjafa, gefa honum lit
  - Fyrir allar hluti í líkaninu skilgreina hvernig þeir endurkasta ljósinu

Fyrirlestraæfingar

1. Hvaða útgáfa af OpenGL samsvarar best WebGL 2.0?
2. Hverju þyrfti að breyta í `triangle`-forritinu til að teikna bláan þríhyrning sem er á hvolfi (þ.e. oddurinn snýr niður)?
3. Hvað þarf marga punkta til að teikna ferning með því að nota:
   a. `gl.TRIANGLES`
   b. `gl.TRIANGLE_FAN`

WebGL forritun

- WebGL kann aðeins að teikna punkta, línur og þríhyrninga í ýmsum litum
  - Við verðum að skilgreina allt annað í kóða!

- Dæmigert WebGL forrit:
  i. Upphafsstilling (skilgr. teiknisvæði, hlaða inn liturum, ...)
  ii. Skilgreina stöðu (*state*)
  iii. Flytja gögn yfir á grafíkkort
  iv. Láta grafíkkort teikna (**`drawArrays`** fallið)

Nokkrar tegundir af gögnum:
- Fylki fyrir hnúta
- Stök gildi
- Mynstur (*texture*)

Dæmigert WebGL forrit

- Samanstendur af tveimur skrám:
  - HTML skrá
    - Lýsir síðunni
    - Skilgreinir HTML virkni
    - Hleður Javascript skrám
    - Inniheldur litarakóða (*shaders*)
  - Javascript skrá
    - Hleður liturum á grafíkkort
    - Skilgreinir og stýrir grafíkinni

Oftast frekar einföld skrá hjá okkur

Notum einfalda litara í upphafi

Þetta er aðalkóðinn sem við skrifum!

Samsetning WebGL forrita

- Litarar skrifaðir í GLSL

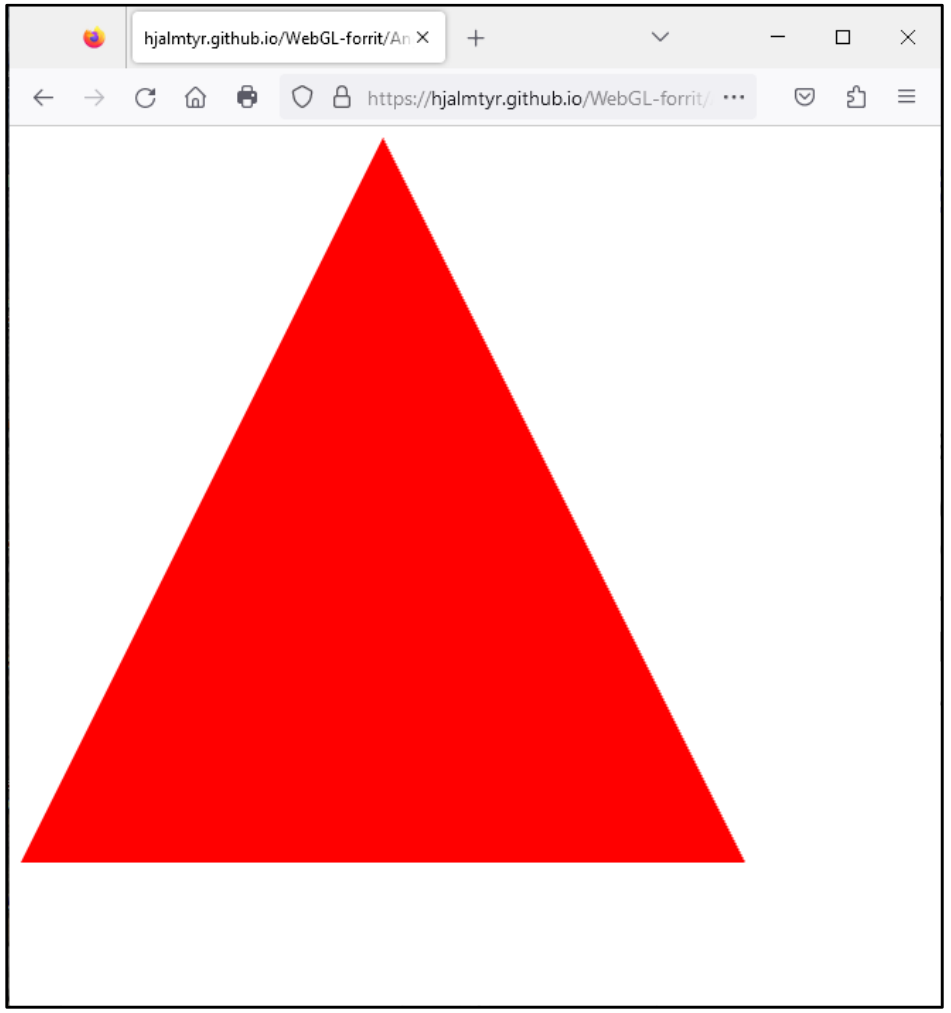
  Forritunarmál fyrir litara, líkist C (sjáum betur síðar)

  - Tveir litarar:
    - Hnútalitari (*vertex shader*) vinnur með einstaka hnúta (þ.e. punkta)
    - Bútalitari (*fragment shader*) vinnum með búta (þ.e. mögulega skjápunkta)
  - Keyra á grafíkörgjörva (***GPU***)

- Javascript föll
  - Keyra í vafra á örgjörva tölvunnar (***CPU***)

triangle.html

- Einfaldur rauður þríhyrningur
  - Þrír hnútar:
    (−1, −1), (0, 1), (1, −1)
  - Hnitakerfið er frá −1.0 til 1.0 í *x*- og *y*-ás



HTML skrá, fyrri hluti

Hnútalitarinn sem textastrengur

Bútalitarinn sem textastrengur

```
<!DOCTYPE html>
<html>
<head>
<script id="vertex-shader" type="x-shader/x-vertex">
attribute vec4 vPosition;
void main()
{
    gl_Position = vPosition;
}
</script>

<script id="fragment-shader" type="x-shader/x-fragment">
precision mediump float;
void main()
{
    gl_FragColor = vec4( 1.0, 0.0, 0.0, 1.0 );
}
</script>

...
```

HTML skrá, seinni hluti

Hleður þremur Javascript skrám

Takið eftir slóðinni á **Common** möppuna

```
...

<script type="text/javascript" src="../Common/webgl-utils.js"></script>
<script type="text/javascript" src="../Common/initShaders.js"></script>
<script type="text/javascript" src="triangle.js"></script>
</head>

<body>
<canvas id="gl-canvas" width="512" height="512">
Oops ... your browser doesn't support the HTML5 canvas element
</canvas>
</body>
</html>
```

Býr til 512x512 HTML5 striga (*canvas*)

Javascript skrár

- Skrárnar sem HTML skráin hleður inn:
  - **`webgl-utils.js`**
    - Skrá frá Google með JS föllum fyrir upphafsstillingu teiknisvæðisins á striganum
  - **`initShaders.js`**
    - Skrá frá Angel (bókarhöfundi) með JS falli til að lesa, þýða og virkja litarana tvo
  - **`triangle.js`**
    - Javascript kóðinn okkar til að teikna þríhyrninginn

`triangle.js`, fyrri hluti

```
var gl;
var points;

window.onload = function init()
{
    var canvas = document.getElementById( "gl-canvas" );
    gl = WebGLUtils.setupWebGL( canvas );
    if ( !gl ) { alert( "WebGL isn't available" ); }

    var vertices = new Float32Array([-1, -1, 0, 1, 1, -1]);

    //  Configure WebGL
    gl.viewport( 0, 0, canvas.width, canvas.height );
    gl.clearColor( 1.0, 1.0, 1.0, 1.0 );

    //  Load shaders and initialize attribute buffers
    var program = initShaders( gl, "vertex-shader", "fragment-shader" );
    gl.useProgram( program );
...
```

Skilgreina striga

Upphafsstilla teiknisvæði á striga

Skilgreina hnútana (-1, -1), (0, 1), (1, -1)

Eiginleikar teiknisvæðis

`initShaders` er fall frá Angel úr skránni **`initShaders.js`**

Ná í litarana tvo, þýða þá, senda þá yfir til grafíkkorts og virkja þá

`triangle.js`, seinni hluti

```
...

    // Load the data into the GPU
    var bufferId = gl.createBuffer();
    gl.bindBuffer( gl.ARRAY_BUFFER, bufferId );
    gl.bufferData( gl.ARRAY_BUFFER,vertices, gl.STATIC_DRAW );

    // Associate shader variables with our data buffer
    var vPosition = gl.getAttribLocation( program, "vPosition" );
    gl.vertexAttribPointer( vPosition, 2, gl.FLOAT, false, 0, 0 );
    gl.enableVertexAttribArray( vPosition );

    render();
};

function render() {
    gl.clear( gl.COLOR_BUFFER_BIT );
    gl.drawArrays( gl.TRIANGLES, 0, 3 );
}
```

Skilgreina minnissvæði á GPU og hlaða inn hnútagildum (**`vertices`**)

Tengja minnissvæðið á GPU við breytu í hnútalitara

Kalla svo á fallið til að teikna

Hreinsa teiknisvæðið og teikna hnútanna þrjá sem þríhyrning

Hvað gerist ef við breytum þessu í **`gl.POINTS`**?

Fyrirlestraæfingar

1. Hvaða útgáfa af OpenGL samsvarar best WebGL 2.0?
2. Hverju þyrfti að breyta í **`triangle`**-forritinu til að teikna bláan þríhyrning sem er á hvolfi (þ.e. oddurinn snýr niður)?
3. Hvað þarf marga punkta til að teikna ferning með því að nota:
   a. `gl.TRIANGLES`
   b. `gl.TRIANGLE_FAN`

`drawArrays` fallið

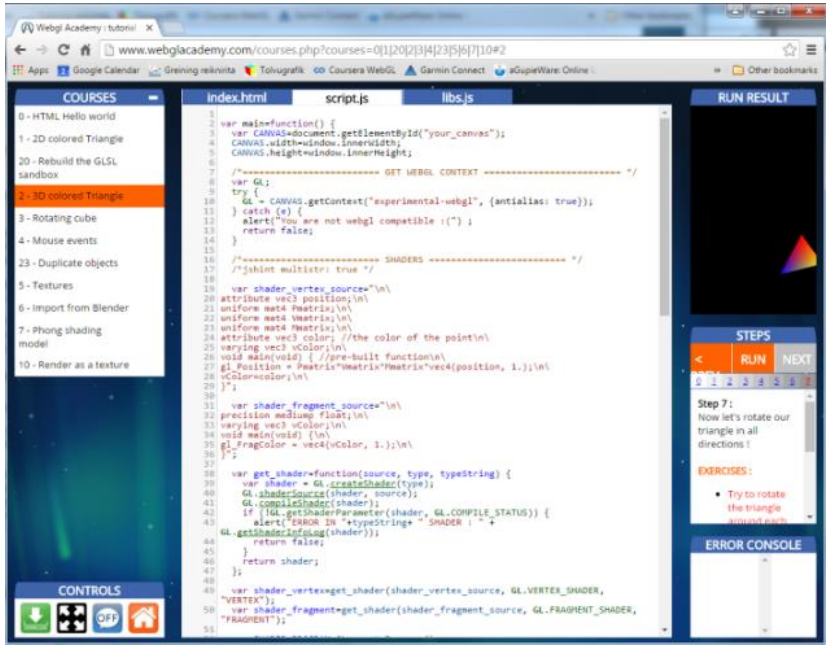
- Teiknar hnúta samkvæmt fyrsta stika (*parameter*)
- Möguleikar:
  - **`gl.POINTS`** punktar
  - **`gl.LINES`** línur
  - **`gl.LINE_STRIP`** samfelldir línubútar
  - **`gl.LINE_LOOP`** samfelldir línubútar í hring
  - **`gl.TRIANGLES`** þríhyrningar
  - **`gl.TRIANGLE_STRIP`** þríhyrningalengja
  - **`gl.TRIANGLE_FAN`** þríhyrningablævængur

Skoðum þessa möguleika betur síðar

Sýnisforritið pointtest.html á forritasíðu

Forritunarumhverfi

- Okkar er textaritill (Notepad++, VS Code, ...), jafnvel vafrari
  - Skrifum aðallega Javascript kóða (plús smá HTML)
  - Frekar einfaldur kóði og óþarfi að nota flókið þróunarumhverfi
- Það er til kennslusíða um WebGL, sem hefur einfalt þróunarumhverfi í vafra: WebGL Academy
  - Aðrar venjur en Angel:
    - Litarar sem strengir í JS forriti
    - Annað forritasafn fyrir vigra/fylki
  - En áhugavert að skoða kóðann



Keyrsla WebGL forrita

- WebGL forrit eru hönnuð til að keyra af vefsíðum
- Áður notuðum við heimasíðusvæði UTS
  - Það hefur nú verið tekið úr notkun! ☹
- Ýmsir möguleikar fyrir ykkur:
  - Github Pages er líklega einfaldast
    - Leiðbeiningar: frá Github, á Youtube, ...
- Forrit námskeiðsins verða á Github síðu
  - hjalmtyr.github.io/WebGL-forrit/

Staðværir vefþjónar

- Þið getið keyrt flest WebGL forrit beint af tölvunni ykkar
  - Tvísmella á HTML-skránna - þá opnast hún í vefsíðu
- Fyrir sum WebGL forrit virkar þetta ekki
  - Það er þegar forritin þurfa að lesa staðværar skrár, t.d. mynstur, líkön, ...
- Lausnir:
  - Flytja forritin á Github Pages síðuna ykkar og keyra þaðan
  - Setja upp staðværan vefþjón á tölvunni ykkar:

Fáið öryggisvillu: "*Cross-Origin Request Blocked*"

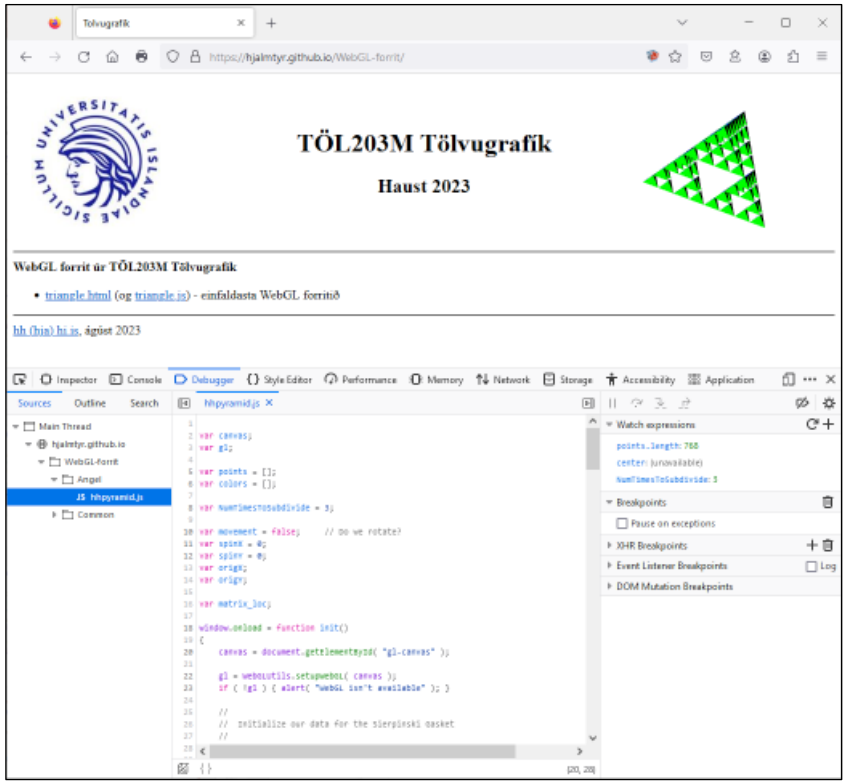
Þurfið þetta fyrir skil á heimadæmum/verkefnum

Staðværir vefþjónar:

• Live Server innan í VSCode	Einföld lausn ef þið notið VSCode í forritun
• Servez forritið	Einfaldur vefþjónn með grafísku viðmóti
• Node.js og npm	Almennasta lausnin (hafið kannski séð í Vefforritun!)

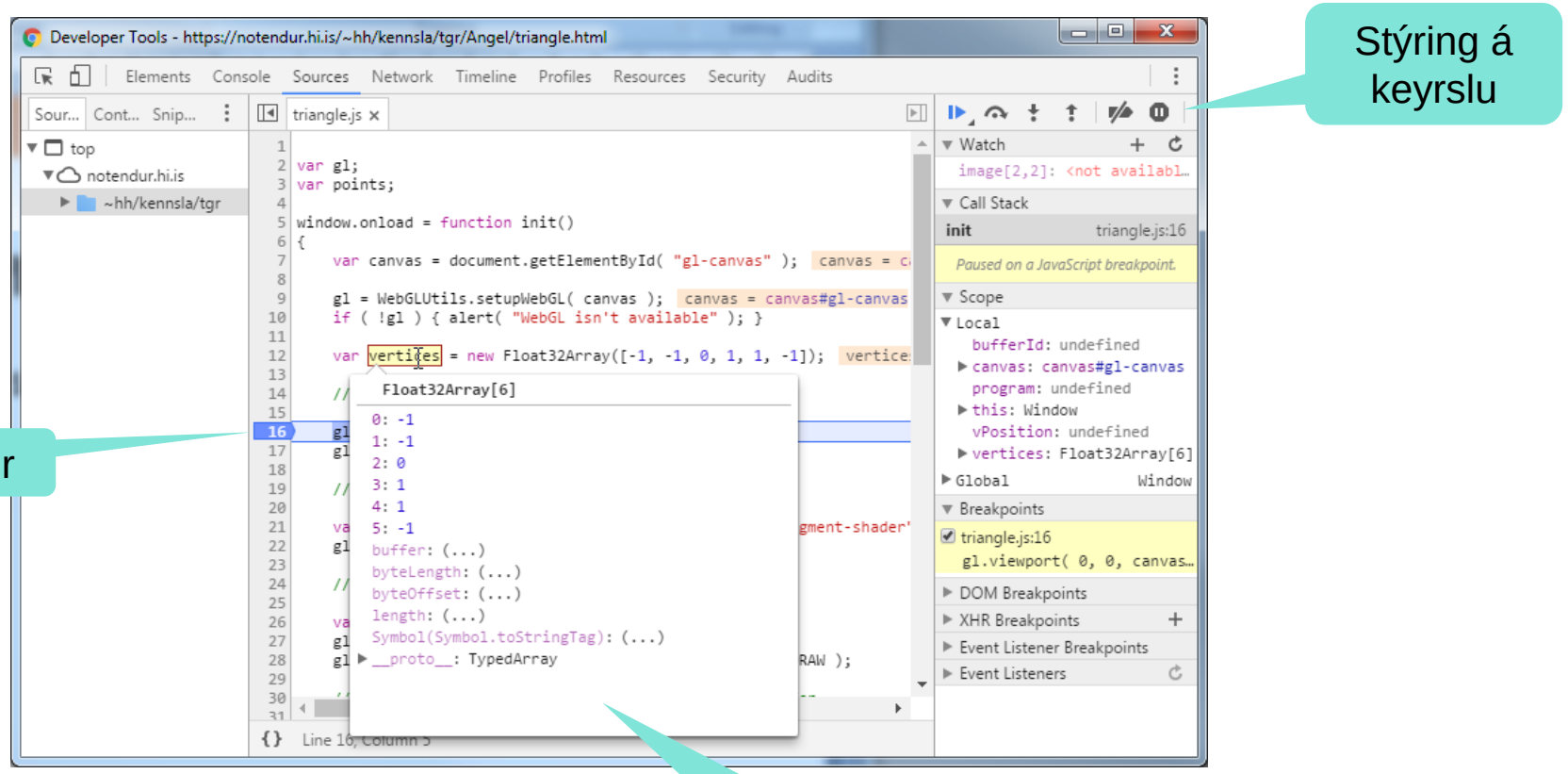
Þróunarverkfæri vafra

- Flestir vafrar hafa innbyggð þróunartól
  - Leyfa notendum að rekja sig í gegnum JS forritið, skoða breytugildi og sjá villur
- Chrome DevTools
  - Opna með **Ctrl+Shift+I**, eða **F12**
  - Velja svo "*Sources*" flipann
- Firefox Developer Tools
  - Opna með **Ctrl+Shift+I**, eða **F12**
  - Velja svo "*Debugger*" flipann



Aflúsun (*debugging*) JS forrita

- Oftast best að setja rofstað (*breakpoint*) og rekja sig svo þaðan



Aflúsun litara (*shaders*)

- Ekki hægt að rekja sig í gegnum keyrslu litara
  - Hnútalitari keyrir einu sinni fyrir hvern hnút
- Ef villa í litara þá kemur hún oftast fram við þýðingu
  - Þýðing litaranna er framkvæmd í **`initShaders`** fallinu
  - Villuskilaboð eru skrifuð á "*Consol*e" glugga, en líka hægt að setja rofstað við kall á **`gl.compileShader`** og skoða skilaboðin

Prófið að breyta hnútalitaranum þannig að hann sé rangur og sjá hvernig það lýsir sér

Sjáum síðar hvernig hægt að aflúsa litara

Fyrirlestraæfingar

1. Hvaða útgáfa af OpenGL samsvarar best WebGL 2.0?
2. Hverju þyrfti að breyta í `triangle`-forritinu til að teikna bláan þríhyrning sem er á hvolfi (þ.e. oddurinn snýr niður)?
3. Hvað þarf marga punkta til að teikna ferning með því að nota:
   a. **`gl.TRIANGLES`**
   b. **`gl.TRIANGLE_FAN`**