PRINSIP BAHASA PEMROGRAMAN

Week 8 – Praktikum 7 HASKELL OpenGL

Disusun oleh:

Muhamad Mathar Rizqi

(221524014)

Jurusan Teknik Komputer dan Informatika

Program Studi D-4 Teknik Informatika

POLITEKNIK NEGERI BANDUNG

Jl. Gegerkalong Hilir, Ciwaruga, Kec. Parongpong, Kabupaten Bandung Barat, Jawa Barat 40559

DAFTAR ISI

Eksplorasi Mandiri

Library

```
import Prelude hiding ( sum )
import Control.Applicative
import Control.Exception ( IOException, catch )
import Control.Monad ( when, unless )
import qualified Data.ByteString as B
import Data.Foldable ( Foldable, sum )
import Data.IORef
import System.Exit
import Graphics.UI.GLUT
```

1. import Prelude hiding (sum): Mengimpor modul standar Haskell (Prelude) tetapi menyembunyikan definisi `sum` agar tidak ada konflik.
2. import Control.Applicative: Mengimpor modul untuk pemrograman aplikatif.
3. import Control.Exception (IOException, catch): Mengimpor modul untuk mengelola pengecualian (exceptions).
4. import Control.Monad (when, unless): Mengimpor modul untuk pemrograman monadik.
5. import qualified Data.ByteString as B: Mengimpor modul untuk manipulasi byte strings dengan alias `B`.
6. import Data.Foldable (Foldable, sum): Mengimpor modul yang berhubungan dengan koleksi data yang dapat dilipat (folded) dan mendefinisikan ulang `sum`.
7. import Data.IORef: Mengimpor modul yang digunakan untuk mengelola mutable state.
8. import System.Exit: Mengimpor modul yang berisi fungsi-fungsi pengakhiran program.
9. import Graphics.UI.GLUT: Mengimpor modul yang berhubungan dengan pemrograman grafis menggunakan GLUT dalam Haskell.

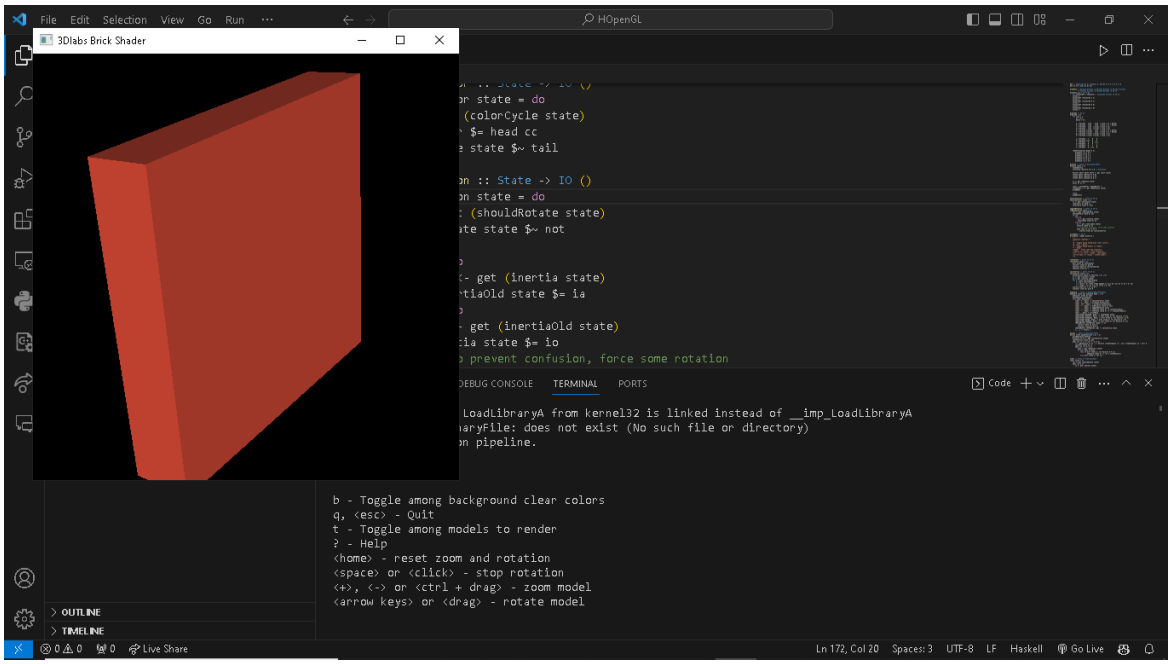
infixl adalah notasi dalam Haskell yang digunakan untuk mendefinisikan operator dengan asosiativitas kiri (left-associative) dan mengatur prioritas operator dalam ekspresi. Ini mempengaruhi urutan evaluasi operator dalam ekspresi.

```haskell
data State = State {
  diff :: IORef (Vector3 GLfloat),
  lastIncr :: IORef (Vector3 GLfloat),
  inertia :: IORef (Vector3 GLfloat),
  inertiaOld :: IORef (Vector3 GLfloat),
  theScale :: IORef GLfloat,
  lastPosition :: IORef Position,
  shouldRotate :: IORef Bool,
  colorCycle :: IORef [Color4 GLclampf],
  modelCycle :: IORef [IO ()],
  modifiers :: IORef Modifiers
  }
```

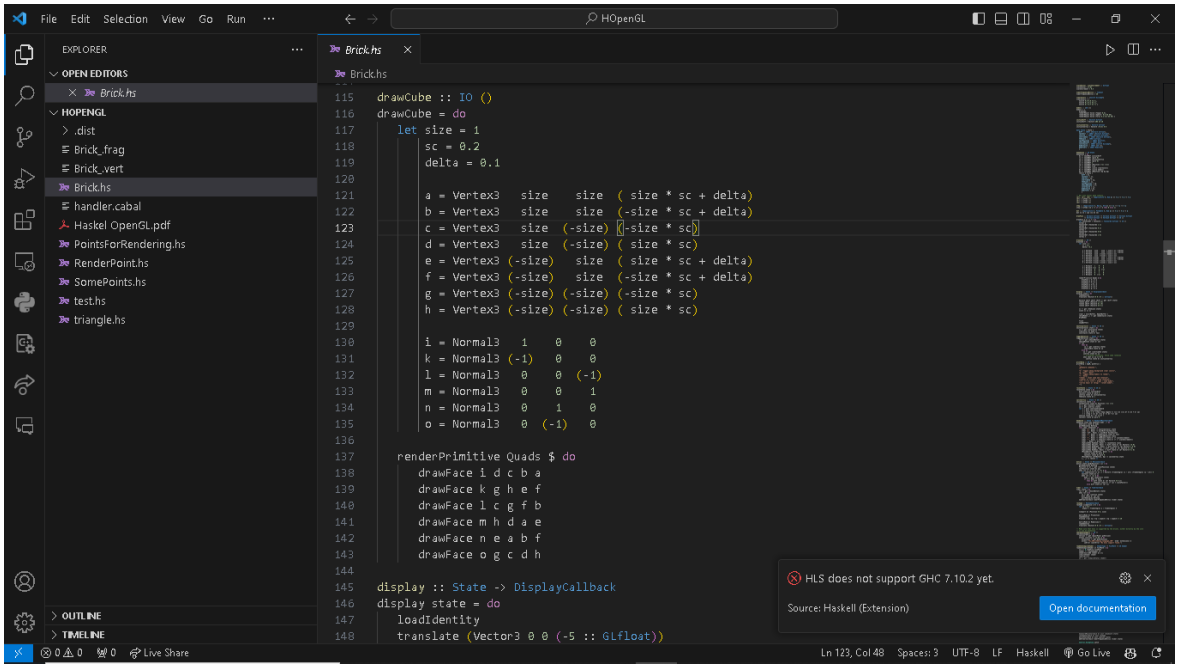
- diff: Ini adalah bidang yang berisi referensi ke IORef yang menyimpan nilai Vector3 GLfloat.
- lastIncr: Mirip dengan diff, bidang ini juga berisi referensi ke IORef yang menyimpan Vector3 GLfloat.
- inertia: Bidang ini berisi referensi ke IORef yang menyimpan Vector3 GLfloat.
- inertiaOld: Seperti inertia, bidang ini juga berisi referensi ke IORef yang menyimpan Vector3 GLfloat.
- theScale: Ini adalah bidang yang berisi referensi ke IORef yang menyimpan nilai bertipe GLfloat.
- lastPosition: Bidang ini berisi referensi ke IORef yang menyimpan data bertipe Position.
- shouldRotate: Ini adalah bidang yang berisi referensi ke IORef yang menyimpan nilai bertipe Bool.
- colorCycle: Bidang ini berisi referensi ke IORef yang menyimpan daftar ([Color4 GLclampf]).
- modelCycle: Bidang ini berisi referensi ke IORef yang menyimpan daftar aksi IO ([IO ()]).
- modifiers: Ini adalah bidang yang berisi referensi ke IORef yang menyimpan data bertipe Modifiers.

Modifikasi Program

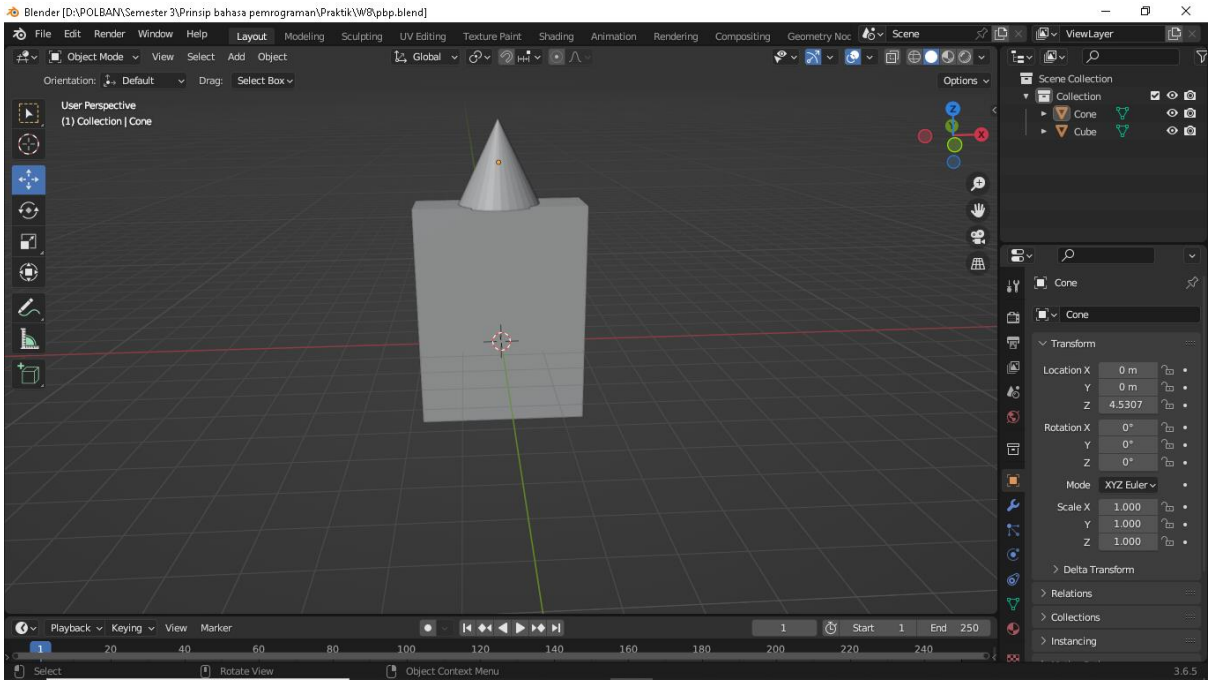
Saat program pertama kali dijalankan akan muncul seperti ini



Saya melakukan modifikasi pada modul drawCube



Saya akan melakukan modifikasi dengan mengganti bata terserbut menjadi sebuah 3D model yang saya buat di aplikasi eksternal yaitu blender



Setelah model terbuat, ekstrak menjadi file dengan ekstensi .obj, setelah itu saya menggunakan code yang dibuat oleh rekan saya Aryo Rakatama untuk mengkonversi file tersebut agar bisa di gunakan di Haskell OpenGL berikut codenya

Code ObjParser

```
obj_file_path = "pbp.obj"
def read_obj_file(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        obj_data = file.read()
    return obj_data

def convert_obj_to_haskell_glut(obj_data):
    vertices = []
    normals = []
    faces = []
    listNormal = []
    numNormal = 0
```

```
    # Membaca data obj
    for line in obj_data.split('\n'):
        tokens = line.split()
        if tokens:
            if tokens[0] == 'v':
                x, y, z = map(float, tokens[1:4])
                vertices.append(f"Vertex3 ({x}) ({y}) ({z})")
            elif tokens[0] == 'vn':
                nx, ny, nz = map(float, tokens[1:4])
                normals.append(f"Normal3 ({nx}) ({ny})
({nz})")
                listNormal.append(numNormal)
                numNormal += 1
            elif tokens[0] == 'f':
                face_indices = [list(map(int, val.split('/')))
for val in tokens[1:]]
                faces.append(face_indices)

    haskell_code = ""
    haskell_code += "  let\n"



    for i, vertex in enumerate(vertices):
        haskell_code += f"      v{i} = {vertex}\n"
    for i, normal in enumerate(normals):
        haskell_code += f"      n{i} = {normal}\n"

    haskell_code += "  renderPrimitive Polygon $ do\n"

    for i, face_indices in enumerate(faces):

        usedNormal = i
        if usedNormal not in listNormal:
            usedNormal = 0
```

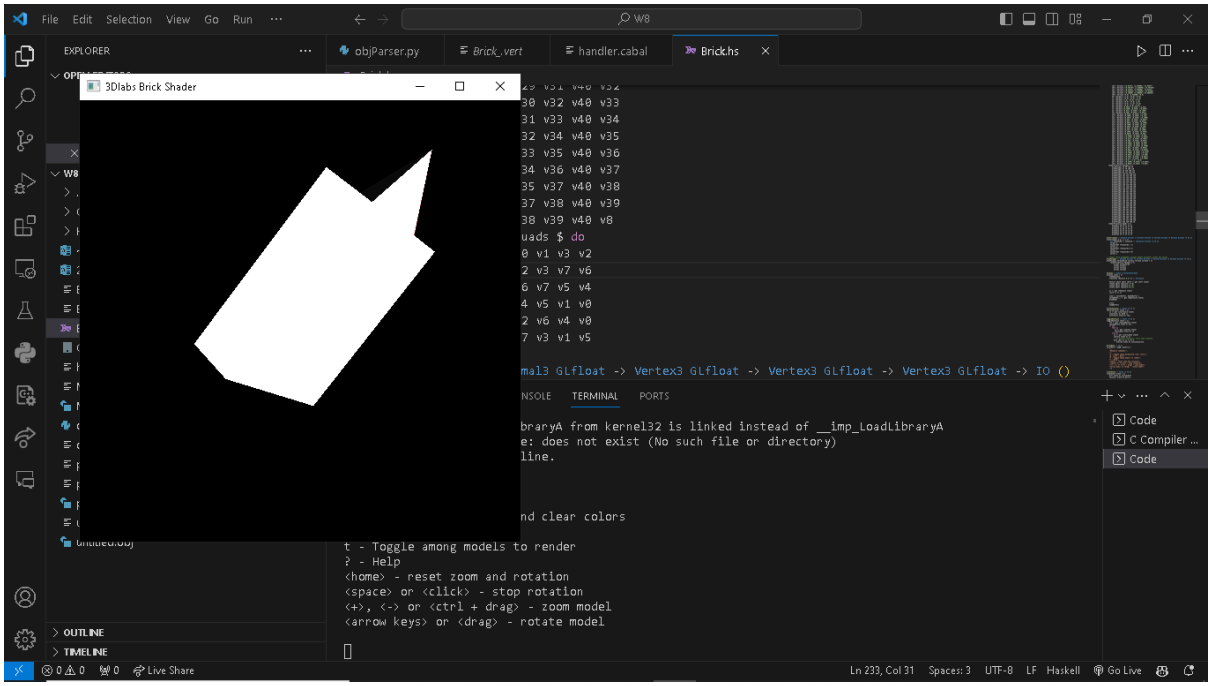
```
        if len(face_indices) == 3:
            haskell_code += f"      drawPolygon n{usedNormal}
"
            for vertex_index, _, normal_index in face_indices:
                vertex_var = f"v{vertex_index - 1}"
                haskell_code += f"{vertex_var} "
            haskell_code += "\n"

    haskell_code += "  renderPrimitive Quads $ do\n"

    for i, face_indices in enumerate(faces):

        usedNormal = i
        if usedNormal not in listNormal:
            usedNormal = 0

        if len(face_indices) == 4:
            haskell_code += f"      drawFace n{usedNormal} "
            for vertex_index, _, normal_index in face_indices:
                vertex_var = f"v{vertex_index - 1}"
                haskell_code += f"{vertex_var} "
            haskell_code += "\n"

    return haskell_code

obj_data = read_obj_file(obj_file_path)

haskell_glut_code = convert_obj_to_haskell_glut(obj_data)

with open("output.txt", "w") as output_file:
    output_file.write(haskell_glut_code)
```

Source code

Karena terlalu Panjang saya upload ke github

https://github.com/Matharrr/Haskell-openGL

Output



Referensi

https://wiki.haskell.org/OpenGLTutorial2

https://github.com/madjestic/Haskell-OpenGL-Tutorial