WebGL and PureScript

Wrangling the **worst API in the world**

What is WebGL?

- Alternative (experimental) HTML Canvas context
- Allows rendering of 3D graphics via JavaScript
- Mirrors the OpenGL API via methods on the context

What is Wrong with WebGL?

- All the known problems with JavaScript *and…*
- Impenetrable state machine interface
- Shaders passed as Strings and compiled ad-hoc
- Unused bindings silently removed
- Silent failure (and error-checking is expensive!)

Super sad face!

PureScript…

…to the rescue!

What makes a good library?

The code:

- **Informs** the user about the domain
- **Assists** the user in performing correct operations
- **Without prohibiting** advanced users / usage

WAI is a great example of this approach!

Let’s write a good library!

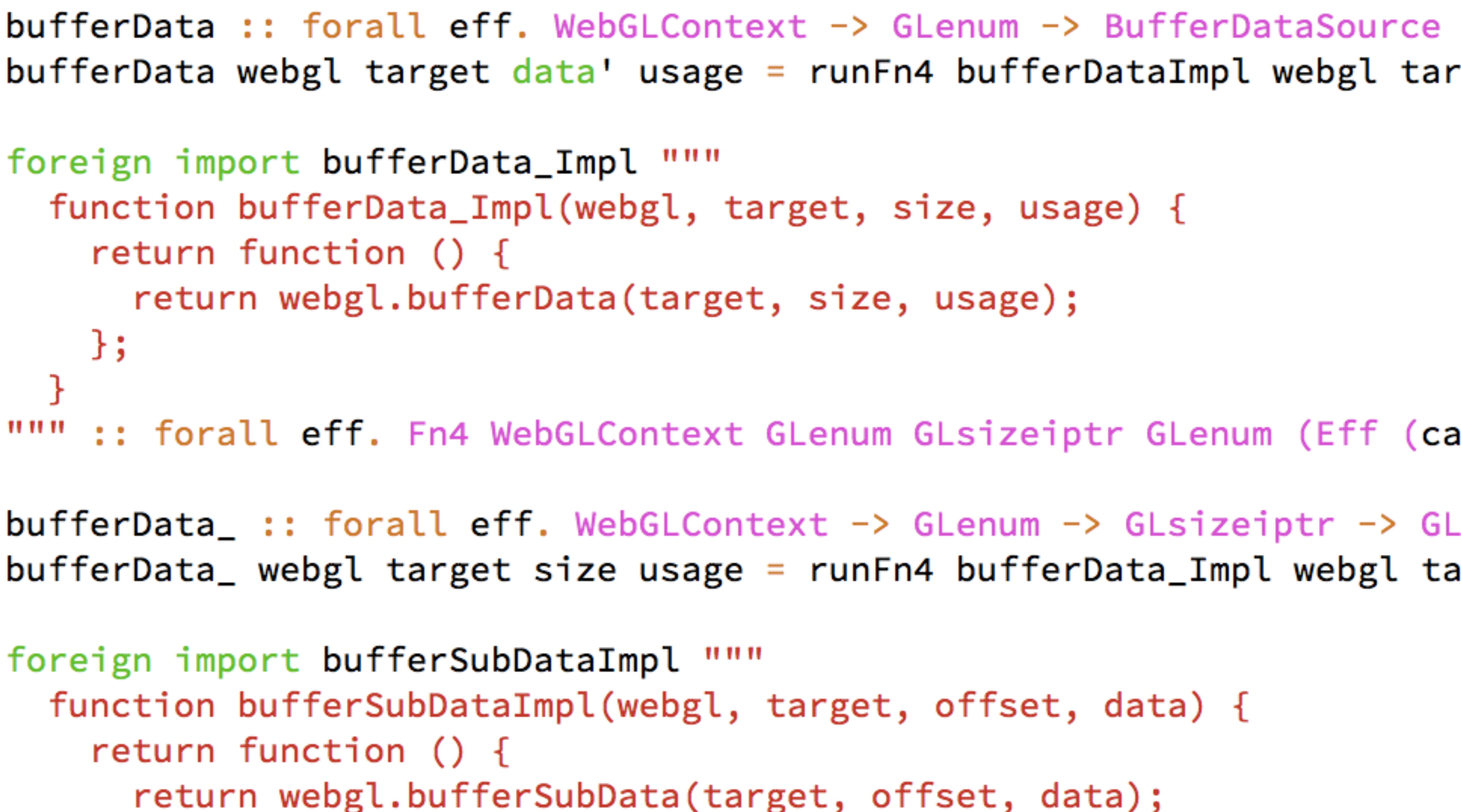
- Start with wrapped low-level operations
- Compose higher-level helpers from these operations
- Guide usage via types

purescript-webgl-raw

Parses the WebGL spec to provide:

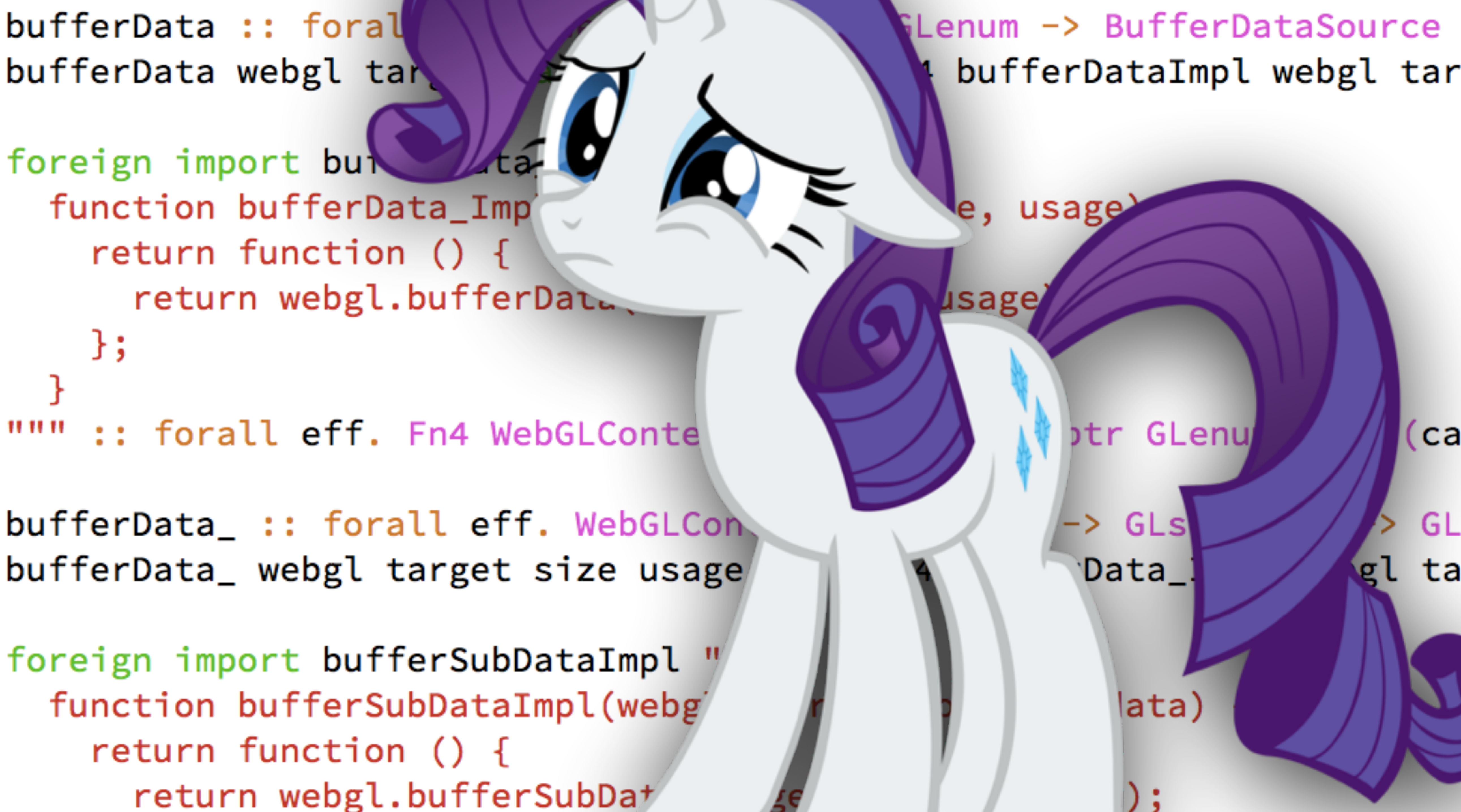
- Standard types (imports and aliases)
- Raw functions with basic type safety
- Enumerables (WebGL constants)

purescript-webgl-raw



```
bufferData :: forall eff. WebGLContext -> GLenum -> BufferDataSource
bufferData webgl target data' usage = runFn4 bufferDataImpl webgl tar

foreign import bufferData_Impl """
  function bufferData_Impl(webgl, target, size, usage) {
    return function () {
      return webgl.bufferData(target, size, usage);
    };
  }
""" :: forall eff. Fn4 WebGLContext GLenum GLsizeiptr GLenum (Eff (ca

bufferData_ :: forall eff. WebGLContext -> GLenum -> GLsizeiptr -> GL
bufferData_ webgl target size usage = runFn4 bufferData_Impl webgl ta

foreign import bufferSubDataImpl """
  function bufferSubDataImpl(webgl, target, offset, data) {
    return function () {
      return webgl.bufferSubData(target, offset, data);
```

purescript-webgl-raw



```
bufferData :: foral                            GLenum -> BufferDataSource
bufferData webgl tar                          4 bufferDataImpl webgl tar

foreign import bu      ta
  function bufferData_Imp                       e, usage)
    return function () {
      return webgl.bufferDat                  usage)
    };
  }
""" :: forall eff. Fn4 WebGLConte                 tr GLenu      (ca

bufferData_ :: forall eff. WebGLCon             -> GLs       -> GL
bufferData_ webgl target size usage      4       Data_        gl ta

foreign import bufferSubDataImpl "
  function bufferSubDataImpl(webg                      ata)
    return function () {
      return webgl.bufferSubDat                      );
```

purescript-webgl-monad

- Threads WebGL context via `ReaderT`
- Catches OpenGL and WebGL errors via `ErrorT`
- Uses `Canvas` effect from `purescript-canvas`

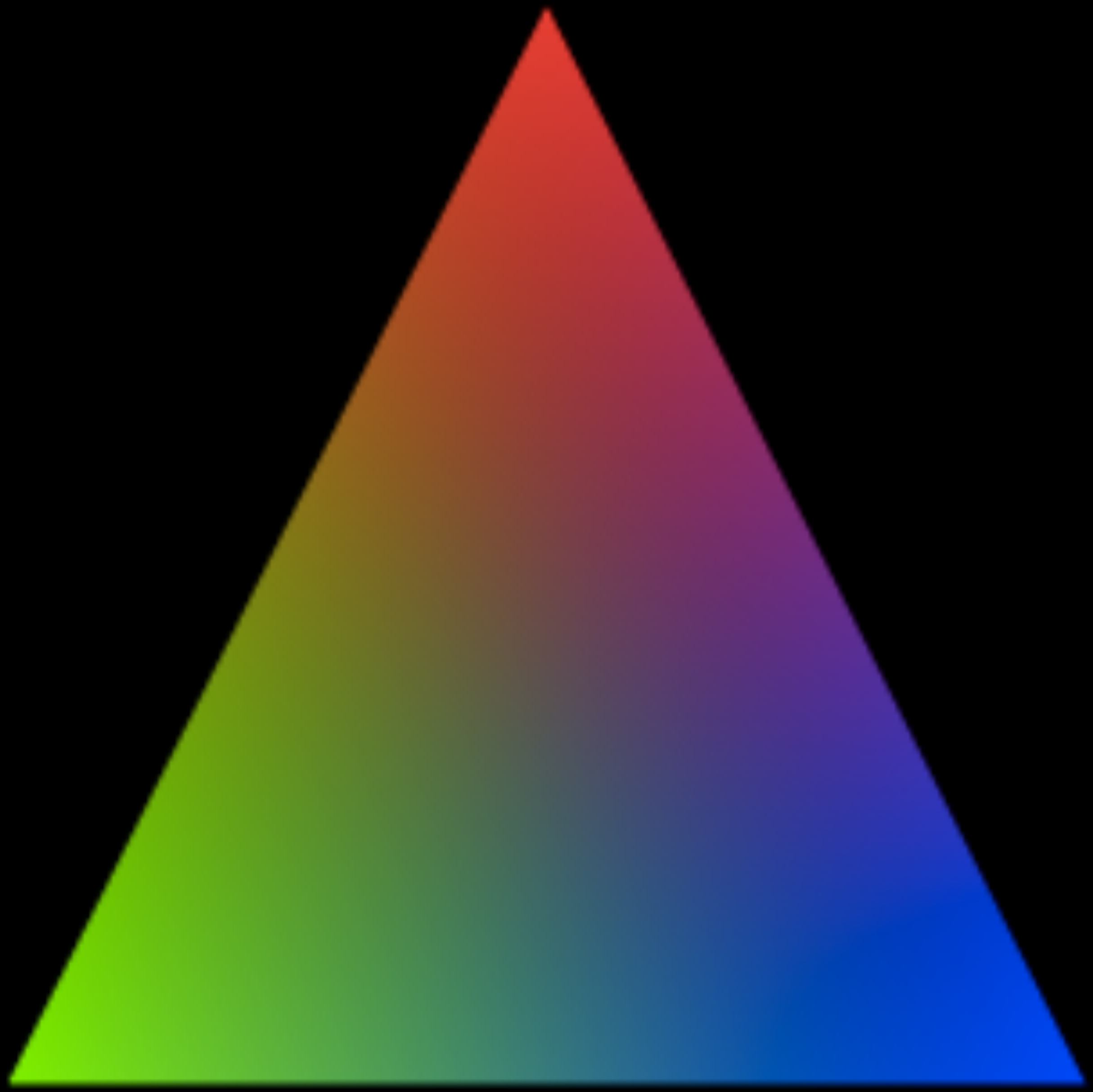
purescript-webgl-monad

```
bindBuffer :: ArrayBufferType -> WebGLBuffer -> WebGL Unit
bindBuffer btype buffer = do
    ctx <- ask
    liftEff $ Raw.bindBuffer ctx (toWebglEnum btype) buffer

bufferData :: ArrayBufferType -> BufferData -> BufferUsage -> WebGL U
bufferData btype datatype usage = do
    ctx <- ask
    liftEff $ case datatype of
      (DataSource ns) -> Raw.bufferData ctx (toWebglEnum btype) ns (t
      (DataSize n)    -> Raw.bufferData_ ctx (toWebglEnum btype) n (t
```

purescript-webgl-monad

```
addShaderToProgram :: WebGLProgram -> ShaderType -> String -> WebGL Uni
addShaderToProgram prog stype src = do
    shader <- GL.createShader stype
    GL.shaderSource shader src
    GL.compileShader shader
    GL.attachShader prog shader

compileShadersIntoProgram :: String -> String -> WebGL WebGLProgram
compileShadersIntoProgram vertSrc fragSrc = do
    prog <- GL.createProgram
    addShaderToProgram prog VertexShader vertSrc
    addShaderToProgram prog FragmentShader fragSrc
    GL.linkProgram prog

    isLinked <- GL.getProgramParameter prog LinkStatus
    when (not isLinked) (throwError shaderLinkError)

    GL.useProgram prog
```

```
main :: WebGLProgram
     -> { a_Position :: Attribute Vec4, a_Color :: Attribute Vec4 }
     -> { }
     -> WebGL Unit
main _ attr _ = do
    let verticies = asFloat32Array
         [  0.0,  0.5, 1.0, 0.0, 0.0
         , -0.5, -0.5, 0.0, 1.0, 0.0
         ,  0.5, -0.5, 0.0, 0.0, 1.0
         ]

    createBuffer >>= bindBuffer ArrayBuffer
    bufferData ArrayBuffer (DataSource verticies) StaticDraw

    vertexAttribPointer attr.a_Position 2 Float false (5*4) 0
    enableVertexAttribArray attr.a_Position
    vertexAttribPointer attr.a_Color 3 Float false (5*4) (2*4)
    enableVertexAttribArray attr.a_Color

    clearColor 0.0 0.0 0.0 1.0
    clear ColorBuffer
    drawArrays Triangles 0 3
```



Halp!

- Many more WebGL methods need monad wrappers
- Need benchmark cases to ensure performance
- Need a matrix math library for transforms
- Those more experienced with WebGL can write more comprehensive high-level functions
- **Github issues and PRs welcome!**

Thanks!

- **Phil Freeman** for apparently never sleeping
- **Jürgen Nicklisch-Franken** for the original Khronos IDL parser