Visual Layout of Graph-Like Models

TAREK SHARBAK

MHDTAREK.SHARBAL@STUDENT.UATNWERPEN.BE

Introduction

- Visual development makes creating complex software a breeze
- The graphical layout of the modelling tools helps with the model comprehension
- Domain-Specific modelling languages need a dynamic layout behavior

Visual Layout

- Treat models as graphs
- Many aspects that can make different graphs easier to read and modify
- Trying to reach an optimization between visual aesthetics
- Use of modelling to model the behavior of the UI of domain specific formalisms

Related Word

- This project was based on the work of Denis Dube in his thesis “Graph Layout for Domain-Specific Modeling”
- The implementation is based on an implementation of rapid UI development using statecharts by Detlev Van Looy

Related Work

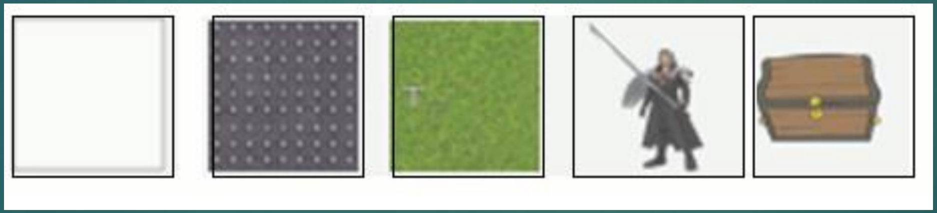
- In Detlev Van Looy's project, he designed a GUI implementation for a statecharts builder along with the abstract syntax and concrete syntax of the UI written in python.
- In this project I reused the components and adapted them to suit my application of making a behavioral UI for the RPG game

RPG Abstract Syntax

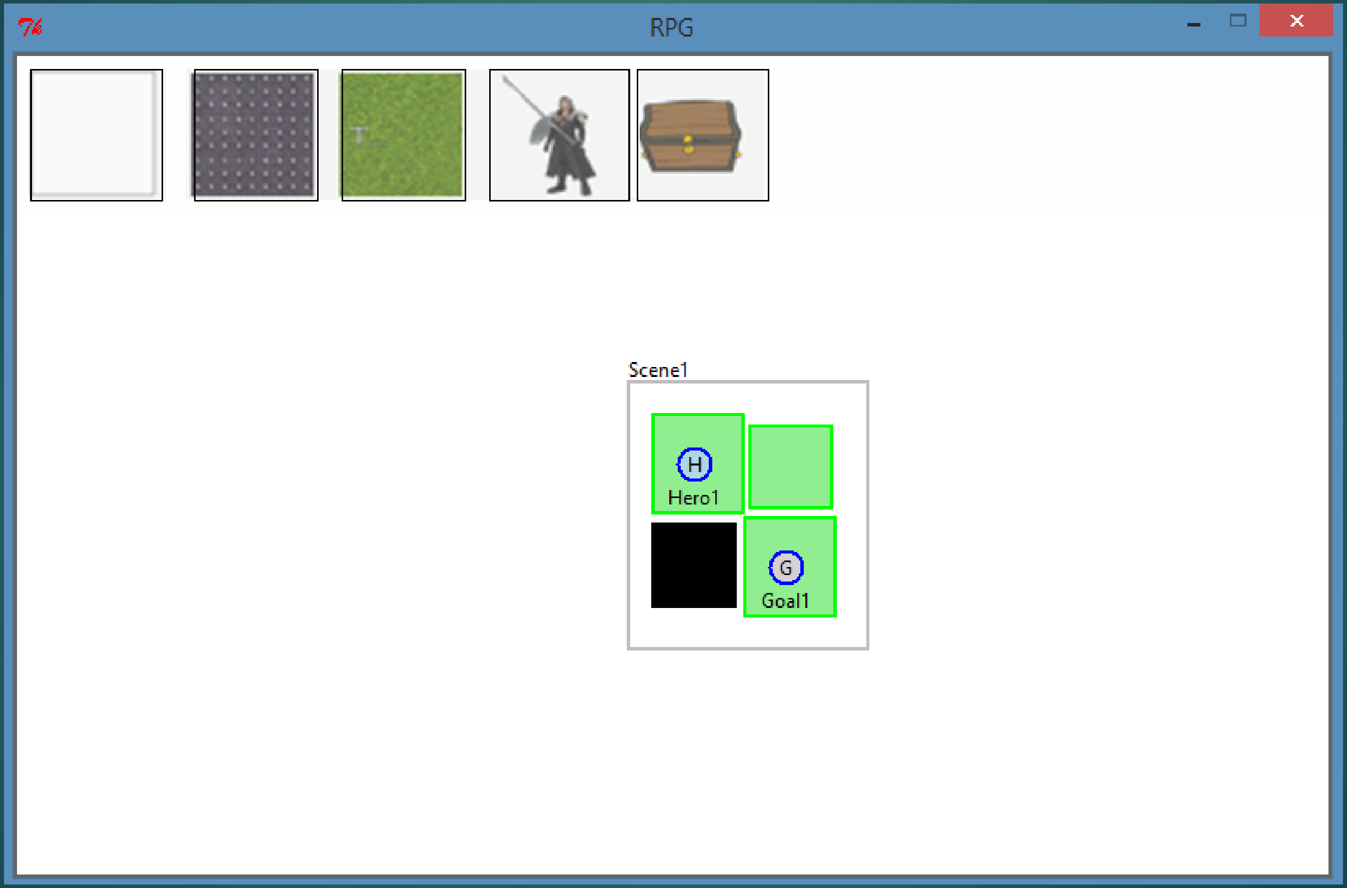
- Written with python
- A class definition for each element of the game (Scene, Standard Tile, Obstacle, Hero, and Goal)
- The implementation of the RPG is minimal since the focus is on the behavioral UI using statecharts

RPG Graphical Interface

- Written in Python
- Use of drawing and positioning functions
- The UI buttons that are used to create the entities



RPG Graphical Interface

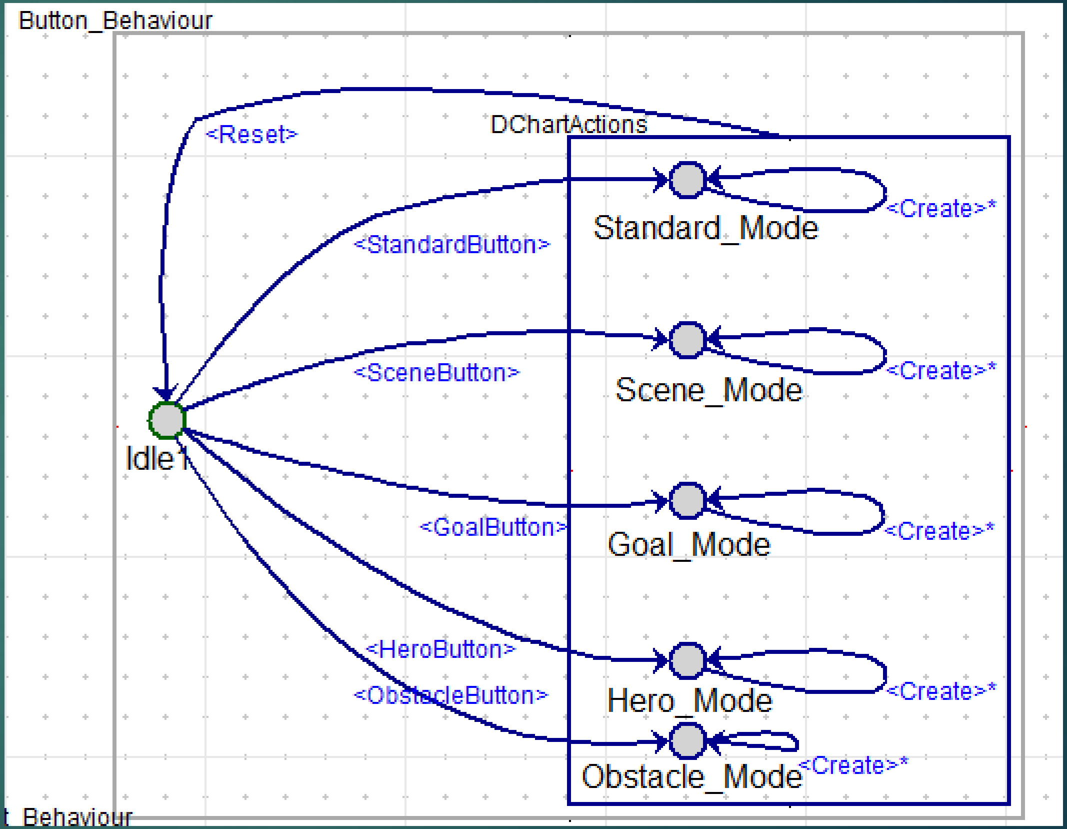


Behavioral Statecharts

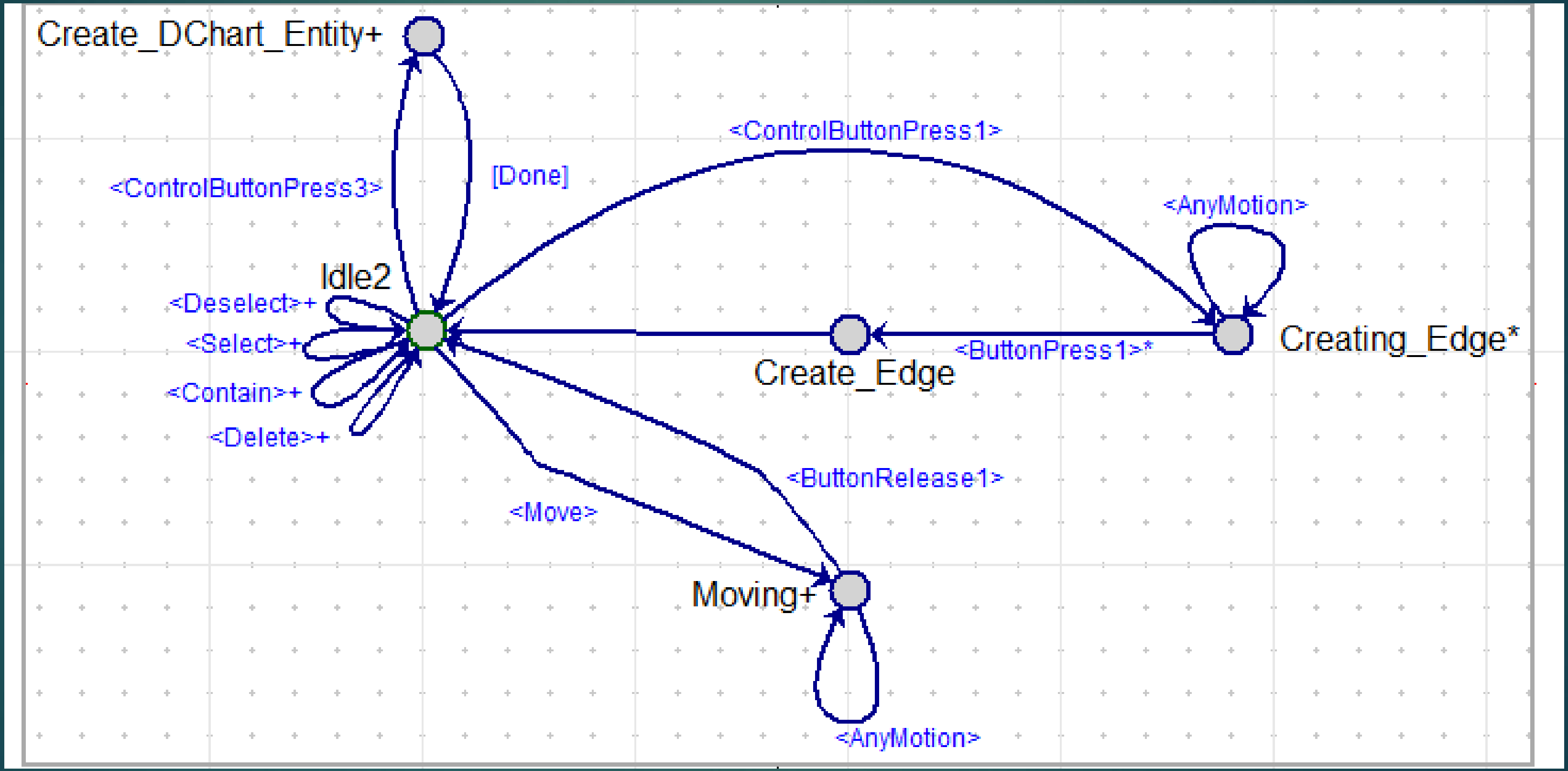
- Implementing the UI behavior using statecharts allowed for a very flexible implementation
- I could reuse a lot of the components of Detlev’s work and adapt it to my needs and fix any issues that existed.

Main Component

- Hierarchical Statecharts
- Implemented using AToM3



Creation of Entities and Edges



Handling the specific entities behaviors

- Some elements like (Scene and Standard Tile) can contain other elements
- This required specific handling of each entity
- When moving an entity that holds other entities, all of the elements should move along
- Like wise, if we delete a scene that has many tiles, all of the tiles along with their items should get deleted

Overview

- Using statecharts to implement the behavior of the UI elements makes understanding the UI much easier and thus modifying it later on or adding more components to it
- The actual drawing and display in this project was done in Python code, where the statecharts communicate with the code via triggers and actions

Conclusion

- Understanding how using models to implement the UI behavior can increase productivity and decrease complexity
- Writing code is prone to errors and bugs
- The need for a unified framework that binds the graphical UI elements with their behavior
- This will guaranty less coding and thus faster production and easy maintenance

References

- Dubé, Denis. "Graph Layout for Domain-Specific Modeling." (2006): 107.
- Denis Dubé, Jacob Beard, H. Vangheluwe, 2009. Rapid development of scoped user interfaces