FELADATKIÍRÁS

Az elektronikusan beadott változatban ez az oldal törlendő. A nyomtatott változatban ennek az oldalnak a helyére a diplomaterv portálról letöltött, jóváhagyott feladatkiírást kell befűzni.



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Automatizálási és Alkalmazott Informatikai Tanszék

Kurdi Barnabás

Creating a textual language for building deck-building games

Somogyi Ferenc Attila

BUDAPEST, 2022

Tartalomjegyzék

[Abstract 5](#_Toc116292049)

[1 Introduction 6](#_Toc116292050)

[1.1 Deck-building board games 6](#_Toc116292051)

[1.2 Creating a text-based programming language 7](#_Toc116292052)

[1.3 Motivation 7](#_Toc116292053)

[1.4 Overview of rest of the paper 8](#_Toc116292054)

[2 Creating Tabl 9](#_Toc116292055)

[3 An interactive experience 10](#_Toc116292056)

[4 Case Studies 11](#_Toc116292057)

[5 Closing thoughts 12](#_Toc116292058)

Hallgatói nyilatkozat

Alulírott **Kurdi Barnabás**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző, cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2022. 10. 10

...…………………………………………….

Kurdi Barnabás

Abstract

Deck-building games have been a staple board game genre ever since the release of *Dominion* in October 2008. In these games players start with a weak set of cards as a personal deck and seek to strengthen their deck with cards from a predetermined array of options, competing against others in the process for victory. No option to date exists for the easy construction and testing of these types of games from scratch in a text-based format.

*Tabl* is a text-based programming language designed to alleviate the aforementioned problem for people who are adept at making deck-building games and curious individuals who want to try their hand at it alike, all while requiring little to no previous programming knowledge. The language comes with its own interpreter written in Python and allows the users to try the card game they have made in a web-based graphical environment developed in Vue.js 3. A file written in *Tabl* is meant to resemble the rulebook of the game being played.

The language is made with further expansion in mind, leaving the possibility of additional development to broaden the scope of it to encapsulate other genres of games.

# Introduction

## Deck-building board games

Board games come in many different shapes and forms: some people enjoy the thrill of rolling the dice for a chance to win big in *Monopoly*, while others prefer crushing their opponents mentally in purely skill-based, deterministic games like chess. These two styles provide a basis for a possible categorization of modern board games. Board-game fans call games that rely more on the luck of the draw and the theatrics of play American-style (or *Ameritrash*), while they dub games that are more abstract and require careful planning and thought throughout *Eurogames*. (examples?)

These two types have very different skill-testing elements. Games that fall into the *Ameritrash* category generally require players to be quick on their feet and fast to adapt to a wide array of situations. *Eurogames* however reward long-term planning and a distinct type of abstract or mathematical thinking.

Naturally, most board games borrow different elements of the two categories and mix them together to provide the best play experience possible. One excellent example of this is the genre of deck-building board games (deck-builders).

The main idea behind deck-builders is the following: what if players were subject to the luck of the draw, but had a chance to tip the scales in their favor by manipulating their own deck during play? The concept of collectible card games (CCGs) is very similar, however the decks used are constructed before play begins, while in deck-builders deck construction is the focus of the main gameplay loop. The first game to fully embrace this idea was *Dominion,* released in 2008.

Most deck-builders follow a very similar play pattern. All the players start with a pre-constructed personal deck of weak cards that don’t have any substantial effect on their own. Players take turns draw a number of these cards, playing them to gain per-turn resources that they can use to buy cards from a common supply. The played cards and the ones players buy during their turn go into the players’ personal discard pile. Once a player’s personal deck is depleted, they shuffle their discard pile to form a new deck of cards to draw from, with their decks now containing the cards they have previously bought.

These cards can have a wide variety of effects. Some of them provide the player with economical advantages, allowing them to buy more cards during subsequent turns. Others make it possible to draw more cards from the deck than normal, allows players to discard cards in order to draw new ones, or enables them to remove the weaker cards from their deck permanently. Several cards facilitate player interaction: making others discard their own cards or stealing resources from them.

The end goals of these games can differ greatly with some requiring the players to accumulate the highest number of victory points. In others, players battle until one of them loses all of their health points. However, no matter the objective, the necessity of players to improve their decks bit by bit remains a constant.

## Creating a text-based programming language

## Motivation

I have always been fascinated and deeply entertained by board games. They provide a safe environment to test one’s mettle against their friends in a competition of wits. After I was introduced to deck-builders I was highly intrigued: they combine luck and strategy in a very distinct way. Deck-builders don’t allow players to calculate their way to a certain victory, requiring critical and strategic thinking in a more meta way. One might not be able to decide what cards they draw in a turn, but they can influence the pool of cards they draw from.

Creating my very own programming language has been a dream of mine ever since I learned of esoteric programming languages like *Shakespeare* or *Whitespace* when I first became interested in coding. The task seemed daunting and a bit too complex for me just to dip my toes into it with no clear goal in mind. Was making a funny or interesting language just for the sake of it worth the time and effort?

I realized that there was a way for me to combine my passion for board games (specifically deck-builders) and my interest in creating my own language. What if there was a way for anyone to create their own deck-building game and test it in a graphical environment online? This is how I came up with the idea of *Tabl.* My goal with Tabl was not only to create a language that could define a deck-builder (one could easily achieve with the use of a data-interchange language like *JSON*), but to make the experience of creating one easy to understand and to structure the language in a way that helps facilitate the ease of translating it into a playable experience. I also wanted to make the files written in the language to be multi-purpose. Not only would they define the author’s custom deck-builder, but they would also look like a traditional board game rulebook, making it easy for anyone to tell the rules of the created game at first glance.

My aspirations for Tabl are two-fold. On one hand I want to make it possible for lovers of the genre with no previous programming experience to try their hand at creating a deck-builder for the first time. On the other, I hope even experienced game designers can find some merit in the concept as well, having a way to effortlessly modify both the minute and bigger, overarching rules of their game and test the changes immediately. This allows for easy evaluation and tweaking of game balance.

## Overview of rest of the paper

In the next section of the paper, I will highlight and explain in detail the different phases of crating the *Tabl* programming language and the interpreter I have developed to process files written in the language. In the section after, I am going to focus on the technological aspects that make testing the created deck-building game possible, describing the different technologies used in both the back- and front-end of the web application.

The third section deals with two case studies: I have created pre-existing card games *Dominion* and *Hero Realms* in *Tabl* and I will be comparing playing these deck-builders through the web application and in real life, testing the product’s viability.

The last section contains my closing thoughts about the project and discussion about the possibility of the expansion of its scope to other types of board games besides deck-builders, or maybe even to board games in general.

# Creating Tabl

# An interactive experience

# Case Studies

# Closing thoughts