Laevade pommitamine

Siim Tanel Laisaar, Joosep Kaimre, Aveli Klaos

Eesmärk

Töö eesmärgiks oli luua laevade pommitamise mäng ning seda mängiv tehisintellekt (st vastane). Soovisime tehisintellekti jaoks kasutada vähemalt kolme erinevat algoritmi ning lasta mängus mängijal nende vahel valida. Lisaks sellele oli eesmärgiks luua graafiline liides ning implementeerida inimese ning arvuti vahelist suhtlust. Hiljem seadsime endale eesmärgiks luua ka võimalus mängijal programmile algne lauaseis pildina ette anda.

Kasutatud ideed

Algoritmide ideed tulid siit ja siit lingilt.
Pildituvastuseks on kasutatud Tehisintellekti 11. praktikumi ja 12. praktikumi materjale (ka lahendustega failidest) ning ka 6. kodutöös tehtut.

Töö autorite panused

Siim Tanel Laisaar
- GUI loomine (mängulaua kujutamine ja laevade sisestamise meetodi valik)
- Arvuti ja kasutaja vaheline suhtlus
- Pildituvastus (kasutaja laevade sisselugemine etteantud pildilt)
- Abimeetodite loomine
- Testimine

Joosep Kaimre
- Tehisintellekti algoritmide loomine
- GUI menüü loomine (algoritmide valik)
- Abimeetodite loomine
- Testimine

Aveli Klaos
- GUI loomine (mängulaua kujutamine ja laevade sisestamise meetodi valik)
- Arvuti ja kasutaja vaheline suhtlus
- Pildituvastus (kasutaja laevade sisselugemine etteantud pildilt)
- Abimeetodite loomine
- Testimine

Siim Tanel ja Aveli tegid suure osa ajast paarisprogrammeerimist. Seega pole commit'i autor GitHub-is otseselt täpne.

Programmi testimisvõimalused

Selleks, et meie programm töötaks, tuleks kõigepealt installida pygame, numpy ning opencv-python. Seda saab teha näiteks järgnevate käskude abil:

- `pip install pygame`
- `pip install numpy`
- `pip install opencv-python`

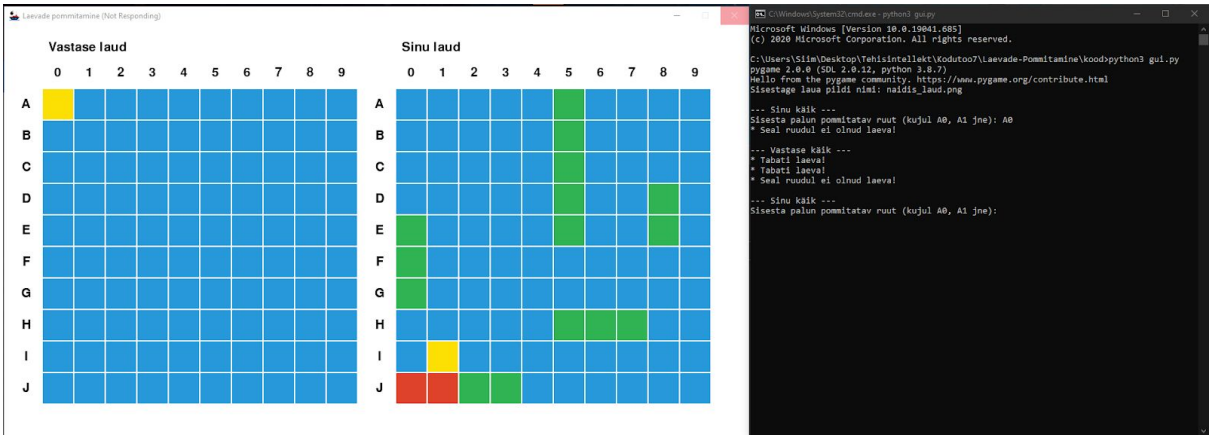
Sammud programmi jooksutamiseks ja testimiseks

1. Navigeeru käsureal kataloogi, kus asuvad koodifailid.
2. Käivita programmi GUI näiteks käsuga `python3 gui.py`.
3. Vali nupuvajutuse abil algoritm, mille vastu mängida.
4. Vali nupuvajutuse abil, kuidas soovid enda mängulauda sisestada.
    a. Käsurealt
        i. Käsurida küsib sinult järjest, mis ilmakaare suunas ning alates mis koordinaatidest laeva lisada. Vasta neile küsimustele vastavalt juhistele.
    b. Pildituvastusega
        i. Ava fail `Kasutaja_laua_tabel.docx` ning järgi failisiseseid juhiseid.
        ii. Sisesta loodud pildifaili nimi käsureal esineva küsimuse vastuseks.
5. Vasta käsureal esinevatele küsimustele, mis küsivad, mis ruute soovid pommitada.

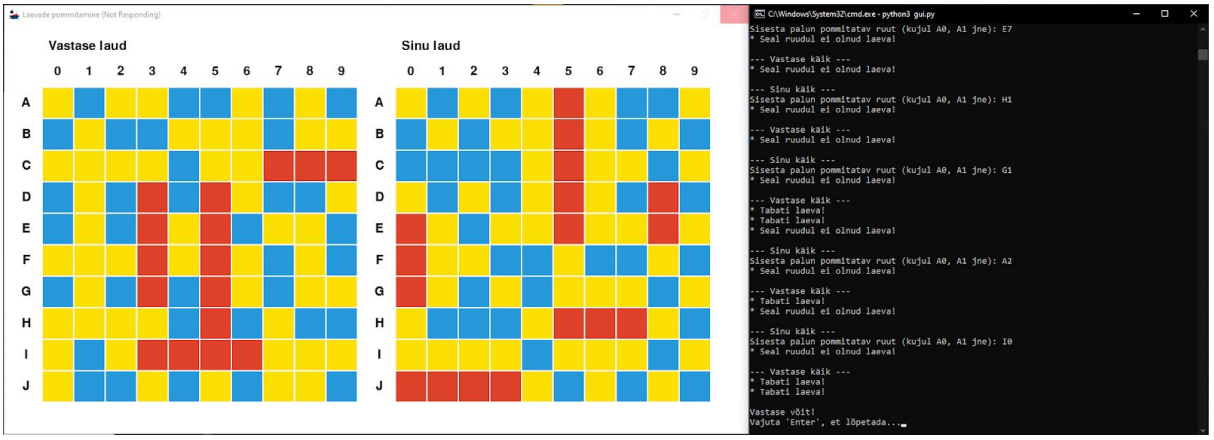
Mängu testimiseks saab eelnimetatud samme järgides mängu läbi mängida erinevate algoritmide vastu ning võrrelda mängu võitmiseks/kaotamiseks kulunud käikude arvu. Samuti saab GUI's vaadelda algoritmide töötamise põhimõtteid. Lisaks saab testida pildituvastust erinevate ise loodud piltidega (ja ka näidispildiga), kuid sealjuures tasub siiski järgida faili `Kasutaja_laua_tabel.docx` juhiseid.

Testimistulemused

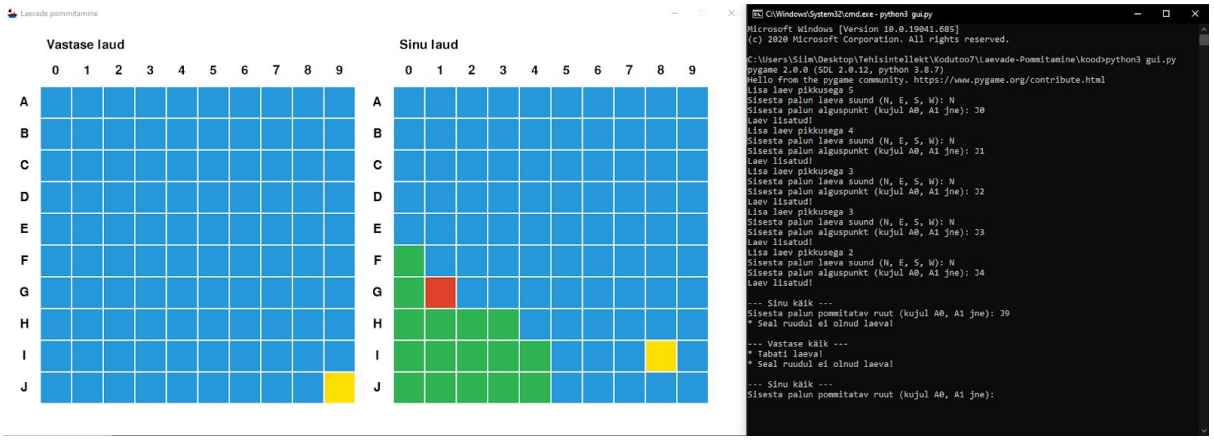
Mänguseis pärast laevade sisselugemist failist `naidis_laud.png` ning pärast kasutaja ja vastase (algoritmiga Hunt koos paarsusega) ühte käiku:



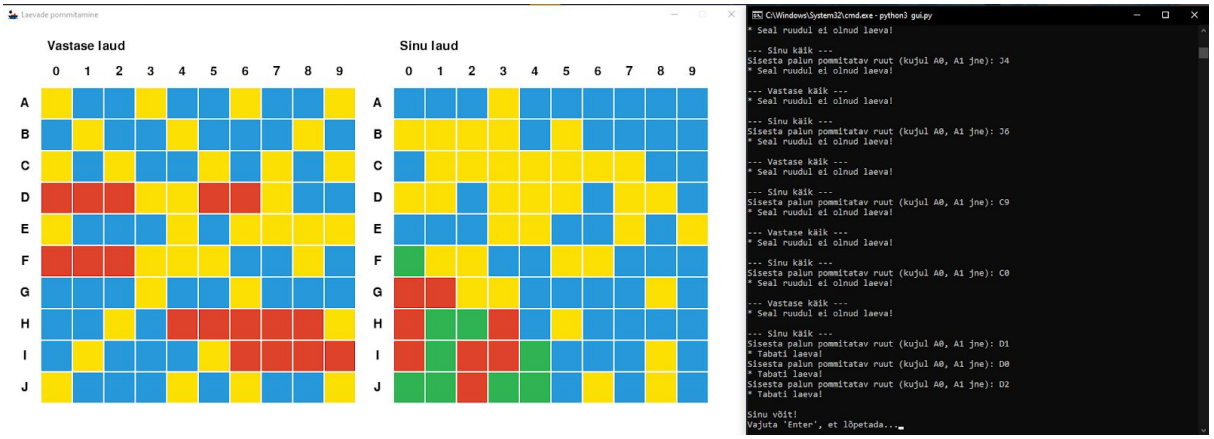
Mänguseis pärast vastase (algoritmiga Hunt koos paarsusega) võitu mängija vastu:



Mänguseis pärast laevade sisse lugemist käsurealt ning pärast kasutaja ja vastase (algoritmiga Random) ühte käiku:



Mänguseis pärast kasutaja võitu vastase (algoritmiga Random) vastu:



Töö käigu kirjeldus

Töö käigus implementeerisime 4 algoritmi (random, Hunti algoritm, Hunti algoritmi edasiarendus koos ruutude paarsusega ning tõenäosuslik algoritm). Kõik seatud eesmärgid said realiseeritud ning töö käigus tekkiski soov luua mängijale võimalus lisada laevu lauale ka pildi abil, mitte ainult käsurea abil (seega me otsustasime pildituvastust kasutada).

Peamised probleemid, mis meil esinesid, olid seotud *pygame* mooduliga. Tihtipeale *pygame*'i aken kas hangus või hoopis ei uuendanud ennast iga kord, kui oleksime seda oodanud. Näiteks vahel uuendati GUI's pommitatud ruutu alles pärast järgmist käiku. Nendele probleemidele kulus palju aega ning 100% kindlusega neid korda ei saanudki. Lisaks tooksime probleemine välja ka selle, et meie tõenäosusliku algoritmi implementatsioon ei tööta eriti hästi, kuigi tegelikult peaks see laevade pommitamise jaoks üpriski hea algoritm olema. See probleem jäigi meil lahendamata, sest algoritm on üpriski keerukas ning koodi vaadates me ise mingeid loogikavigu ei märganud ning seega ei oska arvata, milles täpne probleem peituda võiks.

Üldjoontes sujus aga töö väga hästi ning teised algoritmid tunduvad, et töötavad korrektselt. Samuti õnnestus hästi pilditöötlus (kasutaja laua sisselugemiseks), suuresti seetõttu, et me seadsime kasutajale ette pigem ranged reeglid/juhised pildi koostamiseks. Samas on kasutajal juhiseid järgides vägagi kerge järjest uusi lauaseise (ja pilte) luua. Ka GUI loomine õnnestus hästi, kui välja arvata probleemid *pygame* mooduliga. Lisaks eelnevale saime enda meelest hästi hakkama ka kasutaja ning arvuti vahelise suhtlusega.