# WebGL

Szécsi László

3D Grafikus Rendszerek

1. labor

# Gradle

- kezdeti projekt letöltése a Moodle-ről
  - ne legyen a pathban space, spec. karakter, ékezetes betű!!!
- Gradle telepítése

  https://gradle.org/next-steps/?version=6.8.2&format=bin

  unzip: C:\SDK\Gradle
- parancssorból a projektkönyvtárban (vagy system settingsben)

  ```
  path %path%C:\SDK\Gradle\bin;
  set JAVA_HOME="ahol a JDK van a gépen"
  gradle build
  ```
- eredmény
  - `client/src/main/content/` --copy--> `build/web/`
  - `client/src/main/shaders/` --copy--> `build/web/`
  - `client/src/main/kotlin/` --kotlin2js--> `build/web/client.js`

amíg ez nem jó, a hibaüzenet:
`unrecognized binary`

ha elírtuk, a hibaüzenet:
`invalid JAVA_HOME`

# Webszerver

- node.js portable letöltése

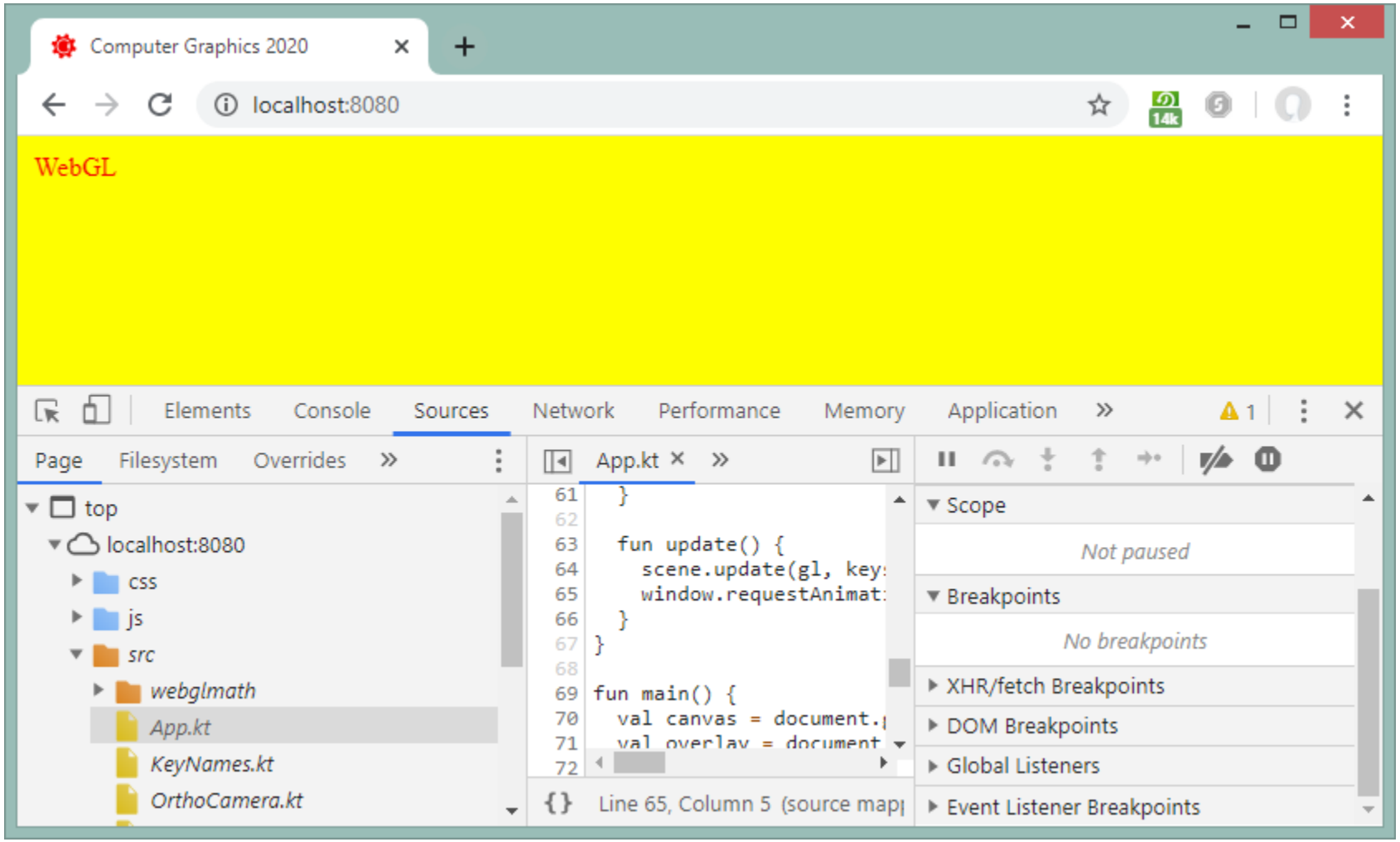
https://github.com/crazy-max/nodejs-portable/releases/download/2.10.0/nodejs-portable.exe

- exe futtatása, 1-es opció mindent feltelepít
- exe futtatása, 2-es opció node-os promptot ad
- navigáljunk a projektkönyvtárba

```
npm install http-server --global
http-server .
```

- ha a tűzfal nem enged ki, az NEM BAJ
- Chrome: http://localhost:8080/build/web
  - vagy ami portot a webszerver kiírt

# Kiindulási állapot



# Debuggolás

- source map van, a `.kt` fileokba breakpoint rakható
- blackboxing nincs, tehát néha oda is be fog lépni, ami nem érdekel minket
  - meneküljünk ilyenkor *step out*-tal
- forrás módosítása és újrafordítása után
  - `ctrl+reload` vagy `ctrl+F5`
  - ha debuggolni akarunk, akkor `ctrl+shift+del` is
    - *cached images and files* törlése
    - különben a source mapekből vett `.kt` kódokat nem frissíti

# Ismerkedés az árnyalókkal

- Változtassa meg a téglalap színét!
  - a képpont-árnyalóban
- Kicsinyítse felére a téglalapot!
  - a csúcspont-árnyalóban
  - bedrótozva (mátrix, változtatható skálafaktor nélkül)
- Változtassa meg a háttérszínt!
  - a `Scene::update`-ben

# Ismerkedés a geometriával

- A `TexturedQuadGeometry` mintájára gyártson egy `TexturedTriangleGeometry` osztályt
  - új `.kt` file
  - kevesebb csúcspont
- A `Scene`-ben hozzon létre egy példányt, és rajzolja ki a `Scene::update`-ben
  - a meglevő programmal mehet ez is

# Ismerkedés az árnyalóprogrammal

- Rajzolja a háromszöget más képpont-árnyalóval!
  - új, árnyalókódot tartalmazó GLSL file
    - a `client/src/main/shaders/solid-fs.glsl` mintájára
    - pl.: `client/src/main/shaders/garish-fs.glsl`
    - adjon vissza egyelőre csak más színt
  - a `Scene`-ben új `Shader`, új `Program` példány létrehozása
  - rajzolás az új programmal

# Csúcspont-attribútumok és árnyaló ki-bementek

- Rajzoljon színátmenetes háromszöget!
  - új csúcspont-árnyaló (pl. `trafo-vs.glsl`)
    - `idle-vs.glsl` mintájára
    - a `vertexPosition` attribútum mellett a `vertexTexCoord` attribútumot is használjuk bemenetként
    - új kimenet: `out vec2 tex;`
    - az attribútum értékét adjuk át a kimenetnek (sima értékadás, =)
    - az új VS-t használjuk a `Program` példányban
  - képpont-árnyalóba
    - új bemenet: `in vec2 tex;`
    - a visszaadott szín két csatornája legyen a `tex.xy`
- Változtassa a színeket a `TexturedTriangleGeometry.kt`-ben!

# Uniform paraméter

- Vegyen fel a csúcspont-árnyalóba egy uniform paramétert.

```
uniform struct{
  mat4 modelMatrix;
} gameObject;
```

- Az árnyalóban transzformálja a csúcspont pozícióját a mátrixszal (Kotlin tömb: row major, GLSL Mat4: column major)
- a `Scene::update`-ben állítsa be a uniform értékét
  - lásd a következő dián

# Uniform beállítása

```
import org.khronos.webgl.Float32Array
```

```
gl.uniformMatrix4fv(
  gl.getUniformLocation(
    garishProgram.glProgram,
   "gameObject.modelMatrix"),
  false,
  Float32Array( arrayOf<Float>(
    1.0f, 0.0f, 0.0f, 0.1f,
    0.0f, 1.0f, 0.0f, 0.2f,
    0.0f, 0.0f, 1.0f, 0.0f,
    0.0f, 0.0f, 0.0f, 1.0f
)))
```

nagyon fapados, nem hatékony, könnyű elrontani és nehéz megkeresni a hibát

column major
persze attól függ az eltolásmátrix, hogy merről szorzunk a shaderben

# Mozgó háromszög 1.

- Vegyen fel a `Scene`-be egy, a shaderbeli uniformnak megfelelő tulajdonságot

```
val modelMatrix = Mat4()
```

- A `Scene::update`-ben változtassa a fenti tulajdonságot, pl.

```
modelMatrix.translate(  )
```

# Mozgó háromszög 2.

- A `Scene::update`-ben töltse fel a uniformba
  - mint eddig, de a `modelMatrix.storage` legyen a `Float32Array` tömb

- eredmény: mozgó háromszög
  - sebesség: képfrissítési frekvenciától függ

# Feladat: simán mozgó háromszög

- eltelt idő

```
//properties
val timeAtFirstFrame = Date().getTime()
var timeAtLastFrame =  timeAtFirstFrame
```

```
//update
val timeAtThisFrame = Date().getTime()
val dt = (timeAtThisFrame - timeAtLastFrame).toFloat() / 1000.0f
val t = (timeAtThisFrame - timeAtFirstFrame).toFloat() / 1000.0f
timeAtLastFrame = timeAtThisFrame
```

- Newton $\mathrm{d}\boldsymbol{s} = \boldsymbol{v} \cdot \mathrm{d}t$

```
modelMatrix.translate(velocity * dt)
```

# Eseménykezelés

- a `Scene::update`-ben csak akkor mozgasson, ha a gomb épp le van nyomva
  - használja a `keysPressed` halmazt
  - kezeljen több gombot, több mozgásirányt

# További feladatok

- háromszög elrejtése/mutatása gombnyomásra
- háromszög színének változtatása gombnyomással
- háromszög mozgatása egérrel
  - eseményfigyelők az App-ban regisztrálva vannak
  - `event.clientX` és `event.clientY` a kattintás koordinátái pixelekben
  - NEM kell a keysPressedhez hasonlóan elmenteni az eseményadatokat és az update-ben reagálni, hanem azonnal lehet módosítani a scene-t. A megjelenítés természetesen az updateben lesz.

# Textúrázott háromszög

- képpont-árnyalóba
  - új uniform: `uniform struct{ sampler2D colorTexture; } material;`
  - szín: `texture(material.colorTexture, tex);`
- `Texture2D` objektum
  - létrehozása a `Scene` propertyjeként

  `"media/asteroid.png"`
  - `Scene::update`-ben hozzárendelés a `colorTexture` uniform változóhoz

```
gl.uniform1i(gl.getUniformLocation(
      garishProgram.glProgram, "material.colorTexture"), 0)
gl.activeTexture(GL.TEXTURE0)
gl.bindTexture(GL.TEXTURE_2D, asteroidTexture.glTexture)
```

# Enable alpha blending

```
gl.enable(GL.BLEND)
gl.blendFunc( GL.SRC_ALPHA, GL.ONE_MINUS_SRC_ALPHA)
```

# 1. HÁZI FELADAT: animált textúra


- jelenítsen meg egy textúrázott téglalapot
- legyen a textúrában egy képsorozat
- a shaderben a textúrakoordináták skálázásával érje el, hogy csak az egyik képelem látszódjon
- aztán adjon hozzá alkalmas offszetértéket úgy, hogy egy másik animációs fázis látszódjon
- legyen az offszet egy uniform paraméter
- Kotlinból változtassa az offszetet úgy, hogy változzon a fázis
  - jobb eleve 2D-s offszetet átadni, mint képpontonként matekozni
- mozgassa a téglalapot úgy, hogy az animációnak megfelelően mozgó karaktert kapjon (ne moonwalkoljon)
- legalább két animáció-szekvenciát tudjon (pl. sétál, fut)