Istanbul Technical University
Faculty of Computer and Informatics
Computer Engineering Department

BLG 458E
Homework 1 Report

Ömer Malik Kalembaşı - 150180112

November 22th, 2023

1 Explanation and how to run

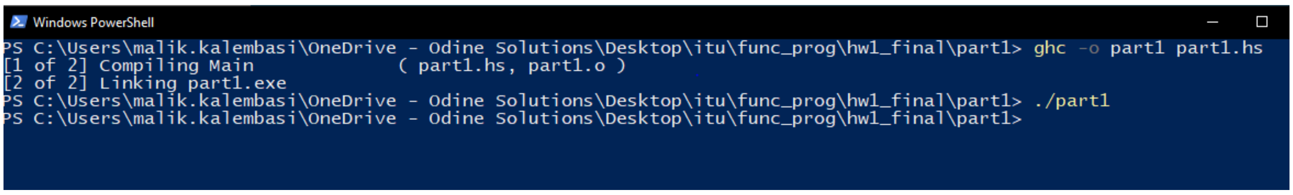
1.1 Part 1 - Sierpinski's triangle

The provided Haskell code generates Sierpinski's triangle fractal. It begins by defining data types for points, triangles, and shapes. The midPoint function calculates the midpoint between two points in 3D space. The subdivideTriangle function takes a triangle and subdivides it into three smaller triangles by finding midpoints of its edges. The core of the code lies in the sierpinskiModel function, which recursively subdivides triangles a specified number of times, creating the Sierpiński pattern. The generateSierpinski function initializes the process with an initial triangle and the desired number of iterations. The code also includes functions to convert triangles and shapes into STL format. Finally, the main function generates the Sierpiński triangle with three iterations and saves it as an STL file named "part1.stl."

Compile and run program with following commands in part1 directory.

```
ghc -o part1 part1.hs
```

```
./part1
```



```
PS C:\Users\malik.kalembasi\OneDrive - Odine Solutions\Desktop\itu\func_prog\hw1_final\part1> ghc -o part1 part1.hs
[1 of 2] Compiling Main             ( part1.hs, part1.o )
[2 of 2] Linking part1.exe
PS C:\Users\malik.kalembasi\OneDrive - Odine Solutions\Desktop\itu\func_prog\hw1_final\part1> ./part1
PS C:\Users\malik.kalembasi\OneDrive - Odine Solutions\Desktop\itu\func_prog\hw1_final\part1>
```

Figure 1: Compile and run part1.hs in the directory part1

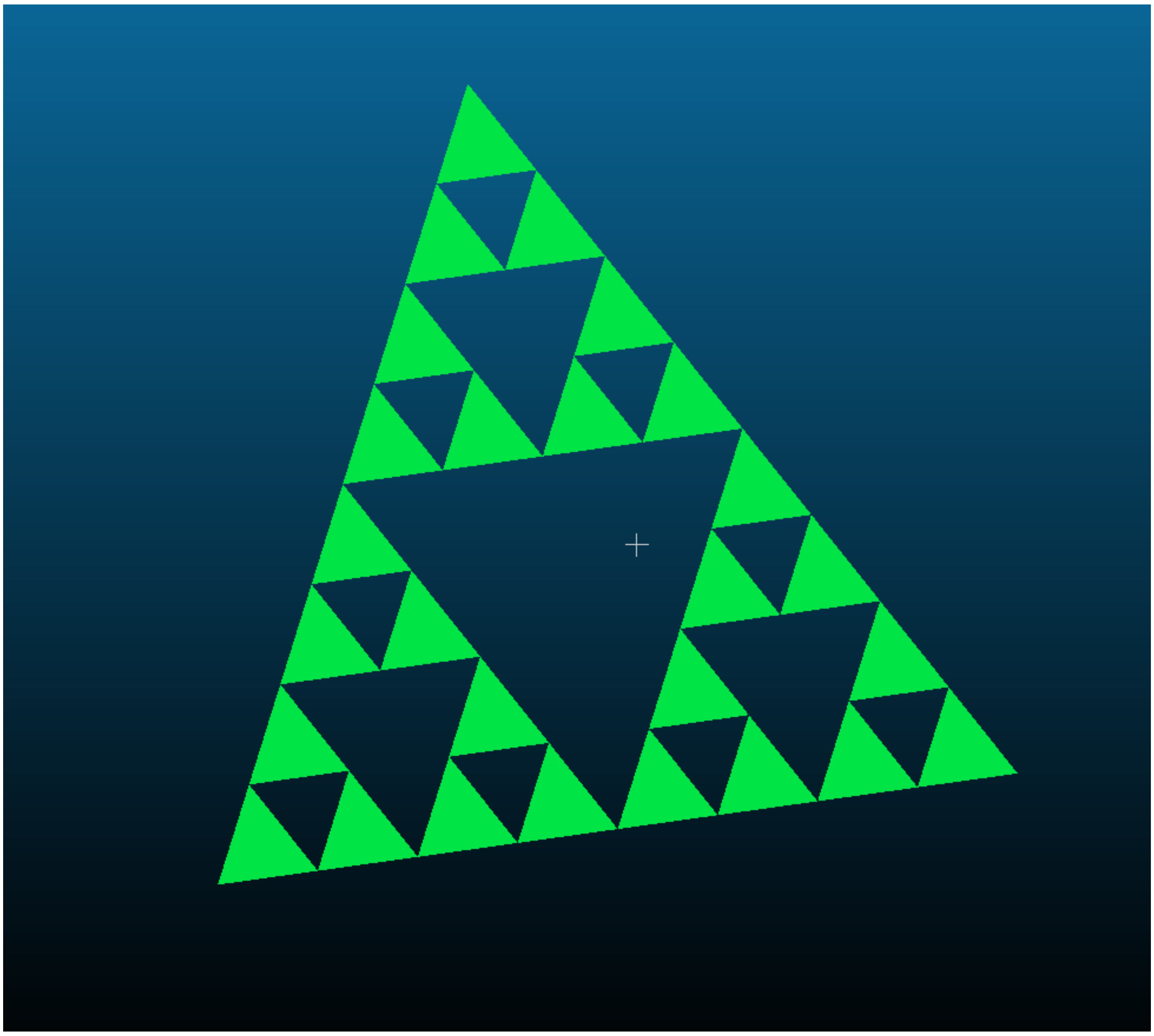


Figure 2: Visual representation of Sierpinski's triangle

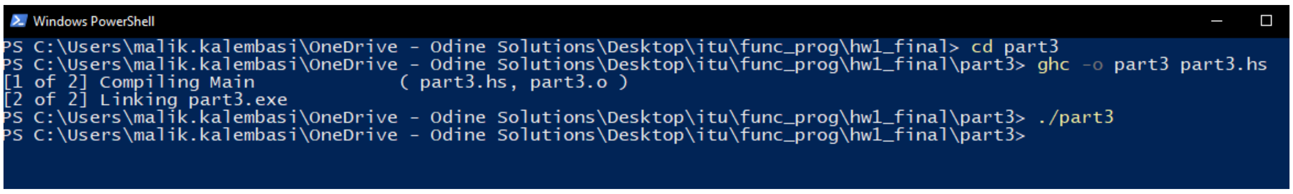
1.2 Part 3 - Cube at a specified position

The provided Haskell code accomplishes the objective of creating a cube centered at a given position with a specified side length. The code defines data types for points, triangles, and shapes, which are essential for working with 3D geometry. The createTriangleDef function converts a triangle into STL format, facilitating the representation of the cube's surfaces as triangles. The createObjectModelString function takes a shape, which consists of triangles, and generates an STL representation of the entire 3D object. The core of the code is the makeCube function, which takes a center point and side length as parameters and constructs a cube composed of 12 triangles. It defines the cube's corners based on the center and half of the side length, ensuring it's centered at the specified position. Finally, the main function creates a cube with a side length of 1 centered at the origin (0, 0, 0) and writes it to an STL file named "part3.stl."

Compile and run program with following commands in part3 directory.

```
ghc -o part3 part3.hs
```

```
./part3
```



```
PS C:\Users\malik.kalembasi\OneDrive - Odine Solutions\Desktop\itu\func_prog\hw1_final> cd part3
PS C:\Users\malik.kalembasi\OneDrive - Odine Solutions\Desktop\itu\func_prog\hw1_final\part3> ghc -o part3 part3.hs
[1 of 2] Compiling Main             ( part3.hs, part3.o )
[2 of 2] Linking part3.exe
PS C:\Users\malik.kalembasi\OneDrive - Odine Solutions\Desktop\itu\func_prog\hw1_final\part3> ./part3
PS C:\Users\malik.kalembasi\OneDrive - Odine Solutions\Desktop\itu\func_prog\hw1_final\part3>
```

Figure 3: Compile and run part3.hs in the directory part3

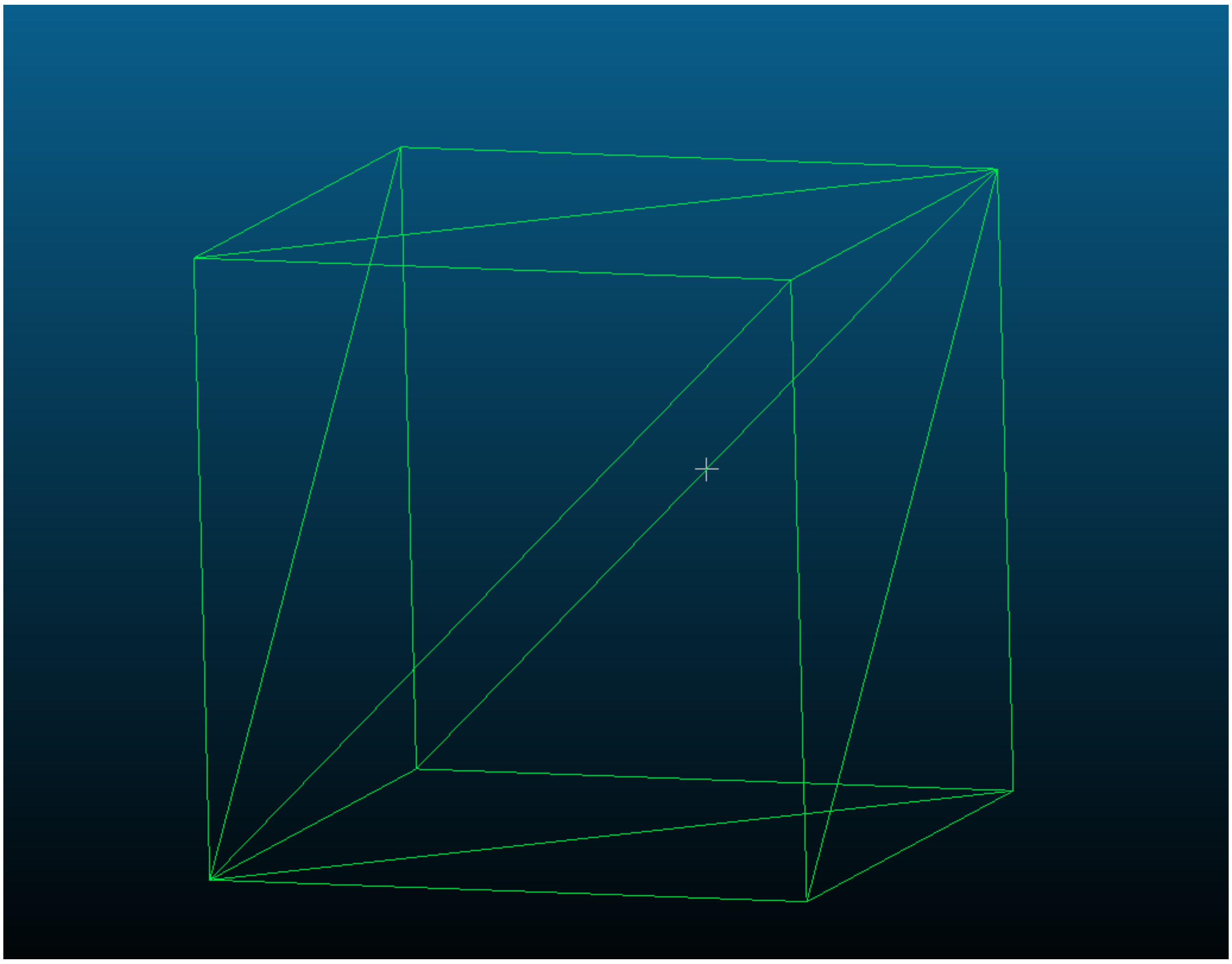


Figure 4: Visual representation of Cube at a specified position

1.3 Part 4 - Cube Pattern

The provided Haskell code accomplishes the objective of generating a fractal composed of cubes centered at a specified position with a given side length. It begins by defining essential data types for points, triangles, and shapes, which are fundamental for working with 3D geometry. The createTriangleDef function converts a triangle into STL format, facilitating the representation of the cubes as triangles. The createObjectModelString function takes a shape, which consists of triangles, and generates an STL representation of the entire 3D fractal object.

The core of the code is the createFractalCubes function, which recursively generates cubes and smaller cubes around the central cube. It allows for the creation of a fractal pattern by specifying the number of iterations, the center position, and the initial cube's side length. The cubeFractal function serves as the entry point, creating the fractal cubes from the origin with the provided number of iterations.

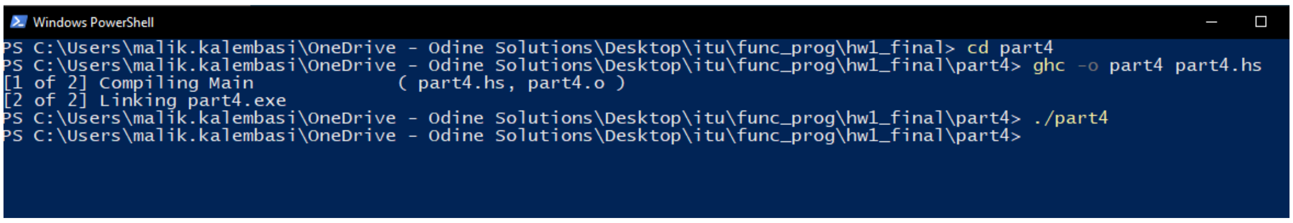
Additionally, the code includes utility functions fst3, snd3, and thd3 to extract the first, second, and third elements of a triple, respectively. These functions are used within the code.

Finally, the main function generates a 3D fractal of cubes with two iterations from the origin, resulting in a visually appealing 3D model and writes it to an STL file named "part4.stl."

Compile and run program with following commands in part4 directory.

```
ghc -o part4 part4.hs
```

```
./part4
```



```
PS C:\Users\malik.kalembasi\OneDrive - Odine Solutions\Desktop\itu\func_prog\hw1_final> cd part4
PS C:\Users\malik.kalembasi\OneDrive - Odine Solutions\Desktop\itu\func_prog\hw1_final\part4> ghc -o part4 part4.hs
[1 of 2] Compiling Main             ( part4.hs, part4.o )
[2 of 2] Linking part4.exe
PS C:\Users\malik.kalembasi\OneDrive - Odine Solutions\Desktop\itu\func_prog\hw1_final\part4> ./part4
PS C:\Users\malik.kalembasi\OneDrive - Odine Solutions\Desktop\itu\func_prog\hw1_final\part4>
```

Figure 5: Compile and run part4.hs in the directory part4

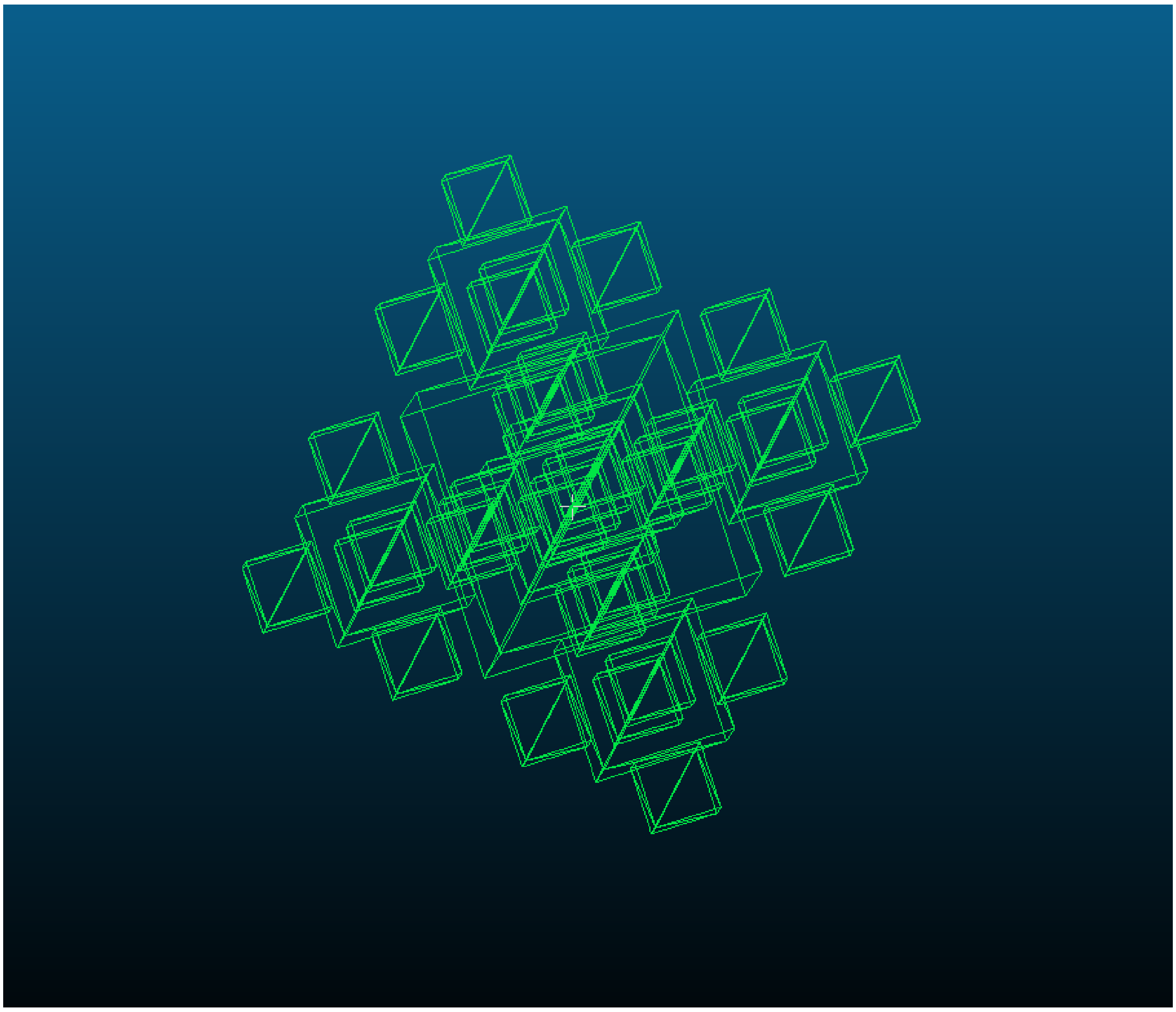


Figure 6: Visual representation of Cube Pattern