Általános információk, a diplomaterv szerkezete

A diplomaterv szerkezete a BME Villamosmérnöki és Informatikai Karán:

1. Diplomaterv feladatkiírás
2. Címoldal
3. Tartalomjegyzék
4. A diplomatervezõ nyilatkozata az önálló munkáról és az elektronikus adatok kezeléséről
5. Tartalmi összefoglaló magyarul és angolul
6. Bevezetés: a feladat értelmezése, a tervezés célja, a feladat indokoltsága, a diplomaterv felépítésének rövid összefoglalása
7. A feladatkiírás pontosítása és részletes elemzése
8. Elõzmények (irodalomkutatás, hasonló alkotások), az ezekbõl levonható következtetések
9. A tervezés részletes leírása, a döntési lehetõségek értékelése és a választott megoldások indoklása
10. A megtervezett mûszaki alkotás értékelése, kritikai elemzése, továbbfejlesztési lehetõségek
11. Esetleges köszönetnyilvánítások
12. Részletes és pontos irodalomjegyzék
13. Függelék(ek)

Felhasználható a következõ oldaltól kezdõdõ LaTeX-Diplomaterv sablon dokumentum tartalma.

A diplomaterv szabványos méretû A4-es lapokra kerüljön. Az oldalak tükörmargóval készüljenek (mindenhol 2.5cm, baloldalon 1cm-es kötéssel). Az alapértelmezett betûkészlet a 12 pontos Times New Roman, másfeles sorközzel.

Minden oldalon - az elsõ négy szerkezeti elem kivételével - szerepelnie kell az oldalszámnak.

A fejezeteket decimális beosztással kell ellátni. Az ábrákat a megfelelõ helyre be kell illeszteni, fejezetenként decimális számmal és kifejezõ címmel kell ellátni. A fejezeteket decimális aláosztással számozzuk, maximálisan 3 aláosztás mélységben (pl. 2.3.4.1.). Az ábrákat, táblázatokat és képleteket célszerû fejezetenként külön számozni (pl. 2.4. ábra, 4.2 táblázat vagy képletnél (3.2)). A fejezetcímeket igazítsuk balra, a normál szövegnél viszont használjunk sorkiegyenlítést. Az ábrákat, táblázatokat és a hozzájuk tartozó címet igazítsuk középre. A cím a jelölt rész alatt helyezkedjen el.

A képeket lehetõleg rajzoló programmal készítsék el, az egyenleteket egyenlet-szerkesztõ segítségével írják le (A LaTeX ehhez kézenfekvõ megoldásokat nyújt).

Az irodalomjegyzék szövegközi hivatkozása történhet a Harvard-rendszerben (a szerzõ és az évszám megadásával) vagy sorszámozva. A teljes lista névsor szerinti sorrendben a szöveg végén szerepeljen (sorszámozott irodalmi hivatkozások esetén hivatkozási sorrendben). A szakirodalmi források címeit azonban mindig az eredeti nyelven kell megadni, esetleg zárójelben a fordítással. A listában szereplõ valamennyi publikációra hivatkozni kell a szövegben (a LaTeX-sablon a BibTeX segítségével mindezt automatikusan kezeli). Minden publikáció a szerzõk után a következõ adatok szerepelnek: folyóirat cikkeknél a pontos cím, a folyóirat címe, évfolyam, szám, oldalszám tól-ig. A folyóirat címeket csak akkor rövidítsük, ha azok nagyon közismertek vagy nagyon hosszúak. Internet hivatkozások megadásakor fontos, hogy az elérési út elõtt megadjuk az oldal tulajdonosát és tartalmát (mivel a link egy idõ után akár elérhetetlenné is válhat), valamint az elérés idõpontját.

Fontos:

- A szakdolgozat készítõ / diplomatervezõ nyilatkozata (a jelen sablonban szereplõ szövegtartalommal) kötelezõ elõírás Karunkon ennek hiányában a szakdolgozat/diplomaterv nem bírálható és nem védhetõ !
- Mind a dolgozat, mind a melléklet maximálisan 15 MB méretû lehet !

Jó munkát, sikeres szakdolgozat készítést ill. diplomatervezést kívánunk !

FELADATKIÍRÁS

A feladatkiírást a tanszéki adminisztrációban lehet átvenni, és a leadott munkába eredeti, tanszéki pecséttel ellátott és a tanszékvezetõ által aláírt lapot kell belefûzni (ezen oldal *helyett*, ez az oldal csak útmutatás). Az elektronikusan feltöltött dolgozatban már nem kell beleszerkeszteni ezt a feladatkiírást.

Budapesti Mûszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Stratégiai játék futtató webes rendszer

Diplomaterv

Készítette
Márkus Krisztián

Konzulens
dr. Dudás Ákos

November 18, 2018

Contents

HALLGATÓI NYILATKOZAT

Alulírott *Márkus Krisztián*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot/ diplomatervet 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, egyértelmûen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerzõ(k), cím, angol és magyar nyelvû tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhetõ elektronikus formában, a munka teljes szövegét pedig az egyetem belsõ hálózatán keresztül (vagy autentikált 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 hozzáférhetõvé.

Budapest, November 18, 2018

Márkus Krisztián
hallgató

Kivonat

Jelen dokumentum egy diplomaterv sablon, amely formai keretet ad a BME Villamosmérnöki és Informatikai Karán végzõ hallgatók által elkészítendõ szakdolgozatnak és diplomatervnek. A sablon használata opcionális. Ez a sablon LaTeX alapú, a *TeXLive* TeX-implementációval és a PDF-LaTeX fordítóval mûködõképes.

Abstract

This document is a LaTeX-based skeleton for BSc/MSc theses of students at the Electrical Engineering and Informatics Faculty, Budapest University of Technology and Economics. The usage of this skeleton is optional. It has been tested with the *TeXLive* TeX implementation, and it requires the PDF-LaTeX compiler.

Introduction

A bevezetõ tartalmazza a diplomaterv-kiírás elemzését, történelmi elõzményeit, a feladat indokoltságát (a motiváció leírását), az eddigi megoldásokat, és ennek tükrében a hallgató megoldásának összefoglalását.

A bevezetõ szokás szerint a diplomaterv felépítésével záródik, azaz annak rövid leírásával, hogy melyik fejezet mivel foglalkozik.

Glossary

Game

- A playable entity for which playing bots can be written. It consists of a game API, a game engine and other meta data including a unique name and player number information.
- An actual event of a game being played by a bot or bots and guided by the game's engine.

Challenge A single player game, whose result (if not an error) is a whole number representing the score received by the playing bot, and an optional maximum points limit.

Match A two player, competitive game which can be played by two different bots and results in a three way outcome or an error.

Game Runtime API A java library containing type definitions which allows games and bots to be created. It is to be used as a provided (non included) dependency for the game engine, game api and bot implementations.

Engine Runtime API A java library containing type definitions and abstract base implementations which allow game engines to be written. It is to be used as a provided (non included) dependency for the game engine.

(Game) Engine An executable library which provides the necessary logic and implementation for a game to work according to its rules.

Game API A java library which contains type definitions and utility code that enable users to create bots for the specific game.

Bot interface An interface which extends the BotInterface marker interface defined in the game runtime api. It defines the game specific methods through which the game engine can interact with the bot.

Bot A virtual player built to play a specific game - alone if that is a challenge or against another bot if that is a match. It is implemented as a class realising the game's bot interface.

Actor A collective term referring to a game engine and the bot or bots playing its game in a given challenge or match.

Actor client A separate process which handles and interacts with a single actor throughout a game.

Chapter 1

Game Model

In order to be operable by the runtime, the games and their respective bots must match certain type level requirements and abide to some contracts.

1.1 Game API

The game api is a library which contains everything that is required or may help to develop bots for the game. This includes the bot interface, all types used in communicating between the engine and a bot and various helpers and utilities to ease bot development.

Bot interface

The bot interface is a simple java interface that extends the BotInterface marker interface found in the universal game runtime api and defines the operations a bot must support. Through its methods will the engine communicate with a bot - which must implement this interface. In order to ensure better compatibility with different programming paradigms - such as functional programming - and generally make bot development easier, is is advised to operate bots as stateless entities and pass the state of the game to them in these methods.

API types

These domain classes and interfaces are to represent the internal model of the game according to its logic. They can be passed to the bot through the bot interface's methods or returned from them. To allow the engine and bots to run on separate virtual machines, these must be serializable. Due to their vm separation, when passing one of these objects to a bot, a replica will be created and therefore changes made to by the engine or bot to their objects will not be visible to the other's instances. If changes are to be made on such parameter by the bot then the returning of the changed value should be used. It is generally a good idea to define these types as simple immutable value classes to help reduce bugs created by not properly understood sharing mechanism.

Utilities

As a game is generally created to be played by many bots, it is very helpful from the game developer to take time and create well usable utilities to help bot development efforts. These helpers usually operate on the domain classes and provide functionality which most bot developers would have to write on their own because of their frequent necessity.

1.2 Game engine

The game engine is a library which contains the implementation of the game logic (the engine itself) and optionally other relevant classes and interface. There are two requirements for being a valid game engine:

- Be a concrete class that implements the generic GameEngine interface from the engine runtime api with the proper bot interface defined in the game api being its type parameter.
- Have a public constructor that takes either one or two instances of the previously mentioned bot interface as parameters, based on weather the game is a challenge or a match.
- Have the realised playGame method return an instance of ChallengeResult or MatchResult based on weather the game is a challenge or a match.

However, in engine runtime api two abstract classes are also defined to help (type-safe) game engine development, namely ChallengeGameEngine and DuelGameEngine.

A game engine is able to log messages during a game for both itself and the playing bot(s) as well. It can do so by acquiring a GameEngineLogger instance either using the EngineLoggerFactory's static getLogger method or if already extending one of the aforementioned abstract helpers, through its logger protected member.

ChallengeGameEngine

ChallengeGameEngine is an abstract class that ensures proper return type (ChallengeResult) while also providing helper methods for easier result creation and logging.

DuelGameEngine

Much like ChallengeGameEngine, DuelGameEngine is also an abstract helper with proper result safety, logging and result creation helpers, however it also implements a basic turn-based logic. Its extender only has to define the mechanism of a turn of the game, while keeping track of the active player is being taken care of.

Other classes

The library can contain any other classes that help the engine work, such as inner types or utilities that it wishes not to share with the bots.

1.3 Bot

Bots are created as libraries that contain a concrete implementation of the specific game's bot interface and have a public, no-argument constructor. The bot library may contain other helpers and utilities as well.

Much like game engines, bots can log messages during matches, however they can only send these to themselves (so only their creator can view them). To create log entries with a bot, a GameBotLogger instance is needed. A bot can access one using the BotLogger-Factory's static getLogger method.

Köszönetnyilvánítás

Ez nem kötelezõ, akár törölhetõ is. Ha a szerzõ szükségét érzi, itt lehet köszönetet nyilvánítani azoknak, akik hozzájárultak munkájukkal ahhoz, hogy a hallgató a szakdolgozatban vagy diplomamunkában leírt feladatokat sikeresen elvégezze. A konzulensnek való köszönetnyilvánítás sem kötelezõ, a konzulensnek hivatalosan is dolga, hogy a hallgatót konzultálja.

Bibliography

Függelék

F.1 A TeXnicCenter felülete

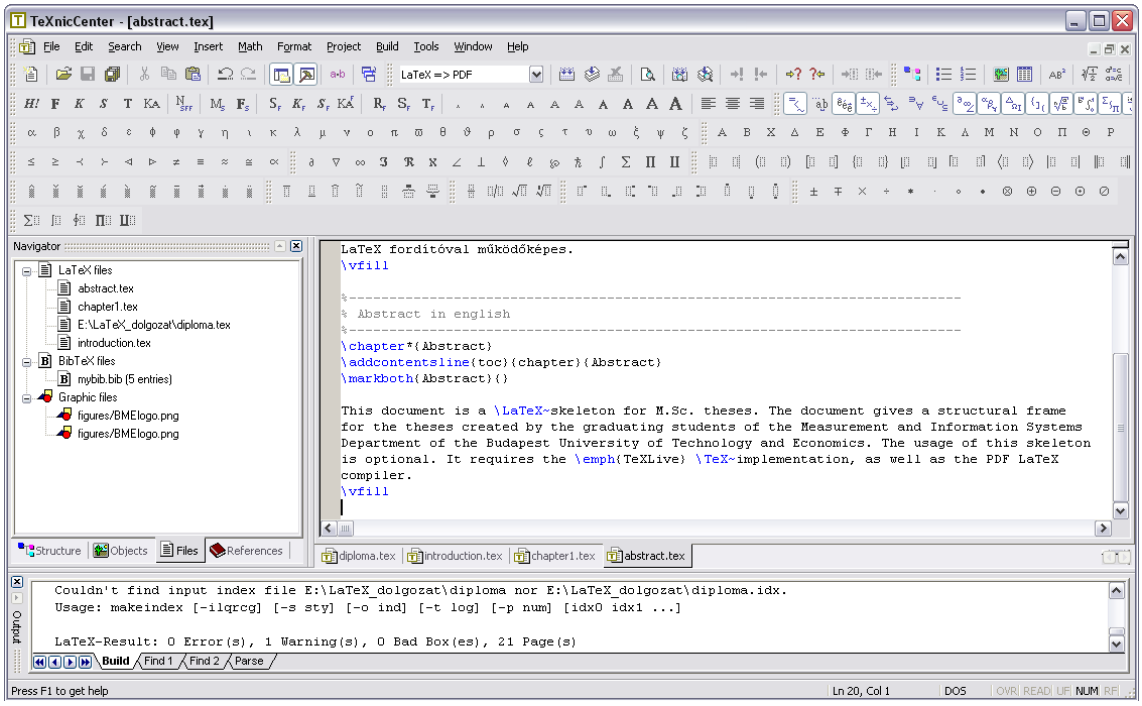

Figure F.1.1: *A TeXnicCenter Windows alapú LaTeX-szerkesztõ.*

F.2 Válasz az „Élet, a világmindenség, meg minden” kérdésére

A Pitagorasz-tételbõl levezetve

$$c^2 = a^2 + b^2 = 42. \tag{F.2.1}$$

A Faraday-indukciós törvénybõl levezetve

$$\operatorname{rot} E = -\frac{dB}{dt} \qquad \longrightarrow \qquad U_i = \oint_{\mathbf{L}} \mathbf{Edl} = -\frac{d}{dt} \int_{A} \mathbf{Bda} = 42. \tag{F.2.2}$$