Predictions – Part 2

Bounses

- 3 Themes – enable with combo box in header (default, barbie world, alien world).
- 2 Animations – enable with check box in header (fade and carousel).
- Graphic view of the world – can view the entities moving in the world when selecting a simulation in result screen.
- Trip in time (past – present) – ability to view the world as it was when simulation is paused.
- Trip in time (future) – ability to manually forward the ticks when simulation is paused.

About

The purpose of the system is to enable the definition of a "world" in which a population exists, and to apply to this population a set of "laws" that change in one way or another (statistically) some of the entities in the world.
The system is a generic system that will allow defining the various entities and the laws that apply to them, thus actually forming an infrastructure and fertile ground for the use of as many simulations as required.
Now you can run simulations in parallel, view the world and analyze their results through a convenient user interface.

Getting Started

Once the program is up, you can choose your favorite theme. It could be a journey in the barbie world, or you can take a rocket ship to the alien world.
Then, you must load a proper simulation file.
The system will alert you if the file does not meet the standard.
After loading the file, you can run your first simulation!
You can choose values for the environment variables or let the system choose them randomly.
Now, you can move to results screen and view the simulation results and see how the world changes over time. (**Bonus** – see the grid that describes the world with different entities in it)
Bonus - you can trip in time when a simulation is paused.

You can click the help button in every screen for more help.

Main Classes

- Validator - Responsible for verifying the integrity of the simulation file input. If a problem is found in the file, an inductive error will be thrown.
- Engine - Responsible for the relationship between the internal operations and the ui, will return dtos.
- HistoryManager - Holds all the information about the completed simulations. We used this class as a base for the system state.
- Manager - Responsible for managing the ui, the requests come from it to the engine and he routes them. We implemented it as a singleton.
- AppController – Responsible for the initialization of all the screens and moving from one to another.
- ResultController – Responsible to fetch data from the engine and display it on the results screen. We create another thread that fetch the data and updates the simulations' values in the table view, moving the entities around the world.

Notes

- We apply regex check in numeric text boxes.
- We decide to alert the user if he wants to load new simulation but a simulation is running or waiting in the queue.

Authors

Omer Hadar, 207388331, omer.hadar14@gmail.com
Dor Wachner, 208277343, wachner.dor@gmail.com