*Lior Shtaimberg ,313418469, ליאור שטיימברג, liorst@mta.ac.il*

*Dor Wolfgor ,205892177, דור וולפגור,* *dorwo@mta.ac.il*

Overview

```

The Program, as expected, creates and handles a target graph that a task can run on

As User Interface you get a beautiful display of an automatic directed graph running on the JavaFx framework,

The Engine will run your task using multithreading and analyze your graph for path and circuits.

```

*UI*

```

GraphFx Module:

A lot of the base FX features are thanks to the library "JavaFXSmartGraph" by brunobrunomnsilva which i "forked" and changed to fit the task in hand

a lot of disecting and adding features later we got a root to leaf top to bottom graph with pressing and changing states capabalities you can use to see graph info

and task info even while running on it.

DesktopUI:

this Module is the ui dedicated module to run on tasks, show paths, show circuits and more and serves as the wrapper module to GraphFx.

utilizing the GraphFx module, the center is the graph with all its marvel, automatic layout holds it together, zoom to resize the Pane,

and a reverse button to... oh well try it yourself \*wink

On the right you can see a SideController with the Action Button that give you the control on the graph.

each action has settings that become visible when you enter the "target choosing state", use reset to switch between actions.

on the bottom of the Side Controller you have the theme chooser and graph info.

```

*Engine*

```

Through the engine our whole task running and target analyzing system works

In this exercise we added the serial sets and implemented the task run by running parallel using the thread pool.

1. the graph methods and logic where the directed graph is represented by the `TargetGraph.AdjacentMap` Class that is two Dictioneries of targets names to their childeren in the graph and to their parents

2. The task runner is the task running dedicated class that controls the thread pool using threadExcecutor and a queue that holds the waiting targets and execute it when the terms are met.

3. the Tasks are Simulation and Compilation both with giving update on all the task running and gives a status and a result in the end

```

Task running on the graph implementation

In the TargetGraph class we have two TargetGraph.AdjacentMap graphs

\* the Original that was loaded from the xml file

\* the Current graph that we would be running the task on

We start from the RunTask button opening a task settings window, and waiting for the corrct task information for the client.

from there in case the task already ran on the graph and the user can run on it from the last point

we create a new graph and set the `targetGraphToRunOn` as that or as the original one

Then the user can choose the targets to run on implemented by ChoosingController,

the listener for the chooser is added in the graph panel with onClicked consumer we send it with its initializing

the bottom controller helps you choose all or WhatIf depends or required quickly

If were not running from scratch that targets get filtered from all the Succesful and Warning targets and run again

we create a Queue with the leafs and Independent targets of the relevant graph

then we go through the queue poping targets and checking their status if its waiting or frozen

then we check if all their childeren finished successfully and execute it

if it failed we update all of its ancecstors to skipped.

we keep a counter that counts the finished and skipped and update it when neccecery.

when all the targets done or skipped we are done.

Bonuses

1. Animations - We implemented a visual graph where all the vertices and edges move according to their positions (roots up and leaves down or upside down).

2. Skin replacement for the system - in the main window of the application you can change the skin according to 3 types of skin (dark, blue, white).

3. Presenting the target graph visually - We implemented the graph visually with external library and adapted it according to our needs.

4. Display the target graph in a hierarchical way - in the main window of the application you can change the view by clicking the reverse button.