Matopeli

Harjoitustyö on pythonilla tehty klassinen matopeli. Tulostuu merkkeinä konsoliin, käyttäjä voi valita kartan suuruuden ja

Peli on koodattu modulaarisesti. Koska osat eivät ole keskenään kovin riippuvaisia toisistaan, on koodin eri osien uudistaminen ja uusien toimintojen lisääminen helpompaa. Vaikka peli ei kauheaa optimointia välttämättä tarvitse, toiminnot on pyritty tekemään siten, että mahdollisimman vähän muutoksia tapahtuu.

Pelin eri osat on jaettu luokkiin, joista jokainen sisältää omat funktiot ja tietonsa. Peli sisältää tiedot eri objekteista kuten pelaajasta ja omenoista sekä kartasta. Pelaajalla ja omenalla on molemmilla tieto tyhjistä soluista, joita käytetään omenan uudelleensijoittamisessa. Pelaajalla on tieto omasta pituudestaan, entisistä sijainneistaan, nykyisestä sijainnistaan, suunnastaan ja siitä onko pelaaja kasvanut kuvaruudun aikana. Omenalla on tieto sijainnistaan, entisestä sijainnistaan ja siitä onko se vaihtanut paikkaa kuvaruudun aikana.

Luokka Vec2:

Helpottaa perus kaksiulotteisten vektori matematiikka laskujen ja muiden toimintojen kanssa.

- add(self, B)
- radd(self, B)
- eq(self, B)
- neg(self)
- hash(self)
- set(self, B)
- abs(self)
- copy(self)

Luokka Game:

Hoitaa itse pelin pyörittämisen kooten pelin eri elementit. Hyödyntää threading.Thread(), time.sleep(), time.perf_counter() ja os.system().

- isGameOver(self)
    - Tarkistaa onko pelaaja osunut seiniin tai itseensä kuvaruudun aikana
- start(self)
    - Aloittaa näppäinpainalluksia seuraavan threadin ja peliä pyörittävän silmukan.
- end(self)
    - Asettaa Game.game_over arvon epätodeksi lopettaen peliä pyörittävän silmukan

- update(self, player_direction: Vec2)
    - Tekee yhden kuvaruudun toiminnot. Aluksi pelaajaa liikutetaan Player.move(), jonka jälkeen tarkistetaan pitäisikö pelaajan kuolla uudessa paikassa Game.isGameOver(). Tämän jälkeen katsotaan onko pelaaja syönyt yhtään omenaa Apple.tryConsume(). Jos peli ei ole päättynyt seuraava kuvaruutu tulostetaan.
- saveScore(self, name: str)
    - Tallentaa pelaajan saaman pistemäärän tiedostoon scoreboard.txt, jos se on kymmenen parhaan joukossa.

Luokka GUI:

Hoitaa käyttöliittymän ja pelien valmistelun. Hyödyntää os.system().

- startMenu(self)
    - Tulostaa aloitusnäytön ja määrää luokan muiden funktioiden suoritusjärjestyksen.
- setupMenu(self)
    - Pyytää pelaajaa määrittelemään pelin kartan koon.
- endMenu(self)
    - Näyttää pelaajan saaman pistemäärän ja pyytää siihen yhdistettävää nimimerkkiä.
- scoreboard(self)
    - Tulostaa pistetaulun scoreboard.txt.
- startGame(self)
    - Kutsuu Game.start() funktiota aloittaen pelin.

Luokka Renderer:

Hoitaa pelin elementtien tulostamisen näytölle. Renderer.buffer sisältää kaikki kuvaruudun tulostettavat merkit. Hyödyntää numpy kirjaston taulukkoa.

- clearBuffer(self, buffer)
    - Vaihtaa Renderer.buffer listan kaikki merkit välilyönneiksi, jotta uusi kuvaruutu näyttää aluksi tyhjältä.
- updateBuffer(self)
    - Luo uuden kuvaruudun kutsumalla renderöitävien objektien funktiota rasterize(). Renderöitävät objektit hoitavat itse itsensä kuvaruutuun lisäämisen.
- drawBuffer(self)
    - Tulostaa Renderer.buffer listan.
- erase(self)
    - Peittää vanhan kuvaruudun.
- cycle(self)

    - Kutsuu muita luokan funktioita. Päivittää kuvaruudun Renderer.updateBuffer(), peittää vanhan kuvaruudun Renderer.erase() ja piirtää uuden kuvaruudun Renderer.drawBuffer().

Luokka Player:

Hoitaa ja säilyttää pelaajaan (matoon) liittyvät toiminnot ja tiedot.

- move(self)
    - Liikuttaa pelaajaa vektorin mukaiseen suuntaan. Poistaa aikaisemman sijainnit Player.cache listasta jos sen pituus ylittää pelaajan pituuden Player.length. Muistiin tallentuu yksi ylimääräinen sijainti, jotta tiedetään sen hännän sijainti edeltävässä kuvaruudussa.
- grow(self)
    - Kasvattaa pelaajan kehoa yhdellä ruudulla.
- isCollidingBorder(self, map_size: Vec2)
    - Tarkistaa onko pelaaja pelialueen ulkopuolella.
- isCollidingSelf(self)
    - Tarkistaa osuuko pelaajan pää omaan kehoonsa.
- rasterize(self, buffer: list[list])
    - Mikäli pelaaja on kasvanut, tyhjentää Renderer.buffer listasta edeltävän kuvaruudun häntänsä sijainnin. Lisää aina merkin listaan päänsä kohdalle. Mikäli pelaaja on muuttanut suuntaa, vaihtaa myös edeltävän kuvaruudun pään sijainnin kulma palaan.
- updateEmptyCells(self)
    - Mikäli pelaaja ei ole kasvanut, lisää pelaajan edeltävän kuvaruudun hännän sijainnin takaisin Game.empty_cells listalle. Poistaa aina pelaajan pään sijainnin listasta.

Luokka Apple:

Hoitaa ja säilyttää omenoihin liittyvät toiminnot ja tiedot. Hyödyntää random.choice().

- tryConsume(self, player: Player)
    - Katsoo onko pelaaja syönyt omenan. Hakee itselleen uuden sijainnin Apple.regeneratePosition() ja päivittää tyhjät solut Apple.updateEmptyCells() jos niin. Palauttaa arvon siitä, kuinka paljon pisteitä pelaajan pitäisi saada.
- regeneratePosition(self)
    - Hakee itselleen uuden sijainnin tyhjistä ruuduista eli Game.empty_cells listan alkioista.
- rasterize(self, buffer: list[list])
    - Päivittää tiedot itsestään Renderer.buffer listassa. Tyhjentää aikaisemman sijainnin merkin listassa ja sijoittaa merkin uudelle sijainilleen listassa.
- updateEmptyCells(self, clear_prev: bool = False)

    - Päivittää tiedot itsestään Game.emtpy_cells listassa. clear_prev voi estää omenaa lisäämästään itseään listaan.

Syöte funktiot:

Hoitavat käyttäjän syötteen lukemiseen liittyviä asioita. Hyödyntää keyboard.is_pressed(), msvcrt.getch() ja msvcrt.kbhit().

- whichKeysPressed(*keys: list[str])
    - Palauttaa kaikki keys listassa olevat painetut näppäimet.
- waitForAnyKey()
    - Odottaa kunnes mikä tahansa näppäintä painetaan.
- threadedMovementInput(output: Vec2, kill: list[bool], game: Game)
    - Kuuntelee pelaajan näppäinpainalluksia madon ohjaamiseen. Suoritetaan toisessa threadissä, jotta pelin suorituksen aikainen time.sleep() kutsuminen ei estä syötteen antamista.

Ulkoiset kirjastot

- Numpy (ei sisälly pythonin standardi kirjastoihin)
    - Tehokkaampi listojen käyttö.
- Threading
    - Syötteen lukeminen pääsilmukan odottaessa.
- Random
    - Satunnaisvalinnat
- Keyboard (ei sisälly pythonin standardi kirjastoihin)
    - Näppäinpainallusten lukeminen
- MSVCRT
    - Näppäinpainallusten lukeminen
- Time
    - Kuvataajuuden säätely
- OS
    - Konsolin tyhjentäminen

Vastuualueiden jakautuminen

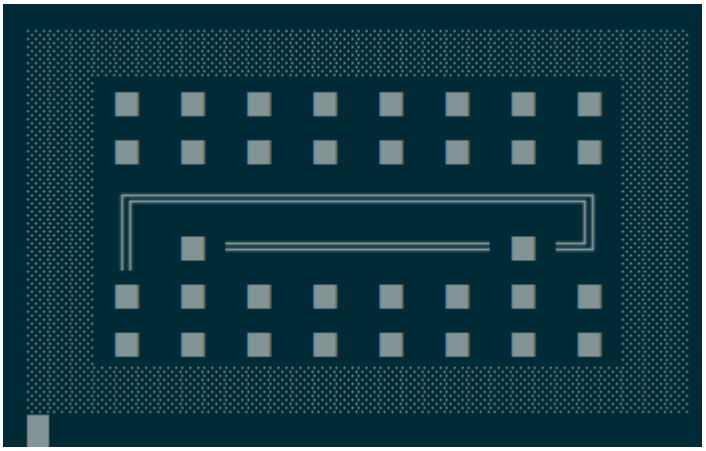
Vastuualueet eivät jakautuneet selkeästi. Harjoitustyössä toteutetut tehtävät suoritettiin keskenään. Molemmat pohtivat ja toteuttivat ohjelmiston toimintoja yhdessä.

Ohjelmiston käyttäminen

Suorita main.py tiedosto ja lataa ulkoiset kirjastot. Tiedosto scoreboard.txt pitää myös olla samassa tiedostosijainnissa kuin main.py tiedosto.

Huomiot

Peliin pystyy lisäämään enemmän kuin yhden omenan. Tämä voi kuitenkin aiheuttaa ongelmia omenien uudelleen sijoittamisessa, mikäli tyhjiä ruutuja ei ole riittävästi. Omenat jäävät aikaisemmalle sijainnilleen ja voivat pinoutua tai ylikirjoittaa pelaajan kehon Renderer.buffer muuttujassa. Kuvassa näkyy kyseinen ongelma.



Syötettä pyytäessä pythonin input() funktiolla kaikki pelin aikana painetut näppäimet kirjoittuvat syöttökenttään.

Koodin rakenteen luonteen takia on myös mahdollista toteuttaa esimerkiksi moninpeli, jossa on kaksi matoa. Tämä voi kuitenkin vaatia pieniä muutoksia ohjelman muussa toiminnassa. On myös mahdollista lisätä suhteellisen helposti erilaisia pelimuotoja ja elementtejä peliin, kuten satunnaisesti spawnatut seinämät.

Tekijät: Leevi Sirkkilä, Ron Puolimatka
23.12.2023