

chess.py

Konsoli-ikkunassa pelattava shakkipeli.

Ratkaisuperiaate

Shakkipeli itse pyörii pythonin ulkoisen `chess` -kirjaston kautta, jonka tarjoama `Board` -luokka pitää tallessa pelin tilan. Työn tavoitteena oli siis lähinnä luoda sille jollain tavalla käytännöllinen ja kustomoitava komentopohjainen käyttöliittymä. Käyttöliittymän kieleksi valitsin englannin, koska se teki `argparse` -kirjaston käytöstä helpompaa.

Ohjelma pitää sisäistä tilaansa tallessa objektissa, johon voi erinäisten komentojen avulla vaikuttaa. Sisäisesti jokainen komento on oma funktionsa, joka saa erääksi argumentikseen kyseisen objektin. Tämä tekee komentojen lisäämisestä ohjelmaan melko yksinkertaista.

Ulkoiset kirjastot

`chess`

Käytettiin shakkipelin toiminnallisuuksien implementointiin.

`colorama`

Käytettiin mahdollistamaan värillinen tulostus ja kursorin liikuttaminen ANSI-koodien avulla käyttöalustasta riippumatta.

Ohjeet

Ohjelman päämoduuli on nimellisesti `main.py`.

Ohjelmaa ohjataan seuraavilla komennoilla:

- `quit`

Sulkee ohjelman. Ei tallenna pelin tilaa.

- `help`

Tulostaa konsoliin kaikkien komentojen nimet ja yksinkertaiset ohjeet niiden käyttöön.

- `replay`

Aloittaa pelin uudelleentoiston.

Uudelleentoiston suuntaa voi vaihtaa komennolla `rev`. Uudelleentoistotilasta poistutaan komennolla `end`. Mikä tahansa muu tai tyhjä komento etenee yhdellä siirrolla eteen- tai taaksepäin.

- `history`

Tulostaa konsoliin kaikki tähän asti tehdyt siirrot.

- `move [--list] [-l] [MOVE]`

Suorittaa siirron, joka annetaan positionaaliseen argumenttiin `MOVE`. Siirto annetaan englanninkielisellä [algebrallisella merkinnällä](#). Siirrolle voi myös antaa arvon "undo" tai "pass". "undo" kumoaa edellisen siirron, ja "pass" luovuttaa vuoron seuraavalle pelaajalle, mikäli mahdollista.

Vipu `--list` tai sen lyhenne `-l` listaa kaikki siirtävän pelaajan mahdolliset siirrot.

- `reset [--fen FEN]`

Resetoi laudan alkuasemaan. Vivulla `--fen` voi myös määritellä oman alkuaseman [FEN-notaatiolla](#).

- `style FILE`

Lataa laudan tyylin `FILE` -nimisestä JSON-tiedostosta "styles"-kansioista.

- `save FILE`

Tallentaa pelin `FILE` -nimiseen tekstitiedostoon "saves"-kansioon.

- `load FILE`

Lataa pelin `FILE` -nimisestä tekstitiedostosta "saves"-kansioista.

- `ai [--on {w, b} [{w, b} ...]] [--off {w, b} [{w, b} ...]]`

Käynnistää tai sammuttaa pelaajien tietokoneohjauksen. Tietokoneohjatut pelaajat suorittavat vuorollaan automaattisesti satunnaisen laillisen siirron.

Kaikille komennoille voi antaa vivun `-h` tai `--help`, joka tulostaa ohjeet komennon käyttöön.

Jos pelilauta näyttää sekamelskalta satunnaisia merkkejä, ohjelman väritulostuksessa saattaa olla ongelma. Silloin kannattaa kokeilla komentoa `style colorless`.

Pelitallennukset eivät säilytä siirtohistoriaa. `history`, `replay` ja `move undo` -komennot eivät siis "näe" kuin ne siirrot, jotka on tehty senhetkisessä pelisessiossa.

Jos yksittäiseen komennon argumenttiin haluaa sisältää tyhjiä merkkejä, kuten välilyöntejä, täytyy se antaa lainausmerkeissä. Esimerkiksi komennossa `reset --fen "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1"` välilyöntejä sisältävä FEN-koodi on suljettava lainausmerkeihin.

Dokumentaatio

Moduuli `main`

- Muuttuja `SAVES_DIR: str = "saves"`

Tallennustiedostojen kansion polku.

- Muuttuja `SAVE_EXT: str = ".txt"`

Tallennustiedostojen tiedostotyyppi.

- Muuttuja `STYLES_DIR: str = "styles"`

Tyylitiedostojen kansion polku.

- Muuttuja `STYLE_EXT: str = ".json"`

Tyylitiedostojen tiedostotyyppi.

- Luokka `Context`

Sisältää ohjelman sisäisen tilan, jota komennot käsittelevät.

- Muuttuja `board: chess.Board`

- Muuttuja `cmds: commands.Commands`

- Muuttuja `ais: set[chess.Color]`

- Muuttuja `style: style.BoardStyle`

- Metodi `__init__(self)`

`board` -muuttujan arvo initialisoituu laudaksi, joka on perinteisen shakin alkutilassa.

`cmds` -muuttujan arvo initialisoituu `commands` -moduulin `init` -funktion paluuarvoksi.

`style` -muuttujan arvo luetaan "default"-nimisestä tyylitiedostosta. Tiedoston polku hankitaan `style_path` -funktiolla ja tiedosto luetaan `style` -moduulin funktiolla `load_file`.

- Funktio `style_path(name: str) -> str`

Palauttaa annetun nimisen tyylitiedoston täyden tiedostopolun. Nimi annetaan ilman tiedostotyyppiä.

Täydennetty polku riippuu globaalien muuttujien `SAVES_DIR` ja `SAVE_EXT` arvoista.

- Funktio `save_path(name: str) -> str`

Palauttaa annetun nimisen tallennustiedoston täyden tiedostopolun. Nimi annetaan ilman tiedostotyyppiä.

Täydennetty polku riippuu globaalien muuttujien `STYLES_DIR` ja `STYLE_EXT` arvoista.

- Funktio `display_outcome(outcome: chess.Outcome) -> str`

Luo merkkijonoesityksen annetusta `chess.Outcome` -objektista konsolitulostusta varten.

Moduuli `commands`

- Luokka `Commands`

- Muuttuja `commands: dict[str, tuple[Callable, argparse.ArgumentParser]]`

- Metodi `__init__(self)`

- Metodi `add_cmd(self, name: str, cmd: Callable, description: str = "") -> argparse.ArgumentParser`

Lisää `self`-instanssiin `name`-nimisen komennon, joka suorittaa funktion `func`. Komennolle voi antaa myös kuvauksen määrittämällä `description`-merkkijonon.

Palauttaa `argparse.ArgumentParser`-objektin, jota muokkaamalla voi määrittää, miten komennon argumentit luetaan.

- Metodi `run(self, args: list[str], *func_args)`

Suorittaa `self`-instanssiin lisätyn komennon, jonka nimi vastaa ensimmäistä `args`-listan merkkijonoa.

Komentofunktion argumenteiksi annetaan komentoa vastaavan `argparse.ArgumentParser`-instanssin `parse_args`-metodin palauttama objekti, sekä kaikki `func_args`-argumentit.

- Funktio `init() -> Commands`

Initialisoi ja palauttaa uuden `Commands`-instanssin, johon on lisätty kaikki ohjelman tarvitsemat komennot.

- Funktio `argsplit(string: str) -> list[str]`

Hajottaa annetun merkkijonon osajonoihin käyttäen tyhjiä merkkejä erottimena, jättäen aina kaiken lainausmerkeissä olevan tekstin yhdeksi osajonoksi.

Esimerkiksi jono `'abc abc "abc abc"'` hajoaisi osiin `["abc", "abc", "abc abc"]`. Huomaa, että useampaa tyhjää merkkiä kohdeltiin vain yhtenä erottimena, ja että lainausmerkit eivät sisältyneet viimeiseen osajonoon.

- Funktio `cmd_quit(namespace, ctx: main.Context)`

`quit`-komennon funktio.

- Funktio `cmd_help(namespace, ctx: main.Context)`

`help`-komennon funktio.

- Funktio `cmd_replay(namespace, ctx: main.Context)`

`replay`-komennon funktio.

- Funktio `cmd_history(namespace, ctx: main.Context)`

`history`-komennon funktio.

- Funktio `cmd_move(namespace, ctx: main.Context)`

`move`-komennon funktio.

- Funktio `cmd_reset(namespace, ctx: main.Context)`

`reset`-komennon funktio.

- Funktio `cmd_style(namespace, ctx: main.Context)`

`style`-komennon funktio.

- Funktio `cmd_save(namespace, ctx: main.Context)`

`save`-komennon funktio.

- Funktio `cmd_load(namespace, ctx: main.Context)`

`load`-komennon funktio.

- Funktio `cmd_ai(namespace, ctx: main.Context)`

`ai`-komennon funktio.

Moduuli `style`

- Luokka `StyleStr`

Edustaa merkkijonoa, jonka tulostuksen etu- ja taustaväritystä voidaan muuttaa ANSI-koodeilla.

- Muuttuja `string: str`

ANSI-koodi, joka määrittää tekstin värin.

- Muuttuja `back: str`

ANSI-koodi, joka määrittää tekstin taustavärin.

- Muuttuja `style: str`

ANSI-koodi, joka määrittää värityksen tyylin.

- Metodi `__init__(self, string: str = "", fore: str = "", back: str = "", style: str = "")`

- Metodi `copy(self) -> Self`

- Metodi `__str__(self) -> str`

Palauttaa tulostusta varten merkkijonoesityksen `self`-instancsista, jossa `self.fore`, `self.back` ja `self.style` -ANSI-koodit on yhdistetty `self.string`-muuttujan arvoon.

Palautetun jonon loppuun lisätään myös ANSI-koodi, joka resetoit kaiken tekstin tyylityksen, jotteivat tekstiin vaikuttavat tyylit "vuoda" eteenpäin. Tätä koodia ei kuitenkaan lisätä, mikäli `self.fore`, `self.back` ja `self.style` ovat kaikki tyhjiä merkkijonoja.

- Luokka `CheckerPattern`

Edustaa ruutukuviointia, joka kartoittaa shakkiruutuja `StyleStr`-objekteihin.

- Muuttuja `light: StyleStr`

"Vaalean" ruudun tyylitys. Joka toinen ruutu on vaalea niin, että mm. ruudut a2 ja b1 ovat vaaleita.

- Muuttuja `dark: StyleStr`

"Tumman" ruudun tyylitys. Joka toinen ruutu on tumma niin, että mm. ruudut a1 ja b2 ovat tummia.

- Metodi `__init__(self, light: StyleStr, dark: StyleStr)`

- Metodi `__getitem__(self, key: chess.Square) -> StyleStr`

- Luokka `BoardStyle`

Edustaa shakkilaudan tyyliä, jonka avulla laudasta voidaan luoda merkkijonoesitys.

- Muuttuja `pattern`

Objekti, jonka `__getitem__`-metodi kartoittaa laudan ruudut `StyleStr`-objekteihin.

- Muuttuja `pieces`

Objekti, jonka `__getitem__`-metodi kartoittaa laudan nappulat `StyleStr`-objekteihin.

- Metodi `__init__(self, pattern, pieces)`

- Metodi `display(self, board: Board, orientation: chess.Color = chess.WHITE)`

Luo annetusta laudasta merkkijonoesityksen tulostusta varten. `orientation` merkitsee sen pelaajan väriä, jonka näkökulmasta lauta piirretään.

- Funktio `parse_color(string: str, fore: bool = True) -> str`

Palauttaa ANSI-koodin annetun nimiselle värille. Boolean-arvo `fore` merkitsee, palautetaanko väri etu- vai taustavärinä. Syötteenä olevan merkkijonon kirjainkoko ei huomioida.

Mikäli annettu merkkijono ei ole kelvollinen, funktio tuottaa `ValueError`-virheen.

- Funktio `parse_style(string: str) -> str`

Palauttaa ANSI-koodin annetun nimiselle värityksen tyyliille. Syötteenä olevan merkkijonon kirjainkoko ei huomioida.

Mikäli annettu merkkijono ei ole kelvollinen, funktio tuottaa `ValueError`-virheen.

- Funktio `parse_json_style(json: str | dict, default_fore: bool = True) -> StyleStr`

Muuntaa annetun JSON-tiedostosta saadun merkkijonon tai sanakirjan `StyleStr`-objektiksi. Muuntamisessa hyödynnetään `parse_color` ja `parse_style`-funktioita. Argumentti `default_fore` merkitsee, oletetaanko väriä edustavan merkkijonon olevan etu- vai taustaväri.

Esimerkiksi merkkijono `"white"` muuntuisi arvoksi `StyleStr(fore = colorama.Fore.WHITE)` tai `StyleStr(back = colorama.Back.WHITE)` riippuen `default_fore`-argumentin arvosta. Merkkijono `"Tekstiä"` muuntuisi arvoksi `StyleStr("Tekstiä")`.

Sanakirja `{"text": "Tekstiä", "fore": "white", "style": "bright"}` puolestaan muuttuisi arvoksi `StyleStr("Tekstiä", fore = colorama.Fore.WHITE, style = colorama.Style.BRIGHT)`. Huomaa, että puuttuva arvo `back` jätettiin yksinkertaisesti huomioimatta, eikä tässä tapauksessa argumentilla `default_fore` ollut vaikutusta.

- Funktio `parse_json_piece(json: str | dict, color: str) -> StyleStr`

Muuntaa annetun JSON-tiedostosta saadun merkkijonon tai sanakirjan pelinappulaa edustavaksi `StyleStr`-objektiksi. Muuntamisessa käytetään `parse_json_style`-funktioita.

Mikäli annettu arvo on sanakirja, joka sisältää `color`-avainarvon, syötetään silloin funktiolle `parse_json_style` arvo `json[color]`. Muussa tapauksessa argumentin `json` arvo syötetään funktiolle sellaisenaan.

- Funktio `load_file(path: str) -> BoardStyle`

Lukee JSON-tiedoston annetusta polusta ja palauttaa sen määrittelemän `BoardStyle`-objektin.

Tiedoston käsittelyssä hyödynnetään funktioita `parse_json_style` ja `parse_json_piece`.